b01lers

Purdue University

Jacob White
Siddharth Muralee
Muhammad Ibrahim
Bo-Shiun Yen
Ashwin Nambiar
Abhishek Reddypalle

Advisors :
Dr. Antonio Bianchi
Dr. Aravind Machiry

Outline

- Design Overview
  - Design Philosophy
  - Threat Model
  - System Design
  - Improvements
- Attack Phase
  - Stack Leak
  - Weak PRNG
  - Common Attacks
- Final Comments and Lessons Learned

b01lers

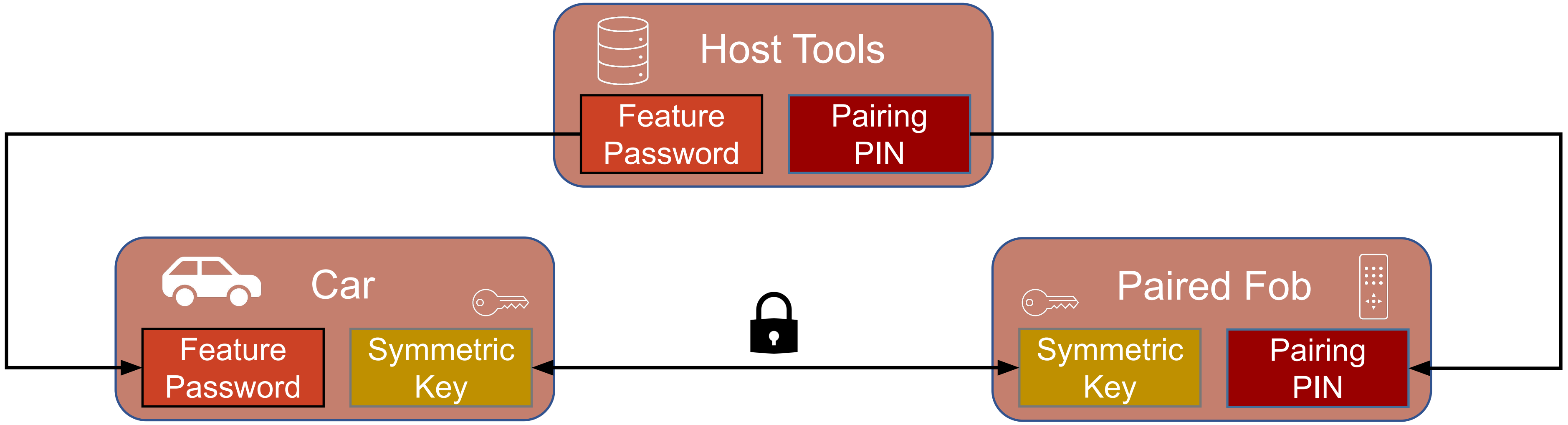
Design Phase

b01lers

Design Philosophy

- **Define a comprehensive threat model**, especially for buffer overflows and side-channels
- **Avoid over-engineering our protocols**, to reduce risk of introducing vulnerabilities
- **Limit the impact and scope of exploits**, even if compromise does occur

System Design



- **Replay attacks**
  - Secure PRNG + many entropy sources
  - Large random unlock challenge
  - Paired fob proves by decrypting with key
- **Buffer overflows**
  - memset uninitialized + unused data to 0
  - Known lengths for UART + processing
- **Side channels**
  - Use side-channel resilient cryptography
  - No secret-dep. computation or branching
  - EEPROM Layout Randomization
- **Brute force**
  - Salt PIN hash to prevent guessing if leaked
  - Persistent ~5 sec. timeout on any error

b01lers

Threat/Capability Matrix

Attacker Goal / Capability	Brute forcing pairing Pin	Unauthorized car unlock	Unauthorized car features	Unauthorized fob duplication
Access to car	No PIN on car	Symmetric keys on car/fob	Unique feature passwords	No PIN on car
Temporary fob access	Delay	Unique challenge-response	Unique feature passwords	Salt-then-hash pairing PIN
Access to car with features	No PIN on car	Symmetric keys on car/fob	Unique feature passwords	No PIN on car

Protocol Overview

Unlock Car

- Symmetric key AEAD Encryption using ASCON
- Randomized challenge-response by car to fob

