Scaling up Vulnerability Analysis of IoT Devices with Heuristics and Binary Code Similarity

Advanced Penetration Testing Group
Dongkwan Kim @ Samsung SDS

Who Am I: Dongkwan Kim

- Passionate, self-motivated security researcher
- Education
  - KAIST Ph.D. ’22 (M.S. ’16 and B.S ’14)
- Newbie researcher
  - 7 top-tier papers (NDSS, USENIX Security, ACM CCS, …)
  - 19+8 papers, 713 citations (as of Oct. 21, 2023)
- CTF Player
  - Defcon finalist (’12, ’14, ’16, ’18, ’19)
  - CTF winner (Whitehat, HDCON, Codegate, …)

WHY IS IOT SECURITY IMPORTANT?

Internet of Things Devices Increase Risk of Cyber Attacks on Industrial Sector: Lloyd's

Practical IoT Hacking:
e Internet of Things.

Home » Cybersecurity » Cyberlaw » Cybercriminals Are Infiltrating Netgear Routers with Ancient Attack Methods

ent

Whistleblower: Ubiquiti Breach "Catastrophic"

CES 2021: Router swarms
i
where you are)

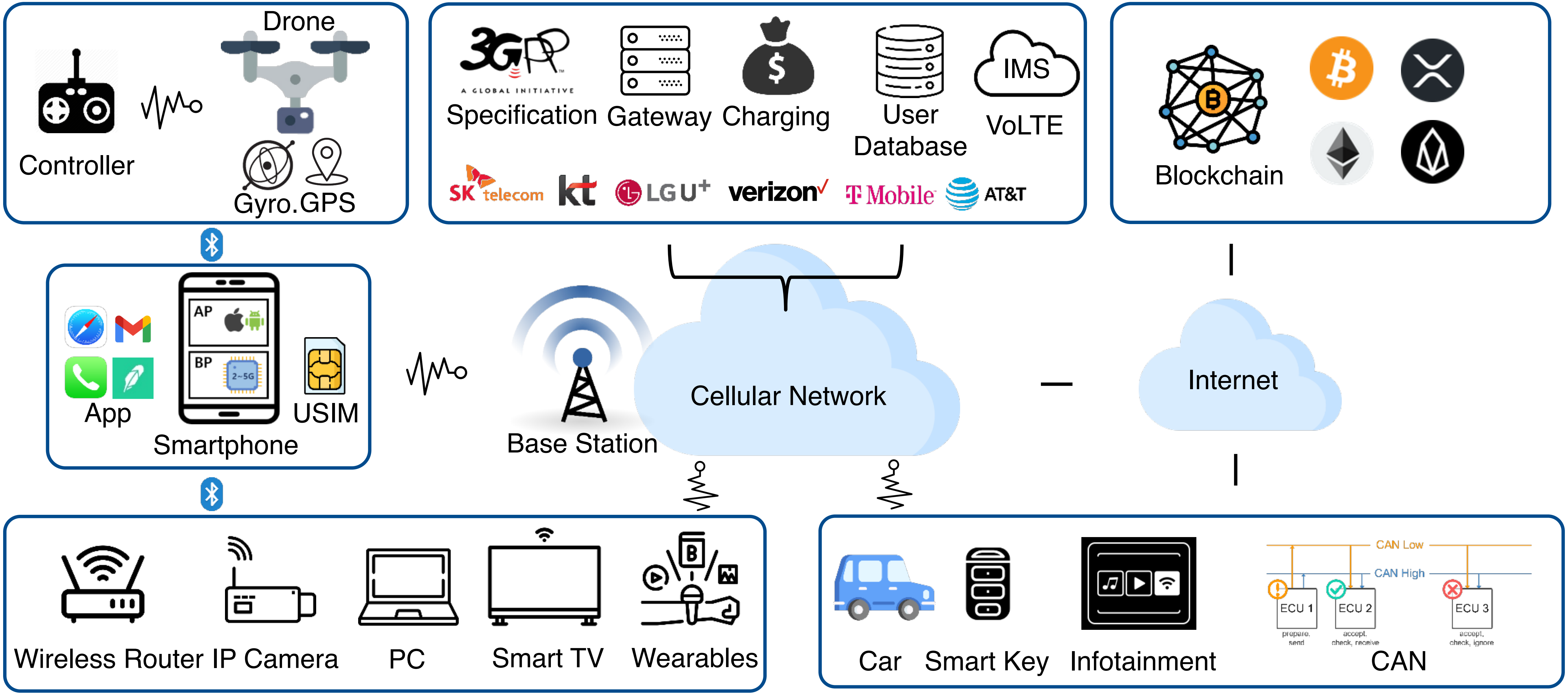
New Mirai Variant and ZHtrap Botnet Malware Emerge in the Wild

New mesh Wi-Fi routers may be the answer to your
and security?

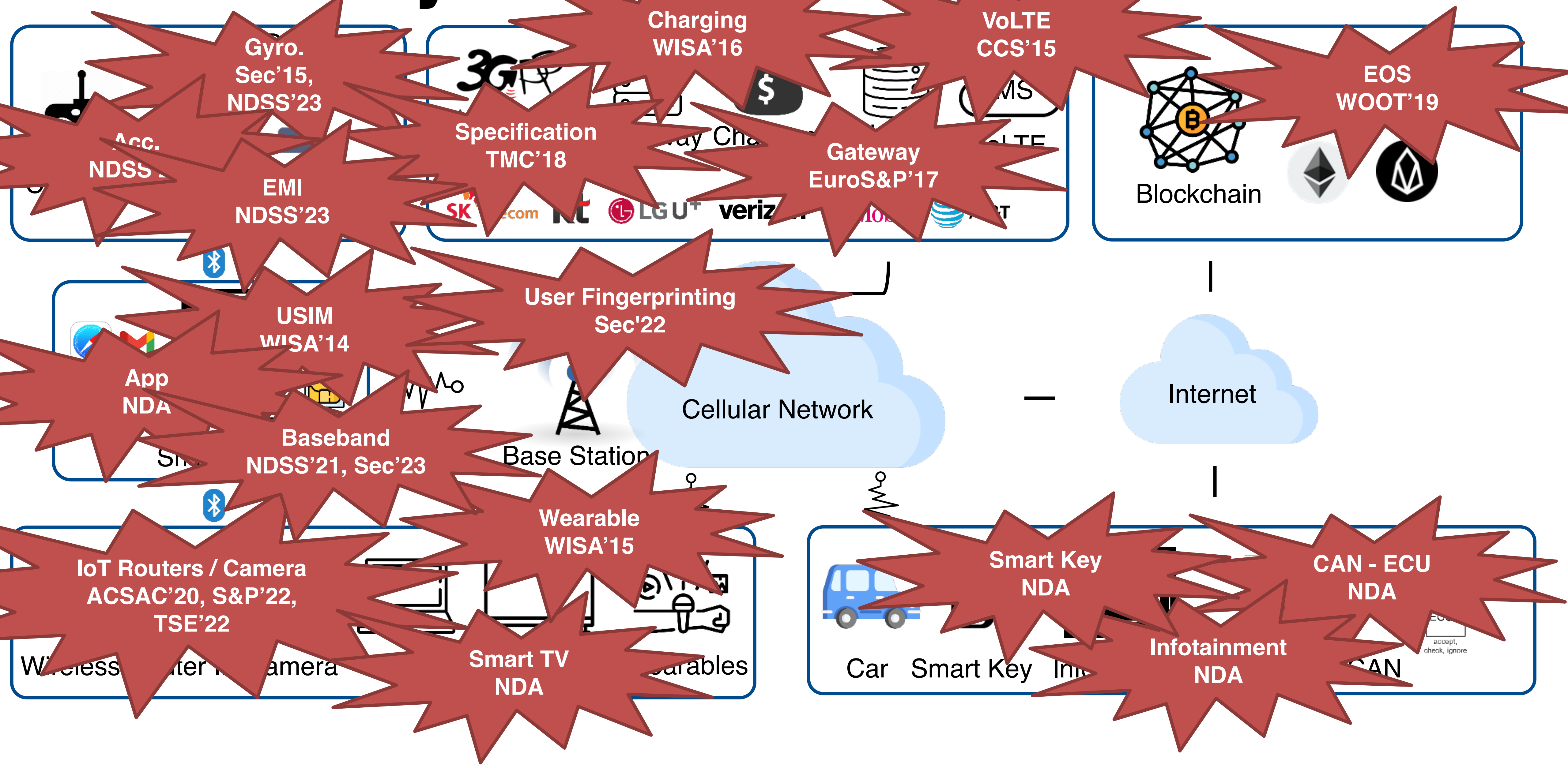
ISS Today >

Critical infrastructure attacks: why

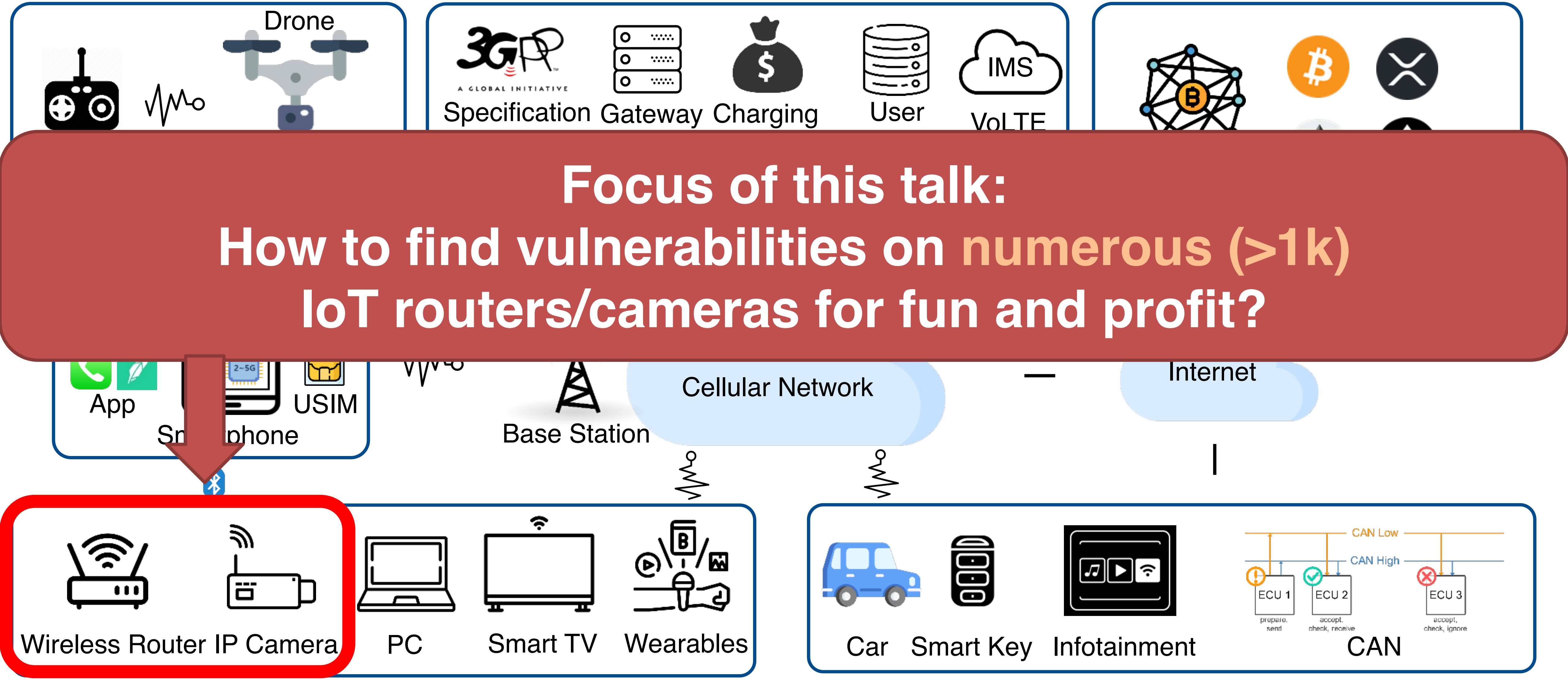
IoT Ecosystem (In)Security



IoT Ecosystem (In)Security



IoT Ecosystem (In)Security

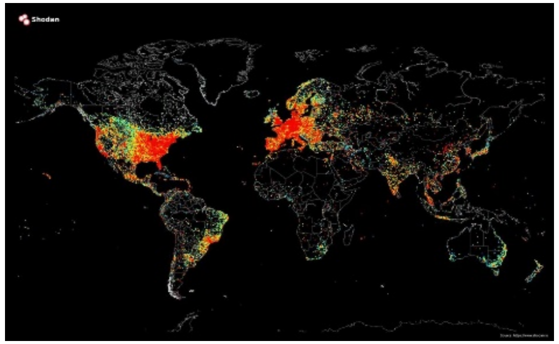


(In)Security of Linux-Based IoT Devices

- **34.2 billion** embedded devices will be in use in 2025*
  - Wireless routers, IP cameras, ...

- Many **botnets** target IoT devices
  - Mirai (Aug. 2016)
  - Satori (Dec. 2017)
  - Crypto (May. 2018)
  - ECHOBOT (Dec. 2019)
  - New Mirai variant (July 2020, 2021, 2023~)
  - ➔ DDoS attacks: DynDNS (2016), GitHub (2018), ...

- **Exposed to the Internet**, especially **web interfaces**
  - Shodan, ZoomEye
  - Over 30 exploits in Mirai variants



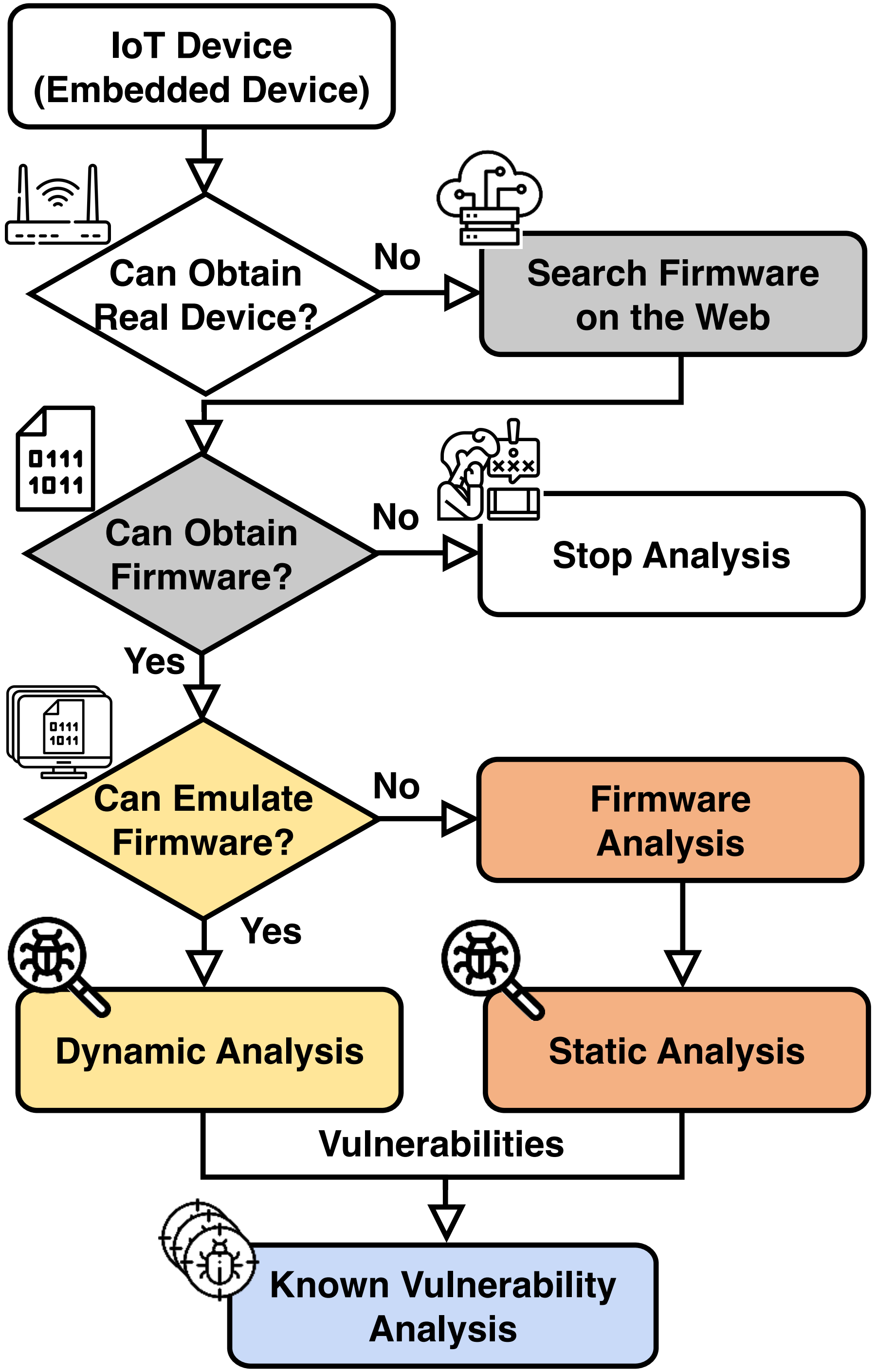
*https://iot-analytics.com/state-of-the-iot-update-q1-q2-2018-number-of-iot-devices-now-7b/

Challenges in IoT Security Analysis

- ❖ The number of IoT devices are **rapidly increasing**
- ➔ **Scalability is the key** to analyze their threats

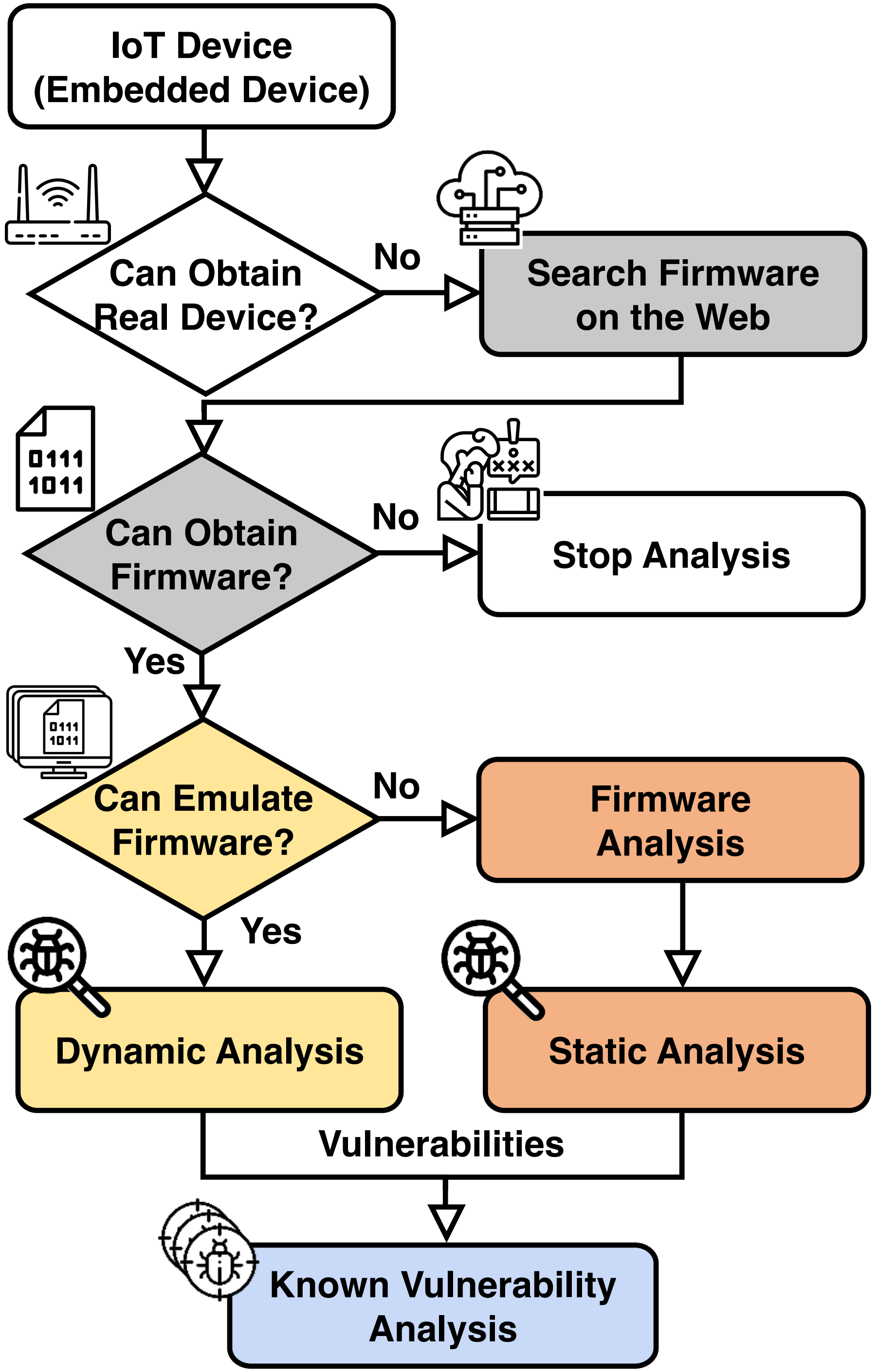
- ❖ Challenge: **no development standards**
  - – Opacity (Obscurity)
    - ▪ Vendors **do not release** implementation details
  - – Diversity
    - ▪ **Numerous vendors**, complex hardware/implementation diversity
- ➔ **Scaling up** the vulnerability analysis is **challenging**

IoT Analysis Procedure



IoT Analysis Procedure

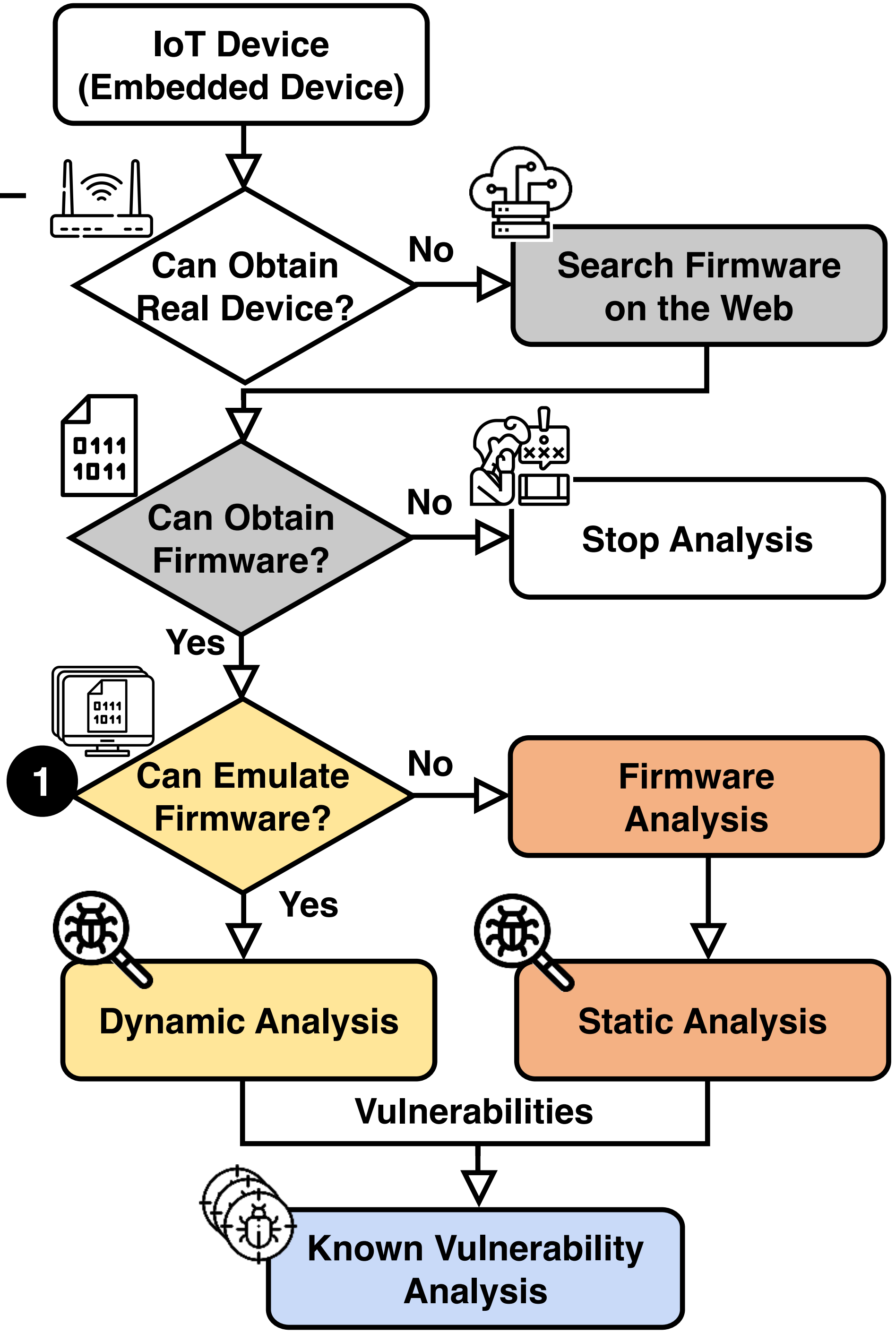
- Firmware collection
  - Physically obtaining numerous devices is infeasible
  - Download firmware images from vendors websites



IoT Analysis Procedure

- ❖ Firmware collection
  - Physically obtaining numerous devices is infeasible
  - Download firmware images from vendors websites

1. Firmware emulation and dynamic analysis
   - Build a virtual environment mimicking a real device
   - Run automated pentesting (e.g., Metasploit)
   - Run fuzzers (e.g., AFL)

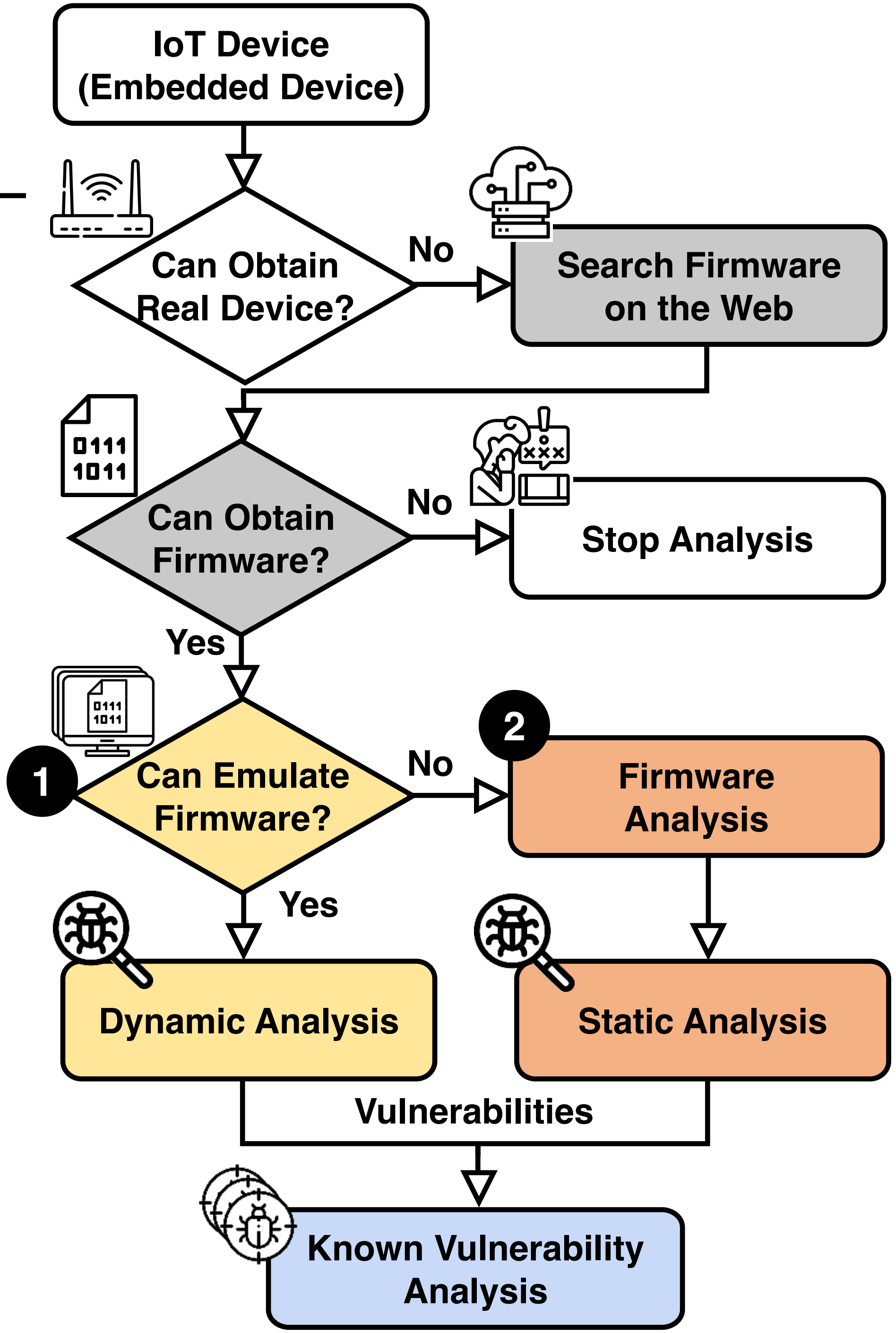


IoT Analysis Procedure

- Firmware collection
  - Physically obtaining numerous devices is infeasible
  - Download firmware images from vendors websites

1. Firmware emulation and dynamic analysis
   - Build a virtual environment mimicking a real device
   - Run automated pentesting (e.g., Metasploit)
   - Run fuzzers (e.g., AFL)

2. Firmware and static analysis
   - Analyze firmware structure and memory layout
   - Identify target functions
   - Run symbolic execution (e.g., angr)



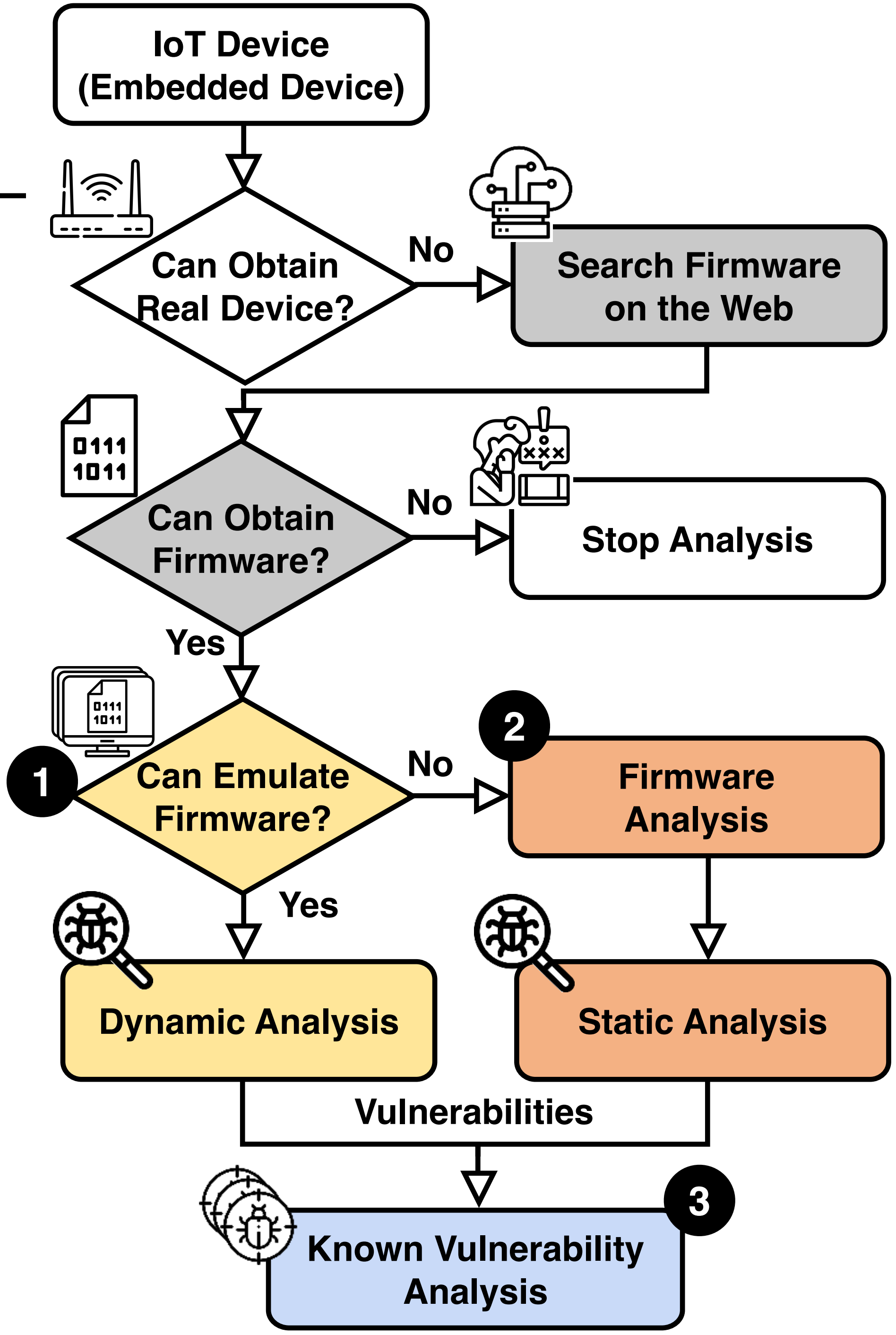
IoT Analysis Procedure

- ❖ Firmware collection
  - Physically obtaining numerous devices is infeasible
  - Download firmware images from vendors websites

1. Firmware emulation and dynamic analysis
   - Build a virtual environment mimicking a real device
   - Run automated pentesting (e.g., Metasploit)
   - Run fuzzers (e.g., AFL)

2. Firmware and static analysis
   - Analyze firmware structure and memory layout
   - Identify target functions
   - Run symbolic execution (e.g., angr)

3. Known vulnerability (1-day-based) analysis
   - Build PoC exploits and run them (e.g., Metasploit)
   - Build signatures and search them (e.g., BCSA)



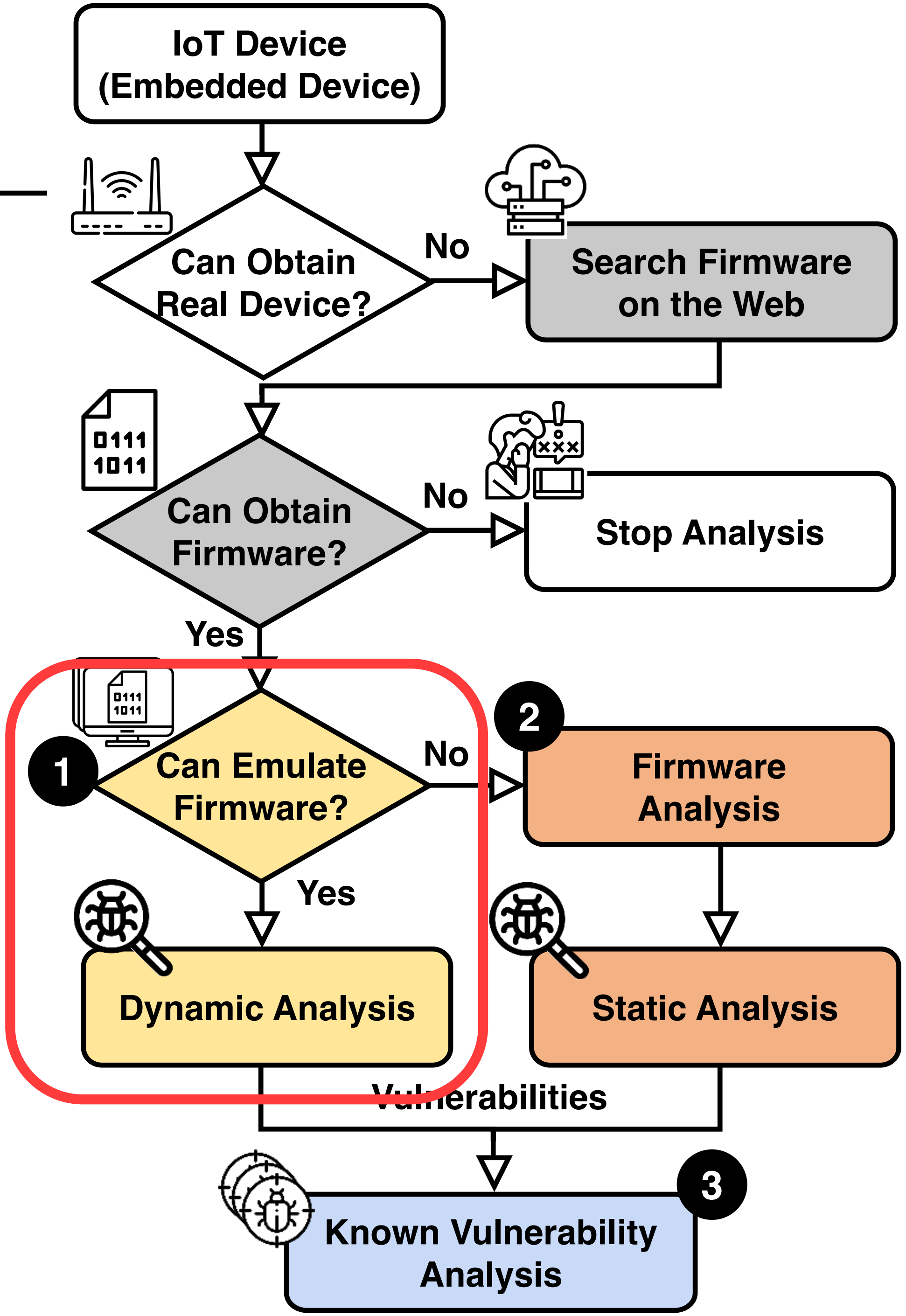
IoT Analysis Procedure

❖ Firmware collection
  – Physically obtaining numerous devices is infeasible
  – Download firmware images from vendors websites

