b01lers

Purdue University

Jacob White
Jack Roscoe
Gabriel Samide
Jihun Hwang (Jimmy)
Kevin Yu
Phillip Frey

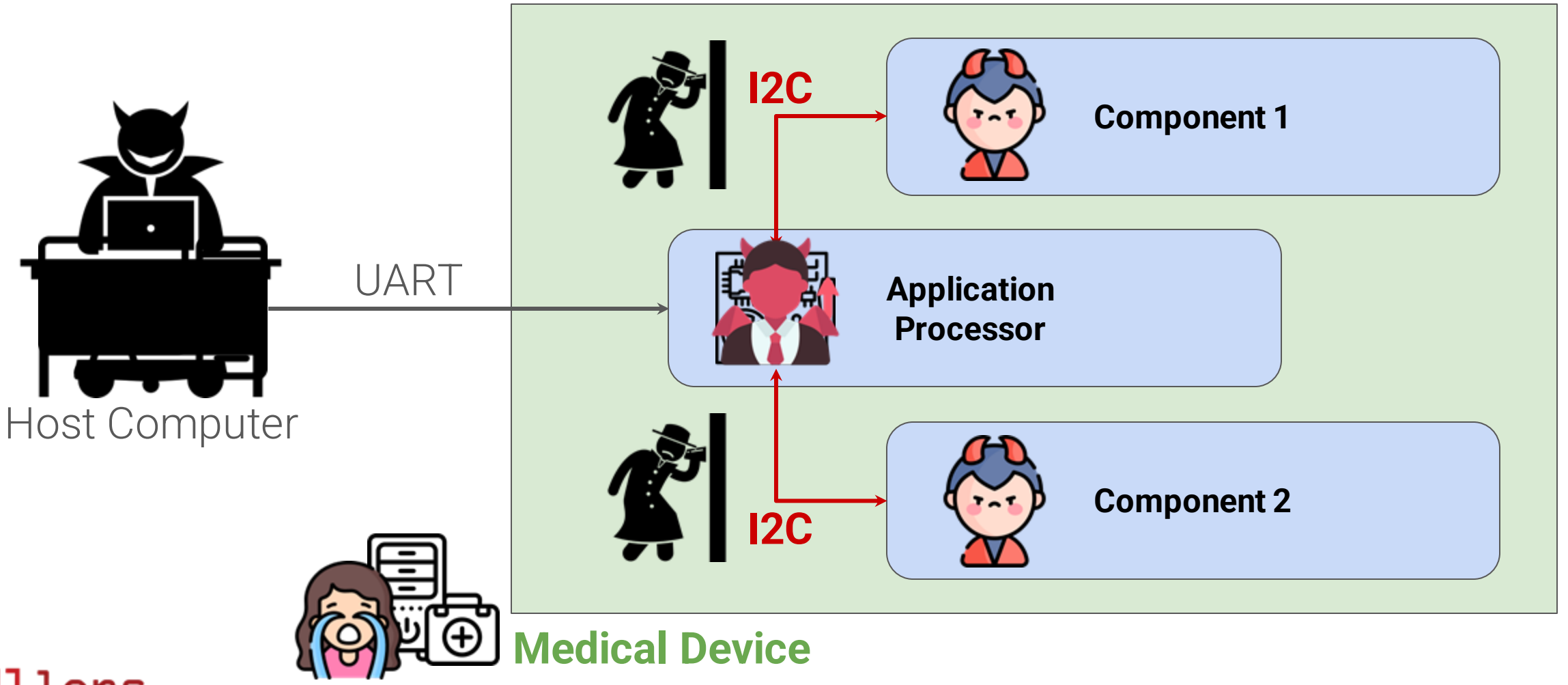
b01lers… Assemble!

Nick Andry, Philip Frey, Jorge Hernandez, Neil Van Eikema Hommes, Jihun Hwang, Siddharth Muralee, Jaxson Pahukula, Mihir Patil, Adrian Persaud, Jack Roscoe, Gabriel Samide, Lucas Tan, Vinh Pham Ngoc Thanh, Vivan Tiwari, Jacob White, Susan Wu, Kevin Yu

