Burp suite: Intruder

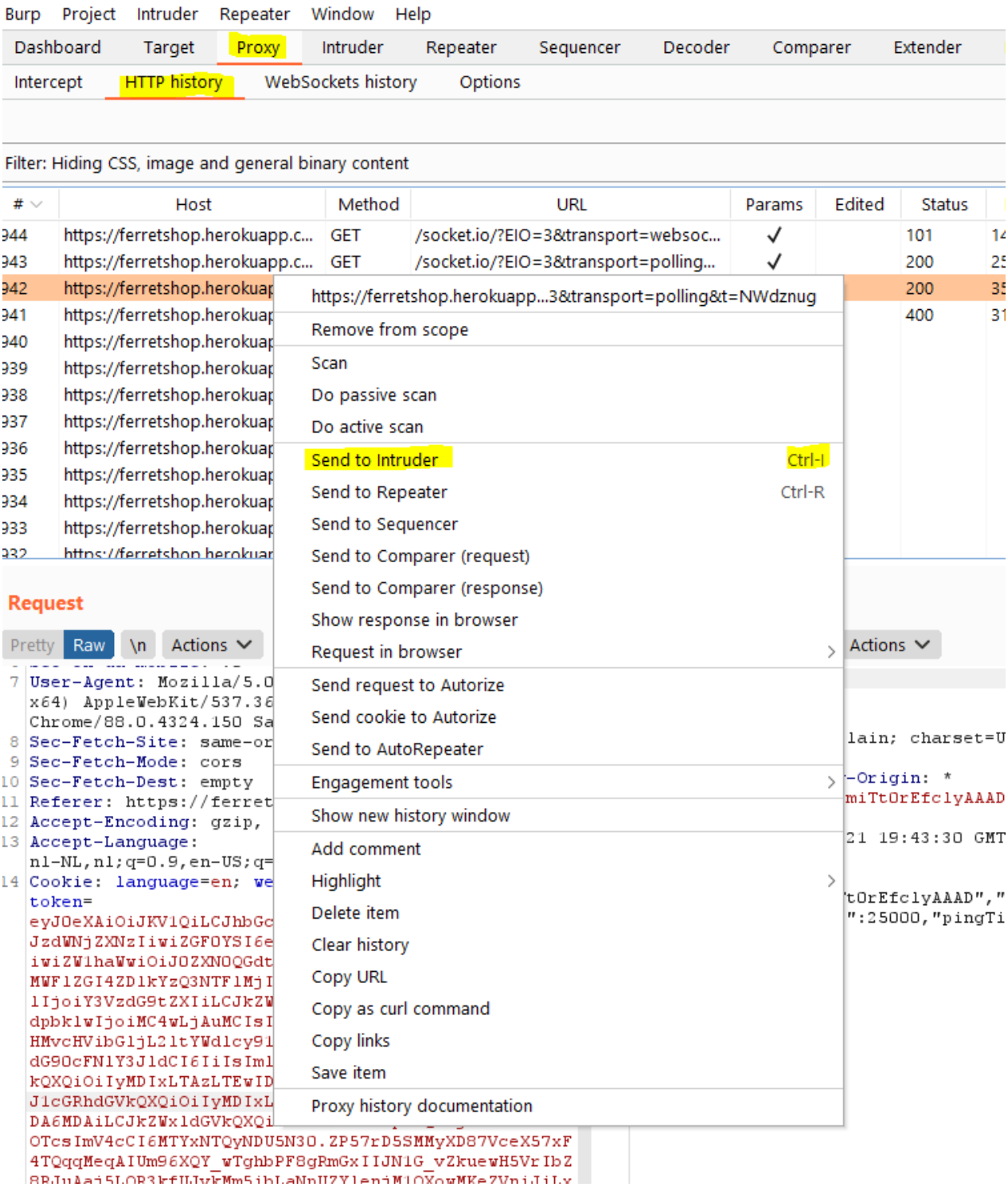
Introduction

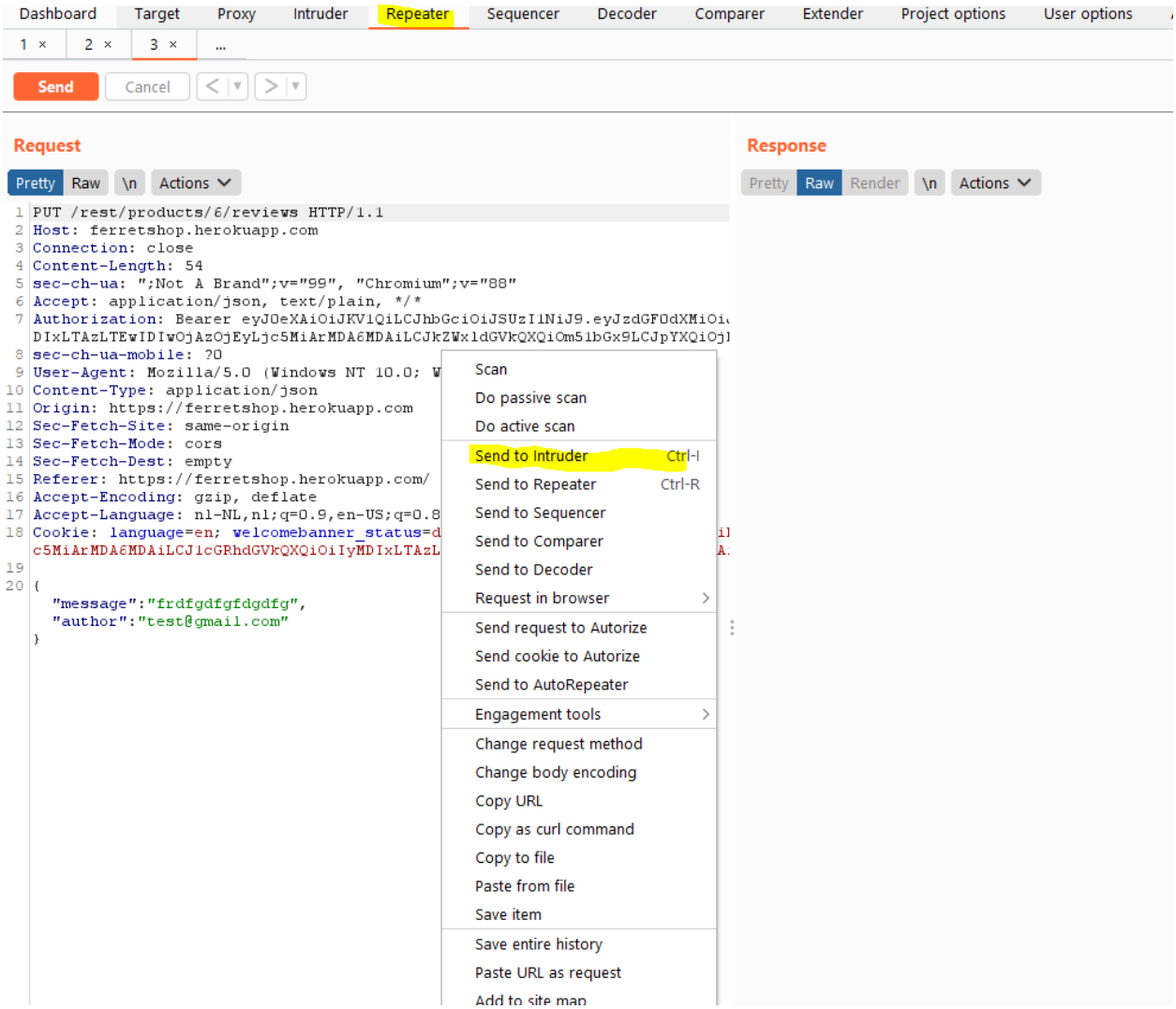
When we are hunting, we sometimes find requests that are interesting to us but they might not be directly usefull due to some system filtering our input like a WAF for example or due to a range of other possibilties. In some of these cases, burp intruder might bring rescue. If we want to try many different