Report on Denial of Service 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 Denial of Service attack?

The goal of a Denial of Service (DoS) attack is to make a resource (a website, an application, or a server) inaccessible for the reason for which it was created. By manipulating network packets, programming, logical, or resource management weaknesses, among other things, it is possible to render a service unavailable to authorized users. If a service receives a huge number of requests, it may be removed off the market for legitimate users. A service may also come to a halt if a programming flaw is exploited, or if the way the service manages the resources it utilizes is compromised.

Implementation of the Attack

My website currently has two routes. One is the base route (‘/’) and another one is the Information route (‘/information’). Other than this, if the user requests for any content which is available under my public directory e.g., ‘index.html’, ‘.js files’, and ‘.css files’, the ‘server’ function will provide that content. But if the user requests for any resource which is not available, e.g., ‘index1.html’, ‘abc.dc’, etc. Then the server will crash. Eventually it will stop providing services to the subsequent users. The code snippet for ‘server’ function inside ‘httpd.js’ file is given below.

```
const server = http.createServer((req, res) => {
  let extname = String(path.extname(req.url)).toLowerCase();
  let reqURL = url.parse(req.url, true);
  let PathName = decodeURIComponent(reqURL.pathname);
  if (extname != ".ico" && extname != "") {
    filePath = "public/" + req.url;
    content = fs.readFileSync(filePath, { encoding: "utf-8" });
    contentType = mimeTypes[extname];
    sendResponse(res);
  }
  if (PathName == "/") {
    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") {
    const params = reqURL.query;
    content = fs.readFileSync("templates/information.html", {encoding: "utf-
8",});
    let queries = "";
    Object.keys(params).forEach((item) => {
      queries += "<li>" + item + ": " + 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);
  }
  res.end();
});
```

Protecting the Website from DoS Attacks

Now, we can prevent DoS attacks by adding some checks. If the resource the user is looking for is unavailable. We will show an Error 404. So, while searching for the resource in the server directory using the filesystem library of NodeJS, we will use a try-catch block. In the catch block, the content to be shown will be replaced with "Now, we can prevent DoS attacks by adding some checks. If the resource the user is looking for is unavailable. We will show an Error 404. So, while searching for the resource in the server directory using the filesystem library of NodeJS, we will use a try-catch block. In the catch block, the content to be shown will be replaced with <H1>Error 404!</H1>. The code snippet for 'server' function inside the 'httpd-patched.js' file is given below. If the requested resource or route does not match anything available then the flag will remain constant and <H1>Error 404!</H1> will be shown instead of stopping the server so that other users can get uninterrupted service.

```
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();
});
```