ACES: AUTOMATIC COMPARTMENTS FOR EMBEDDED SYSTEMS

Abraham A. Clements
clemen19@purdue.edu

Naif Saleh Almakhdhub,
nalmakhd@purdue.edu

Saurabh Bagchi
sbagchi@purdue.edu

Mathias Payer
mathias.payer@nebelwelt.net

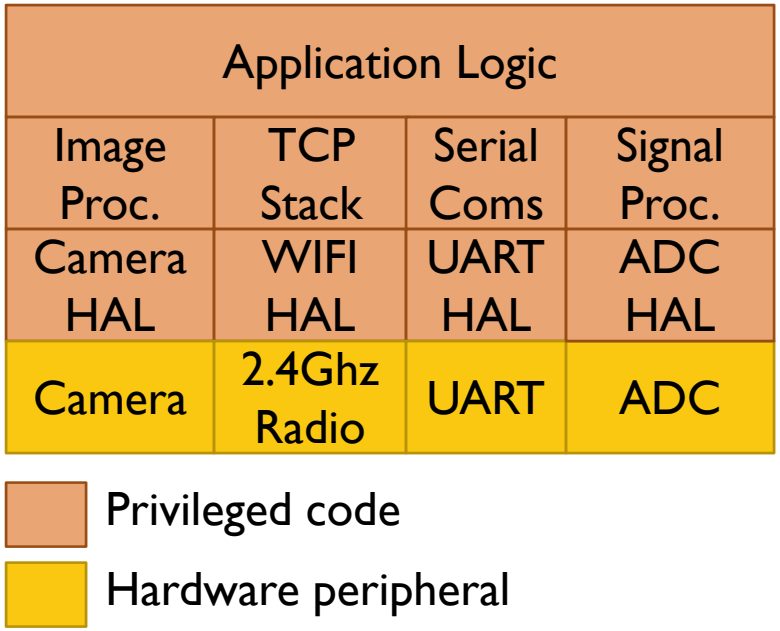
Sandia National Laboratories is a multimission laboratory managed and operated by National Technology & Engineering Solutions of Sandia LLC, a wholly owned subsidiary of Honeywell International Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525. SAND2018-8907 C

PROBLEM STATEMENT

- Bare-metal IoT devices/ SoC's are vulnerable
  - Google P0's Broadcom SoC

    CVE-2017-6957
- Legacy code
  - IoT devices, vehicles, etc.
- No defenses
  - Even DEP is missing
  - No separation of privileges

Bare-metal Application

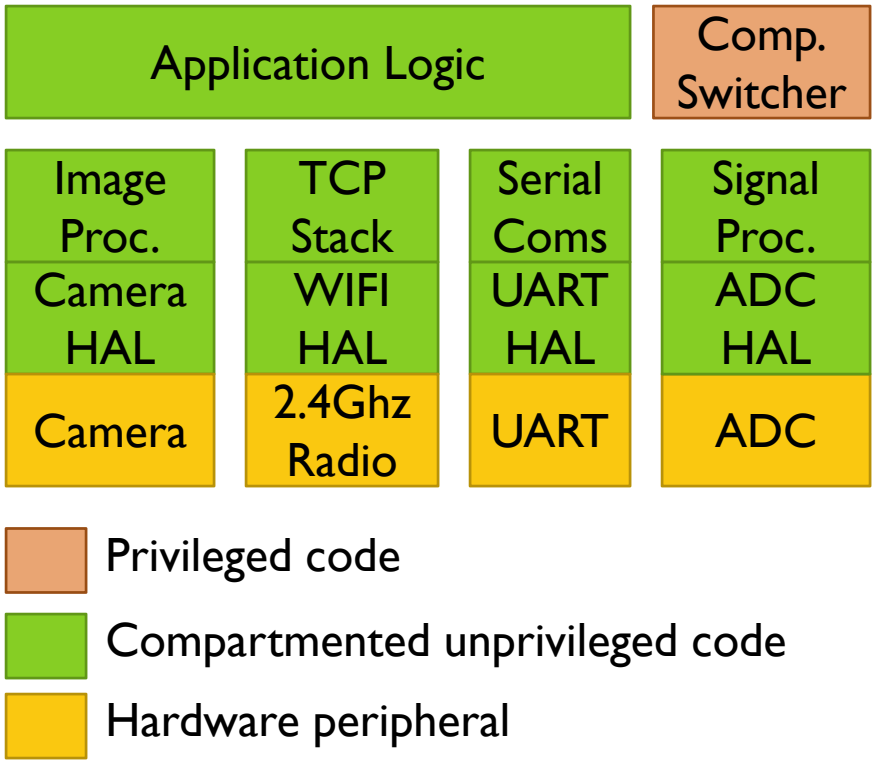


Result: Single vulnerability compromises the entire system

ACES OVERVIEW

- ACES creates many compartments
  - Applies least privileges
  - Creates sub-thread compartments
  - Protects **integrity** of sensitive data and peripherals
- Uses static analysis to automatically infer compartments using a policy
- Separates compartmentalization from application development
- Each compartment has associated data/peripherals

