Catch Me If You Can: Automated Tools for Embedded Software Analysis

Marc Fyrbiak | January 17th 2019 | AI Studienprojekt
**Table 116. Lock Bit Protection Modes**

<table>
<thead>
<tr>
<th>Memory Lock Bits</th>
<th>Protection Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>LB mode</td>
<td>LB2</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
</tr>
</tbody>
</table>
Analysis of

In this report we summarize our findings on the system. First, we state our findings on the deployed data flow and the protocol for communication from the [redacted] to the [redacted] (see Section 1). Specifically, we describe how the input data is composed to form the buffer for the encryption before it is sent. Second, we detail five data samples (see Section 2) which include input data and the corresponding plaintext, ciphertext, and transmitted frames.

Disclaimer: Our results are based on assumptions about firmware interpretation. Thus, our findings may be incomplete or reflect unintended behavior.
Our Solution - Technical Details

Static Program Analysis and Transformation Framework

Compilation   Protection   Assembling
1. Development of Program Analysis / Transformation Technique(s)
   ○ Analysis case study: dataflow analysis
   ○ Transformation case study: custom virtual machine obfuscation

2. Custom Embedded Development Tools
   ○ Extend debugging capabilities for obfuscated embedded systems
   ○ RTOS testsuite

Application is mandatory! Contact us via: studienprojekt@empproof.de
Requirements and What You Learn

**Requirements:**
- 2 students (Master)
- Background in C++, Python
- Understanding of at least one of the following:
  - Embedded Systems and Assembler
  - Compilers
  - Reverse-Engineering / Obfuscation

**What You Learn:**
- Deep embedded systems knowledge
- Compiler internals (and their quirks)
- How systems are reverse-engineered