Hold the pickle

Official HACon CTF write up

Author:they can't stop us all

This challenge is a mashup of some of my favourite python features. This is my first CTF challenge and I hope you all had fun playing.

Step one: Running the programme

First thing we need to do is run the programme to see what it gives us. We can do this by opening a terminal and running it like any python script. CD into the directory it is saved into then run the following command

Python holdthepickle.pyc

Upon running the programme were given some dialog, upon passing through the dialog using the enter key we're given a hint. "Check the directory you copied this to" interesting… Lets save this for later.

Step two: .pict?

Now we have a python script and a .pict file. We need to see what the code is actually doing in the background. To do this we can use a python library called uncompyle6. If you don't have this installed you can follow the link and it will show you how to pip install.

Decompile the script using

```
uncompyle6 holdthepickle.cpython-37.pyc
```

This will return it into the terminal copy and paste this into a .py file.

The way I approach reversing is to follow the "flow" of information, so with that in mind let's start.

Looking at what's imported we can see that time has been imported and pickle. Time is a useful library but it might not give us any answers, pickle however, is very very interesting…..

Looking at the first function "dweeb" this function takes one positional argument, returns it as an integer and then converts it into binary, interesting…

```
def dweeb(x):
     return(int(bin(x)[2:]))
```

The next function is "flagwrap" it has no positional arguments. Before anything is done it calls dweeb within a variable. It's passing a lot of numbers? If you know binary and how they match up with numerical values then you may have got the flag already. It then returns the flag format with the previously mentioned variable.

```
def flagwrap():
```

```
 y = dweeb(65), dweeb(115), dweeb(116), dweeb(114),
dweeb(97),dweeb(102),dweeb(116),dweeb(119)
 return("HAC{",y,"}")
```

The final function is “friends”. Most of this we saw whilst running the script. It contains some prints and some if’s. Most of it is just garbage and can be ignored but anything after the else in the if statement is quite interesting. It has a variable called “pickle_out” which creates the .pict file. It then dumps in “flagwrap” then closes the file and gives you the hint.

```
def friends():
     print("Bonsoir") #opening challenges
     time.sleep(2)
     print("welcome to my challenge")
     time.sleep(2)
     print("I hope you like")
     time.sleep(2)
     user = input("Greetings....")
     if user == "hi":
          print("hello! So simple to crack isn't it?")
          print("Here's your flag")
          print("HAC{lol}")
          return
     else:
          pickle_out = open("flag.pict","wb")
          pickle.dump(flagwrap(), pickle_out)
          pickle_out.close()
          print("Check the directory you copied this to ;)")
          extra = input("Anything else I can do for you?")
          if extra == "y":
               input("What is it then?")
               print("sorry I can't do that, have you tried
holding a pickle?")
          else:
               time.sleep(2)
               print("Okay, don't forget to hold the pickle.")
```

Step three: The solve

There's three ways we can solve this challenge. If you know python it's quite an easy solve. Let's look at the first function again.

```
def dweeb(x):
     return(int(bin(x)[2:]))
```

If you're au fait with the bin function in python you will know it converts a value into binary. This function will take the passed argument and convert it into binary.

Let's take a look at function 2:

```
def flagwrap():
 y = dweeb(65), dweeb(115), dweeb(116), dweeb(114),
dweeb(97),dweeb(102),dweeb(116),dweeb(119)
 return("HAC{",y,"}")
```

The Y variable calls the dweeb function with some numbers. As previously mentioned if you know how binary works with numbers then you will know that these numbers are being converted to binary which in turn we can convert to ASCII. So if we convert the return statement in flagwrap to print, remove everything underneath and call flagwrap instead of friends then the script will print the binary value of the flag! It would look like so:-

```
import time
import pickle


def dweeb(x):
     return(int(bin(x)[2:]))

def flagwrap():
 y = dweeb(65), dweeb(115), dweeb(116), dweeb(114),
dweeb(97),dweeb(102),dweeb(116),dweeb(119)
 return("HAC{",y,"}")

flagwrap()
```

Once we have the binary value we can convert this online using rapidtables BOOM! We have the flag!

The second way to solve the above is to reverse the .pict file. This is super simple to do. If we open the .pict file it's a garbled mess. We will have to reverse it to get the flag. If you have followed the steps above and restore the file to as it was when you first opened it. To reverse the .pict file then let's look at how it is created. If you know the pickle library you will know that to reverse a pickle script is quite simple.

Pickle is a serialisation library basically in layman's terms, it takes a value and makes it unreadable. We don't need to do any master hacking to get access to the file as our computer has created this file. To save you the time I have wrote a simple solution script for you. It is as follows:-

```
Import pickle
Def solution():
     Astra = open("flag.pict", "rb")
     New_dict = pickle.load(Astra)
     astra.close()
     print(new_dict)

Solution()
```

All this script does is open the .pict file as read bytes. We create a variable that uses the pickle.load function and print it! Great, we've got the flag!

Step four: Closing comments

If you have learnt something from this challenge that is great, I learnt pickle by playing CTF's and pulling mine and d1pr1de's hair out at it! I really do hope you enjoyed my challenge and if you would like to play some pickle and decompile challenges I'd recommend the "Peak Hill" room on tryhackme. Keep pwning folks and remember Astra Cyber Team FTW.

If you would like more info on Astra you can find it here