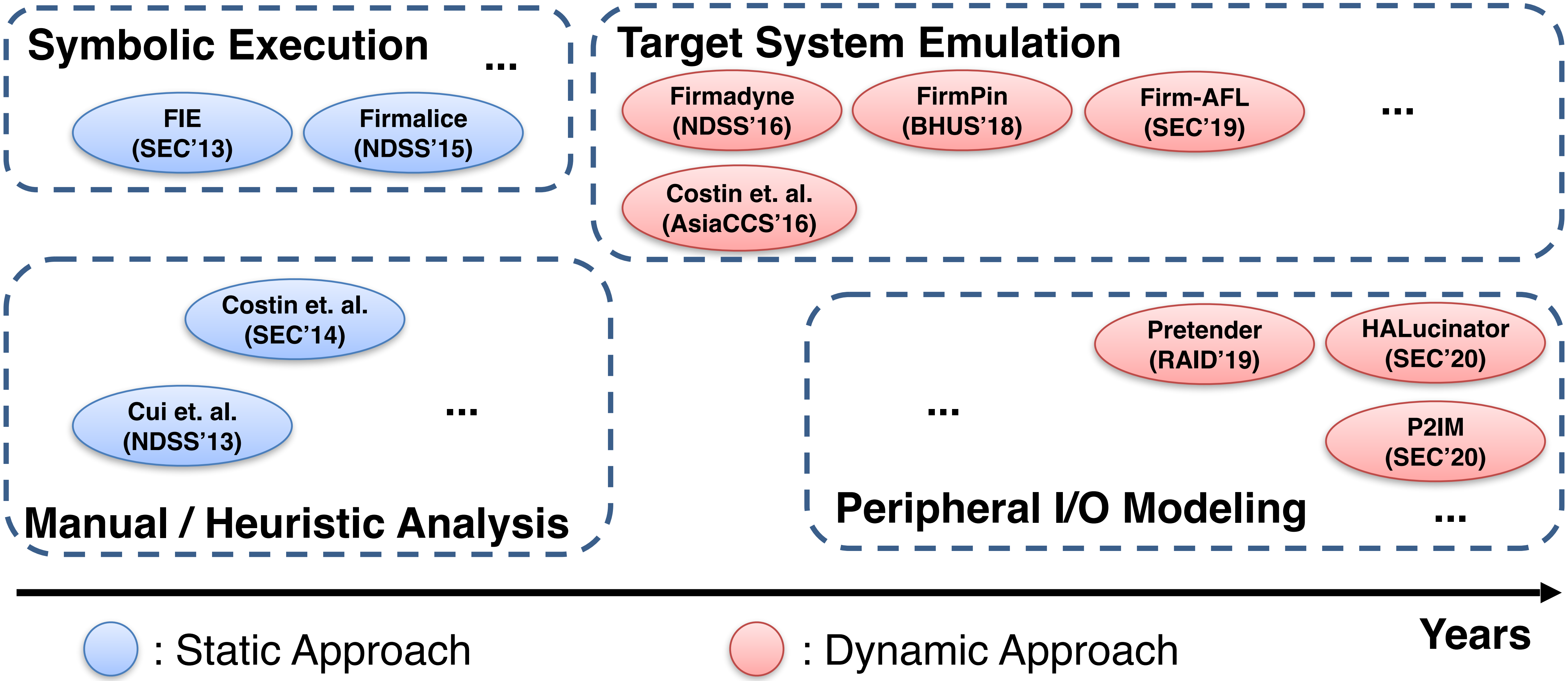
1 Firmware emulation and dynamic analysis
  – Build a virtual environment mimicking a real device
  – Run automated pentesting (e.g., Metasploit)
  – Run fuzzers (e.g., AFL)

2 Firmware and static analysis
  – Analyze firmware structure and memory layout
  – Identify target functions
  – Run symbolic execution (e.g., angr)

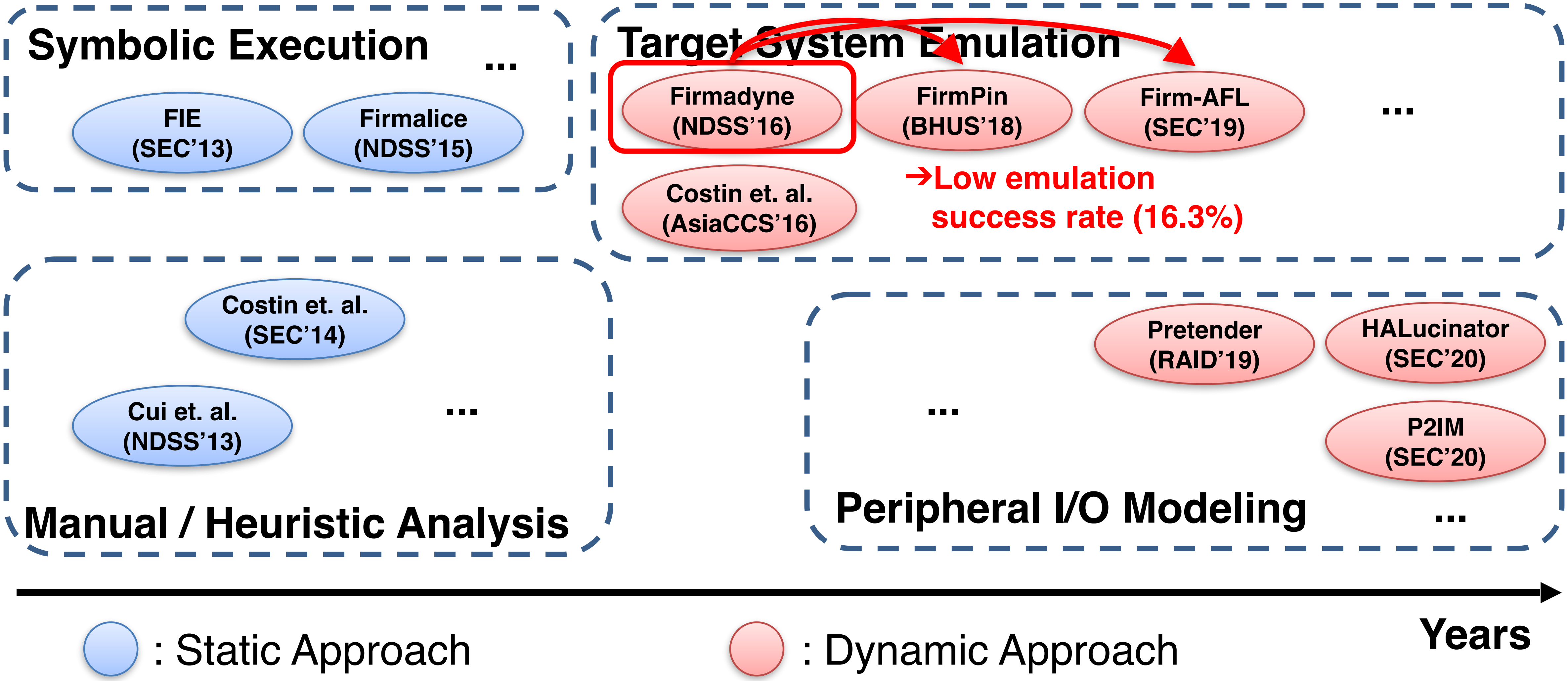
3 Known vulnerability (1-day-based) analysis
  – Build PoC exploits and run them (e.g., Metasploit)
  – Build signatures and search them (e.g., BCSA)



Existing Approaches

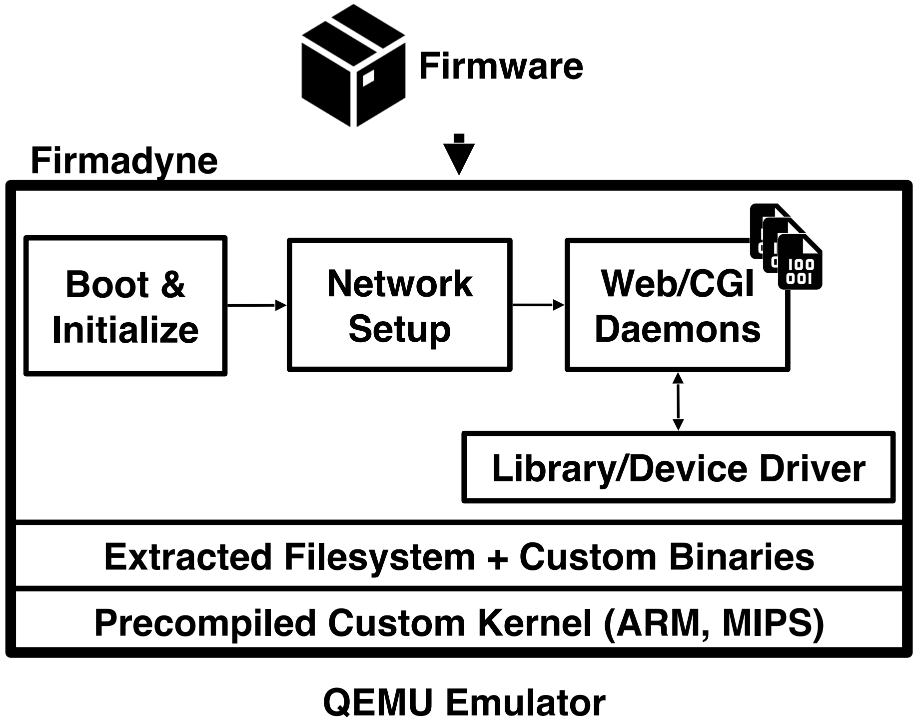


Existing Approaches



Firmadyne: state-of-the-art firmware emulator

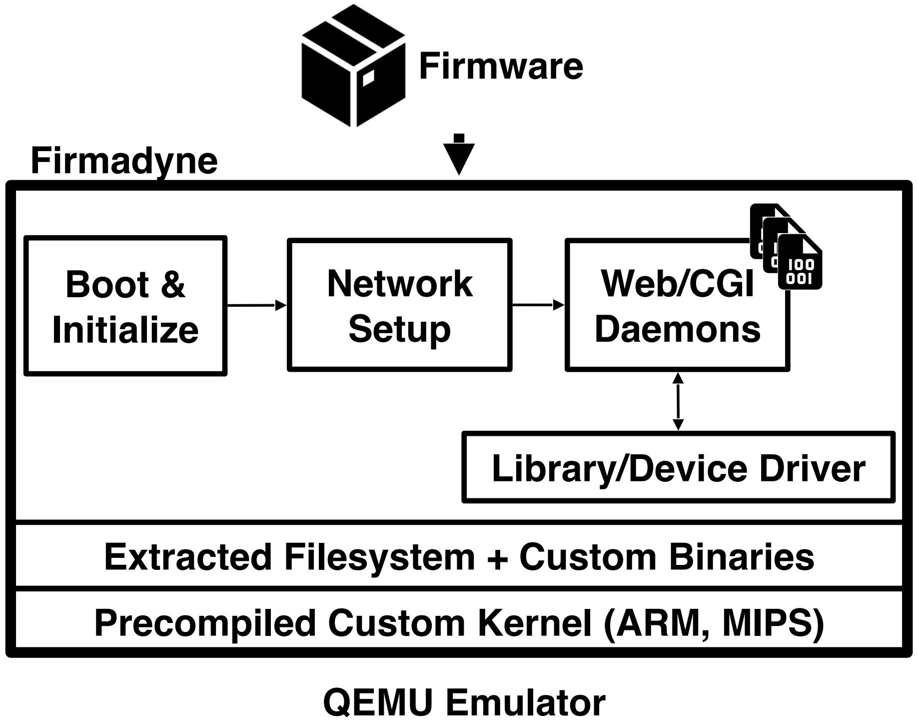
- Custom kernel and library
  - Hook system calls
  - Mimic NVRAM-related functions
    - *NVRAM: flash memory
- Emulating target firmware twice
  - Collect useful logs (IP address, device name)
  - Configure the system with the logs



Firmadyne: state-of-the-art firmware emulator

- Custom kernel and library
  - Hook system calls
  - Mimic NVRAM-related functions
    - *NVRAM: flash memory

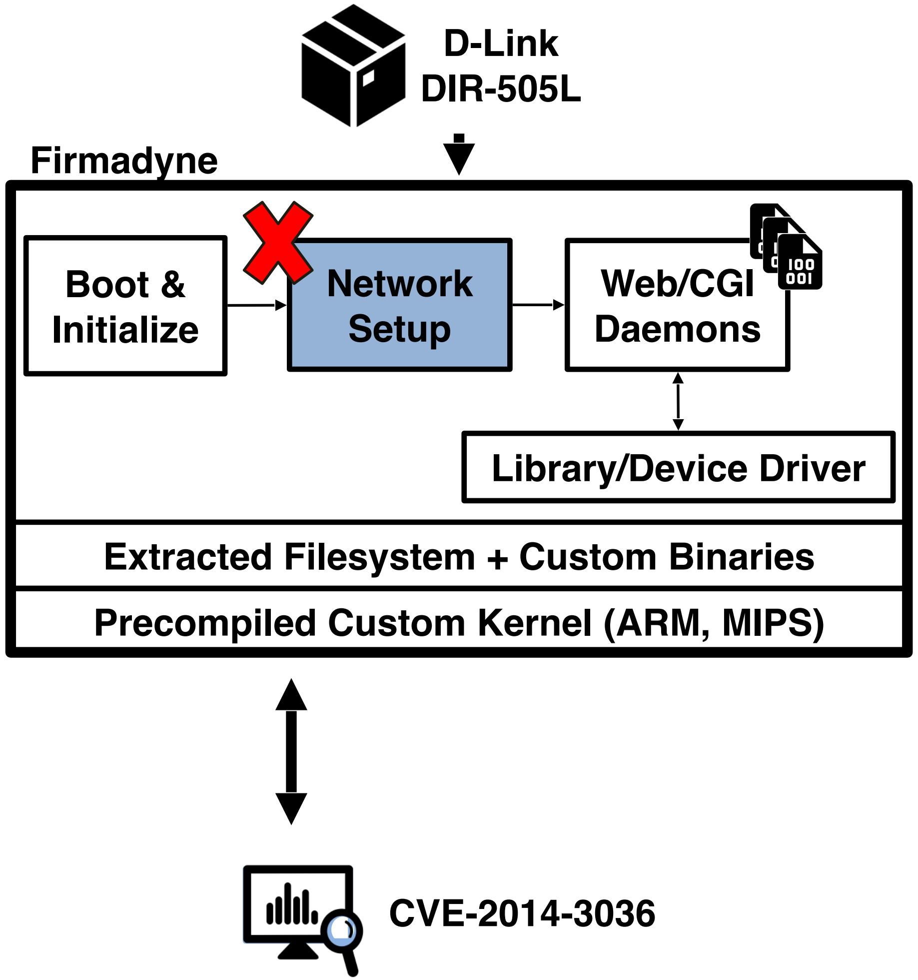
- Emulating target firmware twice
  - Collect useful logs (IP address, device name)
  - Configure the system with the logs



Firmadyne can emulate only 183 of 1,124 (16.3%) firmware images for web services

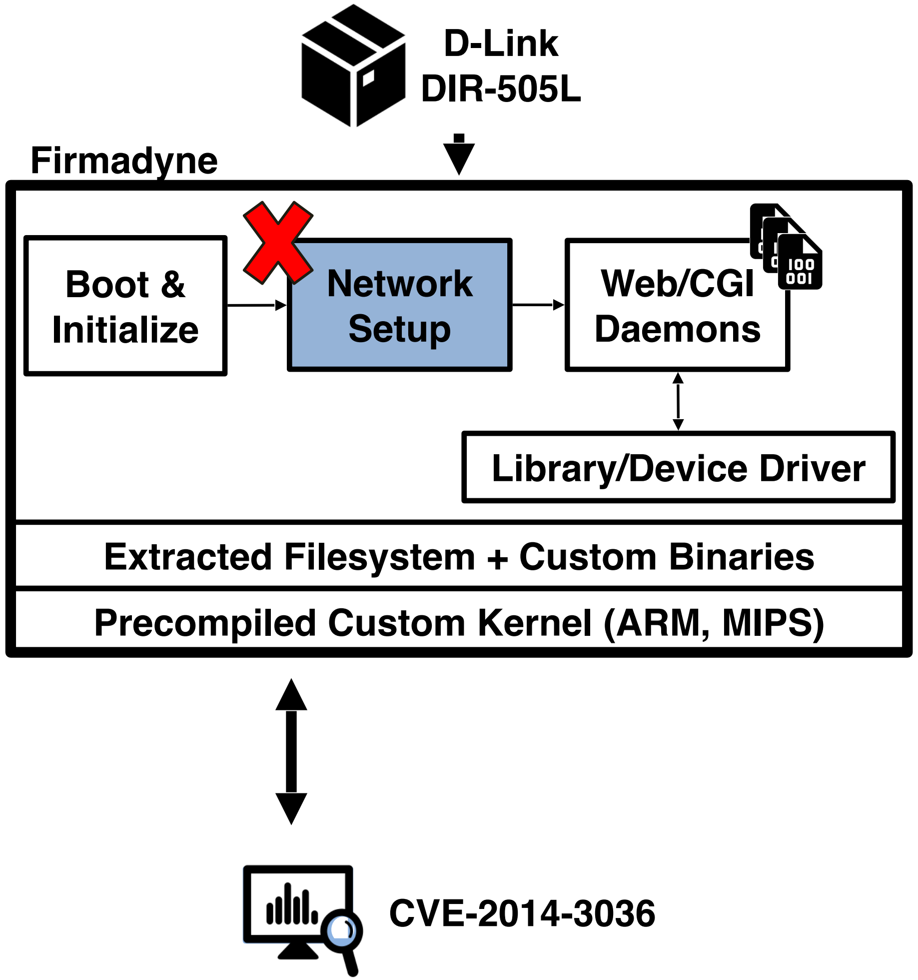
Motivating example: CVE-2014-3936

- ❖ Target
  - D-Link DIR-505L
- ❖ Symptom
  - Fails to configure network interface
- ❖ Possible causes
  - Access to unsupported peripherals
  - Retrieve unknown/improper values
- ❖ How to address
  - Forcibly set up the network interface



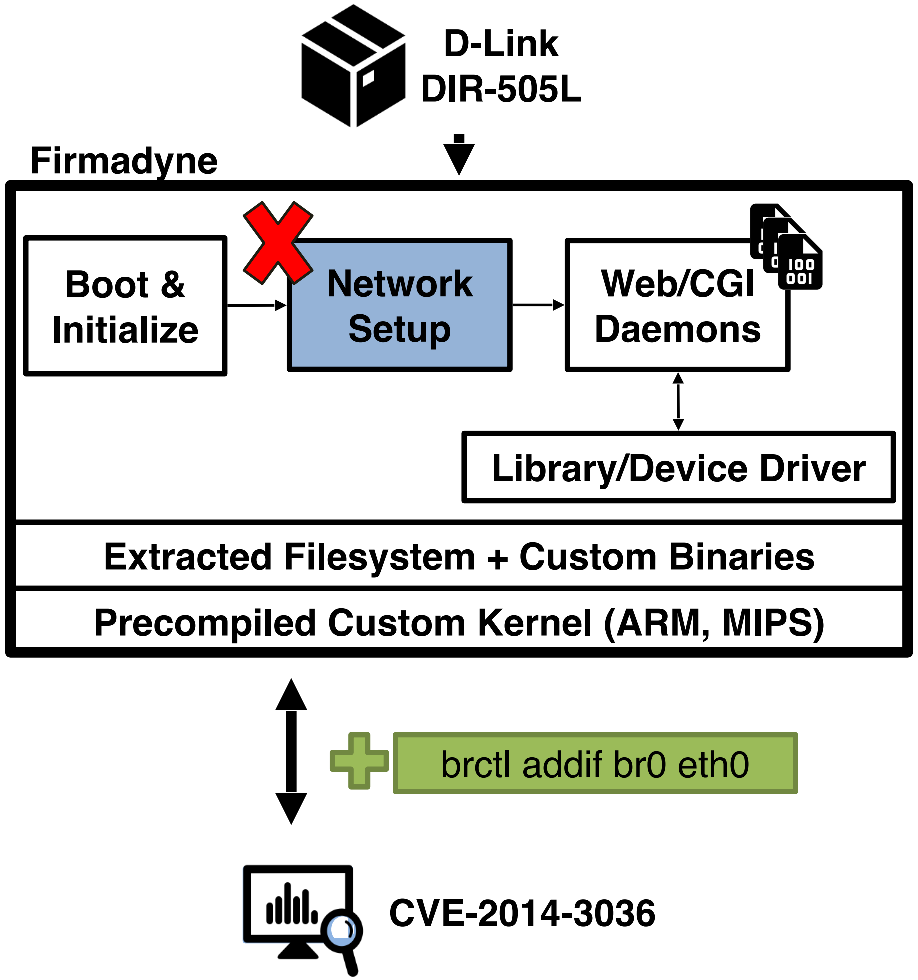
Motivating example: CVE-2014-3936

- Target
  - D-Link DIR-505L
- Symptom
  - Fails to configure network interface
- Possible causes
  - Access to unsupported peripherals
  - Retrieve unknown/improper values
- How to address
  - Forcibly set up the network interface



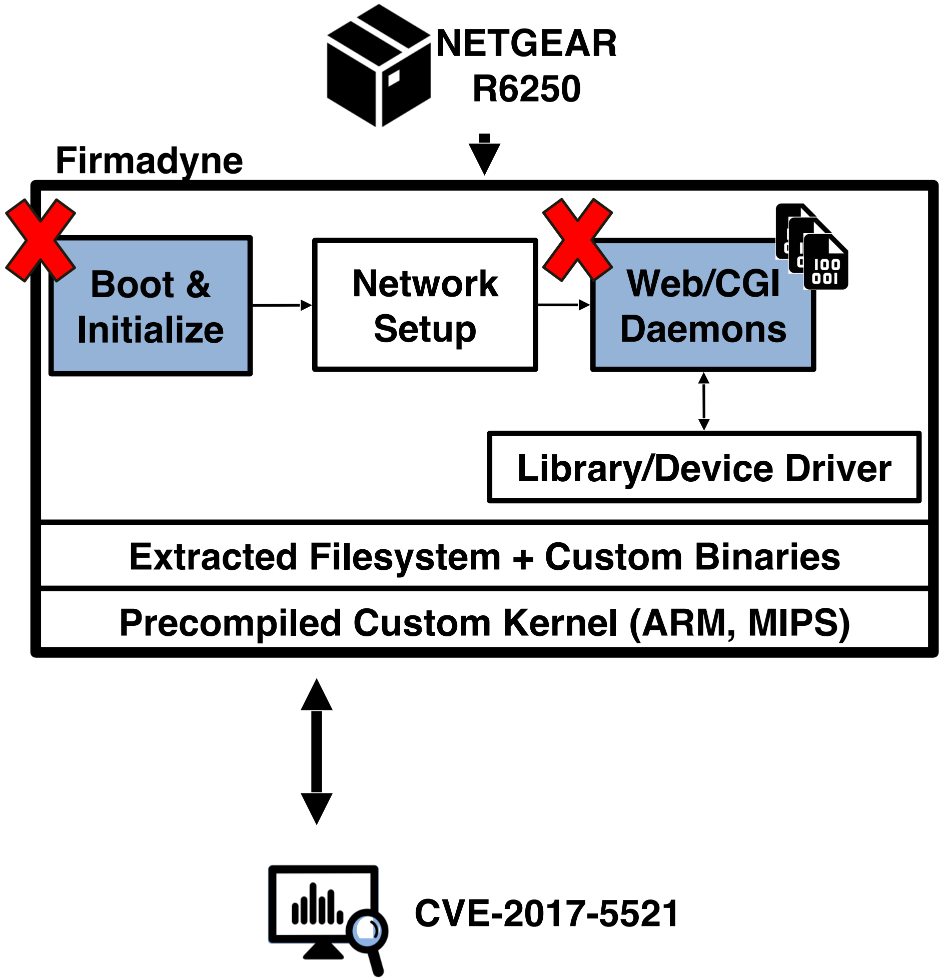
Motivating example: CVE-2014-3936

- Target
  - D-Link DIR-505L
- Symptom
  - Fails to configure network interface
- Possible causes
  - Access to unsupported peripherals
  - Retrieve unknown/improper values
- How to address
  - Forcibly set up the network interface



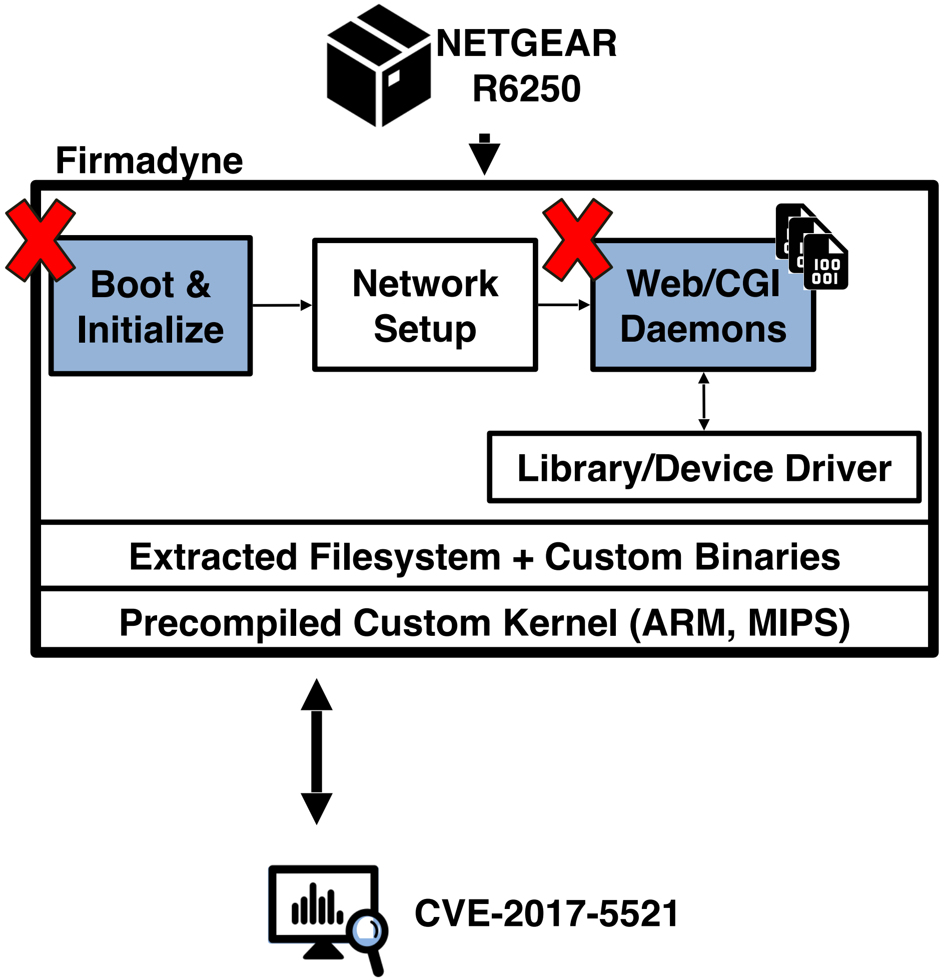
Motivating example: CVE-2017-5521

- Target
  - NETGEAR R6250
- Symptom
  - Fails to boot and run the web service
- Possible causes
  - Incorrect init program
  - Missing kernel module to handle IOCTL
- How to address
  - Set the correct init program path
  - Add an IOCTL wrapper



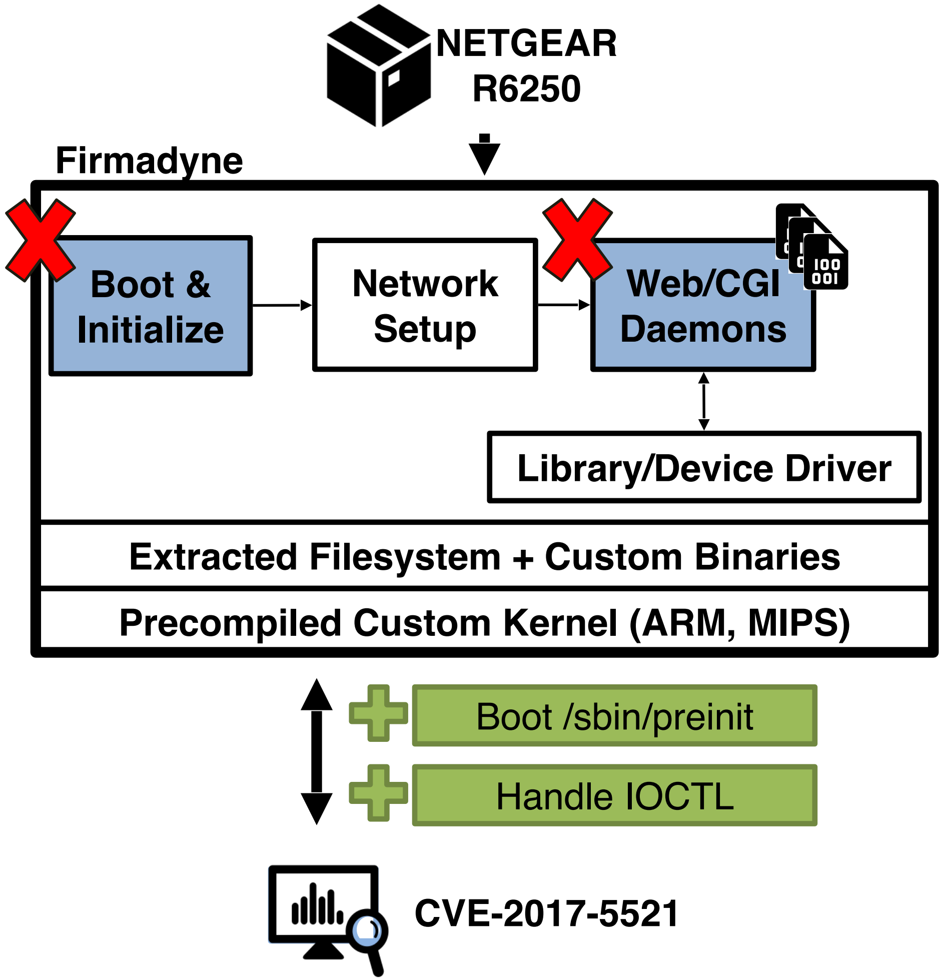
Motivating example: CVE-2017-5521

- Target
  - NETGEAR R6250
- Symptom
  - Fails to boot and run the web service
- Possible causes
  - Incorrect init program
  - Missing kernel module to handle IOCTL
- How to address
  - Set the correct init program path
  - Add an IOCTL wrapper



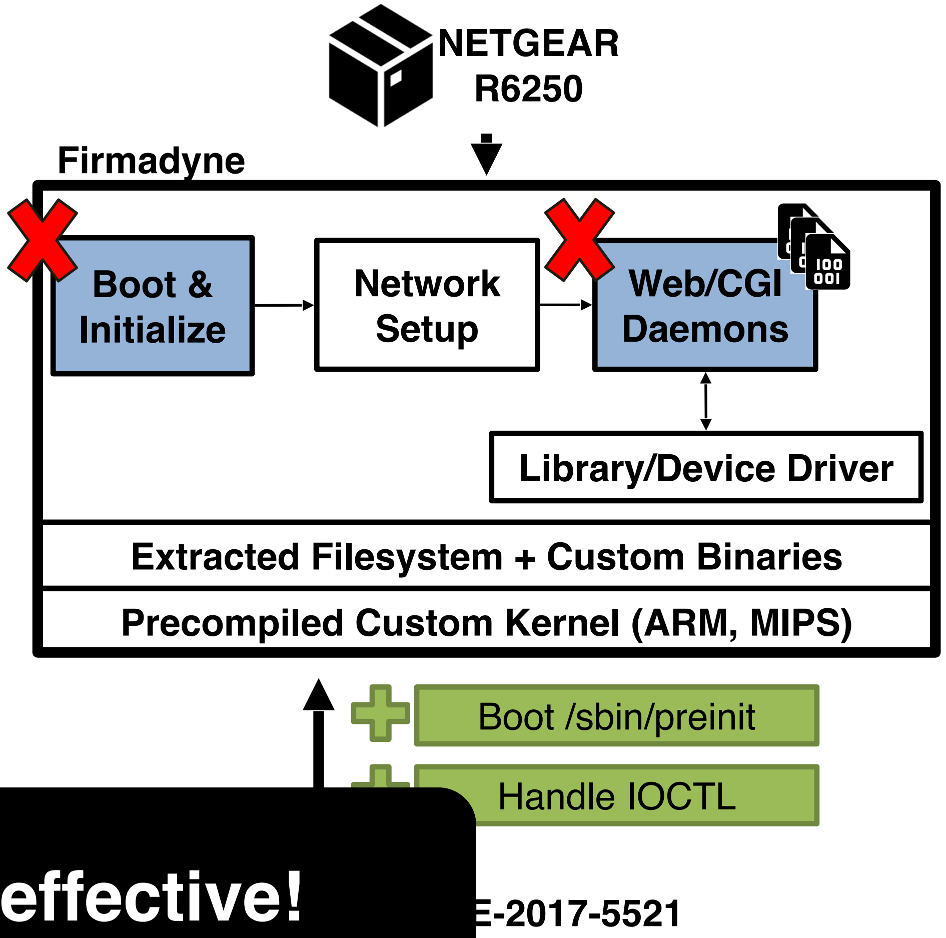
Motivating example: CVE-2017-5521

- Target
  - NETGEAR R6250
- Symptom
  - Fails to boot and run the web service
- Possible causes
  - Incorrect init program
  - Missing kernel module to handle IOCTL
- How to address
  - Set the correct init program path
  - Add an IOCTL wrapper



Motivating example: CVE-2017-5521

- Target
  - NETGEAR R6250
- Symptom
  - Fails to boot and run the web service
- Possible causes
  - Incorrect init program
  - Missing kernel module to handle IOCTL
- How to address
  - Set the correct i
  - Add an IOCTL w

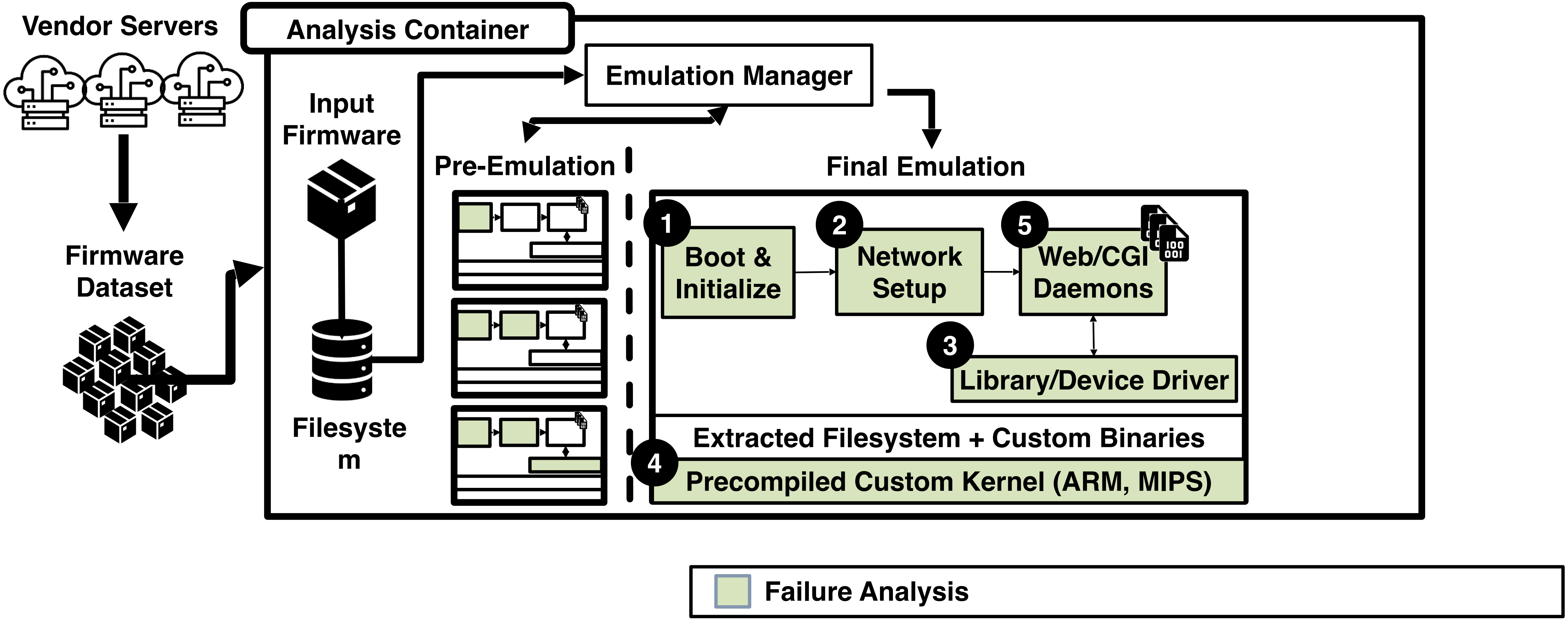


Simple heuristics are effective!

