Manta: Hybrid-Sensitive Type Inference Toward Type-Assisted Bug Detection for Stripped Binaries

1 Appendix

1.1 Type Lattice

Figure 1 shows the lattice used in our type inference for 64-bit binaries. In the lattice, numeric types include integers and floating-point numbers, while pointer type is a subtype of the 64-bit register.

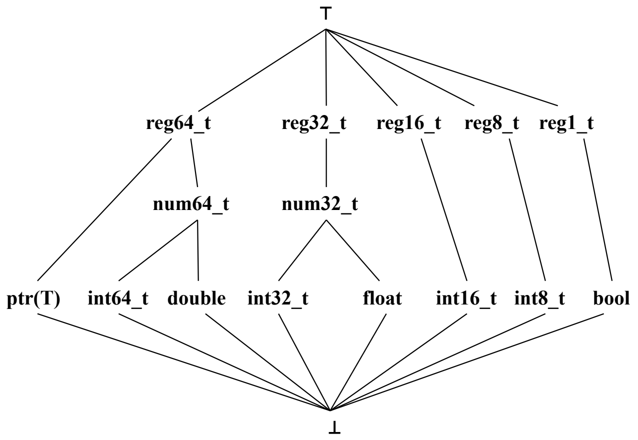

Figure 1. Type lattice in Manta

1.2 Firmware Samples

Table 1 shows the detailed information of our benchmarks, including the firmware series, the binary used for evaluation, and the size and architecture of the binary.

1.3 Detailed Data of Infeasible Data Dependency Pruning Experiment

Table 2 shows the detailed FP, FN, TP of the program slicing result.

Table 1. Firmware samples.

Vendor	Series	Binary	Size(KB)	Arch
Netgear	SXR80	net-cgi	1413	AARCH64 (LE)
Zyxel	NR7101	zhttpd	351	MIPS32 (LE)
Tenda	A15	httpd	913	MIPS32 (LE)
TRENDnet	TEW-755AP	ssi	1027	MIPS32 (LE)
ASUS	RT-AX56U	httpd	551	ARM32 (LE)
TOTOLink	LR350	cstecgi.cgi	215	MIPS32 (LE)
TOTOLink	NR1800X	cstecgi.cgi	250	MIPS32 (LE)
TP-Link	TL-WR940N	httpd	1874	MIPS32 (LE)
H3C	Magic R200	webs	659	MIPS32 (LE)

1.4 Confirmed Bugs

Table 3 shows the list of bugs that have been confirmed by the developers, including the firmware series on which the bug is found, the kind of bug, and the details of assigned CVE/PSV IDs.

A friendly reminder that currently, reviewers are advised against searching for the CVE/PSV IDs on the internet, as it may lead to the disclosure of author information and violate the double-anonymous guidelines.

1.5 Vulnerability Specification

Table 4 lists the specifications for the five vulnerabilities.

Table 2. Program slicing result compared with Pinpoint in 14 large-scale open source projects. `Report` denotes the number of source-sink pairs detected by Pinpoint. `FP` denotes false positives, `FN` denotes false negatives, and `TP` denotes True positives when taking the result of Pinpoint as ground truth.

Project	#Report	Dirty [2]			Ghidra [1]			Retypd [4]			RetDec [3]			Manta											
														FI			FS			FI + FS			**FI + CS + FS**		
		#FP	#FN	#TP	#FP	#FN	#TP	#FP	#FN	#TP	#FP	#FN	#TP	#FP	#FN	#TP	#FP	#FN	#TP	#FP	#FN	#TP	#FP	#FN	#TP
vsftpd	0	7	0	0	18	0	0	18	0	0	1	0	0	13	0	0	7	0	0	0	0	0	0	0	0
libuv	5	4	4	1	4	5	0	4	4	1	0	5	0	1	4	1	4	4	1	1	4	1	1	4	1
memcached	13	25	1	12	23	10	3	24	1	12	1	12	1	10	1	12	16	1	12	9	1	12	8	1	12
lighttpd	18	86	10	8	0	18	0		△		0	18	0	54	10	8	54	10	8	24	10	8	9	10	8
tmux	78	41	26	52	35	47	31				2	78	0	33	20	58	48	25	53	30	25	53	24	20	58
openssh	20	76	5	15	75	12	8	71	11	9	28	12	8	28	5	15	74	4	16	26	5	15	27	5	15
wolfSSL	3	51	1	2	53	2	1	54	1	2	4	3	0	37	1	2	45	1	2	4	1	2	4	1	2
redis	192	160	71	121	155	102	90		△		1	192	0	170	46	146	128	51	141	99	47	145	87	52	140
libicu	168	225	56	112	217	68	100				12	167	1	12	167	1	189	59	109	147	55	113	149	58	110
vim	175		‡		287	114	61				4	174	1	187	70	105	286	73	102	107	76	99	89	76	99
python	146				141	70	76				0	146	0	93	54	92	100	75	71	45	63	83	37	44	102
wrk	126	181	53	73	166	47	79				2	126	0	55	95	31	171	41	85	93	39	87	87	20	106
ffmpeg	187	2468	47	140	2334	95	92				7	186	1	206	45	142	1763	41	146	161	46	141	112	48	139
php	264	291	167	97	223	193	71				7	258	6	268	135	129	280	116	148	169	107	157	147	96	168
Total.	**1,395**	**3,615**	**441**	**633**	**3,731**	**783**	**612**	**171**	**17**	**24**	**69**	**1,377**	**18**	**1,326**	**541**	**854**	**3,165**	**501**	**894**	**915**	**479**	**916**	**781**	**435**	**960**
Rate.	-	**85.1%**	**41.1%**	**14.9%**	**85.9%**	**56.1%**	**14.1%**	**87.7%**	**41.5%**	**12.3%**	**79.3%**	**98.7%**	**20.7%**	**60.8%**	**38.8%**	**39.2%**	**78.0%**	**35.9%**	**22.0%**	**50.0%**	**34.3%**	**50.0%**	**44.9%**	**31.2%**	**55.1%**