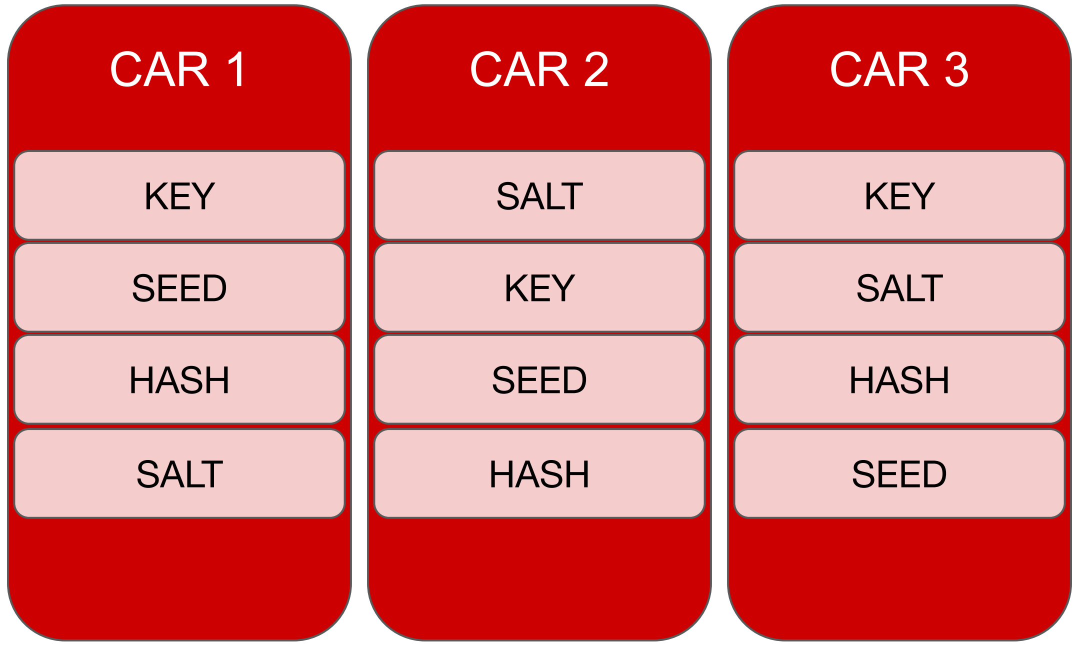
Pair Fob

- Salted and hashed 6-digit pairing PIN
- Persistent 4 sec. timeout on each PIN attempt

Enable Feature

- Unique 32-bit feature password for each car
- Salted and hashed feature stored on car

EEPROM Layout Randomization (ELR)



Our manufacturing process involves the creation of a randomized EEPROM layout for each car produced. This security measure ensures that any attacker who gains access to the EEPROM will be unable to discern the location and content of stored data without reversing the code.

Possible Improvements to Design

Binary Layout Randomization (Compile-Time). Modifying our defense strategy to encompass randomized layout for other sections, such as the .text and .stack, would have resulted in a more formidable challenge for teams seeking to attack our design.

Better PRNG entropy and implementation. We could have looked harder for an existing PRNG implementation instead of rolling our own. We could have improved entropy by sampling (smallest bit of) temperature and sourcing from many samples.

Mutual car-fob and fob-fob authentication. We didn't fully capture the impact of authenticating fobs in the protocols, or how AEAD encryption supports this on unlock. Using signatures or even HMACs would have made it harder to impersonate fobs.

Digitally sign features. Instead of relying on the uniqueness of feature passwords and minimal number of cars when authorizing each car's features, we could have digitally signed a unique feature ID.

Prevent fault injection attacks. Verify each of the conditional checks multiple times to prevent any possibility of glitching.

