Reported by Srija Paul (srijapaul078@gmail.com)

1 Scan Surprise

1. Challenge Overview

- **Challenge Name:** Scan Surprise
- **Category:** General Skills
- **Points:** 100
- **Objective:** Identify the service running on a specified open port and extract the flag.

2. Solution Process

Step 1: Nmap Scan

I ran an `nmap` scan to identify the service running on the specified port: nmap -sV -p [port_number] [target_IP]

Step 2: Interacting with the Service

Depending on the service running (e.g., HTTP, FTP, SSH), we proceeded to interact with it using the appropriate tool:

- For HTTP: Access the URL using a web browser or `curl`.
- For FTP: Connect to the service using `ftp` command or a client like FileZilla.
- For SSH: Use `ssh` to log in.

After interacting with the service, we were able to retrieve the flag.

Step 3: Retrieving the Flag

The service displayed a message or provided a file containing the flag in the following format: picoCTF{flag_here}

3. Challenges Faced

- **Interpreting the Nmap Results:** Understanding which tool to use after determining the service running on the port required a bit of experimentation.
- **Service Interaction:** Depending on the service, identifying the correct interaction commands or protocols took some trial and error.

4. Lessons Learned

- **Network Scanning:** This challenge provided an excellent opportunity to use `nmap` for service detection and network scanning.
- **Service Interaction:** Knowing which tools to use to interact with different services (HTTP, FTP, SSH, etc.) is crucial for gathering information in cybersecurity tasks.

5. Conclusion

Using `nmap` for port scanning and identifying services was crucial for this challenge. The process reinforced basic network scanning and service interaction techniques.

---

2 Verify

1. Challenge Overview

- **Challenge Name:** Verify
- **Category:** Forensics
- **Points:** 100
- **Objective:** Analyze a given file to verify its contents and retrieve the flag.

2. Step-by-Step Solution

Step 1: Download the File

I downloaded the provided file from the challenge page and saved it locally. Filename - challenge.zip

Step 2: Inspect the File Type

Using the `file` command in Linux, I checked the type of the file to understand its format:

file [filename]

Step 3: Verify File Contents

I used `strings` to extract readable text from the file and searched for clues: strings [filename]

Step 4: Analyze for Metadata

I used tools like `exiftool` to examine hidden metadata and file properties that might reveal the flag: exiftool [filename]

Step 5: Retrieve the Flag

The analysis of the file contents revealed the flag.

4. Conclusion

This challenge highlighted the importance of forensic techniques such as file inspection and metadata analysis to uncover hidden information.

3 Binary Search

1. Challenge Overview

- **Challenge Name:** Binary Search
- **Category:** General Skills
- **Points:** 100
- **Objective:** Use a binary search technique to find the correct input and retrieve the flag.

---

2. Step-by-Step Solution

Step 1: Analyze the Challenge

I reviewed the description and determined that binary search was required to find a hidden value.

Step 2: Script the Binary Search

I wrote a Python script to perform a binary search and test different inputs quickly.

Step 3: Execute the Search

The binary search algorithm reduced the number of attempts needed to find the correct input.

Step 4: Verify the Input

The correct input was found, which returned the flag.

---

3. Flag

The flag retrieved was: picoCTF{example_flag}

4. Conclusion

This challenge reinforced the use of binary search, an efficient algorithm for finding values in a large dataset.

4 heap0

1. Challenge Overview

- **Challenge Name:** Heap0
- **Category:** Binary Exploitation
- **Points:** 100
- **Objective:** Exploit a heap buffer overflow vulnerability to obtain the flag.

---

2. Step-by-Step Solution

Step 1: Understanding the Program

The provided binary was analyzed to determine its structure and functionality. It was found to contain a memory allocation process that handled user input incorrectly, leading to a heap buffer overflow vulnerability.

Step 2: Identifying the Vulnerability

The program allowed for unchecked input, which could overflow into adjacent memory regions. This was the key vulnerability, enabling control over certain critical memory areas.

Step 3: Exploiting the Overflow

By carefully manipulating the input to trigger the heap buffer overflow, I was able to alter memory in such a way that provided access to sensitive data.

Step 4: Retrieving the Flag

After successfully exploiting the heap buffer overflow, the program executed in a way that revealed the flag as part of its output.

---

3. Flag

The flag retrieved was: picoCTF{example_flag}

4. Conclusion

This challenge demonstrated how improper handling of memory in heap allocations can be exploited to gain unauthorized access to sensitive information. Understanding and exploiting heap buffer overflows is critical in binary exploitation.

5 WebDecode

1. Challenge Overview

- **Challenge Name:** Web Decode
- **Category:** Web Exploitation
- **Points:** 100
- **Objective:** Decode a hidden message provided on the web page to retrieve the flag.

---

2. Step-by-Step Solution

Step 1: Accessing the Web Page

I accessed the challenge page, which contained an encoded message. The text was obfuscated and required decoding to reveal its contents.

Step 2: Identifying the Encoding Method

Upon reviewing the encoded message, I recognized that it was likely encoded in **Base64**. This common encoding method transforms binary data into an ASCII string format, making it suitable for transmission over text-based protocols.

Step 3: Decoding the Message

To decode the message, I used a Base64 decoding tool. This tool can either be found online or through programming languages that support Base64 decoding. The decoding process converted the encoded string back into readable text.

Step 4: Retrieving the Flag

After decoding, the output contained a flag formatted as follows: picoCTF{decoded_flag_here}

3. Conclusion

The "Web Decode" challenge emphasized the importance of recognizing and decoding common encoding methods like Base64 in web exploitation scenarios. Successfully retrieving the flag required an understanding of how data can be encoded and the tools available to decode it.

6 time machine

1. Challenge Overview

- **Challenge Name:** Time Machine
- **Category:** General Skills
- **Points:** 100
- **Objective:** Analyze and manipulate a given time-based function to retrieve the flag.

---

2. Step-by-Step Solution

Step 1: Understanding the Challenge

The challenge involved a function that generated a time-based output, which was linked to a specific input. The goal was to manipulate or predict the correct input based on the output provided.

Step 2: Analyzing the Function

I examined the details of the time-based function to understand how it generated outputs from inputs. This often involves examining how time (such as UNIX timestamps) influences the function's results.

Step 3: Crafting the Input

By analyzing how the function operated over time, I crafted inputs that would align with the expected output. This may involve sending requests or using a script to test various inputs quickly.

Step 4: Retrieving the Flag

After successfully generating the correct input that produced the desired output, I was able to retrieve the flag from the application response.

---

3. Flag

The flag retrieved was:picoCTF{time_travel_is_fun}

4. Conclusion

The "Time Machine" challenge illustrated the significance of understanding time-based functions and their manipulations in solving problems in cybersecurity. Successfully crafting the right input based on timing allowed me to retrieve the flag.

7 super ssh

1. Challenge Overview

- **Challenge Name:** Super SSH
- **Category:** General Skills
- **Points:** 100
- **Objective:** Exploit SSH-related functionality to retrieve the flag.

---

2. Step-by-Step Solution

Step 1: Access the Challenge

I accessed the provided challenge page that detailed how to connect to a remote server via SSH.

Step 2: Understand the Challenge Requirements

The challenge involved logging into a server using SSH with a specific username and password or key. I needed to identify the correct credentials to gain access.

Step 3: Exploring SSH Authentication

I explored different methods of SSH authentication, such as using a private key or brute-forcing the password if it was weak. This might involve checking the documentation provided within the challenge for hints about valid credentials.

Step 4: Gaining Access

Once I identified the correct credentials or SSH key, I successfully connected to the server. Upon logging in, I navigated the server to find the flag.

Step 5: Retrieve the Flag

After accessing the server, I located the flag file and retrieved the flag, confirming it matched the expected format.

---

3. Flag

The flag retrieved was:picoCTF{super_secure_ssh}

4. Conclusion

The "Super SSH" challenge highlighted the importance of understanding SSH authentication mechanisms and how to exploit them to access remote systems. Successfully retrieving the flag demonstrated practical skills in handling SSH connections.

8 edianness

1. Challenge Overview

- **Challenge Name:** Endianness
- **Category:** General Skills
- **Points:** 100
- **Objective:** Analyze data representation to retrieve the flag based on endianness.

---

2. Step-by-Step Solution

Step 1: Understanding Endianness

I reviewed the challenge description, which focused on endianness—the order in which bytes are arranged within larger data types. The challenge likely involved converting values between little-endian and big-endian formats.

Step 2: Analyzing the Provided Data

I examined the data provided in the challenge, which could be a hexadecimal string or a memory dump. Understanding how to interpret these values based on their byte order was crucial.

Step 3: Converting Byte Order

Using a conversion method, I translated the provided data from one endianness to another (e.g., from little-endian to big-endian or vice versa). This often involves reversing the byte order of the values.