Our approach

- Goal
  - Run admin web services for dynamic security analysis
- Requirements
  - Emulated system should be reachable from the host
  - Web services should be available
- Approach
  - Investigate failure cases of Firmadyne
  - Develop heuristics to satisfy the emulation requirements

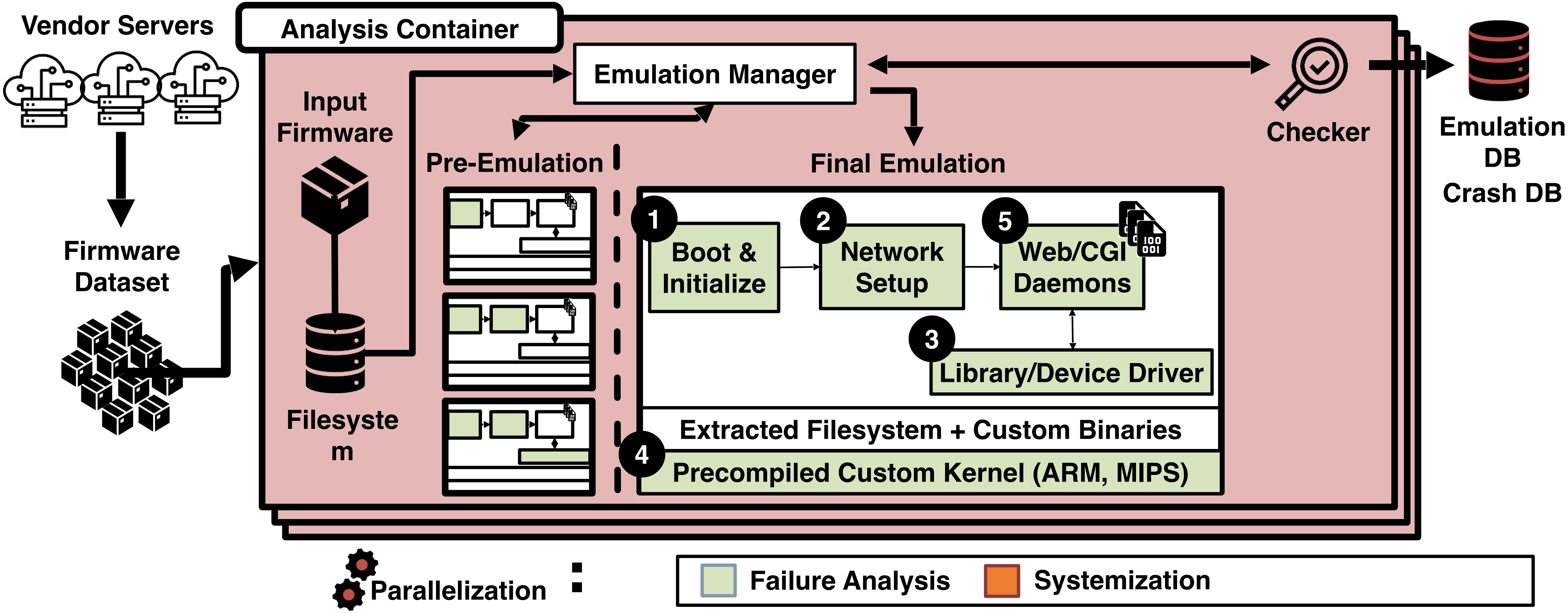
FirmAE overview



Examples of Developed Heuristics

Where	Problem	Heuristics
Boot	Missing files or directories	Extract path strings and create them (e.g., /var, /etc)
Library for Virtualization	Unknown configuration values	Search filesystem and original kernel (e.g., /etc/nvram.default)
Network	No network interface	Forcibly configure a default interface (e.g., eth0, 192.168.0.1)
Programs	Unexecuted web server	Forcibly run the server (e.g., run httpd)

FirmAE overview



Emulation Results (vs Firmadyne)

Wireless Routers

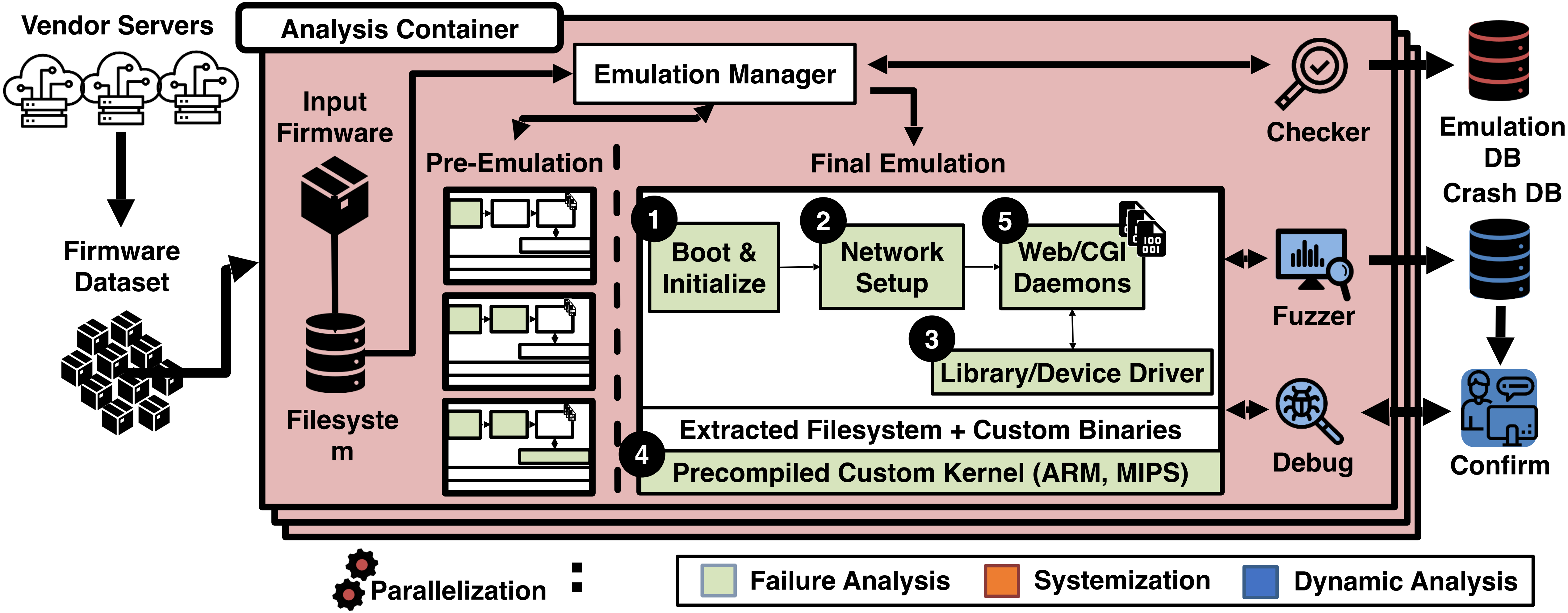
IP Cameras

Dataset	Vendor	Images	Firmadyne Web	FirmAE Web
AnalysisSet (**Outdated**)	D-Link	179	54 (30.17%)	167 (93.30%)
	NETGEAR	73	5 (6.85%)	59 (80.82%)
	TP-Link	274	30 (10.95%)	257 (93.80%)
Sub Total		**526**	**89 (16.92%)**	**483 (91.83%)**
LatestSet (**Latest**)	D-Link	58	17 (29.31%)	48 (82.76%)
	TP-Link	69	10 (14.49%)	54 (78.26%)
	NETGEAR	101	7 (6.93%)	79 (78.22%)
	TRENDnet	106	23 (21.70%)	63 (59.43%)
	ASUS	107	25 (23.36%)	62 (57.94%)
	Belkin	37	2 (5.41%)	22 (59.46%)
	Linksys	55	8 (14.55%)	44 (80.00%)
	Zyxel	20	0 (0%)	10 (50.00%)
Sub Total		**553**	**92 (16.64%)**	**382 (69.08%)**
CamSet (**Latest**)	D-Link	26	0 (0%)	17 (65.38%)
	TP-Link	6	0 (0%)	0 (0%)
	TRENDnet	13	2 (15.38%)	10 (76.92%)
Sub Total		**45**	**2 (4.44%)**	**27 (60.00%)**
Total		**1124**	**183 (16.28%)**	**892 (79.36%)**

x5

*Latest firmware images are checked as of Dec. 2018

FirmAE overview



Dynamic Analysis Results

- Dynamic security analysis
  - Known vulnerabilities
    - RouterSploit (set of known exploits)
    - **14 (Firmadyne) ➔ 320 (FirmAE)**

Description	Total Vulns (Devices)
Information Leak	**8 (157)**
Command Injection	**23 (112)**
Authentication Bypass	**2 (5)**
Buffer Overflow	**5 (7)**

  - New vulnerabilities
    - RouterSploit + Simple custom fuzzer
    - **23 vulns from 95 latest devices (affecting 6 vendors)**

Motivating Observation

- Example vulnerability: CVE-2018-10106
  - Permission bypass in “cgibin” reveals users’ private key
  - Parameter can be over-written with a newline character (0x0a)

```
response = self.http_request(
    method="POST",
    path="/getcfg.php?A=A%0a_POST_SERVICES%3dDEVICE.ACCOUNT%0aAUTHORIZED_GROUP%3d1",
    headers=headers
)
```

  - Still appears in newer device versions (D-Link)
    - CVE-2018-10106, CVE-2019-17506, CVE-2019-20213, CVE-2020-9376
  - Appears in different venders (TRENDnet)
    - CVE-2018-7034

- Potential reasons
  - Improper version/update management
  - Copy and paste buggy code

Known Vulnerability Analysis

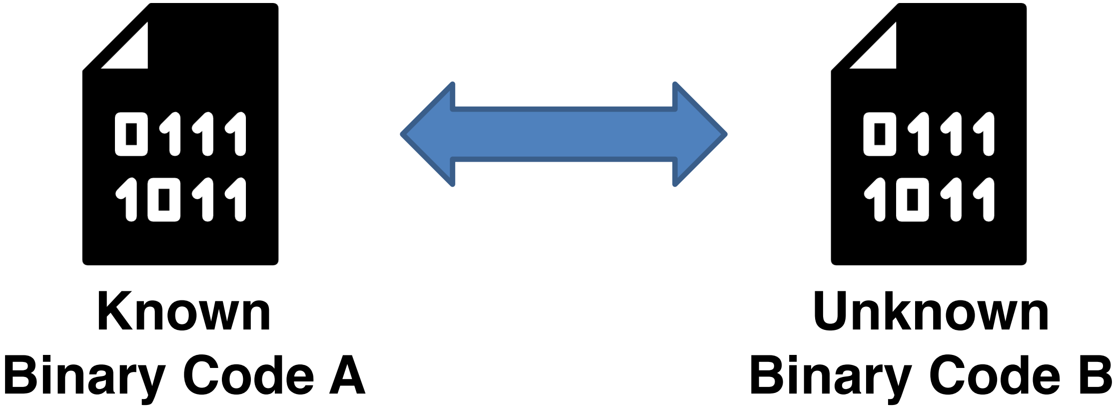
- ❖ Dynamic analysis
  - – Build PoC exploits and run them
  - ☞ Require successful emulation
    - ☞ Architecture challenges (e.g., ARM, MIPS, PowerPC, Hexagon, ...)
    - ☞ Dependency issues in peripherals (e.g., Camera, LED, MMIO access, ...)
  - ☞ Require time for emulation and testing

- ❖ Static analysis
  - – Match known signatures
  - – Leverage Binary code similarity analysis (BCSA)
  - ➔ **Apply BCSA to find same/similar vulnerabilities in newer devices**

Increasing Scalability
Preserving Low False Positive Rate

Binary Code Similarity Analysis

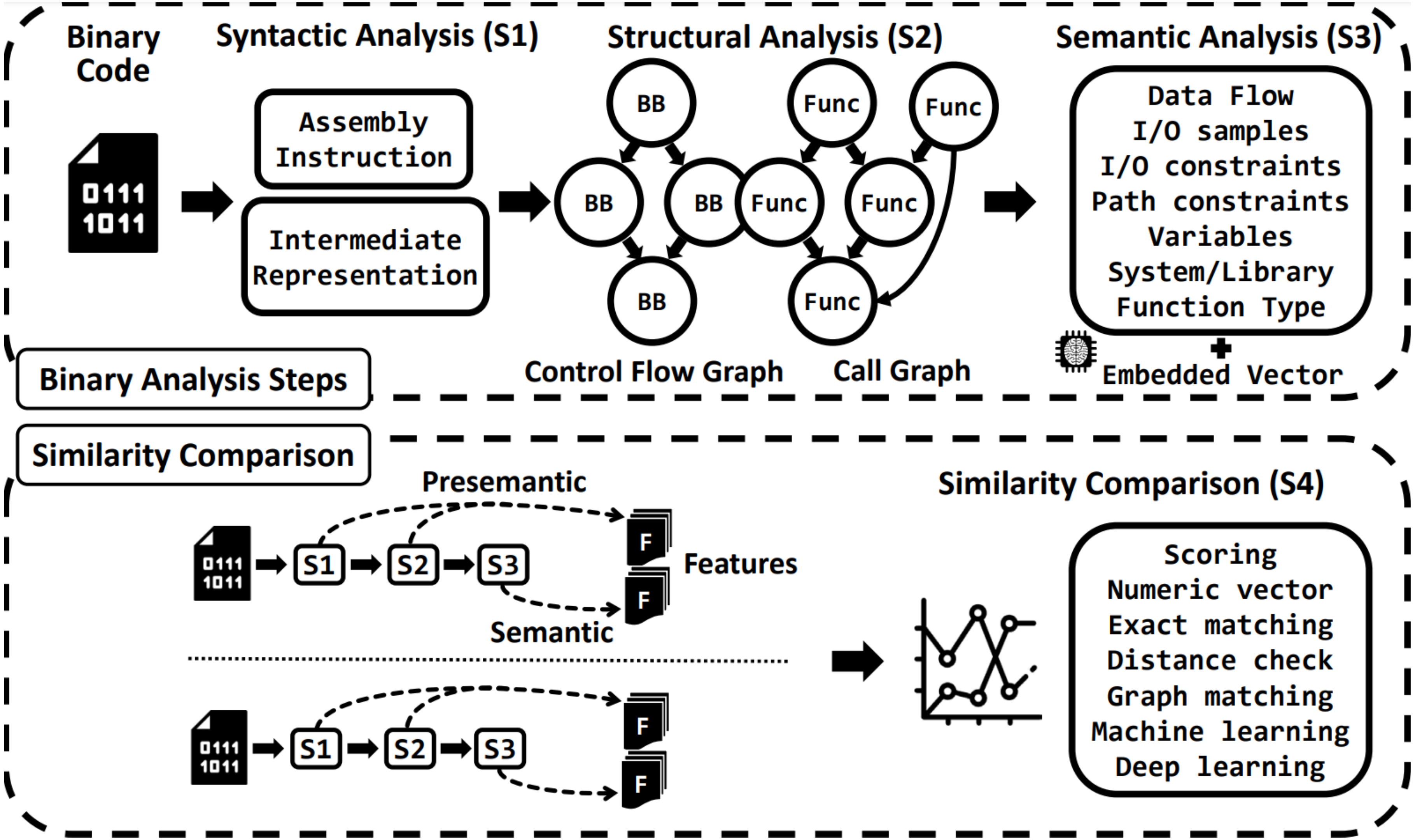
❖ Binary code similarity analysis (BCSA)



❖ Popular tasks
- Malware detection
- Plagiarism detection
- Authorship identification
- **Vulnerability discovery**

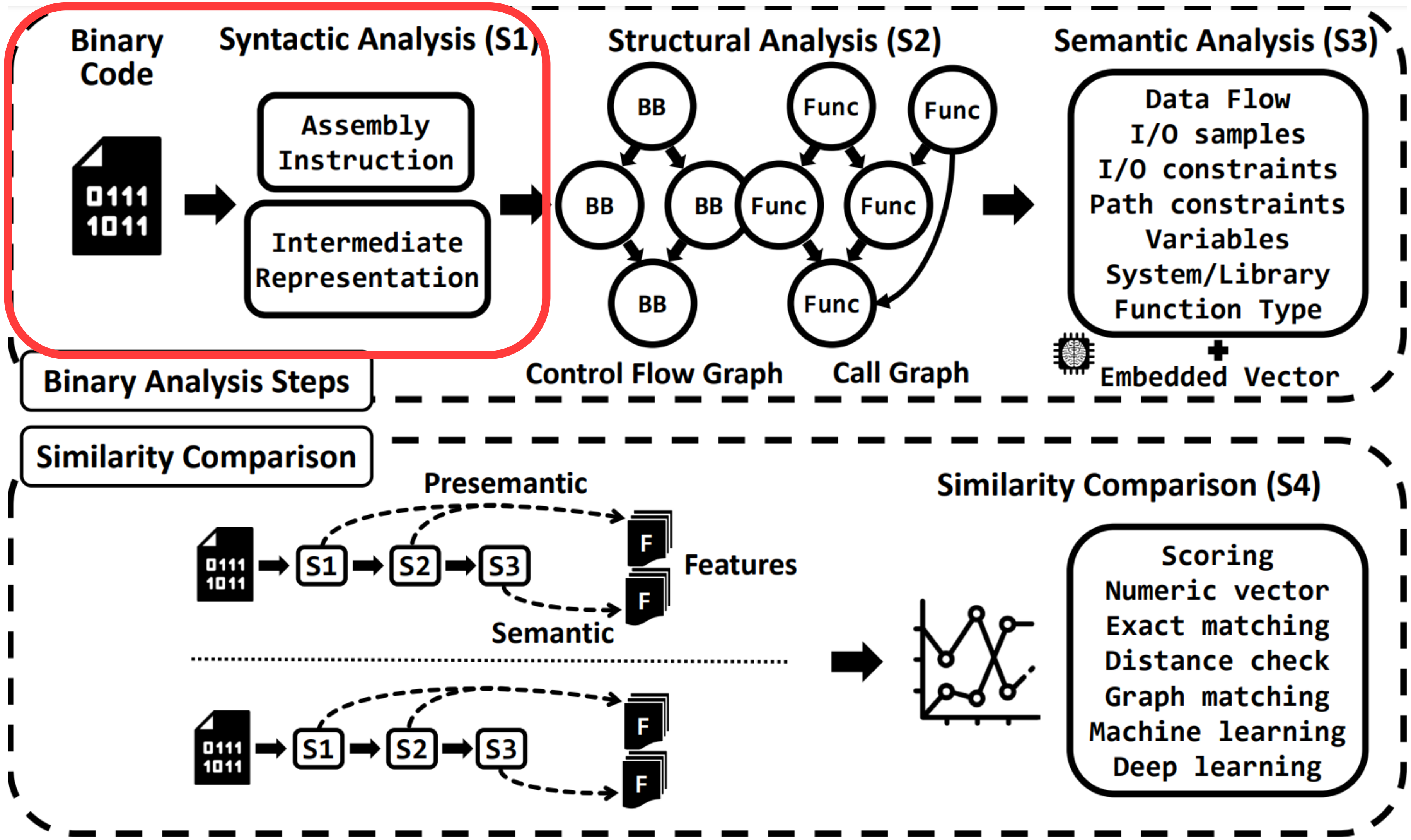
❖ Target
- Architecture (e.g., x86 -> ARM)
- Compiler (e.g., gcc -> clang)
- Optimization (e.g., O1 -> O3)
- Obfuscation (e.g., LLVM-Obfuscator)

BCSA Workflow



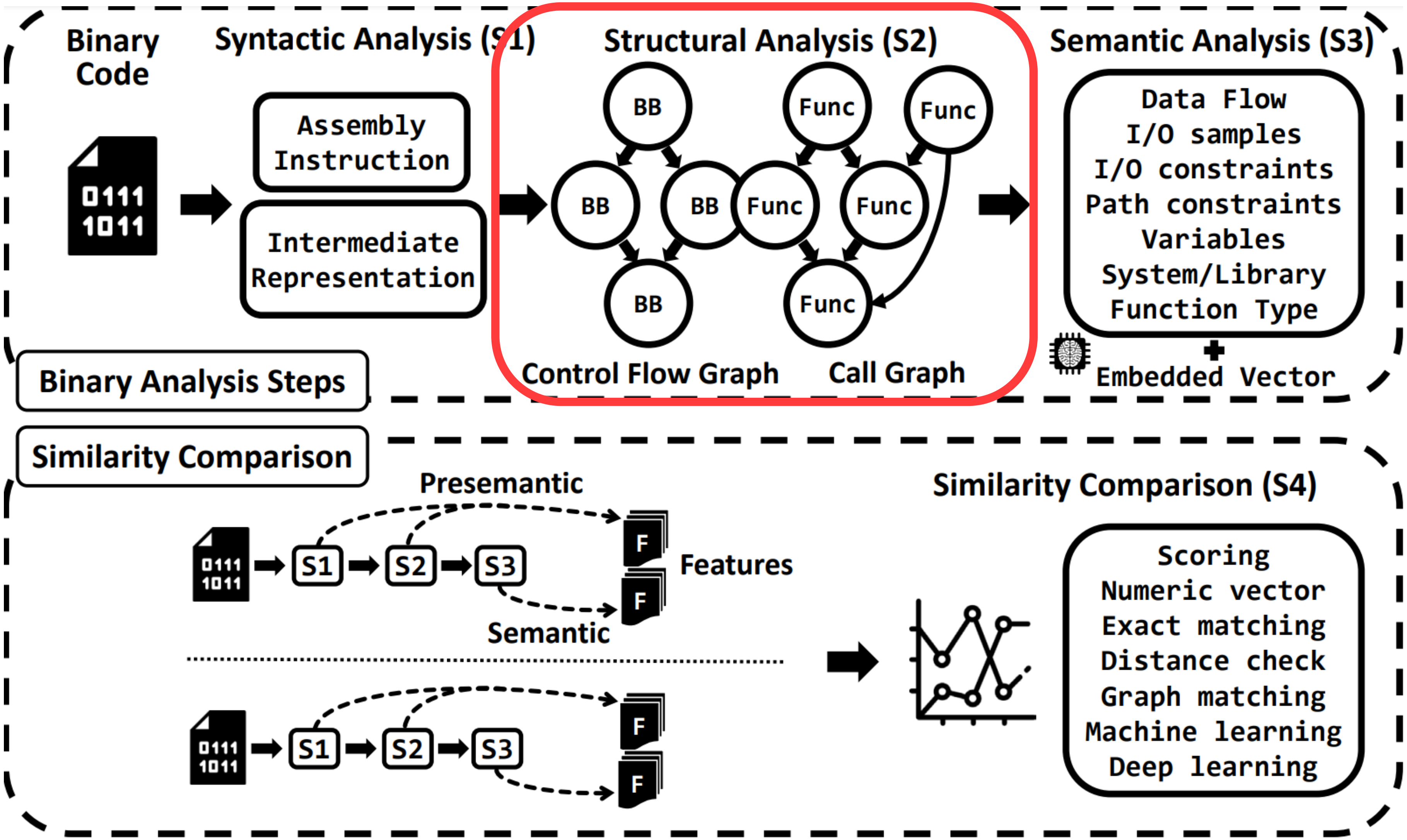
Similarity Score: 0.0 ~ 1.0

BCSA Workflow



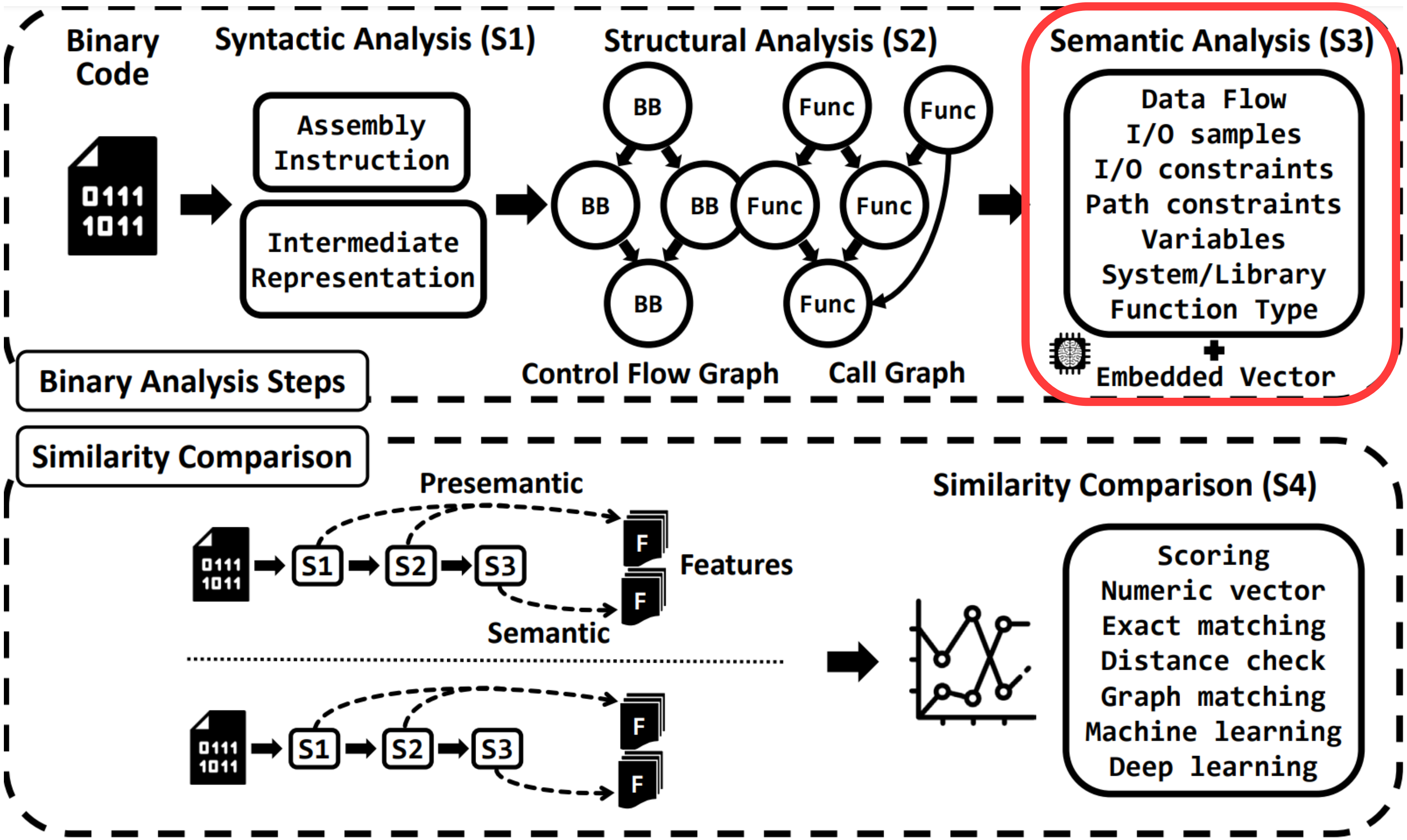
Similarity Score: 0.0 ~ 1.0

BCSA Workflow



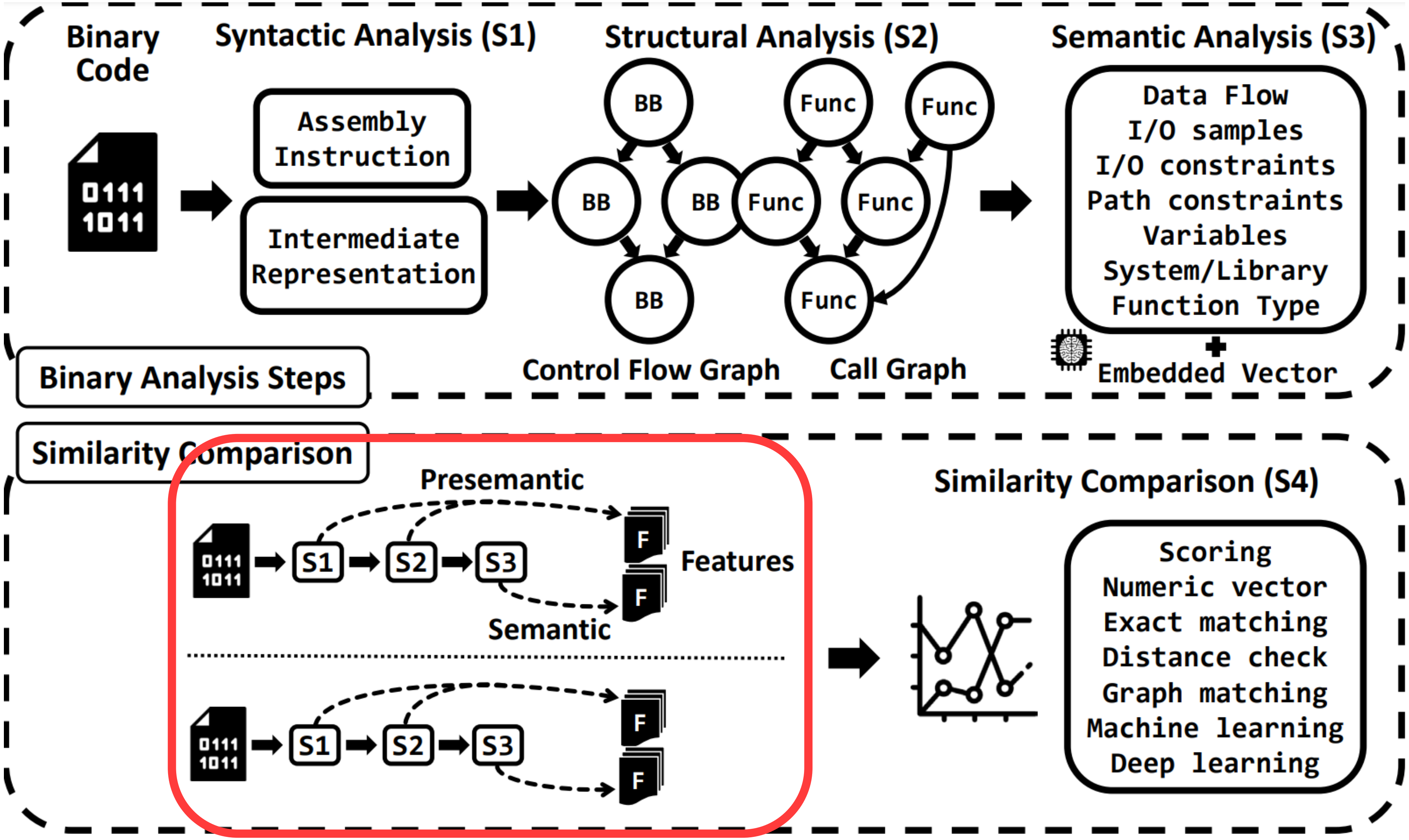
Similarity Score: 0.0 ~ 1.0

BCSA Workflow



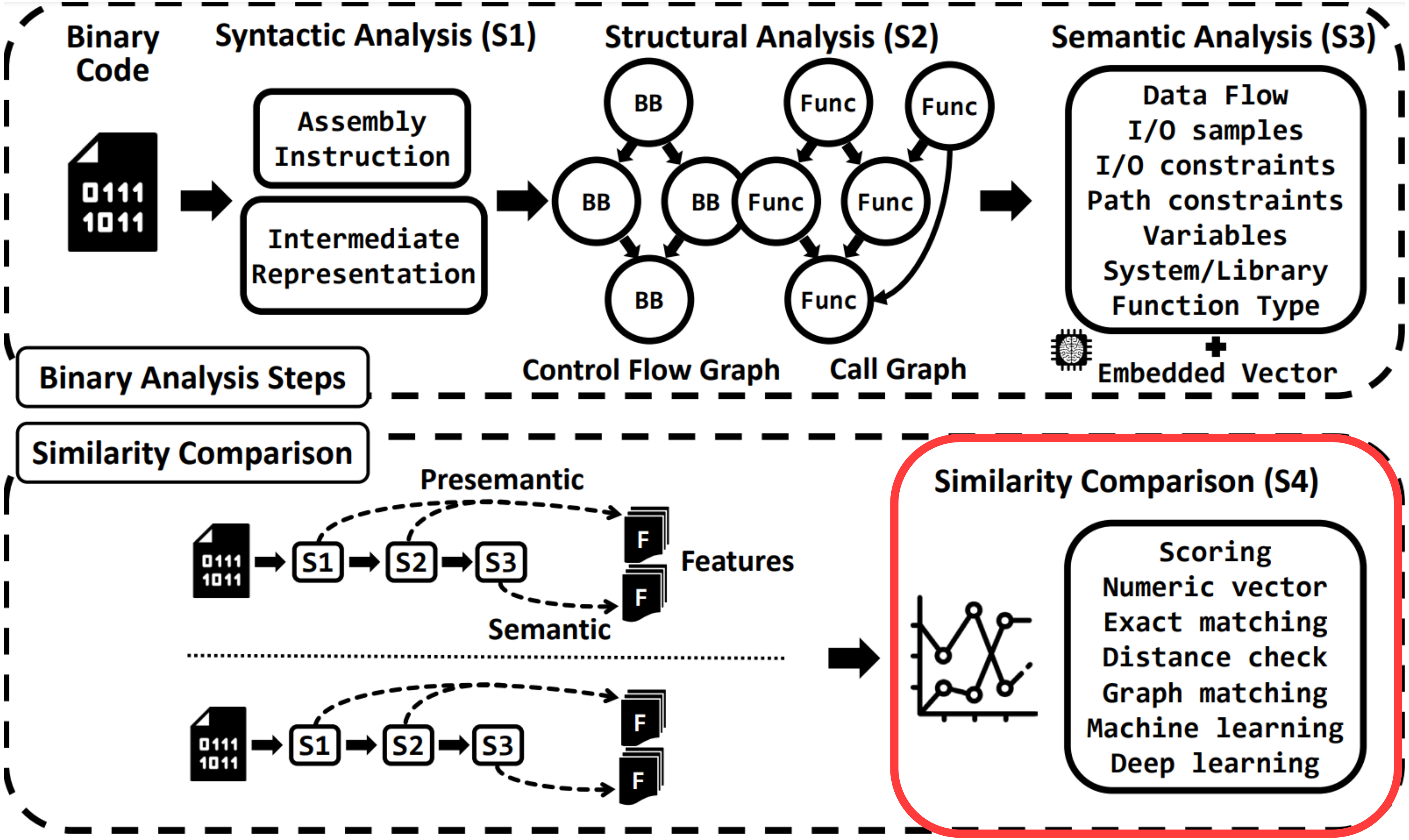
Similarity Score: 0.0 ~ 1.0

BCSA Workflow



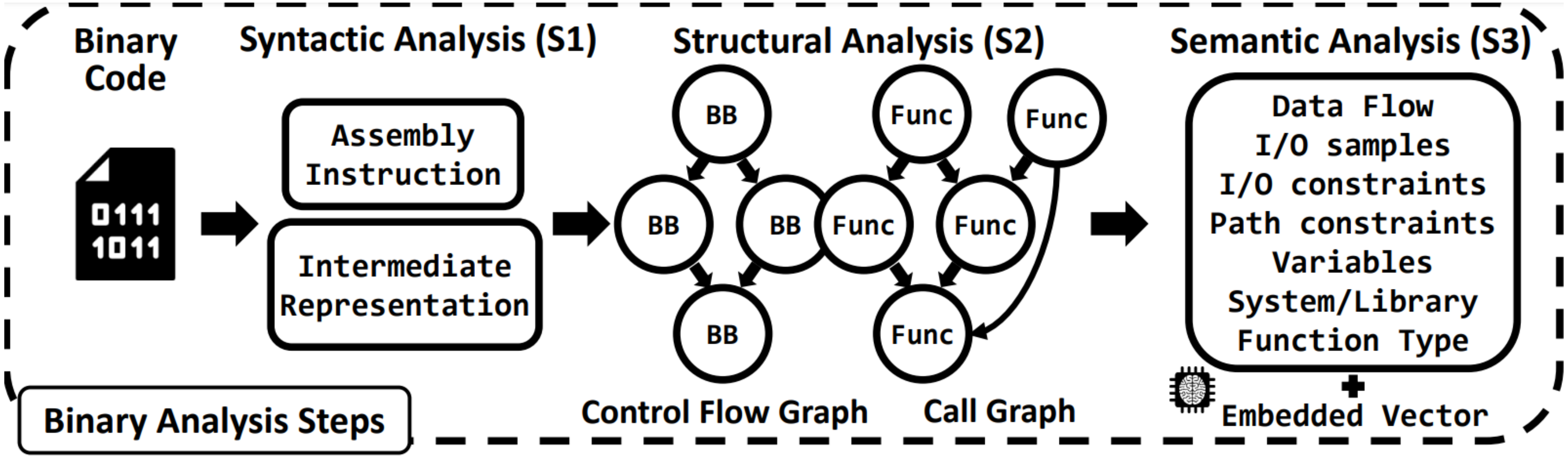
Similarity Score: 0.0 ~ 1.0

BCSA Workflow



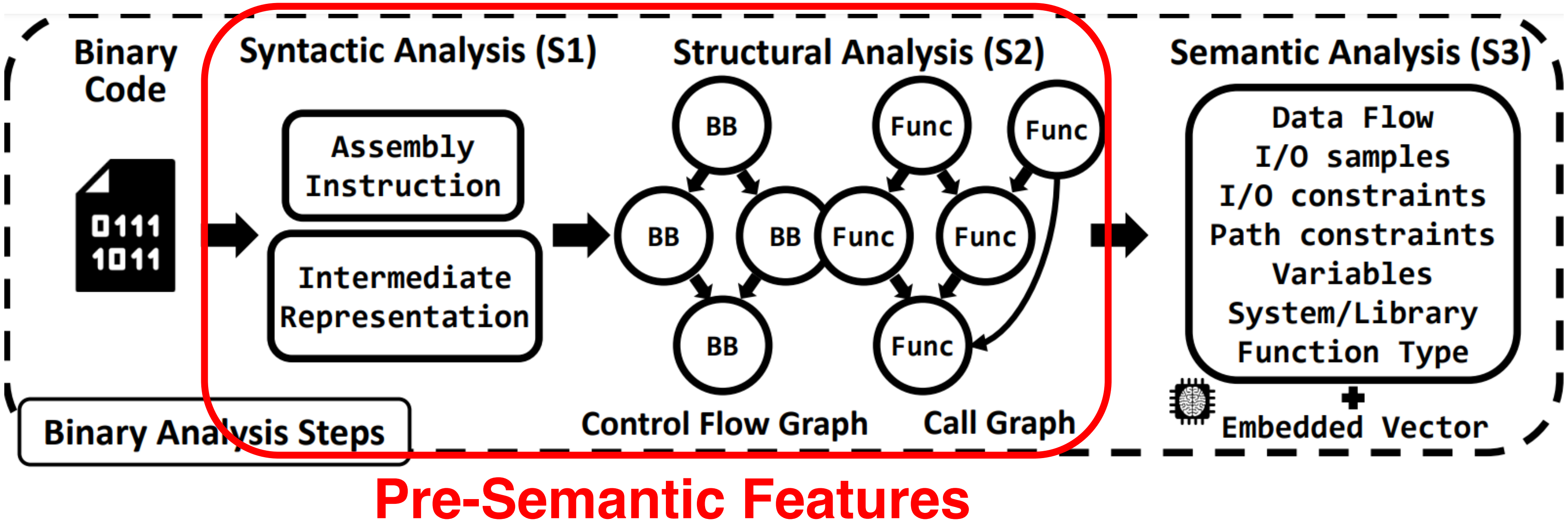
Similarity Score: 0.0 ~ 1.0

BCSA Workflow



*Revisiting Binary Code Similarity Analysis using Interpretable Feature Engineering and Lessons Learned

