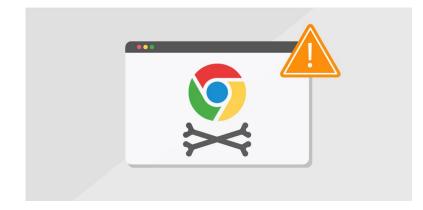
RecIPE: Revisiting the Evaluation of Memory Error Defenses

Yuancheng Jiang, Roland H.C. Yap, Zhenkai Liang, Hubert Rosier AsiaCCS 2022



• Memory errors usually form critical vulnerabilities in low-level languages



• Memory errors usually form critical vulnerabilities in low-level languages

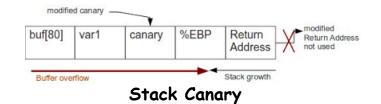


- The primary source of vulnerabilities
 - 70% severe security bugs in Chrome

• Memory errors usually form critical vulnerabilities in low-level languages

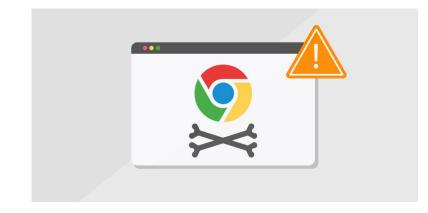


• Defenses against memory errors

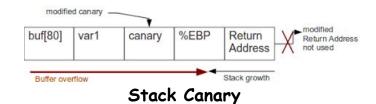


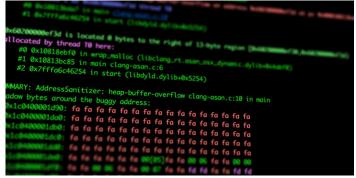
- The primary source of vulnerabilities
 - 70% severe security bugs in Chrome

• Memory errors usually form critical vulnerabilities in low-level languages



• Defenses against memory errors





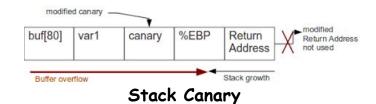
Address Sanitizer

- The primary source of vulnerabilities
 - 70% severe security bugs in Chrome

• Memory errors usually form critical vulnerabilities in low-level languages



• Defenses against memory errors





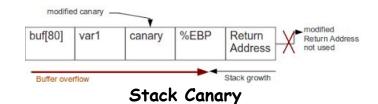
Address Sanitizer

- The primary source of vulnerabilities
 70% severe security bugs in Chrome
- Tradeoffs due to constraints from overhead and compatibility.

• Memory errors usually form critical vulnerabilities in low-level languages



• Defenses against memory errors





Address Sanitizer

- The primary source of vulnerabilities
 70% severe security bugs in Chrome
- Tradeoffs due to constraints from overhead and compatibility.

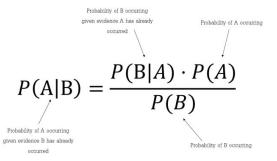
Question: How to accurately evaluate the security effectiveness of defenses?

Theoretical Validation

Experimental Validation

Theoretical Validation

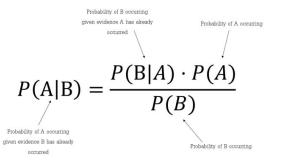
Experimental Validation



- Showing Probabilities
 - Hash Collision
 - Bypassing ASLR

Theoretical Validation

Experimental Validation

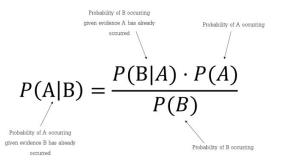


- Showing Probabilities
 - Hash Collision
 - Bypassing ASLR

GAP between T. and P. (•_•`

Theoretical Validation

Experimental Validation







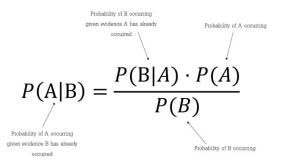
Real-world Case Studies

- Showing Probabilities
 - Hash Collision
 - Bypassing ASLR

GAP between T. and P.

Theoretical Validation





CVEs



- Showing Probabilities
 - Hash Collision
 - Bypassing ASLR

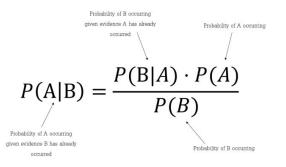
GAP between T. and P.

Limited Scope
 Biased Choice

Real-world Case Studies

Theoretical Validation

Experimental Validation





CVEs

Showing Probabilities

- Hash Collision
- Bypassing ASLR

GAP between T. and P.

Real-world Case Studies

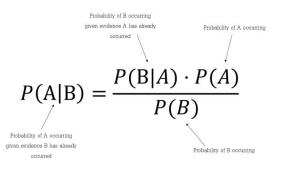
Limited Scope
 Biased Choice

Benchmarks



Synthesized test cases

Theoretical Validation



Experimental Validation

Common Vulnerabilities and Exposures

CVEs

- Showing Probabilities
 - Hash Collision
 Bypassing ASLR
- GAP between T. and P.

Real-world Case Studies





Synthesized test cases



• Runtime Intrusion Prevention Evaluator(RIPE), ACSAC'11

• RIPE generates hundreds of test cases via five dimensions

• Runtime Intrusion Prevention Evaluator(RIPE), ACSAC'11

• RIPE generates hundreds of test cases via five dimensions

RIPE is widely used in evaluating memory error defenses' effectiveness
LBC, COTS CFI, Griffin, Fine-CFI, etc.

• Runtime Intrusion Prevention Evaluator(RIPE), ACSAC'11

• RIPE generates hundreds of test cases via five dimensions

RIPE is widely used in evaluating memory error defenses' effectiveness
LBC, COTS CFI, Griffin, Fine-CFI, etc.

