CTF Report: Discovering Hidden Flags via File Includes

Ayoub Goubraim

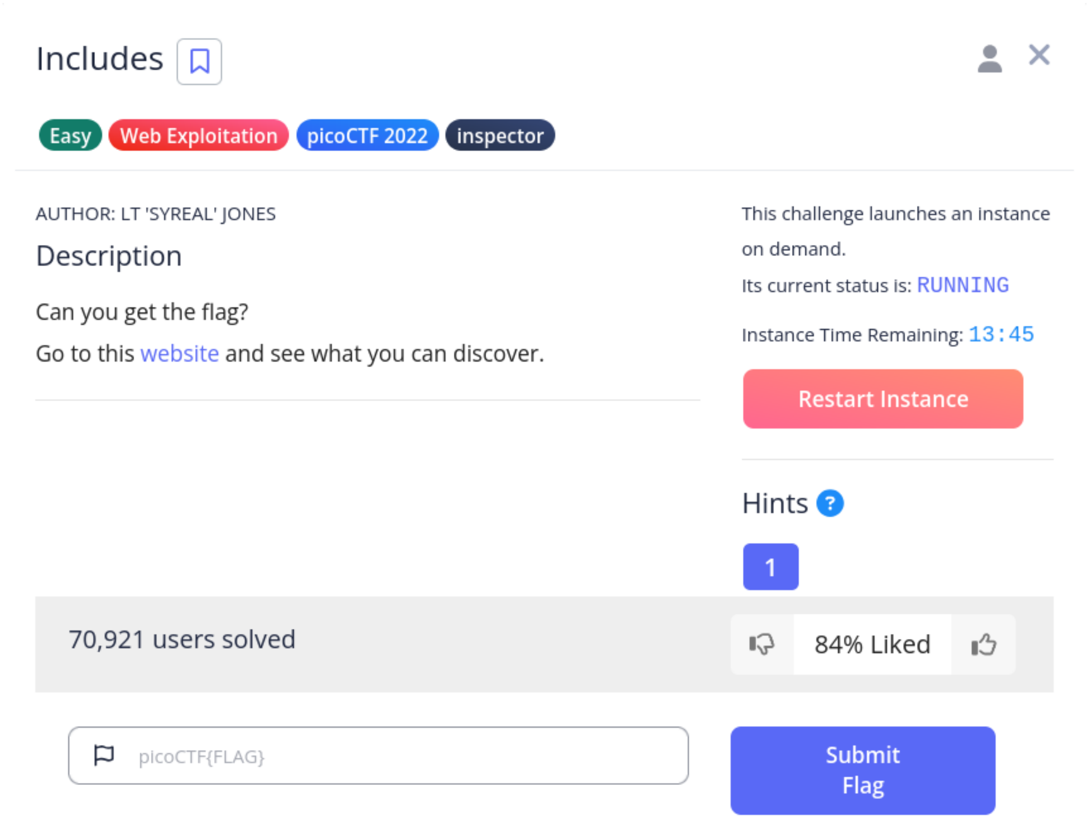
September 9, 2025

Abstract

This report documents the picoCTF 2022 challenge **Includes**. The exercise demonstrates how inspecting external resources (JavaScript and CSS includes) can reveal hidden information. By carefully analyzing linked files, we piece together fragments of the flag and reconstruct the complete solution.

1 Challenge Overview

The challenge statement asks if we can find the flag by exploring the website and its resources.



Includes

Easy | Web Exploitation | picoCTF 2022 | inspector

AUTHOR: LT 'SYREAL' JONES

Description

Can you get the flag?

Go to this website and see what you can discover.

This challenge launches an instance on demand.

