ILLINOIS
Information Trust Institute

Slide-deck for
Graduate Computer Security
(CS563)
Fall 2018
University of Illinois
Prof. Adam Bates

Securing Real-Time Microcontroller Systems through Customized Memory View Switching

ILLINOIS
Information Trust Institute

What are Real-time Microcontrollers?

- Application-specific compute unit.
  - Lack of resources (especially RAM)
  - Low Power
  - Lower footprint e.g., kilobytes vs. megabytes.
  - Runs light-weight OS'es
  - E.g., Automotive, Consumer Electronics, Sensor Networks

Atmel AVR

AVR

ATX Mega

ATmega 328P

PIC 18F877A

8051

Arduino

ARM

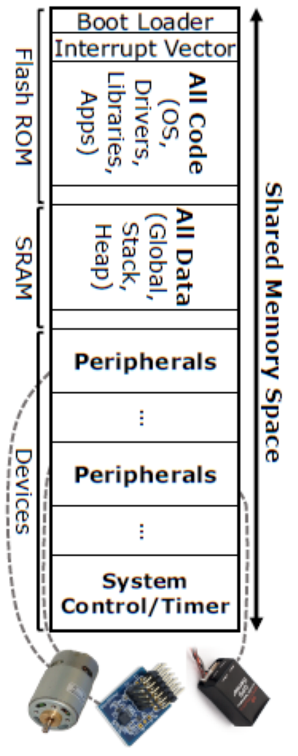
Features

- High resolution clocks and timers
  - RT application development requires support of timer services with resolution ~ µs
- Static Priority Levels
  - RMA & EDF assume static priorities
  - Dynamic Priorities are used for better response to tasks blocked on I/O
- Fast task Preemption
  - High priority task on arrival should instantly get CPU from low priority task
- Predictable and Fast Interrupt Latency
  - Bottom Half

Features

- Memory management Requirements
  - Memory Isolation(MI) : Process MI & Kernel MI
  - No Process MI
    - Lack of Virtual Memory support
      - Worst-case vs. Average (t_{mem})
  - No Kernel Memory Isolation
    - Lack of privilege separation
    - Context switching leads to performance issues.
      - Save memory bits
      - Light-weight system call

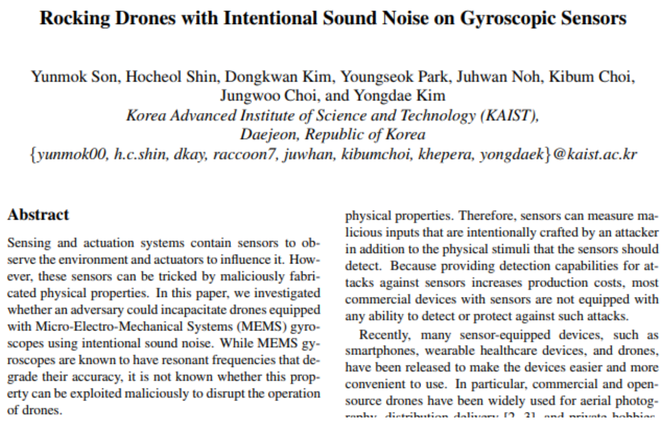
SECURITY?



Memory Protection Units(MPU)

- Hardware-based memory isolation
  - Can be enabled in the firmware of the microcontroller

- Access control rules for memory regions(usually 8 to 16)
  - If a memory region is accessed without the required permission, the processor raises a *Protection Fault*



