NDSS SYMPOSIUM

Facilitating Non-Intrusive In-Vivo Firmware Testing with Stateless Instrumentation

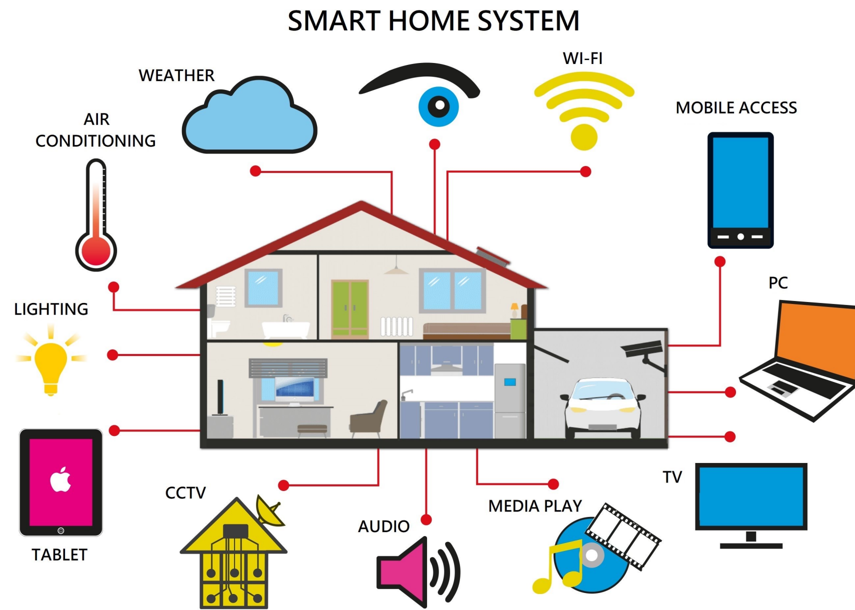
NDSS Symposium 2024

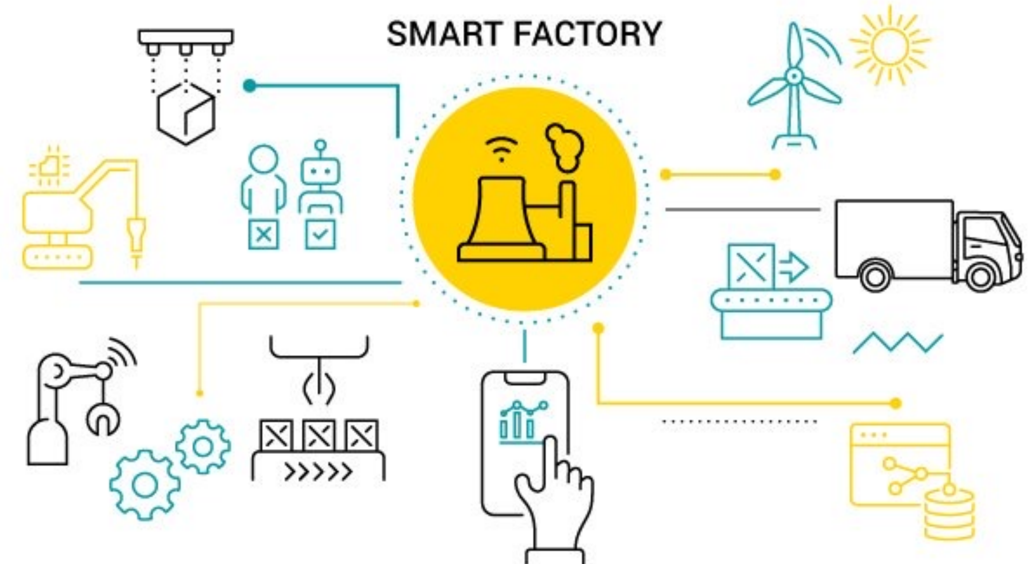
Jiameng Shi*, Wenqiang Li#, Wenwen Wang*, Le Guan*

* The University of Georgia

Independent Researcher

MCU-based Devices are Ubiquitous





Img src: http://visioforce.com/smarthome.html https://www.avsystem.com/blog/smart-factory/

But at the Same Time, They Are Extremely Vulnerable

Multiple Vulnerabilities in Treck TCP/IP Stack Could Allow for Arbitrary Code Execution

MS-ISAC ADVISORY NUMBER:
2020-171

DATE(S) ISSUED:
12/21/2020

OVERVIEW:

Multiple vulnerabilities have been discovered in Treck TCP/IP Stack, the most severe of which could result in arbitrary code execution. Treck TCP/IP Stack are networking protocols libraries specifically designed for embedded systems and are widely used. Successful exploitation of the most severe of these vulnerabilities could allow an attacker to execute arbitrary code in the context of the application. Depending on the privileges associated with the application, an attacker could install programs; view, change, or delete data; or create new accounts with full user rights. If this application has been configured to have fewer user rights on the system, exploitation of the most severe of these vulnerabilities could have less impact than if it was configured with administrative rights.

THREAT INTELLIGENCE:

There are currently no reports of these vulnerabilities being exploited in the wild.

But at the Same Time, They Are Extremely Vulnerable

Multiple Vulnerabilities in Treck
TCP/IP Stac
Arbitrary C

MS-ISAC ADVISORY NUMB
2020-171

DATE(S) ISSUED:
12/21/2020

OVERVIEW:

Multiple vulnerabilities have been d
code execution. Treck TCP/IP Stack
are widely used. Successful exploita
arbitrary code in the context of the
could install programs; view, chang
been configured to have fewer user
have less impact than if it was confi

THREAT INTELLIGENCE:

There are currently no reports of th

The hack attack led to failures in plant equipment and forced the fast shut down of a furnace

But at the Same Time, They Are Extremely Vulnerable

Multiple Vulnerabilities in Treck TCP/IP Stac Arbitrary C

MS-ISAC ADVISORY NUME
2020-171

DATE(S) ISSUED:
12/21/2020

OVERVIEW:

Multiple vulnerabilities have been d
code execution. Treck TCP/IP Stack
are widely used. Successful exploita
arbitrary code in the context of the
could install programs; view, chang
been configured to have fewer user
have less impact than if it was confi

THREAT INTELLIGENCE:

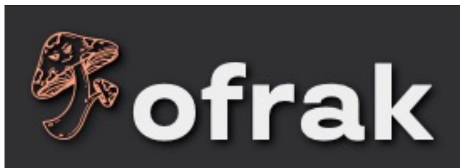
There are currently no reports of th

The hack attack led to failures in plant equipment and forced the fast shut down of a furnac

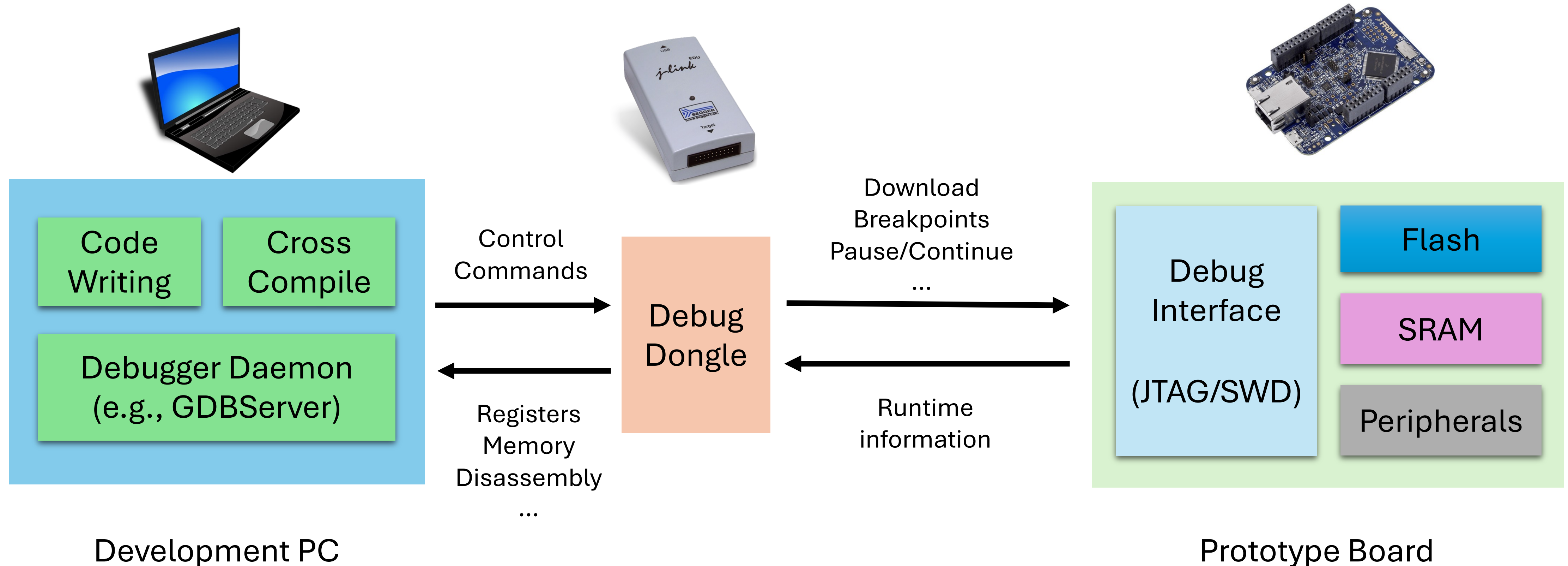
Hacking risk leads to recall of 500,000 pacemakers due to patient death fears

FDA overseeing crucial firmware update in US to patch security holes and prevent hijacking of pacemakers implanted in half a million people

