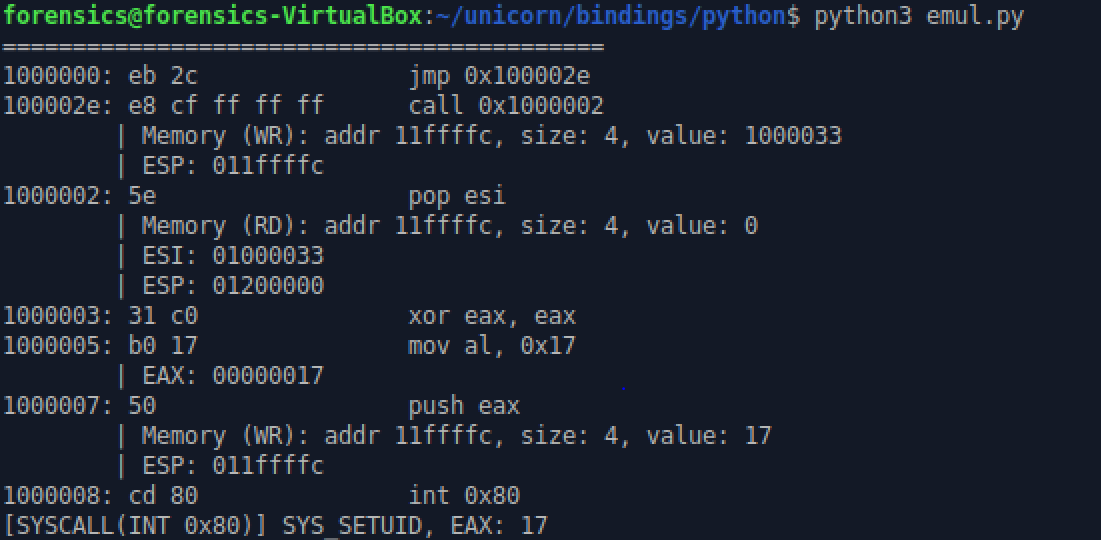
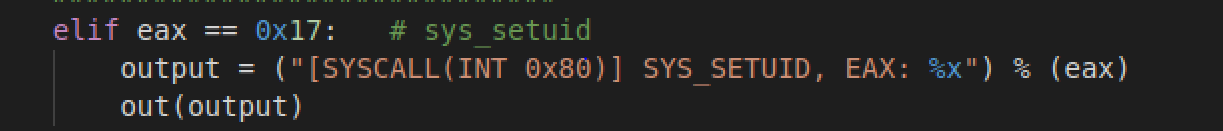
Software Security via Program Analysis

Project 2: Emulating Partial Program

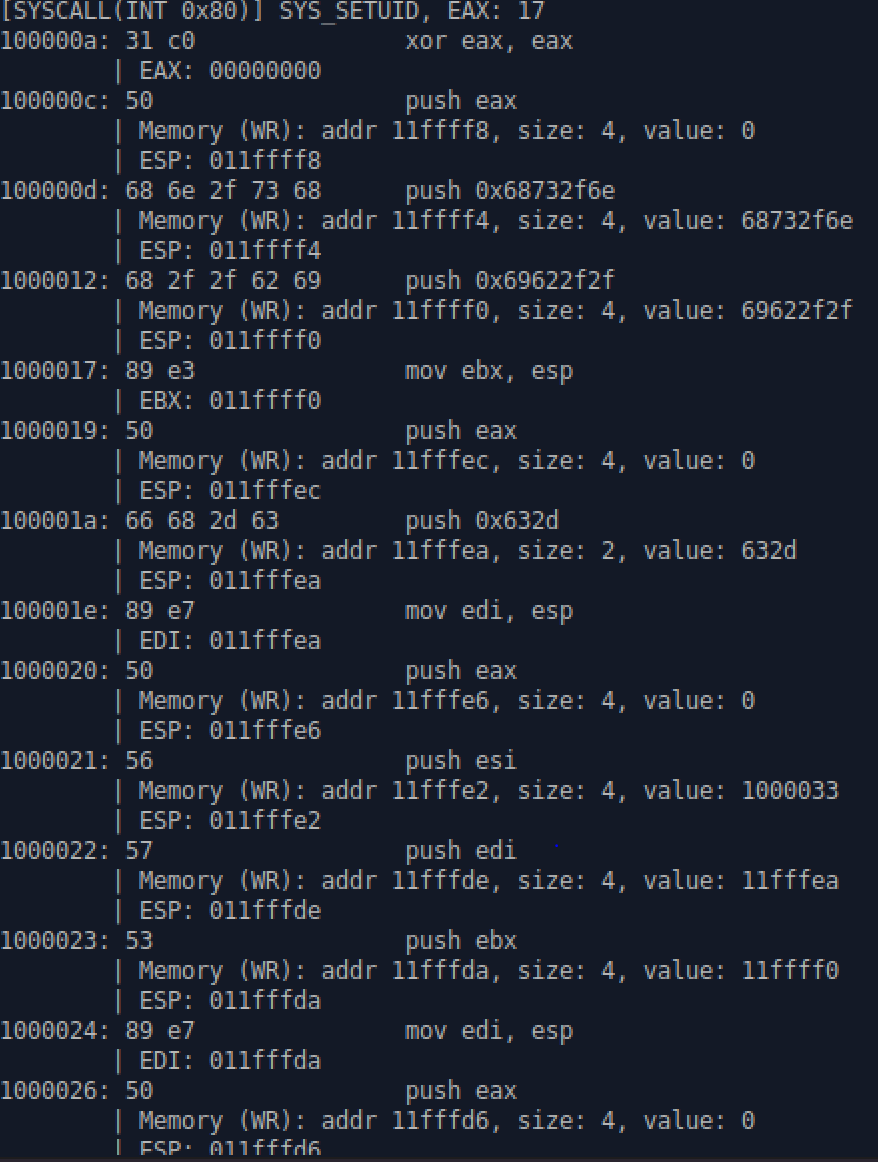
Yu-Sheng Wang (bqk3dj)

