Repeating Requests

In the previous sections, we successfully bypassed the input validation to use a non-numeric input to reach command injection on the remote server. If we want to repeat the same process with a different command, we would have to intercept the request again, provide a different payload, forward it again, and finally check our browser to get the final result.

As you can imagine, if we would do this for each command, it would take us forever to enumerate a system, as each command would require 5-6 steps to get executed. However, for such repetitive tasks, we can utilize request repeating to make this process significantly easier.

Request repeating allows us to resend any web request that has previously gone through the web proxy. This allows us to make quick changes to any request before we send it, then get the response within our tools without intercepting and modifying each request.

Proxy History

To start, we can view the HTTP requests history in `Burp` at (`Proxy>HTTP History`):

In `ZAP` HUD, we can find it in the bottom History pane or ZAP's main UI at the bottom `History` tab as well:

Both tools also provide filtering and sorting options for requests history, which may be helpful if we deal with a huge number of requests and want to locate a specific request. `Try to see how filters work on both tools.`

> Note: Both tools also maintain WebSockets history, which shows all connections initiated by the web application even after being loaded, like asynchronous updates and data fetching. WebSockets can be useful when performing advanced web penetration testing, and are out of the scope of this module.

If we click on any request in the history in either tool, its details will be shown:

Burp:

ZAP:

> Tip: While ZAP only shows the final/modified request that was sent, Burp provides the ability to examine both the original request and the modified request. If a request was edited, the pane header would say `Original Request`, and we can click on it and select `Edited Request` to examine the final request that was sent.

Repeating Requests

Burp

Once we locate the request we want to repeat, we can click on `[CTRL+R]` in Burp to send it to the `Repeater` tab, and then we can either navigate to the `Repeater` tab or click `[CTRL+SHIFT+R]` to go to it directly. Once in `Repeater`, we can click on `Send` to send the request:

> Tip: We can also right-click on the request and select `Change Request Method` to change the HTTP method between POST/GET without having to rewrite the entire request.

Cheat Sheet

? Go to Questions

My Workstation

OFFLINE

⦿ Start Instance

∞ / 1 spawns left

ZAP

In ZAP, once we locate our request, we can right-click on it and select `Open/Resend with Request Editor`, which would open the request editor window, and allow us to resend the request with the `Send` button to send our request:
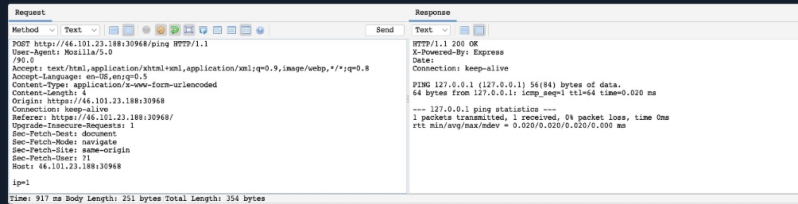
We can also see the `Method` drop-down menu, allowing us to quickly switch the request method to any other HTTP method.

> Tip: By default, the Request Editor window in ZAP has the Request/Response in different tabs. You can click on the display buttons to change how they are organized. To match the above look choose the same display options shown in the screenshot.

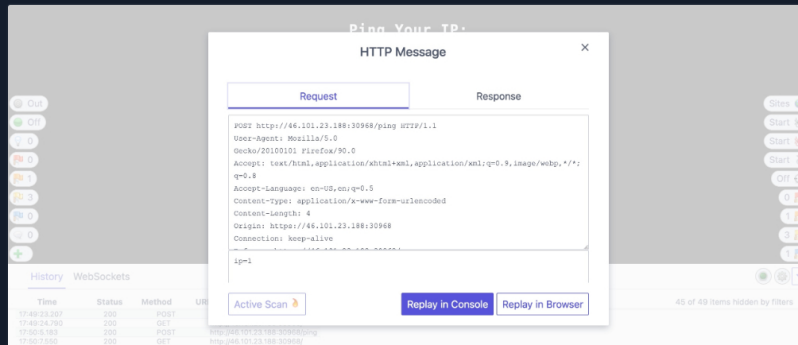We can achieve the same result within the pre-configured browser with `ZAP HUD`. We can locate the request in the bottom History pane, and once we click on it, the `Request Editor` window will show, allowing us to resend it. We can select `Replay in Console` to get the response in the same `HUD` window, or select `Replay in Browser` to see the response rendered in the browser:

So, let us try to modify our request and send it. In all three options (`Burp Repeater`, `ZAP Request Editor`, and `ZAP HUD`), we see that the requests are modifiable, and we can select the text we want to change and replace it with whatever we want, and then click the `Send` button to send it again:
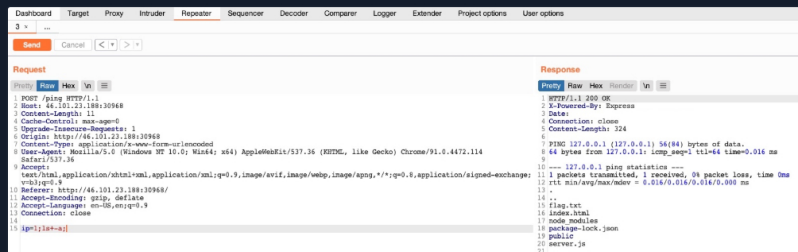
As we can see, we could easily modify the command and instantly get its output by using Burp `Repeater`. Try doing the same in `ZAP Request Editor` and `ZAP HUD` to see how they work.

Finally, we can see in our previous POST request that the data is URL-encoded. This is an essential part of sending custom HTTP requests, which we will discuss in the next section.

Connect to Pwnbox
Your own web-based Parrot Linux instance to play our labs.

Pwnbox Location

UK	137ms ▼

⚠ Terminate Pwnbox to switch location

Start Instance

∞ / 1 spawns left

Enable step-by-step solutions for all questions ⓘ

Questions

Answer the question(s) below to complete this Section and earn cubes!

Cheat Sheet

Target(s): Click here to spawn the target system!

+1 🧊 Try using request repeating to be able to quickly test commands. With that, try looking for the other flag.

HTB{qu1ckly_r3p3471n6_r3qu3575}

🏳 Submit   ▦ Hint

← Previous   Next →   ✅ Mark Complete & Next

Enable step-by-step solutions for all questions ⓘ