CS326 – Systems Security

Lecture 16

Program Analysis and Applications

Elias Athanasopoulos
athanasopoulos.elias@ucy.ac.cy

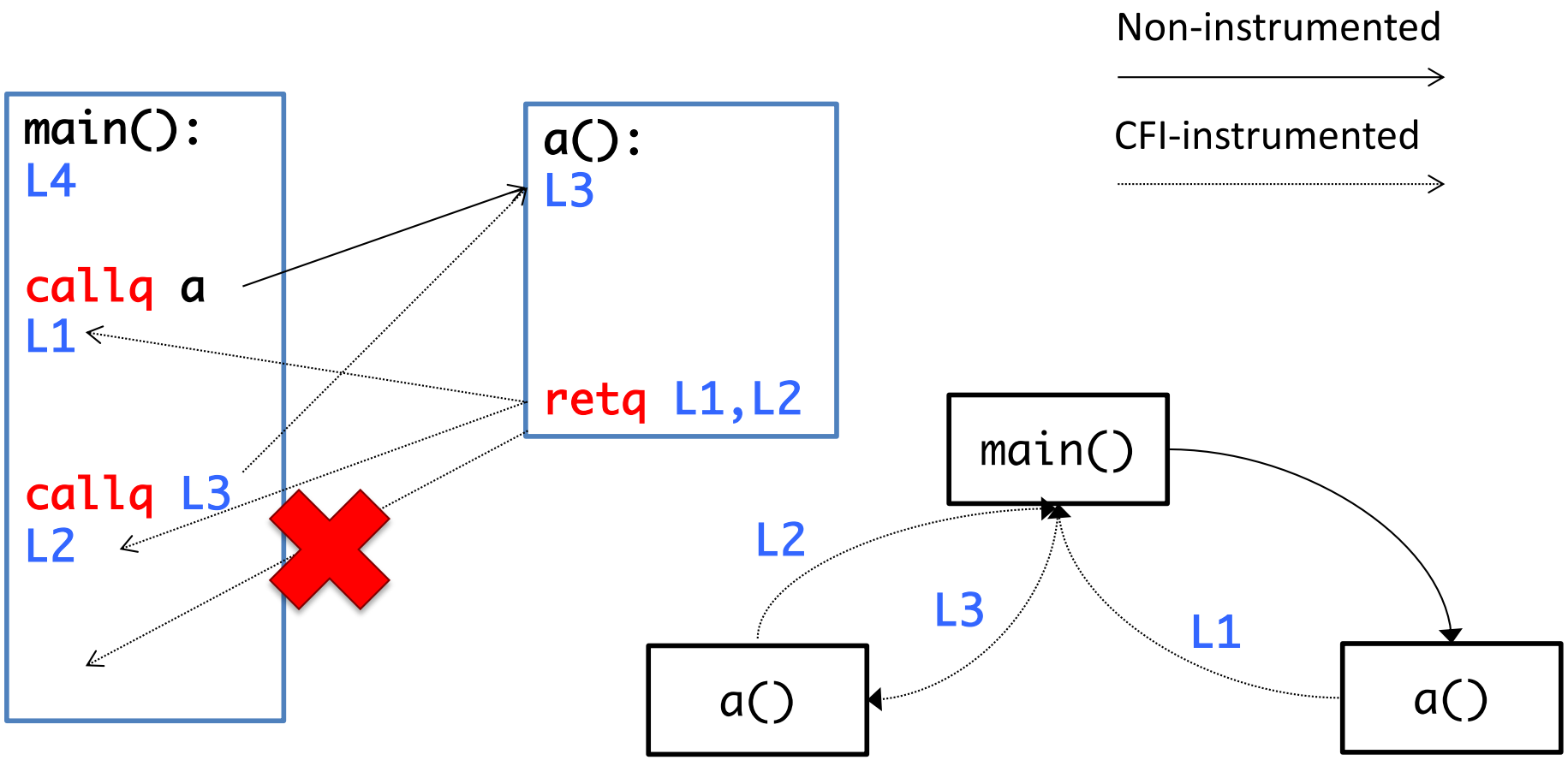
Modern Software Hardening

- Techniques for defending software
  - Against an attacker with arbitrary read/write capabilities
- Can be applied directly to binaries
  - When source is not available
  - Legacy software
- Can be applied to source code
  - Needs software re-compilation
- Performance overheads

