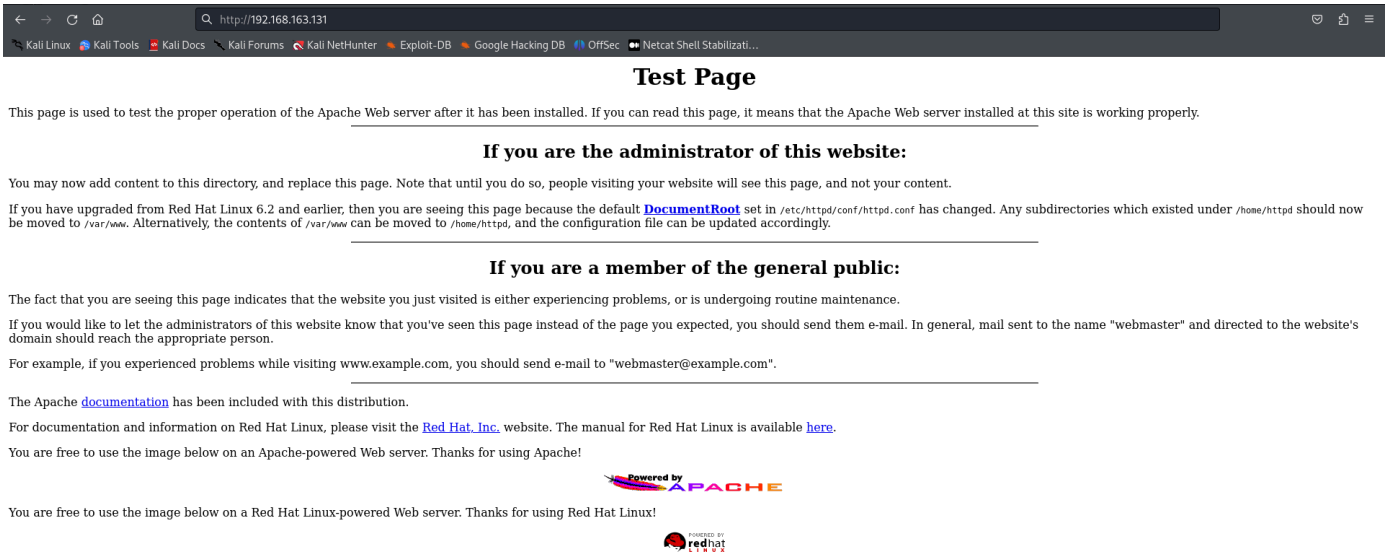
HTTP/HTTPS

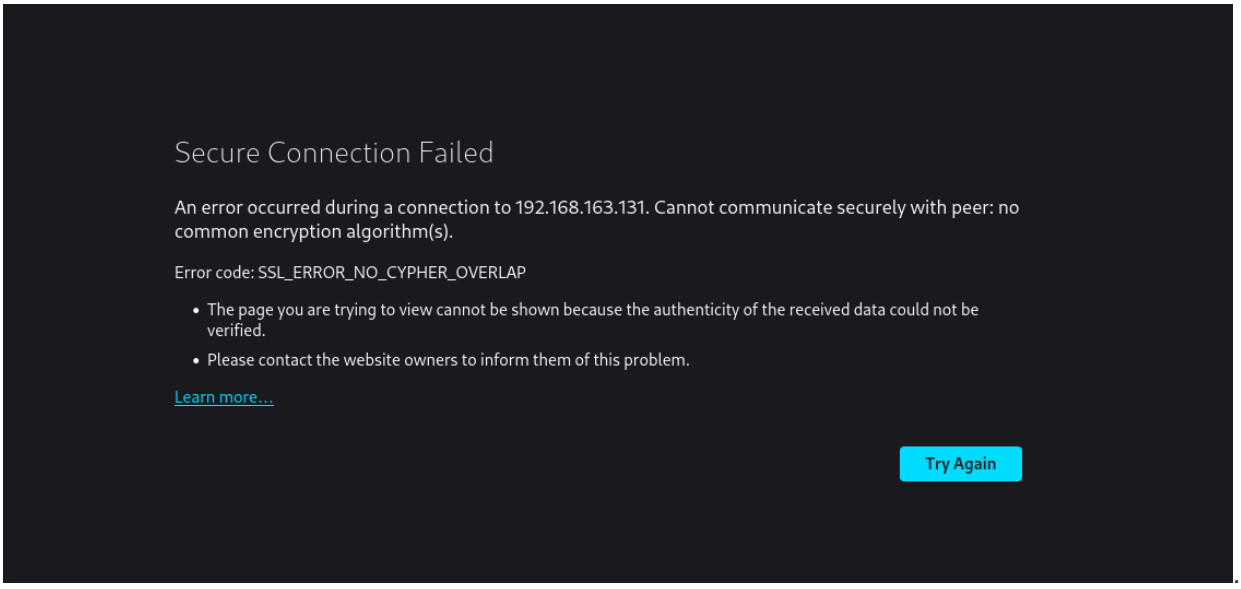
We are going to start enumerating both of these services: HTTP, and HTTPS.