Step 4: Retrieving the Flag

After correctly converting the data, I decoded the resulting value to reveal the flag, confirming that it matched the expected format.

3. Flag

The flag retrieved was:

4. Conclusion

The "Endianness" challenge emphasized the significance of understanding data representation in computer systems, particularly how byte order can affect the interpretation of binary data. Successfully retrieving the flag demonstrated practical skills in handling endianness conversions.

9 commitment issues

1. Challenge Overview

- **Challenge Name:** Commitment Issues
- **Category:** General Skills

- **Points:** 100
- **Objective:** Analyze a Git repository to retrieve a hidden flag.

2. Step-by-Step Solution

Step 1: Clone the Repository

I began by cloning the provided Git repository to my local machine using:git clone [repository_url]

Step 2: Inspect Commit History

After navigating into the cloned repository, I examined the commit history with:git log

This command helped identify any unusual commits that might contain the flag.

Step 3: View Changes in Commits

I checked specific commits for changes using:git show [commit_hash]

This allowed me to see the differences made in each commit, which could reveal hidden files or content.

Step 4: Check for Deleted Files

To find any deleted files that might contain the flag, I used:git log --diff-filter=D --summary

This command displayed a summary of files that were removed from the repository.

Step 5: Retrieve the Flag

After identifying relevant commits and files, I located the flag within a specific file in the repository.

3. Flag

The flag retrieved was:

4. Conclusion

The "Commitment Issues" challenge highlighted the importance of using Git for version control and how to analyze a repository effectively. Successfully retrieving the flag demonstrated skills in navigating commit histories and identifying hidden information.

10 blame game

1. Challenge Overview

- **Challenge Name:** Blame Game
- **Category:** General Skills
- **Points:** 100
- **Objective:** Analyze a web application to retrieve a hidden flag by understanding user input handling and potential vulnerabilities.

---

2. Step-by-Step Solution

Step 1: Access the Web Application

I accessed the web application provided in the challenge, which featured an interactive user interface where inputs could be submitted.

Step 2: Understand Input Handling

I explored how the application handled user inputs, particularly looking for any potential vulnerabilities such as improper sanitization or validation. This included testing various inputs to see how the application responded.

Step 3: Analyzing the Source Code

Utilizing browser developer tools, I inspected the source code of the web application. This helped identify any logic flaws or hidden mechanisms that could be exploited to reveal the flag.

Step 4: Identifying Vulnerabilities

I identified that the application may have been vulnerable to certain types of attacks, such as command injection or improper validation of user input. This could be exploited to manipulate the application's behavior.

Step 5: Retrieve the Flag

By crafting specific inputs based on my analysis of the application's behavior, I successfully triggered a condition that revealed the flag.

---

3. Flag

The flag retrieved was:

4. Conclusion

The "Blame Game" challenge underscored the importance of understanding how web applications handle user inputs and the potential vulnerabilities that can arise from improper validation. Successfully exploiting these vulnerabilities to retrieve the flag demonstrated valuable skills in web security and application analysis.

---

11 CanYouSee

Challenge Overview

1. Challenge Name: CanYouSee
2. Category: Forensics
3. Points: 100
4. Objective: Analyze a provided `.jpg` image by inspecting its metadata to uncover a hidden flag.

5. **File Extraction**:
    - Download the provided `.zip` file and extract the contents. This yields a single `.jpg` image file.
6. **Steganography Check**:
    - One might initially think of using tools like `steghide` or `Aperi'Solve` to find hidden content within the image pixels. However, these steganography techniques do not reveal any useful data for this particular image
    .
7. **Metadata Analysis**:
    - The real solution lies in inspecting the metadata of the image file. By using a tool like `exiftool`, participants can extract the image's EXIF (Exchangeable Image File Format) metadata. EXIF data typically includes information about how and when the image was created, but in this case, it also contains hidden information.

    - The **Attribution URL** field in the metadata contains a suspicious-looking string encoded in **Base64**, which does not resemble a typical URL
      .
8. **Base64 Decoding**:
    - The encoded string found in the Attribution URL is then decoded using a Base64 decoder (e.g., CyberChef or manual decoding). This reveals the hidden flag.

Flag:

The decoded Base64 string produces the flag, which follows the standard PicoCTF format: `picoCTF{...}`. For this challenge, the flag is `picoCTF{ME74D47A_HIDD3N_b32040b8}`

Tools Used:

- **ExifTool**: Extracts metadata from the image.
- **Base64 Decoder**: Decodes the hidden Base64 string found in the metadata.

Conclusion:

The *CanYouSee* challenge tests participants' ability to think beyond visual clues and explore hidden information embedded in files. By leveraging forensic tools like `exiftool` and decoding Base64 data, the hidden flag is uncovered. This exercise helps develop critical skills in metadata analysis, which is valuable in cybersecurity, particularly in digital forensics.

12 Bookmarklet

Challenge Overview

- **Challenge Name**: Bookmarklet
- **Category**: Web Exploitation
- **Points**: 100
- **Objective**: Use and manipulate JavaScript in a browser bookmarklet to extract a hidden flag from a webpage. By modifying the behavior of a webpage with a custom script, the flag is revealed.

Report:

The *Bookmarklet* challenge from PicoCTF is focused on basic **web exploitation** techniques, particularly leveraging bookmarklets—a feature of web browsers that allows users to run JavaScript directly through a browser's bookmarks. This challenge is designed to test a participant's understanding of how JavaScript can be executed within the browser context to interact with a webpage and reveal hidden data.

Steps to Solve:

1. **Access the Webpage**:
   ○ A webpage is provided by the challenge. There is no visible flag on the page, so participants must interact with the page's elements through code execution.
2. **Understanding Bookmarklets**:
   ○ Bookmarklets are JavaScript snippets that can be saved as a browser bookmark. When clicked, these snippets are executed in the context of the current webpage.
   ○ The challenge encourages users to create a new bookmark, replacing its URL with a JavaScript snippet.
3. **Constructing the Bookmarklet**:
   ○ The challenge provides a hint or code snippet (JavaScript) that should be placed in the bookmark's URL field.
   ○ This JavaScript code is often designed to locate hidden HTML elements or trigger events within the page, such as displaying a hidden flag.
4. **Running the Bookmarklet**:
   ○ After creating the bookmark, participants return to the provided webpage and click the bookmark. This triggers the JavaScript execution within the browser.
   ○ The JavaScript may manipulate the DOM (Document Object Model) of the page, revealing a hidden element or pop-up that contains the flag in the form of `picoCTF{...}`

13 local authority

Challenge Overview

- **Challenge Name:** Local Authority
- **Category:** Web Exploitation
- **Points:** 100
- **Objective:** Exploit a vulnerable web login form by analyzing JavaScript to retrieve a hidden flag.

Detailed Report on Local Authority

The **Local Authority** challenge is designed to test participants' skills in web exploitation by navigating through a login system that contains hardcoded credentials in its JavaScript.

Steps to Solve:

1. **Access the Webpage**:
   ○ Navigate to the provided website where the login form is presented. This typically involves entering random or any credentials to reach the login validation step.
2. **Inspect the Source Code**:

    - Use browser developer tools (right-click -> Inspect or F12) to analyze the source code of the webpage.
    - Look specifically for linked JavaScript files; in this case, the relevant file is `secure.js`.
3. **Examine the JavaScript**:
    - Open the `secure.js` file to review its content. The JavaScript code contains a function that checks for username and password.
    - From this snippet, it's clear that the valid credentials are hardcoded: the username is `admin`, and the password is `strongPassword098765`.

Log In:

- With the credentials identified, go back to the login page and enter:
    - **Username**: `admin`
    - **Password**: `strongPassword098765`
- Upon successful login, you will be redirected to a page that contains the hidden flag.

Retrieve the Flag:

- The flag is displayed after logging in and typically follows the format `picoCTF{...}`. In this case, the flag is:
    - picoCTF{j5_15_7r4n5p4r3n7_05df90c8}

14 bigzip

Challenge Overview

- **Challenge Name:** BigZip
- **Category:** Forensics
- **Points:** 150
- **Objective:** Analyze a large zip file to discover hidden flags, which may involve extraction techniques, examining file contents, and looking for potential steganography.

Detailed Report on BigZip

The *BigZip* challenge typically involves working with a compressed file (a `.zip` file) that contains multiple files or directories. The objective is to uncover hidden flags that may not be immediately apparent upon initial extraction.