Instrumentation
Stack Canaries

No stack canaries	Stack canaries
leave ret	**mov -0xc(%ebp),%eax** **xor %gs:0x14,%eax** **je 96 <authenticate_root+0x96>** **call 92 <authenticate_root+0x92>** leave ret

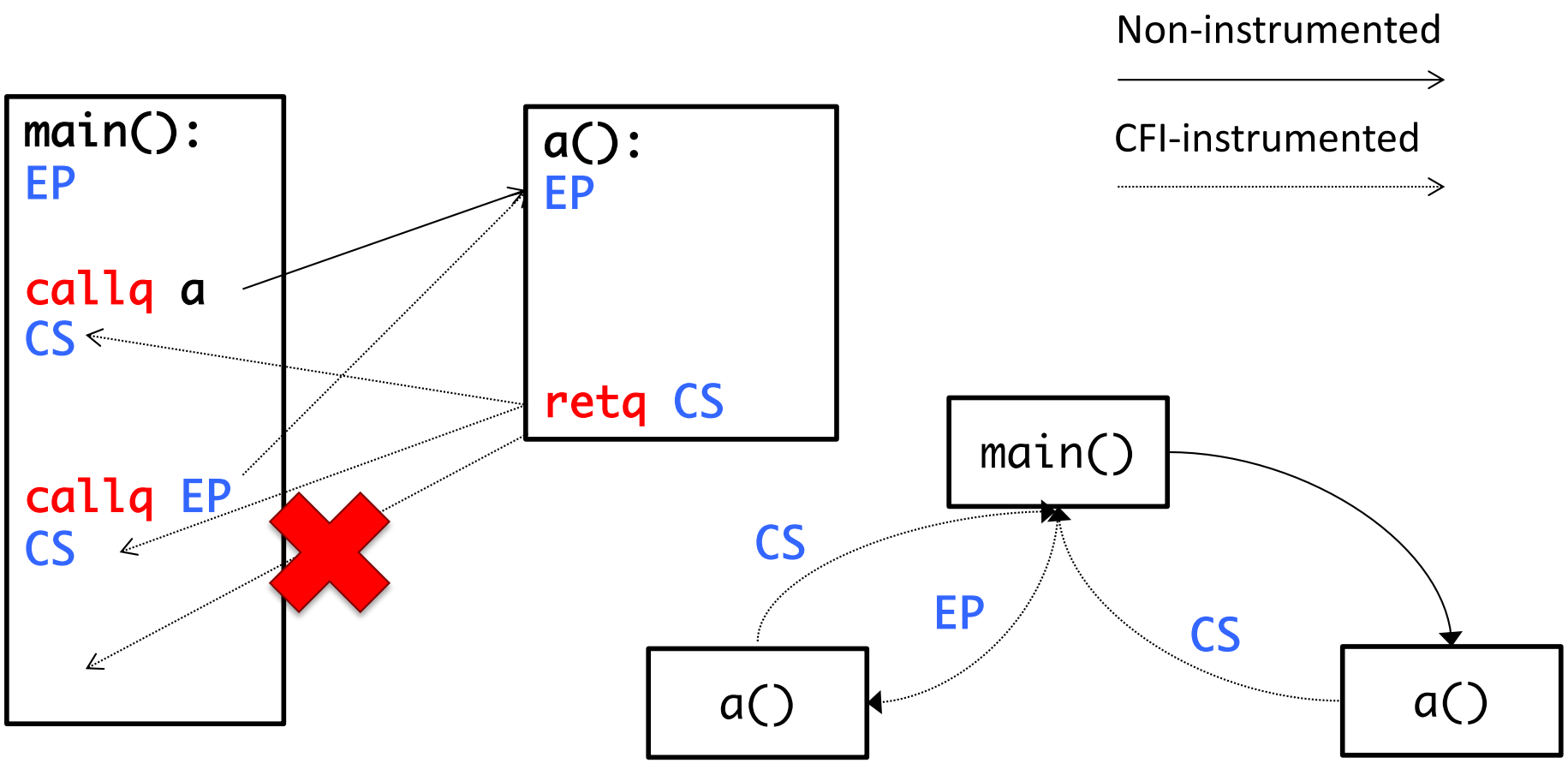
Control-flow Integrity (CFI)