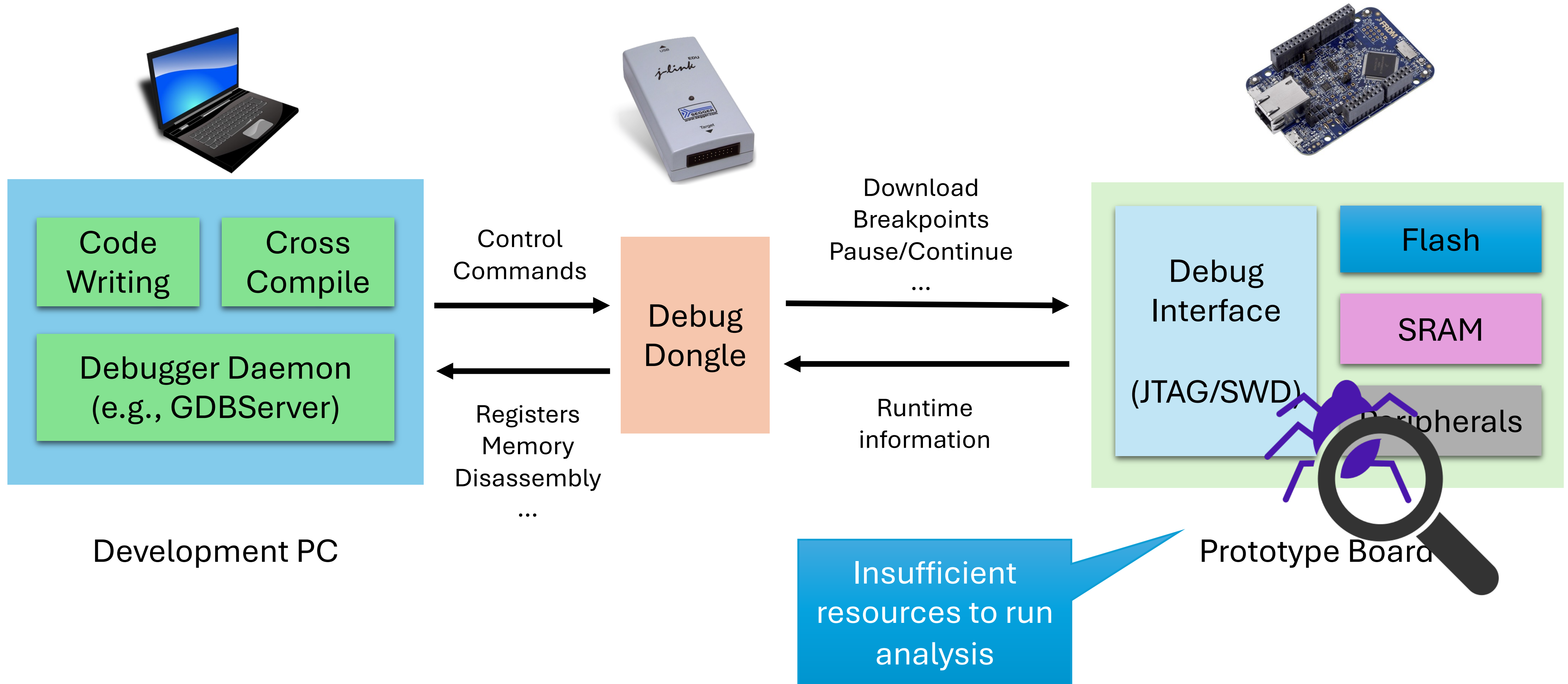
How to Effectively Find Firmware Bugs?



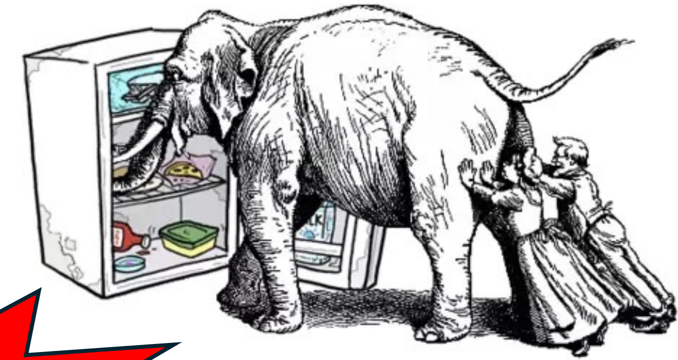
Firmware Development

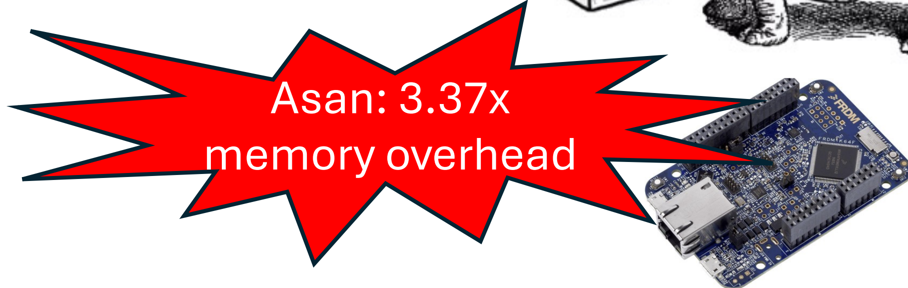


Analysis on the MCU Board?



Analysis on the MCU Board?





Code Writing

Cross Compile

Debugger Daemon (e.g., GDBServer)

Development PC

Control Commands

Registers
Memory
Disassembly
...

Debug Dongle

Download
Breakpoints
Pause/Continue
...

Runtime information

Debug Interface

(JTAG/SWD)

Flash

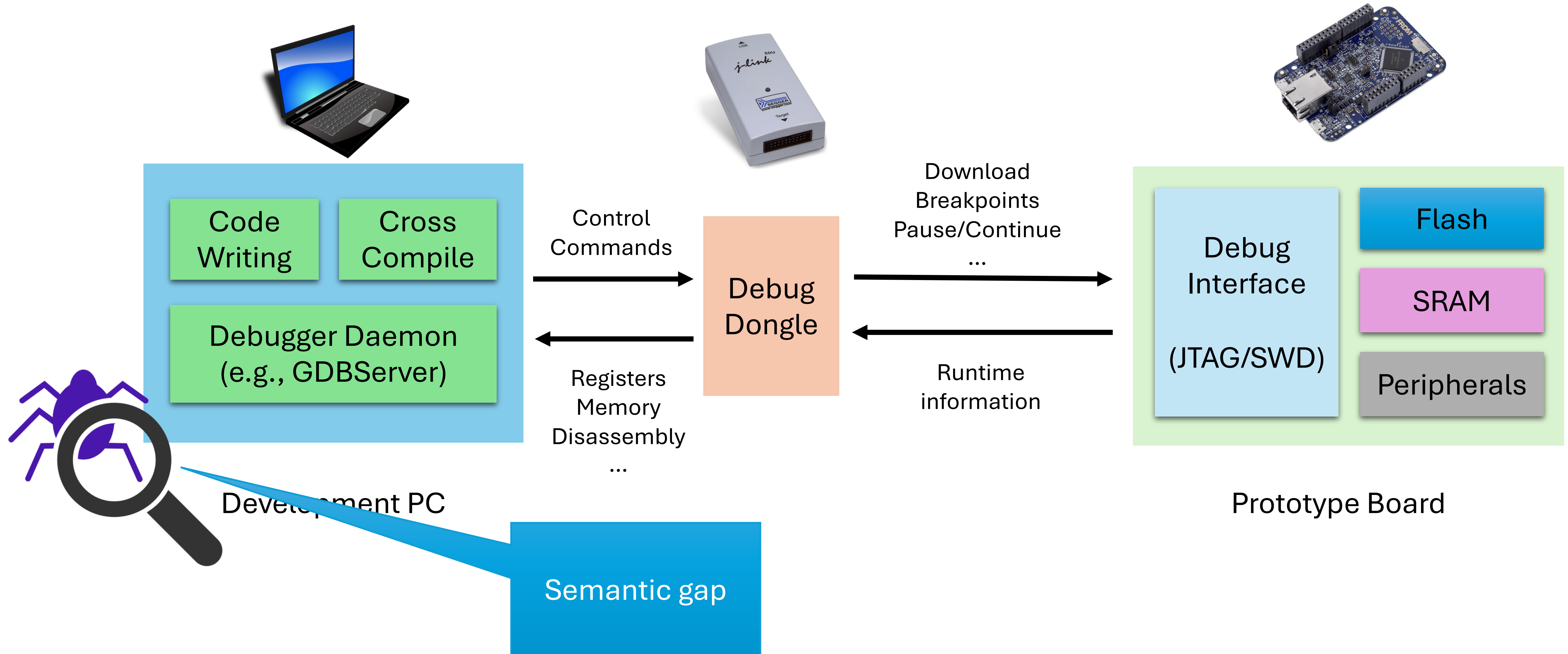
SRAM

Peripherals

Prototype Board

Insufficient resources to run analysis

Analysis on Development Workstation?



Analysis on Development Workstation?

Code Writing

Cross Compile

Debugger D mon
(e.g., GDBS ver)

Develo m

Control Commands →

← Registers Memory Disassembly …

Debug Dongle

Download Breakpoints Pause/Continue … →

← Runtime information

Debug Interface

(JTAG/SWD)

Flash

SRAM

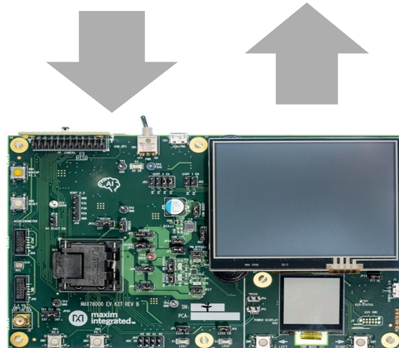
Peripherals

Prototype Board

Semantic gap

State-of-the-Art Solutions

```
american fuzzy lop ++4.07a {default} (./opencv_test_wechat_qrcode) [fast]
 process timing                                      overall results
        run time : 0 days, 0 hrs, 1 min, 21 sec      cycles done : 0
   last new find : 0 days, 0 hrs, 0 min, 7 sec       corpus count : 40
last saved crash : none seen yet                     saved crashes : 0
 last saved hang : none seen yet                      saved hangs : 0
 cycle progress                              map coverage
  now processing : 23.1 (57.5%)                map density : 8.45% / 15.91%
  runs timed out : 0 (0.00%)                count coverage : 3.36 bits/tuple
 stage progress                              findings in depth
  now trying : splice 8                      favored items : 21 (52.50%)
 stage execs : 10/12 (83.33%)                 new edges on : 26 (65.00%)
 total execs : 905                           total crashes : 0 (0 saved)
  exec speed : 7.89/sec (zzzz...)             total tmouts : 0 (0 saved)
 fuzzing strategy yields                         item geometry
   bit flips : disabled (default, enable with -D)    levels : 2
  byte flips : disabled (default, enable with -D)   pending : 38
 arithmetics : disabled (default, enable with -D)  pend fav : 20
  known ints : disabled (default, enable with -D) own finds : 4
  dictionary : n/a                                 imported : 0
havoc/splice : 3/108, 1/276                       stability : 100.00%
py/custom/rq : unused, unused, unused, unused
    trim/eff : 0.00%/237, disabled                         [cpu000:  2%]
                                               ^C
```