Compartmented Application

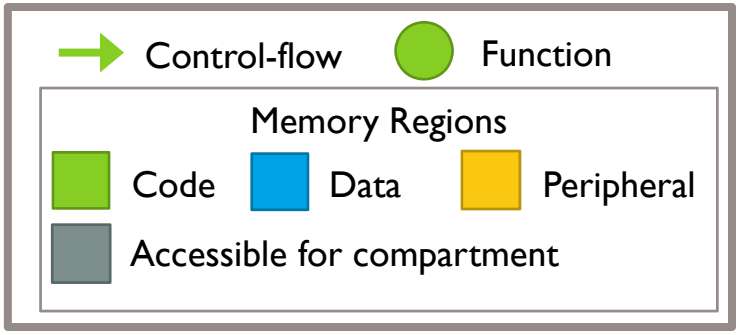


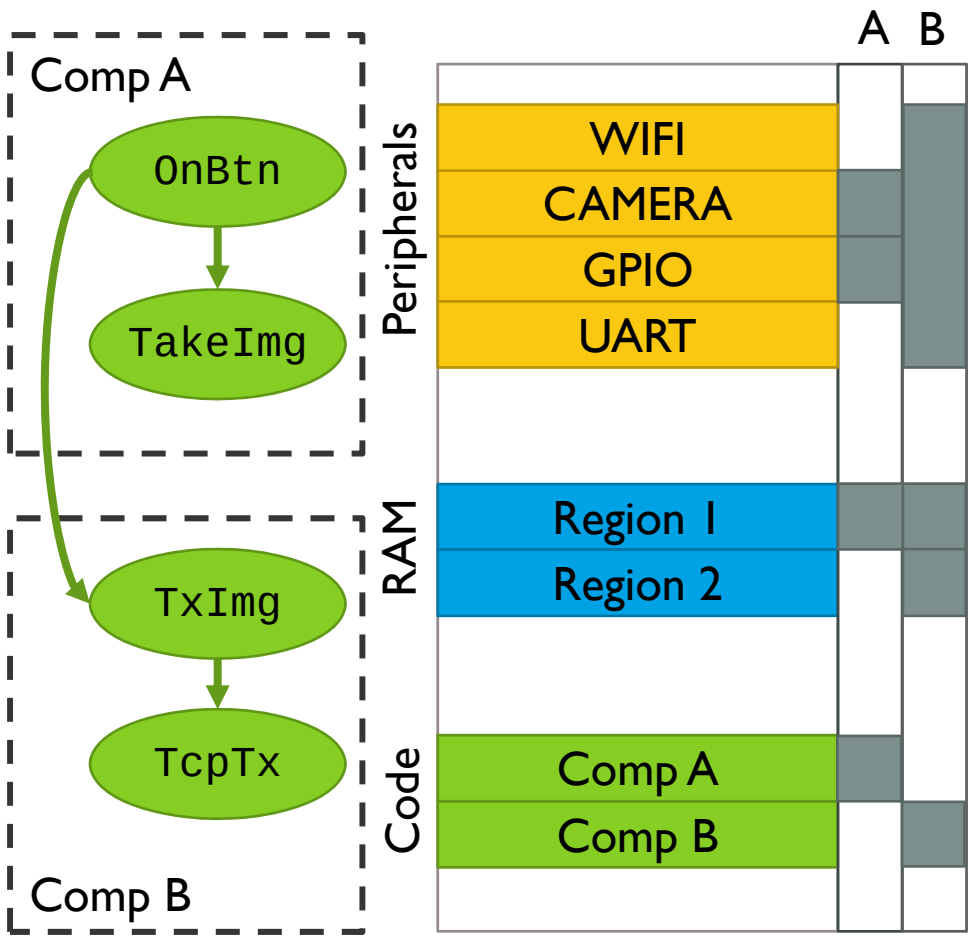
RELATED WORK

- EPOXY – DEP, diversity, and stack protections for bare-metal systems
  - Does not address least privileges
  - Clements A. A. et al. Oakland 2017 (Our work)
- Mbed uVisor – runtime and API to manually create compartments
  - Intermixes source and compartmentalization policy
  - Requires Mbed OS
  - https://www.mbed.com/en/technologies/security/uvisor/
- MINION – Creates thread level compartments on micro-controllers
  - Fixed algorithm used to determine compartments
  - Kim, C. H. et al. NDSS 2018

COMPARTMENTS

- Set of concurrently accessible memory regions and authorized control-flows between them
- Compartments restrict
  - Accessible memory
  - Control-flow between compartments