Advised By: Professors Christina Garman, Kazem Taram, Santiago Torres-Arias

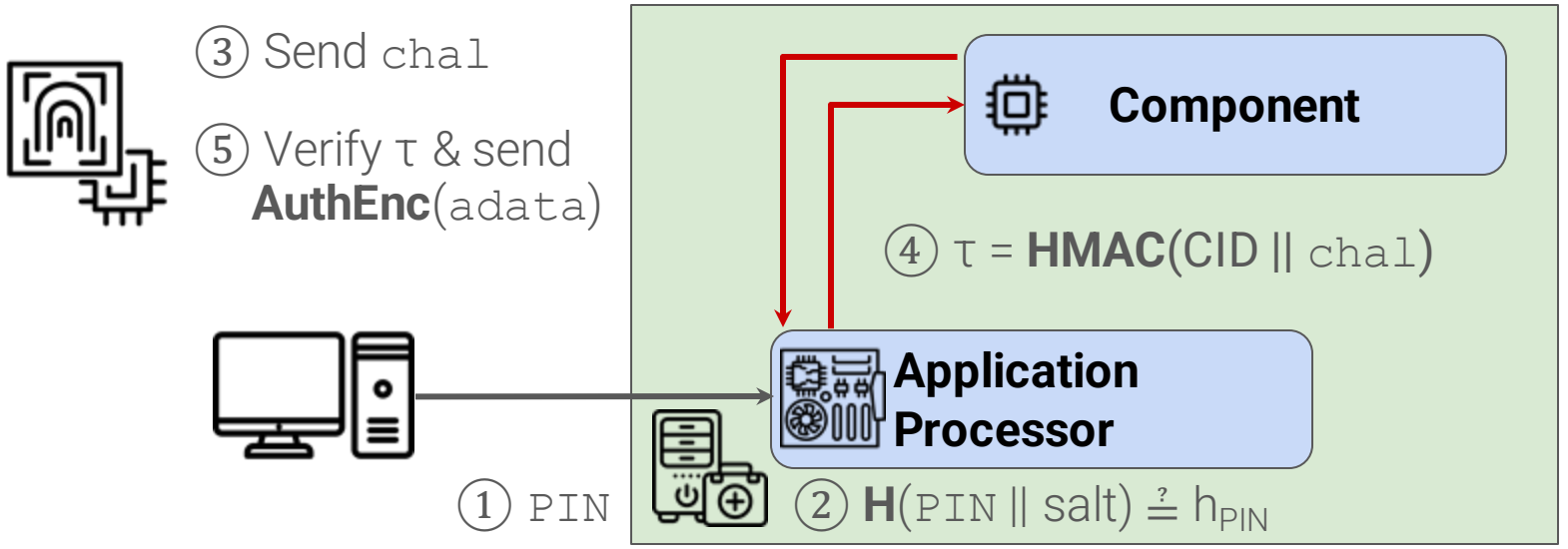
Design Phase

Design Goals & Overview



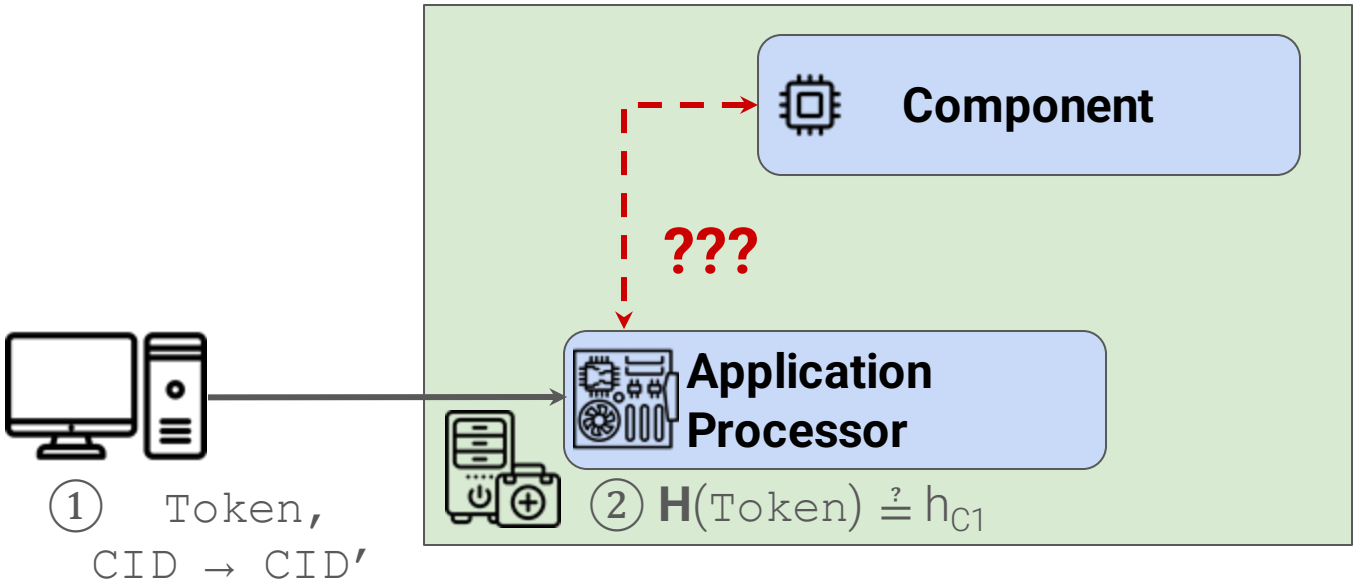
Attest Protocol

- Hash & Check PIN (**`0x000000–0xFFFFFF`**) like a password.
- Component challenges the AP to respond with expected ID.
- Component encrypts attest data for valid AP to decrypt.



