# Burp Suite

Please refer to binder: Burp Suite for more information on settings etc. Used in conjunction with firefox and the foxyproxy extension.

Run firefox - Foxyproxy addon top right.
“Add” to start, this allows us to customise our own proxy settings.

1. Add title eg: Burp
2. Proxy ip eg: 127.0.0.1
3. Port eg: 8080
4. Save the configuration
5. Activate proxy configuration.
6. Reopen foxyproxy top right on the extension and select the saved preset.

Once done, this will **redirect browser traffic through the IP on the proxy.** Burp suite must also be running in order for this to work.

Once they are both running, search a website and boom! Method: GET / POST for example, it will capture these requests.

**Remember the following:**

- When the proxy configuration is active, and the intercept is switched on in Burp Suite, your browser will hang whenever you make a request.
- Be cautious not to leave the intercept switched on unintentionally, as it can prevent your browser from making any requests.
- Right-clicking on a request in Burp Suite allows you to perform various actions, such as forwarding, dropping, sending to other tools, or selecting options from the right-click menu.

TryHackMe | Burp Suite: The Basics

Target tab provides 3 key sub-tabs.

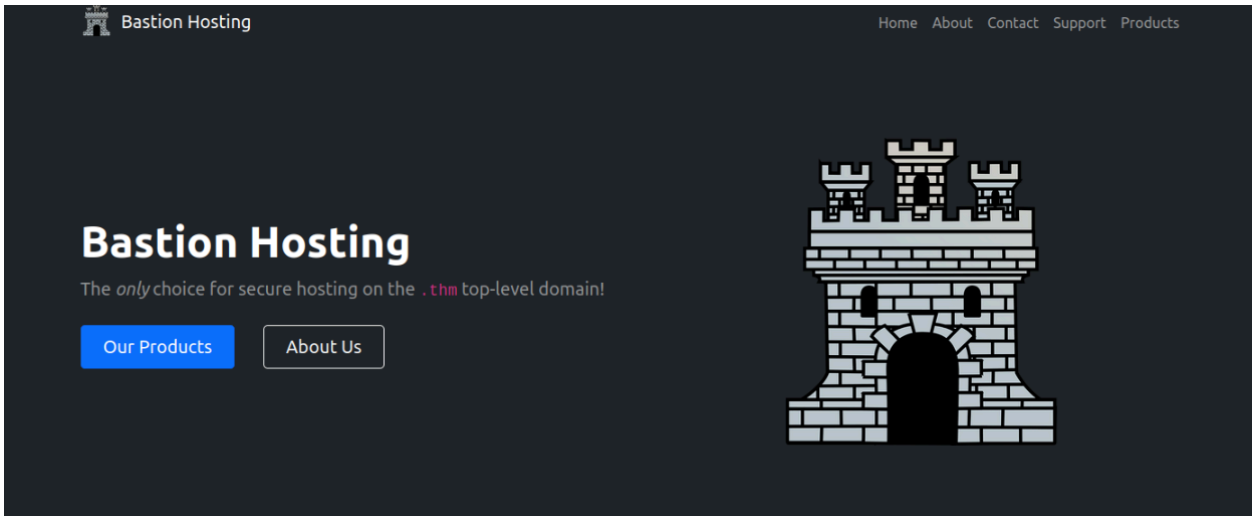
**Site Map:** Map out web applications we are targeting in a tree structure. Every page we visit while the proxy is active will be displayed on the site map. This enables us to automatically generate a site map simply by browsing the web application.

**Issue Definition:** Provides an extensive list of web vulnerabilities complete with descriptions and references. Can be valuable when assessing risks and referencing vulnerabilities when reporting / describing vulnerabilities that may have been identified.

**Scope Settings:** Control the target scope. Include or exclude certain domains/IPs to define the scope of our testing. By managing the scope we can focus on the web applications we are specifically targeting to avoid unnecessary traffic.

