Understanding Binary Code Similarity for Real-World Vulnerability Detection A Large-Scale Empirical Study

FSE 2026

Jingdong Guo, Chaopeng Dong, Yimo Ren, Siyuan Li, Jie Liu, Hong Li, Hongsong Zhu
(Institute of Information Engineering, Hangzhou Dianzi University, Shandong University)

Why BCSD Matters for Firmware Vulnerability Detection

Third-party libraries in firmware

- Firmware widely reuses Third-Party Libraries (TPL) such as OpenSSL and zlib
- **Reused code propagates TPL vulnerabilities across vendors**

Binary Code Similarity Detection (BCSD)

- Compares a known vulnerable function against firmware code
- Extract features → encode functions → rank by similarity

Objective of this study

- **Understand what affects BCSD performance in real-world firmware at scale**

Introduction

Limitations of prior evaluations

- Small-scale datasets cover limited devices, vendors, and TPLs
- Prior work typically evaluates only **10–20 CVEs**
- Prior evaluations validate methods rather than analyze **performance factors**

Research Questions

1. How do different **vulnerable-function versions** affect BCSD performance?
2. How do different **search spaces** affect BCSD performance?
3. How does **function size** affect BCSD performance?

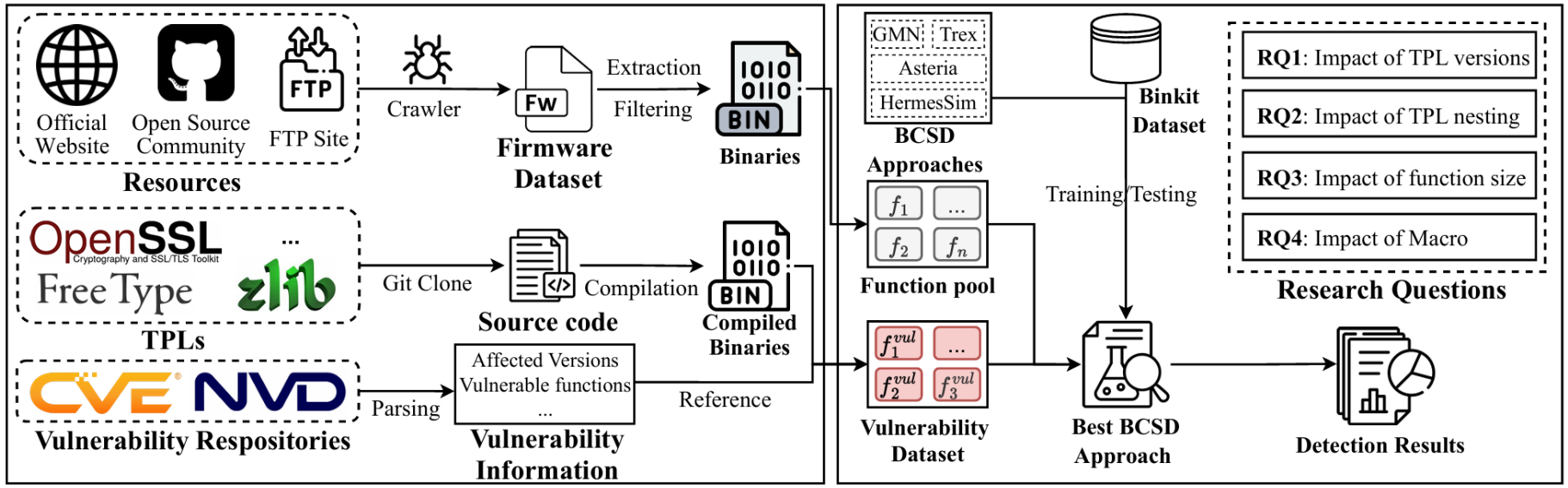
Study Design: Overview

Stage 1. Data collection and preprocessing

- Collect and preprocess firmware images to construct the firmware function pool
- Extract vulnerable functions and affected versions from CVEs, then compile the corresponding TPL source versions

Stage 2. Vulnerability detection and evaluation

- Evaluate BCSD approaches on BinKit, select the best-performing approach, and perform vulnerability detection



Study Design: Firmware Dataset

Scale

- **59,374 images from nearly 200 vendors; 46,802 unpacked**
- 4.4M binaries after deduplication

Characteristics

- Architectures dominated by ARM (45.62%) and MIPS (44.91%)
- 96.51% of binaries are stripped

Construction principles

- Metadata normalized into a unified manifest
- Checksum verification and hash-based deduplication

