FirmUp: Precise Static Detection of Common Vulnerabilities in Firmware

Yaniv David, Nimrod Partush, Eran Yahav @ Technion

*The research leading to these results has received funding from the European Union's Seventh Framework Programme (FP7) under grant agreement no. 615688 - ERC-COG-PRIME and the Israel Ministry of Science and Technology, grant no. 3-9779.

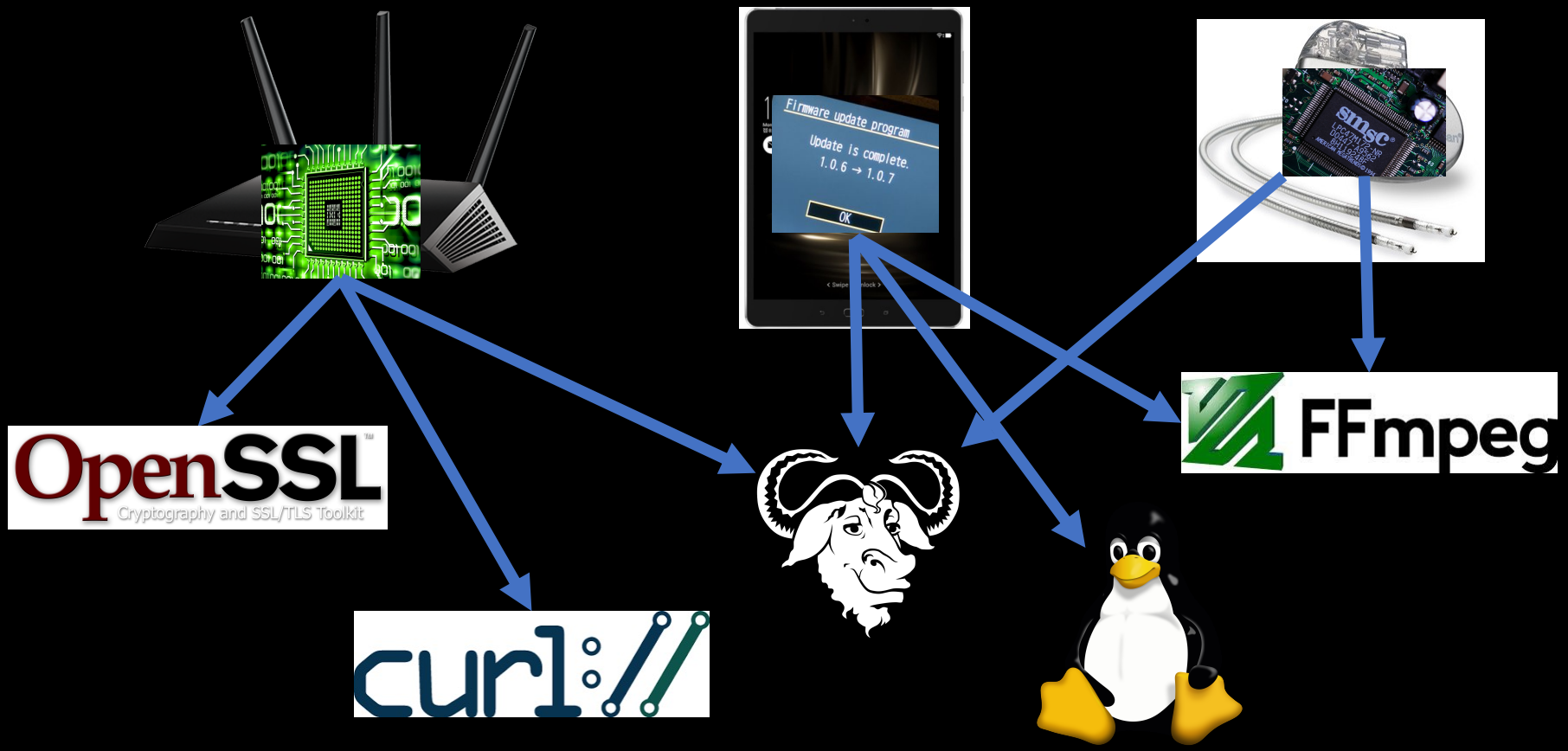
Motivation


NETGEAR

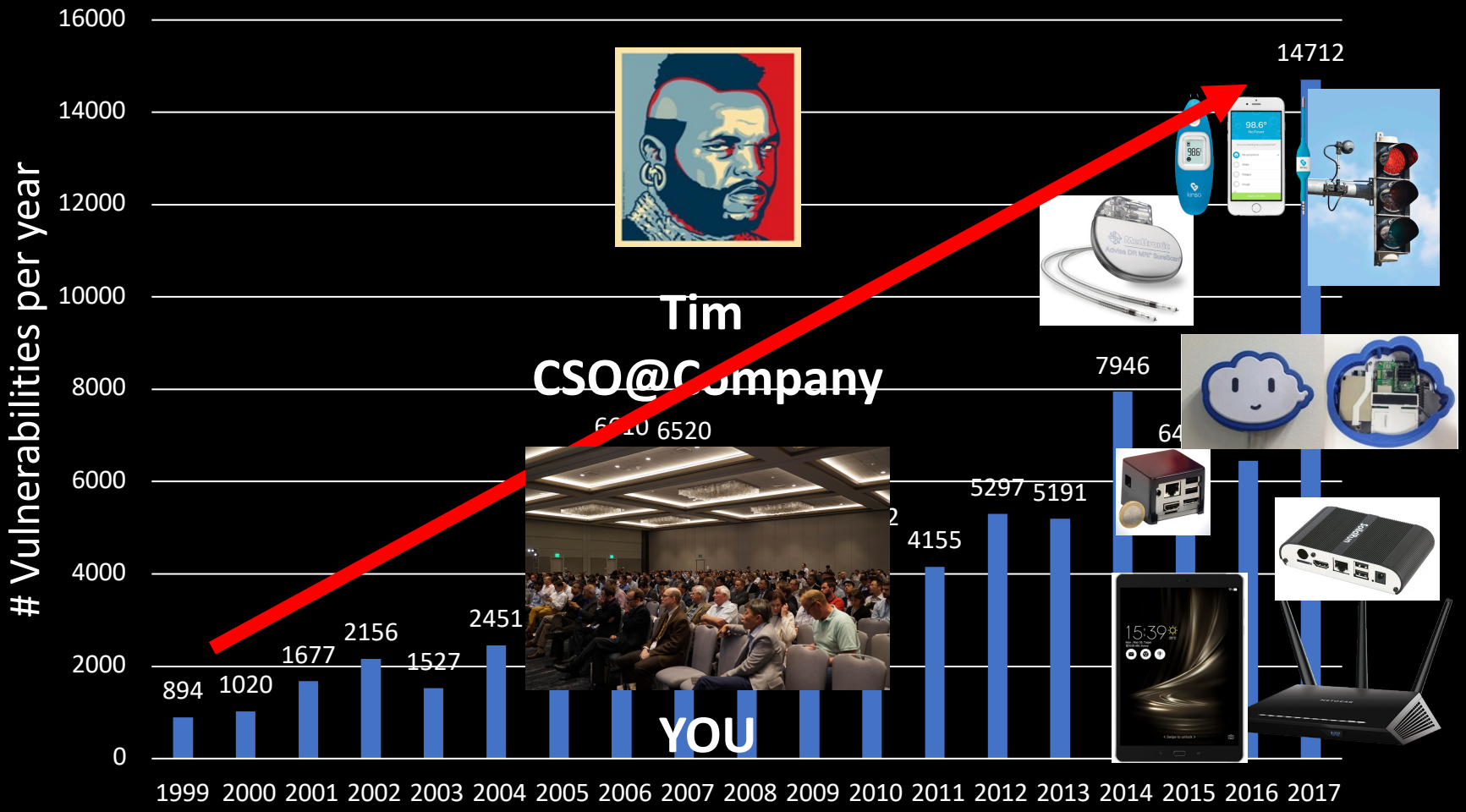

98.6°
No Fever
kinsa
Medtronic
Advisa DR MRI SureScan


15:39
Swipe to unlock

Open Source Is Everywhere



Open Source Vulnerabilities on the Rise

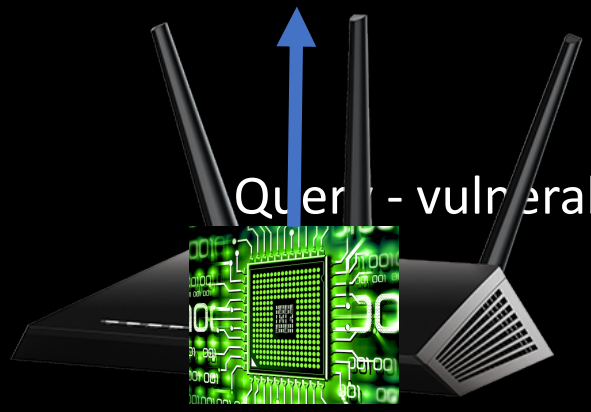


https://www.cvedetails.com/browse-by-date.php

Finding a known vulnerability in a Firmware

wget

CVE-ID	
CVE-2014-4877	Learn more at National Vulnerability Database (NVD) • CVSS Severity Rating • Fix Information • Vulnerable Software Versions • SCAP Mappings • CPE Information
Description	
Absolute path traversal vulnerability in GNU Wget before 1.16, when recursion is enabled, allows remote FTP servers to write to arbitrary files, and consequently execute arbitrary code, via a LIST response that references the same filename within two entries, one of which indicates that the filename is for a symlink.	