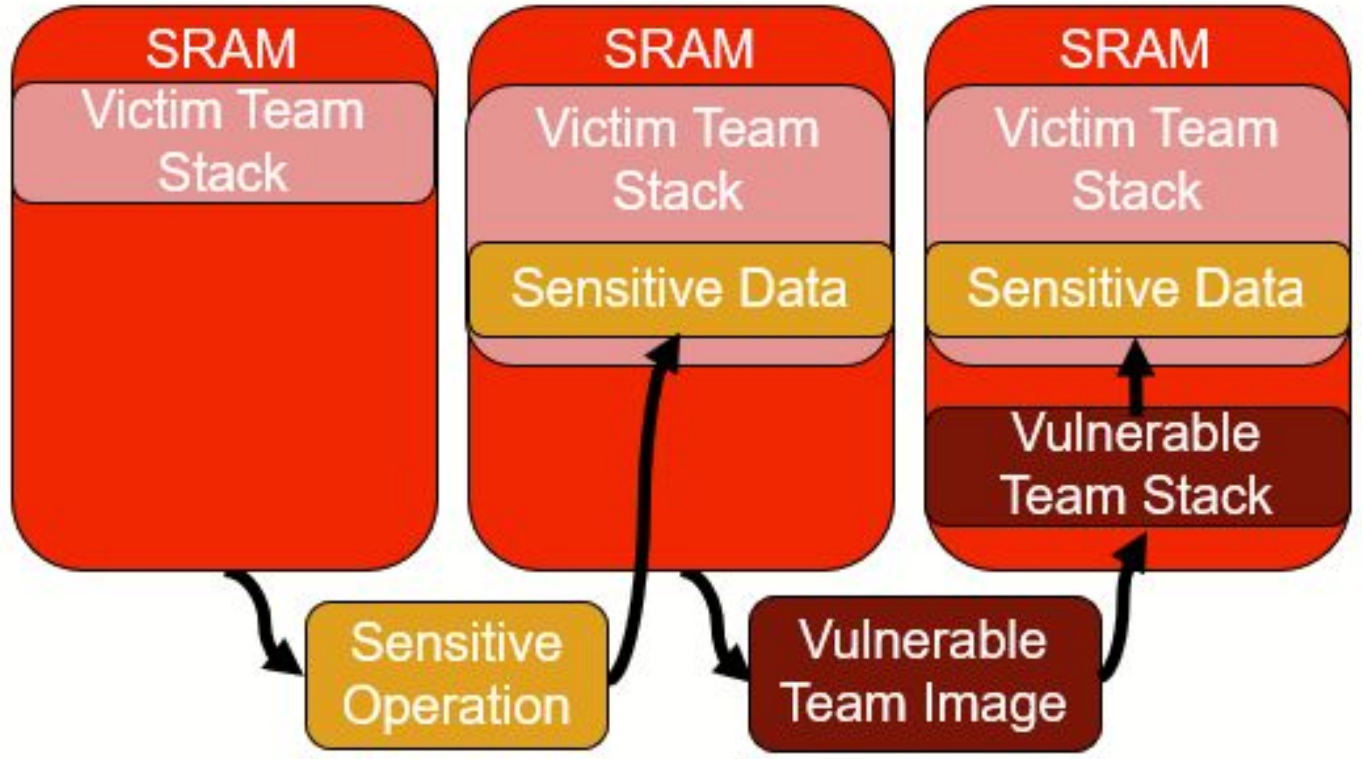
Attack Phase

b01lers

Attack Highlight

Stack Leak

- Boards with flags can only run signed firmware images. However, the attacker can flash any correctly signed firmware at any point on the car/fob.
- By flashing a vulnerable and a victim firmware on the car/fob, we leveraged the vulnerable firmware to extract sensitive data left behind from victim firmware images.

