CO3: Concolic Co-execution for Firmware

Changming Liu*, Alejandro Mera*, Engin Kirda*, Meng Xu^, Long Lu*

*Northeastern University, ^ University of Waterloo

Embedded Systems and firmware

- Microcontrollers (MCUs):
  - Highly efficient and optimized (firmware and hardware).
  - Resourced-constrained.
  - Widely-deployed.
  - Single-chip computers.

Dire security situation.

WORLD ECONOMIC FORUM

Join us Sign in

CYBERSECURITY

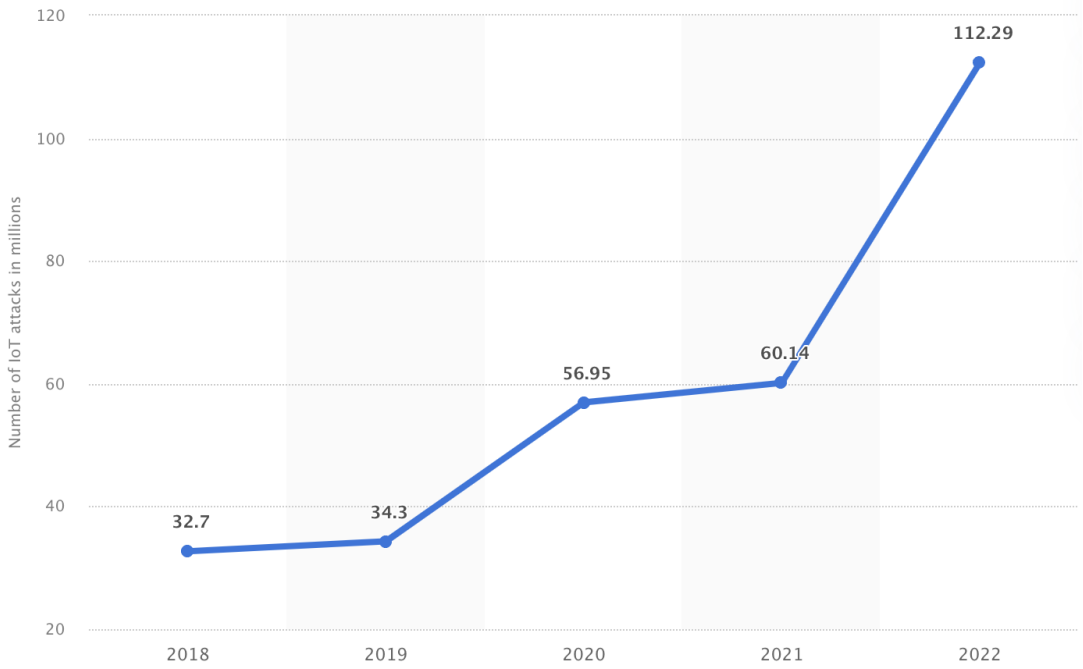
How the Internet of Things (IoT) became a dark web target – and what to do about it