- µAFL [ICSE'22]
- GDBFuzz [ISSTA’23]
- Idea:
  - Run the firmware on device
  - Use special hardware to collect execution information
  - Stream the collected info on PC

- Limitations:
  - Limited to fuzzing
    - Only collect code coverage
  - Require hardware features
    - ETM, breakpoint

IPEA: In-vivo Probe, Ex-vivo Analysis

Non-intrusive

- IPEA seamlessly integrates into existing firmware development workflow and does not rely on additional hardware

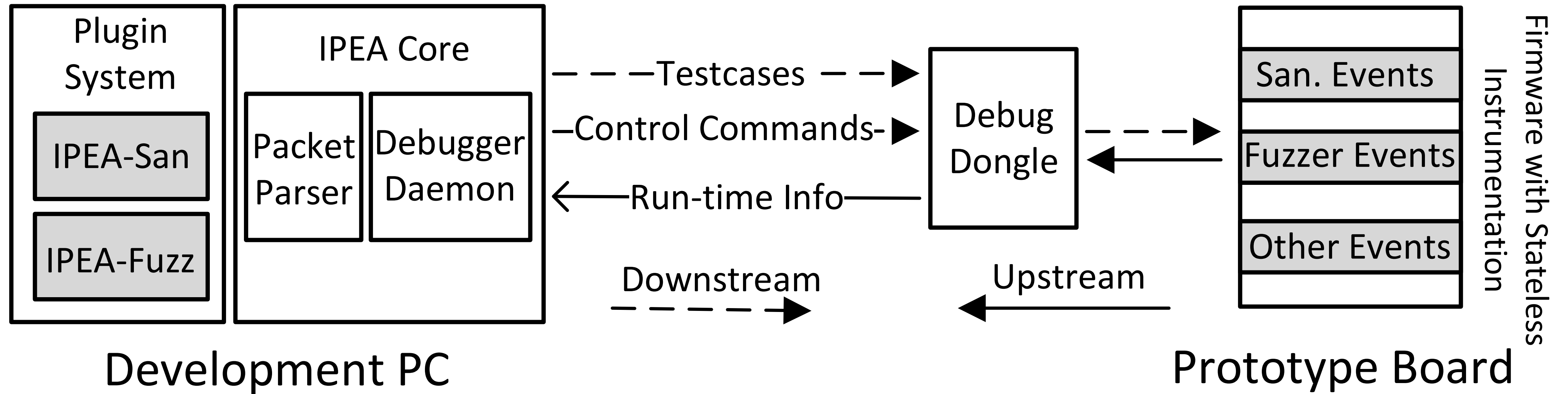
In-vivo

- The instrumented firmware collects arbitrary run-time information essential for firmware analysis, without dependent on special hardware

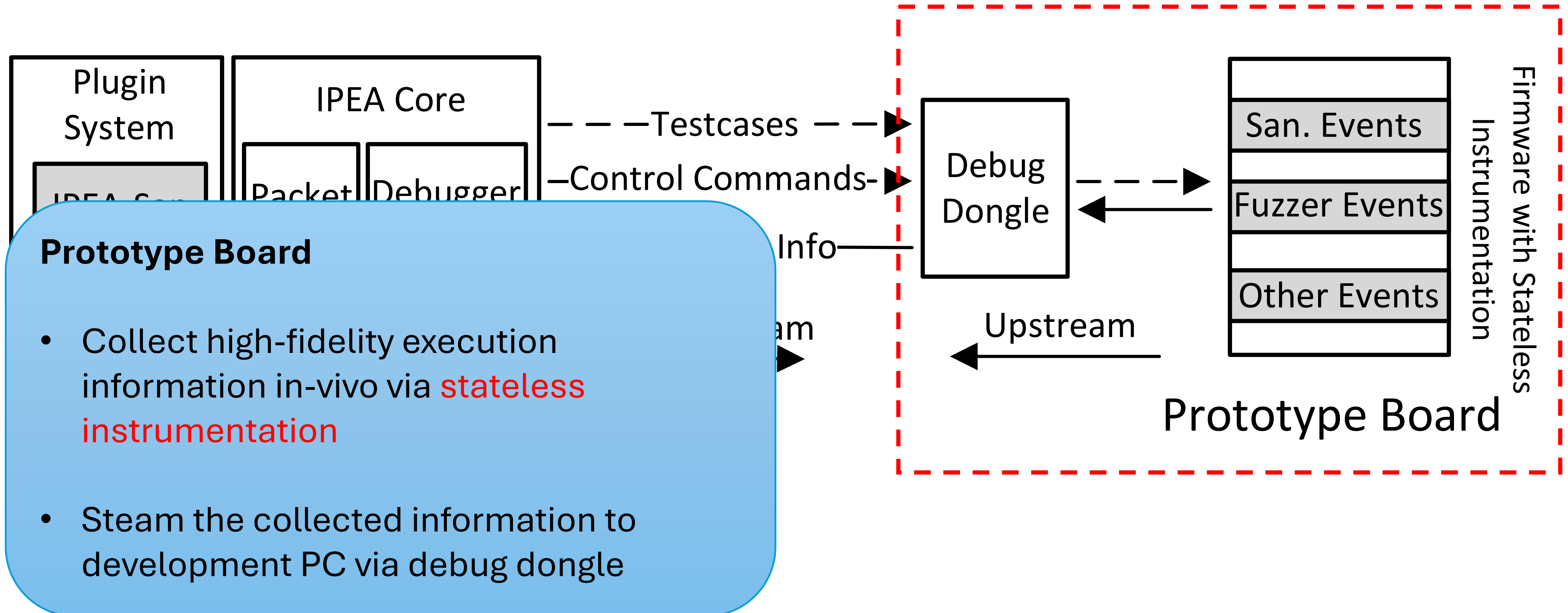
Lightweight

- Offload analysis to the PC, leaving the instrumentation on the device lightweight