`http://10.10.22.39/`

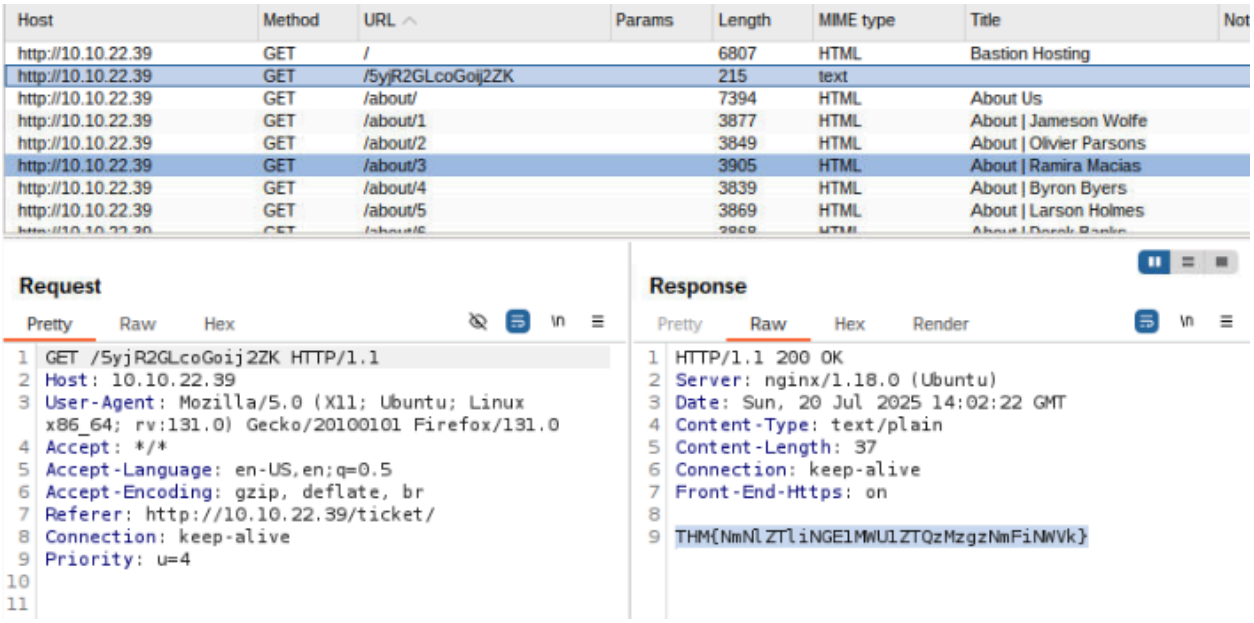
Tryhack me gave me this link which gave me a website




I clicked on every single link and saw in the "Target Scope" site map begin to map every link associated with the website, and it was starting to paint a picture (metaphorically) into what information it was getting from me surfing the web page. These were all GET requests because it was me manually clicking everything.

So why is this helpful? Because it avoids us needing to write everything down, every link, every integrated page within the domain, and instead does it all for us. It also tells us, what the type is, in this case the THM labs was asking for:
What is the flag you receive after visiting the unusual endpoint?

The unusual endpoint I found was this random .txt file lingering inside the web page!



| Host | Method | URL | Params | Length | MIME type | Title | Not |
|---|---|---|---|---|---|---|---|
| http://10.10.22.39 | GET | / | | 6807 | HTML | Bastion Hosting | |
| http://10.10.22.39 | GET | /5yjR2GLcoGoij2ZK | | 215 | text | | |
| http://10.10.22.39 | GET | /about/ | | 7394 | HTML | About Us | |
| http://10.10.22.39 | GET | /about/1 | | 3877 | HTML | About \| Jameson Wolfe | |
| http://10.10.22.39 | GET | /about/2 | | 3849 | HTML | About \| Olivier Parsons | |
| http://10.10.22.39 | GET | /about/3 | | 3905 | HTML | About \| Ramira Macias | |
| http://10.10.22.39 | GET | /about/4 | | 3839 | HTML | About \| Byron Byers | |
| http://10.10.22.39 | GET | /about/5 | | 3869 | HTML | About \| Larson Holmes | |

**Request**

Pretty Raw Hex

```
1 GET /5yjR2GLcoGoij2ZK HTTP/1.1
2 Host: 10.10.22.39
3 User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux
  x86_64; rv:131.0) Gecko/20100101 Firefox/131.0
4 Accept: */*
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate, br
7 Referer: http://10.10.22.39/ticket/
8 Connection: keep-alive
9 Priority: u=4
10
11
```

**Response**

Pretty Raw Hex Render

```
1 HTTP/1.1 200 OK
2 Server: nginx/1.18.0 (Ubuntu)
3 Date: Sun, 20 Jul 2025 14:02:22 GMT
4 Content-Type: text/plain
5 Content-Length: 37
6 Connection: keep-alive
7 Front-End-Https: on
8
9 THM{NmNlZTliNGE1MWU1ZTQzMzgzNmFiNWVk}
```

I clicked the link and bam! It revealed the contents to me within the .txt file which I could then extract.

