Diversify Sensor Nodes to Improve Security of Sensor Networks

Wenliang Du
Department of Electrical Engineering and Computer Science
Syracuse University, Syracuse, NY 13244-1240 USA

A **fundamental challenge** in securing sensor networks is that sensor nodes can be physically compromised. Most of the security mechanisms relies on the secrecy of some important data that is stored on sensor nodes. For example, for encryption, the security depends on the secrecy of keys. Because of the lack of physical security and memory protection, sensors can be captured by adversaries, and secret keys stored in memories can be compromised. Once those secrets are disclosed, a sensor is *completely* compromised, i.e., adversaries can command the sensor to behave maliciously. It is important to protect those sensitive data even if sensor nodes are compromised.

Our goal is not restricted to protect each node, but instead, to protect a significant number of sensors from being compromised. To avoid failure caused by a few malfunctioned or malicious sensors, sensor-network applications often adopt fault-tolerance technologies, so the compromise of a small number of sensor nodes does not compromise the entire mission. However, when a significant number of sensors are compromised, the trusted computing infrastructure depended upon by sensor networks can be compromised.

Challenges

Challenge 1: Disguising Sensitive Data. Secret data are normally stored in memories, so once adversaries have understood the memory layout, they can easily retrieve the sensitive data by dumping the entire memory. To defeat such naive memory-dumping attacks, these data need to be disguised, so knowing the memory layout alone cannot find the data. Adversaries must also understand the program in order to find out where each the data are stored. The challenges is how to automate such data disguising process.

Challenge 2: Obfuscating Code With the modern code analysis, debugging, and reverse engineering tools, adversaries can understand the program and then find the sensitive data. It is essential to make code understanding difficult. Code obfuscation has been extensively studied for traditional systems; its main goal is to increase the complexity of code to defeat reverse engineering efforts. During the past, a number of interesting techniques have been developed in the literature. However, these techniques were developed for traditional systems with abundant power and resources. It is quite challenging to directly adopt them for sensor networks and embedded systems.

Challenge 3: Diversifying Code Code obfuscation is not foolproof; dedicated adversaries can eventually get the confidential data from a captured node. Although it might take adversaries quite a significant amount of time to succeed, if the programs running on different sensors are the same or similar, once a node is compromised, compromising another node takes much less time. We need to use *diversity* techniques to turn the same piece of software into many diversified versions, such that a comparative study of an already-compromised node and a newly-captured node is still difficult. A great challenge is how to diversity code to defeat both static and dynamic matching attacks, without consuming too much resources.

Innovations

Due to the energy constraints of sensor networks, any viable disguising, obfuscation, and diversification method should be energy efficient. This will lead to studies and innovations that are significantly different from the traditional code obfuscation. Moreover, sensor networks and embedded systems have their own unique properties, some of which might benefit code obfuscation.

Code diversification has been used extensively to provide robustness to systems; however there is not much study in using it to enhance security. Robustness mostly deals with accidental fault, but for security, we face intelligent adversaries with sophisticated tools. Therefore, this research can lead to innovative technologies that can be applied to not only sensor networks, but also embedded systems.

Diversifying Sensors to Improve Network Resilience

Wenliang (Kevin) Du
Electrical Engineering & Computer Science
Syracuse University

Hiding Secrets

- Secrets are essential for sensor networks
  - Pre-distributed keys
  - Pair-wise keys
  - Private keys
  - Other secrets
- Fundamental Challenge: hiding secrets is difficult

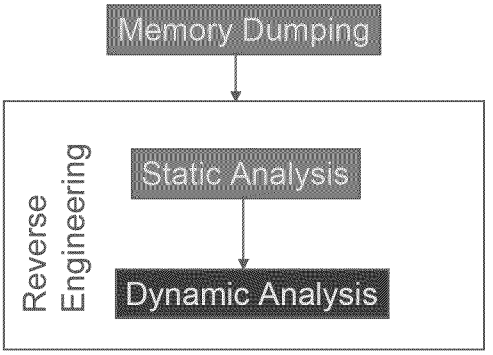
Existing Approaches

- Physical security is difficult to achieve
- Hardware approaches are expensive
- Software approaches
  - Code obfuscation: extensively studied in traditional systems
  - *Bad news*: adversaries eventually win

Rethinking of Software Approaches

- Observation: fault tolerance of sensor networks
  - Should be able to tolerate a small # of bad sensors
- Ideal Goals
  - Hiding secrets in sensor nodes
  - Make it difficult to derive secrets from each sensor
  - Make it N times difficult to derive secrets from N sensors

Threat Model: Physical Compromise



5

Proposed Approach

- Data Obfuscation (Secret Hiding)
  - Memory dump: difficult to find secrets
  - Adversaries must understand the program
- Code Obfuscation
  - Make it difficult to understand one program
- Code Diversification (Randomization)
  - Make adversary's effort non-repeatable

6

Data/Code Obfuscation

- Existing Techniques
  - Code flattening
  - Self-modification code
  - White-box encryption algorithms
  - Various techniques against reverse engineering
- Challenges
  - Achieving obfuscation with limited Memory
  - Computation can't be too expensive
  - Tradeoff needs to be made (optimization)
  - Quantify code complexity

