Compact Control Flow Integerity in Linux

Introduction

The system supported stack plays an important role in running any application. If someone has the control of the stack then he/she has the control over the flow of the program. This is possible when some malicious input is given to the program and the attacker then redirects the control flow to a harmfull code segment. We intend to prevent this by modifying the binary therefore fortifying it.

Attack

The stack becomes vulnerable whenever a function is called either directly or indirectly.

- (*i*) *Indirect Call* :- The control flow is vulnerable during both call and return.
- *(ii) Direct Call:-* The control flow is vulnerable only during return.

Everytime when a program returns from a function, the return address is read from the stack. Whenever a function is called indirectly, the function address is read from the registers which is inturn read from the stack (happens in case of function pointers). The attacker can give some malicious inputs that overwrites the target adresses in the stack thus gaining power over the control flow.

How do we fortify?

This problem can be solved by clubing all the targets into a single place (which the attacker does not have access to) and redirecting the calls here. We call this address space **the springboard**.

We add a new section called the springboard and all the function calls are made from here and the original call instructions are replaced by the jmp instruction that branch to the corresponding springboard target.

Major steps in fortification

- (*i*) Find all the calls and replace it with jumps that branches to the springboard along with adding call instuctions followed by jmp instructions that jumps back to the original calling address.
- *(ii)* Insert checkpoints before every return statement that checks if the return address lies within the springboard section.

Technical details

Adding a Section to a binary:

We can use the *readelf* --sections command to print the sections of a binary.

vignesh	n@vignes	h-Precision-Tower-581	0: ~/Desktop/ADSCde	emo		1∎ En ≪ 17:06 🕸
		cd: Desktop: No su				
(O)		h@vignesh-Precisio are 31 section hea			adelfsections add	
	mere	are si sección nea	ders, starting at	offset oxiaso:		
	Sectio	n Headers:				
	[Nr]	Name	Туре	Address	Offset	
	[0]	Size	EntSize NULL	Flags Link Info		
	[0]	000000000000000000000000000000000000000	0000000000000000000	0 0 0		
	[1]	.interp	PROGBITS	0000000000400238		
		000000000000001c	00000000000000000	A 0 0		
	[2]	.note.ABI-tag 0000000000000020	NOTE 000000000000000000	0000000000400254 A 0 0		
	[3]	.note.gnu.build-i		0000000000400274		
			00000000000000000	A 0 0		
	[4]	.gnu.hash 000000000000001c	GNU_HASH 000000000000000000	0000000000400298 A 5 0		
	[5]	.dynsym	DYNSYM	00000000004002b8		
		000000000000078	0000000000000018			
	[6]	.dynstr 0000000000000056	STRTAB 00000000000000000	0000000000400330 A 0 0		
	[7]	.gnu.version	VERSYM	0000000000400386		
A		6000000000000000	00000000000000000	A 5 0		
	[8]	.gnu.version_r 00000000000000030	VERNEED 000000000000000000	0000000000400390 A 6 1		
	[9]	.rela.dyn	RELA	00000000004003c0		
<u>a</u> ,		0000000000000018	0000000000000018			
	[10]	.rela.plt 0000000000000048	RELA 00000000000000018	00000000004003d8 AI 5 24		
1 1 1	[11]	.init	PROGBITS	000000000000400420		
		000000000000001a	000000000000000000	AX 0 0		
	[12]	.plt 00000000000000040	PROGBITS	6000000000400440 AX 0 0		
· >_	1 [13]		00000000000000000000000000000000000000	00000000000400480		
		0000000000000000	00000000000000000			
	[14]	.text	PROGBITS	0000000000400490 AX 0 0		
	[15]	00000000000001b2 .fini	00000000000000000 PROGBITS	AX 0 0 0000000000400644		
		0000000000000009	000000000000000000	AX 0 0		
	[16]	.rodata	PROGBITS 000000000000000000	0000000000400650 A 0 0		
	[17]	00000000000000016 .eh frame hdr	PROGBITS	A 0 0 0000000000400668		
		000000000000003c	00000000000000000	A 0 0		
_	[18]	.eh_frame	PROGBITS	00000000004006a8		
	[19]	0000000000000114 .init_array	INIT_ARRAY	A 0 0 000000000000000000000000000000000		
	(1)	000000000000000000000000000000000000000	000000000000000000000000000000000000000	WA 0 0		
	[20]	.fini_array	FINI_ARRAY	0000000000000e18		
	[21]	0000000000000008 .jcr	00000000000000000 PROGBITS	WA 0 0 0000000000600e20		
		0000000000000000	00000000000000000			
	[22]	.dynamic	DYNAMIC	0000000000000e28		
_	[23]	00000000000001d0 .got	00000000000000000000000000000000000000	WA 6 0 0000000000600ff8		
_		0000000000000008	00000000000000008	WA 0 0		
_	[24]	.got.plt	PROGBITS	00000000000000000000000		
	[25]	0000000000000030 .data	00000000000000000 PROGBITS	WA 0 0 000000000000001030		
_	[23]	000000000000000000000000000000000000000	000000000000000000			
100	[26]		NOBITS	0000000000601040		
	[27]	0000000000000008 .comment	00000000000000000 PROGBITS	WA 0 0		