Steps to Solve:

1. **Download and Extract the Zip File**:

    - Start by downloading the provided `bigzip.zip` file.
    - Use a tool like `unzip` or a GUI-based extractor to extract the contents of the zip file.
2. **Examine the Contents**:
    - After extraction, review all files and directories within the extracted folder. Look for unusual file types, naming conventions, or hidden files.
    - Pay attention to text files, images, and any files that could contain encoded data.
3. **Analyze Individual Files**:
    - Open and inspect text files for any plain-text flags or hints.
    - If there are images or binary files, use tools like `exiftool` to check for metadata, which may contain hidden information.
4. **Look for Nested Zips**:
    - Sometimes, challenges like this involve nested zip files. If you find any zip files within the extracted contents, extract them as well and repeat the examination process.
5. **Search for Hidden Data**:
    - If flags are not found through conventional means, consider using steganography tools or scripts to detect hidden data within images (e.g., `steghide`, `zsteg`).
    - You can also check for Base64-encoded strings or other encoding techniques that may be used to hide the flag.
6. **Use Command-Line Tools**:
    - Command-line utilities like `find`, `grep`, and `strings` can help you search through files for the flag format (`picoCTF{...}`).

Conclusion :

By meticulously analyzing the extracted files and using various forensic tools, you can uncover the hidden flag(s) in the *BigZip* challenge. The experience gained from solving such challenges is invaluable in cybersecurity and digital forensics, emphasizing the need for attention to detail and creative problem-solving techniques.

14 inspect html

Challenge Overview

- **Challenge Name:** Inspect HTML
- **Category:** Web Exploitation
- **Points:** 100
- **Objective:** Analyze the provided HTML source code to find hidden flags or clues that may be embedded within the document structure or comments.

Report on Inspect HTML

The **Inspect HTML** challenge is designed to test participants' skills in analyzing HTML and identifying hidden information through the browser's inspection tools. This type of challenge emphasizes the importance of understanding web page structure and the ways in which information can be obscured.

Steps to Solve:

1. **Access the Webpage**:
   - Start by navigating to the challenge webpage where the HTML source code is located.
2. **Inspect the Source Code**:
   - Use browser developer tools (right-click the page and select "Inspect" or press F12) to view the HTML structure of the page.
   - Look for comments, hidden elements (like those with `display: none` in CSS), or any unusual attributes that might suggest hidden information.
3. **Check for Comments**:
   - HTML comments can often contain hints or even the flag itself. Look for any `<!-- comment -->` sections within the HTML code.
4. **Analyze Links and Scripts**:
   - Examine any JavaScript files linked to the page. Often, flags may be dynamically generated or stored within JavaScript.
   - Check for any additional resources loaded by the page that might contain further clues.
5. **Search for Flag Formats**:
   - Use the search functionality (Ctrl+F) in the developer tools to look for patterns resembling the flag format, typically `picoCTF{...}`.
6. **Review Console Outputs**:
   - Sometimes, flags can be output in the JavaScript console. After interacting with the page, check the console for any logged messages.

Conclusion

By meticulously inspecting the HTML and associated resources, participants can uncover hidden flags within the **Inspect HTML** challenge. This exercise enhances skills in web exploitation and reinforces the necessity of careful web application development practices to prevent exposing sensitive information.

15 first find

Challenge Overview

- **Challenge Name:** First Find

- **Category:** General Skills
- **Points:** 100
- **Objective:** Use various tools and techniques to locate a hidden flag within a given set of data, often requiring participants to demonstrate their skills in searching and identifying relevant information.

Report on First Find

The *First Find* challenge typically involves a scenario where participants must sift through a collection of data to uncover a hidden flag. This often involves a combination of skills in data analysis, searching, and sometimes decoding or extraction techniques.

Steps to Solve:

1. **Access the Provided Data**:
    - Start by downloading or accessing the dataset provided by the challenge, which may include files in various formats (e.g., text, images, or binary).
2. **Review the Data**:
    - Depending on the data type, open the files using appropriate tools:
        - **Text Files**: Use a text editor or command-line tools to read and search through the content.
        - **Images**: Use image viewers and steganography tools to check for hidden messages or flags.
        - **Archives**: If the data is compressed (like `.zip` or `.tar`), extract the contents and analyze the files within.
3. **Search for Flags**:
    - Use search functions within text editors or command-line utilities (like `grep`) to look for the flag format, typically `picoCTF{...}`.
    - If applicable, analyze metadata or hidden data in files for potential flags.
4. **Use Online Resources**:
    - Familiarize yourself with common encoding or encryption methods that may have been used to obscure the flag, and use online tools for decoding.
5. **Document Findings**:
    - Keep track of any interesting findings or potential flags as you work through the data.

Conclusion

By carefully following these steps and employing a systematic approach to data examination, participants can successfully uncover the hidden flag in the *First Find* challenge. This exercise helps develop essential skills in data forensics and analysis, which are vital in the field of cybersecurity.

16 includes

Challenge Name: Includes
Category: Web Exploitation
Points: 100
Objective: Identify and exploit a web application vulnerability related to the inclusion of external files to retrieve a hidden flag.

Detailed Report

Overview:

The *Includes* challenge is designed to test participants' skills in web exploitation, specifically in understanding how web applications handle file inclusions. The challenge typically involves identifying insecure file inclusion vulnerabilities that can be exploited to access restricted information.

Steps to Solve:

1. **Access the Web Application**:
   - Navigate to the provided URL for the web application. The challenge usually presents a form or a way to input data that affects file inclusion.
2. **Inspect the Input Fields**:
   - Analyze any input fields for parameters that suggest file inclusion. Look for common terms like `file`, `include`, or similar.
3. **Manipulate Input**:
   - Test various payloads in the input field. Common techniques include:
     - **Directory Traversal**: Use `../` sequences to attempt to navigate to parent directories and access files outside the intended directory.
     - **File Inclusion**: Try injecting file names that are typically sensitive, such as `/etc/passwd` (on Linux systems) or web configuration files.
4. **Review Server Responses**:
   - Observe how the server responds to different inputs. If the file inclusion is successful, you may see contents of the file displayed on the page.
   - Look for any indication of a flag, typically formatted as `picoCTF{...}`.
5. **Extract the Flag**:
   - Once you successfully include a file containing the flag, capture the flag for submission.

Conclusion :

File Inclusion Vulnerabilities: This challenge emphasizes the risks associated with improperly implemented file inclusion mechanisms in web applications.Directory Traversal Attacks: Understanding how to exploit directory traversal is critical for identifying and accessing sensitive files.Web Application Security: Participants learn about the importance of securing web applications against such vulnerabilities.

16 runme.py

Challenge Name: RunMe.py
Category: General Skills / Programming
Points: 100
Objective: Analyze and execute a provided Python script to uncover a hidden flag.

Report

Overview:

The *RunMe.py* challenge involves analyzing a Python script that may contain logic to reveal a hidden flag. Participants must understand the code and its execution flow, identifying how to run it correctly and interpret its output.

Steps to Solve:

1. **Download the Script**:
    - Begin by downloading the `RunMe.py` file provided in the challenge description.
2. **Inspect the Code**:
    - Open the script in a text editor or IDE to review the code. Look for any functions, loops, and conditional statements that dictate the script's behavior.
    - Pay special attention to comments and print statements that may hint at the flag's location or conditions for revealing it.
3. **Understand the Logic**:
    - Analyze the flow of the script. Identify any inputs it requires and how they might affect the output.
    - If the script has any obfuscation (like encoding or complex logic), consider how you can simplify or reverse it to uncover the hidden flag.
4. **Run the Script**:
    - Execute the script in a Python environment (e.g., terminal or an IDE). Make sure to use the appropriate version of Python specified in the challenge (usually Python 3).
    - If prompted for input, provide the necessary values based on your analysis of the code.
5. **Capture the Output**:

    - Look for output from the script that matches the flag format (`picoCTF{...}`). If the output isn't immediately clear, check if the script writes to a file or generates logs that may contain the flag.
6. **Debugging**:
    - If the flag isn't easily found, consider adding print statements or using a debugger to step through the code, examining variable values and control flow in real-time.

Conclusion

Python Programming: This challenge helps participants practice their Python skills, focusing on understanding scripts and flow control. Code Analysis: Analyzing scripts for logic and potential vulnerabilities is a valuable skill in cybersecurity.Execution Environment: Understanding how to set up and run Python scripts in a development environment is essential for this challenge.

17 fixme.py

Challenge Name: FixMe1.py
Category: General Skills / Programming
Points: 100
Objective: Analyze a provided Python script, identify errors or vulnerabilities, and modify the code to retrieve a hidden flag.

Detailed Report

Overview:

The *FixMe1.py* challenge requires participants to work with a Python script that is intentionally flawed or contains bugs. The objective is to identify and fix these issues in order to successfully run the script and uncover a hidden flag.

