**Software Security via Program Analysis**

**Protect Against hack (HW3)**

Yu-Sheng Wang (bqk3dj)

**Pre-Work**

* Change the makefile: Add debug information (-g) and remove optimization

Text

Description automatically generated

* Disable/Enable Randomize:

echo 0 | sudo tee /proc/sys/kernel/randomize\_va\_space

* Change *collision* to *s\_collision & r\_collision* in flappybird.cpp

**Pintool Program**

Search *s\_collision* in *flappyBird.S*. The return value %eax is stored in -0xe4(%rbp).

Trace instructions in 1c5d & 1d03.

A screenshot of a computer

Description automatically generated with medium confidence

A screenshot of a computer

Description automatically generated with medium confidence

A screenshot of a computer

Description automatically generated with medium confidence

**Profile**

In the located instructions, we can monitor the values written out to memory via the following program.

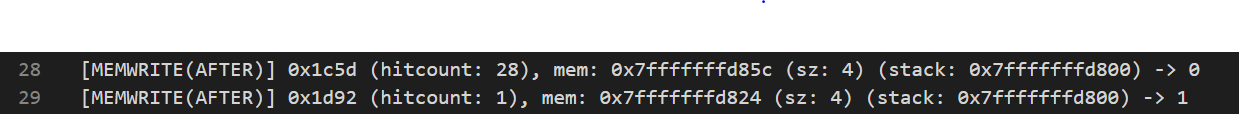
A screenshot of a computer

Description automatically generated with medium confidence

Text

Description automatically generated

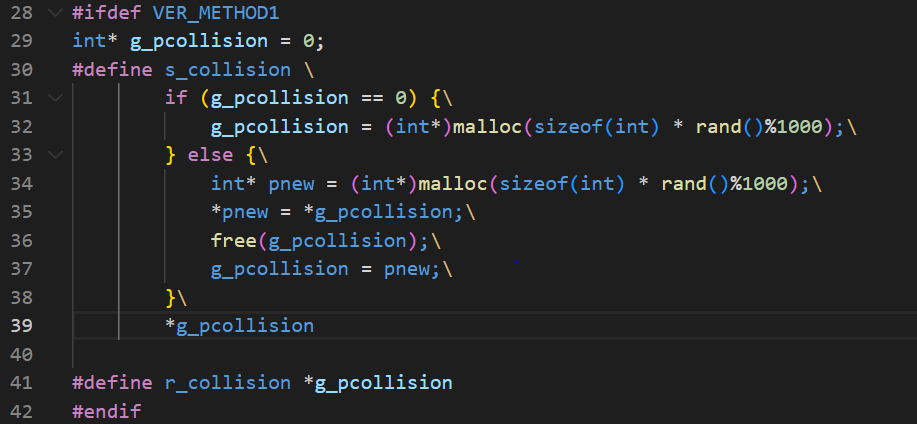
Through profiling, we can find the memory address of variable *collision* (0x7fffffffd85c) and *isOver* (0x7fffffffd824).

  
  
Now, we modify the memories of *isOver* and *collision* via Pintool. The bird won’t die when hitting the piles or the ground.  
A screenshot of a computer

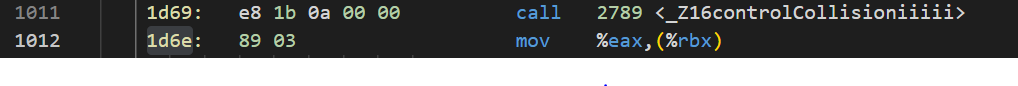
Description automatically generated

**Method 1**

At method 1, we keep changing the memory address of the critical variable via allocating memeories with random size.



Lok into the dump code and trace the instruction at 0x1d6e.

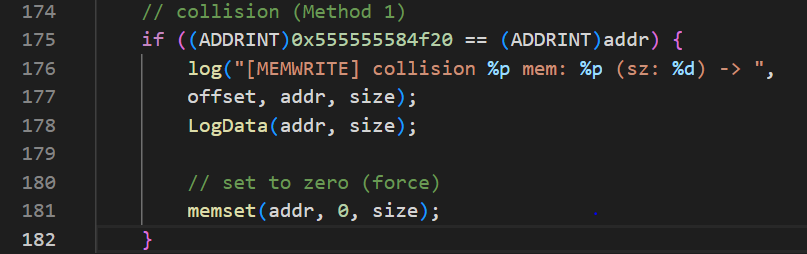


Profile the memory address via Pintool, we can see that the memory address changes every time it’s begin called.

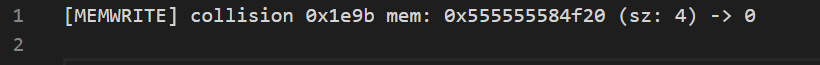
A picture containing text

Description automatically generated

Even if we try to modify the value in memory address 0x555555584f20, the hacking still fails and the bird dies when hitting the ground/piles. It’s because the critical variables keep changing its memory addresses via malloc and free.



The Pintool locates the correct memory address but loses track after.



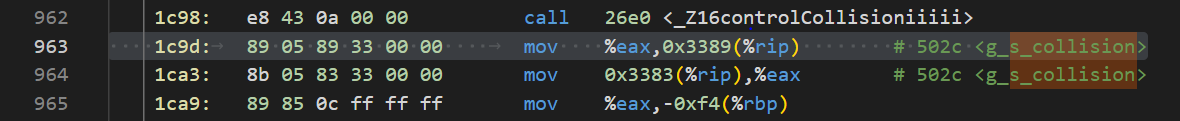
**Method 2**

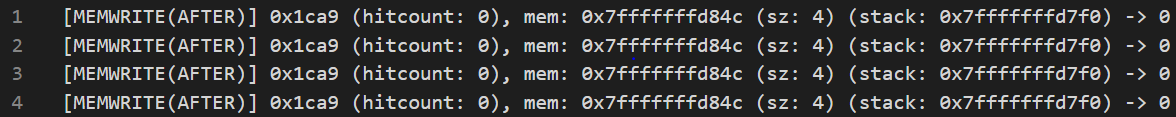
In method 2, we record the critical value to another global variable when the value is called/modified.

A screenshot of a computer

Description automatically generated with medium confidence

Now, we profile again with Pintool and identify the instruction at 0x1ca9 where the value of *collision* is modified. In a real application, the modification of g\_s\_collision should happen somewhere else so that the hacker should not notice. The memory is located at 0x7fffffffd84c.





Now, we launch the attack via Pintool.

A screenshot of a computer

Description automatically generated with medium confidence

Run the game, and we can find out that the program exits and report corrupted when hitting the piles.

Chart, waterfall chart

Description automatically generated