Port 80, and 443.

First thing is to go to the website. We know there are both 80, and 443, so we request "http://TARGET_IP_ADDRESS", and "https://TARGET_IP_ADDRESS". And, take a look at their website, just to have a feeling. In this case, we see the website on the HTTP protocol as shown in the image



, but the HTTPS does not seem to load, and it does not seem to have a way to "continue" to navigate to the site, even if it is not secure. The following is the error message when I try to access HTTPS:



When I click in the learn more, I am brought to the Firefox website providing more information on the problem.

I tried to fix the issues in a couple ways, so here is what I did:

6. Modify Firefox Security Settings

Temporarily modify security settings to bypass the error:

- Open Firefox.
- In the address bar, type `about:config` and press Enter.
- Click "Accept the Risk and Continue."
- Search for `security.ssl.enable_ocsp_stapling` and set it to `false` by double-clicking it.
- Search for `security.OCSP.require` and set it to `false` by double-clicking it.

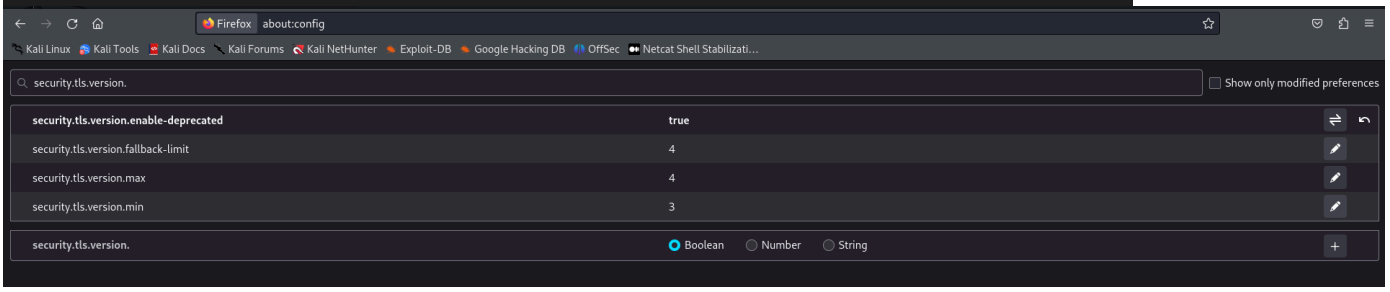
Note: This step reduces security and should only be used as a temporary measure.