Rocking Drones with Intentional Sound Noise on Gyroscopic Sensors

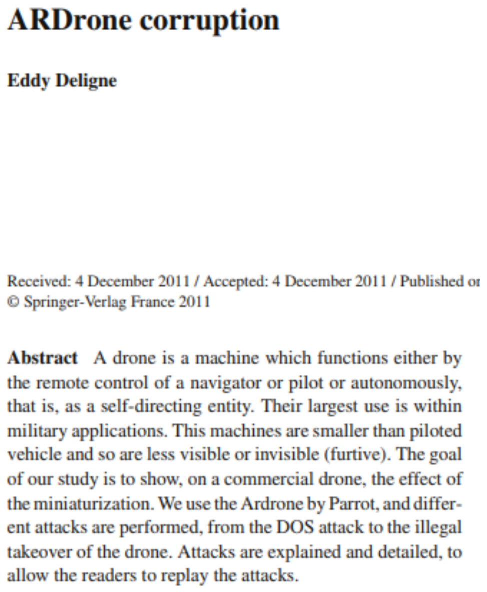
Yunmok Son, Hocheol Shin, Dongkwan Kim, Youngseok Park, Juhwan Noh, Kibum Choi, Jungwoo Choi, and Yongdae Kim
Korea Advanced Institute of Science and Technology (KAIST), Daejeon, Republic of Korea
{yunmok00, h.c.shin, dkay, raccoon7, juwhan, kibumchoi, khepera, yongdaek}@kaist.ac.kr

Abstract

Sensing and actuation systems contain sensors to observe the environment and actuators to influence it. However, these sensors can be tricked by maliciously fabricated physical properties. In this paper, we investigated whether an adversary could incapacitate drones equipped with Micro-Electro-Mechanical Systems (MEMS) gyroscopes using intentional sound noise. While MEMS gyroscopes are known to have resonant frequencies that degrade their accuracy, it is not known whether this property can be exploited maliciously to disrupt the operation of drones.

physical properties. Therefore, sensors can measure malicious inputs that are intentionally crafted by an attacker in addition to the physical stimuli that the sensors should detect. Because providing detection capabilities for attacks against sensors increases production costs, most commercial devices with sensors are not equipped with any ability to detect or protect against such attacks.

Recently, many sensor-equipped devices, such as smartphones, wearable healthcare devices, and drones, have been released to make the devices easier and more convenient to use. In particular, commercial and open-source drones have been widely used for aerial photog-

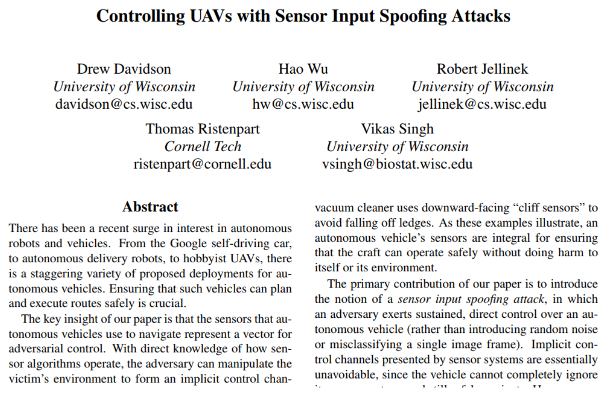


ARDrone corruption

Eddy Deligne

Received: 4 December 2011 / Accepted: 4 December 2011 / Published or
© Springer-Verlag France 2011

Abstract A drone is a machine which functions either by the remote control of a navigator or pilot or autonomously, that is, as a self-directing entity. Their largest use is within military applications. This machines are smaller than piloted vehicle and so are less visible or invisible (furtive). The goal of our study is to show, on a commercial drone, the effect of the miniaturization. We use the Ardrone by Parrot, and different attacks are performed, from the DOS attack to the illegal takeover of the drone. Attacks are explained and detailed, to allow the readers to replay the attacks.



Controlling UAVs with Sensor Input Spoofing Attacks

Drew Davidson, *University of Wisconsin*, davidson@cs.wisc.edu
Hao Wu, *University of Wisconsin*, hw@cs.wisc.edu
Robert Jellinek, *University of Wisconsin*, jellinek@cs.wisc.edu
Thomas Ristenpart, *Cornell Tech*, ristenpart@cornell.edu
Vikas Singh, *University of Wisconsin*, vsingh@biostat.wisc.edu

Abstract

There has been a recent surge in interest in autonomous robots and vehicles. From the Google self-driving car, to autonomous delivery robots, to hobbyist UAVs, there is a staggering variety of proposed deployments for autonomous vehicles. Ensuring that such vehicles can plan and execute routes safely is crucial.