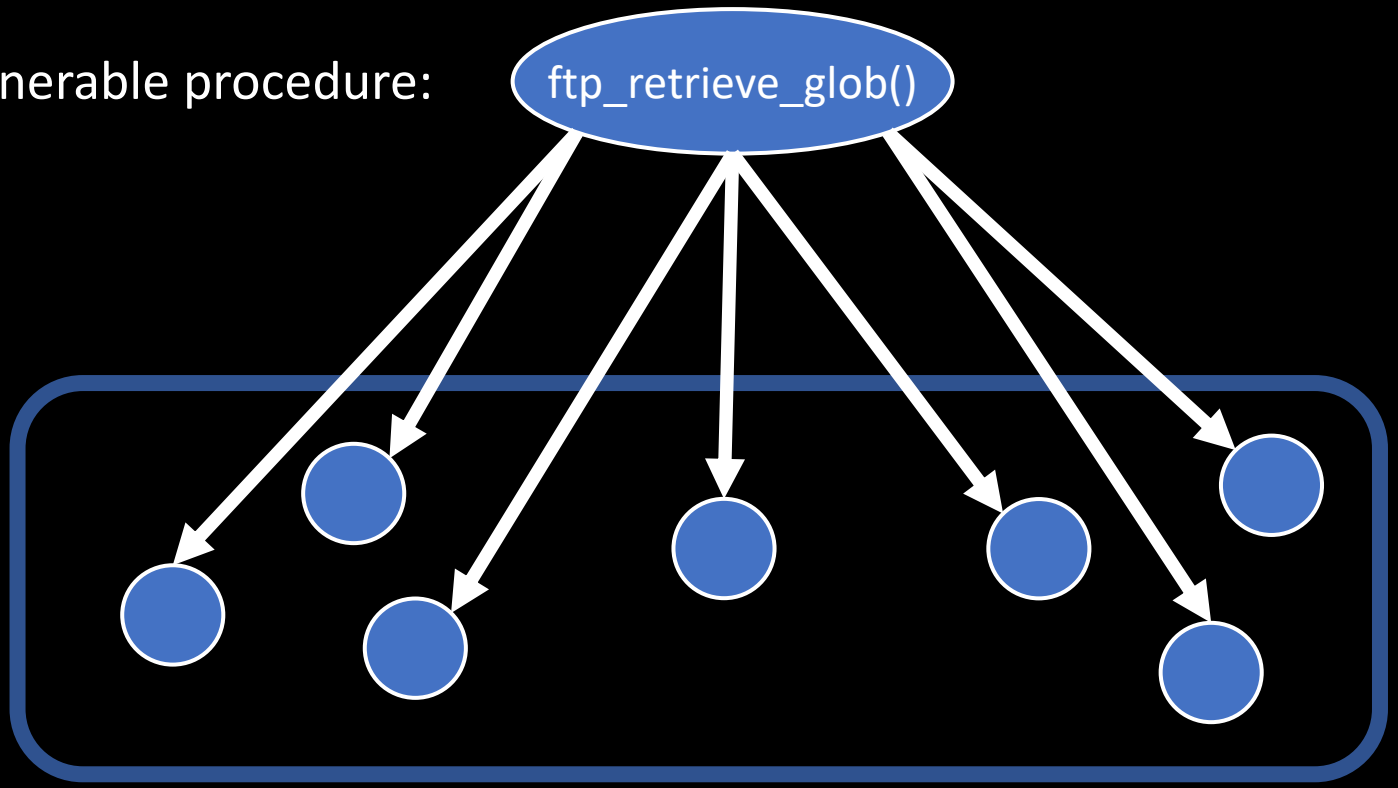
Query - vulnerable procedure:

`ftp_retrieve_glob()`

https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2014-4877

Finding a known vulnerability in a Firmware

Query - vulnerable procedure:



Target Firmware:

Challenge: Code Is Syntactically Different

```
jalr    t9
move    s2, a0
move    s5, v0
li      v0, 0x1F
lw      gp, 0x28+sp
bne     s5, v0, 0x40E744
move    v0, s5
```

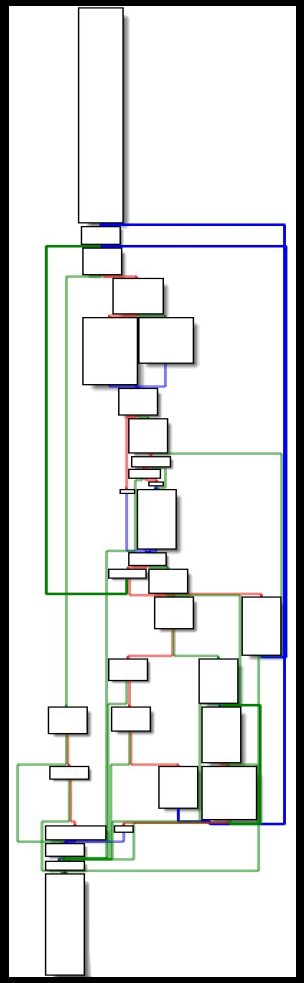
gcc v5.2 –O2

```
addiu   a2, sp, 0x20
move    s4, a1
jal     0x40B2AC
move    s5, a0
li      v1, 0x1F
beq     v0, v1, 0x40B518
lui     s6, 0x47
```