> "the exploits are all very similar; ... the sole purpose of overflowing buffers; ... it makes strong assumptions about the compiler and the operating systems.", <u>High system-code security with low overhead</u>, Wagner, SP'15

• Runtime Intrusion Prevention Evaluator(RIPE), ACSAC'11

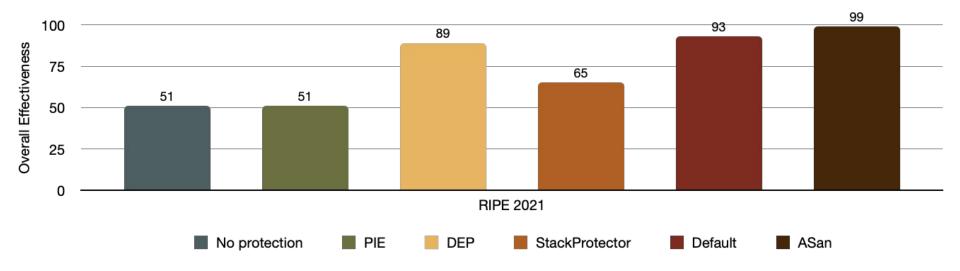
• RIPE generates hundreds of test cases via five dimensions

RIPE is widely used in evaluating memory error defenses' effectiveness
LBC, COTS CFI, Griffin, Fine-CFI, etc.

> "the RIPE testbed only considers a limited number of buffer overflow vulnerabilities. There could be many other types of vulnerabilities and exploit skills in practice", <u>Finding Cracks in Shields</u>: On the Security of <u>Control Flow Integrity Mechanisms</u>, Yuan, CCS'20

RIPE(2011) Performance in 2021

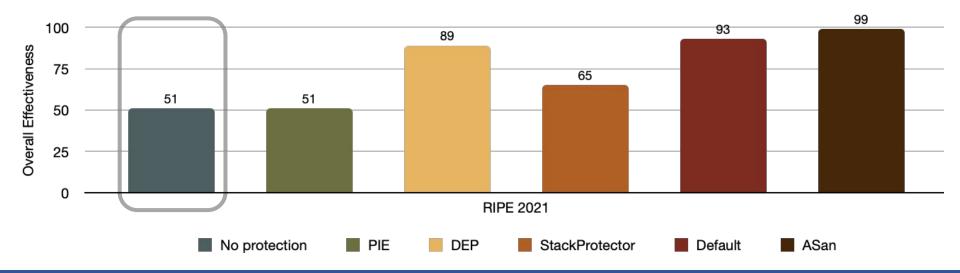
Effectiveness = Attacks/Exploits Prevented Rate



RIPE(2011) Performance in 2021

Effectiveness = Attacks/Exploits Prevented Rate

🗩 Inaccurate Baseline Result - 51% (many RIPE exploits fail)

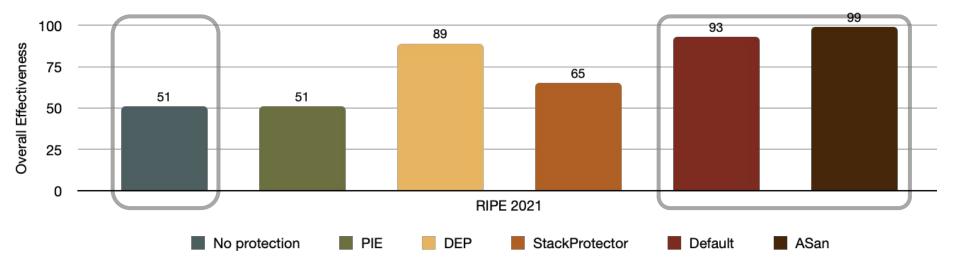


RIPE(2011) Performance in 2021

Effectiveness = Attacks/Exploits Prevented Rate

🗩 Inaccurate Baseline Result - 51% (many RIPE exploits fail)

=> Overrated Effectivenesses on defenses - over 90%



RecIPE

An extensible and comprehensive successor to RIPE

Design Goals

• Easy to extend and customize



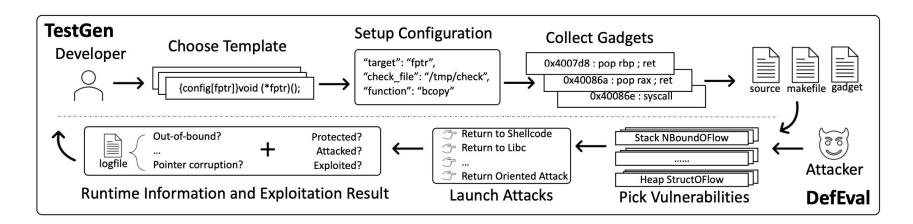
• Accurate and comprehensive measurement

• Support both 32 and 64 bits architecture

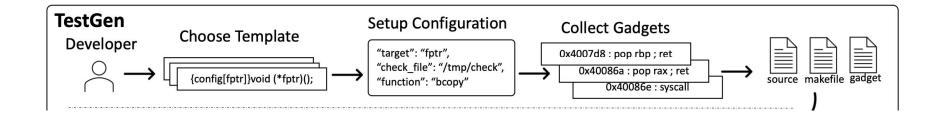
Benchmark Overview

Two Components:

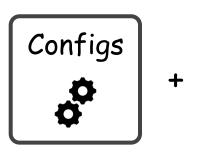
- TestGen: Generating Testcases to Individual Folders
- DefEval: Mimicing Attackers; Giving Analysis Results