CFI Problems

- Hard to compute perfect Control-flow Graph
  - No source code
  - Loadable Modules
  - Dynamic code
- Performance

Coarse-grained CFI (2 labels)



Deployed CFI

- Supported by modern compilers
  - Control-flow Guard (CFG), Microsoft
  - VTV, LLVM/Google
- Hardware support
  - Intel has announced hardware instructions for CFI
  - Shadow stacks (return addresses are stored in a separate h/w memory)

Instrumentation
CFI

Target	Destination
`jmp ecx ; computed jump`	`mov eax, [esp+4] ; dst` `...`

Target	Destination

`; comp ID/dst`
`cmp [ecx], 12345678h`
`; if != fail`
`jne error_label`
`; skip ID at dst`
`lea ecx, [ecx+4]`
`; jump to dst`
`jmp ecx`
	`78 56 34 12 ; data 12345678h` `;ID`
	`mov eax, [esp+4]  ; dst ...`

Program Analysis

- Static
  - Before execution
  - Source-based or binary-based
  - No adaptation to particular inputs
- Dynamic
  - During execution
  - Adapt to certain inputs

Static Analysis

- Analyze the source of the program without executing it (no inputs)
  - Source can be in high-level language (C/C++)
  - Source can be bytecode
  - Source can be machine code (*binary analysis*)

Source-based static analysis

- Usually performed at the compiler-level
- LLVM (Low Level Virtual Machine)
  - Compiler infrastructure that allows to add custom passes
  - All phases in the compilation are represented using LLVM IR (*intermediate representation*)

Binary Analysis

- Disassemble a binary for analysis
  - Open problem for x86
  - Variable-length instructions
  - Data are mixed with code
- Recursive disassembly
  - Follow jumps and disassemble targets
- Linear disassembly
  - Linearly disassemble code
- Suggested read
  - https://syssec.flux.re/papers/sec-2016.pdf

Tools

- `objdump -d`
  - Simple tool for disassembling binaries in Linux, part of binutils
- `otool`
  - Simple tool for disassembling binaries in OSX
- IDA Pro
  - Commercial and powerful disassembler

Dynamic Analysis

- Analyze program while executing
  - Usually, slow
  - Analysis observes actual inputs
- The analysis' code runs *in parallel* with the program's code

Tools

- gdb
  - A debugger which is attached to the program and can perform various tasks (breakpoints, step instruction, inspect memory, etc.)
- PinTool
  - Intel framework for dynamically instrumenting binaries
  - A *pintool* is attached to the analyzed program
  - The *pintool* can execute instructions, account for the program's instructions, inspect memory accesses, etc.