However, if you are running Burp Suite on Linux as the root user (as is the case with the AttackBox), you may encounter an error preventing the Burp Browser from starting due to the inability to create a sandbox environment.

There are two simple solutions to this:

1. **Smart option:** Create a new user and run Burp Suite under a low-privilege account to allow the Burp Browser to run without issues.
2. **Easy option:** Go to `Settings -> Tools -> Burp's browser` and check the `Allow Burp's browser to run without a sandbox` option. Enabling this option will allow the browser to start without a sandbox. However, please be aware that this option is disabled by default for security reasons. If you choose to enable it, exercise caution, as compromising the browser could grant an attacker access to your entire machine. In the training environment of the AttackBox, this is unlikely to be a significant issue, but use it responsibly.

I read this message and rest assured I will only stick to the basics for now… I have no intention of doing any of this outside of the sandbox environment as I do not have a VPN.

So now we have done that task we move onto scoping and why it is important. Scoping is when we want to save being overwhelmed by the potentially insanely large number of requests. So scoping means we can focus on the host URLs we want to focus on!

To do this:

1. On site map right click the host URL we wish to scope and select “Add to scope”
2. Simply go to scope and there it is!

However, even if you disabled logging for out-of-scope traffic, the proxy will still intercept everything. To prevent this, go to Proxy settings.

Burp suite - Settings - Proxy - Request Interception Rules + Response Interception Rules - Enable “And” operators “URL” is in the target scope.

**Enabling this ensures that the proxy ignored all other traffic that is not within the defined scope! Makes it much cleaner and clearer!**

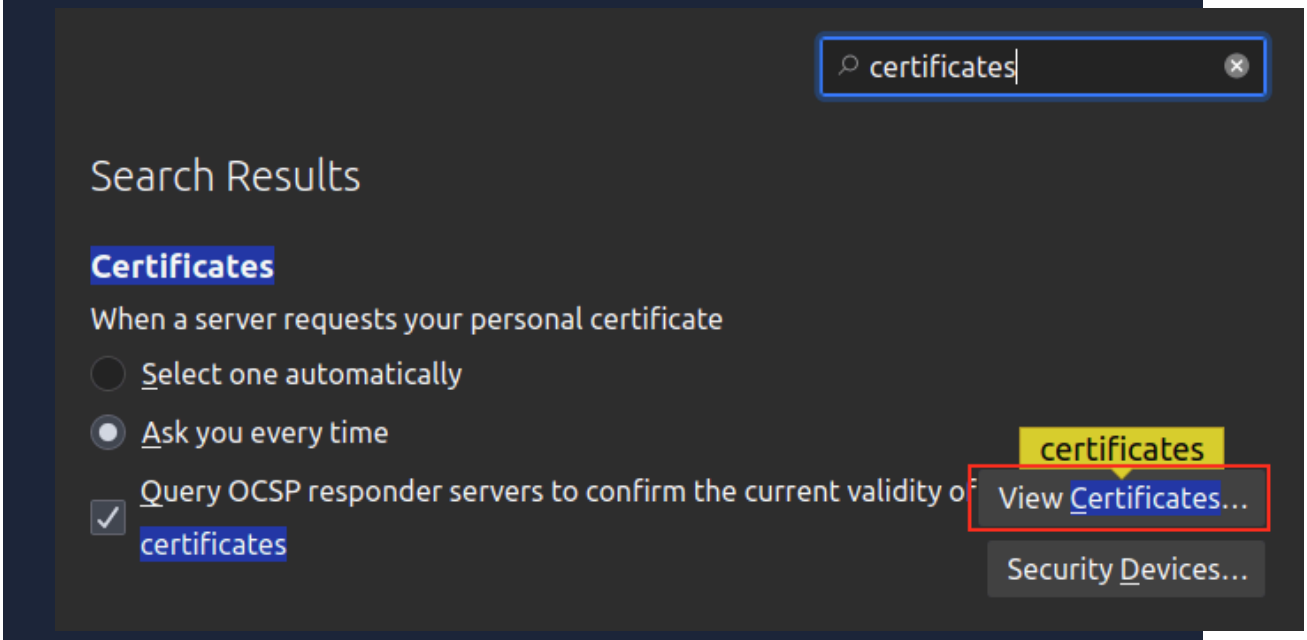
When intercepting HTTP traffic we may encounter an issue when navigating to sites with TSL enabled. Eg, Google.com may receive an error saying PortSwigger Certified Authority is not authorised to secure the connection.