Table 1. Top 30 Vendors by Firmware Count and Other Statistics

Vendor	#Firm	Category			Filesystem				Architecture				32-bits (%)	Stripped (%)
		NAS	Router	Other	Ext	jffs2	Squashfs	Other	ARM	MIPS	x86	Other		
NETGEAR	4,604	389	1,012	3,203	29	216	1,972	2,387	349,695	193,010	115,409	5,770	84.00%	97.84%
D-Link	3,677	20	265	3,392	117	20	922	2,618	69,801	87,441	1,894	27,031	98.32%	98.98%
ASUS	3,657	6	1,192	2,459	3	99	1,973	1,582	469,426	202,064	25	365	99.55%	93.08%
Tomato	3,319	-	18	3,301	0	-	3,316	3	30,469	546,131	0	0	100.00%	100.00%
MikroTik	2,894	-	2,774	120	4	-	2,803	87	55,318	208,507	58,860	123,426	99.76%	97.62%
ZyXEL	2,471	26	46	2,399	69	178	206	2,018	34,360	54,246	96	4,814	99.88%	99.47%
Intel	2,398	-	0	2,398	1	101	889	1,407	376,400	1	1,807	177	99.62%	85.50%
Ubiquiti	1,207	-	204	1,003	0	45	293	869	37,742	188,282	0	735	98.57%	97.78%
Gargoyle	1,046	-	12	1,034	9	15	907	115	3,942	156,278	2,167	0	99.70%	99.85%
digi	1,007	-	0	1,007	4	64	18	921	666	10,046	738	5,900	93.48%	99.79%
TP-LINK	934	-	583	351	-	10	528	396	17,179	37,756	118	368	100.00%	98.97%
CANON	812	-	-	812	-	0	0	812	148	-	10	0	93.67%	94.30%
pelco	800	-	-	800	-	122	475	203	222,969	-	3,307	165	96.98%	96.35%
Schneider	555	-	-	555	-	4	1	550	1,918	1	118	1,647	98.51%	98.83%
TRENDnet	543	2	131	410	15	15	235	278	13,250	18,899	350	34	100.00%	98.96%
Supermicro	528	0	-	528	0	0	0	528	7,667	-	192	24	99.53%	100.00%
Dell	526	1	-	525	1	6	9	510	4,695	-	3,766	4,490	72.85%	99.95%
Conceptronic	367	55	31	281	24	1	71	271	1,485	5,357	39	469	100.00%	96.54%
Tplink	366	0	0	366	3	11	133	219	3,279	5,913	20	2	100.00%	99.91%
AirLive	356	3	9	344	76	2	97	181	20,890	4,986	3,215	-	99.99%	95.38%
Tenda	342	0	76	266	0	3	154	185	10,949	10,234	0	-	100.00%	100.00%
QNAP	341	322	-	19	1	-	2	338	5,466	112	746	-	100.00%	99.92%
UTT	327	-	-	327	0	-	3	324	0	20,677	0	4,040	100.00%	100.00%
Samsung	319	-	-	319	8	5	20	286	38,102	-	104	-	99.99%	99.73%
Hikvision	317	-	-	317	49	3	0	265	4,631	-	0	-	100.00%	100.00%
Wayos	307	-	-	307	6	0	2	299	0	27,351	585	6	96.81%	100.00%
idis	290	-	-	290	-	257	1	32	861	-	-	-	62.83%	100.00%
Sony	277	-	-	277	-	-	4	273	905	-	-	-	100.00%	100.00%
Pioneer	272	-	-	272	11	-	9	252	6,678	9	-	-	100.00%	96.83%
Belkin	260	-	153	107	1	-	123	136	3,200	14,502	66	-	100.00%	99.14%
Other	6,124	353	580	5,191	155	197	1,472	4,300	217,565	183,011	36,453	8,238	89.34%	98.33%

Study Design: Vulnerability Dataset

Each vulnerability consists of

- Vulnerable function name
- Affected versions
- Corresponding binary function used for matching

CVE Mining and Structuring

1. Regex pre-extraction
2. Prompted LLM extraction
3. Patch-message confirmation
4. Human verification

Compilation to ARM Binaries

- Oldest affected, latest affected, and several intermediate versions
- Older Docker images and manual dependency installation when required

Result

- **402 CVEs (2004–2024) across 9 TPLs**

Table 2. Summary of Vulnerability Metrics for TPLs. Abbreviations: "Vuln. Funcs" = Vulnerable Functions, "Comp. Vers." = Compiled Versions, "Comp. Bins" = Compiled Binaries.

