Practical Control Flow Integrity & Randomization for Binary Executables

Group 14

Tsen-Yen Hsiao, Chieh-Shen Chen, Chien Tu, Rick Tsai

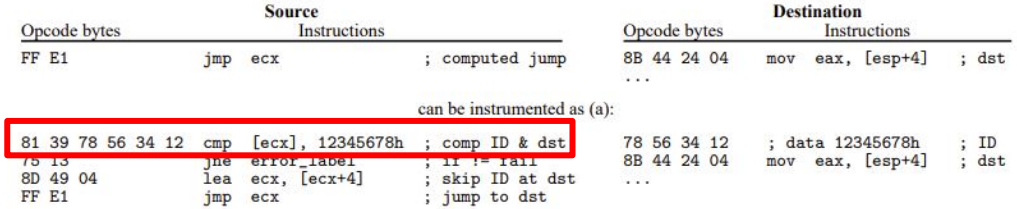
Outline

- CFI Introduction
- CCFIR High Level View
- CCFIR Details
- Performance and Remarks

Outline

- CFI Introduction
- CCFIR High Level View
- CCFIR Details
- Performance and Remarks

CFI Introduction



Source			Destination		
Opcode bytes	Instructions		Opcode bytes	Instructions	
FF E1	jmp ecx	; computed jump	8B 44 24 04	mov eax, [esp+4]	; dst
			...		

can be instrumented as (a):

81 39 78 56 34 12	cmp [ecx], 12345678h	; comp ID & dst	78 56 34 12	; data 12345678h	; ID
75 13	jne error_label	; if != fail	8B 44 24 04	mov eax, [esp+4]	; dst
8D 49 04	lea ecx, [ecx+4]	; skip ID at dst	...		
FF E1	jmp ecx	; jump to dst			

- Why?

  Protect from hijacking by changing target address

  code injection, control flow hijack, ROP, return-to-libc

- What?

  Indirect jump/call and return instruction

- How?

  Mark valid targets of indirect control transfer (IDs)

  Check IDs once with indirect jumps

CFI vs CCFIR

- CFI

  v: strong protection

  x: high overhead, required source code and debug info for precise CFG

- CCFIR

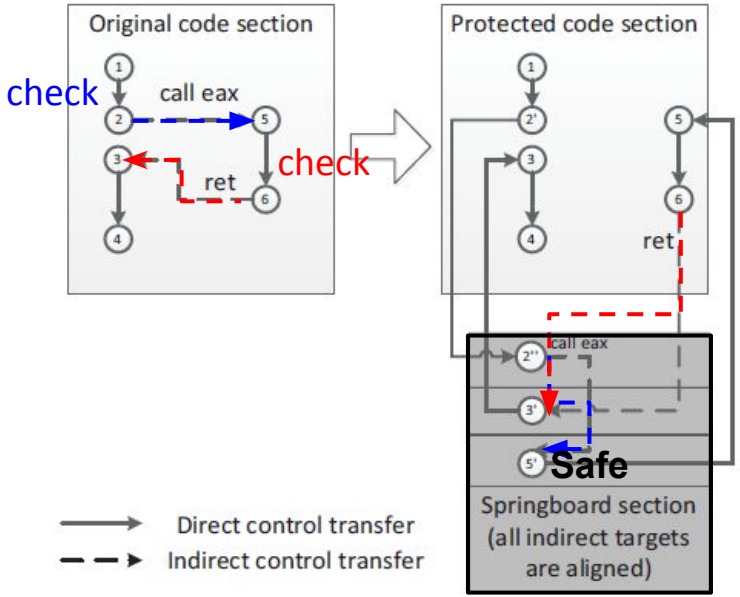
  v: low overhead, binary only, support incremental deployment

  x: still have unprotected attacks

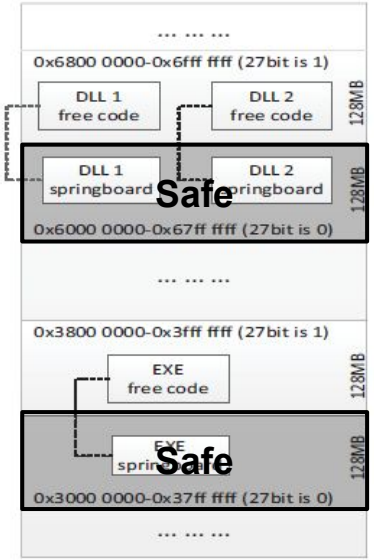
Outline

- CFI Introduction
- CCFIR High Level View
- CCFIR Details
- Performance and Remarks

CCFIR High Level View



▲ CCFIR concept



▲ Memory layout after CCFIR

Springboard Stub Classification

How to identify valid address in bit operation?

