# Penetration Testing Report

## 1. Executive Summary

This penetration testing report provides an assessment of the security vulnerabilities identified during the HackThisSite Basic Challenges (Levels 1-11). The primary goal of this test was to analyze common security flaws in web applications and provide recommendations to mitigate potential risks.

**Basic Hacking**

[www.hackthissite.org](http://www.hackthissite.org)

## 2. Scope of Testing

Target Application: HackThisSite Basic Challenges  
Testing Methodology: Black-box testing  
Tools Used: Web browser developer tools, HTTP request analysis, and manual code review

## 3. Vulnerability Description & Key Findings

### Level 1 - HTML Source Code Exposure

Vulnerability: Password stored in plain text within the HTML source code.

Exploit Method: Viewing page source (`Ctrl + U`) and extracting the password from a hidden field or comment.

Recommendation: Never store sensitive credentials in client-side HTML. Use server-side authentication.

### Level 2 - HTML Comment Disclosure

Vulnerability: Password stored within an HTML comment.

Exploit Method: Inspecting the HTML source (`Ctrl + U`) and identifying the comment containing credentials.

Recommendation: Remove sensitive data from comments; use environment variables for authentication mechanisms.

### Level 3 - Hidden Form Fields

Vulnerability: Password embedded within a hidden input field.

Exploit Method: Viewing the source code or modifying form fields using browser developer tools.

