Why fuzz about security?

Mathias Payer infosec.exchange/@gannimo







Fact 1: Software Has Bugs

Barne Annevation / Brewity

Microsoft: 70 percent of all security bugs are memory safety issues

Percentage of memory safety issues has been hovering at 70 percent for the past 12 years.



fritten by Estatin Gimpane, Contributor on Seb. 9, 9015



syzbot	Sinu:	-		
· Open 1310-	1	involence.	· Fixed (+621)	 Invalid

	Name	TWE WEAK	Lingune					e (
	di gama sentrova	104	31544				- 10	
	ri-genu sodarum-20	104	20lb					-
	dime-forme-in	****	340	1				1
	ri-panet Armiti	104	294	and the second second		411	H٩	-
	sirgened worth coupel.	104	3424			111	10	1
	oter-Förma-Eurerep-in	10w	304	and the second second	-	<u> </u>	· · ·	4
	di-permu2-chart4	100	4.000	ing locon		1000		
	si apstevam hpi hasan poo	Mar	440.	2444	1000.4	. K.	MARP	
	d-apernam-top-pettikasaa-pen		474	217-6	12903	- 10	22-09	54
	di apoletoam pos-armi-4	hiter.	4.985	9614	13853	43	118411	- 55
	G-aps/rsam-por-Isak	104	11.00	540.00	arrient.		10-94	10
	d-apoleoana baoan-goo	101	(artista	+0+0	152015		30030	1.0
	d-upstram-kasas-eon-386	104	27/0e	34040	28.080	12	1.0120	14
	singplerometheoastpressol.	104	8934	10.00	1980.8		10010	14
	ri-apstroam kasaa-aon salinare roct	104	29th	3424	1826X.T		32/94	14
	ci-applements instant-poptisticacies con-	108	340e	7208	GLINA.	28	481346	
	ci-spatroam lansan grut	104	40%	404	426962	120	28089	14
nis Mitch	singelummeringen 300	1.0w	4.000	15.2018	15.00.5	1.0	2018	15
	d-apetream-linex-nect-kasan-pos-root	tile/	34%	7100	1/2289	- 16	36476	
	di-appletoans peri-lastion-port	100	7.08	3049	12004		38380	
	si upstream net tais uasan ges	10w	Tel Sa	1000	43.2525	10	13430	
	GIDITVADO		2108	240.00	10.000	246	6.04910	1
	då-spetroan lossen ope	Admi-	3454	5404	18/24/8		5 (948)	1.0
	G-up/renub	104	PLC P		28.4		2010	

	ci-aperovato epo-menti isassa-pero		40%	121-4	124034		22-04	121204-02109-0518	change in the	#07e	SUBMER	1000
	ci apoletoam pes-armi/4	Report	4.000	96546	12012	43	118411	Showed STREETS		ACCO.	1120-040	0.00
	G-aps/cham.pcm/bs8k	104	21,000	541.0	17567.0	-	10-94	MONTH NOTICE	1000.00	1104	10.000000	10000
	d-apotroanabaran-poo	100	2 mills	+0+0	112015		30030	NUMBER OF TAXABLE	camfile.	Date of	SCE and	1.00
	d-upstrnam-kanas-goo-386	10w	3700A	34040	26.0973	12	1.01249	NONE YOUR	4439534	276-8	11.0-010	1000
	singstrometheous press.	10w	8936	10.00.00	1960.8		10010	Sales and South As	100534	press.	SMAR	1.09.
	ri-apstroam kasaa-ane salinaro mat	now	200	3424	16268.7		32/94	MARKEN POLICY	starf is	216 de	CO. Balance	64995
	ci-appletoatte laanate-goot tettakck-1008	100	340a	7208	11011	28	481346	two/reactions	combin	1004	DUDwink	4.00.
	ci-spates are less an gru	10w	448	454	(2002)	120	28089	Mex81812-04	.cofia	5.00	10.5440	64995
nis Mitch	singplummeringsargaw-303	104	4400	15.2016	15.00.5	1.0	2015	Mademacht	100534	1.04	SHAM	1.09
	d-apetream-linex-nect-kasan-pra-root	hiter .	3456	7100	1/2099		36476	081626-81	.coffie	140 Min	CC5-read	64991
	ci-upsersampret-lassam-port	100	7.08	3049	12204		38380	HERMONY.	camfile.	PED-	Dibetel	4.00
	di upstenam net Galo sanan geo	10w	Tel Sa	1009	0.2525	10	13434P	ablics.#Chila	and in	and the	COLUMN	6.00
	GSDITMERS		2004	2008	10000	246	6.0499	teams will be	cambas.	2100	Skillende	1000
	di upetroamienen ore	Adapt.	34.04	5444	285348		5 9489	GOOD MILES	.confile	5165-ba	005444	4.00
	(C-sp/manach	104	MOR.	100	385.6		1030	March 440	1000.00	10.0	1AD BIRD	10701
	CKO K	E				~					•	
d	Crop			У	9	S	e				91 y	