TPL	TPL Category	# CVEs	# CWEs	# Vuln. Funcs	# Comp. Vers.	# Comp. Bins
OpenSSL	TLS/SSL and Crypto	118	32	139	141	282
libtiff	Image Codec	73	13	77	15	339
FreeType	Font Rendering	66	12	68	30	30
curl	Networking/Data Transfer	54	26	55	53	53
libexpat	XML Parser	28	11	32	16	16
BusyBox	System Utilities	20	11	20	14	14
libpng	Image Codec	19	9	26	23	23
binutils	Binary Toolchain	16	7	18	6	101
zlib	Compression	8	4	8	51	51

Study Design: Ground-Truth Labeling

Filter 1. Symbol-based function identification

- readelf parses each deduplicated binary's symbol table for function names
- Nearly 97% of binaries are stripped, but many still retain some function names

Filter 2. TPL version triangulation

- Curated regexes parse embedded version strings such as "OpenSSL 1.1.1l"
- Binaries without a parsable version string are excluded

Filter 3. Vulnerability label generation

- Label only if the symbol name matches a CVE function and the version is in range

Result

- **111,025 high-confidence triples: binary, function name, CVE-ID**

Study Design: Approach Selection and Workflow

Approach selection

- Baselines: GMN (CFG), Trex (micro-traces), Asteria (AST), HermesSim (SOG)
- **HermesSim achieves the best performance on BinKit in all settings**

Detection and evaluation workflow

- Disassemble binaries with IDA Pro and Ghidra, extract SOGs
- Encode every function into a fixed-length embedding
- Rank firmware candidates by cosine similarity per query
- Evaluate with MRR and Recall@K against the labeling dataset as ground truth

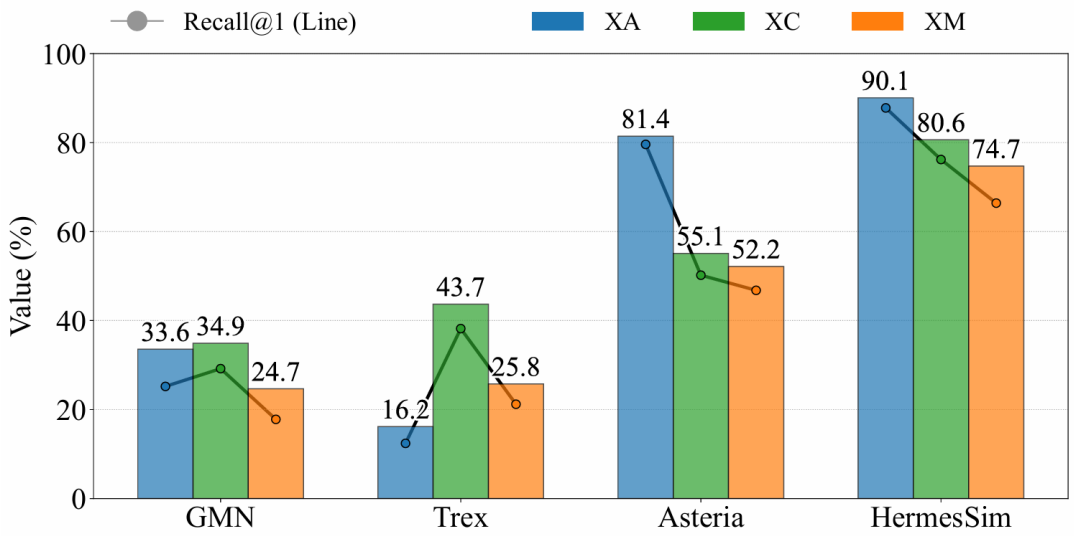


Fig. 3. Performance on the BinKit benchmark (trained and evaluated on BinKit using its standard split). Bars show MRR; points show Recall@1. HermesSim consistently leads across XA, XC, and XM.

Evaluation: RQ1 (Impact of TPL Versions)

Experimental setup

- V_same: versions exactly matching the firmware TPL
- V_oldest: oldest affected version
- V_newest: newest affected version