May 17, 2024

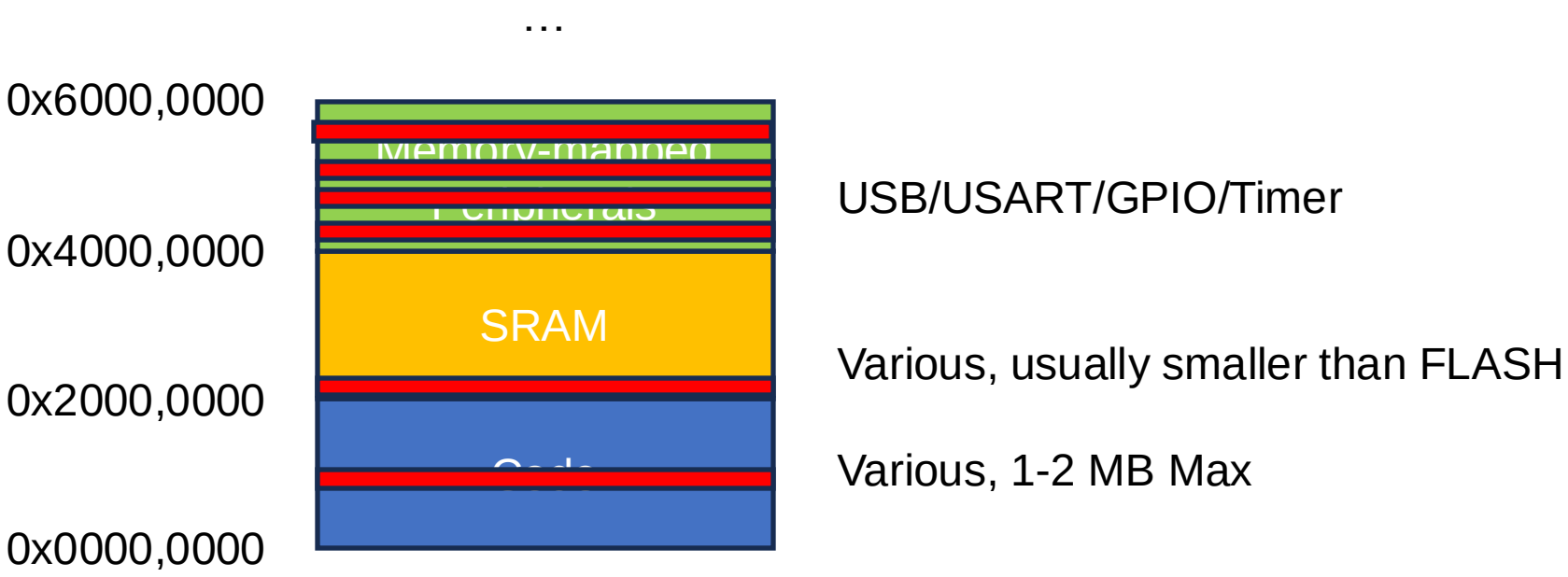
Security Cybersecurity

IoT Malware Attacks Jump 400% Since 2022, Report

Manufacturing was the primary target for malware attacks over the past year, though all industries adopting connected devices are at risk



Problems with securing the firmware on the MCUs

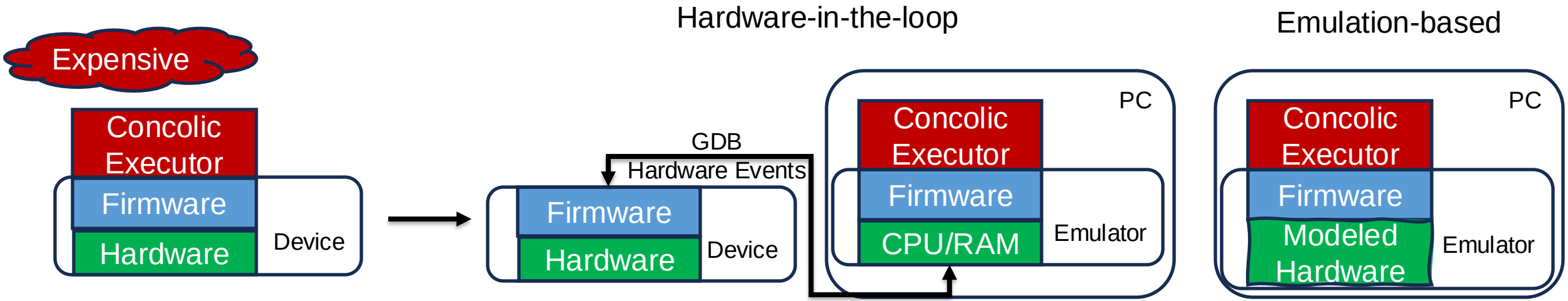


1. Resource constrained.
2. Highly heterogenous physical environment.
    1. A lot of peripherals (types, vendors)[1].
    2. Function like a black-box.

[1] Fasano, Andrew, et al. "Sok: Enabling security analyses of embedded systems via rehosting." *Proceedings of the 2021 ACM Asia conference on computer and communications security*. 2021.

Problems with securing the firmware on the MCUs- Cont

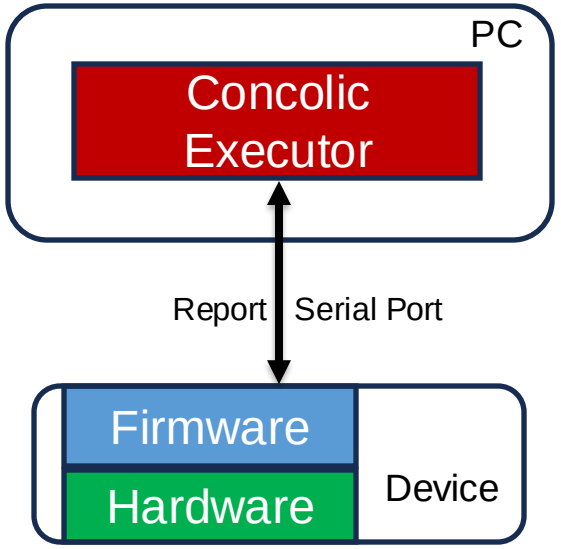
- For the past decade, almost all works **rehost** the firmware.
- Hardware-in-the-loop:
  - Expensive (7 hardware access per second).
  - Hard to support all hardware (e.g., DMA).
  - CPU halting (breaks real-time operation).
  - GDB interface (high in price)
- Emulation: modeled hardware is bad.



