Securing Real-Time Internet-of-Thing

Chien-Ying Chen , Monowar Hasan and Sibin Mohan
Presented by: Cuidi Wei

What will threaten RT-IoT Security?

Security Threats for Vendor-based model

Execute malicious codes

Malicious codes Injection

```
event message_t*
Receive.receive(message_t* bufP
                uint8_t len){
  // BUFF_LEN is defined somew
  uint8_t tmp_buff[BUFF_LEN];
  rcm = (radio_count_msg_t*)payl

  // copy the content in a buffer f
  for (i=0;i<rcm->buff_len; i++
       tmp_buff[i]=rcm->buf
  }
  return bufPtr;
}
```

(a) Sample buffer managemen

Memory address	Usage	normal value	value after overflow
0x10FF	End Mem		
⋮	⋮	⋮	⋮
0x1062	other	0xXX	$ADDR_H$
0x1061	other	0xXX	$ADDR_L$
0x1060	$@ret_H$	0x38	0x2b
0x105F	$@ret_L$	0x22	0x58
0x105E	tmpbuff[3]	0	0x03
0x105D	tmpbuff[2]	0	0x02
0x105C	tmpbuff[1]	0	0x01
0x105B	tmpbuff[0]	0	0x00

(c) Buffer overflow with a packet containing the bytes shown in Figure 2(b).

Malicious codes Injection

```
event  message_t*
Receive.receive(message_t* bu
                uint8_t len){
  // BUFF_LEN is defined some
  uint8_t tmp_buff[BUFF_LEN
  rcm = (radio_count_msg_t*)pa

  // copy the content in a buffer
  for (i=0;i<rcm->buff_len; i+
        tmp_buff[i]=rcm->bu
  }
  return bufPtr;
}
```

(a) Sample buffer management vulnerability.

```
uint8_t payload[ ]={
 0x00,0x01,0x02,0x03, // padding
 0x58,0x2b,           // Address of gadget 1
 ADDR_L,ADDR_H,       // address to write
 0x00,                // Padding
 DATA,                // data to write
 0x00,0x00,0x00,      // padding
 0x85,0x01,           // address of gadget 2
 0x3a,0x07,           // address of gadget 3
 0x00,0x00            // Soft reboot address
 };
```

(b) Payload of the injection packet.

Security Threats for Vendor-based model

Execute malicious codes

Infer critical system information

Side-Channel Attacks

- A side-channel attack manipulates previously unknown channels to acquire useful information from the victim.

  Eg.

  Memory/cache access time

  Power consumption traces

Security Threats for Vendor-based model

Execute malicious codes

Infer critical system information

Target the communication interfaces

Attacks on Communication Channels

- **Cryptographic approaches** vs **Performance**
- **Performance** vs **QoS** ???

Security Threats for Vendor-based model

Execute malicious codes

Infer critical system information

Target the communication interfaces

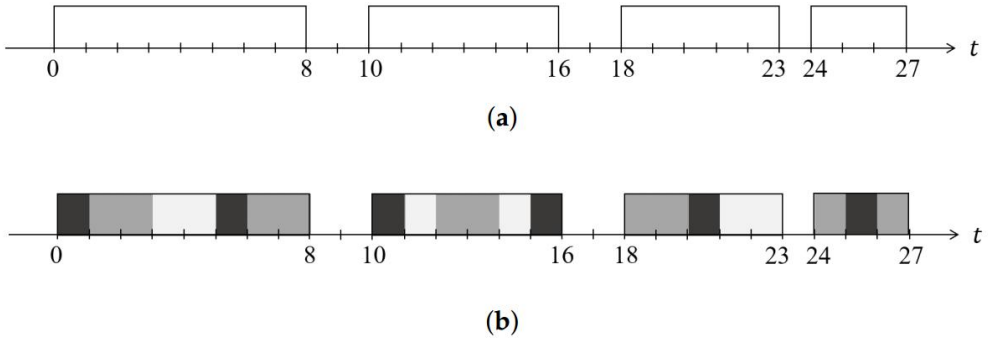
Perform denial of service attacks(DoS, DDoS)

