2021 Edition

OWASP Top 10 Report

OWASP Top 10-2021 report

29 October 2024 12:13

Task 1 OWASP Introduction

i. So OWAPS full form is open web application security protocol that every 3 year publish vulnerabilities list API vulnerabilities
- Broken Access Control
- Cryptographic Failures
- Injection
- Insecure Design
- Security Misconfiguration
- Vulnerable and Outdated Components
- Identification and Authentication Failures
- Software and Data Integrity Failures
- Security Logging & Monitoring Failures
- Server-Side Request Forgery (SSRF)

Task 2 Accessing Machines

a. In this module, we deploy our lab for practice.

b. We have two methods for this: we can use the TryHackMe virtual box on the website, or use our Kali Linux to establish a connection.

c. When we successfully deploy the labs, we get a flag to verify the connection.

Task 3 1. Broken Access Control

1. Website pages are protected from visitors; only an admin user can manage the website and modify access credential files. This ensures proper management and protection. However, a website may have broken access control.
2. **Types of Visitors:**
    1. **Regular Visitor**
        - Able to view sensitive information from other users
        - Accessing unauthorized functionality
    2. **Website Visitor**
3. Can only access their own information
4. These types of vulnerabilities allow attackers to bypass authorization, enabling them to view sensitive data and perform malicious activities.

Task 4 Broken Access Control (IDOR Challenge)

1. **Insecure Direct Object Reference (IDOR)**
2. This refers to access control vulnerabilities where users can access resources that do not belong to them.
3. One example is when a user logs into a banking application and successfully authenticates themselves. The application then redirects the user to a different page where they can perform various actions.
4. If the user leaves the session without logging out, they may assume everything is perfectly secure. However, there is a significant problem: an attacker could use the same URL left behind, modify certain parameters, and gain unauthorized access to other users' details.
5. By changing some parameters of the URL, attackers can potentially access other people's credential information.

Task 5: Cryptographic Failures

1. **Cryptographic Failure**
2. Messages are encrypted using cryptographic algorithms to protect them, ensuring only the authorized recipient can access them. If someone without permission misuses cryptographic algorithms to access messages, files, or data, it is called a cryptographic failure.
    - **Encrypting data in transit**: This refers to data transmitted over the internet using cryptographic algorithms. Even if someone captures the packets, they cannot read the data because it is encrypted.
    - **Encrypting data at rest**: This refers to data stored on a server. Even the server owner cannot access your data, as it is protected by encryption. This is known as encrypting data at rest.

Task 6: Cryptographic Failures (Supporting Material 1)

In this section, I will provide practical examples of how to access a database using the terminal.

1. To find out information about any file, we use the command:
   file file_name

For example, if our file is `example.db` (SQLite), we use:

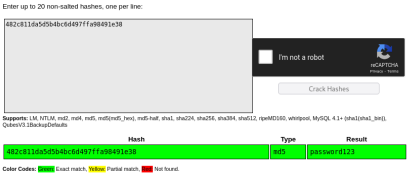
   file example.db

   This will display information about the file.

2. To learn more about the `.db` file, we first enter the SQLite terminal using the command:
   sqlite3 example.db
3. To find out information about a table, we use:
   PRAGMA table_info(table_name);
4. To see the contents of a table, we use:
   SELECT * FROM table_name;

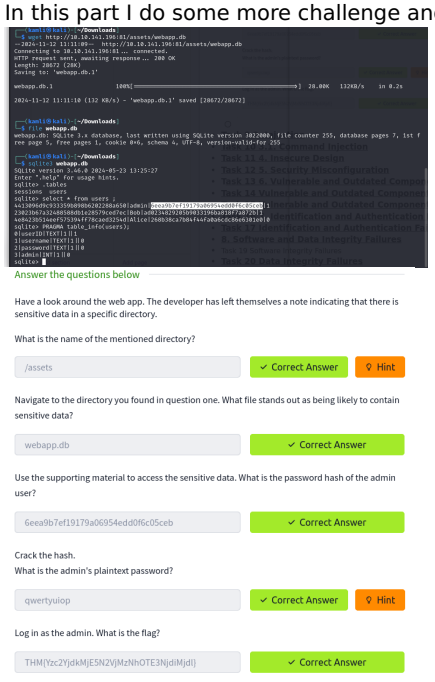
Task 7 Cryptographic Failures (Supporting Material 2)

- **In this part, we decode some hashes using CrackStation**
- For example, if we have the hash 482c811da5d5b4bc6d497ffa98491e38, we can go to CrackStation, input this hash, and after a few seconds, the website will crack the password



Task 8 Cryptographic Failures (Challenge)

- In this part I do some more challenge and solve them_



Task 9 3. Injection

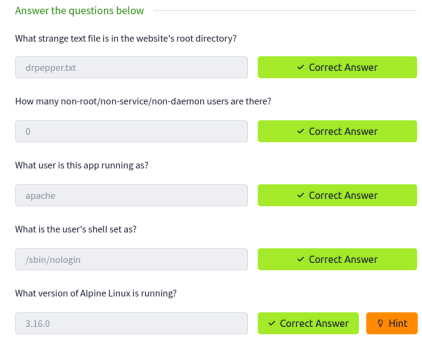
1. **Injection**
2. In today's world, we interact with servers using client browsers where we pass some inputs through URLs. By modifying these input parameters and making requests, the server may process them as valid and return sophisticated results. This is called injection, and it has two types:
3. **SQL Injection** – This injection targets database servers. It involves user-controlled

input in SQL queries (typically malformed SQL queries). When the error message is returned by the database server, the attacker analyzes it and gathers information about the database structure.

4. **Command Injection** – This injection occurs in the system command layer, where an attacker executes arbitrary system commands on the application server. This type of attack allows them to gain access to the user's system.
5. **Prevention** – Several techniques to prevent injection attacks:
   - **Using an Allow List** – The input coming from users is first checked against a predefined list (safe list). If the input matches correctly, it is considered safe; otherwise, it is rejected, and an error may be thrown.
   - **Stripping Input** – If the input contains dangerous characters, they are removed before processing.

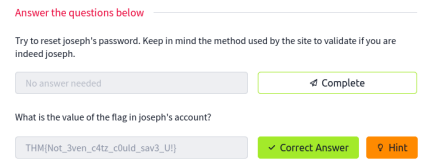
Task 10 3.1. Command Injection

- I carefully read this part and understand how bash code run inside
- Then using this knowledge I solve some practice question



Task 11 4. Insecure Design

- This vulnerability arises from poorly configured and implemented architecture. Sometimes, during testing, the tester disables certain security methods for convenience. When the application moves to the deployment phase, they forget to re-enable these security measures, which results in an insecure vulnerability.
- For example, suppose your application authenticates users using OTP (One-Time Password). During the testing phase, the tester or developer may disable this functionality for ease of testing. However, if this OTP function is not re-enabled before the application is published, it creates an insecure vulnerability.



Task 12 5. Security Misconfiguration

- **Security Misconfiguration**

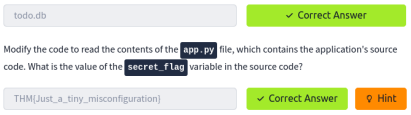
  Security misconfigurations are distinct from other top 10 vulnerabilities because they occur when security could have been appropriately configured but was not. Even if you download the latest updates, misconfigurations can still leave your application vulnerable.

  Examples of Security Misconfiguration include:

  - Poorly configured permissions on services
  - Having unnecessary features enabled, such as unused services
  - Default accounts with unchanged passwords

  Interactive Console Challenge

  I went to the provided link, and for the interactive console, I simply changed the URL by adding an extra /console. This gave me a Python prompt, where I solved each question.



Task 13 6. Vulnerable and Outdated Components

- In this type of vulnerability, what happens is that we provide update patches, but for some reason, we forget to apply them. As a result, the vulnerability remains in the application, and attackers can take advantage of these vulnerabilities.
- This is called **Vulnerable and Outdated Components**.

Task 15 Vulnerable and Outdated Components – Lab

- I carefully read the chapter and solve this challenge
-

Task 16 7. Identification and Authentication Failures

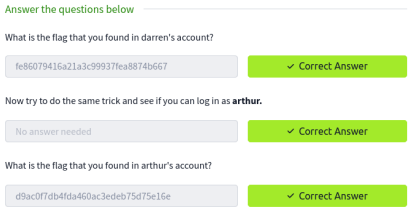
1. Authentication and session management are core components of modern web applications.
2. Authentication allows users to gain access to web applications by verifying their identities. The most common method to identify a user is through a username and password.
3. The server verifies the user, and if the credentials are correct, the server creates session cookies that store the user's behavior and activities.
4. If an attacker is able to find flaws in an authentication mechanism, they might successfully gain access to other users' accounts and sensitive information.
5. **Common Flaws in Authentication:**
    1. **Brute Force Attack** – If the web application uses a username and password, an attacker can attempt a brute force attack by trying multiple username and password combinations.
    2. **Use of Weak Credentials** – If the web application allows weak passwords like "password123" or "password1", attackers can easily guess them.
    3. **Weak Session Cookies** – Session cookies are used to store temporary user data and track activity. Attackers can create their own cookies by guessing predefined values and exploit the vulnerabilities.