Objectives

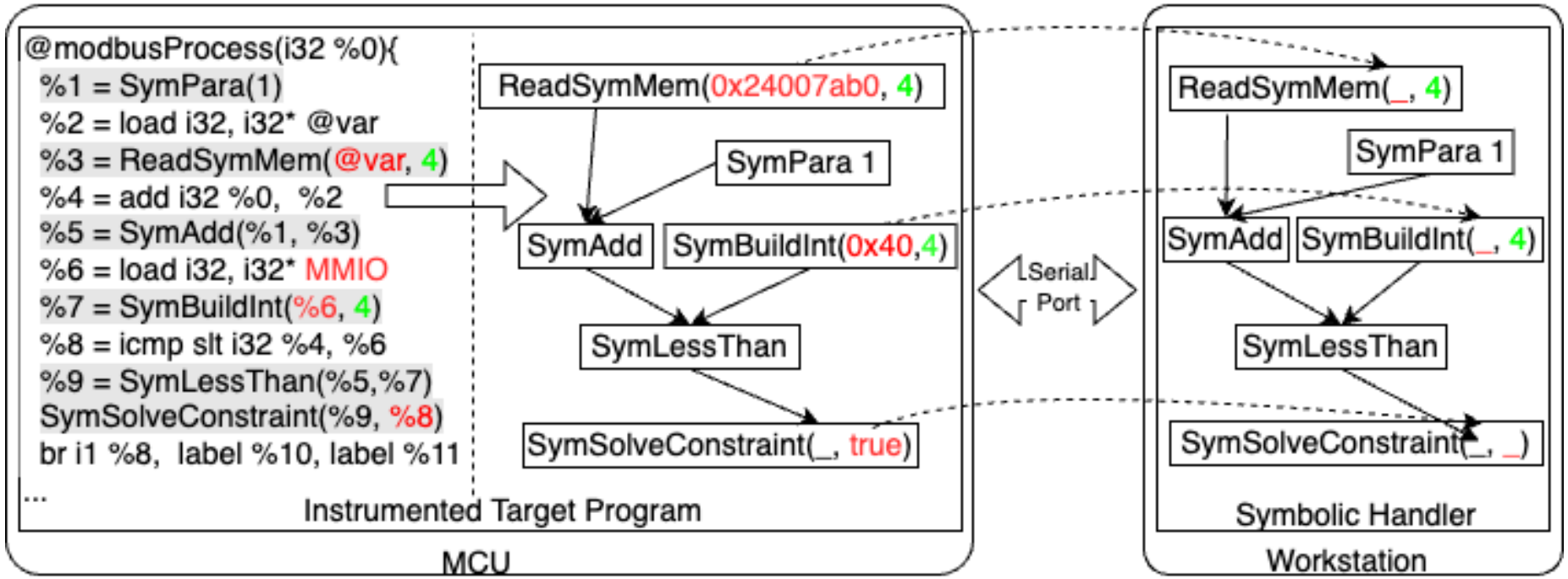
- High-quality peripheral access.
- Performant concolic execution.
- Universally applicable.
- Support All peripherals and hardware.

1. Simpler Communication.
   - No hardware events.
2. No emulator.
3. only need Serial Port (i.e., USART/USB-CDC)
   - No GDB.
   - No CPU Halting.
4. Real hardware and peripherals.

