Guilherme Alt Chagas Merklein

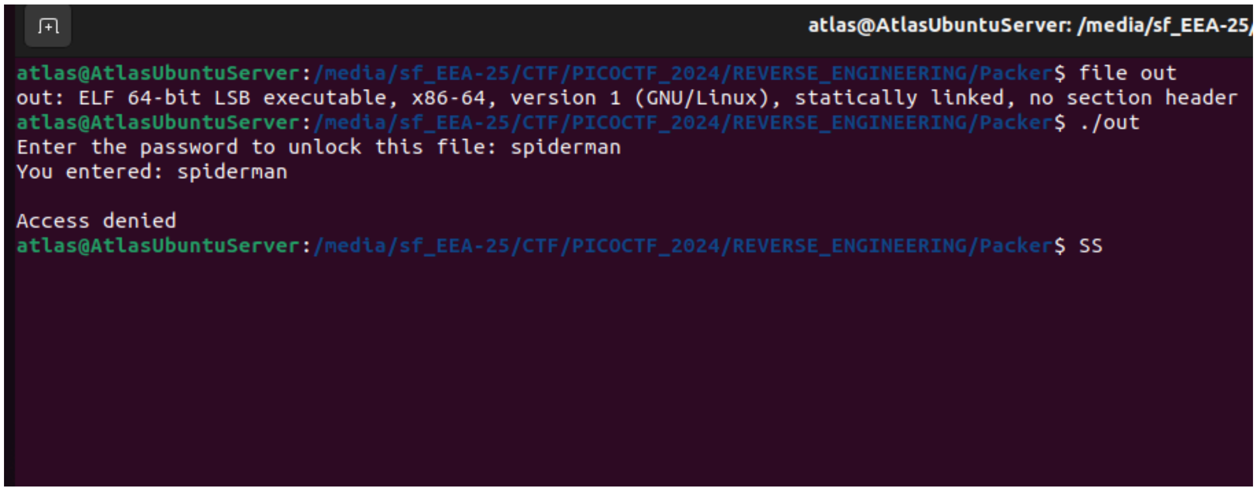
PicoCTF 2024 - Reverse Engineering

flag: picoCTF{U9X_UnP4ck1N6_B1n4Ri3S_e190c3f3}

27/03/2024

Packer

In this challenge, you would be asked to reverse a linux executable. They provided a file named "out". First thing I did was start my ubuntu VM, since my main OS is windows, and then run a "file" command to check the file type (image 1). It definitely seemed like an *ELF* executable, which is the main executable format for Linux binaries (Windows uses another format, named *PE*). I also ran the file once to see what the program was about. It just asks for the user to enter a password (also image 1).

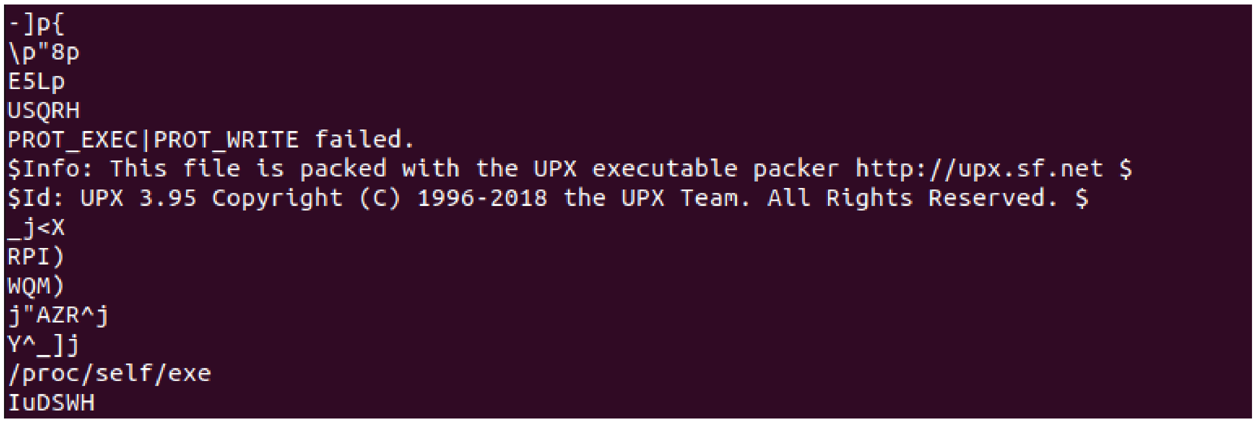


```
atlas@AtlasUbuntuServer: /media/sf_EEA-25/
atlas@AtlasUbuntuServer:/media/sf_EEA-25/CTF/PICOCTF_2024/REVERSE_ENGINEERING/Packer$ file out
out: ELF 64-bit LSB executable, x86-64, version 1 (GNU/Linux), statically linked, no section header
atlas@AtlasUbuntuServer:/media/sf_EEA-25/CTF/PICOCTF_2024/REVERSE_ENGINEERING/Packer$ ./out
Enter the password to unlock this file: spiderman
You entered: spiderman

Access denied
atlas@AtlasUbuntuServer:/media/sf_EEA-25/CTF/PICOCTF_2024/REVERSE_ENGINEERING/Packer$ SS
```