**Steps to overcome:**

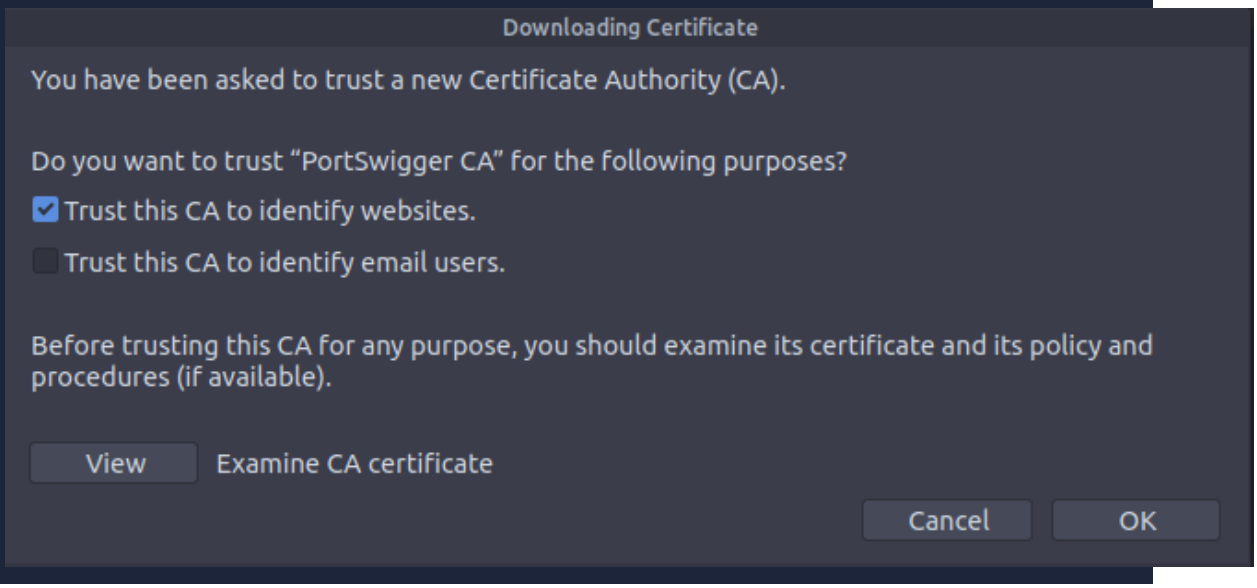
**To overcome this issue, we can manually add the PortSwigger CA certificate to our browser's list of trusted certificate authorities. Here's how to do it:**

1. **Download the CA Certificate: With the Burp Proxy activated, navigate to http://burp/cert. This will download a file called `cacert.der`. Save this file somewhere on your machine.**
2. **Access Firefox Certificate Settings: Type `about:preferences` into your Firefox URL bar and press Enter. This will take you to the Firefox settings page. Search the page for "certificates" and click on the View Certificates**

button.




3. Import the CA Certificate: In the Certificate Manager window, click on the Import button. Select the `cacert.der` file that you downloaded in the previous step.
4. Set Trust for the CA Certificate: In the subsequent window that appears, check the box that says "Trust this CA to identify websites" and click OK.