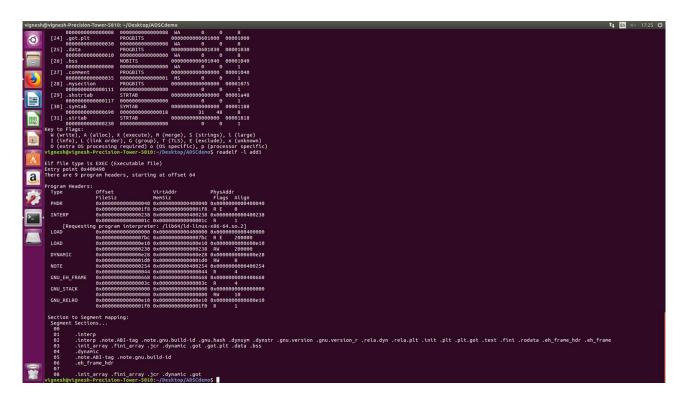
To add a section use *objcopy --add-section .mysection=mydata inptfile outptfile* to add a new section with a name .mysection and has a data found in the file mydata. Objcopy, by default, adds the section after the comment section.

Below is the output of readelf –sections add1...where add1 is the output file.

ianesha	Dviane	sh-Precision-Tower-581	0: ~/Desktop/ADSCd	emo		1 4 E 4× 1
	0]		NULL	000000000000000000000000000000000000000	0000000	
-			000000000000000000000000000000000000000			
2	E 10] .interp	PROGBITS	0000000000400238		
		000000000000000000000000000000000000000	000000000000000000000000000000000000000		1	
	F 2] .note.ABI-tag	NOTE	0000000000400254	00000254	
			000000000000000000000000000000000000000		4	
	F 3	1 .note.gnu.build-i		0000000000400274	00000274	
1			000000000000000000000000000000000000000		4	
	F 4] .gnu.hash	GNU HASH	0000000000400298	00000298	
2			000000000000000000000000000000000000000		8	
	۲ S].dynsym	DYNSYM	00000000004002b8	00000268	
		0000000000000078	00000000000000018	A 6 1	8	
	F 6] .dynstr	STRTAB	0000000000400330	00000330	
		0000000000000056	000000000000000000000000000000000000000	A 0 0	1	
100	Г 7] .gnu.version	VERSYM	0000000000400386	00000386	
		00000000000000000	000000000000000000	A 5 0		
9	[8]] .gnu.version_r	VERNEED	000000000400390	00000390	
		000000000000000000000000000000000000000	000000000000000000000000000000000000000			
	[9]] .rela.dyn	RELA	00000000004003c0	000003c0	
2		000000000000018	0000000000000018			
	[10]] .rela.plt	RELA	00000000004003d8		
		0000000000000048	0000000000000018			
	[11]].init	PROGBITS	000000000400420		
		000000000000001a	000000000000000000000000000000000000000			
	[12]].plt	PROGBITS	000000000400440		
a 📗		000000000000040	000000000000000000000000000000000000000		16	
-	[13]] .plt.got	PROGBITS	000000000400480		
		0000000000000008	000000000000000000000000000000000000000			
19	[14].text	PROGBITS	000000000400490		
		0000000000001b2	000000000000000000000000000000000000000		16	
	[15].fini	PROGBITS	0000000000400644 AX 0 0	4	
	E 14	00000000000000000009] .rodata	00000000000000000000000000000000000000	0000000000400650		
	[10	00000000000000016			4	
	147] .eh_frame_hdr	PROGBITS	0000000000400668		
	110	0000000000000003c			4	
	F10	1 .eh frame	PROGBITS	00000000004006a8		
100	1.10	0000000000000114	000000000000000000000000000000000000000		8	
	F19] .init array	INIT ARRAY	0000000000600e10		
	1.10		000000000000000000000000000000000000000		8	
	[20]] .fini array	FINI ARRAY	0000000000600e18		
		0000000000000000	000000000000000000000000000000000000000		8	
	[21]].jcr	PROGBITS	0000000000600e20		
		0000000000000000	000000000000000000000000000000000000000		8	
	[22]	1 .dvnamic	DYNAMIC	0000000000600e28		
		00000000000001d0	000000000000000000000000000000000000000		8	
	[23]].got	PROGBITS	0000000000600ff8	00000ff8	
		0000000000000000	00000000000000000			
	[24]] .got.plt	PROGBITS	0000000000601000	00001000	
		0000000000000030	0000000000000000			
	[25]].data	PROGBITS	0000000000601030		
		000000000000000000000000000000000000000	000000000000000000000000000000000000000			
	[26]].bss	NOBITS	0000000000601040		
		0000000000000008	000000000000000000000000000000000000000			
	[27]].comment	PROGBITS	000000000000000000000000000000000000000		
		000000000000035	000000000000000000000000000000000000000			
	[28]] .mysection	PROGBITS	0000000000000000		
		0000000000000111	000000000000000000000000000000000000000			
	[29]] .shstrtab	STRTAB	0000000000000000000000		
1	1000	0000000000000117	000000000000000000000000000000000000000		1	
	[30]] .symtab	SYMTAB	000000000000000000000000000000000000000		
		0000000000006690	00000000000000018	31 48	8	

Though we have added a new section, it is not yet executable. If we try to redirect the control flow there we will recieve a segmentation fault. This is because the new section is not yet in the loadable segment. So our section will not be loaded during the runtime.