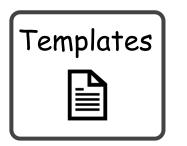


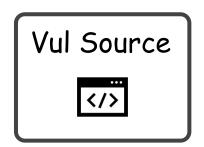
TestGen

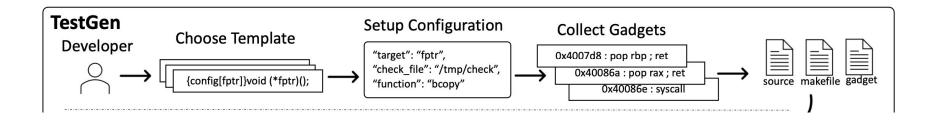


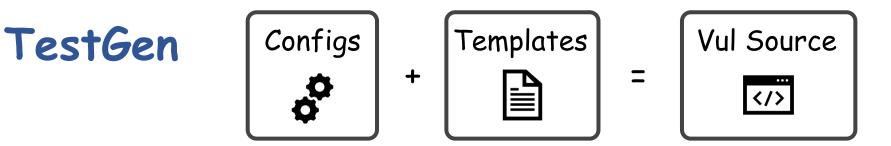
TestGen



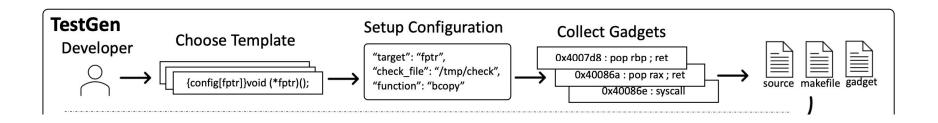








Configurable Templates => Extensibility & Customizability

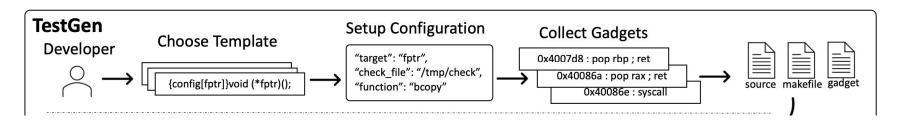




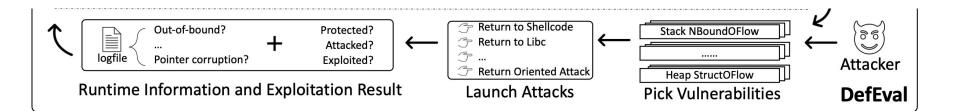


Configurable Templates => Extensibility & Customizability

- 斉
- Challenges:
- To be compatible with various attributes
- To switch context under various scenarios



DefEval

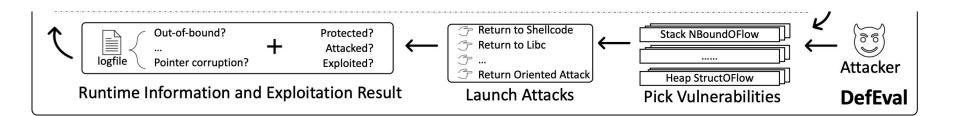




Attacker

Equipped with exploits:

- return-to-shellcode
- return-to-libc
- ROP, SROP





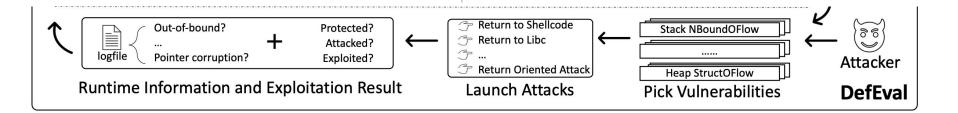
Attacker

Equipped with exploits:

- return-to-shellcode
- return-to-libc
- ROP, SROP



- Runtime Log => Attacked?
- File Existence => Exploited?





. Attacker

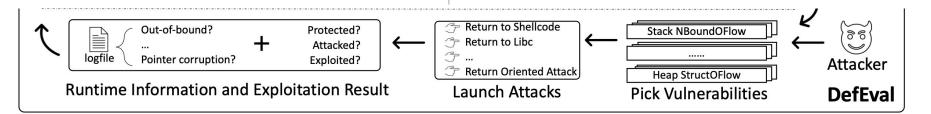
Equipped with exploits:

- return-to-shellcode
- return-to-libc
- ROP, SROP

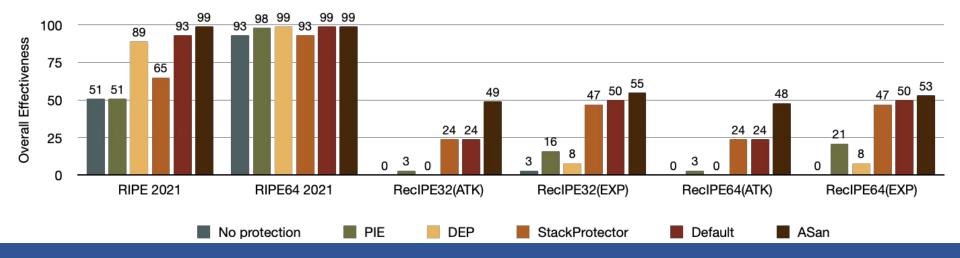


- Runtime Log => Attacked?
- File Existence => Exploited?