BCSA Workflow

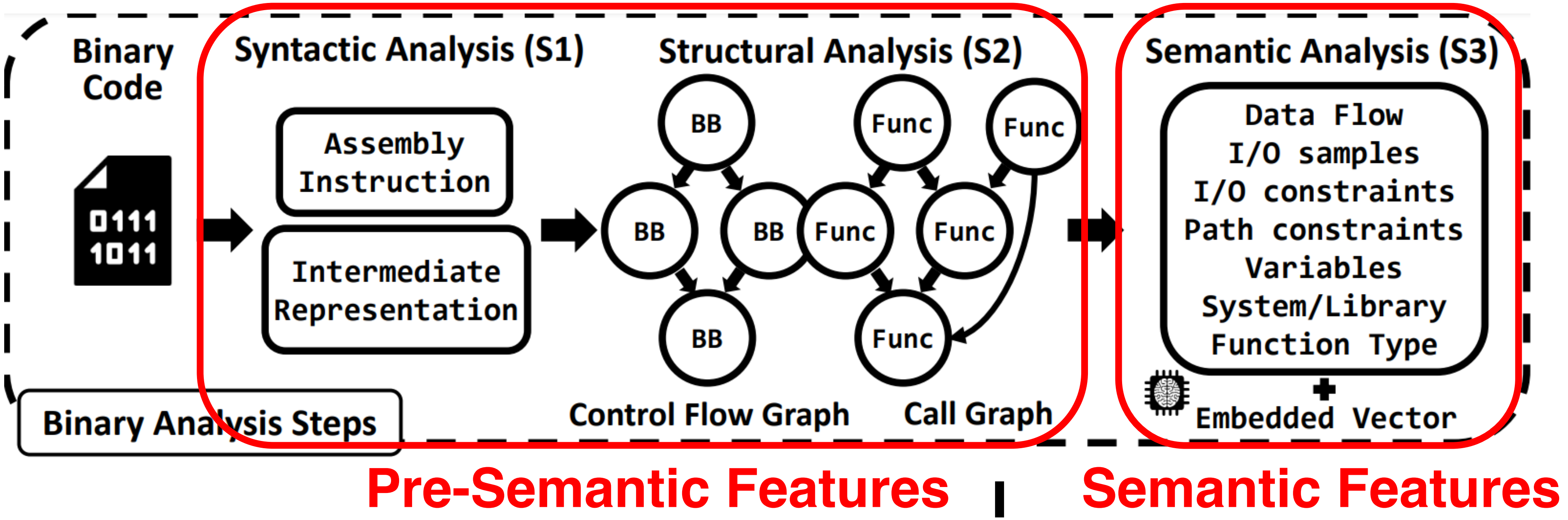


❖ **Numeric features**
- BB-level: # of instructions, …
- CFG-level: # of basic blocks, …
- CG-level: # of callers, …

❖ **Non-Numeric features**
- Raw bytes: N-gram, …
- Instructions: Assembly, IR, …
- Functions: Name, …

*Revisiting Binary Code Similarity Analysis using Interpretable Feature Engineering and Lessons Learned

BCSA Workflow



❖ **Numeric features**
– BB-level: # of instructions, …
– CFG-level: # of basic blocks, …
– CG-level: # of callers, …

❖ **Non-Numeric features**
– Raw bytes: N-gram, …
– Instructions: Assembly, IR, …
– Functions: Name, …

❖ **Semantic features**
– Symbolic constraints
– Runtime behavior (memory values, …)
– Program slices (data flow, …)
– Embedded vector (machine learning)
– …

*Revisiting Binary Code Similarity Analysis using Interpretable Feature Engineering and Lessons Learned

2014	2015	2016	2017	2018	2019	2020
TEDEM	Multi-k-MH	DiscovRE	GitZ	FirmUp	InnerEye	DeepBinDiff
Tracy		Esh	Xmatch	SANER18	Asm2Vec	ImOpt
CoP		Genius	Gemini	BinGo-E	SAFE	ACCESS20
LoPD		BinGo	BinSim	WSB	BAR19i	Patcheko
BLEX		MockingBird	IMF-SIM	BinMatch	BAR19ii	
BinClone		Kam1n0	CACompare	MASES18	FuncNet	
		BinDNN	BinSign	Zeek		
			BinSequence	BinArm		
			ASE17	*a*Diff		
				VulSeeker		

Studied 43 papers in 27 venues

GitZ (PLDI'17)

- Remove/Rearrange IR instructions
- Rename variables/instructions

(i) Assembly → **(ii) Lifted** → **(iii) Canonical** → **(iv) Canonical & Normalized**

(A) ARM-64 GCC 4.8 -O0

(i) Assembly
```
mov     x0,  x20
add     x0,  x0, 1
sub     x21, x21, x0
cmn     x21, 2
```

(ii) Lifted
```
t3  = load i64, i64* x20
t4  = sext i8 0 to i64
t5  = shl  i64  t3, t4
t6  = or   i64  0,  t5
store i64  t6,  i64* x0
t17 = load i64, i64* x0
t18 = add  i64  t17, 1
store i64  t18, i64* x0
t38 = load i64, i64* x21
...
```

(iii) Canonical
```
t3  = load i64, i64* x20
t18 = add  i64  t3,  1
store i64  t18, i64* x0
t38 = load i64, i64* x21
t42 = sub  i64  t38, t18
store i64  t42, i64* x21
t57 = add  i64  t42, 2
ret   i64  t57
```

(B) X86-64 icc 15.0.3 -O3

(i) Assembly
```
lea     r15, [rax+1]
sub     r13, r15
cmp     r13, -2
```

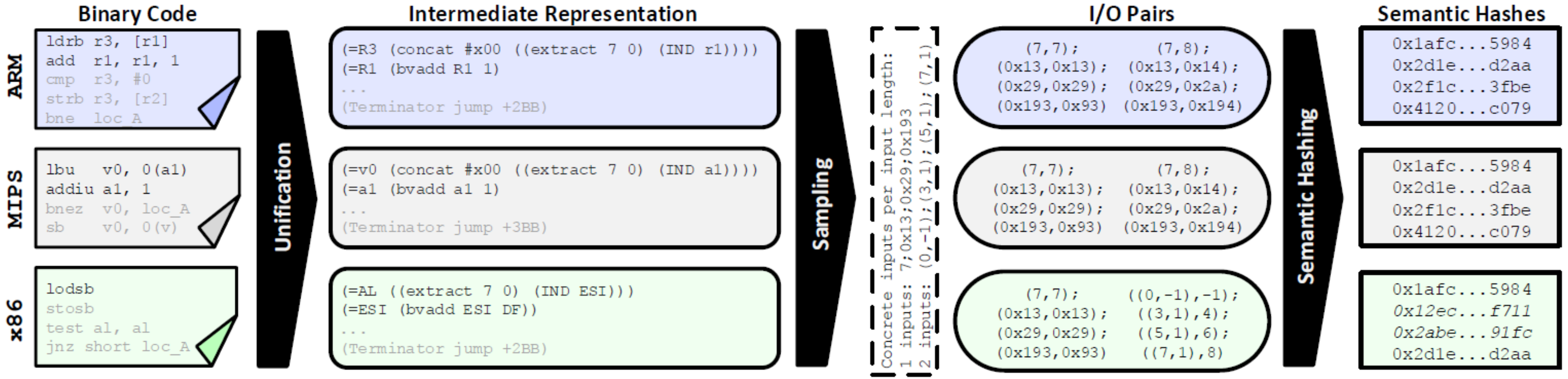
(ii) Lifted
```
t18 = load i64, i64* rax
t19 = add  i64  t18, 1
store i64  t19, i64* r15
t23 = load i64, i64* r13
t24 = load i64, i64* r15
t25 = sub  i64  t23, t24
store i64  t25, i64* r13
t37 = load i64, i64* r13
t38 = sub  i64  t37, -2
...
```

(iii) Canonical
```
t18 = load i64, i64* rax
t19 = add  i64  t18, 1
store i64  t19, i64* r15
t23 = load i64, i64* r13
t25 = sub  i64  t23, t19
store i64  t25, i64* r13
t38 = add  i64  t25, 2
ret   i64  t38
```

(iv) Canonical & Normalized
```
t0  = load i64, i64* r0
t1 = add  i64  t0,  1
store i64  t1, i64* r1
t2 = load i64, i64* r2
t3 = sub  i64  t2, t1
store i64  t3, i64* r2
t4 = add  i64  t3, 2
ret   i64  t4
```

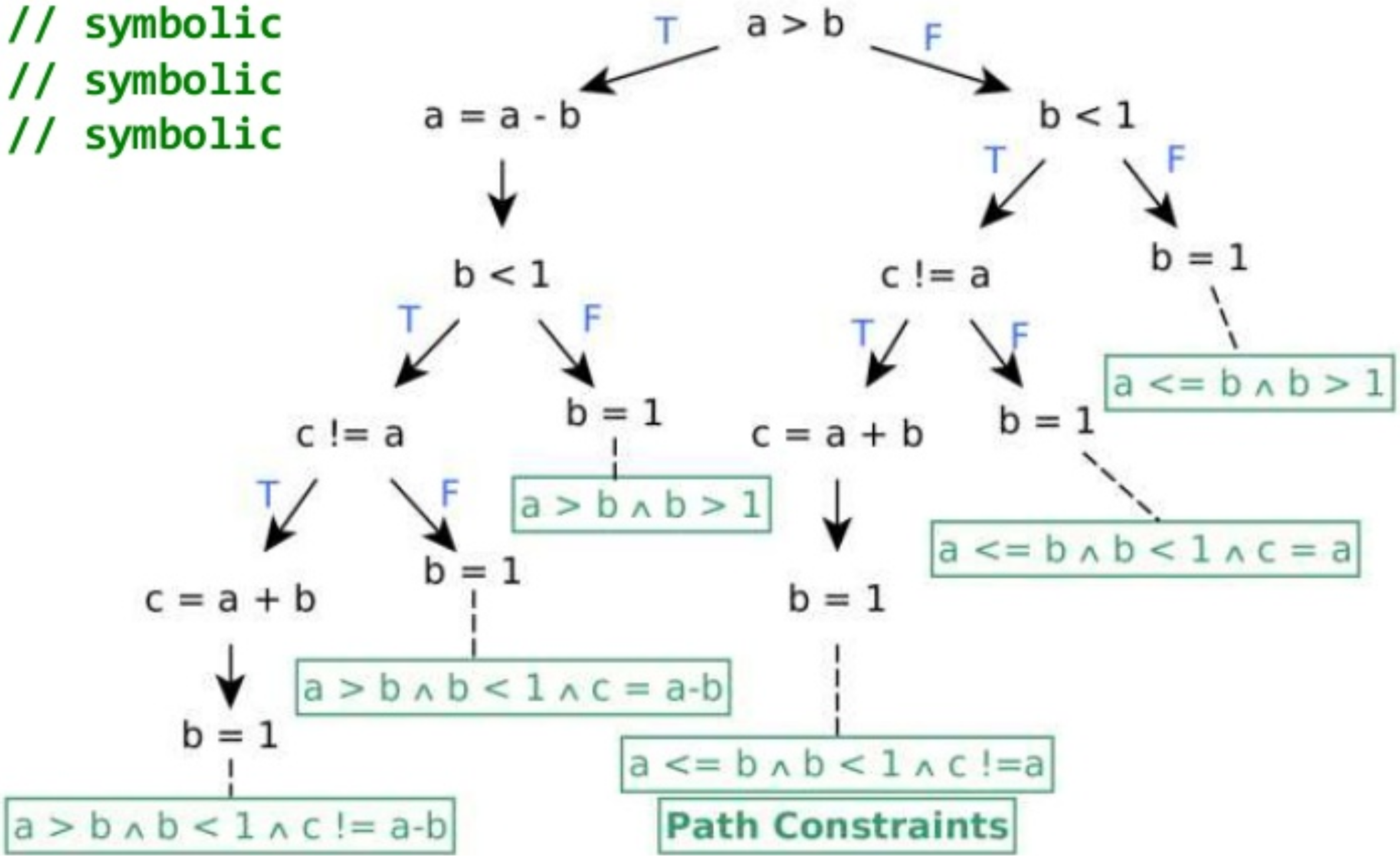
k-MinHash (SP'15)

- Disassemble with IDA Pro, translate to IR with pyvex
- For each bb, generate random inputs with Z3 and collect outputs
- Check k-multi MinHash for I/O pairs
- Propagate basic block matching to whole function



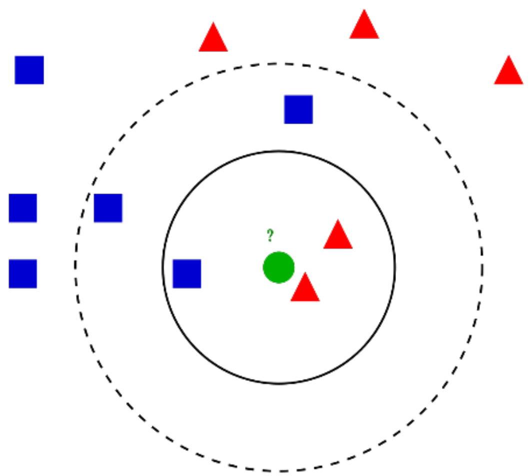
LoPD (ISSRE'14, TOR'16)

```c
int main(int ac, char* av[]){
  int a = atoi(av[1]); // symbolic
  int b = atoi(av[2]); // symbolic
  int c = atoi(av[3]); // symbolic

  if (a > b)
    a = a - b;

  if (b < 1) {
    if (c != a) {
      c = a + b;
    }
  }
  b = 1;
  return 0;
}
```



discovRE (NDSS'16)

- Use numeric features
- Filter features based on their correlation and standard deviation
  - highly correlated features help similar function detection
  - features should not change according to compile options
- Filter target functions
  - k-Nearest Neighbors (kNN)

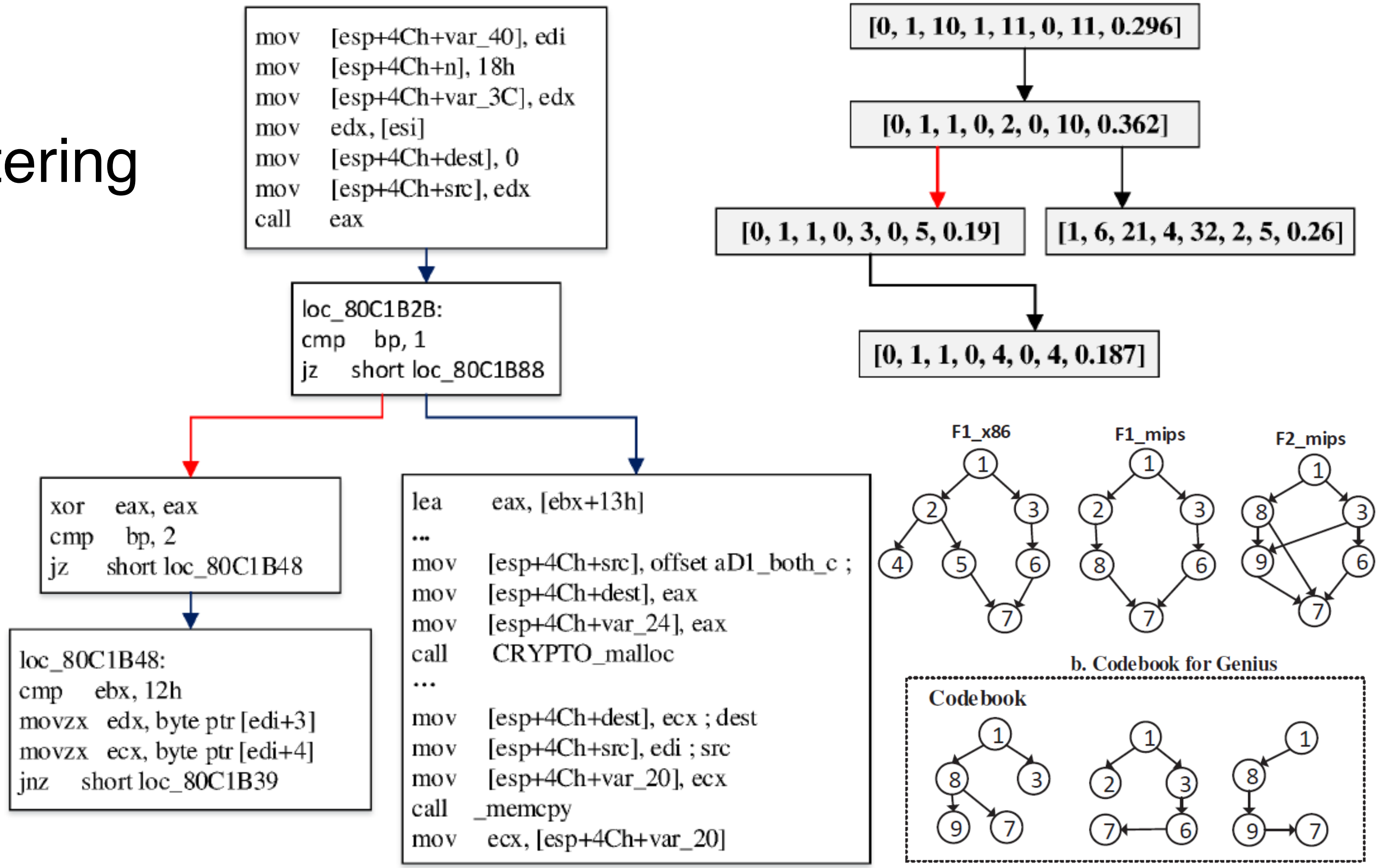


Feature	sd(values)	values	avg.cor	sd(cor)
Arithmetic Instr.	39.483	623	0.907	0.109
Function Calls	22.980	273	0.983	0.073
Logic Instr.	49.607	625	0.953	0.067
Redirections	40.104	556	0.978	0.066
Transfer Instr.	163.443	1,635	0.961	0.075
Local Vars.	2.78E6	890	0.983	0.099
Basic Blocks	48.194	619	0.978	0.067
scc	25.078	389	0.942	0.128
Edges	76.932	835	0.979	0.066
Incoming Calls	46.608	261	0.975	0.086
Instr.	295.408	2,447	0.970	0.069
Parameters	2.157	38	0.720	0.228

Genius (CCS'16)

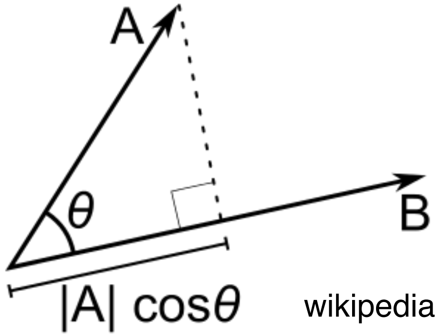
- ❖ Same numeric features
- ❖ Attributed CFG (ACFG)
- ❖ Feature encoding
  - – codebook with spectral clustering

Type	Feature Name
Statistical Features	String Constants
	Numeric Constants
	No. of Transfer Instructions
	No. of Calls
	No. of Instructions
	No. of Arithmetic Instructions
Structural Features	No. of offspring
	Betweeness



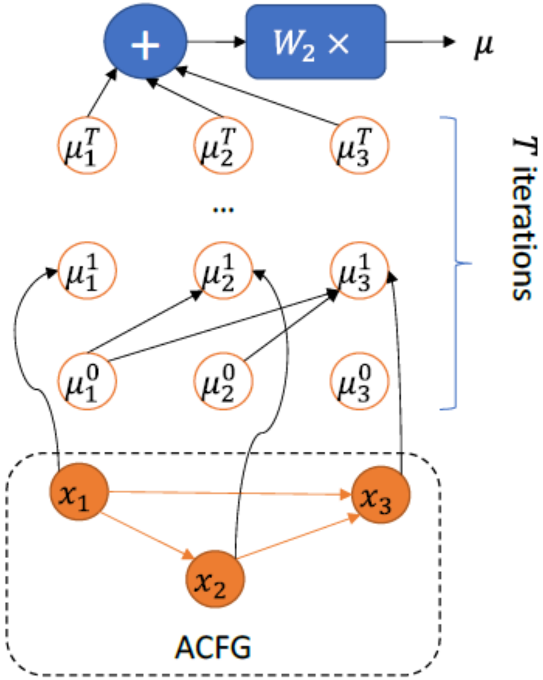
Gemini (CCS'17)

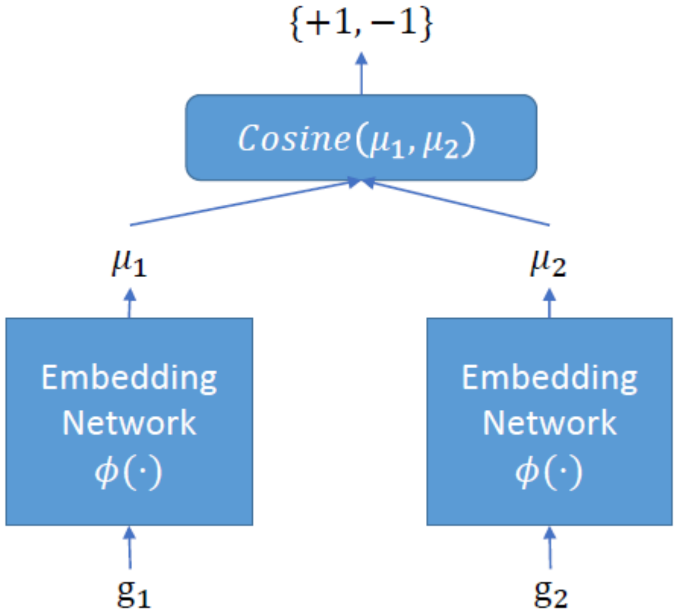
- Same numeric features and create ACFG
- Convert ACFG to a vector using Structure2Vec
- Compare two ACFGs with Siamese architecture



wikipedia

Type	Feature Name
Statistical Features	String Constants Numeric Constants No. of Transfer Instructions No. of Calls No. of Instructions No. of Arithmetic Instructions
Structural Features	No. of offspring Betweeness

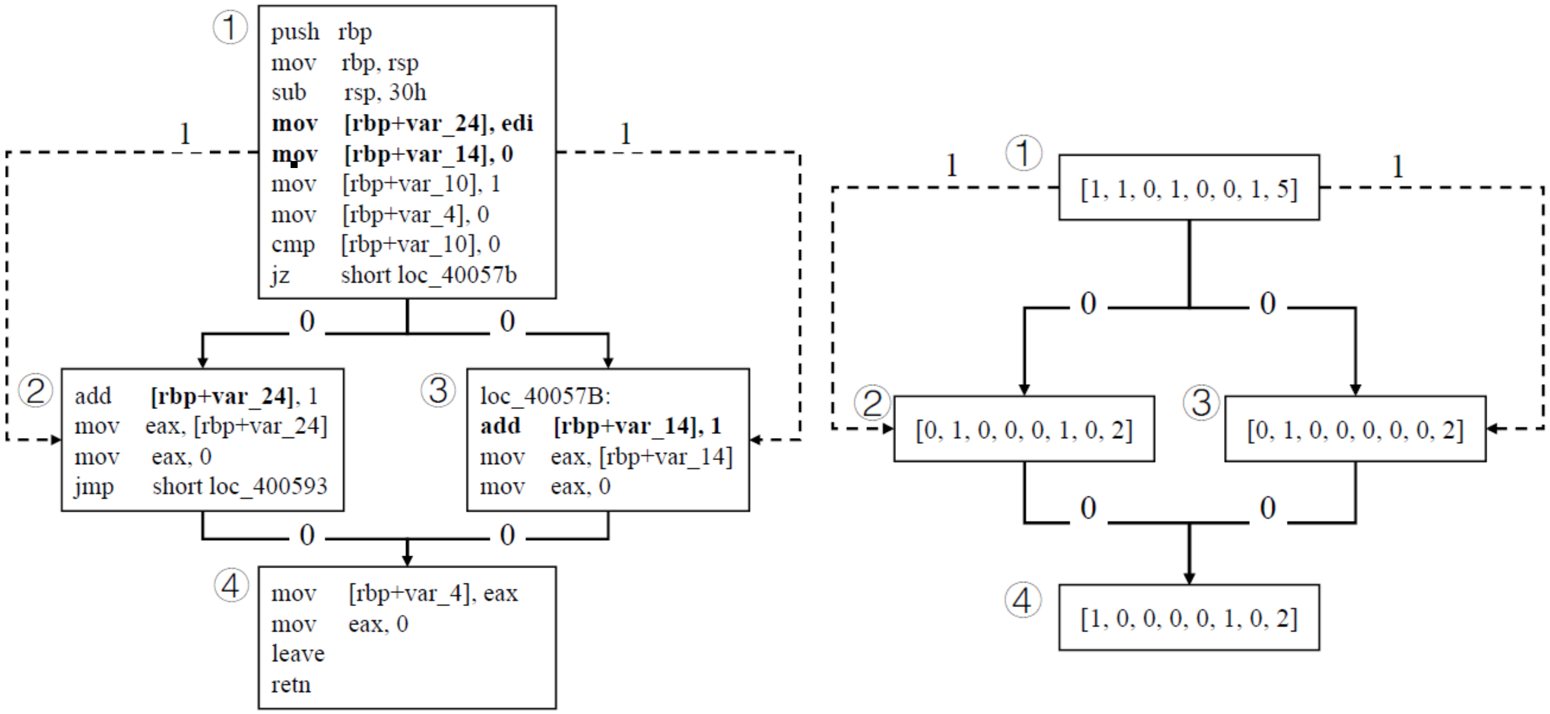




VulSeeker (ASE'18)

- ❖ Use only instruction numeric features
- ❖ Same architecture with Gemini (CCS'17)
- ❖ Add program dependence graph

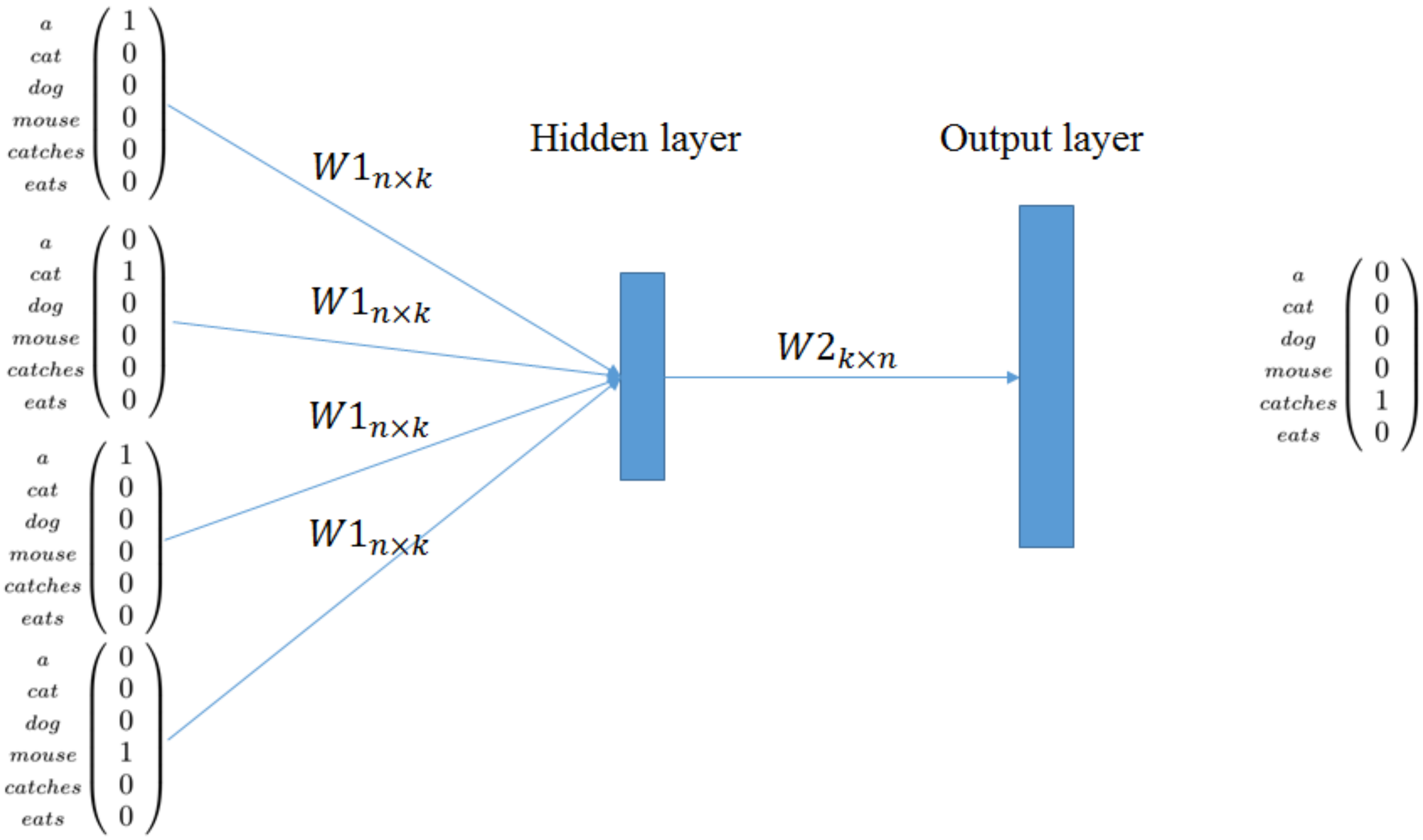
Feature Name	Example
No. of stack operation instructions	push, pop
No. of arithmetic instructions	add, sub
No. of logical instructions	and, or
No. of comparative instructions	test
No. of library function calls	call printf
No. of unconditional jump instructions	jmp
No. of conditional jump instructions	jne, jb
No. of generic instructions	mov, lea



Asm2Vec (SP'19)

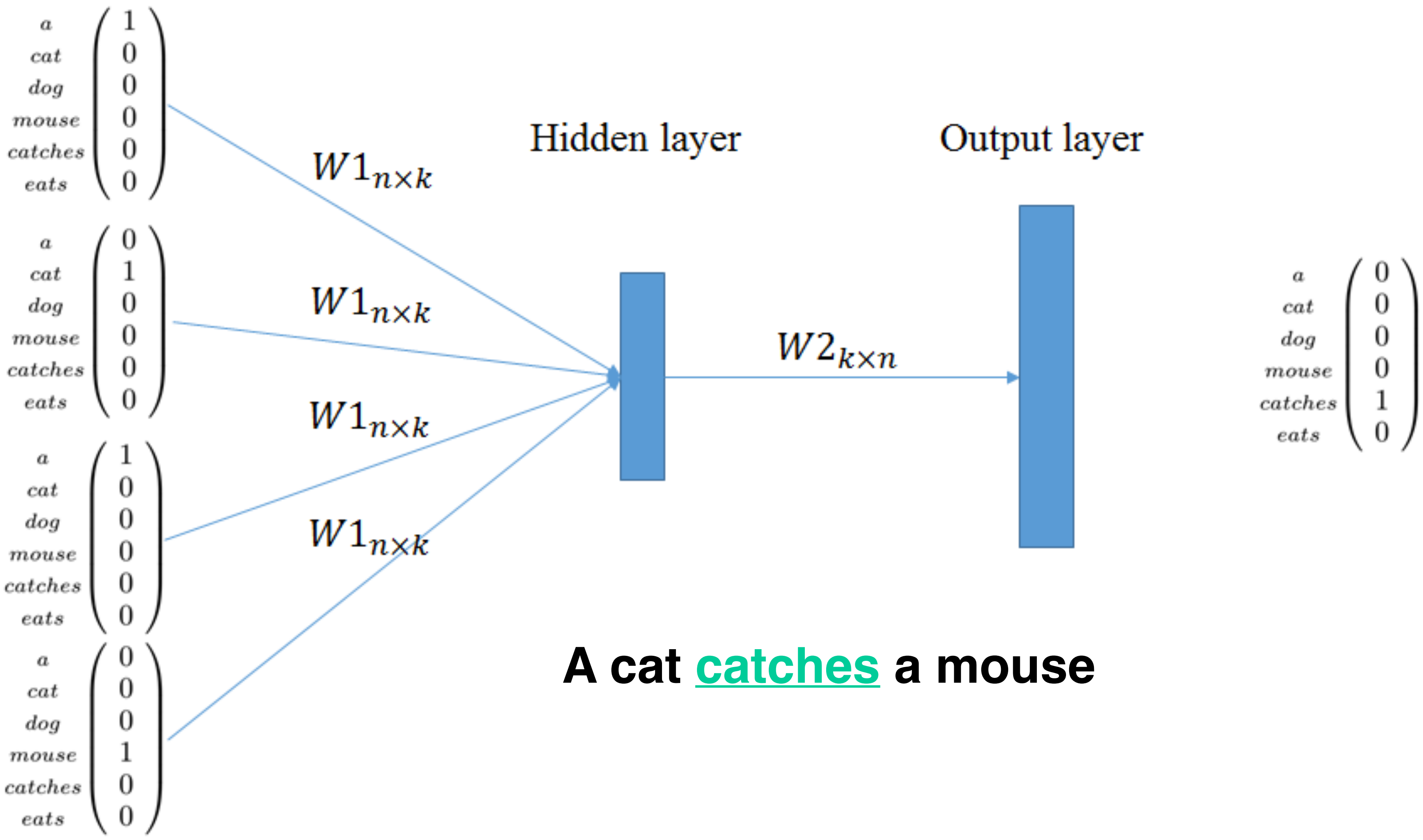
- Applying natural language processing (NLP) to BCSA
- Modify PV-DM to fit x86 assembly instructions

