On the Difficulty of Software-Based Attestation of Embedded Devices

- introduction
- paper summary
- critique

- If a remote node can only use software to prove that it is still running as it should, it is difficult to do so.
- The paper shows two general attack methods that make the node appear to be uncompromised when required to prove itself.
- It also shows attacks against specific techniques, and how modifications can prevent the attacks.

- assumptions and previous work
- generic attacks
  - return-oriented rootkit
  - code compression
- difficulties with specific attestation proposals
  - SWATT

- Software code attestation
  - Remotely verify a node has not been compromised
  - Verify via memory checksum + nonce
- Attack goals
  - Modify executable memory
  - Still pass attestation

- General assumptions
  - Compromised device doesn't interact with other malicious nodes
  - Unmodified hardware (not tamper-resistant)
  - Verifier aware of configuration
- Hardware: MicaZ
  - COTS wireless sensor
  - Atmel AVR
  - Harvard memory architecture (program, data, and external memories)
- Paper contents applicable to similar micro-controllers

- return-oriented programming (ROP)
  - executes existing code (no code changes necessary)
  - Arbitrary functionality (given large enough code size)
  - Manipulates program stack so return executes desired code
  - Segment starts near a return statement, segments strung together
  - If existing code known, compilers make creation of ROP easy

– Attack uses ROP rootkit

- ROP root-kit attack
  - Start of attestation code modified to initiate cleanup sequence
  - Cleanup modifies return address on stack
  - Attestation occurs
  - Returns to ROP that initiates re-infection code

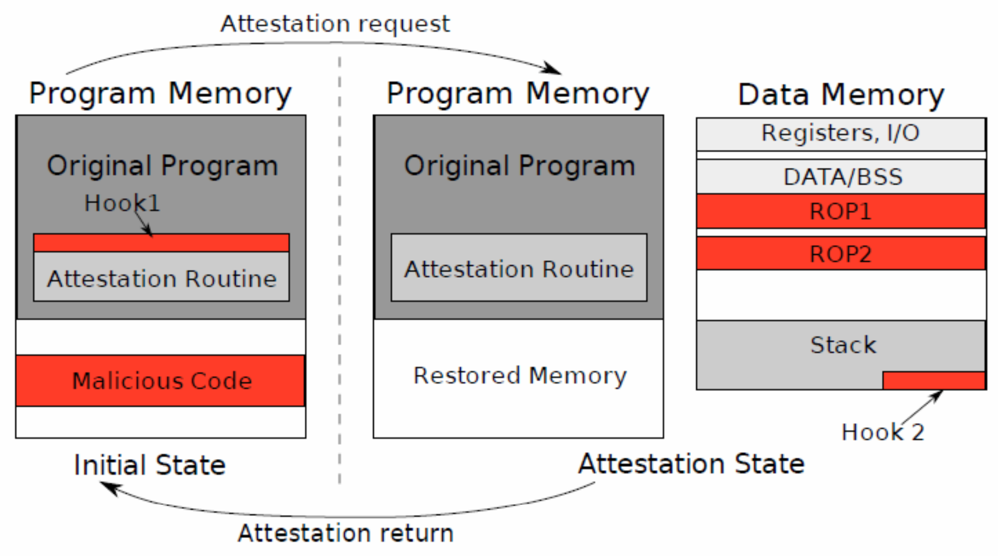

Figure 3: Return-Oriented Programming attack.

- Compression attack
  - Previously, unused program space filled with pseudorandom values so attacker cannot use them.
  - Compress code to make space for attack code
  - Decompressed on-the-fly during attestation
  - Achieved average of 11.6% compression

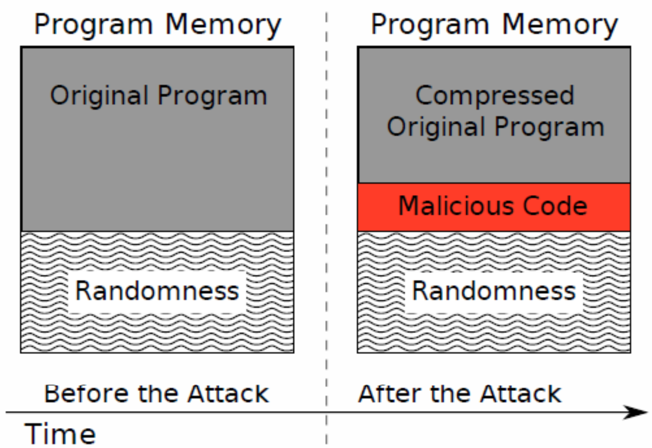

Figure 5: Compression Attack.

- SoftWare-based ATTestation (SWATT), Seshadri et. al.
    - Introduces time-to-respond
    - Attacker would slow down function if redirecting memory
    - Relies on fastest redirection and checksum known
- Paper introduces faster redirection
    - Requires half program memory unused
    - Redirect 0x11xx…xx accesses to 0x10xx…xx and store malicious code in 0x11xx…xx
    - 2 cycles vs previously fastest 3 cycles.
    - Still detectable .. relies on processor capabilities
- Porting SWATT required rewrite of algorithm, changed timing

- Preventing rootkit attack on SWATT
  - Data memory not verified, allows attack
  - Verify memory or clean memory after attestation
  - Verification difficult
    - Architecture uses different address space, instructions
    - Pseudorandom verification requires branch
    - Unpredictible contents (registers, I/O, stack)
  - Clean memory and reboot
    - Disrupts rootkit attack, not shadow attack