6 Mitigation Strategies to Avoid These Attacks:

1. To avoid password guessing, the application should provide a strong password policy.
2. To prevent brute force attacks, set a mechanism to limit password attempts.
3. Implement multi-factor authentication (MFA) to enhance the security of credential data.

Task 17 Identification and Authentication Failures Practical

- In this chapter I carefully study the content usingthoes information I solve the practice challenge



18. Software and Data Integrity Failures

1. **Data Integrity**
2. The data we have should not be modified or changed. If the data looks modified, then it lacks integrity.
3. In cybersecurity, we often download many applications, so we need to ensure that the file we download is original and has not been modified. To verify this, when we download a file, we also get a hash, such as MD5, SHA-1, or SHA-256. We then recalculate the hash of the downloaded file and compare it to the original hash. If both hashes are equal, it means we have downloaded the original file.
4. **Types of Data Integrity:**

- **Software Integrity** – Ensures that the software has not been altered or tampered with.
- **Data Integrity** – Ensures that the data has not been modified, corrupted, or lost during transmission or storage.

Task 19 Software Integrity Failures

- **I read chapter carefullly and solve the challenges**

Task 20 Data Integrity Failures

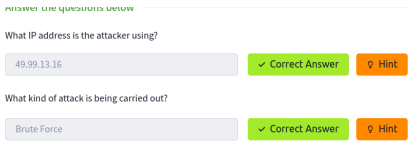
1. **Session Management**
2. When a user logs into a website, they are assigned a token, and this token maintains the session until it ends.
3. Sessions come in many forms, but they are usually handled via cookies.
4. Cookies are key-value pairs stored in the user's browser and are automatically sent with each request to the website that issued them.
5. For example, when you log into a web application, a cookie is assigned to you. This cookie contains your information, like your username. With each request, the web application understands that the request is coming from you by using these cookies.
6. Cookies are stored on the user's browser, and if the user modifies that cookie and sends a request, they could access another user's information, thereby breaking data integrity.
7. One solution to ensure data integrity is using **JSON Web Tokens (JWT)**. JWT is a token that allows you to store key-value pairs and provides integrity as part of the token. This token is given to the user, and the user cannot change or alter it. The structure of the token is as follows:

i. 

   - **Header**: Contains metadata indicating that it is a JWT and specifies the signing algorithm in use (e.g., HS256).
   - **Payload**: Contains the key-value pairs with the data that the web application wants the client to store.
   - **Signature**: Similar to a hash, it is used to verify the authenticity of the signature. If the hash doesn't match the payload, it indicates that the JWT has been tampered with.
8. Each of the three parts of the token is base64-encoded in plain text. The signature contains binary data, so even if you decode it, you won't be able to make much sense of it.

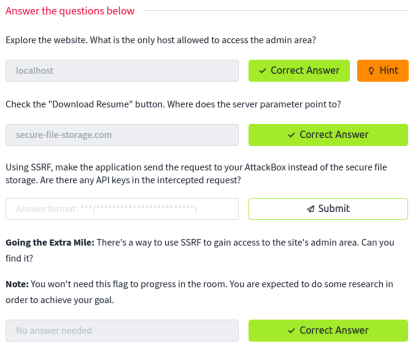
Task 21 9. Security Logging and Monitoring Failures

1. When an application is set up, every action performed by the user is logged. This helps to track the user's activity.
2. Once their actions are traced, the impact and risks can be determined. Without logging:
3. **If an attacker gains access to a particular web application, the significant impacts include:**
   1. **Regulatory Damage**: If an attacker gains access to personally identifiable user information and there is no record of this, it can negatively affect the organization's regulatory standing and compliance.
   2. **Risk of Further Attacks**: If an attacker's presence remains undetected due to the lack of logging, it could allow them to launch further attacks against the web application owners. This could involve stealing credentials, attacking infrastructure, and more.
4. **Information Stored in Logs:**
   - HTTP status codes
   - Time stamps
   - Usernames
   - API endpoints/page locations
   - IP addresses
5. These pieces of information are sensitive, so they should be stored in a secure place with multiple copies in different locations.
6. Using this information and task files, I was able to solve the challenges.



Task 22 10. Server-Side Request Forgery (SSRF)

- When the attacker demands access to files and directories that they do not have permission to access, but the server is not properly configured and allows unauthorized access, this is called **Server Side Request Forgery (SSRF)**.
- Using this information, I was able to solve the challenges.



PicoCTF Report

PicoCTF Report

29 October 2024 13:47

General Skill section

---

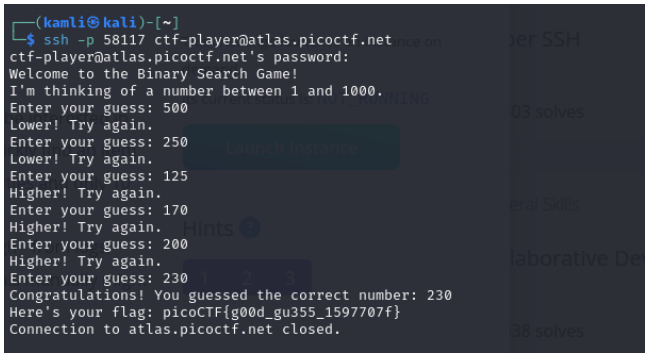
1 Binary Search(General skill)

Description - Want to play a game? As you use more of the shell, you might be interested Binary search is a classic algorithm used to quickly find an item in a sorted list. Can you find 1000 possibilities and only 10 guesses.

Process -

- **Log in**: Use the provided credentials to access the instance.
- **Number guessing game**: The program asks you to guess a number until you get it ri
- **Success**: After several attempts, you guess the correct number.
- **Submit the flag**: Once you guess correctly, you submit the flag as instructed.

Solution



2 Time Machine (general skill)

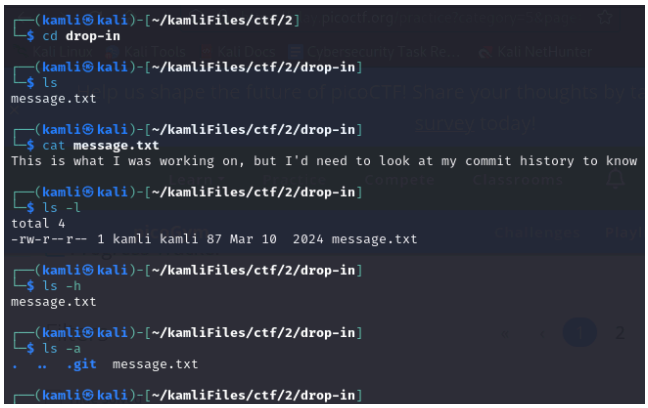
Description

What was I last working on? I remember writing a note to help me remember... You can d challenge files here:

Process

- **First, I download the ZIP file.**
- **Then, I unzip it using the appropriate command.**
- **There are multiple files, so I navigate into the directory.**
- **Inside, I see a message.txt file, which I open using a text editor.**
- **I then use ls -a to check for any hidden files.**
- **I find a .git directory.**

Solution



```
└─$ git log
commit b92bdd8ec87a21ba45e77bd9bed3e4893faafd0f (HEAD → master)
Author: picoCTF <ops@picoctf.com>
Date:   Sat Mar 9 21:10:29 2024 +0000

    picoCTF{t1m3m@ch1n3_5cde9075}
```

3 Super SSH (general skill)

Description

Using a Secure Shell (SSH) is going to be pretty important.
Additional details will be available after launching your challenge instance

Process

- **First, I establish a connection using the information provided to me.**
- **Then, I type "yes" to confirm any prompts (such as adding the host to known h**
- **Next, I enter the provided password.**
- **Finally, I obtain the flag.**