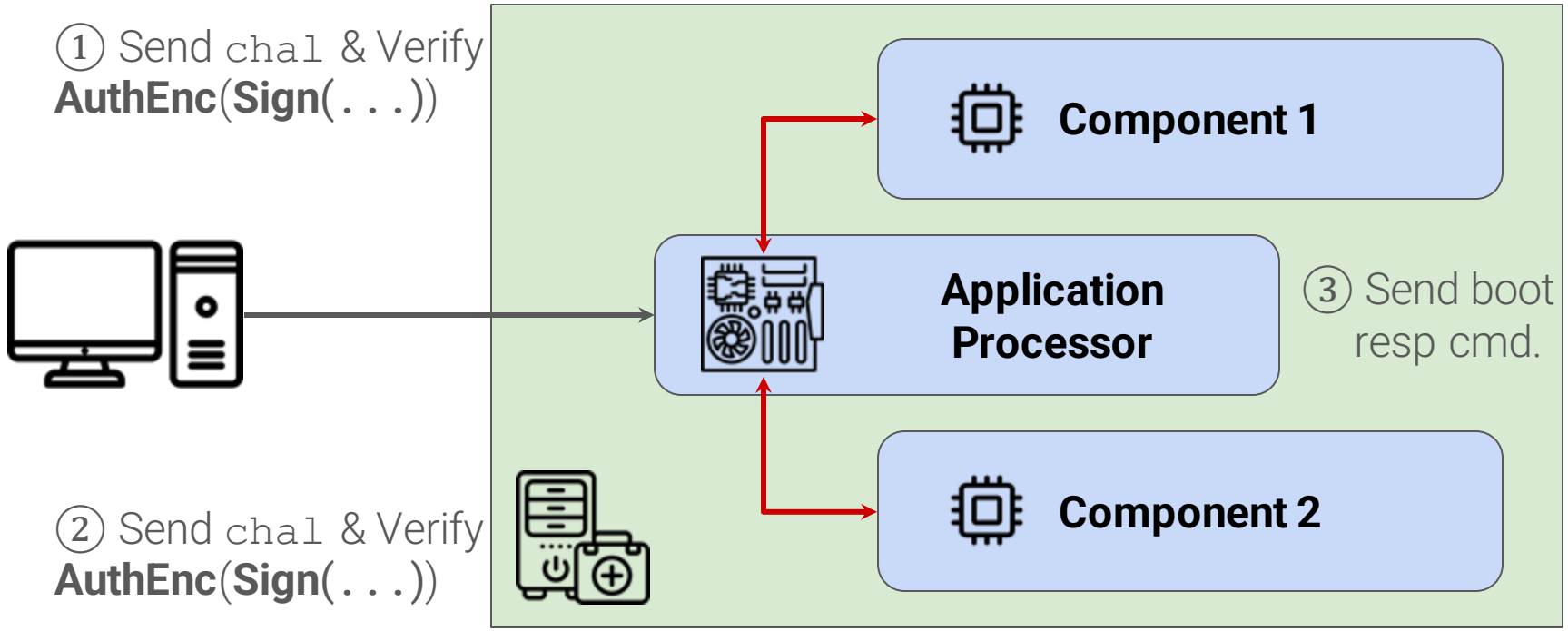
Replace Protocol

- Hash & Check **Replacement Token (16 B!)** like a password…



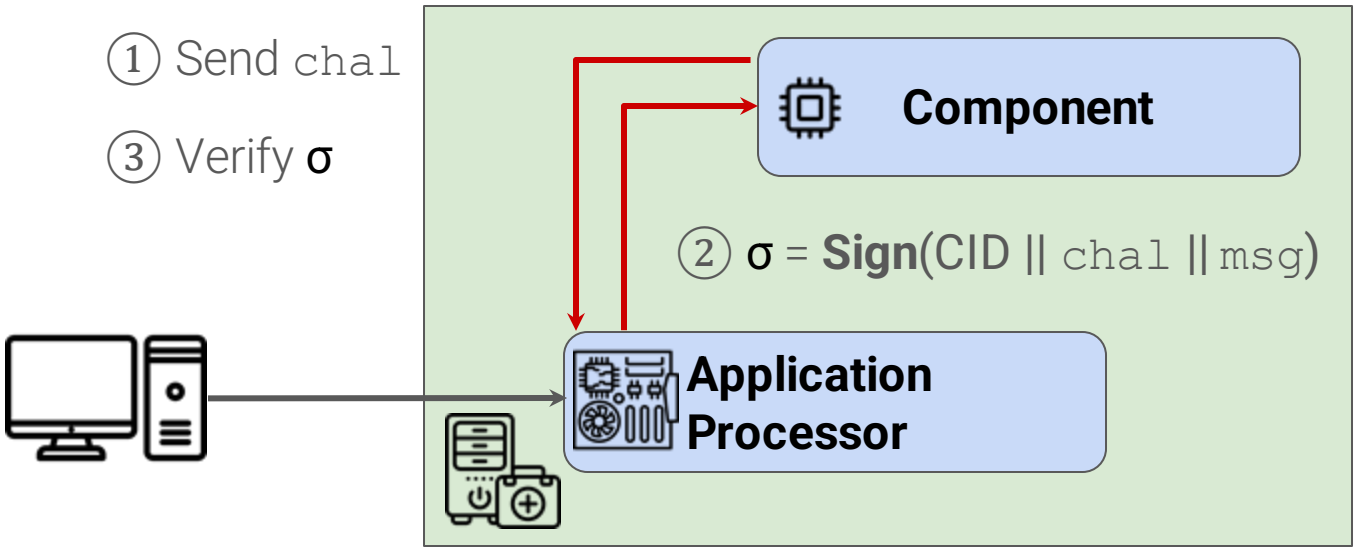
Boot Protocol

- Components and AP challenge each other to check who they say they are.
- AP and Components wait to verify each other and respond before booting.



Messaging Protocol

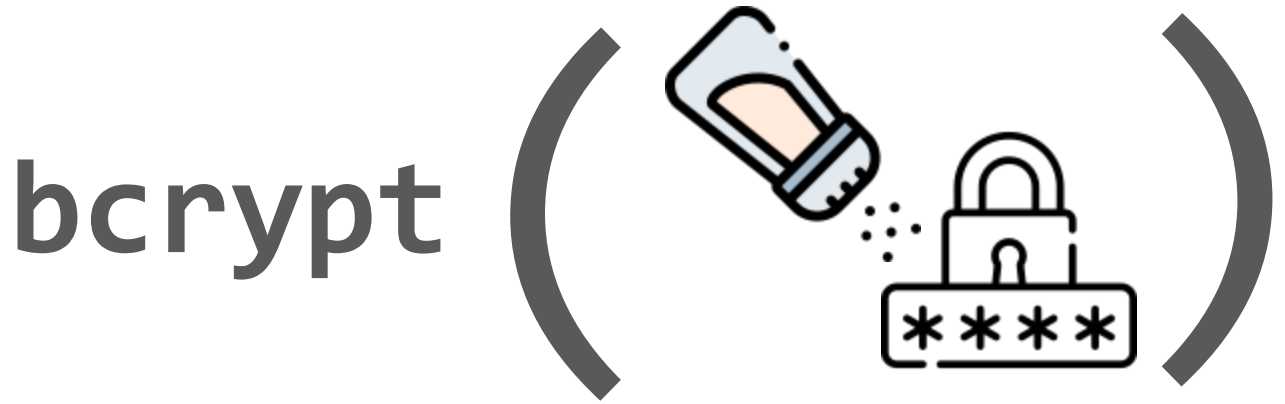
- Generates random challenge to include with signed message.
- Signatures verify the specific identity of Component or AP's messages.
  - Actuator shouldn't pretend to be an insulin pump.
  - Components shouldn't pretend to be AP.



Security Features

Passwords (PINs)

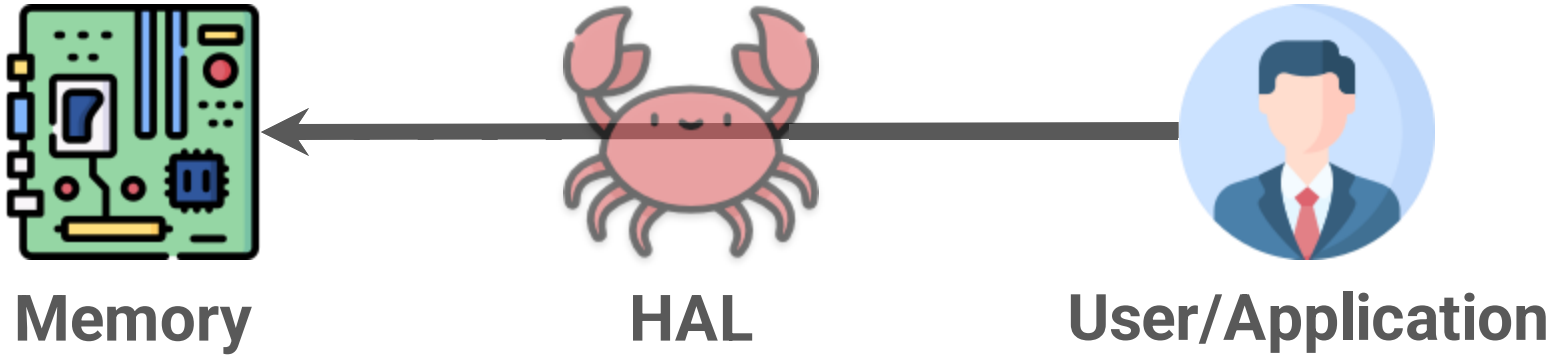
- Salted and hashed with a “slow” hash function (**`bcrypt`**).
- Defends against both offline and online dictionary attacks.



Security Features (Cont.)

