RIDAS: Real-time identification of attack sources on controller area networks

Jiwoo Shin*
Soongsil University
sy9254@gmail.com

Hyunghoon Kim*
Soongsil University
axolotl0210@gmail.com

Seyoung Lee
Korea University
seyoung0131@korea.ac.kr

Wonsuk Choi
Korea University
wonsuk85.choi@gmail.com

Dong Hoon Lee
Korea University
donghlee@korea.ac.kr

Hyo Jin Jo
Soongsil University
hyojinjo86@gmail.com

* Co-first Authors

USENIX Security 2023, 9-11 August 2023
Motivation

Background

Our Method

Evaluation

Discussion

Conclusion
**In-vehicle Communication System**

- **ECU (Electronic Control Unit)**
  - A small device in a vehicle’s body that is responsible for controlling a driving-related function

- **CAN (Controller Area Network)**
  - In-vehicle network designed to communicate between ECUs
  - ISO 11898
  - Broadcasting
  - No data encryption
  - No sender/receiver authentication

→ **Security is needed for CAN**
Intrusion detection system (IDS)

- Lots of Rule-based or AI-based methods have been proposed

→ Attack detection only, attack source cannot be identified
Intrusion detection system (IDS)

- Able to identify the compromised ECU
  - The ECU’s clock skewness-based method proposed in [1]
    → The ECU’s clock skew was found to be corrupted by modifying the timing of transmitted messages [2]
  - The ECU’s physical layer signal-based method proposed in [3,4]
    → Need such a type of electronic test instrument that measures voltage signals
    → In addition, this device cannot identify the attack sources with 100% accuracy due to environmental factors such as its battery level, humidity, etc.
Contributions

Motivation

- Proposal of a novel real-time attack node identification method, called RIDAS
  - Using the error handling rule of CAN

- Proposal of a methodology that deals with RIDAS-aware attackers

- Evaluation of RIDAS on a CAN bus prototype and a real vehicle
Outline

- Motivation
- Background
- Our Method
- Evaluation
- Discussion
- Conclusion
### Controller Area Network (CAN)

**Background**

- **Error handling and fault confinement**
  - ECU has two registers: TEC, REC
  - ECU's error state: Active, Passive, Bus-off
    - **The active state**: default
      - ECU with error active state
        - Time
        - 3 bits (inter-frame space (IFS))
        - ECU with error active state
        - Time
        - 3 bits (IFS) + 8 bits (suspend transmission)
    - **The passive state**: penalty in message sending
      - ECU with error passive state
        - Time
        - 3 bits (IFS) + 8 bits (suspend transmission)
    - **The bus-off state**: blocked from the network
      - ECU with bus-off state
        - Time
The occurrence only in an error passive state

- It is that messages with lower priority are transmitted before messages with higher priority
  - ex) message priority: ID A and ID B > ID C
Outline

- Motivation
- Background
- Our Method
- Evaluation
- Discussion
- Conclusion
### Attack Model

**Naïve attacker**
- Using the default setting of the CAN controller

**RIDAS-aware attacker**
- Exploiting CAN controller’s functions to evade RIDAS
  - CAN controller reset
  - One-shot mode
  - Fast message transmission

Our Method

![Diagram showing the attack model]
**System overview**

- Four modules
- ECU mapping table
- Two modes
  - For the naïve attacker
  - For the RIDAS-aware attacker

**RIDAS: Workflow**

Our Method

<table>
<thead>
<tr>
<th></th>
<th>CAN ID</th>
<th>Transmission cycle</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0x001</td>
<td>20ms</td>
</tr>
<tr>
<td></td>
<td>0x002</td>
<td>10ms</td>
</tr>
<tr>
<td>B</td>
<td>0x003</td>
<td>20ms</td>
</tr>
<tr>
<td></td>
<td>0x004</td>
<td>10ms</td>
</tr>
<tr>
<td>C</td>
<td>0x005</td>
<td>20ms</td>
</tr>
</tbody>
</table>

**ECU**

- **CAN**
- **IDS**
- **TEC Emulation Module**
- **RIDAS-aware Attack Source Identification (RASI) Module**
- **Attack Handling (AH) Module**
- **Naïve Attack Source Identification (NASI) Module**
- **RIDAS Mapping Table**

**Working start**

**Detection attack!**
First, initialization before starting RIDAS

- Start the TEC emulation
  - Monitors the CAN bus in real-time and emulates the TEC of each ECU
- Set each representative ID (RID) for all ECUs
  - CAN ID with the fastest transmission cycle and higher priority

<table>
<thead>
<tr>
<th>ECU</th>
<th>CAN ID</th>
<th>Transmission cycle</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0x001</td>
<td>10ms</td>
</tr>
<tr>
<td></td>
<td>0x002</td>
<td>20ms</td>
</tr>
<tr>
<td>B</td>
<td>0x003</td>
<td>20ms</td>
</tr>
<tr>
<td></td>
<td>0x004</td>
<td>10ms</td>
</tr>
<tr>
<td>C</td>
<td>0x005</td>
<td>20ms</td>
</tr>
</tbody>
</table>
Second, Transition the error state of the compromised ECU

- When an attack message is detected, the AH module injects continuous errors before the message transmission is completed
- AH module aims to transition the compromised ECU from the error active state to the error passive state to induce the priority reduction
Third, Identification of the ECU where the error state has transitioned