payloads quickly, intruder is going to be the perfect tool for us as it is very flexible and it has a range of other options.

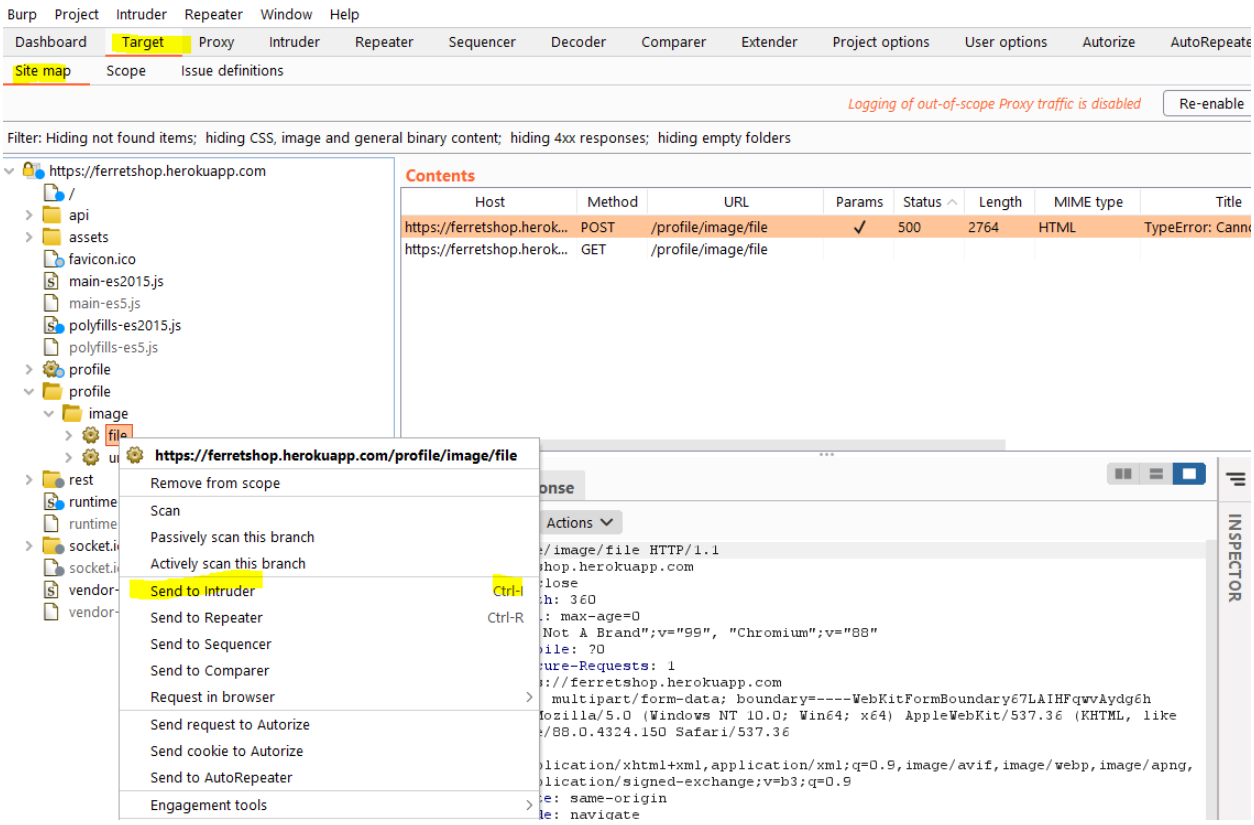
Sending a request to the intruder

We can build the requests we want to send to the intruder manually each time but that would not be very efficient. Instead we have the option to send requests to the intruder from anywhere within burp if we right click them.

