eCTF10

10 YEARS OF THE EMBEDDED CAPTURE THE FLAG

2025 eCTF Award Ceremony

April 25, 2025

CROWDSTRIKE

ROLLS ROYCE

MITRE | SOLVING PROBLEMS FOR A SAFER WORLD®

eCTF10

10 YEARS OF THE EMBEDDED CAPTURE THE FLAG

2025 eCTF Award Ceremony

April 25, 2025

WELCOME!

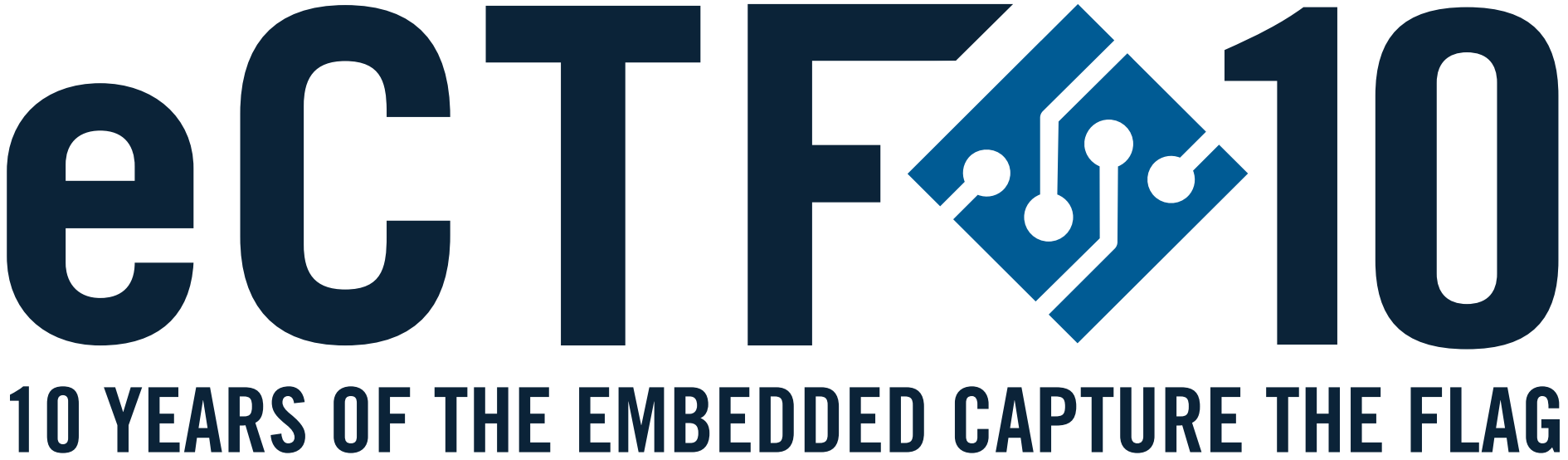
WE WILL GET STARTED SHORTLY

CROWDSTRIKE

ROLLS ROYCE

MITRE | SOLVING PROBLEMS FOR A SAFER WORLD®

Welcome!



MITRE

eCTF10

10 YEARS OF THE EMBEDDED CAPTURE THE FLAG

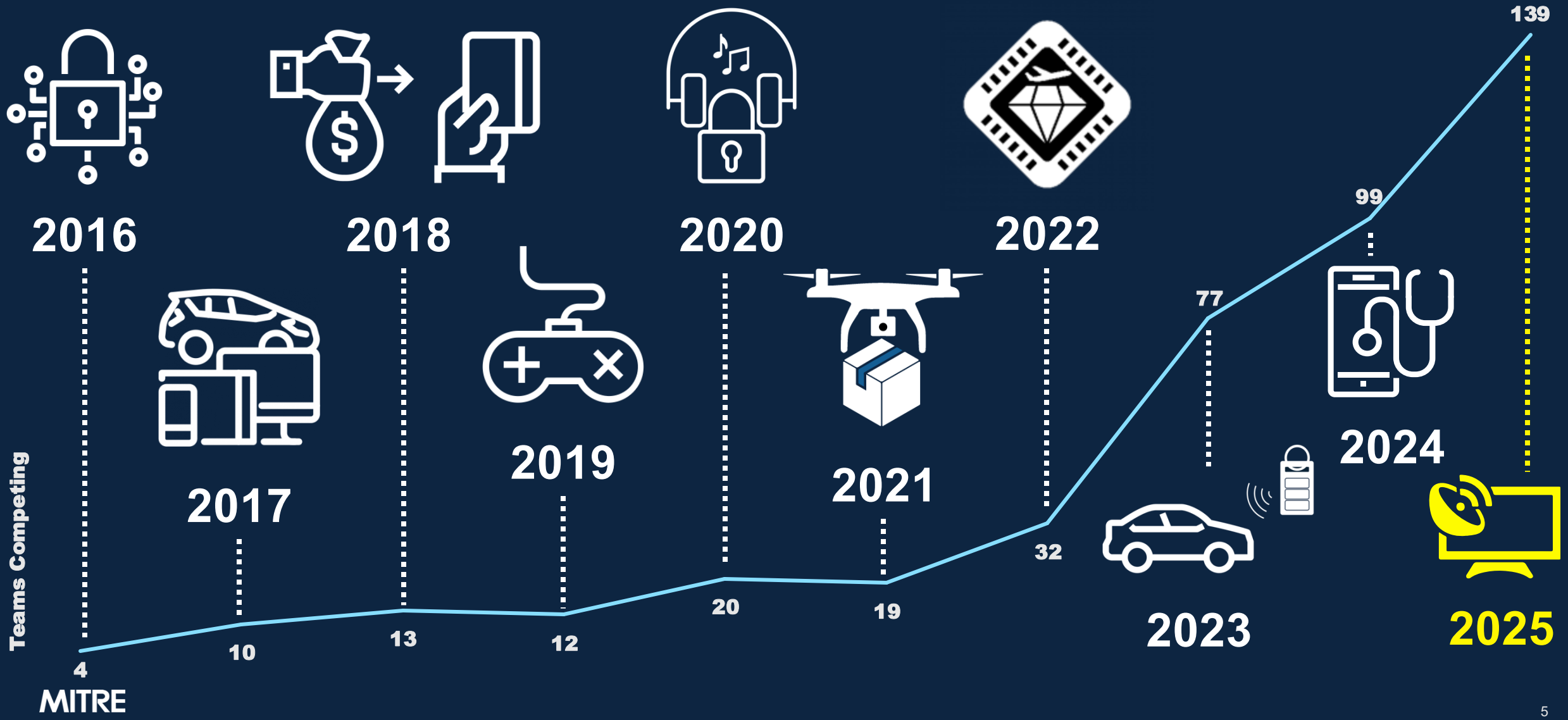
Competition Overview

Time (EST)	Title
15:30	Competition Overview
15:50	NEMC Guest Speaker – John Petrozzelli
16:00	City College of San Francisco
16:15	Mountain View High School
16:30	University of Michigan
16:45	Keynote – Mark Peters, Ph.D.
17:00	Refreshment Break
17:15	Purdue University
17:30	University of California, Los Angeles
17:45	University of Illinois Urbana-Champaign
18:00	Carnegie Mellon University
18:15	Award Presentations
18:25	Closing Remarks – Dan Walters
18:30	Networking Reception

MITRE

Growth of the eCTF

eCTF10
10 YEARS OF THE EMBEDDED CAPTURE THE FLAG



MITRE

Thank You Sponsors!

eCTF10
10 YEARS OF THE EMBEDDED CAPTURE THE FLAG

NEMC HUB

CROWDSTRIKE

ROLLS ROYCE

sysdig

FORTINET

ANALOG DEVICES

BCI

MITRE

Participating Schools

eCTF10
10 YEARS OF THE EMBEDDED CAPTURE THE FLAG

Amherst College	Florida International University	University of Michigan	Royal Melbourne Institute of Technology	University of California, Los Angeles
Amrita Vishwa Vidyapeetham University	Flinders University	Michigan State University	Rossview High School	University of Connecticut
Asia Pacific University of Technology and Innovation	Faculté des sciences Aïn Chock de Casablanca	Middlesex College	Rensselaer Polytechnic Institute	University of California, Santa Cruz
Binghamton University	Guru Gobind Singh Indraprastha University	Mineola High School	Rutgers University	Universidad de las Américas Ecuador
Blacksburg High School	George Mason University	Massachusetts Institute of Technology	Santa Ana College	University of Illinois Urbana-Champaign
Baldwin Wallace University	Georgia Institute of Technology	Morgan State University	Saddleback College	University of Massachusetts Amherst
Brigham Young University	Hampton University	Mount Si High School	Springfield-Clark County Career Technology Center	University of New Hampshire
CyberAegis	HTL Pinkafeld	Michigan Technological University	SDU University	University of New Haven
Colombe Academy of Technology	IFAPME	Mountain View High School	Southeast Missouri State University	University of Nebraska Omaha
City College of San Francisco	Indiana Institute of Technology	North Carolina State University	SESAME University	Universidad Politécnica de Cartagena
Chaitanya Engineering College	Indian Institute of Technology Dharwad	Northeastern University	San Francisco State University	University of South Alabama
Carnegie Mellon University	Indian Institute of Technology Indore	New Mexico State University	Smithtown High School West	United States Air Force Academy
Colorado State University	Indian Institute of Technology Madras	Nova Southeastern University	Singapore Management University	United States Coast Guard Academy
California State University, Fullerton	Institut Bisnis dan Teknologi Indonesia	Northern Virginia Community College	Southern University and A&M College	University of Texas at Arlington
California State University, Los Angeles	ISD 196	NorthWest Arkansas Community College	Sathyabama University of Science and Technology	The University of Texas at El Paso
Delaware Area Career Center	Istituto di Istruzione Secondaria Superiore Marie Curie	New York institute of Technology	Tri-City College Prep	The University of Tulsa
Dakota State University	Johnson C. Smith University	Oklahoma Christian University	University of Trento	University of Washington
El Paso Community College	Johns Hopkins University	The Ohio State University	Tennessee Tech University	Willis College
Ecole Royale de l'Air	Kansas State University	Pace University	Tufts University	Worcester Polytechnic Institute
Embry-Riddle Aeronautical University, Prescott	Kilgore College	Parkway Spark!	The University of Alabama	Xavier University
East Tennessee State University	KL University	Politeknik Kelapa Sawit Citra Widya Edukasi	University of Arkansas	
Fairport High School	Louisiana Tech University	The Pennsylvania State University	University of Colorado, Colorado Springs	
Florida A&M University	Lenoir Community College	Purdue University	University of Central Florida	
Florida Atlantic University	University of Maribor	Rice University	University of California, Irvine	

Original Competitors | **New Schools** | **2024 Winner**

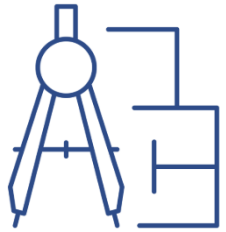
MITRE

Competition Phases

eCTF10
10 YEARS OF THE EMBEDDED CAPTURE THE FLAG

DESIGN PHASE

Teams design and implement systems that meet security and functionality requirements



HANDOFF

Organizers test each design for functionality

ATTACK PHASE

Teams analyze and attack each other's designs for points

2025

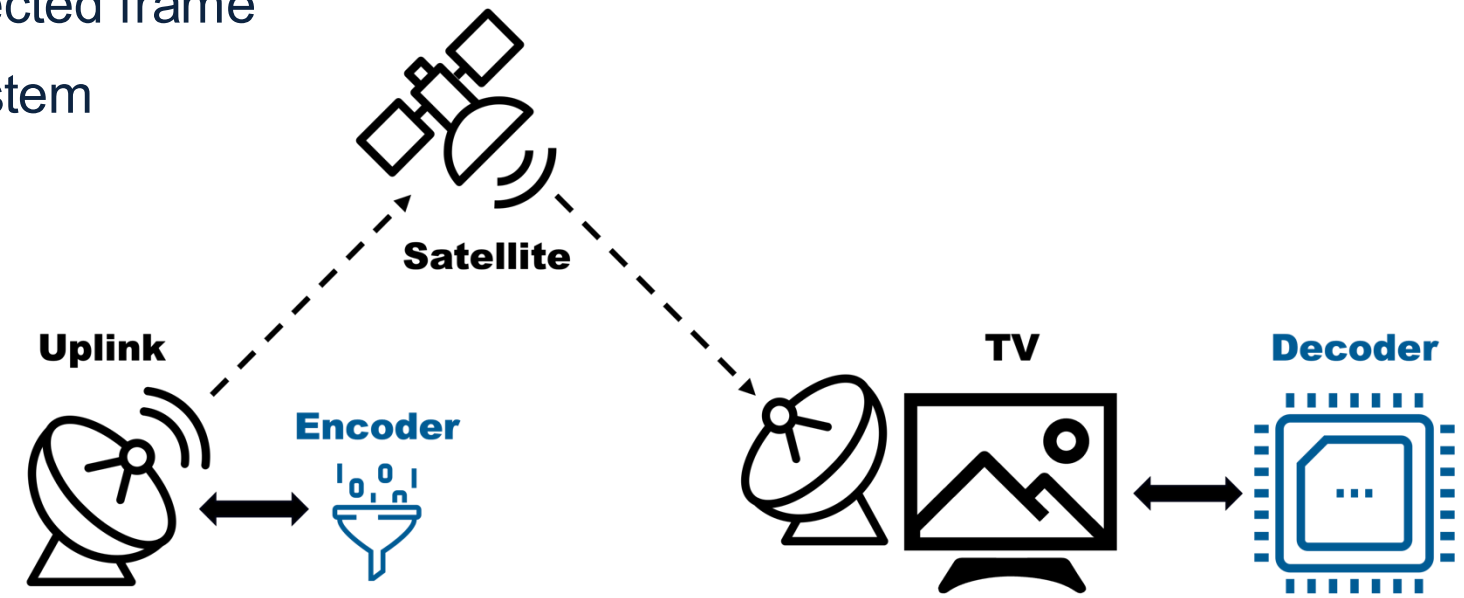
Sep	Dec	Jan 15	Feb 26	Apr 19	Apr 26
Team Registration Opens	Individual Competitor Registration Opens	eCTF Kickoff	Handoff Begins	Attack Phase Ends	Award Ceremony

MITRE

Challenge Overview

- Teams were tasked with designing a secure implementation for a satellite TV

- Teams delivered:
  - Encoder to generate protected frames
  - Decoder firmware to decode protected frame
  - Tools to build and manage the system



Background

eCTF10
10 YEARS OF THE EMBEDDED CAPTURE THE FLAG

CYBERSCOOP

Satellite hack on eve of Ukraine war was a coordinated, multi-pronged assault

The satellite hack that took the world by storm was more complex than initially thought, according to a Viasat executive.

NL#TIMES

Cyber attack on TV channel BabyTV: Toddlers suddenly exposed to Russian propaganda

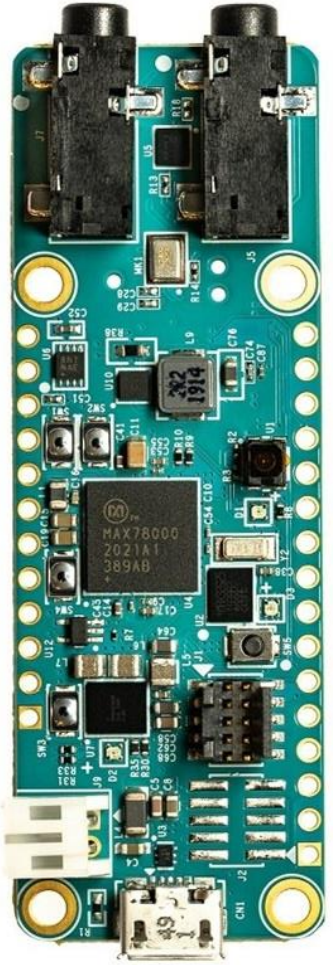
Forbes

The Tale of Captain Midnight, TV Hacker And Folk Hero

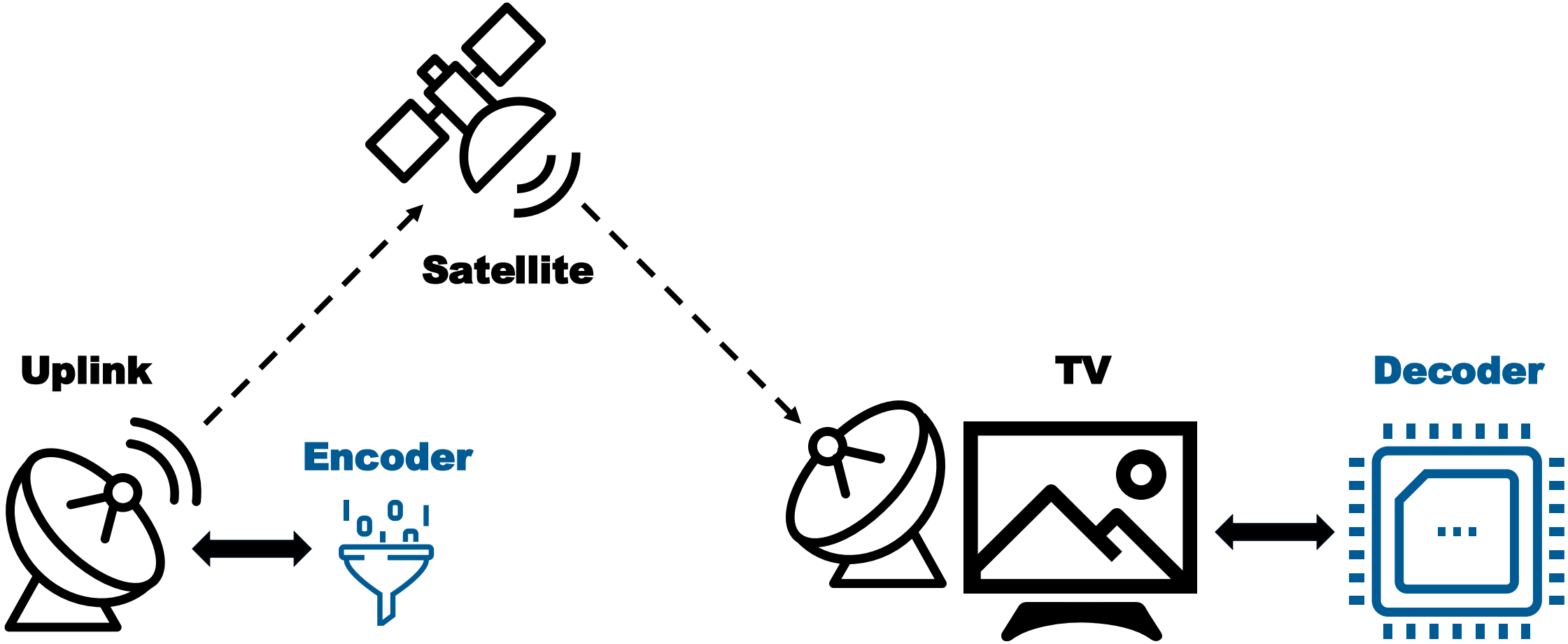
MITRE

Platform

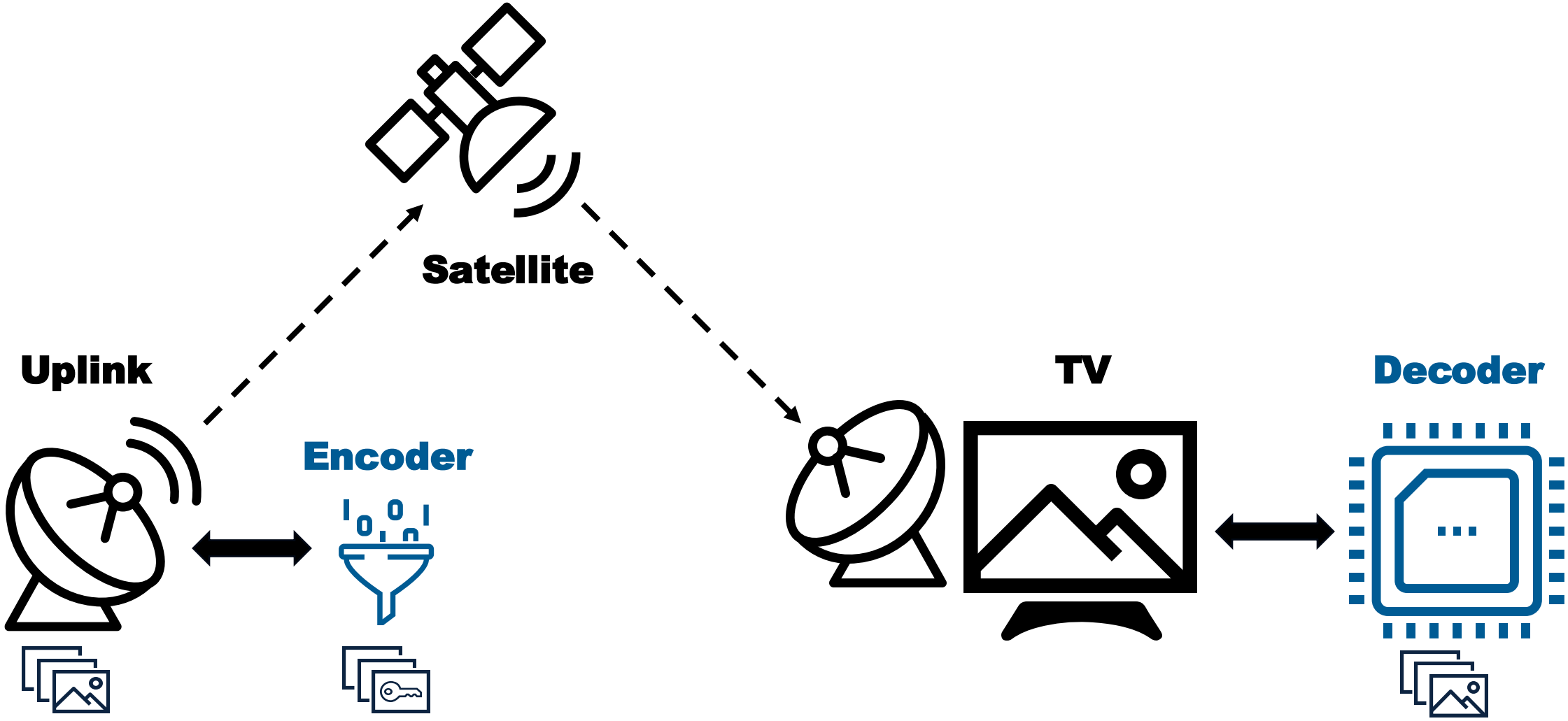
- Analog Devices MAX78000FTHR Evaluation Kit
  - Dual Core MCU
    - 100MHz Arm Cortex M4F
    - 60MHz RISC-V Coprocessor
  - 512KB Flash
  - 128KB SRAM
  - On-board peripherals
    - Camera, PMIC, DAPLink debugger, Audio CODEC, LEDs, Push buttons
  - UART, SPI, I2C



Scenario



Scenario



Security Requirements

1. An attacker should not be able to decode TV frames without a Decoder that has a valid, active subscription to that channel

2. The decoder should only decode valid TV frames generated by the Satellite System the Decoder was provisioned for

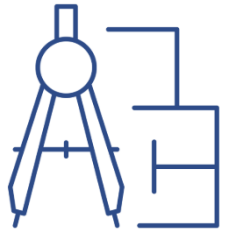
3. The Decoder should only decode frames with monotonically increasing timestamps

Competition Phases

eCTF10
10 YEARS OF THE EMBEDDED CAPTURE THE FLAG

DESIGN PHASE

Teams design and implement systems that meet security and functionality requirements



HANDOFF

Organizers test each design for functionality

ATTACK PHASE

Teams analyze and attack each other's designs for points

2025

Sep	Dec	Jan 15	Feb 26	Apr 19	Apr 26
Team Registration Opens	Individual Competitor Registration Opens	eCTF Kickoff	Handoff Begins	Attack Phase Ends	Award Ceremony

MITRE

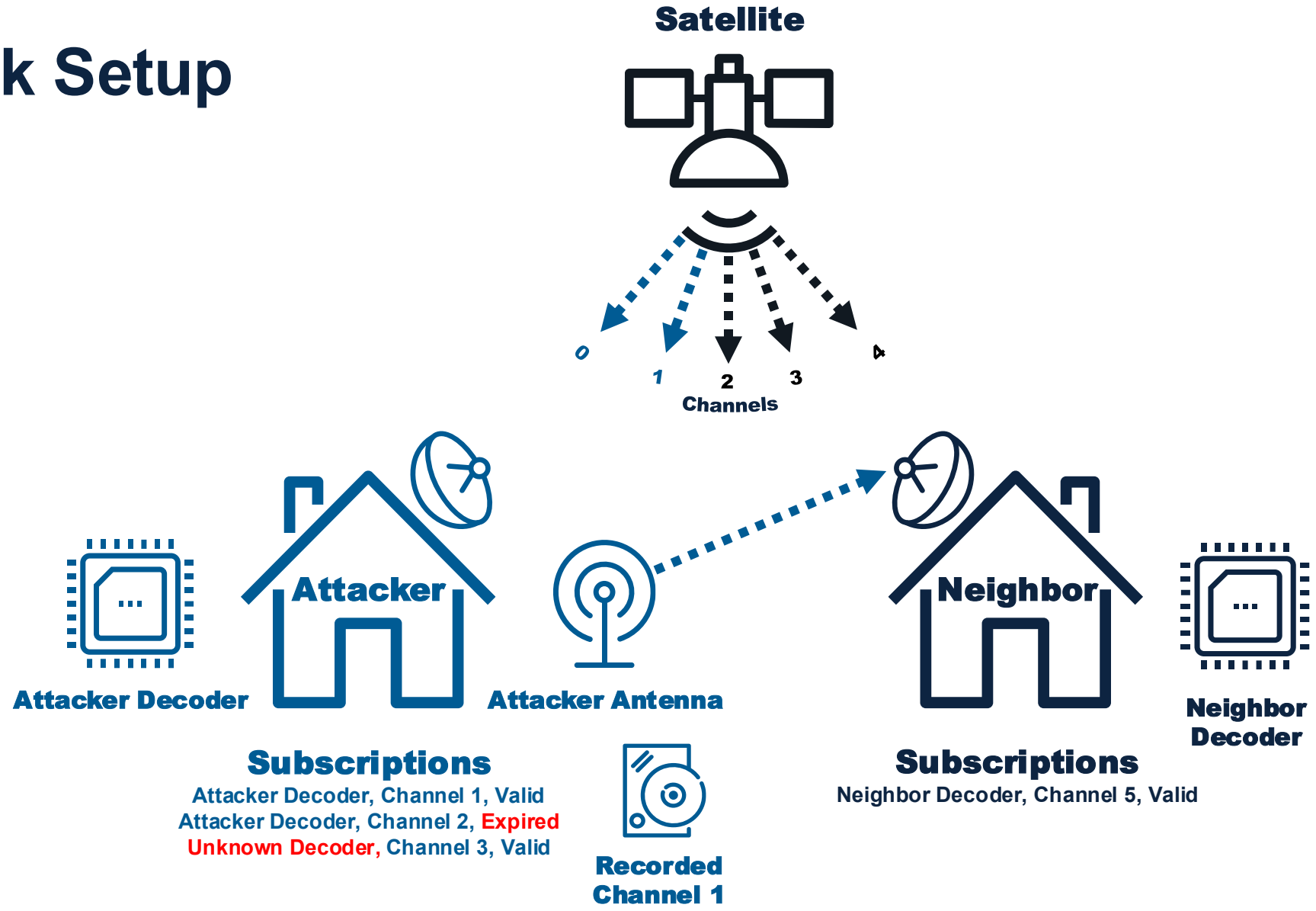
Attacker Resources

eCTF10
10 YEARS OF THE EMBEDDED CAPTURE THE FLAG

Attacking teams had access to:

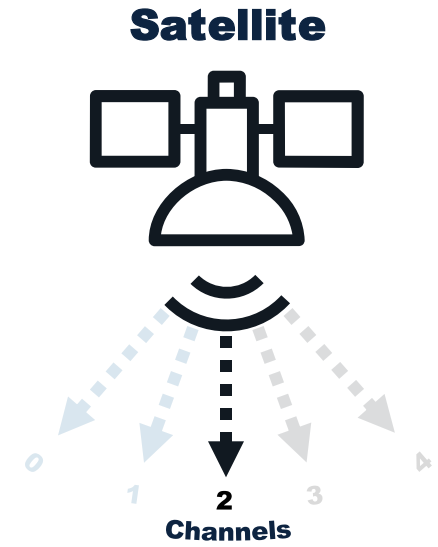
- Secure firmware images as shown on the Attacker Resources slide
  - These images can only be flashed onto Secure Attack Phase boards.
- All source code
  - Includes Decoder, Encoder, and Subscription Generator
  - With .git directly removed (no history)
- The most recent documentation
- The standard host tools

Attack Setup



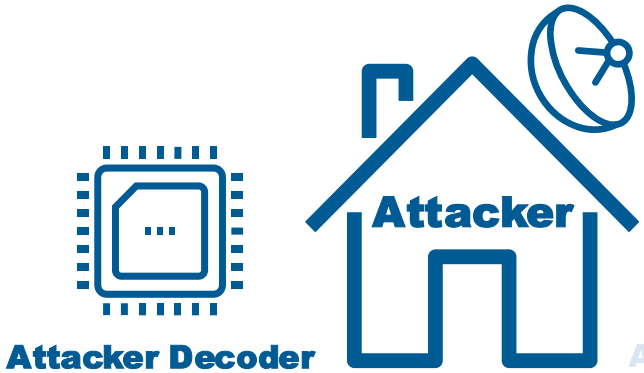
Expired Subscription

Read frames from a channel you have an expired subscription for



Pirated Subscription

Read frames from a channel you have a pirated subscription for



Subscriptions

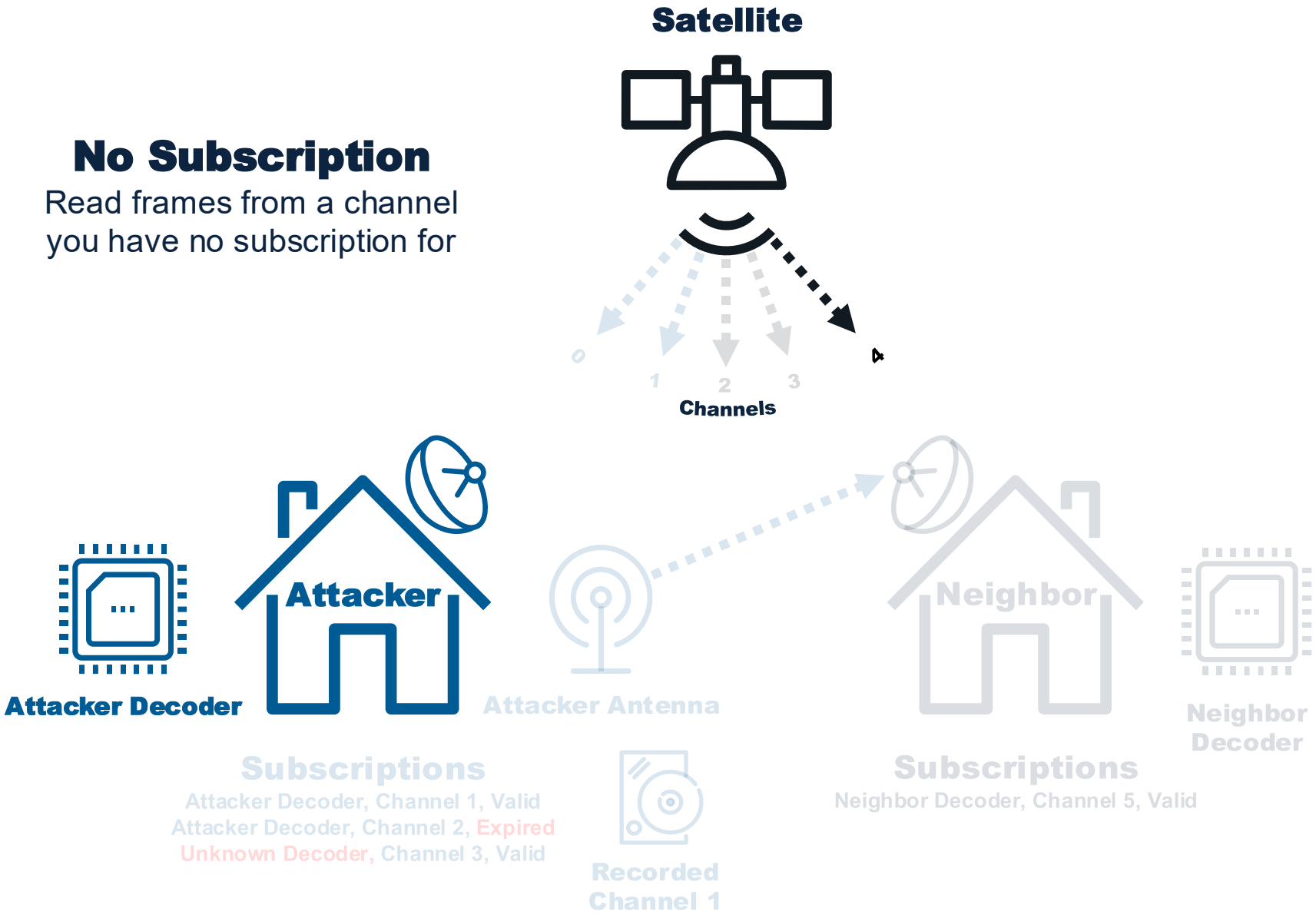
Attacker Decoder, Channel 1, Valid
Attacker Decoder, Channel 2, Expired
Unknown Decoder, Channel 3, Valid

Subscriptions

Neighbor Decoder, Channel 5, Valid

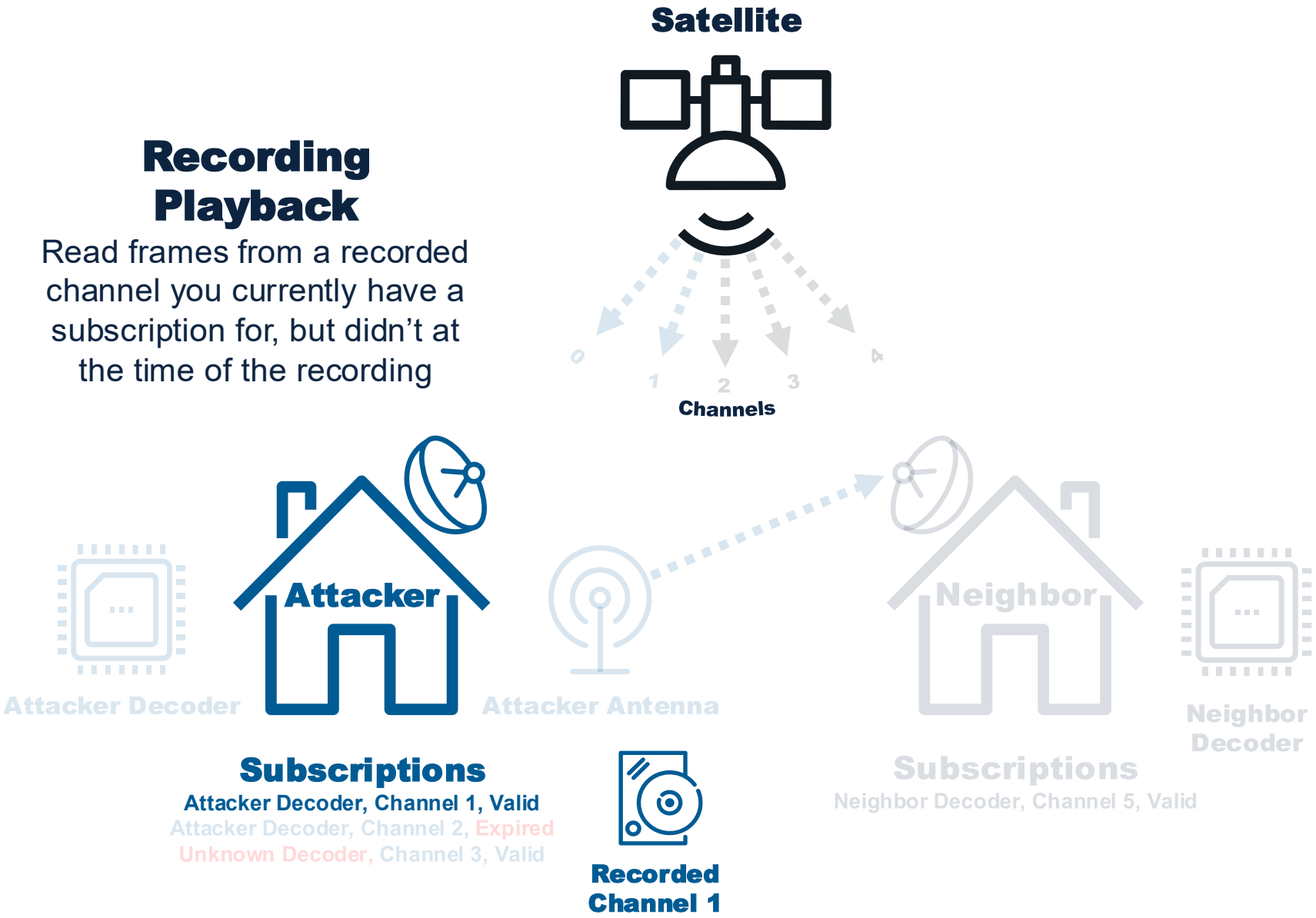
No Subscription

Read frames from a channel you have no subscription for



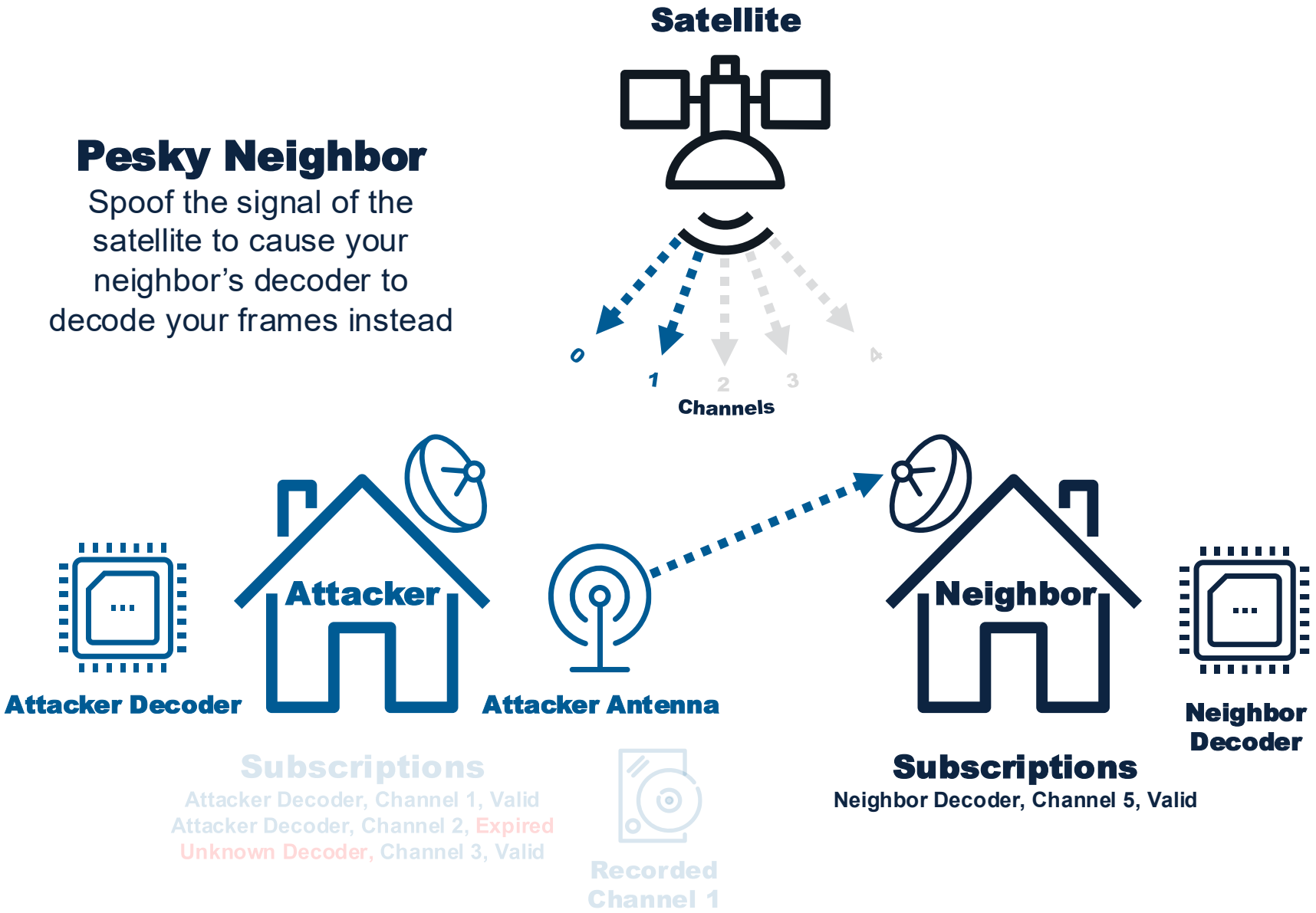
Recording Playback

Read frames from a recorded channel you currently have a subscription for, but didn't at the time of the recording