Attacker Reconnaissance

1. Scheduleak

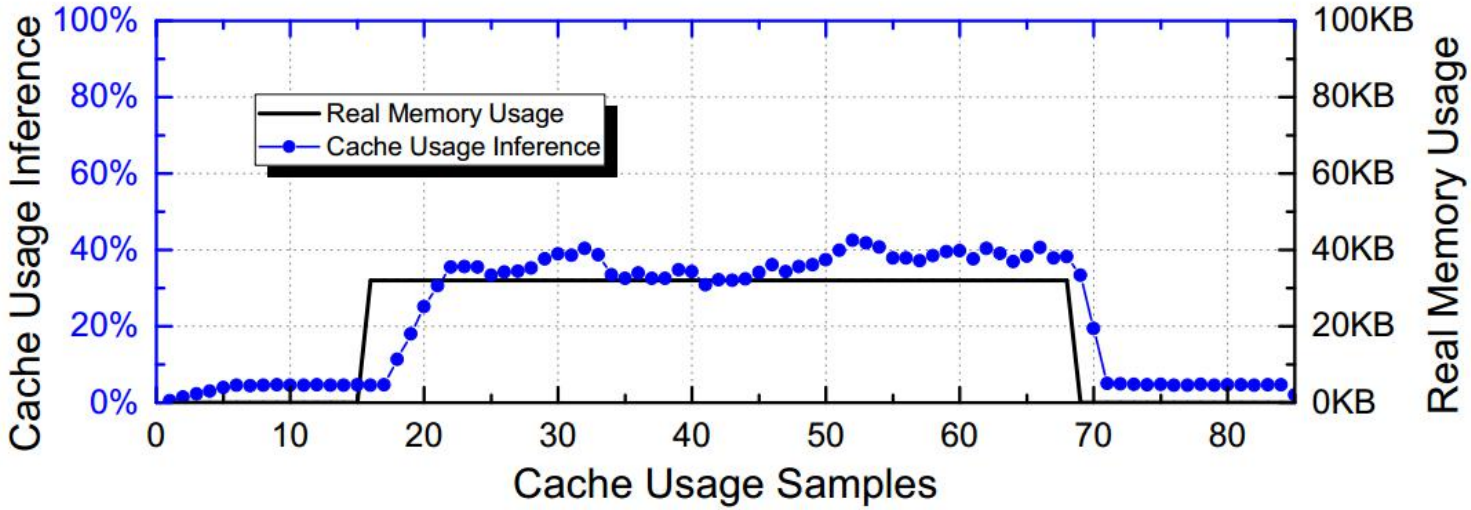
It utilizes an observer task that has the lowest priority in the victim system to observe busy intervals.

R(G, W) = J, where W is a set of observed busy intervals and J is the inferred schedule information that can be used to pinpoint the possible start time of any particular victim task.



2.Targeted attacks

It combined side-channel attack with the Scheduleak algorithm and carried out a cache-timing attack.



Cache-timing attack

In cryptography, a timing attack is a side-channel attack in which the attacker attempts to compromise a cryptosystem by analyzing the time taken to execute cryptographic algorithms.

Q: What is the (Cache-)timing attack used for?

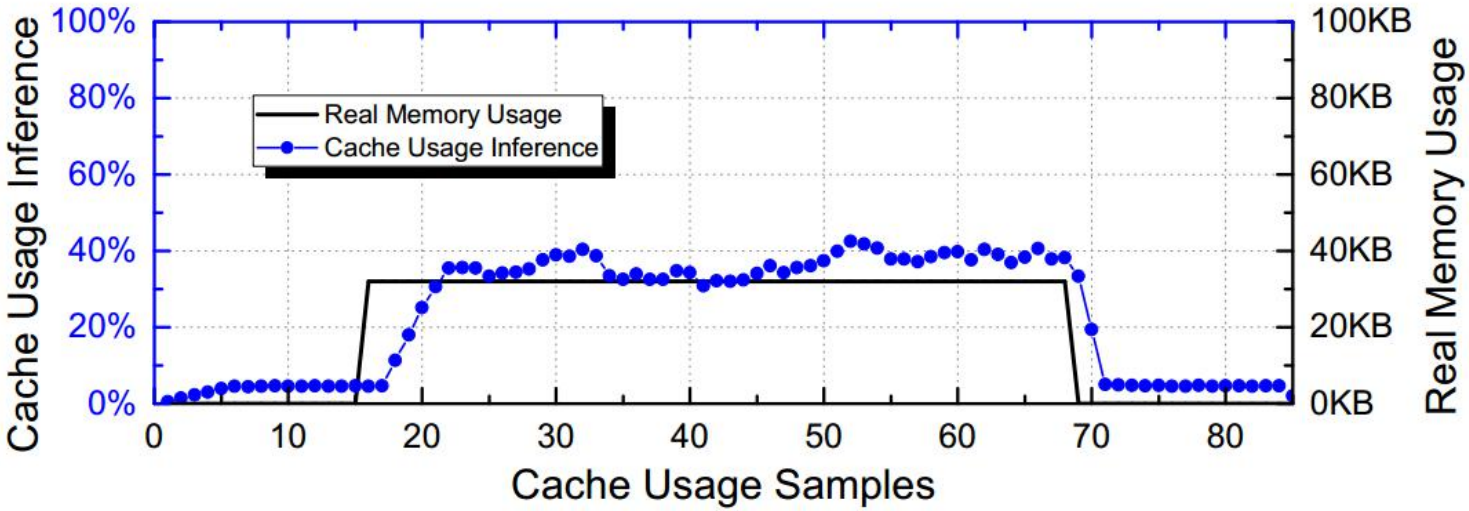
Q: What can the (Cache-)timing attack be used for?