Note that we can also use the shortcut ctrl - i

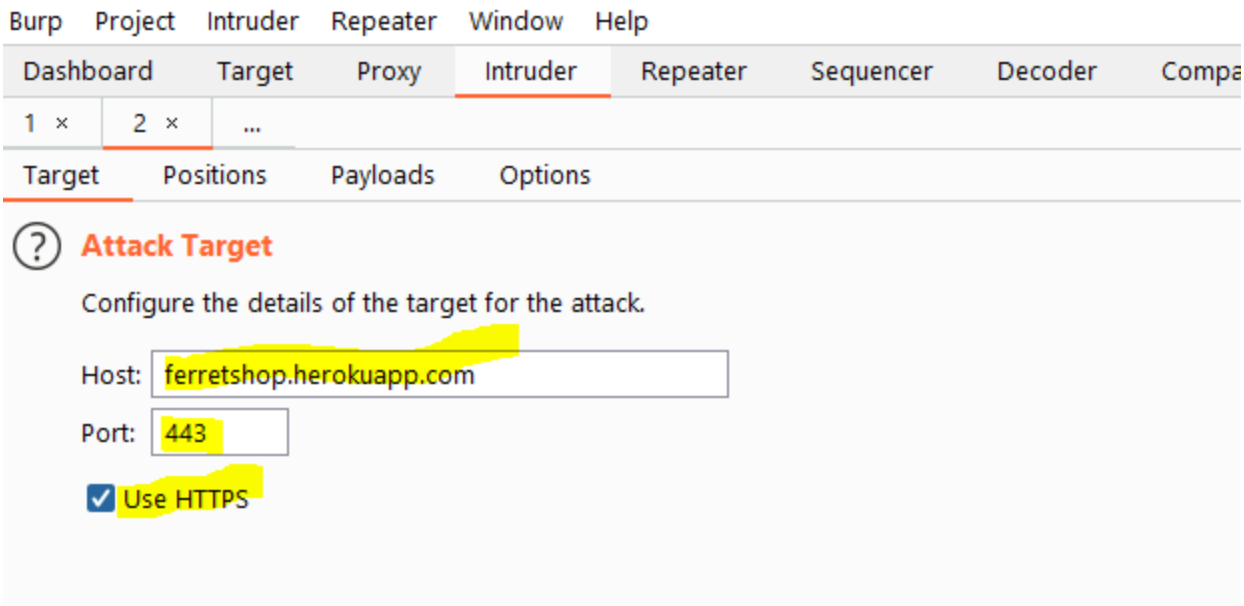



Dashboard
Target
Proxy
Intruder
Repeater
Sequencer
Decoder
Comparer
Extender
Project options
User options
1 ×
2 ×
3 ×
...
Send
Cancel
Request
Pretty
Raw
\n
Actions
PUT /rest/products/6/reviews HTTP/1.1
Host: ferretshop.herokuapp.com
Connection: close
Content-Length: 54
sec-ch-ua: ";Not A Brand";v="99", "Chromium";v="88"
Accept: application/json, text/plain, */*
Authorization: Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJzdGF0dXMiOi
DIxLTAzLTEwIDIwOjAzOjEyLjc5MiArMDA6MDAiLCJkZWx1dGVkQXQiOm51bGx9LCJpYXQiOj
sec-ch-ua-mobile: ?0
User-Agent: Mozilla/5.0 (Windows NT 10.0; W
Content-Type: application/json
Origin: https://ferretshop.herokuapp.com
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: cors
Sec-Fetch-Dest: empty
Referer: https://ferretshop.herokuapp.com/
Accept-Encoding: gzip, deflate
Accept-Language: nl-NL,nl;q=0.9,en-US;q=0.8
Cookie: language=en; welcomebanner_status=d
c5MiArMDA6MDAiLCJlcGRhdGVkQXQiOiIyMDIxLTAzL
{
"message":"frdfgdfgfdgdfg",
"author":"test@gmail.com"
}
Response
Pretty
Raw
Render
\n
Actions
Scan
Do passive scan
Do active scan
Send to Intruder Ctrl-I
Send to Repeater Ctrl-R
Send to Sequencer
Send to Comparer
Send to Decoder
Request in browser
Send request to Autorize
Send cookie to Autorize
Send to AutoRepeater
Engagement tools
Change request method
Change body encoding
Copy URL
Copy as curl command
Copy to file
Paste from file
Save item
Save entire history
Paste URL as request
Add to site map