Pesky Neighbor

Spoof the signal of the satellite to cause your neighbor's decoder to decode your frames instead



John Petrozzelli

Director

MassCyberCenter

eCTF10

10 YEARS OF THE EMBEDDED CAPTURE THE FLAG

City College of San Francisco

Currently in 3rd place with 16,788 points

Not final score

Time (EST)	Title
15:30	Competition Overview
15:50	NEMC Guest Speaker – John Petrozzelli
16:00	City College of San Francisco
16:15	Mountain View High School
16:30	University of Michigan
16:45	Keynote – Mark Peters, Ph.D.
17:00	Refreshment Break
17:15	Purdue University
17:30	University of California, Los Angeles
17:45	University of Illinois Urbana-Champaign
18:00	Carnegie Mellon University
18:15	Award Presentations
18:25	Closing Remarks – Dan Walters
18:30	Networking Reception

MITRE

SATELLITE TV SYSTEM SECURITY

CCSF Team Presentation for MITRE eCTF

April 25, 2025

MEMBERS

Annat Koren

Liviu Tancau

Christine Hoang

Victor Cai

John Refling

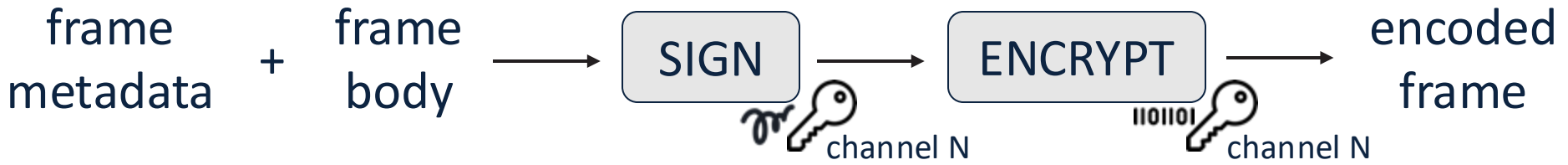
ADVISORS

Samuel Bowne
Cybersecurity

Elizabeth Biddlecome
Cybersecurity

Jonathan Potter
Computer Science

Design Overview



Key Defensive Feature: Random Delays

After reading from UART

```
read_packet();
random_delay();
```

Before cryptographic operation

```
random_delay();
decrypt(...);

random_delay();
verify_signature(...);
```

Before redundant operation

```
if (!is_subscribed())
    ...
random_delay();
if (!is_subscribed())
    ...
```

Some Additional Defensive Features

Decoy cryptographic operations

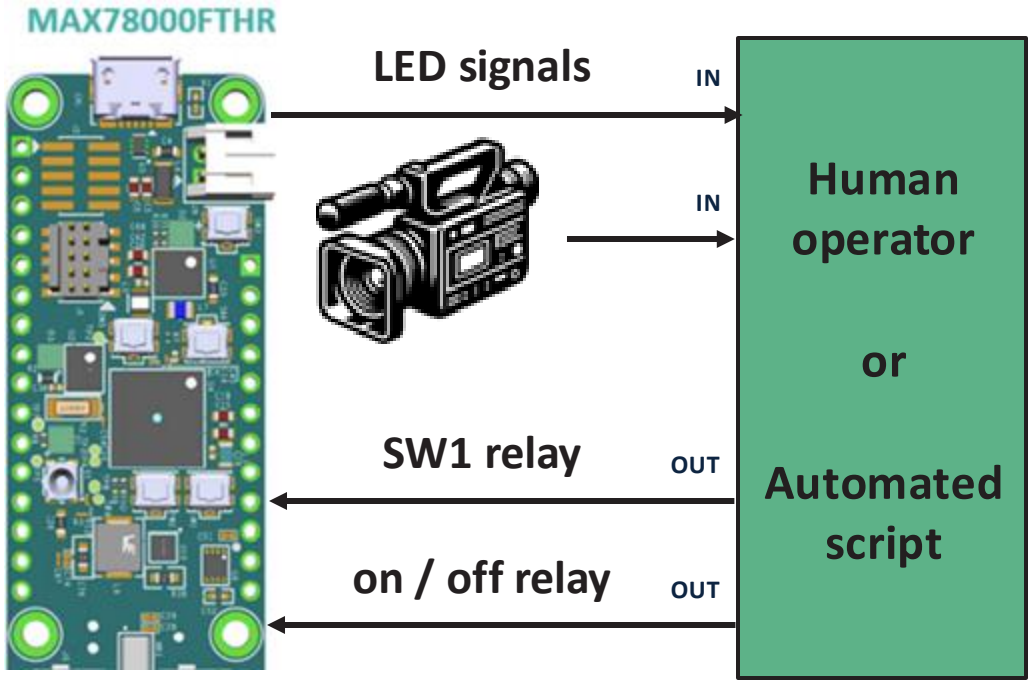
```
decrypt(data, random_key);
decrypt(data, real_key);
decrypt(data, random_key);
```

Constant time response

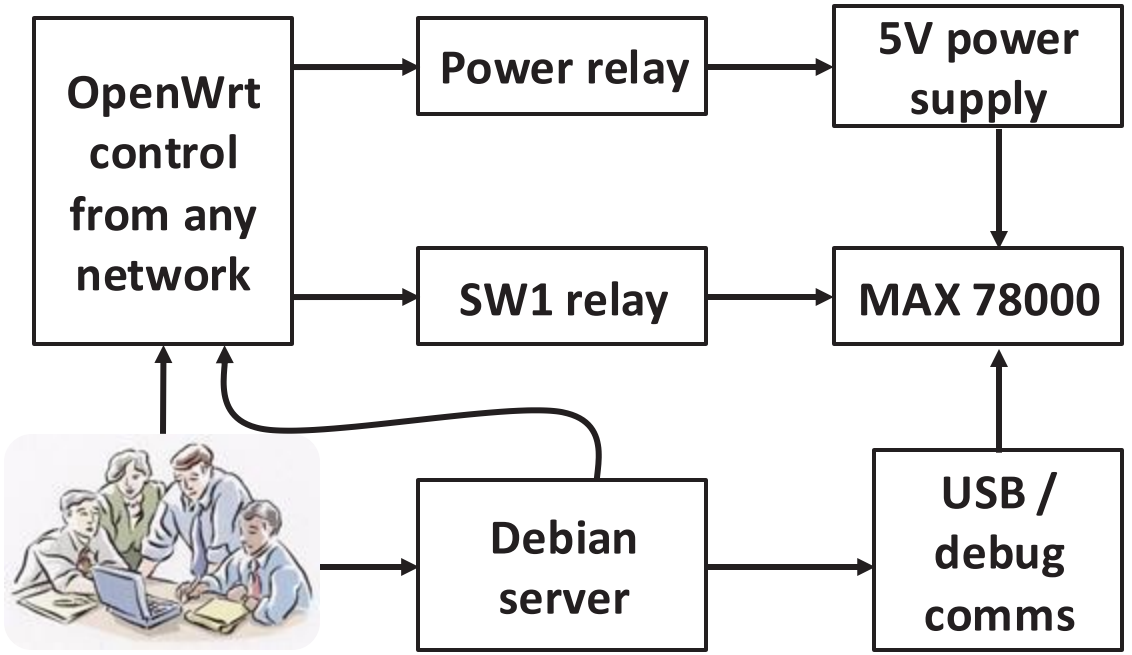
```
start_time = GetTime();
...
while (GetTime() - start_time
       < max_allowed_time) {}
```

Remote Control

Remote Control via SSH

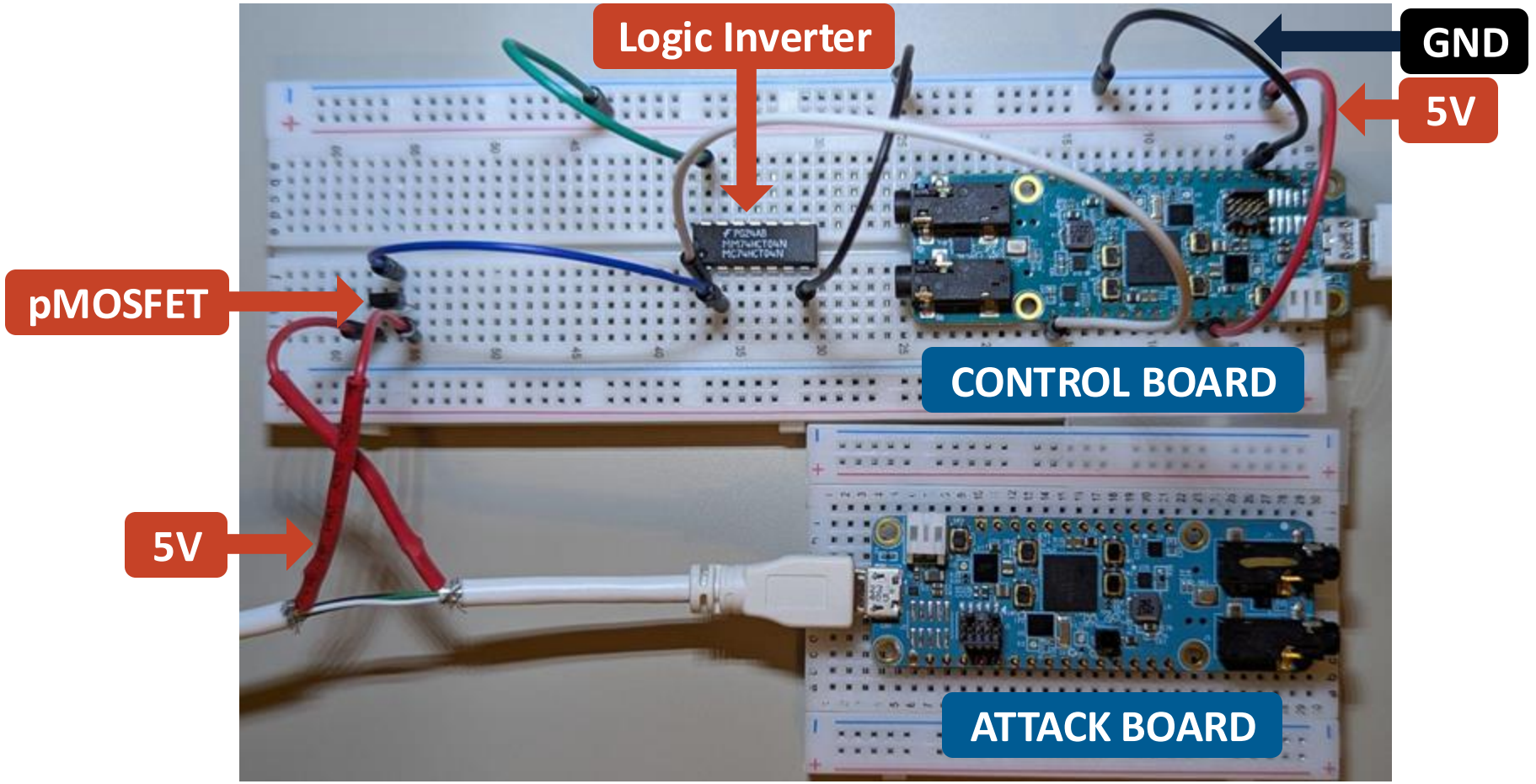


Feedback loop

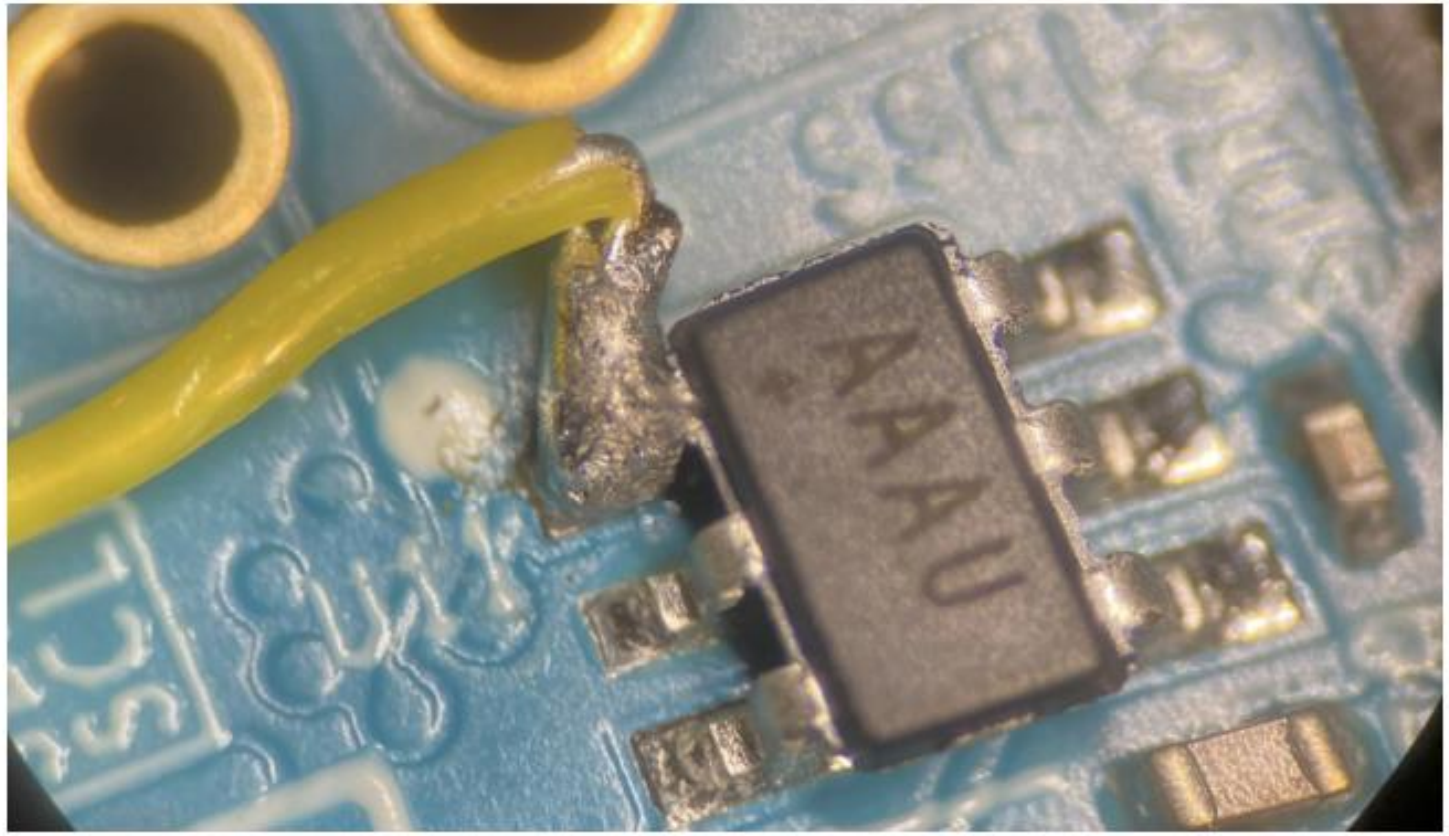


Infrastructure

Power Control of Attack Board



Reflash Control of Attack Board



Wire soldered to SW1 debouncer chip

Automation & Scripts

- Satellite recorder
- Frame replay
- Pesky Neighbor packager (for common vulnerabilities)
- Subscription / Frame modification (team-specific)
- Brute force timing-based attack
- Other bespoke attacks

Attacks

Initial Attacks

CRYPTOGRAPHIC ATTACKS

- AES-CBC IV modification
- AES-CTR XOR attacks
- AES-ECB block substitution
- AES-GCM authentication tag forgery
- Custom attacks based on teams' designs

MEMORY DISCLOSURE EXPLOITS

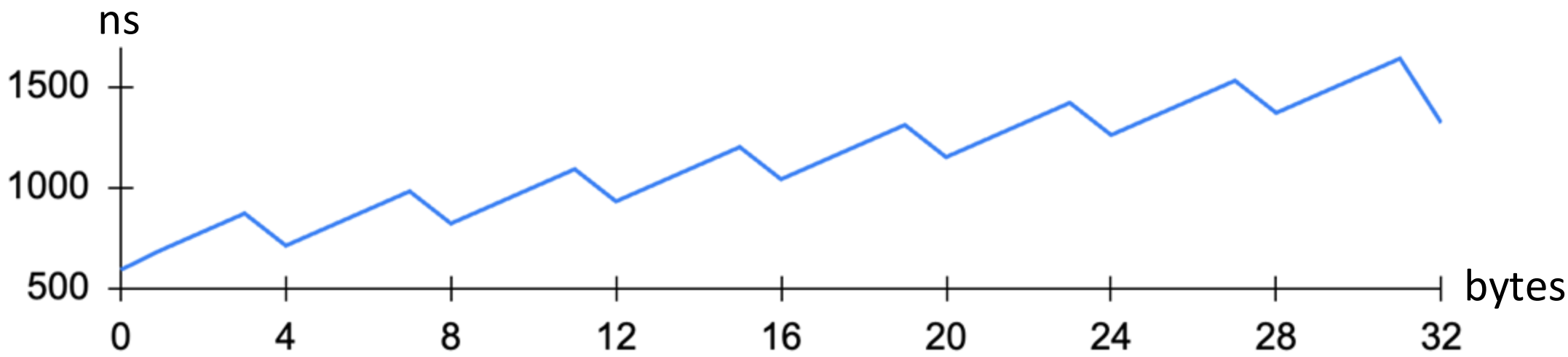
SIMPLE PESKY NEIGHBOR ATTACKS

Hardware Attacks

Vulnerable HMAC Verification

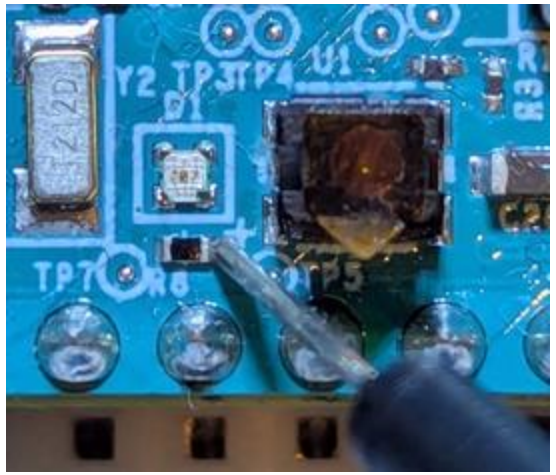
```
if (!memcmp(computed_hmac, extracted_hmac, LEN))
{
  return error;
}
```

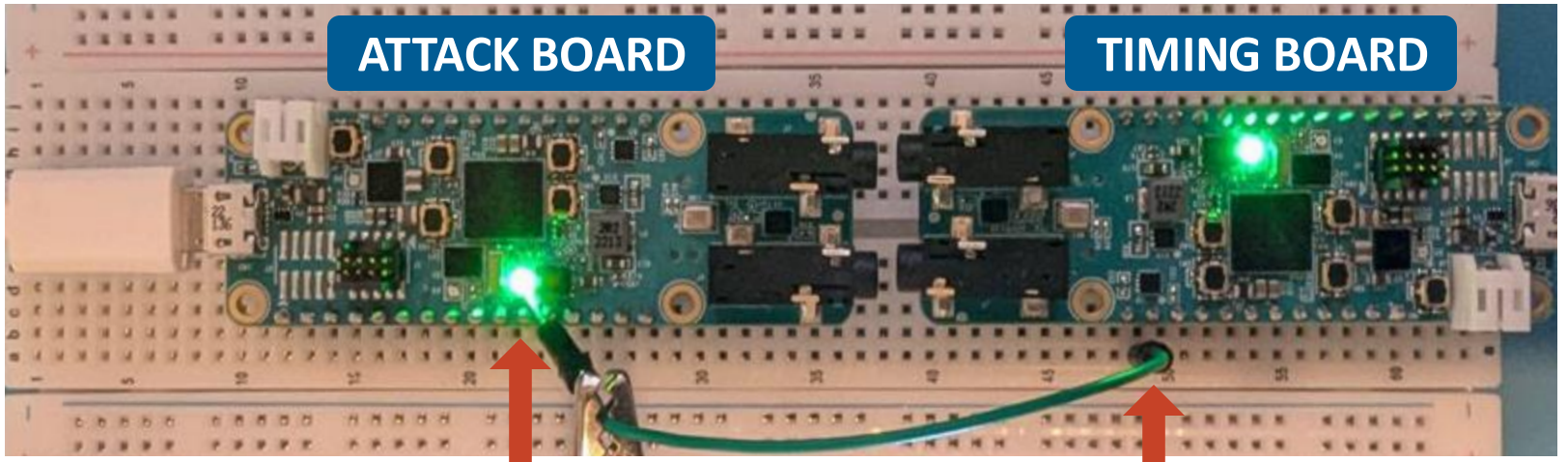
memcmp Timing Profile



32-byte memcmp processing time, by number of matching bytes.

Hardware Setup





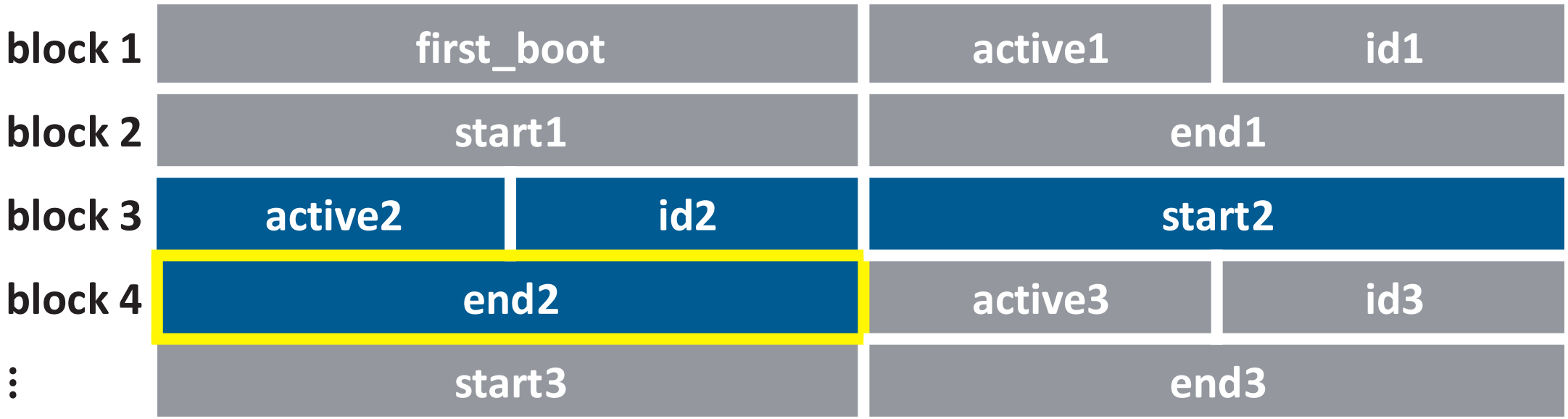
Brute-Forcing Algorithm

```
prefix = ""
for each HMAC byte {
  for guess in range(256) {
    times[guess] = send(forgery + prefix + guess + 0s)
  }
  prefix += find_outlier(times)
}
```

Hardware Attacks, part II

Subscription Data in Memory

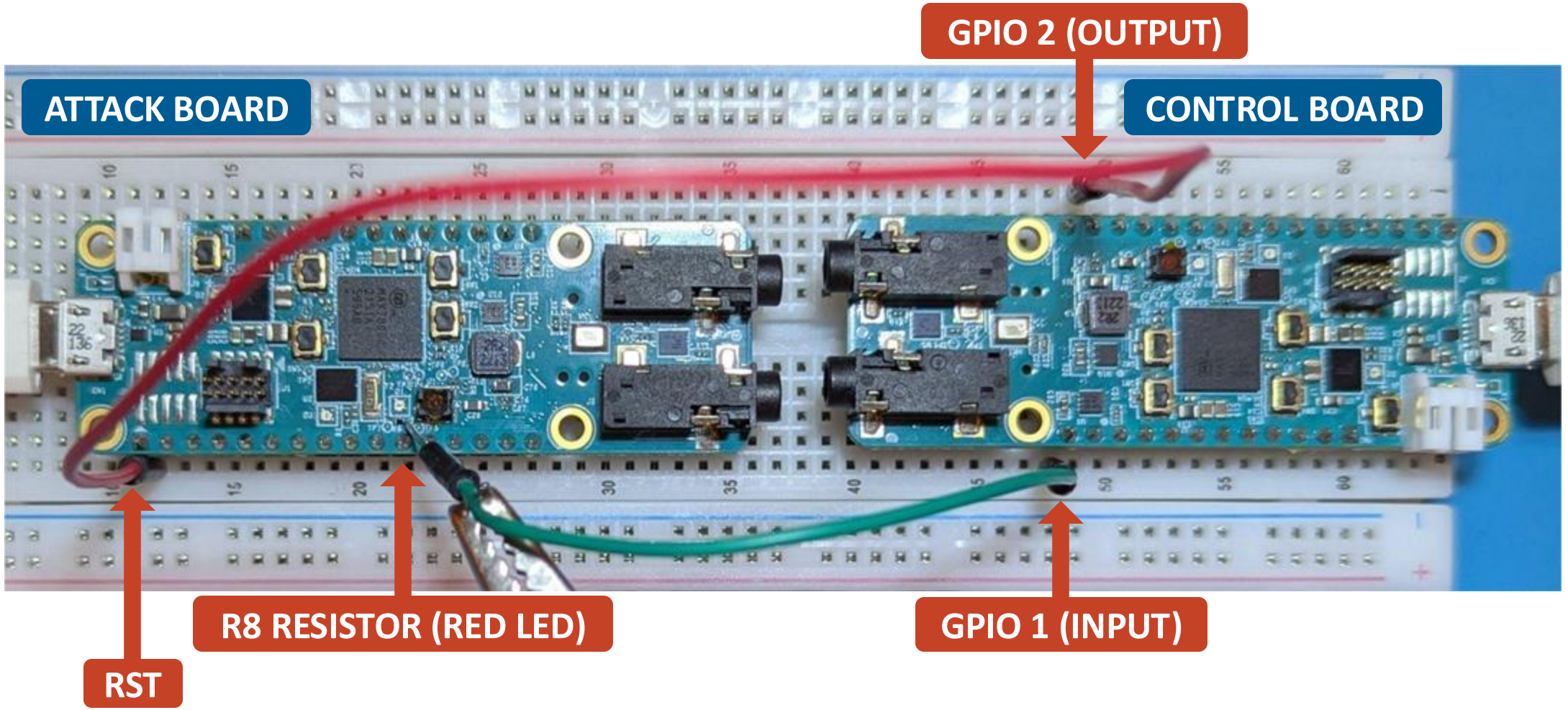
Memory Layout



Exploit

	Complete write	Partial write
block 1	efbeaddeffffffff0100000001000000	efbeaddeffffffff0100000001000000
block 2	6a1551a146350700a6d9e3a1096a0700	6a1551a146350700a6d9e3a1096a0700
block 3	0100000002000000656f668060c80600	0100000002000000656f668060c80600
block 4	9400a538eb090700000000000000000	ffffffffffffffffffffffffffffffff
⋮	ffffffffffffffffffffffffffffffff	ffffffffffffffffffffffffffffffff

Hardware Setup



Result

```
Found subscription: Channel 1 2028902307403114:2086914440288678
Found subscription: Channel 2 190916665687638 9:18446744073709551615
Found subscription: Channel 4294967295 18446744073709551615:18446744073709551615
Found subscription: Channel 4294967295 18446744073709551615:18446744073709551615
Found subscription: Channel 4294967295 18446744073709551615:18446744073709551615
Found subscription: Channel 4294967295 18446744073709551615:18446744073709551615
Found subscription: Channel 4294967295 18446744073709551615:18446744073709551615
Found subscription: Channel 4294967295 18446744073709551615:18446744073709551615
List successful
```

Takeaways

What We Learned

1. **Random delays are *extremely* valuable.**

What We Learned

1. **Random delays are *extremely* valuable.**

1. **Automation will help you capture flags faster.**

What We Learned

1. **Random delays are *extremely* valuable.**

1. **Automation will help you capture flags faster.**

1. **Embrace the tree!** 

What We Learned

1. **Random delays are *extremely* valuable.**

1. **Automation will help you capture flags faster.**

1. **Embrace the tree!** 

1. **You can accomplish a lot with just the provided hardware.**

Questions?

eCTF10

10 YEARS OF THE EMBEDDED CAPTURE THE FLAG

Mountain View High School

Currently in 10th place with 12,167 points

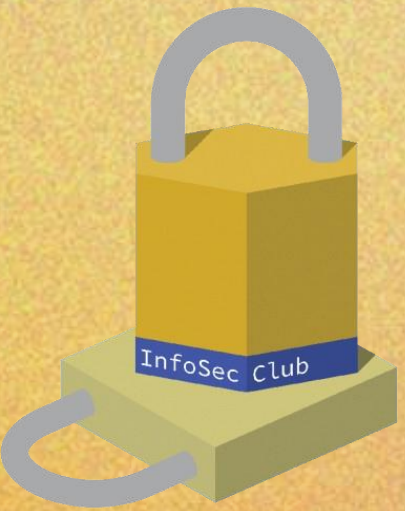
Not final score

Time (EST)	Title
15:30	Competition Overview
15:50	NEMC Guest Speaker – John Petrozzelli
16:00	City College of San Francisco
16:15	Mountain View High School
16:30	University of Michigan
16:45	Keynote – Mark Peters, Ph.D.
17:00	Refreshment Break
17:15	Purdue University
17:30	University of California, Los Angeles
17:45	University of Illinois Urbana-Champaign
18:00	Carnegie Mellon University
18:15	Award Presentations
18:25	Closing Remarks – Dan Walters
18:30	Networking Reception

MITRE

MVHS

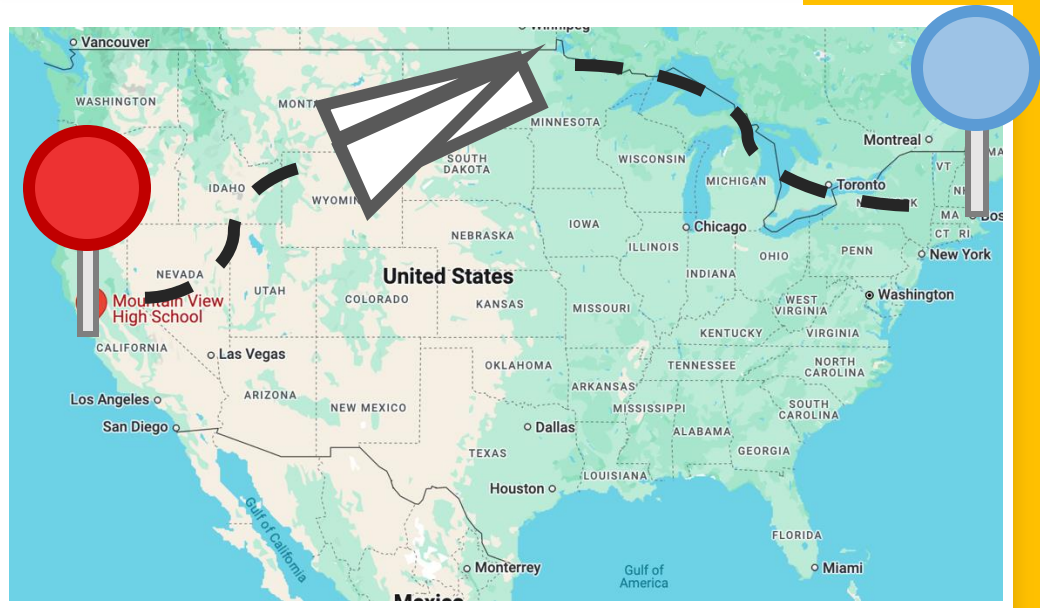
Mountain View High School



Ryan Yin, Arthur Cheong, Jeremy Yu, Jayden Stenfort, Trent McCauley, Justin Cheong, Ryan Chan, Asher Copeland, Ethan Wing, Jeff Zhang

Who are we?

- High school students from Mountain View, CA
  - Silicon Valley
    - "Heart of Google"
  - ~2200 students



Who are we?

- Information Security Club
  - 100+ member club
  - (almost) 3 years old
  - eCTF 2025 is our first time

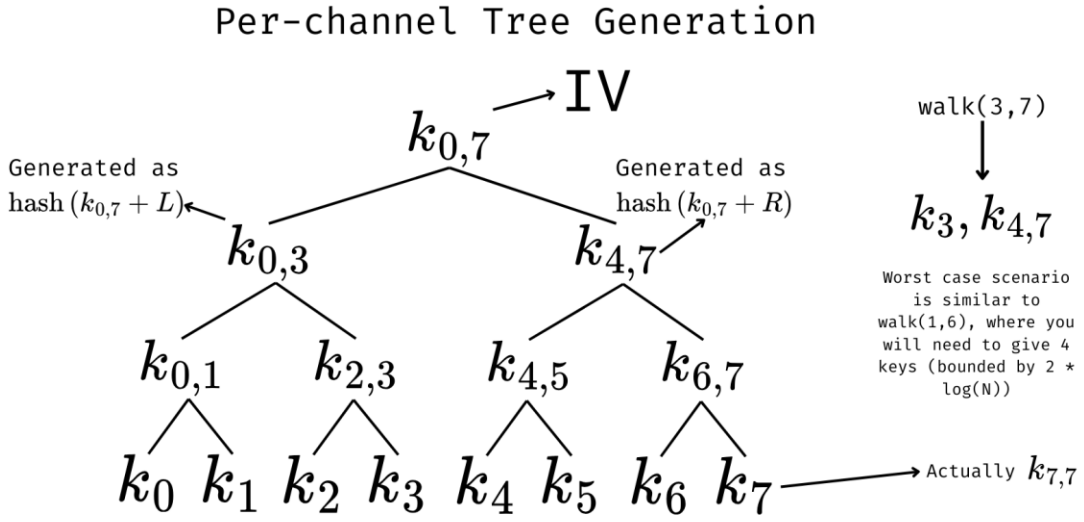
Design

Attack

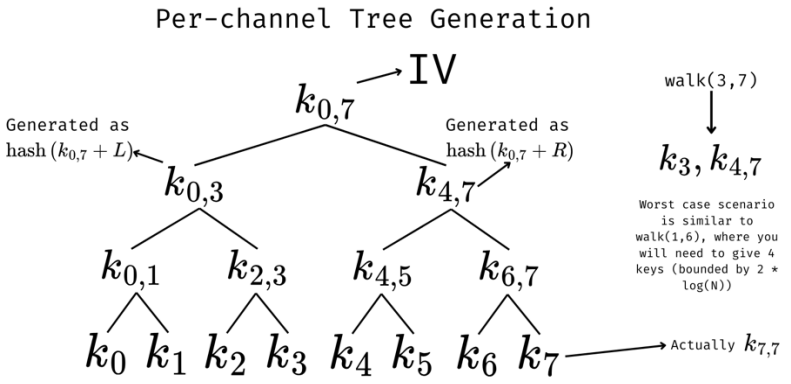
Comments

Design

- Per-channel Binary Tree
  - Uses a unique key to encrypt every timestamp.
  - Zero-trust architecture.
  - Economic (maximum keys bounded by 2 * log(N), which is at most 128 for 64-bit timestamps



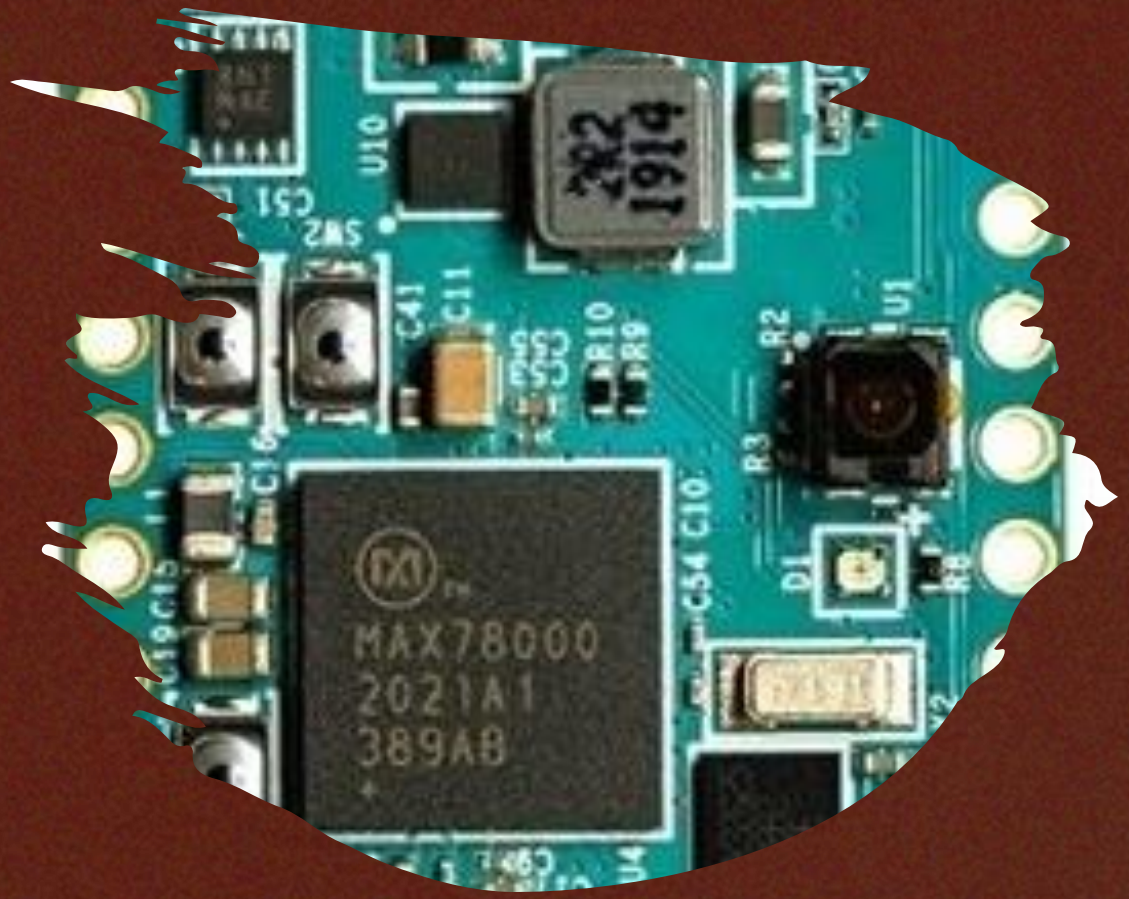
Design



- Each node is a key
  - child nodes derived with hash of key + data
- Keys revealed are then encrypted with key derived from the Decoder ID + secret data
- TV frames are encrypted with the key derived from the tree, then signed

Other Design Decisions

- ✓ Rust
  - Memory Safety
- ✓ Ascon-HashA, Ascon-128A
  - Security
  - Good with embedded devices
  - Efficiency
- ✓ Ed25519
  - Resistance to side channel attacks and poor random number generation