Use Bit Mask of stubs in Springboard.

Executable	Bits				Meaning
	27	26	3	2-0	
no	*	*	*	***	Non-executable section
yes	1	*	*	***	Normal code section
yes	0	*	*	!000	Springboard's invalid entry
yes	0	*	1	000	Springboard's function pointer stub
yes	0	1	0	000	Springboard's sensitive return stub
yes	0	0	0	000	Springboard's normal return stub

CCFIR = 3-ID CFI

Enforcing Control Flow Integrity

1. Indirect call/jump can only jump to function pointer stubs

   => 8 B not 16 B aligned

2. Normal return can only jump to normal return address not sensitive

   => 16 B aligned with 26th 0

3. Return in sensitive functions can jump to any return address

   => 16 B aligned

4. Byte aligned means no jumps to the middle. Random initialization.

Outline

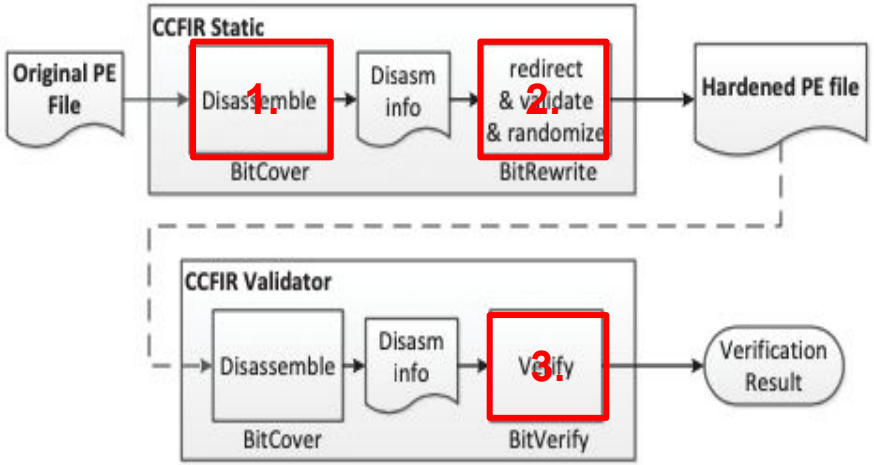
- CFI Introduction
- CCFIR High Level View
- CCFIR Design and Implementation
- Performance and Remarks

CCFIR Implementation - Overview

1. BitCover
   - Disassemble for input PE file
   - Identify indirect inst and indirect control-transfer targets

2. BitRewrite
   - insert Springboard Section
   - encode valid transfer target
   - run time check

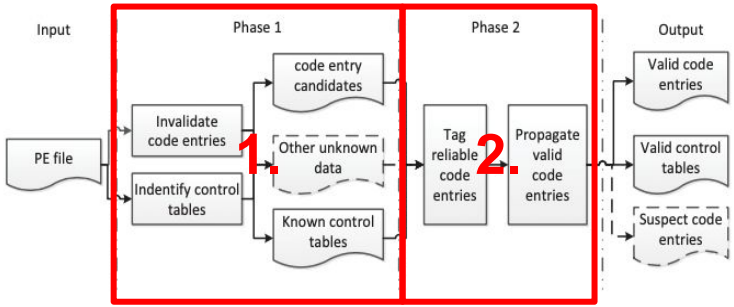
3. BitVerify
   - verify our defined security policy



CCFIR Implementation - BitCover

Identify Indirect Control Transfer & Valid Targets **(= disassemble)**

1. Explore Data and Code
   - Entry Point in export table/Relocation Entries
   - Must or May - Terminate Function
   - Switch Jump Table



2. Refine Disassembling Result
   - remove unreachable entries from relocation entry
   - remove invalid entries

CCFIR Implementation - BitRewrite

1. Redirecting Indirect Control Transfer Targets for
   a. Return Address
   b. Function Pointer
2. Check For the Springboard Mask and Slice
3. Compatibility Issue, eg DLL and import function

