

# Burp Suite Intruder

**Sniper:** Default and most commonly used. Used for effective single-position attacks such as password - brute force or fuzzing for API endpoints. In a sniper attack we provide a set of payloads which can be a wordlist or range of numbers, and intruder inserts each payload in a defined position in the request.

Let's refer to our example template from before:

```
Example Positions

POST /support/login/ HTTP/1.1
Host: MACHINE_IP
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:80.0) Gecko/20100101 Firefox/80.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded
Content-Length: 37
Origin: http://MACHINE_IP
Connection: close
Referer: http://MACHINE_IP/support/login/
Upgrade-Insecure-Requests: 1

username=$pentester$&password=$Exploited$
```

In this example, we have two positions defined for the `username` and `password` body parameters. In a Sniper attack, Intruder takes each payload from the payload set and substitutes it into each defined position in turn.

Assuming we have a wordlist with three words: `burp`, `suite`, and `intruder`, Intruder would generate six requests:

Request Number	Request Body
1	username=burp&password=Exploited
2	username=suite&password=Exploited
3	username=intruder&password=Exploited
4	username=pentester&password=burp
5	username=pentester&password=suite
6	username=pentester&password=intruder

Observe how Intruder starts with the first position (`username`) and substitutes each payload into it, then moves to the second position (`password`) and performs the same substitution with the payloads. The total number of requests made by Intruder Sniper can be calculated as `requests = numberOfWords * numberOfPositions`.

The Sniper attack type is beneficial when we want to perform tests with single-position attacks, utilizing different payloads for each position. It allows for precise testing and analysis of different payload variations.

## Answer the questions below

If you were using Sniper to fuzz three parameters in a request with a wordlist containing 100 words, how many requests would Burp Suite need to send to complete the attack?

✓ Correct Answer

How many sets of payloads will Sniper accept for conducting an attack?

✓ Correct Answer

**Battering-Ram:** Differs from sniper in that it places the same payload into every position simultaneously rather than substituting each payload into each position in turn.

Let's refer back to our previous example template:

Example Positions

```
POST /support/login/ HTTP/1.1
Host: MACHINE_IP
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:80.0) Gecko/20100101 Firefox/80.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded
Content-Length: 37
Origin: http://MACHINE_IP
Connection: close
Referer: http://MACHINE_IP/support/login/
Upgrade-Insecure-Requests: 1

username=$pentester$&password=$Exp101ted$
```

Using the Battering Ram attack type with the same wordlist from before ( `burp` , `suite` , and `intruder` ), Intruder would generate three requests:

Request Number	Request Body
1	<code>username=burp&amp;password=burp</code>
2	<code>username=suite&amp;password=suite</code>
3	<code>username=intruder&amp;password=intruder</code>

As shown in the table, each payload from the wordlist is inserted into every position for each request made. In a Battering Ram attack, the same payload is thrown at every defined position simultaneously, providing a brute-force-like approach to testing.

The Battering Ram attack type is useful when we want to test the same payload against multiple positions at once without the need for sequential substitution.

In the upcoming tasks, we will explore further configurations and settings related to Intruder's Battering Ram attack type and examine its applications in different scenarios.

Answer the questions below

As a hypothetical question: You need to perform a Battering ram Intruder attack on the example request above.

If you have a wordlist with two words in it ( `admin` and `Guest` ) and the positions in the request template look like this:

```
username=$pentester$&password=$Exp101ted$
```

What would the body parameters of the *first* request that Burp Suite sends be?

username=admin&password=admin

✓ Correct Answer

🔍 Hint

**Pitchfork:** Similar to having multiple sniper attacks running simultaneously. While sniper only uses 1 payload, pitchfork utilises one payload set per position (up to a max of 20) and iterates through them all simultaneously.

Eg two worklists:

1. **First wordlist contains usernames joel, harriet and alex**
2. **Second wordlist contains passwords jo3l, Emma1815 and sk1ll.**

We can use these two lists to perform a Pitchfork attack on the login form. Each request made during the attack would look like this:

Request Number	Request Body
1	<code>username=joel&amp;password=Jo3l</code>
2	<code>username=harriet&amp;password=Emma1815</code>
3	<code>username=alex&amp;password=Sk1ll</code>

As shown in the table, Pitchfork takes the first item from each list and substitutes them into the request, one per position. It then repeats this process for the next request by taking the second item from each list and substituting it into the template. Intruder continues this iteration until one or all of the lists run out of items. It's important to note that Intruder stops testing as soon as one of the lists is complete. Therefore, in Pitchfork attacks, it is ideal for the payload sets to have the same length. If the lengths of the payload sets differ, Intruder will only make requests until the shorter list is exhausted, and the remaining items in the longer list will not be tested.

The Pitchfork attack type is especially useful when conducting credential-stuffing attacks or when multiple positions require separate payload sets. It allows for simultaneous testing of multiple positions with different payloads.

In the upcoming tasks, we will explore further configurations and settings related to Intruder's Pitchfork attack type and explore its applications in different scenarios, including credential-stuffing attacks.

Answer the questions below

What is the maximum number of payload sets we can load into Intruder in Pitchfork mode?

20

✓ Correct Answer

**Cluster Bomb:** Allows us to choose multiple payload sets, one per position. Up to a max of 20 of course. Unlike pitchfork where all the payload sets are tested simultaneously, cluster bomb iterates through each payload set individually, ensuring that every possible combination of payload is tested.

To illustrate the Cluster bomb attack type, let's use the same wordlists as before:

- Usernames: `joel`, `harriet`, and `alex`.
- Passwords: `J031`, `Emma1815`, and `Sk111`.