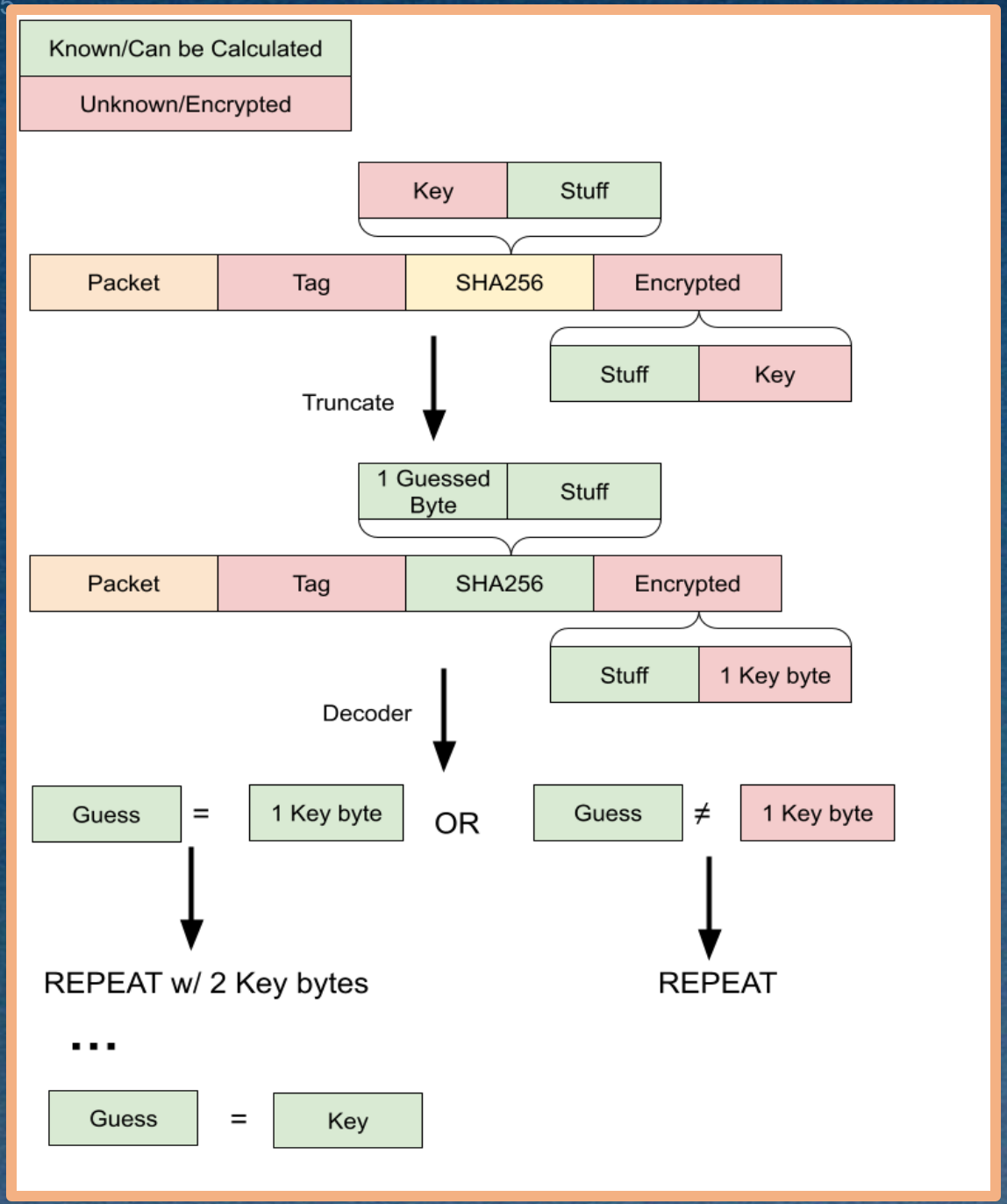
Design

Attack

Comments

Attack Highlight 1

- **Design Security Features**
  - ChaCha20-Poly1305
  - SHA256 Hash w/secret data
  - Encrypted data includes per-channel key, but everything is encrypted with a main key
- **Our Attack**
  - ChaCha20-Poly1305 ignored
  - Take subscription update, truncate
    - Brute force decoder
    - Use TV Frame decode function
  - Use bit-flipping and mint correct SHA hash



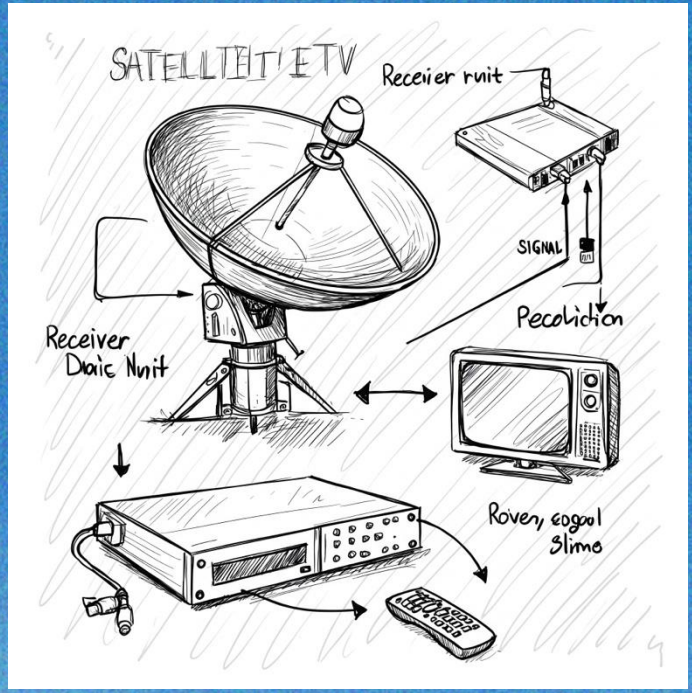
Attack

IMPACTS

- Given any subscription that has ever been subscribed to a channel, an attacker can decode any frame in that channel
- Web pirating

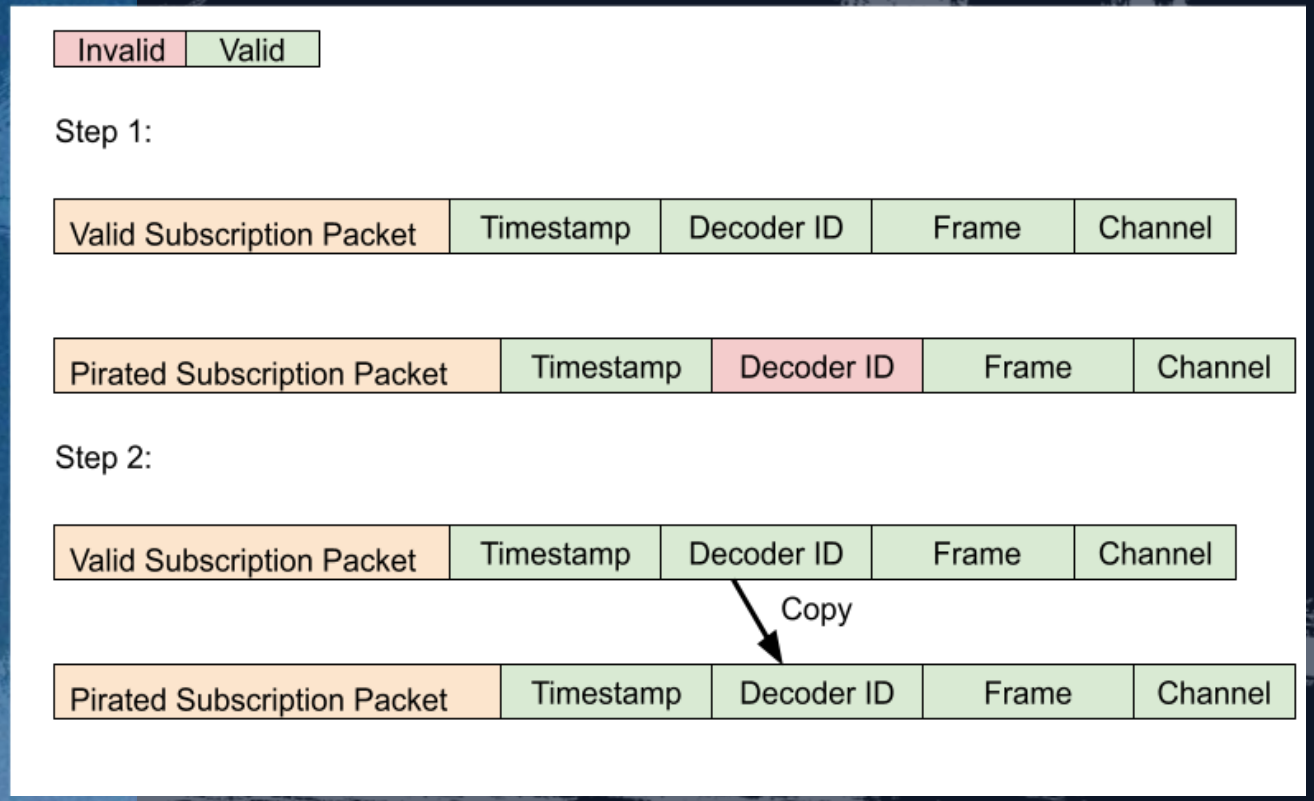
COUNTERMEASURES

- Utilize ChaCha20-Poly1305 tag



Attack Highlight 2

- Same key is used for all subscriptions
- AES Mode ECB
- Do Block Switching



Attack Impacts & Countermeasures

- Web Pirating
- Roku/Netflix loses lots of money

- Use a different AES version
- Use a signature

```
if (memcmp(body->sig, packet_signature, HMAC_LEN))
{
    sprintf(
        output_buf,
        "Authenticity Check Failed.\nSignature: %.8x%.8x%.8x%.8x%.8x%.8x%.8x%.8x",
        ((uint32_t *)packet_signature)[0], ((uint32_t *)packet_signature)[1],
        ((uint32_t *)packet_signature)[2], ((uint32_t *)packet_signature)[3],
        ((uint32_t *)packet_signature)[4], ((uint32_t *)packet_signature)[5],
        ((uint32_t *)packet_signature)[6], ((uint32_t *)packet_signature)[7]);
    print_error(output_buf);
```

Special Attack

- Misunderstanding Implementations
  - Per-channel timestamps
  - Strictly monotonically increasing vs monotonically increasing
  - Emergency Channel exemption
  - Hard-coded keys
  - Uncommented debug lines

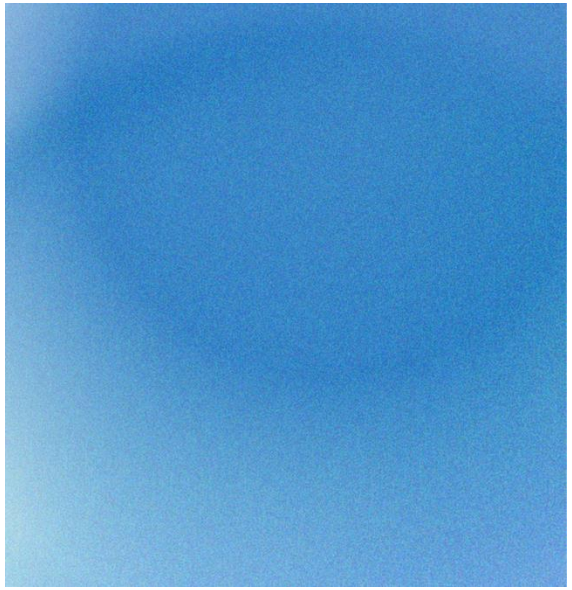
```
"aes_key": "24f6754364b954cfb33671c831da582edb88b942442357d9c95ae94e9f8fe20d",
"cbc_iv": "c6d30c124fbe47e86a02fc2a64b230bf"
```

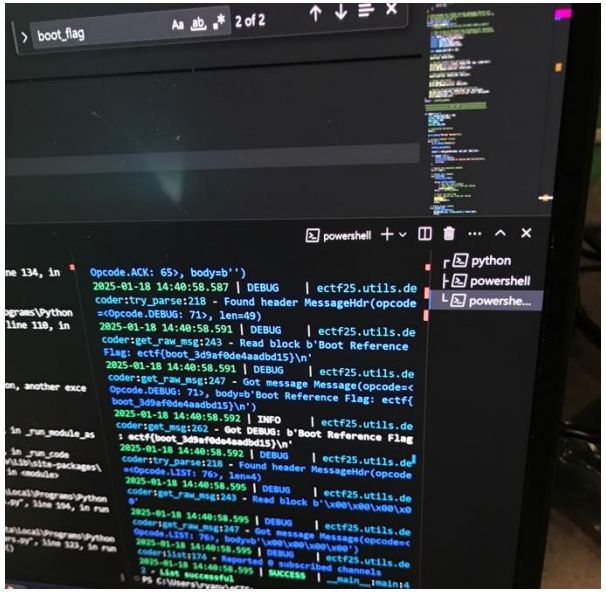
```
.current_timestamp >= timestamp)
```

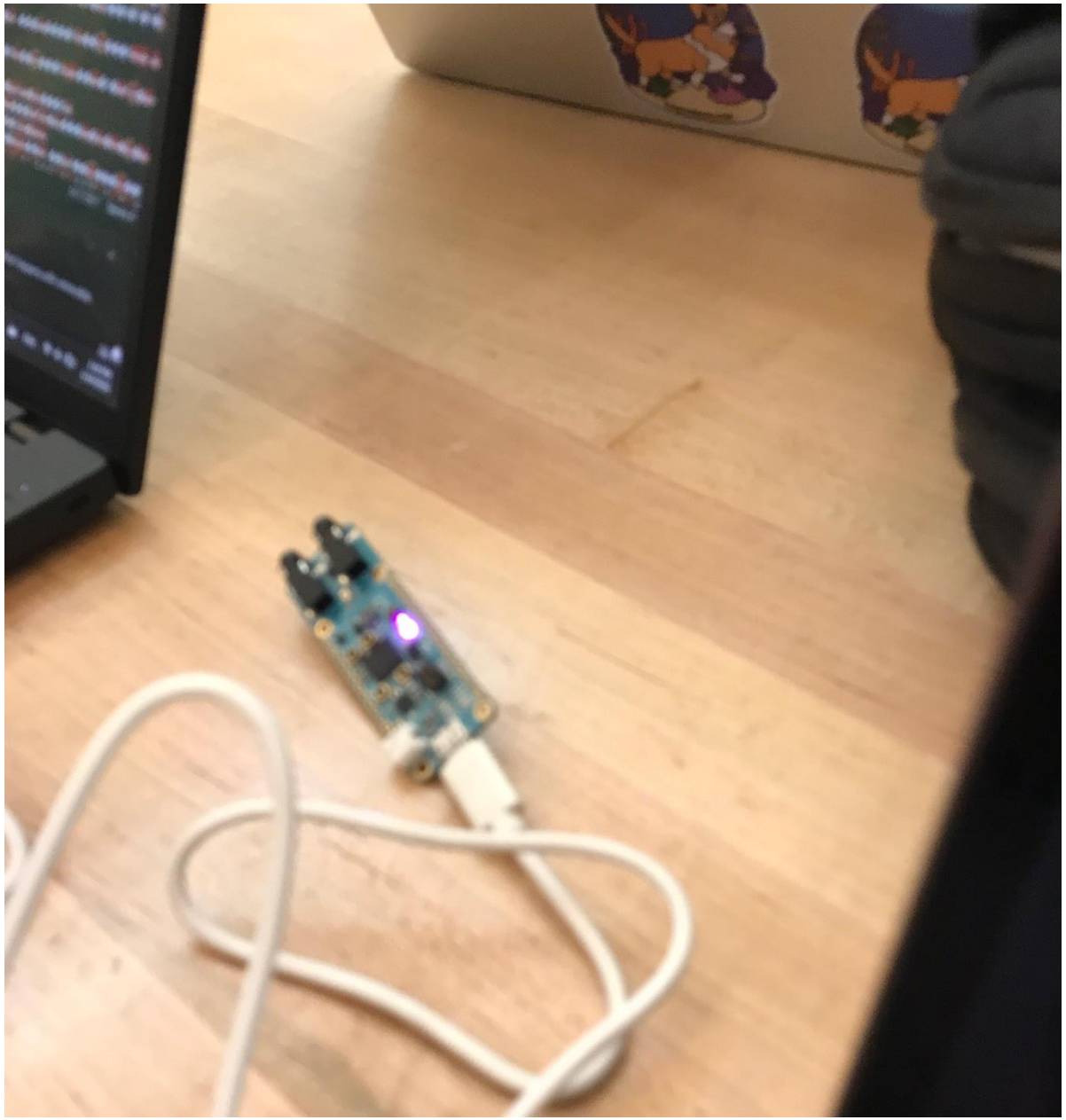
```
// Special handling for emergency channel - always decode regardless of timestamp
if (channel == EMERGENCY_CHANNEL) {
```

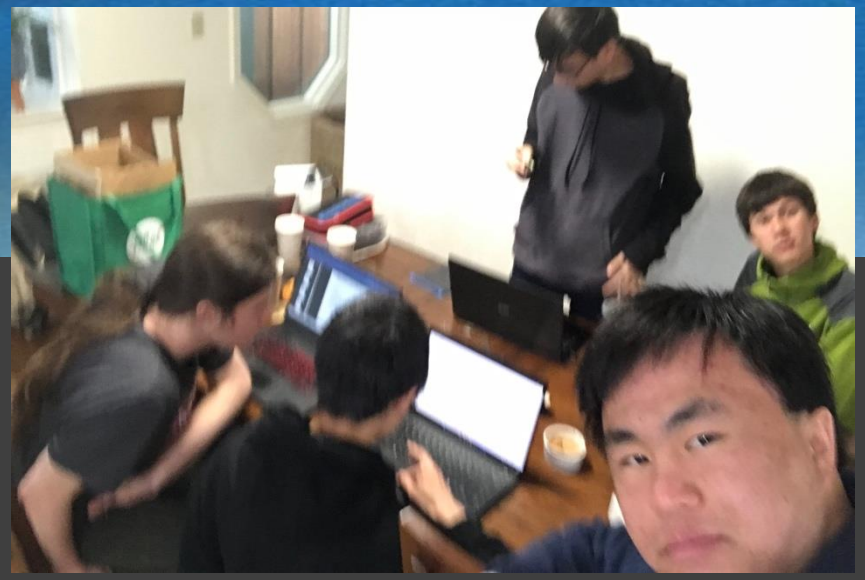
```
int tracker_index = -1;
for (int i = 0; i <= MAX_CHANNEL_COUNT; i++) {
    if (last_timestamps[i].channel_id == channel) {
        tracker_index = i;
        break;
    }
}
```


PENN



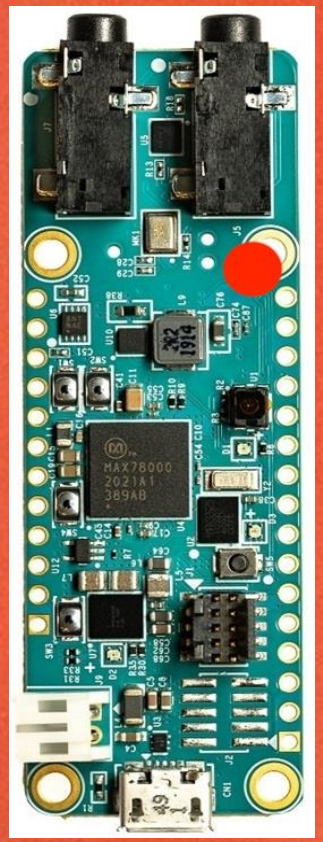

boot_flag
2 of 2
powershell
python
powershell
powershe...
Opcode.ACK: 65>, body=b'')
2025-01-18 14:40:58.587 | DEBUG | ectf25.utils.de
coder:try_parse:218 - Found header MessageHdr(opcode
=<Opcode.DEBUG: 71>, len=49)
2025-01-18 14:40:58.591 | DEBUG | ectf25.utils.de
coder:get_raw_msg:243 - Read block b'Boot Reference
Flag: ectf{boot_3d9af0de4aadbd15}\n'
2025-01-18 14:40:58.591 | DEBUG | ectf25.utils.de
coder:get_raw_msg:247 - Got message Message(opcode=<
Opcode.DEBUG: 71>, body=b'Boot Reference Flag: ectf{
boot_3d9af0de4aadbd15}\n')
2025-01-18 14:40:58.592 | INFO | ectf25.utils.de
coder:get_msg:262 - Got DEBUG: b'Boot Reference Flag
: ectf{boot_3d9af0de4aadbd15}\n'
2025-01-18 14:40:58.592 | DEBUG | ectf25.utils.de
coder:try_parse:218 - Found header MessageHdr(opcode
=<Opcode.LIST: 76>, len=4)
2025-01-18 14:40:58.595 | DEBUG | ectf25.utils.de
coder:get_raw_msg:243 - Read block b'\x00\x00\x00\x0
0'
2025-01-18 14:40:58.595 | DEBUG | ectf25.utils.de
coder:get_raw_msg:247 - Got message Message(opcode=<
Opcode.LIST: 76>, body=b'\x00\x00\x00\x00')
2025-01-18 14:40:58.595 | DEBUG | ectf25.utils.de
coder:list:174 - Reported 0 subscribed channels
2025-01-18 14:40:58.595 | SUCCESS | __main__:main:14
2 - List successful






MYSTERY
SCIENCE
THEATER

Design



Attack

Comments

Comments

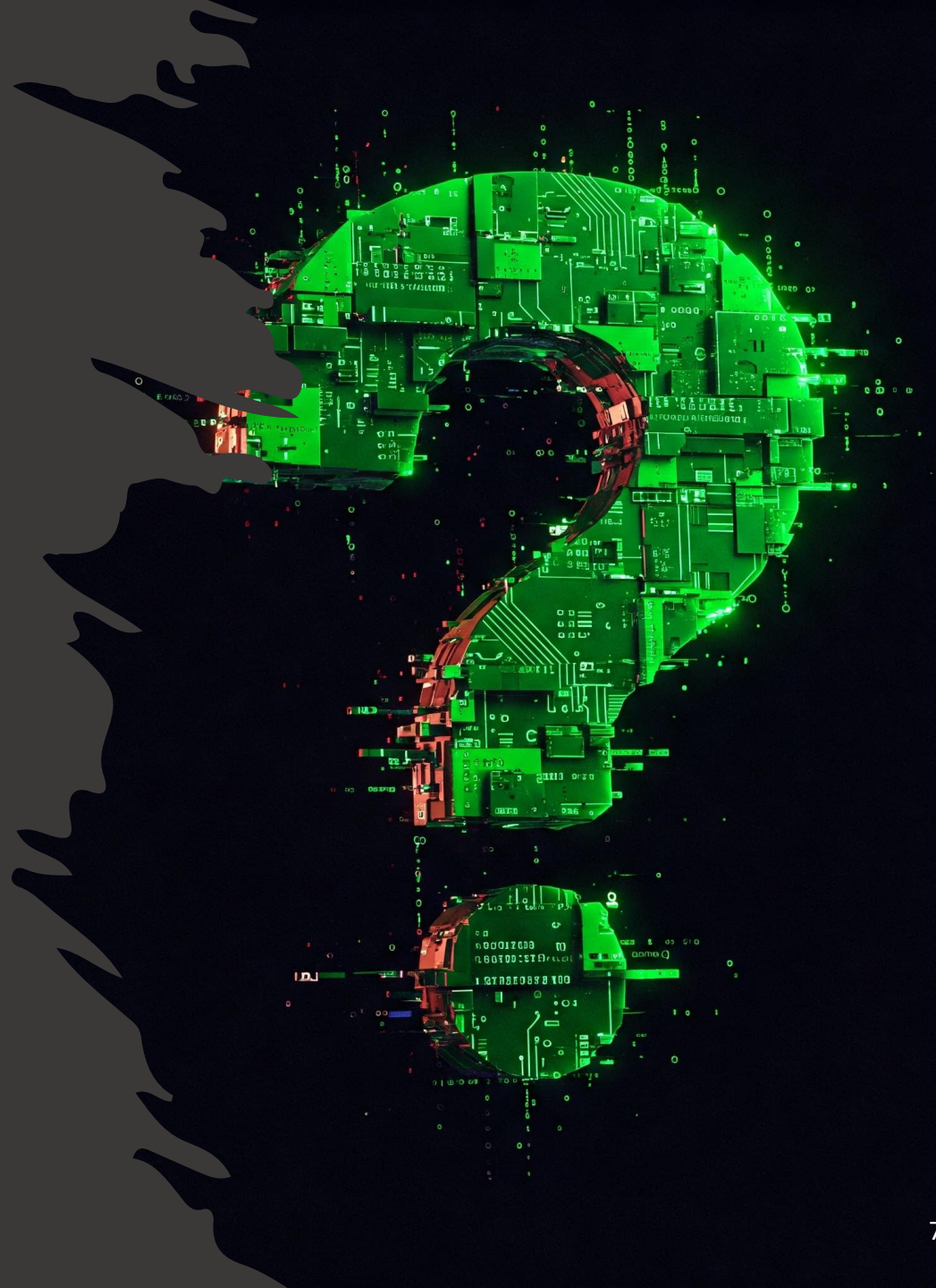
Congratulations to everyone!

Elsewise:

- Signatures seemed to be very important
- We learned a lot about cryptography, and how different ciphers worked
- Embedded devices are a lot of work :)

Questions?

Credit: slidescarnival.com



eCTF10

10 YEARS OF THE EMBEDDED CAPTURE THE FLAG

University of Michigan

Currently in 4th place with 16,445 points

Not final score

Time (EST)	Title
15:30	Competition Overview
15:50	NEMC Guest Speaker – John Petrozzelli
16:00	City College of San Francisco
16:15	Mountain View High School
16:30	University of Michigan
16:45	Keynote – Mark Peters, Ph.D.
17:00	Refreshment Break
17:15	Purdue University
17:30	University of California, Los Angeles
17:45	University of Illinois Urbana-Champaign
18:00	Carnegie Mellon University
18:15	Award Presentations
18:25	Closing Remarks – Dan Walters
18:30	Networking Reception

MITRE

WolvSec (Mich)

University of Michigan

eCTF10

10 YEARS OF THE EMBEDDED CAPTURE THE FLAG

Our Team

Advised by Prof. Paul Grubbs and Matthew Bernath

Andrew Plotner
Andre Quimper Osores
Brennen Daudlin
Matthew Dowling
Elliot Kupchik
Ethan McKean
Alexandra Enders
Faizan Darsot
Muhammad Khan
Hawon Cho
Charlie Herz
Jacob Marchionda
Jackson Donaldson
Katelyn Ha
Liam Domegan
Angela Matta
Nikye Nixon
Nicholas Gamota
Jeremy Shere
Shin Lee
Sahil Sawant
Shinjo Satoh
Sudeepti Rao
Tanay Sharma
Tanishka Nalawade
Sage Ullman
Allie Yuan
Ian Zhang

Outline

- Secure Design
  - Design Features
  - Protocol Description
  - Design Highlight: Scoped Key Distribution
- Attack Phase Highlights
  - Collecting Ciphertext Plaintext Pairs
  - Timing Side Channels
- Final Comments and Lessons Learned

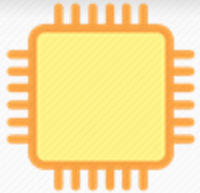
Design Features

Cryptography

ASCON AEAD Encryption

Tree Based Key Derivation Scheme

Ed25519 Signatures



Software Hardening

TRNG for Stack Randomization

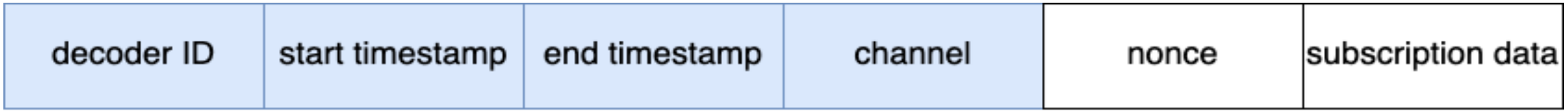
Set Memory as NX with MPU

Stack Protector + Bounds Checking

Mich

Protocol Description

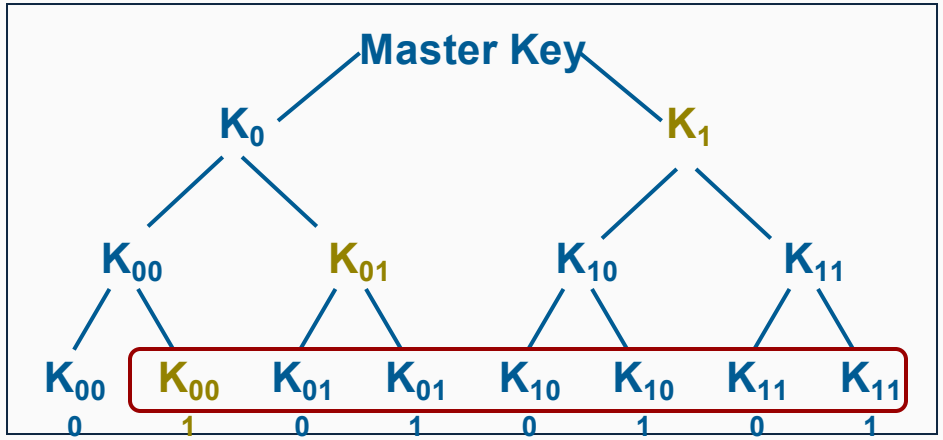
Frame data contains the frame payload encrypted with a unique key and signed with an asymmetric key. Metadata is sent unencrypted as Authenticated Data.



Subscription data contains a minimal key set derived from a binary key tree and keys are encrypted via a *decoder specific* subscription key

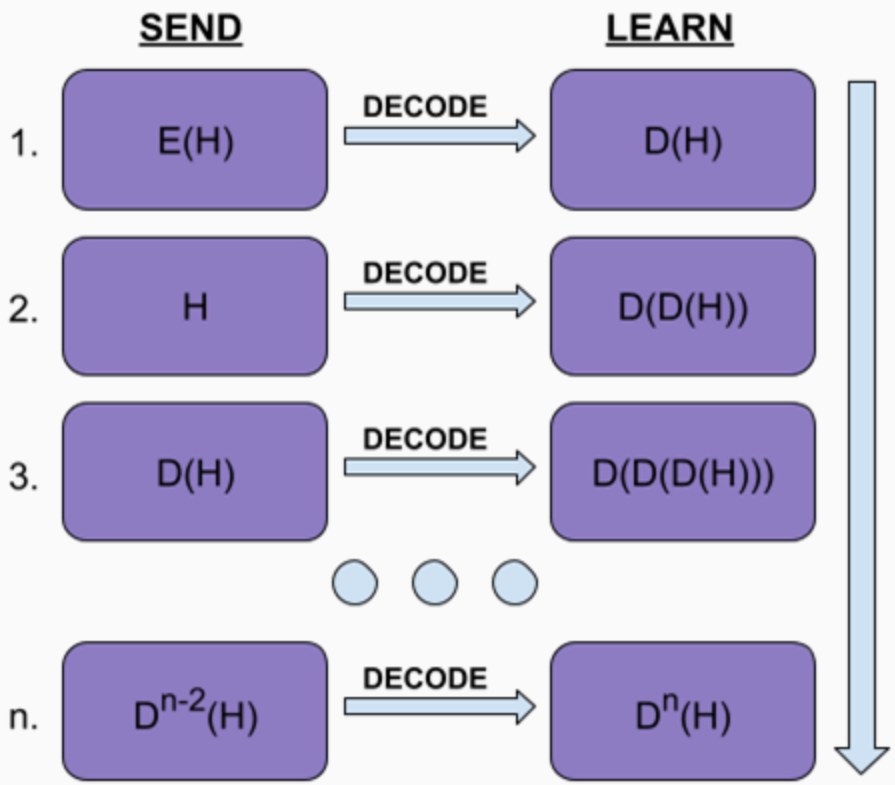
Design Highlight: Scoped Key Distribution

- Tree Based Key Derivation Scheme
  - Each channel has a root key
  - Keys are derived for each timestamp from root key using HKDF
  - ***No device contains more information than it needs***
- Per device subscription keys
  - Subscription keys are derived from global secrets and Device ID
  - ***Master subscription key stored in encoder, inaccessible to attackers***
- If implementation is correct, system is secure ***even under full device compromise***



Attack Highlight: The Farm

- In some cases knowing ciphertext-plaintext pairs enables other attacks
- Set up an attack board with a raspberry pi. Its job: ***Farm ciphertext-plaintext pairs 24/7***
- In one design we found a double decryption oracle
- After ~5 days of collecting data, we found something that decrypted to the memory address that contained the ***MASTER KEY***
  - Leveraged a buffer overflow on the decryption buffer to overwrite pointers and leak master key
  - Also considered finding addresses that we controlled to leverage in control flow redirection



Mich

Attack Highlight: Timing Side Channel

- Used `memcmp`'s byte-by-byte comparison as a timing oracle to forge signatures (or frames)

  UART → UART: measured time between ack vs. error messages

  UART → LED: measured time between ack messages vs. red LED (error)

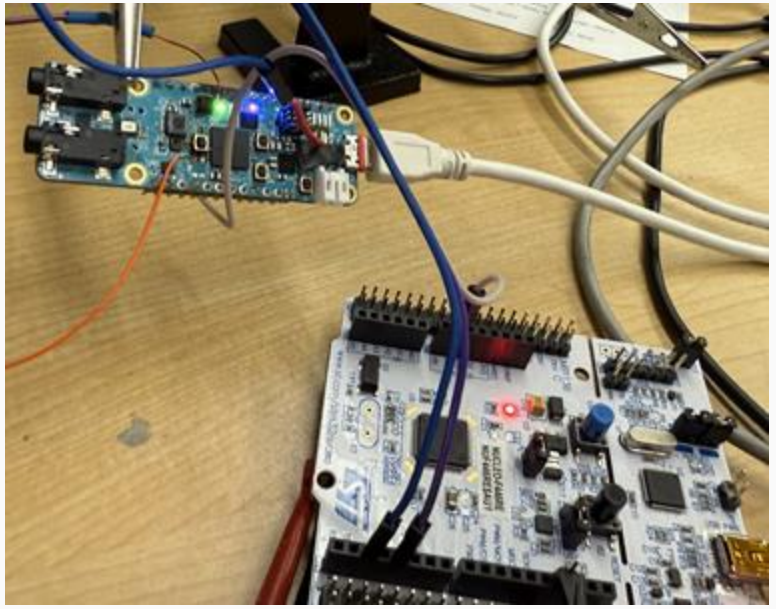
  LED → LED: measured time between yellow LED (ack) vs. red LED (error)

- Attack Implementation

  Wires soldered to resistor connected to the red LED and to UART on MAX78000 board.

  STM32 hooked up to the wires with a timer, sending pulse time durations over serial.

  Python script on host computer received durations from STM32 and performed statistical analysis on recorded durations to determine the correct signature byte.



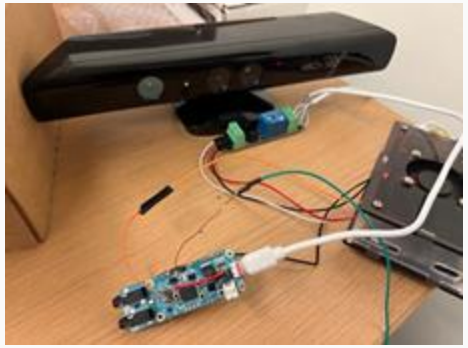
Timing Side Channel Impacts and Countermeasures

- What is the impact of this attack?
  - Leaks information based on how long operations take to execute
  - Attackers can infer sensitive data (e.g., passwords, keys) from small timing differences
  - Error handling that varies in timing can unintentionally reveal internal logic

- Suggested Countermeasure
  - Avoid functions like memcmp as it can leak information by returning early
  - Use constant time comparison functions to check

Other Attacks / Misc Infrastructure

- QEMU/AFL Integration
  - Ended up not being very successful due to buffer overflow in `read_packet()` crashing the decoder
- Remote Setup with Raspberry Pi
  - For easy access to hardware and brute forcing needs



- Automated Pesky Neighbor Script
  - Automated the workflow of attempting trivial attacks on pesky neighbor on a single generalized python script

Final Comments

- With more time and resources, what other things would you have done?
  - **Design Phase:** BLAKE3 instead of SHA1, Randomize Binary Layout, Random Delays to defend against fault injection, Double-If compiler pass if performance allowed it. Better structure for less experienced members.
  - **Attack Phase:** Voltage Glitching, more stable automated Pesky Script, better remote setup. More infrastructure and automation.
- What was the most valuable thing you learned during the competition?
  - Invest in documentation to break down knowledge silos
  - Don’t underestimate potential attack vectors

Thank you!
Questions?

Mich

eCTF10
10 YEARS OF THE EMBEDDED CAPTURE THE FLAG

Mark Peters, Ph.D.

President & CEO

MITRE

Refreshment Break

Please return to your seats by 17:15

eCTF10

10 YEARS OF THE EMBEDDED CAPTURE THE FLAG

Purdue University

Currently in 2nd place with 16,827 points

Not final score

Time (EST)	Title
15:30	Competition Overview
15:50	NEMC Guest Speaker – John Petrozzelli
16:00	City College of San Francisco
16:15	Mountain View High School
16:30	University of Michigan
16:45	Keynote – Mark Peters, Ph.D.
17:00	Refreshment Break
17:15	Purdue University
17:30	University of California, Los Angeles
17:45	University of Illinois Urbana-Champaign
18:00	Carnegie Mellon University
18:15	Award Presentations
18:25	Closing Remarks – Dan Walters
18:30	Networking Reception

MITRE

b01lers

Purdue University

Jack Roscoe
Gabriel Samide
Sebastian Toro
Kevin Yu

Straight b01ling it!