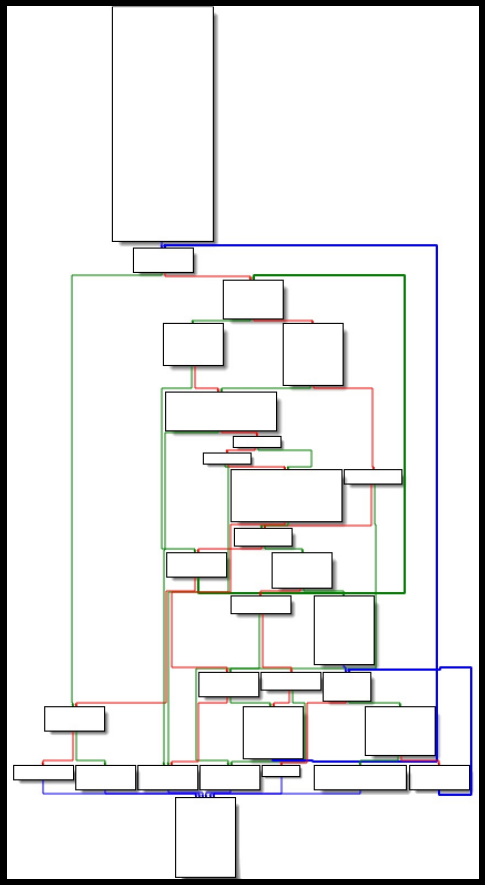
NETGEAR product firmware

Challenge: Control-Flow-Graph Will Not Help

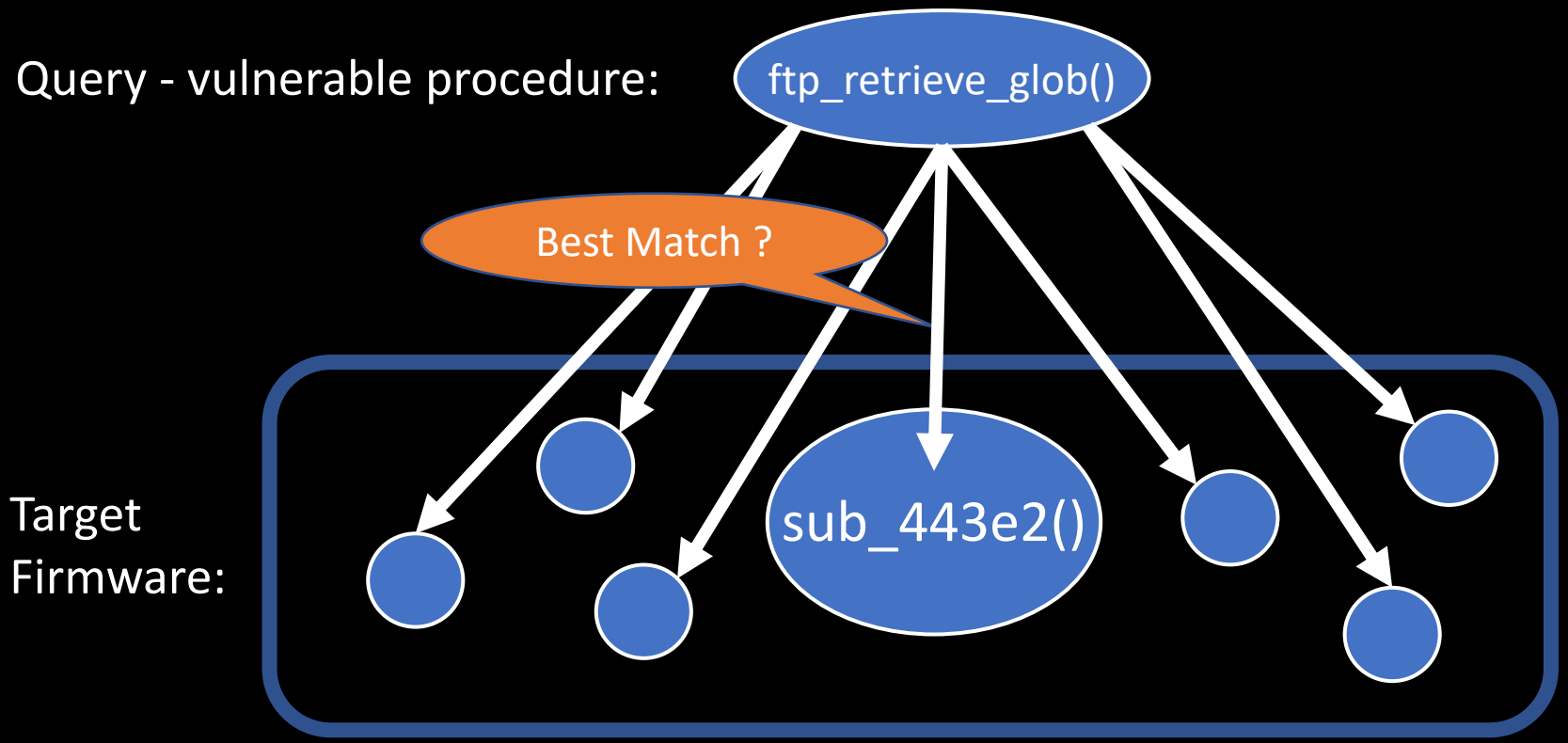
Query CFG:



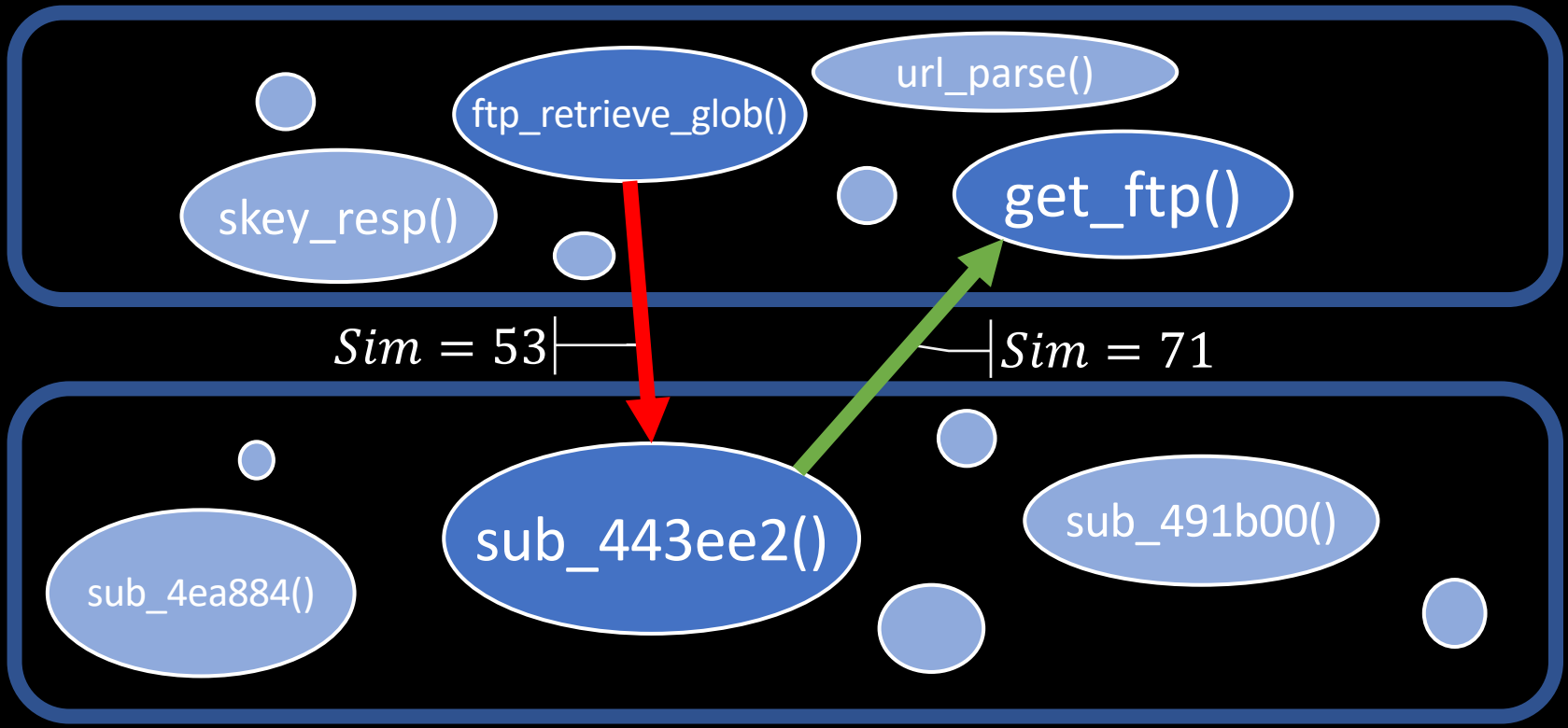
Target CFG:



Finding a known vulnerability in a Firmware

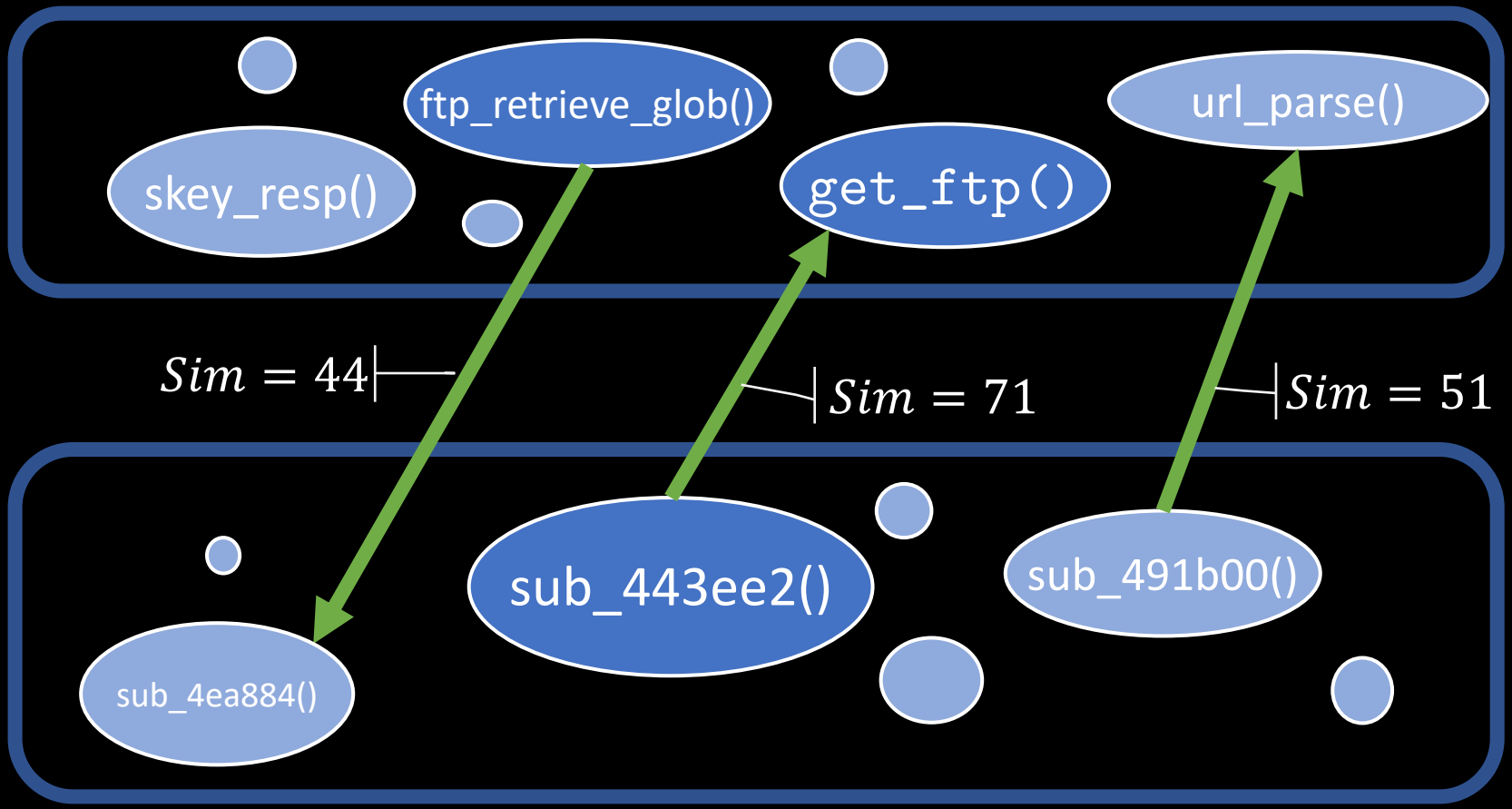


Procedure–Centric Search Misses

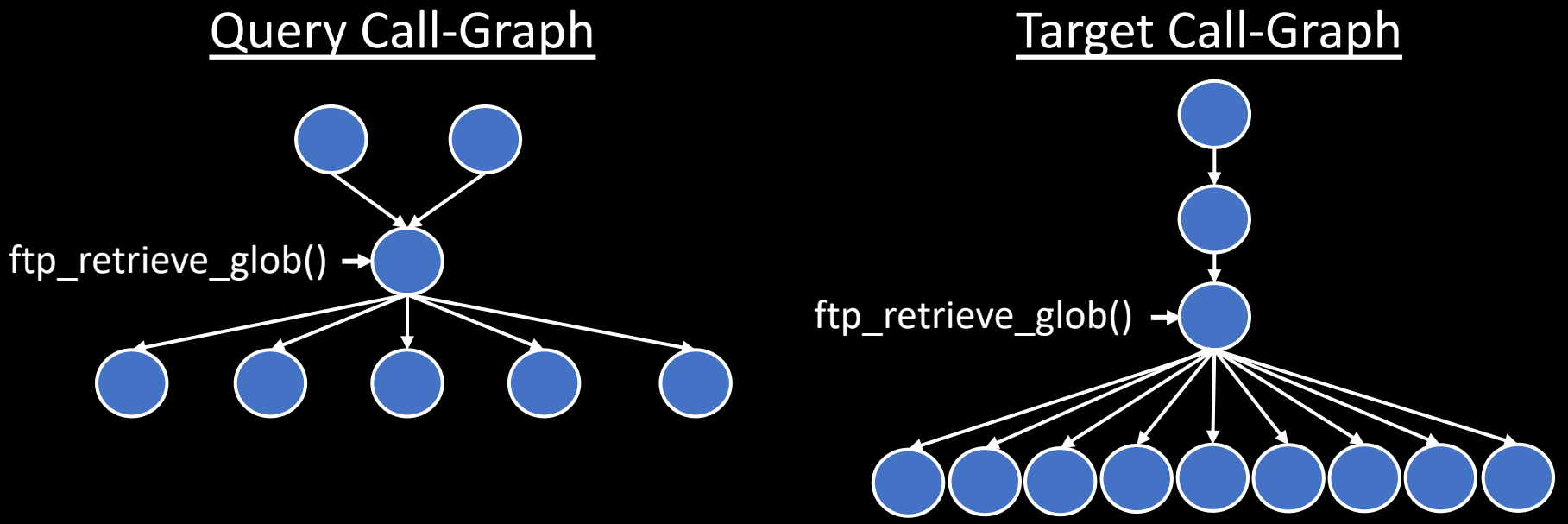


Best match relation – is **not symmetric**

Using **Executable**–Centric Search



Challenge: Procedure Structure Will Not Help



Finding Vulnerabilities

- Precise - avoid false positives
- Flexible – find similarities even when using:
  - Various CPU Architectures (ISA differs syntactically)
  - Custom tool-chains (Compiler vendors, -O123s)
- Scalable – fast enough to work in our scenario
  - Only the minimal partial-matching is calculated

Our Approach

I want to play a game

The Rules of the Game

- The game is played by a player and a rival

- Player needs to create a **partial matching**
  - Must contain q_v - the vulnerable procedure
- Rival tries to find **inconsistencies** in player's matches
- "skipping" a best match allowed **only by** expanding the partial-match

The Rules of the Game (2)

- Player wins the game by finding a **consistent** match
- Rival wins when player gives up (or by timeout / too many game steps)

- This is a two-player game in the formal sense
- Here we only provide some intuition
- Full details in the paper