The key insight of our paper is that the sensors that autonomous vehicles use to navigate represent a vector for adversarial control. With direct knowledge of how sensor algorithms operate, the adversary can manipulate the victim's environment to form an implicit control chan-

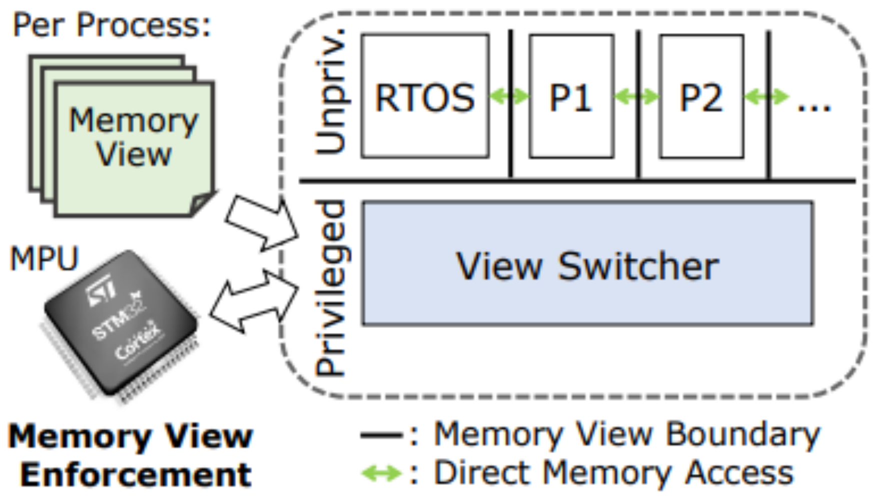
vacuum cleaner uses downward-facing “cliff sensors” to avoid falling off ledges. As these examples illustrate, an autonomous vehicle's sensors are integral for ensuring that the craft can operate safely without doing harm to itself or its environment.

The primary contribution of our paper is to introduce the notion of a *sensor input spoofing attack*, in which an adversary exerts sustained, direct control over an autonomous vehicle (rather than introducing random noise or misclassifying a single image frame). Implicit control channels presented by sensor systems are essentially unavoidable, since the vehicle cannot completely ignore

Attacks are (way too) common!

Security Solution : MINION

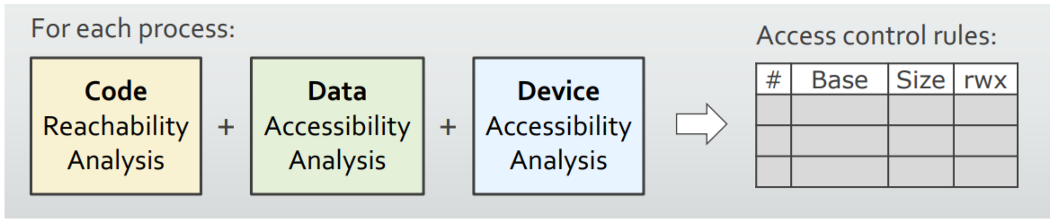
- Memory View
  - **Minimum set of memory regions** essential to correctly operate **each process**
  - Identified by static program analysis
- View Switcher
  - Trusted Compute Base(TCB)
  - Component that loads memory-views
  - Isolated from RTOS and other unprivileged processes



Overall architecture of MINION.

