Report on Reflected Cross-site Scripting Attack

Tasnim Ahmed

DVA489 – Web Security

October 4, 2021

***For ease of Implementation I have used only one third party library (**Nodemon**) so that I do not have to restart the server manually every time I make any small change.

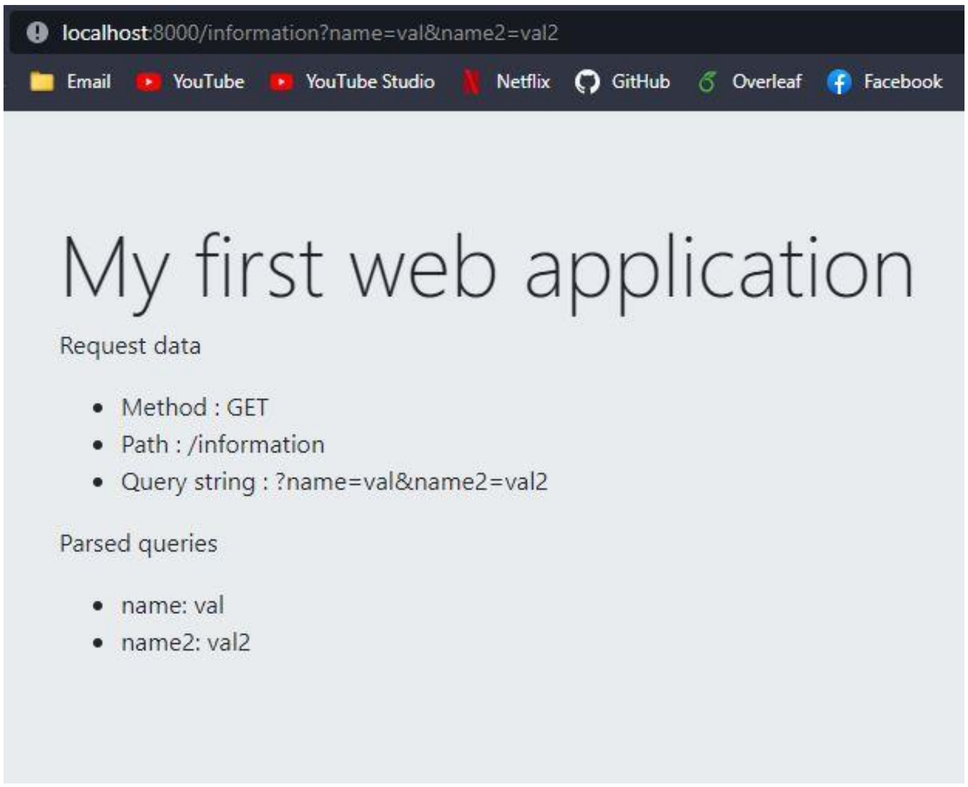
***Github Repo: Assignment One

What is Reflected Cross-site Scripting attack?

Cross-site Scripting (XSS) attacks include injecting data into a web application in such a way that it is executed on the victim's PC. Malicious information is typically written in JavaScript, but it is not restricted to that. Any content that is performed on the victim's PC, such as Flash, can be used. However, we will concentrate on JavaScript injection in this attack. XSS attacks are potent attacks that can be used to steal personal information, redirect the victim to an attacker-controlled site, or do other actions while posing as a benign website. The possibilities are almost endless.

Implementation of the Attack

When a user goes to the 'information' route, then he can see the key-value parameters as the output. An example is depicted below.



My first web application

Request data

- Method : GET
- Path : /information
- Query string : ?name=val&name2=val2

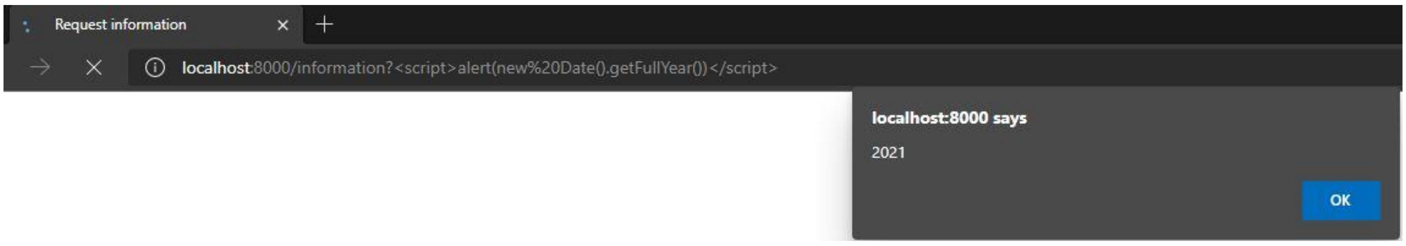
Parsed queries

- name: val
- name2: val2

Here, the query parameters are 'name=val&name2=val2'. This query parameters are directly replaced inside 'information.html' using the following lines of code without checking if there is any special characters inside the queries.

```
content = content.replace("{{method}}", req.method);
content = content.replace("{{path}}", PathName);
content = content.replace("{{query}}", reqURL.search);
content = content.replace("{{queries}}", queries);
```

Now, if I replace the query with a small javascript (In this case, '<script>alert(new Date().getFullYear())</script>'), then the contents of the script will run on the client side. An example is given below.