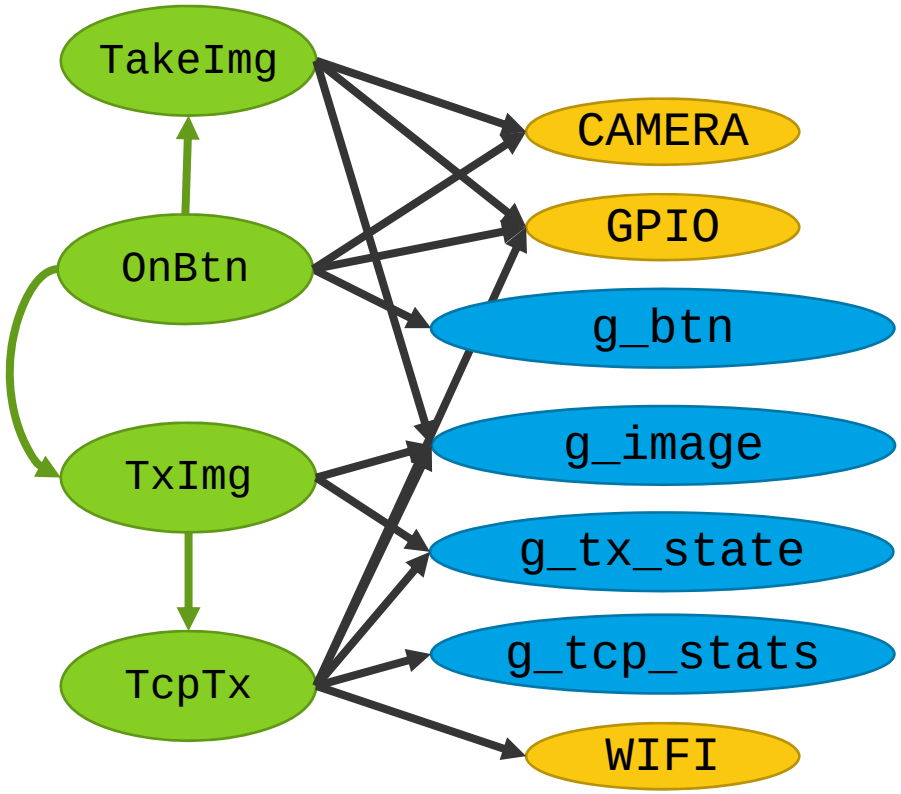


CREATING COMPARTMENT

- Static analysis identifies code, data, and peripheral dependencies
  - ACES ensures compartments can access all required data and peripherals
  - Mirco-Emulator used to dynamically identify missed due to dependencies aliasing
- Compartments are code centric
  - i.e. code belongs to only one compartment
- Policy determines how functions, global variables, and peripherals are grouped to create compartments
- Different policies possible
  - We evaluate: Naïve Filename, Optimize Filename, and Peripheral policies
- MPU is used to restrict access to memory regions

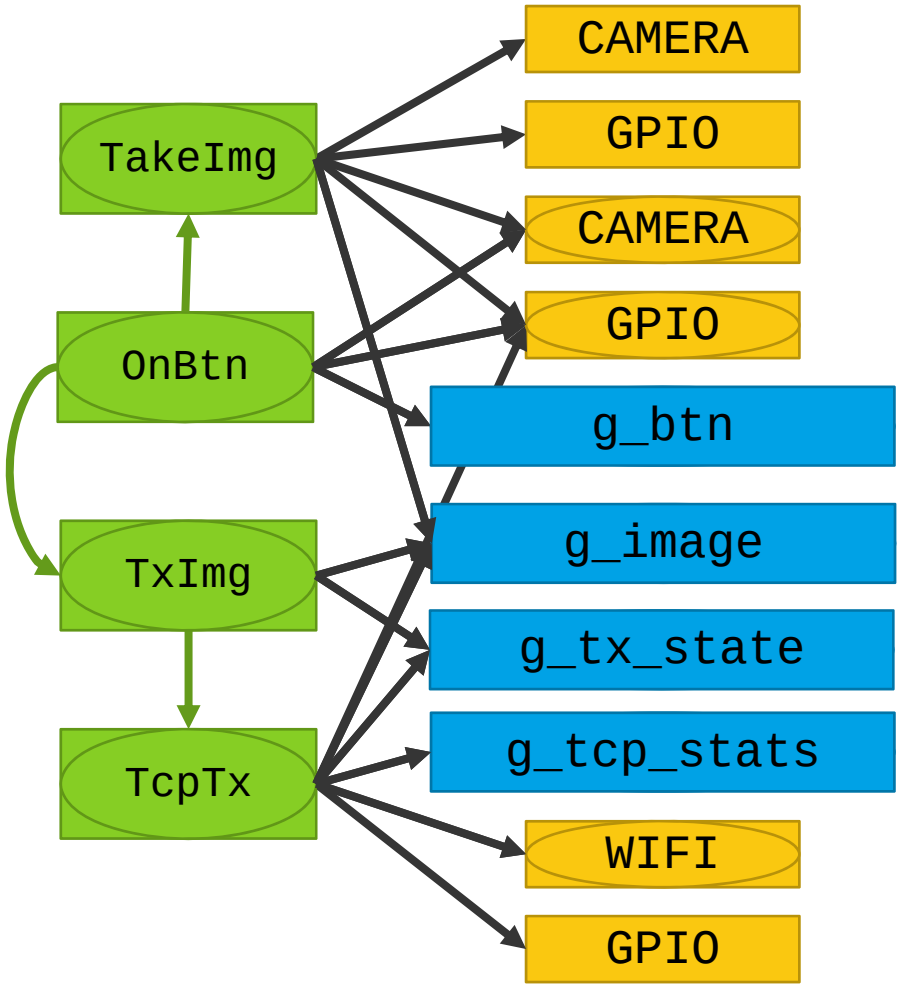
PROGRAM DEPENDENCY GRAPH

```
void OnBtn()
  GPIO.led = ON
  CAMERA.pwr = ON
  g_btn = PUSHED
  TakeImg()
  TxImg()

void TakeImg()
  CAMERA.take_pic = 1
  g_image.buf = CAMERA.rx_reg
  GPIO.led = OFF

void TxImg()
   g_tx_state = ACTIVE
   TcpTx(g_image.buf)
   g_image.sent = TRUE

void TcpTx(*buf)
  GPIO.led_tx = ON
  WIFI.tx_reg = buf
  g_tcp_stats.tx_count++
  g_tx_state = IDLE
```