Solution

```
┌──(kamli㉿kali)-[~/kamliFiles/ctf/2/drop-in]
└─$ ssh -p 63138 ctf-player@titan.picoctf.net
ssh: Could not resolve hostname titan.picoctf.net: Temporary failure in name re

┌──(kamli㉿kali)-[~/kamliFiles/ctf/2/drop-in]
└─$ ssh -p 63138  ctf-player@titan.picoctf.net
The authenticity of host '[titan.picoctf.net]:63138 ([3.139.174.234]:63138)' ca
ED25519 key fingerprint is SHA256:4S9EbTSSRZm32I+cdM5TyzthpQryv5kudRP9PIKT7XQ.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '[titan.picoctf.net]:63138' (ED25519) to the list of
ctf-player@titan.picoctf.net's password:
Welcome ctf-player, here's your flag: picoCTF{s3cur3_c0nn3ct10n_3e293eea}
Connection to titan.picoctf.net closed.

┌──(kamli㉿kali)-[~/kamliFiles/ctf/2/drop-in]
└─$
```

4 Commitment Issues(General skill)

Description

I accidentally wrote the flag down. Good thing I deleted it! You download the challenge files

Process

- **First, I download the challenge file.**
- **Then, I unzip the challenge file.**
- **Next, I navigate to the extracted directory.**
- **I see a message.txt file, and I use cat to view its contents.**
- **I run ls -a to check for any hidden files.**
- **I find a .git directory.**
- **I run git log to find a commit.**
- **Using the commit hash, I run git checkout <commit-hash>.**
- **Finally, I find and retrieve the flag.**

Solution

```
┌──(kamli㉿kali)-[~/kamliFiles/ctf/3]
└─$ ls
'challenge(1).zip'   drop-in

┌──(kamli㉿kali)-[~/kamliFiles/ctf/3]
└─$ cd drop-in

┌──(kamli㉿kali)-[~/kamliFiles/ctf/3/drop-in]
└─$ ls
message.txt

┌──(kamli㉿kali)-[~/kamliFiles/ctf/3/drop-in]
└─$ cat message.txt
TOP SECRET

┌──(kamli㉿kali)-[~/kamliFiles/ctf/3/drop-in]
└─$ ls -a
.  ..  .git  message.txt

┌──(kamli㉿kali)-[~/kamliFiles/ctf/3/drop-in]
└─$ git log
commit ef0b7cc6b98367fa168573c931e0f7098ef59182 (HEAD → master)
Author: picoCTF <ops@picoctf.com>
Date:   Tue Mar 12 00:06:20 2024 +0000

    remove sensitive info

commit ea859bf3b5d94ee74ce5ee1afa3edd7d4d6b35f0
Author: picoCTF <ops@picoctf.com>
Date:   Tue Mar 12 00:06:20 2024 +0000

    create flag

┌──(kamli㉿kali)-[~/kamliFiles/ctf/3/drop-in]
└─$ git checkout ea859bf3b5d94ee74ce5ee1afa3edd7d4d6b35f0

Note: switching to 'ea859bf3b5d94ee74ce5ee1afa3edd7d4d6b35f0'.

You are in 'detached HEAD' state. You can look around, make experimental
changes and commit them, and you can discard any commits you make in this
state without impacting any branches by switching back to a branch.

If you want to create a new branch to retain commits you create, you may
do so (now or later) by using -c with the switch command. Example:

  git switch -c <new-branch-name>

Or undo this operation with:

  git switch -

Turn off this advice by setting config variable advice.detachedHead to false

HEAD is now at ea859bf create flag

┌──(kamli㉿kali)-[~/kamliFiles/ctf/3/drop-in]
└─$
```

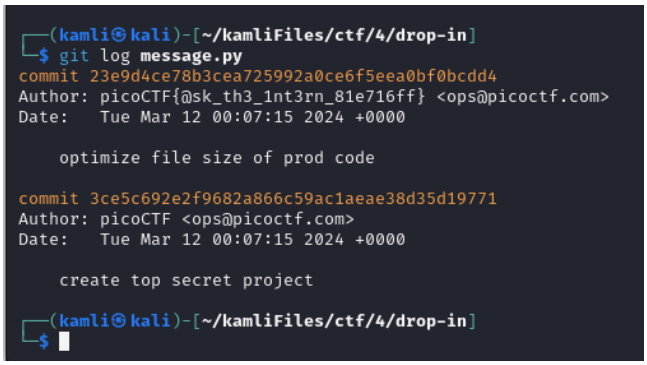
5 Blame Game(General skill)

Description

Someone's commits seems to be preventing the program from working. Who is it?

Process

- **First, I unzip the challenge file and navigate to the extracted directory.**
- **Inside, I see a message.txt file, which I open.**
- **I don’t understand the contents of the message.txt file.**
- **I run the command git log message.py to check the version history of messa**
- **Finally, I find the flag within the commit history.**



6 repetitions(General skill)

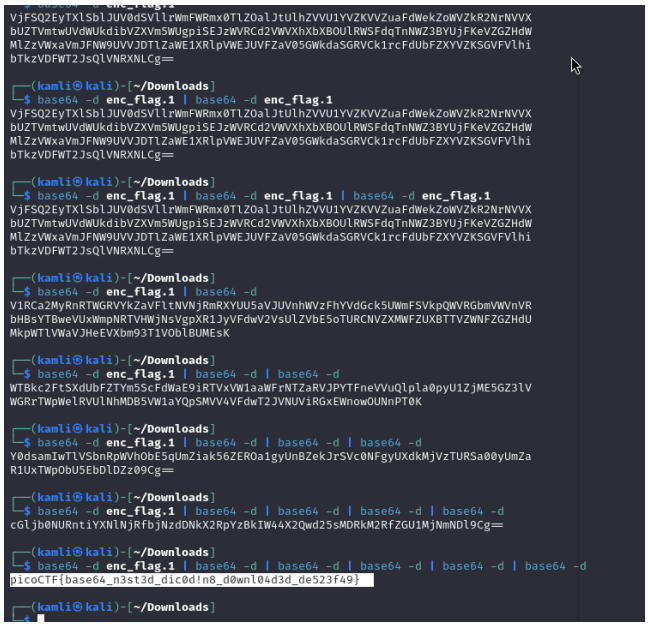
Description

Can you make sense of this file?

Process

- **I download the file.**
- **Then, I review its contents.**
- **I notice that the file is base64 encoded, so I decide to decode it.**
- **I perform multiple rounds of decoding until I reach the final decoded content.**

Solution



7 Big Zip(general skill)

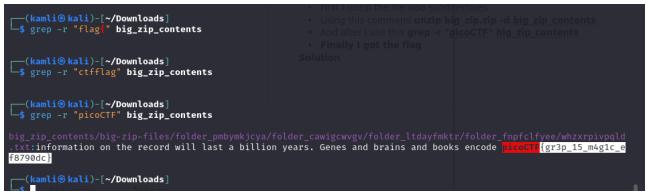
Description

Unzip this archive and find the flag.

Process

- **First, I unzip the file and its subdirectories using the following command:** unzip big big_zip_contents
- Then, I search for the flag within the unzipped content using: grep -r "picoCTF" big_zip_c
- Finally, I find the flag.

Solution



8 First Find (general skill)

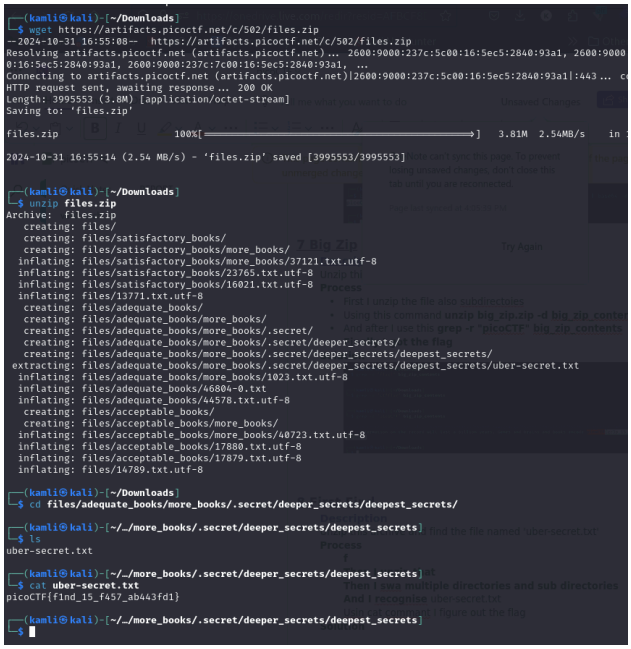
Description

Unzip this archive and find the file named 'uber-secret.txt'

Process

- **First, I download the ZIP file.**
- **Then, I unzip it.**
- **I see multiple directories and subdirectories.**
- **I recognize a file named uber-secret.txt.**
- **Using the cat command, I read the file and find the flag:**
- **cat uber-secret.txt**