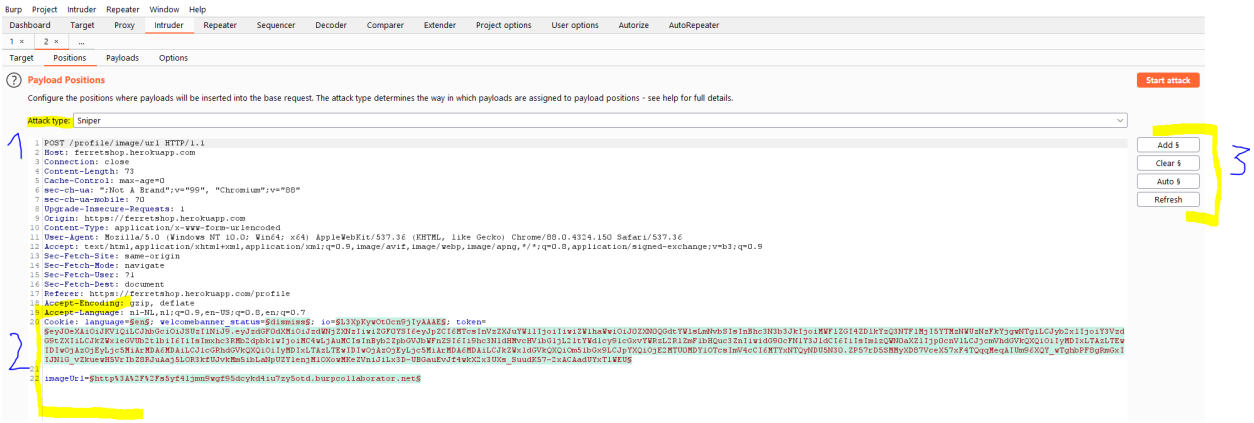


Target tab

This tab is where we define the properties of our target. We can set things like host or port and whether or not to use HTTPS. This can be usefull for example if we want to execute the same attack but on a different target. (For example a production and a staging subdomain)



Positions



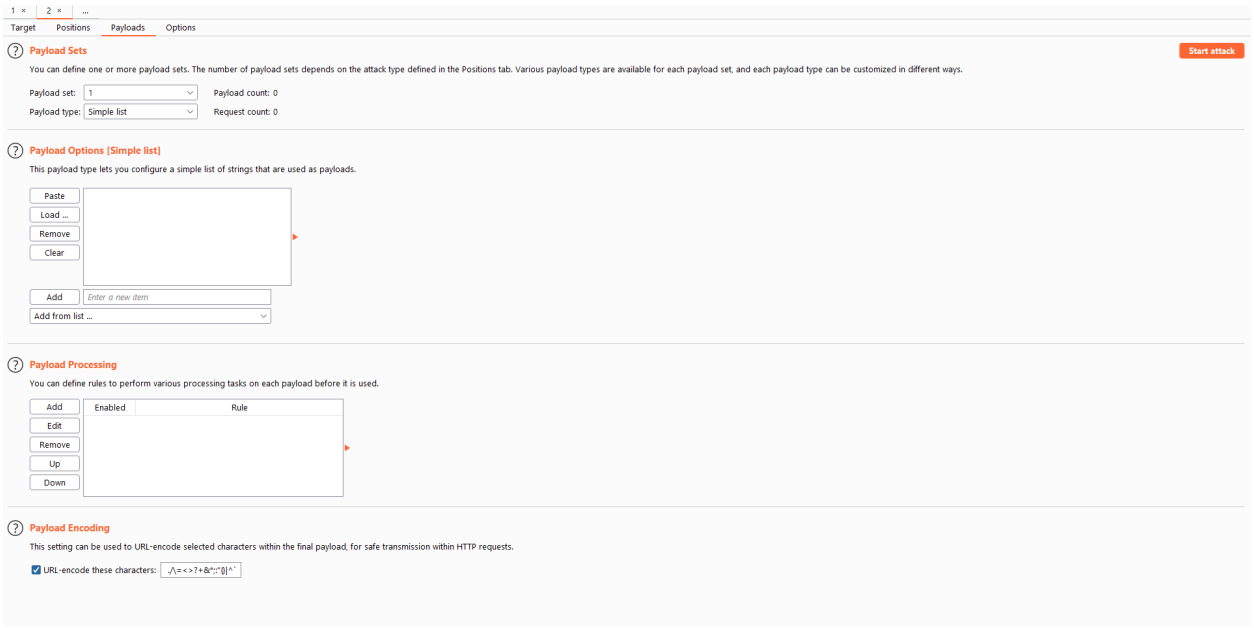
1. The attack type will determine how intruder will handle our payloads. These are the ones i use most
    1. Sniper: One list of payloads which be inserted into every single value. Combinations will be made of all different items on the list. This means that if we have a big list and a lot of parameters we are going to have exponentially more requests and our attack will take a lot longer.

2. Battering ram: It also uses one list but it will insert the same payload into every position. So for example if our list is a,b,c and we have 3 parameters it will first set all parameters to a, then to b and finally to c.
3. Pitchfork: This attack uses as much lists of payloads as there are parameters. This attack type will go through the lists and put every value into the parameters. For example if we have 3 parameters, we will need 3 lists. If we have 10 values in our list, the attack will first pick the first value from all the lists and put it in the respective parameter. Next it will move on to the second item in the lists and so on.
4. Cluster bomb: This also uses as much lists as there are parameters but this method will test every combination of list items possible. This attack takes a long time as you can imagine.

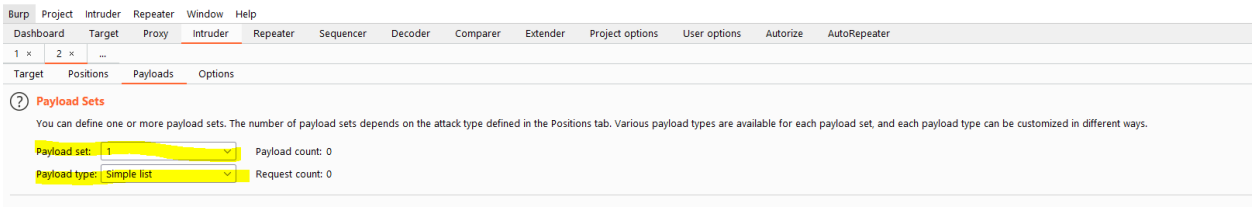
2. This is where we define our parameters. By default the cookies are also selected as parameters. I usually click click in the next step and add my parameters from a blank entry.
3. We can add, remove or clear parameters that we want to test. Auto assigning the template will again assign every parameter burp suite can find.

