Poster: \( \mu \text{RAI}: \) Securing Embedded Systems with Return Address Integrity

Naif Saleh Almakhdhub  
Purdue University and  
King Saud University

Abraham A. Clements  
Sandia National Laboratories

Saurabh Bagchi  
Purdue University

Mathias Payer  
EPFL

Abstract

Embedded systems are deployed in security critical environments and have become a prominent target for remote attacks. Microcontroller-based systems (MCUS) are particularly vulnerable due to a combination of limited resources and low level programming which leads to bugs. Since MCUS are often a part of larger systems, vulnerabilities may jeopardize not just the security of the device itself but that of other systems as well. For example, exploiting a WiFi System on Chip (SoC) allows an attacker to hijack the smart phone’s application processor.

Control-flow hijacking targeting the backward edge (e.g., Return-Oriented Programming–ROP) remains a threat for MCUS. Current defenses are either susceptible to ROP-style attacks or require special hardware such as a Trusted Execution Environment (TEE) that is not commonly available on MCUS.

We present \( \mu \text{RAI} \), a compiler-based mitigation to prevent control-flow hijacking attacks targeting backward edges by enforcing the Return Address Integrity (RAI) property on MCUS. \( \mu \text{RAI} \) does not require any additional hardware such as TEE, making it applicable to the wide majority of MCUS. To achieve this, \( \mu \text{RAI} \) introduces a technique that moves return addresses from writable memory, to readable and executable memory. It re-purposes a single general purpose register that is never spilled, and uses it to resolve the correct return location. We evaluate against the different control-flow hijacking attacks scenarios targeting return addresses (e.g., arbitrary write), and demonstrate how \( \mu \text{RAI} \) prevents them all. Moreover, our evaluation shows that \( \mu \text{RAI} \) enforces its protection with negligible overhead.

1 Reference

This work will appear at NDSS 2020:


2 DOI

Network and Distributed Systems Security (NDSS) Symposium 2020  
23-26 February 2020, San Diego, CA, USA  
ISBN 1-891562-61-4  
https://dx.doi.org/10.14722/ndss.2020.24016  
www.ndss-symposium.org

\(^1\)https://github.com/embedded-sec/uRAI
μRAI : Securing Embedded Systems with Return Address Integrity[1]

Naif Saleh Almakhdhub  
Purdue and King Saud University

Abraham A. Clements  
Sandia National Laboratories

Saurabh Bagchi  
Purdue University

Mathias Payer  
EPFL

Problem

- Embedded systems and IoT are run on Microcontroller systems (MCUS)
- MCUS lack basic mitigations and are prone to control-flow hijacking attacks such as Return Oriented Programming (ROP)
- Proposed defenses have limited security guarantees, high runtime overhead, or require special hardware features

Objective

- Return Address Integrity (RAI) prevents ROP attacks on MCUS
- RAI results in low runtime overhead
- RAI does not require special hardware

μRAI

- Identifies the possible return targets of each function in the call graph
- Transforms the set of return targets to a jump table in R+X memory
- Introduces a State Register (SR), which is never spilled and is exclusively used by μRAI
- Uses the SR at run time to resolve the correct return location from the jump table
- Enforces the RAI property since the SR and jump table are inaccessible to an adversary
- Protects sensitive Memory Mapped IO (MMIO) by enforcing Software-based Fault Isolation (SFI) on functions callable within an exception handler context to protect sensitive such as the MPU
- Partitions the SR into segments to curb path explosion
- Applies a type-based CFI for forward edges

Compiler Transformation

- Identify the possible return targets of each function in the call graph
- Transform the set of return targets to a jump table in R+X memory
- Introduce a State Register (SR), which is never spilled and is exclusively used by μRAI
- Use the SR at run time to resolve the correct return location from the jump table
- Enforce the RAI property since the SR and jump table are inaccessible to an adversary
- Protect sensitive Memory Mapped IO (MMIO) by enforcing Software-based Fault Isolation (SFI) on functions callable within an exception handler context to protect sensitive such as the MPU
- Partition the SR into segments to curb path explosion
- Apply a type-based CFI for forward edges

Evaluation

- μRAI enforces the RAI property with low overhead in contrast to mechanisms requiring full-SFI

Comparison to backward edge

<table>
<thead>
<tr>
<th>App</th>
<th>Type-based CFI Target Set</th>
<th>Max.</th>
<th>Ave.</th>
</tr>
</thead>
<tbody>
<tr>
<td>PinLock</td>
<td></td>
<td>8</td>
<td>3</td>
</tr>
<tr>
<td>FatFs_uSD</td>
<td>94</td>
<td>21</td>
<td></td>
</tr>
<tr>
<td>FatFs_RAM</td>
<td>94</td>
<td>27</td>
<td></td>
</tr>
<tr>
<td>LCD_uSD</td>
<td>49</td>
<td>11</td>
<td></td>
</tr>
<tr>
<td>Animation</td>
<td>49</td>
<td>11</td>
<td></td>
</tr>
<tr>
<td>CoreMark</td>
<td>52</td>
<td>12</td>
<td></td>
</tr>
</tbody>
</table>

μRAI eliminates the remaining attack surface for control-flow bending attacks

Security

- Prevented
- Buffer overflow: ✓
- Arbitrary write: ✓
- Stack pivot: ✓

μRAI prevents all control-flow hijacking attack scenarios targeting return addresses

References


Sandia National Laboratories is a multimission laboratory managed and operated by National Technology & Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International Inc., for the U.S. Department of Energy’s National Nuclear Security Administration under contract DE-NA0003525. SAND2020-0912 C.