Word2Vec - CBOW



https://www.researchgate.net/figure/word2vec-CBOW-model_fig1_313247648

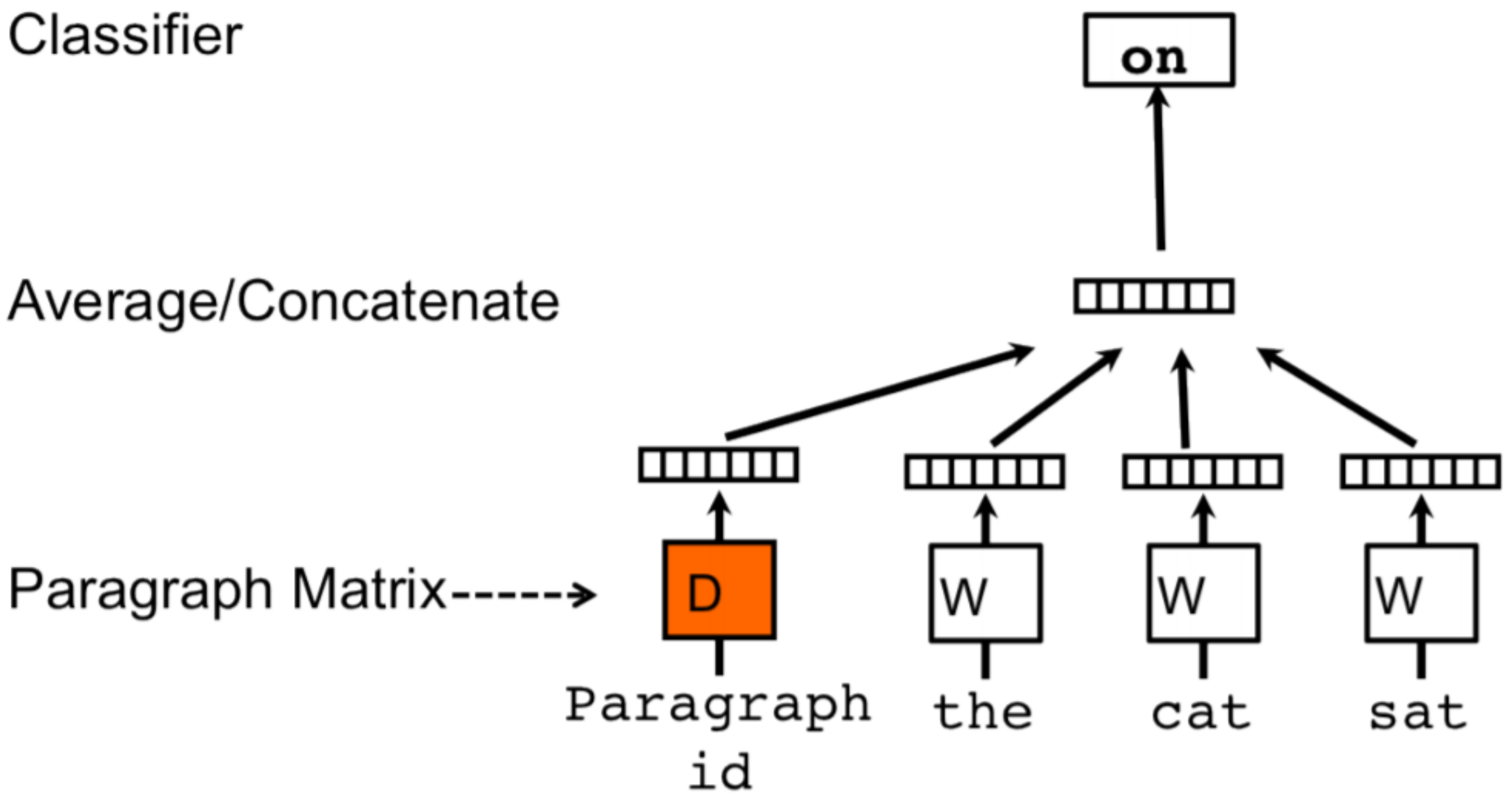
Word2Vec - CBOW



A cat catches a mouse

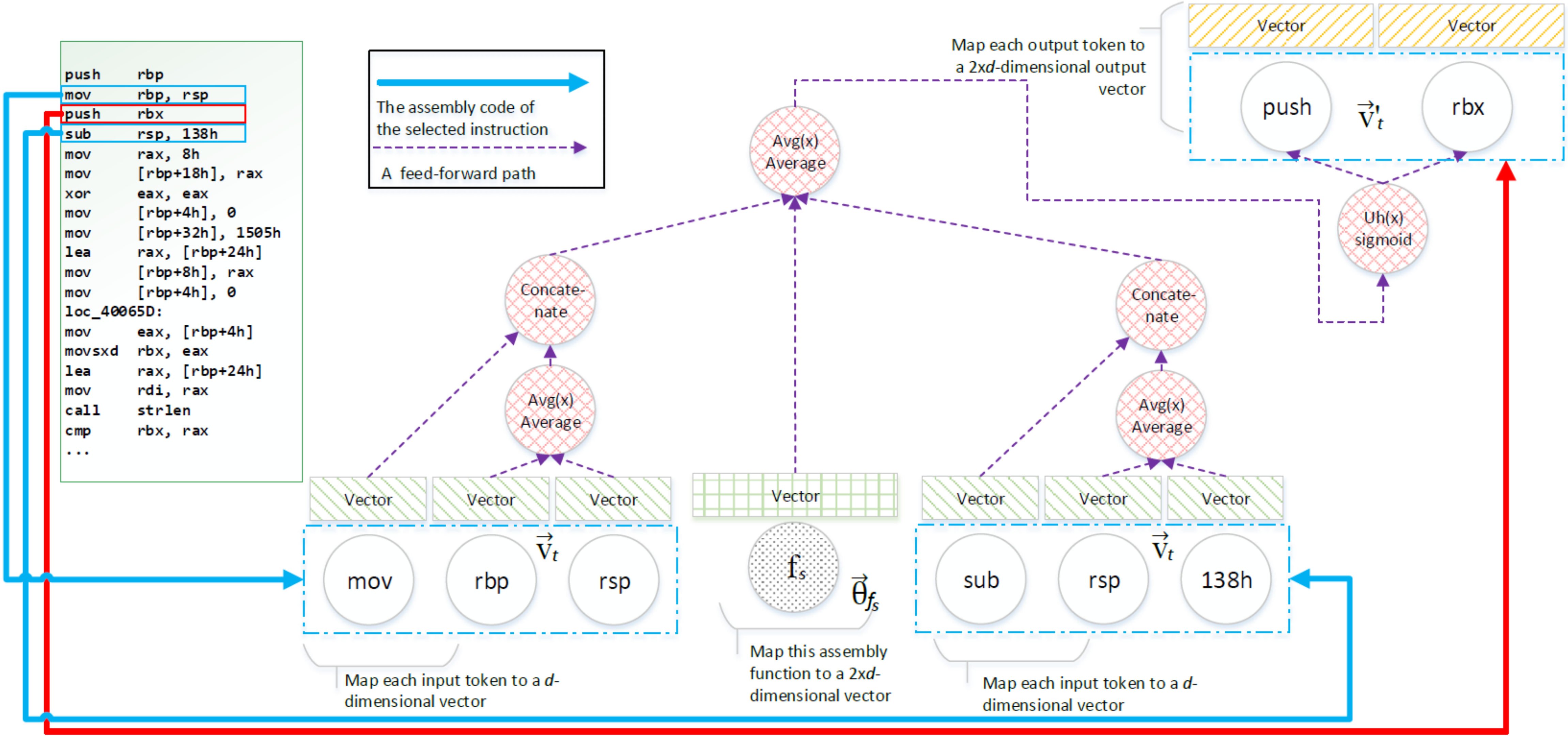
https://www.researchgate.net/figure/word2vec-CBOW-model_fig1_313247648

Doc2Vec - PV-DM

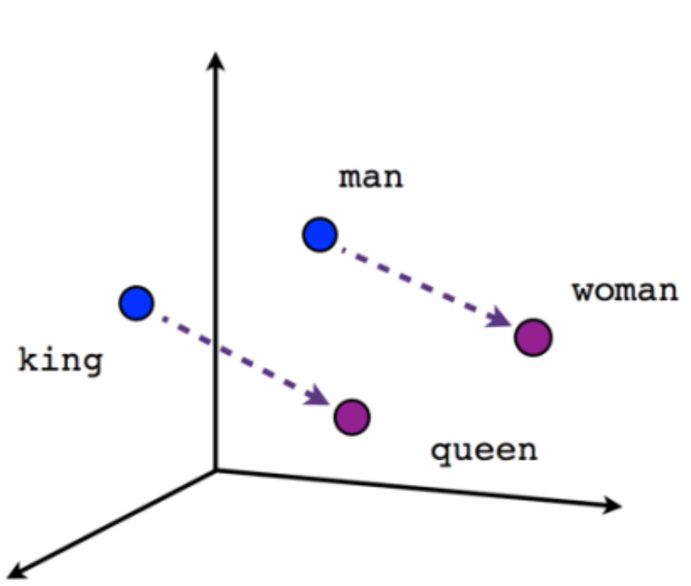


https://medium.com, amarbudhiraja, understanding-document-embeddings-of-doc2vec

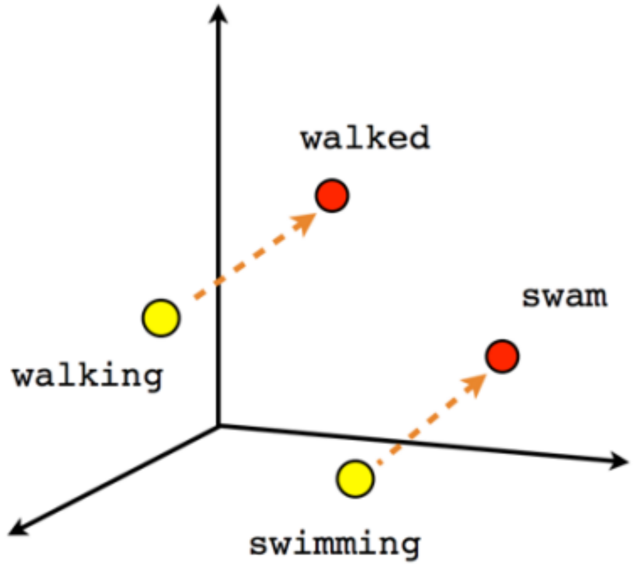
Asm2Vec (SP'19)



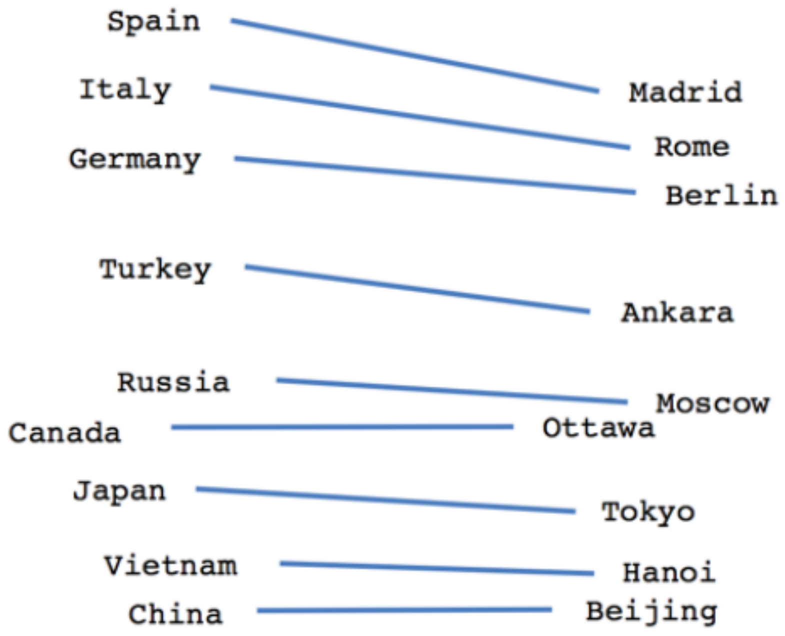
Word2Vec - Vectors



Male-Female

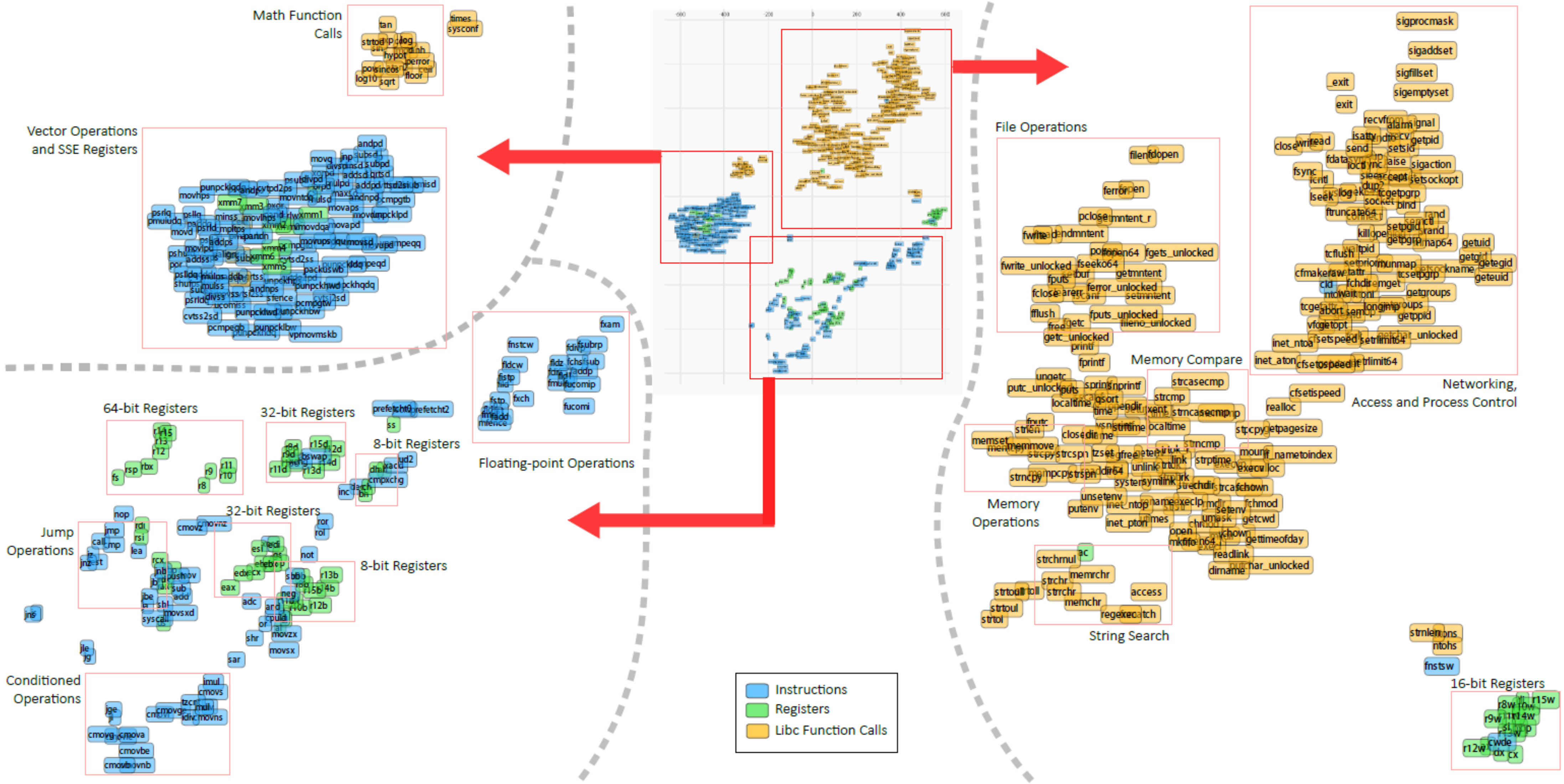


Verb tense



Country-Capital

Asm2Vec (SP'19) - Vectors



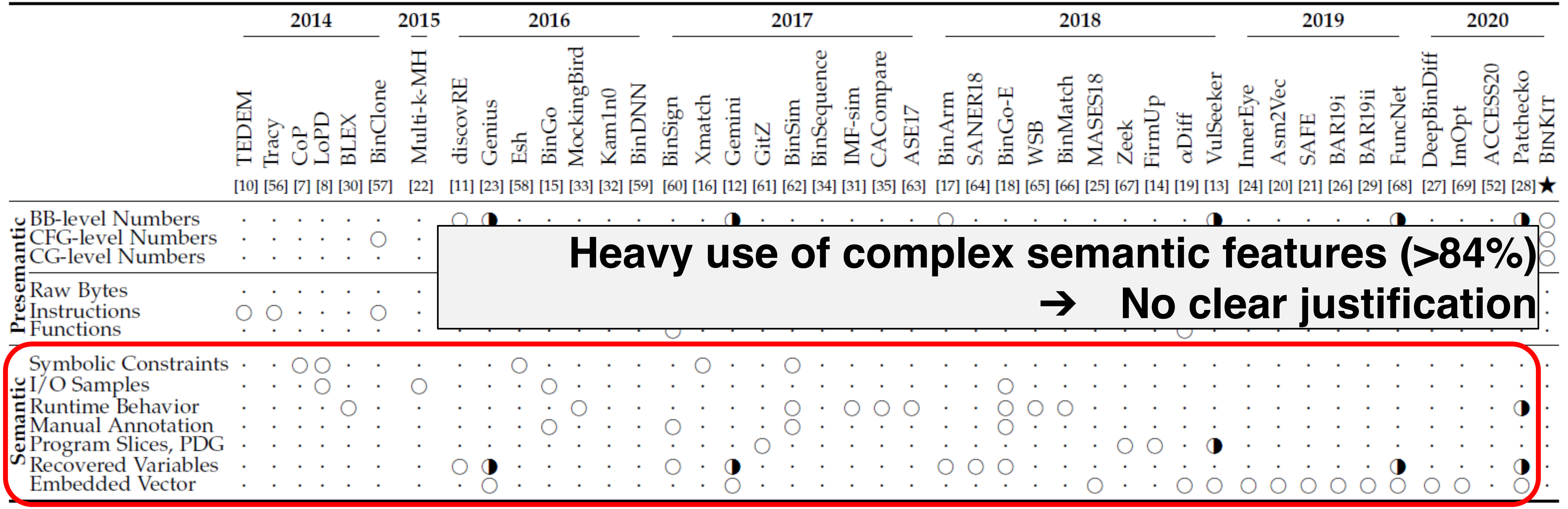
BCSA Features in Previous Literature

		TEDEM [10]	Tracy [56]	CoP [7]	LoPD [8]	BLEX [30]	BinClone [57]	Multi-k-MH [22]	discovRE [11]	Genius [23]	Esh [58]	BinGo [15]	MockingBird [33]	Kam1n0 [32]	BinDNN [59]	BinSign [60]	Xmatch [16]	Gemini [12]	GitZ [61]	BinSim [62]	BinSequence [34]	IMF-sim [31]	CACompare [35]	ASE17 [63]	BinArm [17]	SANER18 [64]	BinGo-E [18]	WSB [65]	BinMatch [66]	MASES18 [25]	Zeek [67]	FirmUp [14]	αDiff [19]	VulSeeker [13]	InnerEye [24]	Asm2Vec [20]	SAFE [21]	BAR19i [26]	BAR19ii [29]	FuncNet [68]	DeepBinDiff [27]	ImOpt [69]	ACCESS20 [52]	Patchecko [28]	BinKit ★
	Year	2014						2015	2016							2017									2018										2019						2020				
Presemantic	BB-level Numbers	·	·	·	·	·	·	·	○	◑	·	·	·	·	·	·	·	◑	·	·	·	·	·	·	○	·	·	·	·	·	·	·	·	◑	·	·	·	·	·	◑	·	·	·	◑	○
Presemantic	CFG-level Numbers	·	·	·	·	·	○	·	○	◑	·	·	·	·	·	○	·	◑	·	·	○	·	·	·	○	○	○	·	·	·	·	·	·	◑	·	·	·	·	·	◑	·	·	·	◑	○
Presemantic	CG-level Numbers	·	·	·	·	·	·	·	○	·	·	○	·	·	·	○	·	·	·	·	·	·	·	·	○	○	○	·	·	·	·	·	○	·	·	◑	·	·	·	·	·	·	·	◑	○
Presemantic	Raw Bytes	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	◑	·	·	◑	·	·	·	·	·	·	·	·	·	○	·	·
Presemantic	Instructions	○	○	·	·	·	○	·	·	·	·	○	·	○	○	·	·	·	·	·	○	·	·	·	·	·	○	·	·	·	·	·	·	·	◑	◑	◑	◑	◑	·	◑	◑	·	·	·
Presemantic	Functions	·	·	·	·	·	·	·	·	·	·	·	·	·	·	○	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	○	·	·	·	·	·	·	·	·	·	·	·	·
Semantic	Symbolic Constraints	·	·	○	○	·	·	·	·	·	○	·	·	·	·	·	○	·	·	○	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·
Semantic	I/O Samples	·	·	·	○	·	·	○	·	·	·	○	·	·	·	·	·	·	·	·	·	·	·	·	·	·	○	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·
Semantic	Runtime Behavior	·	·	·	·	○	·	·	·	·	·	·	○	·	·	·	·	·	·	○	·	○	○	○	·	·	○	○	○	·	·	·	·	·	·	·	·	·	·	·	·	·	·	◑	·
Semantic	Manual Annotation	·	·	·	·	·	·	·	·	·	·	○	·	·	·	○	·	·	·	○	·	·	·	·	·	·	○	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·
Semantic	Program Slices, PDG	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	○	·	·	·	·	·	·	·	·	·	·	·	○	○	·	◑	·	·	·	·	·	·	·	·	·	·	·
Semantic	Recovered Variables	·	·	·	·	·	·	·	○	◑	·	·	·	·	·	○	·	◑	·	·	·	·	·	·	○	○	○	·	·	·	·	·	·	·	·	·	·	·	·	◑	·	·	·	◑	·
Semantic	Embedded Vector	·	·	·	·	·	·	·	·	○	·	·	·	·	·	·	·	○	·	·	·	·	·	·	·	·	·	·	·	○	·	·	○	○	○	○	○	○	○	○	○	○	·	○	·

◑: used with machine learning

*Revisiting Binary Code Similarity Analysis using Interpretable Feature Engineering and Lessons Learned

BCSA Features in Previous Literature



Heavy use of complex semantic features (>84%)

➔ No clear justification

◑: used with machine learning

*Revisiting Binary Code Similarity Analysis using Interpretable Feature Engineering and Lessons Learned

BCSA Features in Previous Literature

Group	Feature	TEDEM [10] (2014)	Tracy [56] (2014)	CoP [7] (2014)	LoPD [8] (2014)	BLEX [30] (2014)	BinClone [57] (2014)	Multi-k-MH [22] (2015)	discovRE [11] (2016)	Genius [23] (2016)	Esh [58] (2016)	BinGo [15] (2016)	MockingBird [33] (2016)	Kam1n0 [32] (2016)	BinDNN [59] (2016)
Presemantic	BB-level Numbers	·	·	·	·	·	·	·	○	◑	·	·	·	·	·
Presemantic	CFG-level Numbers	·	·	·	·	·	○	·	○	◑	·	·	·	·	·
Presemantic	CG-level Numbers	·	·	·	·	·	·	·	○	·	·	○	·	·	·
Presemantic	Raw Bytes	·	·	·	·	·	·	·	·	·	·	·	·	·	·
Presemantic	Instructions	○	○	·	·	·	○	·	·	·	·	○	·	○	○
Presemantic	Functions	·	·	·	·	·	·	·	·	·	·	·	·	·	·
Semantic	Symbolic Constraints	·	·	○	○	·	·	·	·	·	○	·	·	·	·
Semantic	I/O Sam	[illegible]	[illegible]	[illegible]	○	[illegible]	[illegible]	○	[illegible]	[illegible]	[illegible]	○	[illegible]	[illegible]	[illegible]
Semantic	Runtime	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
Semantic	Manual	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
Semantic	Program	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
Semantic	Recovere	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
Semantic	Embedd	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	○	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]

Group	Feature	BinSign [60] (2017)	Xmatch [16] (2017)	Gemini [12] (2017)	GitZ [61] (2017)	BinSim [62] (2017)	BinSequence [34] (2017)	IMF-sim [31] (2017)	CACompare [35] (2017)	ASE17 [63] (2017)	BinArm [17] (2018)	SANER18 [64] (2018)	BinGo-E [18] (2018)	WSB [65] (2018)	BinMatch [66] (2018)	MASES18 [25] (2018)	Zeek [67] (2018)	FirmUp [14] (2018)	αDiff [19] (2018)	VulSeeker [13] (2018)
Presemantic	BB-level Numbers	·	·	◑	·	·	·	·	·	·	○	·	·	·	·	·	·	·	·	◑
Presemantic	CFG-level Numbers	○	·	◑	·	·	○	·	·	·	○	○	○	·	·	·	·	·	·	◑
Presemantic	CG-level Numbers	○	·	·	·	·	·	·	·	·	○	○	○	·	·	·	·	·	○	·
Presemantic	Raw Bytes	·	·	·	·	·	·	·	·	·	·	·	·	·	·	◑	·	·	◑	·
Presemantic	Instructions	·	·	·	·	·	○	·	·	·	·	·	○	·	·	·	·	·	·	·
Presemantic	Functions	○	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	○	·
Semantic	Symbolic Constraints	·	○	·	·	○	·	·	·	·	·	·	·	·	·	·	·	·	·	·
Semantic	I/O Sam	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	○	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	·
Semantic	Runtime	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	·
Semantic	Manual	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	·
Semantic	Program	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	◑
Semantic	Recovere	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	·
Semantic	Embedd	[illegible]	[illegible]	○	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	○	[illegible]	[illegible]	○	○

Group	Feature	InnerEye [24] (2019)	Asm2Vec [20] (2019)	SAFE [21] (2019)	BAR19i [26] (2019)	BAR19ii [29] (2019)	FuncNet [68] (2019)	DeepBinDiff [27] (2020)	ImOpt [69] (2020)	ACCESS20 [52] (2020)	Patchecko [28] (2020)	BINKIT ★
Presemantic	BB-level Numbers	·	·	·	·	·	◑	·	·	·	◑	○
Presemantic	CFG-level Numbers	·	·	·	·	·	◑	·	·	·	◑	○
Presemantic	CG-level Numbers	·	◑	·	·	·	·	·	·	·	◑	○
Presemantic	Raw Bytes	·	·	·	·	·	·	·	·	○	·	·
Presemantic	Instructions	◑	◑	◑	◑	◑	·	◑	◑	·	·	·
Presemantic	Functions	·	·	·	·	·	·	·	·	·	·	·
Semantic	Symbolic Constraints	·	·	·	·	·	·	·	·	·	·	·
Semantic	I/O Sam	·	·	·	·	·	·	·	·	·	·	·
Semantic	Runtime	·	·	·	·	·	·	·	·	·	◑	·
Semantic	Manual	·	·	·	·	·	·	·	·	·	·	·
Semantic	Program	·	·	·	·	·	·	·	·	·	·	·
Semantic	Recovere	·	·	·	·	·	◑	·	·	·	◑	·
Semantic	Embedd	○	○	○	○	○	○	○	○	·	○	·

Heavy use of complex machine learning (>90%)

➔ Hard to interpret/understand the results

◑: used with machine learning

*Revisiting Binary Code Similarity Analysis using Interpretable Feature Engineering and Lessons Learned

BCSA Dataset in Previous Literature

		Source*		Architecture*								Optimization*					Compiler†*													Extra				Info.		
Year	Tool [Paper]	Binaries	Firmware	x86	x64	arm	aarch64	mips	mips64	mipseb	mips64eb	O0	O1	O2	O3	Os	GCC 3	GCC 4	GCC 5	GCC 6	GCC 7	GCC 8	Clang 3	Clang 4	Clang 5	Clang 6	Clang 7	etc.	Total #	Noinline	PIE	LTO	Obfus.	Code	Dataset	IDA
	TEDEM [10]	14	·	○	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	○
	Tracy [56]	(115)	·	·	○	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	○	·	○
2014	CoP [7]	(214)	·	·	·	·	·	·	·	·	·	○	○	○	○	○	·	1	·	·	·	·	·	·	·	·	·	1	2	·	·	·	○	·	·	○
	LoPD [8]	48	·	○	·	·	·	·	·	·	·	○	○	○	○	○	·	1	·	·	·	·	·	·	·	·	·	1	2	·	·	○	○	·	·	·
	BLEX [30]	1,140	·	·	○	·	·	·	·	·	·	○	○	○	○	·	·	1	·	·	·	·	1	·	·	·	·	1	3	·	·	·	·	·	·	○
	BinClone [57]	90	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	○	·	○
2015	Multi-k-MH [22]	60	6	○	·	○	·	○	·	·	·	○	○	○	○	·	·	2	·	·	·	·	1	·	·	·	·	·	3	·	·	·	·	·	·	○
	discovRE [11]	593	3	○	·	○	·	○	·	·	·	○	○	○	○	○	·	1	·	·	·	·	1	·	·	·	·	2	4	○	·	·	·	○	◑	○
	Genius [23]	(7,848)	8,128	○	·	○	·	○	·	·	·	○	○	○	○	·	·	2	·	·	·	·	1	·	·	·	·	·	3	·	·	·	·	·	·	○
	Esh [58]	(833)	·	·	○	·	·	·	·	·	·	·	·	·	·	·	·	3	·	·	·	·	2	·	·	·	·	2	7	·	·	·	·	○	○	○
2016	BinGo [15]	(5,143)	·	○	○	○	·	·	·	·	·	○	○	○	○	·	·	3	·	·	·	·	1	·	·	·	·	1	5	·	·	·	·	·	·	○
	MockingBird [33]	80	·	○	·	○	·	○	·	·	·	○	·	○	○	·	·	1	·	·	·	·	1	·	·	·	·	·	2	·	·	·	·	·	·	○
	Kam1n0 [32]	96	·	○	○	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	○	·	○
	BinDNN [59]	2,064	·	○	○	○	·	·	·	·	·	○	○	○	○	·	·	1	·	·	·	·	·	·	·	·	·	1	2	·	·	·	·	·	·	○
	BinSign [60]	(31)	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	2	2	·	·	·	○	·	·	○
	Xmatch [16]	72	1	○	·	·	·	○	·	·	·	·	·	·	·	·	·	2	·	·	·	·	1	·	·	·	·	·	3	○	·	·	·	·	·	○
	Gemini [12]	18,269	8,128	○	·	○	·	○	·	·	·	○	○	○	○	·	·	·	1	·	·	·	·	·	·	·	·	·	1	·	·	·	·	·	·	○
	GitZ [61]	44	·	·	○	·	○	·	·	·	·	○	○	○	○	○	·	3	·	·	·	·	2	1	·	·	·	·	6	·	·	·	·	·	·	·
2017	BinSim [62]	1062	·	○	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	○	·	·	○
	BinSequence [34]	(1,718)	·	·	○	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	○
	IMF-sim [31]	1,140	·	·	○	·	·	·	·	·	·	○	·	○	○	·	·	1	·	·	·	·	1	·	·	·	·	1	3	·	·	·	○	·	·	·
	CACompare [35]	72	·	○	·	○	·	○	·	·	·	○	·	○	○	·	·	1	·	·	·	·	1	·	·	·	·	·	2	·	·	·	·	·	·	○
	ASE17 [63]	55	·	○	○	·	·	·	·	·	·	○	·	○	○	·	·	1	·	·	·	·	1	·	·	·	·	·	2	·	·	·	○	·	·	·
	BinArm [17]	·	2,628	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	○
	SANER18 [64]	7	·	○	○	·	·	·	·	·	·	·	·	·	·	·	·	1	1	1	·	·	1	·	·	·	·	1	5	·	·	·	·	·	·	○
	BinGo-E [18]	(5,145)	·	○	○	○	·	·	·	·	·	○	○	○	○	·	·	3	·	·	·	·	1	·	·	·	·	1	5	·	·	·	·	·	·	○
	WSB [65]	(173)	·	·	·	·	·	·	·	·	·	·	○	○	○	·	·	1	·	·	·	·	1	·	·	·	·	·	2	·	·	·	○	·	·	·
2018	BinMatch [66]	80	·	○	·	·	·	·	·	·	·	○	·	○	○	·	·	1	·	·	·	·	1	·	·	·	·	·	2	·	·	·	○	·	·	○
	MASES18 [25]	47	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	○	·	·	·
	Zeek [67]	(20,680)	·	○	·	○	○	·	·	·	·	○	○	○	○	○	·	3	·	·	·	·	4	1	·	·	·	2	10	·	·	·	·	·	·	·
	FirmUp [14]	·	2,000	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	○
	αDiff [19]	(69,989)	2	○	○	○	·	·	·	·	·	○	○	○	○	·	·	2	1	·	·	·	2	·	·	·	·	·	5	·	·	·	·	◑	◑	○
	VulSeeker [13]	(10,512)	4,643	○	○	○	○	○	○	·	·	○	○	○	○	·	·	1	1	·	·	·	·	·	·	·	·	·	2	·	·	·	·	○	·	○
	InnerEye [24]	(844)	·	·	○	○	·	·	·	·	·	·	○	○	○	·	·	·	·	·	·	·	·	·	·	1	·	·	1	·	·	·	·	◑	◑	·
	Asm2Vec [20]	68	·	·	○	·	·	·	·	·	·	○	○	○	○	·	·	1	1	·	·	·	2	·	·	·	·	·	4	·	·	·	○	○	·	○
2019	SAFE [21]	(5001)	·	·	○	○	·	·	·	·	·	○	○	○	○	·	1	3	1	1	1	·	2	1	1	1	·	·	12	·	·	·	·	○	◑	○
	BAR19i [26]	(804)	·	·	○	○	·	·	·	·	·	·	○	○	○	·	·	·	·	·	·	·	·	·	·	1	·	·	1	·	·	·	·	○	·	·
	BAR19ii [29]	(11244)	·	○	○	○	·	·	·	·	·	○	○	○	○	·	1	3	1	·	·	·	2	1	1	·	·	2	11	·	·	·	·	·	·	·
	FuncNet [68]	(180)	·	○	·	○	·	○	·	·	·	○	○	○	○	○	·	·	·	1	·	·	·	·	·	·	·	·	1	·	·	·	·	·	·	○
	DeepBinDiff [27]	2114	·	·	·	·	·	·	·	·	·	○	○	○	○	·	·	1	·	·	·	·	·	·	·	·	·	·	1	·	·	·	·	○	○	○
	ImOpt [69]	18	·	·	○	·	·	·	·	·	·	○	·	○	○	·	·	·	1	·	·	·	·	·	·	·	·	·	1	·	·	·	○	·	·	·
2020	ACCESS20 [52]	12,000	·	○	○	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·
	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	○	○	○	○	○	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	·	·	·	·	·	·	○
	BinKit ★	243,128	·	○	○	○	○	○	○	○	○	○	○	○	○	○	·	1	1	1	1	1	·	1	1	1	1	·	9	○	○	○	○	○	○	○

*Revisiting Binary Code S

BCSA Dataset in Previous Literature

		Source*		Architecture*								Optimization*					Compiler†*													Extra				Info.		
Year	Tool [Paper]	Binaries	Firmware	x86	x64	arm	aarch64	mips	mips64	mipseb	mips64eb	O0	O1	O2	O3	Os	GCC 3	GCC 4	GCC 5	GCC 6	GCC 7	GCC 8	Clang 3	Clang 4	Clang 5	Clang 6	Clang 7	etc.	Total #	Noinline	PIE	LTO	Obfus.	Code	Dataset	IDA
2014	TEDEM [10]	14	·	○	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	○
	Tracy [56]	(115)	·	·	○	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	○	·	○
	CoP [7]	(214)	·	·	·	·	·	·	·	·	·	○	○	○	○	○	·	1	·	·	·	·	·	·	·	·	·	1	2	·	·	·	○	·	·	○
	LoPD [8]	48	·	○	·	·	·	·	·	·	·	○	○	○	○	○	·	1	·	·	·	·	·	·	·	·	·	1	2	·	·	○	○	·	·	·
	BLEX [30]	1,140	·	·	○	·	·	·	·	·	·	○	○	○	○	·	·	1	·	·	·	·	1	·	·	·	·	1	3	·	·	·	·	·	·	○
	BinClone [57]	90	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	○	·	○
2015	Multi-k-MH [22]	60	6	○	·	○	·	○	·	·	·	○	○	○	○	·	·	2	·	·	·	·	1	·	·	·	·	·	3	·	·	·	·	·	·	○
2016	discovRE [11]	593	3	○	·	○	·	○	·	·	·	○	○	○	○	○	·	1	·	·	·	·	1	·	·	·	·	2	4	○	·	·	·	○	◑	○
	Genius [23]	(7,848)	8,128	○	·	○	·	○	·	·	·	○	○	○	○	·	·	2	·	·	·	·	1	·	·	·	·	·	3	·	·	·	·	·	·	○
	Esh [58]	(833)	·	·	○	·	·	·	·	·	·	·	·	·	·	·	·	3	·	·	·	·	2	·	·	·	·	2	7	·	·	·	·	○	○	○
	BinGo [15]	(5,143)	·	○	○	○	·	·	·	·	·	○	○	○	○	·	·	3	·	·	·	·	1	·	·	·	·	1	5	·	·	·	·	·	·	○
	MockingBird [33]	80	·	○	·	○	·	○	·	·	·	○	·	○	○	·	·	1	·	·	·	·	1	·	·	·	·	·	2	·	·	·	·	·	·	○
	Kam1n0 [32]	96	·	○	○	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	○	·	○
	BinDNN [59]	2,064	·	○	○	○	·	·	·	·	·	○	○	○	○	·	·	1	·	·	·	·	·	·	·	·	·	1	2	·	·	·	·	·	·	○
2017	BinSign [60]	(31)	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	2	2	·	·	·	○	·	·	○
	Xmatch [16]	72	1	○	·	·	·	○	·	·	·	·	·	·	·	·	·	2	·	·	·	·	1	·	·	·	·	·	3	○	·	·	·	·	·	○
	Gemini [12]	18,269	8,128	○	·	○	·	○	·	·	·	○	○	○	○	·	·	·	1	·	·	·	·	·	·	·	·	·	1	·	·	·	·	·	·	○
	GitZ [61]	44	·	·	○	·	○	·	·	·	·	○	○	○	○	○	·	3	·	·	·	·	2	1	·	·	·	·	6	·	·	·	·	·	·	·
	BinSim [62]	1062	·	○	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	○	·	·	○
	BinSequence [34]	(1,718)	·	·	○	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	○
	IMF-sim [31]	1,140	·	·	○	·	·	·	·	·	·	○	·	○	○	·	·	1	·	·	·	·	1	·	·	·	·	1	3	·	·	·	○	·	·	·
	CACompare [35]	72	·	○	·	○	·	○	·	·	·	○	·	○	○	·	·	1	·	·	·	·	1	·	·	·	·	·	2	·	·	·	·	·	·	○
	ASE17 [63]	55	·	○	○	·	·	·	·	·	·	○	·	○	○	·	·	1	·	·	·	·	1	·	·	·	·	·	2	·	·	·	○	·	·	·
2018	BinArm [17]	·	2,628	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	○
	SANER18 [64]	7	·	○	○	·	·	·	·	·	·	·	·	·	·	·	·	1	1	1	·	·	1	·	·	·	·	1	5	·	·	·	·	·	·	○
	BinGo-E [18]	(5,145)	·	○	○	○	·	·	·	·	·	○	○	○	○	·	·	3	·	·	·	·	1	·	·	·	·	1	5	·	·	·	·	·	·	○
	WSB [65]	(173)	·	·	·	·	·	·	·	·	·	·	○	○	○	·	·	1	·	·	·	·	1	·	·	·	·	·	2	·	·	·	○	·	·	·
	BinMatch [66]	80	·	○	·	·	·	·	·	·	·	○	·	○	○	·	·	1	·	·	·	·	1	·	·	·	·	·	2	·	·	·	○	·	·	○
	MASES18 [25]	47	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	○	·	·	·
	Zeek [67]	(20,680)	·	○	·	○	○	·	·	·	·	○	○	○	○	○	·	3	·	·	·	·	4	1	·	·	·	2	10	·	·	·	·	·	·	·
	FirmUp [14]	·	2,000	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	○
	αDiff [19]	(69,989)	2	○	○	○	·	·	·	·	·	○	○	○	○	·	·	2	1	·	·	·	2	·	·	·	·	·	5	·	·	·	·	◑	◑	○
	VulSeeker [13]	(10,512)	4,643	○	○	○	○	○	○	·	·	○	○	○	○	·	·	1	1	·	·	·	·	·	·	·	·	·	2	·	·	·	·	○	·	○
2019	InnerEye [24]	(844)	·	·	○	○	·	·	·	·	·	·	○	○	○	·	·	·	·	·	·	·	·	·	·	1	·	·	1	·	·	·	·	◑	◑	·
	Asm2Vec [20]	68	·	·	○	·	·	·	·	·	·	○	○	○	○	·	·	1	1	·	·	·	2	·	·	·	·	·	4	·	·	·	○	○	·	○
	SAFE [21]	(5001)	·	·	○	○	·	·	·	·	·	○	○	○	○	○	1	3	1	1	1	·	2	1	1	1	·	·	12	·	·	·	·	○	◑	○
	BAR19i [26]	(804)	·	·	○	○	·	·	·	·	·	·	○	○	○	·	·	·	·	·	·	·	·	·	·	1	·	·	1	·	·	·	·	○	·	·
	BAR19ii [29]	(11244)	·	○	○	○	·	·	·	·	·	○	○	○	○	·	1	3	1	·	·	·	2	1	1	·	·	2	11	·	·	·	·	·	·	·
	FuncNet [68]	(180)	·	○	·	○	·	○	·	·	·	○	○	○	○	○	·	·	·	1	·	·	·	·	·	·	·	·	1	·	·	·	·	·	·	○
2020	DeepBinDiff [27]	2114	·	·	·	·	·																											○	○	○
	ImOpt [69]	18	·	·	○	·	·																											·	·	·
	ACCESS20 [52]	12,000	·	○	○	·	·																											·	·	·
	Patchecko [28]	2,108	2	○	○	○	○					○	○	○	○	○																		·	·	○
	BinKit ★	243,128	·	○	○	○	○	○	○	○	○	○	○	○	○	○	·	1	1	1	1	1	·	1	1	1	1	·	9	○	○	○	○	○	○	○