Game: Find Match for q_v

Q: Query Executable

T: Target Executable

Tim thinks this is best match in T

q_v

t_1

Stack of procedures to match

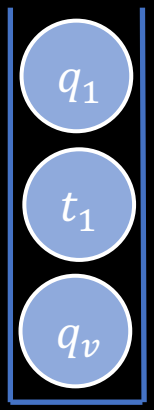
$q_v \leftrightarrow t_1$

q_v

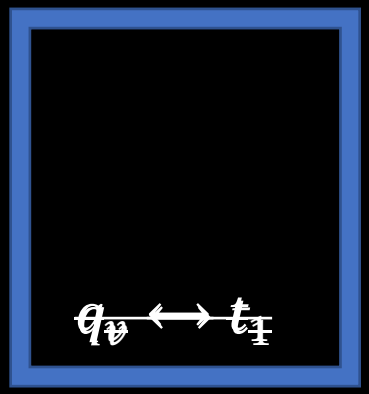
To Match

Matches

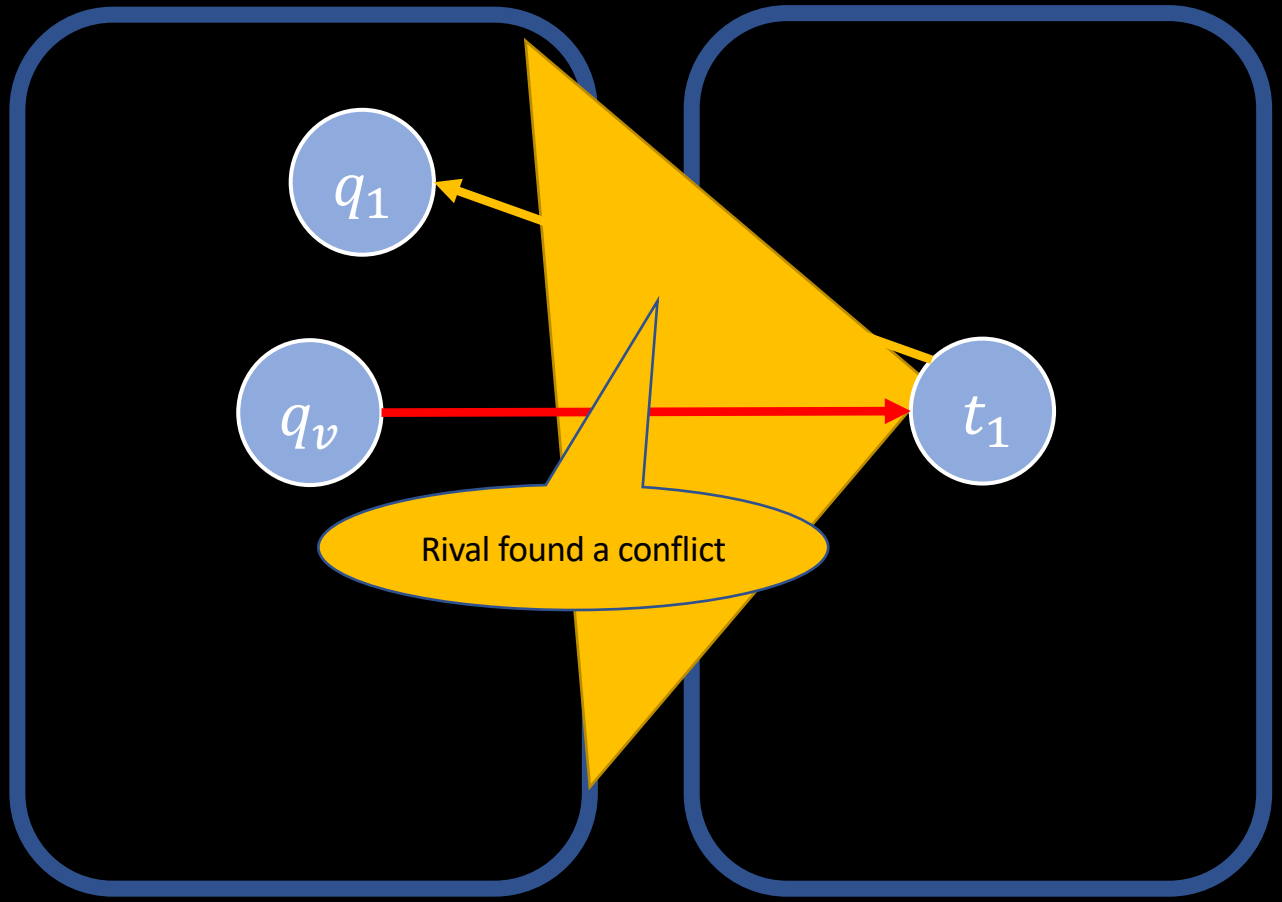
Game: Reverse Search by Rival



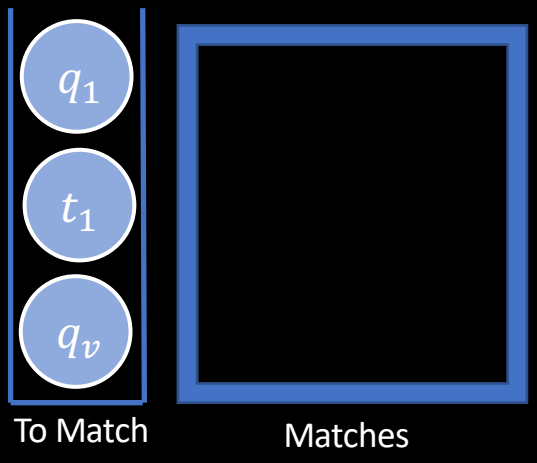
To Match



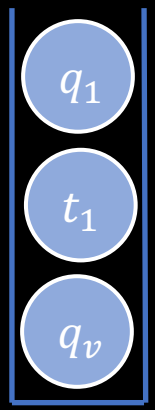
Matches



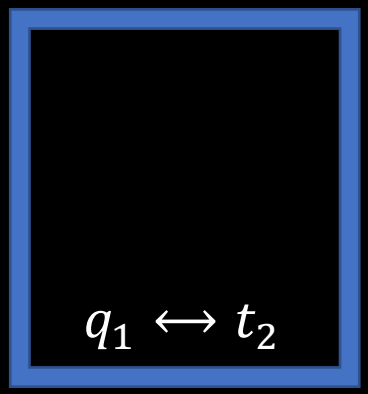
Game: Explain the Skip



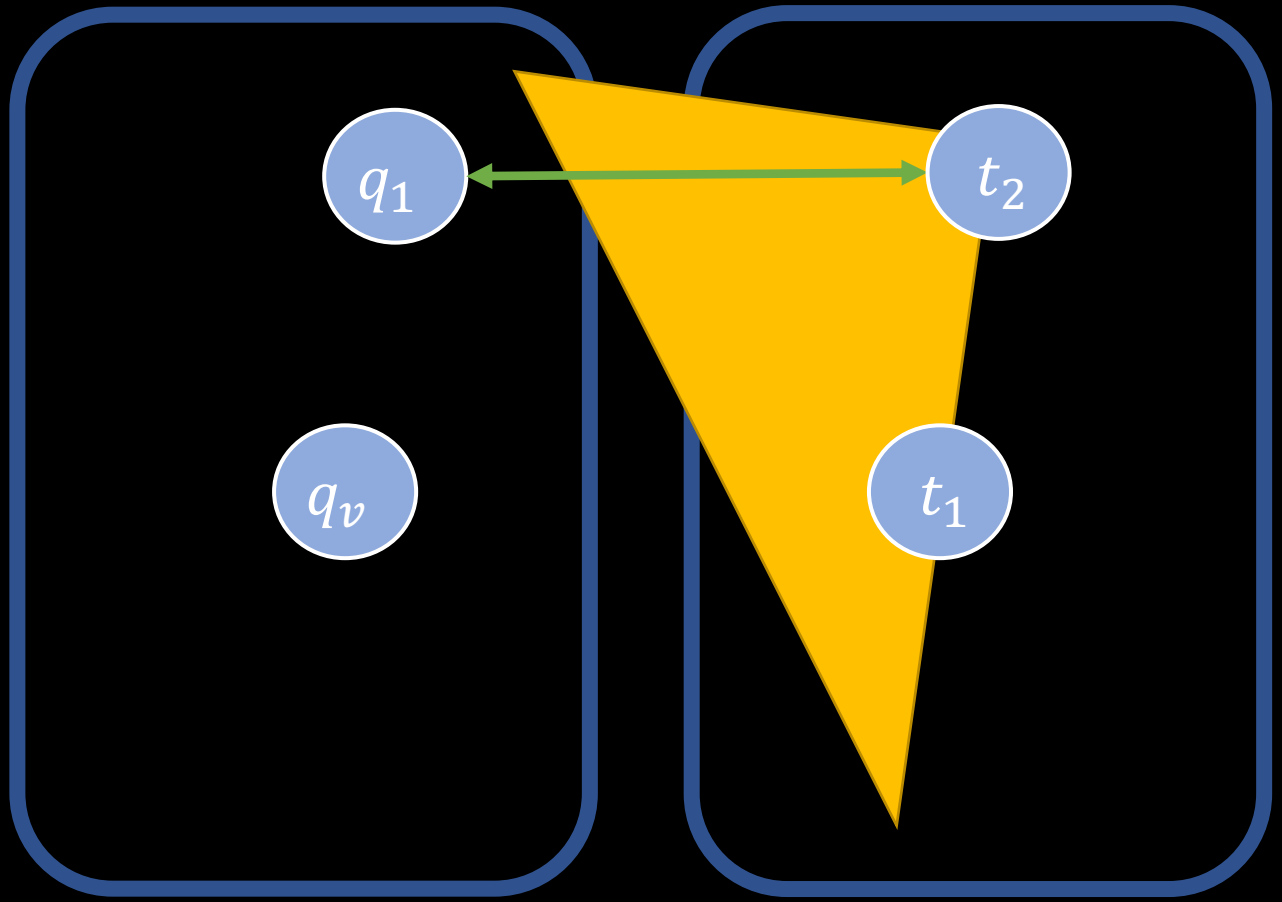
Game: First Match



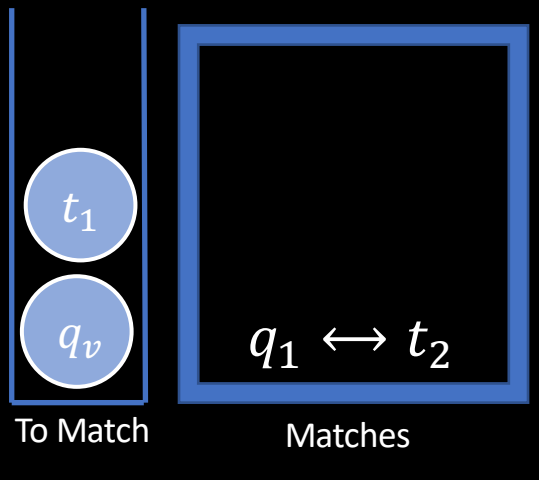
To Match

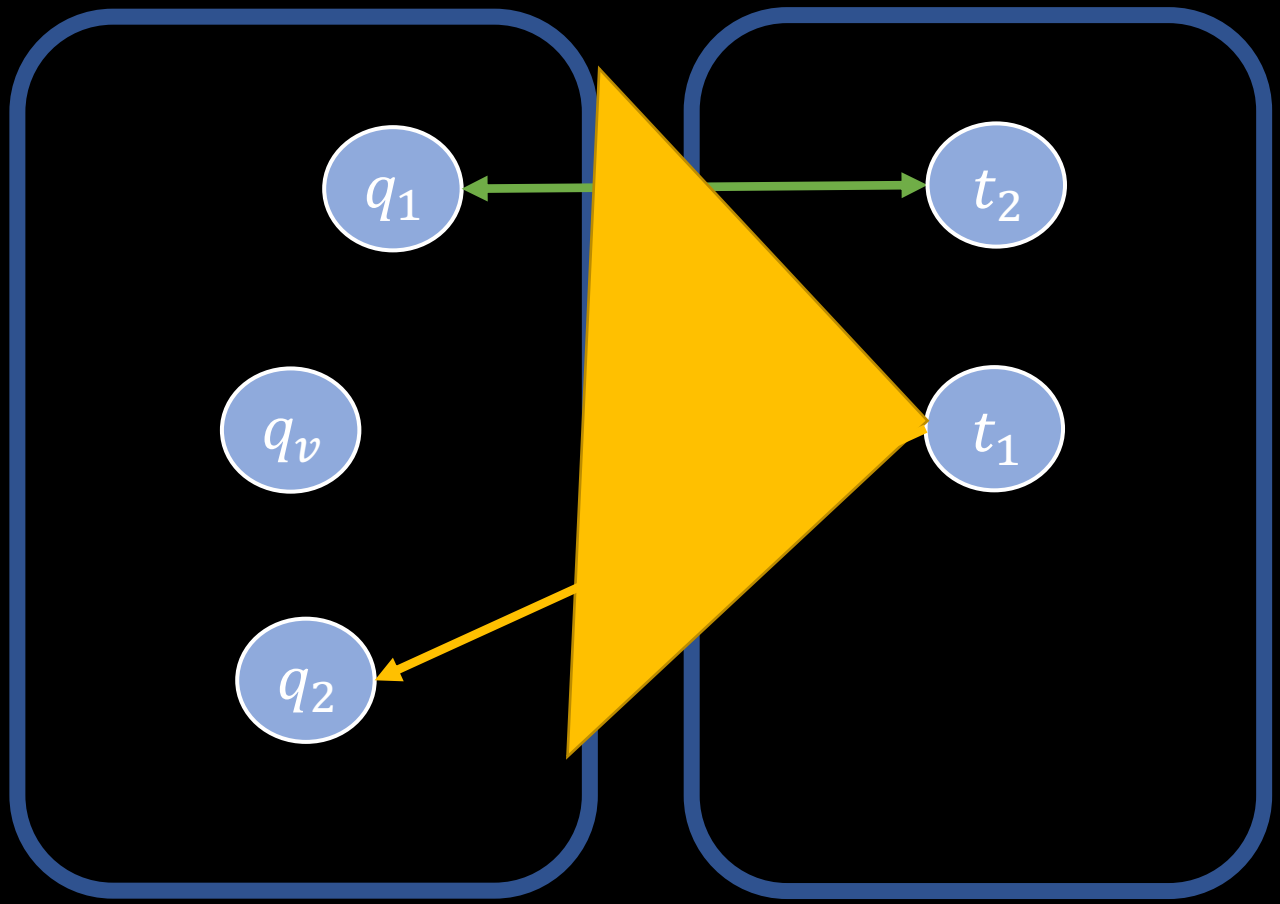


Matches



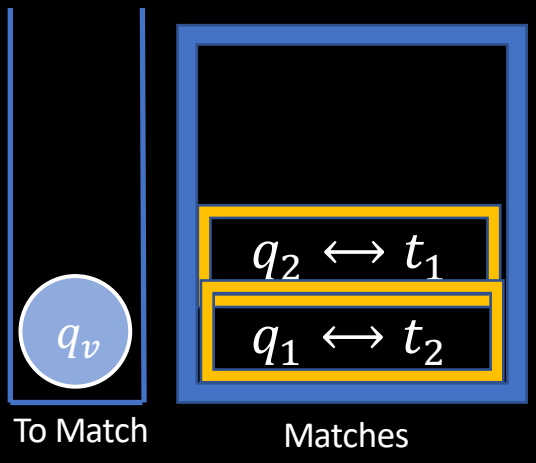
Game: Working Up the Stack