Eg. There is a function which is used for comparing the passwords inputted by users with the passwords stored in the system.

If the function compares these two passwords word by word, the function will stop when the two words are different. By computing the speed of response we can figure out which word is the first different word between the two passwords.

2.Targeted attacks

It combined side-channel attack with the Scheduleak algorithm and carried out a cache-timing attack.



How to improve the RT-IoT security?

Security Approaches

References	Approach	Attack Surface	Overhead/Costs
Simplex-based security [27–31]	Use verified/secure hardware module to monitor system behavior (e.g., timing [28] and execution pattern [27], memory access [29], system call usage [30], control flow [31])	Code injection attacks	Require custom hardware or monitoring unit
Security by platform-level reset [32,49]	Periodically and/or asynchronously (e.g., upon detection of a malicious activity) restart the platform and load an uncompromised OS image	Code injection, side channel and DoS attacks	Extra hardware to ensure safety during periodic/asynchronous restart events
Cache flushing [33,44]	Flush the shared medium (e.g., cache) between the consecutive execution of high-priority (security sensitive) and low-priority (potentially vulnerable) tasks	Side-channel (cache) attacks	Overhead of cache flushing reduces task-set schedulability
Schedule randomization [50]	Randomize the task execution order (i.e., schedule) to reduce the predictability	Side-channel attacks	Extra context switch

Security Approaches

References	Approach	Attack Surface	Overhead/Costs
Security task integration for legacy RT-IoT [35,37]	Execute monitoring/intrusion detection tasks with a priority lower than real-time task to preserve the real-time task parameters (e.g., period, WCET and execution order)	Code injection, side-channel, DoS and/or communication attacks depending on the what monitoring tasks are used	Running security task with lower priority may cause longer detection time due to high interference (e.g., preemption) from real-time tasks
Adaptive security task integration [36]	Execute monitoring/intrusion detection tasks with a lowest priority most of the time (e.g., during normal system operation)—however change the mode of operation execute with a higher priority (for a limited amount of time) if any anomalous behavior is suspected	Code injection, side-channel, DoS and/or communication attacks depending on the what monitoring tasks are used	False positive detection may cause unnecessary mode switches

Hardware Support

- Secure System Simplex Architecture(S3A)
- Secure Core Framework
- Control Flow Monitoring --- Check integrity of code flow
- Resecure --- Reset the system frequently enough