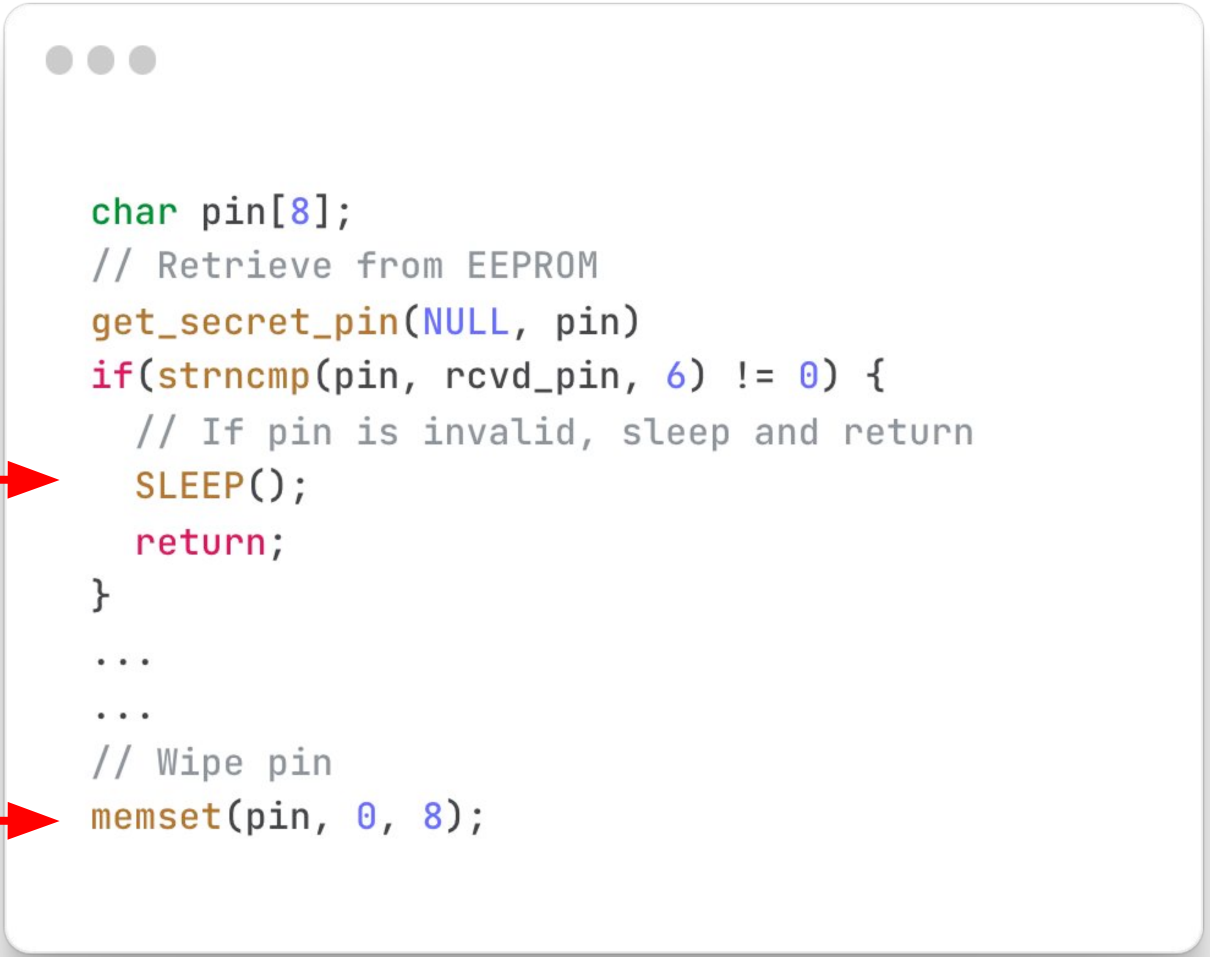


By leveraging these leaks, we successfully extracted private keys and pairing pins on the test boards. However, this attack did not work on keyed boards since the bootloader clears the SRAM and removes any sensitive data left by the victim team.

b01lers

Countermeasure to Stack Leak

- **Countermeasure:** Ensure that variables stored on stack are wiped when not needed



```
char pin[8];
// Retrieve from EEPROM
get_secret_pin(NULL, pin)
if(strncmp(pin, rcvd_pin, 6) != 0) {
  // If pin is invalid, sleep and return
  SLEEP();
  return;
}
...
...
// Wipe pin
memset(pin, 0, 8);
```

b01lers

Weak PRNG

- Secure cryptography requires good randomness. However, the attacker effectively has a save state of the car through the distributed firmware.
- By flashing a vulnerable car/fob, we reset the PRNG to the same initial state if the team doesn't design it properly.
- We can first flash the car/fob, observe the PRNG output through the challenges, then re-flash the car/fob and get the same challenge.
- **Countermeasure:** Introduce entropy on flash / first boot / first randomness output. There are several sources of good entropy on board that can be used to seed the PRNG, which mitigates the issue.

Common Attacks

- Shared Secrets : Shared secrets allowed reusing fobs on other cars.
- Brute Force : No limits on the number of attempts allowed to brute force the PIN on the fob.
- Buffer Overflow : We wrote exploits to leak flags and pins from various teams.

Final Comments

- What features of opponents' systems made things difficult for you as an attacker?
  - Secure designs with high-entropy RNG
  - Inconsistency between comments, docs, and code
    - This is **NOT** a recommendation to use security by obscurity…
  - Memory-safe Rust
- Would some of your attacks also be successful against your own system?
  - Likely, replay attacks on weak PRNG
  - Stack leak is also potentially possible against our system

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

Questions?