Solution



9 Runme.py(general skill)

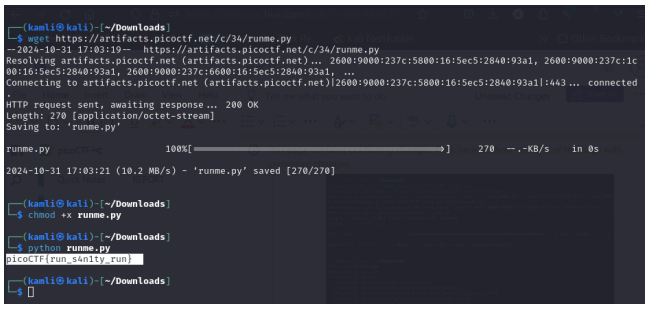
Description

Run the runme.py script to get the flag. Download the script with your browser or with wget i

Process

- **First I download the file using wget <link>**
- **Then I saw there is a file runme.py**
- **I run that file**
- **Finally I get the flag**

Solution



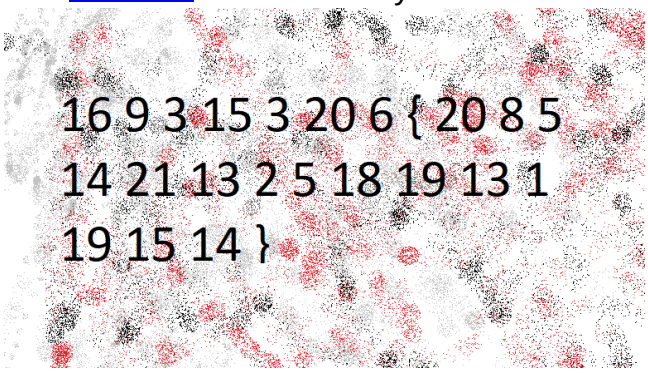
Cryptography section

---

11 The Numbers

Description

The numbers... what do they mean?



Process -

- **First, I notice that each number represents a letter of the alphabet.**
- **I convert each number to its corresponding letter (e.g., 1 = A, 2 = B, etc.).**
- **The hint specifies that the result should be in capital letters, so I ensure each le**
- **Finally, I obtain the flag: PICOCTF{THENUMBERSMASON}.**

12 13

Description

Cryptography can be easy, do you know what ROT13 is? cvpbPGS{abg_gbb_onq_bs_n

Processs

- **ROT13 is a cipher where each letter is replaced by the letter 13 places after i**
- **I applied the ROT13 transformation to each letter of the given text.**
- **I obtained a flag, but it was missing a letter, so I added the missing characte**
- **Finally, I got the complete flag: picoCTF{not_too_bad_of_a_problem}.**

13 interencdec

Description

Can you get the real meaning from this file. Download the file here.

Process

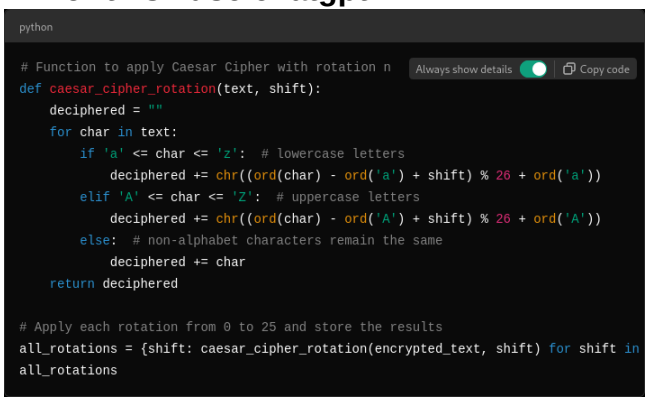
- **First, I download the file.**
- **Then, I decode the file using the base64 -d command: base64 -d <file>**
- **I copy the decoded content, which is inside the quotes.**
- **I paste the copied content into the website https://www.base64decode.or**
- **I decode the output again.**
- **Finally, I get the flag: picoCTF{caesar_d3cr9pt3d_a47c6d69}.**

14 rotation

Description

You will find the flag after decrypting this file Download the encrypted flag here.

- **Process**
  - **for this I use chatgpt**



```python
# Function to apply Caesar Cipher with rotation n
def caesar_cipher_rotation(text, shift):
    deciphered = ""
    for char in text:
        if 'a' <= char <= 'z':  # lowercase letters
            deciphered += chr((ord(char) - ord('a') + shift) % 26 + ord('a'))
        elif 'A' <= char <= 'Z':  # uppercase letters
            deciphered += chr((ord(char) - ord('A') + shift) % 26 + ord('A'))
        else:  # non-alphabet characters remain the same
            deciphered += char
    return deciphered

# Apply each rotation from 0 to 25 and store the results
all_rotations = {shift: caesar_cipher_rotation(encrypted_text, shift) for shift in
all_rotations
```

- **First, I try to use ROT13 on the flag, but it doesn't give the correct answer.**
- **Then, I try cipher rotation, applying multiple rotations.**
- **After several rotations, I find that applying a rotation of 18 gives the correct res**
- **Finally, I get the flag.**

```
{0: 'xqkwKBN{z0bib1wv_l3kzgxb3l_4k71n5j0}',
 1: 'yrlxLCO{a0cjc1xw_m3lahyc3m_4l71o5k0}',
 2: 'zsmyMDP{b0dkd1yx_n3mbizd3n_4m71p5l0}',
 3: 'atnzNEQ{c0ele1zy_o3ncjae3o_4n71q5m0}',
 4: 'buoaOFR{d0fmf1az_p3odkbf3p_4o71r5n0}',
 5: 'cvpbPGS{e0gng1ba_q3pelcg3q_4p71s5o0}',
 6: 'dwqcQHT{f0hoh1cb_r3qfmdh3r_4q71t5p0}',
 7: 'exrdRIU{g0ipi1dc_s3rgnei3s_4r71u5q0}',
 8: 'fyseSJV{h0jqj1ed_t3shofj3t_4s71v5r0}',
 9: 'gztfTKW{i0krk1fe_u3tipgk3u_4t71w5s0}',
 10: 'haugULX{j0lsl1gf_v3ujqhl3v_4u71x5t0}',
 11: 'ibvhVMY{k0mtm1hg_w3vkrim3w_4v71y5u0}',
 12: 'jcwiWNZ{l0nun1ih_x3wlsjn3x_4w71z5v0}',
 13: 'kdxjXOA{m0ovo1ji_y3xmtko3y_4x71a5w0}',
 14: 'leykYPB{n0pwp1kj_z3ynulp3z_4y71b5x0}',
 15: 'mfzlZQC{o0qxq1lk_a3zovmq3a_4z71c5y0}',
 16: 'ngamARD{p0ryr1ml_b3apwnr3b_4a71d5z0}',
 17: 'ohbnBSE{q0szs1nm_c3bqxos3c_4b71e5a0}',
 18: 'picoCTF{r0tat1on_d3crypt3d_4c71f5b0}',
 19: 'qjdpDUG{s0ubu1po_e3dszqu3e_4d71g5c0}',
 20: 'rkeqEVH{t0vcv1qp_f3etarv3f_4e71h5d0}',
 21: 'slfrFWI{u0wdw1rq_g3fubsw3g_4f71i5e0}',
 22: 'tmgsGXJ{v0xex1sr_h3gvctx3h_4g71j5f0}',
 23: 'unhtHYK{w0yfy1ts_i3hwduy3i_4h7 g0}',
```


- **picoCTF{r0tat1on_d3crypt3d_4c71f5b0}**

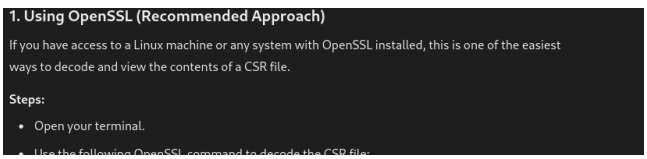
15 ReadMyCert

Description

How about we take you on an adventure on exploring certificate signing requests Tak file here.

