Ret2Rev

Texas A&M University

Abhishek Bhattacharyya, Bode Raymond, Glenn Fitzpatrick, Liam Haber, Matthew Le, Nathan Nguyen

Advised by: Dr. Martin Carlisle

Design Overview

Unlocking

- Each car is intended to have 2 unique keys, one of which is for ensuring the legitimacy of challenges so that fobs don't solve arbitrary ciphertexts, and so that it can verify responses in the challenge-response system.
- Fob authenticates w/ the car in a challenge-response system which verifies fob has the requisite paired key. ONLY after successful authentication, car sends unlock flag and enabled features message to host tools. See ***Figure 1***.

Security Requirements

- Challenge-response system is used to ensure only paired fobs can unlock the car w/ the correct computed response.
- This mitigates attacks from non-authentic fobs as well as replay attacks. Attempting to replay the unlock command will issue a new challenge from the car, which can only be completed using the paired key from an authenticated fob.
- Package contents of fob features are hashed and signed with a private key to verify authenticity of the package.
- Each fob verifies the carID of a loaded package matches the fob's own carID to prevent mix-and-match attacks.
- There is a timeout to prevent PIN brute-force. When a pairing attempt fails, the fob will time out for 5 seconds.

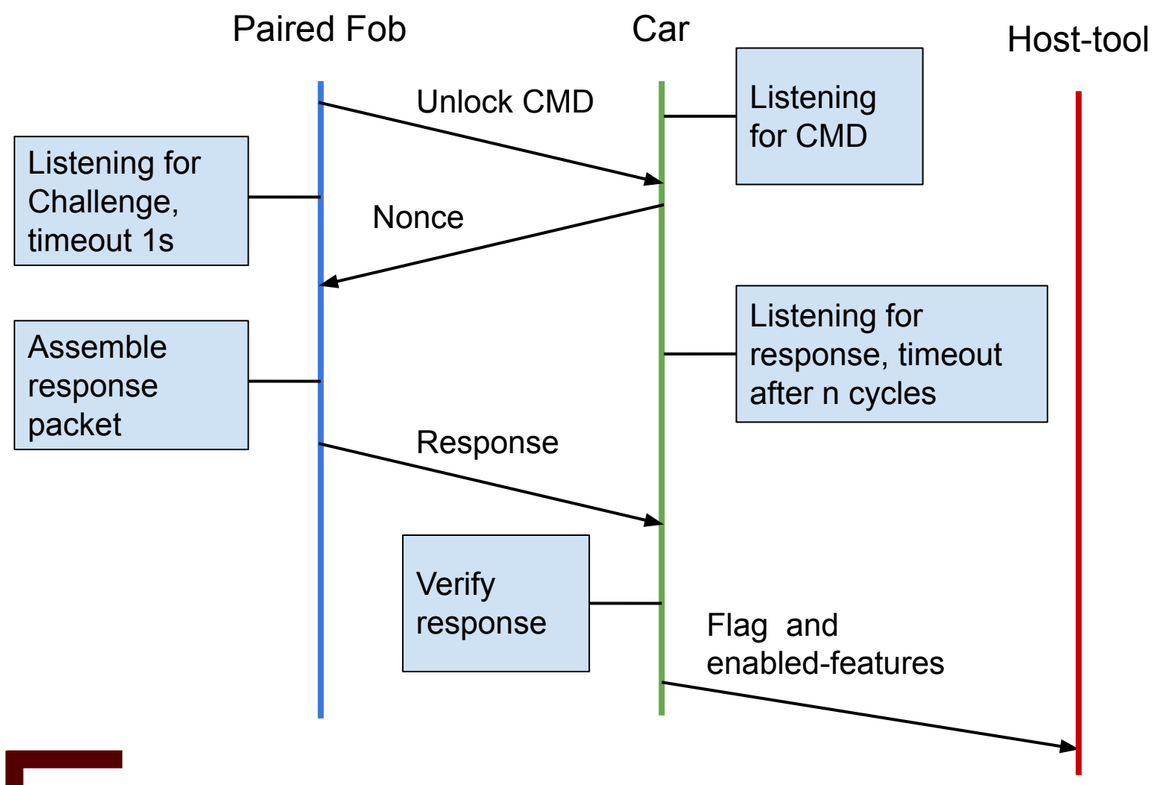
Defensive Highlight

As data needs to be accessed on the device, it is decrypted into a buffer in SRAM along with its hash. At time of use, the integrity of the data is verified using the hash. This provides strong security for data stored on SRAM, mitigating hardware attacks such as flash trojans. Short random delays during code execution are also used to make hardware attacks that rely on precise timings much more difficult to accomplish.

Throughout the firmware design, special attention was paid to ensure all required security features and exploit mitigations met each aspect of CIA triad (Confidentiality, Integrity, and Availability) within the rules of the competition framework.

One area where flaws may be present is in cloning attacks. Without hardware attestation, traditional cloning attacks on the fob are difficult to mitigate. Further investigation is needed to determine if a unique hardware fingerprint can be acquired to implement mitigations against cloning.

Figure 1: Car/Fob Communication and Host Tools Workflow



Offensive Highlight

Our primary attack vector was a replay attack predicated on the car providing predictable nonces for the unlock command in some designs. If the fob does not verify with the car when it receives a nonce, any plaintext nonce can be provided for the fob to assemble a response packet. This allowed us to replay valid unlocks we captured, and can used this to gain the 'temporary fob access' and 'passive unlock' flags. Additionally, for the enable feature flag, it was discovered if a plaintext payload of form *featurenum+carid* is sent to the fob as the nonce, the fob will return the corresponding ciphertext used to unlock that feature. Then, sending this ciphertext back to the paired fob enables the feature on the fob.

Additionally, the reference implementation had a buffer overflow in the uart_readline(); allowing attackers to write arbitrarily large messages to the buffer. This in turns lets an attacker overwrite the saved link register and gain control of the program counter (see ***Figure 2***). While several designs were vulnerable, we could not implement this due to time constraints.

To mitigate the buffer overflow attack, implementers should add bounds checking to the loop in main which handles polling for uart commands. This is done by checking the loop counter against the buffer size on every iteration of the loop to ensure data input does not exceed the buffer size. Once the buffer is full, the loop must exit and the buffer be cleared before the fob can poll for another UART command.

Figure 2: Buffer Overflow Exploit