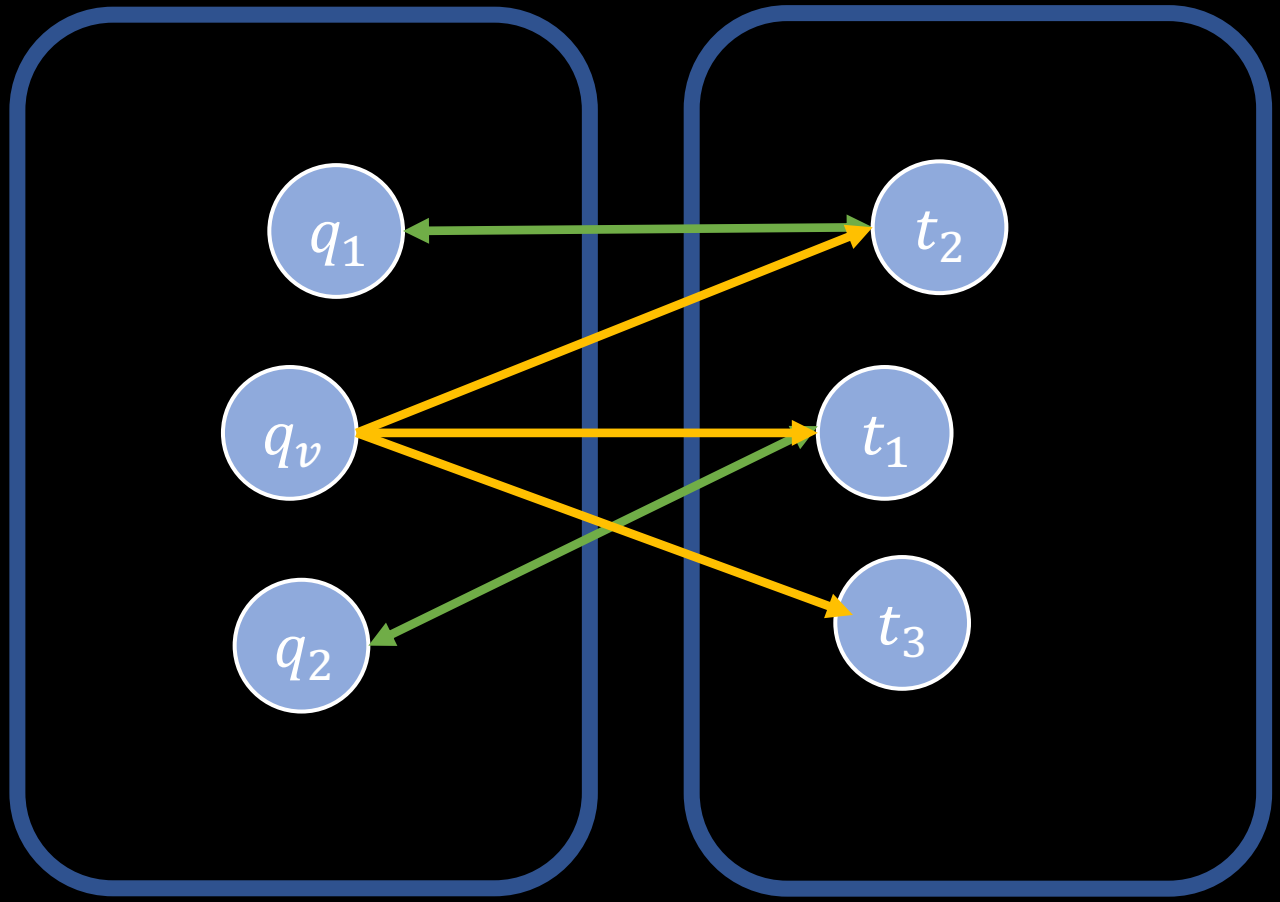




Game: Three Steps Forward

Game: Getting Back To q_v





Game: Partial Match Found

A full match is too **expensive**

q_1 t_2 q_v t_1 q_2 t_3

Partial matching provides **confidence**

q_v

To Match

$q_2 \leftrightarrow t_1$

$q_1 \leftrightarrow t_2$

Matches

Evaluation

Prototype of our approach - FirmUp

Evaluation

- Corpus
  - **~5000** Firmware images crawled from public repositories
    - NETGEAR, D-Link, ASUS
  - ~2000 contained relevant executables (Arch + OS)
    - 32bit Architectures: MIPS TECHNOLOGIES, ARM, intel, PowerPC
  - Total of **~200,000** executables
  - Containing **~40,000,000** procedures

- Queries - 7 real world public vulnerabilities (CVE) from diverse types
  - DOS, BOF, input validation, information disclosure, and path traversal