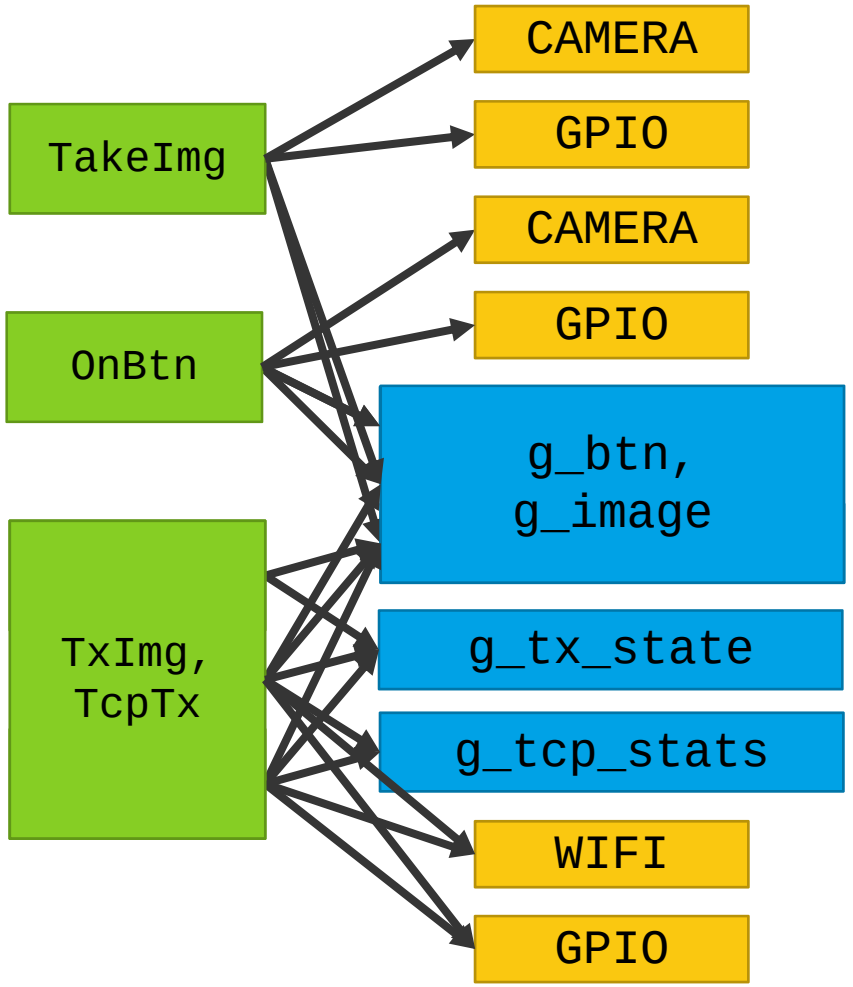
REGION GRAPH

- PDG mapped to a Region Graph
- Functions 1:1
  - Control edges **not** transferred
- Data 1:1
  - Data edges transferred
- Peripherals 1:Many
  - Unique region created per dependency edge



