Burp Intruder

Both Burp and ZAP provide additional features other than the default web proxy, which are essential for web application penetration testing. Two of the most important extra features are `web fuzzers` and `web scanners`. The built-in web fuzzers are powerful tools that act as web fuzzing, enumeration, and brute-forcing tools. This may also act as an alternative for many of the CLI-based fuzzers we use, like `ffuf`, `dirbuster`, `gobuster`, `wfuzz`, among others.

Burp's web fuzzer is called `Burp Intruder`, and can be used to fuzz pages, directories, sub-domains, parameters, parameters values, and many other things. Though it is much more advanced than most CLI-based web fuzzing tools, the free `Burp Community` version is throttled at a speed of 1 request per second, making it extremely slow compared to CLI-based web fuzzing tools, which can usually read up to 10k requests per second. This is why we would only use the free version of Burp Intruder for short queries. The `Pro` version has unlimited speed, which can rival common web fuzzing tools, in addition to the very useful features of Burp Intruder. This makes it one of the best web fuzzing and brute-forcing tools.

In this section, we will demonstrate the various uses of Burp Intruder for web fuzzing and enumeration.

Target

As usual, we'll start up Burp and its pre-configured browser and then visit the web application from the exercise at the end of this section. Once we do, we can go to the Proxy History, locate our request, then right-click on the request and select `Send to Intruder`, or use the shortcut [`CTRL+I`] to send it to `Intruder`.

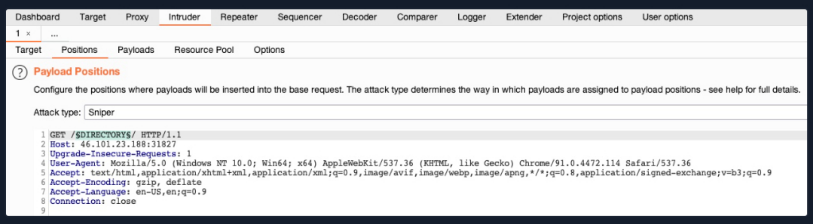
We can then go to `Intruder` by clicking on its tab or with the shortcut [`CTRL+SHIFT+I`], which takes us right to `Burp Intruder`:

On the first tab, '`Target`', we see the details of the target we will be fuzzing, which is fed from the request we sent to `Intruder`.

Positions

The second tab, '`Positions`', is where we place the payload position pointer, which is the point where words from our wordlist will be placed and iterated over. We will be demonstrating how to fuzz web directories, which is similar to what's done by tools like `ffuf` or `gobuster`.

To check whether a web directory exists, our fuzzing should be in '`GET /DIRECTORY/`', such that existing pages would return `200 OK`, otherwise we'd get `404 NOT FOUND`. So, we will need to select `DIRECTORY` as the payload position, by either wrapping it with `§` or by selecting the word `DIRECTORY` and clicking on the `Add §` button:



> Tip: the `DIRECTORY` in this case is the pointer's name, which can be anything, and can be used to refer to each pointer, in case we are using more than position with different wordlists for each.

The final thing to select in the target tab is the `Attack Type`. The attack type defines how many payload pointers are used and determines which payload is assigned to which position. For simplicity, we'll stick to the first type, `Sniper`, which uses only one position. Try clicking on the `?` at the top of the window to read more about attack types, or check out this link.

> Note: Be sure to leave the extra two lines at the end of the request, otherwise we may get an error response from the server.