Attack? + Exploit? => Effectiveness Results

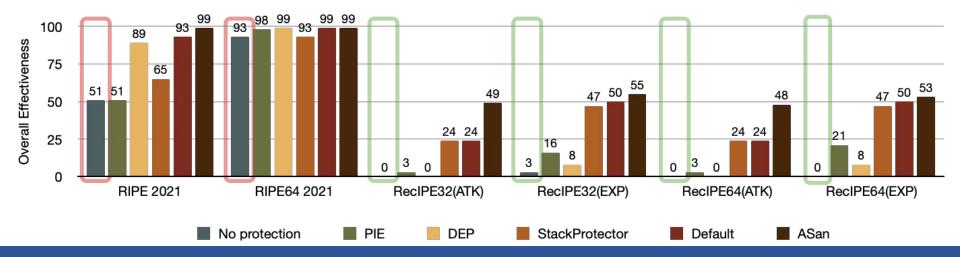


Effectiveness Evaluation



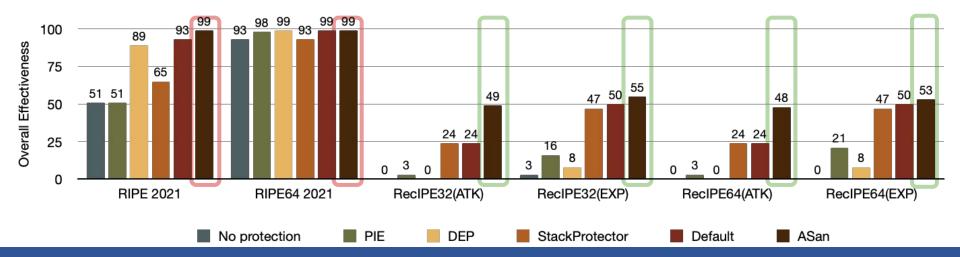
Effectiveness Evaluation

⊙ Accurate Baseline — No Protection 0%



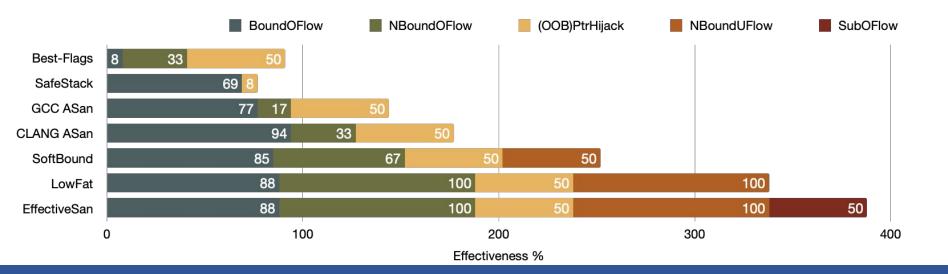
Effectiveness Evaluation

- ⊙ Accurate Baseline No Protection 0%
- Moderate Score ASan 50%
 => Reserve space for stronger defense



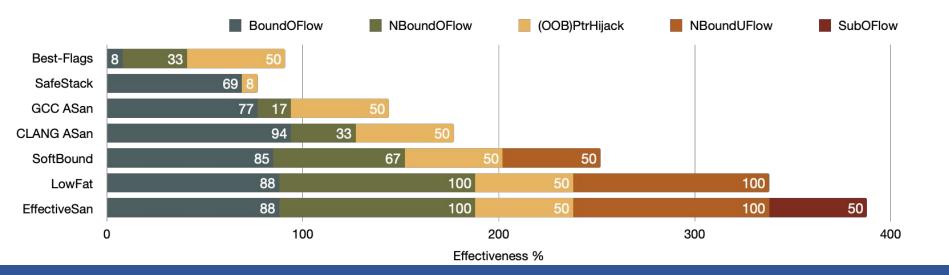
Effectiveness towards various defenses

• Clearly present pros and cons of each defense



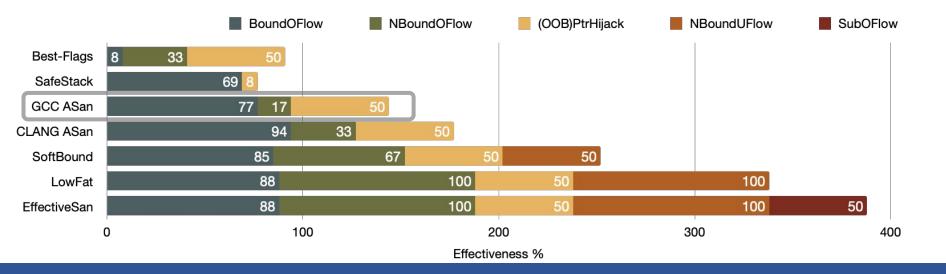
Effectiveness towards various defenses

- Clearly present pros and cons of each defense
- Help to understand limitations:



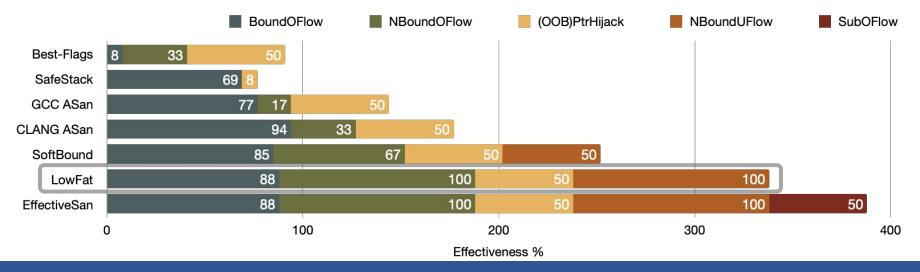
Effectiveness towards various defenses

- Clearly present pros and cons of each defense
- Help to understand limitations:
 - ASan does not protect non-linear OOB