No same benchmark

*Revisiting Binary Code S

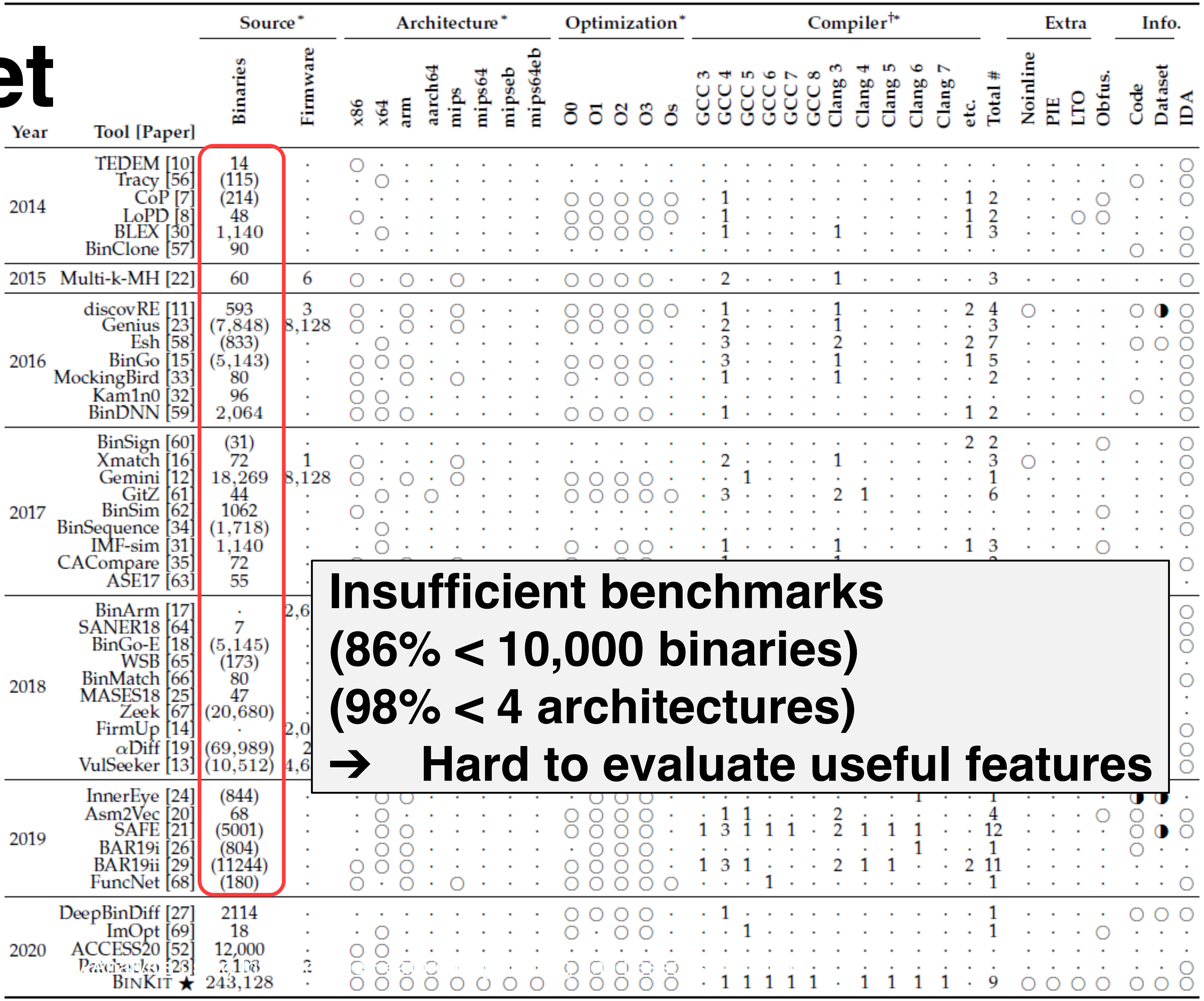
BCSA Dataset in Previous Literature

		Source*		Architecture*								Optimization*					Compiler†*													Extra				Info.		
Year	Tool [Paper]	Binaries	Firmware	x86	x64	arm	aarch64	mips	mips64	mipseb	mips64eb	O0	O1	O2	O3	Os	GCC 3	GCC 4	GCC 5	GCC 6	GCC 7	GCC 8	Clang 3	Clang 4	Clang 5	Clang 6	Clang 7	etc.	Total #	Noinline	PIE	LTO	Obfus.	Code	Dataset	IDA
2014	TEDEM [10]	14	·	○	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	○
	Tracy [56]	(115)	·	·	○	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	○	·	○
	CoP [7]	(214)	·	·	·	·	·	·	·	·	·	○	○	○	○	○	·	1	·	·	·	·	·	·	·	·	·	1	2	·	·	·	○	·	·	○
	LoPD [8]	48	·	○	·	·	·	·	·	·	·	○	○	○	○	○	·	1	·	·	·	·	·	·	·	·	·	1	2	·	·	○	○	·	·	·
	BLEX [30]	1,140	·	·	○	·	·	·	·	·	·	○	○	○	○	·	·	1	·	·	·	·	1	·	·	·	·	1	3	·	·	·	·	·	·	○
	BinClone [57]	90	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	○	·	○
2015	Multi-k-MH [22]	60	6	○	·	○	·	○	·	·	·	○	○	○	○	·	·	2	·	·	·	·	1	·	·	·	·	·	3	·	·	·	·	·	·	○
2016	discovRE [11]	593	3	○	·	○	·	○	·	·	·	○	○	○	○	○	·	1	·	·	·	·	1	·	·	·	·	2	4	○	·	·	·	○	◑	○
	Genius [23]	(7,848)	8,128	○	·	○	·	○	·	·	·	○	○	○	○	·	·	2	·	·	·	·	1	·	·	·	·	·	3	·	·	·	·	·	·	○
	Esh [58]	(833)	·	·	○	·	·	·	·	·	·	·	·	·	·	·	·	3	·	·	·	·	2	·	·	·	·	2	7	·	·	·	·	○	○	○
	BinGo [15]	(5,143)	·	○	○	○	·	·	·	·	·	○	○	○	○	·	·	3	·	·	·	·	1	·	·	·	·	1	5	·	·	·	·	·	·	○
	MockingBird [33]	80	·	○	·	○	·	○	·	·	·	○	·	○	○	·	·	1	·	·	·	·	1	·	·	·	·	·	2	·	·	·	·	·	·	○
	Kam1n0 [32]	96	·	○	○	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	○	·	○
	BinDNN [59]	2,064	·	○	○	○	·	·	·	·	·	○	○	○	○	·	·	1	·	·	·	·	·	·	·	·	·	1	2	·	·	·	·	·	·	○
2017	BinSign [60]	(31)	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	2	2	·	·	·	○	·	·	○
	Xmatch [16]	72	1	○	·	·	·	○	·	·	·	·	·	·	·	·	·	2	·	·	·	·	1	·	·	·	·	·	3	○	·	·	·	·	·	○
	Gemini [12]	18,269	8,128	○	·	○	·	○	·	·	·	○	○	○	○	·	·	·	1	·	·	·	·	·	·	·	·	·	1	·	·	·	·	·	·	○
	GitZ [61]	44	·	·	○	·	○	·	·	·	·	○	○	○	○	○	·	3	·	·	·	·	2	1	·	·	·	·	6	·	·	·	·	·	·	·
	BinSim [62]	1062	·	○	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	○	·	·	○
	BinSequence [34]	(1,718)	·	·	○	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	○
	IMF-sim [31]	1,140	·	·	○	·	·	·	·	·	·	○	·	○	○	·	·	1	·	·	·	·	1	·	·	·	·	1	3	·	·	·	○	·	·	·
	CACompare [35]	72	·	○	·	○	·	○	·	·	·	○	·	○	○	·	·	1	·	·	·	·	1	·	·	·	·	·	2	·	·	·	·	·	·	○
	ASE17 [63]	55	·	○	○	·	·	·	·	·	·	○	·	○	○	·	·	1	·	·	·	·	1	·	·	·	·	·	2	·	·	·	○	·	·	·
2018	BinArm [17]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	·	·	·	○
	SANER18 [64]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	·	·	·	○
	BinGo-E [18]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	·	·	·	○
	WSB [65]	(173)	·	·	·	·	·	·	·	·	·	·	○	○	○	·	·	1	·	·	·	·	1	·	·	·	·	·	2	·	·	·	○	·	·	·
	BinMatch [66]	80	·	○	·	·	·	·	·	·	·	○	·	○	○	·	·	1	·	·	·	·	1	·	·	·	·	·	2	·	·	·	○	·	·	○
	MASES18 [25]	47	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	○	·	·	·
	Zeek [67]	(20,680)	·	○	·	○	○	·	·	·	·	○	○	○	○	○	·	3	·	·	·	·	4	1	·	·	·	2	10	·	·	·	·	·	·	·
	FirmUp [14]	·	2,000	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	○
	αDiff [19]	(69,989)	2	○	○	○	·	·	·	·	·	○	○	○	○	·	·	2	1	·	·	·	2	·	·	·	·	·	5	·	·	·	·	◑	◑	○
	VulSeeker [13]	(10,512)	4,643	○	○	○	○	○	○	·	·	○	○	○	○	·	·	1	1	·	·	·	·	·	·	·	·	·	2	·	·	·	·	○	·	○
2019	InnerEye [24]	(844)	·	·	○	○	·	·	·	·	·	·	○	○	○	·	·	·	·	·	·	·	·	·	·	1	·	·	1	·	·	·	·	◑	◑	·
	Asm2Vec [20]	68	·	·	○	·	·	·	·	·	·	○	○	○	○	·	·	1	1	·	·	·	2	·	·	·	·	·	4	·	·	·	○	○	·	○
	SAFE [21]	(5001)	·	·	○	○	·	·	·	·	·	○	○	○	○	○	1	3	1	1	1	·	2	1	1	1	·	·	12	·	·	·	·	○	◑	○
	BAR19i [26]	(804)	·	·	○	○	·	·	·	·	·	·	○	○	○	·	·	·	·	·	·	·	·	·	·	1	·	·	1	·	·	·	·	○	·	·
	BAR19ii [29]	(11244)	·	○	○	○	·	·	·	·	·	○	○	○	○	·	1	3	1	·	·	·	2	1	1	·	·	2	11	·	·	·	·	·	·	·
	FuncNet [68]	(180)	·	○	·	○	·	○	·	·	·	○	○	○	○	○	·	·	·	1	·	·	·	·	·	·	·	·	1	·	·	·	·	·	·	○
2020	DeepBinDiff [27]	2114	·	·	·	·	·	·	·	·	·	○	○	○	○	·	·	1	·	·	·	·	·	·	·	·	·	·	1	·	·	·	·	○	○	○
	ImOpt [69]	18	·	·	○	·	·	·	·	·	·	○	·	○	○	·	·	·	1	·	·	·	·	·	·	·	·	·	1	·	·	·	○	·	·	·
	ACCESS20 [52]	12,000	·	○	○	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·
	[illegible] [28]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	·	·	○
	BinKit ★	243,128	·	○	○	○	○	○	○	○	○	○	○	○	○	○	·	1	1	1	1	1	·	1	1	1	1	·	9	○	○	○	○	○	○	○

Only 2 released full dataset

*Revisiting Binary Code S

BCSA Dataset in Previous Literature



Year	Tool [Paper]	Binaries	Firmware
2014	TEDEM [10]	14	·
	Tracy [56]	(115)	·
	CoP [7]	(214)	·
	LoPD [8]	48	·
	BLEX [30]	1,140	·
	BinClone [57]	90	·
2015	Multi-k-MH [22]	60	6
2016	discovRE [11]	593	3
	Genius [23]	(7,848)	8,128
	Esh [58]	(833)	·
	BinGo [15]	(5,143)	·
	MockingBird [33]	80	·
	Kam1n0 [32]	96	·
	BinDNN [59]	2,064	·
2017	BinSign [60]	(31)	·
	Xmatch [16]	72	1
	Gemini [12]	18,269	8,128
	GitZ [61]	44	·
	BinSim [62]	1062	·
	BinSequence [34]	(1,718)	·
	IMF-sim [31]	1,140	·
	CACompare [35]	72	·
	ASE17 [63]	55	·
2018	BinArm [17]	·	2,6
	SANER18 [64]	7	
	BinGo-E [18]	(5,145)	
	WSB [65]	(173)	
	BinMatch [66]	80	
	MASES18 [25]	47	
	Zeek [67]	(20,680)	
	FirmUp [14]	·	2,0
	αDiff [19]	(69,989)	2
	VulSeeker [13]	(10,512)	4,6
2019	InnerEye [24]	(844)	·
	Asm2Vec [20]	68	·
	SAFE [21]	(5001)	·
	BAR19i [26]	(804)	·
	BAR19ii [29]	(11244)	·
	FuncNet [68]	(180)	·
2020	DeepBinDiff [27]	2114	·
	ImOpt [69]	18	·
	ACCESS20 [52]	12,000	·
	Patchecko [28]	2,108	3
	BinKit ★	243,128	·

Insufficient benchmarks
(86% < 10,000 binaries)
(98% < 4 architectures)
➔ Hard to evaluate useful features

*Revisiting Binary Code S

BCSA Dataset in Previous Literature

Year	Tool [Paper]	Source*		Architecture*								Optimization*					Compiler†*													Extra				Info.		
		Binaries	Firmware	x86	x64	arm	aarch64	mips	mips64	mipseb	mips64eb	O0	O1	O2	O3	Os	GCC 3	GCC 4	GCC 5	GCC 6	GCC 7	GCC 8	Clang 3	Clang 4	Clang 5	Clang 6	Clang 7	etc.	Total #	Noinline	PIE	LTO	Obfus.	Code	Dataset	IDA
2014	TEDEM [10]	14	·	○	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	○
	Tracy [56]	(115)	·	·	○	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	○	·	○
	CoP [7]	(214)	·	·	·	·	·	·	·	·	·	○	○	○	○	○	·	1	·	·	·	·	·	·	·	·	·	1	2	·	·	·	○	·	·	○
	LoPD [8]	48	·	○	·	·	·	·	·	·	·	○	○	○	○	○	·	1	·	·	·	·	·	·	·	·	·	1	2	·	·	○	○	·	·	·
	BLEX [30]	1,140	·	·	○	·	·	·	·	·	·	○	○	○	○	·	·	1	·	·	·	·	1	·	·	·	·	1	3	·	·	·	·	·	·	○
	BinClone [57]	90	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	○	·	○
2015	Multi-k-MH [22]	60	6	○	·	○	·	○	·	·	·	○	○	○	○	·	·	2	·	·	·	·	1	·	·	·	·	·	3	·	·	·	·	·	·	○
2016	discovRE [11]	593	3	○	·	○	·	○	·	·	·	○	○	○	○	○	·	1	·	·	·	·	1	·	·	·	·	2	4	○	·	·	·	○	◑	○
	Genius [23]	(7,848)	8,128	○	·	○	·	○	·	·	·	○	○	○	○	·	·	2	·	·	·	·	1	·	·	·	·	·	3	·	·	·	·	·	·	○
	Esh [58]	(833)	·	·	○	·	·	·	·	·	·	·	·	·	·	·	·	3	·	·	·	·	2	·	·	·	·	2	7	·	·	·	·	○	○	○
	BinGo [15]	(5,143)	·	○	○	○	·	·	·	·	·	○	○	○	○	·	·	3	·	·	·	·	1	·	·	·	·	1	5	·	·	·	·	·	·	○
	MockingBird [33]	80	·	○	·	○	·	○	·	·	·	○	·	○	○	·	·	1	·	·	·	·	1	·	·	·	·	·	2	·	·	·	·	·	·	○
	Kam1n0 [32]	96	·	○	○	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	○	·	○
	BinDNN [59]	2,064	·	○	○	○	·	·	·	·	·	○	○	○	○	·	·	1	·	·	·	·	·	·	·	·	·	1	2	·	·	·	·	·	·	○
2017	BinSign [60]	(31)	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	2	2	·	·	·	○	·	·	○
	Xmatch [16]	72	1	○	·	·	·	○	·	·	·	·	·	·	·	·	·	2	·	·	·	·	1	·	·	·	·	·	3	○	·	·	·	·	·	○
	Gemini [12]	18,269	8,128	○	·	○	·	○	·	·	·	○	○	○	○	·	·	·	1	·	·	·	·	·	·	·	·	·	1	·	·	·	·	·	·	○
	GitZ [61]	44	·	·	○	·	○	·	·	·	·	○	○	○	○	○	·	3	·	·	·	·	2	1	·	·	·	·	6	·	·	·	·	·	·	·
	BinSim [62]	1062	·	○	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	○	·	·	○
	BinSequence [34]	(1,718)	·	·	○	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	○
	IMF-sim [31]	1,140	·	·	○	·	·	·	·	·	·	○	·	○	○	·	·	1	·	·	·	·	1	·	·	·	·	1	3	·	·	·	○	·	·	·
	CACompare [35]	72	·	○	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	·	·	○
	ASE17 [63]	55	·	○	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	·	·	·
2018	BinArm [17]	·	2,628	·	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	·	·	○
	SANER18 [64]	7	·	○	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	·	·	○
	BinGo-E [18]	(5,145)	·	○	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	·	·	○
	WSB [65]	(173)	·	·	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	·	·	·
	BinMatch [66]	80	·	○	·	·	·	·	·	·	·	○	·	○	○	·	·	1	·	·	·	·	1	·	·	·	·	·	2	·	·	·	○	·	·	○
	MASES18 [25]	47	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	○	·	·	·
	Zeek [67]	(20,680)	·	○	·	○	○	·	·	·	·	○	○	○	○	○	·	3	·	·	·	·	4	1	·	·	·	2	10	·	·	·	·	·	·	·
	FirmUp [14]	·	2,000	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	○
	αDiff [19]	(69,989)	2	○	○	○	·	·	·	·	·	○	○	○	○	·	·	2	1	·	·	·	2	·	·	·	·	·	5	·	·	·	·	◑	◑	○
	VulSeeker [13]	(10,512)	4,643	○	○	○	○	○	○	·	·	○	○	○	○	·	·	1	1	·	·	·	·	·	·	·	·	·	2	·	·	·	·	○	·	○
2019	InnerEye [24]	(844)	·	·	○	○	·	·	·	·	·	·	○	○	○	·	·	·	·	·	·	·	·	·	·	1	·	·	1	·	·	·	·	◑	◑	·
	Asm2Vec [20]	68	·	·	○	·	·	·	·	·	·	○	○	○	○	·	·	1	1	·	·	·	2	·	·	·	·	·	4	·	·	·	○	○	·	○
	SAFE [21]	(5001)	·	·	○	○	·	·	·	·	·	○	○	○	○	·	1	3	1	1	1	·	2	1	1	1	·	·	12	·	·	·	·	○	◑	○
	BAR19i [26]	(804)	·	·	○	○	·	·	·	·	·	·	○	○	○	·	·	·	·	·	·	·	·	·	·	1	·	·	1	·	·	·	·	○	·	·
	BAR19ii [29]	(11244)	·	○	○	○	·	·	·	·	·	○	○	○	○	·	1	3	1	·	·	·	2	1	1	·	·	2	11	·	·	·	·	·	·	·
	FuncNet [68]	(180)	·	○	·	○	·	○	·	·	·	○	○	○	○	○	·	·	·	1	·	·	·	·	·	·	·	·	1	·	·	·	·	·	·	○
2020	DeepBinDiff [27]	2114	·	·	·	·	·	·	·	·	·	○	○	○	○	·	·	1	·	·	·	·	·	·	·	·	·	·	1	·	·	·	·	○	○	○
	ImOpt [69]	18	·	·	○	·	·	·	·	·	·	○	·	○	○	·	·	·	1	·	·	·	·	·	·	·	·	·	1	·	·	·	○	·	·	·
	ACCESS20 [52]	12,000	·	○	○	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·
	Patchecko [28]	[illegible]	[illegible]	○	○	○	○	[illegible]	[illegible]	·	·	○	○	○	○	○	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	·	·	·	·	·	·	·	·	·	·	·	·	○
	BinKit ★	243,128	·	○	○	○	○	○	○	○	○	○	○	○	○	○	·	1	1	1	1	1	·	1	1	1	1	·	9	○	○	○	○	○	○	○

A few focused on IoT vulnerability Analysis

*Revisiting Binary Code S

Problems of Existing Studies

- In IoT devices, vulnerabilities can exist in
  - Libraries or utility binaries — **What BCSA studies have focused on**
  - Custom binaries (mostly, CGI binaries) — **None of BCSA studies targeted**

- Existing studies focus on **only libraries or utility binaries**
  - Open-source packages (e.g., OpenSSL, bash, vsftpd, ...)
  - Easy to generate training dataset

- **None** has analyzed **custom binaries** (e.g., CGI binaries)
  - No available dataset (or vulnerability details)
  - Not enough samples

Problems of Existing Studies

- **No** available open-source tools
  - Among 43 BCSA studies, 10 released their source code
  - Among these 10 tools,
    - **Only 2 supports x86, ARM, MIPS** (i.e., Gemini, **VulSeeker**)
  - Most IoT devices are based on **ARM/MIPS**

- Limitations of Gemini and VulSeeker
  - Do not have full source code
  - Based on complex machine learning ➔ Hard to interpret/understand the results
  - **How about performance?**

Motivating Example: CVE-2015-1791

- VulSeeker released partial results without full source code
  - Target firmware: Tomato Cisco M10v2 (router)
  - Target vulnerability: *ssl3_get_new_session_ticket* in *libssl.so*
  - Race condition causes double free (DoS)
- Approach
  - Compile vulnerable OpenSSL package (v1.0.1f) with 48 compiler options
  - Query each of the 48 functions in the target firmware
  - Average the similarity scores for all functions
- Result
  - VulSeeker found the vulnerability at **Rank 21**

Motivating Example: CVE-2015-1791

- ❖ VulSeeker released partial results without full source code
  - Target firmware: Tomato Cisco M10v2 (router)
  - Target vulnerability: *ssl3_get_new_session_ticket* in *libssl.so*
  - Race condition causes double free (DoS)
- ❖ Approach
  - Compile vulnerable OpenSSL package (v1.0.1f) with 48 compiler options
  - Query each of the 48 functions in the target firmware
  - Average the similarity scores for all functions
- ❖ Result
  - VulSeeker found the vulnerability at **Rank 21**

→ **Enough?**

Our Approach

- Fundamental problems of existing BCSA studies
  - No available dataset ➔ Establish a baseline benchmark (BinKit)
  - Heavy use of machine learning ➔ Develop a simple & interpretable model (TikNib)
  - Heavy use of semantic features ➔ Investigate pre-semantic features

- Problems of BCSA-based IoT vulnerability analysis
  - No analysis on custom binaries ➔ Establish ground truth dataset (FirmKit)
  - No available tool & Not enough studies ➔ Empirically analyze firmware images

Our Approach

- ❖ Fundamental problems of existing BCSA studies
  - – No available dataset ➔ Establish a baseline benchmark (BinKit)
  - – Heavy use of machine learning ➔ Develop a simple & interpretable model (TikNib)
  - – Heavy use of semantic features ➔ Investigate pre-semantic features

- ❖ Problems of BCSA-based IoT vulnerability analysis
  - – No analysis on custom binaries ➔ Establish ground truth dataset (FirmKit)
  - – No available tool & Not enough studies ➔ Empirically analyze firmware images

Building a Comprehensive Benchmark (BinKit)

- ❖ Compile GNU software packages
- ❖ Build ground truth by leveraging source file names and line numbers

Category	Previous Options	Our Options (Count)
Architecture	98% tested 4	x86, arm ,mips, mipseb for 32, 64 bits (4x2=8)
Compiler	95% tested 5	GCC: v4~v8 (5) Clang: v4~v7 (4)
Optimization	16% tested all opti-levels	O0, O1, O2, O3, Os (5)
Noinline	5% tested	Include (1)
PIE	0% tested	Include (1)
Link Time Optimization	2% tested	Include (1)
Obfuscation	26% tested	Obfuscator-LLVM (4)

Building a Comprehensive Benchmark (BinKit)

- ❖ Compile GNU software packages
- ❖ Build ground truth by leveraging source file names and line numbers

Category	Previous Options	Our Options (Count)
Architecture	98% tested 4	x86, arm ,mips, mipseb for 32, 64 bits (4x2=8)
Compiler	95% tested 5	GCC: v4~v8 (5) Clang: v4~v7 (4)
Optimization	16% tested all opti-levels	O0, O1, O2, O3, Os (5)
Noinline	5% tested	Include (1)
PIE	0% tested	Include (1)
Link Time Optimization	2% tested	Include (1)
Obfus[illegible]	[illegible]	[illegible]VM (4)

243,128 binaries for 36,256,322 functions

Analyze Pre-Semantic Features

- Justify semantic features (84%) and machine learning (90% after 2019)
  - ➔ **Cannot understand** the results
- Simple pre-semantic features
  - ➔ **Can understand** the results

Numeric Level	Feature Category	Example
CFG-Level (41 Features)	Graphic	Basic Blocks, Edges, …
	Computing	Arithmetic, Logic, …
	Data Manipulating	Copy, Addressing, …
	Control Transferring	Jmp, Conditional Jmp, …
	Category Mixing	Arithmetic + Shifting, …
CG-Level (6 Features)	Counting Unique	Callers, Callees, Imported Callees
	Including Duplicates	Incoming Calls, Outgoing Calls, Imported Calls

Design an Interpretable Model (TikNib)

- An intuitive model to easily understand the results

- Relative difference of feature ***f*** of function ***A*** and ***B***

$$rdiff(Af - Bf) = \frac{\left| Af - Bf \right|}{\left| max(Af, Bf) \right|}$$

Design an Interpretable Model (TikNib)

- An intuitive model to easily understand the results

- Relative difference of feature ***f*** of function ***A*** and ***B***

$$rdiff\left(Af - Bf\right) = \frac{\left|Af - Bf\right|}{\left|max\left(Af, Bf\right)\right|}$$

- Similarity score of function ***A*** and ***B***
  - Average of the relative differences of all features from ***f1*** to ***fN***

$$score(A, B) = \frac{rdiff\left(Af1, Bf1\right) + \cdots + rdiff\left(AfN, BfN\right)}{N}$$

  - Any other scoring metric can be integrated (e.g., Jaccard index)

Experiment Methodology

- ❖ There exist over 36M functions
  - ➔ We need a fast approach to obtain the tendency

- ❖ Utilize TP/TN pairs for each function **λ** (same as Gemini, VulSeeker)

Different Compiler Option

$(\lambda^{\text{Target}}, \lambda^{\text{True Positive}})$ $(\lambda^{\text{Target}}, \lambda^{\text{True Negative}})$

Same Compiler Option

- ❖ Greedily select features with ROC AUC
- ❖ 10-fold cross validation for each test

: Exist in all 10 tests			Opt Level			Compiler				Arch						vs. SizeOpt†			vs. Extra†			vs. Obfus.†				Bad‡	
		Rand.	O0 vs. O3	O2 vs. O3	Rand.	GCC v4 vs. GCC v8	Clang v4 vs. Clang v7	GCC vs. Clang	Rand.	x86 vs. ARM	x86 vs. MIPS	ARM vs. MIPS	32 vs. 64	LE vs. BE	Rand.	O0 vs. Os	O1 vs. Os	O3 vs. Os	PIE	NoInline	LTO	BCF	FLA	SUB	All	Norm.	Norm. vs. Obfus.†
Feature	CFG Avg. # of edges	0.33	0.26	0.42	0.34	0.44	0.46	0.37	0.41	0.43	0.37	0.37	0.43	0.47	0.41	0.34	0.42	0.36	0.45	0.44	0.40	0.26	0.34	0.47	0.22	0.32	0.19
	CFG # of backedges	0.39	0.33	0.44	0.39	0.46	0.45	0.41	0.43	0.47	0.45	0.45	0.46	0.48	0.46	0.39	0.42	0.38	0.50	0.40	0.46	0.23	0.08	0.47	0.05	0.32	0.03
	CFG # of edges	0.47	0.37	0.63	0.48	0.66	0.69	0.52	0.60	0.65	0.57	0.57	0.65	0.72	0.61	0.46	0.59	0.52	0.71	0.61	0.64	0.25	0.23	0.72	0.10	0.42	0.06
	CFG # of loops	0.40	0.34	0.44	0.40	0.46	0.46	0.41	0.44	0.47	0.45	0.45	0.46	0.47	0.46	0.40	0.42	0.39	0.50	0.40	0.46	0.23	0.13	0.47	0.10	0.33	0.08
	CFG # of basic blocks	0.41	0.36	0.59	0.46	0.62	0.65	0.48	0.56	0.44	0.55	0.39	0.62	0.69	0.52	0.43	0.56	0.50	0.67	0.57	0.60	0.26	0.23	0.67	0.10	0.26	0.00
	CG # of callees	0.50	0.43	0.59	0.52	0.62	0.63	0.52	0.58	0.63	0.54	0.55	0.57	0.64	0.57	0.50	0.59	0.53	0.60	0.57	0.57	0.60	0.59	0.64	0.56	0.46	0.47
	CG # of callers	0.45	0.40	0.54	0.48	0.59	0.58	0.49	0.54	0.53	0.41	0.45	0.54	0.60	0.50	0.50	0.57	0.50	0.56	0.54	0.52	0.54	0.54	0.58	0.52	0.37	0.41
	CG # of imported callees	0.44	0.39	0.54	0.47	0.58	0.56	0.48	0.52	0.59	0.44	0.45	0.55	0.55	0.50	0.46	0.53	0.48	0.55	0.50	0.52	0.53	0.53	0.56	0.50	0.36	0.43
	CG # of imported calls	0.45	0.38	0.56	0.48	0.60	0.58	0.48	0.54	0.61	0.45	0.47	0.57	0.57	0.52	0.46	0.54	0.48	0.57	0.52	0.54	0.49	0.54	0.59	0.46	0.37	0.40
	CG # of incoming calls	0.46	0.41	0.56	0.50	0.61	0.60	0.50	0.56	0.55	0.42	0.46	0.56	0.62	0.52	0.52	0.58	0.50	0.58	0.57	0.55	0.50	0.56	0.60	0.47	0.37	0.38
	CG # of outgoing calls	0.52	0.44	0.62	0.54	0.66	0.66	0.54	0.60	0.67	0.57	0.58	0.61	0.68	0.60	0.52	0.61	0.55	0.64	0.60	0.61	0.53	0.62	0.67	0.50	0.48	0.44
	Inst Avg. # of arith+shift	0.17	0.16	0.51	0.30	0.50	0.50	0.27	0.39	0.21	0.08	0.10	0.29	0.52	0.21	0.19	0.43	0.41	0.49	0.50	0.43	0.28	0.22	0.46	0.17	0.07	0.12
	Inst Avg. # of ctransfer	0.17	0.15	0.28	0.20	0.28	0.30	0.20	0.25	0.26	0.19	0.21	0.25	0.32	0.22	0.19	0.24	0.22	0.30	0.27	0.27	0.19	0.18	0.31	0.12	0.12	0.07
	Inst Avg. # of dtransfer+misc	0.17	0.08	0.44	0.22	0.42	0.46	0.27	0.36	0.30	0.15	0.17	0.31	0.49	0.25	0.09	0.36	0.35	0.45	0.44	0.36	0.28	0.26	0.45	0.17	0.12	0.08
	Inst Avg. # of dtransfer	0.19	0.10	0.45	0.23	0.43	0.48	0.28	0.37	0.30	0.20	0.22	0.32	0.53	0.28	0.11	0.38	0.36	0.46	0.46	0.38	0.28	0.27	0.47	0.18	0.10	0.08
	Inst Avg. # of instrs.	0.17	0.11	0.38	0.21	0.37	0.41	0.25	0.33	0.30	0.15	0.15	0.28	0.45	0.24	0.12	0.32	0.31	0.40	0.38	0.33	0.26	0.25	0.37	0.17	0.14	0.10
	Inst Avg. # of logic	0.24	0.23	0.51	0.34	0.45	0.56	0.27	0.39	0.22	0.21	0.25	0.40	0.54	0.31	0.25	0.40	0.42	0.56	0.48	0.54	0.23	0.22	0.40	0.12	0.24	0.05
	Inst # of arith+shift	0.24	0.26	0.59	0.40	0.59	0.60	0.38	0.49	0.28	0.11	0.12	0.37	0.61	0.27	0.28	0.52	0.48	0.58	0.56	0.53	0.26	0.41	0.55	0.14	0.13	0.02
	Inst # of ctransfer	0.42	0.35	0.56	0.43	0.57	0.62	0.43	0.51	0.57	0.51	0.54	0.56	0.64	0.54	0.38	0.51	0.46	0.62	0.54	0.57	0.25	0.23	0.64	0.10	0.32	0.03
	Inst # of dtransfer+misc	0.27	0.15	0.58	0.30	0.58	0.60	0.43	0.51	0.46	0.26	0.29	0.46	0.63	0.38	0.11	0.52	0.47	0.60	0.57	0.51	0.28	0.22	0.59	0.09	0.25	0.01
	Inst # of arith	0.24	0.26	0.59	0.39	0.59	0.60	0.38	0.49	0.27	0.12	0.13	0.38	0.61	0.27	0.28	0.52	0.48	0.57	0.57	0.53	0.25	0.41	0.55	0.14	0.12	0.01
	Inst # of bit-manipulating	0.09	0.12	0.34	0.21	0.31	0.23	0.18	0.24	0.07	0.04	0.06	0.22	0.20	0.10	0.14	0.31	0.28	0.36	0.32	0.34	0.17	0.16	0.20	0.05	0.05	0.01
	Inst # of compare	0.47	0.42	0.62	0.53	0.67	0.68	0.55	0.62	0.39	0.58	0.32	0.61	0.72	0.55	0.54	0.61	0.52	0.71	0.60	0.64	0.23	0.25	0.68	0.09	0.40	0.06
	Inst # of cond ctransfer	0.53	0.42	0.62	0.54	0.67	0.70	0.58	0.63	0.67	0.65	0.65	0.68	0.72	0.66	0.55	0.62	0.51	0.72	0.60	0.65	0.23	0.23	0.71	0.10	0.42	0.06
	Inst # of dtransfer	0.28	0.16	0.58	0.31	0.58	0.60	0.43	0.51	0.45	0.29	0.35	0.46	0.63	0.41	0.13	0.53	0.48	0.60	0.57	0.52	0.31	0.22	0.59	0.09	0.23	0.03
	Inst # of float instrs.	0.09	0.09	0.27	0.16	0.16	0.28	0.12	0.17	0.09	0.10	0.08	0.17	0.32	0.11	0.09	0.22	0.23	0.27	0.23	0.17	0.24	0.12	0.30	0.08	0.00	0.00
	Inst # instrs.	0.31	0.21	0.57	0.34	0.58	0.60	0.45	0.52	0.52	0.29	0.30	0.48	0.62	0.42	0.18	0.52	0.47	0.60	0.55	0.52	0.26	0.22	0.56	0.08	0.30	0.02
	Inst # of misc	0.10	0.04	0.47	0.17	0.40	0.46	0.20	0.32	0.06	0.11	0.02	0.36	0.67	0.19	0.04	0.29	0.27	0.47	0.44	0.37	0.16	0.19	0.49	0.09	0.00	0.00
	Inst # of shift	0.22	0.23	0.42	0.30	0.42	0.39	0.27	0.34	0.22	0.20	0.19	0.31	0.54	0.26	0.24	0.35	0.34	0.48	0.37	0.41	0.40	0.34	0.42	0.31	0.19	0.23
Avg. TP-TN Gap		0.31	0.26	0.49	0.35	0.49	0.51	0.36	0.43	0.39	0.33	0.32	0.43	0.54	0.38	0.30	0.44	0.40	0.52	0.47	0.47	0.28	0.26	0.50	0.17	0.24	0.11
Avg. of Grey		0.43	0.34	0.42	0.48	0.57	0.59	0.49	0.50	0.55	0.44	0.49	0.57	0.61	0.52	0.46	0.56	0.45	0.55	0.57	0.51	0.44	0.47	0.55	0.41	0.33	0.33
ROC AUC		0.94	0.90	0.97	0.95	0.99	1.00	0.96	0.98	0.99	0.98	0.98	0.99	1.00	0.98	0.96	0.98	0.95	1.00	0.97	0.98	0.98	0.98	1.00	0.95	0.91	0.91
Std. of ROC AUC		0.01	0.01	0.01	0.00	0.00	0.00	0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.01	0.00	0.01	0.00	0.01	0.00	0.00	0.00	0.00	0.00	0.02	0.01

Examples of Findings

Architecture has a small impact	ROC AUC
x86 vs ARM	0.99
x86 vs MIPS	0.98
ARM vs MIPS	0.98
32-bit vs 64-bit (Bits)	0.99
Little vs Big (Endian)	1.00

Optimization is largely influential	
O0 vs O3	0.90
O2 vs O3	0.97

Compiler version has almost no effect	
GCCv4 vs GCCv8	0.99
Clangv4 vs Clangv7	1.00

GCC and Clang have diverse characteristics	ROC AUC
GCC vs Clang	0.96

Extra Options are less effective	
vs PIE	1.00
vs Noinline	0.97
vs LTO	0.98

O-LLVM is insufficient for evaluation	
vs Bogus Control Flow	0.98
vs Control Flow Flattening	0.98
vs Instruction Substitution	1.00
vs All Three Options	0.95

Examples of Findings

Architecture has a small impact	ROC AUC
x86 vs ARM	0.99
x86 vs MIPS	0.98
ARM vs MIPS	0.98
32-bit vs 64-bit (Bits)	0.99
Little vs Big (Endian)	1.00

Optimization is largely influential	
O0 vs O3	0.90
O2 vs O3	0.97

Compiler version has almost no effect	
GCCv4 vs GCCv8	0.99
Clangv4 vs Clangv7	1.00

GCC and Clang have diverse characteristics	ROC AUC
GCC vs Clang	0.96

Extra Options are less effective	
vs PIE	1.00
vs Noinline	0.97
vs LTO	0.98

O-LLVM is insufficient for evaluation	
vs Bogus Control Flow	0.98
vs Control Flow Flattening	0.98
vs Instruction Substitution	1.00
vs All Three Options	0.95

Examples of Findings

Architecture has a small impact	ROC AUC
x86 vs ARM	0.99
x86 vs MIPS	0.98
ARM vs MIPS	0.98
32-bit vs 64-bit (Bits)	0.99
Little vs Big (Endian)	1.00

Optimization is largely influential	
O0 vs O3	0.90
O2 vs O3	0.97

Compiler version has almost no effect	
GCCv4 vs GCCv8	0.99
Clangv4 vs Clangv7	1.00

GCC and Clang have diverse characteristics	ROC AUC
GCC vs Clang	0.96

Extra Options are less effective	
vs PIE	1.00
vs Noinline	0.97
vs LTO	0.98

O-LLVM is insufficient for evaluation	
vs Bogus Control Flow	0.98
vs Control Flow Flattening	0.98
vs Instruction Substitution	1.00
vs All Three Options	0.95

Examples of Findings

Architecture has a small impact	ROC AUC
x86 vs ARM	0.99
x86 vs MIPS	0.98
ARM vs MIPS	0.98
32-bit vs 64-bit (Bits)	0.99
Little vs Big (Endian)	1.00

Optimization is largely influential	
O0 vs O3	0.90
O2 vs O3	0.97

Compiler version has almost no effect	
GCCv4 vs GCCv8	0.99
Clangv4 vs Clangv7	1.00

GCC and Clang have diverse characteristics	ROC AUC
GCC vs Clang	0.96

Extra Options are less effective	
vs PIE	1.00
vs Noinline	0.97
vs LTO	0.98

O-LLVM is insufficient for evaluation	
vs Bogus Control Flow	0.98
vs Control Flow Flattening	0.98
vs Instruction Substitution	1.00
vs All Three Options	0.95

Examples of Findings

Architecture has a small impact	ROC AUC
x86 vs ARM	0.99
x86 vs MIPS	0.98
ARM vs MIPS	0.98
32-bit vs 64-bit (Bits)	0.99
Little vs Big (Endian)	1.00

Optimization is largely influential	
O0 vs O3	0.90
O2 vs O3	0.97

Compiler version has almost no effect	
GCCv4 vs GCCv8	0.99
Clangv4 vs Clangv7	1.00

GCC and Clang have diverse characteristics	ROC AUC
GCC vs Clang	0.96

Extra Options are less effective	
vs PIE	1.00
vs Noinline	0.97
vs LTO	0.98

O-LLVM is insufficient for evaluation	
vs Bogus Control Flow	0.98
vs Control Flow Flattening	0.98
vs Instruction Substitution	1.00
vs All Three Options	0.95

Pre-semantic Features Are Effective!

