UCSC

University of California, Santa Cruz

Brian Mak, Steven Mak, Jeffrey Zhang, Victor Ho, Jackson Kohls, Nancy Lau, Iakov Taranenko, Stephen Lu, Chiara Knicker, Eya Badal Abdisho
Advised by: Professor Alvaro Cardenas
April 24, 2023

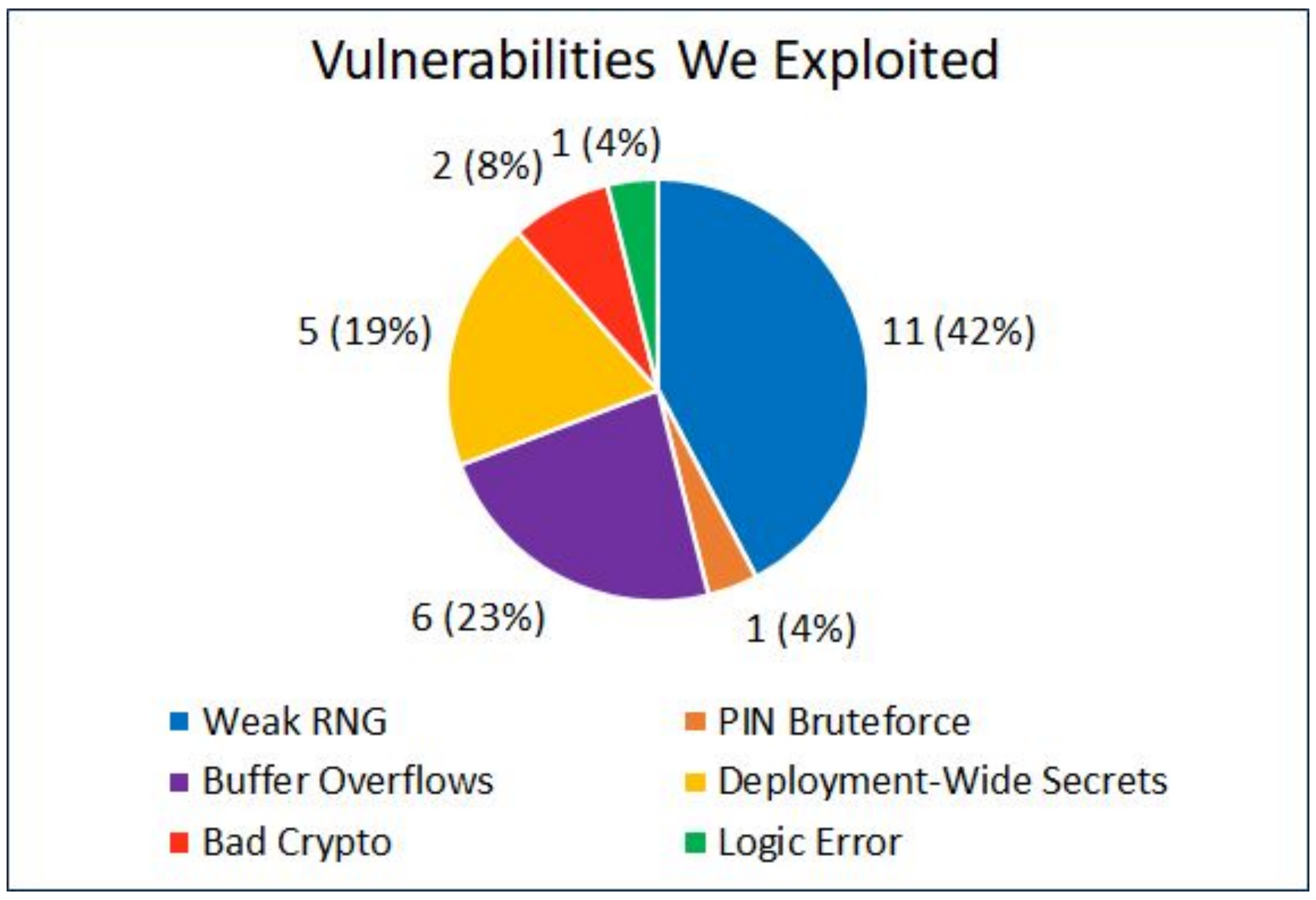
Design Overview

Unlock: Challenge-response protocol with a randomly generated number as the challenge

Pairing: ECDHE to generate keys for the challenge-response protocol, and increased delay after incorrect PIN

Features: Car ID & feature no. signed with manufacturer key

- Used multiple entropy sources to ensure generation of secure and unpredictable random numbers
- Secure RNG combined with built-in encryption in communication stack ensures SR1, SR2, and SR3
- Increasing delay after an incorrect PIN makes brute-force attack unfeasible, ensuring SR4
- Digital signatures and including the car ID in the feature package ensure SR5 and SR6



Defensive Highlight

Security measures we decided to include in our design:

- Rust was used to ensure memory safety
- RNG used four sources of entropy to ensure secure and unpredictable random numbers
- Widely-used Rust crates were chosen to ensure the security of cryptographic functions used
- Reusable abstractions for security-critical code used, such as built-in encryption in communication stack
- Code review and dependency auditing done to prevent errors and ensure dependencies were up-to-date
- Stack space was moved below .data and .bss sections to prevent stack overflows from leaking sensitive data

We decided to include a variety of sources of entropy for redundancy in case one or more of our sources were able to be controlled by an attacker. We acquire approximately 256 bits of entropy from each of our sources: a device-unique seed base, clock drift, the temperature sensor, and uninitialized memory. We then hash the input with SHA3-256 to whiten it. This worked very well in our favor considering that the temperature sensor can be potentially controlled. Additionally, the secure bootloader zeros out the SRAM on startup, rendering uninitialized memory useless as well. Despite all of this, we still have working sources of entropy that are secure. In the future, we could build upon this by putting the output of each entropy source through a Von Neumann extractor before hashing to gather a precise number of bits of entropy. This would provide us protection against entropy source tampering that attempts to reduce entropy.

Offensive Highlight

Attacks that our team developed:

- Timing attacks to break timing-reliant RNG
- ROP chains to exploit buffer overflows
- Replay attacks with deployment-wide secrets
- Message forging with secrets leaked in features
- Nonexistent or improperly placed delays allowing PIN brute force
- Misuse of strncmp for memory comparisons allowing comparisons to be bypassed
- Custom key exchange protocol allowing decryption of all traffic and forging of unlock requests
- Lack of authentication or encryption allowing enabling of features

A couple of teams use the internal temperature sensor readings as part of their RNG. This sensor is read using the onboard ADCs, which rely on two reference pins VDDA and GNDA. By disconnecting GNDA from the board and shorting it with VDDA, we are able to lower the ADC reading by 0x200.
This lowers the required temperature to saturate the ADC reading from around 145°C to 117°C. One team uses a Von Neumann extractor to generate random numbers, so saturating the ADC reading alone would not work. Instead, we can toggle a relay connecting VDDA and GNDA at the sample rate of the ADC, ensuring that readings will alternate between saturated and non-saturated, hopefully producing a predictable value.
In the end, we didn’t have the time to carry out the attack.
A variety of different RNG sources like we used in our design would largely mitigate this sort of attack.