Process

- **For this, I asked ChatGPT how to solve it.**
- **ChatGPT suggested using the OpenSSL approach.**

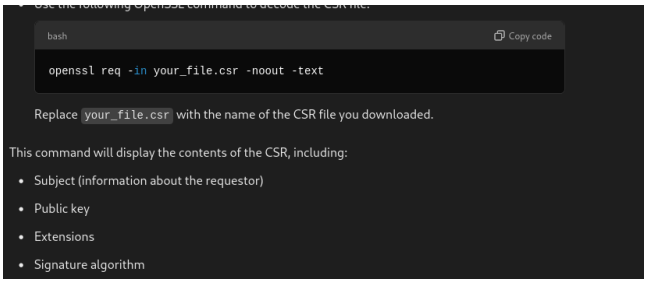


1. Using OpenSSL (Recommended Approach)

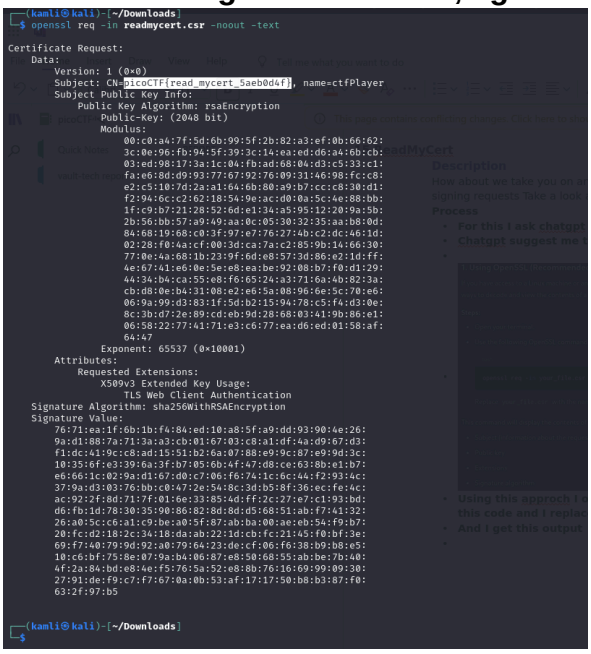
If you have access to a Linux machine or any system with OpenSSL installed, this is one of the easiest ways to decode and view the contents of a CSR file.

Steps:

- Open your terminal.



- **Using this approach, I open my Kali Linux terminal.**
- **I use the suggested OpenSSL command, replacing the file with the appropri**
- **After executing the command, I get the desired output**



- **After executing the command, I get the output.**
- **Finally, I obtain the flag: picoCTF{read_mycert_5aeb0d4f}.**

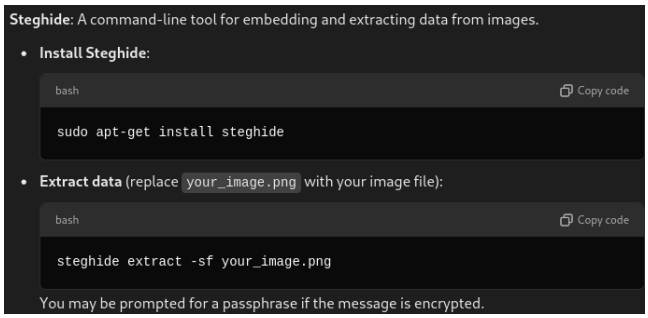
16 HideToSee

Description

How about some hide and seek heh? Look at this image here.

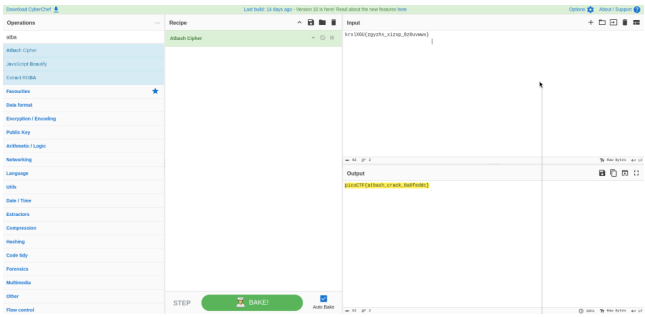
Process

- **For this, I asked ChatGPT how to solve it.**
- **ChatGPT provided several approaches.**
- **I decided to use steghide to solve the challenge.**



- **I replace the image name as instructed.**
- **Running steghide generates an encrypted.txt file.**
- **I open the encrypted.txt file to check its contents.**

- **I find a text that looks like a flag, but it seems scrambled.**
- **I use CipherChef, select the Atbash cipher, and apply it to the text.**
- **After decoding, I paste the resulting text.**
- **Finally, I get the correct flag.**
-



17 substitution0

Description

A message has come in but it seems to be all scrambled. Luckily it seems to have the
Can you crack this substitution cipher?

Process

- **According to the hint, I carefully observe the beginning of the message.**
- **I notice that each character represents an original letter of the alphabet.**
- **I create a table to map each character to its corresponding alphabet letter.**
-

Original Alphabet	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
Scrambled Text	O	H	N	F	U	M	W	S	V	Z	L	X	E	G	C	P	T	A	J	D	Y	I	R	K	Q	B

- **At the end, the message looks like a flag.**
- **I compare that message to the table I created.**
- **After mapping the characters using the table, I obtain the flag:**
- **picoCTF{5UB5717U710N_3V0LU710N_03055505}**

18 substitution2

Description

It seems that another encrypted message has been intercepted. The encryptor se
their lesson though and now there isn't any punctuation! Can you still crack the cip

Process

- **For this, I go to dcode.fr.**
- **On the site, I select the "Monoalphabetic Substitution Decoder."**
- **I paste the encoded message into the decoder.**
- **After decoding, I get the flag:**
- **PICOCTF{N6R4M_4N41Y515_15_73D10U5_702F03FC}**

19 spelling-quiz

Description

I found the flag, but my brother wrote a program to encrypt all his text files. He has a s
guide too, but I don't know if that helps.

Process

- **First, I unzip the file.**
- **Inside, I find three files: two .txt files and one .py file.**

- **I analyze each file, but they don't make any sense.**
- **Then, I visit the website www.boxentriq.com to analyze the message.**
- **I paste the message into the website.**
- **The website tells me that it's a monoalphabetic cipher and provides a link to cipher tool.**
- **I go to the tool and paste words from quiz.txt. I also paste the encrypted flag**
- **After some trial and error, I manage to decode part of the message. Some se making sense. I put the sentence inside picoCTF{} and finally get the compl**
- **The final flag is:**
- **picoCTF{perhaps_the_dog_jumped_over_was_just_tired}**

Reverse Engineering section

---

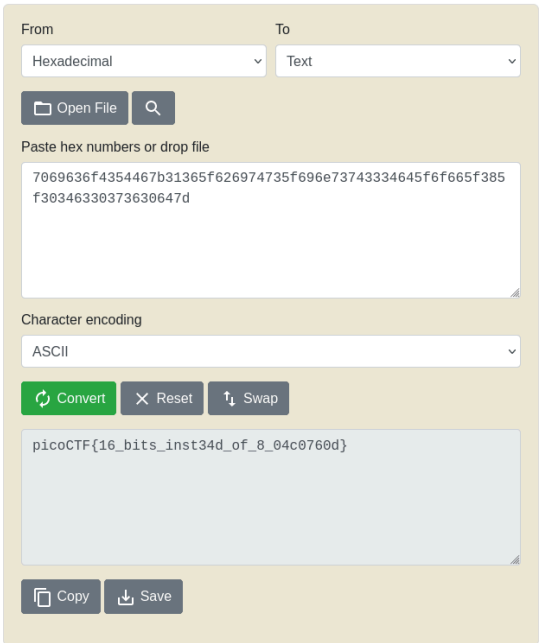
20 Transformation

Description

I wonder what this really is... [enc](#) ''.join([chr((ord(flag[i]) << 8) + ord(flag[i + 1])) for i in 2)])

Process

- First I download the enc file
- Then I use a python script **([picoCTF 2021 Transformation](#))**
- 

- Here I get a hex text I copy this and use a web site that convert this hex to ascii
- **https://www.rapidtables.com/convert/number/hex-to-ascii.html?x=**
- 

- Finally I get the flag
- **picoCTF{16_bits_inst34d_of_8_04c0760d}**

21 bloat.py

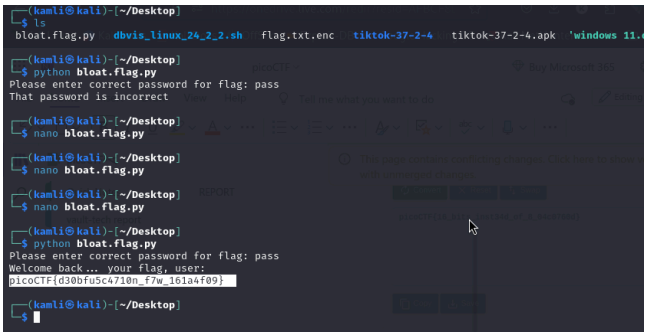
Description

Can you get the flag? Run this Python program in the same directory as this encrypte

Process

- **First, I download the files.**
- **I try to run the program, but my password is incorrect multiple times.**
- **Then, I remove the part of the code where it handles the login process.**
- **After modifying the code, I try logging in again and successfully log in.**
- **I find a YouTube video that helps me locate the flag: picoCTF 2022 | Reve bloat.py.**
- 

- **Finally, I find the flag.**



22 Bbbbloat

Description

Can you get the flag? Reverse engineer this binary.