In this example, let's assume that we don't know which password belongs to which user. We have three users and three passwords, but the mappings are unknown. In this case, we can use a Cluster bomb attack to try every combination of values. The request table for our username and password positions would look like this:

Request Number	Request Body
1	<code>username=joel&amp;password=J031</code>
2	<code>username=harriet&amp;password=J031</code>
3	<code>username=alex&amp;password=J031</code>
4	<code>username=joel&amp;password=Emma1815</code>
5	<code>username=harriet&amp;password=Emma1815</code>
6	<code>username=alex&amp;password=Emma1815</code>
7	<code>username=joel&amp;password=Sk111</code>
8	<code>username=harriet&amp;password=Sk111</code>
9	<code>username=alex&amp;password=Sk111</code>

As shown in the table, the Cluster bomb attack type iterates through every combination of the provided payload sets. It tests every possibility by substituting each value from each payload set into the corresponding position in the request.

Cluster bomb attacks can generate a significant amount of traffic as it tests every combination. The number of requests made by a Cluster bomb attack can be calculated by multiplying the number of lines in each payload set together. It's important to be cautious when using this attack type, especially when dealing with large payload sets. Additionally, when using Burp Community and its Intruder rate-limiting, the execution of a Cluster bomb attack with a moderately sized payload set can take a significantly longer time.

The Cluster bomb attack type is particularly useful for credential brute-forcing scenarios where the mapping between usernames and passwords is unknown.

In the upcoming tasks, we will explore further configurations and settings related to Intruder's Cluster bomb attack type and examine its applications in different scenarios.

100 (Set 1) × 2 (Set 2) × 30 (Set 3) = 6,000 requests

✓ Answer: 6000

Answer the questions below

We have three payload sets. The first set contains 100 lines, the second contains 2 lines, and the third contains 30 lines.

How many requests will Intruder make using these payload sets in a Cluster bomb attack?

6000

✓ Correct Answer

🔍 Hint

### Practical:

We have the IP address of: <http://10.10.250.250/support/login/> and upon viewing the source code, no protective measures have been implemented.

```
Support Login Form Source Code

---
<form method="POST">
  <div class="form-floating mb-3">
    <input class="form-control" type="text" name=username placeholder="Username" required>
    <label for="username">Username</label>
  </div>
  <div class="form-floating mb-3">
    <input class="form-control" type="password" name=password placeholder="Password" required>
    <label for="password">Password</label>
  </div>
  <div class="d-grid"><button class="btn btn-primary btn-lg" type="submit">Login!</button></div>
</form>
---
```

Given the absence of protective measures we have multiple options to exploit in this form, including cluster bomb attack for brute forcing credentials, but we do have an easier approach.

Approximately 3 months ago Bastio Hosting fell victim to a cyber attack compromising 3 employee usernames, email addresses and plain text passwords. While the affected employees were instructed to change their passwords promptly there is a possibility that someone disregarded this advice.

As we possess a list of known usernames, each accompanied by a corresponding password, we can leverage a credential stuffing attack instead of a straight forward brute force attack. This method provides an advantageous and significantly quicker attack, especially when utilising the rate limited edition we have on the community burp suite.

In order to access the leaked credentials we must visit: `wget`

<http://10.10.250.250:9999/Credentials/BastionHostingCreds.zip>

```
root@ip-10-10-62-144:~# wget http://10.10.250.250:9999/Credentials/BastionHostin
gCreds.zip
--2025-07-23 16:13:06-- http://10.10.250.250:9999/Credentials/BastionHostingCre
ds.zip
Connecting to 10.10.250.250:9999... connected.
HTTP request sent, awaiting response... 200 OK
Length: 3505 (3.4K) [application/zip]
Saving to: 'BastionHostingCreds.zip'

BastionHostingCreds 100%[=====>] 3.42K --.-KB/s in 0s
2025-07-23 16:13:06 (114 MB/s) - 'BastionHostingCreds.zip' saved [3505/3505]

root@ip-10-10-62-144:~#
```