△ denotes the type inference cannot finish analysis in 72 hours.

‡ denotes the type inference crashes.

Table 3. Confirmed bugs and assigned CVE and PSV IDs.

Vendor	Device Series	Type	Vulnerabilities IDs
Netgear	SXR80	NPD	PSV-2022-165, PSV-2022-166, PSV-2022-167, PSV-2022-168, PSV-2022-169 8 unassigned, fixed
		BOF	PSV-2023-0077
Zyxel	NR7101	BOF	CVE-2023-27989 2 unassigned, pending fixed
Tenda	A15	BOF	CVE-2022-47115, CVE-2022-47116, CVE-2022-47117, CVE-2022-47118 CVE-2022-47119, CVE-2022-47120, CVE-2022-47121, CVE-2022-47122 CVE-2022-47123, CVE-2022-47124, CVE-2022-47125, CVE-2022-47126, CVE-2022-47127, CVE-2022-47128
TP-Link	WR940N	RSA	4 unassigned, not fixed
TRENDNet	TEW-755AP	NPD	3 unassigned, fixed
		RSA	1 unassigned, fixed
		CI	CVE-2022-45597, CVE-2022-45598
		BOF	CVE-2022-45580, CVE-2022-45581, CVE-2022-45582, CVE-2022-45583 CVE-2022-45584, CVE-2022-45585, CVE-2022-45586, CVE-2022-45588 CVE-2022-45589, CVE-2022-45590, CVE-2022-45591, CVE-2022-45592 CVE-2022-45593, CVE-2022-45594, CVE-2022-45596, CVE-2022-45599, CVE-2022-45560, CVE-2022-45561
ASUS	RT-AX56U	RSA	1 unassigned, not fixed
		NPD	2 unassigned, fixed
TOTOLink	NR1800X	CI	CVE-2022-41518, CVE-2022-41525
		BOF	CVE-2022-41517, CVE-2022-41520, CVE-2022-41521 CVE-2022-41522, CVE-2022-41523, CVE-2022-41524, CVE-2022-41526, CVE-2022-41527, CVE-2022-41528
	LR350	CI	CVE-2022-44249, CVE-2022-44250, CVE-2022-44251, CVE-2022-44252
		BOF	CVE-2022-44253, CVE-2022-44254, CVE-2022-44255, CVE-2022-44256 CVE-2022-44257, CVE-2022-44258, CVE-2022-44259, CVE-2022-44260
H3C	MagicR200	NPD	1 unassigned, fixed

Table 4. Rules for finding source-sink bugs in MANTA.

Vulnerability	**Source**	**Sink**
❶ Null Pointer Dereference	constant zero value, null return function (e.g., `malloc()`)	Pointer Dereference
❷ OS Command Injection	`getenv()`, `gets()`...	`system()`, `popen()`...
❸ Use After Free	1st arg. of `free()`, `kfree()`...	Pointer Dereference
❹ Buffer Overflow	`read()`, `gets()`...	`strcpy()`, `sprintf()`...
❺ Return of Stack Address	`alloca()`	`ret` instruction

References

[1] N. S. Agency. Ghidra reverse engineering tool. https://www.nsa.gov/resources/everyone/ghidra/.

[2] Qibin Chen, Jeremy Lacomis, Edward J. Schwartz, Claire Le Goues, Graham Neubig, and Bogdan Vasilescu. Augmenting decompiler output with learned variable names and types. In *USENIX Security Symposium*, 2022.

[3] P. Matula J. Kˇroustek. Retdec: An open-source machine-code decompiler. Presented at Pass the SALT 2018, Lille, FR, July 2018.

[4] Matt Noonan, Alexey Loginov, and David Cok. Polymorphic type inference for machine code. *SIGPLAN Not.*, 51(6):27–41, jun 2016.