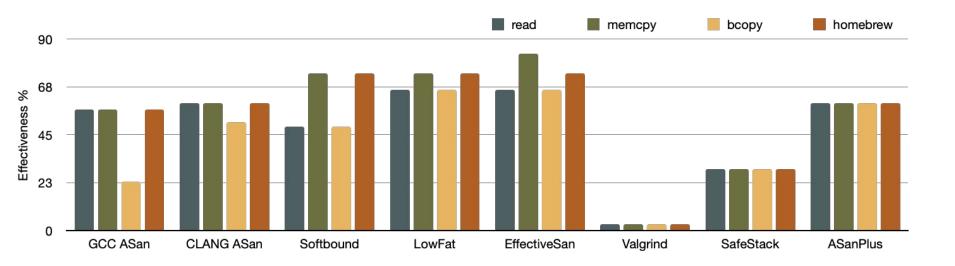


Effectiveness towards various defenses

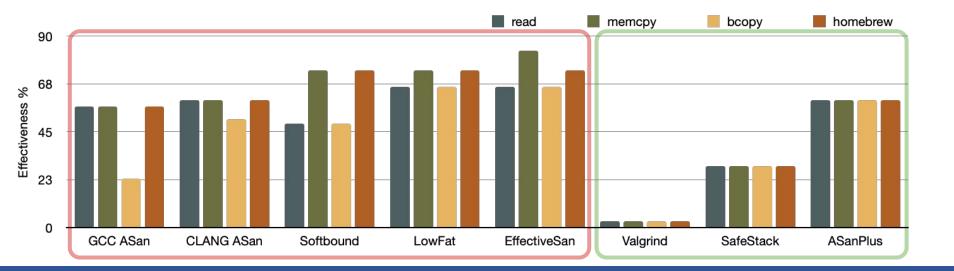
- Clearly present pros and cons of each defense
- Help to understand limitations:
 - ASan does not protect non-linear OOB
 - LowFat does not protect Sub-Object Overflow



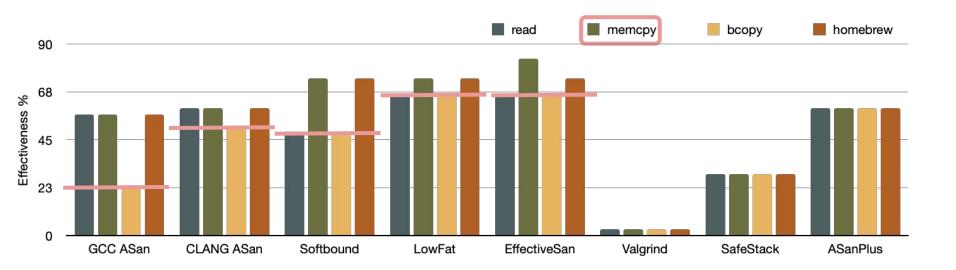
• Special Hardenings on Special Functions



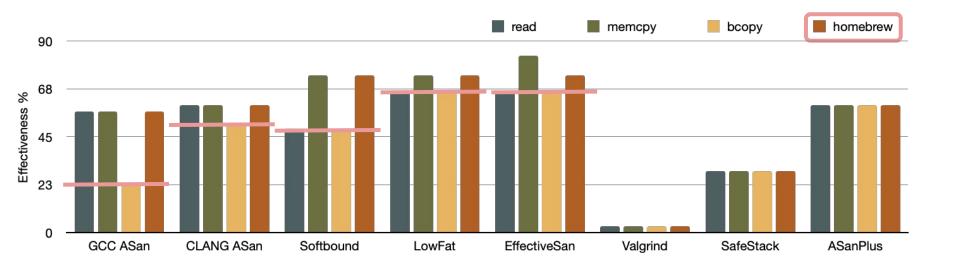
• Special Hardenings on Special Functions



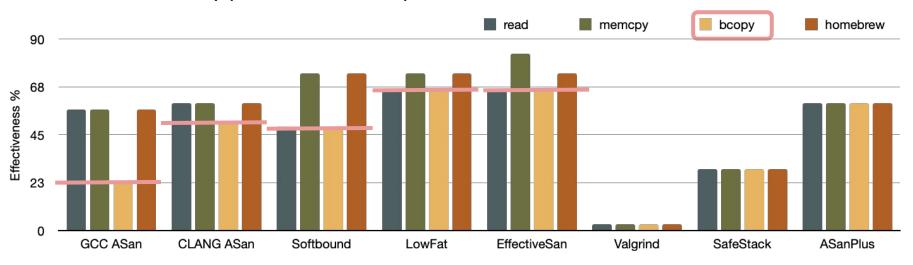
- Special Hardenings on Special Functions
 - o memcpy => "memcpy_safe"



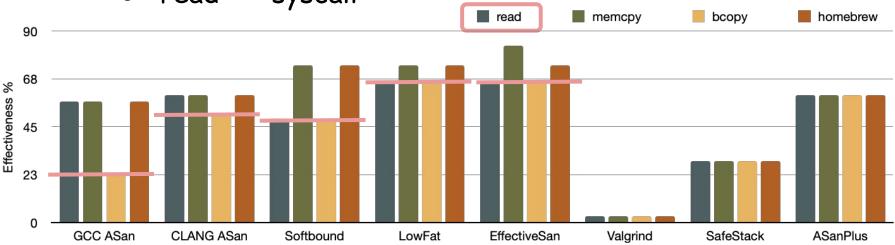
- Special Hardenings on Special Functions
 - o memcpy => "memcpy_safe"
 - homebrew memcpy => get "instrumented"



- Special Hardenings on Special Functions
 - o memcpy => "memcpy_safe"
 - homebrew memcpy => get "instrumented"
 - bcopy => not widely considered



- Special Hardenings on Special Functions
 - o memcpy => "memcpy_safe"
 - homebrew memcpy => get "instrumented"
 - bcopy => not widely considered
 - o read => "syscall"



More Detailed Results

Table 1: Evaluating security flags and sanitizers from region and technique attributes