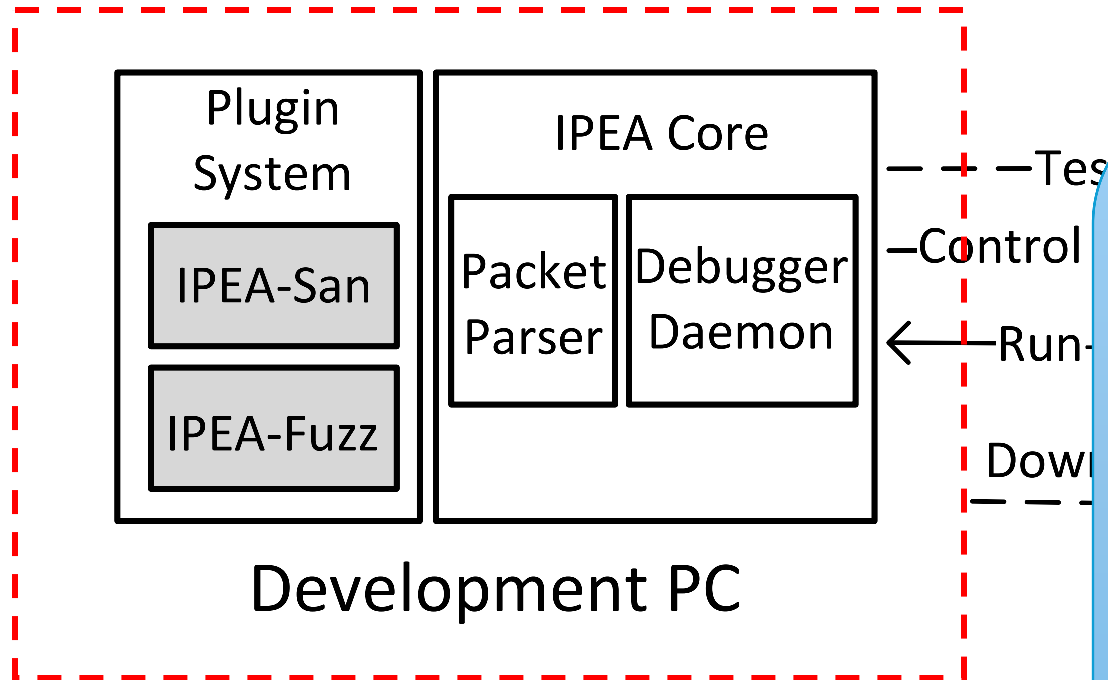
Design



Design



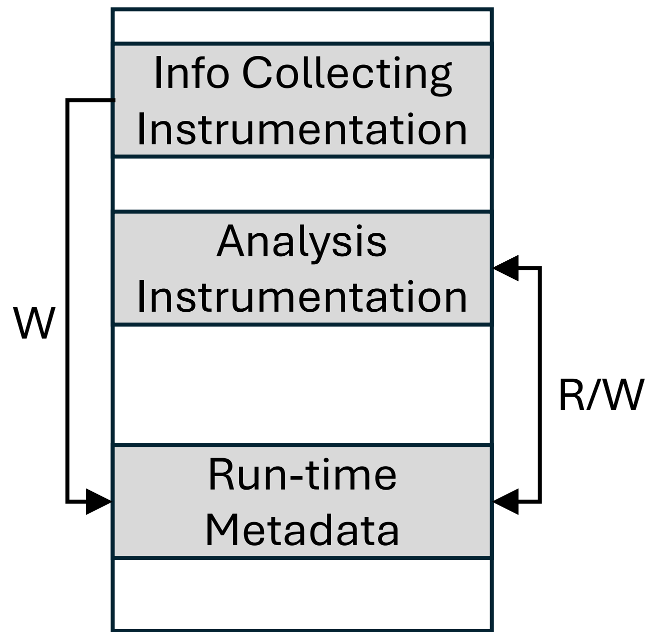
Design



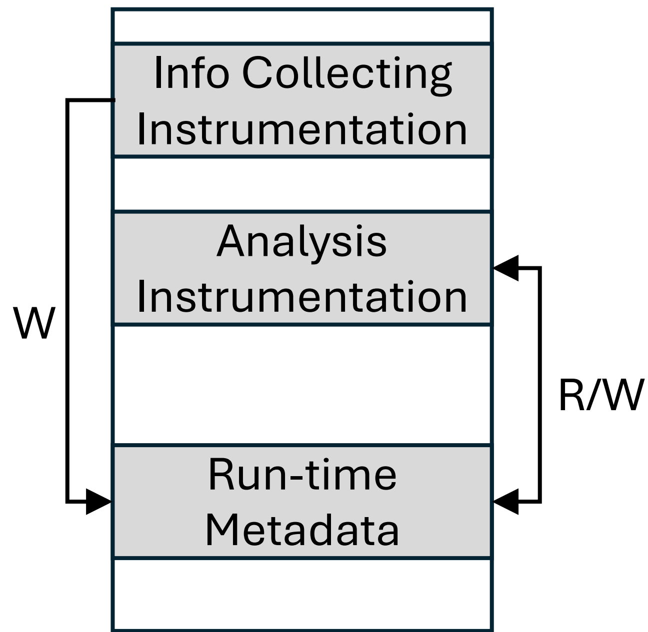
Development PC

- **IPEA-Core**: Download testcases to the target device, parse the collected information and deliver to the intended plugin
- **Analysis plugins**
  - IPEA-San: a pointer-based sanitizer
  - IPEA-Fuzz: a graybox fuzzer with edge coverage as feedback

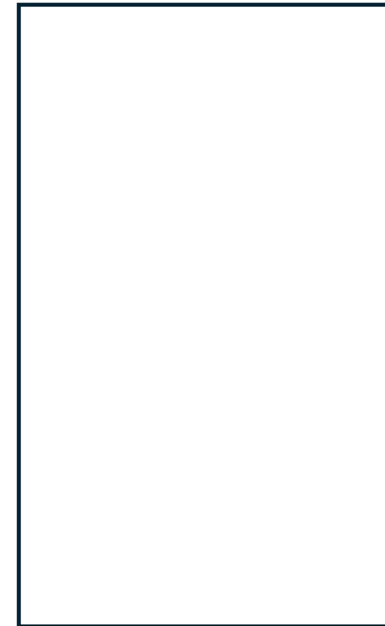
Stateless Instrumentation



Stateless Instrumentation

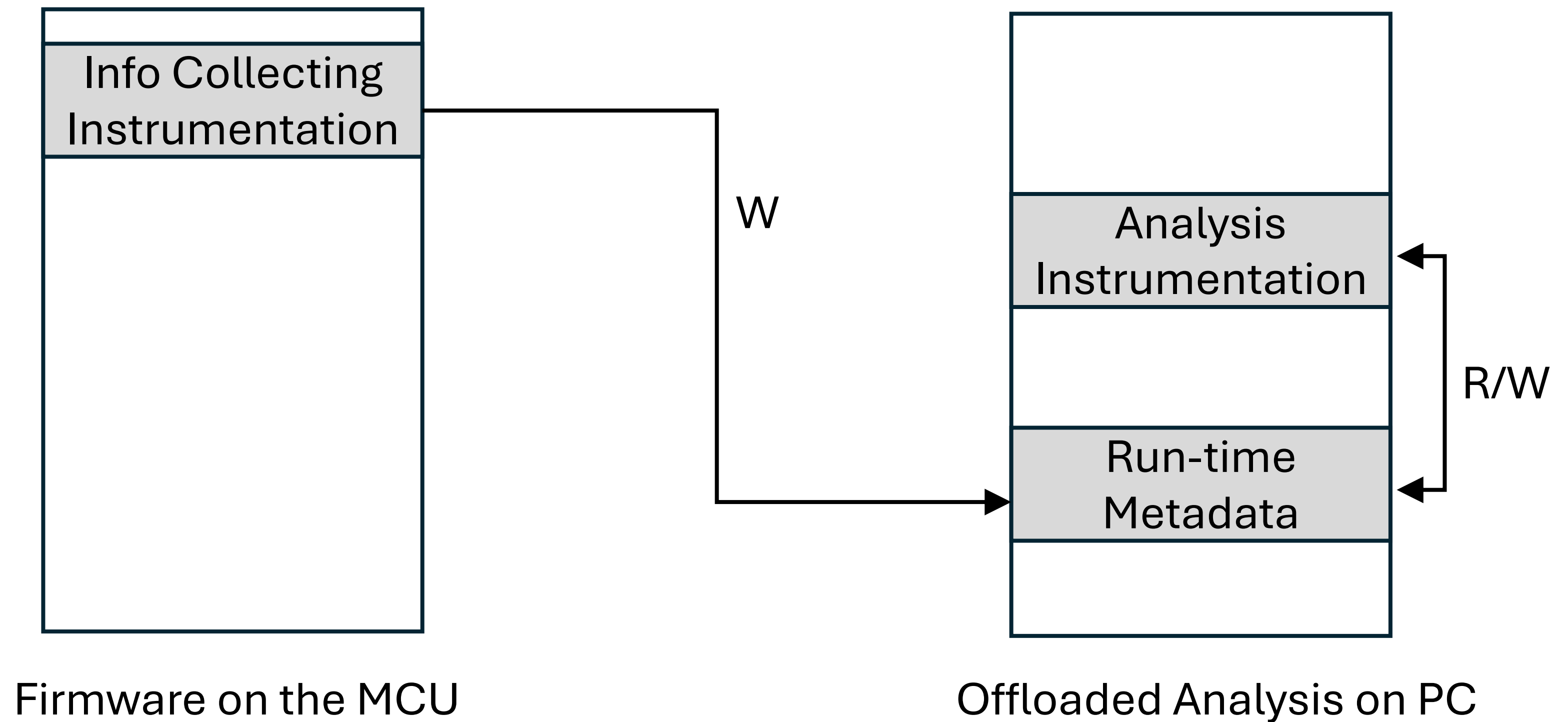


Firmware on the MCU



Offloaded Analysis on PC

Stateless Instrumentation



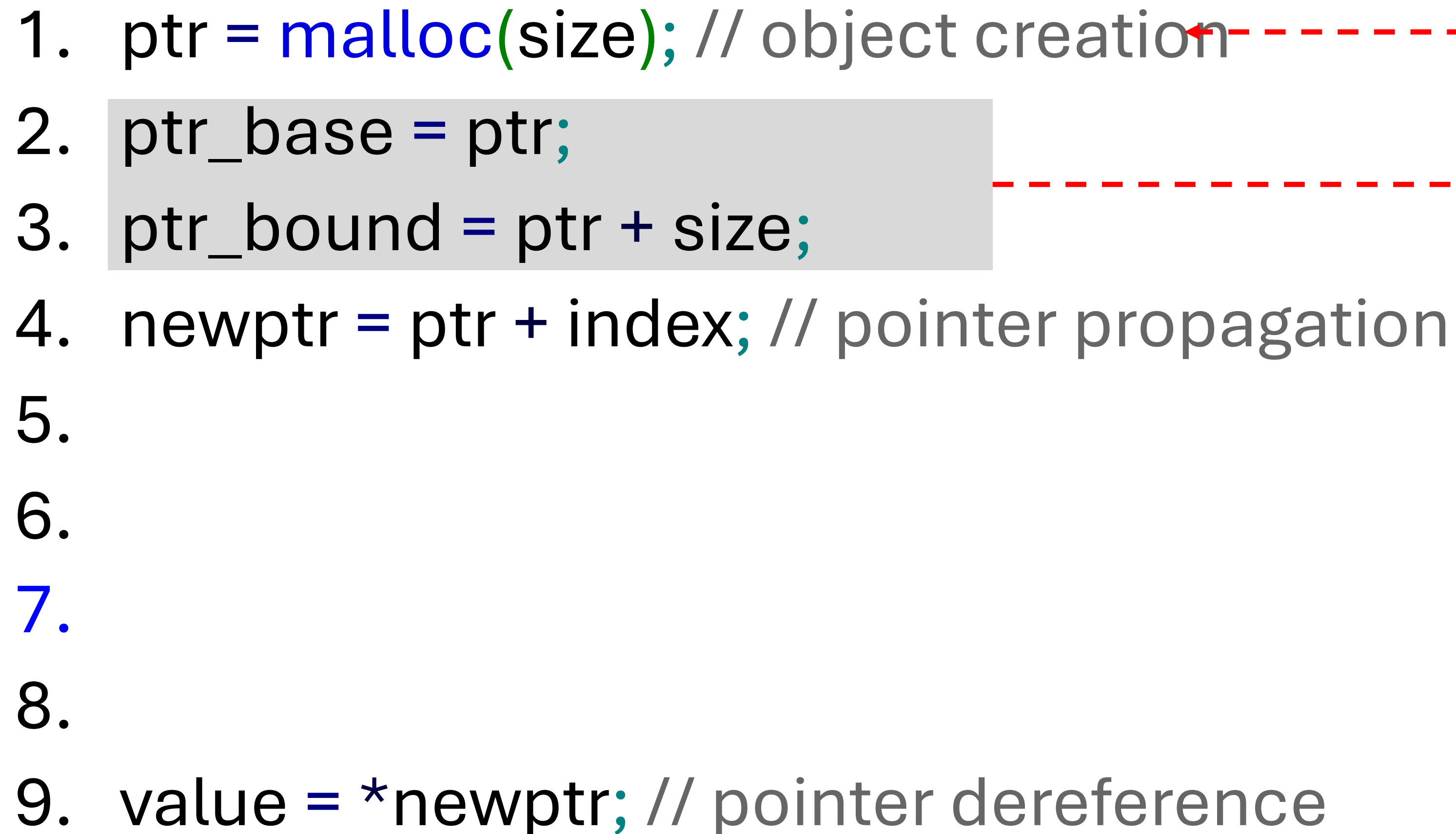
IPEA-San Plugin

- Pointer-based sanitizer
  - Avoid shadow memory
  - Each pointer is associated with metadata of its capability (bounds and validity)
  - E.g., SoftBound [PLDI'09]
- Memory tagging
  - A variant of pointer- based sanitizer
    - When an object is allocated, its memory and receiving pointer are given the same tag
    - All accesses to that memory must be made by a pointer having the same tag
  - E.g., Arm MTE
- We virtually extend the capability of MCUs by emulating an enhanced Arm MTE
  - Cooperation between the instrumented firmware and IPEA-San plugin on PC
  - Support unlimited number of tags compared with Arm MTE