We can check the loadable segments in the program header with *readelf* -l command.



We can see that our section is not there in the loadable segment(no. 3).

To make our section executable, we have to make some mannual changes in the binary. One can use a hex editor like 010Editor to do these changes.

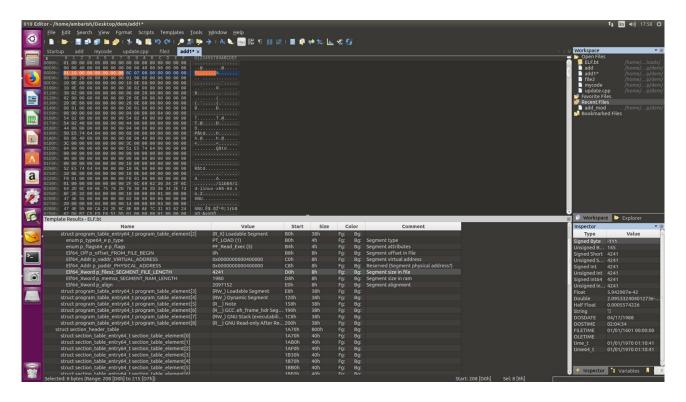
• Change the flag of the section in the section header table to 6.

010 Editor - /home/ambarish/Desktop/dem/add1*							t∎ En 40) 17:46 ⊀0
File Edit Search View Format Scripts Templates Too	ls Window Help						
🔇 ା 🗈 ାଳ ା 📾 📾 📾 🚔 ା 😓 📲 📽 ୨୦ ୧୦ ୮ 🗩 輝 ା				Jan 812			
and the second		· · · ·	· · · ·	4 103			
Startup add mycode update.cpp file2 add1* :							<>
	123456789ABCDEF						Open Files ELF.bt /home/loads/
							add /home/p/dem
2160h: 01 00 00 00 00 00 00 01 00 00 00 00 00							add1* /home/p/dem
	· · · · · · · · · · · · · · · · · · ·						ifile2 /home/p/dem
							mycode /home/p/dem
21A0h: 01 00 00 00 00 00 00 00 00 00 00 00 00							update.cpp /home/p/den
							Recent Files
							add mod /home/p/den
							🛃 Bookmarked Files
2210h: 98 86 88 88 88 88 88 88 88 88 88 88 88 88							
2250h: 30 02 00 00 00 00 00 00 00 00 00 00 00 00							
2250h: 01 00 00 00 00 00 00 00 00 00 00 00 00							
22701:							
a							
Template Results - ELF.bt							😠 💿 Workspace 🗁 Explorer
Name	Value	Start	Size	Color	Comment		Inspector •
struct section_table_entry64_t section_table_element[18]		1EF0h	40h	Fg: Bg:			Type Value
struct section_table_entry64_t section_table_element[19]		1F30h	40h	Fg: Bg:			Signed Byte 6
struct section table entry64 t section table element[20]		1F70h	40h	Fg: Bg:			Unsigned B 6
struct section table entry64 t section table element[21]		1FB0h	40h	Fg: Bg:			Signed Short 6
struct section table entry64 t section table element[22]		1FF0h		Fq: Bq:			Unsigned S 6
struct section table entry64 t section table element[23]		2030h		Fq: Bq:			Signed Int 6
struct section_table_entry64_t section_table_element[24]		2070h		Fg: Bg:			Unsigned Int 6
struct section_table_entry64_t section_table_element[25]		20B0h		Fg: Bg:			Signed Int64 6
struct section_table_entry64_t section_table_element[26]		20F0h					Unsigned In 6
struct section_table_entry64_t section_table_element[27]		2130h		Fg: Bg:			Float 8.407791e-45
struct section_table_entry64_t section_table_element[28]				Fg: Bg:			Double 2.96439387504748e
struct s_name64_t s_name	.mysection	2170h		Fg: Bg:			Half Float 3.576279e-07
enum s_type64_e s_type	SHT_PROGBITS (1)	2174h	4h	Fg: Bg:			String 🛛
enum s_flags64_e s_flags		2178h	8h	Fg: Bg:			DOSDATE
Elf64_Addr s_addr	0x000000000000000	2180h	8h	Fg: Bg:			DOSTIME 00:00:12
Elf64_Off s_offset	1075h	2188h	8h	Fg: Bg:			FILETIME 01/01/1601 00:00:00
Elf64_Xword s_size Elf64 Word s_link	28	2190h 2198h	8h 4h	Fg: Bg: Fg: Bg:			OLETIME
Elf64 Word s info		2198h 219Ch	4n 4h				time_t 01/01/1970 00:00:06
Elf64_Word s_info Elf64_Xword s_addralign		219Ch 21A0h	4n 8h	Fg: Bg: Fg: Bg:			time64_t 01/01/1970 00:00:06
Elf64 Xword s entsize		21A0h 21A8h	an 8h	Fg: Bg: Fg: Bg:			
char data[28]	the/attack/starts/here!/;0@	1075h	an 1Ch	Fg: Bg: Fg: Bg:			
struct section table entru64 t section table element[29]	encyactacity scarts/nere/20@						😴 🛷 Inspector 👎 Variables 👢 🔇
		21R0h				Start: 8568 [2178h] Sel: 8 [8h]	

• Make sure the address of the section = base address + offset, the base is same as that of other loadable sections. In this case base = 0x400000, offset = 0x1075. The base address can be obtained by looking at the other loadable sections and using the above formula.