Nick Andry, William Boulton, Philip Frey, Neil Van Eikema Hommes, Sam Guber, Jihun Hwang, Jaxson Pahukula, Jack Reynolds, Jack Roscoe, Gabriel Samide, Lucas Tan, Vinh Pham Ngoc Thanh, Vivan Tiwari, Jacob White, Lawrence Xue, Bo-Shiun Yen, Kevin Yu

Advised By: Santiago Torres-Arias

Design

Design Philosophy

- Assume secrets on flash can be leaked by attacker
  - Helps defend hardware attacks
  - Also mitigates effectiveness of software attacks which gain code execution

Design Process

- Utilise symmetric encryption
  - Prevents attacker from decoding channel without secret key
- Improvement: Key Per Channel, send keys in encrypted subscription
  - No Subscription scenario is now impossible
- Improvement: Unique subscription key per decoder
  - Pirated Subscription scenario is now impossible

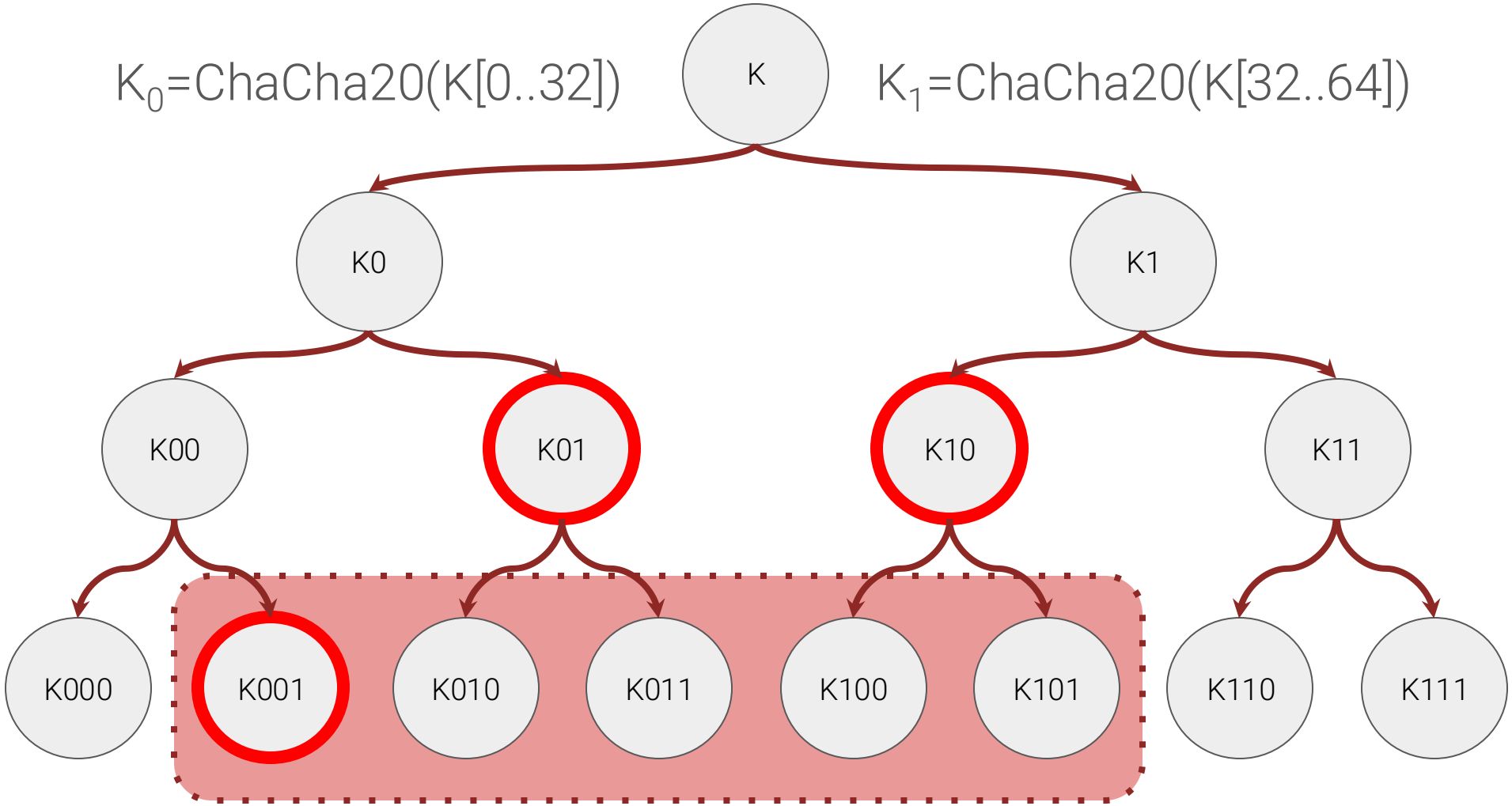
Key Per Timestamp

- In order to make Expired Subscription and Recording Playback impossible, we want to follow same trick as before, with key per timestamp
  - Unfortunately, too many timestamps to send every key in subscription

Hash Tree

- Utilize tree structure with hashes
  - Based on bit of timestamp, hash secret with different salt depending on left or right node
  - Intermediate nodes can be sent in subscription, using O(log n) space

PRNG Tree



b0llers

Digital Signatures

- Pesky Neighbor still vulnerable since attacker can recover key encrypting frames if the neighbor has a subscription to the same time region (or the emergency channel)
- Solution: sign subscriptions and frames with EdDSA to prevent forgeries

Additional Security Mitigations

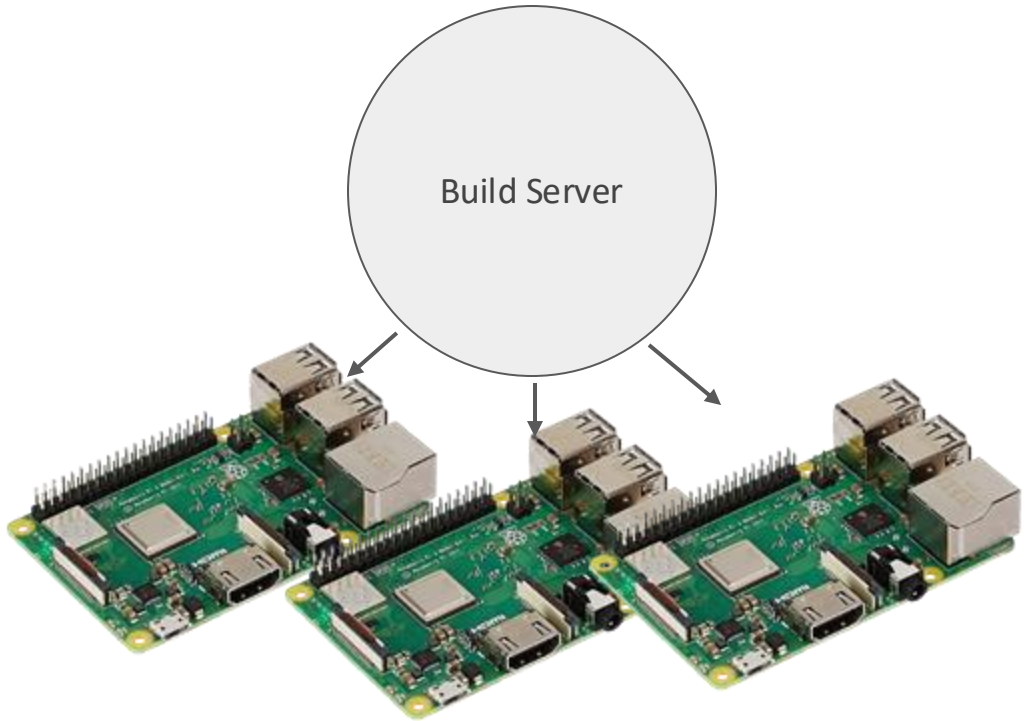
- Utilise Rust with custom HAL implementation
  - Borrow checker helps prevent memory corruption
  - Same board from last year -> saved time making a HAL
- MPU used to forbid RWX memory
  - Makes memory corruption bugs harder to exploit
- Compile time randomization of all sections
  - Unfortunately had to disable this due to issues with flash controller code

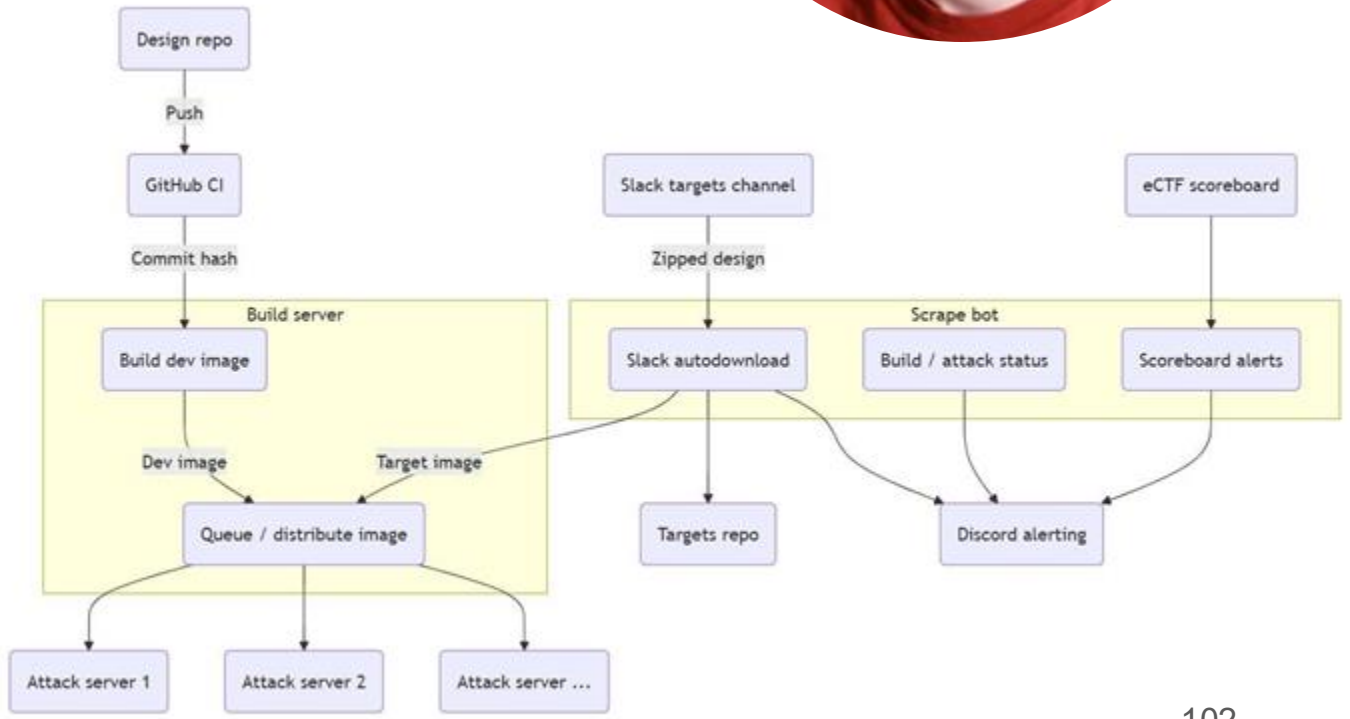
Infra

High level overview

Infra split into 2 main parts:

- scrape-bot (aka Tom Scott) — Discord/Slack/etc. bot
- build-server — build/distribution system

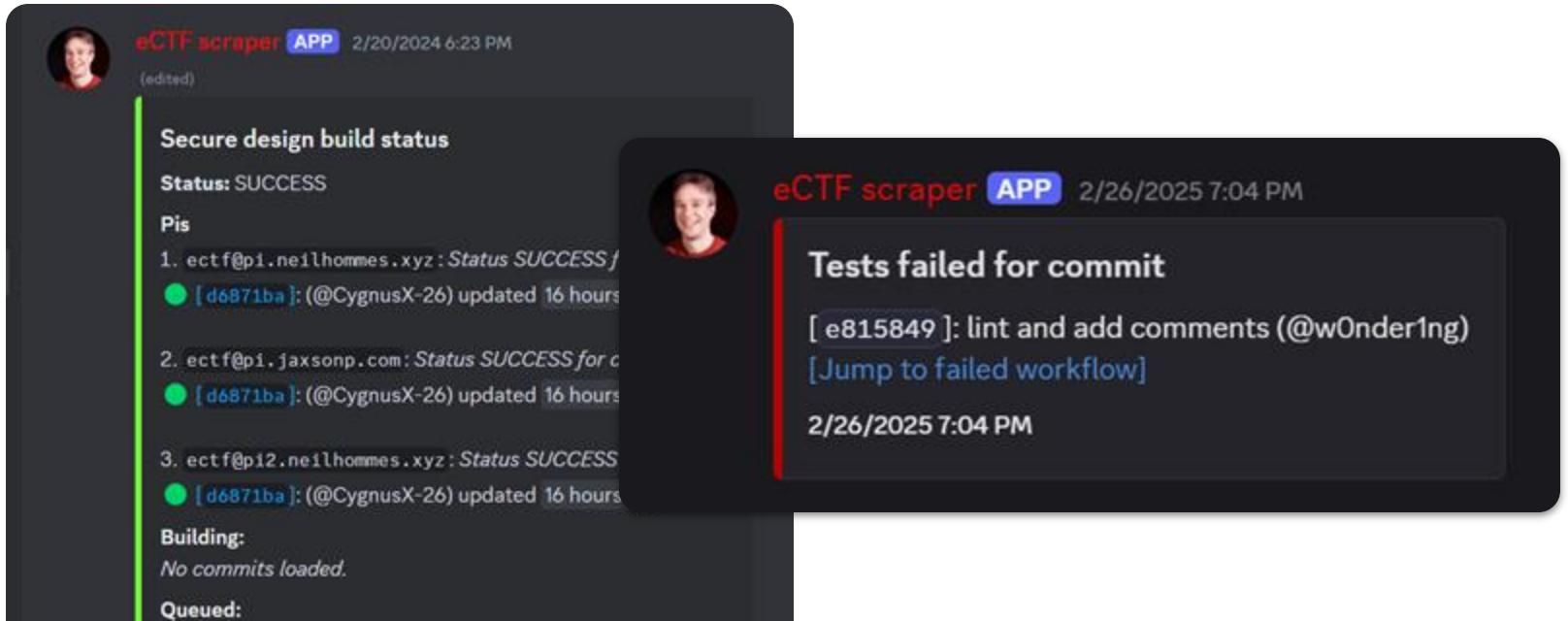


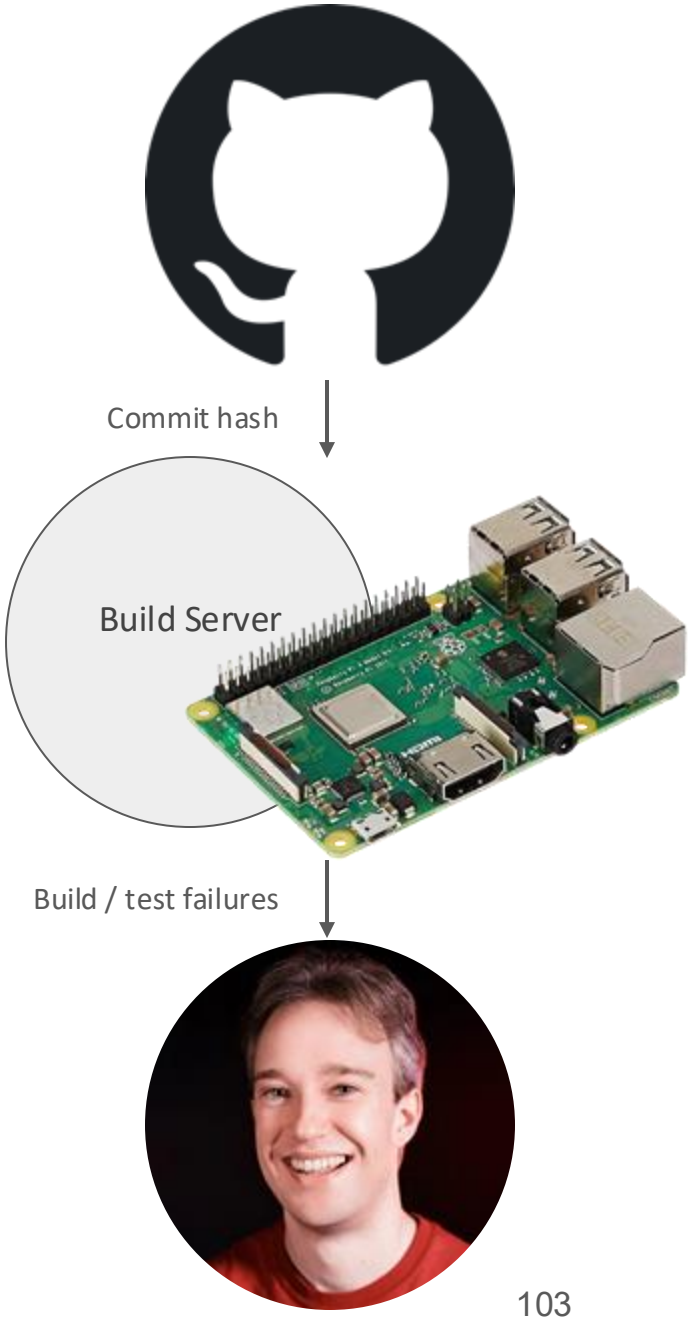


Dev phase

GitHub CI:

- Build new design
- Distribute image to attack pis
- Test attacks / functional requirements
- Report build / test failures in Discord

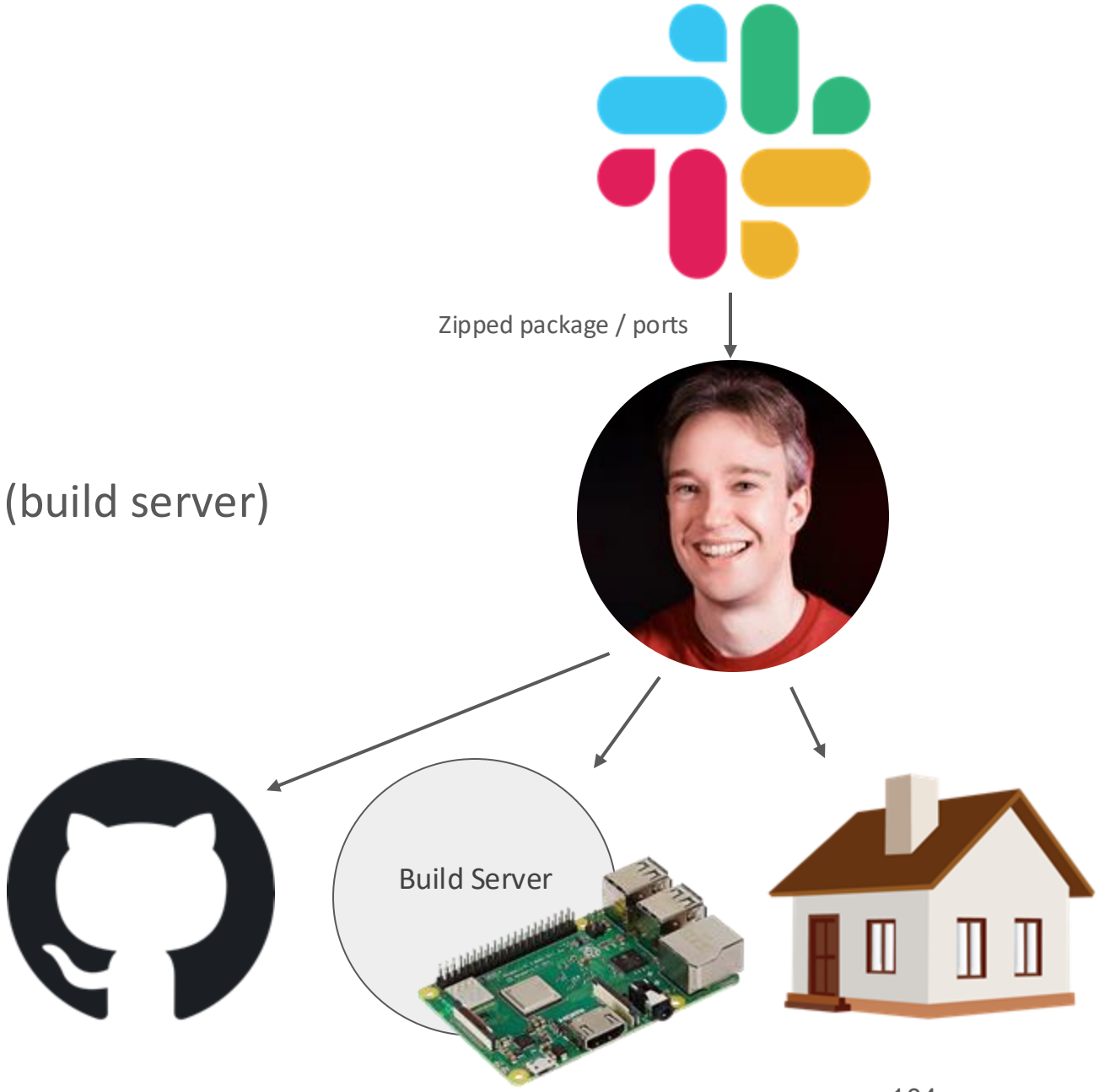




Attack phase

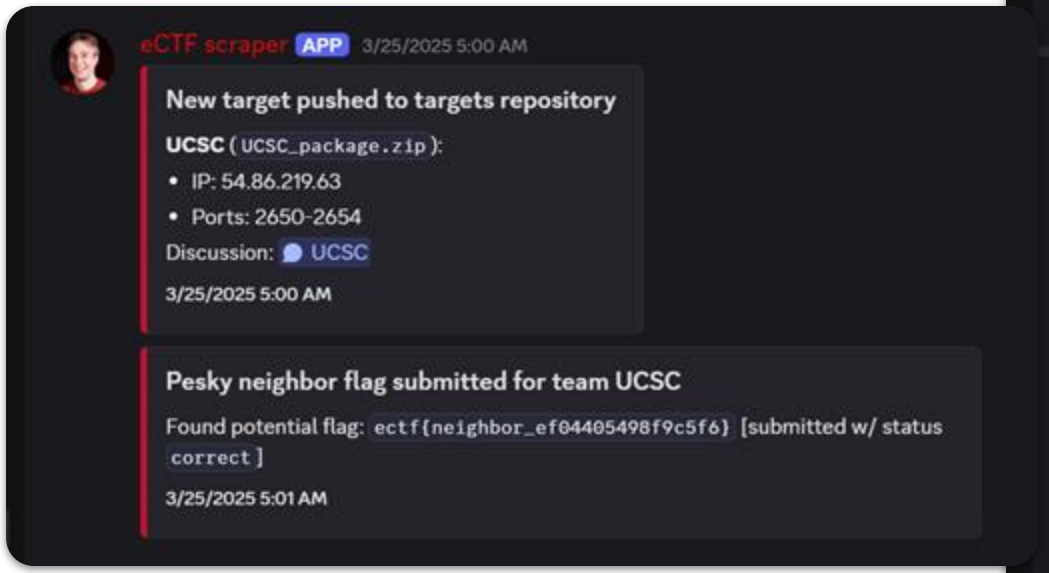
On new push to the #attack-targets channel,

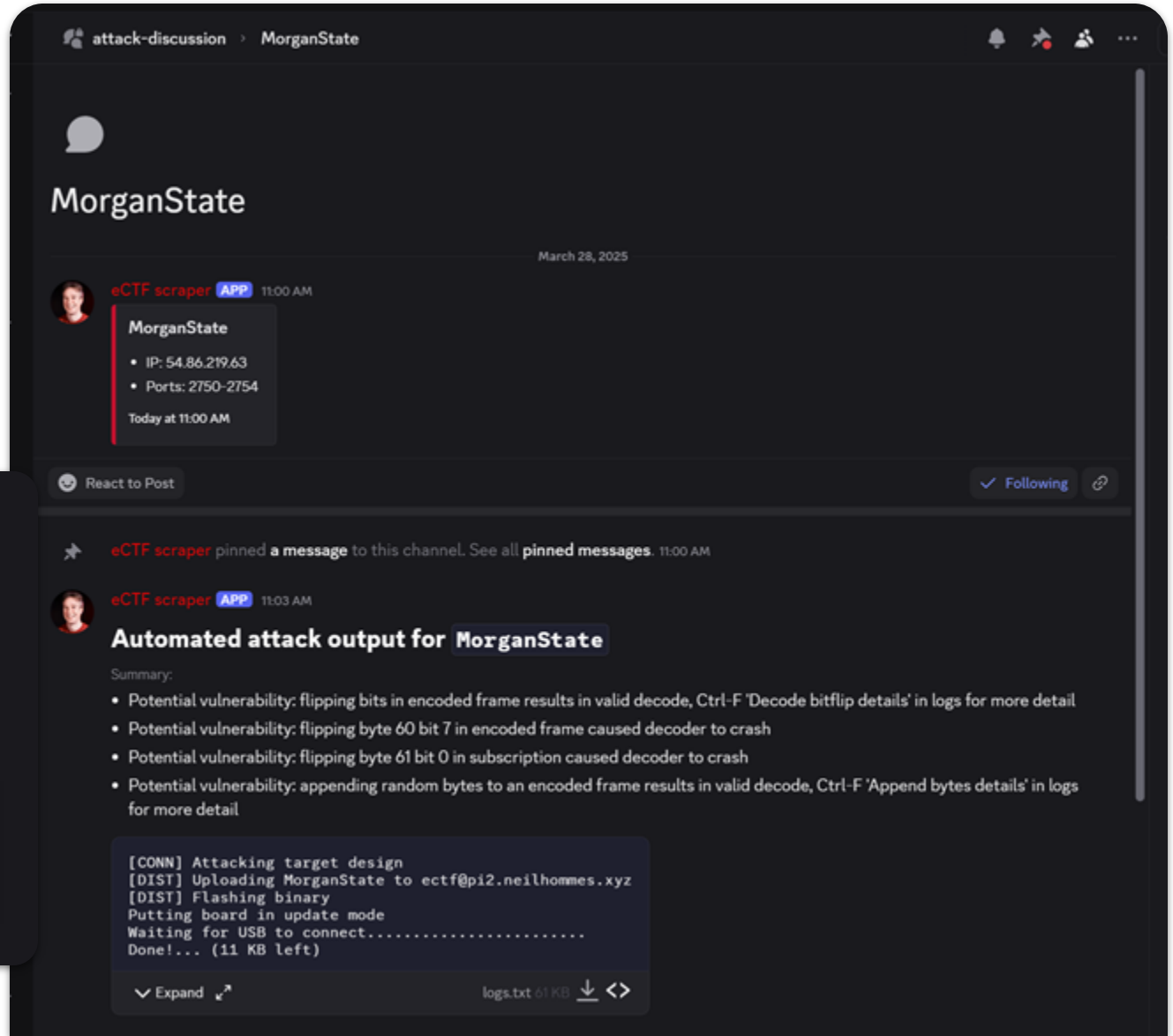
- Create new attack channel in discord
- Push new design to targets repo
- Run automated attacks on new package (build server)
- Queue default pesky neighbor attack



Attack phase (cont.)

- ~18 automated flag submissions
- ~5 automated first bloods

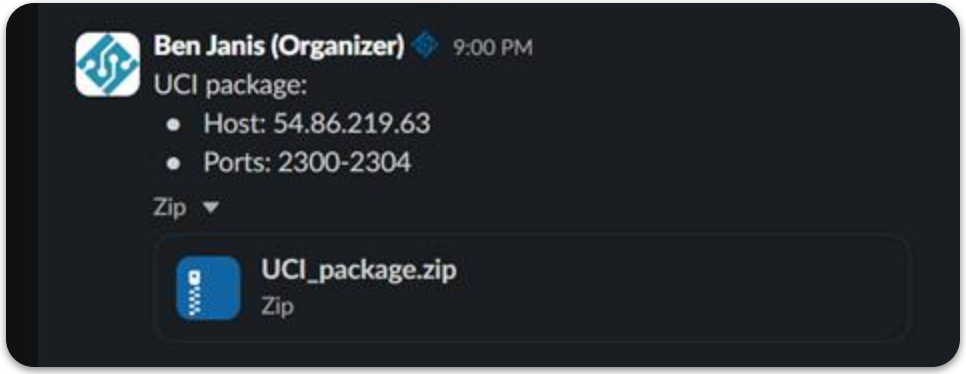


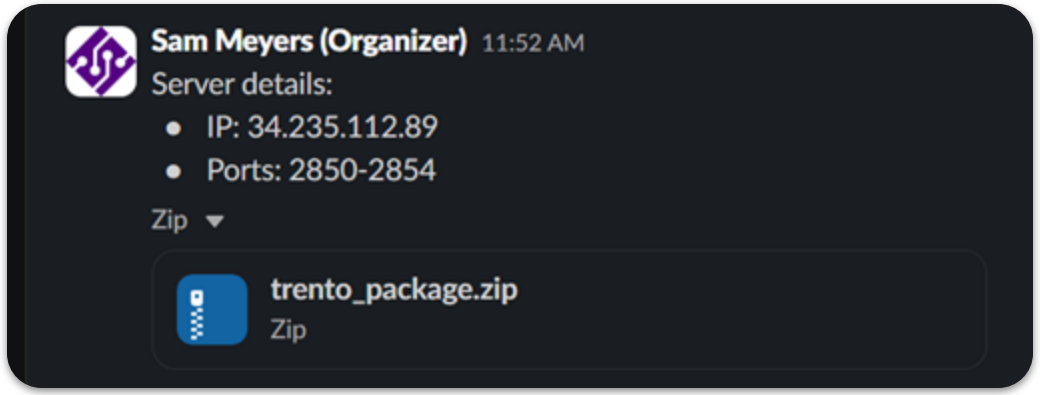


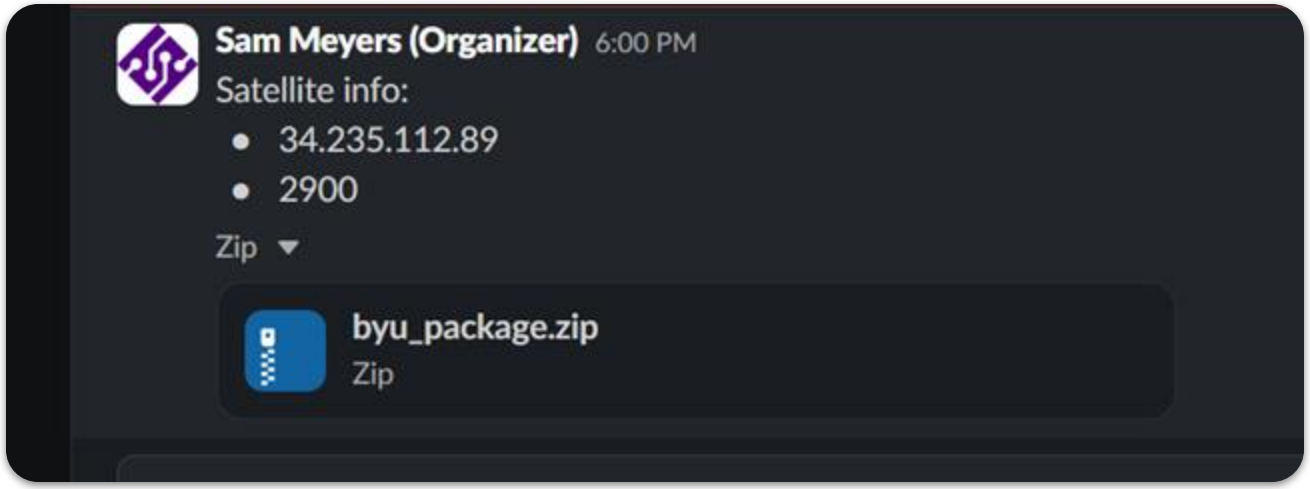
Some problems

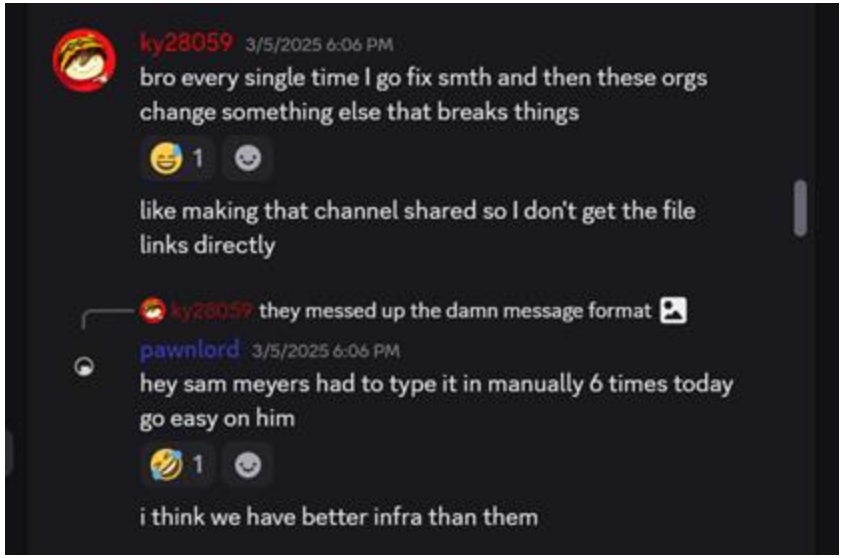
Format of attack target messages kept changing

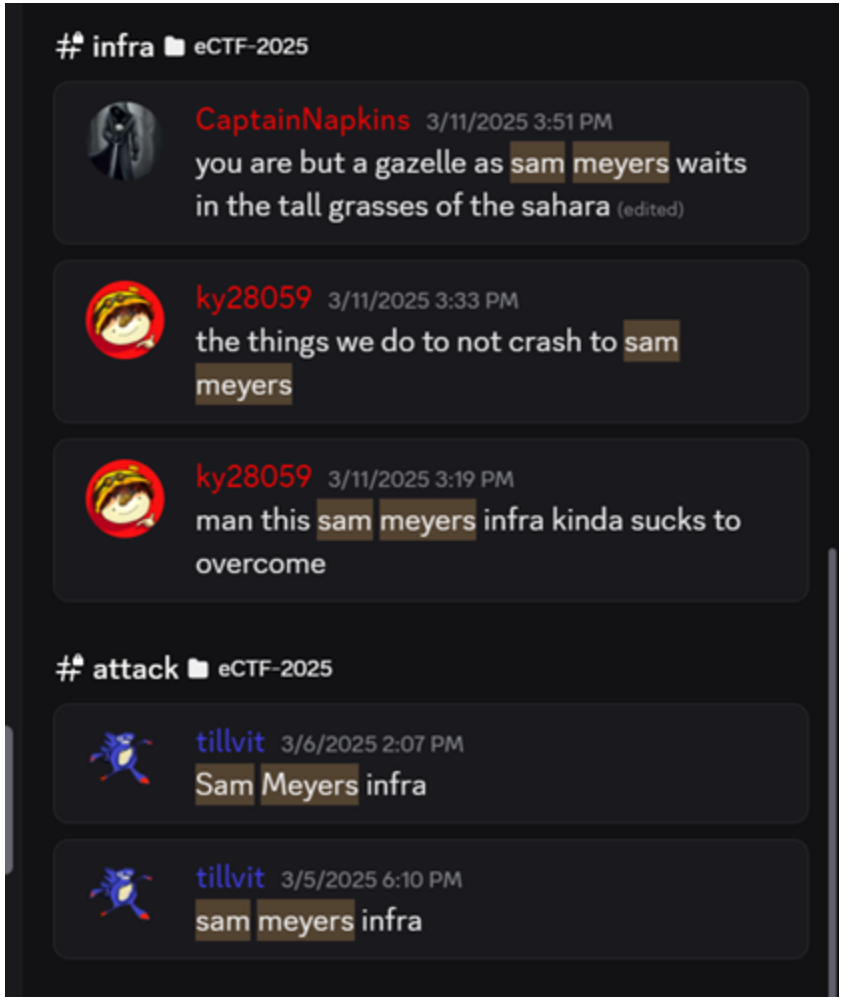
- Missing ports
- Different labels
- The bot would crash each time…







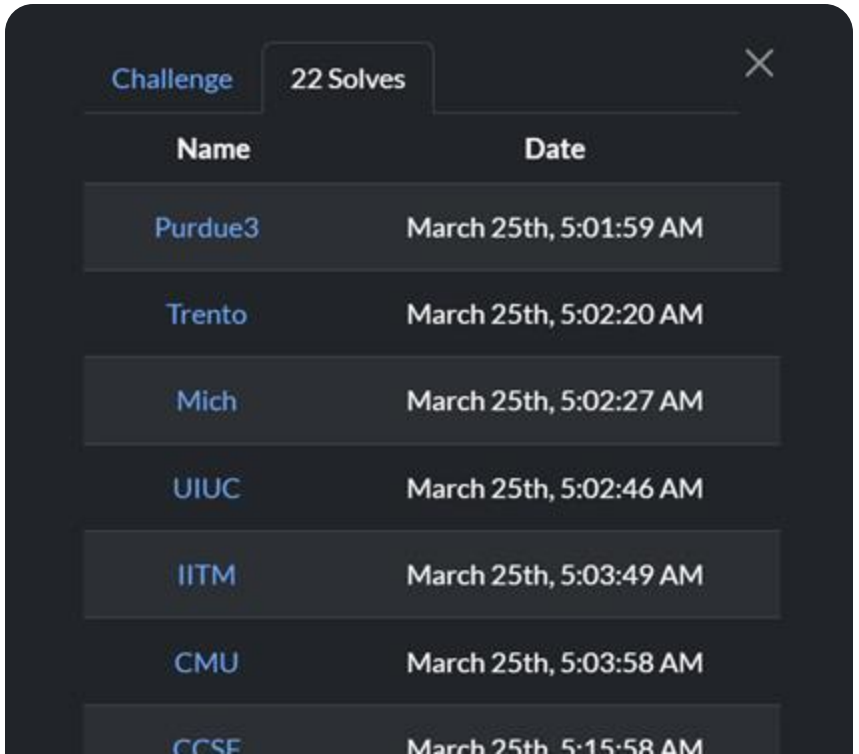


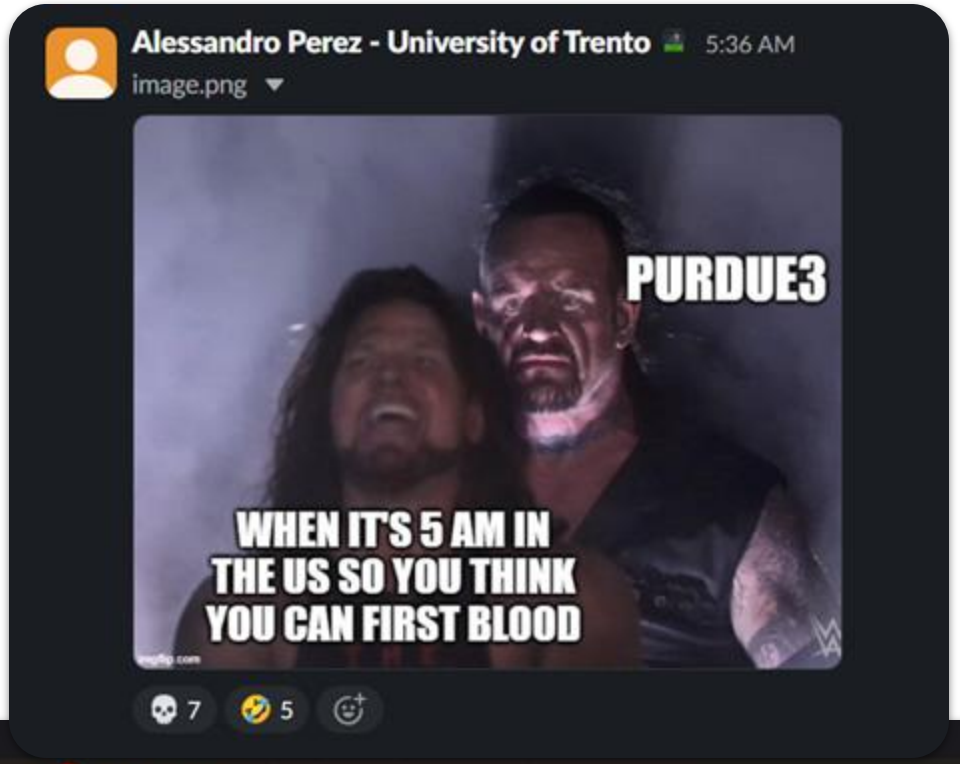


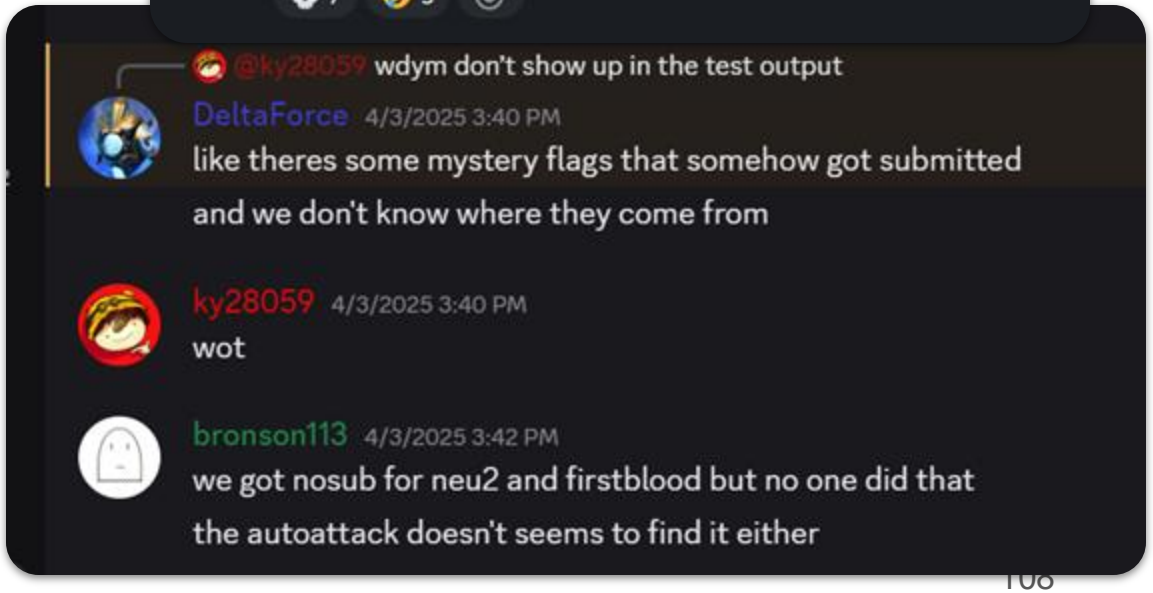
(jk, we love you Sam Meyers <3)

Some highlights

- Beating Trento to UCSC pesky neighbor while entire team was asleep
- Mysterious neu2 first blood…







Seems Familiar…

Source: Purdue eCTF 2024 slides

In other words, it pays to have infra :)

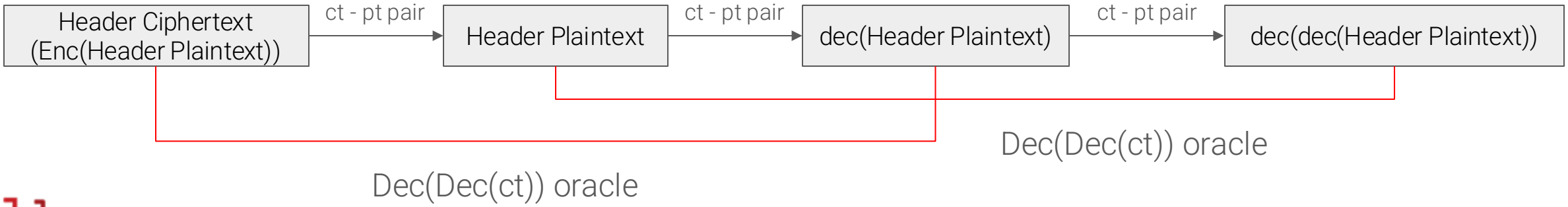
Attack Phase

Common Attacks

- Buffer Overflow to overwrite return address and get code execution
- Frame decode length not checked enabling leaking secrets off of stack
- Hardcoded Secrets
- Leaked Secrets in package
- Default pesky neighbor
  - Many teams misread rules and check monotonically increasing timestamp per channel
  - Some teams also check if next timestamp >= last timestamp instead of >
- No or incorrect MAC / signature allowing packets or subscriptions and frames to be manipulated and forged
- Cryptographic issue like reused nonce