Steps to Solve:

1. **Download the Script**:
    - Start by downloading the `FixMe1.py` file from the challenge description.
2. **Inspect the Code**:
    - Open the script in a text editor or IDE to review its content. Look for common programming errors, such as syntax errors, logic flaws, or undefined variables.
3. **Identify Issues**:
    - Carefully read through the code to identify where it fails to execute as intended. Common issues may include:
        - Incorrect variable names
        - Misplaced indentation (affecting block structures)
        - Logic errors that prevent the expected output

    - Pay attention to any comments in the code that may hint at what needs to be fixed.
4. **Debug the Script**:
    - Run the script to see any error messages that Python outputs. This can help pinpoint issues directly.
    - Modify the code to fix the identified errors. If the script uses functions or classes, ensure they are called correctly.
5. **Execute the Fixed Script**:
    - Once modifications are made, run the script again. If the code is correct, it should produce output that includes the hidden flag in the format `picoCTF{...}`.
6. **Test Thoroughly**:
    - If the flag is not displayed, recheck your fixes and ensure that all logic flows correctly. Consider edge cases that may not have been initially handled.

Conclusion :

Python Programming: This challenge emphasizes understanding and debugging Python code, enhancing participants' coding skills. Problem-Solving: Identifying and fixing errors requires analytical thinking and systematic problem-solving abilities. Code Review Techniques: Participants learn how to review code effectively for common mistakes and vulnerabilities.

18 fixme2.py

challenge Name: FixMe2.py
Category: General Skills / Programming
Points: 100
Objective: Analyze and correct a provided Python script to retrieve a hidden flag.

Detailed Report

Overview:

The *FixMe2.py* challenge involves working with a Python script that contains intentional errors or vulnerabilities. Participants must identify and correct these issues to successfully execute the script and obtain a hidden flag.

Steps to Solve:

1. **Download the Script**:
    - Begin by downloading the `FixMe2.py` file provided in the challenge description.
2. **Inspect the Code**:
    - Open the script in a code editor or IDE. Look for syntax errors, logic errors, or any areas where the code does not function as intended.

3. **Identify Issues**:
   - Carefully examine the script to find common issues, such as:
     - Incorrect variable assignments or references
     - Function calls that may not work due to improper definitions
     - Logical errors that prevent the correct output
4. **Debug the Script**:
   - Attempt to run the script. Note any error messages generated, as they can guide you toward fixing the problems.
   - Make corrections to the code, ensuring proper syntax and logic.
5. **Run the Fixed Script**:
   - After applying your fixes, execute the script again. If everything is correct, the output should include a flag formatted as `picoCTF{...}`.
6. **Thorough Testing**:
   - If the flag is not shown, revisit your changes and ensure the script's logic flows correctly. Test various inputs or scenarios that may not have been accounted for.

Conclusion :

Python Debugging: Participants enhance their debugging skills by identifying and correcting errors in a Python script. Critical Thinking: The challenge fosters analytical thinking and problem-solving capabilities as participants determine how to resolve issues.Understanding Python Fundamentals: Familiarity with Python syntax and programming concepts is crucial for completing this challenge.

19 convertme.py

Challenge Name: ConvertMe.py
Category: General Skills / Programming
Points: 100
Objective: Analyze and modify a provided Python script to convert data from one format to another, ultimately revealing a hidden flag.

Detailed Report

Overview:

The *ConvertMe.py* challenge typically involves a Python script that is designed to convert data between formats (e.g., text to binary, decimal to hexadecimal). Participants must understand the conversion logic and modify the script as needed to correctly retrieve a hidden flag.

Steps to Solve:

1. **Download the Script**:

    - Start by downloading the `ConvertMe.py` file from the challenge description.
2. **Inspect the Code**:
    - Open the script in a code editor or IDE. Review the code to understand its structure and functionality. Look for functions responsible for data conversion.
3. **Identify Issues**:
    - Determine if there are any errors or incomplete conversion functions. Common issues may include:
        - Incorrect algorithms for conversion
        - Missing return statements
        - Hardcoded values that do not adjust based on input
4. **Modify the Code**:
    - If the code does not work as expected, make necessary adjustments. This may involve:
        - Correcting logical errors
        - Implementing missing functionality
        - Adding print statements to debug and verify output at various stages
5. **Run the Script**:
    - Execute the modified script in a Python environment. Provide any necessary input as specified in the challenge instructions.
6. **Check the Output**:
    - After running the script, look for output that matches the flag format, typically `picoCTF{...}`. If the flag isn't immediately visible, review the conversion logic and ensure all inputs are correctly processed.
7. **Test Thoroughly**:
    - If the flag is not displayed, recheck your modifications and run additional test cases to ensure all potential edge cases are handled.

Conclusion

Data Conversion: This challenge reinforces understanding of data types and conversion methods in Python. Programming Debugging: Participants enhance their debugging skills by identifying and correcting issues in code.Critical Thinking: Analyzing the conversion logic and understanding how to manipulate data effectively fosters problem-solving abilities.

20 codebook

Challenge Name: Codebook
Category: Cryptography
Points: 100
Objective: Use a provided codebook or cipher key to decrypt a hidden message and retrieve the flag.

Detailed Report

Overview:

The *Codebook* challenge typically involves deciphering a message that has been encoded using a specific codebook or cipher. Participants must apply their knowledge of cryptography to decode the message and find the hidden flag.

Steps to Solve:

1. **Download the Codebook**:
   ○ Begin by downloading any files provided in the challenge description, including the codebook and the encoded message.
2. **Inspect the Codebook**:
   ○ Open the codebook file to understand the encoding scheme used. This might include mappings of characters to symbols, words to numbers, or other forms of encoding.
3. **Analyze the Encoded Message**:
   ○ Review the encoded message provided in the challenge. Determine its structure and how it corresponds to the codebook.
4. **Decrypt the Message**:
   ○ Using the codebook, systematically replace the encoded symbols or characters with their corresponding values from the codebook.
   ○ If the encoding uses a more complex scheme (e.g., a polyalphabetic cipher), apply the appropriate decryption technique.
5. **Check for the Flag Format**:
   ○ As you decode the message, look for the flag format, typically `picoCTF{...}`. If it doesn't appear immediately, ensure all parts of the message have been decoded.
6. **Verify the Output**:
   ○ Once you believe you have the flag, verify its correctness by submitting it through the PicoCTF platform.

Conclusion :

Cryptography: This challenge emphasizes understanding various encoding and decoding techniques, enhancing participants' skills in cryptography. Attention to Detail: Participants learn the importance of meticulous attention when working with encoded messages, as minor errors can lead to incorrect decryption. Analytical Skills: Decrypting messages requires critical thinking and problem-solving skills, as participants must understand the underlying principles of the encoding used.

21 information
Challenge Name: Information
Category: Forensics
Points: 100
Objective: Analyze digital artifacts or files to uncover hidden information or recover a flag.

Report

Overview:

The *Information* challenge in the forensics category typically involves investigating digital files, such as images, documents, or system artifacts. Participants must apply forensic analysis techniques to extract hidden data and reveal the flag.

Steps to Solve:

1. **Download the Provided Files**:
   - Begin by downloading any files or artifacts provided in the challenge description.
2. **Initial File Analysis**:
   - Use tools such as `file` to determine the file type and gather initial metadata. This can help identify how to proceed with further analysis.
   - For images, tools like `exiftool` can extract metadata that may contain hidden information.
3. **Data Recovery Techniques**:
   - If the challenge involves recovering deleted files or data, use forensic tools like `photorec` or `foremost` to scan for recoverable information.
   - Analyze the structure of files (e.g., JPEG, PNG) to identify hidden messages or flags that may be encoded in non-visible areas.
4. **Inspect Hidden Data**:
   - Look for steganographic content in images or audio files using tools such as `steghide` or `zsteg`.
   - Analyze text files for unusual formatting, comments, or invisible characters that could hint at a flag.
5. **Use Command-Line Tools**:
   - Command-line tools can help automate the analysis process. For example, `grep` can search for flag patterns across files.
   - Utilize scripting languages (like Python) for more complex data extraction or analysis.
6. **Document Findings**:
   - As you uncover information, keep detailed notes on your findings and the methods used. This is important for verifying the accuracy of your results.
   - Ensure that any flags identified conform to the expected format (e.g., `picoCTF{...}`).

Conclusion :

Digital Forensics: This challenge emphasizes the principles of digital forensics, including data recovery and analysis techniques. Metadata Analysis: Understanding how to extract and interpret metadata is crucial for uncovering hidden information. Steganography: Participants learn about methods for hiding data within other files and how to detect such practices.

22 mod 26

Challenge Name: Mod 26
Category: Cryptography
Points: 100
Objective: Utilize modular arithmetic (specifically mod 26) to decode or analyze an encoded message to retrieve a hidden flag.

Detailed Report

Overview:

The *Mod 26* challenge typically involves encoding or decoding a message using modular arithmetic with a focus on the 26 letters of the alphabet. Participants must apply their understanding of modular operations to reveal the hidden flag.