Image 1: check file type and run it.

Next thing you normally do when reversing an executable would be looking at the strings within the file, to see if the flag is straight up obvious. For this task, I ran the command “strings” and passed the file name as argument. I scrolled all the way to see If I could find anything suspicious, and eventually I found something interesting (Image 2): The file was apparently packed with UPX.

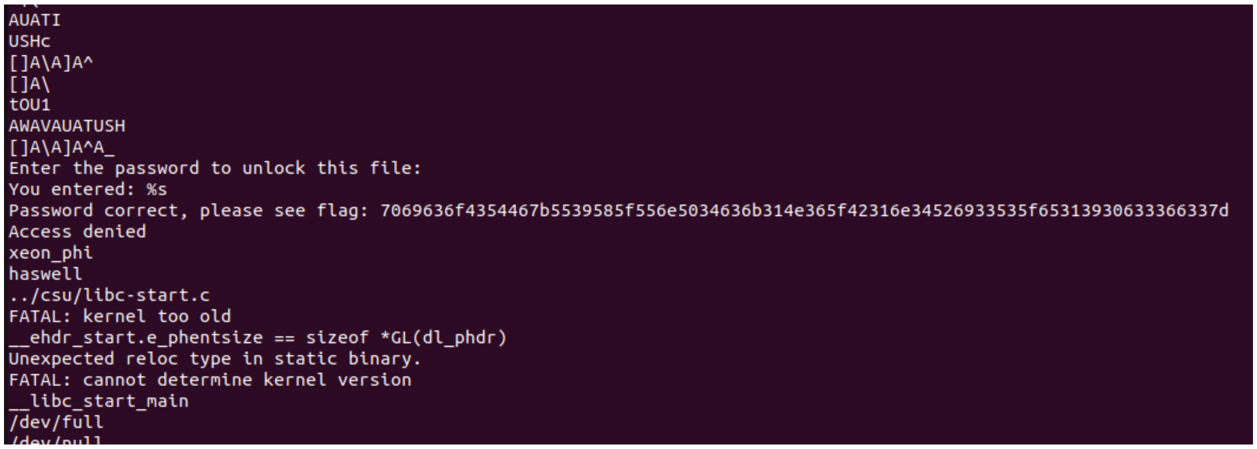


```
-]p{
\p"8p
E5Lp
USQRH
PROT_EXEC|PROT_WRITE failed.
$Info: This file is packed with the UPX executable packer http://upx.sf.net $
$Id: UPX 3.95 Copyright (C) 1996-2018 the UPX Team. All Rights Reserved. $
_j<X
RPI)
WQM)
j"AZR^j
Y^_]j
/proc/self/exe
IuDSWH
```

Image 2: Upx info embedded in the file.

I didn’t know what that was at the time, so I did some research, and it is basically a general tool to compress executable files. I also found good news: You can actually easily unpack it. I ran the command *“upx -d ./out”* on Ubuntu and the file got unpacked, so I renamed that version to out_unpacked.

Next thing I did was run the “strings” command again on the unpacked file, and got some new interesting strings. One of them had a “password correct” phrase (image 3), which I suppose is passed as argument to a printf function when the user enters the correct password.