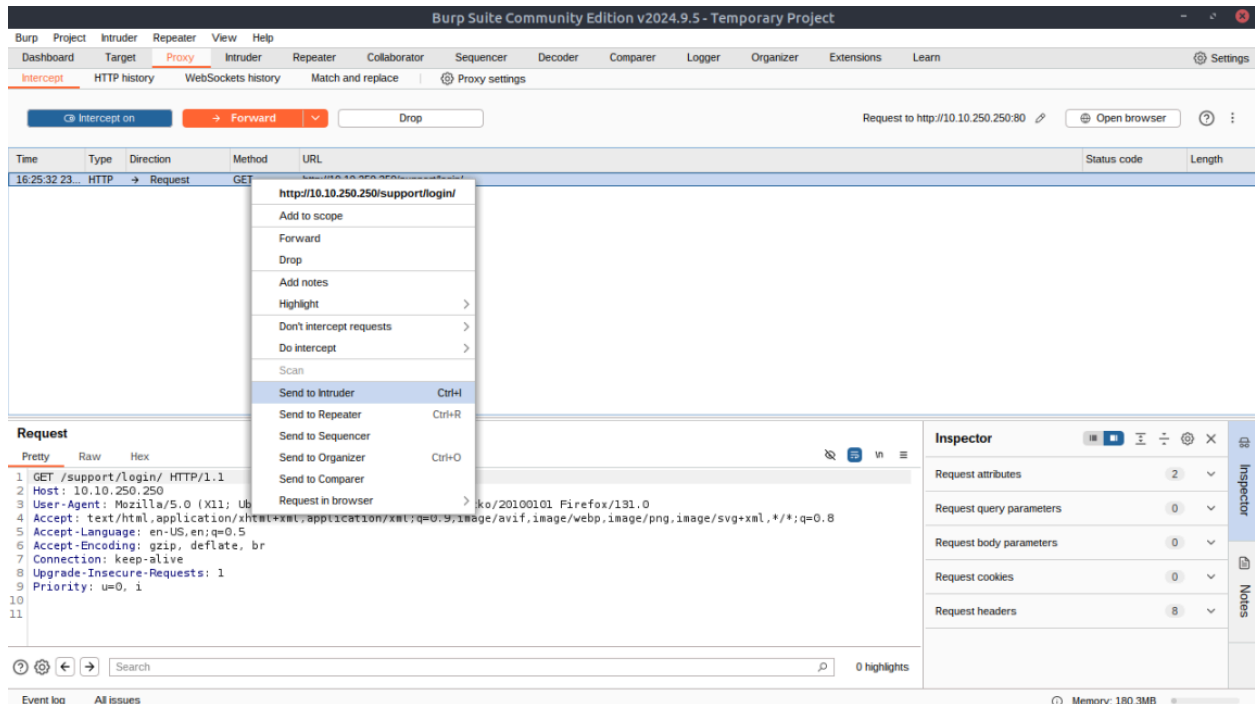
Here we can see it was successfully installed.

```
root@ip-10-10-62-144:~# unzip BastionHostingCreds.zip

Archive: BastionHostingCreds.zip
  inflating: combined.txt
  inflating: emails.txt
  inflating: passwords.txt
  inflating: usernames.txt
```

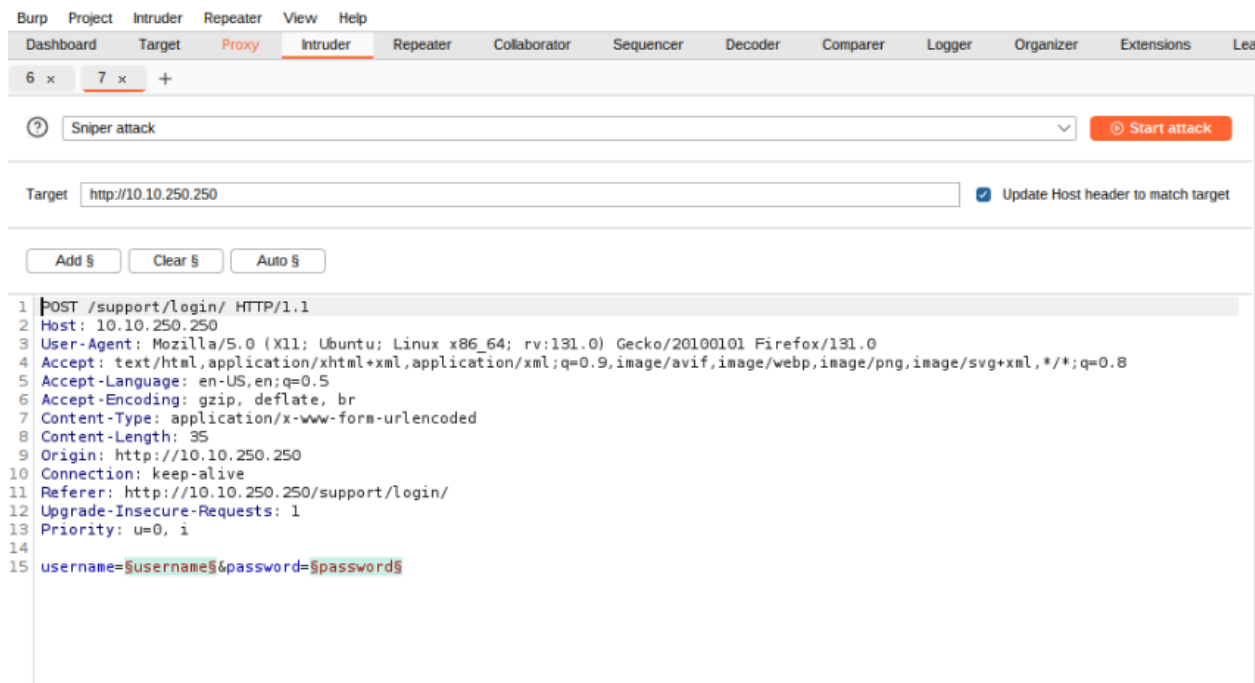
I created sub folders individually, and went through extracting the correct information so I could use it in the intruder attacks.

I then made sure FoxyProxy was enabled and Burp suite were running to capture the proxy request.



I then sent the GET request to the intruder. I then configure the Intruder tab as appropriate and select the “Pitchfork” attack.

But, we need a POST request instead, so we send a random set of info into the username and password fields and enter.



Selecting “Auto” will automatically highlight the username and password field as shown.

Payloads

Payload position: 1

Payload type: Simple list

Payload count: 100

Request count: 0

Payload configuration

This payload type lets you configure a simple list of strings that are used as payloads.

Paste

Load...

Remove

Clear

Deduplicate

l.madden

j.poole

i.hopkins

l.mayo

m.roberts

m.ratliff

j.morrison

r.carroll

t.nichols

d.decker

l.nana

Add

Enter a new item

Add from list... [Pro version only]

Payload processing

You can define rules to perform various processing tasks on each payload before it is used.

Add

Edit

Remove

Use

☐

Enabled

Rule

I then select payload position 1 “username.txt” which loads 100 payload.