010 Edito	or - /home/ambarish/Desktop/dem/add1* _Eile _Edit _Search _View Format Scripts Temp <u>l</u> ates _Tool _ ■ ~ ▶ ~ _ ■ ■ ● ● ● ↓ 为 ■ ■ ◆ へ ~ ↓ ● ★ ▶		: • • •	v * %: ⊾	× 13				tta, En ∢n)) 17:49 ⊀2
		- 1557/85802065 							Vorkspace Vorkspace
22									<u> </u>
0	Template Results - ELF.bt								📧 🕑 Workspace 🗁 Explorer
	Name	Value	Start	Size	Col	lor	Comment		inspector 🔹
X .	struct section_table_entryG4_isection_table_element[18] struct section_table_entryG4_isection_table_element[12] struct section_table_entryG4_isection_table_element[22] struct section_table_entryG4_isection_table_element[22] struct section_table_entryG4_isection_table_element[23] struct section_table_entryG4_isection_table_element[25] struct section_table_entryG4_isection_table_element[25] struct section_table_entryG4_isection_table_element[26] struct section_table_entryG4_isection_table_element[26] struct section_table_entryG4_isection_table_element[27] struct section_table_entryG4_isection_table_element[28] struct section_table_entryG4_isection_table_element[28] struct section_table_entryG4_isection_table_element[28] struct section_table_entryG4_isection_table_element[28] struct section_table_entryG4_isection_table_element[28] struct section_table_entryG4_isection_table_element[28] struct section_table_entryG4_isection_table_element[28] struct section_table_entryG4_isection_table_element[28] struct_section_table_entryG4_isection_table_element[28] struct_section_table_entryG4_isection_table_element[28] struct_section_table_entryG4_isection_table_element[28] struct_section_table_entryG4_isection_table_element[28] struct_section_table_entryG4_isection_table_element[28]	mysection SHT_PROGBITS (1) SF44_Write_Alloc(6) bxc0000000000401075 1075h 28	1EF0h 1F30h 1F70h 1FF0h 2030h 2070h 2070h 2070h 2130h 2170h 2170h 2170h 2174h 2178h 2180h 2188h 2190h 2190h	40h 40h 40h 40h 40h 40h 40h 40h 40h 40h	Fg: Fg: Fg: Fg: Fg: Fg: Fg: Fg: Fg: Fg:	Bg: Bg: Bg: Bg: Bg: Bg: Bg: Bg: Bg: Bg:			Type Value Signed Stre 117 Unsigned B., 117 Signed Stort 4213 Unsigned S., 4213 Unsigned S., 4213 Unsigned S., 4213 Unsigned Int. 4198317 Unsigned Int. 4198317 Float 2.07348/0318045€ Half Float 0.0005440712 String u00 DOSDATE 0.72/1/988 DOSTATE 0.27/1986 OCELETINE 0.70/1/601 00:00:00
	Elf64_Word s_link Elf64_Word s_info		2198h 219Ch	4h 4h		Bg: Bg:			time_t 02/18/1970 14:15:17 time64_t 02/18/1970 14:15:17
	Elf64_Xword s_addralign Elf64_Xword s_entsize		21A0h 21A8h	8h 8h		Bg: Bg:			
	char data[28] struct section table entrums t section table element[29] Edited template variable.	the/attack/starts/here!/¿D@	1075h 2180h	1Ch 40h		Bg: Bo:		Start: 8576 [2180h] Sel: 8 [8h]	🗧 🖋 Inspector 👎 Variables 👢 🦉

Increase the size of the loadable segment $(3^{rd} \text{ segment}, 02)$ in the program header table to include our new section. Change both the file length and ram length. The size has to be one greater than the address of the last byte of our section. To get the value for Elf64_xword p_filesz_segment_file_length and Elf64_xword p_memsz_segment_ram_length (they have to be set the same): Approach 1, to look at the size and offset of our added section (.mysection), add them together (and plus 1) to get 0x1091. Approach 2 is to look at the last byte (plus 1) of chardata field of the section.



