Ali’i CFG: Improving the Accuracy and Completeness of Control Flow Graphs

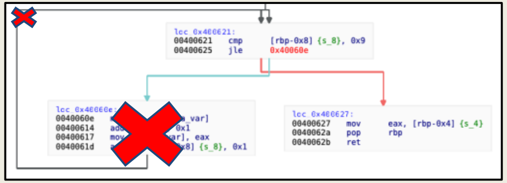
Zion Leonahenahe Basque, Computer Science
Mentor: Royu “Fish” Wang, Assistant Professor
CIDSE

Motivation

Modern day automated security analysis **depends on Control Flow Graphs** (CFG) that are assumed to be **complete** for sourceless executables. Often, these CFGs are flawed due to a lack of resolving indirect control flow. To increase automated analysis of executables, we must increase the completeness of CFG’s by resolving these indirect control locations.

On average executable programs contain nearly **91 unresolved** control flow locations, accounting for nearly **50%** of all indirect control flow in a executable. This is a major analysis flaw.

Approach: Skipping costly structures

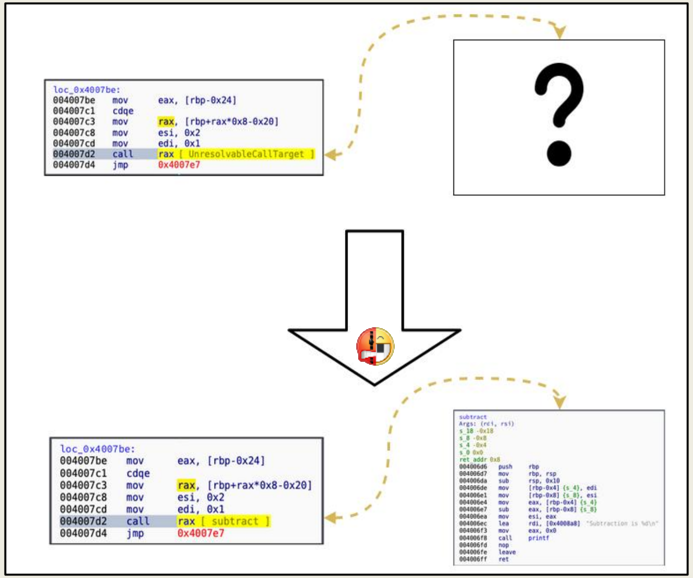


```
loc_0x400621:
00400621   cmp    [rbp-0x8] {s_8}, 0x9
00400625   jle    0x40060e
```

```
loc_0x40060e:
0040060e              _var]
00400614   add        0x1
00400617   mov        ar], eax
0040061d   a          x8] {s_8}, 0x1
```

```
loc_0x400627:
00400627   mov    eax, [rbp-0x4] {s_4}
0040062a   pop    rbp
0040062b   ret
```

Approach: Resolving Indirect Calls in CFGs



```
loc_0x4007be:
004007be   mov    eax, [rbp-0x24]
004007c1   cdqe
004007c3   mov    rax, [rbp+rax*0x8-0x20]
004007c8   mov    esi, 0x2
004007cd   mov    edi, 0x1
004007d2   call   rax [ UnresolvableCallTarget ]
004007d4   jmp    0x4007e7
```

```
loc_0x4007be:
004007be   mov    eax, [rbp-0x24]
004007c1   cdqe
004007c3   mov    rax, [rbp+rax*0x8-0x20]
004007c8   mov    esi, 0x2
004007cd   mov    edi, 0x1
004007d2   call   rax [ subtract ]
004007d4   jmp    0x4007e7
```

```
subtract
Args: (rdi, rsi)
s_18 -0x18
s_8 -0x8
s_4 -0x4
s_0 0x0
ret addr 0x8
004006d6   push   rbp
004006d7   mov    rbp, rsp
004006da   sub    rsp, 0x10
004006de   mov    [rbp-0x4] {s_4}, edi
004006e1   mov    [rbp-0x8] {s_8}, esi
004006e4   mov    eax, [rbp-0x4] {s_4}
004006e7   sub    eax, [rbp-0x8] {s_8}
004006ea   mov    esi, eax
004006ec   lea    rdi, [0x4008a8]  "Subtraction is %d\n"
004006f3   mov    eax, 0x0
004006f8   call   printf
004006fd   nop
004006fe   leave
004006ff   ret
```

Approach: Pointer Propagation

```
int main()
{
    void (*func_ptr_arr[])(int, int) =
{add, subtract, multiply};

    int loc = 1;
    if(loc >= 0)
        (*func_ptr_arr[loc])(1, 2);
    else
         printf("Bad Loc”)
    return 0; }
```

Results

- Resolving around **25% more** indirect control flow locations on average
- Ability to resolve constant **propagation across functions** and resolved structs in memory
- **Created a Python framework** for extending with more modern resolving methods like Andersen Analysis (future work)

FURI

Ira A. Fulton Schools of Engineering
Arizona State University