Software Fault Isolation using the CompCert compiler

Alexandre Dang

Team Celtique

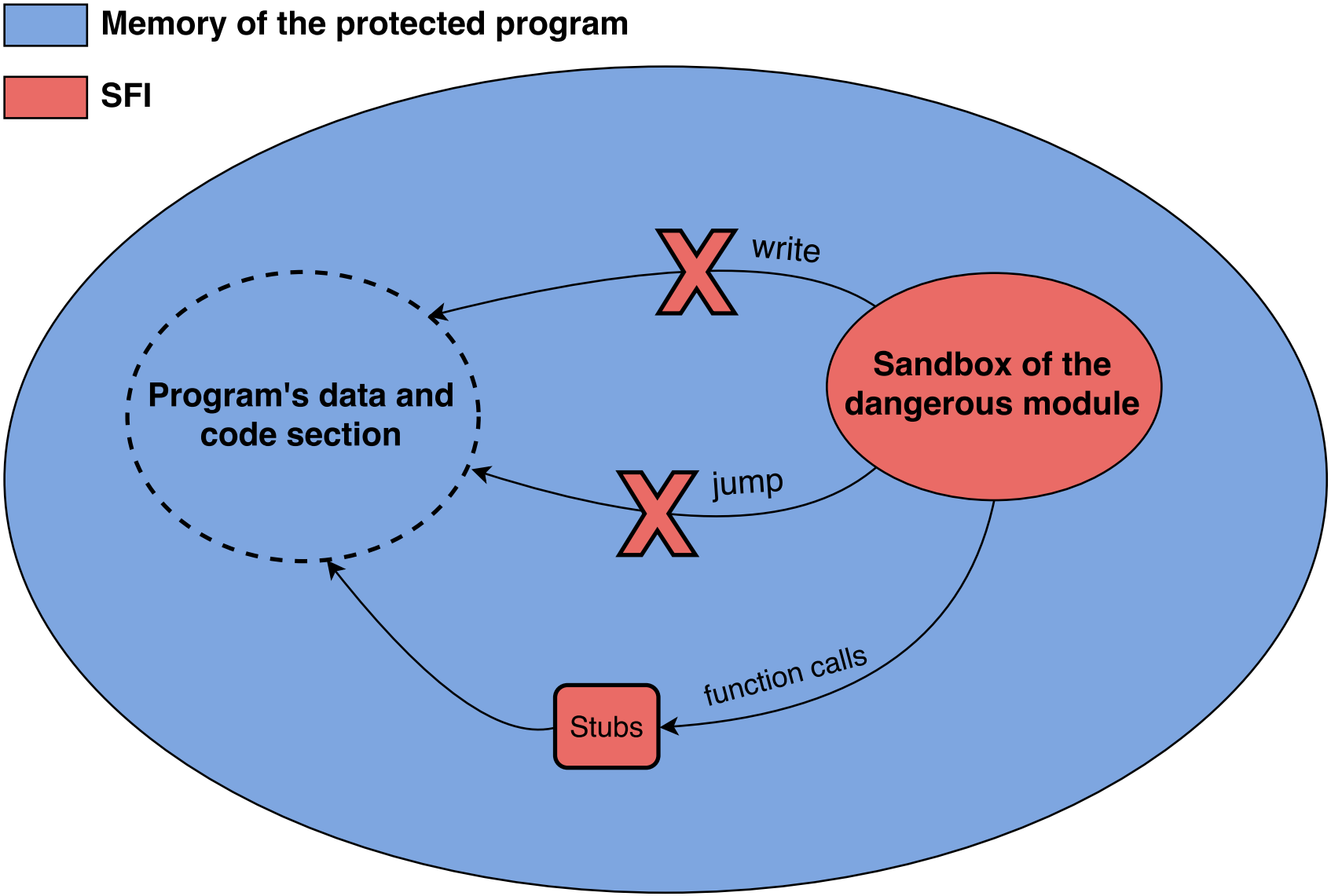
June 6, 2016

Flash sux

Goals of Software Fault Isolation (SFI)

- SFI aims to allow a protected program to execute dangerous modules in its own memory space without dangers.
- SFI confines the execution of the dangerous modules in a reserved area called sandbox
- `jump` and `write` instructions are protected by runtime checks
- function calls to the protected programs are controlled by SFI

Goals of SFI


Memory of the protected program
SFI
write
Program's data and code section
Sandbox of the dangerous module
jump
function calls
Stubs

Overview of SFI

SFI chain is composed of two elements: a generator and a verifier

- the generator transforms the assembly code of the dangerous modules in order to confine the modules in their sandbox
- the verifier checks that the SFI transformations are present and valid before loading the code in memory

Sandboxing

Sandbox are continuous area identified by a tag For example the sandbox [`0xda000000` - `0xdaffffff`] has the tag `0xda`

NativeClient

Google im

SFI for CompCert

Advantages of SFI

Problematics of SFI

Return Oriented Programing attacks

Example (1/2)

```
1  void evil_code() {
2  printf("Argh, we got hacked!\n");
3  }
4
5  void foo(char* input){
6  char buf[1];
7  ... code ...
8  strcpy(buf, input);
9  ... code ...
10 }
```

Example (2/2)

terminal$ ./buffer $(python -c 'print
13*"a"+"\x7b\x84\x04\x08"')
Address of evil_code = 0x0804847b
Stack before:
0xf7712000
...
0xf77828f8
0xff957998
0x08048510 //Return address of *foo*

Stack after :
0xff958161

0xff957978 //Buffer overflow
0x61593d00 //"a"
0x61616161 //"aaaa"
0x61616161 //"aaaa"
0x61616161 //"aaaa"

Modern ROP attacks

Goals of our approach

CompCert stack

Transformations of the stack layout

Injection of runtime checks

Conditions of our approach

Discussion of the approach

Evaluation of security

Evaluation of performance

Discussion