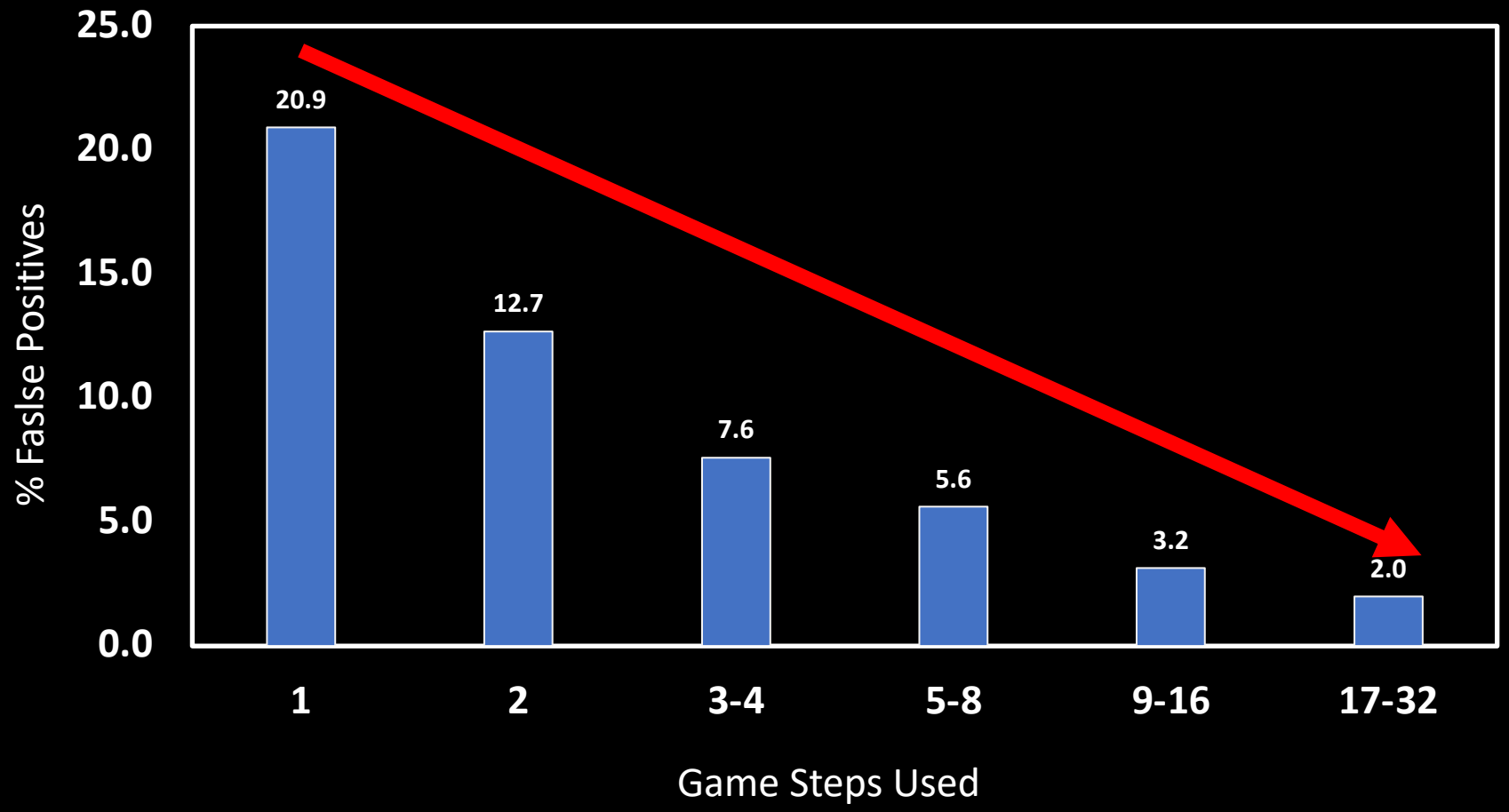
Finding Vulnerabilities Using FirmUp

373 confirmed vulnerabilities, 147 in the latest available Firmware

#	CVE	Package						
1	2011-0762	vsftpd	vsf_filename_passe_				29	2m
2	2009-4593	bftpd	bftpdutmp_log	63		NETGEAR	15	4m
3	2012-0036	libcurl	curl_easy_unescape	1		NETGEAR	0	12s
4	2013-1944	libcurl	tailmatch	5		ASUS,D-Link	2	1m
5	2013-2168	dbus	printf_string_upper_bound	10		D-Link, NETGEAR	5	7m
6	2014-4877	wget	ftp_retrieve_glob	69	14	ASUS, NETGEAR	35	18m
7	2016-8618	libcurl	alloc_addbyter	149	0	ASUS,D-Link, NETGEAR	61	25m

New versions => new procedures => symmetry is broken

The Importance of the Game

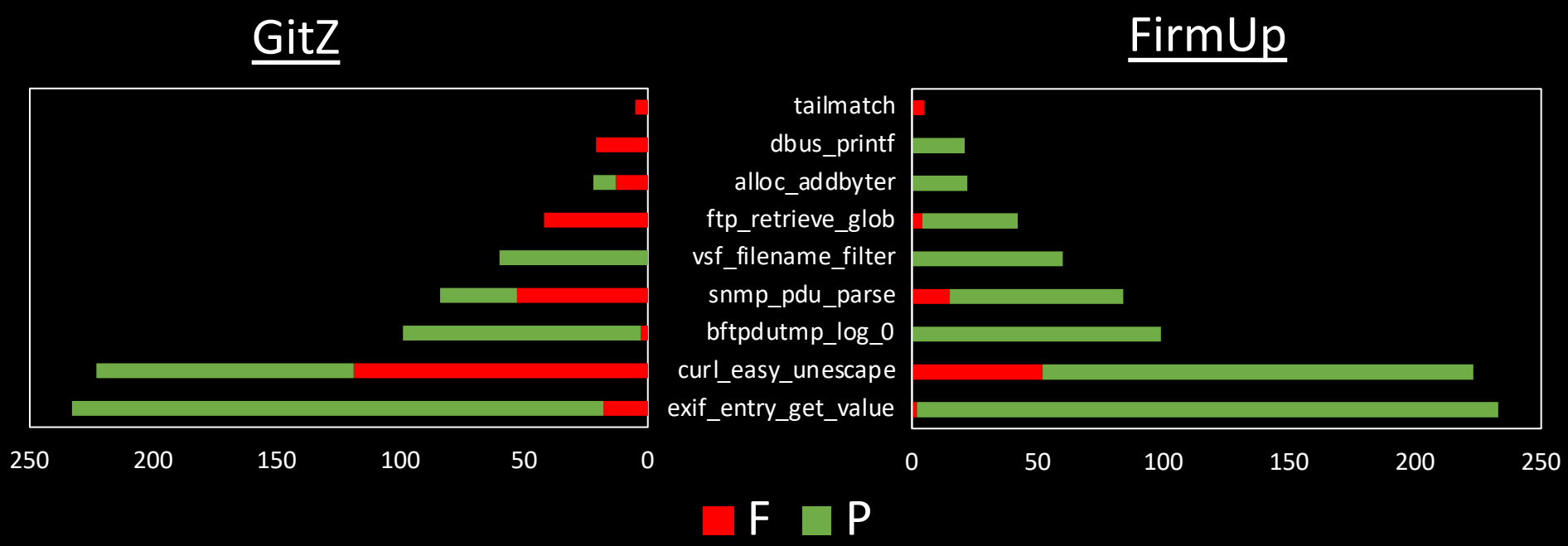


Wait, Where Are the False-Negatives?