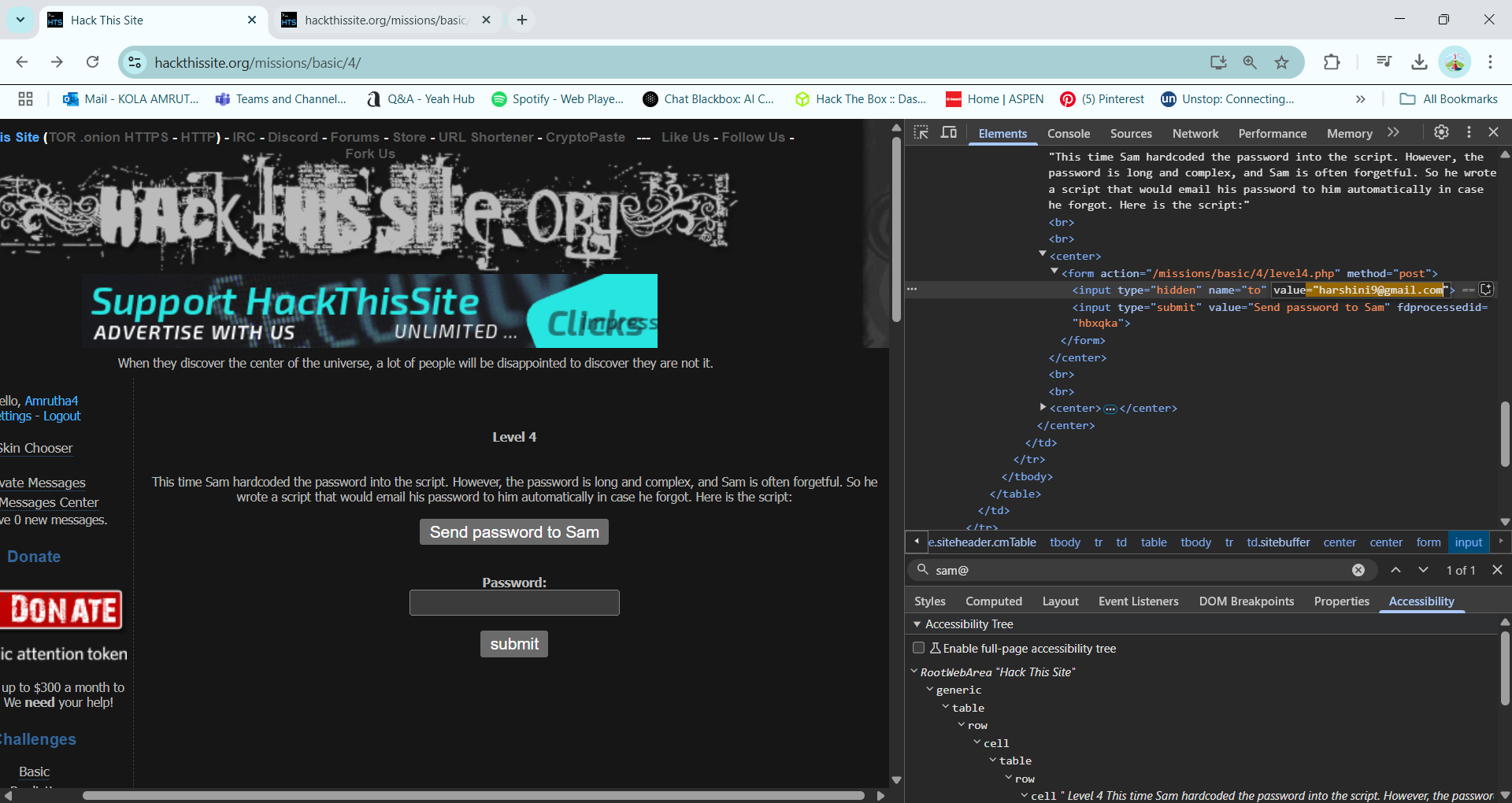
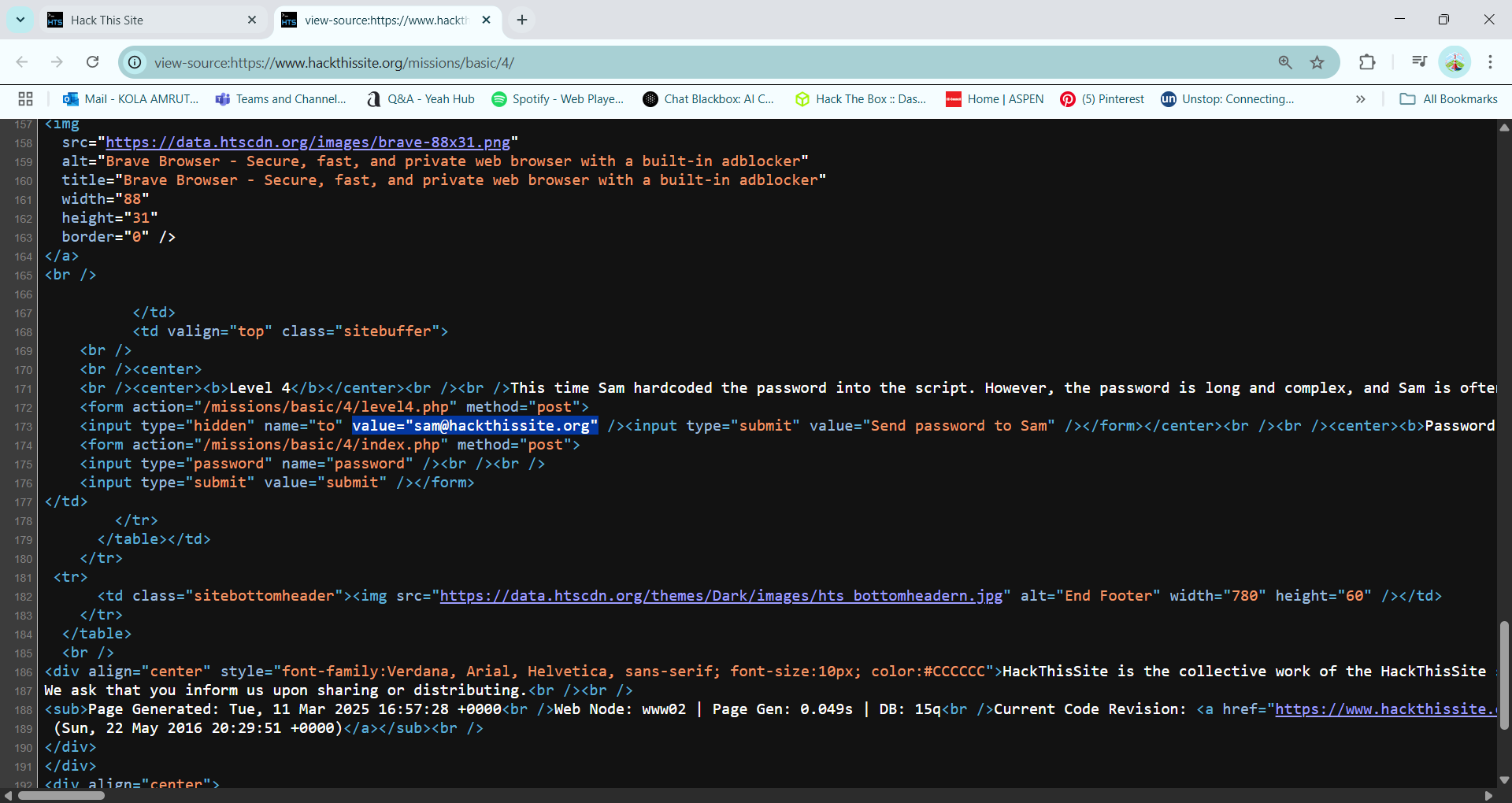
Recommendation: Do not trust hidden fields for authentication. Always validate input on the server side.