Fact 2: Many Bugs are Exploitable



Fact 3: Software is Incredibly Complex

Google Chrome: 76 MLoC

Gnome:

9 MLoC

Xorg/Wayland:

glibc:

Linux kernel:

17 MLoC

1 MLoC

Margaret Hamilton with code for Apollo Guidance Computer (NASA, '69)



Chrome and OS ~100 MLoC, 27 lines/page, 0.1mm/page ≈ 370m

Software Security: Key Research Questions

- RQ1: *Efficiently* detect security violations
- RQ2: Automatically generate test cases
- RQ3: *Scale* testing to *large* source repositories
- RQ4: *Effectively* test complex interfaces
- RQ5: *Mitigation* co-design based on feedback



hexhive

Software Testing

- Goal: prune bugs
- A tool for developers



Mitigation

- Goal: stop exploitation
- Last line of defense



Compartments

- Goal: least privilege
- Divide & conquer security



Fuzzing in a Nutshell

\$./testme --help
Usage: testme <int32_arg>

\$./testme AAAA
Please enter an integer!



```
$ cat fuzzer.sh
while :
do
    len=$(($RANDOM % 255))
    input="$(dd if=/dev/urandom bs=$len count=1)"
    ./testme $input || echo $input >> crash_seeds
done
```

Fuzzing: Automated (Fuzz) Testing



Effective Fuzzing 101

Test cases must *reach bugs*

Exploration through coverage-guided fuzzing

The fuzzer must *detect bugs*

• Exploitation through sanitization and triaging

Performance is key (zero sum game)!

• Finite cycles/time, must spend resources wisely!





Greybox Fuzzers: A Genealogy



datAFLow: Toward a Data-Flow-Guided Fuzzer. Adrian Herrera, Mathias Payer, and Antony Hosking. In TOSEM'23 FishFuzz: Catch Deeper Bugs by Throwing Larger Nets. Han Zheng, Jiayuan Zhang, Yuhang Huang, Zezhong Ren, He Wang, Chunjie Cao, Yuqing Zhang, Flavio Toffalini, and Mathias Payer. In SEC'23

Spill the TeA and TEEzz: Trusted Applications on Android Devices



From Crashes to Ranked Bugs

Fuzzing produces (many) crashes, mapping to real bugs

- Programmers are overwhelmed by large amount of crashes
- Crashes need to be distilled into bugs to be useful
- Bugs need complete descriptions

Our findings

- Minimizing path length of seeds enables similarity matching
- Igor groups 254'000 crashes across 39 bugs into 48 distinct clusters
- Evocatio summarizes bug capabilities, bypasses 7 out of 16 CVEs

Igor: Crash Deduplication Through root-Cause Clustering Zhiyuan Jiang, Xiyue Jiang, Ahmad Hazimeh,
 Chaojing Tang, Chao Zhang, and Mathias Payer. In CCS'21.
 Evocatio: Conjuring Bug Capabilities from a Single PoC Zhiyuan Jiang, Shuitao Gan, Adrian Herrera, Flavio Toffalini,
 Lucio Romerio, Chaojing Tang, Manuel Egele, Chao Zhang, and Mathias Payer. In CCS'22.

Fuzzing is Maturing. What's next?

Metrics for starting seed corpora and how to generate them

Cross-distillation to reuse seeds among targets

Stateful programs and network protocols

Handling (diverse) peripherals for embedded systems

Helping developers cope with inferred information

Distinguishing exploration and exploitation phases



Conclusion



EPFL

Join us on this research journey!



















Join the Software Security Fun Ride!

Bugs are ubiquitous and a (re-)growing resource:

- Software testing weeds them out early
- Mitigations stop attack classes
- Compartmentalization limits their impact

Our research focuses on:

- Specializing fuzzing to new environments
- Enable developers to "understand" bugs
- Customize mitigations per-program
- Infer strong compartmentalization mechanisms





Mathias Payer (infosec.exchange/@gannimo)