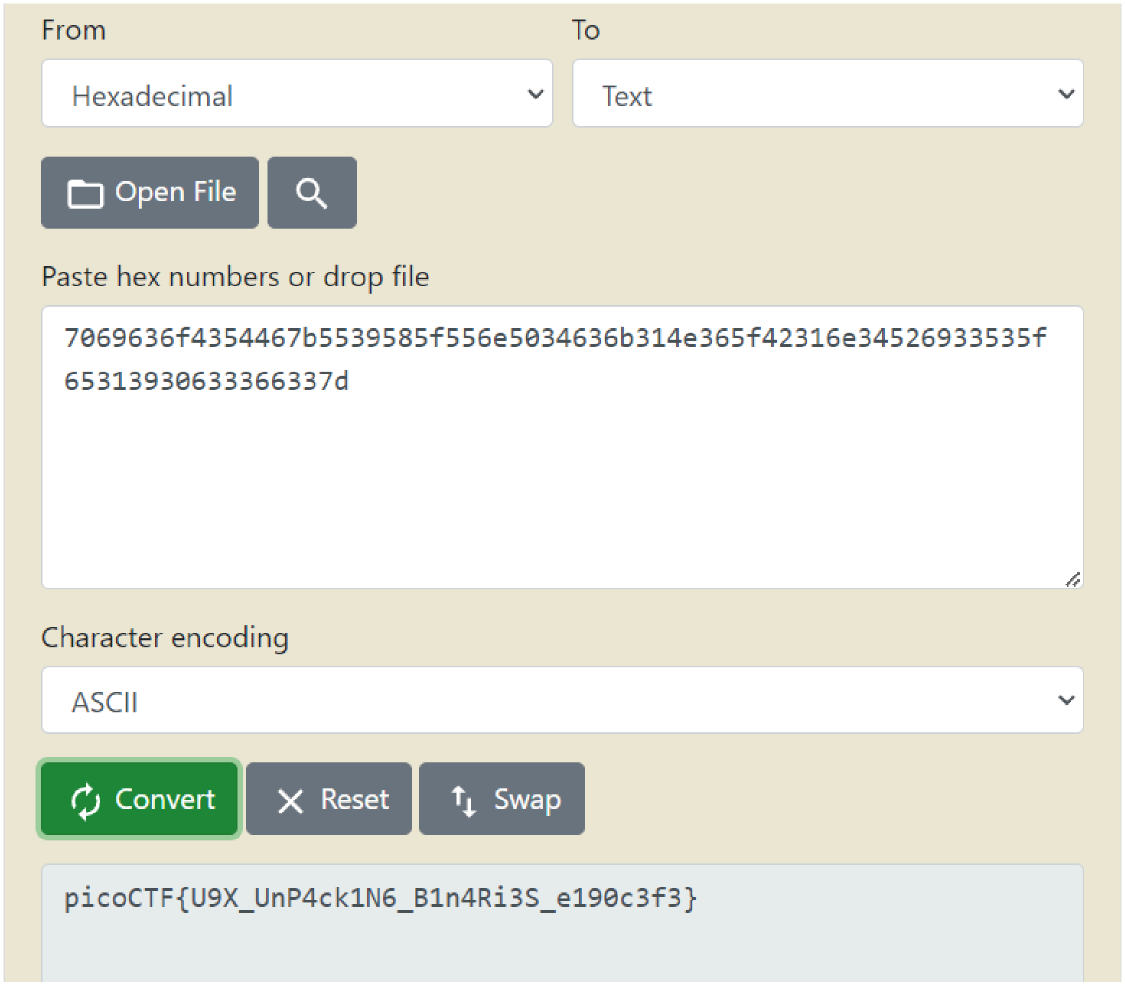


```
AUATI
USHc
[]A\A]A^
[]A\
tOU1
AWAVAUATUSH
[]A\A]A^A_
Enter the password to unlock this file:
You entered: %s
Password correct, please see flag: 7069636f4354467b5539585f556e5034636b314e365f42316e34526933535f65313930633366337d
Access denied
xeon_phi
haswell
../csu/libc-start.c
FATAL: kernel too old
__ehdr_start.e_phentsize == sizeof *GL(dl_phdr)
Unexpected reloc type in static binary.
FATAL: cannot determine kernel version
__libc_start_main
/dev/full
```

Image 3: password correct string.

My first thought was that the flag was base-64 encoded, but that was not it, so I thought - and I was wrong - that there was more to be done, And I would actually have to debug this problem, so I went to sleep.

Next day I noticed that if you separate the string into chunks of 2 digits each, the values could actually be the hexadecimal representations of ascii characters (I thought this because ascii characters usually have a small hex value representation, and by that I mean "digit_code < 0x80"), so I tried a hex to ascii decoder, and got the flag (image 4).



From: Hexadecimal

To: Text

Paste hex numbers or drop file

```
7069636f4354467b5539585f556e5034636b314e365f42316e34526933535f
65313930633366337d
```

Character encoding: ASCII

```
picoCTF{U9X_UnP4ck1N6_B1n4Ri3S_e190c3f3}
```

Image 4: flag.