**Example 4**





The code first run the system call 0x17 (setuid).



Next, there are multiple pushed where the addresses and arguments of the instructions are recorded.

Text

Description automatically generated

Next, it runs the system call 0x11 (sys\_execv) with path “//bin/sh” and argument “/sbin/kload /tmp/o.o”.

Text

Description automatically generated

The rest of the code is string and we can skip executing them with the following code:

Text

Description automatically generated

**Example 5**

Text

Description automatically generated

Text

Description automatically generated

The program stops when encountering repeated instructions (loop).

Text

Description automatically generated

Let’s allow the program to run repeated instructions up to 100 times with the count array *cnt\_repeated*. By doing so, we can get the following result:

Text

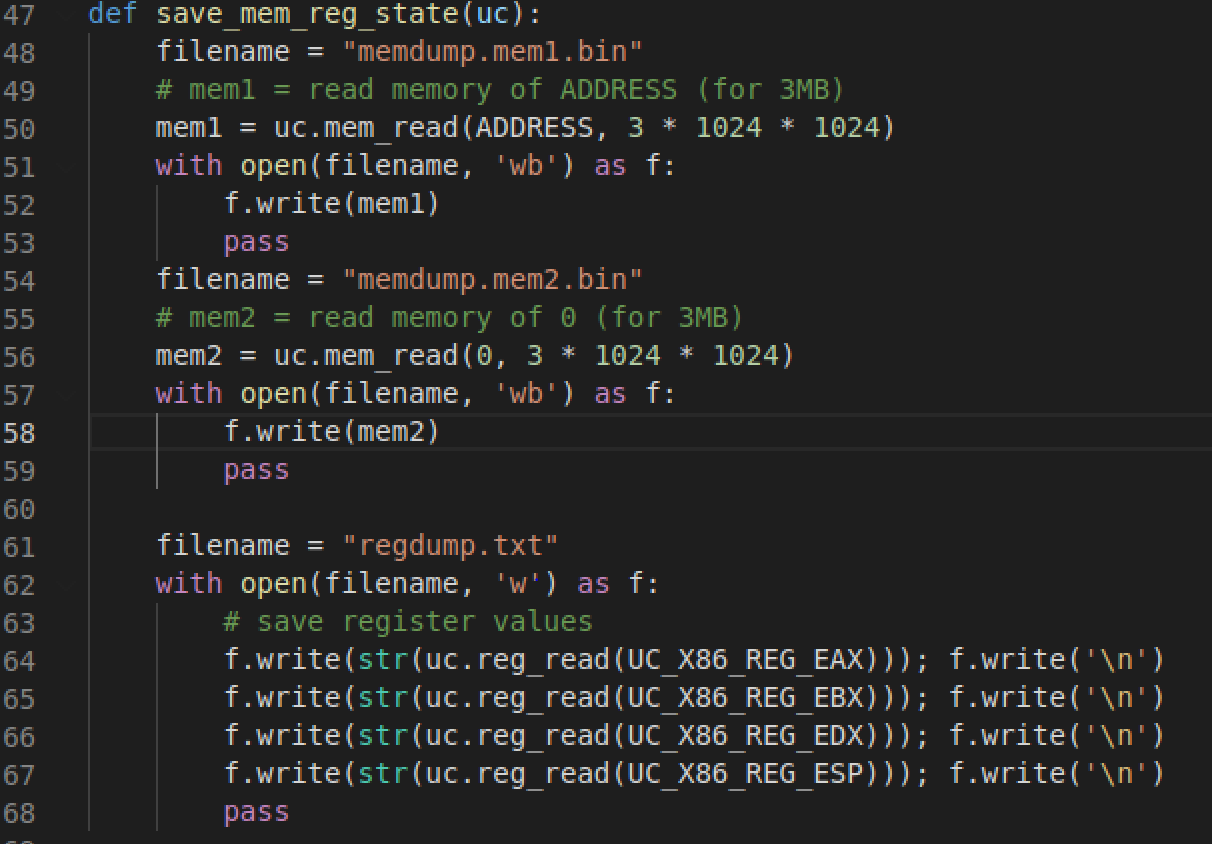
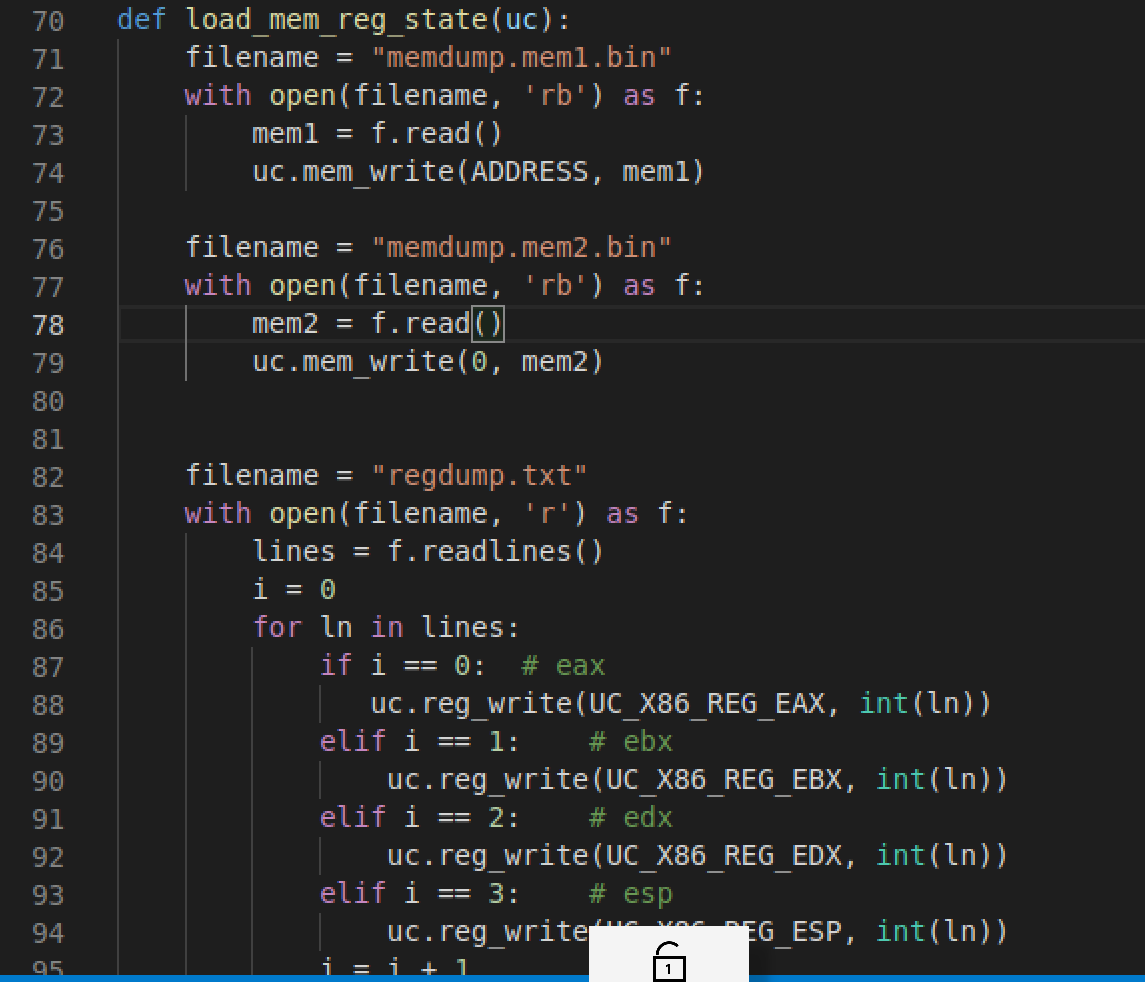
Description automatically generated

The program executes *sys\_execv* with path */bin/sh.* However, it fails when we run continue because we lost the previous state memory

Text

Description automatically generated

To solve this issue, we write the memory & register state when the execution stops and read them back in the next execution:

Finally, the program runs *sys\_execv* twice*:*

Text

Description automatically generatedText

Description automatically generated