ECB Decrypted Overflow

- On one team, we had ability to overwrite return address, but overwrite is first decrypted with AES ECB
- Outer layer decrypted with master key, inner layer with channel key
  - However, on channel 0, master key is used as inner key
  - We can use decode functionality as Dec(Dec(ciphertext)) oracle
- First block is header with known plaintext, encrypted with outer key
  - Repeatedly use oracle to generate chain of known plaintext ciphertext pairs



ECB Decrypted Overflow

- With ability to generate many plaintext ciphertext pairs, we eventually hope one of them decrypts to useful return address
- After ~5 hours and a few million pairs, one pair decrypts to address inside of `MXC_GPIO_Init`, which doesn't crash and pops return address off of stack
  - Return address is inside of UART buffer, so we can return to shellcode and leak keys

Double CBC Decryption Oracle

- One team utilized AES-CBC mode with fixed IVs, but did not use a MAC
- Subscription data has 3 blocks:

$(Channel_{ID} \| 0) \oplus IV_{master}$	$T_{end} \| T_{start}$	$Enc((Device_{ID} \| 0) \oplus IV_{Channel}), key = Key_{Channel})$

- These 3 blocks are then encrypted in CBC with the Master Key
- To get expired subscription, we must manipulate subscription as follows:
  - First block must be the same, there is only 1 possible value that matches the $Channel_{ID}$
  - Second block can be changed to whatever we want, since any random 2nd block has about a 1 in 1000 chance to give timestamps which allow expired decode
  - 3rd block must be exactly the same for $Device_{ID}$ check to pass

Double CBC Decryption Oracle

$(Channel_{ID} \| 0) \oplus IV_{master}$	$T_{end} \| T_{start}$	$Enc((Device_{ID} \| 0) \oplus IV_{Channel}), key = Key_{Channel})$

- Problem: changing second block changes 3rd block plaintext due to CBC decrypt
- If we can find an oracle to decrypt a block with the master key, we can learn `Enc((DeviceID || 0) ⊕ IVChannel), key = KeyChannel)`
  - Then with any known plaintext - ciphertext pair for the 3rd block, we can construct a second block which causes it to xor to correct value, and have valid timestamps with 1 / 1000 chance

Double CBC Decryption Oracle

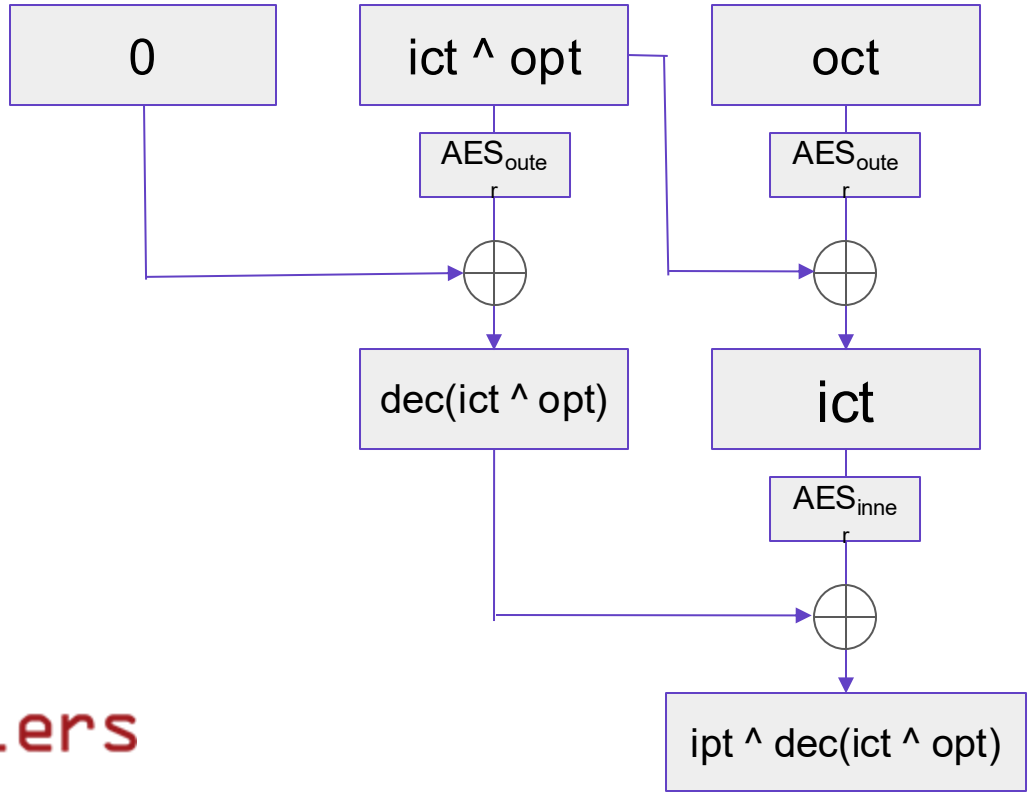
- We can use the decode functionality to try to decrypt the correct block
- Decode functionality decrypts payload in 2 layers
  - First decrypt outer layer in CBC mode with master key
  - Then decrypt inner layer in CBC mode with channel specific key
- This double layer decrypt complicates things, as the inner layer obscures decryption results from outer layer

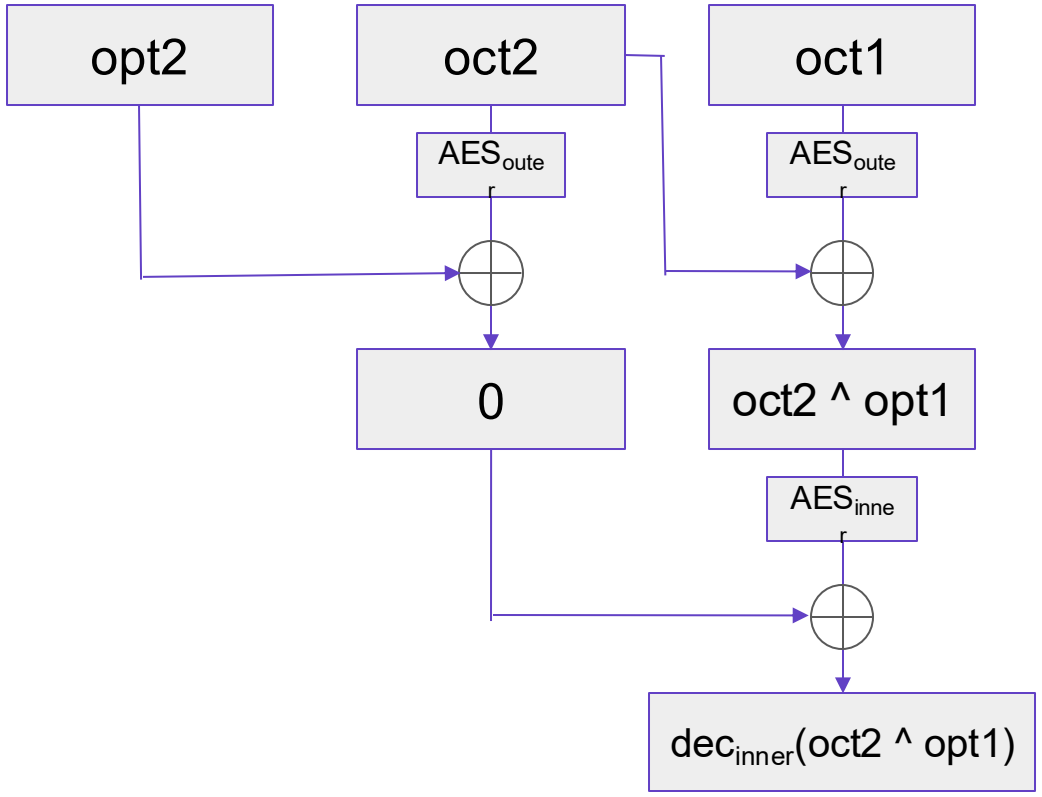
Double CBC Decryption Oracle

- Our first observation is we have 2 known plaintext - ciphertext (pt - ct) pairs for the master key
  - These correspond to the blocks containing timestamp in subscription data
  - Subscription can be uploaded to device and list can be used to get timestamps

Double CBC Decryption Oracle

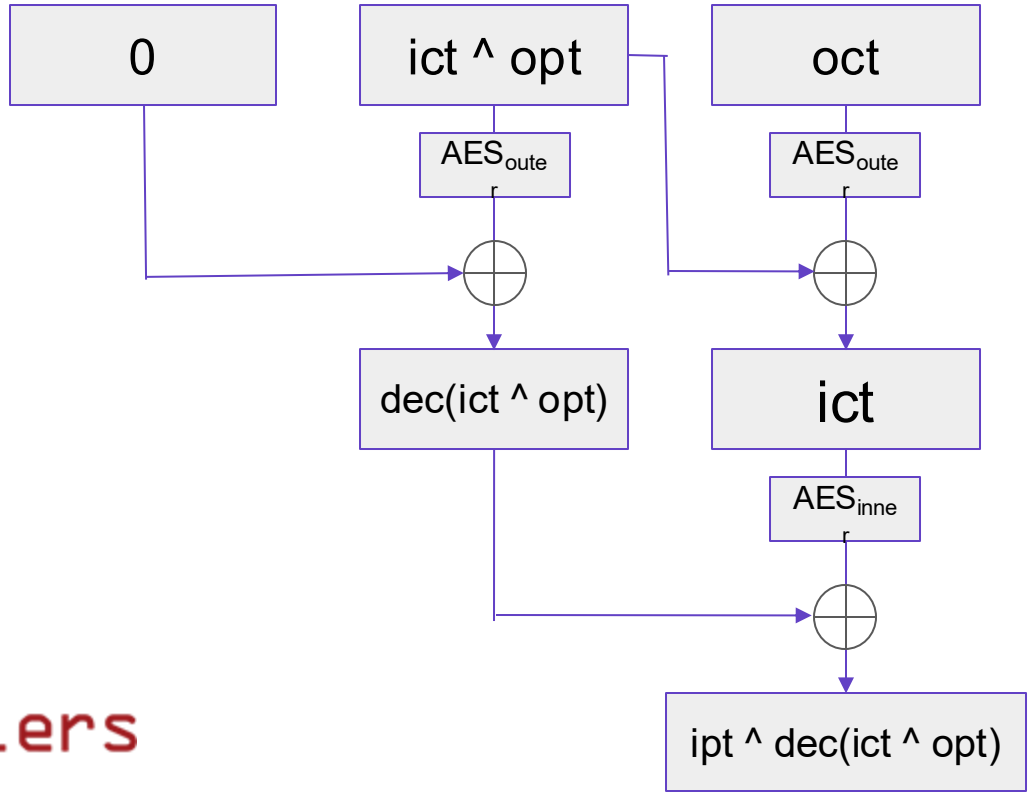
- We can construct the following 2 oracles:
  - let oct, opt denote an outer ciphertext and plaintext pair (with master key)
  - let ict, ipt denote an inner ciphertext and plaintext pair (with channel 0 key)

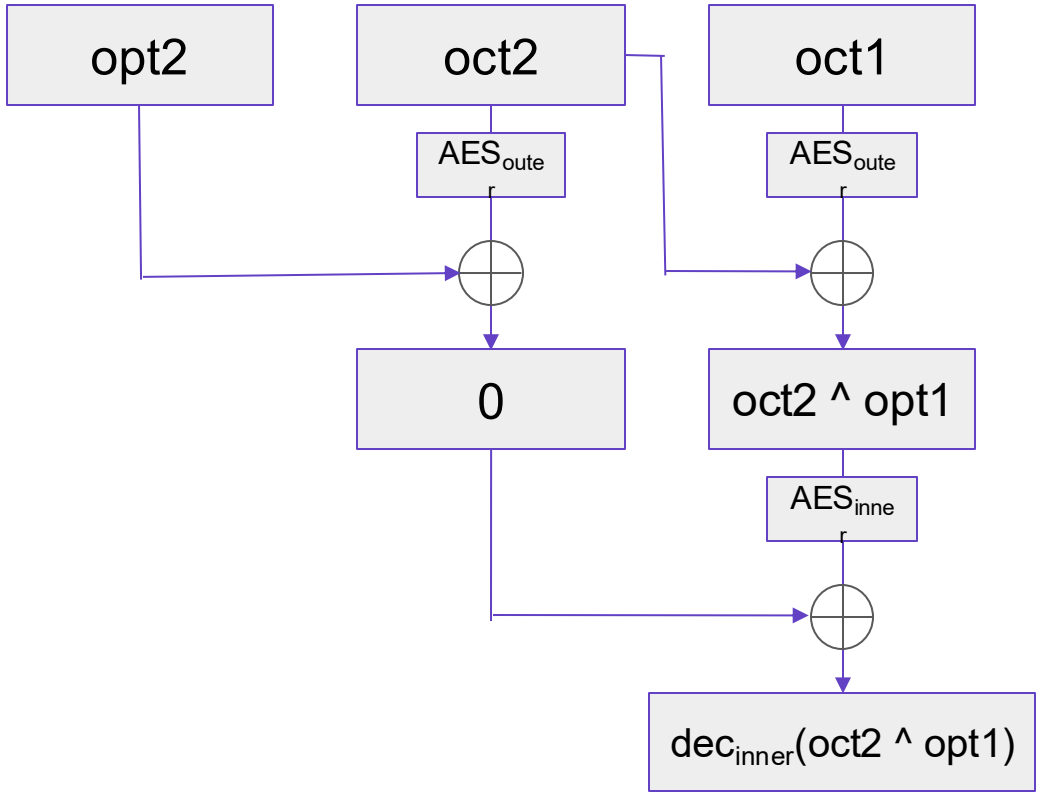




Double CBC Decryption Oracle

- With these oracles, we can learn the following:
  - $dec_{master}(ict \oplus opt)$
  - $dec_{channel}(oct_2 \oplus opt_1)$





Double CBC Decryption Oracle

- With these oracles, we can learn the following:
  - Outer oracle: $dec_{master}(ict \oplus opt)$
  - Inner oracle: $dec_{channel}(oct_2 \oplus opt_1)$
- First we can generate a known ict - ipt pair, using inner oracle
  - Send same outer pair for both pair1 and pair2
    - $ict = oct \oplus opt$, $ipt = dec_{channel}(oct \oplus opt)$
- Using this generated inner pair and the other outer pair, we can generate an arbitrary number of outer pairs using outer oracle
  - $oct_{new} = ict \oplus opt_{old}$, $ipt_{new} = dec_{master}(ict \oplus opt_{old})$
  - This can be repeated many times

Double CBC Decryption Oracle

- With these oracles, we can learn the following:
  - Outer oracle: $dec_{master}(ict \oplus opt)$
  - Inner oracle: $dec_{channel}(oct_2 \oplus opt_1)$
- Observe that using both oracles, we can compute $dec_{master}(ict \oplus opt_1 \oplus \ldots opt_n)$
  - In outer oracle, we can substitute ict with in invocation of inner oracle
  - Since inner oracle ciphertext = $oct_2 \oplus opt_1$, we can then substitute oct_2 with outer oracle
  - We can use this mutual recursion to choose many opt to xor together
    - Then we pick one ict as base case

Double CBC Decryption Oracle

- If we generate 128 opt, we can represent them as 128 dimensional vectors, which span the entire vector space
  - we can use RREF to pick which opt to xor together to get the term

    $Enc_{master}(Enc_{channel}((Device_{ID} \| 0) \oplus IV_{Channel})) \oplus oct_{base}$

    $Dec_{master}(opt_1 \oplus opt_2 \oplus \dots \oplus opt_n \oplus oct_{base})$

    $= Dec_{master}(Enc_{master}(Enc_{channel}((Device_{ID} \| 0) \oplus IV_{Channel})) \oplus oct_{base} \oplus oct_{base})$

    $= Dec_{master}(Enc_{master}(Enc_{channel}((Device_{ID} \| 0) \oplus IV_{Channel})))$

    $= Enc_{channel}((Device_{ID} \| 0) \oplus IV_{Channel})$
  - This is exactly value we wanted to decrypt earlier, which enables us to solve expired subscription, along with the method of generating oct - opt pairs

Encrypted Vtable Overwrite (Theoretical)

- One team had a buffer overflow, but it was first decrypted by AES-GCM
  - Unfortunately, they used mbed_tls which zeroes decryption buffer if GCM check fails (unlike wolfssl)
- However, frames are decrypted into a global buffer, which is right before the AES-GCM struct
  - AES-GCM has a pointer to cipher info as first element
    - This cipher info contains a vtable pointer

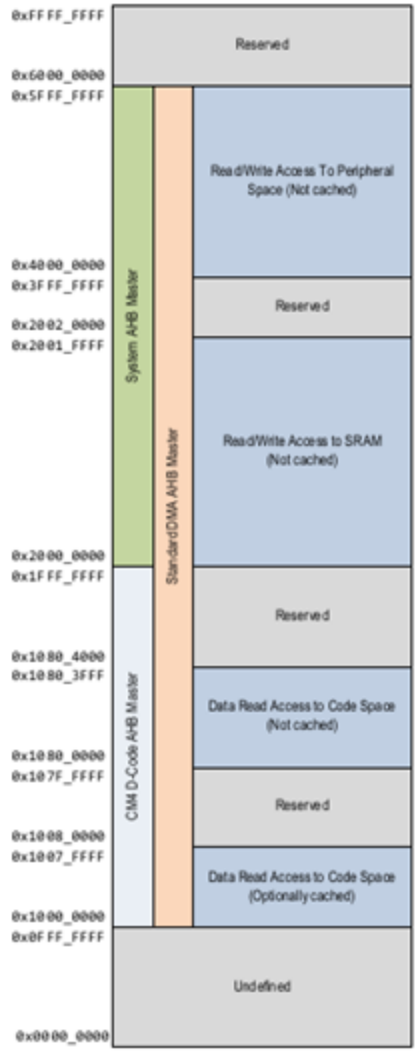
```
/**
 * Generic cipher context.
 */
typedef struct mbedtls_cipher_context_t
{
    /** Information about the associated cipher. */
    const mbedtls_cipher_info_t *cipher_info;

    /** Key length to use. */
    int key_bitlen;

    /** Operation that the key of the context has been
     * initialized for.
     */
    mbedtls_operation_t operation;
```

Encrypted Vtable Overwrite (Theoretical)

- As this pointer is used as decryption is occurring, before GCM is verified, we can overwrite the pointer without having it be zeroed
  - Unfortunately overwrite is encrypted (xored with unknown value)
- Based on the fields of the cipher_info, if it points to some random memory, it will likely return an error (specific values have to be set)
  - However, if it points to invalid memory, board will fault

Encrypted Vtable Overwrite (Theoretical)

- Using this crash or error info, we had the idea to flip high order bits, and narrow on power of 2 size aligned MAX78000 memory map region
  - Once we hit valid memory, it will return error instead of crashing, so we learn upper bits of keystream
  - Repeat process with progressively smaller section of memory map
  - We estimate it would take a few thousand reboots
    - Unfortunately we couldn't figure out how to automatically reboot board, so this was infeasible
    - Memory map also didn't match what documentation said



memcmp Timing Side Channel

- At first we thought memcmp checks 4 bytes at a time (similar to last year)
  - Any byte by byte brute force using timing side channel would require 2^{32} attempts per byte, which is too much to do over UART
- However we realised last day it actually falls back to byte by byte if 4 byte fails
- It is perhaps possible to measure this timing delay, but we didn't have time to get it working
  - This would enable attacking 4 or so additional teams

Final Comments

- Improvements to be made:
  - Improve attack infra automation–fuzzing and better loading
  - Hardware attacks–power analysis and glitching

We immensely enjoyed the competition; thank you to the MITRE organizers and eCTF sponsors for your hard work in making this event possible.

See You Next Year!

eCTF10

10 YEARS OF THE EMBEDDED CAPTURE THE FLAG

University of California, Los Angeles

Currently in 7th place with 14,272 points

Not final score

Time (EST)	Title
15:30	Competition Overview
15:50	NEMC Guest Speaker – John Petrozzelli
16:00	City College of San Francisco
16:15	Mountain View High School
16:30	University of Michigan
16:45	Keynote – Mark Peters, Ph.D.
17:00	Refreshment Break
17:15	Purdue University
17:30	University of California, Los Angeles
17:45	University of Illinois Urbana-Champaign
18:00	Carnegie Mellon University
18:15	Award Presentations
18:25	Closing Remarks – Dan Walters
18:30	Networking Reception

MITRE

acm.cyber + IEEE AT UCLA

IPEBERE

UCLA

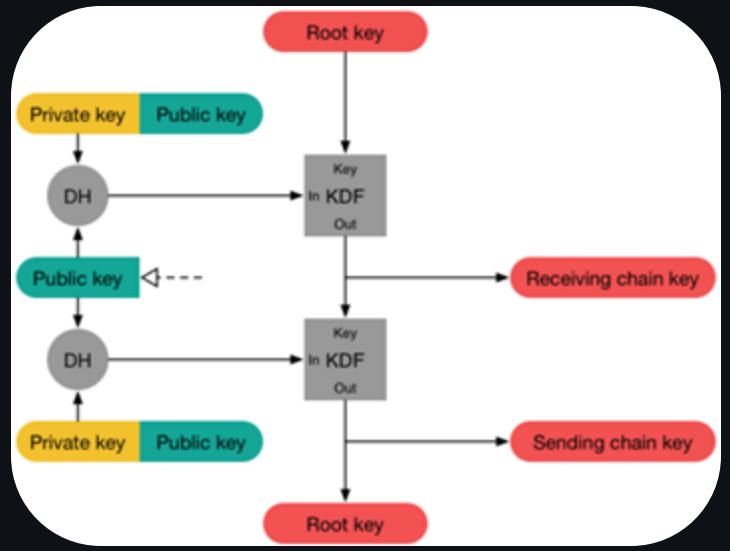
Ronak Badhe, Ryan Chang, Andrew Kuai, Parth Pandhare
Gary Song, Arnav Vora, Kai Wang, Joshua Zhu

Advised by Professor Nader Sehatbakhsh

Our Secure Design

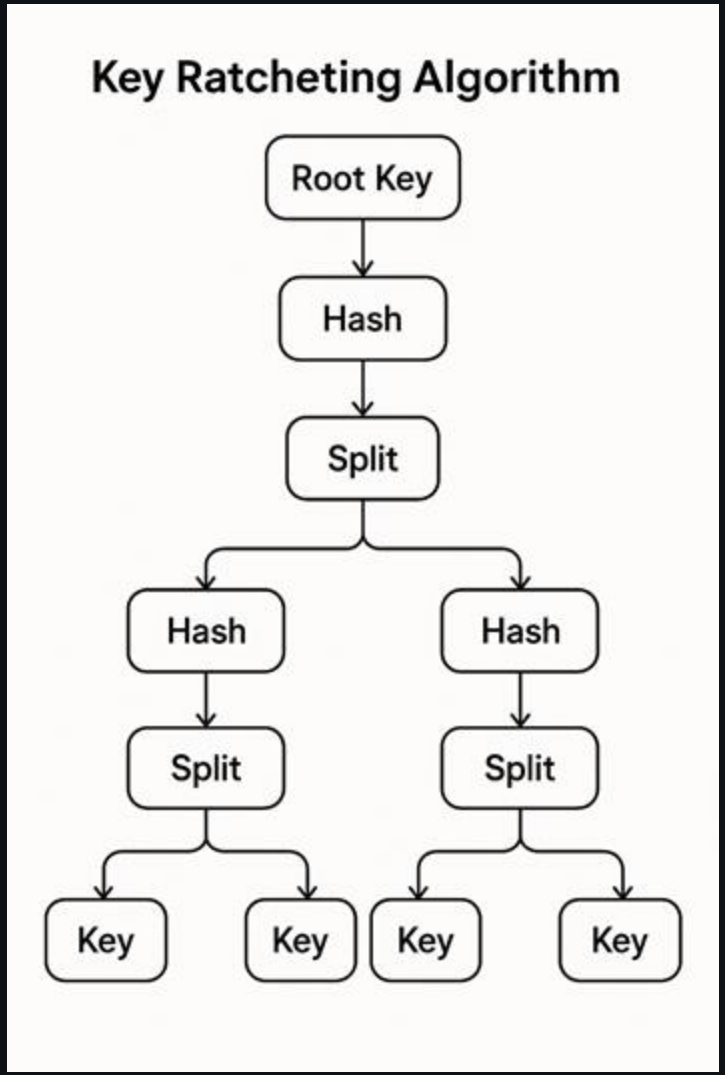
Cryptography

- **Signatures**: ED25519
- **Key derivation**: PBKDF2 using SHA256
- **Authenticated encryption with associated data**: ChaCha20+Poly1305
- **Key ratcheting**



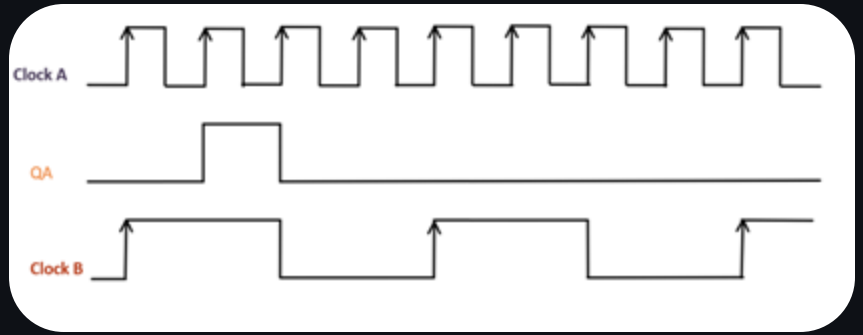
Key Ratcheting

- Channel-specific **root key** for size-2^{64} range
- SHA256 hash for smaller ranges
- Select half using timestamp bit



Optimizing Performance

- **Overclock** the board!
- Reduced prefix key size from 32 bytes to 16 bytes



Things We Would Have Done Differently

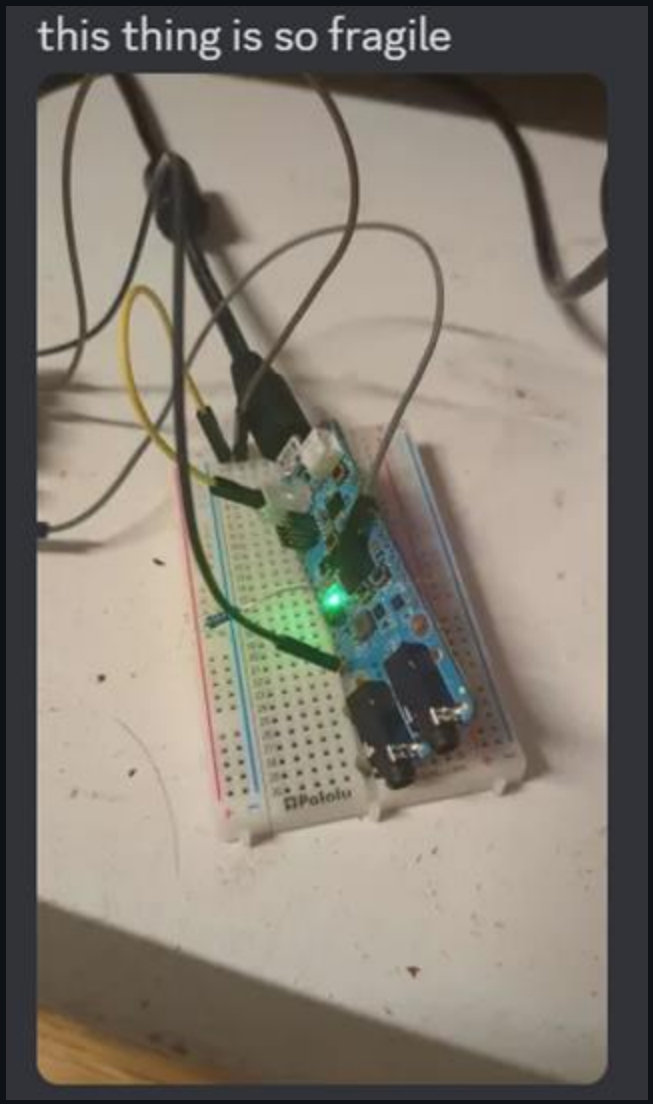
- Add random delays
- Disabling fault injection and unused peripherals
- Fuzz our own design

acm.cyber

Attack Phase Highlights

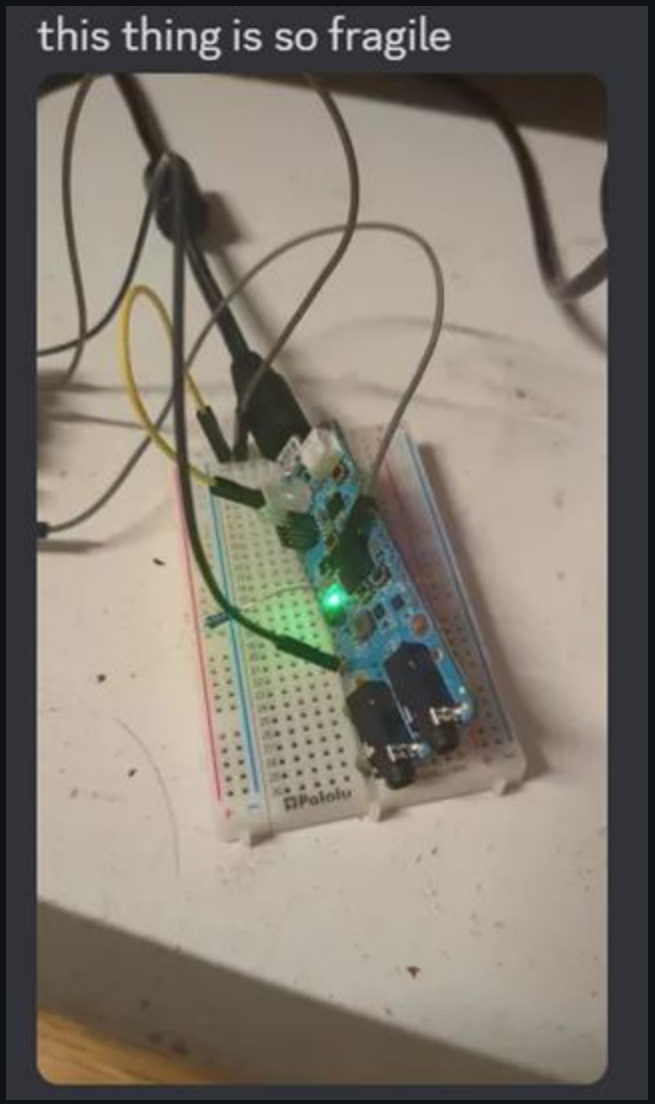
Attack Highlight: Side Channels

- LED blinks before and after memcmp HMAC check
- Side channel opportunity!
- Forge HMAC byte by byte



Suggested Fix

- Provide **AS LITTLE** information as possible
- Constant time operations
- Randomized delays



Other Attacks

- Exposed secrets
- ECB
- Pesky neighbor cheesing
- Padding oracle attack

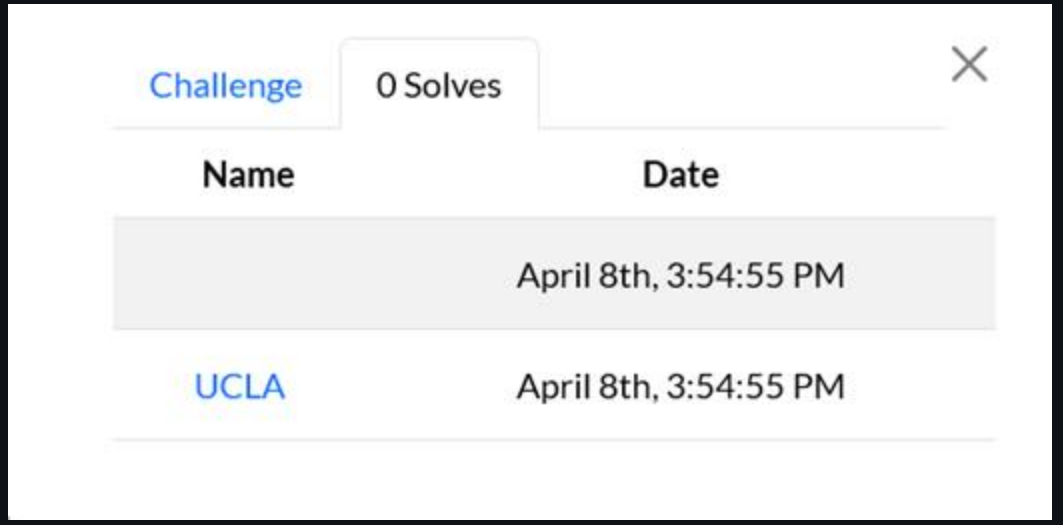


Exhibit A: Us getting ping-diffed

Final Thoughts

- The true pesky neighbor
- Lots of teams had the same secure cryptosystem
- Hardware attacks cool
- Learn Rust?

Questions?

eCTF10

10 YEARS OF THE EMBEDDED CAPTURE THE FLAG

University of Illinois Urbana-Champaign

Currently in 5th place with 15,276 points

Not final score

Time (EST)	Title
15:30	Competition Overview
15:50	NEMC Guest Speaker – John Petrozzelli
16:00	City College of San Francisco
16:15	Mountain View High School
16:30	University of Michigan
16:45	Keynote – Mark Peters, Ph.D.
17:00	Refreshment Break
17:15	Purdue University
17:30	University of California, Los Angeles
17:45	University of Illinois Urbana-Champaign
18:00	Carnegie Mellon University
18:15	Award Presentations
18:25	Closing Remarks – Dan Walters
18:30	Networking Reception

MITRE

University of Illinois Urbana-Champaign (UIUC)

Advisor

Professor Kirill Levchenko

Members (30)

Minh Duong, Jake Mayer, Nikhil Date, Krishnan Shankar, Suchit Bapatla, Safwan Morshed, Akhil Bharanidhar, Stephen Cao, Sagnik Chakraborty, Sanay Doshi, Yotam Dubiner, Timothy Fong, Serkan Gozel, Yash Gupta, Adish Jain, Peter Karlos, Swetha Karthikeyan, Adarsh Krishnan, Ananth Madan, Mason Miao, Julie Oh, Ben Pegg, Jupiter Peng, Phillip Raczka, Liam Ramsey, Neil Rayu, Rohan Seth, Preetesh Shah, Arpan Swaroop, Saipranav Venkatakrishnan

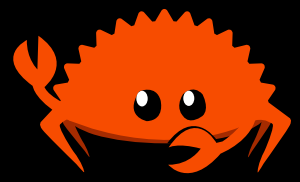
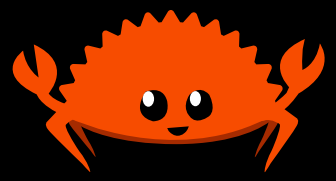
Our Design

- Rust (with our custom HAL) for memory safety
- Rust-based encoder using PyO3 (Python bindings for Rust)
- Masked Ascon-128 cipher implementation
- Decoder-specific and channel-specific keys for subscriptions
- No tree construction 😢

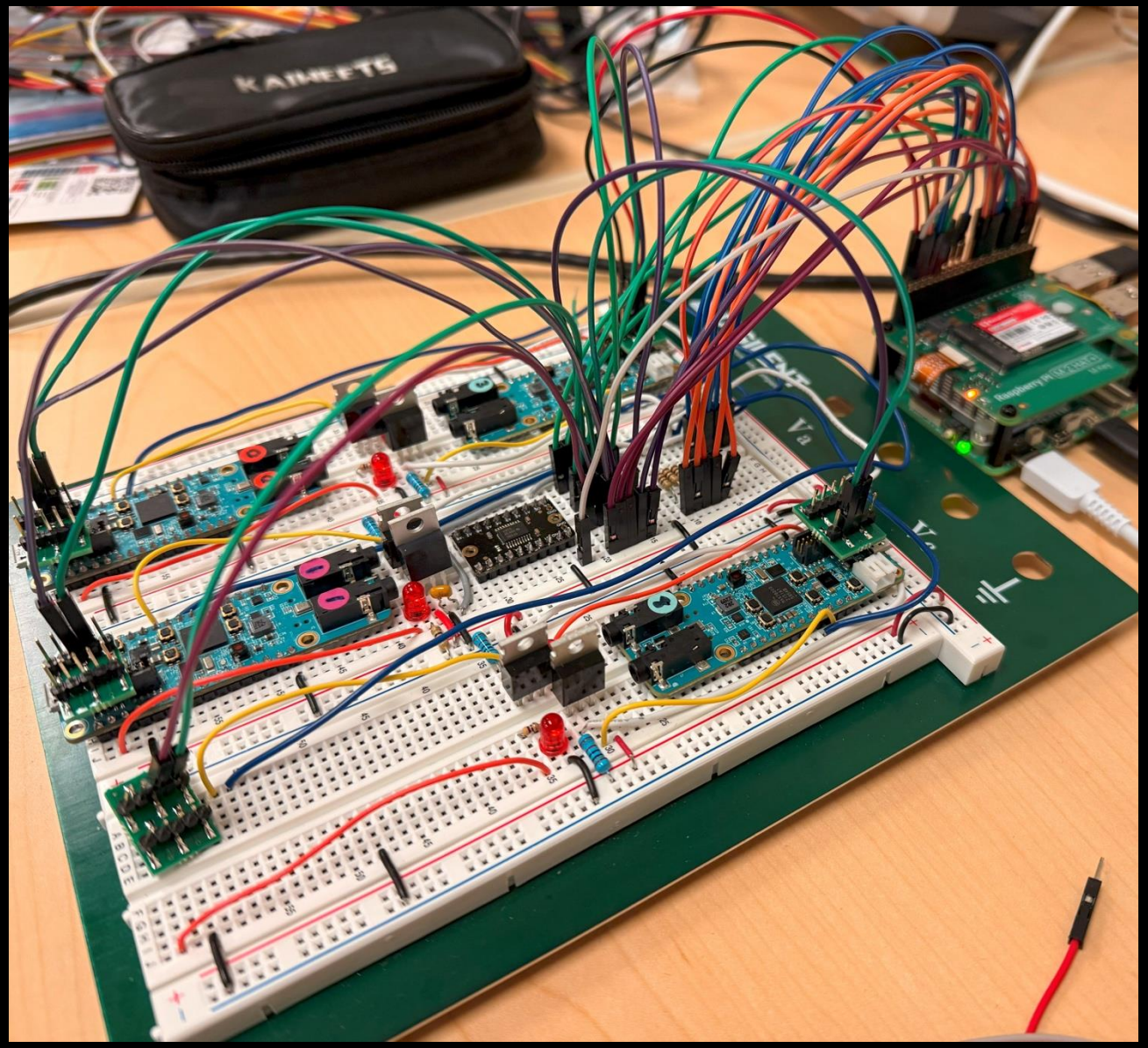
Rust HAL

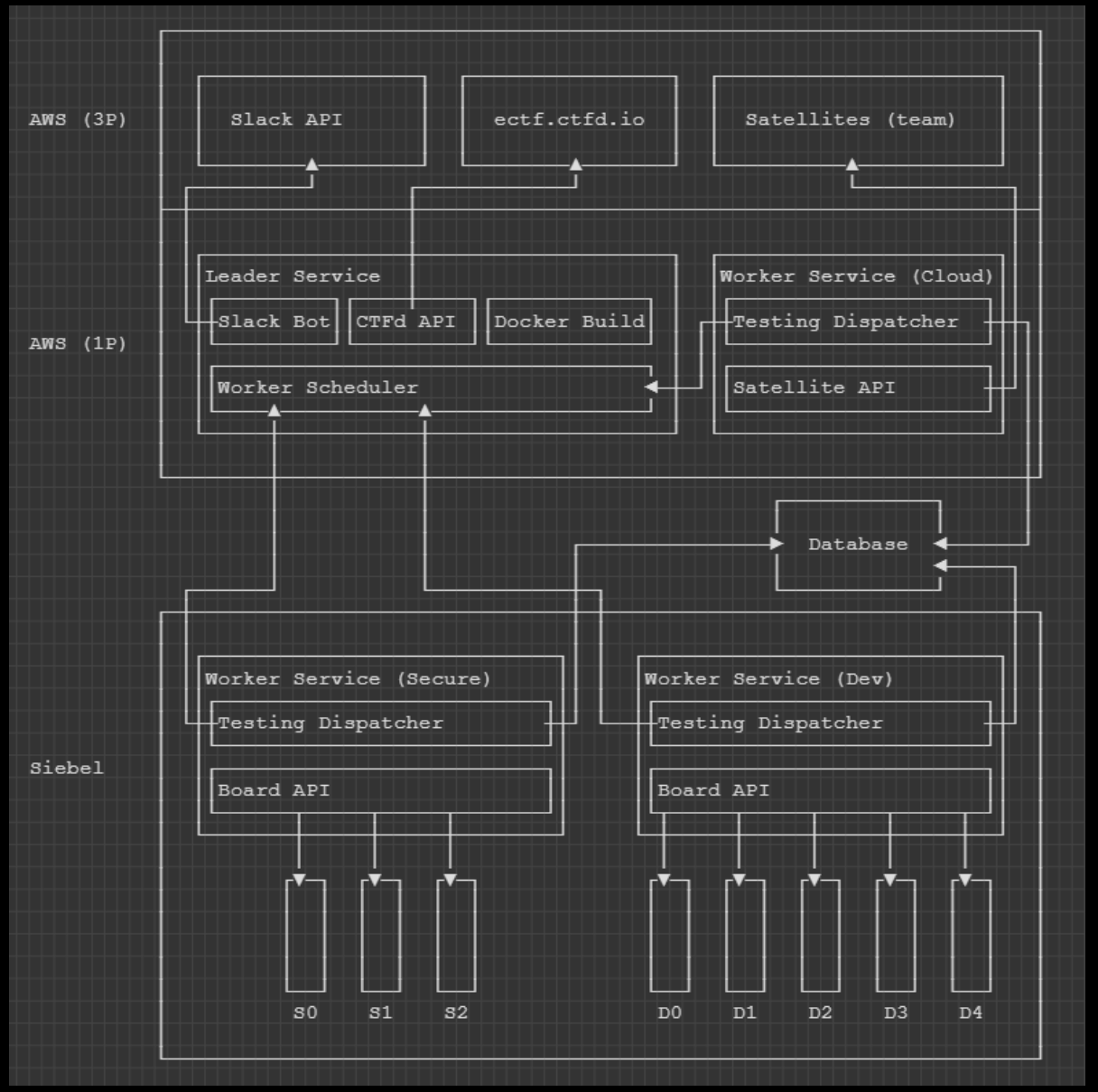
- In 2024, two teams wrote their design in Rust (us + Purdue)
  - 7% of overall teams
- This year, we created and open-sourced a generic Rust hardware abstraction layer for the MAX78000
  - Our goal was to have more teams writing Rust code!