localhost:8000/information?<script>alert(new%20Date().getFullYear())</script>

localhost:8000 says

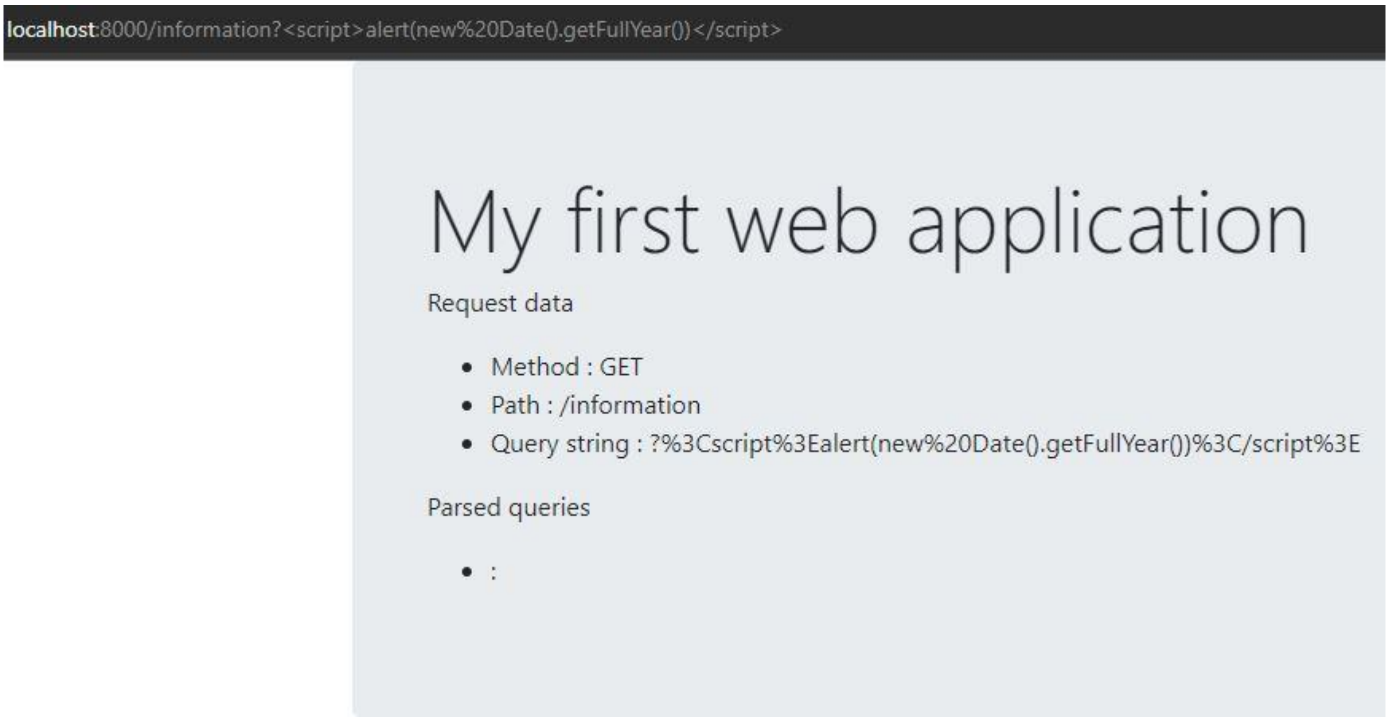
2021

Protecting the Website from XSS Attacks

To protect the website from XSS attacks we have replaced special characters using the following character escaping function.

```
const escaping = (params) => {
  let new_params = params;
  new_params = new_params.replace("<", "&lt");
  new_params = new_params.replace(">", "&gt");
  new_params = new_params.replace("&", "&amp");
  new_params = new_params.replace('"', "&quot");
  new_params = new_params.replace("'", "&#x27");
  return new_params;
};
```

If the query parameters contain any special characters or vulnerable characters it will be replaced according to the function. After getting replaced, the query parameters will be shown in the ‘information route’. An example is given below.



localhost:8000/information?<script>alert(new%20Date().getFullYear())</script>

My first web application

Request data

- Method : GET
- Path : /information
- Query string : ?%3Cscript%3Ealert(new%20Date().getFullYear())%3C/script%3E

Parsed queries

- :

The final patched code for the server function is given below.

```
const server = http.createServer((req, res) => {
  let flag = false;
  let extname = String(path.extname(req.url)).toLowerCase();
  let reqURL = url.parse(req.url, true);
  let PathName = decodeURIComponent(reqURL.pathname);
  if (extname == ".js" || extname == ".css" || extname == ".html") {
    flag = true;
    filePath = "public/" + req.url;
    try {
      content = fs.readFileSync(filePath, { encoding: "utf-8" });
      contentType = mimeTypes[extname];
    } catch (error) {
      content = "<H1>Error 404!</H1>";
      contentType = "text/html";
    }
    sendResponse(res);
  }
  if (PathName == "/") {
    flag = true;
    if (req.method == "POST") {
      let data = "";
      req.on("data", (chunk) => {
        data += chunk;
      });
      req.on("end", () => {
        console.log(data);
      });
    }
    content = fs.readFileSync("public/index.html", { encoding: "utf-8" });
    contentType = "text/html";
    sendResponse(res);
  }
  if (PathName == "/information") {
    flag = true;
    let params = reqURL.query;
    content = fs.readFileSync("templates/information.html", {
      encoding: "utf-8",
    });
    let queries = "";
    Object.keys(params).forEach((item) => {
      queries +=
        "<li>" + escaping(item) + ": " + escaping(params[item]) + "</li>";
    });
    content = content.replace("{{method}}", req.method);
    content = content.replace("{{path}}", PathName);
    content = content.replace("{{query}}", reqURL.search);
    content = content.replace("{{queries}}", queries);
    contentType = "text/html";
    sendResponse(res);
  }
  if (flag == false) {
    content = "<H1>Error 404!</H1>";
    contentType = "text/html";
    sendResponse(res);
  }
  res.end();
});
```