Running Example – w/o IPEA-San

```
1.  ptr = malloc(size); // object creation
2.
3.
4.  newptr = ptr + index; // pointer propagation
5.
6.
7.
8.
9.  value = *newptr; // pointer dereference
```

Running Example – w/o IPEA-San



Running Example – w/o IPEA-San

```
1. ptr = malloc(size); // object creation
2. ptr_base = ptr;
3. ptr_bound = ptr + size;
4. newptr = ptr + index; // pointer propagation
5. newptr_base = ptr_base;
6. newptr_bound = ptr_bound;
7.
8.
9. value = *newptr; // pointer dereference
```

Running Example – w/o IPEA-San

```
1. ptr = malloc(size); // object creation
2. ptr_base = ptr;
3. ptr_bound = ptr + size;
4. newptr = ptr + index; // pointer propagation
5. newptr_base = ptr_base;
6. newptr_bound = ptr_bound;
7. if ((newptr < newptr_base) || (newptr+sizeof(*newptr) > bound))
8.   abort();
9. value = *newptr; // pointer dereference
```

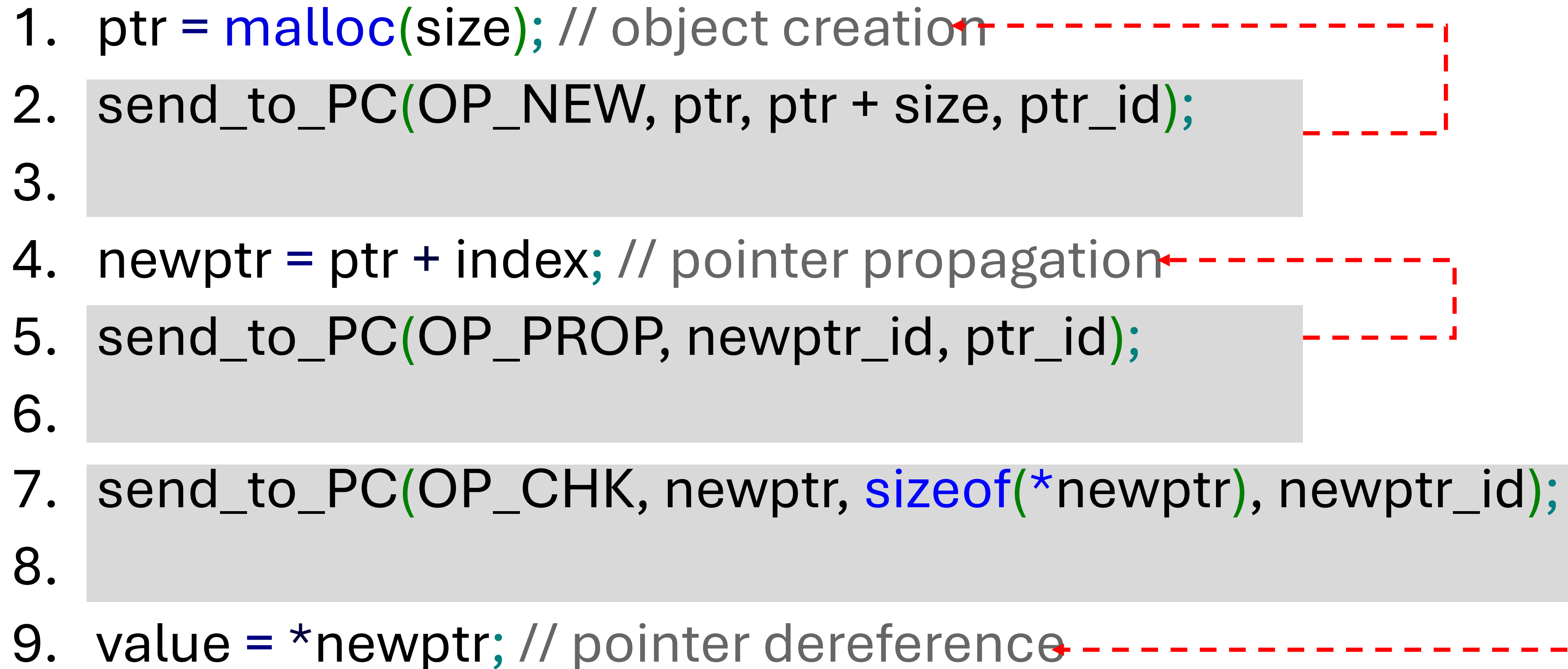
Running Example – w/o IPEA-San

```
1. ptr = malloc(size); // object creation
2. ptr_base = ptr;
3. ptr_bound = ptr + size;
4. newptr = ptr + index; // pointer propagation
5. newptr_base = ptr_base;
6. newptr_bound = ptr_bound;
7. if ((newptr < newptr_base) || (newptr+sizeof(*newptr) > bound))
8.   abort();
9. value = *newptr; // pointer dereference
```

Need a table to store per-pointer metadata

Need on-device computation

Running Example – w/ IPEA-San



Running Example – w/ IPEA-San

```
1. ptr = malloc(size); // object creation
2. send_to_PC(OP_NEW, ptr, ptr + size, ptr_id);
3.
4. newptr = ptr + index; // pointer propagation
5. send_to_PC(OP_PROP, newptr_id, ptr_id);
6.
7. send_to_PC(OP_CHK, newptr, sizeof(*newptr), newptr_id);
8.
9. value = *newptr; // pointer dereference
```

Tables are maintained on the PC

Computation is done on the PC

Metadata Recovery and Maintenance on PC

```
ptr = malloc(size);
send_to_PC(OP_NEW, ptr, size, ptr_id);
newptr = ptr + index;
send_to_PC(OP_PROP, newptr_id, ptr_id);
send_to_PC(OP_CHK, newptr, sizeof(*newptr),\
           newptr_id);
value = *newptr;
```

- Each pointer is associated with a compiler-generated unique ID.
- Instrumentation streams out pointer operations to the PC
- On receiving pointer operations, the IPEA-San plugin on PC recovers the per-pointer metadata.

Metadata Recovery and Maintenance on PC

```
ptr = malloc(size);

newptr = ptr + index;



value = *newptr;
```

Pointer ID	Tag

ID-Tag Table

Address	Tag

Shadow Memory

Metadata Recovery and Maintenance on PC

```
ptr = malloc(size);
send_to_PC(OP_NEW, ptr, size, ptr_id);
newptr = ptr + index;



value = *newptr;
```

ID-Tag Table

Pointer ID	Tag
ptr_id	0x100

Shadow Memory

Address	Tag
ptr	0x100
ptr	0x100
ptr	0x100

Metadata Recovery and Maintenance on PC

```
ptr = malloc(size);
send_to_PC(OP_NEW, ptr, size, ptr_id);
newptr = ptr + index;
send_to_PC(OP_PROP, newptr_id, ptr_id);


value = *newptr;
```

ID-Tag Table

Pointer ID	Tag
ptr_id	0x100
newptr_id	0x100

Shadow Memory

Address	Tag
ptr	0x100
ptr	0x100
ptr	0x100

Metadata Recovery and Maintenance on PC

```
ptr = malloc(size);
send_to_PC(OP_NEW, ptr, size, ptr_id);
newptr = ptr + index;
send_to_PC(OP_PROP, newptr_id, ptr_id);
send_to_PC(OP_CHK, newptr, sizeof(*newptr), \
           newptr_id);
value = *newptr;
```

Pointer ID	Tag
ptr_id	0x100
newptr_id	0x100

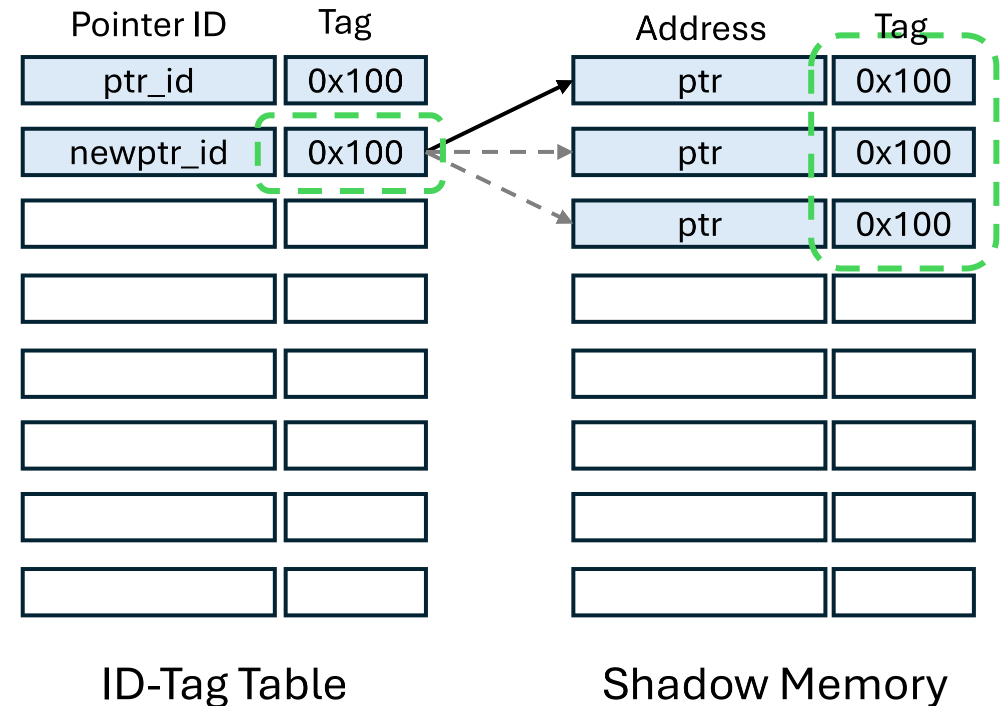
ID-Tag Table

Address	Tag
ptr	0x100
ptr	0x100
ptr	0x100

Shadow Memory

Metadata Recovery and Maintenance on PC

```
ptr = malloc(size);
send_to_PC(OP_NEW, ptr, size, ptr_id);
newptr = ptr + index;
send_to_PC(OP_PROP, newptr_id, ptr_id);
send_to_PC(OP_CHK, newptr, sizeof(*newptr), \
           newptr_id);
value = *newptr;
```