7. Bypass Security Temporarily

As a last resort, you can bypass security warnings temporarily. This is not recommended for regular use as it reduces security:

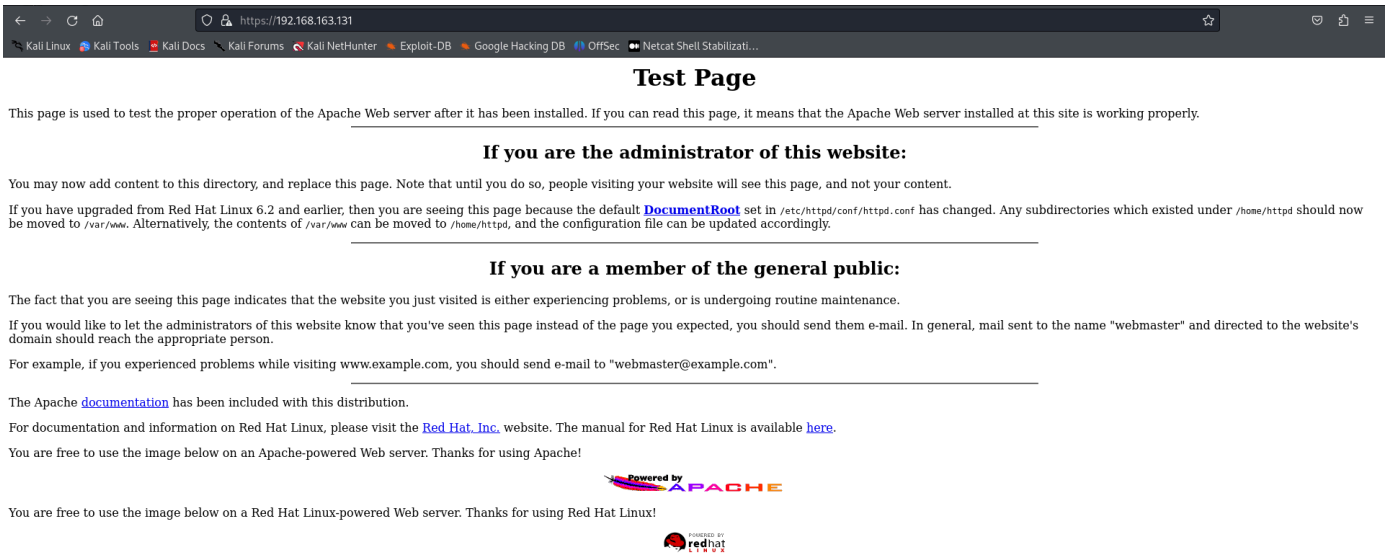
- Open Firefox.
- In the address bar, type `about:config` and press Enter.
- Click "Accept the Risk and Continue."
- Search for `security.ssl.enable_ocsp_stapling` and set it to `false`.
- Search for `security.ssl.require_safe_negotiation` and set it to `false`.

Note: Disabling these security features can make your browser vulnerable to attacks, so use this as a temporary measure only.



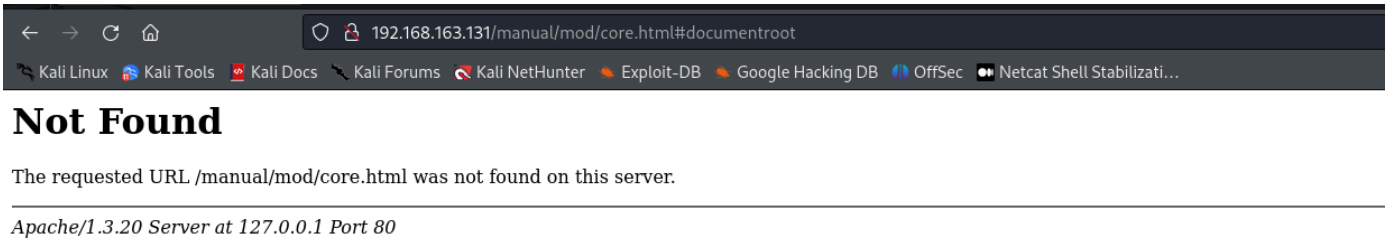
Setting "security.tls.version.enable-deprecated" to "true" seemed to have done the trick. I did set all the previous parameters in number #6, and #7 to false just like it recommended. Now, I do not know if only setting the "security.tls.version.enable-deprecated" to "true" did the trick, or the whole process was necessary, or if only one of those parameters needed to be set to "false" together with "security.tls.version.enable-deprecated". My goal here is not to find the root cause, it is only to make it work. To figure out which one or ones did the trick, we would need to set them as default, and one by one changing it, and seeing how the website respond. It is not very hard, it is just a little extra work, and I am not going to do it.