Its current status is: RUNNING

Instance Time Remaining: 13:45

70,921 users solved

84% Liked

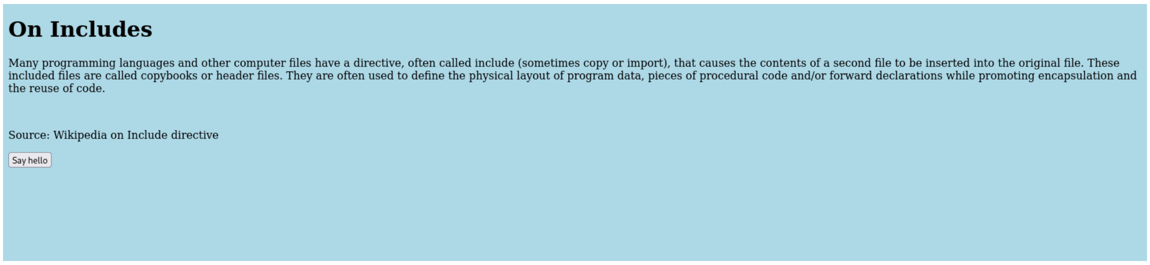
Figure 1: Challenge description for *Includes*.

2 Environment & Tools

- **OS:** Kali Linux.
- **Browser:** Firefox with Developer Tools (Inspector).

3 Reconnaissance

Navigating to the website shows a simple homepage with minimal content. Nothing obvious points to the flag.



On Includes

Many programming languages and other computer files have a directive, often called include (sometimes copy or import), that causes the contents of a second file to be inserted into the original file. These included files are called copybooks or header files. They are often used to define the physical layout of program data, pieces of procedural code and/or forward declarations while promoting encapsulation and the reuse of code.

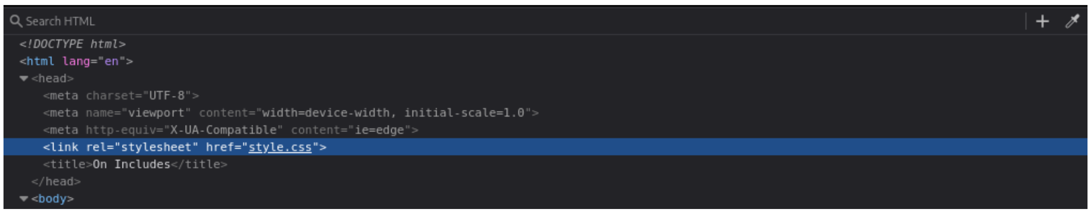
Source: Wikipedia on Include directive

Figure 2: Homepage of the challenge site.

4 Inspecting Source Code

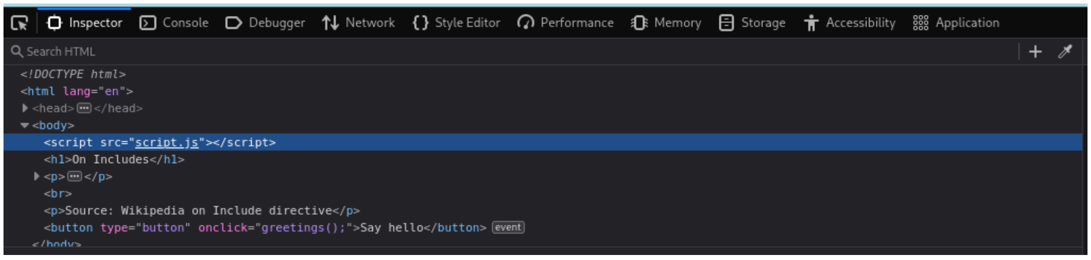
Opening the DOM in the Web Inspector reveals that two files are included:

- `style.css`
- `script.js`



```
<!DOCTYPE html>
<html lang="en">
▾<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <meta http-equiv="X-UA-Compatible" content="ie=edge">
   <link rel="stylesheet" href="style.css">
   <title>On Includes</title>
  </head>
▾<body>
```

Figure 3: Inspector showing `style.css` included in the HTML.