**Payloads**

Payload position: 2

Payload type: Simple list

Payload count: 100

Request count: 100

**Payload configuration**

This payload type lets you configure a simple list of strings that are used as payloads.

Paste: bambam  
Load...: UnitedKingdom123  
Remove: 1q2q3q  
Clear: valencia  
Deduplicate: 258852  
Add: 10031991  
Add from list... [Pro version only]: wolverine  
12111991  
asdfghj  
02011976  
#####

**Payload processing**

You can define rules to perform various processing tasks on each payload before it is used.

Add, Edit, Remove buttons are present. A table with columns 'Enabled' and 'Rule' is shown.

I then select payload position 2 “password.txt” which loads 100 payloads.

Once these are both configured I am ready to launch the attack!

45	lpacheco	iceman	302	4	679
46	o.barton	westham	302	6	679
47	y.williams	lane	302	9	679

Request	Response
POST /support/login/ HTTP/1.1 Host: 10.10.250.250 User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:131.0) Gecko/20100101 Firefox/131.0 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/png,image/svg+xml,*/*;q=0.8 Accept-Language: en-US,en;q=0.5 Accept-Encoding: gzip, deflate, br Content-Type: application/x-www-form-urlencoded Content-Length: 36 Origin: http://10.10.250.250 Connection: keep-alive Referer: http://10.10.250.250/support/login/ Upgrade-Insecure-Requests: 1 Priority: u=0, i username=1s2epacheco&password=iceman	

So we are looking for the shortest response time that indicates a successful login attempt. So we can now try and see if it works, logging in using their credentials. Did not work...

50	m.rivera	letmein1	302	340	598
51	d.banks	1964	302	5	679
52	l.watson	0xc0c0	302	5	679
53	l.holmes	NS446voVxzczZ8xC7	302	5	679
54	a.holland	wrench	302	4	679
55	r.macias	BastionHostingCTFOForTheWin	302	4	679
56	d.vincent	25011985	302	5	679
57	s.ford	sanders	302	5	679
58	j.perry	Sailing-3	302	5	679

Support — Mozilla Firefox

10.10.250.250/support/

TryHackMe | Learn Cy... TryHackMe Support Offline CyberChef Revshell Generator Reverse Shell Cheat S... GitHub - swisskyrepo/...

It looks like you haven't started Firefox in a while. Do you want to clean it up for a fresh, like-new experience? And by the way, welcome back! Refresh Firefox...

Bastion Hosting

Home About Contact Support Products Logout

## Support Home

Your Assigned Tickets:

Ticket ID	Preview	Completed?
6	0day wuz here...	✗
57	We keep getting hack...	✗
78	Hey, I tried to sign...	✓

Or maybe not... The longest response time seemed to work instead?

Answer the questions below

What username and password combination indicates a successful login attempt? The answer format is "username:password".

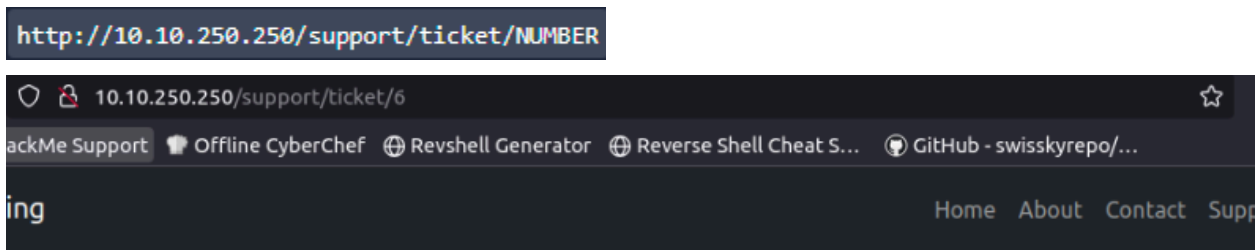
✓ Correct Answer

Trial and error! We did it!!

### THM Challenge:

So having gained access to the support system we now have the opportunity to explore its functionalities and see what actions we can perform.

Upon accessing the home interface I am presented with a table displaying various tickets. Clicking on any row redirects us to a page where we can view the incomplete ticket. By examining the URL structure we observe that the pages are numbered in the following format:



After some wild guessing! Ticket 6 is available. But there could be more and to reduce the time spent doing them all individually let's set up a sniper attack.

`GET /support/ticket/6 HTTP/1.1` this is what is shown in the intruder window after sending the GET request.

`GET /support/ticket/$6$ HTTP/1.1`

I change it to select the number within the section symbol: \$

Payloads

Payload position: All payload positions

Payload type: Numbers

Payload count: 1

Request count: 1

Payload configuration

This payload type generates numeric payloads within a given range and in a specified format.

Number range

Type: ☒ Sequential ☐ Random

From: 0

To:

Step: 1

How many:

Number format

Base: ☒ Decimal ☐ Hex

Min integer digits: 0

Max integer digits: 0

Min fraction digits: 0

Max fraction digits: 0

Examples

0

n

I will select a numbers file form the number 1 - 200.

Payloads

Request count: 200

Payload configuration

This payload type generates numeric payloads within a given range and in a specified format.

Number range

Type:
☒ Sequential
☐ Random

From: 1

To: 200

Step: 1

How many:

Number format

Base:
☒ Decimal
☐ Hex

Min integer digits: 0

Max integer digits: 3

Min fraction digits: 0

Max fraction digits: 0

Examples

1

321

Payload processing

You can define rules to perform various processing tasks on each payload before it is

Going 1 step at a time I assume just means going 1 at a time.. So let's see what happens.