Now, we can see the website sitting on port 443:



Here, our thought are: we have just seem two default webpages. This tells us 2 things. We now have some knowledge on the architecture of the website, and a little bit about the client potential hygiene. We know they are running Apache2 behind the scenes, and the box is potentially running Red Hat Linux, and this should gives us some idea of what is running behind the scenes. The questions that should pop in our hear should be: 1) are there other directories hosted in this website or are they hosting a website that is not in this particular Ip Address? or 2) did they just put it out there by accident?

If we clink on the links in the page, it brings us a error page where it is disclosing some extra information, and it should actually be redirecting us to a error page that did not disclose information regarding the website, just a general statement.



For example, if they are running Apache2, they are very likely to also be running PHP, or have a database PHPMyAdmin, MySQL....

A good tool to get started on scanning vulnerabilities in the website is :

1. Nikto:

"#nikto -h http://TARGET_IP_ADDRESS" -h stands for host

or

"#nikto -h https://TARGET_IP_ADDRESS"

Nikto might be blocked if they have good security measures by a web application firewall.

Nikto scan did not scan HTTPS for some reason.

Results for Nikto scan on HTTP:

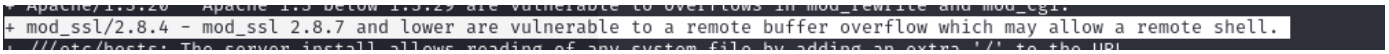
```
┌──(kali㉿kali)-[~/Desktop/PraticalEthicalKacker/Scanning-And-Enumaration]
└─$ nikto -h http://192.168.163.131
- Nikto v2.5.0
---------------------------------------------------------------------------
+ Target IP:          192.168.163.131
+ Target Hostname:    192.168.163.131
+ Target Port:        80
+ Start Time:         2024-06-29 16:46:35 (GMT-4)
---------------------------------------------------------------------------
+ Server: Apache/1.3.20 (Unix)  (Red-Hat/Linux) mod_ssl/2.8.4 OpenSSL/0.9.6b
+ /: Server may leak inodes via ETags, header found with file /, inode: 34821, size: 2890, mtime: Wed Sep  5 23:12:46 2001. See: http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2003-1418
+ /: The anti-clickjacking X-Frame-Options header is not present. See: https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/X-Frame-Options
+ /: The X-Content-Type-Options header is not set. This could allow the user agent to render the content of the site in a different fashion to the MIME type. See: https://www.netsparker.com/web-vulnerability-scanner/vulnerabilities/missing-content-type-header/
+ Apache/1.3.20 appears to be outdated (current is at least Apache/2.4.54). Apache 2.2.34 is the EOL for the 2.x branch.
+ mod_ssl/2.8.4 appears to be outdated (current is at least 2.9.6) (may depend on server version).
+ OpenSSL/0.9.6b appears to be outdated (current is at least 3.0.7). OpenSSL 1.1.1s is current for the 1.x branch and will be supported until Nov 11 2023.
+ /: Apache is vulnerable to XSS via the Expect header. See: http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2006-3918
+ OPTIONS: Allowed HTTP Methods: GET, HEAD, OPTIONS, TRACE .
+ /: HTTP TRACE method is active which suggests the host is vulnerable to XST. See: https://owasp.org/www-community/attacks/Cross_Site_Tracing
+ Apache/1.3.20 - Apache 1.x up 1.2.34 are vulnerable to a remote DoS and possible code execution.