- VulSeeker (ASE'18)
  - State of the art using numeric features
  - Use both pre-semantic and semantic features with deep neural network

- vs VulSeeker

ROC AUC

Dataset	Packages	Arch	Compilers	VulSeeker	Ours
ASE1	2	3	1	0.99	**0.9661**
ASE2	**5**	3	1	-	**0.9610**
ASE3	5	**6**	2	0.8849	**0.9616**
ASE4	5	**8**	**9**	-	**0.9450**

Larger Dataset

Case Study: Heartbleed

- Utilize TikNib to analyze Heartbleed (CVE-2014-0160)
  - Genius, Gemini, Multi-kMH, DiscovRE, SAFE, ...

- Target: *tls1_process_heartbeat, dtls1_process_heartbeat*
  - OpenSSL v1.0.1f (vulnerable), v1.0.1u (patched)
  - Query *tls1_process_heartbeat*

- Average the similarity score rank in each option

Source option to Target option	All to All	ARM to ARM	ARM to MIPS	ARM to x86	MIPS to MIPS	MIPS to ARM	MIPS to x86	x86 to x86	x86 to ARM	x86 to MIPS	O2 to O3	O3 to O2	GCC to Clang	GCC v4 to GCC v8	GCC v8 to GCC v4	Clang v4 to Clang v7	Clang v7 to Clang v4
# of Option Pairs	552	56	64	64	56	64	64	56	64	64	144	144	144	36	36	36	36
Rank (tls, vuln)*	1.19	1.14	1.66	1	1	1.62	1	1	1.25	1	1.18	1.19	1	1.44	1.06	1	1
Precision@1 (tls, vuln)*	0.89	0.86	0.66	1	1	0.75	1	1	0.75	1	0.9	0.89	1	0.78	0.94	1	1
Rank (dtls, vuln)†	4.54	9.82	11.81	3.06	2	4.72	2	2.07	1.75	3.62	4.5	4.38	2.72	3.11	5.06	3.61	3.33
Rank (tls, patched)‡	29.16	12.12	57.69	3.56	3.82	51.62	43.94	4.29	6.38	70.59	27.5	28.96	27.68	32.89	40.89	20.22	22.67
Rank (dtls, patched)‡	76.47	46.95	145.75	7.25	8.21	128	128.94	9.57	11.94	181.03	73.04	75.41	87.31	66.28	87.33	68.44	78

*tls: tls1_process_heartbeat *dtls: dtls1_process_heartbeat

Case Study: Heartbleed

- Utilize TikNib to analyze Heartbleed (CVE-2014-0160)
  - Genius, Gemini, Multi-kMH, DiscovRE, SAFE, ...

- Target: *tls1_process_heartbeat, dtls1_process_heartbeat*
  - OpenSSL v1.0.1f (vulnerable), v1.0.1u (patched)
  - Query *tls1_process_heartbeat*

- Average the similarity score rank in each option

Source option to Target option	All to All	ARM to ARM	ARM to MIPS	ARM to x86	MIPS to MIPS	MIPS to ARM	MIPS to x86	x86 to x86	x86 to ARM	x86 to MIPS	O2 to O3	O3 to O2	GCC to Clang	GCC v4 to GCC v8	GCC v8 to GCC v4	Clang v4 to Clang v7	Clang v7 to Clang v4
# of Option Pairs	552	56	64	64	56	64	64	56	64	64	144	144	144	36	36	36	36
Rank (tls, vuln)*	1.19	1.14	1.66	1	1	1.62	1	1	1.25	1	1.18	1.19	1	1.44	1.06	1	1
Precision@1 (tls, vuln)*	0.89	0.86	0.66	1	1	0.75	1	1	0.75	1	0.9	0.89	1	0.78	0.94	1	1
Rank (dtls, vuln)†	4.54	9.82	11.81	3.06	2	4.72	2	2.07	1.75	3.62	4.5	4.38	2.72	3.11	5.06	3.61	3.33
Rank (tls, patched)‡	29.16	12.12	57.69	3.56	3.82	51.62	43.94	4.29	6.38	70.59	27.5	28.96	27.68	32.89	40.89	20.22	22.67
Rank (dtls, patched)‡	76.47	46.95	145.75	7.25	8.21	128	128.94	9.57	11.94	181.03	73.04	75.41	87.31	66.28	87.33	68.44	78

*tls: tls1_process_heartbeat *dtls: dtls1_process_heartbeat

Case Study: Heartbleed

- Utilize TikNib to analyze Heartbleed (CVE-2014-0160)
  - Genius, Gemini, Multi-kMH, DiscovRE, SAFE, ...
- Target: *tls1_process_heartbeat, dtls1_process_heartbeat*
  - OpenSSL v1.0.1f (vulnerable), v1.0.1u (patched)
  - Query *tls1_process_heartbeat*
- Average the similarity score rank in each option

Source option to Target option	All to All	ARM to ARM	ARM to MIPS	ARM to x86	MIPS to MIPS	MIPS to ARM	MIPS to x86	x86 to x86	x86 to ARM	x86 to MIPS	O2 to O3	O3 to O2	GCC to Clang	GCC v4 to GCC v8	GCC v8 to GCC v4	Clang v4 to Clang v7	Clang v7 to Clang v4
# of Option Pairs	552	56	64	64	56	64	64	56	64	64	144	144	144	36	36	36	36
Rank (tls, vuln)*	1.19	1.14	1.66	1	1	1.62	1	1	1.25	1	1.18	1.19	1	1.44	1.06	1	1
Precision@1 (tls, vuln)*	0.89	0.86	0.66	1	1	0.75	1	1	0.75	1	0.9	0.89	1	0.78	0.94	1	1
Rank (dtls, vuln)†	4.54	9.82	11.81	3.06	2	4.72	2	2.07	1.75	3.62	4.5	4.38	2.72	3.11	5.06	3.61	3.33
Rank (tls, patched)‡	29.16	12.12	57.69	3.56	3.82	51.62	43.94	4.29	6.38	70.59	27.5	28.96	27.68	32.89	40.89	20.22	22.67
Rank (dtls, patched)‡	76.47	46.95	145.75	7.25	8.21	128	128.94	9.57	11.94	181.03	73.04	75.41	87.31	66.28	87.33	68.44	78

*tls: tls1_process_heartbeat *dtls: dtls1_process_heartbeat

Case Study: Heartbleed

- ❖ Utilize TikNib to analyze Heartbleed (CVE-2014-0160)
  - – Genius, Gemini, Multi-kMH, DiscovRE, SAFE, ...

- ❖ Target: *tls1_process_heartbeat, dtls1_process_heartbeat*
  - – OpenSSL v1.0.1f (vulnerable), v1.0.1u (patched)
  - – Query *tls1_process_heartbeat*

- ❖ Average the similarity score rank in each option

Source option to Target option	All to All	ARM to ARM	ARM to MIPS	ARM to x86	MIPS to MIPS	MIPS to ARM	MIPS to x86	x86 to x86	x86 to ARM	x86 to MIPS	O2 to O3	O3 to O2	GCC to Clang	GCC v4 to GCC v8	GCC v8 to GCC v4	Clang v4 to Clang v7	Clang v7 to Clang v4
# of Option Pairs	552	56	64	64	56	64	64	56	64	64	144	144	144	36	36	36	36
Rank (tls, vuln)*	1.19	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	1.06	1	1
Precision@1 (tls, vuln)*	0.89	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	0.94	1	1
Rank (dtls, vuln)†	4.54	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	5.06	3.61	3.33
Rank (tls, patched)‡	29.16	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	40.89	20.22	22.67
Rank (dtls, patched)‡	76.47	[illegible]	[illegible]	[illegible]	[illegible]	128	128.94	[illegible]	11.94	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	87.33	68.44	78

Pre-semantic features with a simple/interpretable model is effective!

*tls: tls1_process_heartbeat   *dtls: dtls1_process_heartbeat

Our Approach

- ❖ Fundamental problems of existing BCSA studies
  - – No available dataset ➔ Establish a baseline benchmark (BinKit)
  - – Heavy use of machine learning ➔ Develop a simple & interpretable model (TikNib)
  - – Heavy use of semantic features ➔ Investigate pre-semantic features
  - ➔ **Proper feature engineering is important**
  - ➔ **Simple model with presemantic features can show promising performance**

- ❖ Problems of BCSA-based IoT vulnerability analysis
  - – No analysis on custom binaries ➔ Establish ground truth dataset (FirmKit)
  - – No available tool & Not enough studies ➔ Empirically analyze firmware images

Our Approach

- ❖ Fundamental problems of existing BCSA studies
  - No available dataset ➔ Establish a baseline benchmark (BinKit)
  - Heavy use of machine learning ➔ Develop a simple & interpretable model (TikNib)
  - Heavy use of semantic features ➔ Investigate pre-semantic features
  - ➔ **Proper feature engineering is important**
  - ➔ **Simple model with presemantic features can show promising performance**

- ❖ Problems of BCSA-based IoT vulnerability analysis
  - No analysis on custom binaries ➔ Establish ground truth dataset (FirmKit)
  - No available tool & Not enough studies ➔ Empirically analyze firmware images

Building Ground Truth Dataset

- ❖ Vulnerabilities from FirmAE
  - – 1,124 firmware images of IoT routers and cameras

- ❖ Target dataset
  - – 1,124 firmware images — 52,086,995 functions
  - – **267 vulnerable functions**
    - ▪ 98 command injection
    - ▪ 162 information leak
    - ▪ 7 buffer overflow
    - ➔ 19 unique vulnerabilities

➔ **Manually marked vulnerable function addresses**

Analyzing Linux-based IoT Devices

- Randomly select one sample for each unique vulnerability
- Query it for each firmware image (1,124 images, 52M funcs)

	Original TikNib	
Top-k	**# of Total Vulns**	**Percent**
1	141 / 267	52.81%
5	167 / 267	62.55%
10	182 / 267	68.16%
50	196 / 267	73.41%
100	**196 / 267**	**73.41%**

Analyzing Linux-based IoT Devices

- Randomly select one sample for each unique vulnerability
- Query it for each firmware image (1,124 images, 52M funcs)

	Original TikNib	
Top-k	**# of Total Vulns**	**Percent**
1	141 / 267	52.81%
5	167 / 267	62.55%
10	182 / 267	68.16%
50	196 / 267	73.41%

How to increase the performance?

Failure Case Study - CVE-2015-2051

- Architecture specific issues
  - ARM -> ARM: detected at Rank 1.75 on average
  - ARM -> MIPS: detected at Rank over 1000

- Arm produces a wrapper function for a library function call (.PLT)
  - ➔ # of callees, # of imported callees, cfg_size, ...

```
v9 = 0;
memset(s, 0, sizeof(s));
v6 = getenv("HTTP_AUTHORIZATION");
haystack = getenv("HTTP_SOAPACTION");
s1 = getenv("REQUEST_METHOD");
```

ARM (Wrapper Function Call)

```
memset(v26, 0, sizeof(v26));
v4 = getenv("HTTP_AUTHORIZATION");
v5 = getenv("HTTP_SOAPACTION");
v6 = getenv("REQUEST_METHOD");
```

MIPS (External Function Call)

Failure Case Study - CVE-2017-5521

```
sub_155D4(a1, "answer1");
sub_155D4(a1, "answer2");
v4 = (const char *)acosNvramConfig_get(&unk_87F8E);
v5 = (const char *)acosNvramConfig_get(&unk_87F9F);
if ( !strcasecmp(v26, v4) && !strcasecmp(v25, v5) )
{
  if ( (int)time(0) > 0x47302D4D )
  {
    time(&timer);
    localtime_r(&timer, &tp);
    v7 = (const char *)sub_6A460("language");
    if ( !strcmp("Japanese", v7) )
    {
      tm_year = tp.tm_year;
      v13 = sub_15FE8("year");
      v14 = tm_year + 1900;
      if ( tp.tm_mon )
      {
        switch ( tp.tm_mon )
        {
          case 1:
            v15 = "month_feb";
            break;
          case 2:
            v15 = "month_mar";
            break;
          case 3:
            v15 = "month_apr";
            break;
          case 4:
            v15 = "month_may";
            break;
          case 5:
            v15 = "month_jun";
```

Different version has an additional check routine

```
websGetVar(a1, "answer1", v10);
websGetVar(a1, "answer2", v11);
v4 = (const char *)acosNvramConfig_get("password_answer1");
v5 = (const char *)acosNvramConfig_get("password_answer2");
if ( !strcasecmp(v10, v4) && !strcasecmp(v11, v5) )
{
  if ( time(0) > 0x47302D4D )
  {
    time(&v9);
    v7 = localtime(&v9);
    v8 = asctime(v7);
    strcpy(v12, v8);
    acosNvramConfig_set("timestamp_of_last_recovery", v12);
  }
  else
  {
    acosNvramConfig_set("timestamp_of_last_recovery", "");
  }
  acosNvramConfig_save();
  sendPage2Client("MNU_accessPassword_recovered.htm", a2);
}
else
{
  sendPage2Client("MNU_accessUnauthorized_checkAnswerAgain.htm", a2);
}
return 0;
}
```

No such routine exists

Failure Case Study - CVE-2017-5521

```
sub_155D4(a1, "answer1");
sub_155D4(a1, "answer2");
v4 = (const char *)acosNvramConfig_get(&unk_87F8E);
v5 = (const char *)acosNvramConfig_get(&unk_87F9F);
if ( !strcasecmp(v26, v4) && !strcasecmp(v25, v5) )
{
  if ( (int)time(0) > 0x47302D4D )
  {
    time(&timer);
    localtime_r(&timer, &tp);
    v7 = (const char *)sub_6A460("language");
    if ( !strcmp("Japanese", v7) )
    {
      tm_year = tp.tm_year;
      v13 = sub_15FE8("year");
      v14 = tm_year + 1900;
      if ( tp.tm_mon )
      {
        switch ( tp.tm_mon )
        {
          case 1:
            v15 = "month_feb";
            break;
          case 2:
            v15 = "month_mar";
            break;
          case 
            v15
            brea
          case 
            v15
            brea
          case 
            v15
```

```
websGetVar(a1, "answer1", v10);
websGetVar(a1, "answer2", v11);
v4 = (const char *)acosNvramConfig_get("password_answer1");
v5 = (const char *)acosNvramConfig_get("password_answer2");
if ( !strcasecmp(v10, v4) && !strcasecmp(v11, v5) )
{
  if ( time(0) > 0x47302D4D )
  {
    time(&v9);
    v7 = localtime(&v9);
    v8 = asctime(v7);
    strcpy(v12, v8);
    acosNvramConfig_set("timestamp_of_last_recovery", v12);
  }
  else
  {
    acosNvramConfig_set("timestamp_of_last_recovery", "");
  }
  acosNvramConfig_save();
  sendPage2Client("MNU_accessPassword_recovered.htm", a2);
}
else
{
  sendPage2Client("MNU_accessUnauthorized_checkAnswerAgain.htm", a2);
```

No such routine exists

Need features robust against different architectures and versions

Leverage Heuristic Features

- IoT binaries often contain function names
  - Use caller and callee names (i.e., internal and library function names)

- Data strings often contain useful information
  - CGI binaries parse URLs with hard-coded strings
    - "HTTP", "POST", "answer1", "password", ...
  - Use words in a string

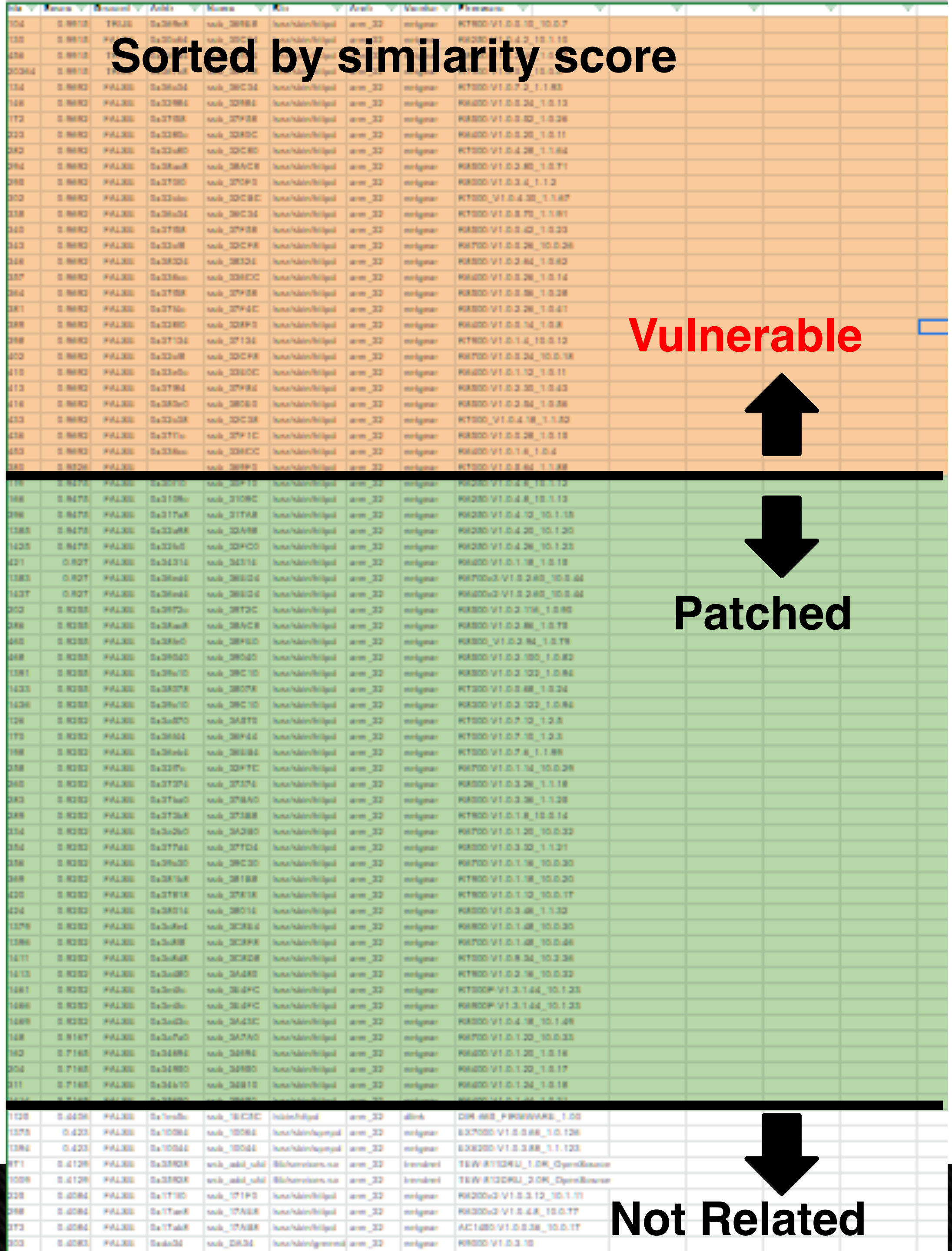
- Compare each word with Jaccard index
  - The score is merged with TikNib

$$Jaccard(A, B) = \frac{|A \cap B|}{|A \cup B|}$$

Final Results of Linux-based IoT Devices

- Randomly select one sample for each unique vulnerability
- Query it for each firmware image (1,124 images, 52M funcs)

	Original TikNib		TikNib (+Heuristic Features)	
Top-k	# of Total Vulns	Percent	# of Total Vulns	Percent
1	141 / 267	52.81%	**263 / 267**	**98.50%**
5	167 / 267	62.55%	263 / 267	98.50%
10	182 / 267	68.16%	266 / 267	99.63%
50	196 / 267	73.41%	266 / 267	99.63%
100	196 / 267	73.41%	267 / 267	100%



Vulnerability†	Range	# of Funcs	Vuln†	Netgear	D-Link	TRENDnet	Belkin	Asus	ZyXEL	Linksys	arm	mips	mipseb	Binary
				Vendor							Arch‡			
CVE-2016-6277 (104)	0.95–1.00	29 (3)	V	✓	·	·	·	·	·	·	✓	·	·	/usr/sbin/httpd
	0.5–0.95	40 (-)	P	✓	·	·	·	·	·	·	✓	·	·	/usr/sbin/httpd
CVE-2015-2051 (619)	0.81–1.00	5 (4)	V	·	✓	·	·	·	·	·	✓	·	·	/htdocs/cgibin
	0.68–0.73	25 (-)	P	·	✓	·	·	·	·	·	✓	·	·	/htdocs/cgibin
	0.58–0.75	6 (5)	V	·	✓	✓	·	·	·	·	·	✓	·	/htdocs/cgibin
	0.53–0.59	3 (-)	P	·	✓	·	·	·	·	·	·	✓	·	/htdocs/cgibin
	0.68	1 (-)	P	·	✓	·	·	·	·	·	·	·	✓	/htdocs/cgibin
	0.58–0.69	15 (14)	V	·	✓	·	·	·	·	·	·	·	✓	/htdocs/cgibin
	0.53	9 (-)	P	·	✓	·	·	·	·	·	·	·	✓	/htdocs/cgibin
	0.49–0.53	17 (-)	P	·	✓	·	·	·	·	·	·	✓	✓	/usr/sbin/upnpkits
CVE-2017-7240 (118)	0.95–1.00	3 (3)	V	·	·	·	✓	·	·	·	·	✓	·	/usr/sbin/httpd
	0.54–0.83	6 (-)	N	·	·	·	✓	·	·	·	·	✓	·	/usr/sbin/httpd
	0.50–0.53	23 (-)	N	·	·	·	·	✓	✓	✓	·	✓	✓	/usr/sbin/httpd
CVE-2018-10106 (2)	0.99–1.00	45 (42)	V	·	✓	✓	·	·	·	·	·	✓	✓	/htdocs/cgibin
	0.48–0.86	42 (41)	V	·	✓	·	·	·	·	·	✓	✓	✓	/htdocs/cgibin
	0.55–0.84	5 (-)	P	·	✓	·	·	·	·	·	✓	✓	✓	/htdocs/cgibin
CVE-2014-2962 (510)	0.96–1.00	2 (2)	V	·	·	·	✓	·	·	·	·	·	✓	/usr/www/cgi-bin/webproc
	0.66–0.86	13 (0)	V*	✓	·	✓	✓	·	·	·	·	·	✓	/usr/www/cgi-bin/webproc
	0.53	1 (-)	P	✓	·	·	·	·	·	·	·	✓	·	/usr/www/cgi-bin/webproc
CVE-2020-15893 (2)	0.86–1.00	43 (40)	V	·	✓	✓	·	·	·	·	·	✓	✓	/htdocs/cgibin
	0.96	1 (-)	P	·	·	✓	·	·	·	·	·	✓	✓	/htdocs/cgibin
	0.85	17 (12)	V	·	✓	·	·	·	·	·	·	✓	✓	/usr/sbin/upnpkits
	0.82	7 (7)	V	·	✓	·	·	·	·	·	✓	·	·	/htdocs/cgibin
	0.74–0.81	42 (-)	P	·	✓	·	·	·	·	·	✓	✓	✓	/htdocs/cgibin
	0.52	1 (1)	V	·	✓	·	·	·	·	·	·	·	✓	/htdocs/cgibin
CVE-2016-11021 (804)	0.97–1.00	11 (1)	V	·	✓	·	·	·	·	·	·	✓	·	/bin/alphapd
	0.97	2 (2)	V	·	✓	·	·	·	·	·	·	✓	·	/bin/goahead
	0.67–0.75	21 (-)	P	·	✓	✓	·	·	·	·	·	✓	·	/bin/alphapd
	0.60–0.67	9 (0)	V	·	✓	·	·	·	·	·	·	✓	·	/bin/alphapd
	0.59	1 (-)	P	·	·	✓	·	·	·	·	·	✓	·	/bin/alphapd
	0.50–0.59	18 (-)	N	·	✓	·	·	·	·	·	·	✓	·	/bin/alphapd
CVE-2017-6077 (186)	0.85–1.00	2 (2)	V	✓	·	·	·	·	·	·	·	·	✓	/usr/sbin/httpd
	0.5–0.85	1 (0)	V	✓	·	·	·	·	·	·	·	·	✓	/usr/sbin/httpd
CVE-2012-2765 (37)	0.72–1.00	7 (3)	V	·	·	·	✓	·	·	·	·	✓	·	/usr/sbin/httpd
	0.66	1 (-)	P	·	·	·	✓	·	·	·	·	✓	·	/usr/sbin/httpd
	0.58	2 (0)	V	·	·	·	✓	·	·	·	·	✓	·	/usr/sbin/httpd
	0.53	1 (-)	N	·	·	·	·	·	·	✓	·	✓	·	/usr/sbin/httpd
Linksys (53)	0.72–1.00	10 (1)	V	·	·	·	·	·	·	✓	·	✓	·	/usr/sbin/httpd
	0.53–0.64	7 (-)	P	·	·	·	·	·	·	✓	·	✓	·	/usr/sbin/httpd
CVE-2017-5521 (99, Stage 1)	0.98–1.00	40 (26)	V	✓	·	·	·	·	·	·	✓	·	·	/usr/sbin/httpd
	0.74–0.83	73 (-)	P	✓	·	·	·	·	·	·	✓	·	·	/usr/sbin/httpd
	0.79	2 (0)	V*	✓	·	·	·	·	·	·	✓	·	·	/usr/sbin/httpd
	0.51–0.52	11 (9)	V	✓	·	·	·	·	·	·	·	✓	·	/usr/sbin/httpd
	0.51–0.59	171 (-)	U	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	22 different binaries
CVE-2017-5521 (99, Stage2)	0.98–1.00	79 (26)	V	✓	·	·	·	·	·	·	✓	·	·	/usr/sbin/httpd
	0.76–0.92	36 (-)	P	✓	·	·	·	·	·	·	✓	·	·	/usr/sbin/httpd
	0.74–0.78	24 (6)	V	✓	·	·	·	·	·	·	·	✓	·	/usr/sbin/httpd
	0.68–0.73	9 (-)	P	✓	·	·	·	·	·	·	·	✓	·	/usr/sbin/httpd
	0.51–0.53	3 (-)	N	✓	·	·	·	·	·	·	·	·	✓	/usr/sbin/httpd
	0.51–0.51	1 (-)	P	✓	·	·	·	·	·	·	·	✓	·	/usr/sbin/upnpd
	0.51–0.51	14 (3)	V	✓	·	·	·	·	·	·	·	✓	·	/usr/sbin/upnpd

Case Study of CVE-2016-6277

- Command injection in CGI parsing (NETGEAR)

```
cmd = cmd.replace(" ", "$IFS")
path = "/cgi-bin/;{}".format(cmd)

self.http_request(
    method="GET",
    path=path
)
```

- Simple patch based on a block list

```
if ( !strchr(uri, ';') && !strchr(uri, '`') && !strchr(uri, '$') && !strstr(uri, "..") )
{
```

Range	# of Samples	Is Vulnerable?	Vendor	Arch
0.95 ~ 1.00	29 (3 Ground Truths)	**Vulnerable**	Netgear	ARM
0.5 ~ 0.95	40	Patched	Netgear	ARM

- **BCSA can distinguish vulnerabilities from the patched ones**

Case Study of CVE-2017-7240

- ❖ Directory traversal in CGI parsing
- ❖ DD-WRT's httpd
  - – Designed to accept only allowed file types
  - – Customized images allow all file types

```
response = self.http_request(
    method="GET",
    path="/etc/passwd"
)
```

Range	# of Samples	Is Vulnerable?	Vendors
0.95 ~ 1.00	3 (3 Ground Truths)	**Vulnerable**	Belkin
0.54 ~ 0.83	6	Not Vulnerable	Belkin
0.50 ~ 0.53	23	Not Vulnerable	Asus, ZyXEL, linksys

- ❖ The vulnerability resides in the data section, but BCSA found it

➔ **BCSA can detect diversities in compile environments**

*DD-WRT: Open source IoT firmware

Case Study of CVE-2018-10106

- Permission bypass with a newline (AUTHORIZED_GROUP)

Range	# of Samples	Is Vulnerable?	Vendor
0.99 ~ 1.00	45 (42 Ground Truths)	**Vulnerable**	**D-Link, TRENDnet**
0.48 ~ 0.86	42 (41 Ground Truths)	**Vulnerable**	D-Link
	5	Patched	D-Link

- Same vulnerability appears in new **versions** (D-Link)
  - CVE-2018-10106, CVE-2019-17506, CVE-2019-20213, CVE-2020-9376
- Same vulnerability appears in different **vendors** (TRENDnet, with score: 1.0)
  - CVE-2018-7034
- Same vulnerability appears in different **architectures** (MIPS, MIPSEB, ARM)
  - MIPS: 0.65~1, ARM: 0.5~0.6

Case Study of CVE-2014-2962

❖ Directory traversal in parsing a “getpage” parameter in CGI

Range	# of Samples	Is Vulnerable?	Vender
0.96 ~ 1.00	2 (2 Ground Truths)	**Vulnerable**	Belkin
0.66 ~ 0.86	13	**Potentially Vulnerable**	**Belkin, TRENDnet, Netgear**
0.53	1	Patched	Netgear

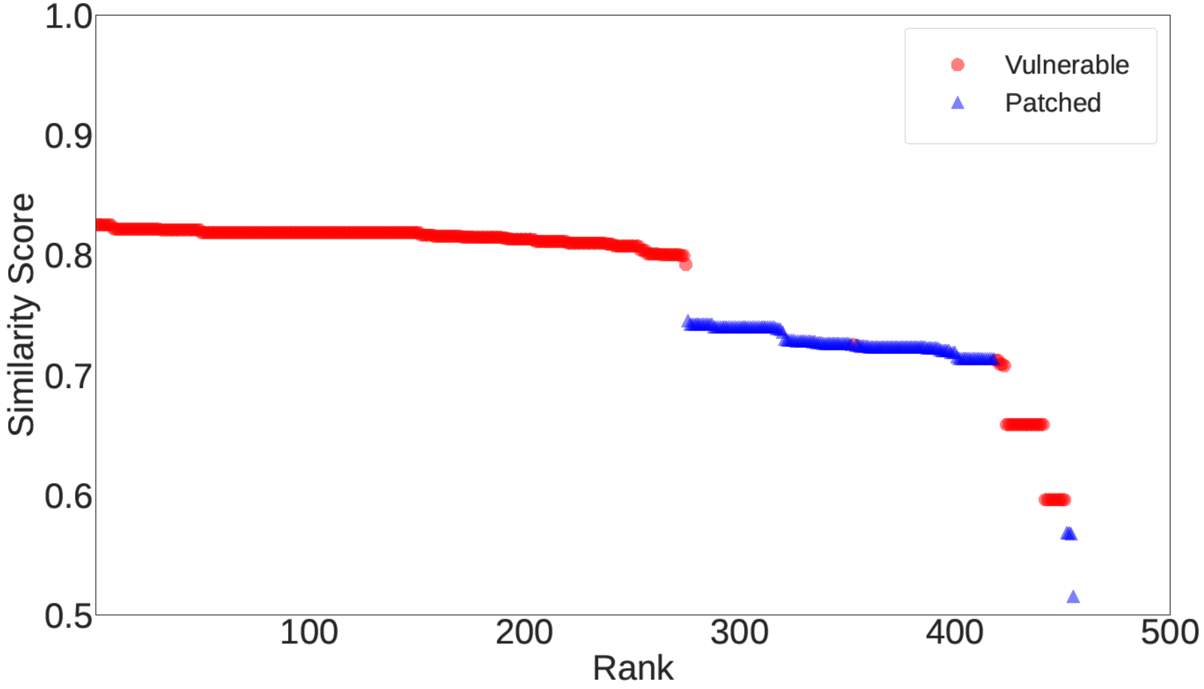
❖ Similar/same vulnerability has existed from 2006 in multiple vendors

– CVE-2006-2337 D-Link
– CVE-2006-5607 Inca
– CVE-2006-5536 D-Link
– CVE-2014-2962 Belkin
– CVE-2015-7250 Zte
– CVE-2017-15647 Fiberhome
– CVE-2017-8770 Twsz

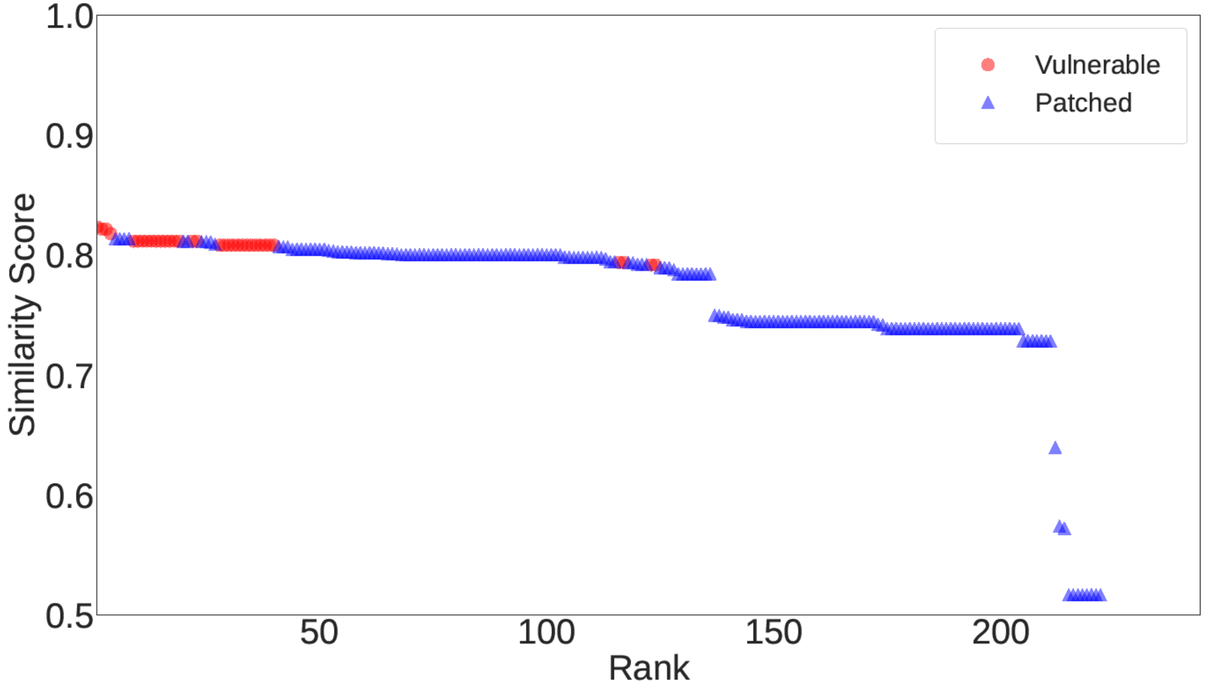
Case Study of OpenSSL Vulnerabilities

- Vulnerable functions are ranked higher than patched functions
  - Queried *OpenSSL v1.0.1f*