Payloads

On the third tab, '`Payloads`', we get to choose and customize our payloads/wordlists. This payload/wordlist is what would be iterated over, and each element/line of it would be placed and tested one by one in the Payload Position we chose earlier. There are four main things we need to configure:

- `Payload Sets`
- `Payload Options`
- `Payload Processing`
- `Payload Encoding`

Cheat Sheet
Go to Questions

Table of Contents

Getting Started
Intro to Web Proxies
Setting Up

Web Proxy
Proxy Setup
Intercepting Web Requests
Intercepting Responses
Automatic Modification
Repeating Requests
Encoding/Decoding
Proxying Tools

Web Fuzzer
Burp Intruder
ZAP Fuzzer

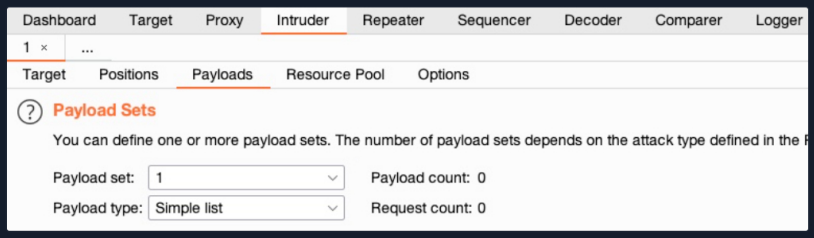
Web Scanner
Burp Scanner
ZAP Scanner
Extensions

Skills Assessment
Skills Assessment - Using Web Proxies

My Workstation
OFFLINE
Start Instance
∞ / 1 spawns left

Payload Sets

The first thing we must configure is the `Payload Set`. The payload set identifies the Payload number, depending on the attack type and number of Payloads we used in the Payload Position Pointers:



In this case, we only have one Payload Set, as we chose the '`Sniper`' Attack type with only one payload position. If we have chosen the '`Cluster Bomb`' attack type, for example, and added several payload positions, we would get more payload sets to choose from and choose different options for each. In our case, we'll select `1` for the payload set.

Next, we need to select the `Payload Type`, which is the type of payloads/wordlists we will be using. Burp provides a variety of Payload Types, each of which acts in a certain way. For example:

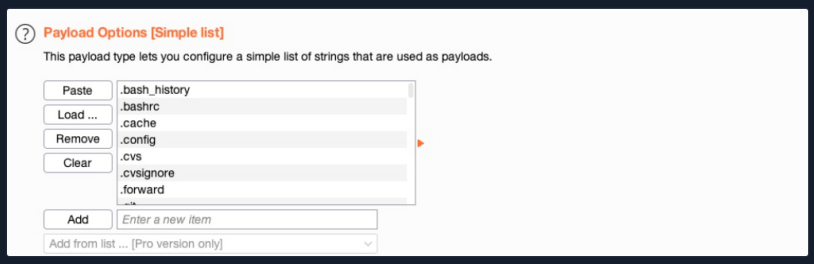
- `Simple List`: The basic and most fundamental type. We provide a wordlist, and Intruder iterates over each line in it.
- `Runtime file`: Similar to `Simple List`, but loads line-by-line as the scan runs to avoid excessive memory usage by Burp.
- `Character Substitution`: Lets us specify a list of characters and their replacements, and Burp Intruder tries all potential permutations.

There are many other Payload Types, each with its own options, and many of which can build custom wordlists for each attack. Try clicking on the `?` next to `Payload Sets`, and then click on `Payload Type`, to learn more about each Payload Type. In our case, we'll be going with a basic `Simple List`.

Payload Options

Next, we must specify the Payload Options, which is different for each Payload Type we select in `Payload Sets`. For a `Simple List`, we have to create or load a wordlist. To do so, we can input each item manually by clicking `Add`, which would build our wordlist on the fly. The other more common option is to click on `Load`, and then select a file to load into Burp Intruder.

We will select `/opt/useful/seclists/Discovery/Web-Content/common.txt` as our wordlist. We can see that Burp Intruder loads all lines of our wordlist into the Payload Options table:



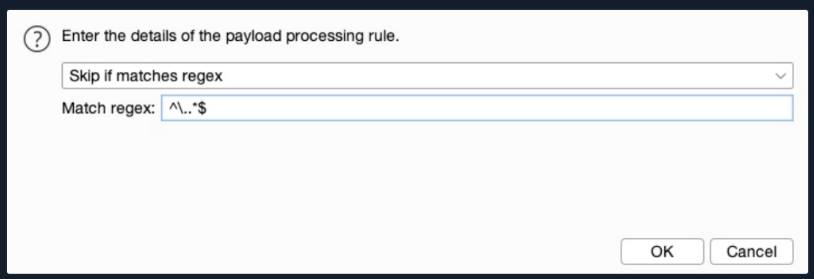
We can add another wordlist or manually add a few items, and they would be appended to the same list of items. We can use this to combine multiple wordlists or create customized wordlists. In Burp Pro, we also can select from a list of existing wordlists contained within Burp by choosing from the `Add from list` menu option.