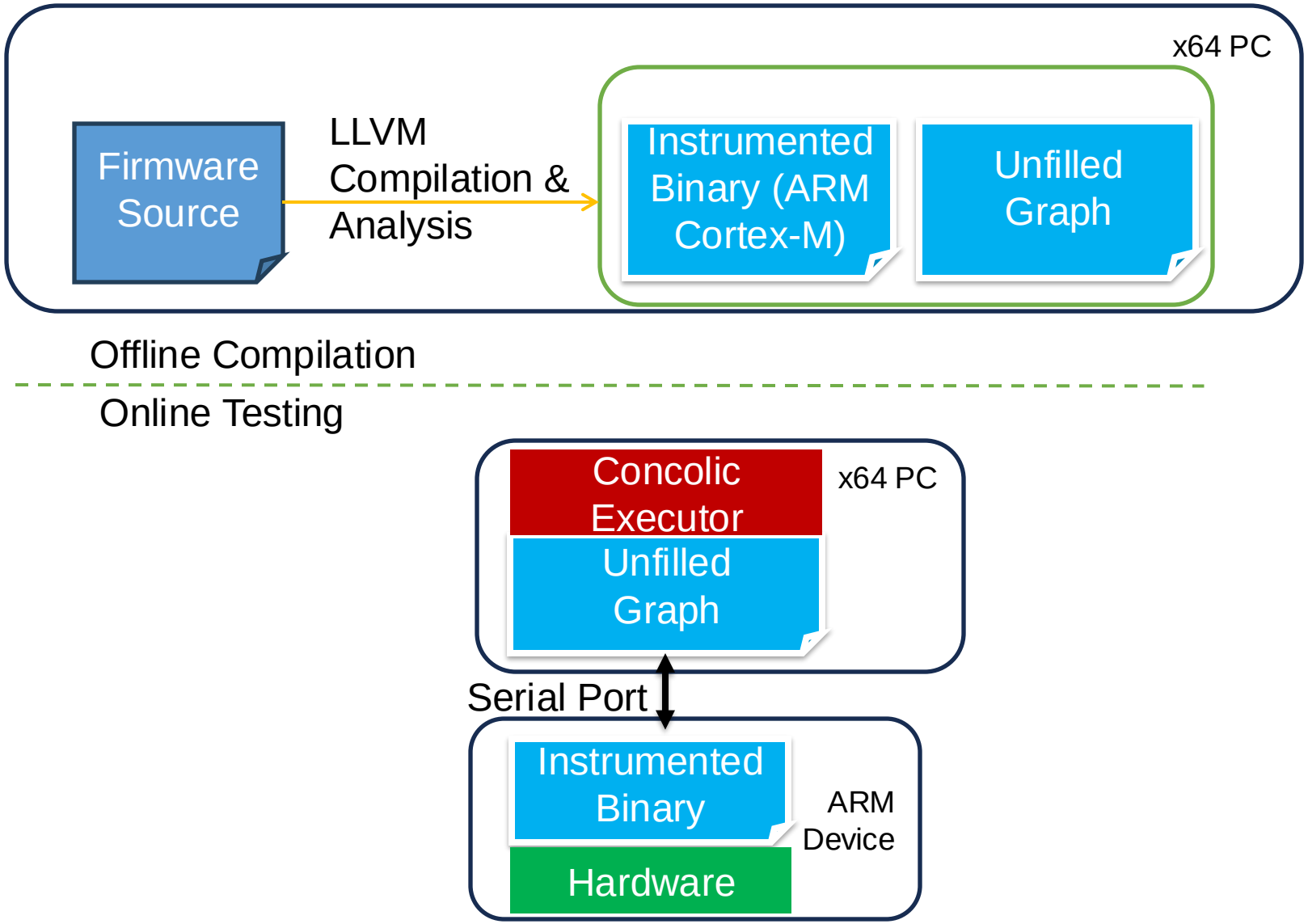


How to achieve this?

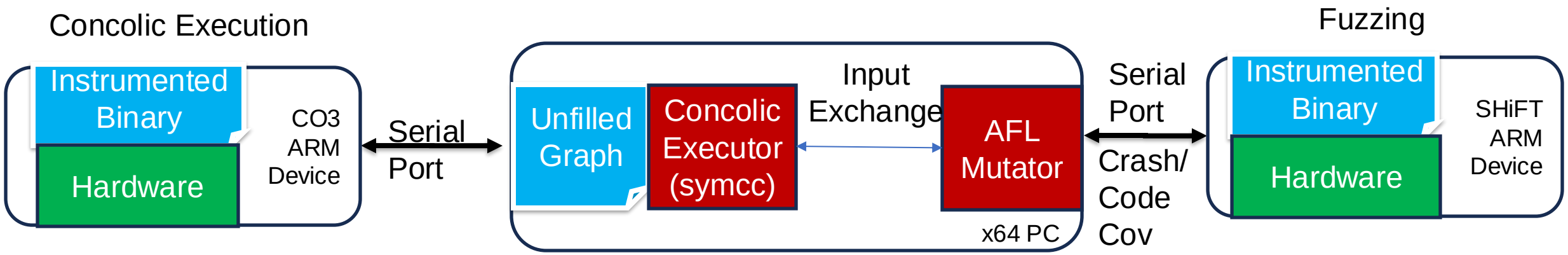
Compile-time analysis + instrumentation.