Metadata Recovery and Maintenance on PC

```
ptr = malloc(size);
send_to_PC(OP_NEW, ptr, size, ptr_id);
newptr = ptr + index;
send_to_PC(OP_PROP, newptr_id, ptr_id);
send_to_PC(OP_CHK, newptr, sizeof(*newptr), \
           newptr_id);
value = *newptr;
```

Pointer ID	Tag
ptr_id	0x100
newptr_id	0x100

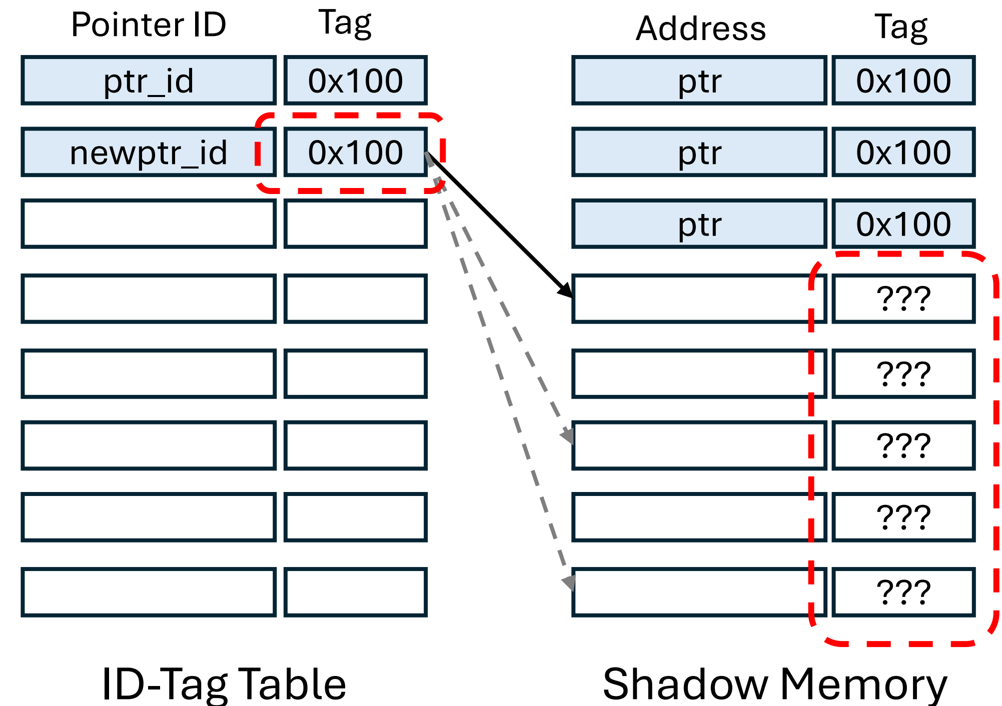
ID-Tag Table

Address	Tag
ptr	0x100
ptr	0x100
ptr	0x100

Shadow Memory

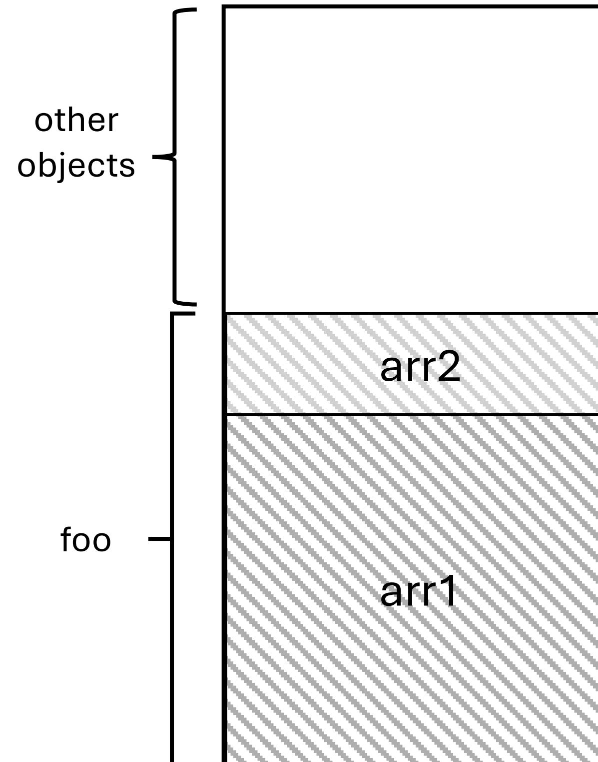
Metadata Recovery and Maintenance on PC

```
ptr = malloc(size);
send_to_PC(OP_NEW, ptr, size, ptr_id);
newptr = ptr + index;
send_to_PC(OP_PROP, newptr_id, ptr_id);
send_to_PC(OP_CHK, newptr, sizeof(*newptr), \
           newptr_id);
value = *newptr;
```



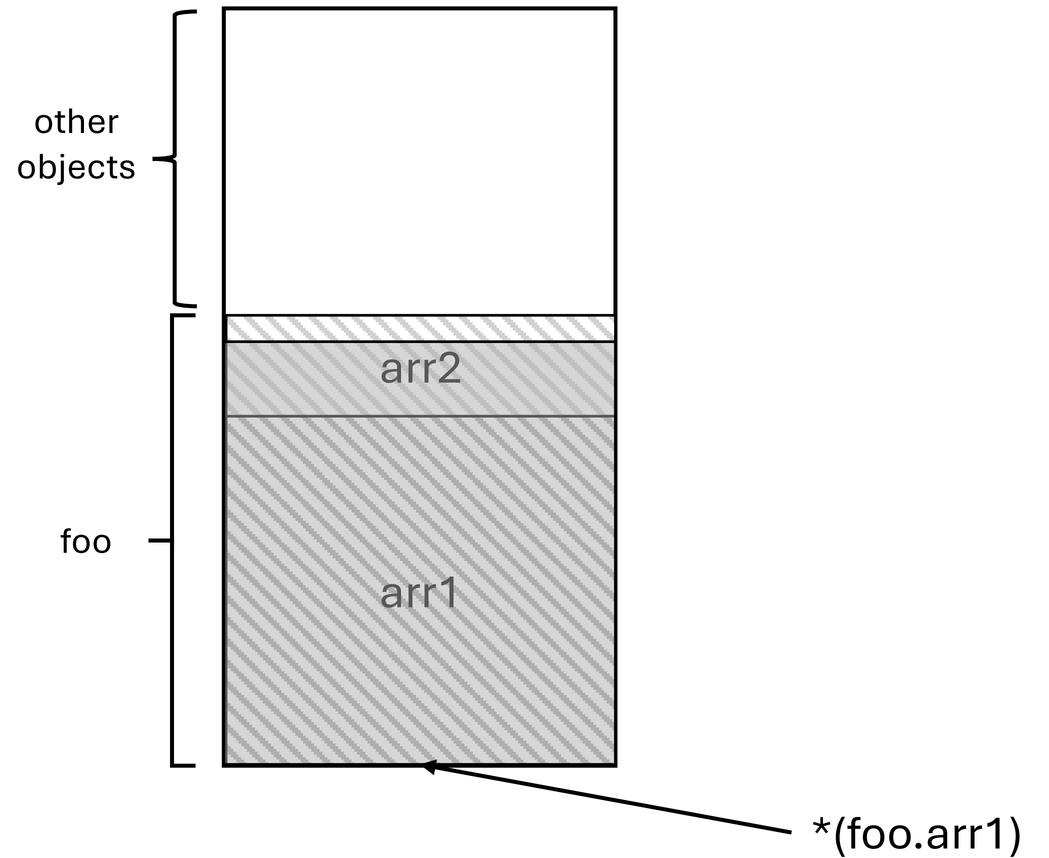
Intra-object Overflow

```
struct foo
{
    char arr1[…];
    char arr2[…];
};
```



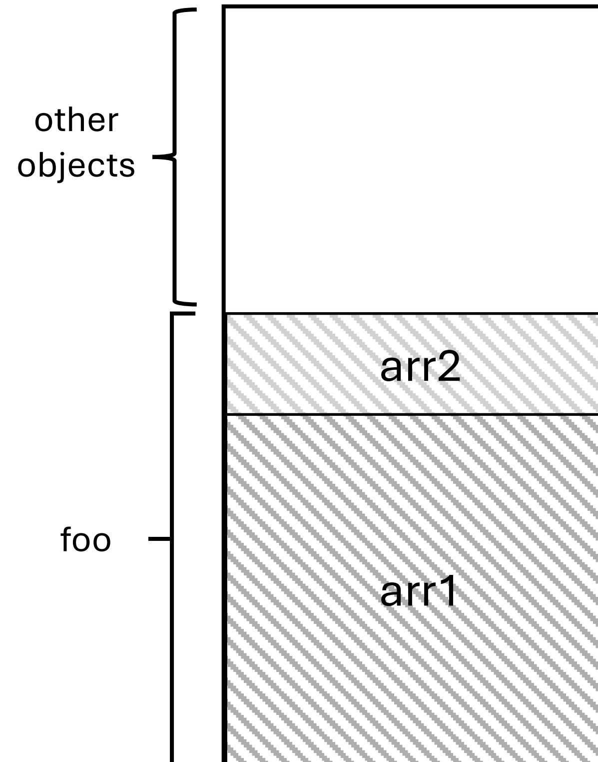
Intra-object Overflow Detection

```
struct foo
{
    char arr1[…];
    char arr2[…];
};
```