- Data from the “wild” is not labeled => no false-negatives
  - To save space debug information **is stripped** in firmware build
- Some non-stripped executables existed in corpus
  - Usually found in early versions of firmware (maybe for debugging)
  - Library procedure names cannot be stripped (importing/calling by name)
- Extended experiment by Including two more CVEs

GitZ Vs FirmUp

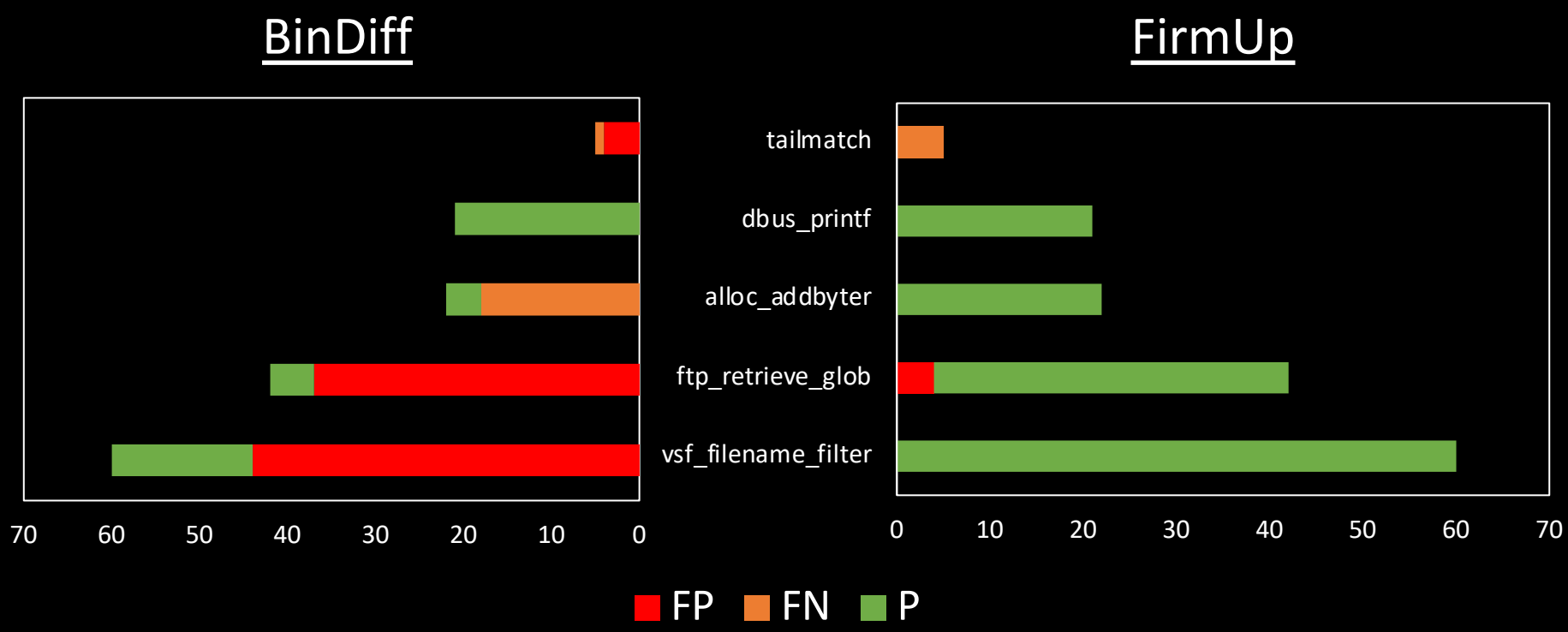
GitZ [PLDI17] - **our** previous work, procedure-centric similarity search



GitZ encountered 34% false positives
compared with 9.88% for FirmUp

BinDiff Vs FirmUp

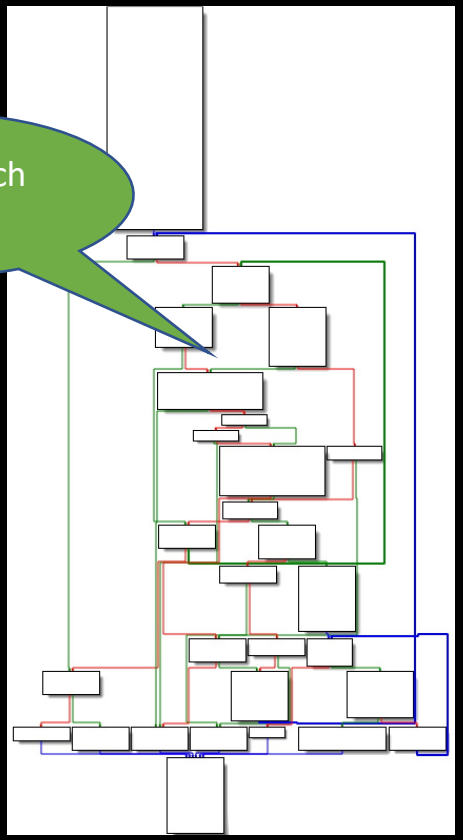
BinDiff - **Industry** standard tool (recently made free)



BinDiff encountered over 69.3% false results overall compared with 6% for FirmUp

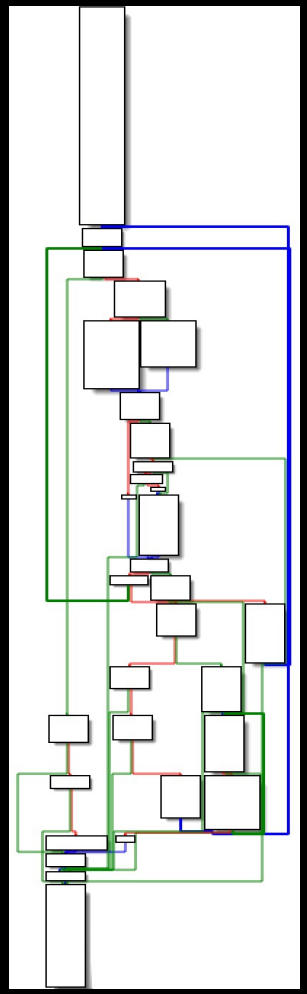
Similar Structure ≠ Similar Semantics

Target #1

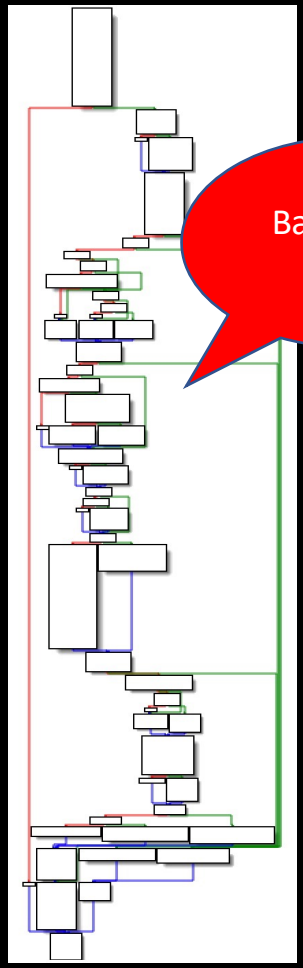


Right Match
FirmUp

Query



Target #2



Bad Match
Bindiff

Summary

- Procedure-centric search is lacking
  - shown in comparison to GitZ
- Executable-centric search uses available information to improve search
- Full-equivalence is expensive, game-inspired partial equivalence instead
- Evaluated on data from the “wild” : 373 confirmed vulnerabilities, 147 in the latest available firmware

Questions?