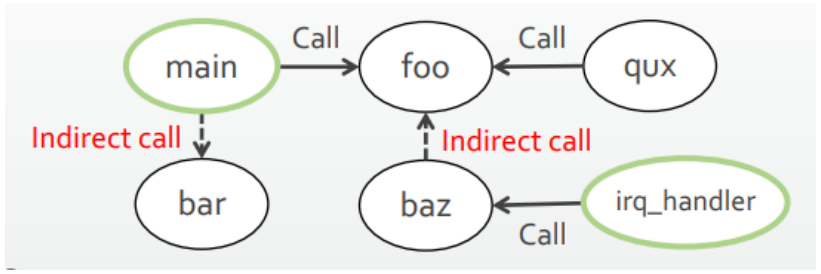
Memory View Tailoring

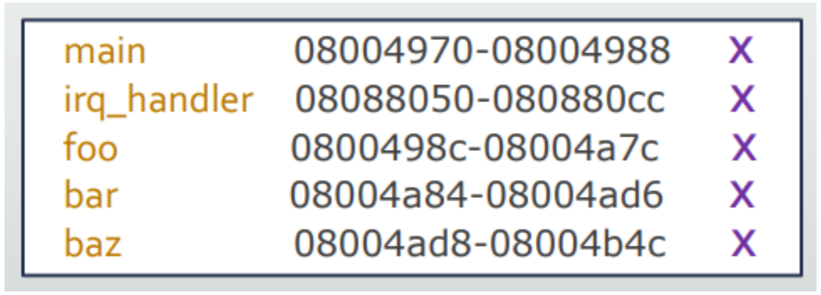
- Find the physical memory regions *essential* , per-process
- Static firmware analysis
- Code injection/reuse,
- Data corruption,
- Physical device abuse


For each process:
Code Reachability Analysis
+
Data Accessibility Analysis
+
Device Accessibility Analysis
Access control rules:
#
Base
Size
rwx

Code Reachability Analysis

- Find all *reachable* functions from the entry functions
- Entry functions
  - Start function & interrupt handlers
  - Identified by analyzing a few RTOS functions
- Indirect calls are handled by inter-procedural points-to analysis
- Build a list of executable memory regions for each process

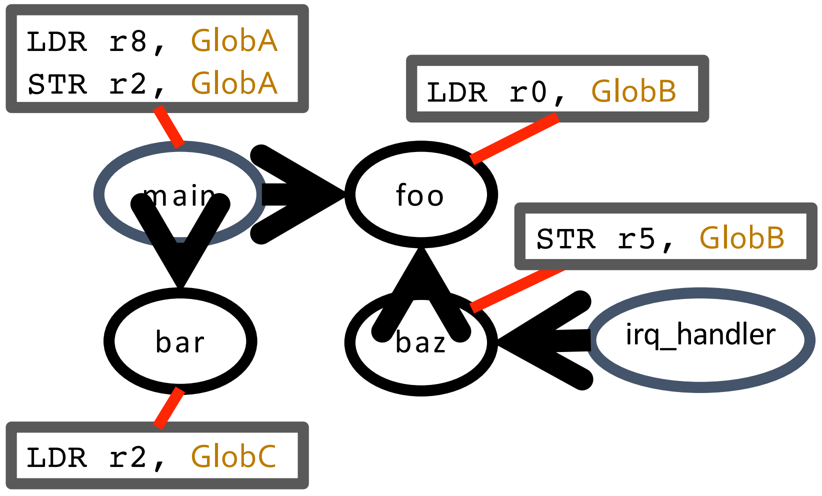




main	08004970-08004988	X
irq_handler	08088050-080880cc	X
foo	0800498c-08004a7c	X
bar	08004a84-08004ad6	X
baz	08004ad8-08004b4c	X

Data Reachability Analysis

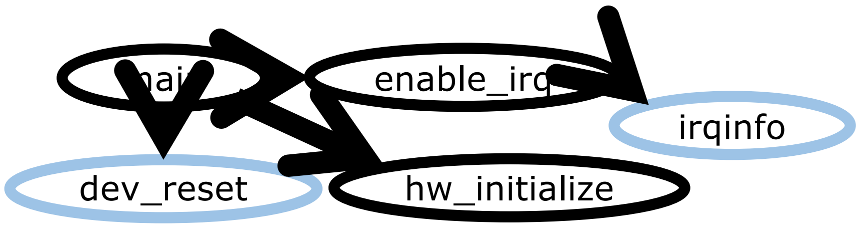
- Global data
  - **Forward slicing** based on inter-procedural value flow graph
  - Build a list of global data for each process
- Stack and Heap data
  - Memory pool **size profiling** with annotated memory allocator
  - **Per-process memory pool** allocation