Process

- **First, I download the executable file.**
- **When I try to execute it, it asks for some number input, but I don't have that**
- **So, I decide to use reverse engineering to find the solution.**
- **I find a YouTube video that helps me with the reverse engineering process.**
- picoCTF 2022 | Reverse Engineering | Bbbbloat

- I follow this step and I get the numer
- Finally I found the flag
- **picoCTF{cu7_7h3_bl047_695036e3}**

23 crackme-py

Description

crackme.py

Process

- **First, I download the file.**
- **I try to understand the challenge, but it's not clear at first.**
- **I find a YouTube video that helps me solve the challenge: picoCTF Walkt py.**
- 

- **After following the video, I find the flag:**
- **picoCTF{1|\/|_4_p34|\|ut_a79b6c2d}**
-

24 Shop

Description

Best Stuff - Cheap Stuff, Buy Buy Buy... Store Instance: source. The shop is open for mercury.picoctf.net 37799.

Process

I follow this youtube video to get flag

picoCTF 2021 shop

Image upload failed. Please try again

picoCTF{b4d_brogrammer_591a895a}

25 " format string_0"

Category: Binary Exploitation

Problem Description: The challenge requires exploiting a heap overflow vulneral source code to retrieve the flag.

Approach:

1. **Run the program** and enter a string longer than 64 bytes. This causes a heap
2. **After exploiting the heap overflow**, check for the flag.
3. **Finally, I retrieve the flag.**

Flag: picoCTF{7h3_cu570m3r_15_n3v3r_SEGFAULT_74f6c0e7}

26 "heap 1"

Category: Binary Exploitation

Problem Description: The challenge requires setting a variable str_var to "pico" i flag.

Approach:

1. **Run the program** and inspect what the str_var is.
2. To set str_var to "pico," **add data to the buffer**. You need to provide input that manipulates the value of str_var.
3. **Enter a string** of the form p%x%x%x%x%x%x%x%x%x%x%x%x%x%x%x%x%x%x%x%p the format string vulnerability to overwrite str_var with "pico".
4. After entering the correct input, **check for the flag** and successfully retrieve it.

Flag: picoCTF{starting_to_get_the_hang_ce5bee9b}

27 "binary Search"

Category: General Skill

Problem Description: The challenge asks you to guess a number between 1 and chances to retrieve the flag.

Approach:

1. **Run the program** and begin guessing numbers one by one.
2. Since the program follows a **binary search algorithm**, it uses a divide-and-co narrow down the possible range of numbers.
3. By leveraging this approach, you can efficiently guess the correct number by h space with each guess.

4. After making the correct guess on the **10th attempt**, you retrieve the flag.

Flag: picoCTF{g00d_gu355_2e90d29b}

28 "time machine"

Category: General Skill

Problem Description: The challenge asks you to find flags within git files.

Approach:

1. **Download the challenge.zip file** and unzip it.
2. After unzipping, **navigate to the drop-in directory** created.
3. Once inside the directory, **run the command git log** to view the commit histor
4. From the git logs, you can identify and retrieve the flag.

Flag: picoCTF{t1m3m@ch1n3_b476ca06}

29 "super ssh"

Category: General Skill

Problem Description: The challenge requires you to log in via the shell to retriev

Approach:

1. **Open the terminal** and log in as the ctf-player user.
2. The system may ask for **fingerprint verification**. Grant the necessary permiss authentication.
3. **Enter the password** when prompted and successfully log in.
4. Once logged in, **retrieve the flag**.

Flag: picoCTF{s3cur3_c0nn3ct10n_07a987ac}

30 "endianness"

Category: General Skill

Problem Description: The challenge asks you to determine the small-endian and representations of a word for the flag.

Approach:

1. **Open the terminal** and run the provided code.
2. **Provide a word** as input.
3. Visit a website like **CipherChef** and paste the word into the tool.
4. Select the **hex filter** to convert the word into its hexadecimal representation.
5. Once you have the hexadecimal code, apply the following transformations:
   - For **small-endian**, reverse the byte order, placing the smaller digits first. Fo output is 78953f4521, the small-endian format would be 21453f9578.
   - For **big-endian**, the hexadecimal code remains unchanged, such as 78953
6. Using the above process, you will find the correct representation and retrieve t
7. **Flag: picoCTF{3ndi4n_sw4p_su33ess_02999450}**

31 "commitment issue"

Category: General Skill

Problem Description: The challenge requires you to find the flag by examining th

Approach:

1. **Download the challenge.zip file** and unzip it.
2. Navigate to the **drop-in directory** where the files are located.
3. Run the command git log to view the commit history. You will find a code (com
4. Use the command git checkout <commit-hash> to check out the specific comm
5. Open the **message.txt** file using the cat command to reveal the flag.

Flag: picoCTF{s@n1t1z3_cf09a485}

33 Glory of the Garden (Forensic)

Description

This garden contains more than it seems.

Process

- **Download the challenge file** (which is an image).
- Visit the website **StegOnline** (https://www.stegonline.com/).
- Upload the image to the website.
- Use the website's tools to **extract the hidden data** from the image.
- After extraction, you will find the flag hidden within the image.
- **Flag:**
- **picoCTF{more_than_m33ts_the_3y3657BaB2C}**

34 "heap0"Category: Binary Exploitation

Problem Description: The challenge asks you to find a way to exploit a heap overflow v provided source code and obtain the flag.

Approach:

i. **Run the program** and select **option 2** from the available choices.
ii. **Enter a string** that is larger than **64 bytes** (to trigger the heap overflow).
iii. As the heap gets exploited, the program will reveal the flag.

Flag: picoCTF{my_first_heap_overflow_1ad0e1a6}

35 " big zip"

Category: General Skill

Problem Description: The challenge requires you to find the flag by using the grep com

Approach:

1. **Download the file** using the provided link.
2. **Unzip the file** to access its contents.
3. **Use the grep command** to search for the term "pico" in the files: grep -rl "pico" big-zi
4. This command will return the path of a .txt file containing the flag. In this case, the pat
5. big-zip-files/folder_pmbymkjcya/folder_cawigcwvgv/folder_ltdayfmktr/folder_fnpfclfyee
6. **Open the file** using the cat command to view the content and find the flag:
7. cat big-zip-files/folder_pmbymkjcya/folder_cawigcwvgv/folder_ltdayfmktr/folder_fnpfcl
8. Flag: picoCTF{gr3p_15_m4g1c_ef8790dc}

36 "first find"

Category: General Skill

Problem Description: The task is to find the flag using the filename “uber-secret.txt”.

Approach:

1. **Download the file** using the provided link.
2. **Unzip the file** to access its contents.
3. **Use the find command** to locate the file named "uber-secret.txt": find files | grep ub
4. The command will return the path of the uber-secret.txt file. In this case, the file path
be: files/adequate_books/more_books/.secret/deeper_secrets/deepest_secrets/uber-
5. **Open the file** using the cat command to view its contents and extract the flag: cat
files/adequate_books/more_books/.secret/deeper_secrets/deepest_secrets/uber-sec
6. Flag: picoCTF{f1nd_15_f457_ab443fd1}

37 "Insp3ct0r"

Category: Web Exploitation

Problem Description: The goal is to inspect the web page and extract the flag from the

Approach:

1. **Open the Web Page:** Navigate to the web page provided in the challenge.
2. **Inspect the Source Code:**
   - Press Ctrl + U (or right-click the page and select "View Page Source") to open the so
   page.
3. **Locate Flag Parts:**
   - **First Part of the Flag:** Search within the HTML part of the source code for the first pa
   be visible as plain text or within an HTML tag.
   - **Second Part of the Flag:** Look for the .css file linked in the HTML code. Open the lin
   search for the second part of the flag.
   - **Final Part of the Flag:** Similarly, locate the .js file referenced in the source code. Op
   for the final part of the flag.
4. **Combine the Flag Parts:** Once you have all the parts from the HTML, CSS, and JS
form the full flag.
   Flag: picoCTF{tru3_d3t3ct1ve_0r_ju5t_lucky?f10be399}

38 "where are the robots"

Category: Web Exploitation

Problem Description: The goal is to find the secret files hidden on a web page.

Approach:

1. **Check the robots.txt File:**
   - Open the web page provided in the challenge.
   - After the URL of the web page, add /robots.txt. For example: http://example.com/robo
   - The robots.txt file typically contains directives for search engines about which pages
   It may also list secret files or directories that are intentionally hidden.