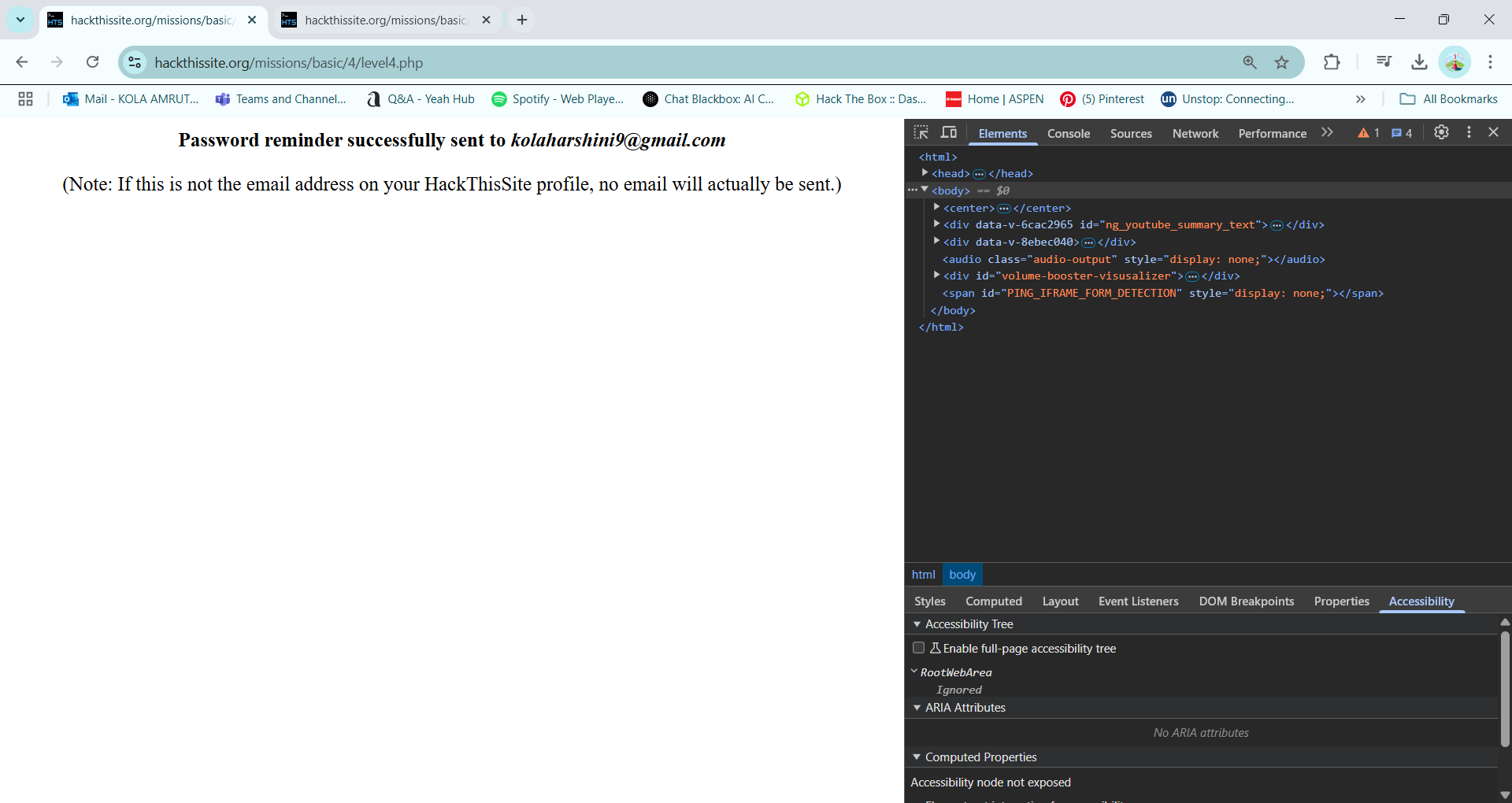
### Level 4 - JavaScript Validation Manipulation