1.S3A Architecture -- Secure System Simplex Architecture

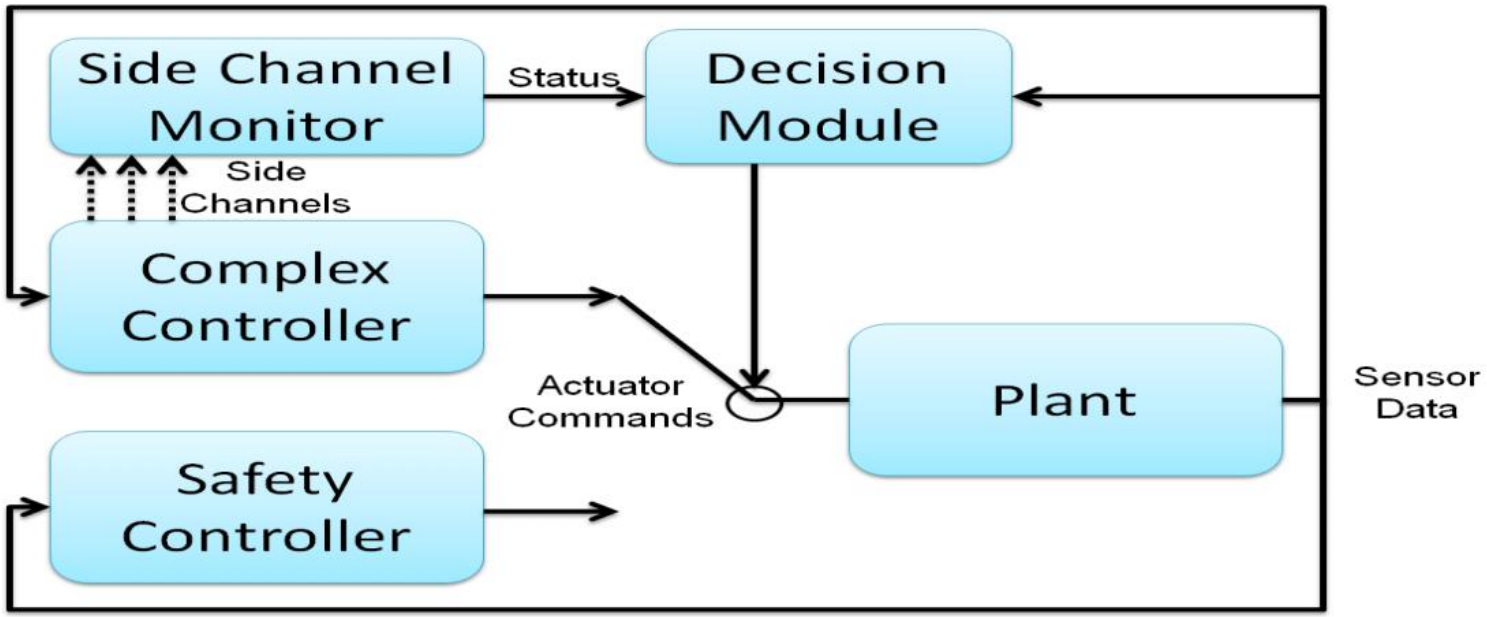
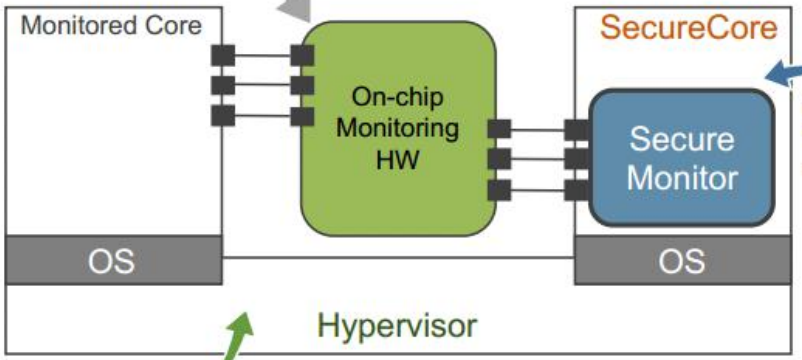


Figure 2: S3A Architecture

2.SecureCore Framework

On-chip Monitoring HW Unit

- Observes the state of monitored cores, I/O activities, physical states, etc.
- Invisible to all but SecureCore, non-intrusive



Secure Monitor

- Software process that performs monitoring and detection using observed behavior

Hypervisor-based SecureCore Protection

- Resource virtualization: memory space separation, I/O device consolidation

4. Resecure

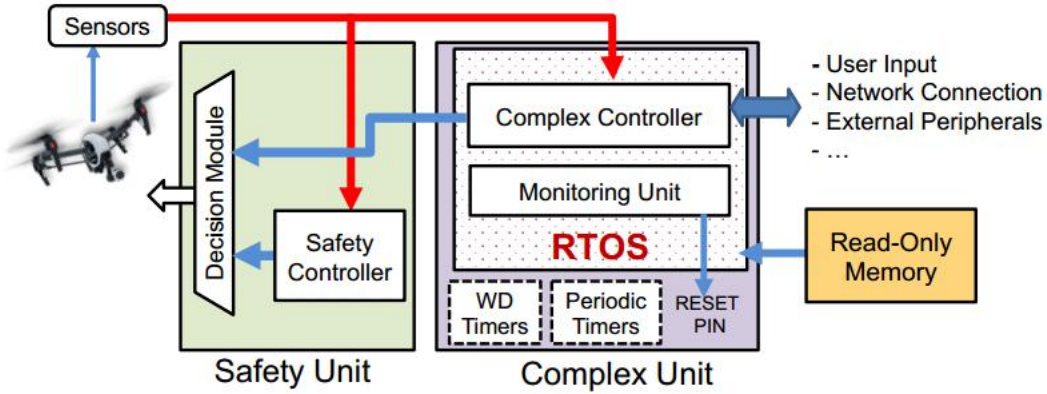
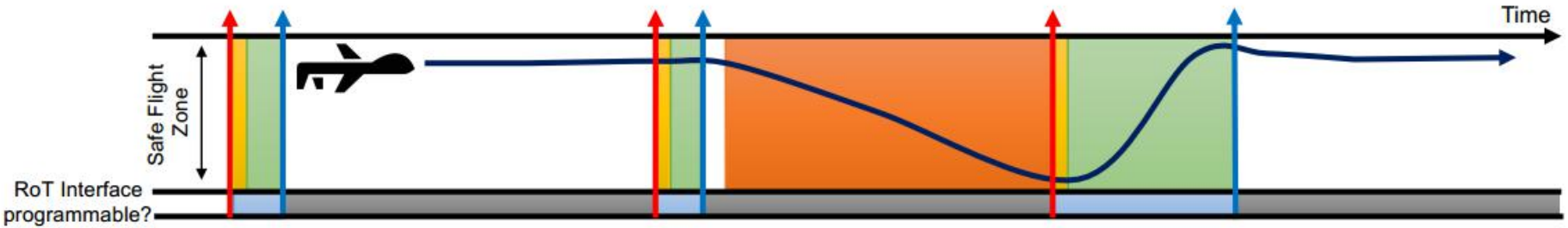