- Result: 11 teams used Rust, **9** teams used our HAL or a fork
  - **22%** of overall teams (**82%** of Rust teams used our HAL)


Automated Attacks


KAIWEETS

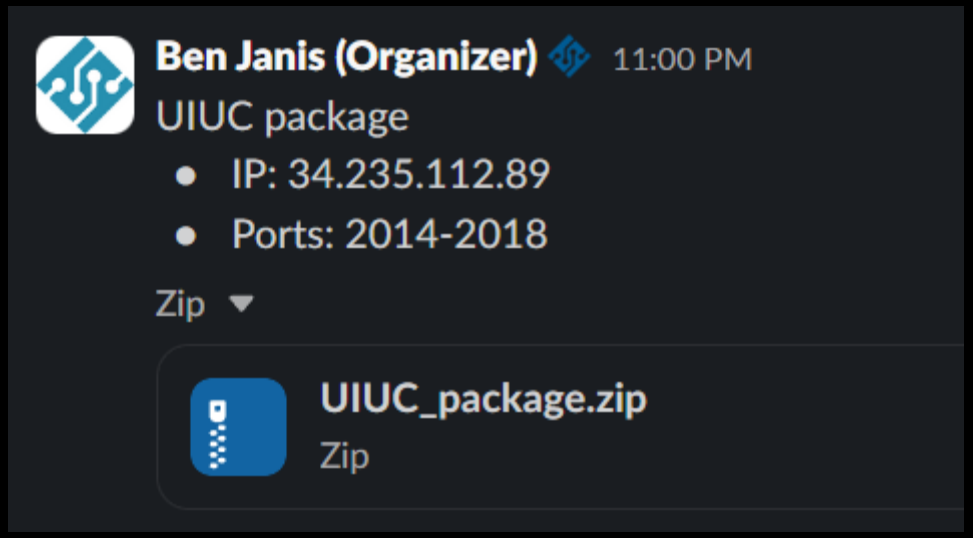

AWS (3P)
Slack API
ectf.ctfd.io
Satellites (team)
AWS (1P)
Leader Service
Slack Bot
CTFd API
Docker Build
Worker Scheduler
Worker Service (Cloud)
Testing Dispatcher
Satellite API
Database
Siebel
Worker Service (Secure)
Testing Dispatcher
Board API
Worker Service (Dev)
Testing Dispatcher
Board API
S0
S1
S2
D0
D1
D2
D3
D4

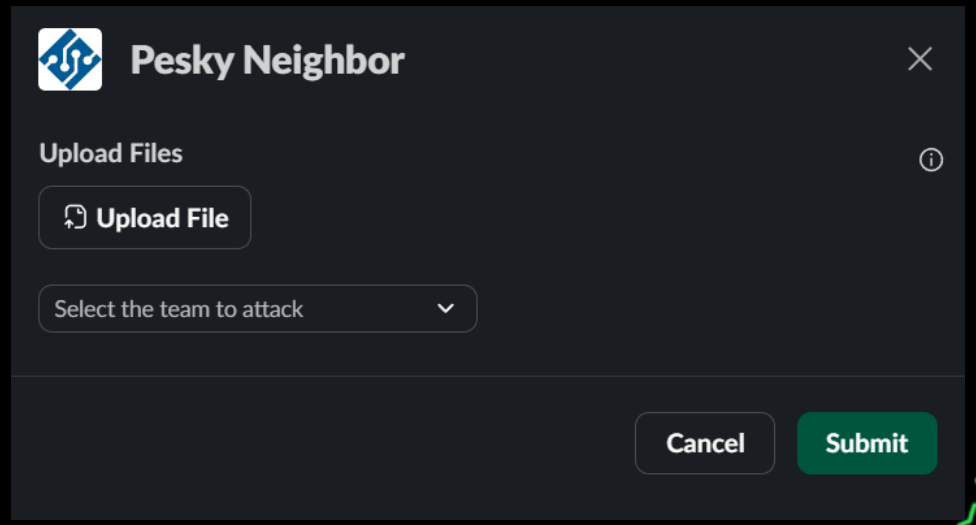
Pesky Neighbor Strategy

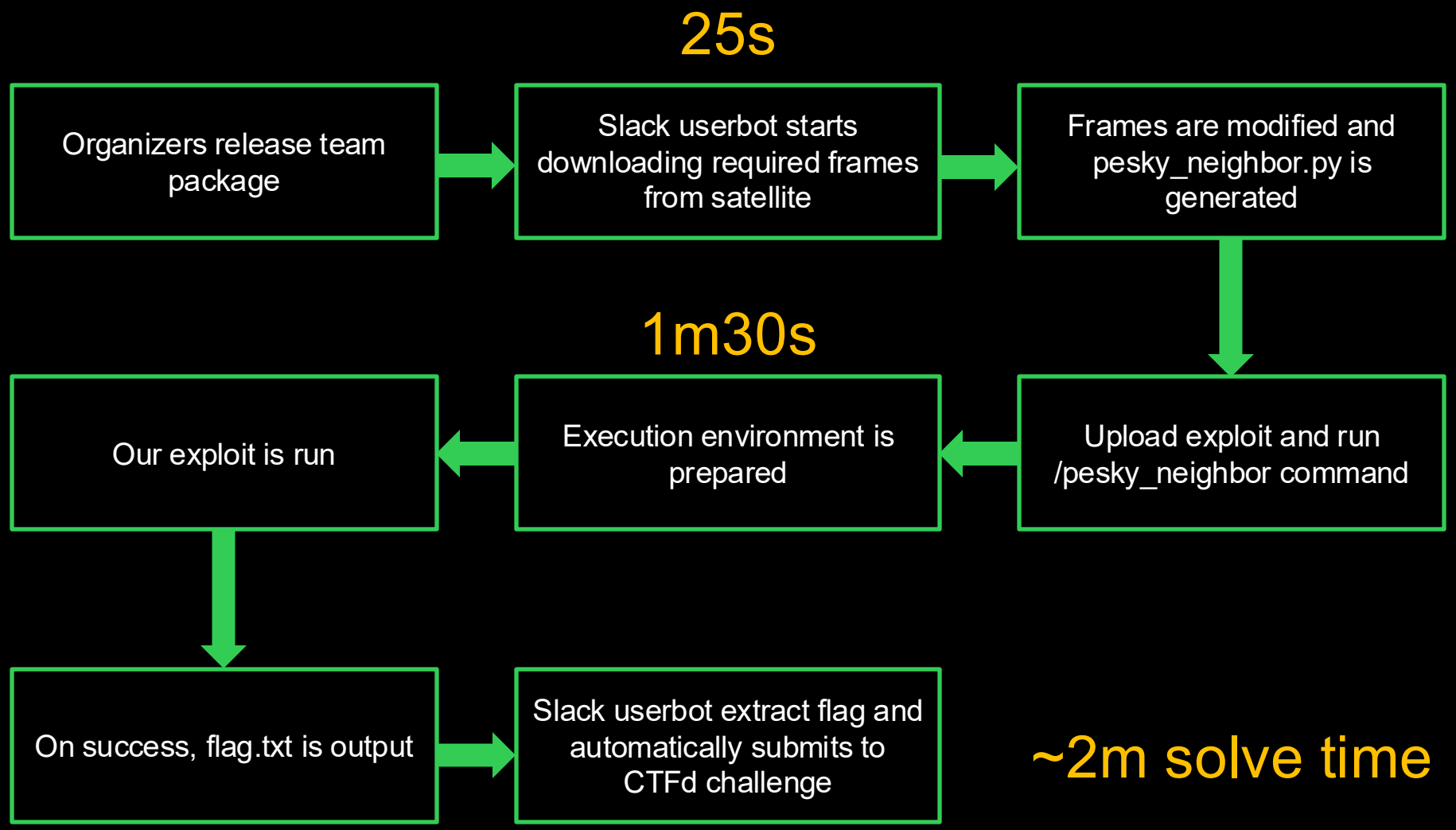
- Collect one frame for channel 1 (frame A1), then three more frames for channel 0 (B0, C0, D0), modify a few bytes of D0 create D0~
  - We primarily collect channel 0 frames since a lot of teams had coded exceptions for emergency channel 0
- Send order: C0, B0, C0, A1, D0~
  - Tests if frames are monotonically increasing because C0 is sent before B0
  - Checks that equal timestamps C0 and C0 are not monotonically increasing
    - Two frames with the same timestamp cannot be sent as later frames should have larger timestamps
  - Per-channel last timestamp attack, violating SR3 (B0, C0, A1)
    - If timestamps are only checked within a channel, send smaller timestamp frame of different channel
  - Unauthenticated frame data where manipulated ciphertext causes manipulated plaintext, violating SR2 (D0~)

Pesky Go Brrrrrr

- Submitting attacks to organizer's pesky service happened over Slack app commands
- We had to reverse engineer Slack's API to build a userbot which could do everything automatically
- CTFd flag submissions were also automated

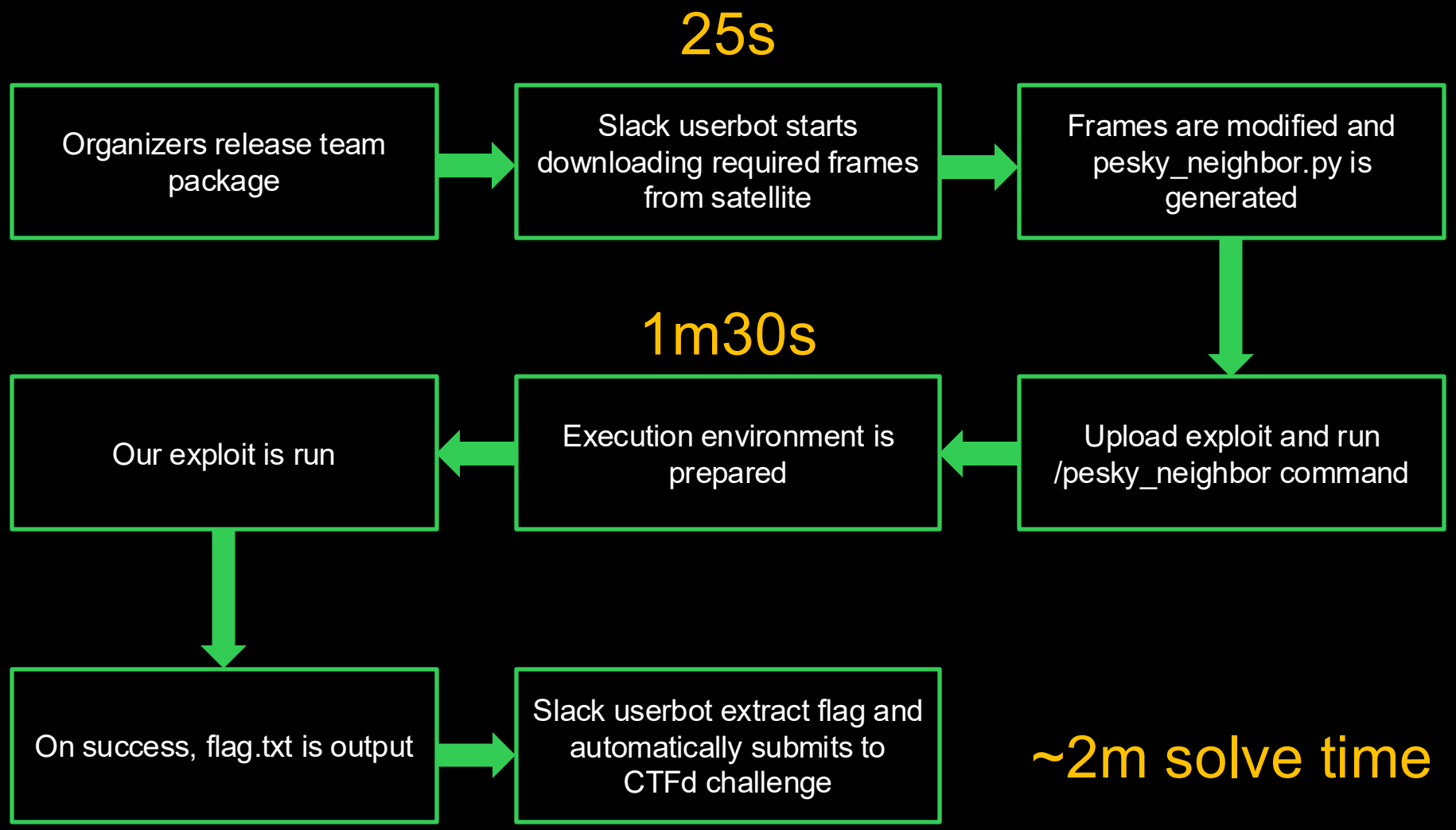


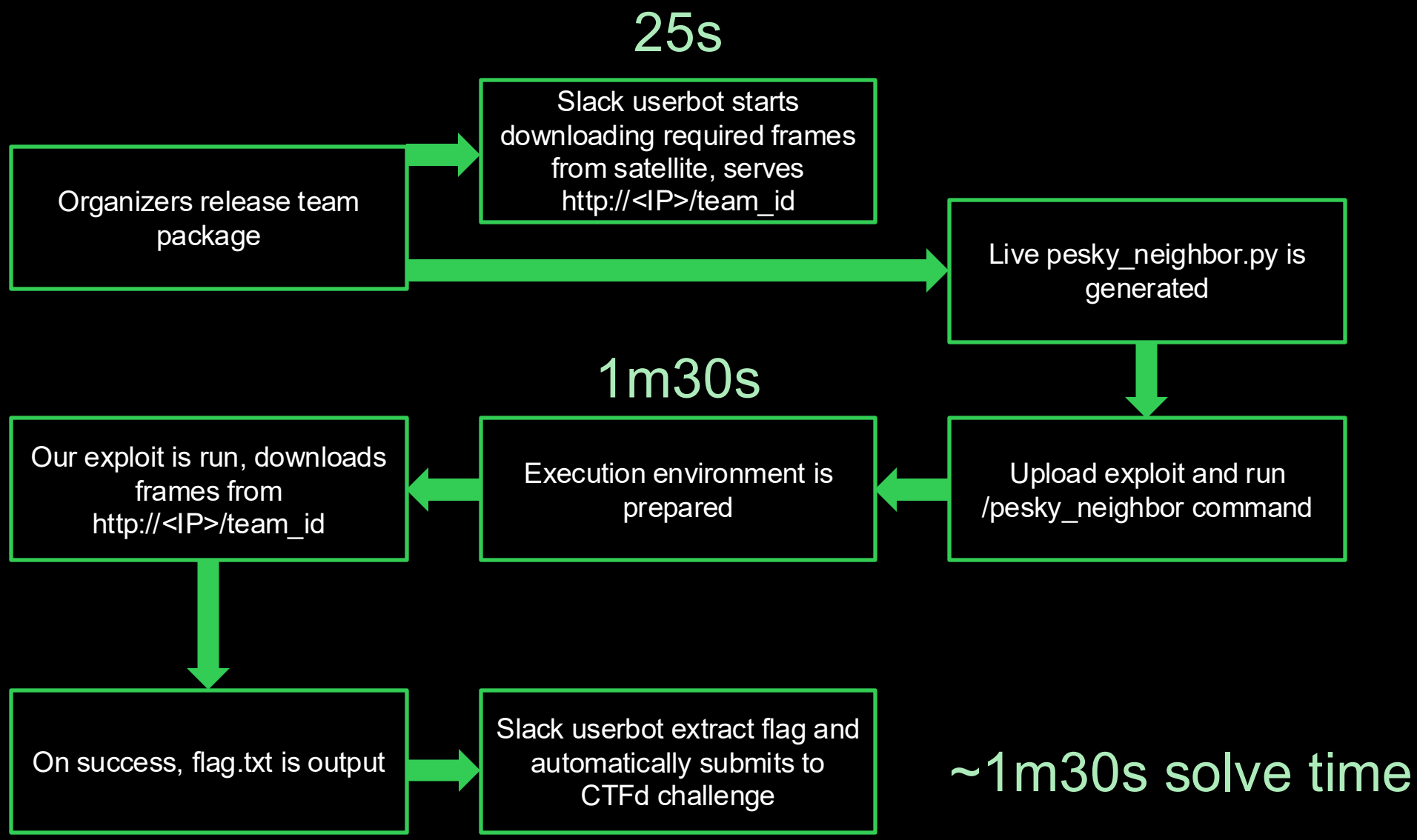



25s
Organizers release team package
Slack userbot starts downloading required frames from satellite
Frames are modified and pesky_neighbor.py is generated
1m30s
Upload exploit and run /pesky_neighbor command
Execution environment is prepared
Our exploit is run
On success, flag.txt is output
Slack userbot extract flag and automatically submits to CTFd challenge
~2m solve time

Pesky Go Brrrrrrrrrrr

- Very important speedup: parallelizing frame collection with command submission
  - Collecting 4 frames (channel 0) from the satellite takes ~25s
  - Organizer setup takes ~1m30s before our script is called
  - These two operations are independent!
- Frames from the satellite are collected on a VPS (cloud server) as soon as the team's ports are released
- When pesky_neighbor.py runs, it downloads these pre-collected frames from our VPS, saving ~25s
- We achieved the fastest pesky submission time of 1m27s!


25s
Organizers release team package
Slack userbot starts downloading required frames from satellite
Frames are modified and pesky_neighbor.py is generated
1m30s
Upload exploit and run /pesky_neighbor command
Execution environment is prepared
Our exploit is run
On success, flag.txt is output
Slack userbot extract flag and automatically submits to CTFd challenge
~2m solve time

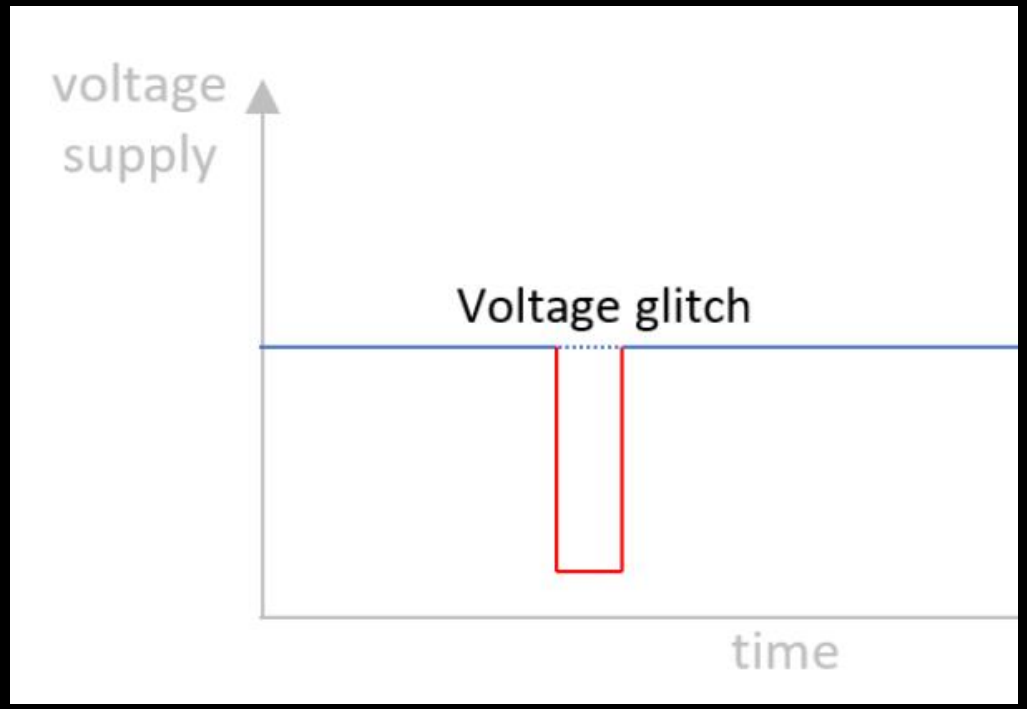

25s
Organizers release team package
Slack userbot starts downloading required frames from satellite, serves http://<IP>/team_id
Live pesky_neighbor.py is generated
1m30s
Upload exploit and run /pesky_neighbor command
Execution environment is prepared
Our exploit is run, downloads frames from http://<IP>/team_id
On success, flag.txt is output
Slack userbot extract flag and automatically submits to CTFd challenge
~1m30s solve time

Hardware Attacks

When "secure" software isn't secure

Intro to Voltage Glitching

- Hardware attack that attempts to corrupt program state (control or data) by briefly pulling microcontroller power to ground
- Common effect is to skip an instruction (some pipeline stages don't get enough power)
- Glitch needs to be precisely timed, so need to trigger based on some IO (GPIO is ideal)



Idealized voltage glitch

Modified subscription

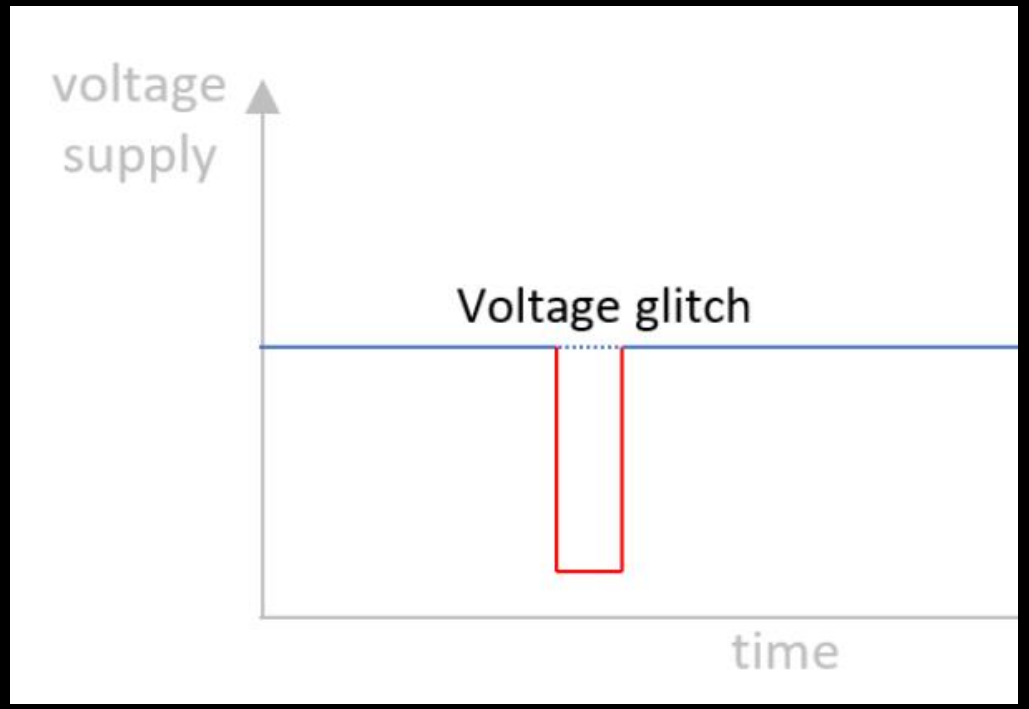
```
... [LED_On]
1000e540 bl decrypt_sym
1000e544 cmp r0 ,#0 // r0 != 0 if authentication fails
1000e546 bne auth_fail
... [r0 == 0 -> process subscription update]
... [r0 != 0 -> process authentication error]
```

Authentication failure! :(

Vulnerable snippet of subscription update logic

Intro to Voltage Glitching

- Hardware attack that attempts to corrupt program state (control or data) by briefly pulling microcontroller power to ground
- Common effect is to skip an instruction (some pipeline stages don't get enough power)
- Glitch needs to be precisely timed, so need to trigger based on some IO (GPIO is ideal)



Idealized voltage glitch

Modified subscription

```
... [LED_On] // trigger on GPIO out
1000e540 bl decrypt_sym
1000e544 cmp r0 ,#0 // r0 != 0 if authentication fails
1000e546 bne auth_fail // skip this instruction
... [r0 == 0 -> process subscription update]
... [r0 != 0 -> process authentication error]
```

Glitch offset

Subscription Update Successful! :)

Vulnerable snippet of subscription update logic

Glitching Challenges

- Glitch duration must be precise
  - If too short, no effect
  - If too long, board hits a hard fault
  - Solution: experimentation + brute-force search for glitch width
- Glitch timing must be precise
  - Need a reliable trigger source
  - Solution: tap LED voltage and use as trigger
  - Need to find the right timing offset
  - Solution: place a trigger in the target to determine the offset
- Board is actively working against you
  - Decoupling capacitors try to keep voltage stable
  - But a sharper glitch is more precise
  - Solution: desolder decoupling capacitors
- Does not always work even with good parameter values
  - Worse, clocks aren't synchronized, so timing isn't perfectly repeatable
  - Solution: automate repeated attempts

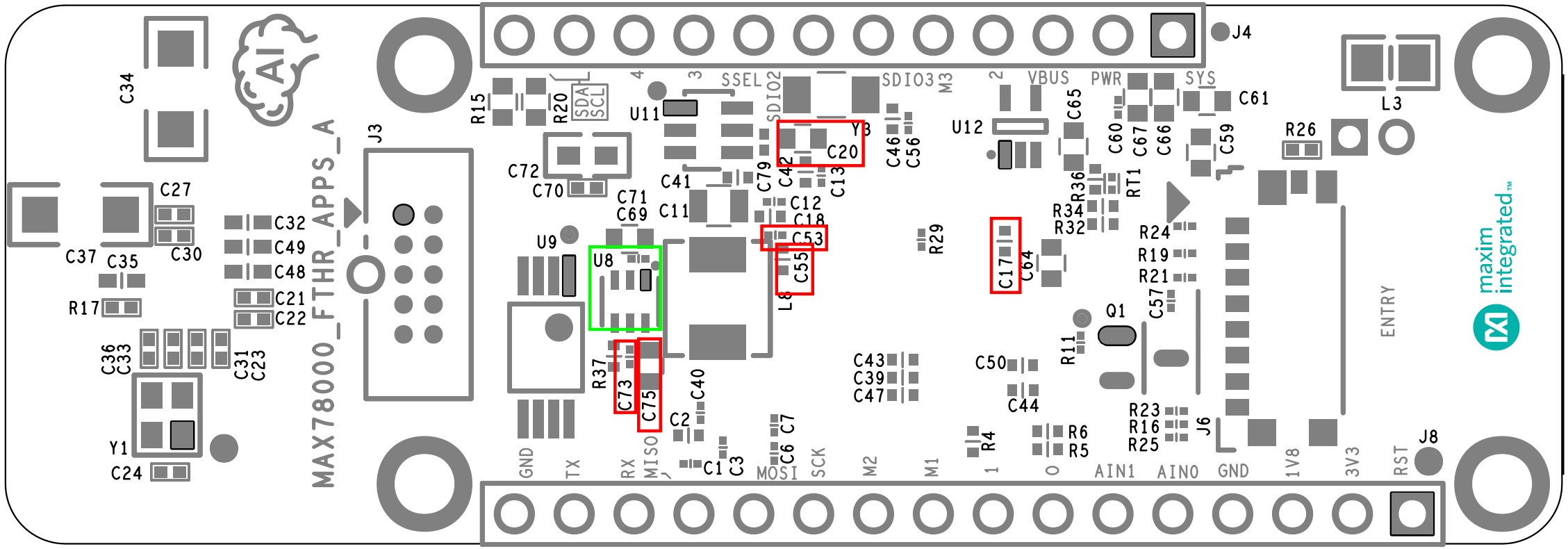
Shopping List

- $600 – ChipWhisperer Husky
- $300 – Oscilloscope
- $130 – PCBite probe kit
- $100 – Additional MAX78000FTHR boards
- $100 – Soldering supplies (station, solder, flux, wick, chisel tip)
- $20 – Multimeter (for continuity testing)
- and more from assorted components/jumper wires…

- Total: **$1,500**

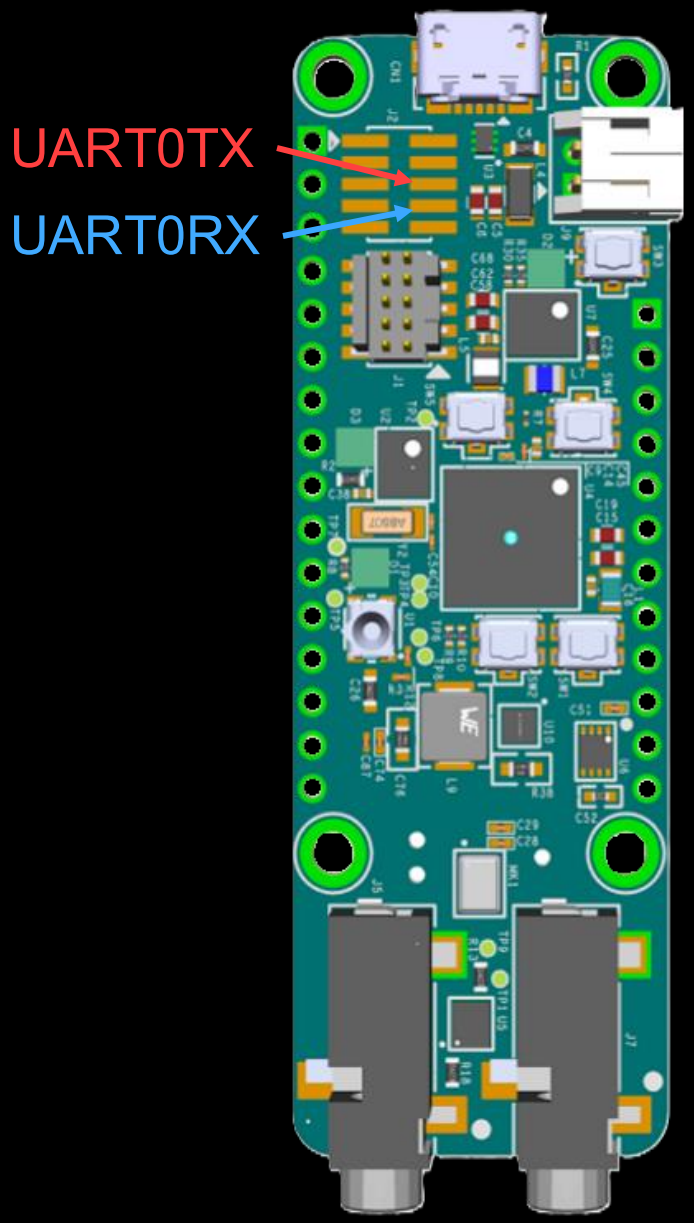
Board Instrumentation

- Remove decoupling capacitors by pushing them off with your chisel tip for soldering iron
- Attach probe/wire between:
  - VCOREA and CW glitch
  - LED and CW trigger pin
  - PCB and schematic PDFs are your friends

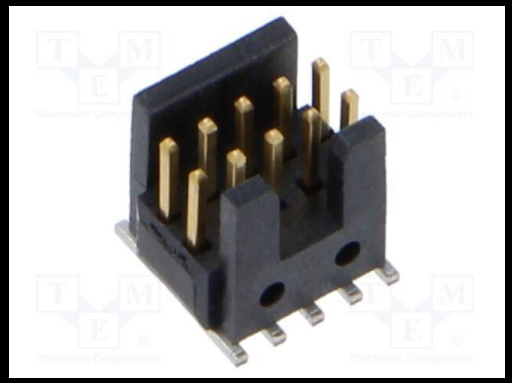


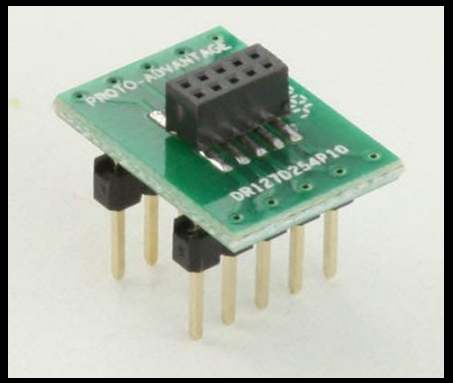
Red: Removal of capacitors C17, C20, C53, C55, C73, C75 (skipped removal of C9 on opposite side)

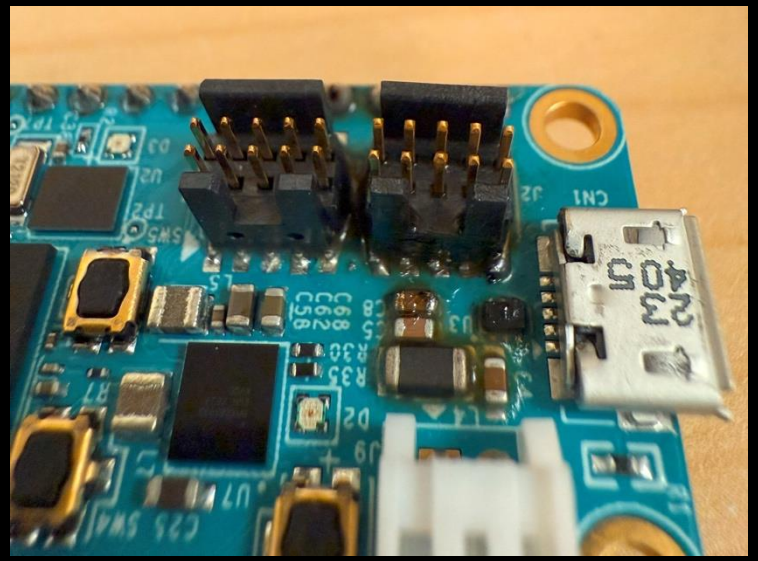
Green: Removal of buck converter (U8)

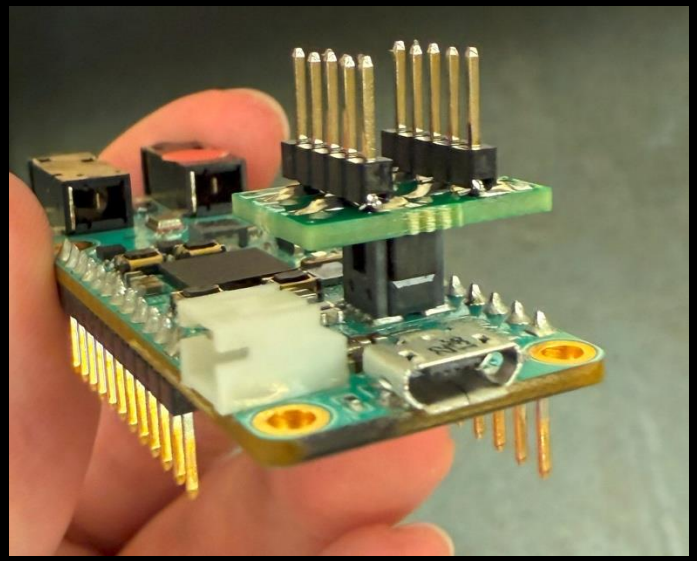


UART0 isn’t available on the existing feather GPIO pins, so we soldered an extra header and used an adapter to break them out (flying wires also would work).