And now our section is successfully in the loadable segment. After this the section is executable.

ambaris	sh@ambarish-Precisio	ion-Tower-S810: ~/Desktop/dem 1	En 🜒) 17:59 🔱
0	NOTE	0x000000000000100 xx000000000103 RV 8 0 0x000000000000000000000000000000000	
0	GNU EH FRAME	0x200000000000000000000000000000000000	
		0x00000000000003c 0x0000000000000000 R 4	
	GNU_STACK	6x0000000000000000	
	GNU_RELRO	6x8000000000000000000000000000000000000	
	Section to Segment Section 00	ment mapping: ons	
	01 .inter	rp rp.note.ABI-tag.note.gnu.bulld-id.gnu.hash.dynsyn.dynstr.gnu.version.gnu.version r.rela.dyn.rela.plt.init.plt.plt.got.text.fini.rodata.eh frame hdr.eh frame	
	03 .init_ 04 .dynam	array .jcri.dynamic .got .got.plt .data .bss mic	
		.ABI-tag .note.gnu.build-id rame_hdr	
	08 .init_	_array .fini_array .jcr .dynamic .got sh-Precision-Tower-5810:-/Desktop/dem\$ readelf -l add1	
A	Elf file type is Entry point 0x40	s EXEC (Executable file)	
a	There are 9 prog	gram headers, starting at offset 64	
-	Program Headers: Type	Offset VirtAddr PhysAddr	
	PHDR	FileSiz HenSiz Flags Align 6x00090600600040 0x00000000400040 0x000000000400040	
1	INTERP	0x000000000000150 xx000000000000156 R E 8 0x00000000000000000000000000000000000	
	[Requestin	ng program tniterpreter; /11b4/14/11uux-x86-64.so.2] 6 %000000000000000000000000000000000000	
S	LOAD	oxaanaosaanaaaaaaaaaaaaaaaaaaaaaaaaaaaaa	
		0x00000000000238 0x000000000238 RW 200000	
·	DYNAMIC	\$x0008000000000000000000000000000000000	
	NOTE	\$x0000000000000400254 (x000000000400254 (x 4) \$x0000000000000400040000000000000000000	
	GNU_EH_FRAME	\$x000000000000000000000000000000000000	
	GNU_STACK	6x000000000000000000000000000000000000	73
	GNU_RELRO	0x000000000000000000000000000000000000	
	Section to Segm	ment mapping:	
	Segment Section		
	01 .inter 02 .inter	rp .note.ABI-tag .note.gnu.build-id .gnu.hash .dynsym .dynstr .gnu.version .gnu.version_r .rela.dyn .rela.plt .init .plt .got .text .fini .rodata .eh_frame_hdr .eh_frame .m	mysection
	04 .dynar		
	05 .note.	ABI-tag .note.gnu.butld-id rame.hdr	
-	07	array .fini_array .jcr .dynamic .got _	
		sh-Precision-Tower-5810:-/Desktop/den\$	

Now we can add our executable instructions and direct the control flow here.

Direct Call

- \bullet 1st add a section called *.springboard*.
- As mentioned earlier redirect the calls to the spring board.

 \bullet To add the checkpoints, we need to insert a piece of code before the return statement but that would change the offsets of many other instructions which is not desirable.

• So we add a new section called *.extention* in which we add the extra piece of code with a return at the end.

vignes	h@vignesh-Precisio	n-Tower-5810: ~/Desktop/AD	SCdemo		14	n «×	18:29 🔱
Q	400604: 400606: 400608: 400605:	74 20 31 db 0f 1f 84 00 00 00 00 00	je xor nopl	400626 <llbc_csu_lnlt+0x56> %ebx,%ebx 0x0(%rax,%rax,i)</llbc_csu_lnlt+0x56>			
	400610: 400613: 400613: 400616: 4006210: 400621: 400624: 400626: 400626: 400626: 400626: 400622: 400622: 400630: 400632: 400633:	4c 89 ea 4c 89 ff 44 89 ff 41 ff 14 dc 48 32 c0 01 48 32 eb 55 54 82 c4 08 55 54 54 55 54 55 54 55 54 55 54 55 54 55 54 55 54 55 55	add cmp jne add pop pop pop pop pop retq	<pre>%r13,Krdx %r14,Krdi %r14,Krdi %r14,Krdi %r40,Krdx,80 \$00,Krdx 400010llbc_csu_init+0x40> \$00,Krdy %r00 Krdy %r14 %r14 %r15</pre>			
	400635: 400636: 40063d:	90 66 2e 0f 1f 84 00 00 00 00 00	пор порж	%cs:0x0(%rax,%rax,1)			
	400640:	40 <libc_csu_fini>: f3 c3</libc_csu_fini>	repz	retq			
a,	Disassembly of	section .fini:					
%	00000000004006 400644: 400648: 40064c: Disassembly of	44 <_fini>: 48 83 ec 08 48 83 c4 08 c3 section .extension:	sub add retq	Şöxö, Krsp Şöxö, Krsp			
	0000000004010 401075: 401077: 401077: 401077: 401077: 401083: 401087: 4010857: 4010857: 4010852: 401091: 401092: 401093: 401094:	75 <.extension>: e9 76 73 7f ff 59 50 80 52 00 00 00 2b 44 24 08 83 78 00 7e 05 e9 1f f7 ff ff 58 50 90	callq nop leave jush lea sub cmp jle jmpq pop retq nop nop	400470 «[soc39_scanf@plt> % % sx % 12(Sr(E)),%eax # 401095 «FRAME_END+0x8dd> 085(%rsp),%eax 40050 «GNU_EH_FRAME_HDR+0x148> %rax			
	00000000004010 401095: 401095: 401096: 401080: 401081: 401081: 401082: 401083: 401084:	section .springboard: 95 <.springboard>: e8 ec f4 ff ff 96 96 98 99 99 99 90 •>recision-Tower-5810:	callq jmpq nop nop nop nop nop -/Deskto	4005b5 ≪malm+0xe>			

• We also have to redirect the end of the function to this section.