So already seeing the progress we have some with the status code of 200 which is good! And 404 means it doesn't exist.. Not so good. But that's okay!

0		200	4
1	1	404	7
2	2	404	4
3	3	404	5
4	4	404	4
5	5	404	5
6	6	200	4
7	7	404	4
8	8	404	4
9	9	404	5
10	10	404	4

See already we picked up on an error of doing it manually I didn't think of putting 0, which means we missed it, and 6 we know was correct because that was a baseline for us, the first one we discovered doing it manually.

So let's make a table of our findings:

0 = OK

6 = OK

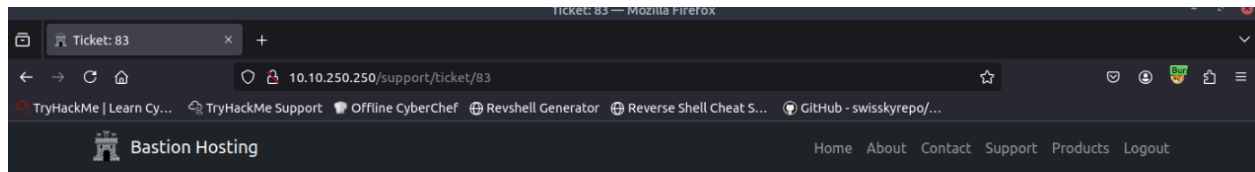
47 = OK

57 = OK

78 = OK

83 = OK

By the looks of it, those are all the results we got back, 6 positive results. Now all we need to do, turn intercept off and manually type those values in to find the flag. This is a good time to mention, doing this manually we would have probably been done by now.. But let's say that there was 100s or 1000s of different IDs, doing it manually would take time, and like me, I could overlook some values very easily, so this negates human error but also identified it in a list you can directly refer from.



## Ticket 83

Email	flag@bastionhosting.thm
Query	THM{MTMxNTg5NTUzMWM0OWRIYzUzMDVjMzJl}

Ticket Resolved: ☒

There we go, in ticket 83 is my sweet, sweet flag.

THM{MTMxNTg5NTUzMWM0OWRIYzUzMDVjMzJl}

## Another Challenge:

In this extra-mile exercise, we will tackle a more challenging variant of the credential-stuffing attack we previously performed. However, this time, additional measures have been implemented to make brute-forcing more difficult. If you are comfortable using Burp Macros, you can attempt this challenge without the instructions below. Otherwise, let's proceed with the step-by-step approach.

So we need to navigate to <http://10.10.250.250/admin/login/>

# Administraction Login

Username  
m.rivera

Password  
\*\*\*\*\*

Login!

Invalid Username or Password.



Worth a try seeing as it was just sitting there! (This was from our previous attempts). First I am going to turn intercept on and get a grab of this page, and because it is now a post request

17:14:00 23 J... HTTP → Request POST http://10.10.250.250/admin/login/

Because I tried to login already to no success but this allows me to manipulate it. So right click and send to the intruder!

?

Pitchfork attack

Start attack

Target

http://10.10.250.250

☒ Update Host header to match target

Add §

Clear §

Auto §

1 POST /admin/login/ HTTP/1.1

2 Host: 10.10.250.250

3 User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86\_64; rv:131.0) Gecko/20100101 Firefox/131.0

4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/png,image/svg+xml,\*/\*;q=0.8

5 Accept-Language: en-US,en;q=0.5

6 Accept-Encoding: gzip, deflate, br

7 Content-Type: application/x-www-form-urlencoded

8 Content-Length: 79

9 Origin: http://10.10.250.250

10 Connection: keep-alive

11 Referer: http://10.10.250.250/admin/login/

12 Cookie: session=eyJTdXBwb3J0QXV0aCI6IiRydWUiLCJTdXBwb3J0SUQiOjMsInRva2VuSUQiOiI4NjVmZTk5QWQxMGZmYTg1ZWQ2MTQ0MmQxN2ZkZTk2NyJ9. aIEKPQ.Fn9j 0GcTvgu9 0StQnyTk0yQETDU

13 Upgrade-Insecure-Requests: 1

14 Priority: u=0, i

15

16 username=\$m.rivera&password=\$letmein&loginToken=67c6f4682ff59562405dd12e188909ed