Regi	on						Sta	ick								He	ap					Glo	bal				
Techn		Bound			dOFlow		OFlow		ndUFlow		trHijack		Hijack	Bound			dOFlow	SubC			lOFlow		ndOFlow	SubO		Countral	Meaning
Defense	Compiler	ATK	EXP	ATK	EXP	ATK	EXP	ATK	EXP	ATK	EXP		EXP	ATK	EXP	ATK	EXP	ATK	EXP	ATK	EXP	ATK	EXP	ATK	EXP		
										-	Securit	ty Flags	3	-											-	0	All ATK/EXP Fail
Baseline	gcc clang				:				:										:		:		:	:	•	•	Half ATK/EXP Fail
	gcc		-	-	-		-	-	-				C 6	-	-		-		-			-	-			•	All ATK/EXP Succeed
FullRELRO	clang	•	è.	•	•	•	•	•		•	e6	•	66	•	•	•	•	•	•	•	•	•	•	•	•	Super	Defense Strategy
DEP	gcc	•	€6	•	•	•	•	•	•	•	•	٠	•	•	•	•	•	•	٠	•	•	•	•	•	٠	01	Memory Layout Change
DEF	clang	•	0 6-	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	01*	clang Memory Layout
PIE	gcc	•	⊖ ⁸ ⊙ ⁸ -	•	•	•	•	:	•	C ⁵	C ³	•	C ⁸	•	•	•	•	•	•	:	•	•	•	•	•	<u>O</u> ²	Guard/Redzone checks
	clang	• •		• •	• •	•	-	-	-		01			•	-	-	-	-	-	•	-	-				O ³	Bound Checks
FORTIFY	gcc clang	•2 •2	04	e ²	e ²	•2	•2 •2	•2 •2	•2 •2		of	•2	•2	•2 •2	•2	•2	•2	-2 •2	•2	•2 •2	•2	•2	•2	-2 •2	•2	O ⁴	Compiler Hardening
StackPrtAll	gcc	ei	O ^{Î,2}	oî	0Î	•	•	•	•	া	0Î	•	•	•	•	•	•	•	•	•	•	•	•	•	•	05	Seperate Stack
StackFILAII	clang	€1	01,2	●1*	• ¹ *	•	•	•	•	O ^{1*}	01*	•	•	•	•	•	•	•	•	•	•	•	•	•	•	06	Memory Permissions
Default	gcc	¢1	○ ^{1,2} ● ⁶	01	01	•	•	•	•	01	01	•	C ⁸	•	•	•	•	•	•	:	•	•	•	•	•	07	Dynamic Instrumentation
	clang	• •1,2	01,2	• •	• •		-			● ○ ¹⁸	0168		e68				-		-	•	-					O ⁸	Randomization
Best-Flags	gcc clang	C1,2	01,2	€ ¹ *	€ ¹ *				:	018	0168		68													O ⁹	Allocator Change
SafeStack	clang	05	05	•	•	•	•	•	•	¢.	C-	•	•	•	•	•	•	•	•	•	•	•	•	•	•	Sub	Defense Weakness
											Sanit	tizers															
ASan	gcc	O1,2,4	O1,2,4	€1,4	€1,4	•	•	•	•	01,2,4	01,2,4	•	C4	02,4	$O_1^{2,4}$	•	•	•	•	02,4	$O_1^{2,4}$	•	•	•	•	O 1	Missed Check
Aban	clang	01,2,4-	O ^{1,2,4} -	O1,4_	O ^{1,4} -	•	٠	•	•		○1,2,4	•	€4	02,4	C ^{2,4}	•	•	•	•	02,4	C2,4	•	•	•	•	O 2	Analysis Limitation
ASanPlus	gcc	01,2,4,6	○1,2,4,6	01,4	O ^{1,4}	•	٠	•	•	○1,2,4	01,2,4	•	€4,6,8	02,4	02,4	•	•	•	٠	02,4	02,4	•	•	•	•	Suffix	Other Reasons
	clang	01,2,4,6	01,2,4,6	01,4	O ^{1,4}	•	•	•	•	01,2,4	01,2,4	•	€ ^{4,6,8}	○2,4	02,4	•	•	•	•	○2,4	02,4	•	•	•	•	0-	ATK/EXP Not Robust
Valgrind	clang	•	C.	•	•	•	•	•	•	•	€9	•	€9	0	07	•	•	•	•	•	•	● ●1,3,4	1,3,4	•	•	-	THICK IS IN THOUSAGE
SoftBound	clang	O ₁ ^{1,3,4}	O ₁ ^{1,3,4}	01,3,4	01,3,4	•	•			01,3,4		•	0-	€1,3,4	$\Theta_{1}^{1,3,4}$	€1,3,4	$\Theta_{1}^{1,3,4}$	•	•	$\Theta_{1}^{1,3,4}$	$\Theta_{1}^{1,3,4}$	1	~1	•	•		
LowFat	clang	01,3,4	0 ^{1,3,4}	01,3,4	01,3,4	•	•	01,3,4	~	01,3,4	○1,3,4	•	69	● ^{3,4}	● ^{3,4}	03,4	03,4	•	•	€1,4	$\Theta_{1}^{3,4}$	03,4	03,4	•	•		
EffectiveSan	clang	01,3,4	01,3,4	01,3,4	01,3,4	e_1^3	e_1^3	01,3,4	01,3,4	01,3,4	01,3,4	•	€9	$\Theta_{1}^{3,4}$	○3,4	03,4	03,4	€ ^{3,4}		€3,4	⊖ ^{3,4}	O ^{3,4}	○3,4	● ^{3,4}	€ ^{3,4}		
ESanPlus	clang	01,3,4	O ^{1,3,4}	01,3,4	01,3,4	e_{1}^{3}	e_{1}^{3}	01,3,4	○1,3,4	01,3,4	01,3,4	•	€6,9	$\Theta_{1}^{3,4}$	⊖3,4	03,4	O ^{3,4}	€ ^{3,4}	$e_{1}^{3,4}$	● ^{3,4}	O ^{3,4}	⊖3,4	⊖3,4	€ ^{3,4}	€ ^{3,4}		