2 Look for Secret Files in robots.txt:

- In the robots.txt file, search for paths or files that are not intended to be publicly index
These files could contain parts of the flag or additional clues.

- For example, you might find something like: Disallow: /477ce.html

3 Access the Secret File:

- Based on the information from robots.txt, go to the secret file by appending the file pa URL. For example:
- http://example.com/477ce.html

4 Retrieve the Flag:

- Once you access the file, you should find the flag.

Flag: picoCTF{ca1cu1at1ng_Mach1n3s_477ce}

39 "packer"

Category: Reverse Engineering

Problem Description: You are tasked with finding the secret files hidden in the pages of

Approach:

- **Download the File:**
  - First, download the file using the link provided in the challenge. This file is likely ar binary file.
- **Make the File Executable:**
  - Open a terminal in the directory where the file was downloaded.
  - Run the following command to make the file executable:
  - chmod +x out
  - This ensures that the file has the proper permissions to be executed.
- **Extract Strings from the File:**
  - Next, use the strings command to extract readable text from the binary file. The grep used to search for the term "flag" within the output. Run the following command: strin
  - This will return an encrypted or encoded string related to the flag. The output could lo hex or another encoded format
- **Decode the Flag:**
  - Once you have the encrypted or encoded flag, head to a website like **CipherChef** (https://www.cipherchef.com) for decoding.
  - Paste the encrypted flag into the tool and select the appropriate filter (for example, "F is in hexadecimal format).
  - The tool will decode the flag for you.
- **Retrieve the Flag:**
  - Once decoded, you will obtain the flag in its readable form.
- Flag: picoCTF{U9X_UnP4ck1N6_B1n4Ri3S_6ff964ef}

40 "PcapPoisoning"

Category: Forensics

Problem Description: You are tasked with finding the flag hidden in a .pcap (packet cap

Approach:

1. **Download the File:**
   - Start by downloading the .pcap file (named trace.pcap) using the provided link.
2. **Extract Strings from the .pcap File:**
   - Use the strings command to extract readable strings from the .pcap file. This com sequences of printable characters in binary files.
   - In the terminal, run the following command: strings trace.pcap | grep picoCTF

- This command will search through the strings extracted from the .pcap file and loc of the string "picoCTF."

3 Find the Flag:

- After running the command, you will find the flag within the output, as it will match picoCTF{...}.
- Flag: picoCTF{P64P_4N4L7S1S_SU55355FUL_f621fa37}

Bug Bounty Report

Website

1. I take this website from https://hackerone.com/23andme_bbp?type=team

23andMe Bug Bounty
http://23andme.com@23andme
23andMe provides personalized DNA insights that empower individuals to explore their ancestry, health risks, and genetic traits!Bug Bounty Program launched in Dec 2023

Information about this website

1. Ip information

2 Ports informatio

```
┌──(kamli㉿kali)-[~]
└─$ nmap 104.16.183.73
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-11-19 14:50 IST
Nmap scan report for 104.16.183.73
Host is up (0.087s latency).
Not shown: 996 filtered tcp ports (no-response)
PORT     STATE SERVICE
80/tcp   open  http
443/tcp  open  https
8080/tcp open  http-proxy
8443/tcp open  https-alt

Nmap done: 1 IP address (1 host up) scanned in 13.15 seconds

┌──(kamli㉿kali)-[~]
└─$
```

3 Subdomain

Subdomains

Search subdomains...

HOSTNAME	IP ADDRESS
piwik.auth.23andme.com	N/A

4 links connected to this website

7	7	0
Found links	Dofollow	Nofollow

External links (7)

- https://customercare.23andme.com/hc/en-us/articles/202907970/
- https://customercare.23andme.com
- https://www.23andme.com/en-ca/legal/terms-of-service/
- https://www.23andme.com/legal/privacy/
- https://www.23andme.com/legal/us-privacy/#washington-consumer-health-data-privacy-policy
- https://www.23andme.com/about/cookies/
- https://www.23andme.com/test-info/

5 Technology Stack

Background

Site title	23andMe Login - Sign Into Your Account	**Date first seen**	March 2018
Site rank	20123	**Primary language**	English
Description	Login to your 23andMe account to access your DNA test results, health and ancestry reports, manage your account and more.		

Network

Site	https://auth.23andme.com	**Domain**	23andme.com
Netblock Owner	Cloudflare, Inc.	**Nameserver**	alina.ns.cloudflare.com
Hosting company	Cloudflare	**Domain registrar**	Unknown
Hosting country	US	**Nameserver organisation**	whois.cloudflare.com
IPv4 address	104.16.182.73 (VirusTotal)	**Organisation**	Unknown
IPv4 autonomous systems	AS13335	**DNS admin**	dns@cloudflare.com
IPv6 address	2606:4700:0:0:0:0:6810:b649	**Top Level Domain**	Commercial entities (.com)
IPv6 autonomous systems	AS13335	**DNS Security Extensions**	Enabled
Reverse DNS	Unknown		

IP delegation

IPv4 address (104.16.182.73)

IP range	Country	Name	Description
::ffff:0.0.0.0/96	United States	IANA-IPV4-MAPPED-ADDRESS	Internet Assigned Numbers Authority
↳ 104.0.0.0-104.255.255.255	United States	NET104	American Registry for Internet Numbers
↳ 104.16.0.0-104.31.255.255	United States	CLOUDFLARENET	Cloudflare, Inc.
↳ 104.16.182.73	United States	CLOUDFLARENET	Cloudflare, Inc.

IPv6 address (2606:4700:0:0:0:0:6810:b649)

IP range	Country	Name	Description
::/0	N/A	ROOT	Root inet6num object
↳ 2600::/12	United States	NET6-2600	American Registry for Internet Numbers
↳ 2606:4700::/32	United States	CLOUDFLARENET	Cloudflare, Inc.
↳ 2606:4700:0:0:0:0:6810:b649	United States	CLOUDFLARENET	Cloudflare, Inc.

SSL/TLS

Assurance	Domain validation	**Perfect Forward Secrecy**	Yes
Common name	23andme.com	**Supported TLS Extensions**	RFC8446 key share, RFC8446 supported versions, RFC4366 server name, RFC7301 application-layer protocol negotiation, RFC4366 status request, RFC8446 early data
Organisation	Not Present	**Application-Layer Protocol Negotiation**	h2
State	Not Present	**Next Protocol Negotiation**	Not Present
Country	Not Present	**Issuing organisation**	Google Trust Services
Organisational unit	Not Present	**Issuer common name**	WE1
Subject Alternative Name	23andme.com, *.23andme.com, *.education.23andme.com, *.medical.23andme.com, *.research.23andme.com, *.therapeutics.23andme.com, *.mg.23andme.com, *.mg-system.23andme.com, *.mg-account.23andme.com	**Issuer unit**	Not Present
Validity period	From Nov 17 2024 to Feb 16 2025 (2 months, 4 weeks)	**Issuer location**	Not Present
Matches hostname	Yes	**Issuer country**	US
Server	cloudflare	**Issuer state**	Not Present
Public key algorithm	id-ecPublicKey	**Certificate Revocation Lists**	http://c.pki.goog/we1/hx34Vi4ORbM.crl
Protocol version	TLSv1.3	**Certificate Hash**	7FjFsxkIa5JZuf++J9l+dDoqJ4k
Public key length	256	**Public Key Hash**	00387843522f5bc32ea122de5eac814fe99cce5ae84d1a4118e7236aa80c0829
Certificate check	ok	**OCSP servers**	http://o.pki.goog/s/we1/MIg
Signature algorithm	ecdsa-with-SHA256	**OCSP stapling response**	Certificate valid
Serial number	0x308847314f61c456118e4219abd1a951	**OCSP data generated**	Nov 18 00:42:44 2024 GMT
Cipher	TLS_AES_256_GCM_SHA384	**OCSP data expires**	Nov 24 23:42:43 2024 GMT

Signed Certificate Timestamps (SCTs)

Source	Log	Timestamp	Signature Verification
Certificate	*Unknown* zxFW7tUufK/zh1vZaS6b6RpxZ0qwF+ysAdJbd87MOwg=	2024-11-18 00:42:44	*Unknown*
Certificate	*Unknown* ouMK5EXvva2bfjjtR2d3U9eCW4SU1yteGyzEuVCkR+c=	2024-11-18 00:42:44	*Unknown*

SSLv3/POODLE

This site does not support the SSL version 3 protocol.