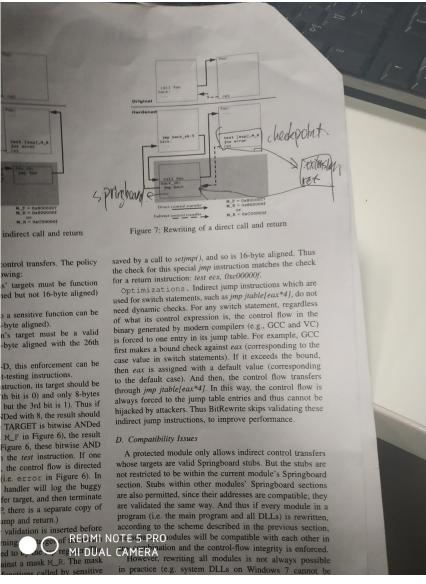
ambarist	n@ambarish-Precisi	on-Tower-	5810: ~//	Desktop/de	èm.	h	En 🜒) 19:07 👯
10000	40053a:	66 Of 1			порм		
0	000000000040054	a da	alobal	dtore aux			
	400540:	80 3d f			cmpb		
	400547: 400549:	75 11 55			jne push	40055a <do_global_dtors_aux+0x1a> %rbo</do_global_dtors_aux+0x1a>	
	40054a:	48 89 e	5		mov	%rsp,%rbp	
	40054d: 400552:	e8 6e f	r fr fr		callq		
	400553:	5d c6 05 e	6 0a 20	00 01	pop movb	%rbp 50x1,0x200ae6(%r\p) # 601040 <tmc_end_></tmc_end_>	
	40055a:	f3 c3			герд г	retq	
	40055c:	0f 1f 4	0 00		nopl	6x0(%rex)	
	666666666666666666666666666666666666666	0 <frame< td=""><td>_dummy></td><td></td><td></td><td></td><td></td></frame<>	_dummy>				
	400560: 400565:	bf 20 0 48 83 3			mov cmpq	\$0x600e20,%edi 50x00,%rdi)	
	400569:	75 05			jne	400570 <frame_dummy+0x10></frame_dummy+0x10>	
	40056b: 40056d:	eb 93 0f 1f 0	9		jmp nopl	400500 <register_tm_clones> (%rax)</register_tm_clones>	
F	400570:	b8 00 0	0 00 00		MOV	\$0x0,%eax	
	400575: 400578:	48 85 c 74 f1	0		test je	%rax,%rax 40056b <frame_dunmy+0xb></frame_dunmy+0xb>	
A	40057a:				push	%rbp	
	40057b: 40057e:	48 89 e ff d0	5		mov	%rsp,%rbp g *%rax	
a	400580:	5d			pop	y Jun Krbp	
<u> </u>	400581:	e9 7a f	f ff ff		jmpq	400500 <register_tn_clones></register_tn_clones>	
	6666666666646658						
	400586: 400587:	55 48 89 e	-		push mov	xrbp Xrsp.Xrbp	
	40058a:	48 83 e			sub	vi su to, vi co solo solo solo solo solo solo solo s	
6	40058e: 400592:	48 8d 4			lea	-0x10(%rbp),%rax	
	400592:	48 89 c bf 54 0			MOV	%rax,%rs1 Fox40054,%edi	
	40059a: 40059f:	b8 00 0 e8 cc f			mov	Soxo, Keax	
	400591:	90 CC T	e tr tr		nop	q 400470 <isoc99_scanf@plt></isoc99_scanf@plt>	
	4005a5:	c9			leaveq		
· >_ ·	4005a6:				retq		
	00000000004005a						
6	4005a7: 4005a8:	55 48 89 e	5		push mov	xrbp xrsp, xrbp	
	4005ab:	b8 00 0			MOV	\$0x0,%eax	
	4005b0: 4005b5:	e8 d1 f bf 57 0			callq mov	q 400586 <\$cn> 50x400677, #ed\	
	4005ba:	e8 91 f	e ff ff		callq	q 400450 <puts@plt></puts@plt>	
	4005bf: 4005c4:	bf 61 0 e8 87 f			mov callo	\$0×480661,≸edi q. 409459 ⊂ puts@pit>	
	4005c9:	b8 00 0			MOV	\$0x0,%eax	
	4005ce: 4005cf:	5d c3			pop retq	%rbp	
	00000000004005d 4005d0:	0 <lib 41 57</lib 	c_csu_t	1112>:	push	%r15	
	4005d2:	41 56			push	%r14	
	4005d4: 4005d7:	41 89 f 41 55	T		push	%edi,%r15d %r13	
	4005d9:	41 54			push	%r12	
100	4005db: 4005e2:	4c 8d 2	5 2e 08	20 00	lea push	0x20082e(%rip),%r12 # 600e10 <frame_dummy_init_array_entry> %rbp</frame_dummy_init_array_entry>	
					p a a n		

