Exercise 2

Group members: Endré Hadzalic, Odin Kvarving

# 1. PyCalc challenge on ctf.idi.ntnu.no

### 1.1 Summary

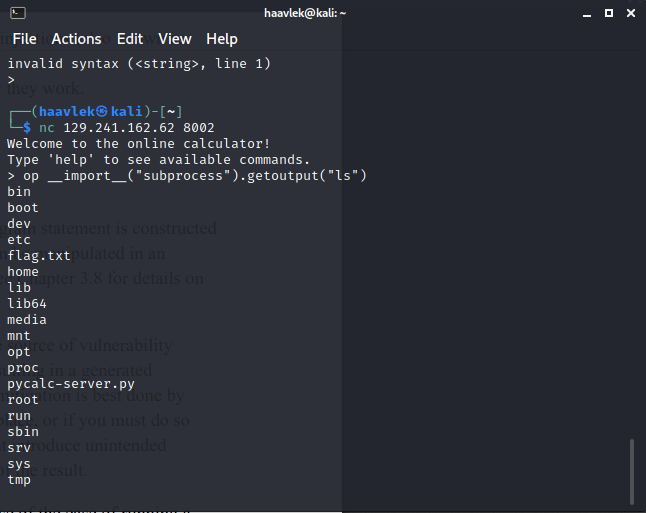
First and foremost, when we opened the calculator we could see that you could use the command ‘help’ to list what commands could be run in the calculator.

There we could see that you could either use: “echo Hello World!” to write the argument to standard output, or “op 2+2” to solve a simple equation. We figured that the calculator might use eval()-function to calculate the expressions, since this is a relatively easy way to calculate expressions. This is because eval evaluates the expression as a Python-expression, so it can be used to solve equations. The downside of using eval however, is that it can also handle other potentially harmful statements.

To test if we could do code-injection through the eval()-function, we tried writing this command:

“op \_\_import\_\_(“subprocess”).getoutput(“ls”)

The subprocess module allows us to run shell code in Python, which we used in combination with the command .getoutput from “ls” to print all files and directories in the present directory. This gave us a list of all the directories that were present in the directory that the calculator was running in:



Now we could see that there was a text-file called flag.txt, which we could read with the same method as mentioned above:

“op \_\_import\_\_(“subprocess”).getoutput(“cat flag.txt”)

cat flag.txt allows us to read the contents of this file, where we eventually found the flag:



### 1.2 Steps to reproduce

Write the following commands in the terminal, when connected to the calculator:

* “op \_\_import\_\_(“subprocess”).getoutput(“ls”)
* “op \_\_import\_\_(“subprocess”).getoutput(“cat flag.txt”)

This will give you the flag that is contained in the text-file.

### 1.3 Impact

‘That the calculator is using the eval()-function to calculate expressions can have potentially major risks, because an attacker can have their own arbitrary code injected and executed with the current user privileges. In this way, an attacker can gain access to files, execute functions, change or delete files from the system etc.

This should not be used to perform simple calculations, since there are much better alternatives which can do the exact same thing.

### 1.4 Timeline

A reasonable time to fix this issue would be no more than 90 days, as it should be fairly simple to fix.

# 2. Flag from Hacker101’s Hello World! CTF

# **Injecting memory address into Stdin to get flag**

Opened bin file in Ghidra

Found a function called print\_flags which had a memory location of:

0x0000000004006ee

So typing in data until you get segmentation fault then injecting the memory location at the end of the string in the URL as such:

ABCDEFGHIJKLMNOPQRSTUVWXYZABasfgdsdssfag%ee%06%40%00%00%00%00%00

Resulted in:

["^FLAG^9c89b32572f3af54caa125ba25f33d550904845df5e2f559f14c253c03710b48$FLAG$"]