Hardware Abstraction Layer (HAL)

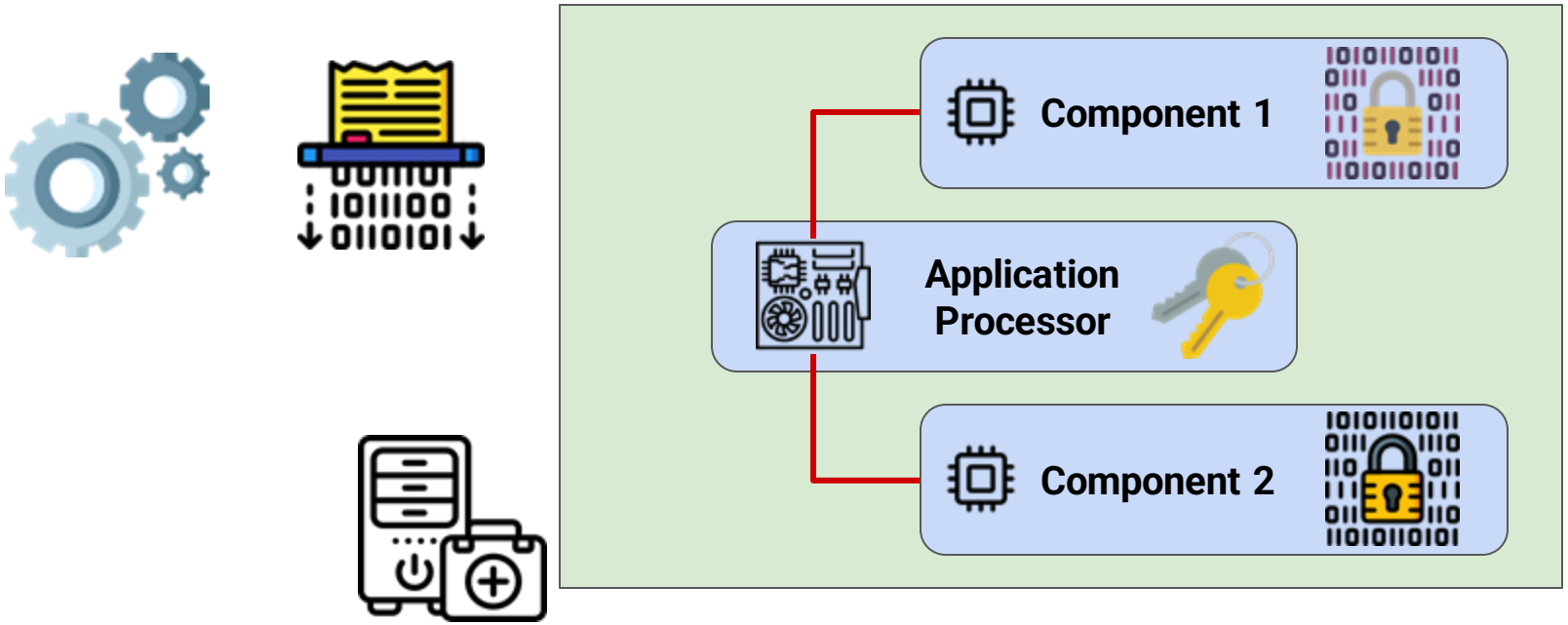
- Re-wrote firmware in Rust.
- Memory safe by default!



Security Features (Cont.)

Build-time Encrypted Data

- Attestation Key is only stored on the AP, not on Components.
- Attestation Data is not directly stored anywhere inside.



Security Features (Cont.)

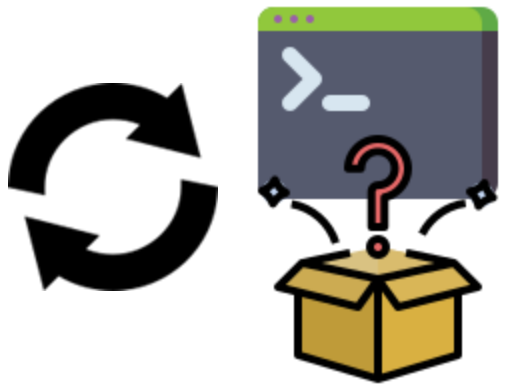
Compile-time address randomization (ASLR)

- Randomized memory section offsets.
- Frustrates buffer overflow or similar attacks.

Security Features (Cont.)

Random Delays and Repeated Checks

- Protection against fault injection, timing and glitching attacks, etc.



Lorem ipsum dignis eium fugit aspelluptat eati odis net exeri rectem lia venihil icipsapid qui dolupient quam aceatemque repedi tem lantiae provid quia sitatia temquibuste voluptatquo comnit fugiat invenit fugia quo ditias ipitatat erspel id utesecuptur solorio. Hari veria dis nis et millic tota comnimet inctor sum aut laboressedis deribus illore non et fugitiosam, soluptat in eatur? Endignatem el et ex endicim re occullu ptatem laut audipita num fugit adis delenti cum sus aut iduntur arumqui blam eos molorum quissimint, nis ut aut adisci odic te as everspe ditati accum alit rempel iumque nobis repudamus.

Epro que pra nis atempor ad quis am sim explitem sum et, illaute mposape lestiores rehenis ante senducidus quatet doluptatur sumquia ntiore, et idus volutem voloreria volorem fugit ento blabore pro et, od quam sam re volori aut etur seque velignia voloreic tem es nosam volo magnam que ad que voles et voloribeatem rati ipsae dolentiatur rempore sequunt, quis ea volupti am secuptaquate cus etum eliqui dolorep erferrum diore quibus re omnisci aditibusda qui aut alit, ide et possequias nonsed est int eos il earciis sit aliquidis eum quiam, quam doluptatis vel eum nos pos dolorat emoditiur? Quid que pe veratiae pero molum volorehendit re, unt pa doluptio cus.

Ga. Nam, ommodic aborrorepe plabo. Neque quaestiat hiciae opta inis dolestrum cum, quam ut fugiam, od mosamus aruptaspel modi rerunt paris a volore pressed ut voluptatius voluptae provid quis volo eumquunte exerumq uaernatia serum cus estis autemquis exeris et ut omni consequis essi to qui ne destoreri sinctes placcae aut labo. Ritia dolestiatur autet omnitatus aliciae repudam, sanditatiis doluptatur? Qui apid ut officabo. Et essi nem et debis re dolorio ssinctatem quodipsuntem explique doluptaqui officidempor ad ut volut la doluptam re dolute eatur sequam exerspero explacc ulliquiatio vendustrum etus sed que desseque sum quos voluptatiame core cuptaquibus exerrum et molupta ssiment ressundantem idiciet omnimus re ni berunt prero veliquis non cus.