> Tip: In case you wanted to use a very large wordlist, it's best to use `Runtime file` as the Payload Type instead of `Simple List`, so that Burp Intruder won't have to load the entire wordlist in advance, which may throttle memory usage.

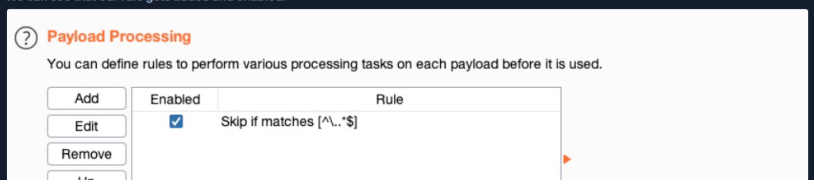
Payload Processing

Another option we can apply is `Payload Processing`, which allows us to determine fuzzing rules over the loaded wordlist. For example, if we wanted to add an extension after our payload item, or if we wanted to filter the wordlist based on specific criteria, we can do so with payload processing.

Let's try adding a rule that skips any lines that start with a `.` (as shown in the wordlist screenshot earlier). We can do that by clicking on the `Add` button and then selecting `Skip if matches regex`, which allows us to provide a regex pattern for items we want to skip. Then, we can provide a regex pattern that matches lines starting with `.`, which is: `^\..*$`:

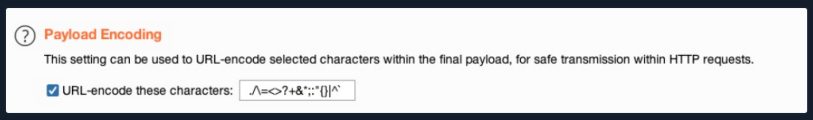


We can see that our rule gets added and enabled:



Payload Encoding

The fourth and final option we can apply is `Payload Encoding`, enabling us to enable or disable Payload URL-encoding.

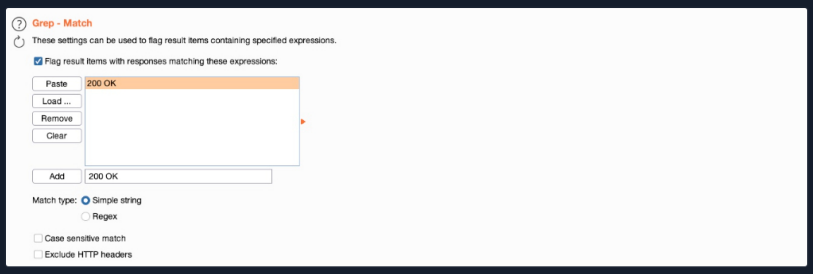


We'll leave it enabled.

Options

Finally, we can customize our attack options from the `Options` tab. There are many options we can customize (or leave at default) for our attack. For example, we can set the number of `retried on failure` and `pause before retry` to 0.

Another useful option is the `Grep - Match`, which enables us to flag specific requests depending on their responses. As we are fuzzing web directories, we are only interested in responses with HTTP code `200 OK`. So, we'll first enable it and then click `Clear` to clear the current list. After that, we can type `200 OK` to match any requests with this string and click `Add` to add the new rule. Finally, we'll also disable `Exclude HTTP Headers`, as what we are looking for is in the HTTP header:



We may also utilize the `Grep - Extract` option, which is useful if the HTTP responses are lengthy, and we're only interested in a certain part of the response. So, this helps us in only showing a specific part of the response. We are only looking for responses with HTTP Code `200 OK`, regardless of their content, so we will not opt for this option.

