# June the 2nd - Fuzzing project report

# Goal (26th May - 2nd June)🏆

- **Improve** and **think** about **loop detection false positive** (function executed many times VS loop). We want **metrics** about:
  - **number of call instructions** (all of call instruction)
  - **number of unique functions** (function executed one time)
  - **number of call instrutions without ret** (function executed but not returned)
- **manual review**: test with ~~Eclipse~~ **Ghidra** with the binary

# What I have done ? 👨‍💻

## Improvements/patches

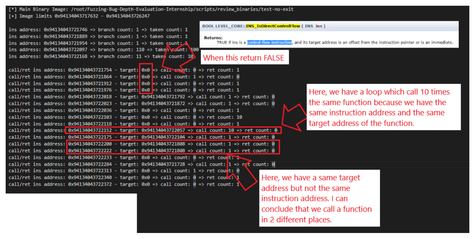
One of the improve is to make the **difference between a function which execute many times in a loop or in different places**.

To do this, we can take the **target address of the function**.

If we have the **same address instruction** and the **same target address** for 10 call, it's a **loop** which execute 10 times the same function.

Otherwise, if we have **different address instruction** but **same target address**, it's the **same function** which execute at different places.

Here is an explanation of a Pin script trace:




Result of the Pin script:

```
The bug depth with conditional branches = 3
Number of all conditional branches = 124
The bug depth with call instructions = 1
Number of unique function = 6
Number of all call instructions = 16
```

Another improvement is to have the **number of unique functions** (unique call instruction).

I based on the **number of call** (only 1) and the **target address function** (unique and not 0x0 address).

Th script: Pin script - bugdepthevaluation.cpp

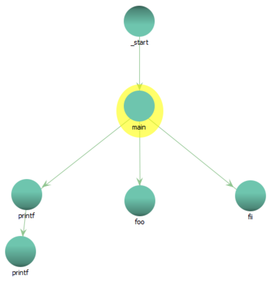
## Manual review (Ghidra)

I use Ghidra to do manual review and to have a function call graph. After these improvements/patches, we have to know if they works.

We have this test script:

```c
#include <stdio.h>

int foo() {
    int i, x = 0;

    // exit-2
    //exit(1);

    for (i = 0; i < 10; i++)
        x += i * 2;

    // exit-3
    //exit(1);

    return x;
}

int fii() {

    // exit-1
    //exit(1);

    return 1;
}


int main()
{
    int y, z = 0;

    for (int i = 0; i < 10; i++) {
        y += foo();
    }

    z = fii();

    printf("%d \n", y);
    printf("%d \n", z);
}
```

On Ghidra, we have this:




And the Pin result:

```
The bug depth with conditional branches = 3
Number of all conditional branches = 124
The bug depth with call instructions = 1
Number of unique function = 6
Number of all call instructions = 16
```

I don't understand why I have 6 for `Number of unique functions` and on Ghidra, 4 functions (main, printf, foo and fii). The function _start is not in the instrumentation I think because we focused on the main executable.

However, Ghidra `can not find precise callgraph for indirect function calls!` - Sanjay Rawat. It's maybe cause of that ?? Or it count the _start function and the printf function in double (because of the dynamically linked of the binary).

The results: Results (libxml and manual reviews)

# Where I stopped ? 🛑

We are in the end of the first part of the internship. I can improve and fix some problems about the development of the Pin script. The goal of the next week s is to concentrate on the second part: **experiment part** !!