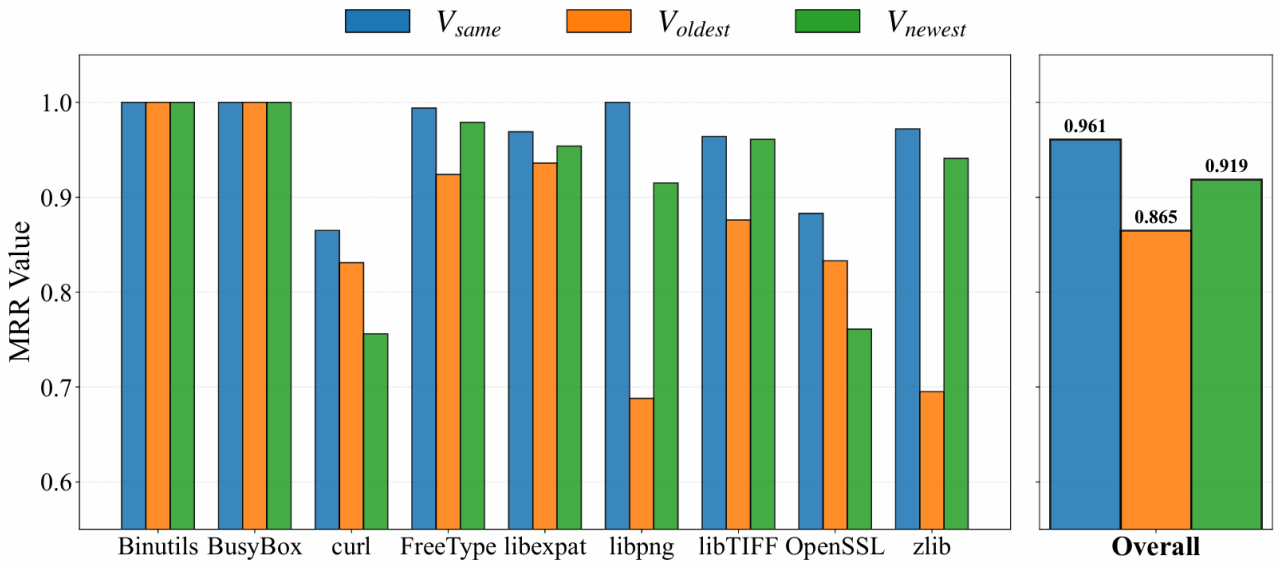


Fig. 4. Impact of different versions of vulnerable functions on BCSD performance.

Evaluation: RQ2 (Impact of Search Spaces)

Two settings

- Agnostic: entire firmware as search space
- Aware: restricted to the corresponding TPL

⇒ **Restricting the search space to the correct TPL reduces irrelevant-function noise**

Table 3. Impact of search spaces on libpng and OpenSSL.

TPL	Search	Recall@1	MRR
libpng	Agnostic	0.53	0.751
	Aware	**1.000**	**1.000**
OpenSSL	Agnostic	0.794	0.838
	Aware	**0.870**	**0.883**

Evaluation: RQ3 (Impact of Function Size)

- Function Size = End Address – Start Address
- Small functions (< 2 KB)
- Mid-sized functions (2–4 KB)
- Large functions (> 4 KB)

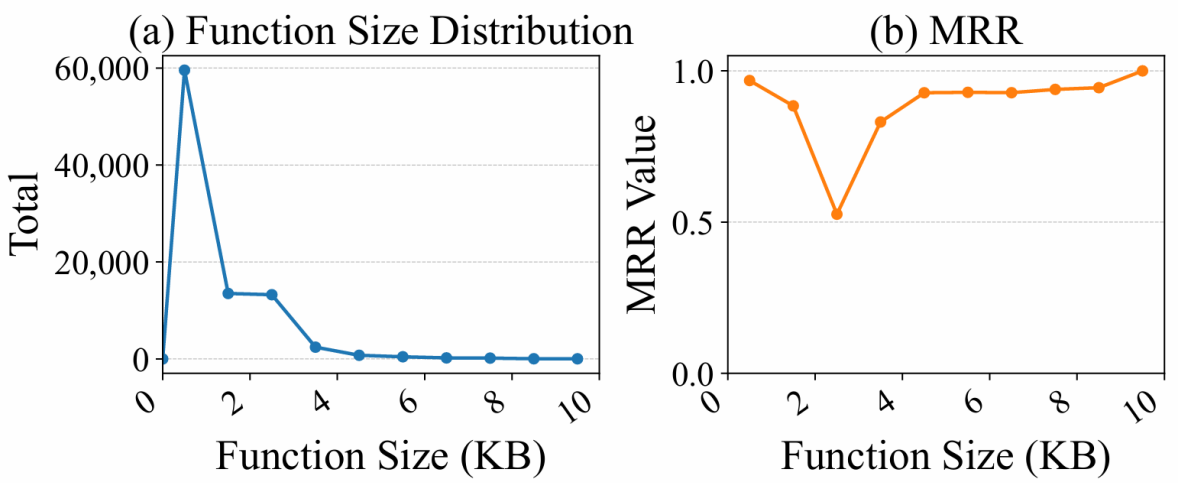


Fig. 5. Non-linear impact of function size on BCSD performance. (a) Long-tailed size distribution in our dataset. (b) U-shaped relationship between size and MRR, with a pronounced performance trough at mid sizes.

Conclusion

Contributions

- Largest firmware and vulnerability dataset: 46,802 images, 402 CVEs, 1,069 binaries
- First large-scale analysis of BCSD for firmware vulnerability detection across four factors

Personal Thoughts

- Queries are compiled for ARM only, yet 44.91% of the corpus is MIPS: cross-architecture matching is untested
- Ground truth depends on surviving symbols, biasing labels toward the least-stripped binaries
- All four RQs run on a single engine selected on BinKit, so the factor effects may be model-specific
- RQ2 presupposes correct TPL identification, and build archetypes come from one toolchain family

Thank You