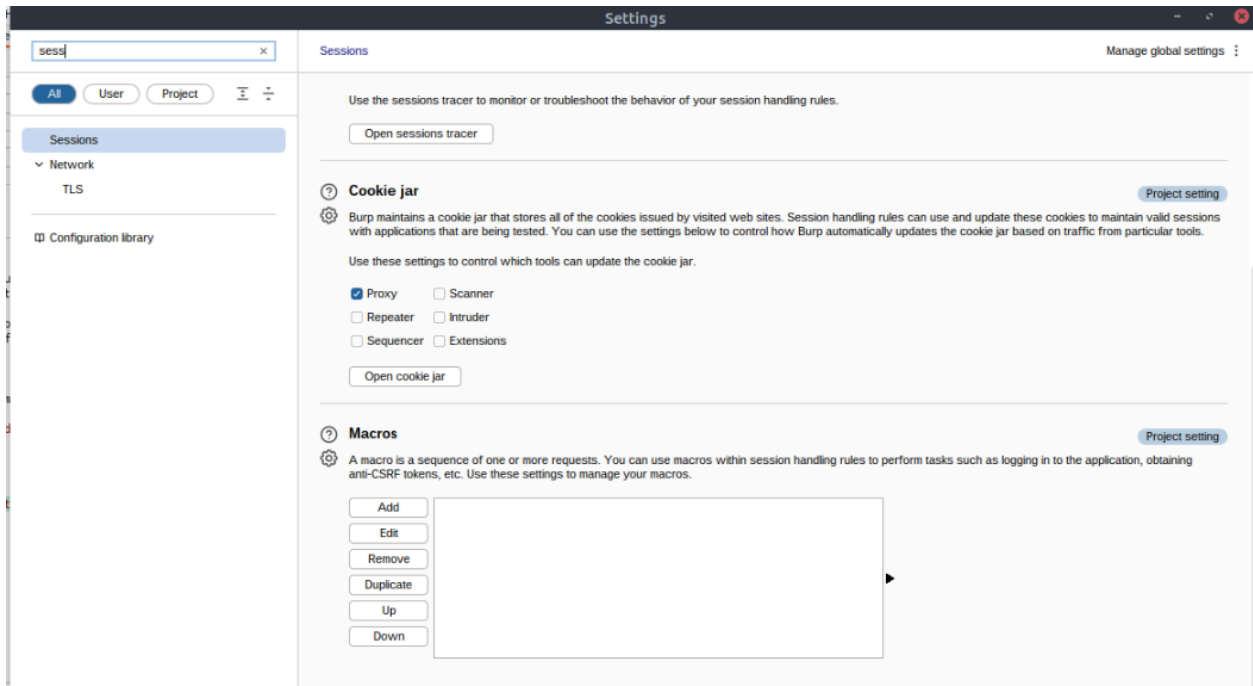
I have configured everything and now we will load the same username and password list from before to test out the credentials again and try find someone with some significance to get past the admin authentication.



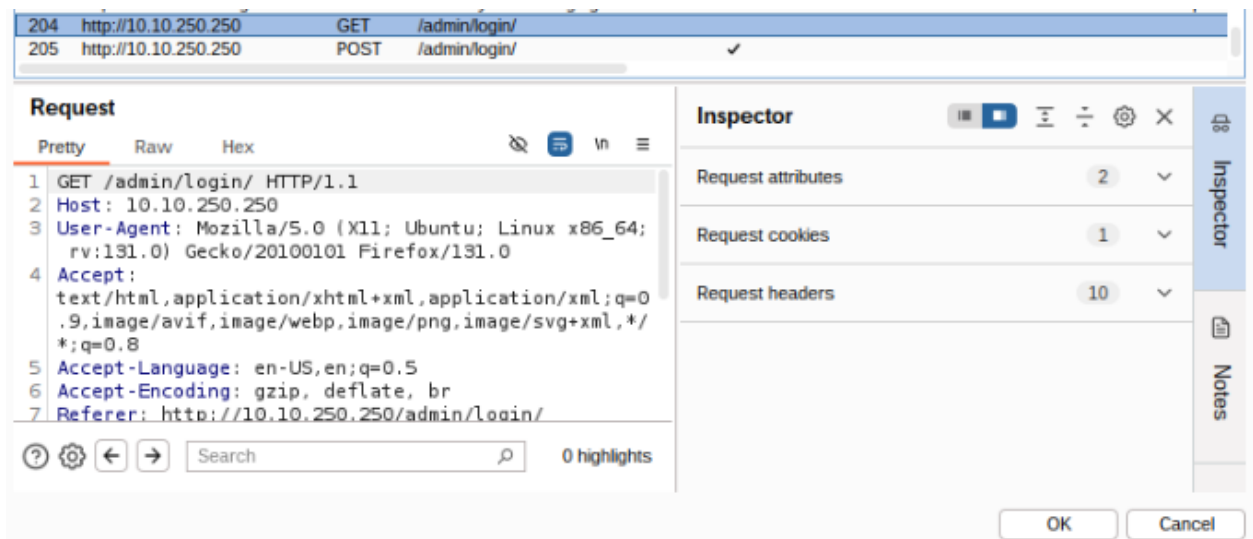
The screenshot shows the 'Payloads' tab in Burp Suite. At the top, there are settings for 'Payload position' (set to 1), 'Payload type' (set to 'Simple list'), 'Payload count' (100), and 'Request count' (0). Below these is the 'Payload configuration' section, which includes a description: 'This payload type lets you configure a simple list of strings that are used as payloads.' On the left of this section are buttons for 'Paste', 'Load...', 'Remove', 'Clear', and 'Deduplicate'. On the right is a list of strings: 'lmadden', 'j.poole', 'i.hopkins', 'l.mayo', 'm.roberts', 'm.ratliff', 'j.morrison', 'r.carroll', 't.nichols', 'd.decker', and 'l.nana'. Below the list is an 'Add' button and a text input field with the placeholder 'Enter a new item'. At the bottom of this section is a dropdown menu labeled 'Add from list... [Pro version only]'. The bottom section is 'Payload processing', which includes a description: 'You can define rules to perform various processing tasks on each payload before it is used.' On the left are buttons for 'Add', 'Edit', and 'Remove'. On the right is a table with two columns: 'Enabled' and 'Rule'. The 'Enabled' column has a checkbox, and the 'Rule' column is currently empty.

Same process as last time just loading the username.txt file and password.txt file on separate payload positions and then I will start the attack.