GlobA	200010f0-200010f4	RW
GlobB	20014618-20014638	RW
GlobC	080b3428-080b3440	R

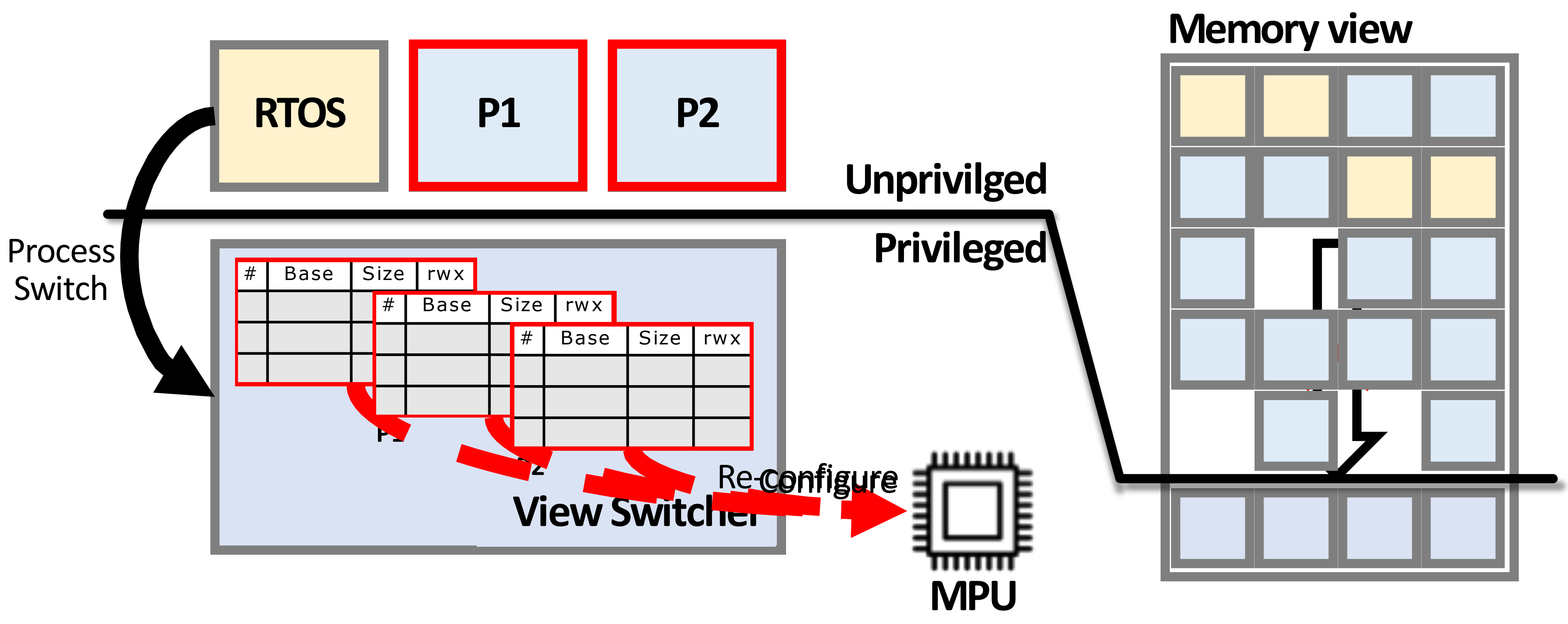
Device Reachability Analysis

- Find load and store instructions with an MMIO address
- **Backward slicing** on inter-procedural value flow graph
- Build a list of **peripheral-mapped** memory regions for each process