- To identify the compromised ECU (i.e., the naïve attacker) who has transitioned to the error passive state, the NASI module sequentially inspects all ECUs.
- NASI module generates bit-errors pre-defined number of times \((k)\) for all RIDs to observe the priority reduction.
■ **Third**, Identification of the ECU where the error state has transitioned

- To **identify the compromised ECU (i.e., the naïve attacker)** who has transitioned to the error passive state, the NASI module sequentially inspects all ECUs
- NASI module **generates bit-errors pre-defined number of times \( k \) for all RIDs** to observe the priority reduction

![Diagram](image)

**Our Method**

RIDAS: Workflow (for the naïve attacker)
**RIDAS: Workflow (for the naïve attacker)**

- **Third**, Identification of the ECU where the error state has transitioned
  - The ECU of RID in which priority reduction has occurred is the compromised ECU

---

**Our Method**

**ECU A**  
**ECU B**  
**ECU C**

- ECU C's RID
- **Lower priority**
- ECU C's RID

**x k times**

**Retransmission**

Observe whether the priority reduction occurs

**RIDAS**

- TEC Emulation Module
- RASI Module
- AH Module
- NASI Module
- ECU Mapping Table

**IDS**
**Forth, Restart RIDAS**

- Before restarting RIDAS, **the AH module reduces increased ECU’s TEC** by generating request messages (e.g., remote frame or UDS message) for all ECUs.
RASi module deals with attackers who evade RIDAS by monitoring the CAN bus

- **CAN controller reset**
  - Detection of *the change in the transmission cycle* of certain CAN packets

- **One-shot mode**
  - Detection of *the non-retransmission*

- **Fast message transmission**
  - Whenever 8 fast messages are detected, a bit-error is injected to restore the compromised node's TEC to its original value
Experimental Setup

- CAN bus prototype
  - ECU: Arduino Uno with CAN Bus Shield (x10)
  - RIDAS: CAN Pico (x2), ECU (x2)
  - Monitoring tool: PCAN-USB Pro FD

- Real vehicle
  - RIDAS
  - Monitoring tool
  - CAN DBC: openDBC
  - Vehicle: Hyundai Avante CN7 2020

openDBC: https://github.com/commaai/opendbc
Evaluation of AH

- The AH module prevents driving the ECU into the bus-off state

Due to NASI's error injection
Due to AH's error injection
Other ECUs' inspection is completed
Start RIDAS
Restart RIDAS

Don't enter the bus-off state
Due to NASI's error injection
Baseline = 79
Start RIDAS
By requesting UDS messages & Restart RIDAS
By requesting remote frames & Restart RIDAS
Evaluation of NASI (identification rate)

- The probability of priority reduction according to the bus load and the number of error injections ($k_{\text{check}}$)

CAN bus prototype

Real vehicle

$\mathbf{k}_{\text{check}}: 7 \sim 8$

$\mathbf{k}_{\text{check}}: 4 \sim 5$

$\mathbf{k}_{\text{check}}: 3 \sim 4$
Evaluation of NASI (identification time)

- The average identification time of the NASI module
  - The identification time does not exceed 500ms in Avante CN7
    - Baseline = 79, $k_{\text{check}} = 5$, bus load = 80%

<table>
<thead>
<tr>
<th>Completion Time (ms)</th>
<th>Compromised ECU</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>G</th>
<th>H</th>
<th>I</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Prototype</td>
<td>164.6</td>
<td>75.7</td>
<td>106.3</td>
<td>73</td>
<td>42.4</td>
<td>43.7</td>
<td>33.2</td>
<td>5.4</td>
<td>563.5</td>
</tr>
<tr>
<td></td>
<td>Avante</td>
<td>150.1</td>
<td>75.8</td>
<td>99.6</td>
<td>72.3</td>
<td>38.4</td>
<td>40.1</td>
<td>29.7</td>
<td>5.1</td>
<td>458.6</td>
</tr>
</tbody>
</table>
Evaluation of RASI (for the RIDAS-aware attacker)

- Response to the ECU reset and the use of one-shot mode
  - There is a notable change in the message transmission cycle of the ECU

Message transmission cycle variations of ECU
(left: reset off / right: reset on)

Message transmission cycle variations of ECU
(left: default mode / right: one-shot mode)
Outline

- Motivation
- Background
- Our Method
- Evaluation
- Discussion
- Conclusion
Discussion

- Intrusion detection system
  - The IDS with RIDAS must detect attacks before the messages are completely transmitted
    - The worst-case response time-based IDS [5]

- Limitation of RIDAS
  - Direct TEC manipulation attack
    - Cannot drive a compromised ECU into the error passive state
  - ID reuse attack
    - Only nodes that attempt a masquerade attack can be identified

Outline

- Motivation
- Background
- Our Method
- Evaluation
- Discussion
- Conclusion
Conclusion

- Proposed a novel real-time attack node identification method, called RIDAS
  - RIDAS identifies an attack source using the priority reduction of an ECU’s error passive state

- Evaluated RIDAS on a CAN bus prototype and a real vehicle
  - RIDAS is capable of identifying the attack source without affecting driving
  - RIDAS is robust against changes in a vehicle’s environment

- In future research, we plan to integrate a lightweight IDS into RIDAS
Q&A
Thank you