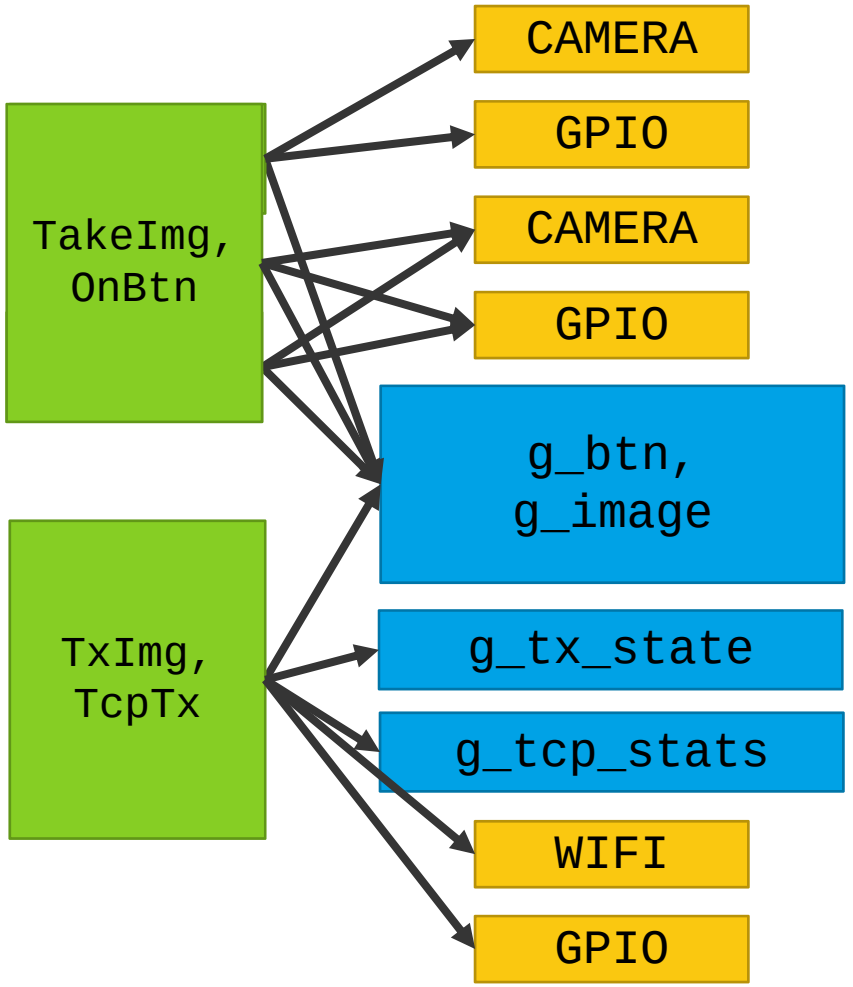
COMPARTMENTALIZATION POLICY

- Defines what should be grouped to form compartments
- Implemented policies
  - Naïve Filename
  - Optimized Filename
  - Peripheral
  - Many more are possible



