

Lecture 2 Program Analysis

CTI	F crunch	Computer and Network Security October 7, 2019		
		Computer Science and Engineering Departmen	Ł	
CSE Dep, ACS, UPB		Lecture 2, Program Analysis	1/79	
				Notes
				INOTES
Prog	ram Analysis			
CSE Dep. ACS, UPB		Lecture 2, Program Analysis	2/79	
CNS		Program	Analysis	Notes
	automatic analys	is of programs		
	property verificat			
		formance) or correctness		
>	static analysis or	dynamic analysis		
CSE Dep, ACS, UPB		Lecture 2, Program Analysis	3/79	
CNS Ò	1	Program	n Model	
OTF orunor		1 105.41		Notes
>	automaton			
•	control flow grap	h (CFG) (set of states and transitions)		
▶	coverage: how m	uch of the CFG can the analysis cover t	0	
	ensure property v	alidation		

Notes

CNS	Static and Dynamic Analysis
► do not execute or e	xecute the program
static analysis on so (executable)	ource code or on binary program
dynamic analysis orsymbolic execution	resource usage and behavior (process)
► fuzzing is dynamic	-
•	ad, may go into path explosion lepth, may miss certain cases
dynamic unarysis. d	epin, may miss certain cases
CSE Dep, ACS, UPB	Lecture 2, Program Andysis 5/79
CNS⊖	Source Code vs Executable
OTF Grunch	
extensive analysis owe don't know wha	n source code but t the compiler / linker does to it, what
optimizations happe it may not be availa	en, how it links to other components
we focus most on s	
CSE Dep, ACS, UPB	Lecture 2, Program Analysis 6/79
CNS⊖	Challenges of Static Binary Analysis
CTF crunch	
	derstand: requires reverse engineering
may be subject to onetypically doubled by	obfuscation, encryption, packing
., p	
CSE Dep, ACS, UPB	Lecture 2, Program Analysis 7/79
CNS⊖	Process as a Goal
OTF orunch	

Notes

Notes			
Notes			

▶ allocate and use memory and other resources

▶ provide functionality ▶ dynamic / run time

	and assemble source code into object f ect files into executable	riles	
3. load exe	ecutable (disk image file) into process (r	memory +	
CPU)			
CSE Dep, ACS, UPB	Lecture 2, Program Analysis	10/79	
CNS		Object File	Notes
▶ binary fi	les		
► headers	and binary code		
▶ may be ▶ data and	disassembled d. code		
▶ sections			
CSE Dep. ACS, UPB	Lecture 2, Program Analysis	11/79	
CNS⊖		Library Files	Notes
ore cronon			Notes
			-
► susting /	Carlle attended a fitte at Class		
➤ arcnive/ ➤ modular	collection of object files ity		
	nking and dynamic linking libraries ing happens at link time		
▶ link	ing happens at load time		
CSE Dep. ACS, UPB	Lecture 2, Program Analysis	12/79	
CNS⊖		Executable Files	Notes
CTF crunch			Notes
► binary fi			
	o object files, consist of object code disassembled		
► created	from object files		
static ar	nd dynamic executables		

CSE Dep, ACS, UPB Lecture 2, Program Analysis 13/70

▶ dynamic: library stubs to library functions

Notes

 $\verb|http://www.roman10.net/2012/11/28/an-intro-to-elf-file-formatpart-1-file-types-and-dual-views/all-file-formatpart-1-file-types-and-dual-views/all-file-formatpart-1-file-types-and-dual-views/all-file-formatpart-1-file-formatp$

CSE Dep, ACS, UPB

Lecture 2, Program Analys

14/79

CNS

Object File Format

- format of a file that contains object code: object file, executable files, dynamic-linking library files
- ▶ headers, sections
- ► data and code
- ▶ may be disassembled
- ▶ PE (Portable Executable) on Windows
- ► COFF (Common Object File Format) on Unix
- ► ELF (Executable and Linking Format) on Linux

CSE Dep. ACS, UPB

Lecture 2, Program Analysi

16/79

CNS

Common Information in Executabile Files

- ► entry point
- program addresses (section addresses)
- section sizes
- symbols (names and addresses)
- permissions

CSE Dep, ACS, UPB

Lecture 2, Program Analysis

17/79

CNS

ELF Format

- ▶ header
- program headers
- sections
- ▶ segments
- ▶ symbols
- ightharpoonup readelf, objdump, nm

