



Verify



EasyForensicspicoCTF 2024grepbrowser_webshell_solvablechecksum

AUTHOR: JEFFERY JOHN

Description

People keep trying to trick my players with imitation flags. I want to make sure they get the real thing! I'm going to provide the SHA-256 hash and a decrypt script to help you know that my flags are legitimate.

You can download the challenge files here:

- [challenge.zip](#)

The same files are accessible via SSH here:

```
ssh -p 63078 ctf-player@rhea.picoctf.net
```

Using the password `6abf4a82`. Accept the fingerprint with `yes`, and `ls` once connected to begin. Remember, in a shell, passwords are hidden!

- Checksum:
b09c99c555e2b39a7e97849181e8996bc6a62501f0149c32447d8
e65e205d6d2
- To decrypt the file once you've verified the hash, run
`./decrypt.sh files/<file>.`

This challenge launches an instance on demand.

Its current status is: **RUNNING**

Instance Time Remaining: **29:52**

Restart Instance

Hints ?

123

PICO GYM

Name: Verify (easy)

Downloadable Files:

[No need to download other files]

Solution

> Launching the instance tells me that the best way to solve this problem was to open up a Linux Terminal and use ssh to access the files. I used this command and password:

```
$ ssh -p 63078 ctf-player@rhea.picoctf.net
```

Password: 6abf4a82

> Using the ls command, I saw three things that I could interact with. The first one is a file that contains the checksum text. The second one is a .sh file that I should use to decrypt the file. The third one is a directory called "files" containing files with file names that are unusable to figure out.

> My first instinct and my first mistake is to think that I can use [decrypt.sh] and the [checksum.txt] and pipe it to the files directory as a way to find the right answer. I failed to understand what is inside decrypt.sh and what it is for.

> hashdeep and sha256deep were commands that were promising but it is not available and I am not able to install them.

> However, I realized that I could use sha256sum to get the hash value. At first, I assumed that the files already have a hash value somewhere but when I realized that they don't have any yet, well... the solution has become clearer.

> To create a hash value for every single file inside the [file] directory, I used the following command:

```
$ sha256sum /home/ctf-player/drop-in/files/* > filescheck.txt
```

> The command allows me to create a hash value of each file using * and put the result inside a text file called "filescheck.txt"

> Now at this point, I could use the cat command on the checksum.txt, get the hash value inside, and compare it to the list, however, there is a better way to find it.

> I used this command to help me find the exact file name that has the same hash value that was provided in the checksum.txt file.

```
$ desired_checksum=$(cat checksum.txt) && grep "$desired_checksum" filescheck.txt
```

- desired_checksum is a variable. The contents of this variable is the result of the command \$(cat checksum.txt).
- and then I used the && to continue and use grep command to get the exact string inside the variable desired checksum and compare it to filescheck.txt

> That command gave me this result:

```
ctf-player@pico-chall$ desired_checksum=$(cat checksum.txt) && grep "$desired_checksum" filescheck.txt
b09c99c555e2b39a7e97849181e8996bc6a62501f0149c32447d8e65e205d6d2 /home/ctf-player/drop-in/files/451fd69b
```

> Now I have the file name and it is 451fd69b. I thought this was the answer but then I realized that this is where the decrypt.sh file comes in. It also solidified my assumption when I used that cat command on it and I got gibberish.

```
ctf-player@pico-chall$ cat 451fd69b
Salted__bK♦♦♦IQ♦L♦♦♦♦[h=♦I♦♦♦♦♦♦♦♦♦♦Q♦♦♦♦♦e# ♦♦)4♦♦(♦♦o♦♦BY♦ctf-player@pic
```

> After reading the problem prompt again, I realized that this is how I'm supposed to use it. Using it has given me the flag

```
ctf-player@pico-chall$ ls
checksum.txt  decrypt.sh  files  filescheck.txt
ctf-player@pico-chall$ ./decrypt.sh files/451fd69b
picoCTF{trust_but_verify_451fd69b}
```

The Flag is: picoCTF{trust_but_verify_451fd69b}

Things I've learned:

- read the problem prompt again.
- Worked with what is available. I should not have spent a lot of time figuring out how to install sha256deep.
- Remember the difference between encryption and hashing.