-> this is before the modification

winnerho	ulanash Deselator	-Tower-5810: ~/Desktop/ADS	C dama	1 ₄ 🛅 4× 1830 🕏
vignesnia	400565:	48 83 3f 00		50x0,(%rdi)
0	400569:	75 05	cmpq jne	30X0(X)UU 400570 cfrane dunny+0X10>
0	40056b:	eb 93	jmp	400500 <register_time_lones></register_time_lones>
	40056d:	0f 1f 00	nopl	(%rax)
	400570:	b8 00 00 00 00	MOV	\$0x0,%eax
	400575:	48 85 c0 74 f1	test	Krax, Krax
	400578: 40057a:	55	je push	40050b <frame_dummy+0xb> %</frame_dummy+0xb>
	40057b:	48 89 e5	mov	Arup Krsp, Krbp
	40057e:	ff d0	callq	*%гах
	400580:	Sd	pop	%rbp
===	400581:	e9 7a ff ff ff	jmpq	400500 <register_tm_clones></register_tm_clones>
1	00000000040058	6 (500)		
	400586:	55	push	*rbp
E	400587:	48 89 e5	mov	%rsp,%rbp
	40058a:	48 83 ec 10	sub	\$8x10,%rsp
1	40058e:	48 8d 45 fe	lea	-Ox10(%rbp),%rax
	400592: 400595:	48 89 c6 bf 54 06 40 00	MOV	%rax,%rs1 50x400534,%ed1
-	400593:	b8 00 00 00 00 00	nov	304700054, ACUL S080, Seax
and the second	40059f:	e9 d1 0a 00 00	jmpq	401875 <frame_end+0x8bd></frame_end+0x8bd>
	4005a4:	98	nop	
	4005a5: 4005a6:	90 98	nop	
	400580:	90	nop	
a	0000000004005a	7 <main>:</main>		
	4005a7:		push	%rbp
100	4005a8:	48 89 e5	MOV	%rsp,%rbp
	4005ab: 4005b0:	b8 00 00 00 00 e9 e0 0a 00 00	mov impg	\$0x0,%eax \$0x0,%eax € FRAME END +0x8dd>
	4005b5:	bf 57 06 40 00	MOV	40,103 \Mint_EndU_0_000/ \$0x400657,%edi
	4005ba:	e8 91 fe ff ff		400450 <puts@plt></puts@plt>
· >_ ·	4005bf:	bf 61 06 40 00	MOV	\$0x400661,%edi
	4005c4:	e8 87 fe ff ff	callq	
0	4005c9: 4005ce:	b8 00 00 00 00 5d	mov pop	\$0x0, %eax ************************************
	4005cf:	c3	retq	
G	0000000004005d	0 <libc_csu_init>:</libc_csu_init>		
	4005d0: 4005d2:	41 57 41 56	push push	%r15 %r14
	4005d4:	41 89 ff	mov	an Len , sr 15d
	4005d7:	41 55	push	%r13
	4005d9:	41 54	push	%r12
	4005db: 4005e2:	4c 8d 25 2e 08 20 00 55	lea push	0x20082e(%rip),%r12
	4005e2: 4005e3:	48 8d 2d 2e 08 20 00	lea	жгор 0x20082e(%rip),%rbp # 600e18 <tnit_array_end></tnit_array_end>
	4005ea:	53	push	Srbx
	4005eb:	49 89 f6	mov	%rst,%r14
	4005ee: 4005f1:	49 89 d5	mov	Srdx, Sr13
	4005f1: 4005f4:	4c 29 e5 48 83 ec 08	sub sub	%r12,%rbp 508,%rsp
	4005f8:	48 c1 fd 03	sar	50x3,%rbp
	4005fc:	e8 1f fe ff ff	callq	400420 <_init>
	400601:	48 85 ed	test	srbp, srbp
	400604:	74 20 31 db	je	400626 <_ Libc_csu_init+0x56>
	400608:	of 1f 84 00 00 00 00	xor nopl	%ebx,%ebx 50%/sxx,%rax,1)
	40060f:	00	and the	
	400610:	4c 89 ea	mov	%r13,%rdx
	400613:	4c 89 f6	MOV	sr14,srsi

-> this is after modification

we delete the required number of instructions to add a jmp instruction and replace the remaining bytes with nop(0x90). So we copy the deleted code in the .extension before we add the checkpoint. The program control flow for the modified program is shown as the figure below.



0000000000400580	ó <s< th=""><th>cn></th><th>:</th><th></th><th></th><th></th><th></th></s<>	cn>	:				
400586:	55					push	%гbр
400587:	48	89	e5			mov	%rsp,%rbp
40058a:	48	83	ec	10		sub	\$0x10,%rsp
40058e:	48	8d	45	f0		lea	-0x10(%rbp),%rax
400592:	48	89	сб			mov	%rax,%rsi
400595:	bf	54	06	40	00	mov	\$0x400654,%edi
40059a:	b 8	00	00	00	00	mov	\$0x0,%eax
40059f:	e8	сс	fe	ff	ff	callq	400470 <isoc99_scanf@plt></isoc99_scanf@plt>
4005a4:	90					nop	
4005a5:	c9					leaveq	
4005a6:	с3					retq	
00000000004005a7	7 <m< td=""><td>ain</td><td>>:</td><td></td><td></td><td></td><td></td></m<>	ain	>:				
4005a7:	55					push	%гbр
4005a8:	48	89	e5			mov	%rsp,%rbp
4005ab:	b 8					mov	\$0x0,%eax
4005b0:	e8 (d1	ff	ff	ff	callq	400586 <scn></scn>
4005b5:	bf	57	06	40	00	mov	\$0x400657,%edi
4005ba:	e8	91	fe	ff	ff	callq	400450 <puts@plt></puts@plt>
4005bf:	bf	61	06	40	00	MOV	\$0x400661,%edi
4005c4:	e8	87	fe	ff	ff	callq	400450 <puts@plt></puts@plt>
4005c9:	b 8	00	00	00	00	MOV	\$0x0,%eax
4005ce:	5d					рор	%гbр
4005cf:	с3					retq	