+ Apache/1.3.20 - Apache 1.3 below 1.3.27 are vulnerable to a local buffer overflow which allows attackers to kill any process on the system.
+ Apache/1.3.20 - Apache 1.3 below 1.3.29 are vulnerable to overflows in mod_rewrite and mod_cgi.
+ mod_ssl/2.8.4 - mod_ssl 2.8.7 and lower are vulnerable to a remote buffer overflow which may allow a remote shell.
+ ///etc/hosts: The server install allows reading of any system file by adding an extra '/' to the URL.
+ /usage/: Webalizer may be installed. Versions lower than 2.01-09 vulnerable to Cross Site Scripting (XSS). See: http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2001-0835
+ /manual/: Directory indexing found.
+ /manual/: Web server manual found.
+ /icons/: Directory indexing found.
+ /icons/README: Apache default file found. See: https://www.vntweb.co.uk/apache-restricting-access-to-iconsreadme/
+ /test.php: This might be interesting.
+ /wp-content/themes/twentyeleven/images/headers/server.php?filesrc=/etc/hosts: A PHP backdoor file manager was found.
+ /wordpress/wp-content/themes/twentyeleven/images/headers/server.php?filesrc=/etc/hosts: A PHP backdoor file manager was found.
+ /wp-includes/Requests/Utility/content-post.php?filesrc=/etc/hosts: A PHP backdoor file manager was found.
+ /wordpress/wp-includes/Requests/Utility/content-post.php?filesrc=/etc/hosts: A PHP backdoor file manager was found.
+ /wp-includes/js/tinymce/themes/modern/Meuhy.php?filesrc=/etc/hosts: A PHP backdoor file manager was found.
+ /wordpress/wp-includes/js/tinymce/themes/modern/Meuhy.php?filesrc=/etc/hosts: A PHP backdoor file manager was found.
+ /assets/mobirise/css/meta.php?filesrc=: A PHP backdoor file manager was found.
+ /login.cgi?cli=aa%20aa%27cat%20/etc/hosts: Some D-Link router remote command execution.
+ /shell?cat+/etc/hosts: A backdoor was identified.
+ /#wp-config.php#: #wp-config.php# file found. This file contains the credentials.
+ 8908 requests: 0 error(s) and 30 item(s) reported on remote host
+ End Time:           2024-06-29 16:47:00 (GMT-4) (25 seconds)
---------------------------------------------------------------------------
+ 1 host(s) tested
```

These are all findings. We can look through and see if they are vulnerable to a specific type of attack.

In this particular case, the website is vulnerable to DOS attack, which is typically out of scope when pentesting. There is a possible code execution, buffer overflow whether local or remote, and rewrite. Remote ones are usually the best findings, mostly because we can potentially exploit it from the "outside" on the comfort of our houses.

Nikto also does some directory busting, and we can see it found a couple directories in the website we were not aware off initially.

Here, lets take note of the potential remote vulnerability found:



Next, We are going to use Dirbuster to bust some directories. There is also the option to use Gobuster, or Dirb.

2)#Dirbuster

Dirbuster - pretty straight forward. News here are that we can search for specific file types. In this case, we are up against apache server which runs "php". Now, if it was a microsoft server, then we would be looking for "asp", "aspx" type of files. We should also be looking for :

txt files, zip files, maybe rar files, pdf, docx .....

We separate them with commas.

In our case, we are going to stick with php only for time purpose.