More Detailed Results

Table 1: Evaluating security flags and sanitizers from region and technique attributes

Degi								Stack								Hea				-			Global																						
Regie Techni		Boun	dOFlow	NBou	undOF1/	w Si			oundUFlow		PtrHijach	Ptel	lijack	Bound	OFlow			Sub	OFlow	Bonn	dOFlow		oundOFle	ow Su	hOFlow	-						_													
	Compiler		EXP	ATK				P ATE		ATK		ATK		ATK			EXP			ATK			K EXP		K EXP	1 [Symbo	l Me	eaning			1													
						1		1		1		ity Flags						1						1		ĪĒ	0	All	1 ATK/E	XP Fail		_													
Baseline	gcc	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	٠	•	•	٠	٠	•	•	1 1	•		alf ATK/			-													
Dasemie	clang	•	e -	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	. 1	•		I ATK/E			-													
FullRELRO	gcc clang	:	:		:		:	:	:	:	C°	:	C°	:	:	:	:	:	:	:	:		:	1	:		Super	_	fense St			=													
	gcc	-	6 6	-							-			-			-		-								Ol	_			21	=													
DEP	clang	•	•	•									•	•	•	•		•		•							01*		emory L			_													
PIE	gcc	•	●8	•	•	•	•	•	•	€ ⁸	€ ⁸	•	€ ⁸	•	•	•	•	•	•	•	•	•	•	•	•	1	02		ang Mer			_													
PIE	clang	•	O ⁸ -	•	٠	•	•	•	•	€8	€8	•	€8	•	•	•	•	•	٠	•	•	•	٠	•	•		0 ³		ard/Red		hecks	_													
FORTIFY	gcc	• ¹ ₂	0	01	02	•2	2 •2	•2	•2	01	01	•2	•2	•2	•2	•2	•2	•2	•2	•2	•2	•2	•2	•2	•2	1 -	04		ound Che			_													
	clang	•2 •1	01,2	2	- <u>C</u> 2	2	2 •2	•2			02	-2	•2	0 2	9 2	2	2	•2	2	•2	2	•2		•2	•2		05		ompiler I		ing	_													
StackPrtAll	gcc clang	e1	01,2	€1*	A1*					01+	01+																		perate S			_													
	gcc	C ¹	01,2	01	01	÷				OI	01	÷	C ⁸	•		•	•	÷	÷	÷		÷			÷	1	06	Me	emory P	ermissi	ions	_													
Default	clang	•	●6	•		•		•	•	•										т	ahle	2. F	valu	ating	T Sec			e an	d sar	nitiza	ers fr	rom	tard	et an	d tec	hnia	110 91	ttribu	ites						
Best-Flags	gcc	€ ^{1,2}	01,2	01	O 1	•	•	•	•	0											abie	2. 1	varu	aime	See	unity	mag	5 an	iu sai	111120	CI 3 II	Um	tai 5	ct an	u ice	ming	uc a		neo						
	clang	€1,2	01,2	€1*	€ ¹ *	•	•	•	•	0																																			
SafeStack	clang	G 2	O2	•	•	•	•	•	•	•		Target				Ret	urn Ado	dress				OldE	EBP					Fun	action Po	inter						Longji	np Buffe	er		GOT		Exit		Hook	
		1,2,4	A1,2,4	●1.4	■1.4						Te	echnique			dOFlow	NE	BoundOl	Flow	(OOB)	PtrHija		Bound			ndOFlow												(OOB)PtrHijac		B)PtrHija		OB)PtrHija)B)PtrHija	ck
ASan	gcc clang	01,2,4	01,2,4	01,4	01.4						Defens	ie Co	ompiler	ATK	EXP	AT	к ех	XP	ATK	EXP	A	ГК	EXP	ATK	EXP	AT	K EXI	P /	ATK EX	XP AT	гк ех	(P	ATK	EXP	ATK	EXP	ATK	EXP	ATF	EXP	AT	K EXP	AT	K EXP	
0.0 0.00	gcc		01,2,4,6		01,4																						5	Security	y Flags																
ASanPlus	clang	01,2,4,6	01,2,4,6	01,4	01,4			•	•	0	Baselir		gcc	•	•	•	•		•	•	•		•	•	•	•	•		• •	•	•		•	•	•	•		•	•	•	•	•	•	:	
Valgrind	clang	•	¢.	•	٠	٠	٠	•	٠	•	1000000000		0	•					-	-	•		0-					-		•						-		-							_
SoftBound	clang	O1,3,4	O ₁ ,3,4	01,3,4	~1		•		^{3,4} ⊖ ^{1,3,4}		FullREL		gcc clang										0												-					06					
LowFat	clang	01,3,4	⊖ ^{1,3,4}	01,3,4	~		•	01,3	~				gcc		6 6				-	-	-		6									-	-	-		-	-								-
EffectiveSan	clang	01,3,4	○1,3,4		01,3				3,4 01,3,4		DEP		lang	•	06					•			0-										•	•											