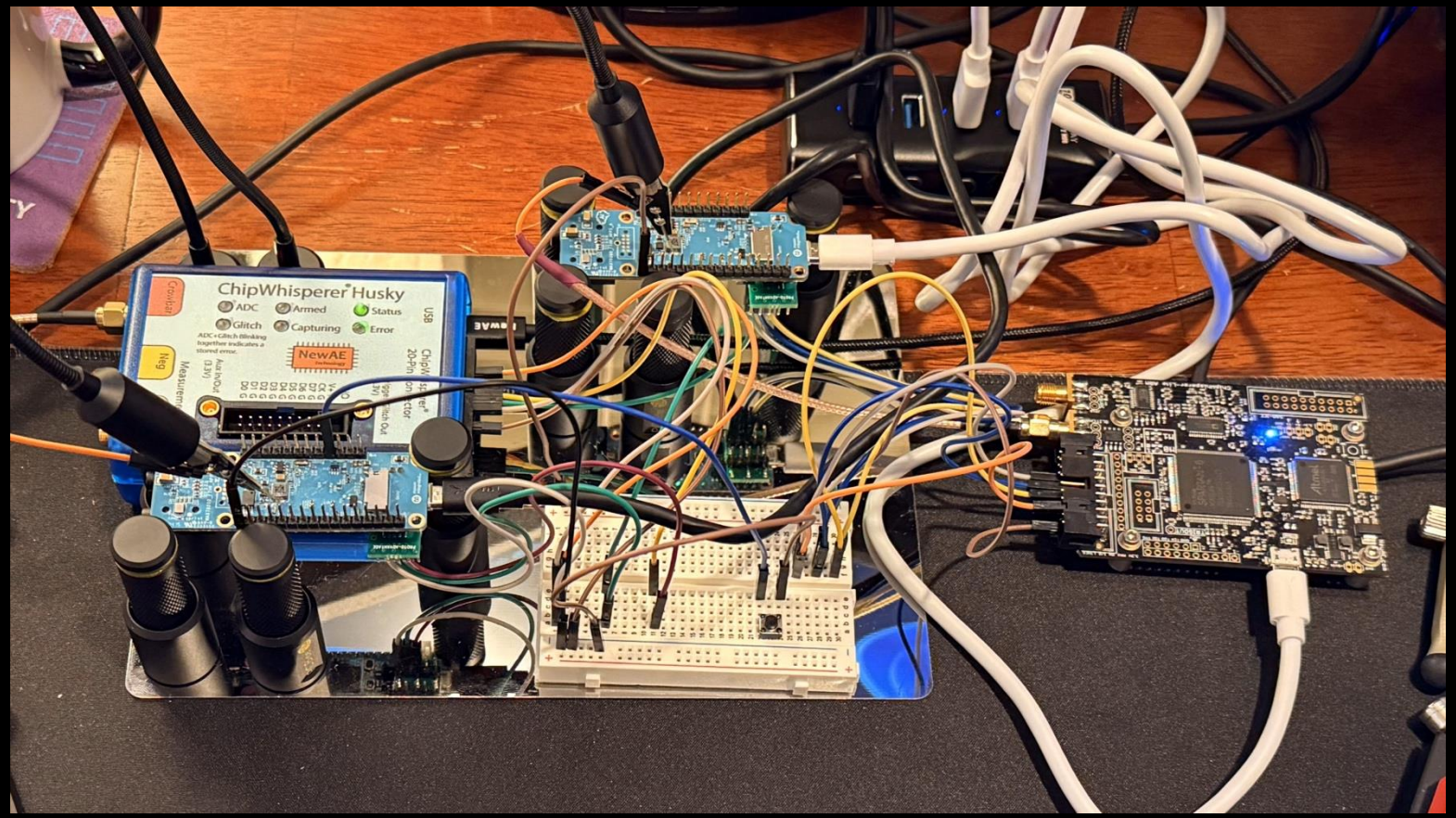


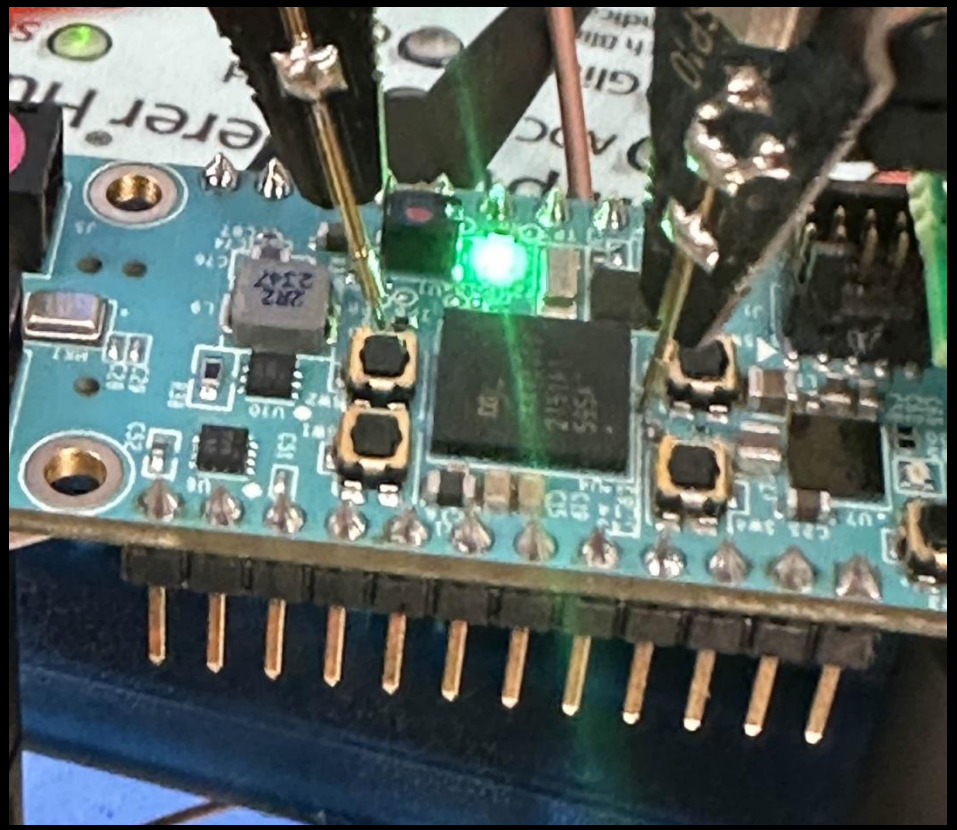
Experimentation

- Validate glitches against *simple* test programs
  - Start with a big loop and manually sending glitches
- Build your way up to programs resembling the targets
- Learn what kind of behavior you can induce with which parameters
  - Write a hard fault handler: see what's happening
- Be systematic: record results, one change at a time
  - Patience is a virtue: <0.1% success rate can happen

Exploitation

- Once you have a glitch in a similar test firmware, it's time to try it for real
- Add a new GPIO trigger adjacent to the targeted code
  - Beware, the compiler may reorder things now!
- Use your scope to measure the offset between the real, production trigger (LED) and instrumented trigger
  - Send a short glitch with the CW at this offset
  - Use the oscilloscope to validate your offset
- Use this offset and the width from before to attempt a glitch





Early glitch attempts above:

- UART triggering with CW-Husky
- Spray and pray with CW-Lite
- Probes are attached to L8 (inductor/VCOREA for glitch)

Left probe: R10 (green LED for trigger)
Right probe: C9 (VCOREA for glitch)

Closing Thoughts

- Voltage glitching is hard
  - Intersection of Computer Science and Electrical Engineering
- We aren't experts: we would love to compare notes with other teams
- The process we outlined is crude: specialized knowledge, equipment, and techniques enable powerful attacks

Lab Setup (Before)

Lab Setup (After)

Thank you! Any questions?

eCTF10

10 YEARS OF THE EMBEDDED CAPTURE THE FLAG

Presentation: Carnegie Mellon University

Currently in 1st place with 55,350 points

Not final score

Time (EST)	Title
15:30	Competition Overview
15:50	NEMC Guest Speaker – John Petrozzelli
16:00	City College of San Francisco
16:15	Mountain View High School
16:30	University of Michigan
16:45	Keynote – Mark Peters, Ph.D.
17:00	Refreshment Break
17:15	Purdue University
17:30	University of California, Los Angeles
17:45	University of Illinois Urbana-Champaign
18:00	Carnegie Mellon University
18:15	Award Presentations
18:25	Closing Remarks – Dan Walters
18:30	Networking Reception

MITRE

Plaid Parliament of Pwning 2025 eCTF Team

Carnegie Mellon University

Om Arora, Sky Bailey, Taha Biyikli, Rohil Chaudhry, Samuel Dinesh, Daniel Ha, Akhil Harikumar, Janice He, Peiyu Lin, Harrison Lo, Leonardo Mouta, Matin Sadeghian, Carson Swoveland, Surya Togaru, and Haonan Yan

Advisors: Anthony Rowe, Patrick Tague, and Maverick Woo

Outline

Design Phase

- **Threat modeling**
- **Highlight: Tree**
- **Hardening & Auditing**

Attack Phase

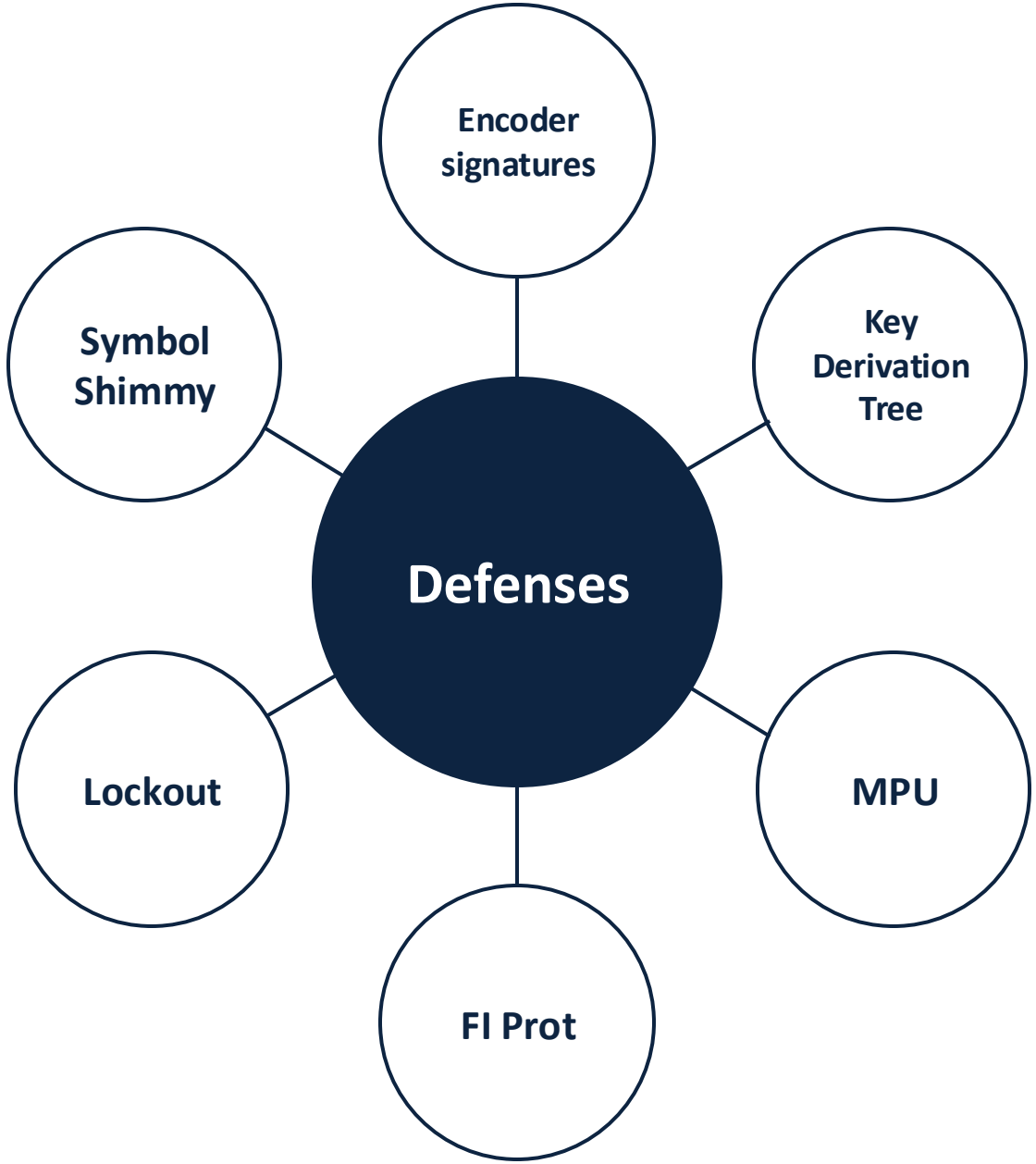
- **Voltage Glitching**
- **FlashCat**

Lessons Learned

- **Team Management**
- **C & Secure Design**

Design Overview

Secure by design



Threat Modeling

Good Protocol Design is the **ONLY 100% Effective Defense**

(for eCTF 2025)

Level-by-level analysis during design phase:

1. Assume attacker has **arbitrary code execution**
   Defend as many flags as possible
2. Assume attacker has **arbitrary read**, but **no code exec**
   Defend as many flags as possible
3. Assume attacker **knows** only **some keys**
   Defend as many flags as possible
4. etc.

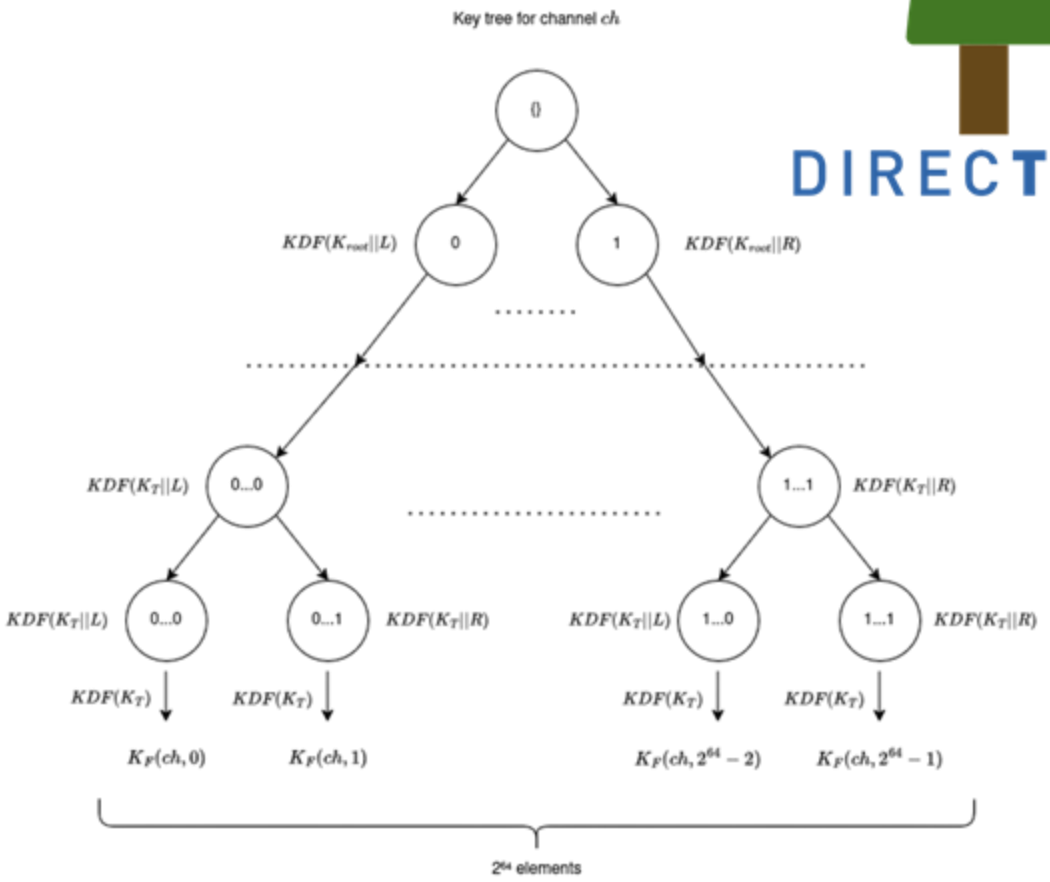
Design Highlight

DIRECTREEV

Binary key-derivation tree

Subscription includes **minimal set of nodes** required for given time range

- Even an attacker with **arbitrary access** still **cannot decode inactive frames**
- Timestamps are enforced by **protocol, not software**



Hardening!

Our protocol protects us even if we get pwned, but we should still make it difficult to do so

- Fault injection -> **randomized delays**
- Brute force attacks -> 5-second lockout **persists across resets**
- Shellcode -> MPU to set NX permissions
- ROP -> Randomize function address at build time

```
fiproc_delay();
frame_ch_t timestamped_frame = {};
if (decrypt_symmetric((uint8_t*)&timestamped_frame, packet->payload.enc_frame,
                      sizeof(timestamped_frame), sub->kch) != OK) {
    // inner decryption is corrupted but signature passes means attack
    attack_detected();
    return ERROR;
}
```

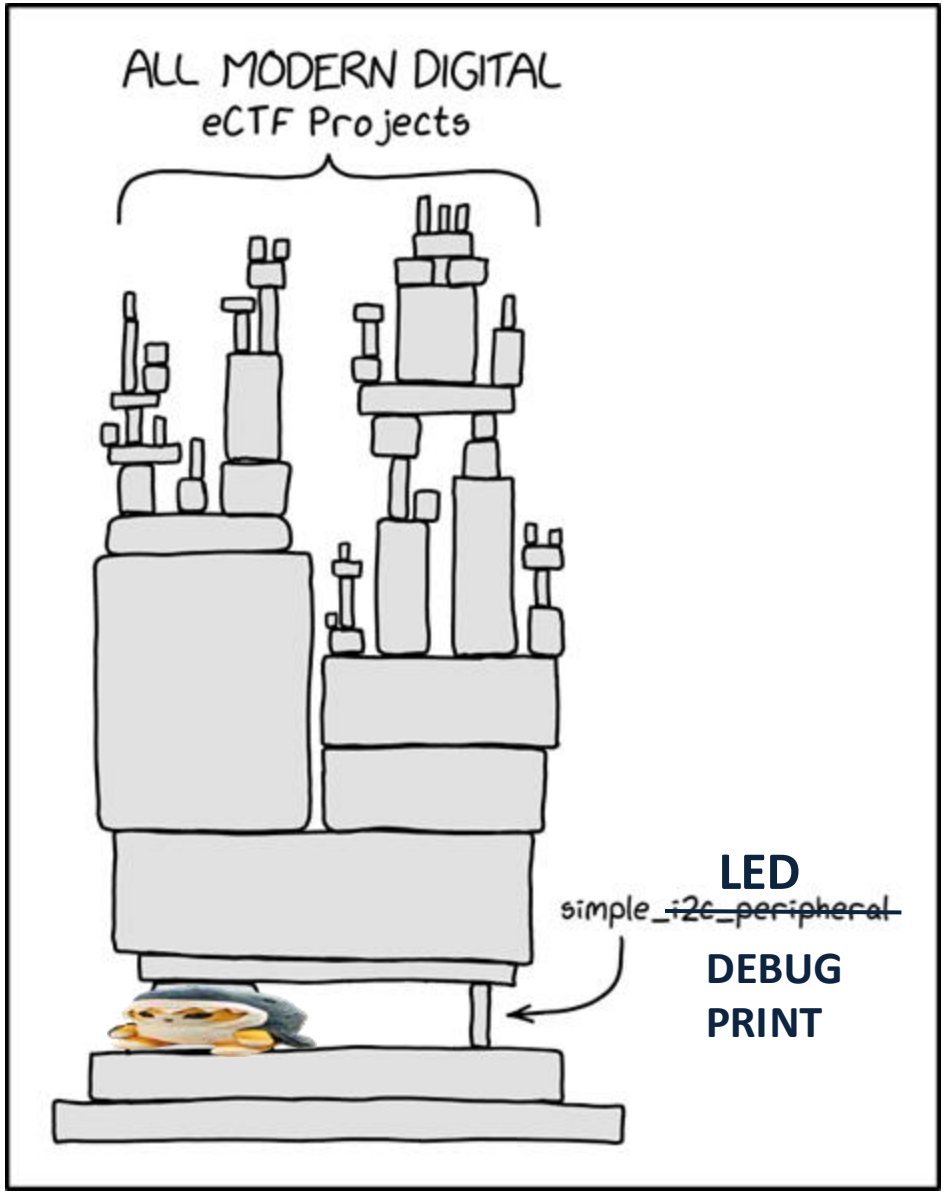
Auditing

Read every single line of source code **and generated assembly**:

- Caught a buffer overflow in host messaging
- Made sure compiler didn’t optimize away our hardening measures
    - Reached out to Monocypher's author to confirm effect of optimization

We experimented w/ SonarCloud, but we did not find anything with it

Attack Overview



Voltage Fault Injection (FI)

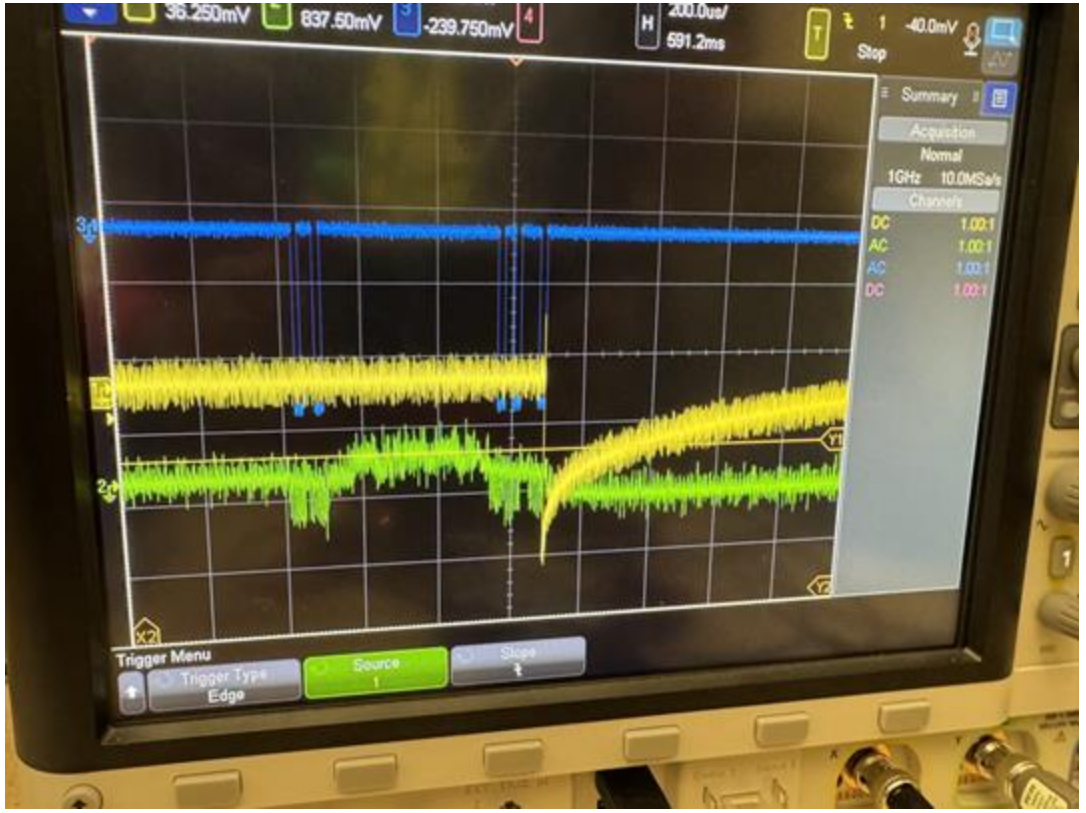
Electronics need power to work

Normal voltage?
CPU executes normally

Voltage too low?
CPU crashes/resets/powers off

Voltage **briefly** too low?
*CPU briefly unstable **but continues to run code***

Most common result: **Instruction Skip**



FI, Continued

Successful FI has **Two Requirements**

Trigger Point

Need *precise timing* to glitch (16ns/instruction!!)

Most Common Trigger: Changing Status LED color

Single Point of Failure

Can only skip **one** instruction

Requires assembly audit

Most Common Target: UART write_bytes()

FlashCat

`write_bytes()` from Insecure Example

```
for (int i = 0; i < len; i++) {
    uart_writebyte(buf[i]);
}
```

After Compiler Optimizations…

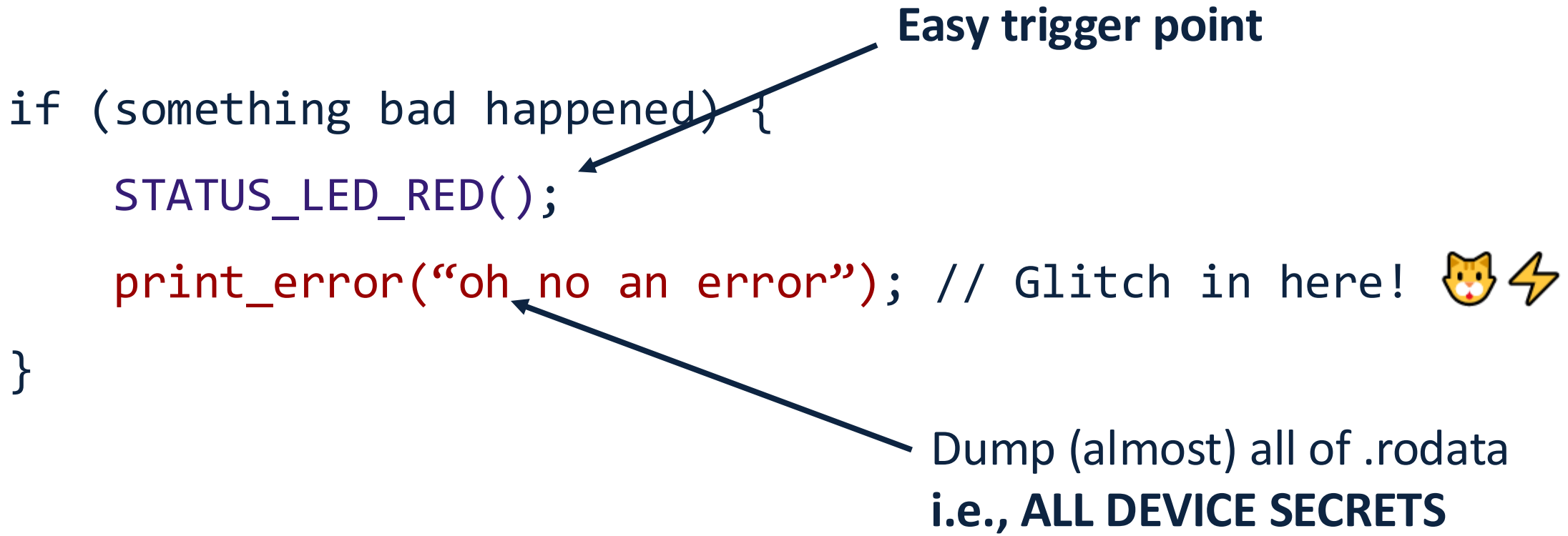
```
i = 0;
p = buf;
while (i++ != len) uart_writebyte(*p++);
```

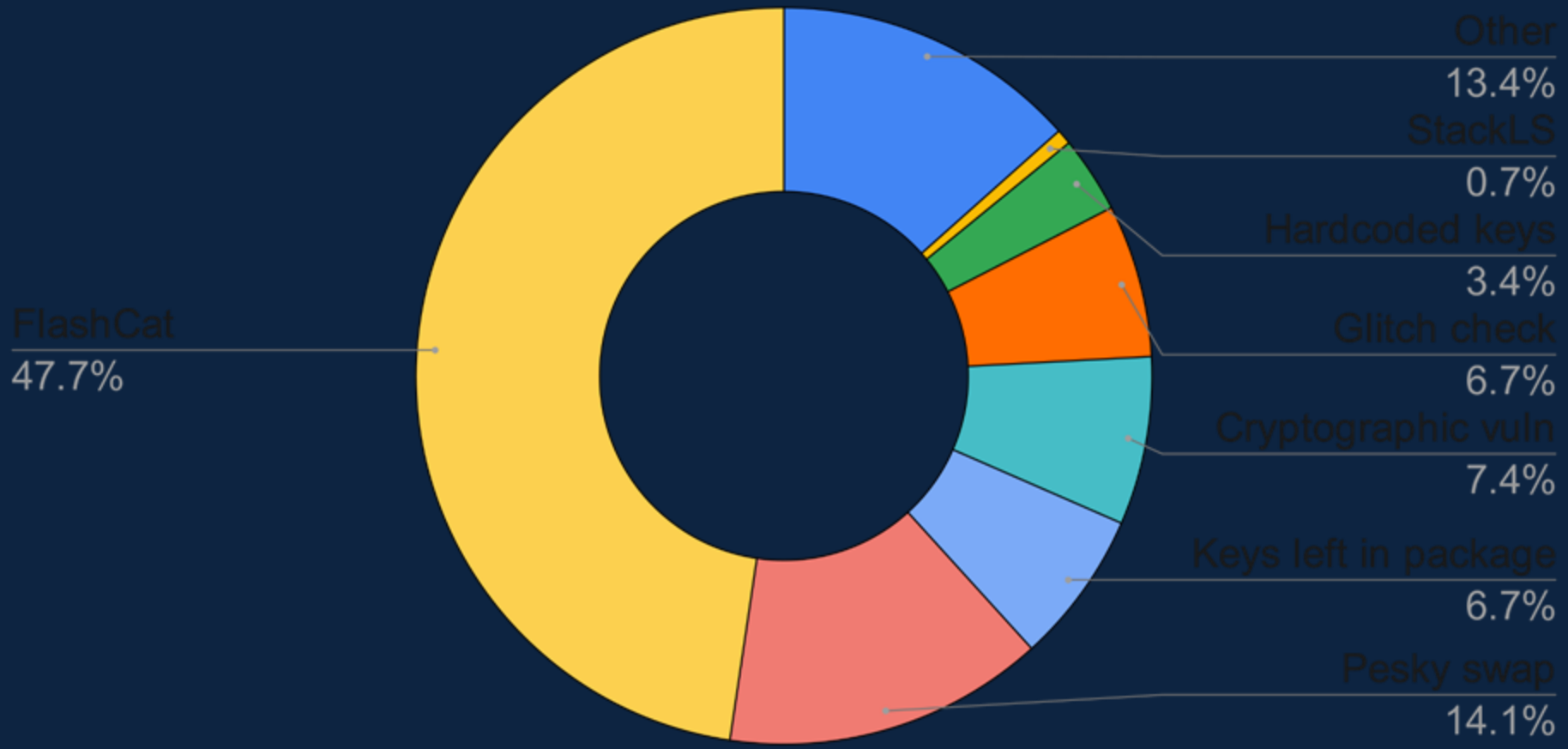
Loop condition never terminates if `i > len`

```
~~i = 0;~~ // SKIPPED!
p = buf;
while (true) uart_writebyte(*p++);
```

Result: print all data that comes after `buf`

FlashCat


Easy trigger point
if (something bad happened) {
STATUS_LED_RED();
print_error("oh no an error"); // Glitch in here!
}
Dump (almost) all of .rodata
i.e., ALL DEVICE SECRETS


Other
13.4%
StackLS
0.7%
Hardcoded keys
3.4%
Glitch check
6.7%
Cryptographic vuln
7.4%
Keys left in package
6.7%
Pesky swap
14.1%
FlashCat
47.7%

Wrap-Up

Why Do We Still Use C?

Comes back to **threat modeling**

eCTF is a **hardware competition** centered around **cryptography**.

Rust **protects against memory unsafety,**
NOT incorrect cryptography and hardware attacks!

Downsides of Rust:

- Most team members were not already familiar with Rust
- Rust (like any compiler) can subtly optimize away hardening
  - **This has directly led to at least two captured flags!**
  - LTO'd code is difficult to audit and verify by hand!

Team Structure to the Rescue!

Project Manager and Tech Lead roles were defined early on, and tasks were assigned to specific people each week.

- Prevents scope creep
- Improves task prioritization and tracking
- Ensures balanced work distribution -> increased efficiency
- Each team member plays to their strengths
- Team achieved better work-life balance (?)

Good organization allowed us to submit our design on schedule.

Lessons Learned

LEDs and prints can be abused

Review object code, not just source code

Fault injection is highly time-intensive

Assume worst case in threat model - set trust boundaries

Delegate clear responsibilities to meet deadlines

Moments to Cherish!

Sponsors Acknowledgement

We acknowledge the generous support of the following sponsors to our team:

CyLab IoT Initiative
AT&T
AWS
Cisco
Infineon
Nokia Bell Labs
Rolls-Royce
Siemens

(Any opinions, findings, and conclusions or recommendations expressed in this material are those of our team and do not necessarily reflect the views of our sponsors.)

Thank you!

Questions?

eCTF10

10 YEARS OF THE EMBEDDED CAPTURE THE FLAG

Award Presentation

Time (EST)	Title
15:30	Competition Overview
15:50	NEMC Guest Speaker – John Petrozzelli
16:00	City College of San Francisco
16:15	Mountain View High School
16:30	University of Michigan
16:45	Keynote – Mark Peters, Ph.D.
17:00	Refreshment Break
17:15	Purdue University
17:30	University of California, Los Angeles
17:45	University of Illinois Urbana-Champaign
18:00	Carnegie Mellon University
18:15	Award Presentations
18:25	Closing Remarks – Dan Walters
18:30	Networking Reception

MITRE

2025 eCTF PROLeague

eCTF10
10 YEARS OF THE EMBEDDED CAPTURE THE FLAG

For the first time, professionals joined the eCTF!

Separate scoreboard (ectf-pro.ctfd.io)

Attack Phase-only using student designs

Thank you to the participating teams

1st **Concentric Codebreakers**

2nd **Arrowhead Research**

3rd **L3ak**

MITRE

Special Award

Top High School

Special Award: Top High School

Mountain View High School
MVHS Infosec

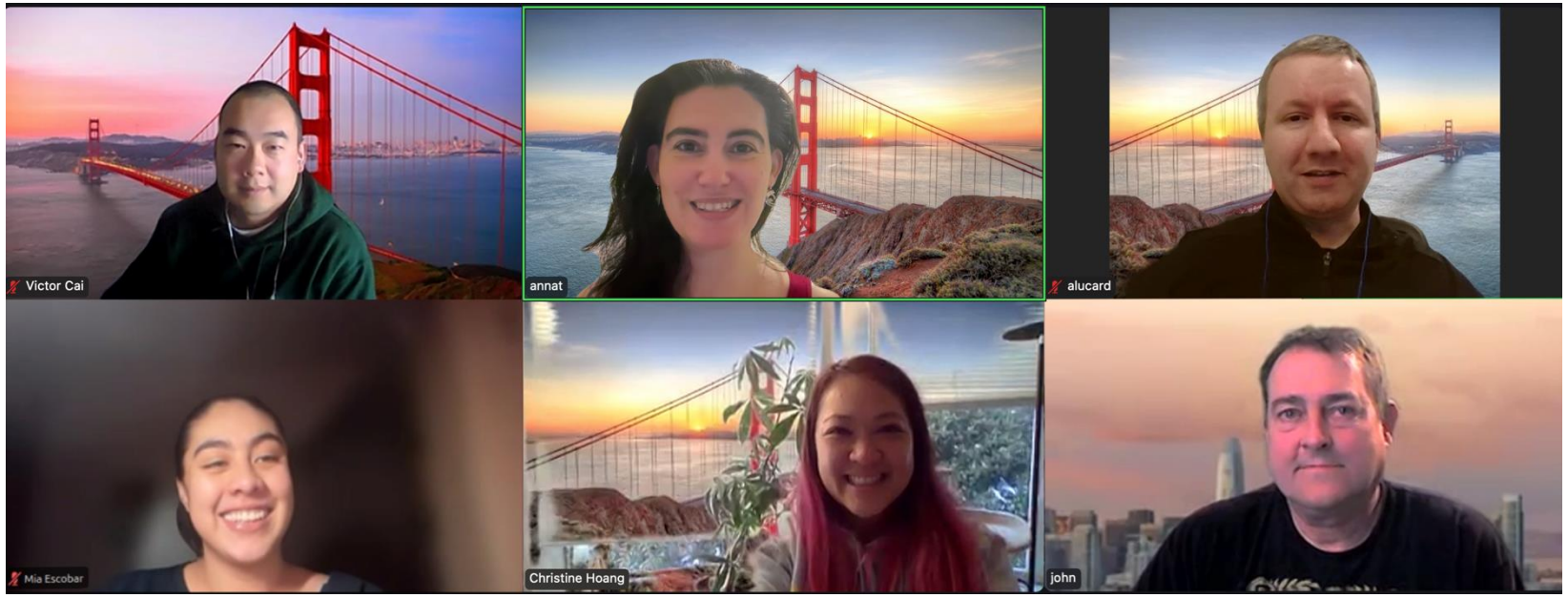
Ryan Yin, Jayden Stenfort, Arthur Cheong, Ethan Wing, Trent McCauley, Ryan Chan, Justin Cheong, Jeremy Yu, Asher Copeland, Jeff Zhang, Max Schexnayder

Advised by: Jennifer Chiu

Special Award

Top Community College

Special Award: Top Community College



City College of San Francisco

Ram Security @ CCSF

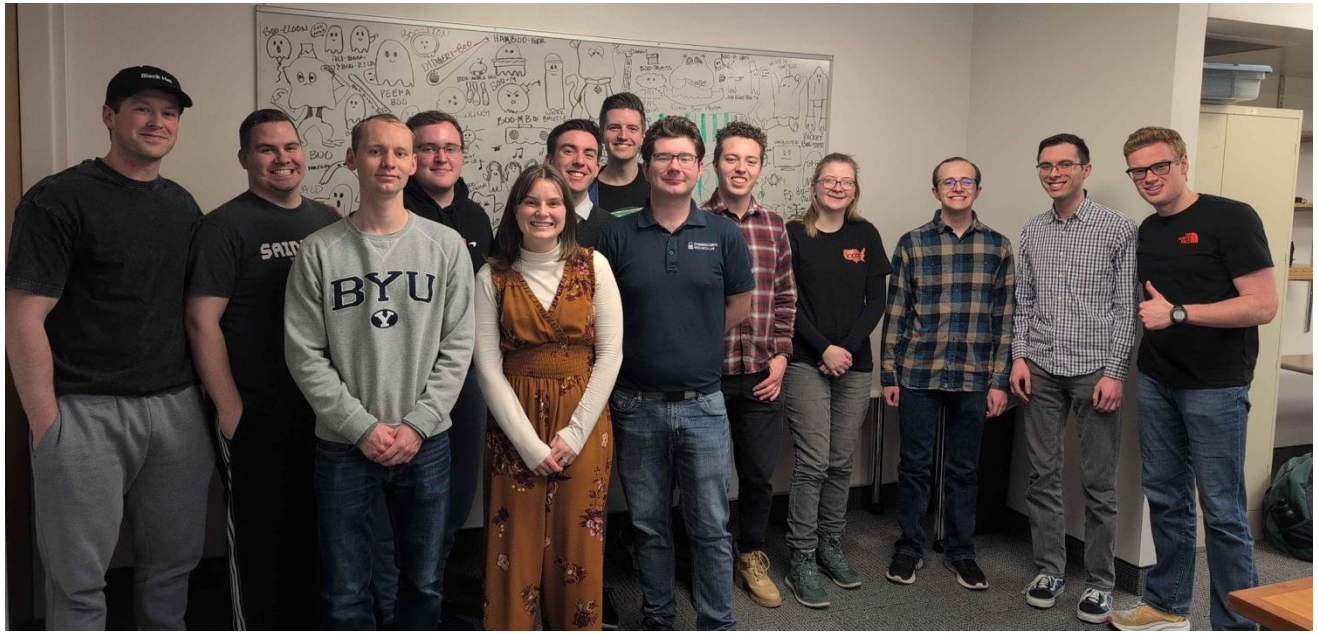
Annat Koren, Christine Hoang, John Refling, Liviu Tancau, Victor Cai

Advised by: Elizabeth Biddlecome, Jonathan Potter, Sam Bowne

Special Award

Attacking Spirit

Special Award: Attacking



Brigham Young University

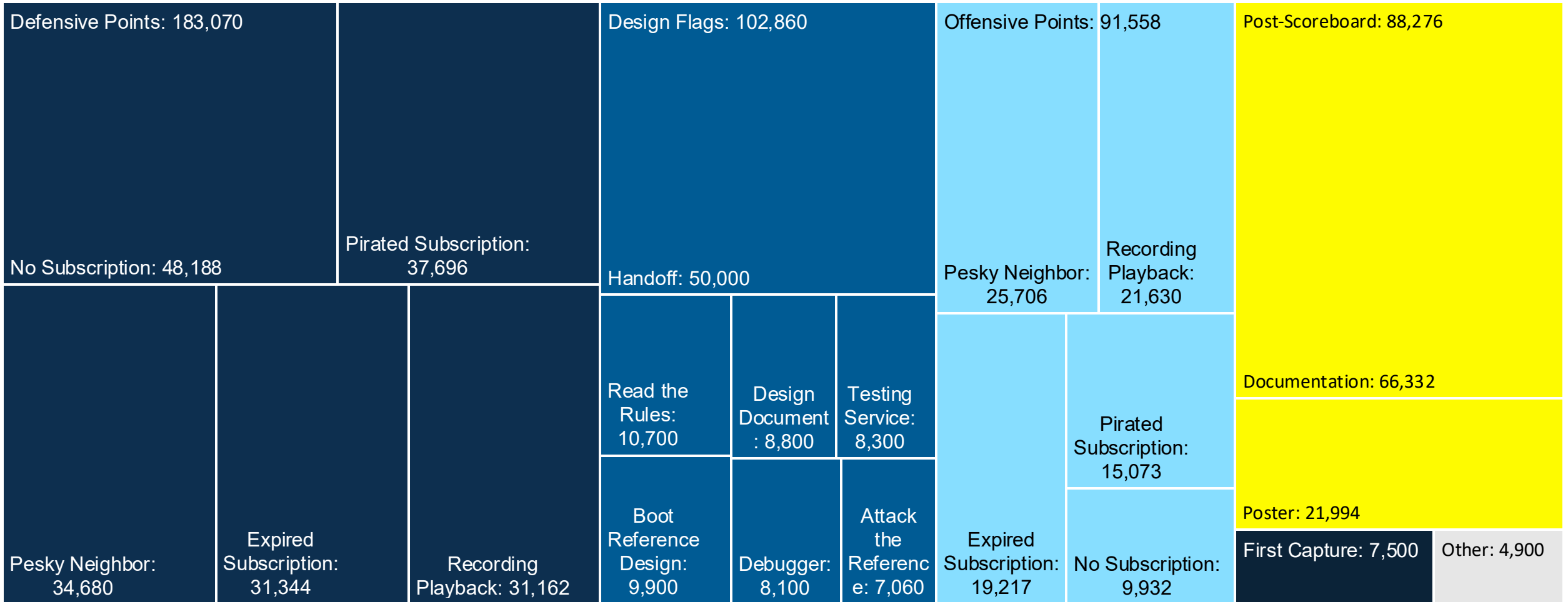
BYU Cyberia

Ava Petersen, Cameron Snider, Caydn Baldwin, Cody Higgins, Daniel Kemp, Evan Cook, Isaac Garfield, Jackson Ouzts, Jordan Williams, Justin Applegate, Kendel Woodburn, Macen Bird, Sam Brescia, Sebastian Hayes, Wyatt Pochman

Advised by: Albert Tay

Final Scoring Breakdown

eCTF10
10 YEARS OF THE EMBEDDED CAPTURE THE FLAG



MITRE

382-point difference!

Scoreboard Scores At Close

Not final scores!