DEVICE_X	50000804-50000808	W
NVIC_A	e000e100-e000e104	RW
NVIC_B	e000e104-e000e108	RW

Run-time Memory View Enforcement



Implementation

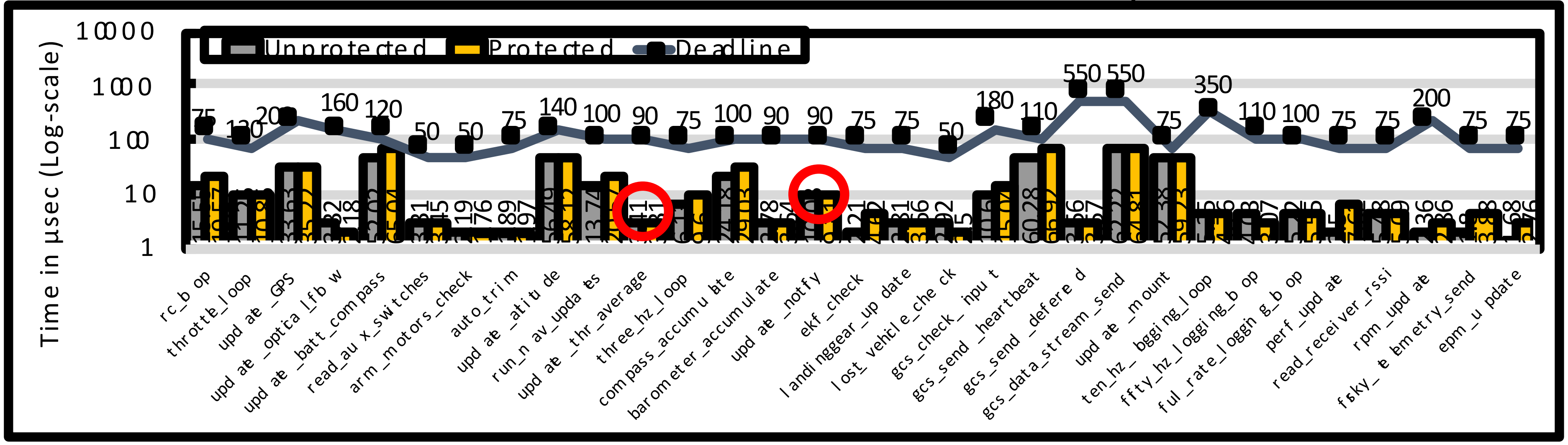
- Drone platform, 3DR-IRIS+ based off Pixhawk µC
- Fail-safe landing feature on illegal memory access

- 787 LOC : View Switcher
- 87 LOC : RTOS

3DR IRIS+

Evaluation: (1)Performance Impact

- Real-time Benchmarks:
  - 31 real-time tasks with deadlines: 2% overhead (every context switch)
  - **All deadline constraints satisfied**
- Micro-Benchmarks:
  - switch_view: 15 µs
  - read_scb : 4 µs
  - write_scb : 5 µs



Evaluation: (2)Security Experiments

Discovery: Memory Corruption Bugs

- 4 new bugs
  - Side-effect of developing Minion

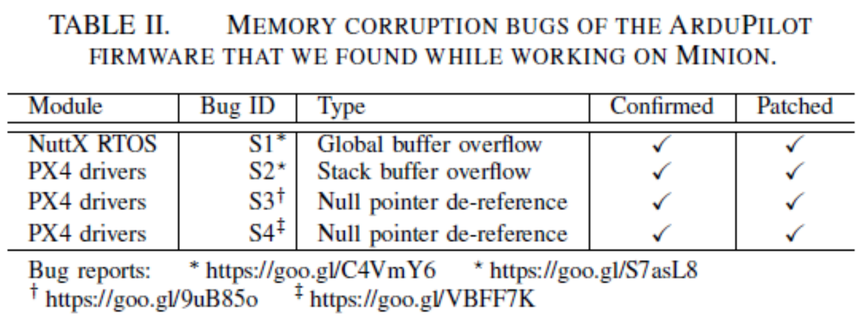


TABLE II. MEMORY CORRUPTION BUGS OF THE ARDUPILOT FIRMWARE THAT WE FOUND WHILE WORKING ON MINION.