Aliquibusdae con prernatur aut res dolora aut aut quam comni tem sequo ommoditatet hiliqua tusciae pedit, atures doloritius dolutat emperis et imus dolupta tentibe rspienis niminus ni videbitatem quos re, temquis untibus undaeribus eum faccus, tem erum nem. Entorum illectus del eatur?

Faccuptatur, se nostrum et fugia peria nossunt fuga. Sedicab orectissitae voles evel ipsus ipsae et, sam vit, conesequе sit et aspe quo quas se quiandusdae solorumquae vel enimus verumendunt.

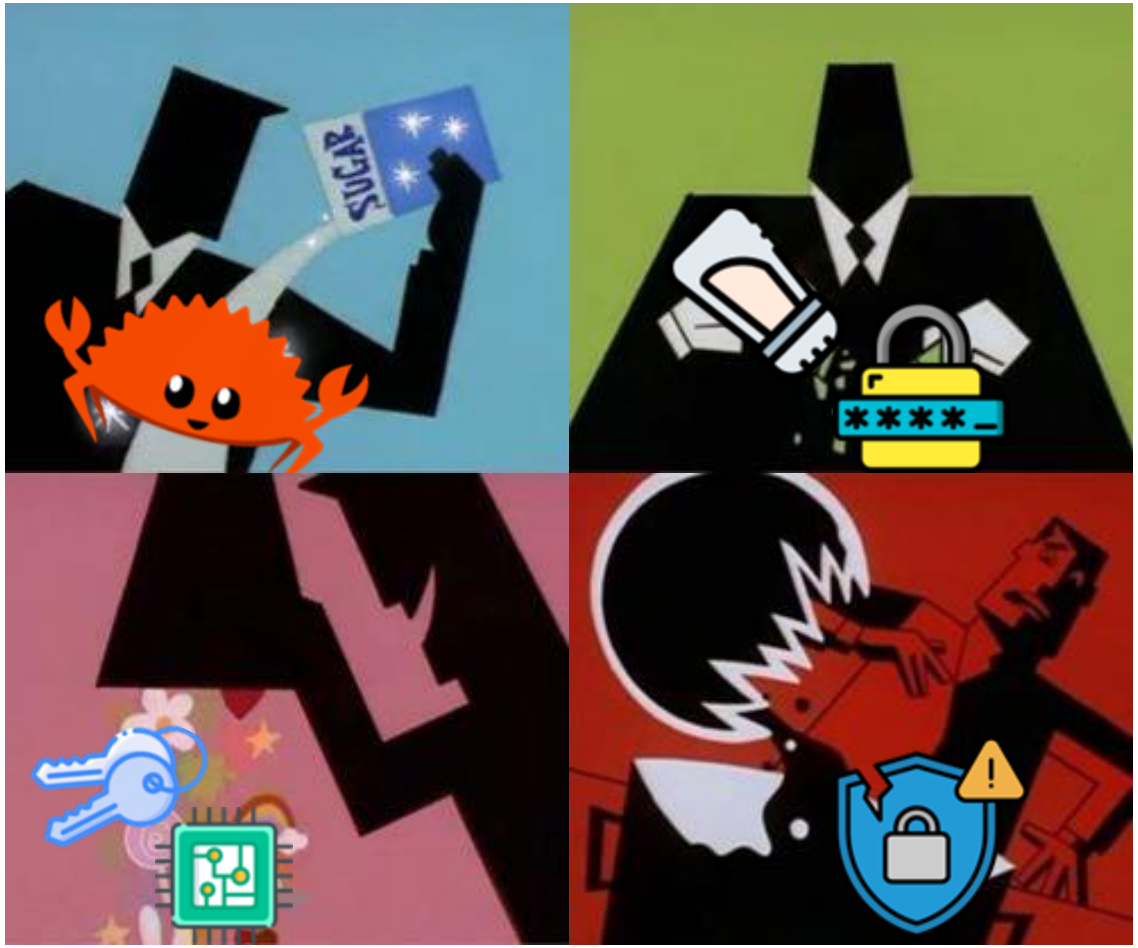
Ab into dolora nossequi tem num quunt maxim del intio. Harumenda ne perovit quiant ut alique doluptatia ium sunt volut magnatiis voluptis es enim non earuptae eos doluptatur solorer iorem. Ur rem que dellores nimpos pra sequasped qui quiatqui conem reiunt.

As debit eatiber itatur, aut lat parchilicae pa pellentum ut ident eatia commolo eum nectibusa apel istruntiore idebit eturiatur?

Lupieniendis aut volorerio. Daectot aectestium latem volenimus ut velis alibus ulliciis aceaquam derovid elitatet que vel minctume iusam dolupti venis am fugit etus vellit re viducid quiatquiant volestisti torehen tiorestem antis militas nes del ilicaturibus et et est harum, ipsant, natem quos es ipsa velit est, es re volupta temolorum este explant.

Pore vid est, audam facia voluptiae pos ut que nulloria core nihilita istio tem quiscia volore nulliae corpos eatur

Design Summary



Potential Improvements

- Require secret sharing from all components & AP to boot.
- Two way C-R and sequence numbers for post-boot.
- Activate the board's Memory Protection Unit (MPU).
- Use stronger memory-hard hashes (e.g. Argon2id) [1] for Attest PIN.

[1] Jeremiah Blocki, Benjamin Harsha, and Samson Zhou. "On the economics of offline password cracking." 2018 IEEE Symposium on Security and Privacy (SP). IEEE, 2018.

