Indirect Control Flow Analysis

Discovering possible ICF targets

LIU Xuebao

Institue of Computing Technology

June 9, 2014

Outline

1. What are ICF?
   - Definition
   - Classification
2. Methodology
   - Where to find the possible ICF targets?
   - How to discover the possible ICF targets?
   - How to reduce false negative?
3. Evaluation
4. Conclusion

Outline

1. What are ICF?
   - Definition
   - Classification
2. Methodology
   - Where to find the possible ICF targets?
   - How to discover the possible ICF targets?
   - How to reduce false negative?
3. Evaluation
4. Conclusion

Definition

Definition

***I**ndirect **C**ontrol **F**low* indicates that the targets of control-flow transfers are determined at runtime.

Key point

Instruction pointer is determined at runtime.

Outline

1. What are ICF?
   - Definition
   - Classification
2. Methodology
   - Where to find the possible ICF targets?
   - How to discover the possible ICF targets?
   - How to reduce false negative?
3. Evaluation
4. Conclusion

Classification

- Indirect control-flow transfer Instuructions(ICFTI): CALL and JMP
- Indirect control-flow transfer Functions(ICFTF): setjmp() and sigsetjmp()
- Signal and Interrupt (???)

Outline

1. What are ICF?
   - Definition
   - Classification
2. Methodology
   - Where to find the possible ICF targets?
   - How to discover the possible ICF targets?
   - How to reduce false negative?
3. Evaluation
4. Conclusion

Characteristics of ICFTI targets

- A register or memory pointer specifies the ICF targets.
- Before used as targets, the registers or memories shall be initialized.
- Most of the initializers(targets) could be found in Code Segment or Data Segment directly.
- A few targets could be obtained by simple arithmetic compution on initializers.

Characteristics of ICFTI targets

- A register or memory pointer specifies the ICF targets.
- Before used as targets, the registers or memories shall be initialized.
- Most of the initializers(targets) could be found in Code Segment or Data Segment directly.
- A few targets could be obtained by simple arithmetic compution on initializers.

Characteristics of ICFTI targets

- A register or memory pointer specifies the ICF targets.
- Before used as targets, the registers or memories shall be initialized.
- Most of the initializers(targets) could be found in Code Segment or Data Segment directly.
- A few targets could be obtained by simple arithmetic compution on initializers.

Characteristics of ICFTI targets

- A register or memory pointer specifies the ICF targets.
- Before used as targets, the registers or memories shall be initialized.
- Most of the initializers(targets) could be found in Code Segment or Data Segment directly.
- A few targets could be obtained by simple arithmetic compution on initializers.

Characteristics of ICFTF targets

- Usually, the targets are addresses of the instructions, which are the successors of insturctions *CALL* to setjmp/sigsetjmp.
- How to find the insturctions *CALL* to setjmp/sigsetjmp?

Characteristics of ICFTF targets

- Usually, the targets are addresses of the instructions, which are the successors of insturctions *CALL* to setjmp/sigsetjmp.
- How to find the insturctions *CALL* to setjmp/sigsetjmp?

Outline

1. What are ICF?
   - Definition
   - Classification
2. Methodology
   - Where to find the possible ICF targets?
   - How to discover the possible ICF targets?
   - How to reduce false negative?
3. Evaluation
4. Conclusion

Discover the possible ICF targets

- MOV **imm**, reg/mem — The immidiate is a candidate target.
- LEA **imm(RIP)**, reg — The (immididate+RIP) is a candidate target.
- LEA+ — If X is a candidate target in LEA format, X+MEM(X) is a candidate target.
- data — Any 8-byte numerics, which meet the constrants, is a candidate target.
- rodata — Any 8-byte numerics, which meet the constrants, is a candidate target.
- got, ctros and dtors — Entries in *.got*, *.ctros* and *.dtors* are candidate targets.

Discover the possible ICF targets

- MOV **imm**, reg/mem — The immidiate is a candidate target.
- LEA **imm(RIP)**, reg — The (immididate+RIP) is a candidate target.
- LEA+ — If X is a candidate target in LEA format, X+MEM(X) is a candidate target.
- data — Any 8-byte numerics, which meet the constrants, is a candidate target.
- rodata — Any 8-byte numerics, which meet the constrants, is a candidate target.
- got, ctros and dtors — Entries in *.got*, *.ctros* and *.dtors* are candidate targets.

Discover the possible ICF targets

- MOV **imm**, reg/mem — The immidiate is a candidate target.
- LEA **imm(RIP)**, reg — The (immididate+RIP) is a candidate target.
- LEA+ — If X is a candidate target in LEA format, X+MEM(X) is a candidate target.
- data — Any 8-byte numerics, which meet the constrants, is a candidate target.
- rodata — Any 8-byte numerics, which meet the constrants, is a candidate target.
- got, ctros and dtors — Entries in *.got*, *.ctros* and *.dtors* are candidate targets.