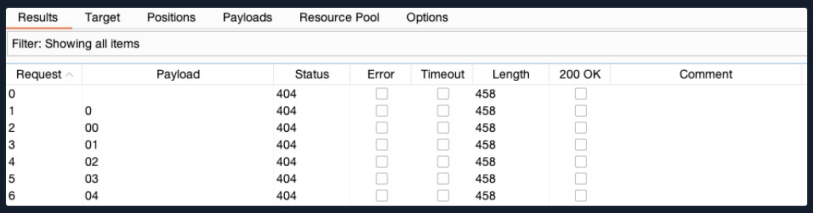
Try other `Intruder` options, and use Burp help by clicking on `?` next to each one to learn more about each option.

> Note: We may also use the `Resource Pool` tab to specify how much network resources Intruder will use, which may be useful for very large attacks. For our example, we'll leave it at its default values.

Attack

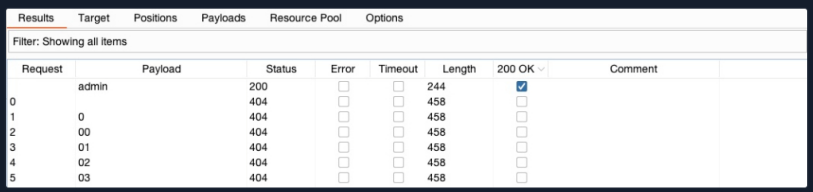
Now that everything is properly set up, we can click on the `Start Attack` button and wait for our attack to finish. Once again, in the free `Community Version`, these attacks would be very slow and take a considerable amount of time for longer wordlists.

The first thing we will notice is that all lines starting with `.` were skipped, and we directly started with the lines after them:



Request	Payload	Status	Error	Timeout	Length	200 OK	Comment
0		404	☐	☐	458	☐	
1	0	404	☐	☐	458	☐	
2	00	404	☐	☐	458	☐	
3	01	404	☐	☐	458	☐	
4	02	404	☐	☐	458	☐	
5	03	404	☐	☐	458	☐	
6	04	404	☐	☐	458	☐	

We can also see the `200 OK` column, which shows requests that match the `200 OK` grep value we specified in the Options tab. We can click on it to sort by it, such that we'll have matching results at the top. Otherwise, we can sort by `status` or by `Length`. Once our scan is done, we see that we get one hit `/admin`:



Request	Payload	Status	Error	Timeout	Length	200 OK	Comment
	admin	200	☐	☐	244	☑	
0		404	☐	☐	458	☐	
1	0	404	☐	☐	458	☐	
2	00	404	☐	☐	458	☐	
3	01	404	☐	☐	458	☐	
4	02	404	☐	☐	458	☐	
5	03	404	☐	☐	458	☐	

We may now manually visit the page `<http://SERVER_IP:PORT/admin/>`, to make sure that it does exist.

Similarly, we can use `Burp Intruder` to do any type of web fuzzing and brute-forcing, including brute forcing for passwords, or fuzzing for certain PHP parameters, and so on. We can even use `Intruder` to perform password spraying against applications that use Active Directory (AD) authentication such as Outlook Web Access (OWA), SSL VPN portals, Remote Desktop Services (RDS), Citrix, custom web applications that use AD authentication, and more. However, as the free version of `Intruder` is extremely throttled, in the next section, we will see ZAP's fuzzer and its various options, which do not have a paid tier.

Terminate Pwnbox to switch location

Start Instance

∞ / 1 spawns left

Waiting to start...

Enable step-by-step solutions for all questions

Questions

Cheat Sheet

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

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

+2 Use Burp Intruder to fuzz for '.html' files under the /admin directory, to find a file containing the flag.

HTB{burp_1n7rud3r_fuzz3r}

Submit Hint

Previous Next

Mark Complete & Next