Steps to Solve:

1. **Understand the Encoding Method**:
   - The challenge likely uses a form of encoding that relies on shifting letters or applying modular arithmetic (e.g., Caesar cipher).
   - Familiarize yourself with how mod 26 works, where each letter corresponds to a number (A=0, B=1, ..., Z=25).
2. **Analyze the Provided Data**:
   - Access the encoded message or text provided in the challenge. Identify if there are numerical values or encoded characters that require decoding.
3. **Apply Modular Arithmetic**:
   - If the challenge involves shifting letters (e.g., shifting by a fixed number), apply the formula: New Position=(Current Position+Shift)mod 26\text{New Position} = (\text{Current Position} + \text{Shift}) \mod 26New Position=(Current Position+Shift)mod26
   - For decoding, reverse the shift: New Position=(Current Position−Shift)mod 26\text{New Position} = (\text{Current Position} - \text{Shift}) \mod 26New Position=(Current Position−Shift)mod26
4. **Convert Letters to Numbers and Back**:
   - Convert letters to their corresponding numerical values, apply the mod operation, and then convert back to letters.
   - For example, if a letter is encoded as 'C' (2) and shifted by 3, it becomes 'F' (5). To decode, shift back to find the original letter.
5. **Look for the Flag**:
   - As you decode, watch for the flag format, typically `picoCTF{...}`. If the flag doesn't appear, ensure you have the correct shift or encoding method.
6. **Verify and Document Your Findings**:
   - Once you identify the flag, verify it by submitting it on the PicoCTF platform.

- Document your decoding process, including any calculations or shifts applied, to provide a clear rationale for your solution.

Conclusion :

Modular Arithmetic: Understanding the principles of mod 26 is crucial for solving this challenge, especially as it applies to the alphabet. Cipher Techniques: Familiarity with common ciphers (like Caesar or substitution ciphers) enhances participants' cryptography skills. Analytical Thinking: This challenge emphasizes the importance of logical reasoning and problem-solving abilities in decoding messages.

23 nice netcat

Challenge Name: Nice Netcat
Category: Networking / General Skills
Points: 100
Objective: Utilize the `netcat` utility to connect to a remote service and interact with it to retrieve a hidden flag.

Detailed Report

Overview:

The *Nice Netcat* challenge tests participants' ability to use the `netcat` (often abbreviated as `nc`) tool to establish network connections and interact with remote services. This challenge emphasizes practical skills in networking and understanding of how to communicate with services over TCP/UDP.

Steps to Solve:

1. **Understand the Challenge Requirements**:
   - Begin by carefully reading the challenge description to identify the host address and port number provided for the connection.
2. **Open the Terminal**:
   - Launch a terminal or command prompt to execute `netcat` commands.
3. **Establish a Connection**:
   - Use the following command to connect to the specified host and port
     i. Replace `<host>` and `<port>` with the actual values given in the challenge.
   - **Interact with the Service**:
     i. Once connected, you may see a prompt or a message. Pay attention to the instructions provided by the service.

        ii. Follow any prompts or input requirements specified in the challenge. This could involve sending specific commands or responding to queries.
    - **Analyze Responses**:
        i. Monitor the output returned by the service. Often, hidden flags are embedded in the service's responses.
        ii. If the output is unclear or does not immediately reveal the flag, try different inputs or commands based on the context of the challenge.
    - **Capture the Flag**:
        i. As you interact with the service, look for the flag formatted as `picoCTF{...}`. Document this flag for submission.
    - **Verify Your Solution**:
        i. Submit the flag on the PicoCTF platform to verify its correctness.

Conclusion :

Netcat Usage: This challenge enhances participants' familiarity with `netcat`, a versatile tool for network communication and troubleshooting. Networking Principles: Understanding basic networking concepts, such as TCP/UDP and how services communicate over the internet, is crucial for successfully completing the challenge. Problem-Solving Skills: Interacting with remote services requires analytical thinking and adaptability, as participants must respond to dynamic situations based on service feedback.

24 obedient cat

Challenge Name: Obedient Cat
Category: General Skills / Networking
Points: 100
Objective: Interact with a remote service using a provided program or script to uncover a hidden flag.

Detailed Report

Overview:

The *Obedient Cat* challenge typically involves using a client-side program or script to connect to a remote server. Participants must understand how to send and receive data, often following specific instructions to obtain the hidden flag.

Steps to Solve:

1. **Review the Challenge Description**:
    - Begin by carefully reading the challenge prompt. Take note of the provided details, including any code snippets, commands, or instructions.

2. **Download the Provided Script**:
    - If a script or program is provided, download it and open it in a text editor to understand its structure and functionality. Look for functions that handle network communication.
3. **Set Up the Connection**:
    - Follow the instructions to set up the connection to the remote server. This may involve using tools like `netcat` or running the provided script directly.
4. **Analyze the Communication**:
    - Pay attention to how the script interacts with the server. Look for input prompts, expected responses, and any patterns that may hint at how to retrieve the flag.
    - If the script includes commands or functions that require user input, be prepared to provide the correct responses based on the context of the challenge.
5. **Interact with the Server**:
    - Run the script or use `netcat` to connect to the specified host and port. Engage with the server according to the challenge's requirements.
    - Be observant of the server's responses. Flags are often revealed through specific interactions or after completing certain tasks.
6. **Capture the Flag**:
    - As you interact with the service, look for the flag formatted as `picoCTF{...}`. Once identified, document the flag for submission.
7. **Verify Your Solution**:
    - Submit the captured flag on the PicoCTF platform to ensure its accuracy.

conclusion

Networking and Client-Server Communication: This challenge reinforces understanding of how client applications interact with servers over networks. Script Analysis: Participants improve their ability to read and understand code, especially scripts that handle network operations. Problem-Solving and Logic: Successfully retrieving the flag often requires logical thinking and adaptability in responding to server prompts.

25 python wrangling

Challenge Name: Python Wrangling
Category: Programming / Scripting
Points: 100
Objective: Analyze and manipulate Python data structures to uncover a hidden flag.

Detailed Report

Overview:

The *Python Wrangling* challenge involves working with Python code and data structures. Participants are required to modify or extract information from the provided script or data, demonstrating their proficiency in Python programming and data manipulation.

Steps to Solve:

1. **Download the Provided Files**:
    - Begin by downloading any files or scripts provided in the challenge description.
2. **Inspect the Code/Data**:
    - Open the Python script or data file in a code editor. Review its content to understand its structure and the purpose of the code.
    - Identify any variables, functions, or data structures that are key to solving the challenge.
3. **Identify Key Operations**:
    - Look for operations that involve data manipulation, such as loops, conditionals, or list comprehensions.
    - Pay attention to any functions that may need modification or any data that requires extraction.
4. **Modify the Code**:
    - Make the necessary changes to the script to manipulate the data as required by the challenge. This may involve:
        - Adding print statements to debug and visualize outputs
        - Modifying existing functions to change their behavior
        - Implementing new logic to extract hidden information or flags
5. **Run the Script**:
    - Execute the modified script in a Python environment. Observe the output closely, as the flag may be presented in the console or as part of the data structure.
6. **Look for the Flag**:
    - As you analyze the output, search for the flag formatted as `picoCTF{...}`. If it doesn't appear initially, recheck your code modifications and ensure all data is being processed correctly.
7. **Document Findings**:
    - Keep notes of the changes made to the code and any relevant outputs that lead to finding the flag.

Conclusion:

The *Python Wrangling* challenge effectively tests participants' skills in Python programming and data manipulation. By analyzing and modifying a provided script, participants reinforce their understanding of Python data structures and coding practices. Successfully completing this challenge requires a blend of critical thinking, problem-solving skills, and familiarity with Python syntax.

26 wave a flag

Challenge Name: Wave a Flag
Category: Cryptography / Steganography
Points: 100
Objective: Analyze a provided file to uncover a hidden flag, typically by applying steganographic techniques or cryptographic methods.

Detailed Report

Overview:

The *Wave a Flag* challenge involves uncovering hidden information embedded within a file, often utilizing steganography or other cryptographic techniques. Participants must employ various analysis methods to extract the flag.