Attack Phase

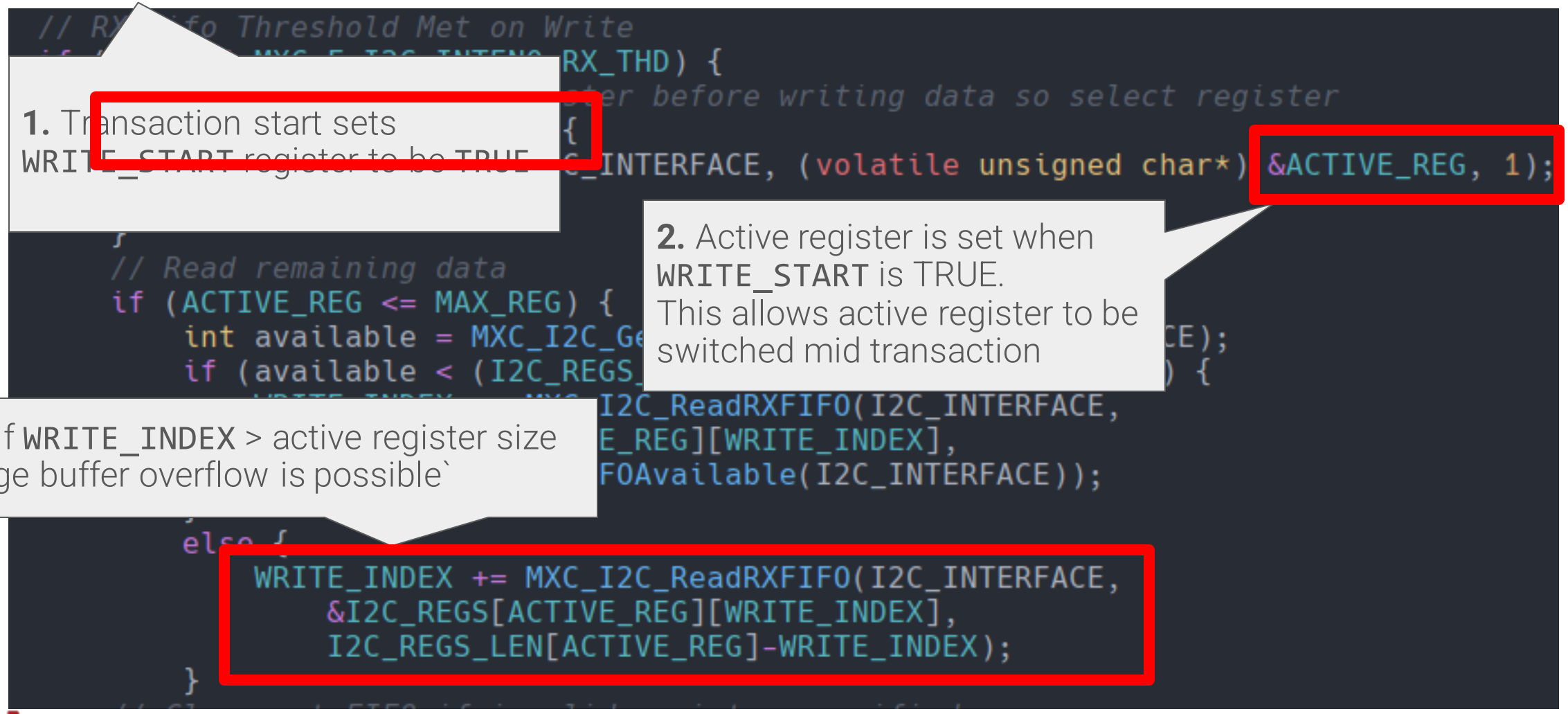
I2C Interrupt Handler Buffer Overflow

The interrupt handler did not properly handle repeated restart.

- Only checks if the stop flag is set for executing end of transaction code.
  - … which is only set when I2C stop code transmitted!
- A repeated restart message can start a transaction without sending an I2C stop code.

```
// Transaction over interrupt
if (Flags & MXC_F_I2C_INTFL0_STOP) {
```

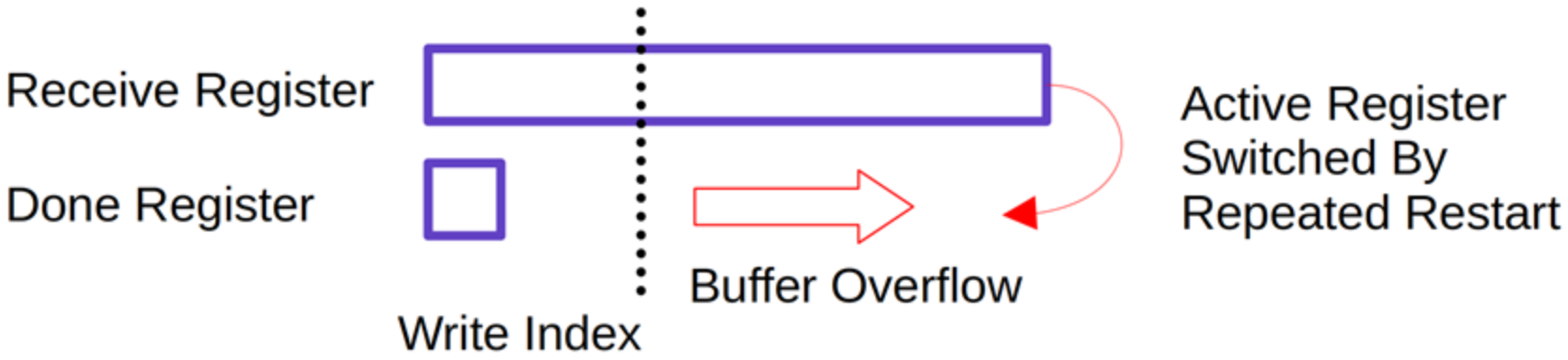
I2C Exploit (Cont.) Transaction Threshold



I2C Exploit Details

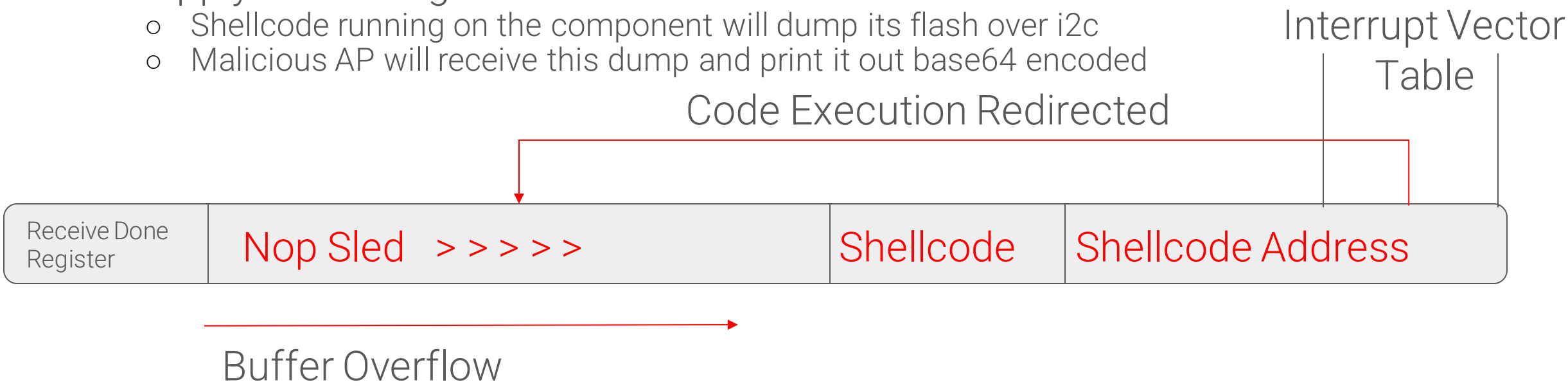
Buffer overflow -> Memory Corruption

- Write bytes, switch from a large register to a small register with repeated restart.
- **`WRITE_INDEX` will be out of bounds!**
- A large buffer overflow occurs, and bytes can be written in from I2C in many repeated restarts
  - Pointer in interrupt vector table overwritten to jump to shellcode