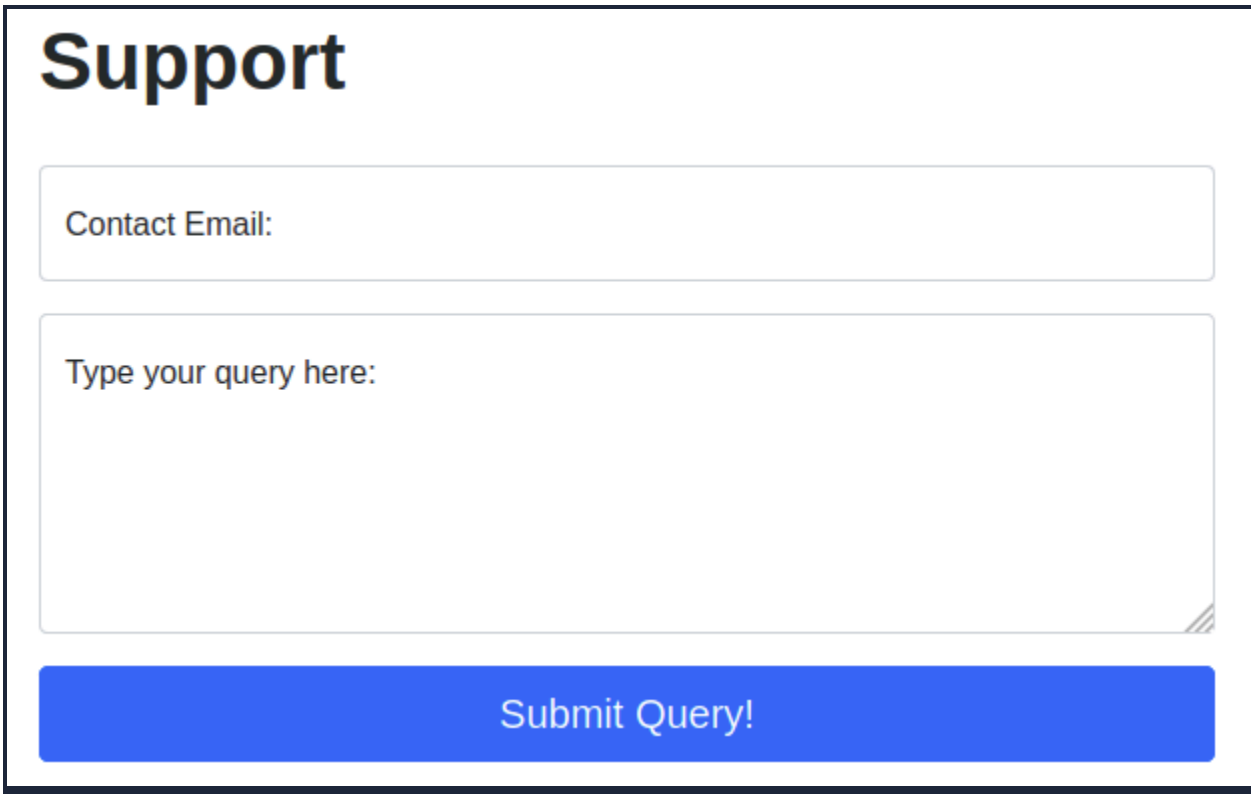



By completing these steps, we have added the PortSwigger CA certificate to our list of trusted certificate authorities. Now, we should be able to visit any TLS-enabled site without encountering the certificate error.

**Example Attack:**

Having looked at how to set up and configure our proxy, let's go through a simplified real-world example.

We will start by taking a look at the support form at `http://10.10.22.39/ticket/`:




In a real-world web app pentest, we would test this for a variety of things, one of which would be Cross-Site Scripting (or XSS). If you have not yet encountered XSS, it can be thought of as injecting a client-side script (usually in Javascript) into a webpage in such a way that it executes. There are various kinds of XSS – the type that we are using here is referred to as "Reflected" XSS, as it only affects the person making the web request.