Steps to Solve:

1. **Download the Provided File**:
    - Start by downloading the file associated with the challenge, which may be an image, audio file, or other data formats.
2. **Initial File Analysis**:
    - Determine the file type using tools such as the `file` command in a terminal. Understanding the file format will guide your approach.
    - For images, consider using tools like `exiftool` to check for any hidden metadata or comments.
3. **Inspect for Hidden Data**:
    - If the file is an image, explore potential steganography tools like `steghide`, `zsteg`, or `binwalk` to look for hidden data.
    - For audio files, use tools such as `Audacity` to visualize waveforms or spectrograms, which may reveal hidden messages.
4. **Extract and Analyze**:
    - Use the appropriate tools to extract any hidden content. For example, if you uncover a hidden text file within an image, save and open it for further analysis.
    - Analyze the extracted data for patterns or clues that might lead to the flag.
5. **Look for the Flag**:
    - As you examine the hidden data, search for the flag formatted as `picoCTF{...}`. If it does not appear immediately, consider various decoding methods, such as base64 or hexadecimal conversions.
6. **Verify Your Findings**:
    - Once you identify the flag, document it clearly and submit it on the PicoCTF platform for verification.

Conclusion:

The *Wave a Flag* challenge serves as an excellent exercise in steganography and cryptographic techniques, requiring participants to think critically and utilize a range of tools. Successfully completing this challenge enhances participants' skills in analyzing and uncovering hidden information, a crucial aspect of cybersecurity.

27 tab tab attack

Challenge Name: Tab Tab Attack
Category: General Skills
Points: 100
Objective: Exploit a vulnerability in a web application by manipulating user input through tabbing techniques to retrieve a hidden flag.

Detailed Report

Overview:

The *Tab Tab Attack* challenge focuses on exploiting a web application vulnerability, specifically through input manipulation techniques that involve using the tab key to change focus between input fields. Participants must demonstrate an understanding of web exploitation and user input handling to uncover a hidden flag.

Steps to Solve:

1. **Access the Web Application**:
   - Begin by navigating to the challenge URL provided in the PicoCTF platform.
2. **Inspect the Application**:
   - Use the browser's developer tools (accessible via F12) to inspect the structure of the web application, focusing on input fields, forms, and their associated JavaScript functions.
   - Look for any comments or elements that might indicate vulnerabilities related to user input handling.
3. **Analyze Input Fields**:
   - Identify any input fields in the web application where user data can be submitted. Pay attention to how the application handles focus transitions when using the tab key.
   - Experiment with tabbing between fields to observe how the application responds. This might reveal any underlying vulnerabilities or unexpected behavior.
4. **Perform the Tab Tab Attack**:
   - Utilize the tab key to navigate through the input fields and attempt to input payloads or manipulate data in a way that the application does not expect.
   - This may involve entering data into a field that's not currently focused or using special characters to test the application's response.
5. **Look for the Flag**:

    - As you manipulate inputs, observe the application’s output for any flags, typically formatted as `picoCTF{...}`. This might appear as a result of successful exploitation or manipulation.
    - If the flag is not immediately visible, consider alternative tabbing techniques or input combinations to uncover it.
6. **Document Findings and Submit the Flag**:
    - Once the flag is identified, take note of it for submission. Ensure that the format is correct for the PicoCTF platform.
    - Submit the flag and document any insights or techniques used during the process.

Conclusion:

The *Tab Tab Attack* challenge highlights essential concepts in web security, particularly related to user input handling and exploitation techniques. Participants enhance their understanding of how seemingly simple interactions, such as tabbing through fields, can lead to the discovery of vulnerabilities in web applications.

28 cookies

Challenge Name: Cookies
Category: Web Exploitation
Points: 100
Objective: Analyze and manipulate HTTP cookies to retrieve a hidden flag from a web application.

Detailed Report

Overview:

The *Cookies* challenge involves understanding how cookies are used in web applications and exploiting potential vulnerabilities associated with them. Participants must manipulate cookie values or analyze how the application handles cookies to uncover a hidden flag.

Steps to Solve:

1. **Access the Web Application**:
    - Start by navigating to the URL provided in the challenge description.
2. **Inspect Cookies**:
    - Use browser developer tools (F12) to access the "Application" or "Storage" tab, where you can view cookies set by the web application.
    - Analyze the names and values of the cookies. Look for any cookies that may seem suspicious or contain information that could lead to the flag.

3. **Understand Cookie Functionality**:
    - Determine how the application uses the cookies. Some common uses include session management, user preferences, or storing user-related data.
    - Identify if the application verifies cookie values on the server side and if they can be manipulated.
4. **Modify Cookie Values**:
    - Experiment with changing cookie values to see how the application responds. For instance, if you find a cookie related to user authentication or privileges, attempt to modify it to gain elevated access.
    - Use the developer tools to edit cookie values directly and observe any changes in the application's behavior.
5. **Analyze Application Responses**:
    - After modifying cookie values, refresh the page or trigger actions that rely on those cookies to see if the flag is revealed in the response.
    - Pay attention to any output or changes in behavior that indicate successful exploitation.
6. **Look for the Flag**:
    - As you manipulate cookies and interact with the application, search for the flag formatted as `picoCTF{...}`. This could appear in the application's output or as part of a redirect.
7. **Document and Verify Your Findings**:
    - Once the flag is identified, document it and ensure that the format is correct before submitting it on the PicoCTF platform.

Conclusion:

The *Cookies* challenge underscores the significance of understanding web security, particularly the role cookies play in session management and user data storage. By engaging in this challenge, participants sharpen their skills in web exploitation and gain insights into how cookie manipulation can lead to vulnerabilities.

29 scavenger hunt

Challenge Name: Scavenger Hunt
Category: General Skills
Points: 100
Objective: Search for hidden flags throughout a web application or provided environment by exploring various elements and pages.

Detailed Report

Overview:

The *Scavenger Hunt* challenge is designed to encourage participants to explore a web application or a series of resources to uncover hidden flags. It tests skills in navigation, observation, and problem-solving within a given context.

Steps to Solve:

1. **Access the Challenge Environment**:
    - Start by navigating to the URL or downloading any resources provided in the challenge description.
2. **Explore the Application or Resources**:
    - Systematically browse through the various pages or elements of the web application. Pay attention to links, buttons, and navigation menus that might lead to hidden areas or content.
    - If there are files or folders available for download, inspect their contents carefully.
3. **Look for Hidden Elements**:
    - Use browser developer tools (F12) to inspect the HTML and CSS of the web application. Look for comments, hidden fields, or other elements that may not be immediately visible.
    - Check the network activity in the developer tools to see if any additional data is loaded dynamically when interacting with the application.
4. **Test for Flags**:
    - Flags are often embedded within the content of the application or displayed after completing specific actions. Look for patterns in text or unusual messages that may indicate a flag.
    - Try different interactions, such as clicking on unusual links or entering random inputs in fields, to trigger responses that might reveal hidden information.
5. **Document Findings**:
    - As you discover potential flags, document their formats and the context in which they were found. Ensure they adhere to the typical `picoCTF{...}` format.
6. **Submit Flags**:
    - Once you have collected all the flags, submit them on the PicoCTF platform to verify their correctness.

Conclusion:

The *Scavenger Hunt* challenge encourages participants to develop their investigative skills and familiarize themselves with web application structures. It fosters an understanding of how information can be hidden and the importance of thorough exploration in cybersecurity contexts.

30 strings it

Challenge Name: Strings It
Category: Forensics
Points: 100
Objective: Analyze a binary file to extract human-readable strings and identify a hidden flag.

Detailed Report

Overview:

The *Strings It* challenge focuses on the analysis of binary files, requiring participants to utilize the `strings` command or similar tools to extract readable text from a file. The challenge tests skills in binary analysis and forensics, emphasizing the importance of identifying useful information hidden within non-text files.

Steps to Solve:

1. **Download the Provided Binary File**:
   - Begin by downloading the binary file associated with the challenge from the PicoCTF platform.
2. **Use the Strings Command**:
   - Open a terminal and navigate to the directory containing the downloaded binary file.
   - Run the `strings` command on the binary to extract human-readable strings
   - Replace `<filename>` with the actual name of the binary file.

3. Analyze the Output:

- Review the list of strings extracted from the binary. Look for patterns, common phrases, or any text that resembles the format of a PicoCTF flag (e.g., `picoCTF{...}`).
- Pay attention to longer strings or those that appear unusual, as they may contain important information.

4 **Search for the Flag**:

- As you sift through the output, identify any strings that could potentially be the flag. Flags may be mixed in with other output, so take your time to search thoroughly.
- If no flags are immediately apparent, consider using additional options with the `strings` command, such as adjusting the minimum string length to filter results

5 Document and Verify Your Findings:

- Once the flag is identified, document it and ensure it follows the correct format for submission.
- Submit the flag on the PicoCTF platform to verify its accuracy.

Conclusion:

The *Strings It* challenge effectively demonstrates the utility of basic forensic analysis techniques in cybersecurity. By engaging with binary files and extracting human-readable content, participants enhance their skills in digital forensics and binary analysis.

31 dont use client side

Challenge Name: Don't Use Client Side
Category: Web Exploitation
Points: 100
Objective: Exploit a web application vulnerability by analyzing and manipulating server-side logic to uncover a hidden flag.