Enforcement:

1. Indirect jump can only jump to function stubs in Springboard
2. Return instructions are constrained to jump to return address stub in Springboard
3. Normal return are not allowed jumping to sensitive return stubs

CCFIR Implementation - BitRewrite

Indirect

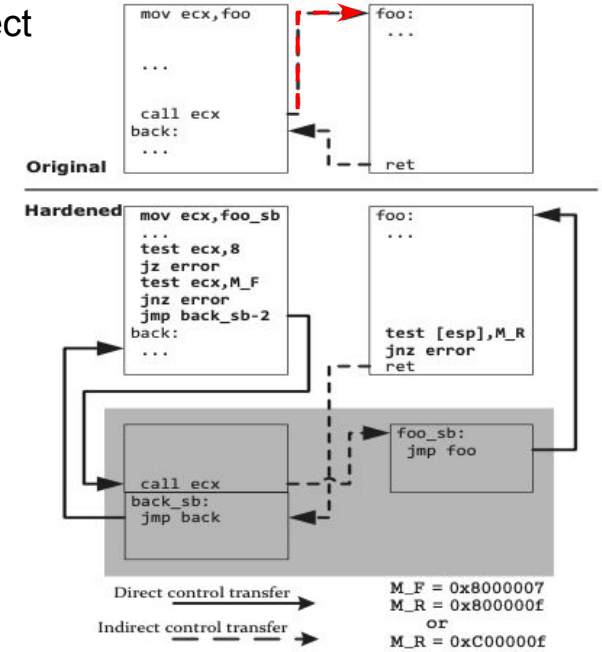


Figure 6: Rewriting of an indirect call and return

Direct

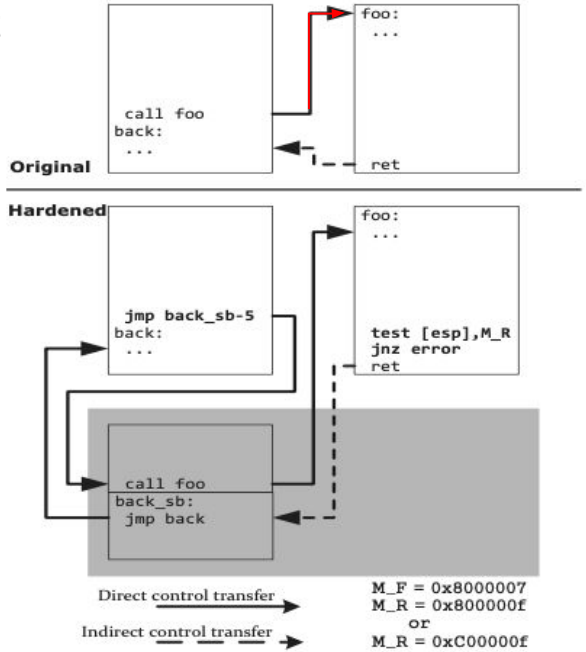
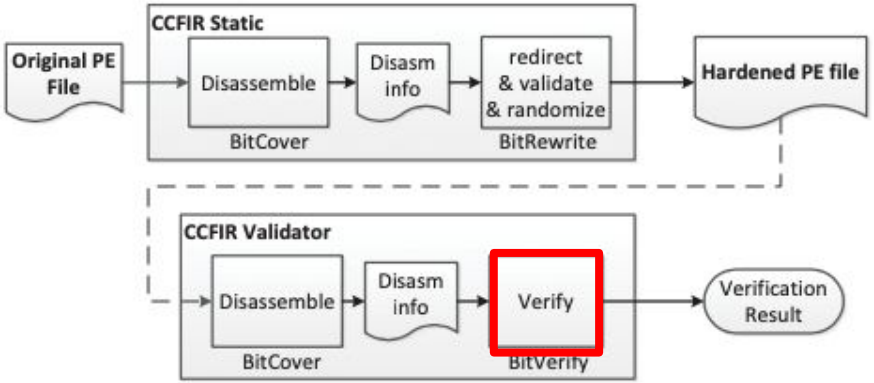


Figure 7: Rewriting of a direct call and return

CCFIR Implementation - BitVerify

- Verification provides independent checks if the target obey the policy

  => guarantee all indirect and ret can only flow to valid code entry



Outline

- CFI Introduction
- CCFIR High Level View
- CCFIR Details
- Performance and Remarks