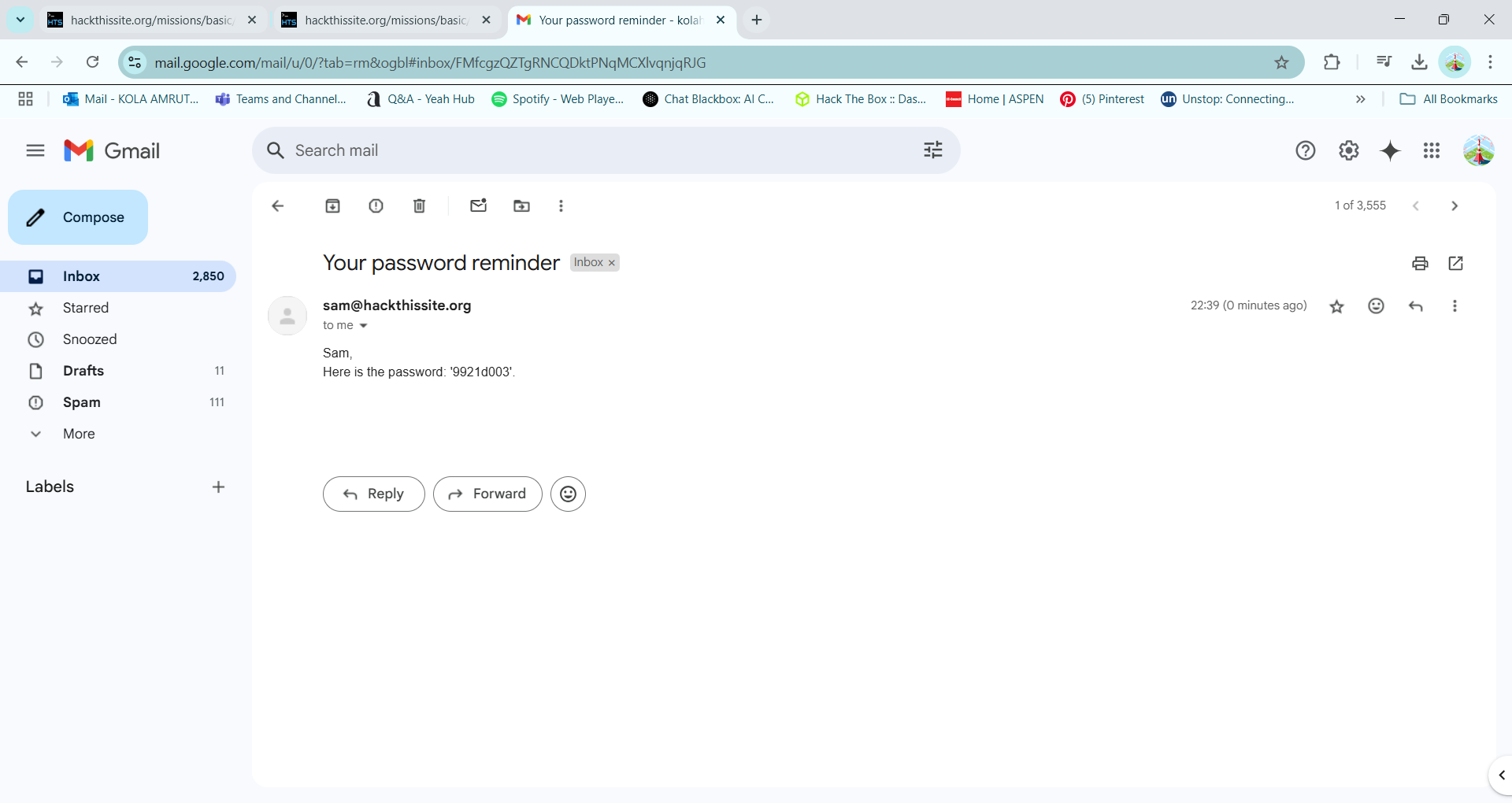
Vulnerability: Password validation performed on the client-side using JavaScript.