Figure 6. The *ReSecure* framework [32]: Safety unit is the bare-metal verified component and complex unit is not verified. The decision module switches between the controllers to provide overall system safety.

4. Resecure



Security without custom HW support

(1)Capture Security Constraints between tasks

(2)Random the Task Schedule - TaskShuffler

(3) Integrating Security for Legacy RT-IoT

Security without custom HW support

(1)Capture Security Constraints between tasks

For any two tasks ti and tj:

if noleak(ti, tj) = True, then information leakage from ti to tj must be prevented;

if noleak(ti, tj) = False, no such constraints need to be enforced.

(2)Random the Task Schedule - TaskShuffler

A randomization protocol for fixed-priority scheduling algorithm, to achieve such randomness in task schedule.

Security without custom HW support

(3) Integrating Security for Legacy RT-IoT

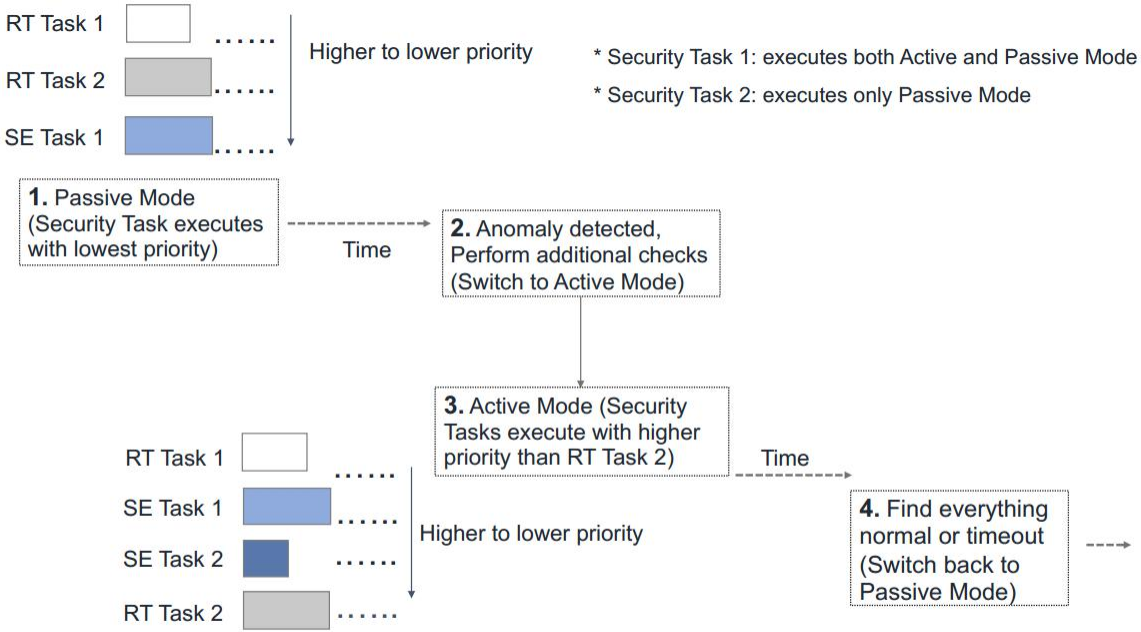


Figure 9. Flow of operations in *Contego* depicting different modes for the security tasks.

Critique

1. For ReSecure, is resetting a great way?
2. How are things like memory access times found by attackers?

References

[34] Code Injection Attacks on Harvard-architecture Devices
[27] Secure system simplex architecture for enhanced security and robustness of cyber-physical systems