CCFIR Overhead

- CCFIR

  Largest overhead: 8.6 %

  Average overhead: 3.6 %

Ref: CFI (Max 26.8 %, Avg 7.7%)

- CCFIR (browser)
- Low Springboard size and Randomizing time

App.	performance			file size (KB)	
	original run time	new run time	overhead	original	new
SPECint2000			Avg: 3.6385%		
gzip	84.30	86.50	2.6097%	101	140
vpr	66.00	66.00	0.0001%	231	301
gcc	38.33	39.90	4.0870%	1181	1642
mcf	31.90	31.97	0.2089%	80	116
crafty	43.03	43.50	1.0845%	290	368
parser	93.23	98.60	5.7562%	159	226
eon	57.50	60.50	5.2174%	440	618
perlbmk	64.47	70.01	8.6002%	605	829
gap	43.30	46.43	7.2363%	439	639
vortex	74.33	78.97	6.2334%	488	648
bzip2	68.87	71.00	3.0978%	91	131
twolf	107.00	107.00	0.0000%	262	332
SPECfp2000			Avg: 0.5855%		
wupwise	171.00	174.00	1.7544%	67	83
swim	347.00	347.00	0.0000%	48	61
mgrid	588.00	588.00	0.0000%	49	63
applu	484.00	484.00	0.0000%	165	181
mesa	72.87	75.77	3.9799%	531	681
galgel	265.00	265.00	0.0000%	281	317
art	31.40	31.43	0.1060%	89	129
equake	46.50	46.83	0.7168%	93	129
facerec	239.00	239.00	0.0000%	127	150
ammp	91.87	93.53	1.8143%	217	279
lucas	151.00	150.00	-0.6623%	121	127
fma3d	200.00	201.00	0.5000%	1429	1633
sixtrack	425.00	425.33	0.0786%	1463	1618
apsi	351.00	351.00	0.0000%	201	236
Browsers					
mshtml.dll				2,995	4,594
xul.dll				11,498	15,620

Protection Effect

- Eliminating ROP Gadgets

App.	modifications							#gadgets		
	redirected fp/ret_addr				validated inst		optimiz.			
	#fp	#imp	#GPA	#call	#indirect call/jmp	#ret	#skipped fp/import	original	new	valid
SPECint2000										
gzip	77	20	3	976	106	430	171	2484	0	0
vpr	85	20	3	1578	110	768	190	4437	0	0
gcc	1000	26	3	12185	263	5628	612	42884	0	0
mcf	73	20	3	896	105	392	173	1791	0	0
crafty	88	23	3	2135	114	930	228	7483	0	0
parser	78	20	3	1600	114	751	173	4400	0	0
eon	1546	28	3	4391	381	2325	316	10366	0	0
perlbmk	924	39	3	7017	203	3229	419	30949	0	0
gap	758	22	3	9991	1352	2672	181	20455	0	0
vortex	164	23	3	3429	124	1715	213	13408	0	0
bzip2	69	20	3	826	103	367	171	1824	0	0
twolf	81	20	3	1385	109	674	183	3987	0	0
SPECfp2000										
wupwise	7	4	0	127	35	31	48	255	0	0
swim	7	4	0	82	30	15	44	116	134	0
mgrid	7	4	0	104	31	21	44	161	166	0
applu	7	4	0	118	29	27	42	182	172	0
mesa	585	22	3	8513	495	3539	345	21696	0	0
galgel	11	4	0	534	62	142	58	1515	952	0
art	73	20	3	895	105	406	174	1874	0	0
equake	69	19	3	862	103	381	154	1710	0	0
facerec	7	4	0	213	55	50	66	826	775	0
ammp	181	20	3	2014	132	901	178	5039	0	0
lucas	7	4	0	98	43	20	55	129	0	0
fma3d	7	4	0	1341	77	438	72	4161	0	0
sixtrack	13	4	0	667	92	208	72	3979	3312	0
apsi	7	4	0	372	40	98	54	1126	878	0

Comment

- Best: Low Runtime and Space to implement

- Weakness:
  1. Race Condition of Return Address

     => Return Address will be modified by another thread

- Limitation:
  1. Cannot support self-modify code
  2. Useless if Springboard is vulnerable to attacker

Q&A