CVE-2015-1791 (309 of 455 are vulns)



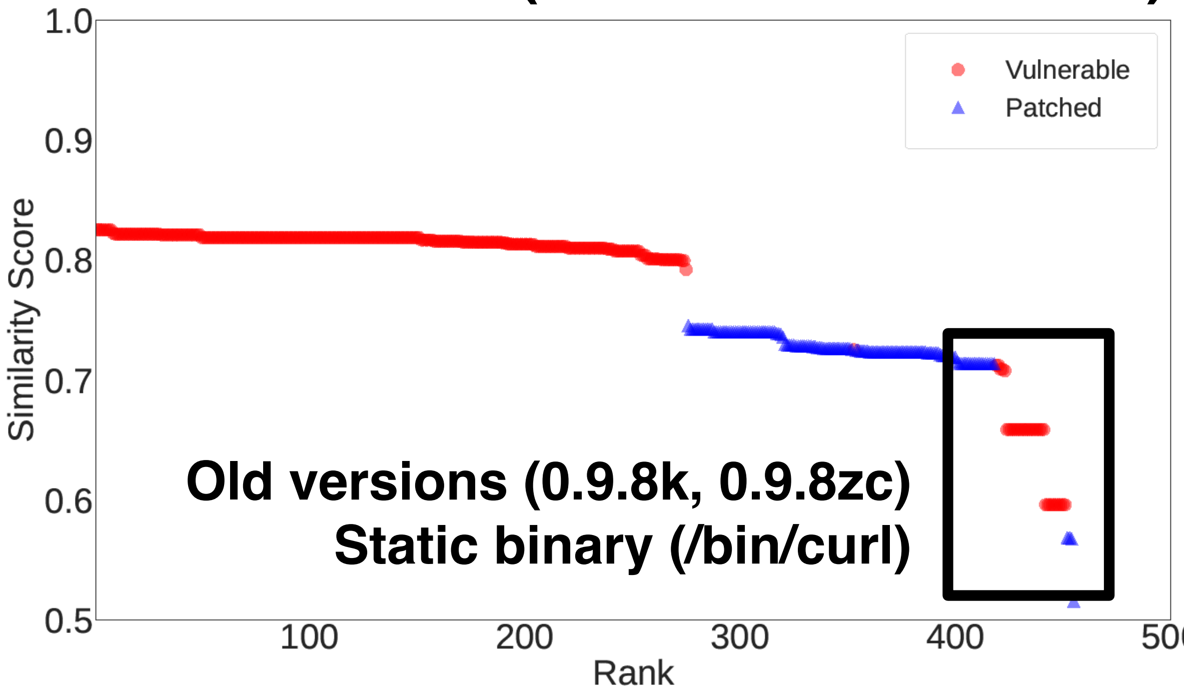
CVE-2014-0160 (34 of 222 are vulns)



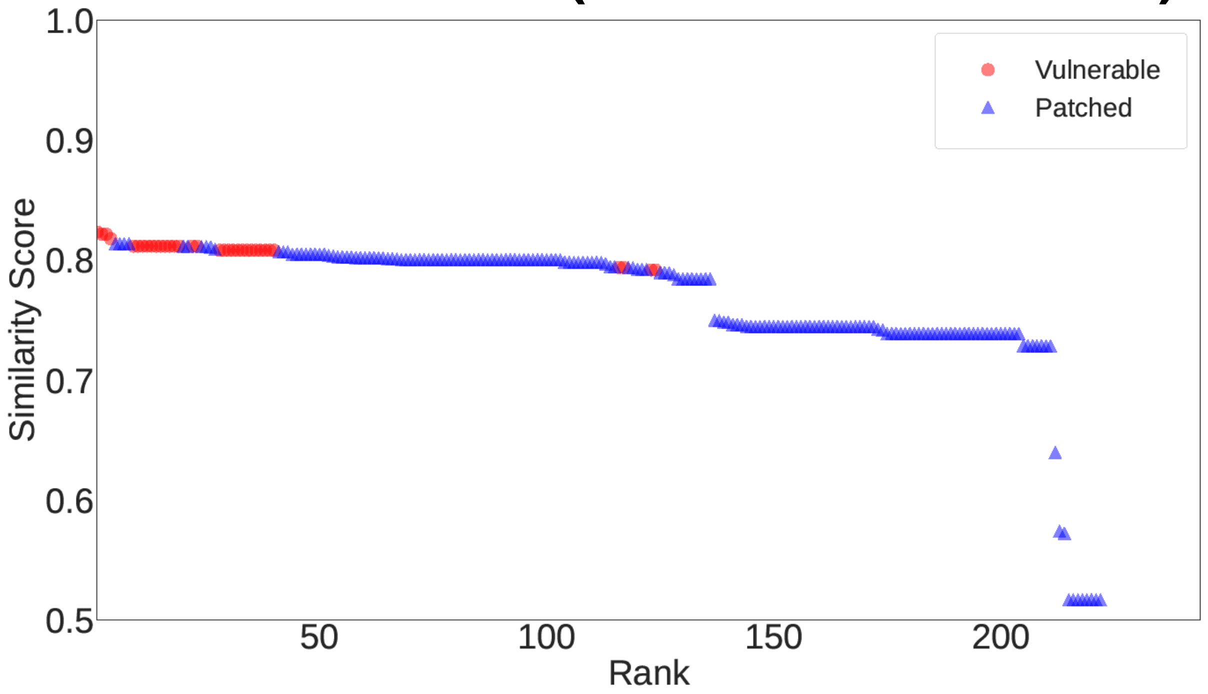
Case Study of OpenSSL Vulnerabilities

- Vulnerable functions are ranked higher than patched functions
  - Queried *OpenSSL v1.0.1f*

CVE-2015-1791 (309 of 455 are vulns)



CVE-2014-0160 (34 of 222 are vulns)



Comparison Results of CVE-2015-1791

- Top-k results of all functions in all firmware images (***NOT*** each image)
- Gemini and VulSeeker utilized 4643 firmware images (**unavailable**)
- TikNib utilized 1,124 firmware images (FirmAE)

	Gemini		VulSeeker	
Top-k	# of Funcs	%	# of Funcs	%
1	1	100%	1	100%
5	2	40%	3	60%
10	4	40%	6	60%
50	36	72%	41	82%
100	75	75%	83	83%

TikNib (O0-O3)		TikNib (O2-O3)		TikNib (+Heuristics)	
# of Funcs	%	# of Funcs	%	# of Funcs	%
1	**100%**	**1**	**100%**	**1**	**100%**
5	**100%**	**5**	**100%**	**5**	**100%**
9	**90%**	**10**	**100%**	**10**	**100%**
19	38%	**46**	**92%**	**50**	**100%**
50	50%	**82**	**82%**	**100**	**100%**

Comparison Results of CVE-2015-1791

- Top-k results of all functions in all firmware images (***NOT*** each image)
- Gemini and VulSeeker utilized 4643 firmware images (**unavailable**)
- TikNib utilized 1,124 firmware images (FirmAE)

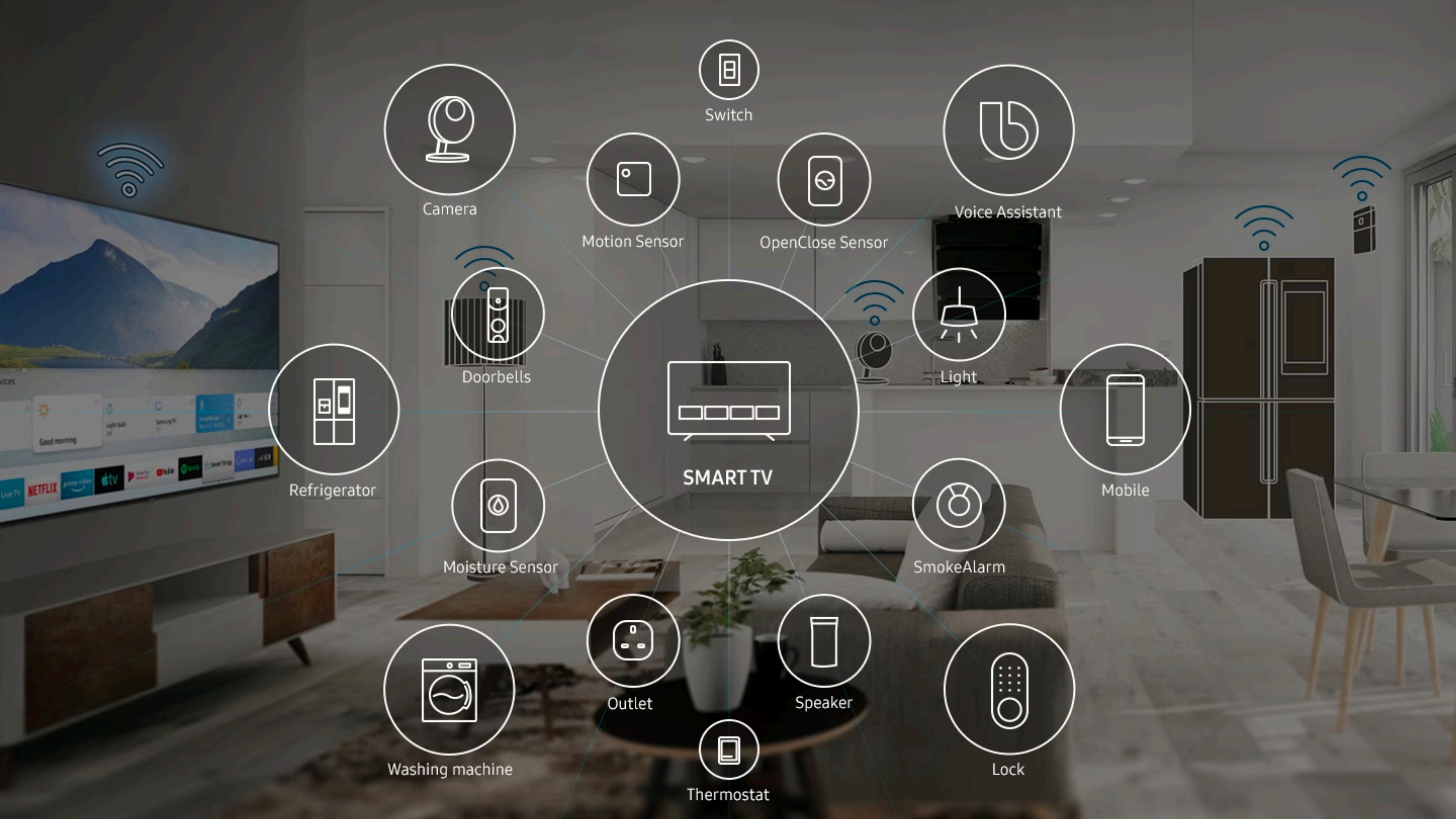
	Gemini		VulSeeker		TikNib (O0-O3)		TikNib (O2-O3)		TikNib (+Heuristics)	
Top-k	# of Funcs	%	# of Funcs	%	# of Funcs	%	# of Funcs	%	# of Funcs	%
1	1	100%	1	100%	**1**	**100%**	**1**	**100%**	**1**	**100%**
5	2	40%	3	60%	**5**	**100%**	**5**	**100%**	**5**	**100%**
10	4	40%	6	60%	**9**	**90%**	**10**	**100%**	**10**	**100%**
50	36	72%	41	82%	19	38%	**46**	**92%**	**50**	**100%**
100	75	75%	83	83%	50	50%	**82**	**82%**	**100**	**100%**

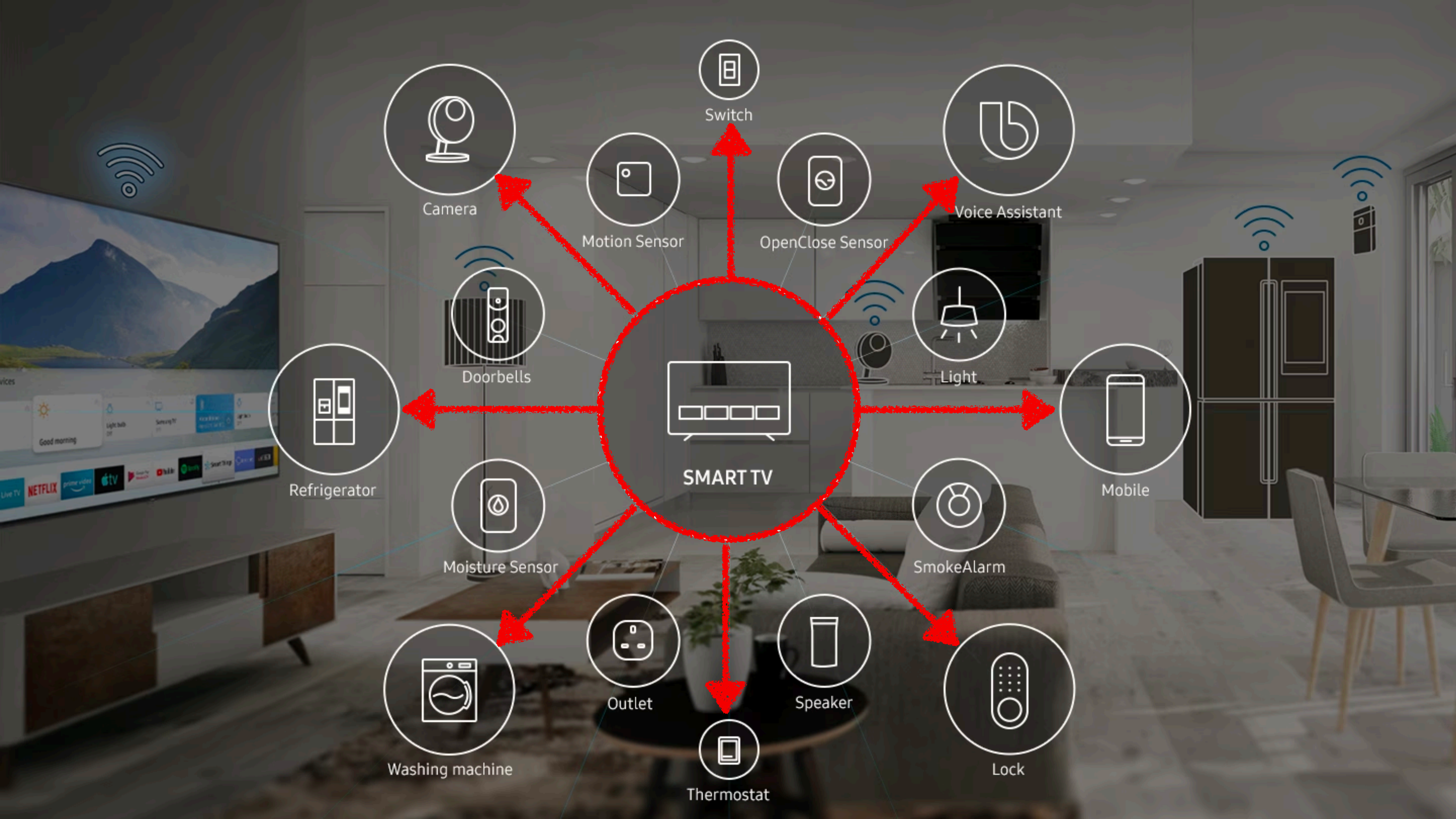
Firmware images are highly likely compiled with O2-O3

Limitation and Future Works

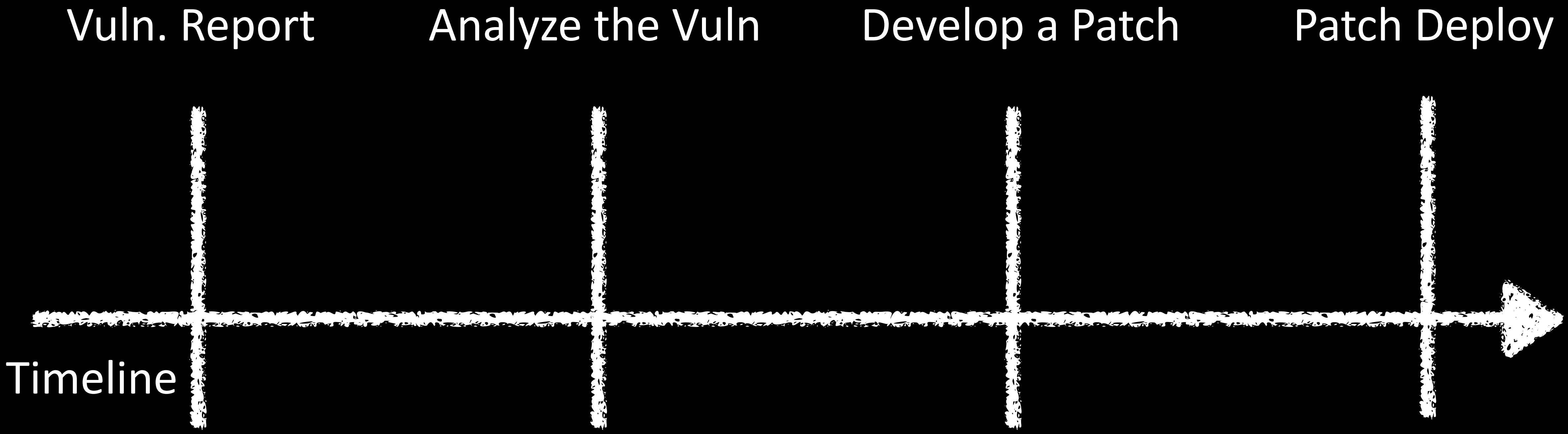
- Developing other effective features
  - Type recovery (NDSS’11, SIGPLAN’13, SEC’17, CCS’18, …)
    - Type-related features are effective
    - # of arguments, each argument type, function return type
    - All benchmark tests achieved ROC AUC close to **1.0**
  - Inter-procedural analysis
    - Optimization affects function in-lining
  - Inter-binary analysis
    - Handle static binaries

- Determining whether a detected function is indeed vulnerable
  - Function-level: e.g., leverage symbolic execution
  - Binary-level: e.g., emulate a target binary and check dynamically
  - Firmware-level: e.g., analyze vulnerabilities spread over multiple binaries
  - ➔ Leave as future work

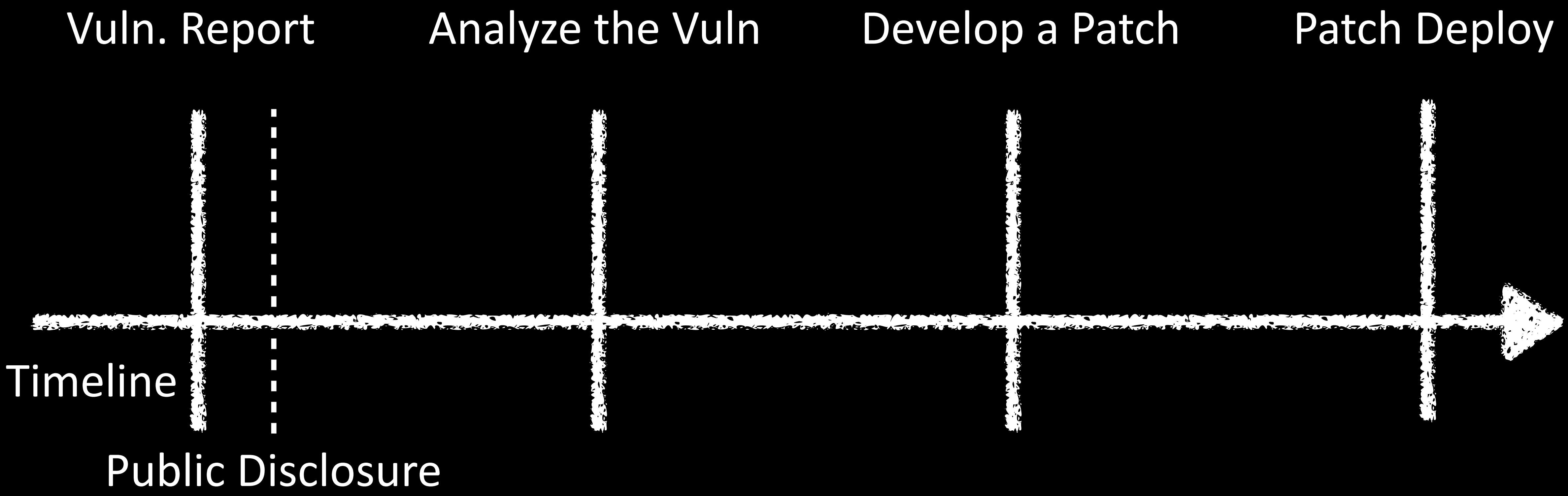

Switch
Camera
Voice Assistant
Motion Sensor
OpenClose Sensor
Doorbells
Light
Refrigerator
SMART TV
Mobile
Moisture Sensor
SmokeAlarm
Outlet
Speaker
Washing machine
Lock
Thermostat
Good morning
NETFLIX


Switch
Camera
Motion Sensor
OpenClose Sensor
Voice Assistant
Doorbells
Light
Refrigerator
SMART TV
Mobile
Moisture Sensor
SmokeAlarm
Outlet
Speaker
Washing machine
Lock
Thermostat

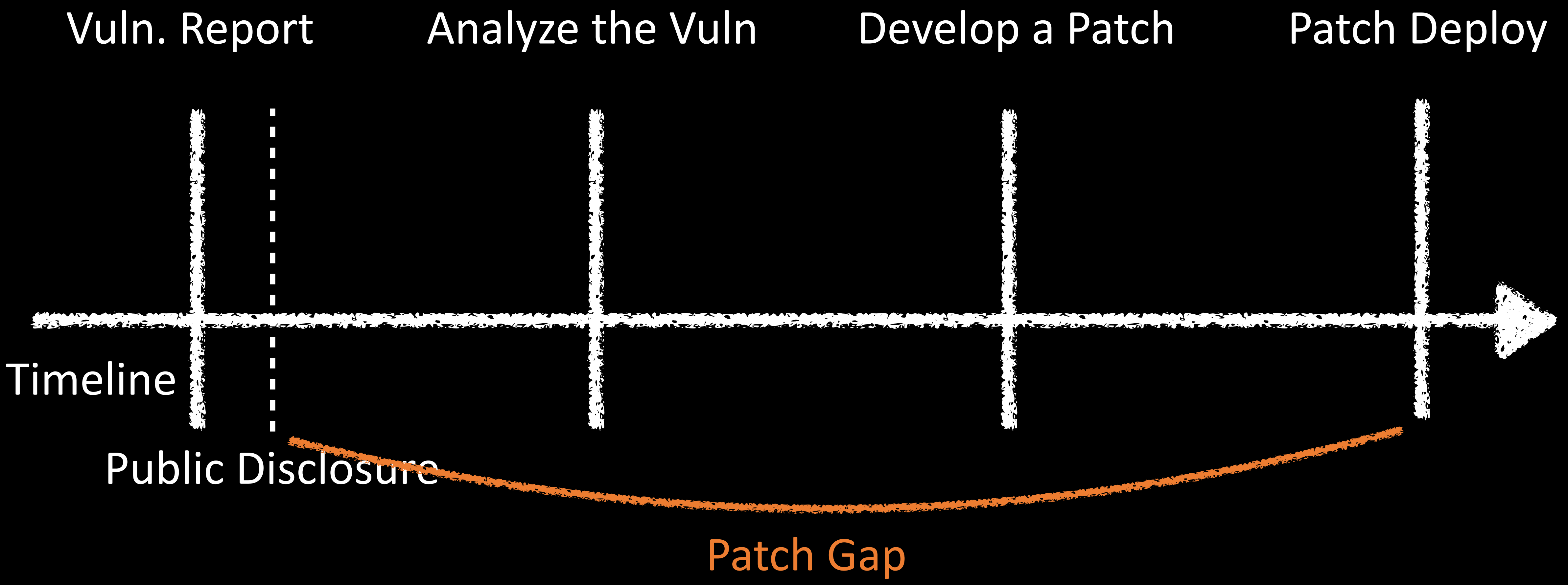
What Is Patch Gap?



What Is Patch Gap?



What Is Patch Gap?



Are Patch Gap Issues that Critical?

- Log4Shell (CVE-2021-44228)
    - Discovered in Dec. 2021
    - Oct. 2022: 72% of Organizations are still vulnerable (by Tenable)

- Citrix Application Delivery Controller and Gateway (CVE-2022-27510, CVE-2022-27518)
    - Discovered in Nov. 2022, respectively
    - Dec. 2022: 42% of Servers are actively exploited (by NSA)

- Arm Mali GPU driver (CVE-2022-22706)
    - Reported in Jun. 2022
    - Sep. 2022: Not patched at all (by Google Project Zero)

Difficulties in Addressing Patch Gap

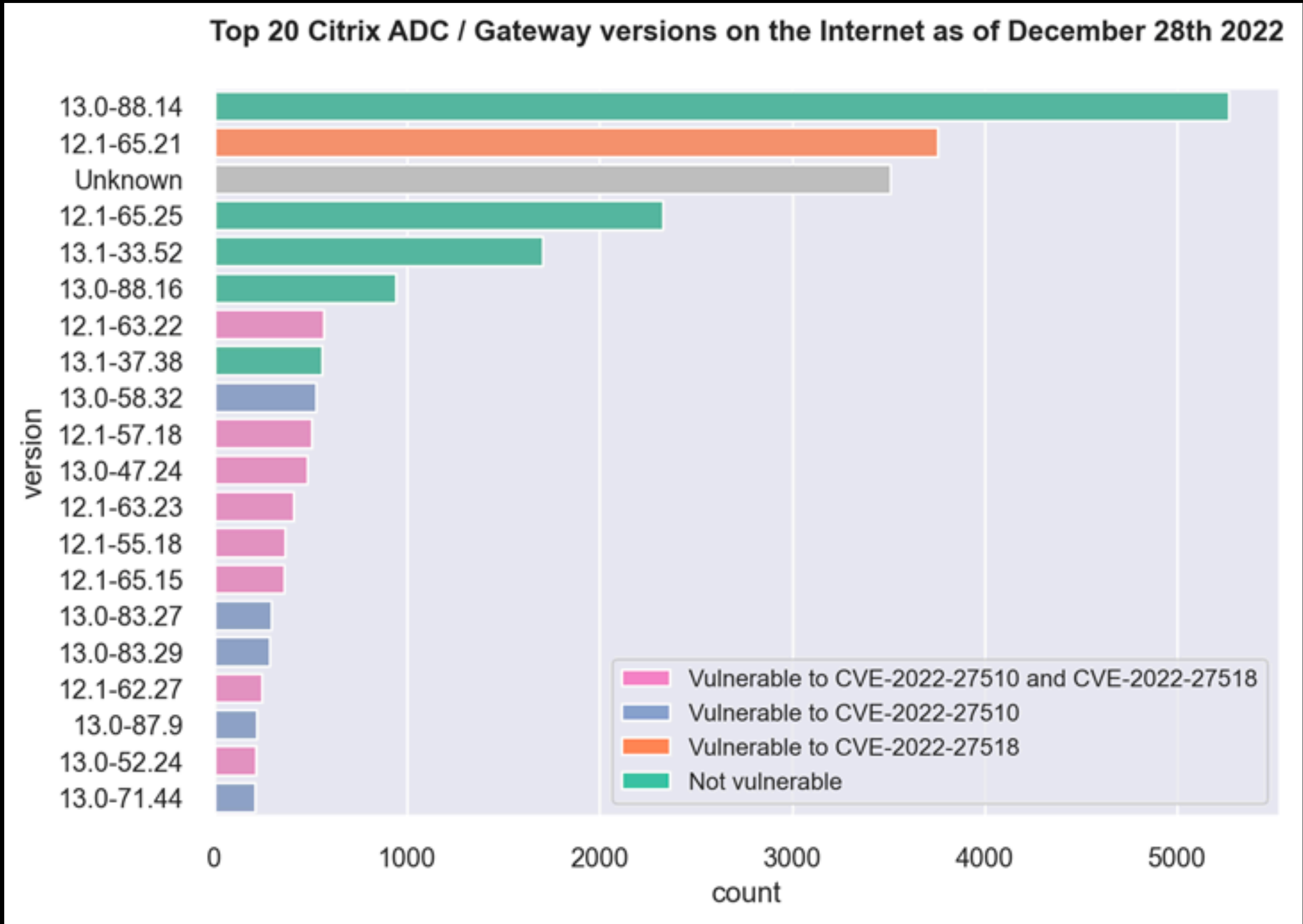
- Too complex software
  - Complex codebase (> 10M LoC, …)
  - Huge dependency of 3rd party libraries
  - …
- Too complex patch ecosystem
  - Example: patching a vulnerability in Galaxy S10?

Difficulties in Addressing Patch Gap

- Too complex software
  - Complex codebase (> 10M LoC, …)
  - Huge dependency of 3rd party libraries
  - …
- Too complex patch ecosystem
  - Example: patching a vulnerability in Galaxy S10?
  - > 1 billion users
  - > 280 carriers to update
  - > 30 device models
  - Takes > 6 months to deploy a patch (See BaseSpec, NDSS'21)

Difficulties in Addressing Patch Gap

- Citrix ADC/Gateway?

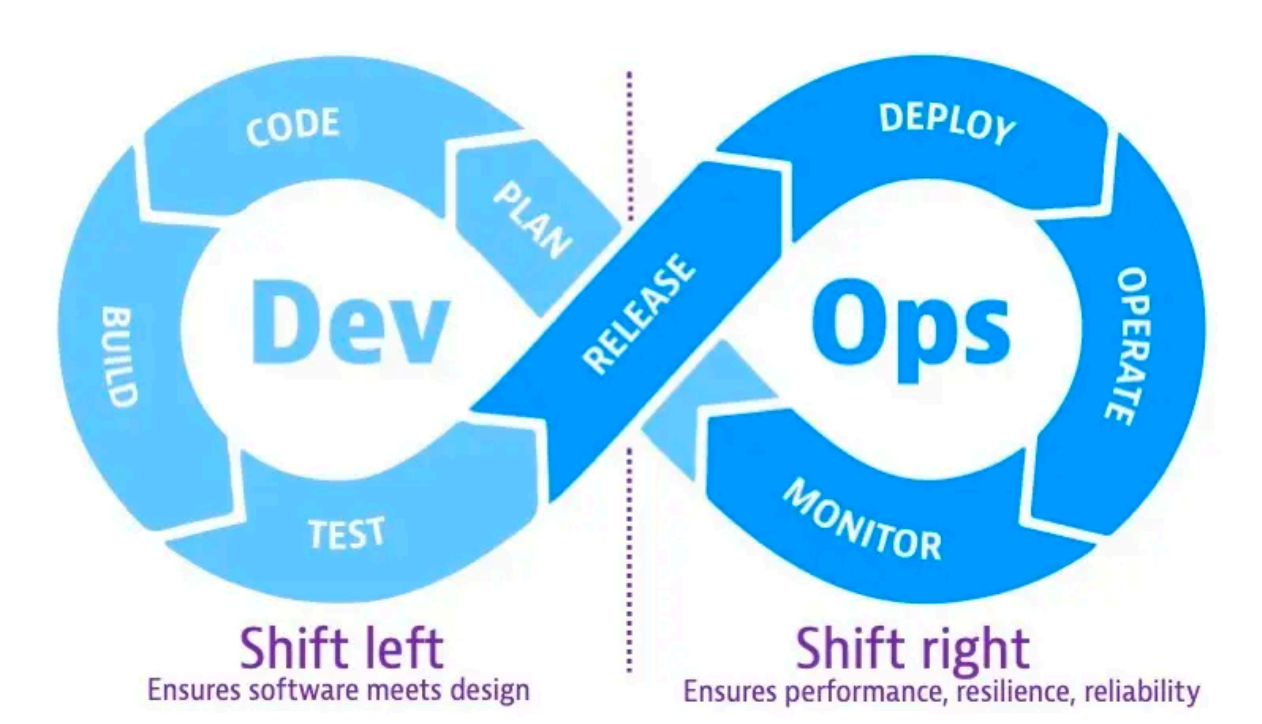


Behind Stories - IoT Routers Vuln Reporting

- D-Link
  - All vulnerabilities are patched by the vender
- ASUS
  - Reported on Apr 2019
  - Confirmed on Jan. 2020 (> 8 months)
- Belkin
  - Reported on 2019
  - No update until now (as of Oct. 2023)

Effort to Mitigate Patch Gap Issues

- Consider security from the design stage of software/system
- Apply source code analysis techniques (white-box)
  - Version check, simple pattern match, …
  - IDE-integrated plugins
  - …
- Fast patch deployment
  - Version/Model management
  - …
- Shift-left (DevSecOps)



Real Difficulties … (Academia vs Industry)

- What if you are a company owner,
  - Limited resource
    - how many workers
    - how much time
    - …
- Can you assess the risk of a security issue?
  - Probability?
  - Potential Impact?
  - How much loss?

Thank You!

dkay@kaist.ac.kr

BACKUP SLIDES

Decompilation?

```c
int ntz(unsigned x) {
    int n;
    int y = -x & (x-1);
    printf ("%x %x\n", x, y);
    n = 0;
    while(y != 0) {
        n = n + 1;
        y = y >> 1;
    }
    return n;
}

int main()
{
    printf ("%x\n", ntz(5));
    printf ("%x\n", ntz(10));
}
```

```
; Attributes: bp-based frame

; int __cdecl main(int argc, const char **argv, const char **envp)
public main
main proc near
push    rbp
mov     rbp, rsp
mov     edi, 5
call    ntz
mov     esi, eax
mov     edi, offset asc_40064B ; "%x\n"
mov     eax, 0
call    _printf
mov     edi, 0Ah
call    ntz
mov     esi, eax
mov     edi, offset asc_40064B ; "%x\n"
mov     eax, 0
call    _printf
mov     eax, 0
pop     rbp
retn
main endp
```

Variable Recovery

```c
int __cdecl main(int argc, const char **arg
{
  __int64 v3; // rsi@1
  __int64 v4; // rdx@1
  unsigned int v5; // eax@1

  v3 = (unsigned int)ntz(5LL, argv, envp);
  printf("%x\n", v3);
  v5 = ntz(10LL, v3, v4);
  printf("%x\n", v5);
  return 0;
}
```

Type Inference

Type Features Should Be Studied

- ❖ Function type does not change unless source code varies
  - – # of arguments
  - – Leverage Jaccard index for checking argument type, return type

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|}$$

- ❖ All benchmark tests achieved ROC AUC over **0.99**

- ❖ vs VulSeeker

ROC AUC

Dataset	Packages	Arch	Compilers	VulSeeker	Ours	Ours (Type)
ASE1	2	3	1	0.99	0.9727	**0.9924**
ASE2	**5**	3	1	-	0.9764	**0.9931**
ASE3	5	**6**	2	0.8849	0.9782	**0.9939**
ASE4	5	**8**	**9**	-	0.9584	**0.9841**

Larger Dataset

➔ **Features from type information is effective**
(NDSS’11, SIGPLAN’13, SEC’17, CCS’18, …)

Failure Case Analysis

- **Errors in IDA Pro (72% use IDA Pro)**
  - Cannot handle some registers in GCC and Clang
    - GCC: ‘gp’, Clang: ‘s0’, ‘v0’
  - incomplete CFGs
    - switch table, data in code section

- **Diversity of compiler backends**
  - Conditional instructions for ARM
    - GCC: MOVLE, MOVGT, Clang: MOV + JLE, MOV + JGT
  - Instruction pointer loading
    - GCC: call __x86.get_pc_thunk.bx, Clang: call $+5

- **Architecture-specific macros**
  - mul_add in OpenSSL

➔ **Need to consider these cases carefully!**

Analyzing Open-Source Vulnerabilities

- Two well-known OpenSSL vulnerabilities
  - CVE-2015-1791: *ssl3_get_new_session_ticket*
    - Genius, Gemini, VulSeeker
  - CVE-2014-0160: *tls1_process_heartbeat*
    - Genius, Gemini, Multi-kMH, DiscovRE, SAFE

- Approach
  - Compile OpenSSL v1.0.1f with combinations of compiler options
  - Search all compiled functions in each firmware image
  - Average the similarity score for each function in each firmware image

- Ground truth
  - Match a function name and version string
  - CVE-2015-1791: 309 of 455 are vulnerable
  - CVE 2014-0160: 34 of 222 are vulnerable

```
SSLv2 part of OpenSSL 1.0.1c 10 May 2012
SSLv3 part of OpenSSL 1.0.1c 10 May 2012
TLSv1 part of OpenSSL 1.0.1c 10 May 2012
DTLSv1 part of OpenSSL 1.0.1c 10 May 2012
```

Version strings in *libssl.so*