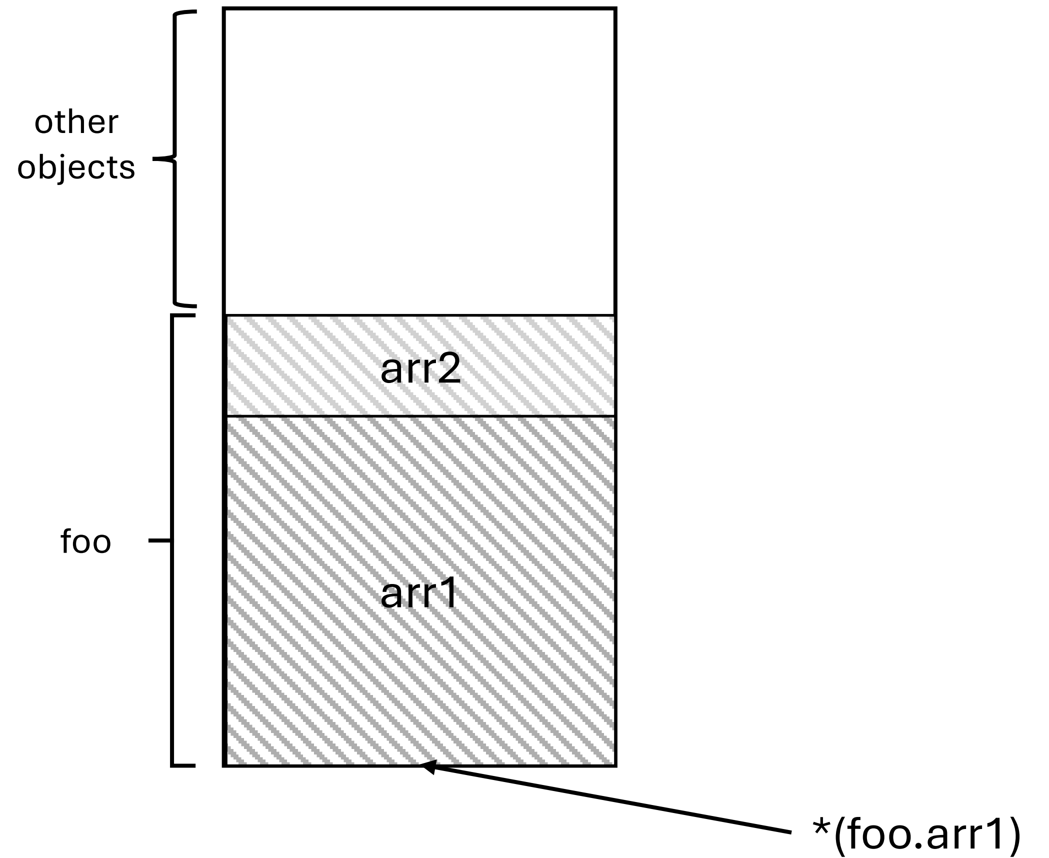
Intra-object Overflow Detection

```
struct foo
{
    char arr1[…];
    char arr2[…];
};
```



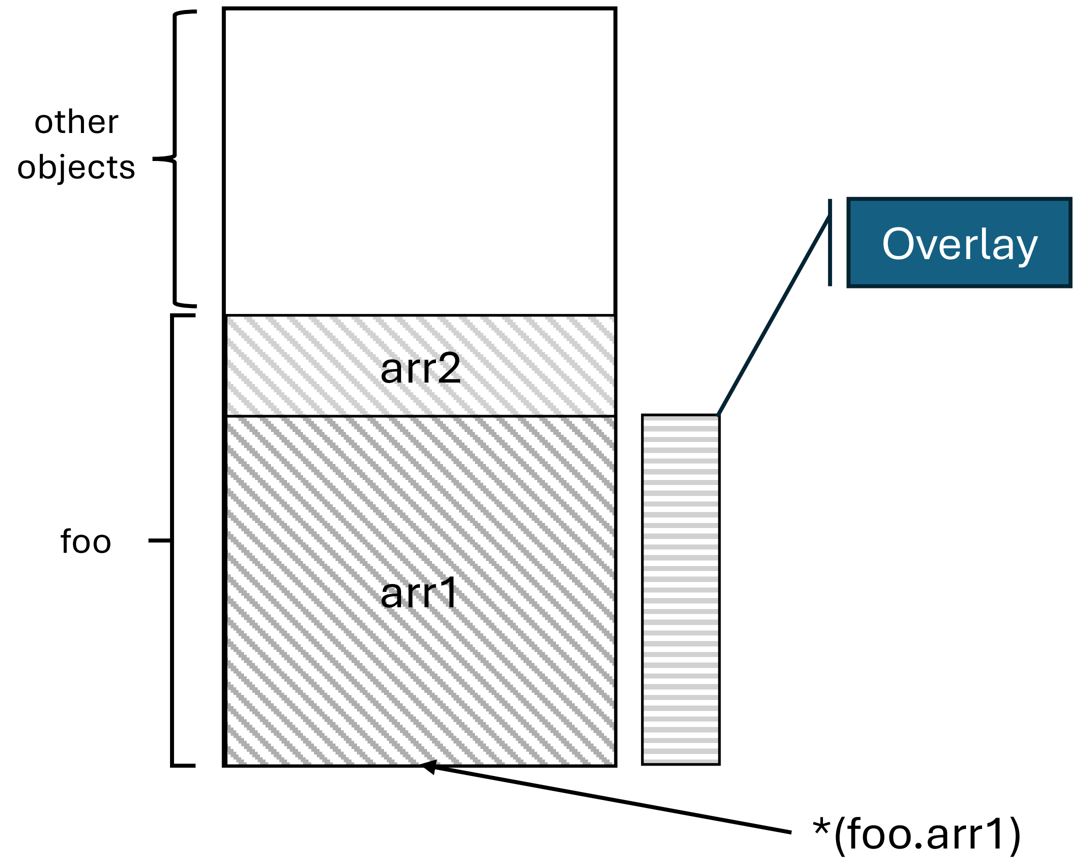
Tag Overlay

```
struct foo
{
    char arr1[…];
    char arr2[…];
};
```



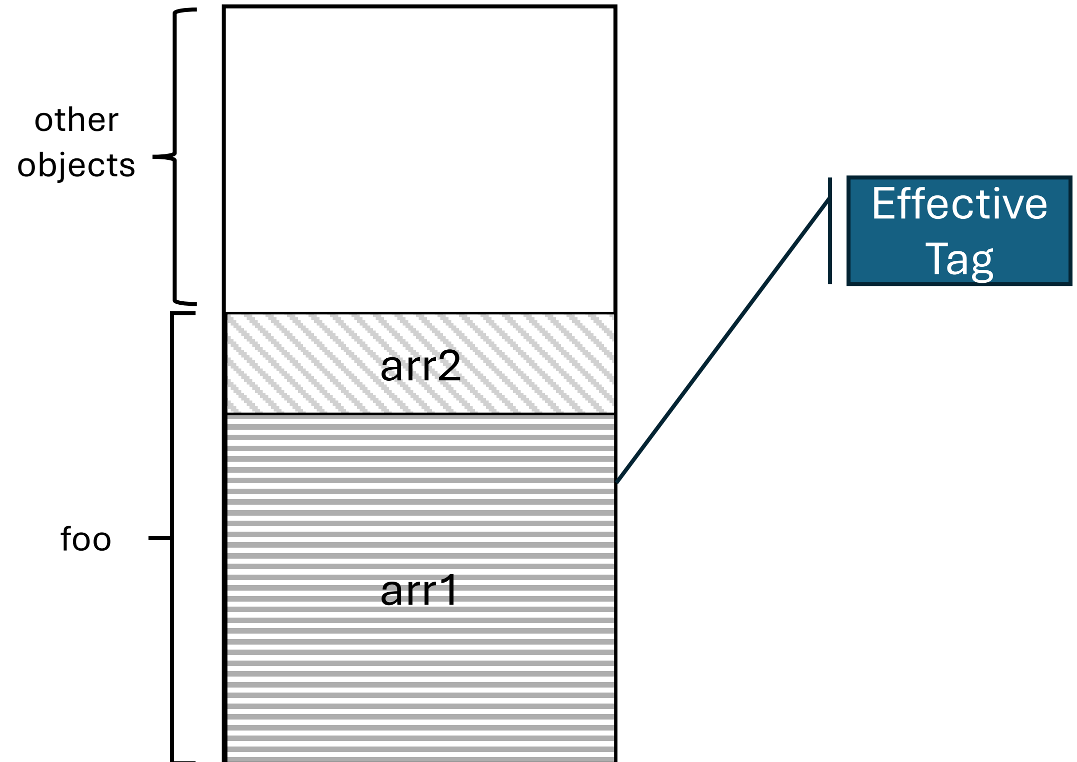
Tag Overlay

```
struct foo
{
    char arr1[…];
    char arr2[…];
};
```