Exploit Method: Editing JavaScript functions using the browser console to bypass validation.

Recommendation: Perform all authentication logic on the server side.







### Level 5 - URL Parameter Manipulation

Vulnerability: Password stored within the URL query string.

Exploit Method: Modifying URL parameters in the address bar to access restricted content.

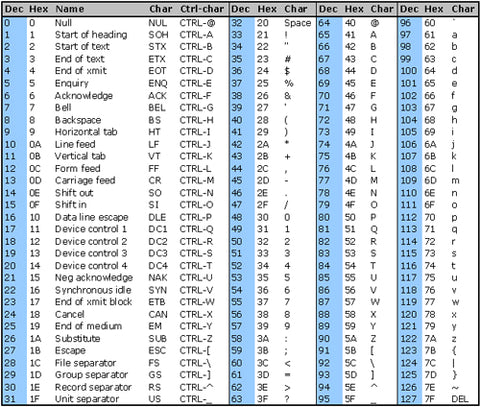
Recommendation: Do not pass sensitive data through URLs. Use session-based authentication.

### Level 6 - ASCII Character Encoding Exploit

Vulnerability: Password stored as ASCII-encoded characters.

Exploit Method: Converting ASCII values to readable text using an ASCII table.

Recommendation: Avoid encoding sensitive data in easily reversible formats.