How to achieve this- Cont



Hybrid Fuzzing



SHACO

Evaluation

- Speed:
  - 1.3-1.7x faster than SymCC (SotA concolic executor)
  - 1000x faster than Avatar2 (Classic hardware-in-the-loop)
- Cover more code within 24 hours.
  - 1.2x more than Symcc
  - 2x more than Avatar2
- Overhead:
  - 27% FLASH overhead
  - 2.9% RAM overhead. (different modes)

Evaluation: bug detection

- Compare SHACO with P2IM/Fuzzware
  - 1000x speed up in detecting all known bugs.
  - Eliminates hundreds of false positives per firmware.
  - Found 3 new bugs.

Ref	#	Firmware	OS	MCU	*SHACO*				SHiFT					P2IM/DICE					Fuzzware				
					Time(s)	UC	TP	FP	Time(s)	SUF	UC	TP	FP	Time(s)	SUF	UC	TP	FP	Time(s)	SUF	UC	TP	FP
P2IM [24]	1	*PLC*	**F**	h743	38	8	4	0	165	4.3x	8	4	0	3873	101.9x	183	4	2	73980	1946x	30	4	2
DICE [42]	2	*Modbus*	**F**	h743	20	4	3	0	38	1.9x	4	3	0	29881	1494x	71	3	2	**I**	**n/a**	25	0	1
	3	*Midi*	**F**	h743	126	20	2	0	129	1.02x	20	2	0	25413	201x	4	2	0	**I**	**n/a**	104	0	2
SHiFT [43]	4	*Synthetic*	**F**	h743	26	20	11	0	340	13.1x	23	11	0	**I**	**n/a**	8	3	1	**I**	**n/a**	486	0	10
	5	*Shelly Dimmer*	**F**	h743	40	6	3	0	262	6.5x	7	3	0	**NB**	**n/a**	0	0	0	**I**	**n/a**	1496	0	1
SHACO	6	*CANopen*	**F**	l4r5	164	7	3	0	525	3.2x	8	3	0	**NB**	**n/a**	0	0	0	**I**	**n/a**	0	0	0
	7	*Stepper*	**F**	l4r5	187	7	2	0	691	3.7x	7	2	0	**NB**	**n/a**	0	0	0	**I**	**n/a**	2355	1	3
	8	*Bldc*	**C**	f429	376	4	2	0	2068	5.5x	4	2	0	**NB**	**n/a**	0	0	0	**NB**	**n/a**	0	0	0