Notes			
Notes			
Notes			

CSE Dep, ACS, UPB

Lecture 2, Program Analysis

18/79

_		~~
r	N	
u	ıv	\mathbf{u}

Sections

Notes

- ▶ storing data or code
- ▶ readelf -S program
- .text, .data, .bss
- .symtab, .strtab

CSE Dep. ACS, UPE

Lecture 2, Program Analysi

10/70

CNS

Sections vs. Segments

- ▶ segments contain 0 ore more sections
- sections are used by linker, some sections may be ditched at runtime
- segments are used by the operating system (loaded into memory)

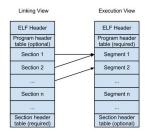
CSE Dep, ACS, UPI

Lecture 2, Program Analysi

20/79

CNS

View of ELF File



http://www.roman10.net/2012/11/28/an-intro-to-elf-file-formatpart-1-file-types-and-dual-views/

CSE Dep, ACS, UPB

Lecture 2, Program Analysis

21/79

CNS

Symbols

- ▶ readelf -s program
- .dynsym and .symtab
- ▶ name, value, type, bind, size

Notes		
Notes		
Notes		

CSE Dep, ACS, UPE

Lecture 2, Program Analysis

22/79

CNS	Debugging Symbols	Notes
		TVOCCS
Man Assaulth instructions to	iabla firmation on line in the	
 Map Assembly instructions to source code 	variable, function or line in the	
► Help mapping stack values wi	th function parameters	
Optimize data flow analysis		
Optimize static and dynamic a	analysis	
On Linux, symbol table is eml	bedded in the ELF file. PE files	
use an external symbols file		
CSE Dep, ACS, UPB Lecture 2	2, Program Analysis 23/79	
CNS⊖	Stripping	
CTF crunch	Этпрриів	Notes
Removing symbol table from	program executable	
 Complicates reverse engineering 		
 Less space used by original bit 		
Less space used by original bil	iai y	_
CSE Dep, ACS, UPB Lecture 2	24/79 24/79	
CNSÒ	Overview of Linking	
OTF crunch		Notes
N. All altitude Classics Colored to act	thanks and the control of the	
► All object files are linked toget		
Input: Object files, static libra	iries, dynamic libraries	
Output: Executable image		
The linker resolved external re	eferences from each object file	
		-
CSE Dep, ACS, UPB Lecture 2	2, Program Analysis 26/79	
CNS⊖	Using 1d	
CTF crunch -		Notes

► Command used in the last compiling phase

▶ Libraries are specified using -1 option

▶ PIE option enables ASLR support

NS 🗟		Static Linking	Notes	
► Executable is more	routines directly into exec portable because all data r			
execute resides in t Faster execution be	he file cause imports are not reso	lved at runtime		
► Uses more space	sause impente une net rese			
p, ACS, UPB	Lecture 2, Program Analysis	28/79	_	
ıs ò		Tools of Trade		
orunch -			Notes	
building machine co				
inspecting machine	code files			
	code files			
inspecting machine	code files			
inspecting machine	code files			
inspecting machine	code files			
inspecting machine	code files			
inspecting machine	code files	30/79		
▶ inspecting machine▶ disassembling mach	code files iine code files	20/79		
 ▶ inspecting machine ▶ disassembling mach 	code files ine code files Lecture 2. Program Analysis			
▶ inspecting machine▶ disassembling mach	code files ine code files Lecture 2. Program Analysis	ilding Executables	Notes	
inspecting machine inspecting machine disassembling mach	code files ine code files Lecture 2. Program Analysis		Notes	
inspecting machine inspecting machine disassembling mach	code files ine code files Lecture 2. Program Analysis		Notes	
inspecting machine inspecting machine disassembling mach	code files ine code files Lecture 2. Program Analysis		Notes	
inspecting machine inspecting machine disassembling mach	code files ine code files Lecture 2. Program Analysis Bu		Notes	
inspecting machine inspecting machine disassembling mach	code files ine code files Lecture 2. Program Analysis Bu		Notes	
inspecting machine inspecting machine disassembling mach	code files ine code files Lecture 2. Program Analysis Bu		Notes	
inspecting machine inspecting machine disassembling machine ACS. UPB SS COURTS COURTS	code files ine code files Lecture 2. Program Analysis Bu		Notes	
inspecting machine inspecting machine disassembling mach	code files ine code files Lecture 2. Program Analysis Bu		Notes	
inspecting machine inspecting machine disassembling machine ACS. UPB SS COURTS COURTS	code files ine code files Lecture 2. Program Analysis Bu		Notes	