How We Used I2C Exploit

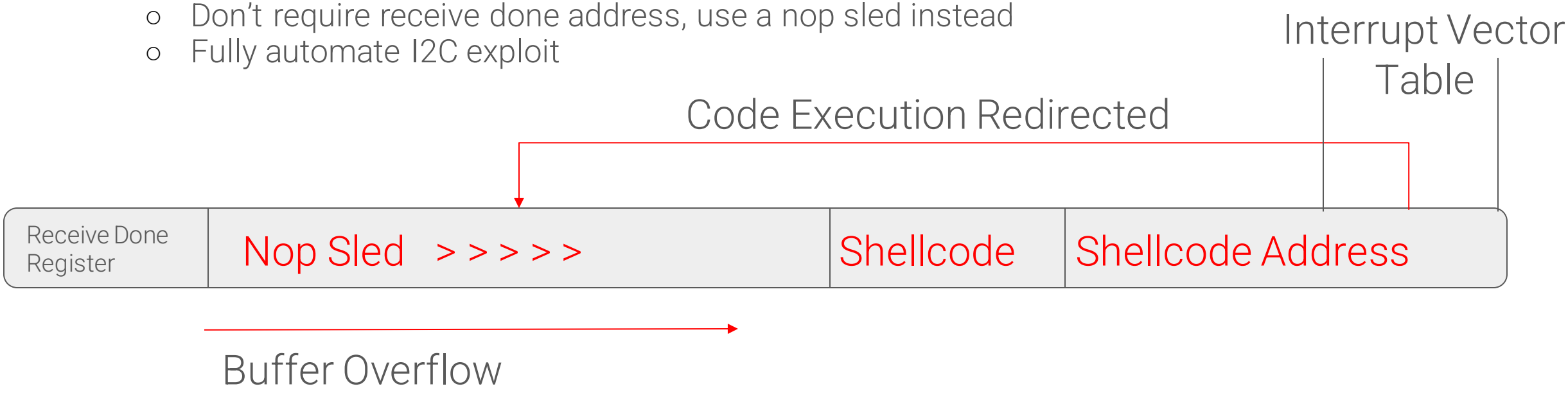
- Attacks with Physical Access
  - Python scripts communicate with malicious AP and shellcode on component to retrieve flash dump
  - Flags and keys can be retrieved from flash dump
- Supply Chain Flags
  - Shellcode running on the component will dump its flash over i2c
  - Malicious AP will receive this dump and print it out base64 encoded



Potential Improvements to I2C Exploit

- Other teams were aware of the exploit, so it was a race to get first bloods
  - We could manually do the easier 4 in physical access flags in about 3-4 minutes, but other teams automated systems could do it in about one minute

- Improvements
  - Don't require receive done address, use a nop sled instead
  - Fully automate I2C exploit



b0llers

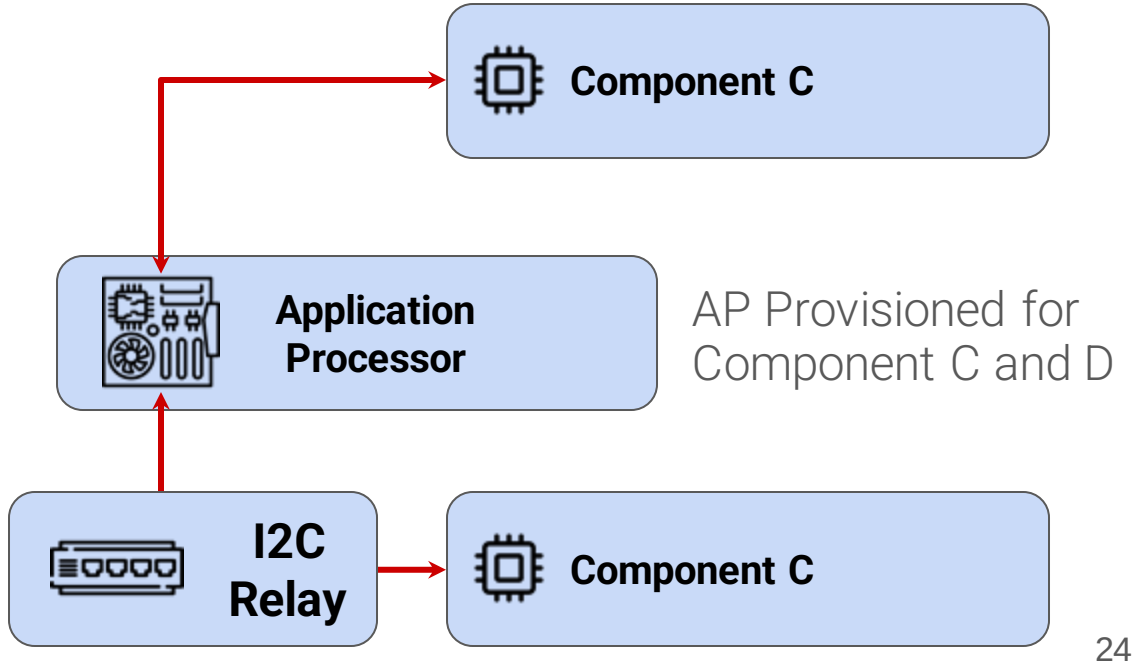
Attack Impacts and Countermeasures

- Potential Impact
  - Arbitrary code execution resulting in numerous potential attacks
    - Exfiltrating attestation data, boot messages, and secret keys stored in the flash, modifying vital hardware registers, and manipulating intended functionality

- Suggested Countermeasures
  - Reset the read and write indexes are reset even after a repeated restart
  - Ensure out of bounds writes are not possible
  - Redesign the interrupt handler

Exploiting Protocol Flaws

- Static token to authenticate AP and components
  - Some teams had a secret token used to authenticate the AP sent in plaintext
  - The token could be received by a malicious component, then replayed by a malicious AP
- Boot authentication does not include component ID
  - e.g. in challenge response authentication for boot