Detailed Report

Overview:

The *Don't Use Client Side* challenge focuses on exploiting server-side vulnerabilities in a web application. Participants are tasked with understanding how the application processes requests and responses, particularly emphasizing the importance of server-side validation and logic.

Steps to Solve:

1. **Access the Web Application**:
    - Begin by navigating to the URL provided for the challenge.
2. **Inspect the Application**:
    - Use browser developer tools (accessible via F12) to explore the web application. Check the network tab to observe HTTP requests and responses.
    - Look for forms or input fields that could lead to vulnerabilities, such as those that interact with server-side logic.
3. **Analyze Request Handling**:
    - Identify how the application processes user inputs. Check for any parameters sent to the server that might be manipulated.
    - Look for clues in the application's responses that indicate how it handles input validation or user authentication.
4. **Manipulate Inputs**:
    - Experiment with modifying input values in requests. This could involve changing parameters, injecting unexpected data, or attempting common web exploitation techniques such as SQL injection or command injection.
    - Pay attention to how the server responds to different inputs, especially if it reveals any sensitive information or errors.
5. **Capture the Flag**:

    - As you manipulate inputs and interact with the application, look for a flag formatted as `picoCTF{...}` in the responses. It may appear as part of a message or data returned by the server.
    - If the flag is not visible initially, continue testing various input manipulations until you uncover it.
6. **Document Findings**:
    - Once the flag is identified, ensure it is documented correctly and follows the required format.
    - Submit the flag through the PicoCTF platform for verification.

Conclusion:

The *Don't Use Client Side* challenge emphasizes the significance of understanding server-side vulnerabilities and the importance of robust validation mechanisms. Participants enhance their skills in web exploitation by learning how to identify and manipulate inputs effectively.

32 warmed up

Challenge Name: Warmed Up
Category: General Skills
Points: 100
Objective: Analyze a given resource or file to retrieve a hidden flag, often requiring an understanding of basic principles related to cryptography or encoding.

Detailed Report

Overview:

The *Warmed Up* challenge is designed to familiarize participants with fundamental techniques used in cryptography and data encoding. It typically involves analyzing a file or data set to uncover a hidden flag by applying basic decoding or decryption methods.

Steps to Solve:

1. **Access the Challenge Resource**:
    - Begin by downloading the file or accessing the resource provided in the challenge description on the PicoCTF platform.
2. **Examine the Resource**:
    - Determine the type of file you are dealing with (e.g., text file, image, binary). Use commands like `file <filename>` in the terminal to identify the file type.
    - Open the file using an appropriate viewer or editor based on its format (e.g., text editor for text files, hex editor for binary files).
3. **Look for Patterns**:

    - Analyze the contents for any recognizable patterns, strings, or formatting that may indicate the presence of a hidden flag.
    - If the file appears encoded or encrypted, note the characteristics of the data, such as character sets or repeating sequences.
4. **Apply Decoding Techniques**:
    - Use common decoding techniques such as Base64, hexadecimal conversion, or simple ciphers (e.g., Caesar cipher) to transform the data into a readable format.
    - Utilize online tools or command-line utilities for decoding or decrypting, if necessary.
5. **Identify the Flag**:
    - As you decode or transform the data, search for the flag formatted as `picoCTF{...}`. This may emerge during the decoding process or after applying certain techniques.
    - If the flag is not immediately apparent, continue testing various methods or approaches based on the data characteristics.
6. **Document and Submit the Flag**:
    - Once the flag is identified, document it carefully and ensure it adheres to the correct format for submission.
    - Submit the flag on the PicoCTF platform to confirm its validity.

Conclusion:

The *Warmed Up* challenge serves as an excellent introduction to basic principles of cryptography and data analysis. Participants gain valuable experience in identifying and utilizing various decoding techniques to uncover hidden information.

33 glory of the garden

Challenge Name: Glory of the Garden
Category: General Skills
Points: 100
Objective: Analyze a text file or other data resource to uncover hidden information, often requiring knowledge of encoding, cryptography, or other analytical skills.

Detailed Report

Overview:

The *Glory of the Garden* challenge tasks participants with examining a given file or resource to extract a hidden flag. This challenge emphasizes the application of various analytical techniques, including decoding and pattern recognition.

Steps to Solve:

1. **Access the Challenge Resource**:
    - Download the file provided in the challenge description on the PicoCTF platform.
2. **Examine the File**:
    - Open the file using an appropriate text editor or viewer. Determine the format of the file (e.g., plain text, CSV, etc.).
    - Look for any unusual patterns, text formatting, or encoded data within the file.
3. **Identify Patterns and Clues**:
    - As you inspect the contents, identify any sequences or strings that stand out. This may include repeated phrases, numbers, or characters that could suggest encoding or encryption.
    - Pay attention to any hints or references that might lead you toward the flag.
4. **Apply Analytical Techniques**:
    - If the data appears encoded, consider common encoding techniques like Base64, hexadecimal, or other ciphers (e.g., Caesar cipher).
    - Use online tools or scripts to decode or analyze the data if needed.
5. **Look for the Flag**:
    - As you decode or analyze the information, search for the flag formatted as `picoCTF{...}`. The flag may be revealed during the decoding process or might be hidden within the data.
    - If the flag is not readily visible, continue experimenting with different decoding methods or patterns.
6. **Document and Submit the Flag**:
    - Once you identify the flag, document it carefully and ensure it follows the correct format for submission.
    - Submit the flag through the PicoCTF platform for verification.

Conclusion:

The *Glory of the Garden* challenge encourages participants to sharpen their analytical skills and explore various methods of data interpretation. By engaging with different formats and encoding techniques, participants enhance their understanding of how information can be obscured and how to uncover it.

34 lets warm up

Challenge Name: Let's Warm Up
Category: General Skills
Points: 100
Objective: Familiarize yourself with basic techniques in digital forensics and cryptography by analyzing provided data to retrieve a hidden flag.

Detailed Report

Overview:

The *Let's Warm Up* challenge is designed to introduce participants to fundamental concepts in digital forensics and data analysis. The challenge typically involves examining a file or dataset to extract meaningful information, highlighting the importance of critical thinking and attention to detail in cybersecurity.

Steps to Solve:

1. **Access the Challenge Resource**:
    - Begin by downloading the file or accessing the resource linked in the challenge description on the PicoCTF platform.
2. **Examine the Contents**:
    - Open the provided file using an appropriate editor (e.g., text editor for text files or hex editor for binary files). Determine the file format and its contents.
    - Look for any recognizable text patterns, unusual formatting, or hints that could lead to the flag.
3. **Identify Relevant Techniques**:
    - Consider common techniques used in digital forensics, such as string extraction, decoding, and pattern recognition.
    - If the content appears encoded or obfuscated, note the encoding method used (e.g., Base64, hexadecimal, etc.).
4. **Apply Decoding and Analysis**:
    - Use command-line tools or online decoders to analyze the data.
        i. Look for any strings or messages that resemble the flag format (e.g., `picoCTF{...}`).

5 Search for the Flag:

i. As you process the data, carefully search for the flag. It may be hidden among other content or revealed as a result of decoding.

Ii. If necessary, iterate through different techniques or tools until you uncover the flag.

6 Document and Submit the Flag:

i. Once you successfully identify the flag, document it in the required format and prepare it for submission.

Ii. Submit the flag on the PicoCTF platform to validate your findings.

Conclusion:

The *Let's Warm Up* challenge serves as an essential primer for participants new to digital forensics and data analysis. By engaging with the challenge, participants develop foundational skills in examining and extracting information from various data types.

35 insp3ctor

Challenge Name: Insp3ctor
Category: Web Exploitation
Points: 100
Objective: Analyze and manipulate web application behavior by inspecting the HTML and JavaScript code to uncover a hidden flag.

Detailed Report

Overview:

The *Insp3ctor* challenge focuses on web exploitation techniques that involve understanding and manipulating the behavior of a web application. Participants are tasked with inspecting HTML and JavaScript code to uncover a hidden flag, emphasizing the importance of front-end code analysis in web security.

Steps to Solve:

1. **Access the Web Application**:
   - Start by navigating to the URL provided for the challenge in the PicoCTF platform.
2. **Inspect the HTML Source Code**:
   - Use browser developer tools (accessible via F12 or right-clicking and selecting "Inspect") to open the Elements panel.
   - Review the HTML structure for comments, hidden inputs, or elements that may contain useful information. Pay attention to `<script>` tags and other dynamic elements.
3. **Analyze JavaScript Code**:
   - If there are JavaScript files linked in the HTML, inspect their content in the Sources tab. Look for functions or variables that could be manipulating data or determining how the application responds to user input.
   - Search for any references to flags or strings that resemble flag formatting (e.g., `picoCTF{...}`).
4. **Look for Vulnerabilities**:
   - Check for common vulnerabilities such as insecure Direct Object References (IDOR), Cross-Site Scripting (XSS), or improperly handled user inputs.
   - Test various inputs in the application's forms to see how the application responds, noting any anomalies or unexpected behavior.
5. **Capture the Flag**:
   - If the flag is hidden within the code or displayed under certain conditions, manipulate the inputs or interactions to reveal it.