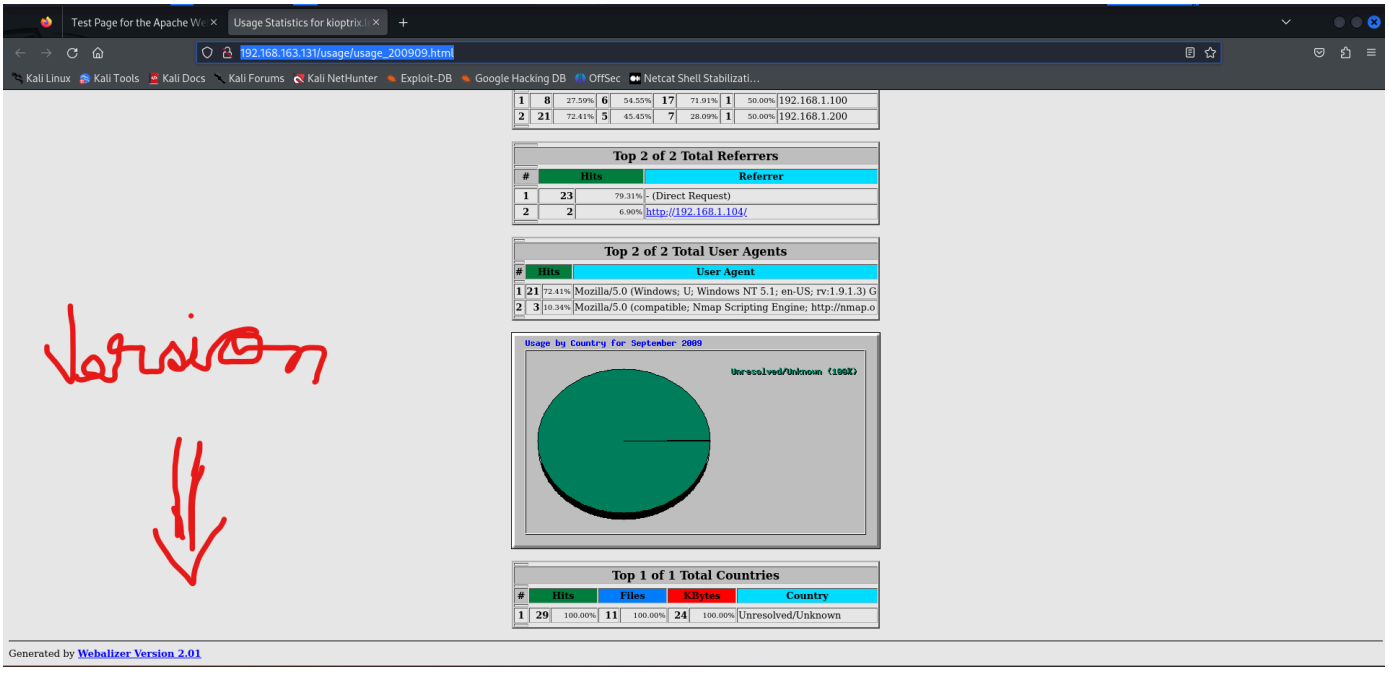
Do not forget, we need to specify the port number on the URL.

And, we are going to use the "/directory-list-2.3-small.txt" wordlist in this case.

Here, we also need to prioritize the findings. We are looking for login pages, html pages found, usage pages,...

In the moment, one of the ones that pops to the eyes is the :