If our website is really big, we can also search the source code at the bottom of the page.

Payloads



Payload sets



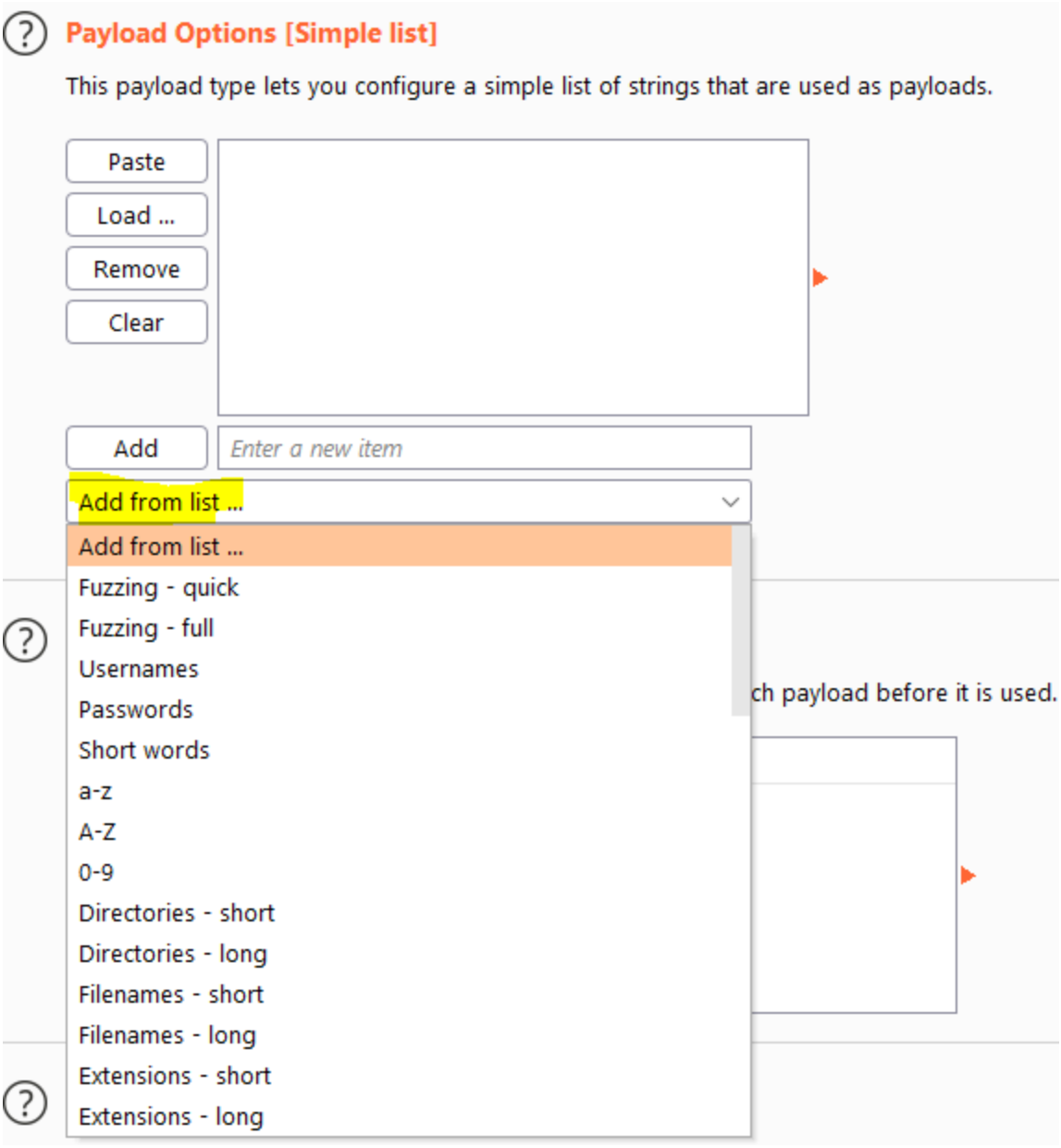
We will have as many payload sets in here as we selected parameters previously if we picked pitchfork or clusterbomb attack forms. Otherwise we will always only have one list available.

The type of list can very. I'm going to go over some popular ones but they speak for themselves.

- Simple list
- Runtime file: This is an external file you can use as a list
- Numbers: For example if you want to test for all numbers between 0000 and 9999 if you are brute forcing
- Brute forcer
- Bit flipper: This is some really cool fuzzing but very hard to understand