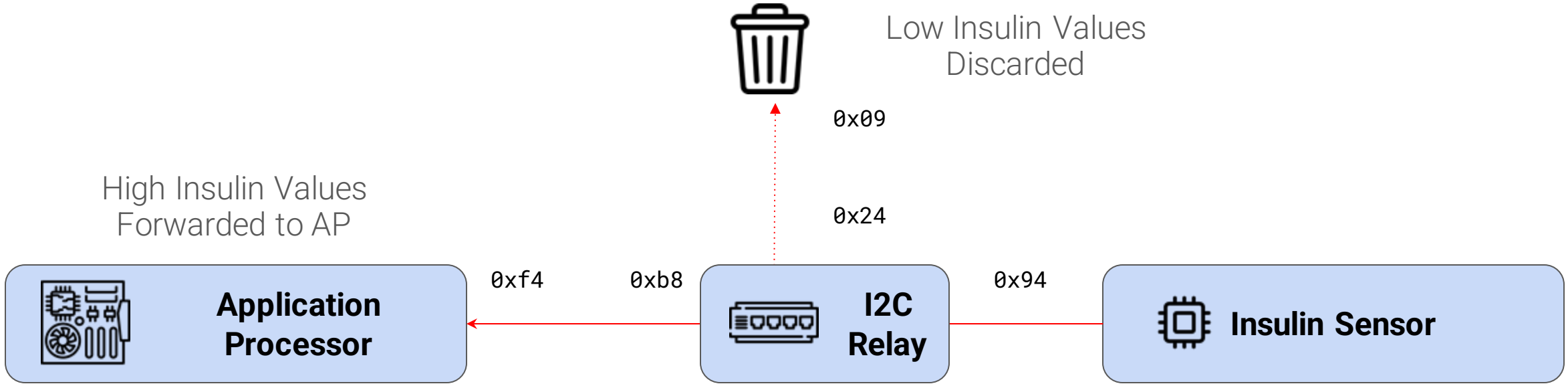
Protocol Flaws Countermeasures

- Potential Impacts
  - Attacker can maliciously impersonate devices and perform operations only authorized devices should be able to perform
  - Includes operations such as booting MISC system and querying attestation data without the need for authentication

- Countermeasures
  - Utilize challenge response instead of just a fixed token
  - Challenge response should include the component ids in some form

Other Attacks

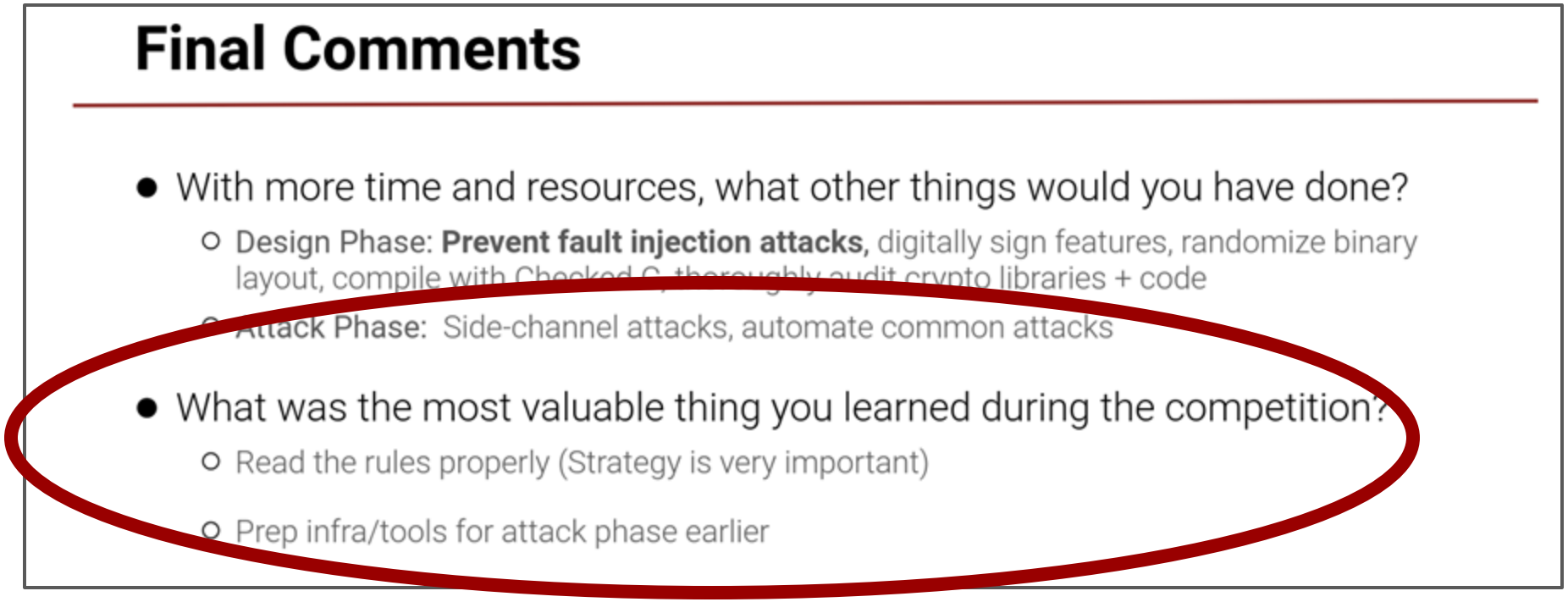
- We investigated possibility of dropping packets in operational pump swap
  - Drop the low insulin reading packets, and only forward the high packets
    - Several teams (us included) did not fully ensure post boot packets arrived in order
  - This ended up not working due to the way post boot messaging was implemented



What We Learned

- Focus on getting infra set up
  - Having to pass around boards manually, and not having a way for everyone to work on development / exploits at the same time hindered us a lot
- Read rules more carefully
  - We had to redesign our replace / boot protocol near the end of design phase because we didn't realise that AP cannot talk to component during replace
  - We didn't realise until the very end that all flags were stored in attest data and boot message, so we can do extra things like encrypting all of them at build time and use some sort of secret sharing scheme to recover decryption keys

Seems Familiar…



Final Comments

- With more time and resources, what other things would you have done?
  - Design Phase: **Prevent fault injection attacks,** digitally sign features, randomize binary layout, compile with Checked C, thoroughly audit crypto libraries + code
  - Attack Phase: Side-channel attacks, automate common attacks
- What was the most valuable thing you learned during the competition?
  - Read the rules properly (Strategy is very important)
  - Prep infra/tools for attack phase earlier

Source: Purdue eCTF 2023 slides

Final Comments

- Improvements to be made:
  - Quickly establish threat model and outline of protocol implementation
  - Spin up (fully…) functioning development and attack infra
  - Immediately start Rusty development - What even is an MSDK?

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!

Backup Slides