http://192.168.163.131/usage/usage_200909.html

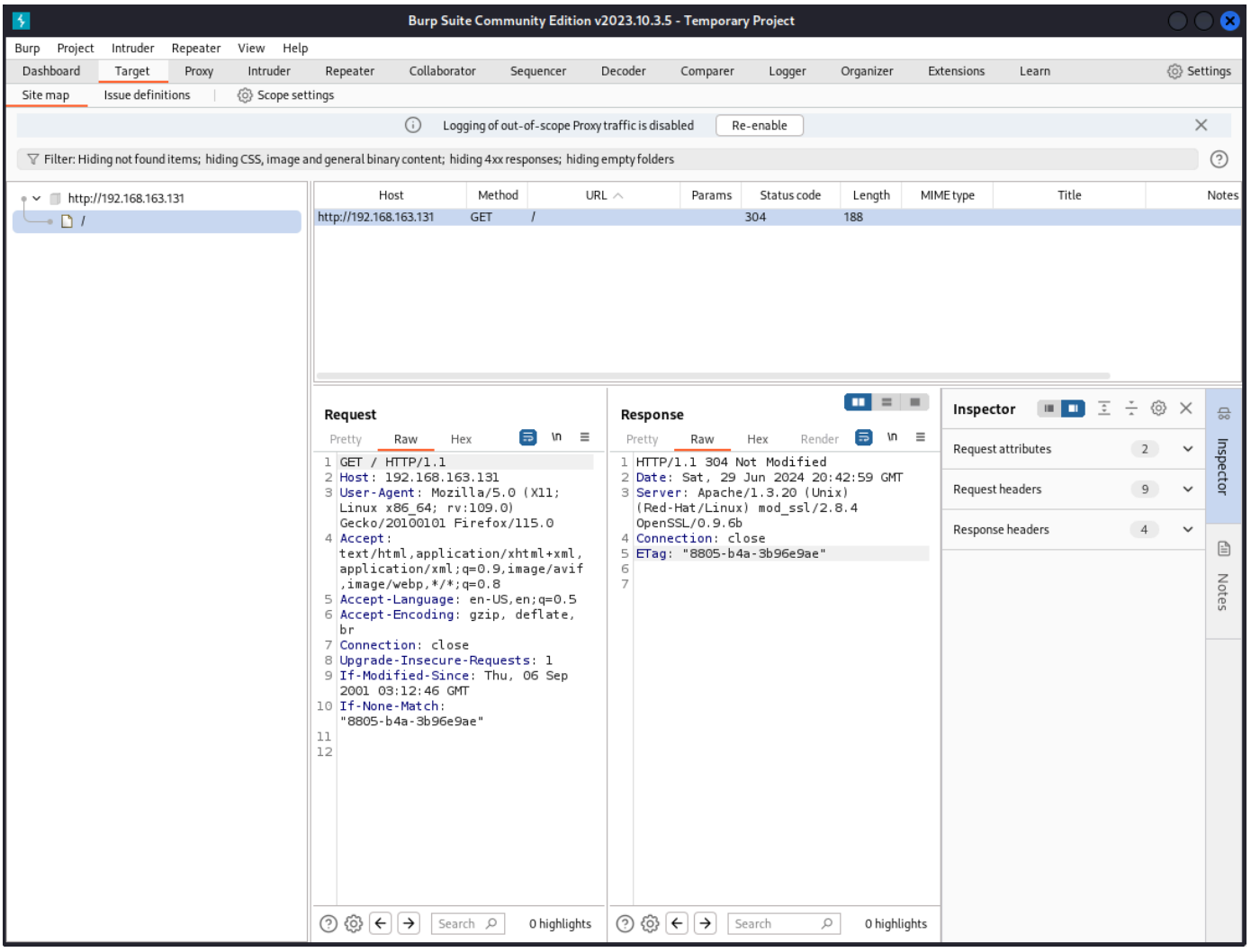


Next up is BurpSuite.

#Burpsuite.

We intercept a request, and then we send it to repeater. Then, we can modify this request, and try it against the server, and see if we get some more information disclosure from the website. We can modify all kind of parameters here.

Another thing we can se is go to the target page, add the target scope, and in this way we are limited to in-scope items. So, we are going to be limited to response from the website. Then, we can see if the server's headers disclose some more info regarding the target.



And, sure enough, it does. This would also be another finding, the server header disclose version information. This was also found in the nikto scan earlier.

This "client" we are working on has some issues with information disclosure.

#Viewing the Source Code:

Another important place to search for information in a website is in the source code. Here in particular, we are looking for comments, information disclosures, keys, passwords, user accounts, or any information in the source code that should not be disclosed.

#Take Away Here:

It is all about the methodology. These are some of the basics that we are looking for: service version info, any sort of back end directories, potential vuln scanning with Nikto, information from Wappalyzer, etc.