ELF Inspection

▶ strings

► xxd

► readelf

▶ nm

Lecture 2, Program Analysis 32/79

Notes		
Notes		
Notes		

CNS	Disassembling	Notes
017 01011011		Notes
		-
► IDA		
▶ objdump		
► radare2		
CSE Dep, ACS, UPB Lecture 2, Program Analy	ysis 33/79	
CNSO CTF or unoh	Not for Static Analysis	Notes
pmaplsof		
▶ ltrace		
▶ strace		
► GDB		
CSE Dep, ACS, UPB Lecture 2, Program Analy	rsis 34/79	
CNS⊖	Dynamic Analysis	
CTF crunch	Dynamic Analysis	Notes
► starts from executable files		-
► investigate processes		
requires process to runruntime analysis		
blackbox analysis		
CSE Dep, ACS, UPB Lecture 2, Program Analy	psis 36/79	
CNSÒ	Processes	
CTF crunch -		Notes
unit of work in the operating system	1	
virtual memory address space, thread		
isolated from each other		
▶ at load time the executable gives bi	rth to a process	

CSE Dep., ACS, UPB Lecture 2, Program Analysis 37/79

(3	1	٧	•	S	ţ	_)	

Process Memory Layout

	Higher memory address	
	System	
	MILY ALGO ALGO	main() fintee pointer
Stack Stack	Auto variables for main()	(ESP) DIESE POUNE
Ē	Auto variable for func()	Stack pointer (ESP), points at the top of the stack -power decreased
	Available for stack growth	tale same reports terrorized
Shared	malloc.o (lib*.so)	Library functions of dramagenily insted
agri Tipe	printf.o (lib*.so)	strani case
	Available for heap growth	box () point
	Neap (mailor(), callor(), men)	
sata	Global variables	Uninitialized data - box
-8	int y = 100;	Installed data - data
	malloc.o (lib*.a)	Libeary functions of stationary functions of
0.0	printf.o (lib*.e)	COMP
Text (Compiled code, a.out)	file.o	
	main.o funci)	◆ The return addrso:
-	crt0.c (startup routine)	

http://www.tenouk.com/Bufferoverflowc/Bufferoverflow1_files/image022.png

CSE Dep, ACS, UP

ecture 2. Program Analysis

38/79

CNS

Interesting Process Information

- ▶ the process memory map (virtual memory areas)
- memory addresses: code, variables
- memory region access rights
- machine code (to be disassembled)
- process state: registers, (call) stack, code

CSE Dep, ACS, UP

Lecture 2, Program Analysis

39/79

CNS♦

Why Dynamic Analysis

- ▶ get output for input (blackbox)
- $\,\blacktriangleright\,$ glimpse into the internals
- ► monitor/inspect resource usage
- $\,\blacktriangleright\,$ debug execution and test attacks (step by step)

CSE Dep, ACS, UPB

Lecture 2, Program Analysis

40/79

CNS

What Do We Investigate?

- code: system calls, library calls, function calls, step-by-step code
- ▶ state: thread information, process maps, open files, resources
- data: registers, variables, raw memory data

Notes			
Notes			
Notes			

CSE Dep., ACS, UPB Lecture 2, Program Analysis

Types of Tools

- blackbox inspection: function call tracers (strace, ltrace, $\mathsf{dtrace}/\mathsf{dtruss}),\ \mathsf{fuzzers}$
- ▶ profilers: most often for performance: perf, callgrind, vTune
- debugging: GDB, LLDB, valgrind

Notes

CNS	Fuzzing	Notes
generate "random" input and detect program flaws		
program is run		
smart fuzzer try to directAFL, libfuzzer		
7.1.2, 110.10.22.01		
CSE Dep, ACS, UPB Lecture 2, Program Analysis	47/79	
CNS stra	ace/Itrace	Notes
▶ strace ./a.out		
strace -e read, write ./a.outstrace -e file ./a.out		
▶ strace -e file -f ./a.out		
strace -e file -s 512 -f ./a.outsimilar options for ltrace		
CSE Dep., ACS, UPB Lecture 2, Program Analysis	48/79	
CNS 🖯	sof/pmap	Notes
▶ PID as argument▶ 1sof -p 12345		
▶ pmap 12345		
CSE Dep. ACS, UPB Lecture 2, Program Analytis	49/79	
CNS ÷	perf	Notes