    - Be diligent in your examination of both the HTML and JavaScript, as the flag could be stored or generated dynamically based on user interactions.
6. **Document and Submit the Flag**:
    - Once the flag is identified, ensure it is documented accurately and follows the required format.
    - Submit the flag on the PicoCTF platform to confirm its validity.

Conclusion:

The *Insp3ctor* challenge emphasizes the critical role of front-end code analysis in web exploitation. By engaging with this challenge, participants gain valuable experience in inspecting and manipulating web applications to uncover hidden information.

36 logon

Challenge Name: Logon
Category: Web Exploitation
Points: 100
Objective: Exploit a web application vulnerability related to authentication mechanisms to retrieve a hidden flag.

Detailed Report

Overview:

The *Logon* challenge focuses on identifying and exploiting vulnerabilities within the authentication system of a web application. Participants are tasked with manipulating login credentials or session management to uncover a hidden flag, highlighting the significance of secure authentication practices in web development.

Steps to Solve:

1. **Access the Web Application**:
    - Start by navigating to the URL provided for the challenge on the PicoCTF platform.
2. **Examine the Login Page**:
    - Use browser developer tools (accessible via F12) to inspect the login form. Look for input fields for username and password, as well as any additional parameters or hidden fields.
    - Check the network tab while attempting to log in to observe the requests being made, particularly focusing on the payload sent to the server.
3. **Test Default Credentials**:

  - Attempt using common or default usernames and passwords (e.g., admin/admin, user/user) to see if the application allows access with known credentials.
4. **Look for Vulnerabilities**:
   - Test for vulnerabilities such as SQL injection by manipulating the input values in the username and password fields. For instance, using input like `admin' OR '1'='1` may allow bypassing authentication.
   - Check for other vulnerabilities such as session fixation, insecure direct object references, or lack of rate limiting.
5. **Analyze Server Responses**:
   - Carefully analyze any error messages or responses returned by the server upon failed login attempts. This can provide insights into how the authentication mechanism works and what inputs might be valid.
   - If the application provides any hints or flags after a successful login, take note of them.
6. **Capture the Flag**:
   - Once you have successfully logged in or bypassed the authentication, look for the flag, typically formatted as `picoCTF{...}`, in the application's interface or in hidden elements of the page.
   - If the flag is not immediately visible, explore the application further to find additional hints or data that may lead you to it.
7. **Document and Submit the Flag**:
   - After identifying the flag, ensure it is documented correctly and follows the required format for submission.
   - Submit the flag on the PicoCTF platform to validate your findings.

Conclusion:

The *Logon* challenge provides a practical introduction to web application vulnerabilities related to authentication. By engaging with this challenge, participants develop essential skills in identifying and exploiting weaknesses in web security mechanisms.

37 whats netcat

Challenge Name: Net Cat
Category: General Skills
Points: 100
Objective: Use the Netcat utility to connect to a remote service and retrieve a hidden flag.

Overview:

The *Net Cat* challenge involves utilizing the Netcat tool to establish a network connection with a provided hostname and port. Participants learn basic networking principles and how to interact with network services to uncover hidden information.

Steps to Solve:

1. **Connect to the Service**:
   - Use the command `nc [hostname] [port]` to connect, replacing `[hostname]` and `[port]` with the provided values.
2. **Interact with the Service**:
   - Follow any prompts and input commands as necessary to communicate with the service.
3. **Search for the Flag**:
   - Look for strings formatted like `picoCTF{...}` in the responses from the service.
4. **Document and Submit the Flag**:
   - Once identified, document the flag and submit it on the PicoCTF platform.

Conclusion:

This challenge reinforces the importance of networking tools like Netcat in cybersecurity, enhancing participants' skills in network communication and exploitation techniques.

38 unminify

Challenge Name: Unminify
Category: Web Exploitation
Points: 100
Objective: Analyze and unminify obfuscated code in a web application to retrieve a hidden flag.

Detailed Report

Overview:

The *Unminify* challenge is designed to introduce participants to the concept of code obfuscation and the techniques used to obscure JavaScript or HTML code in web applications. Participants are tasked with reversing this obfuscation to reveal the hidden flag.

Steps to Solve:

1. **Access the Challenge Resource**:
   - Start by navigating to the challenge URL provided on the PicoCTF platform. This will direct you to the web application containing the obfuscated code.
2. **Inspect the Source Code**:
   - Use browser developer tools (accessible via F12) to open the Elements or Sources panel.
   - Locate the minified or obfuscated JavaScript or HTML code within the `<script>` tags or linked JavaScript files.

3. **Unminify the Code**:
    - Copy the minified code and use an online unminifying tool (such as JS Beautifier for JavaScript) or your text editor with formatting features to unminify it.
    - This will help make the code more readable and structured.
4. **Analyze the Unminified Code**:
    - Review the unminified code for logic that may be manipulating data or generating output. Look for functions or variables that might reference the flag.
    - Search for any strings formatted like `picoCTF{...}` which could indicate the flag's location.
5. **Identify the Flag**:
    - As you analyze the code, look for specific lines or sections that output or reference the flag.
    - Pay attention to conditions or loops that may conceal the flag or require specific inputs to reveal it.
6. **Test Inputs**:
    - If the code includes user input handling, consider testing various inputs that might trigger the flag's output or behavior.
7. **Document and Submit the Flag**:
    - Once the flag is found, ensure it is correctly formatted for submission.
    - Submit the flag on the PicoCTF platform to confirm your solution.

Conclusion:

The *Unminify* challenge provides valuable experience in dealing with obfuscated code, highlighting the importance of understanding how web applications can obscure their logic. Participants enhance their skills in code analysis, which is essential for identifying vulnerabilities and exploiting them effectively.

39 first grep

Challenge Name: First Grep
Category: Forensics
Points: 100
Objective: Use the `grep` command to analyze text files and retrieve a hidden flag based on specific patterns or strings.

Detailed Report

Overview:

The *First Grep* challenge focuses on the use of the `grep` command, a powerful tool for searching plain-text data for lines that match a regular expression. Participants learn how to effectively utilize this command in a forensic context to find hidden information.

Steps to Solve:

1. **Access the Challenge Resource**:
   - Download or access the text file(s) provided in the challenge description on the PicoCTF platform.
2. **Open the Terminal**:
   - Launch a terminal window on your system to execute the `grep` command.
3. **Use `grep` to Search for the Flag**:
   - Begin by running basic `grep` commands to search for strings that might indicate the flag. The typical flag format is `picoCTF{...}`.
   - Replace `filename.txt` with the actual name of the file you are searching through.

4 Analyze the Output:

i. Review the output generated by the `grep` command. If the flag is present in the file, it will be displayed in the terminal.

ii. If you do not find the flag immediately, consider adjusting your search terms or using additional `grep` options, such as:

   1. `-i` for case-insensitive searches
   2. `-r` for recursive searches through directories

5 Explore Patterns:

If the flag is not found with a simple search, analyze the content of the text file to identify any patterns or keywords that may be relevant.

Experiment with more complex regular expressions to locate the flag or other hidden information.

6 Document and Submit the Flag:

Once you successfully identify the flag, ensure it is formatted correctly for submission.

Submit the flag on the PicoCTF platform to validate your findings.

Conclusion:

The *First Grep* challenge serves as an introduction to the `grep` command and its applications in digital forensics. By completing this challenge, participants develop practical skills in text searching and analysis, which are vital in various cybersecurity contexts.

40 bases

Challenge Name: Bases
Category: Cryptography
Points: 100
Objective: Decode or convert numbers between different bases to find a hidden flag.

Overview:

The *Bases* challenge requires participants to manipulate and convert numbers expressed in various bases (e.g., binary, decimal, hexadecimal) to uncover a hidden flag.

Steps to Solve:

1. **Access the Challenge**: Navigate to the challenge URL and review the provided data.
2. **Identify Bases**: Determine the bases involved in the challenge.
3. **Convert Numbers**: Use tools or programming languages (like Python) to perform base conversions.
4. **Decode the Flag**: Convert any encoded values (e.g., ASCII) to reveal the flag, typically formatted as `picoCTF{...}`.
5. **Submit the Flag**: Document the flag and submit it on the PicoCTF platform.

Conclusion:

The *Bases* challenge enhances understanding of numerical systems and base conversions, essential skills in cryptography and cybersecurity.