Payload options

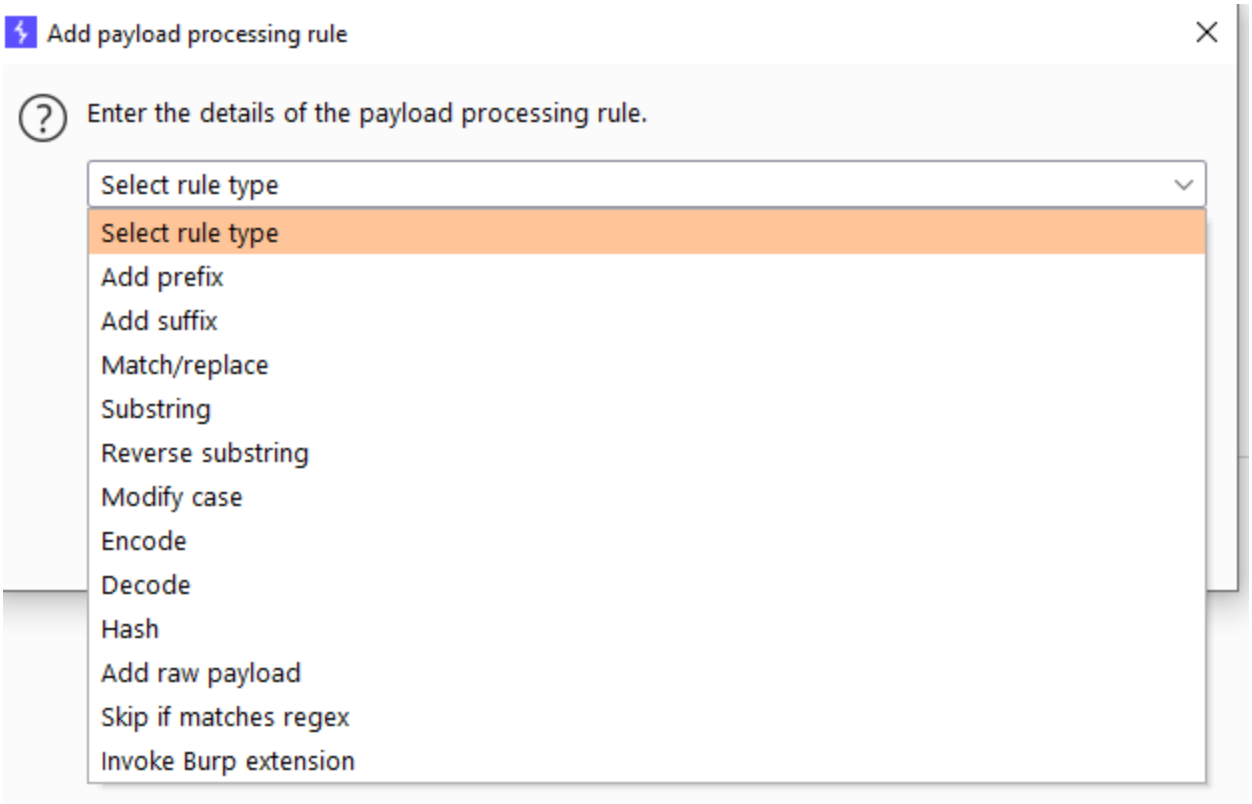
Depending on what payload type you pick, this will differ. One noteable option for the simple list is the option to include fuzzing lists created by burp suite.



Payload processing

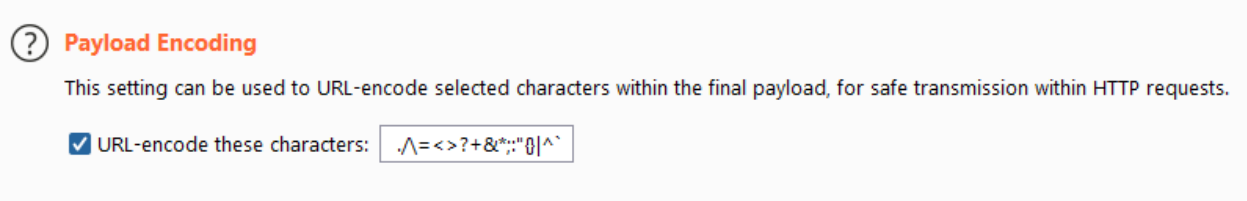
Sometimes we want our payloads to be processed by certain rules before we send them in. For example, some targets require us to base64 encode our payload, this is possible with the payload processings.

We can pick from several different rule types, most speak for themselved.



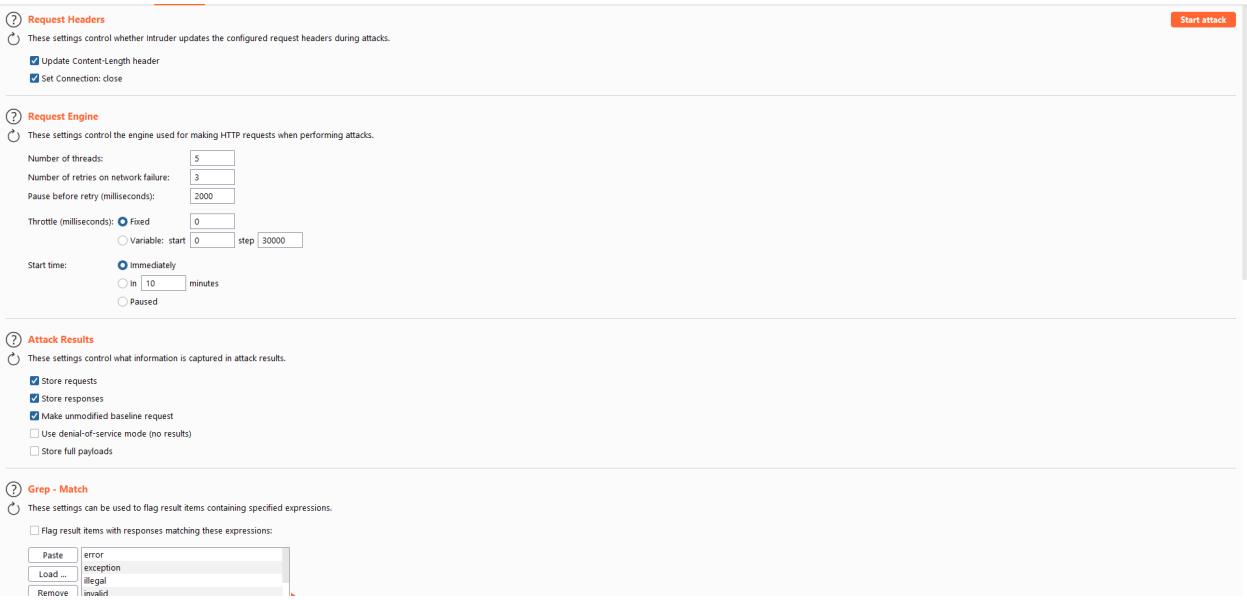
One noteable option we have here is that we can invoke Burp extensions which means we can even write our own code to process the payloads if a target uses custom stuff we can't do with the already existing options.