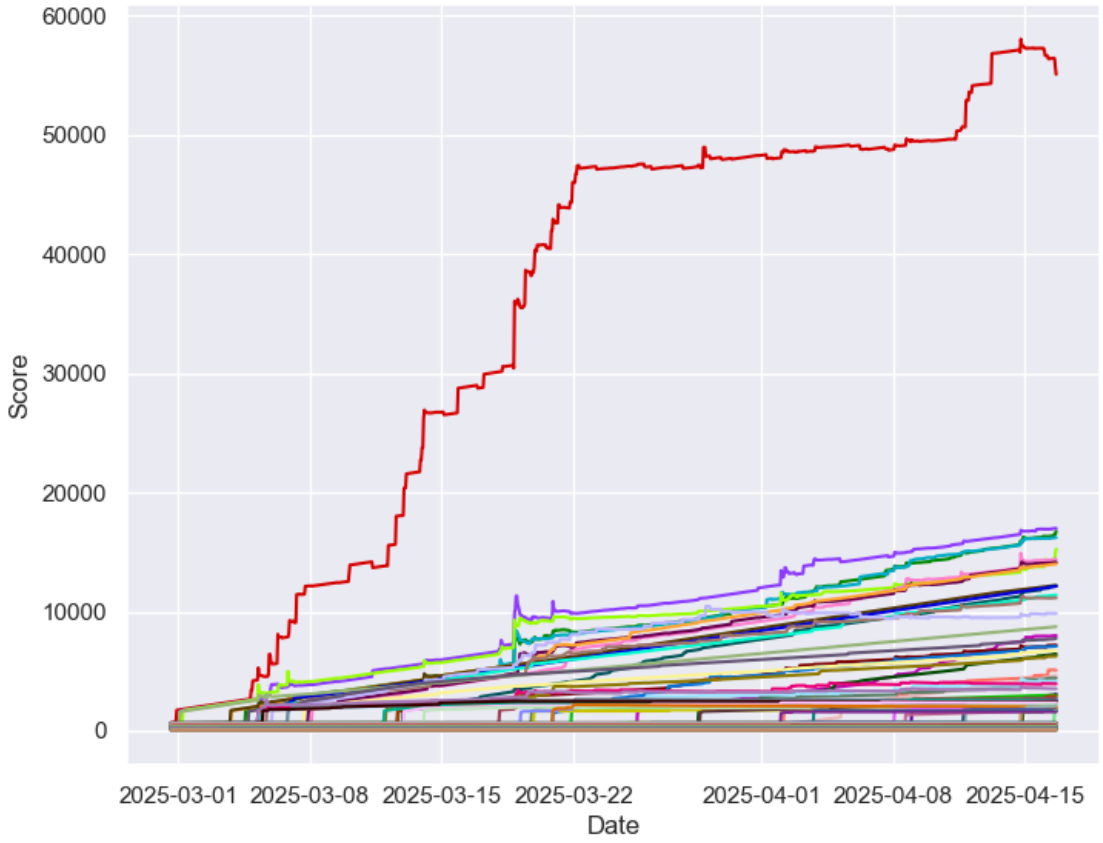
Rank	School	Score
1	Carnegie Mellon University	55,350
2	Purdue University	16,827
3	City College of San Francisco	16,788
4	University of Michigan	16,445
5	University of Illinois Urbana-Champaign	15,276
6	Indian Institute of Technology Madras	14,415
7	University of California, Los Angeles	14,272
8	The Ohio State University	14,115
9	University of South Alabama	12,207
10	Mountain View High School	12,167
11	Massachusetts Institute of Technology	12,133
12	United States Air Force Academy	11,407
13	University of Trento	11,192
14	Brigham Young University	9,884
15	Baldwin Wallace University	8,738
16	University of California, Santa Cruz	8,017
17	University of Connecticut	7,766
18	Georgia Institute of Technology	7,218
19	University of Washington	7,109
20	Northeastern University	6,749
21	Michigan State University	6,465
22	Johns Hopkins University	6,279
23	Purdue University	5,107
24	Parkway Spark!	4,617
25	New York institute of Technology	4,436
26	Binghamton University	4,137

Rank	School	Score
26	Ecole Royale de l'Air	3966
28	Colombe Academy of Technology	3543
29	University of Massachusetts Amherst	3100
31	University of Colorado, Colorado Springs	2991
30	Tufts University	2927
33	University of California, Irvine	2733
32	Tennessee Tech University	2644
34	New York University	2614
35	University of Nebraska Omaha	2592
36	Morgan State University	2514
39	Southeast Missouri State University	2445
38	Flinders University	2346
37	East Tennessee State University	2205
40	Georgia Institute of Technology	2162
41	University of Central Florida	2010
43	The University of Texas at El Paso	1944
44	United States Coast Guard Academy	1764
42	Florida International University	1756
45	University of New Hampshire	1726
48	CyberAegis	1706
47	Northeastern University	1670
45	Rensselaer Polytechnic Institute	1669
49	Indian Institute of Technology Dharwad	1650
50	Oklahoma Christian University	1591
51	Kilgore College	660
52	Rutgers University	650

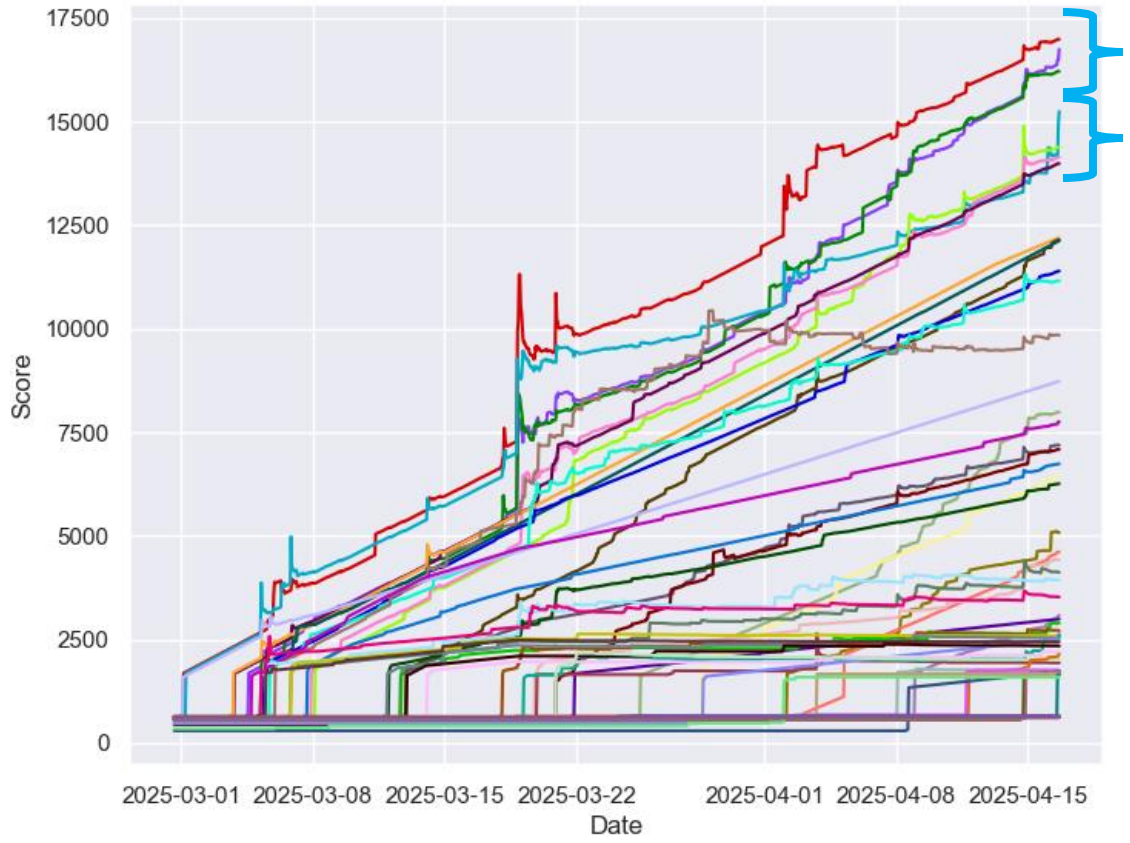
Preliminary Scoreboard Results

eCTF10
10 YEARS OF THE EMBEDDED CAPTURE THE FLAG

Attack Phase



Attack Phase Zoomed

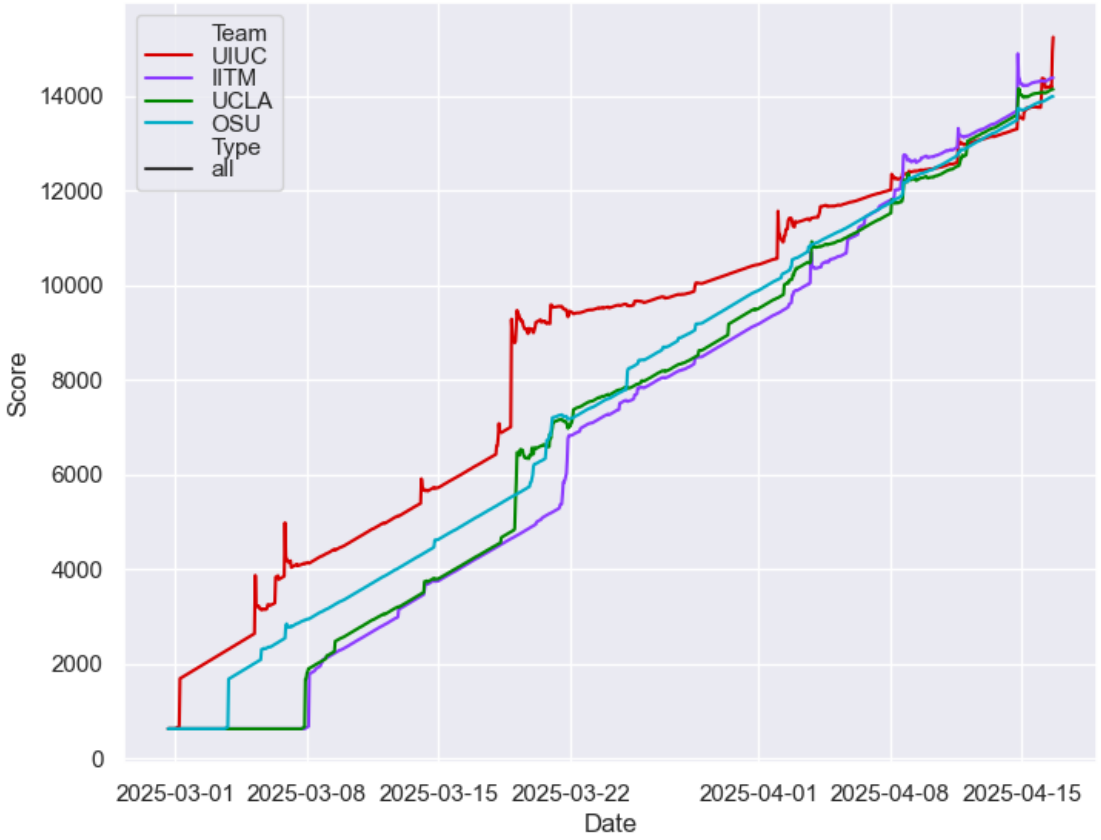


MITRE

Preliminary Scoreboard Results

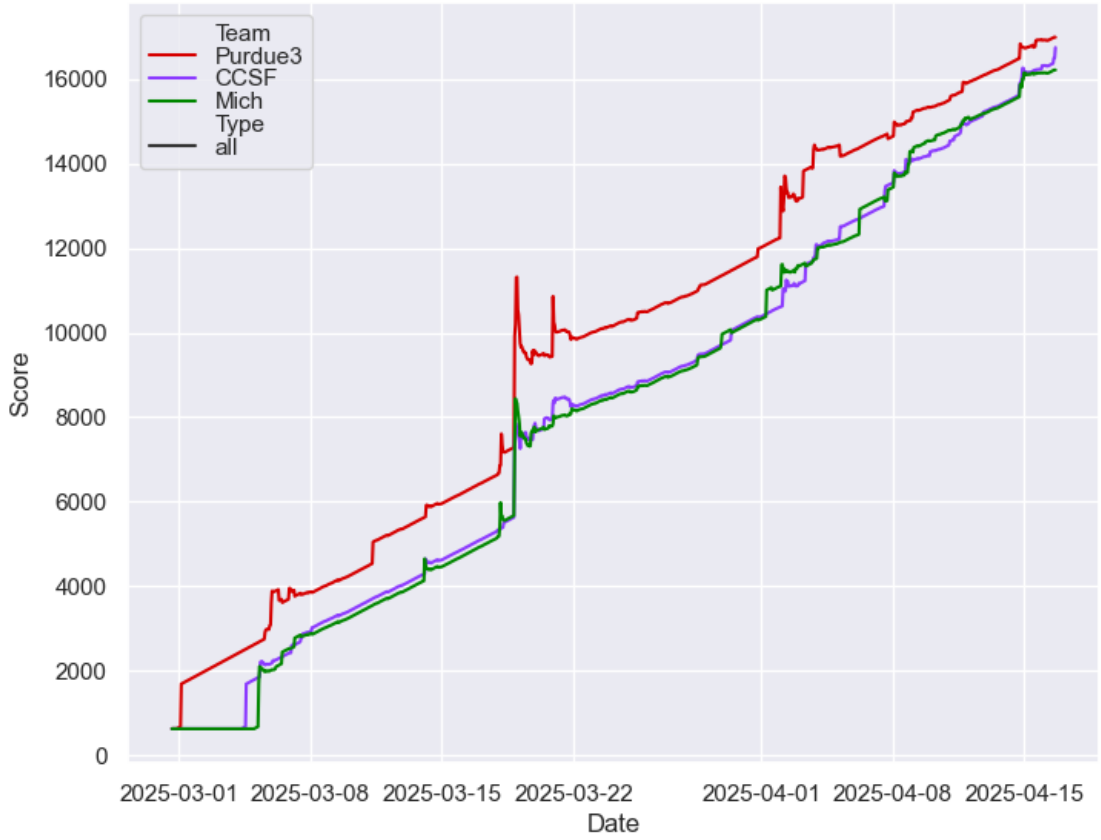
eCTF10
10 YEARS OF THE EMBEDDED CAPTURE THE FLAG

The Race for Fifth



34 position changes!

The Race for Third



25 position changes between CCSF and Michigan!

MITRE

Third Place

$2,500

Third Place

University of Michigan

WolvSec

Allie Yuan, Andre Quimper Osores, Andrew Plotner, Angela Matta, Angie Leong, Brennen Daudlin, Charlie Herz, Elliot Kupchik, Ethan McKean, Faizan Darsot, Haris Khan, Ian Zhang, Jackson Donaldson, Jacob Marchionda, Jeremy Shere, Joshua Cho, Katelyn Ha, Lexa Enders, Liam Domegan, Matthew Dowling, Nick Gamota, Nikye Nixon, Sage Ullman, Sahil Sawant, Shin Lee, Shinjo Satoh, Sudeepti Rao, Tanay Sharma, Tanishka Nalawade, William Zhang

Advised by: Matthew Bernath, Paul Grubbs

Second Place

$5,000

Second Place

Purdue University

b01lers

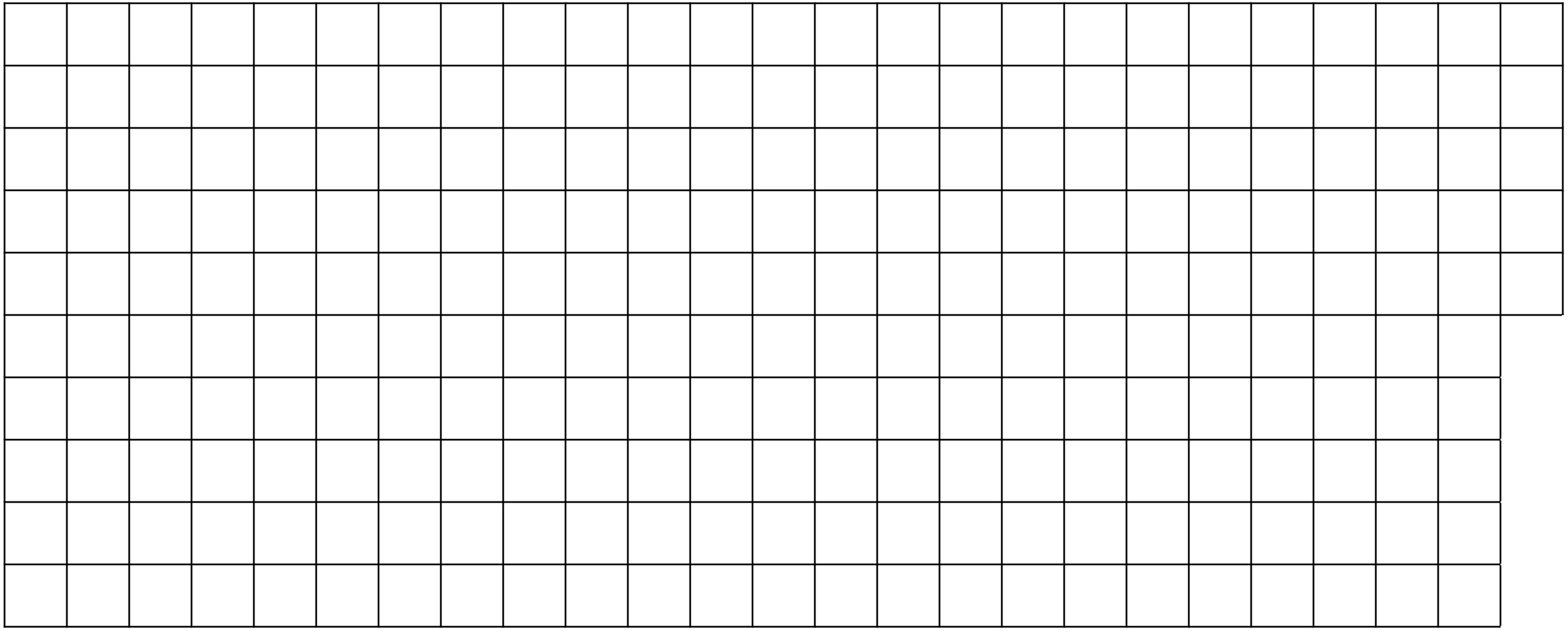
Bronson Yen, Gabe Samide, Jack Reynolds, Jack Roscoe, Jacob White, Jaxson Pahukula, Jimmy Hwang, Jimmy Hwang, Kevin Yu, Larry Xue, Lucas Tan, Neil Van Eikema Hommes, Nick Andry, Philip Frey, Sam Guber, Sebastian Toro, Vinh Pham Ngoc Thanh, vivan tiwari, William Boulton

Advised by: Santiago Torres-Arias

245

eCTF10
10 YEARS OF THE EMBEDDED CAPTURE THE FLAG

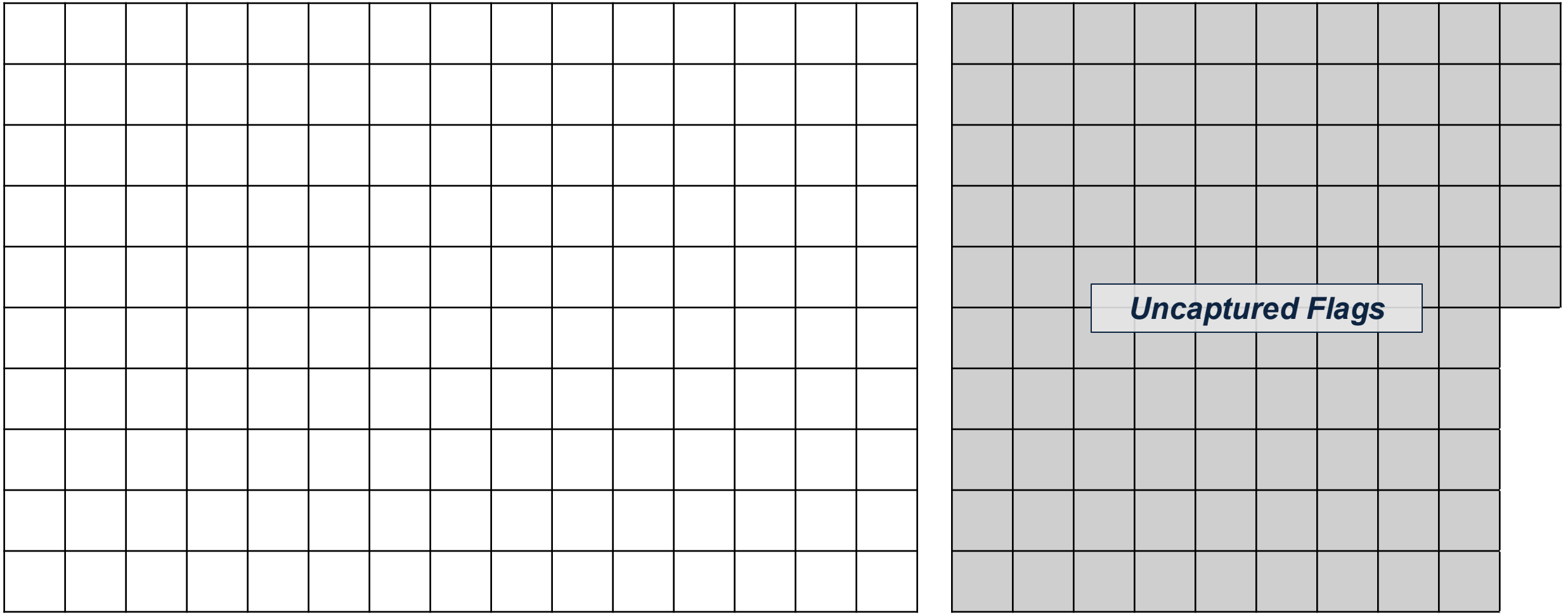
Flags in the Attack Phase



MITRE

95

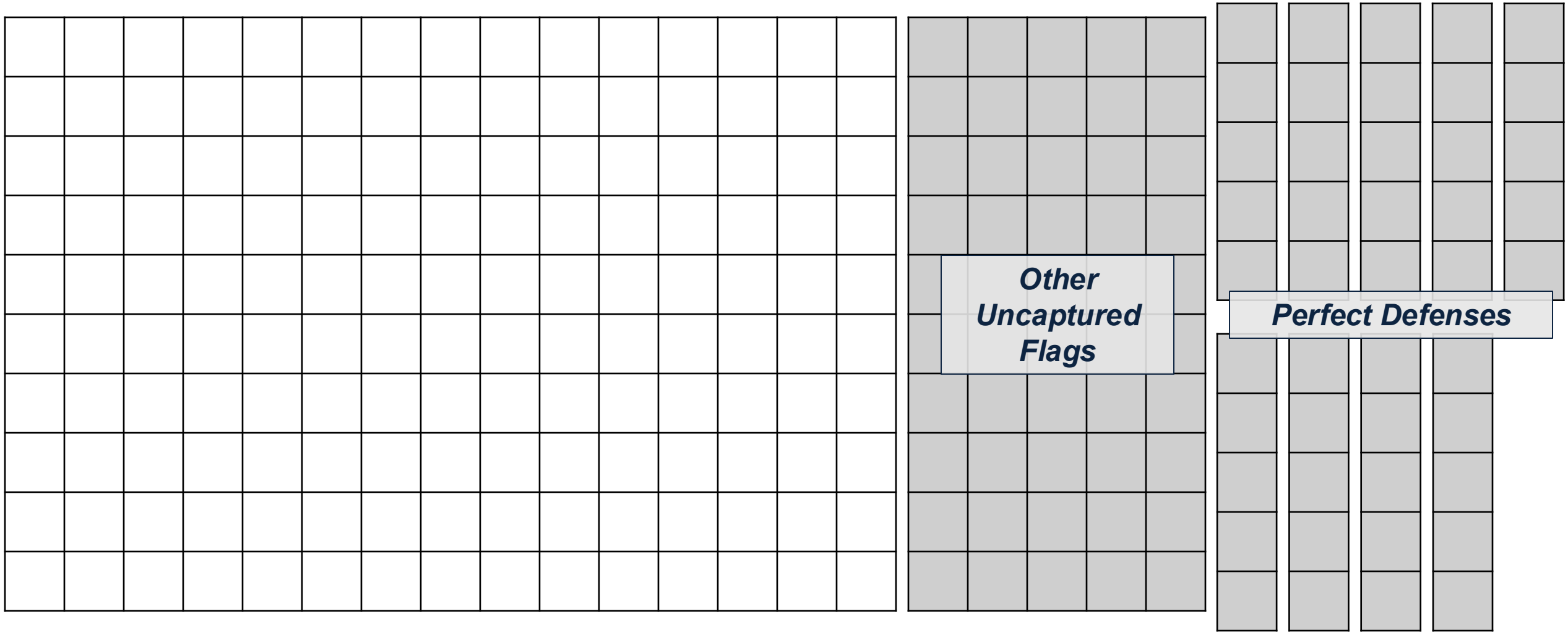
Flags went uncaptured



9

eCTF10
10 YEARS OF THE EMBEDDED CAPTURE THE FLAG

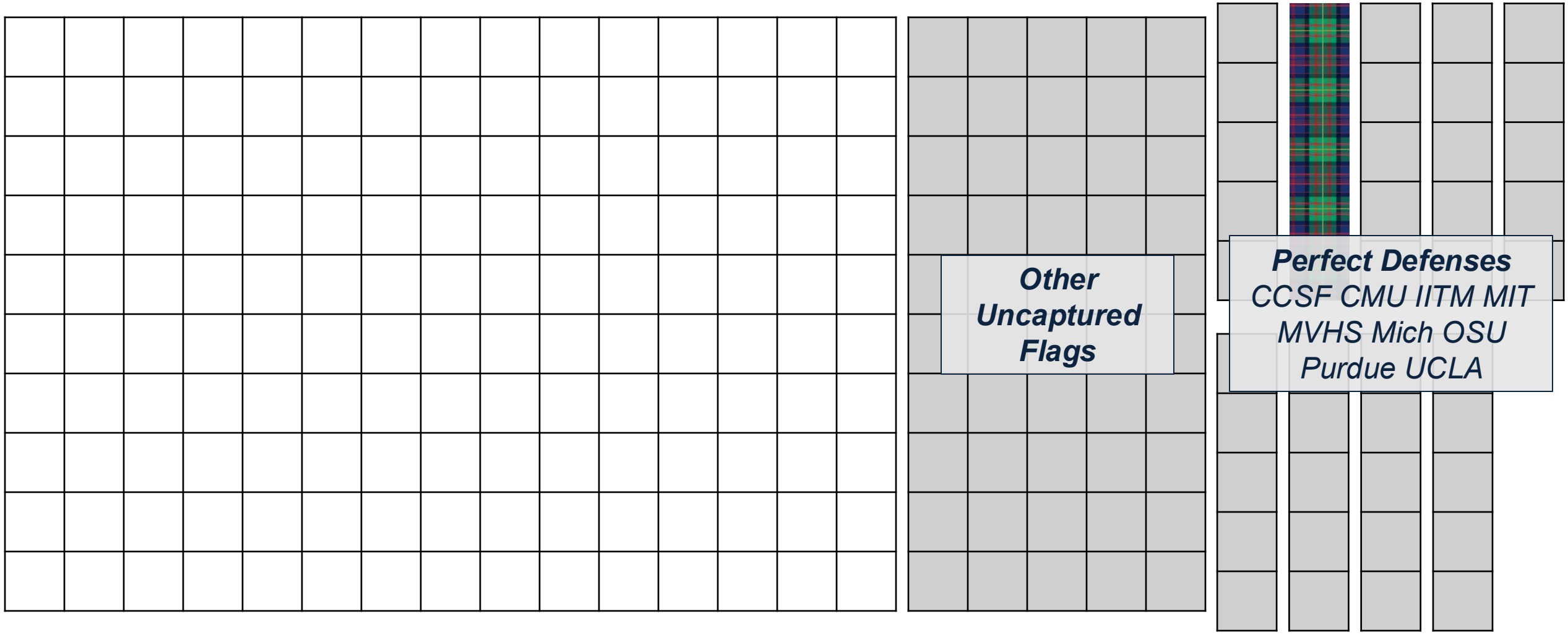
Teams defended all flags



MITRE

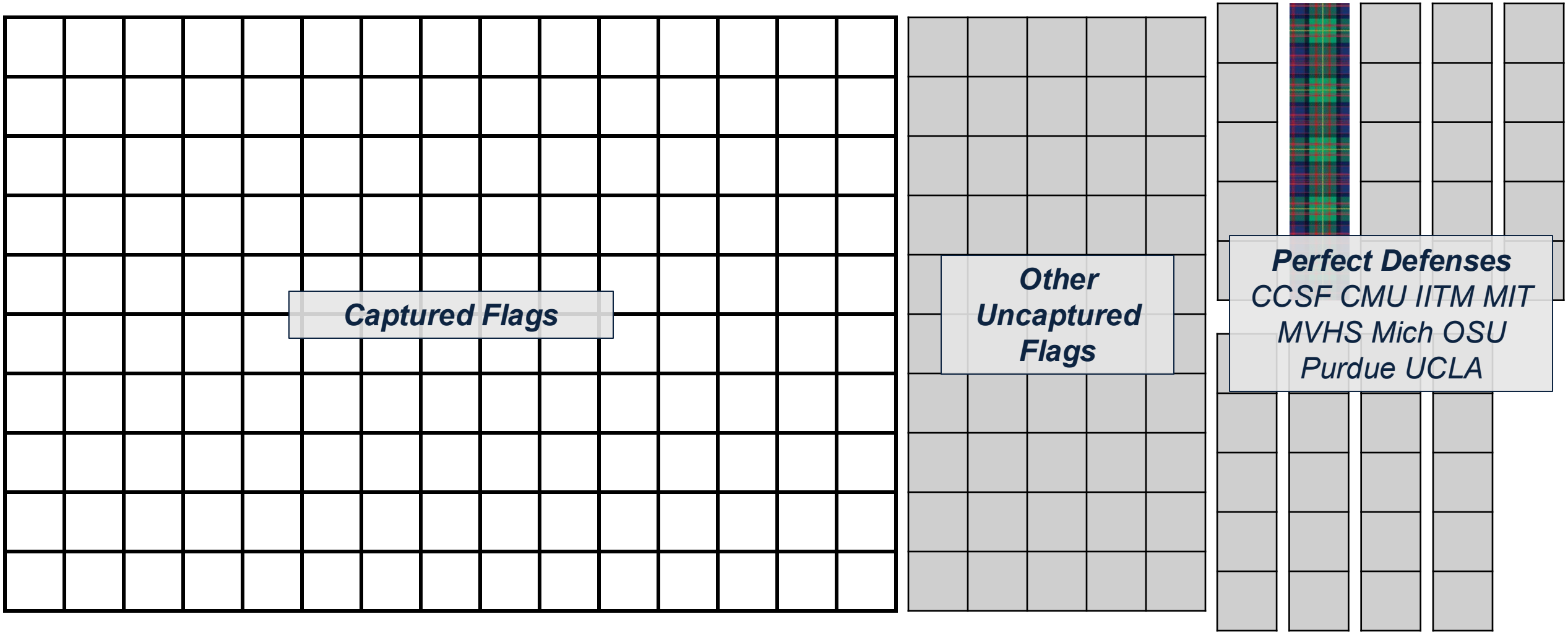
9

Teams defended all flags



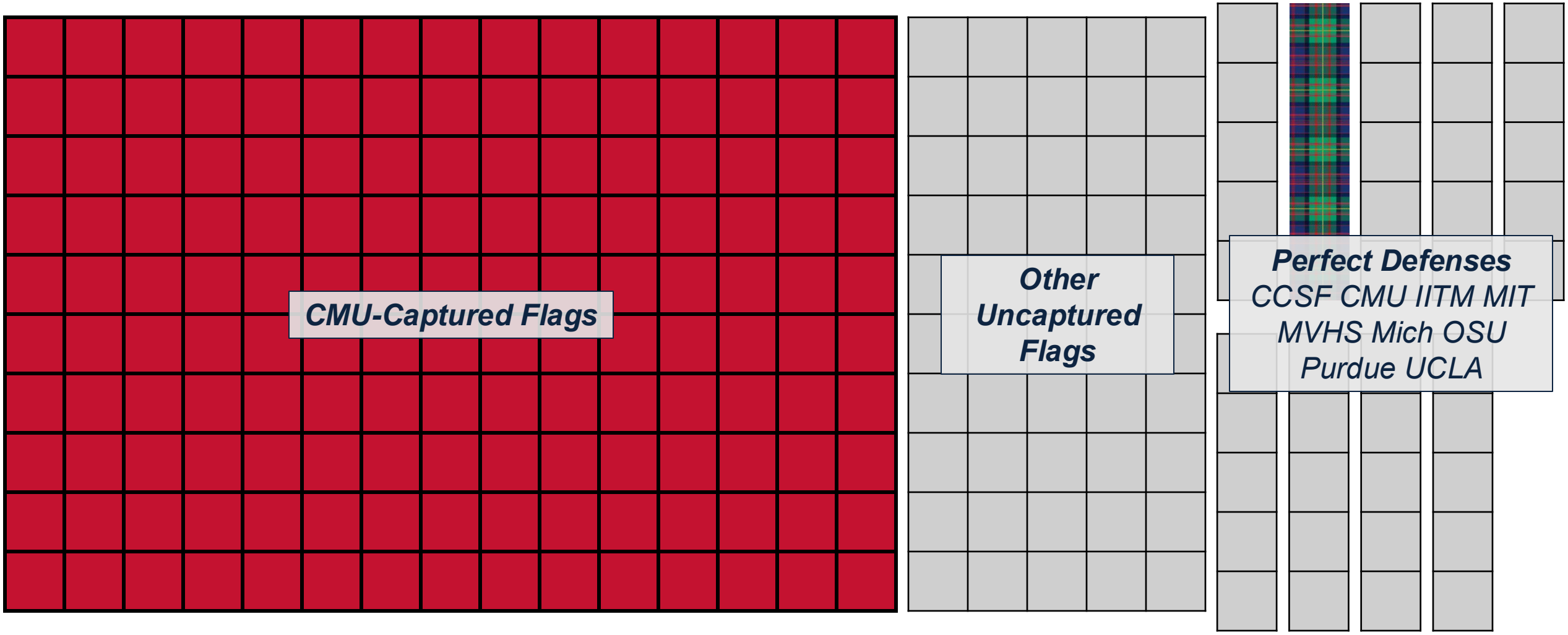
150

Captured flags



150

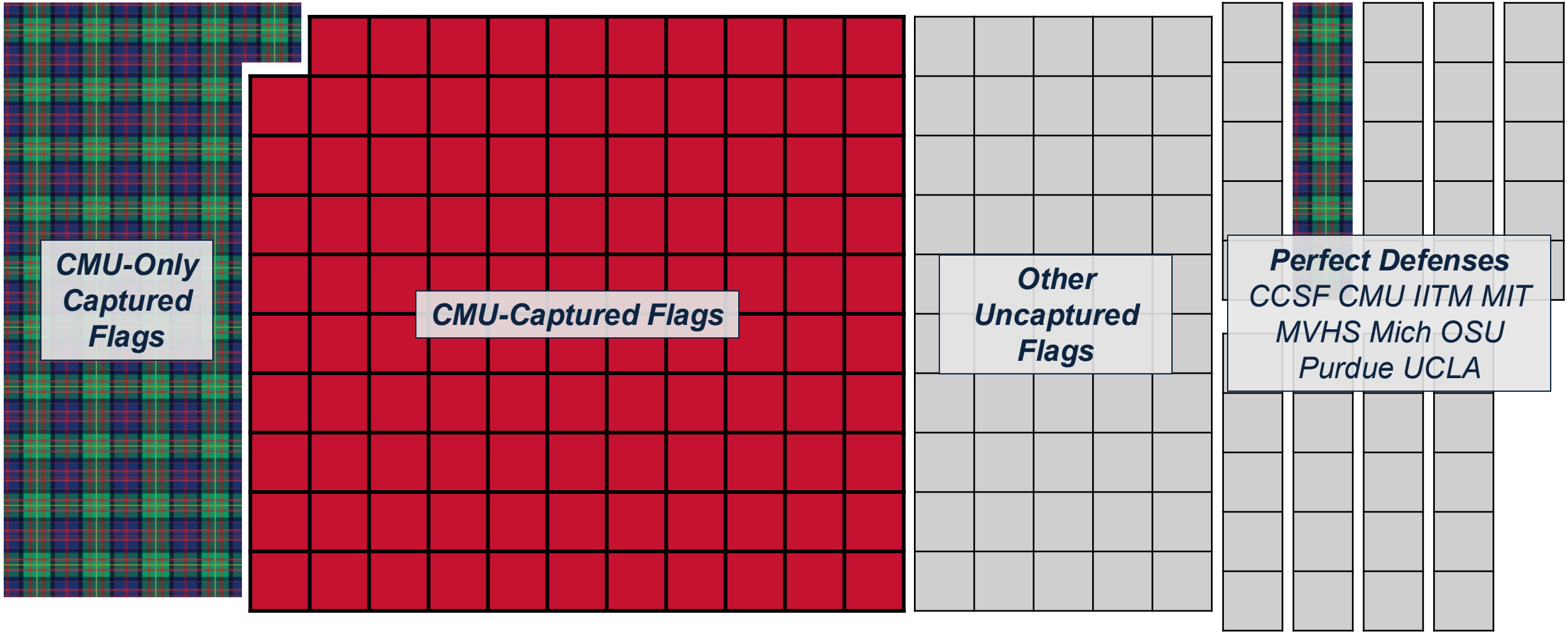
CMU-captured flags



41

eCTF10
10 YEARS OF THE EMBEDDED CAPTURE THE FLAG

Uniquely CMU-captured flags



MITRE

First Place

$10,000

First Place

Carnegie Mellon University

Plaid Parliament of Pwning

Akhil Harikumar, Carson Swoveland, Daniel Ha, Harrison LO, Janice He, Leonardo Mouta Pereira Pinheiro, Matin Sadeghian, Om Arora, Peiyu Lin, Rohil Chaudhry, Samuel Dinesh, Sky Bailey, Surya Togaru, Taha Biyikli,Tony Yan

Advised by: Anthony Rowe, Maverick Woo, Patrick Tague

Rank	School	Score	Position Change
1	Carnegie Mellon University	58,030	0
2	Purdue University	19,420	0
3	University of Michigan	19,112	1
4	City College of San Francisco	18,970	-1
5	University of Illinois Urbana-Champaign	17,710	0
6	The Ohio State University	16,611	2
7	Indian Institute of Technology Madras	16,394	-1
8	University of California, Los Angeles	16,332	-1
9	Mountain View High School	14,302	1
10	United States Air Force Academy	13,755	2
11	Massachusetts Institute of Technology	13,657	0
12	University of South Alabama	13,642	-3
13	University of Trento	13,428	0
14	Brigham Young University	12,173	0
15	University of California, Santa Cruz	10,350	1
16	University of Connecticut	10,215	1
17	Baldwin Wallace University	10,068	-2
18	Northeastern University	8,715	2
19	University of Washington	8,544	0
20	Georgia Institute of Technology	8,377	-2
21	Michigan State University	8,175	0
22	Johns Hopkins University	7,471	0
23	Purdue University	7,087	0
24	Parkway Spark!	6,677	0
25	New York institute of Technology	5,753	0
26	Ecole Royale de l'Air	5,635	1

Rank	School	Score	Position Change
26	Binghamton University	5,635	0
28	Colombe Academy of Technology	5,417	0
29	Tufts University	5,167	2
31	Flinders University	4,791	7
30	University of Colorado, Colorado Springs	4,875	0
33	Tennessee Tech University	4,622	0
32	University of Massachusetts Amherst	4,673	-3
34	University of California, Irvine	4,551	-2
35	University of Nebraska Omaha	4,374	0
36	Morgan State University	3,803	0
39	Indian Institute of Technology Dharwad	3,730	10
38	New York University	3,765	-4
37	East Tennessee State University	3,794	2
40	Georgia Institute of Technology	3,555	0
41	Southeast Missouri State University	3,516	-4
43	Florida International University	3,081	1
44	University of New Hampshire	3,059	1
42	United States Coast Guard Academy	3,142	1
45	Northeastern University	2,920	2
48	University of Central Florida	2,841	-7
47	Oklahoma Christian University	2,868	3
45	The University of Texas at El Paso	2,920	-3
49	CyberAegis	2,163	-3
50	Rensselaer Polytechnic Institute	2,126	-2
51	Pace University	1,149	18
52	SDU University	974	2

Dan Walters

Department Manager
Electronic Systems Security
MITRE

Stay Connected

eCTF10
10 YEARS OF THE EMBEDDED CAPTURE THE FLAG

eCTF Alumni Page

MITRE

Winning Teams

Please stay in your seats after the ceremony for photos!

eCTF10

10 YEARS OF THE EMBEDDED CAPTURE THE FLAG

THANK YOU!

Join us in the atrium for light refreshments

SURVEY AND
SWAG RAFFLE

MITRE | SOLVING PROBLEMS FOR A SAFER WORLD®