Demo

lcm@DESKTOP-DTCDME0: ~ | lcm@lcm-lab-ubuntu: ~

```
time:16.69 / 86400
symcc generated 18 inputs, 0 new edge found

iter:1,cur at 21 from 1 to 123, need 143351 inputs to finish
time:17.24 / 86400
symcc generated 53 inputs, 0 new edge found

^CTraceback (most recent call last):
  File "/home/lcm/github/spear/spear-code/code_coverage/generate_inputs_symcc.py", line 135, in <mod
ule>
    main()
  File "/home/lcm/github/spear/spear-code/code_coverage/generate_inputs_symcc.py", line 131, in main
    num_generated_inputs, num_run_inputs, total_time, coverage = runSymcc(benchmark)
  File "/home/lcm/github/spear/spear-code/code_coverage/generate_inputs_symcc.py", line 72, in runSy
mcc
    p1.wait()
  File "/usr/lib/python3.10/subprocess.py", line 1209, in wait
    return self._wait(timeout=timeout)
  File "/usr/lib/python3.10/subprocess.py", line 1959, in _wait
    (pid, sts) = self._try_wait(0)
  File "/usr/lib/python3.10/subprocess.py", line 1917, in _try_wait
    (pid, sts) = os.waitpid(self.pid, wait_flags)
KeyboardInterrupt

lcm@lcm-lab-ubuntu:~/github/spear/spear-code/code_coverage$ python generate_inputs_symcc.py
iter:0,cur at 0 from 0 to 1, need 73813 inputs to finish
time:1.52 / 86400
symcc generated 122 inputs,

iter:1,cur at 1 from 1 to 123, need 80544 inputs to finish
time:2.79 / 86400
symcc generated 119 inputs,

^CTraceback (most recent call last):
  File "/home/lcm/github/spear/spear-code/code_coverage/generate_inputs_symcc.py", line 135, in <mod
ule>
    main()
  File "/home/lcm/github/spear/spear-code/code_coverage/generate_inputs_symcc.py", line 131, in main
    num_generated_inputs, num_run_inputs, total_time, coverage = runSymcc(benchmark)
  File "/home/lcm/github/spear/spear-code/code_coverage/generate_inputs_symcc.py", line 72, in runSy
mcc
    p1.wait()
  File "/usr/lib/python3.10/subprocess.py", line 1209, in wait
    return self._wait(timeout=timeout)
  File "/usr/lib/python3.10/subprocess.py", line 1959, in _wait
    (pid, sts) = self._try_wait(0)
  File "/usr/lib/python3.10/subprocess.py", line 1917, in _try_wait
    (pid, sts) = os.waitpid(self.pid, wait_flags)
KeyboardInterrupt

lcm@lcm-lab-ubuntu:~/github/spear/spear-code/code_coverage$
```

```
In [1]: e
Out[1]: b''

In [2]: os._exit(1)
lcm@lcm-lab-ubuntu:~/github/spear/CO3/utils$ python co3_firmware.py -p /dev/ttyACM1 -b 7500000
Traceback (most recent call last):
  File "/home/lcm/github/spear/CO3/utils/co3_firmware.py", line 153, in <module>
    main()
  File "/home/lcm/github/spear/CO3/utils/co3_firmware.py", line 150, in main
    runCO3(args)
  File "/home/lcm/github/spear/CO3/utils/co3_firmware.py", line 107, in runCO3
    print("iter:{},cur at {} from {} to {}, edge size:{}, need {} inputs to finish".format(it, cur_input_id,
 batch_input_id_start , batch_input_id_end, estimate_inputs_needed(cur_input_id + 1, total_time, time_budget
)))
IndexError: Replacement index 5 out of range for positional args tuple
lcm@lcm-lab-ubuntu:~/github/spear/CO3/utils$ python co3_firmware.py -p /dev/ttyACM1 -b 7500000
iter:0,cur at 0 from 0 to 1, need 7238 inputs to finish
building time:0.26, transmit 106.45 KB costs:0.19, total time:0.26 / 1440
co3 generated 119 inputs, 0 new edge found

iter:1,cur at 1 from 1 to 120, need 8141 inputs to finish
building time:0.46, transmit 134.09 KB costs:0.29, total time:0.46 / 1440
co3 generated 119 inputs, 0 new edge found

iter:1,cur at 2 from 1 to 120, need 7857 inputs to finish
building time:0.71, transmit 164.19 KB costs:0.38, total time:0.71 / 1440
co3 generated 135 inputs, 0 new edge found

iter:1,cur at 3 from 1 to 120, need 9139 inputs to finish
building time:0.82, transmit 174.29 KB costs:0.45, total time:0.82 / 1440
co3 generated 84 inputs, 0 new edge found

iter:1,cur at 4 from 1 to 120, need 10783 inputs to finish
building time:0.85, transmit 176.54 KB costs:0.50, total time:0.87 / 1440
co3 generated 34 inputs, 0 new edge found

^CTraceback (most recent call last):
  File "/home/lcm/github/spear/CO3/utils/co3_firmware.py", line 153, in <module>
    main()
  File "/home/lcm/github/spear/CO3/utils/co3_firmware.py", line 150, in main
    runCO3(args)
  File "/home/lcm/github/spear/CO3/utils/co3_firmware.py", line 79, in runCO3
    p1.wait(timeout)
  File "/usr/lib/python3.10/subprocess.py", line 1209, in wait
    return self._wait(timeout=timeout)
  File "/usr/lib/python3.10/subprocess.py", line 1953, in _wait
    time.sleep(delay)
KeyboardInterrupt

lcm@lcm-lab-ubuntu:~/github/spear/CO3/utils$
```

Thank you!

- Code available: www.github.com/Lawliar/co3
- MCU is needed to experiment with the firmware.
- Workstation program (e.g., CGC) supported.

- Contact:
- @Law1iar
- charley.ashbringer@gmail.com