Payload encodings



By default, burp will url encode a number of characters. This is a good thing as it makes sure we don't trigger any errors on the webserver. In rare cases we might want to disable this option but beware that it might trigger a lot of errors if have a bad list.

Options

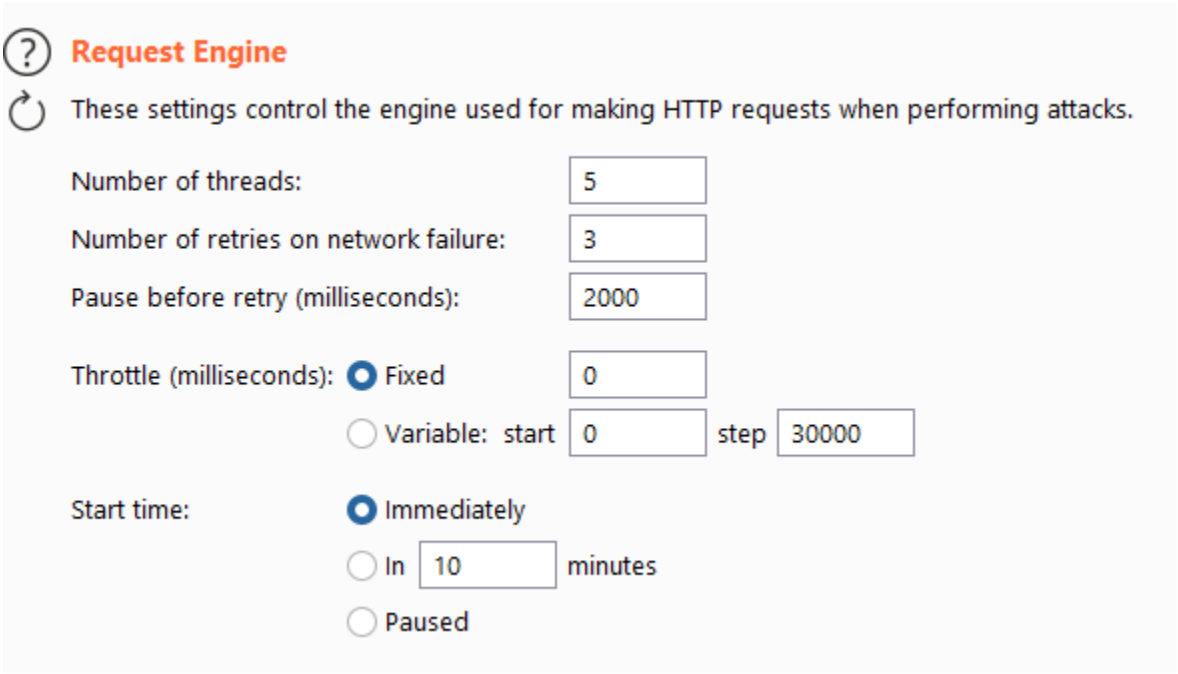


Request headers

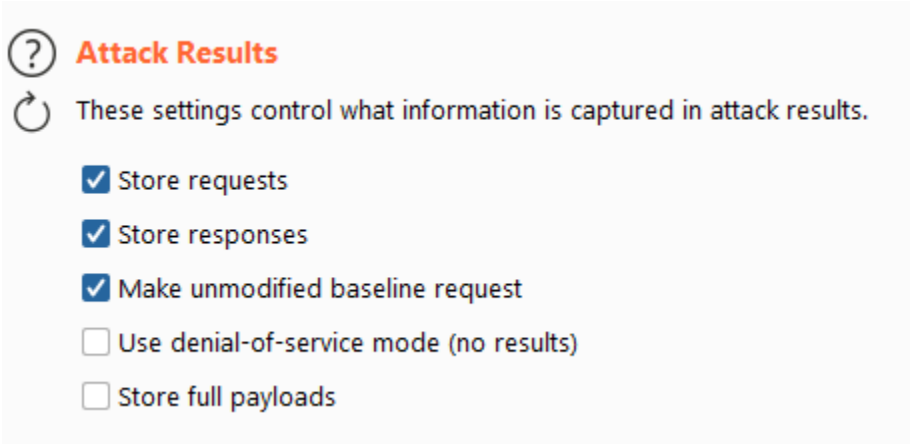
If you change something in the body of a request, something headers should change as well like the content length. Burp suite will automatically do this for us and we can find that option here.

Request engine

This is the most important setting. Some targets will say things like "No automatic scanners allowed". This doesn't mean we can't fuzz perse but we just need to limit our request very much. We can throttle our requests here and tweak several settings such as number of threads or pauses.