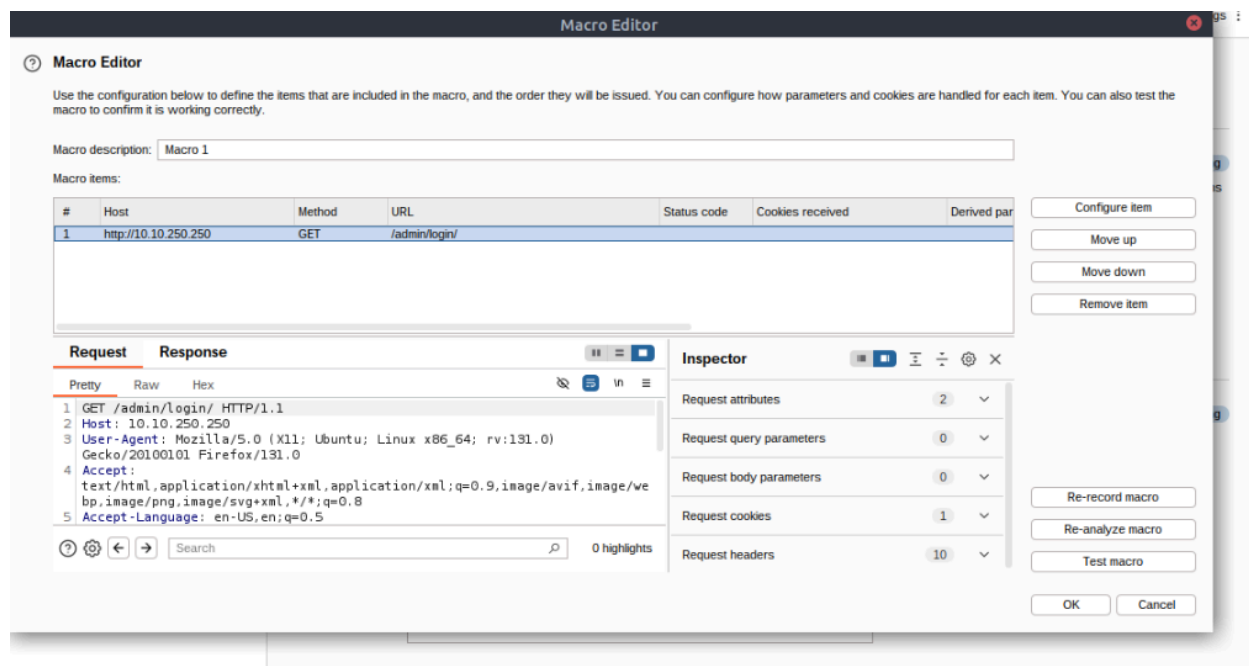
But we now need to find a way to grab the changing login token and session cookie. Recursive grep won't work here due to the direct response. So we will need to build a "Macro". This will allow us to perform the same set of actions repeatedly. In this case we simply want to send a GET request to /admin/login/.



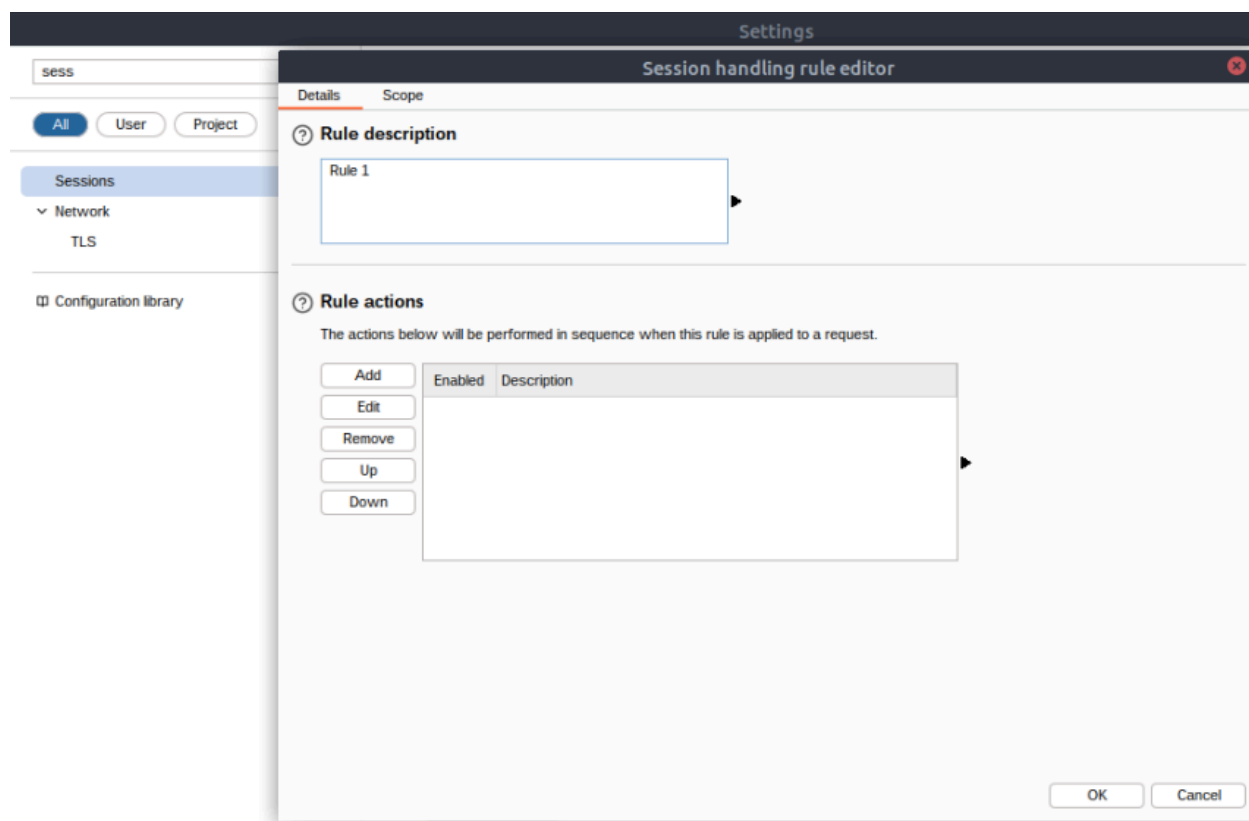
“Add” -



“Ok”



“Ok”



”Add” New rule - Details and scope.