-> the original code

0000000000400	586 <scn>:</scn>	
400586:	55	push %rbp
400587:	48 89 e5	mov %rsp,%rbp
40058a:	48 83 ec 10	sub \$0x10,%rsp
40058e:	48 8d 45 f0	lea -0x10(%rbp),%rax
400592:	48 89 c6	mov %rax,%rsi
400595:	bf 54 06 40 00	mov \$0x400654,%edi
40059a:	b8 00 00 00 00	mov \$0x0,%eax
40059f:	e9 d1 0a 00 00	jmpq
4005a4:	90	nop
4005a5:	90	nop
4005a6:	90	nop
0000000000400	5a7 <main>:</main>	
4005a7:	55	push %rbp
4005a8:	48 89 e5	mov %rsp,%rbp
4005ab:	b8 00 00 00 00	mov \$0x0,%eax
4005b0:	e9 e0 0a 00 00	jmpq 401095 <frame_end+0x8dd></frame_end+0x8dd>
4005b5:	bf 57 06 40 00	mov \$0x400657,%edi
4005ba:	e8 91 fe ff ff	callq 400450 <puts@plt></puts@plt>
4005bf:	bf 61 06 40 00	mov \$0x400661,%edi
4005c4:	e8 87 fe ff ff	callq 400450 <puts@plt></puts@plt>
4005c9:	68 00 00 00 00 D8	mov \$0x0,%eax
4005ce:	5d	рор %гbр
4005cf:	c3	retq

-> the call is modified

00000000040	1075 <.extension>:	
401075:	e8 f6 f3 ff ff	callq 400470 <isoc99_scanf@plt></isoc99_scanf@plt>
40107a:	90	nop
40107b:	C9	leaveq
40107c:	50	push %rax
40107d:	8d 05 12 00 00 00	lea 0x12(%rip),%eax # 401095 <frame_end+0x8dd></frame_end+0x8dd>
401083:	2b 44 24 08	sub 0x8(%rsp),%eax
401087:	83 f8 00	cmp \$0x0,%eax
40108a:	7e 05	jle 401091 <frame_end+0x8d9></frame_end+0x8d9>
40108c:	e9 1f f7 ff ff	jmpq 4007b0 <gnu_eh_frame_hdr+0x148></gnu_eh_frame_hdr+0x148>
401091:	58	рор %гах
401092:	с3	retq
401093:	90	nop
401094:	90	пор
	of section .springboard: 1095 <.springboard>:	
401095:	e8 ec f4 ff ff	callq 400586 <scn></scn>
40109a:	e9 16 f5 ff ff	jmpg 4005b5 <main+0xe></main+0xe>
40109f:	90	nop
4010a0:	90	nop
4010a1:	90	nop
401001.		
4010a2:	90	пор
	90 90	пор пор

-> this is the code in our added sections.

The checkpoint :

push %rax #to save t	he current value of rax
lea 0x12(%rip),%eax	#loading the springboard address into rax
sub 0x8(%rsp),%eax	#subtracting the return address from the stack and rax (rsp pointing to the 8 bytes below the return address)
cmp 0x0,%eax	#checking if the return address is greater than the springboard address (we need to make sure the return address is inside springboard section)
jle 401091	#jumping to return if everything is fine
jmpq 4007b0	#error handling, just a random address in the #binary to cause seg fault.
pop %rax	#restore the value of rax
retq	#return

Indirect Call

• The return statements are handled the same way as the direct call.

 \bullet As for the call, it also is dealt like the direct call but we dont know the actuall address of the function being called as it is read from the register.

• Right now, we dont have an automatic technique for this but one can manually see the address loaded in the register by tracing back few assembly instructions.

Benchmark

We tried our approach a to benchmark binary : kill. The results are as below.

• Number of direct calls = 245(220 library calls, 25 normal calls)

• Number of indirect calls = 2

• Number of returns = 14 (2 sensitive returns and 12 normal ones)

we remove our checkpoint for the 2 sensitive returns as they are called from the system and return to the system.

The funcionality has been tested by trying a few commands like:

• ./kill

• ./kill --help

• ./kill --version

0 ./kill -9 \$PID

Performance has been tested by creating and killing the process 10,000 times(I wrote a c++ script).

Original

0.859882 0.854217 0.854568 0.854012

= 0.85570 + 0.003, deviation = 0.35%

Fortified

0.861137 0.861522 0.850526 0.852335

= 0.85640 + 0.006, deviation = 0.70%

overhead = 0.08%

The above times are in seconds. The overhead is so small(even smaller than the deviation) as kill is a very small binary.