**Step-by-Step Decryption:**

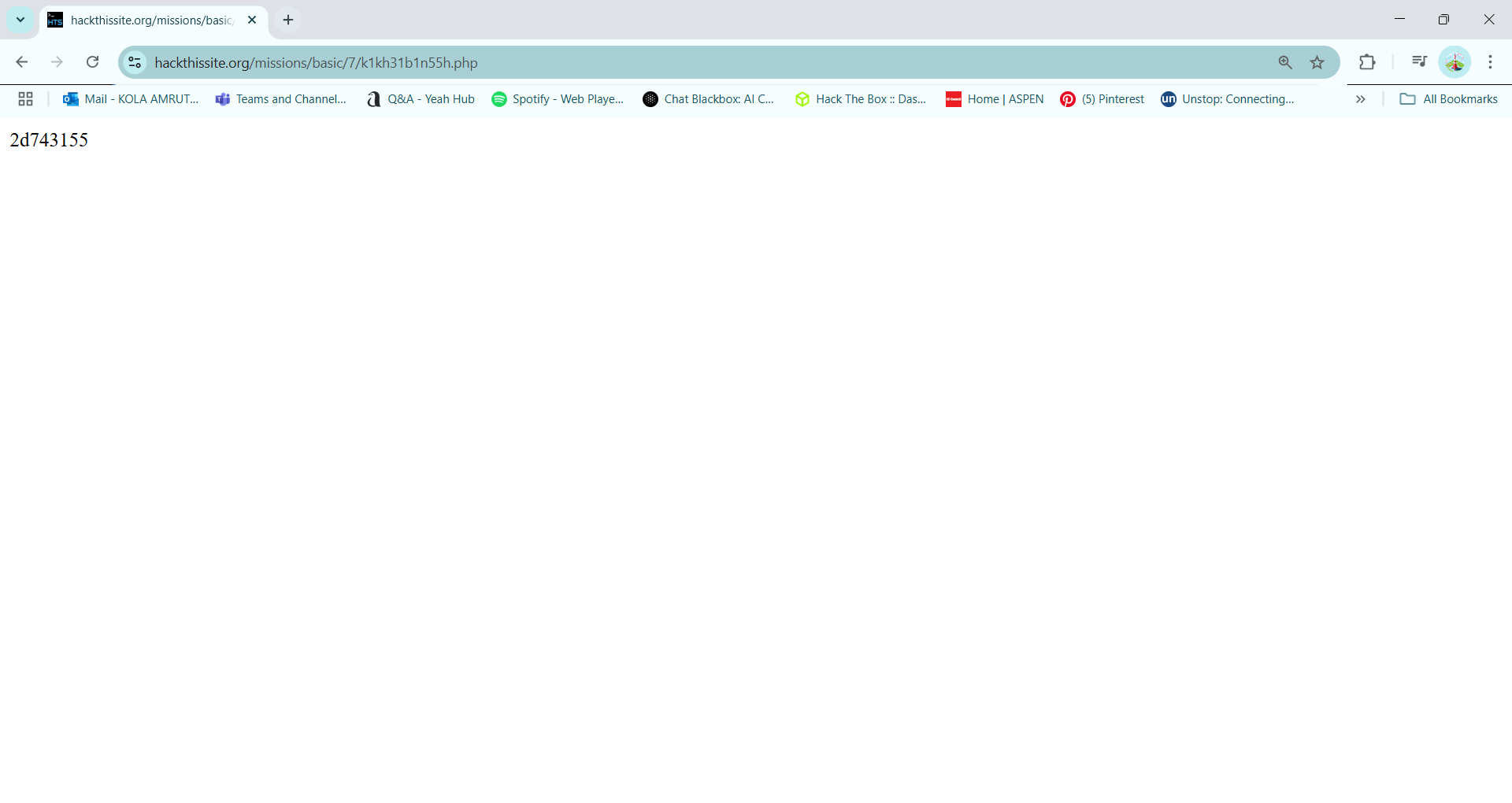
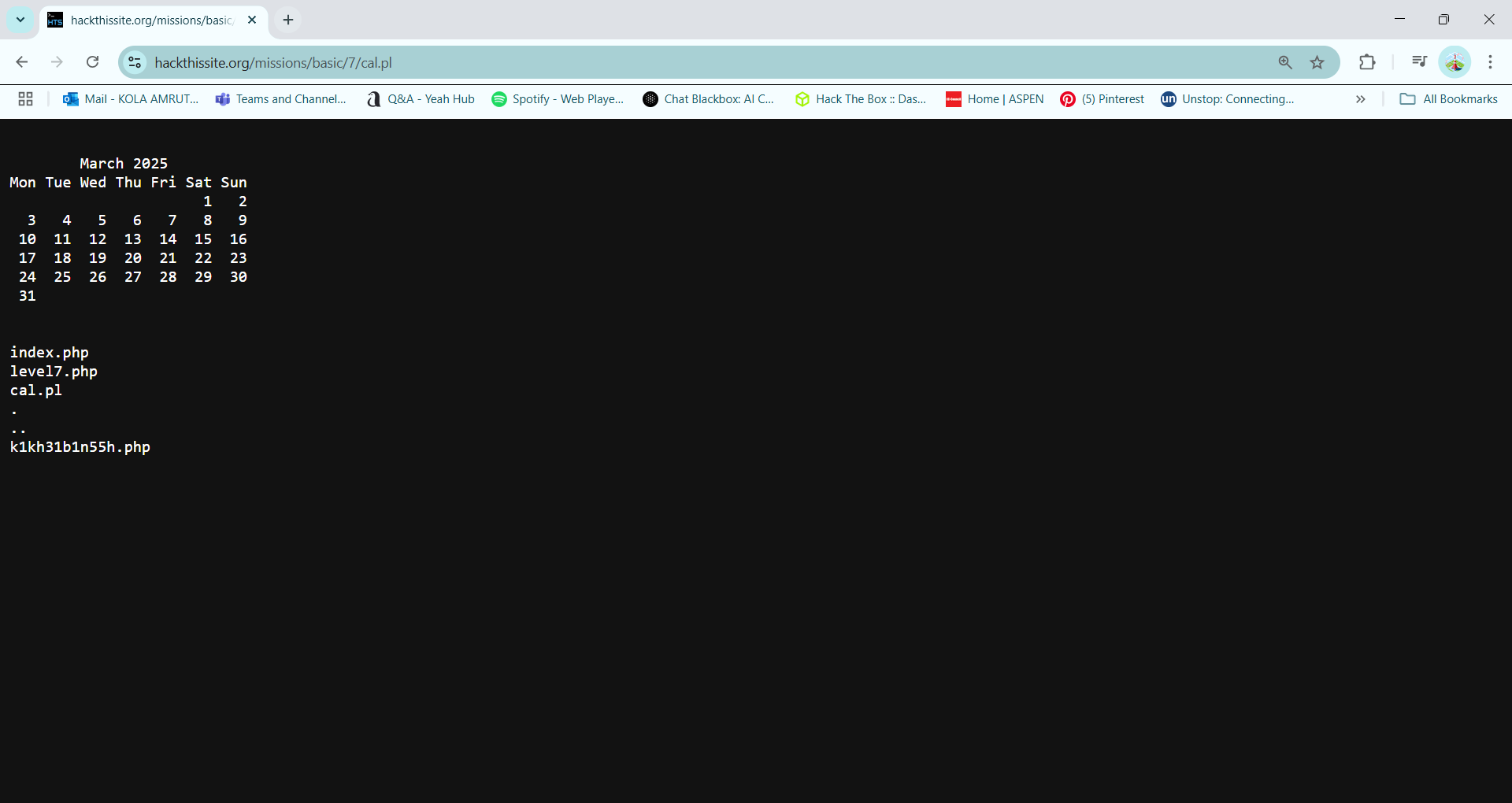
| **Character** | **ASCII (Encrypted)** | **Position** | **Decrypted ASCII** | **Decrypted Character** |
| --- | --- | --- | --- | --- |
| f | 102 | 0 | 102 - 0 = **102** | f |
| 6 | 54 | 1 | 54 - 1 = **53** | 5 |
| : | 58 | 2 | 58 - 2 = **56** | 8 |
| h | 104 | 3 | 104 - 3 = **101** | e |
| h | 104 | 4 | 104 - 4 = **100** | d |
| > | 62 | 5 | 62 - 5 = **57** | 9 |
| j | 106 | 6 | 106 - 6 = **100** | d |
| < | 60 | 7 | 60 - 7 = **53** | 5 |