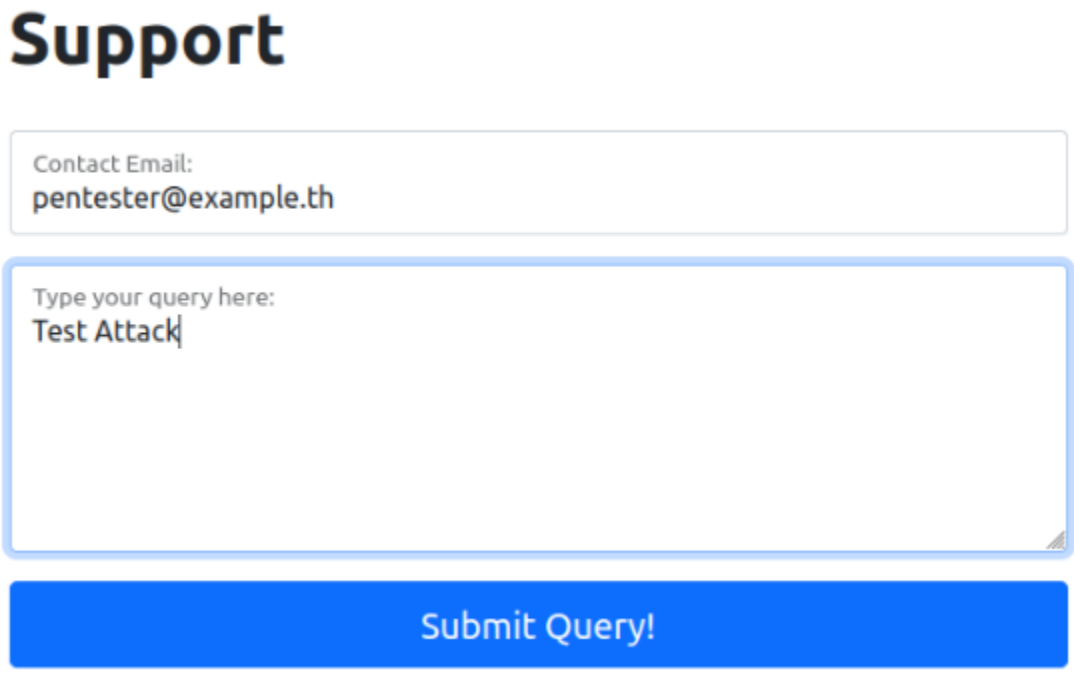
Walkthrough

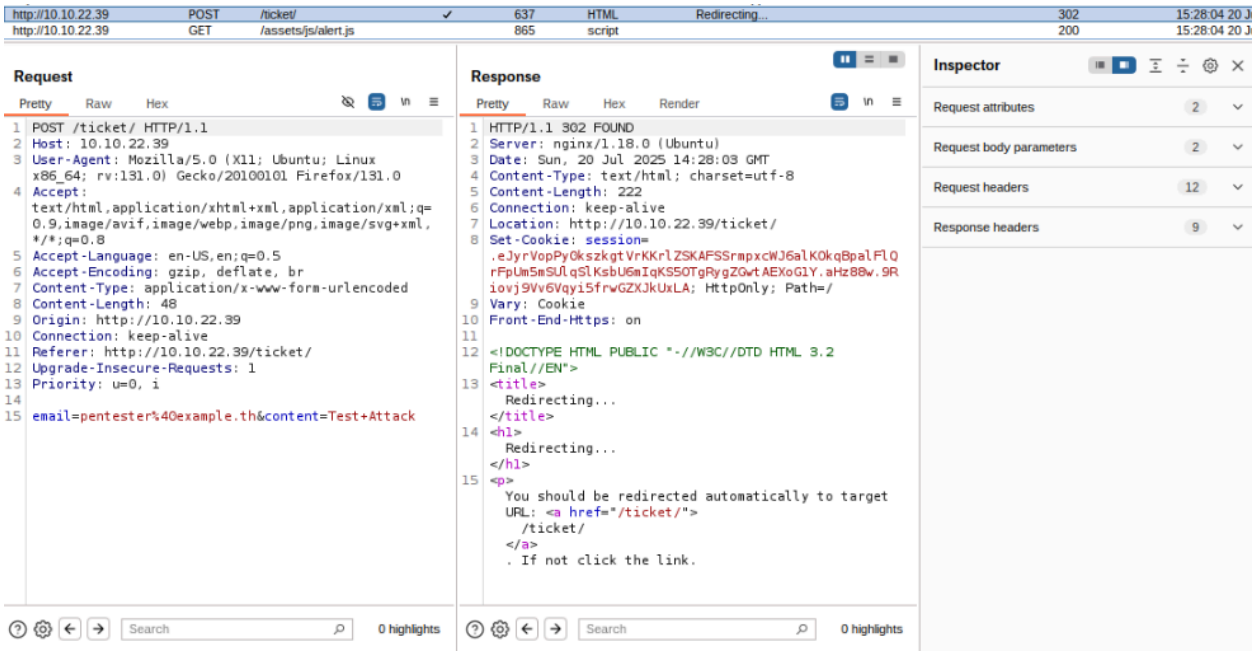
Try typing: `<script>alert("Succ3ssful XSS")</script>`, into the "Contact Email" field. You should find that there is a client-side filter in place which

prevents you from adding any special characters that aren't allowed in email addresses:

**So the script doesn't work, which indicates to me there is some sort of advanced filtering at play here, the web developer may have thought about this.**

**Now I am going to use Burp to try and get over this task, and by pass this client.**



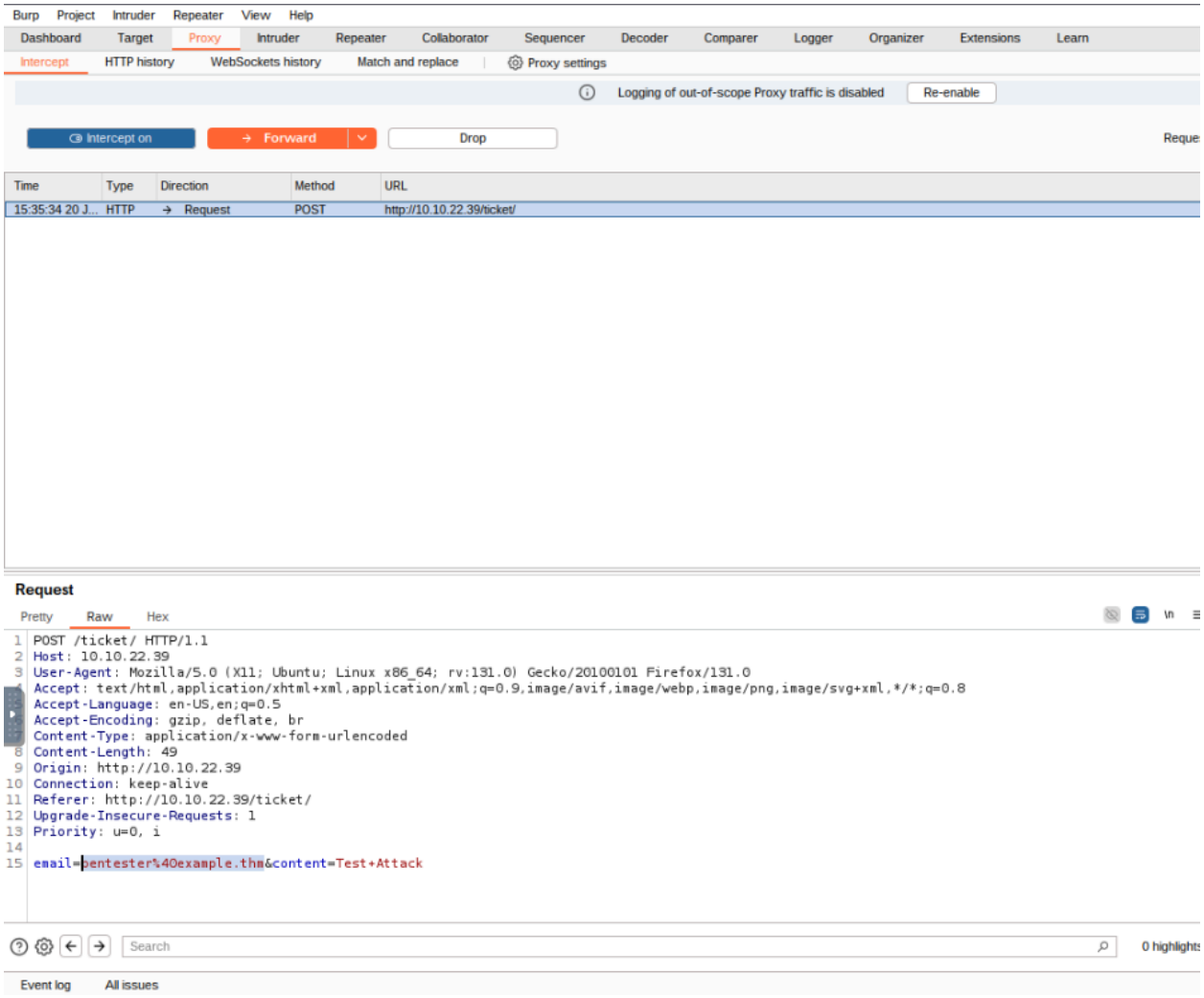



**Please zoom in if unable to read!**

So as we can see, the request went through! As per the instruction our email is visible, so we can manipulate the email link from.
email=pentester%40example.th&content=Test+Attack
Injecting our own payload into it:

First, I submitted the query again with the intercept on so it could capture the intercept request I made with the email and query string.

