picoCTF - picoGYM walkthrough: Beginner picoMini 2022

Codebook

Objectives

> - To be able to download and execute a python script

1. Start by downloading the required files using the Webshell

```
wget https://artifacts.picoctf.net/c/102/code.py

wget https://artifacts.picoctf.net/c/102/codebook.txt
```

2. You can use the command `ls` to check that the relevant files have been downloaded locally. Make sure that both `code.py` and `codebook.txt` are in the same directory
3. Executing the python file `code.py` will then print the challenge flag. You can do this by entering the following command

```
python3 code.py
```

runme.py

Objectives

> - To be able to use basic command line tools

1. Use the wget command to download the file

▼ Hint?

Right click on the link of the file you wish to download. Then simply run the following in the terminal `wget <link_to_file>`

fixme1.py

Objectives

> - To be able to fix issues in a python script

1. Start by downloading the required python script using the Webshell `console wget https://artifacts.picoctf.net/c/39/fixme1.py`
2. Run the python script `fixme1.py` to see what type of error is raised and take note of the line number on where the error is found `console python3 fixme1.py`
3. Fix the error in the python script by using a terminal editor

```
vi fixme1.py
```

▼ Hint?

You should encounter a `IndentationError` on **line 20**, to begin fixing this issue

```
flag = str_xor(flag_enc, 'enkidu')
print('That is correct! Here\'s your flag: ' + flag)
```

fixme2.py

Objectives

> - To be able to fix issues in a python script

1. Start by downloading the required python script using the Webshell `console wget https://artifacts.picoctf.net/c/65/fixme2.py`
2. Run the python script `fixme2.py` to see what type of error is raised and take note of the line number on where the error is found `console python3 fixme2.py`
3. Fix the error in the python script by using a terminal editor

```
vi fixme1.py
```

▼ Hint?

With the python file opened, navigate to **line 22** and fix the incorrect Syntax. The conditional operator `if` in Python requires two equal signs instead of one.

```
if flag == "":
print('String XOR encountered a problem, quitting.')
```

serpentine.py

Objectives

> - To be able to edit and exploit application code

1. Run `serpentine.py`. Try to select option `b`. Oh it didn't work, well we'll have to dive deeper into the code to get the flag.
2. Open `serpentine.py` in a terminal editor
3. Extract the flag

▼ Hint?

Was your first instinct to decode each individual byte in the flag and piece them together?

Well what if I told you there was an easier way! Let the program do the heavy lifting for you.

Notice in `line 20` that there is a method called `print_flag`. We just need to append it to the output when selecting option `b` on `line 69`.

convertme.py

Objectives

> - To be able to convert between decimal and binary

1. Start by downloading the required python script using the Webshell
2. Run the python script `convertme.py` `console python3 convertme.py`
3. Convert the decimal number to one in binary

▼ Hint?

You should see a randomly generated decimal value, converting this value into binary will provide the required flag. There are many websites available for converting decimal to binary, e.g. https://www.rapidtables.com/convert/number/decimal-to-binary.html

Glitch Cat

Objectives

> - To be able to use networking tool netcat
> - To be able to convert hex characters to ascii

1. netcat is a computer networking utility for reading from and writing to network connections using TCP or UDP. The command `nc` will allow us to connect to the the flag printing server using netcat `console nc saturn.picoctf.net 51109`
2. Like the challenge mentions, the service is broken. The printed flag contains ASCII characters, which are those `chr(0xXX)` values where XX is a integer number. To retrieve the correct format flag, we must convert these ASCII characters
3. The ASCII characters can be directly converted inside the Webshell using Python.

▼ Hint?

First open up Python in the webshell by entering the command `python3`. Then you may use the `print` function with the incorrected formatted flag to acquire the valid flag

HashingJobApp

Objectives

> - To be able to use networking tool netcat
> - To be able to carry out md5 hashing

1. netcat is a computer networking utility for reading from and writing to network connections using TCP or UDP. The command `nc` will allow us to connect to the the flag printing server using netcat `console nc saturn.picoctf.net 57689`
2. Using md5, hash the quote and provide it as an answer
3. Repeat this for the next 2 quotes too

▼ Hint?

To access the flag you must `md5 hash` a randomly generated string. This can be done in web apps like the following: https://www.md5hashgenerator.com, or in your command line by using the `md5sum` tool. `console echo -n 'string' | md5sum`

PW Crack 1

Objectives

> - To be able to understand and exploit application code

Steps:

1. Open `level1.py` in your terminal
2. Go through the code line by line, ignoring the section that has been highlighted for you to ignore

▼ Hint?

Notice on `line 19` that the program runs a conditional check for the user's password. Fortunately for us the password is in plain text for us to copy.

3. Run `python3 level1.py level1.flag.txt.enc` and enter the correct password you have found when prompted
4. Congratulations! You have found the flag. Go ahead and submit it!

PW Crack 2

Objectives

> - To be able to understand and exploit application code
> - To be able to convert from hexadecimal to ascii characters

1. Open `level2.py` in your terminal
2. Go through the code line by line, ignoring the section that has been highlighted for you to ignore. Do you notice anything *different* this time?

▼ Hint?

Notice on `line 18` that the program runs a conditional check for the user's password. Unlike the previous challenge this is not in a human readable format. The prefix `0x` is used to denote hexadecimal bytes. Let's convert it to something readable by using this

online hexadecimal converter

3. Convert the password to a regular ascii string
4. Run `level2.py level2.flag.txt.enc` and enter the correct password you have found when prompted
5. Congratulations! You have found the flag. Go ahead and sumbit it!

PW Crack 3

Objectives

> - To be able to understand and exploit application code
> - To be able to conceptualise brute force attacks

1. Open `level3.py` in your terminal
2. Notice the list of potential passwords on `line 44`? You can try something called a brute force attack where you essentially go through the list of passwords and try them out until you stumble upon the correct one

▼ Want to learn more?

It may not seem particularly clever but it is very useful way of gaining the correct password/keys. Reason being that computers have advanced to the point where they can 'crack' a password in mere moments.

How do you counter this? First off never hardcode secrets into your application code. Secondly, always use a strong password to stave off these brute force attacks.

Password strength checker

Note do not enter your actual password into these types of websites, just make up your own to test with

PW Crack 4

Objectives

> - To be able to understand and exploit application code
> - To be able to conceptualise brute force attacks
> - To be able to create short scripts to carry out brute force attacks

1. Open `level4.py` in your terminal. Do you notice anything *different*? We are now dealing with more passwords that we can manually handle. I don't know about you but I don't want to sit here and type out every single password. Why not let the computer do it for us?
2. Try to edit the the program `level4.py` in your favourite terminal editor. Your aim is to figure out a way to go through the list of possible passwords.

▼ Hint?

Copy the `pos_pw_list` into the `level_4_pw_check` method. From there create a **for loop** to go through the list of possible passwords

PW Crack 5

This time our list of passwords have gotten much longer and are now placed in a completely separate file called `dictionary.txt`. How will we brute force it this time around?

Objectives

> - To be able to understand and exploit application code
> - To be able to conceptualise brute force attacks
> - To be able to create short scripts to carry out brute force attacks

1. Edit `level5.py` file to open the `dictionary.txt`
2. Use a **for loop** to iterate through the potential passwords in the dictionary

▼ Hint?

Try using pythons inbuilt function called `open()` to read the file. Its default behaviour is to read through the file line by line which is convenient for us since this is how the passwords are separated in `dictionary.txt`.

> Still stuck? Remember that there are invisible characters you have to be wary of. These are whitespace characters like `\t\r\n` which affects the hashing of the string. You need to trim these before hashing.