### Level 7 - Command Injection (Linux 'cal' command)

Vulnerability: User input directly interacting with system commands.

Exploit Method: Running Linux commands using input fields.

Recommendation: Sanitize and validate user inputs to prevent command execution.



### Level 8 - Server-Side Includes (SSI) Injection

Vulnerability: Web application vulnerable to SSI injection.

Exploit Method: Mission 8 asks us for our name.  When we enter our name, it adds it to a document.  If we use SSI, we can also get a directory listing (ls).

First, test the script by putting your name in it.

Next, enter the following

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

Space is very important.  It gives us a listing of the files in this directory.  To do a listing of the parent directory, we merely modify the above

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

Recommendation: Disable SSI processing if not required.

### Level 9 - SSI Directory Traversal

Vulnerability: SSI allowing access to sensitive files.

Exploit Method: Mission 9 is very similar, and uses, mission 8.  To get this password, we simply need to use the ls command SSI injection from mission 8 to go to the directory for mission 9.  Simply go back to basic missions, select mission 8, and enter the following in the name field:

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

This gives us the ability to go up to the directory for 8, the directory above 8 called basic, and into the directory for 9.  It gives us the following listing:

Hi, index.php p91e283zc3.php!

Your name contains 24 characters.  
You can then use p91e283zc3.php to open up the password file.  It gives us the link:  
https://www.hackthissite.org/missions/basic/9/p91e283zc3.php

Recommendation: Implement strict access controls and disable unnecessary directives.

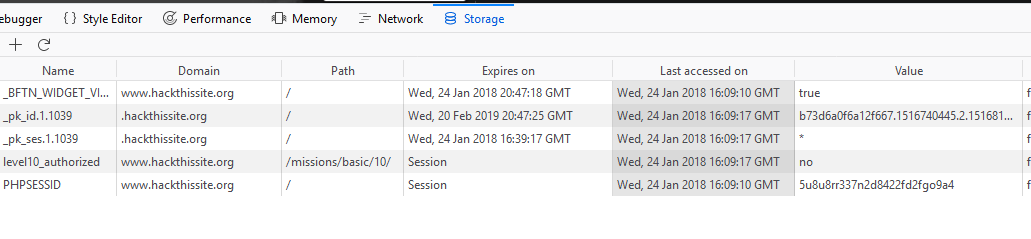
### Level 10 - Cookie Manipulation

Vulnerability: Authentication mechanism relying on modifiable cookies.

Exploit Method: Editing browser cookies to gain unauthorized access.

Mission 10 uses javascript and cookies for loggin in.  It is simply a case of modifying your login session to yes.

This time we are going to use the Firefox built in Web development tools, but we are going to use the Storage tab.  Best achieved by hitting Shift + F9 on your key board, and look for a cookie named “level10\_authorized” as below:



One of the values is the cookie, and for the cookie, one of the entries is level10\_authorized.  It is set to no, double click on the word “no” and set it to yes, click away, and click Submit on the form

### Level 11 - .htaccess Misconfiguration

Vulnerability: Weak .htaccess configurations allowing unauthorized access.

Exploit Method: Modifying HTTP headers or URLs to bypass authentication.

Every time we refresh new song is displayed those are songs of Elton john. we try to bypass http header with these letters, .htaccess file enforcing it shows the folder , to the answer, we can try navigating to index.php to submit the answer

Recommendation: Properly configure `.htaccess` files to enforce access control rules.

## 4. Conclusion & Recommendations

Through this penetration testing exercise, we identified multiple security vulnerabilities in the target web application. These vulnerabilities highlight common web security flaws that can be exploited by attackers. Implementing the following recommendations can significantly enhance security:

Secure Authentication Mechanisms: Ensure passwords are never stored in client-side code or transmitted in plaintext.

Input Validation: Implement proper input sanitization to prevent SQL injection, XSS, and command injection attacks.

Server-Side Security: Move all critical security checks and authentication mechanisms to the server side.

Use Secure Cookies: Implement HttpOnly, Secure, and SameSite attributes to protect session cookies.

Access Control Policies: Properly configure .htaccess and ensure least privilege principles for user access.

By following these recommendations, web applications can significantly reduce their attack surface and protect against common vulnerabilities.