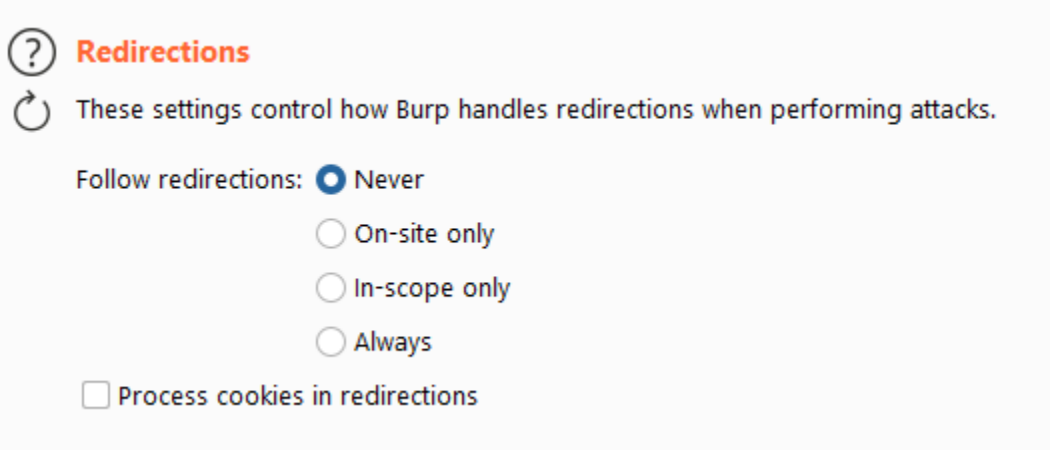


Attack results



This section determines what results are being logged. This can be handy if you are trying to execute a DoS for example. If intruder needs to log stuff, that takes time and if we want to DoS an application, we need to send as much requests as possible. This is something you should rarely if ever do however at full speed as a crashing server in production could spell disaster.

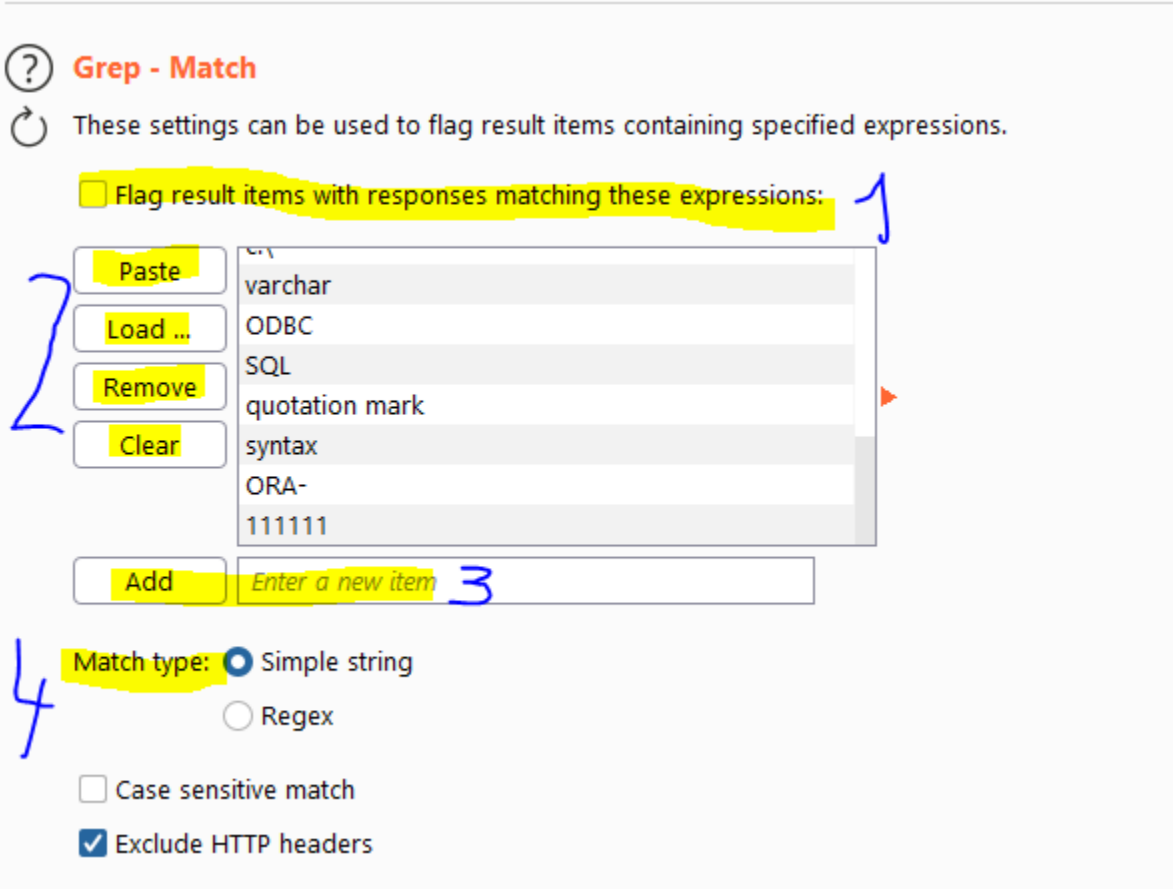
Redirections



Sometimes a website will return a redirection. If that response is the exact same for logged in users as it is for non logged in uses, we might need to follow that redirect to see a difference in responses, this option makes that possible for us.

Grep - Matches

This option allows us to flag specific requests if the response has texts matching the ones we give it in the list. This can be useful for example if you are looking for SQli and expecting a SQL error, you might want to look for a word resembling "SQL" automatically and flag the results.



1. Be aware, you do need to activate this function, it's not active by default
2. You can Paste a list from your clipboard, load it from a file, remove a specific item from the list or clear the whole list
3. Add one item to the list
4. We can either match for full strings or we can even match regular expressions