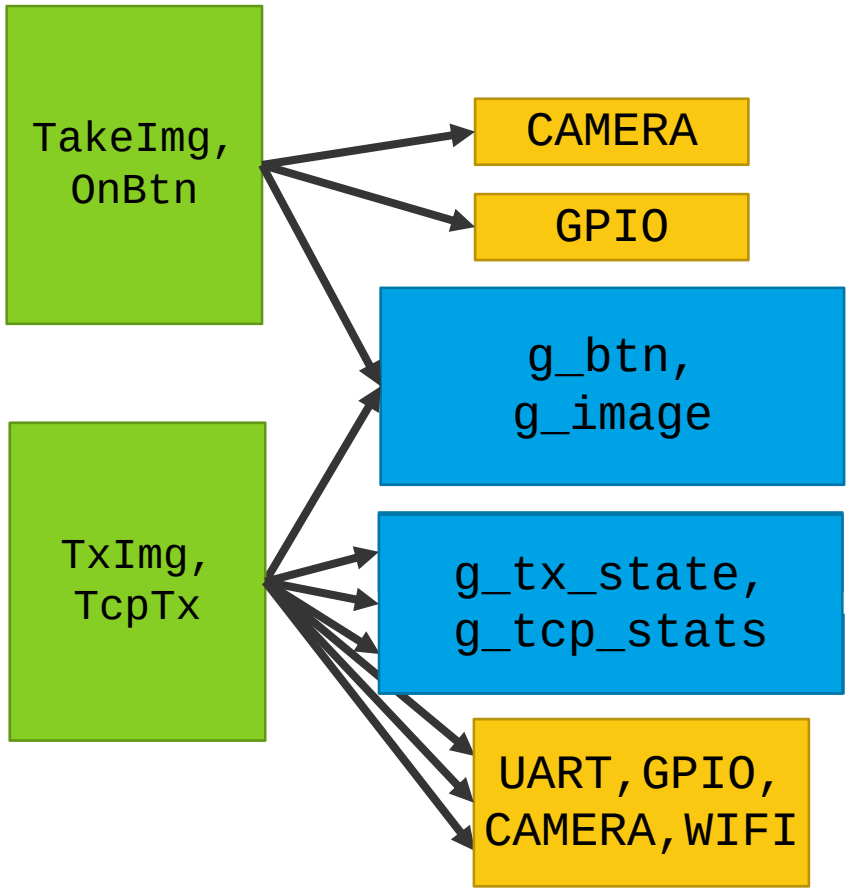
OPTIMIZE

- Improve security
- Improve performance

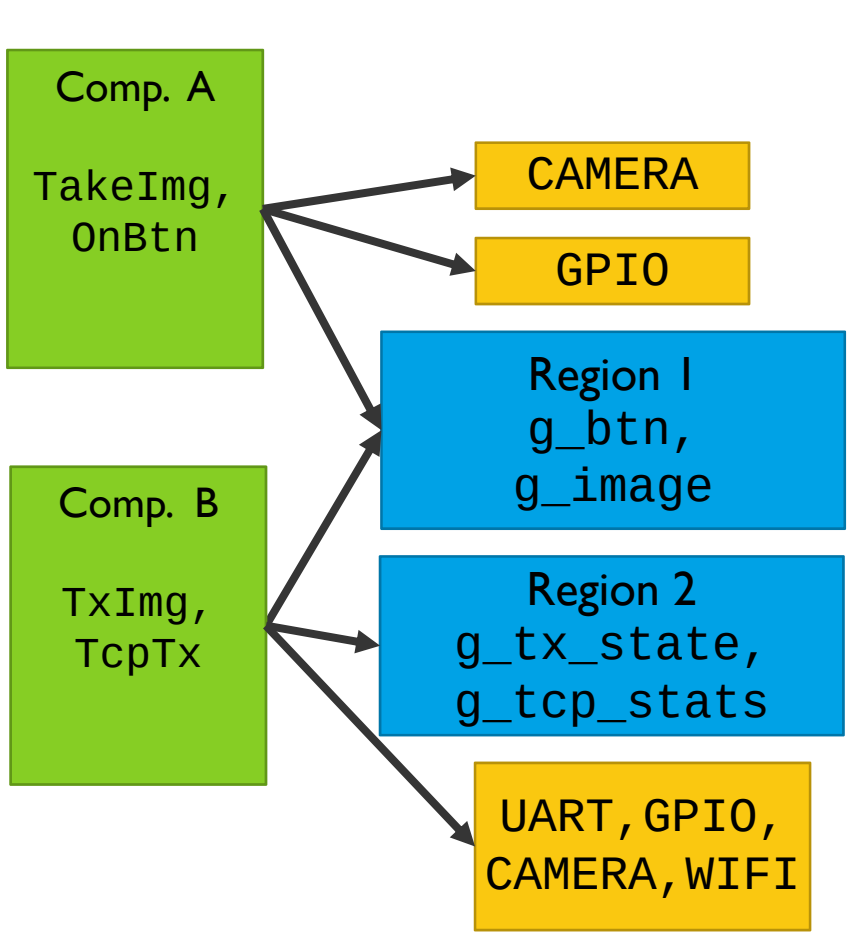


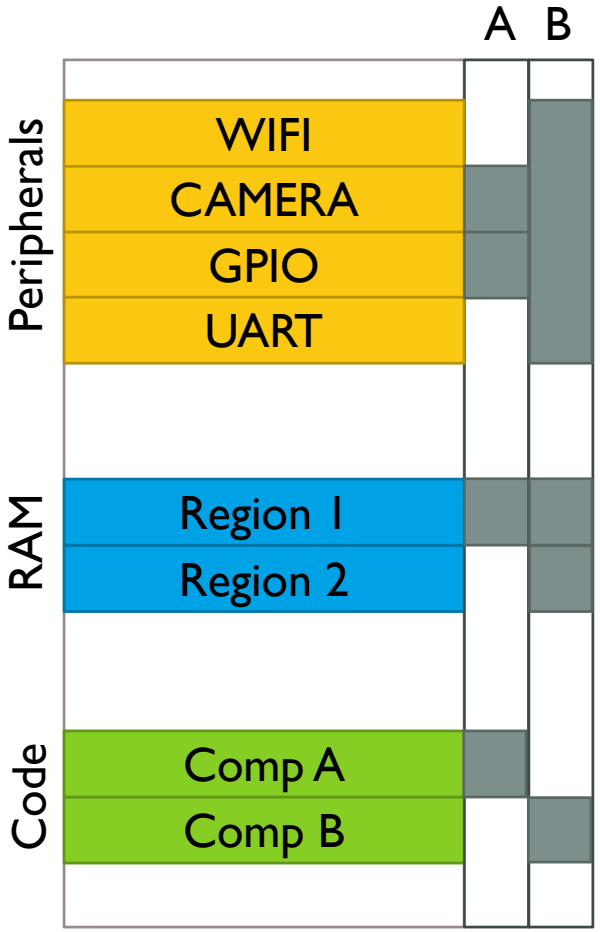
LOWERING

- Reduces graph to meet HW constraints
  - Degree of each code regions must be less than the number of MPU regions
- Lowest cost regions merged until constraints are met
- Lowing may increases permissions of compartments
- Merging peripherals may capture additional peripherals