Session handling rule editor

Details

Scope

?

Rule description

Rule 1

?

Rule actions

The actions below will be performed in sequence when this rule is applied to a request.

Add

Edit

Remove

Up

Down

Enabled	Description
<input checked="" type="checkbox"/>	run macro: Macro 1

OK

Cancel

Run macro: Macro 1.

Session handling rule editor

Details

Scope

Tools scope

Select the tools that this rule will be applied to.

☐ Target

☐ Scanner

☐ Repeater

☒ Intruder

☐ Sequencer

☐ Extensions

☐ Proxy (use with caution)

URL scope

Use the configuration below to control which URLs this rule applies to.

☐ Include all URLs

☒ Use suite scope [defined in Target tab]

☐ Use custom scope

Parameter scope

You can restrict the rule to requests containing specific parameters if required.

☐ Restrict to requests containing these parameters:

Edit

OK

Cancel

Configure scope settings.

Update Macro settings to update only the following parameters and headers “loginToken” and “sessions” ..

ID	IP	Host	User-Agent	Accept	Accept-Encoding	Content-Type	Content-Length	Origin	Connection	Referer	Cookies	Upgrade-Insecure-Requests	Priority
35	b.hennet	dbel											
36	b.buckner	efant											
37	a.peters	78945812											
38	d.whitney	ravett											

Request	Response
GET / HTTP/1.1	HTTP/1.1 200 OK

```
POST /admin/login/ HTTP/1.1
Host: 10.10.250.250
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:131.0) Gecko/20100101 Firefox/131.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/svg+xml,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Content-Type: application/x-www-form-urlencoded
Content-Length: 80
Origin: http://10.10.250.250
Connection: keep-alive
Referer: http://10.10.250.250/admin/login/
Cookie: session=eyJTdXBwZXJ0dGQzOTUyZmRlLnRlcjYxZWVudS01Q1I4NnZkYyOGNmMGZmZTg1ZWZMT09HMDhkdWZkZDkyZm9yLmFkEPQ.F5nj0GcTvvgU9t9rOnTyOjQETDU
Upgrade-Insecure-Requests: 1
Priority: u=0, i=1

username=02zhennett&password=bellaAI&loginToken=67cf6f682ff59f62405fd12ba18890e9ad
```

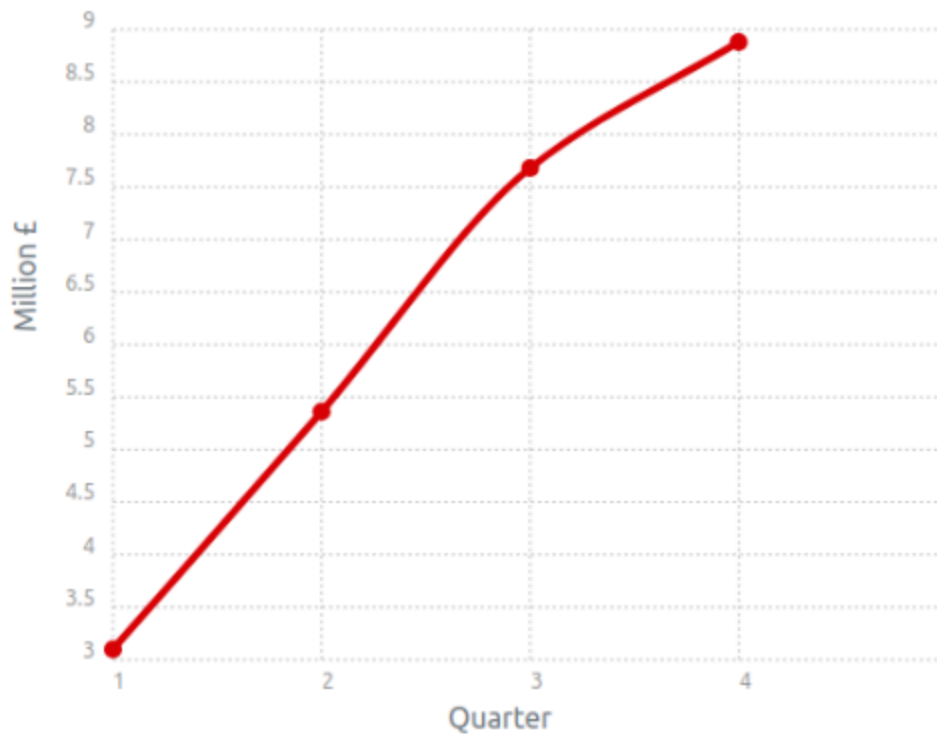
Username: o.bennett

Password: bella1

Let's now see if we can use this to login.

# Admin Home

## Profits for 2024



We did it!

Answer the questions below

What username and password combination indicates a successful login attempt? The answer format is "username:password".

o.bennett:bella1

✓ Correct Answer

Conclusion:

This was certainly more advanced than the repeater, using Macros and really piecing bits of information together using wordlists can really generate some sensitive information. I love how using these techniques we can bypass security features such as cookies and login tokens. It just goes to show how effective tools and applications can be when you know how to configure them. This was very enjoyable, to see how different attacks can be performed, and how to overcome issues should they arise.