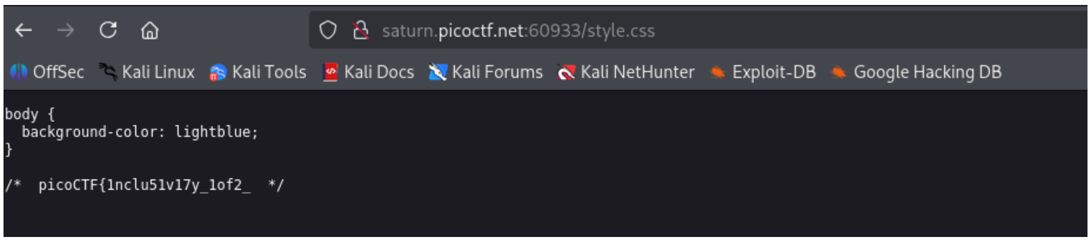


```
<!DOCTYPE html>
<html lang="en">
▸<head>…</head>
▾<body>
   <script src="script.js"></script>
   <h1>On Includes</h1>
  ▸<p>…</p>
   <br>
   <p>Source: Wikipedia on Include directive</p>
   <button type="button" onclick="greetings();">Say hello</button> event
  </body>
```

Figure 4: Inspector showing `script.js` included in the HTML.

5 Exploring Included Files

The `style.css` file contains a hidden comment with the first fragment of the flag.

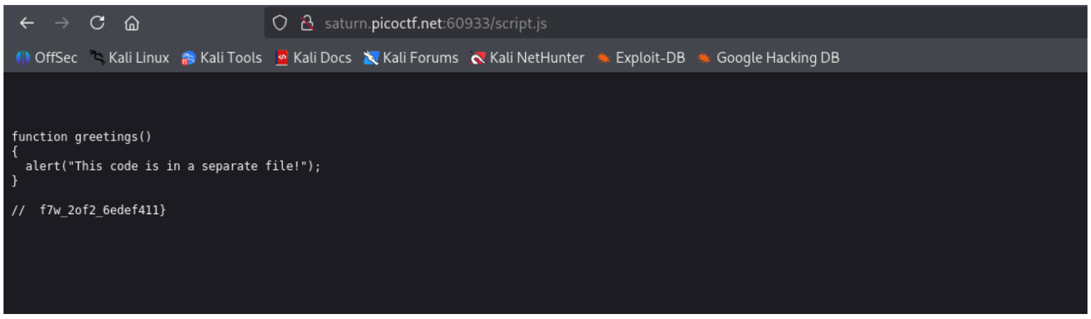


```
body {
  background-color: lightblue;
}

/*  picoCTF{1nclu51v17y_1of2_  */
```

Figure 5: Inspecting `style.css` reveals the first part of the flag.

The `script.js` file contains another fragment, hidden as a comment at the bottom.



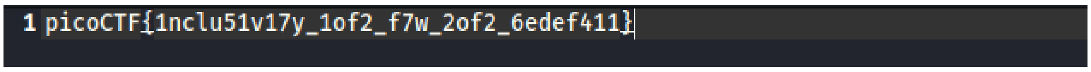
```
function greetings()
{
  alert("This code is in a separate file!");
}

//  f7w_2of2_6edef411}
```

Figure 6: Inspecting `script.js` reveals the second part of the flag.

6 Flag Reconstruction

By concatenating both fragments, we obtain the complete flag.



```
picoCTF{1nclu51v17y_1of2_f7w_2of2_6edef411}
```

Figure 7: The final flag assembled from both fragments.

7 Discussion & Takeaways

This challenge illustrates:

- Always inspect external resources (`.js`, `.css`, comments).
- Sensitive data should never be hidden in client-side files.
- Proper content auditing and code reviews are essential to prevent leakage.

8 Conclusion

The *Includes* challenge highlights the importance of not relying on client-side secrecy. By checking CSS and JavaScript includes, we successfully recovered all parts of the flag.