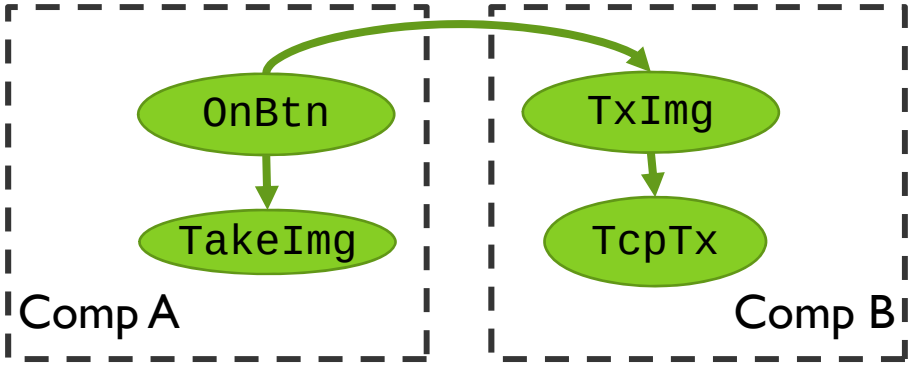
MAPPING TO MEMORY

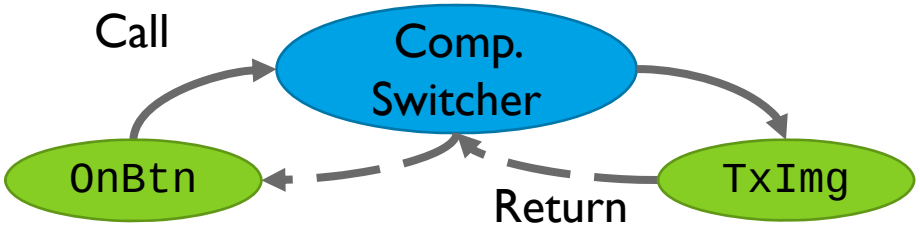




RESTRICTING CONTROL-FLOW

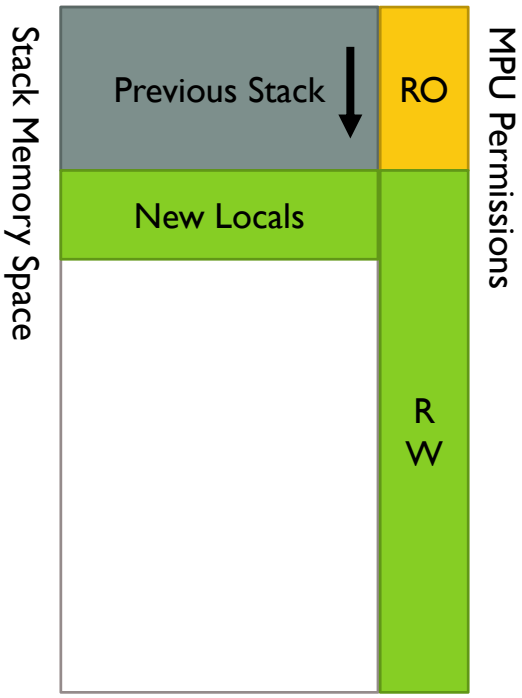
- Instrument calls and returns crossing compartment boundaries
  - Invoke compartment switcher
- Compartment switcher authenticates both directions
  - Memory permissions only changed for valid transitions





MICRO-EMULATOR

- Emulates store instructions in software
  - Overcomes limitations of static analysis in generating PDG
  - Authenticated writes outside a compartment's regions
- Dynamic profiling run creates white-list of accesses per compartment
- Used for stack protection



EVALUATION

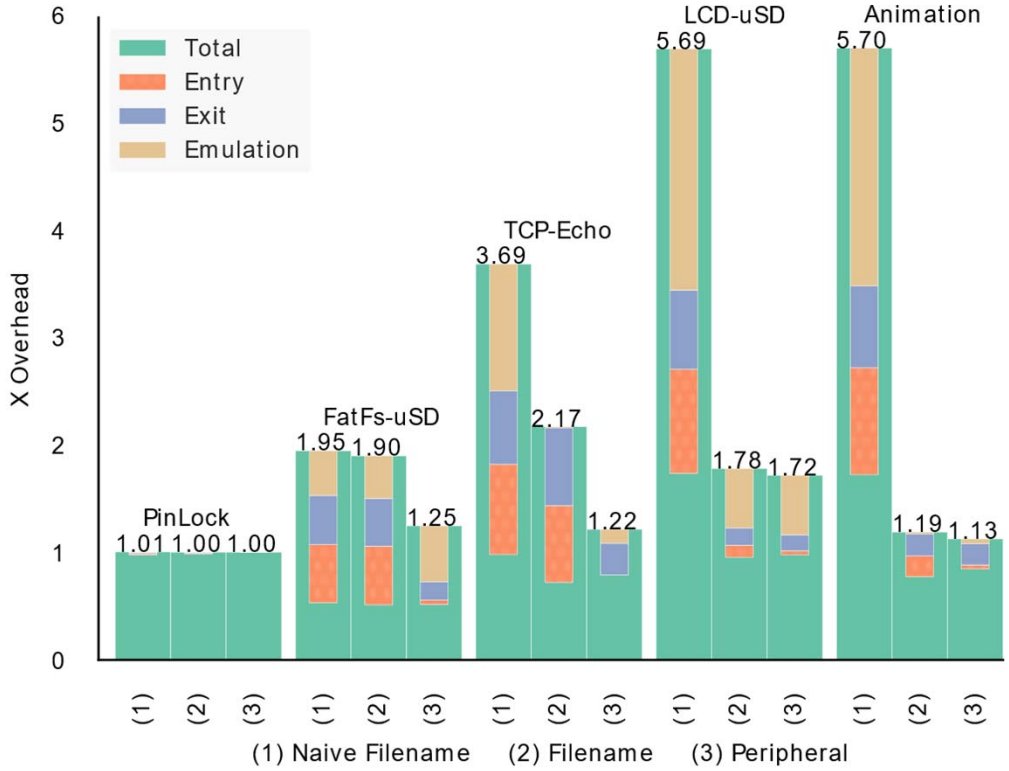
- Evaluated policies for security, runtime, resource usage
  - Naïve Filename
  - Optimized Filename
  - Peripheral
- Five applications run on Cortex-M4
  - PinLock
  - FatFs-uSD
  - TCP-Echo
  - LCD-uSD
  - Animation

