Secure Programming for Fun and Profit
(Real World Experiences in Secure Programming)

Scott D. Miller
Security Analyst
Arxan Research, Inc.
Doctoral Student in CS
Advisors: Aditya Mathur; Ray DeCarlo
Build a Better Mousetrap…

• For most, it’s a game.
  – Much ego involved

• Secure programming boils down to
  – Enforcing access policy
    • E.g. code execution rights
  – Anti-tamper/tamper detection
  – Protection of intellectual property
Attacker Objectives

- Network-based attacks
  - Unauthorized code execution
  - Key reconstruction
- Malicious users
  - Circumventing digital rights management
  - “Cracking” (Red-team) and unauthorized distribution
- Well-funded nation and corporate adversaries
  - Tampering and unadvertised functionality
  - Recovery of sensitive IP
Attack Method

• Analysis of Software Code
  – Statistical properties
  – Disassembling

• Analysis of Running Software
  – Timing/Power Analysis
  – Debugging and Emulating

• Injection/Modification of Code
  – Patching, loading libraries, etc.

• Stimulation with malicious data
  – Buffer overflows, unexpected values
Some Interesting Scenarios

Secure Programming Examples in Industry
Code Injection

• Unexpected injection mechanisms
  – Through “social engineering” (old BBS days.)
  – Through the context menu!

• When prevention fails, use detection
  – Statistical profiling of system calls can be effective [For97, Hof98, Mic02, Mar00].
  – (U.S. Air Force proposal.)
Vulnerability Broadcasting

• Consider a security patch to Apache, IE, etc.
• Diff the patched vs. un-patched version.

• How many people put off downloading security updates?

• Obfuscation and execution path randomization can hide the patch from static and dynamic analysis (funded in part by U.S. Air Force.)
Interesting Obfuscation

• Eventually, they will get the code [And96].
  – Remember the U.S. fighter jet’s emergency landing in China?

• Obfuscating to match statistical code properties.

• Through numerical transforms
  – Data splitting (funded in part by U.S. Army.)
Our Approach

Program with Sensitive Content

Non-sensitive Program

Sensitive Program

Our Approach
Non-performance degrading AT

• For real-time systems (e.g. OSD Anti tamper requirements on all new weapons systems.)

• Security “co-processors” in FPGA (funded in part by Missile Defense Agency.)
Our Approach

- **Sensitive Software**
- **Anti-tamper Components**
- **RAM**
- **FPGA**
- **CPU**
- **Peripherals**
FPGA-Aided Encryption Protection

- Encrypted
- Decrypted
- Unprotected

CPU

Protected Program

Decryption Start Trigger

FPGA

Encryption Start Trigger

January 10, 2006
Summary

• There is no shortage of work for Secure Programming
  – Commercial, too (e.g. Microsoft, Boeing, Lockheed Martin, etc.)

• The threats are ever-evolving
  – Never do the same thing twice!
Questions?
References


