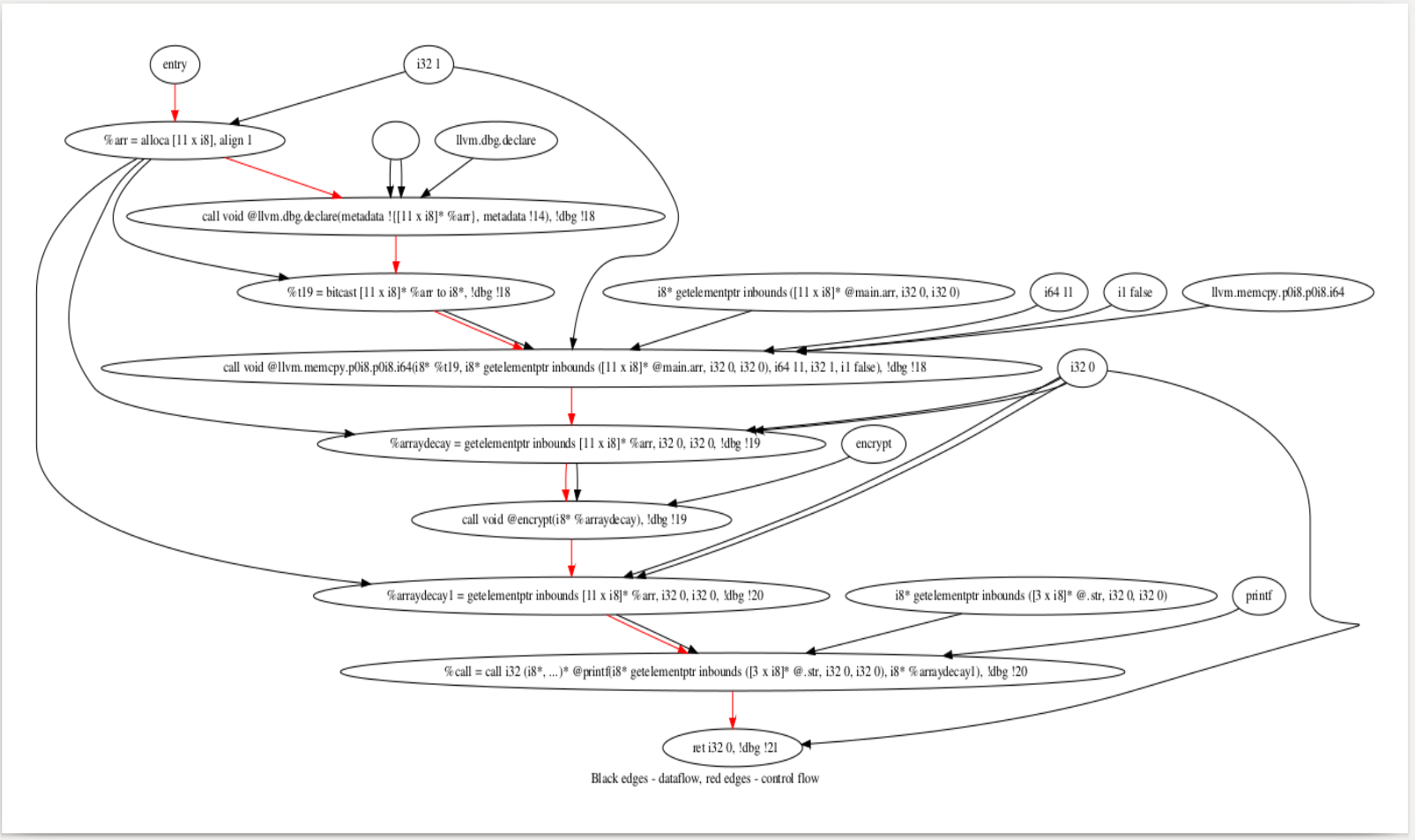
SSE Course Project

Static Analysis of code



P Girinath CS18B032

Meesa Shivaram Prasad CS18B056

28.05.2021

CS6650 - Course Project Report

AIM

Consider a code like a crypto-cipher, which has some secret key. Assume the cipher is invoked by a function, like encrypt() or decrypt(). One requirement of a good implementation is that after the function executes, there should be no traces of the key either in memory / registers / or anywhere else. This project is to write a script that uses gdb, to identify if any portion of the secret key is present in the memory or registers. This requires

(1) first an analysis to identify the memory locations of key / key related data.

(2) ensure that when the function returns, all these addresses are set to 0 after the cipher completes its execution

APPROACH

The approach we followed is as follows :

1. For obtaining the data flow, we used python-llvm-analyzer which required installation of llvm-3.3 and clang-3.3.
2. From the above, we get a Directed Acyclic Graph (DAG) which is the data flow graph.
3. The user says which variable in the code is to be tracked ie. the key.
4. Using the data flow graph, we do a Depth First Search to find all other variables which depend on the key, and store them in a list.
5. We then run the program given using gdb and print the values of these variables at the end of function.
6. From this, the user can figure out whether these locations have a random value or some sensitive value.

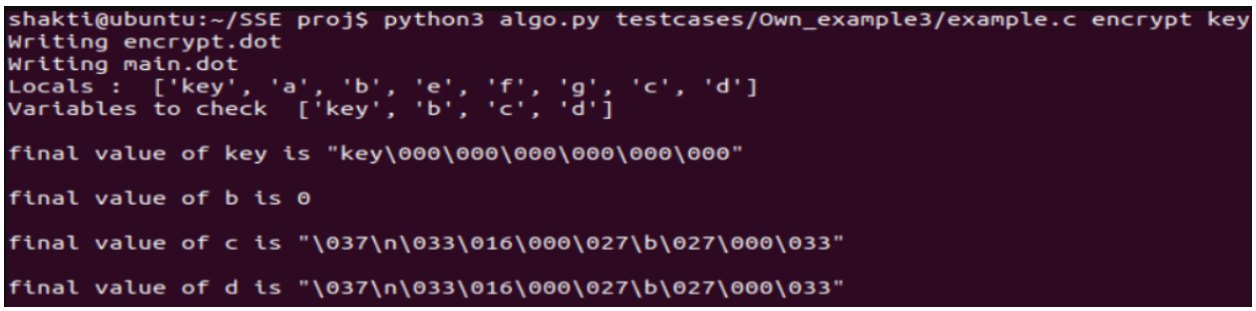
STEPS TO RUN

1. Install llvm-3.3 and clang-3.3
2. Download python-llvm-analyzer and make sure algo.py is in the same folder as it.
3. In algo.py, make sure to change the clang_path to the path/to/clang in your pc.
4. Run the python script algo.py with <path_to_c_file> , <function_name> , <key> as

command line arguments

- <path_to_c_file> : Provide path to the C file here
- <function_name>: name of the C function that u wish to inspect
- <key> : name of the key variable name that u feel is important and check whether all the variables that are dependent on key data are made 0 at the end of the function

SCREENSHOT OF RUN



DEMONSTRATION VIDEO LINK

https://drive.google.com/file/d/1m2NTkPr198wfmv-6HgbT2RraijKAM9Yt/view

GITHUB LINK FOR CODE

https://github.com/Giri2801/SSE-Proj

FUTURE WORK

1. In depth analysis of control flow and dynamic checking of dependency during the run itself.
2. Building the data flow graph from objdump, by which we can track all memory locations and registers which have some information about the key.

REFERENCES

1. Visualizing code structure in LLVM
2. Codes in test cases are taken from here and are slightly modified.