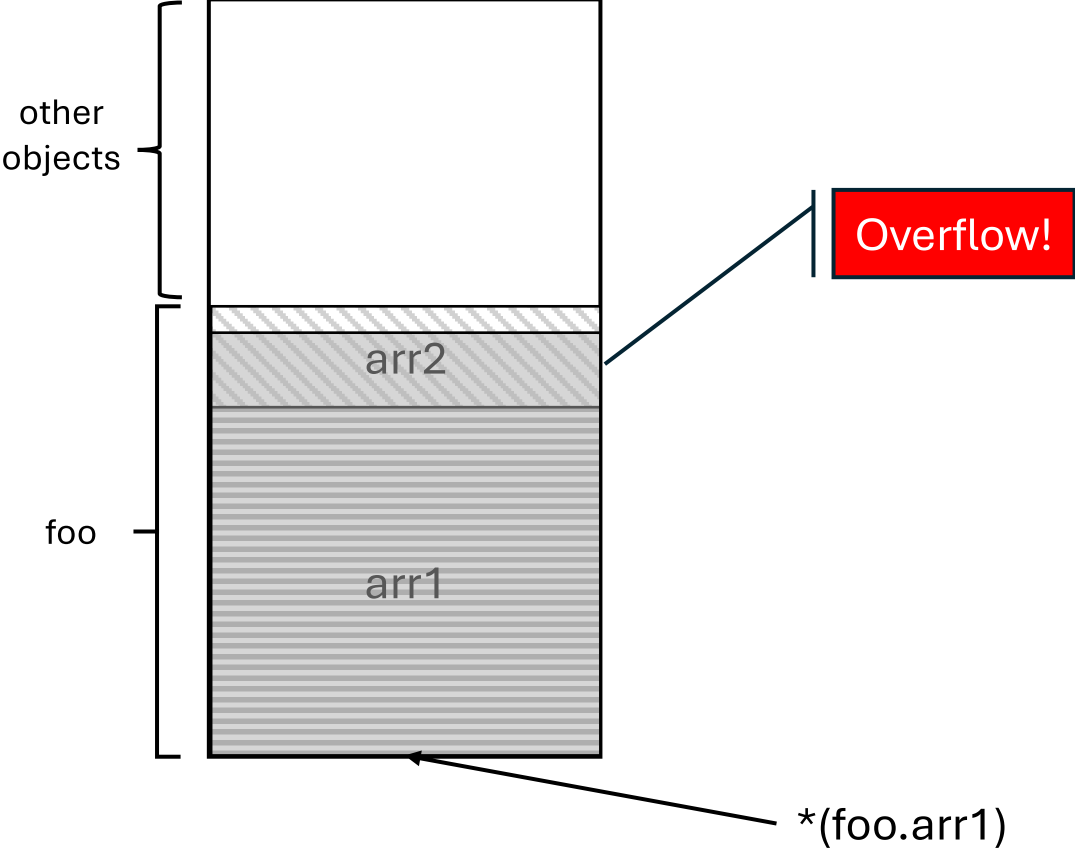
Tag Overlay

```
struct foo
{
    char arr1[…];
    char arr2[…];
};
```



Tag Overlay

```
struct foo
{
    char arr1[…];
    char arr2[…];
};
```



IPEA-Fuzz Plugin

- Firmware Instrumentation
  - A random number is generated at compile-time to identify a basic block.
  - The random numbers are sent to IPEA-Fuzz plugin at run-time.

- IPEA-Fuzz Plugin
  - Maintains a 64KB bitmap shared with AFL to record the number of hits for a particular edge.

Experiment Setup

- Tested prototype boards (7)
  - NXP FRDM-K64F (1MB flash / 256KB SRAM)
  - NXP FRDM-K66F (2MB flash / 256KB SRAM)
  - NXP LPCXpresso55S69 (640KB flash / 320KB SRAM)
  - STM32-NucleoF411R (512KB flash / 128KB)
  - STM32H7B3I-DK (1MB flash / 1MB SRAM)
  - Raspberry Pi Pico (2MB flash / 264KB SRAM)
  - nRF52-DK (192KB flash / 24KB SRAM)
- Debug dongle
  - SGGER J-Link Edu

Evaluation

- **RQ1**: What kind of memory errors can be captured by IPEA-San?
- **RQ2**: Can IPEA-San reduce resource consumption on MCUs?
- **RQ3**: Can the combination of IPEA-San and IPEA-Fuzz find bugs?

RQ1: IPEA-San Capability

CWE Index	Description	# Tests
CWE 121	Stack-based Buffer Overflow	2,432
CWE 122	Heap-based Buffer Overflow	1,594
CWE 124	Buffer Underwrite	682
CWE 126	Buffer Overread	524
CWE 127	Buffer Underread	682
CWE 415	Double Free	190
CWE 416	Use After Free	118
CWE 476	NULL Pointer Dereference	234
CWE 761	Invalid Heap Pointer Free	152

Selected Testcases in Juliet Testsuite

RQ1: IPEA-San Capability

CWE Index	IPEA-San		ASan	
	False Negative (%)	False Positive (%)	False Negative (%)	False Positive (%)
CWE 121	0	0	1,220 (50.16%)	0
CWE 122	0	0	42 (2.26%)	0
CWE 124	0	0	170 (24.92%)	0
CWE 126	0	0	179 (34.16%)	0
CWE 127	0	0	170 (24.92%)	0
CWE 415	0	0	0	0
CWE 416	0	0	0	0
CWE 476	0	0	0	0
CWE 761	0	0	0	0
Total	**0**	**0**	**1,781 (26.95%)**	**0**

RQ2: IPEA-San Overhead

Firmware	Baseline					*IPEA-San*						ASan					
	Flash (Bytes)	Stack (Bytes)	Heap (Bytes)	Global (Bytes)	Total RAM (Bytes)	Flash (x)	Stack (x)	Heap (x)	Global (x)	RTT† (Bytes)	Total RAM (x)	Flash (x)	Stack (x)	Heap (x)	Global (x)	Shadow‡ (Bytes)	Total RAM (x)
aha-compress	4,356	112	-	184	296	1.81	1.14	-	1.00	1,076	4.69	1.94	3.00	-	1.63	2,085	9.19
ctl-stack	4,796	120	816	204	1,140	2.12	1.13	1.00	1.00	1,076	1.96	1.91	2.80	3.13	3.92	2,148	5.12
ctl-string	8,076	104	232	212	548	1.90	1.00	1.00	1.00	1,076	2.97	1.62	2.00	3.03	5.83	2,202	7.94
frac	7,456	240	-	268	508	1.46	1.03	-	1.00	1,076	3.14	1.52	1.60	-	1.64	2,103	5.76
huffbench	4,960	7,824	1,001	188	9,013	2.08	1.01	1.00	1.00	1,076	1.13	2.00	1.13	4.60	2.17	2,099	1.77
sglib-hashtable	6,408	224	800	664	1,688	2.12	1.00	1.00	1.00	1,076	1.64	1.63	2.07	2.14	1.39	2,163	3.12
Geo. Mean	-	-	-	-	-	**1.90**	-	-	-	-	**2.32**	**1.76**	-	-	-	-	**4.78**
PinLock	20,956	416	-	760	1,176	1.67	1.06	-	1.00	1,076	1.94	3.88	1.81	-	72.33	8,919	54.97
CNC	74,800	344	-	12,708	13,052	1.44	1.17	-	1.00	1,076	1.09	1.37	1.51	-	2.00	3,684	2.27
nRF52-Keyboard	52,428	478	-	6,224	6,702	2.34	1.05	-	1.09	1,076	1.25	1.70	2.17	-	1.80	1,540	2.05
ClockAndWeather	286,184	732	5,456	179,164	185,360	1.21	2.03	1.00	1.00	1,076	1.01	1.30	1.57	1.01	1.84	45,390	2.06
AudioPlayer	101,832	872	-	13,992	14,864	1.56	1.11	-	1.07	1,076	1.14	1.39	2.26	-	4.01	7,272	4.40
WeighScale	24,052	768	-	21,060	21,828	2.79	1.66	-	1.02	1,076	1.01	2.07	2.32	-	2.86	7,754	3.19
HttpServer*	71,932	840	-	65,408	66,248	2.67	1.05	-	1.01	1,076	1.03	-	-	-	-	-	-
U-Disk*	33,356	188	12,168	44,632	56,988	2.75	1.96	1.00	1.00	1,076	1.02	-	-	-	-	-	-
MQTT-Echo*	63,976	632	10,436	59,020	70,088	2.56	1.40	1.00	1.00	1,076	1.02	-	-	-	-	-	-
App-Scheduling	355,108	124	-	7,764	7,888	1.10	1.22	-	1.00	1,076	1.13	1.09	1.42	-	1.32	1,250	1.48
App-Timers	346,816	112	-	7,696	7,808	1.11	1.21	-	1.00	1,076	1.14	1.08	1.42	-	1.32	1,250	1.48
App-IRQs	356,588	108	-	7,800	7,908	1.12	1.17	-	1.00	1,076	1.14	1.07	1.41	-	1.29	1,250	1.45
Geo. Mean	-	-	-	-	-	**1.74**	-	-	-	-	**1.14**	**1.52**	-	-	-	-	**3.06**

†: RTT buffer plus other metadata. ‡: Shadow memory plus other metadata. *: Failed to compile using ASan.

RQ3: IPEA-Fuzz Evaluation

Firmware	Target MCU	Time (s)	# Execution	Exec/sec	Paths	Crashes/Hangs
Toy	K64F	7,200	1,990,296	276.43	9	8/0
Expat XML	STM32H7	86,400	1,539,648	17.82	1,098	5/0
JPEGDEC	STM32H7	83,752	805,308	9.61	903	17/4
PNGdec	STM32H7	84,011	848,512	10.10	1,001	22/10
UART	K64F	86,471	537,849	6.22	22	0/0
WiFi	STM32H7	151,212	32,218	0.21	153	1/5
USB Host	K64F	327,910	127,884	0.39	99	22/47
USB Host	STM32H7	347,774	141,277	0.45	96	0/57
emUSB-Host	LPC55S69	122,452	87,619	0.88	106	3/23
microSD	K64F	86,457	30,240	0.35	77	0/6

RQ3: IPEA-Fuzz Evaluation

Firmware	Target MCU	Time (s)	# Execution	Exec/sec	Paths	Crashes/Hangs
Toy	K64F	7,200	1,990,296	276.43	9	8/0
Expat XML	STM32H7	86,400	1,539,648	17.82	1,098	5/0
JPEGDEC	STM32H7	83,752	805,308	9.61	903	17/4
PNGdec	STM32H7	84,011	848,512	10.10	1,001	22/10
UART	K64F	86,471	537,849	6.22	22	0/0
WiFi	STM32H7	151,212	32,218	0.21	153	1/5
USB Host	K64F	327,910	127,884	0.39	99	22/47
USB Host	STM32H7	347,774	141,277	0.45	96	0/57
emUSB-Host	LPC55S69	122,452	87,619	0.88	106	3/23
microSD	K64F	86,457	30,240	0.35	77	0/6

Found 7 zero-day bugs:

- 3 in popular IoT libraries
- 4 in peripheral driver code, including a commercial product

Conclusions

- We present the design and implementation of IPEA firmware analysis framework and two analysis plugins for it.
- They seamlessly integrate into existing firmware development environments, allowing developers to run advanced firmware testing while developing firmware.
- By offloading analysis to the development PCs, the proposed analysis techniques significantly reduce memory overhead compared with solutions that run entirely on MCUs.

Thank you!!

jiameng@uga.edu

https://github.com/MCUSec/IPEA