ESanPlus	clang	01,3,4	01,3,4	01,3,4	01,3	.4 C	${}^{3}_{1} \bullet^{3}_{1}$	01,3	3,4 01,3,4	0			gcc	•	●8-	•	•		•	•	•		0 ⁸	•	•	•	•	-		•	•	-	•	•	•	•	•	•	●8-	08-	•	•	•	•	-
											PIE		lang	•	●8-	•			•	•	•		O ⁸ -	•	•	•	•	•	• •	•	•		•	•	•	•	•	•	€8-	O ⁸ -	•	•	•	•	
											FORTI	ZV.	gcc	•2	•2	•2	•	2	Θ_2^1	Θ_2^1	•2		0-	€ ⁴ ₂	e_2^4	e_2^4	\mathbf{e}_2^4	•	•2 •		•		•2	• ₂	•2	•2		Θ_2^1	Θ_2^1		Θ_2^1		Θ_2^1		
											FORTI	· ·	clang	•2	•2	•2	•	2	•2	•2	•2		0-	•2 ²	•2	e_2^{a}	e_2^4	•	•2 •		•		•2	•2	•2	•2	\ominus_2^{I}	\odot_2^{I}	Θ_2^2			2	● ¹ ₂	\ominus_2^{I}	
											StackPrt		gcc	•	01,2	•	•		01	● ¹	•		O ¹ -	C1	•1	•1	C1	•	• •	01	•		•	•	01	01	O ¹	\odot^1	01	⊖î	• ¹	● ¹	⊖ ¹	● ¹	
										F		(clang	•	O ^{1,2}	•	•		01	01	•		O1-	C1	e1	€ ¹	C'		•••	01	01		•	•	•	•	•	•	● ¹	●1 ○1,6,8	•	• •	● ¹	● ¹	_
											Defau		gcc	:	1,2				0.+0	01,2			01- 0-		0.					0.		. 1		:	0.	0.		•	-	01,0,0		•	-	<u> </u>	
										H					01,2				A1.2	01,2			0-	C1			.						-	-		01	e	-	01,8	01,6,8					_
											Best-Fla		gcc clang		01,2				1,2	01,2			0-	e1	e1	C1	C ¹				i õi	i			ě	ě	0 1	1	1,8						
										ŀ	SafeSta		clang	05	05					•	05		05			•				•			•	•	•	•	-5	6 5	•				ě		-
										ŀ	Jacoba			-			-			-				1				Sanit								-	1.								-
										F			gcc	01,2,4	01,2,4	•			€1,4	€1,4	0	1,2,4	O ^{1,2,4} -	01,2,4	0,1,2,4	4 C1.4	4 C1,4	•	• •	• • 1	• •1	1	•	•	•	•	●1	\odot^1	●1	\odot^1	●1	● ¹	●1	04	-
											ASan		clang	01,2,4_	01,2,4				€1,4	€1,4	0	,2,4_	01,2,4_	01,2,4	01,2,4	4 01,4	4 €1,4	4		01	•	i	•	•	01*	01*	●1	\odot^1	●1	\odot^1	●1	\odot^1	●1	O ⁴	
										F	ASanPl		gcc		○1,2,4		•		€1,4,	€1,4			○1,2,4,6				4 C ^{1,4}	•	• •	• •1	• • •	. I	•	•	01	O1	●1	\odot^1	●1	$O^{1,6,8}$			● ¹		
												(clang	01,2,4,6		.6	٠		€1,4,	€1,4	01	1,2,4,6	○1,2,4,6	01,2,4,4	6 O ^{1,2,4}	4,6 01,4	4 C1,	•	• •	01	•		•	•	\bigcirc^1	O^1	●1	\odot^1	●1	$O^{1,6,8}$	-	\odot^1	\odot^1	Ŷ	
											Valgrir		clang	•	€7-	٠	•		٠	•	•		•	€²	€7	•	٠		• •	•	•		•	•	•	٠	•	•	•	•	٠	•	•	ಿ	
											SoftBou	nd o	clang	01,3,4	○1,3,4		~		€1,3,4	01,3,4	- 10		01,3,4	€1,3,4	● ^{1,3,4}	ª ⊖1,:	^{3,4} ⊖ ^{1,2}	3,4 (• •	●1	- U		~1	$\Theta_{1}^{1,3,4}$	01,3,4	01,3,4	€1,3,4	0	●1,3,					3,4 O-	
											LowFa	it c	clang	01,3,4			,3,4 ()		€1,3,4	-1			○1,3,4	01,3,4	<u> </u>						,3,4 ⊖			01,3,4	01,3,4		€1,3,4	<u> </u>	10					3,4 01,3,4	
											Effective		clang	○1,3,4	01,3,4		,3,4 ()						○1,3,4	01,3,4					€ ^{3,4} €	3,4	,3,4 🔿			○1,3,4	01,3,4		€1,3,4							3,4 01,3,4	
										[ESanPl	us o	clang	○1,3,4	01,3,4	01	,3,4 ()	1,3,4	€1,3,4	● ^{1,3,4}	01	1,3,4	$O^{1,3,4}$	01,3,4	⊖1,3,4	4 01,3	3,4 01,3	3,4 (0 ^{3,4} 0	^{3,4} ⊖ ¹	,3,4 🔿	,3,4	01,3,4	○1,3,4	01,3,4	○1,3,4	€1,3,4	$\Theta_{1}^{1,3,4}$	€1,3,	4 O1,3,4	4,6 ⊖1,	,3,4 ⊖ ^{1,3,•}	4 01,3	3,4 01,3,4	4.9



- We propose a newly designed benchmark, RecIPE, for evaluating memory error defenses' effectiveness.
- RecIPE is extensible, comprehensive, and accurate.
- RecIPE helps to understand the security coverage of memory error defense and even implementation details.
- RecIPE is available at

<u>https://github.com/YuanchengJiang/recipe-benchmark</u>

Thanks