default profiler on Linux

▶ sampling profiler, doesn't instrument the code

uses events sampling

▶ perf stat -e cache-misses -a ./mem-walk

▶ sudo perf list

 $\,\blacktriangleright\,$ some actions and events may require privileged access

SE Dep, ACS, UPB Lecture 2, Program Analysis 50/

CNS Ò	GDB	Notes
		Notes
Adfault debugger on CNIII/I in w. dietzibutions		
 default debugger on GNU/Linux distributions command line; there are some GUI front-ends 		
▶ incorporated in Linux-based IDEs		
debugging, dynamic analysis / process investigation		
▶ gdb ./a.out▶ gdb -q ./a.out		
3 - 1 - 1 - 1 - 1 - 1 - 1 - 1 - 1 - 1 -		
CSE Dsp. ACS, UPB Lecture 2, Program Analysis	51/79	
CNS OTF OTUNON	LLDB	Notes
► LLVM Debugger		
▶ used on Mac OS X		
► similar features to GDB		
command line; most commands are equivalent to GDBhttp://lldb.llvm.org/lldb-gdb.html		
CSE Dep. ACS, UPB Lecture 2, Program Analysis	52/79	
CNS Hardware Support for D	ebugging	Notes
OTF OTBION		Notes
useful for debugging embedded devices		
 JTAG: Joint Test Action Group uses dedicated debug port 		
► Lauterbach Trace32: in circuit debugger (device using J	TAG)	
		_
CSE Dep. ACS, UPB Lecture 2, Program Analysis	53/79	
CNS GDB and	Security	
GDB and	Jecurity	Notes

▶ not just for debugging

▶ follow what a process does (step instructions)

▶ inspect data (memory, registers)

CSE Dep., ACS, UPB Lecture 2, Program Analysis 55/7

OTF crunch	Notes
process state inspection	
register inspection	
► (machine) code inspection	- <u></u>
memory inspection	
memory alteration	
► function call tracing	
CSE Dep. ACS, UPB Lecture 2, Program Analysis 56/70	
CNS GDB Basics	
CTF crunch —	Notes
▶ starting a process	
stepping instructions	
► breakpoints	
disassemble	
▶ show registers	
▶ display data	
▶ trace function calls	
▶ alter data	
CSE Dep, ACS, UPB Lecture 2, Program Analysis 57/79	
C: D	
CNS Starting a Process	Notes
▶ run	
run < input file	
run arg1 arg2 arg3	
set args arg1 arg2 arg3 and then issue run	
start: breakpoint at main / starting point	
7 6	
CSE Dep. ACS, UPB Lecture 2, Program Analysis 58/79	
CNS Stepping Instructions	Notes
vir grungn -	NOTES
▶ si and ni	
▶ ni doesn't go into nested functions	
very useful for understanding programs and validating attacks	

CSE Dep, ACS, UPB Lecture 2, Program Analysis 59/76

b -	h	evmhol-nama

- ▶ b *address: b *0x80123456
- ▶ continue: continue until the next breakpoint
- ▶ help breakpoints

CSE Dep. ACS, UF

Lecture 2, Program Analysi

60/70

CNS

Disassembling

- ► during runtime
- ▶ disass symbol-name: disass printf
- ▶ help disassemble

CSE Dep, ACS, UPB

Lecture 2, Program Analysis

61/79

CNS

Displaying Data

- $\,\blacktriangleright\,$ show memory data or registers
- ▶ info registers
- ▶ p \$eax
- ▶ p *0x80123456
- x/10x 0x12345678: examine memory and display in hex
- ▶ x/10s 0x12345678: examine memory and display in string
- x/10i 0x12345678: examine memory and display in instructions
- ▶ help p
- ▶ help x

CSE Dep, ACS, UPB

Lecture 2, Program Analysi

62/79

CNS

Find Data in Memory

- ▶ find "sh"
- ightharpoonup find 0x01020304
- ▶ find 0x400000, 100000, "sh"