7

Diversifying Code

- Turn the same piece of software into many diversified versions
- Difference from traditional diversity
  - Diversity for fault tolerance
  - Diversity for attack tolerance (vulnerabilities)
    - Attacks are quite fragile
  - Diversity for code-analysis tolerance
    - Attacks are adaptive and intelligent (human involved)

8

Diversifying Code: Challenges

- Quantify diversity and manageability
  - Manageability prefers uniformity
  - Diversity destroys uniformity
  - Manageability is application dependent
  - Optimal tradeoff
- Comparative study: already compromised node and newly-captured node
- Static matching attacks
- Dynamic matching attacks

9

Difference from Protecting Intellectual Right

- Intellectual Right
  - Success = breaking one copy
- Sensor Networks
  - Success = breaking more than k copies

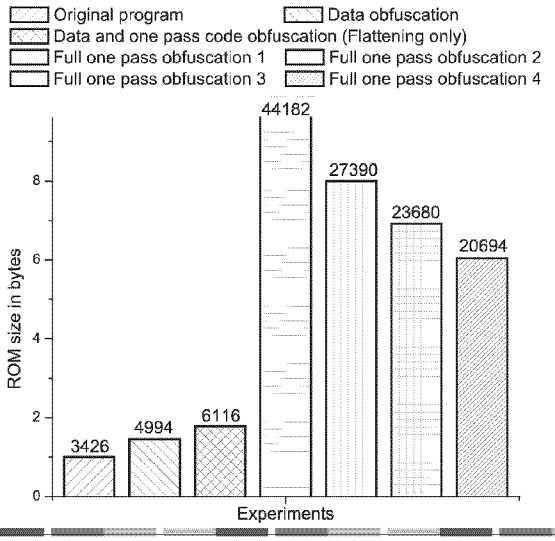
10

Unique Properties of Sensor Networks

- Code usually has small size
- Some applications has static configurations
  - The OS can be obfuscated too
- Hardware specific code obfuscation

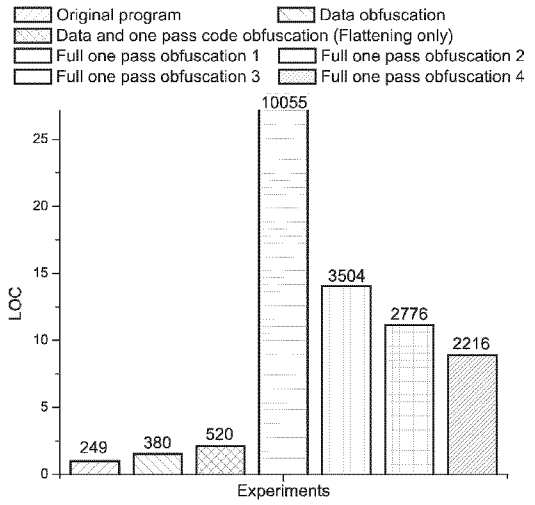
11

Preliminary Results: SASN'06



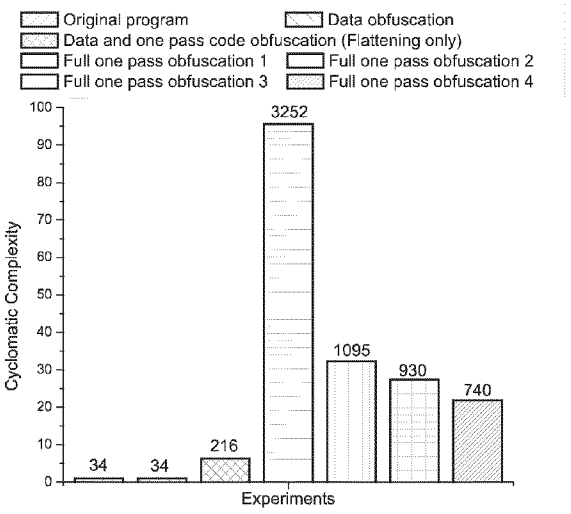
12

Complexity: Line of Code



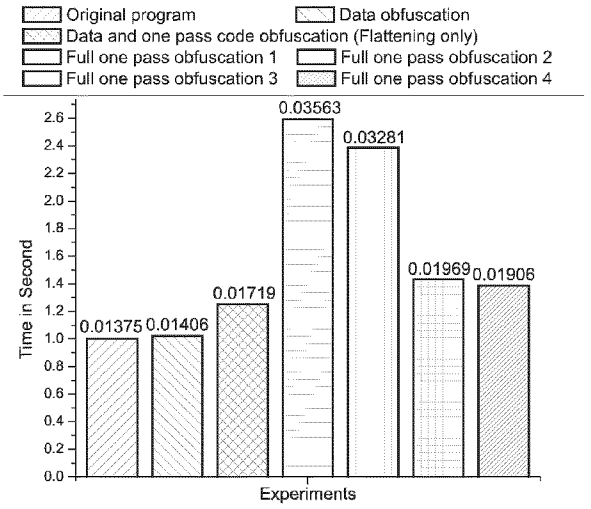
13

Cyclomatic Complexity



14

Running Time



Summary

- Diversified code obfuscation is quite unique for sensor networks
- Require understanding from both engineering and theory perspectives