PINLOCK CASE STUDY

- Attacker trying to unlock lock
- Assume write-what-where vulnerability in HAL_UART_Receive_IT

	Overwrite		**Control Hijack**	
Policy	Global	GPIO	Direct	Deputy
Naïve Filename	✓	✓	✓	✓
Opt. Filename	✓	✓	✓	✓
Peripheral	✗	✓	✗	✗

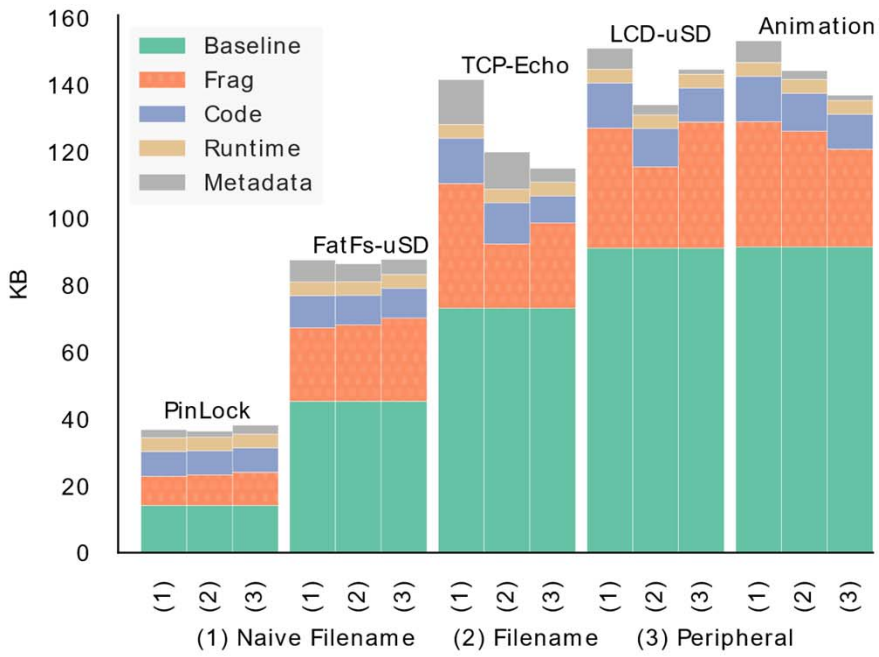
RUNTIME EVALUATION



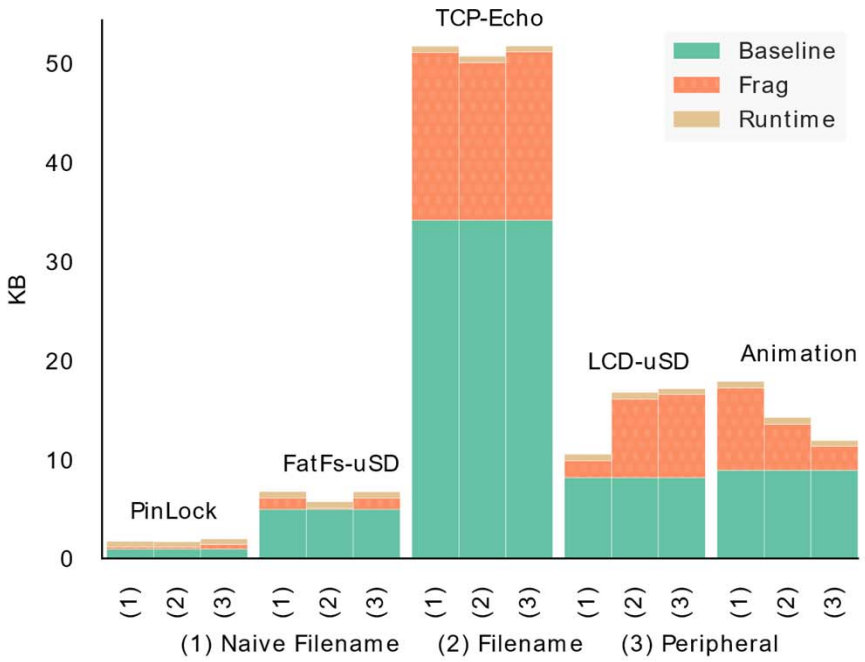
- Can have moderate runtime impact
- Emulating instructions accounts for largest increase in execution time

MEMORY OVERHEAD

Flash Overhead



RAM Overhead



CONCLUSION

- Applies least privileges bare-metal IoT devices
  - Does not require changes to application logic
  - Uses existing hardware
- ACES automatically creates and enforces sub-thread
  - Decouples security policy from application
  - Frees developer from having to manage underlying security hardware
  - Enables research in creating compartmentalization policies
- Code will be available at:

  https://github.com/embedded-sec/ACES