An Automated and Principled Security Analysis Framework for Bluetooth LE Implementations

Syed Rafiul Hussain*, Victoria C. Moore†, Elisa Bertino*
*Purdue University, †Intel Corporation

Bluetooth Low Energy For Proximity-based Communication

Smart devices are connected to IoT gateways, e.g., Smartphone

Vulnerabilities in Bluetooth Protocol Implementation

Nest security cameras can be knocked out via Bluetooth

Why Such Vulnerabilities?

(1) Parsing Errors: BLE Implementations do not correctly parse and process the BLE packets.
(2) Semantic Bugs: Implementations deviate from Bluetooth standard specifications and hence contains functional or semantic bugs.
(3) Memory Corruption Bugs: Use-after-free, buffer overflow, etc.
(4) Weak Cryptographic Primitives: Cryptographic building blocks used in the protocol are prone to existing attacks.

Problem Objective

Fuzzing:
- Cannot explore the functional bugs.
- Cannot point out the location of the bug.
- Poor code coverage

Symbolic Execution:
- State explosion problem.

Our Proposed Approach

(1) Extract Finite State Machine
- Using a combination of
  - static analysis
  - symbolic execution
(2) Security Evaluation
- Find missing checks
- Use model checking to find property violation
- Perform differential testing by comparing two FSMs

Find Missing Checks

Malicious packets may get accepted by an implementation if certain checks are missed

Solutions:
1. Compare path constraints for two different implementations
2. Find the relevant fields of a packet in the list of path constraints.

Find Property Violation

- Select important security property from standard specification.
  - "The length of the pin code must not exceed 128 bits"
- Convert this property to a logical formula.

Differential Testing

Difference between two FSMs refers to possible discrepancy

Conclusion

Though, developers often optimize the complex part of the specification for embedded devices, they need to make sure the implementation complies with specification.

*This work is supported by Intel Corporation

Overflow PIN Code Memory in BlueDroid

If a malicious client sets a pin that was too long it would overflow the pin code memory.

Overflow PIN Code Memory in
BlueDroid

Porsche’s Car Kit Authentication

Car Kit’s authentication bypass with Android Phone
Goes directly to BLE_PAIR_AUTH_COMPLETE state if there is a saved PIN code.

Why Existing Techniques Fall short?

Fuzzing:
- Cannot explore the functional bugs.
- Cannot point out the location of the bug.
- Poor code coverage

Symbolic Execution:
- State explosion problem.

Find Property from specification

Model Checker

YES

NO

Counter-example

Find Missing Checks

Find Property Violation

PC7

Implementation 1

Implementation 2

Implementation 1

Implementation 2

Difference between two FSMs refers to possible discrepancy

Conclusion

Though, developers often optimize the complex part of the specification for embedded devices, they need to make sure the implementation complies with specification.

*This work is supported by Intel Corporation

Why Existing Techniques Fall short?

Fuzzing:
- Cannot explore the functional bugs.
- Cannot point out the location of the bug.
- Poor code coverage

Symbolic Execution:
- State explosion problem.

Find Missing Checks

Malicious packets may get accepted by an implementation if certain checks are missed

Solutions:
1. Compare path constraints for two different implementations
2. Find the relevant fields of a packet in the list of path constraints.

Find Property Violation

- Select important security property from standard specification.
  - "The length of the pin code must not exceed 128 bits"
- Convert this property to a logical formula.

Differential Testing

Difference between two FSMs refers to possible discrepancy

Conclusion

Though, developers often optimize the complex part of the specification for embedded devices, they need to make sure the implementation complies with specification.

*This work is supported by Intel Corporation

Why Existing Techniques Fall short?

Fuzzing:
- Cannot explore the functional bugs.
- Cannot point out the location of the bug.
- Poor code coverage

Symbolic Execution:
- State explosion problem.

Find Missing Checks

Malicious packets may get accepted by an implementation if certain checks are missed

Solutions:
1. Compare path constraints for two different implementations
2. Find the relevant fields of a packet in the list of path constraints.

Find Property Violation

- Select important security property from standard specification.
  - "The length of the pin code must not exceed 128 bits"
- Convert this property to a logical formula.

Differential Testing

Difference between two FSMs refers to possible discrepancy

Conclusion

Though, developers often optimize the complex part of the specification for embedded devices, they need to make sure the implementation complies with specification.

*This work is supported by Intel Corporation