NI .	
Notes	
	_
Notes	
	-
Notes	

CSE Dep. ACS, UPB Lecture 2, Program Analysis

CNS		Trace Function Calls	Notes
>	backtrace: show function trace		
>	up, down: update current call stack		
•	http://web.mit.edu/gnu/doc/html/gd	lb_8.html	
CSE Dep, ACS, UPB	Lecture 2, Program Analysis	64/79	
CNS		Alter Data	Notes
	set variable num = 10		
	set {int}0x8038290 = 10 set \$eax = 0x12345678		
CSE Dep, ACS, UPB	Lecture 2, Program Analysis	65/79	
CNS		PEDA	Notes
•	Python Exploit Development Assistance enhancement for GDB		
>	create cyclic patterns Return Oriented Programming features		
>	custom view: code, registers, stack shellcode features telescope an address (follow pointers)		
	(
CSE Dep, ACS, UPB	Lecture 2, Program Analysis	06/70	
CNS	Times in Getting from So	ource Code to Process	Notes

- ▶ compile time: when translating source code to object code in
- $\,\blacktriangleright\,$ link time: when aggregating multiple object files into an
- is created (using ./program)

object files (using gcc, gas, nasm) executable file (using gcc, ld) $\,\blacktriangleright\,$ load time: when executable is loaded in memory and a process $\,\blacktriangleright\,$ run time: while the process is running (using strace -p, lsof -p)

\sim	k i	07
U	IN	30

Linking and Loading

▶	linking is getting object	files	together	into	an	executable	or
	dynamic-linking file						

- ▶ for the linker, object files are input and executables are output
- ▶ loading is getting an executable into memory and starting a process
- $\,\blacktriangleright\,$ for the loader, executable file is input, process is output

CSE Dep. ACS. U

Lecture 2, Program Analysi

60/70

CNS

Static linking

- ▶ all symols are solved at link time
- ▶ all code is part of the executable
- static executables
- ▶ large executable files, but with no dependencies, highly portable

CSE Dep, ACS, UPI

Lecture 2, Program Analysi

70/79

CNS

Load Time Dynamic Linking

- ▶ symbols are marked as stubs inside the executable file
- symbols are solved at load time, the moment the process is created
- symbols are picked from dynamic-linking library files
- provides reduced size executable files but requires dependencies to be satisfied

CSE Dep. ACS, UPB

Lecture 2, Program Analysis

71/79



Run Time Dynamic Linking

- ▶ linking (and loading) is done at runtime
- lacktriangle it may be implicit (lazy binding) or explicit
- dlopen, dlsym for the explicit case: explicitly load a library and locate a symbol

Notes	Notes		
Notes			
	Notes		
	Notes		
Notes			
	Notes		

CSE Dep, ACS, UPB Lecture 2, Program Analysis 72/

▶	postpone	linking	of a	symbol	until	it	is	called

- usually done for functions through the use of a trampoline section (PLT for ELF)
- ▶ the first time a function is called, the dynamic linker also does the binding

Locating Libraries

- ▶ for stating linking, use the -L argument to gcc
- ▶ for dynamic linking, the dynamic linker/loader is used: ld-linux.so
- ▶ man ld-linux.so
- searches for
 - 1. values in LD_LIBRARY_PATH
 - the /etc/ld.so.cache file; populated by ldconfig
 the default /lib and /usr/lib library folders

CNS

PLT

- ▶ used for external library function calls
- generic trampoline code to jump to initially jump to per-function binder (.plt in ELF)
- writable data area storing function pointers (.got.plt)
 - initially store pointers to binder code (symbol solver)
 - ▶ after the first call store actual pointer to function call

CNS

GOT

- ► Global Offset Table
- .got in ELF for global variables
- .got.plt in ELF for external library function pointers
- local uses of external library symbol point to GOT
- $\,\blacktriangleright\,$ GOT if filled by the dynamic linker at the beginning

Notes		
Notes		
Notes		

	- 1 <i>C</i> /
 static analysis dynamic analysis executable ELF readelf section segment disassembling objdump symbols linker process 	<pre>▶ Isof / pmap ▶ perf ▶ GDB ▶ breakpoint ▶ info ▶ examine ▶ ni, si ▶ backtrace, up, down ▶ write ▶ searchmem ▶ dynamic linking ▶ dynamic loading ▶ lazy binding ▶ trampoline</pre>
cse by Res Strace / Itrace	Lecture 2, Program Analysis PLT GOT

Useful Links

- http://www.skyfree.org/linux/references/ELF_Format.pdf
- http://ith.gnu.org/old-gnu/Manuals/1d-2.9.1/html_node/ld_3.html
 https://msdn.microsoft.com/en-us/library/windows/desktop/
 ee416588(v=vs.85).aspx
- https://www.technovelty.org/linux/ plt-and-got-the-key-to-code-sharing-and-dynamic-libraries.html

Notes	
	 -
Notes	
Notes	
Notes	