Symbolic Execution

Code can be found here: https://github.com/c2003-tamu/413

Environment Setup

- Clone git repository
- Ensure docker is installed
    - sudo apt install docker.io

Source Code

My source code (seen below) essentially just takes in 2 integers using argv. Using these 2 integers, it puts “Granted” to stdout if a^2/b == 1 and “Denied” if not.

```c
#include <stdio.h>
#include <stdlib.h>

int main(int argc, char *argv[]) {
        if(argc != 3){
                printf("Usage: %s <num1> <num2>\n", argv[0]);
                return 1;
        }
        int num1, num2;
        num1 = atoi(argv[1]);
        num2 = atoi(argv[2]);

        if (((num1 * num1) / num2) == 1) {
                printf("Granted\n");
        } else {
                printf("Denied\n");
        }

        return 0;
}
```

Running Angr

- Run command: docker pull angr/angr

- Navigate to 413/symbolic directory
- Run command: docker run -v $(pwd):/home/angr/symbolic --rm -it angr/angr
- Inside container
    - Navigate to /home/angr/symbolic directory
    - Run command: python3 solve.py
    - Based on the output generated by angr, run command: ./crackme <arg1> <arg2>

```
vboxuser@meow:~/Desktop/413/symbolic$ docker run -v $(pwd):/home/angr/symbolic --rm -it angr/angr
(angr) angr@6ceec9b22465:~$ cd symbolic/
(angr) angr@6ceec9b22465:~/symbolic$ ls
crackme  crackme.c  solve.py
(angr) angr@6ceec9b22465:~/symbolic$ python3 solve.py
Found inputs: 758135088 , 942682160
(angr) angr@6ceec9b22465:~/symbolic$ ./crackme 758135088 942682160
Granted
(angr) angr@6ceec9b22465:~/symbolic$
```

Angr Script / Findings

Using angr, we found that not only can we get access in the intended way (for example, ./crackme 2 4), but we can also abuse integer overflow in order to gain access to the application.

Below is the angr script used to make these findings:

```
vboxuser@meow:~/Desktop/413/symbolic$ cat solve.py
import angr
import claripy

project = angr.Project('./crackme', auto_load_libs=False)

arg1 = claripy.BVS('arg1', 32)
arg2 = claripy.BVS('arg2', 32)

argv = ['./crackme', arg1, arg2]
state = project.factory.full_init_state(args=argv)

simulation = project.factory.simgr(state)

simulation.explore(find=lambda s: b"Granted" in s.posix.dumps(1))

if simulation.found:
    print(f"Found inputs: {simulation.found[0].solver.eval(arg1, cast_to=int)} , {simulation.found[0].solver.eval(arg2,
cast_to=int)}")
else:
    print("inputs not found")
```

NOTE: chatgpt was used to give an overview of how to make an angr script. Prompt: give me an overview of angr scripting to detect input outcomes in a c binary. I did this in order to get an overview and broad idea of how angr scripting works and the syntax that goes into it. This output was used in conjunction with the demonstration video in order to make this angr script.