6 Domain information

```
#
# ARIN WHOIS data and services are subject to the Terms of Use
# available at: https://www.arin.net/resources/registry/whois/tou/
#
# If you see inaccuracies in the results, please report at
# https://www.arin.net/resources/registry/whois/inaccuracy_reporting/
#
# Copyright 1997-2024, American Registry for Internet Numbers, Ltd.
#


NetRange:       104.16.0.0 - 104.31.255.255
CIDR:           104.16.0.0/12
NetName:        CLOUDFLARENET
NetHandle:      NET-104-16-0-0-1
Parent:         NET104 (NET-104-0-0-0-0)
```



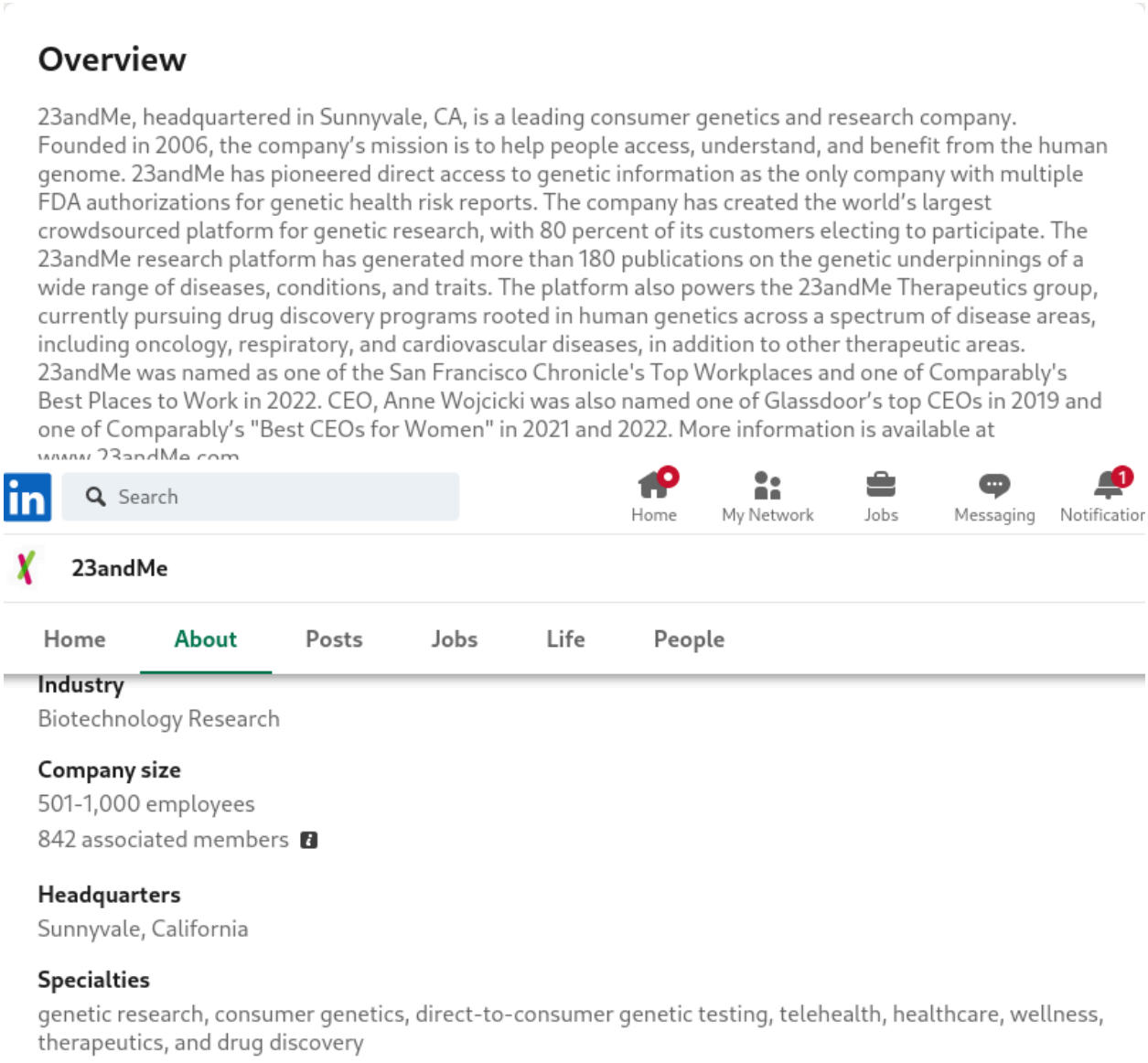
```
Organization:   Cloudflare, Inc. (CLOUD14)
RegDate:        2014-03-28
Updated:        2024-09-04
Comment:        All Cloudflare abuse reporting can be done via https://www.cloudflare.com
Comment:        Geofeed: https://api.cloudflare.com/local-ip-ranges.csv
Ref:            https://rdap.arin.net/registry/ip/104.16.0.0



OrgName:        Cloudflare, Inc.
OrgId:          CLOUD14
Address:        101 Townsend Street
City:           San Francisco
StateProv:      CA
PostalCode:     94107
Country:        US
RegDate:        2010-07-09
Updated:        2021-07-01
Ref:            https://rdap.arin.net/registry/entity/CLOUD14
```

```
OrgRoutingHandle: CLOUD146-ARIN
OrgRoutingName:   Cloudflare-NOC
OrgRoutingPhone:  +1-650-319-8930
OrgRoutingEmail:  noc@cloudflare.com
OrgRoutingRef:    https://rdap.arin.net/registry/entity/CLOUD146-ARIN

OrgNOCHandle: CLOUD146-ARIN
OrgNOCName:   Cloudflare-NOC
OrgNOCPhone:  +1-650-319-8930
OrgNOCEmail:  noc@cloudflare.com
OrgNOCRef:    https://rdap.arin.net/registry/entity/CLOUD146-ARIN

OrgTechHandle: ADMIN2521-ARIN
OrgTechName:   Admin
OrgTechPhone:  +1-650-319-8930
```



```
OrgAbuseHandle: ABUSE2916-ARIN
OrgAbuseName:   Abuse
OrgAbusePhone:  +1-650-319-8930
OrgAbuseEmail:  abuse@cloudflare.com
OrgAbuseRef:    https://rdap.arin.net/registry/entity/ABUSE2916-ARIN

RNOCHandle: NOC11962-ARIN
RNOCName:   NOC
RNOCPhone:  +1-650-319-8930
RNOCEmail:  noc@cloudflare.com
RNOCRef:    https://rdap.arin.net/registry/entity/NOC11962-ARIN

RTechHandle: ADMIN2521-ARIN
RTechName:   Admin
RTechPhone:  +1-650-319-8930
RTechEmail:  rir@cloudflare.com
RTechRef:    https://rdap.arin.net/registry/entity/ADMIN2521-ARIN

RAbuseHandle: ABUSE2916-ARIN
RAbuseName:   Abuse
RAbusePhone:  +1-650-319-8930
RAbuseEmail:  abuse@cloudflare.com
RAbuseRef:    https://rdap.arin.net/registry/entity/ABUSE2916-ARIN


#
# ARIN WHOIS data and services are subject to the Terms of Use
# available at: https://www.arin.net/resources/registry/whois/tou/
#
# If you see inaccuracies in the results, please report at
# https://www.arin.net/resources/registry/whois/inaccuracy_reporting/
#
# Copyright 1997-2024, American Registry for Internet Numbers, Ltd.
#
```

7 information from social media



Overview

23andMe, headquartered in Sunnyvale, CA, is a leading consumer genetics and research company. Founded in 2006, the company's mission is to help people access, understand, and benefit from the human genome. 23andMe has pioneered direct access to genetic information as the only company with multiple FDA authorizations for genetic health risk reports. The company has created the world's largest crowdsourced platform for genetic research, with 80 percent of its customers electing to participate. The 23andMe research platform has generated more than 180 publications on the genetic underpinnings of a wide range of diseases, conditions, and traits. The platform also powers the 23andMe Therapeutics group, currently pursuing drug discovery programs rooted in human genetics across a spectrum of disease areas, including oncology, respiratory, and cardiovascular diseases, in addition to other therapeutic areas. 23andMe was named as one of the San Francisco Chronicle's Top Workplaces and one of Comparably's Best Places to Work in 2022. CEO, Anne Wojcicki was also named one of Glassdoor's top CEOs in 2019 and one of Comparably's "Best CEOs for Women" in 2021 and 2022. More information is available at www.23andMe.com

Search

Home My Network Jobs Messaging Notificatior

23andMe

Home About Posts Jobs Life People

Industry
Biotechnology Research

Company size
501-1,000 employees
842 associated members

Headquarters
Sunnyvale, California

Specialties
genetic research, consumer genetics, direct-to-consumer genetic testing, telehealth, healthcare, wellness, therapeutics, and drug discovery