Module	Bug ID	Type	Confirmed	Patched
NuttX RTOS	S1*	Global buffer overflow	✓	✓
PX4 drivers	S2⋆	Stack buffer overflow	✓	✓
PX4 drivers	S3†	Null pointer de-reference	✓	✓
PX4 drivers	S4‡	Null pointer de-reference	✓	✓

Bug reports: * https://goo.gl/C4VmY6 ⋆ https://goo.gl/S7asL8
† https://goo.gl/9uB85o ‡ https://goo.gl/VBFF7K

Evaluation: (2)Security Experiments

Attack Cases

- Exploits buffer overflow bug in PX4 driver

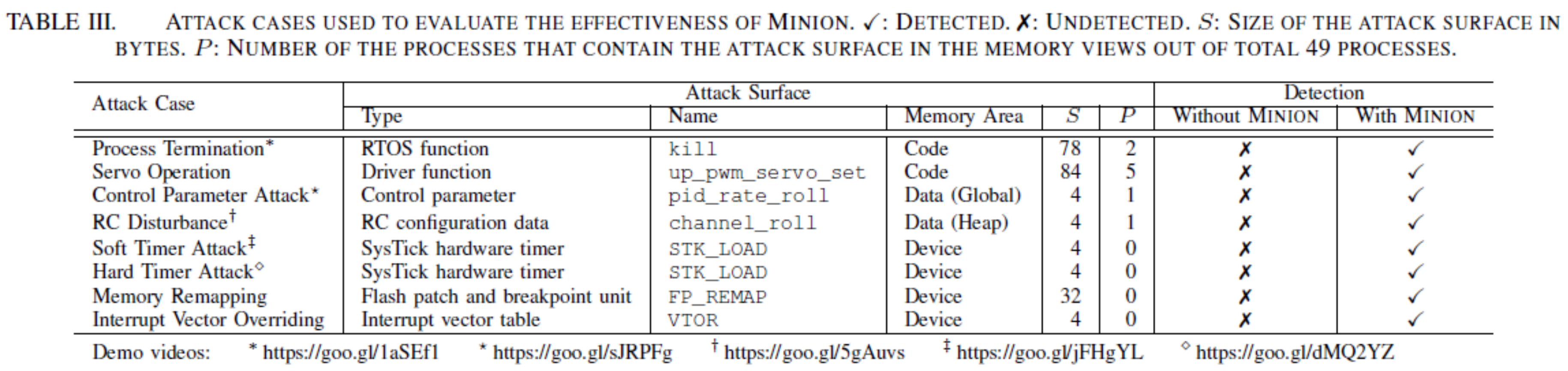


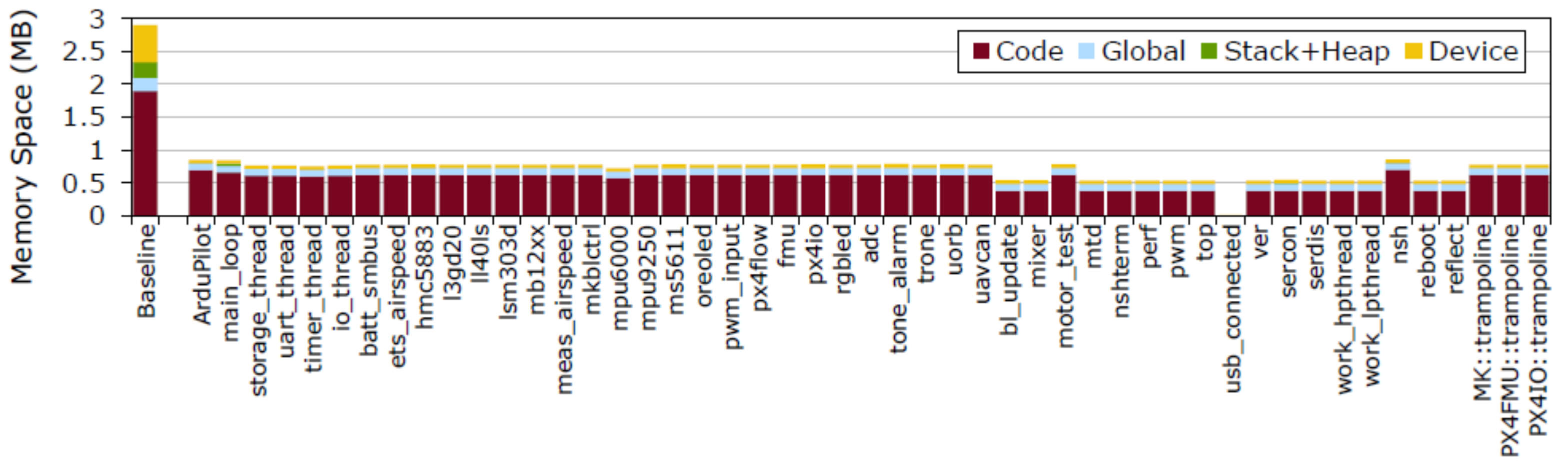
TABLE III. ATTACK CASES USED TO EVALUATE THE EFFECTIVENESS OF MINION. ✓: DETECTED. ✗: UNDETECTED. S: SIZE OF THE ATTACK SURFACE IN BYTES. P: NUMBER OF THE PROCESSES THAT CONTAIN THE ATTACK SURFACE IN THE MEMORY VIEWS OUT OF TOTAL 49 PROCESSES.

Attack Case	Attack Surface					Detection	
	Type	Name	Memory Area	S	P	Without MINION	With MINION
Process Termination*	RTOS function	`kill`	Code	78	2	✗	✓
Servo Operation	Driver function	`up_pwm_servo_set`	Code	84	5	✗	✓
Control Parameter Attack⋆	Control parameter	`pid_rate_roll`	Data (Global)	4	1	✗	✓
RC Disturbance†	RC configuration data	`channel_roll`	Data (Heap)	4	1	✗	✓
Soft Timer Attack‡	SysTick hardware timer	`STK_LOAD`	Device	4	0	✗	✓
Hard Timer Attack◇	SysTick hardware timer	`STK_LOAD`	Device	4	0	✗	✓