Applications

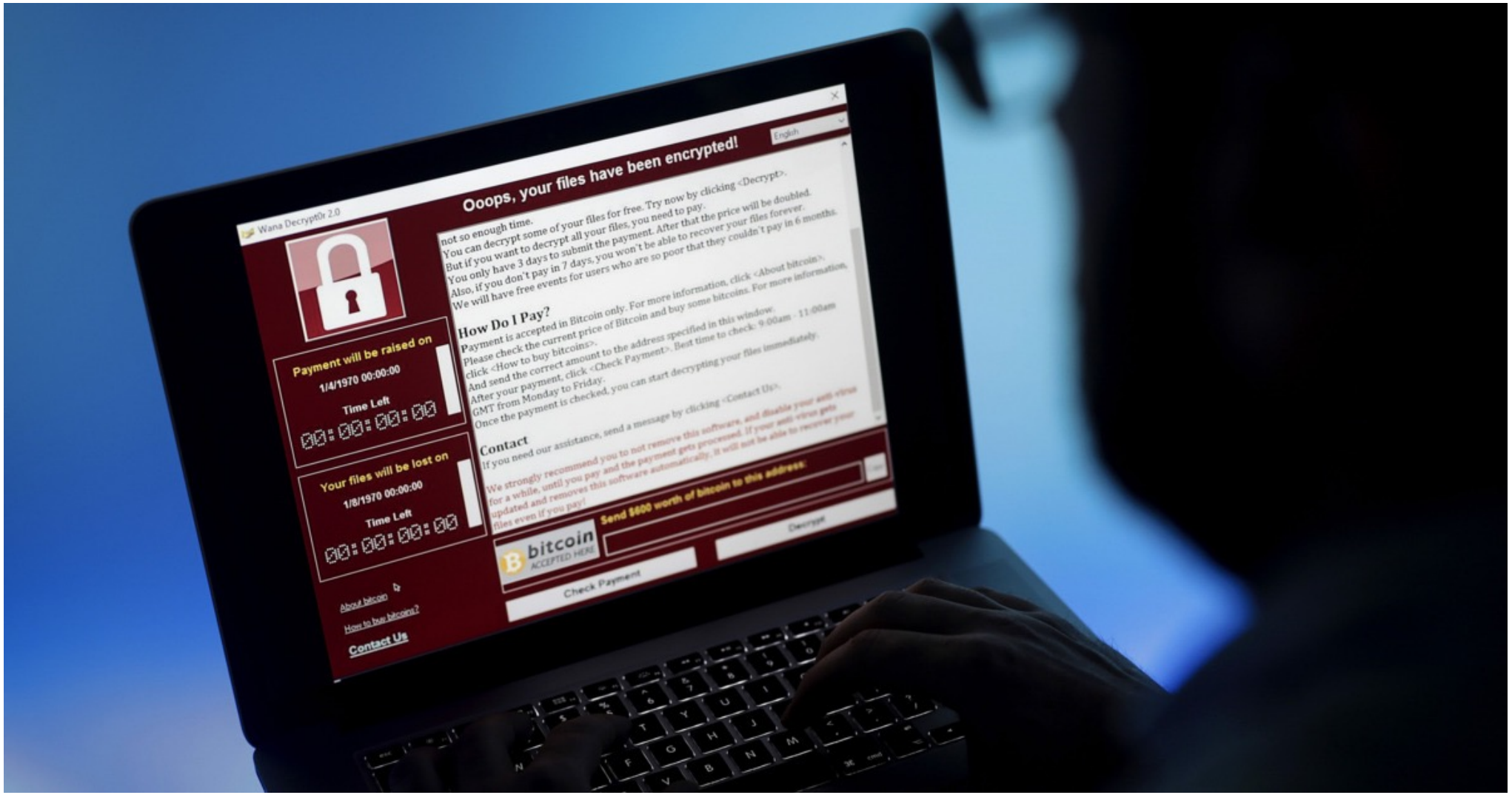
- Program instrumentation
  - Software hardening
- Bug finding
  - Assisted *fuzzing*
- Malware identification
  - Check if a downloaded program is malware or not

Binary Preloading

- Dynamically linked binaries
  - Code is loaded at run-time using the dynamic loader
  - `ld-linux.so` for Linux
- A symbol can be in several libraries
  - The dynamic loader uses the first found one
- We can *hook* code in library calls
  - As long as we load our code first

Example
Hooking `malloc()`

- Create a shared library with a custom `malloc()`
- Use LD_PRELOAD to load the custom library first
- Inside the custom `malloc()` you can load the *real* `malloc()`
- The custom `malloc()` can do some work and then run the *real* `malloc()`


Wana Decrypt0r 2.0
Ooops, your files have been encrypted!
English
not so enough time.
You can decrypt some of your files for free. Try now by clicking <Decrypt>.
But if you want to decrypt all your files, you need to pay.
You only have 3 days to submit the payment. After that the price will be doubled.
Also, if you don't pay in 7 days, you won't be able to recover your files forever.
We will have free events for users who are so poor that they couldn't pay in 6 months.
How Do I Pay?
Payment is accepted in Bitcoin only. For more information, click <About bitcoin>.
Please check the current price of Bitcoin and buy some bitcoins. For more information,
click <How to buy bitcoins>.
And send the correct amount to the address specified in this window.
After your payment, click <Check Payment>. Best time to check: 9:00am - 11:00am
GMT from Monday to Friday.
Once the payment is checked, you can start decrypting your files immediately.
Contact
If you need our assistance, send a message by clicking <Contact Us>.
We strongly recommend you to not remove this software, and disable your anti-virus
for a while, until you pay and the payment gets processed. If your anti-virus gets
updated and removes this software automatically, it will not be able to recover your
files even if you pay!
Payment will be raised on
1/4/1970 00:00:00
Time Left
Your files will be lost on
1/8/1970 00:00:00
Time Left
Send $600 worth of bitcoin to this address:
bitcoin
ACCEPTED HERE
Check Payment
Decrypt
About bitcoin
How to buy bitcoins?
Contact Us