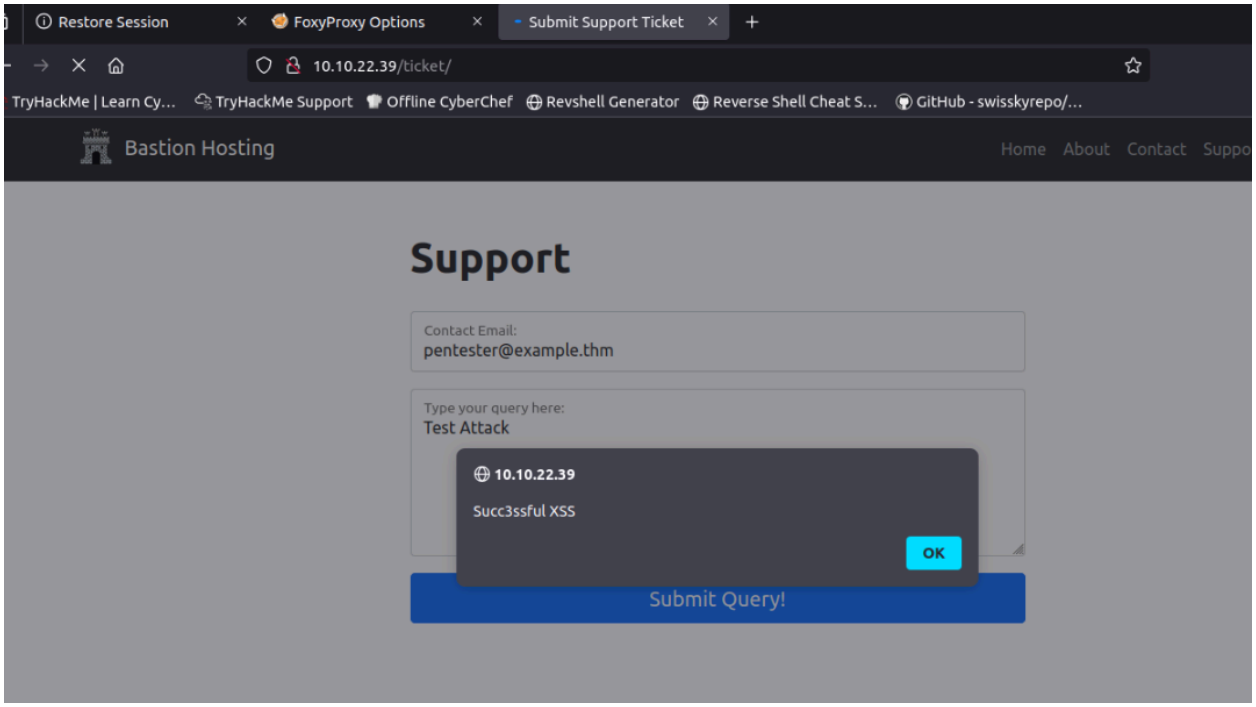
Then with the captured request it has essentially “Paused” the servers response and we are sneaking around in the middle.
So I go into the request field and change the email section with our payload.




`<script>alert("Succ3ssful XSS")</script>`

Is what we will be changing it to.

Like so!
Then we select “Forward” which essentially resumes the process again.




Exceptionally done. We completed a XSS attack by simply pausing the web page, changing it in transit, then forwarding it on when we were done with it. Very sneaky indeed.

**Conclusion:**

So I was in the middle of the Pen Tester path and I kept seeing the word “Burp” being thrown around and realised it wasn’t just trapped gas! In all seriousness, I kept getting stuck on tasks and peeked at a few youtube videos to see what I was doing wrong in comparison, often it was because they were flying through it with “Burp Suite” which I did see the extension on my browser but being so unfamiliar with it, and it not being directly mentioned in the TryHackMe contents I ignored it.
So, with this tool, I can now confidently say, it makes the process of penetrating a lot easier. Because not only does it map out what the contents are, but we can easily pause requests in such a smooth and clear way! For example, before I was using debugging and web developer tools, and viewing the page source. This allowed me to complete the task but this tool certainly simplified the process. I cannot wait to apply this knowledge to later tasks.