Memory Remapping	Flash patch and breakpoint unit	`FP_REMAP`	Device	32	0	✗	✓
Interrupt Vector Overriding	Interrupt vector table	`VTOR`	Device	4	0	✗	✓

Demo videos: * https://goo.gl/1aSEf1 ⋆ https://goo.gl/sJRPFg † https://goo.gl/5gAuvs ‡ https://goo.gl/jFHgYL ◇ https://goo.gl/dMQ2YZ

ILLINOIS
Information Trust Institute

Evaluation: (3)Memory Space Reduction



(a) 8 MPU regions (ARM Cortex-M).

Conclusion

- Memory protection in RT microcontrollers
- **Minion**: New architecture to bring memory isolation to RT microcontroller systems
- Significant memory space reduction with maintained RT responsiveness
- Attack cases and vulnerability discovery

Analysis

Strengths

- Uncovers new security holes in the firmware
- Real-time guarantees still satisfied

Weaknesses

- Attack windows still exists due to providing limited access control protection between views
- Requires root access to the microcontroller and redeployment.

Opportunities/Future Work

- Using trusted execution environments can provide increased isolation guarantees between RTOS and the View Switcher
- Better static/dynamic analysis techniques

Threats

- View switcher is a single point of failure
- Calculation of deadlines not explained