Discover the possible ICF targets

- MOV **imm**, reg/mem — The immidiate is a candidate target.
- LEA **imm(RIP)**, reg — The (immididate+RIP) is a candidate target.
- LEA+ — If X is a candidate target in LEA format, X+MEM(X) is a candidate target.
- data — Any 8-byte numerics, which meet the constrants, is a candidate target.
- rodata — Any 8-byte numerics, which meet the constrants, is a candidate target.
- got, ctros and dtors — Entries in *.got*, *.ctros* and *.dtors* are candidate targets.

Discover the possible ICF targets

- MOV **imm**, reg/mem — The immidiate is a candidate target.
- LEA **imm(RIP)**, reg — The (immididate+RIP) is a candidate target.
- LEA+ — If X is a candidate target in LEA format, X+MEM(X) is a candidate target.
- data — Any 8-byte numerics, which meet the constrants, is a candidate target.
- rodata — Any 8-byte numerics, which meet the constrants, is a candidate target.
- got, ctros and dtors — Entries in *.got*, *.ctros* and *.dtors* are candidate targets.

Category I – MOV

Format

MOV **imm**, reg
MOV **imm**, mem

Example

```
0000000000402a90 <_start>:
    ...
  402a9f: mov     $0x5b4ac0,%r8
  402aa6: mov     $0x5b4ad0,%rcx
  402aad: mov     $0x402bf0,%rdi
  402ab4: callq   402620 <__libc_start_main@plt>
    ...
```

Category II – LEA

Format

LEA **imm(RIP)**, reg

Example

```
   ...
404c3d: lea    0x2e(%rip),%r11  # 404c72
404c44: sub    %rax,%r11
404c47: lea    0xbf(%rsp),%rax
404c4f: jmpq   *%r11
   ...
404c6e: movaps %xmm0,-0x7f(%rax)
404c72: mov    %rsi,%r14
   ...
```

Category III – LEA+

Format

LEA **imm(RIP)**, reg

Example

```
  ...
5af47e: lea    0xeb(%rip),%r11   # 5af570
  ...
5af48c: mov    (%r11,%r9,8),%r10
5af490: lea    (%r10,%r11,1),%r11
5af494: jmpq   *%r11
5af497: mov    (%rdx),%r11b
  ...
5af4bd: mov    (%rdx),%r10w
  ...
```

Category IV – data

Example

```
842b70 e0004500 00000000 70094100 00000000
842b80 f0004500 00000000 f0014500 00000000
```

```
    ...
0000000000410970 <Perl_pp_pushmark>:
  410970: push   %r12
  410972: push   %rbx
  410973: push   %rsi
    ...
```

Category V – rodata

Example

```
5bda60 46864000 00000000 31864000 00000000
5bda70 1c864000 00000000 07864000 00000000
5bda80 f7854000 00000000 00000000 00000000
```

```
   ...
4085f7: mov     0x0(%r13),%rcx
4085fb: mov     %rcx,0x44499e(%rip)
408602: jmpq    40b30c
408607: mov     0x0(%r13),%rdi
40860b: callq   40cd80 <Perl_scope>
408610: mov     %rax,0x444989(%rip)
408617: jmpq    40b30c
40861c: mov     0x0(%r13),%rdi
```

Outline

1. What are ICF?
   - Definition
   - Classification
2. Methodology
   - Where to find the possible ICF targets?
   - How to discover the possible ICF targets?
   - How to reduce false negative?
3. Evaluation
4. Conclusion

Constraints

- Candidate targets (exclude those from got, ctors and dtors) must not point to the destinations outside the current module.
- Candidate targets must not point to destinations inside any one instruction.
- . . .

Constraints

- Candidate targets (exclude those from got, ctors and dtors) must not point to the destinations outside the current module.
- Candidate targets must not point to destinations inside any one instruction.
- . . .

Constraints

- Candidate targets (exclude those from got, ctors and dtors) must not point to the destinations outside the current module.
- Candidate targets must not point to destinations inside any one instruction.
- . . .

Evaluation

- Benchmarks
    - 4 programs of SPEC2006
- Compiler
    - icc
- Platform
    - X86_64 GNU/Linux

Results

	pin	sicfa	same	lost	false
400.perl	1620	5818	1617	3	4201
401.bzip	370	833	370	0	463
403.gcc	3468	9887	3468	0	6419
433.milc	24	527	24	0	503

Conclusion

- A static analysis for indirect control-flow targets is useful.
- Challenges
  - Subset and superset
  - Obfuscation of Executable Code

Conclusion

- A static analysis for indirect control-flow targets is useful.
- Challenges
  - Subset and superset
  - Obfuscation of Executable Code

Conclusion

- A static analysis for indirect control-flow targets is useful.
- Challenges
  - Subset and superset
  - Obfuscation of Executable Code

The End