Vulnerability and Information Flow Analysis of COTS

Somesh Jha, Bart Miller, Tom Reps
{jha,bart,reps}@cs.wisc.edu
Computer Sciences Department
University of Wisconsin
1210 W. Dayton Street
Madison, WI 53706-1685
Phone: 608-262-9519
FAX: 608-262-9777
Cost of Software Development Motivates Use of COTS software

• High cost of software development
  • increased complexity
  • increasing degree of concurrency
  • increasing quality-assurance demands
  • other factors . . .

• Increased deployment of COTS

• CIP/SW TOPIC #6
  - Protecting COTS from the inside
COTS Spending on the Rise

• In 1991 DoD’s SAI initiative mandates defense contractors to consider COTS in their programs
• Today, a significant percentage of their IT budget is allocated to COTS
• Other countries are taking similar steps

Defense Contractor COTS Spending

Source: Jane’s Information Group
http://www.janes.com/
Note: IT Budget refers to total spent on
Advantages and Disadvantages of COTS

- **Advantages**
  - reduced cost
  - promotes modular design
  - partitions the testing effort

- **Disadvantages**
  - higher risk of vulnerabilities
  - general quality-assurance issues
Unsafe Malicious Code

• **Viruses**
  - *Gain access through infected files*

• **Worms**
  - *Spread over the network*

• **Trojans**
  - *Hide harmful behavior under the guise of useful programs*

• **Most often: combined code**
  - *worm + virus + trojan*

• **Distinguishing characteristics:** something observable happens
Malicious Code Example:

*Internet worm Sobig.E*

- Install worm code:
  - into the Windows folder
  - as a Win2K service

- Auto-update itself from a list of master servers:
  - relay spam
  - steal confidential data
  - install keyboard loggers

Windows Shares
E-mail

Feb 2005, ONR Review
S. Jha, B. Miller, and T. Reps
What Is Spyware?

- **Spyware is software that**
  - Is non-destructive (unlike a virus)
  - Operates in background—not easily observable
  - Is often installed silently by other software
  - Usually integrated with desired functionality

- **Privacy-violating malicious code**
  - Provides useful functionality
  - But, “leaks” sensitive information
KaZaa in Operation

Spyware
- Collecting user information
- Download/install programs
- Modify system settings

Sending collected information
Spyware Summary

• Install a useful program
  - Play DVDs

• But ...
  - Also install “spy” software, which monitors user behavior
    • Example: Monitor web traffic

• Aureate Media, Real Networks

• Consult
  - http://grc.com/optout.htm

• Maybe can be used by advisors/managers😊
Problems and Challenges

• Cannot expect to have source code for COTS software
  - Solution: we target executables

• Should handle unsafe and privacy-violating malicious code
  - Solution: initially targeted unsafe malicious code, but have started work on Spyware

• Certain executables are very hard to analyze statically
  - Solution: developed a sandboxing technology
WiSA and SandboX86: Static and Dynamic Approaches for COTS

- We have proposed the Wisconsin Safety Analyzer
  - vulnerability analysis
    - Handles unsafe malicious code
  - information flow analysis of COTS
    - Handles privacy-violating malicious code (Spyware)
- Develop technology for static and dynamic analysis of binaries
  - Original plan to focus on static analysis
  - Realized that we need multiple-lines of defense
  - Started working on dynamic analysis as well and developed a sandboxing system called SandboX86
- Investigate applications
Tools for Reducing the Risk of COTS Deployment

- Static analysis and rewriting of executables
- Sandboxing and dynamic slicing
- Evaluation and testing
Tools for Reducing the Risk of COTS Deployment

- Static analysis and rewriting of executables
- Malicious code detection
  - Model-based HIDS
  - Program Obfuscation
- Sandboxing and dynamic slicing
- Containing malicious behavior
  - Discovering potential privacy violations
- Evaluation and testing
  - Testing malware detectors
  - Testing NIDS
IDA Pro

• Decompilation tool
• Supports several executable file formats like COFF, ELF ....
• Gather as much information as possible
  • e.g. Names of functions, parameters to functions
• Is extensible through a built-in C-like language
Codesurfer

• A program-understanding tool
• Analyzes the data and control dependences
  - stores in System Dependence Graph (SDG)
  - Helpful in static analysis
• API to access information stored in IRs
  - Platform for additional static analysis
• The API can be extended
Architecture

Binary

IDA Pro
- Parse Binary
- Build CFGs

Connector
- Memory Analysis
- BREW Rewrite
- Generate Code

Codesurfer
- Build SDG
- Browse

Clients
- Detect Malicious Code
- Detect Buffer Overrun

System Dependence Graph

Control Flow Graphs

Generated Binary
Code Generation

Binary

IDA Pro
- Parse Binary
- Build CFGs

Connector
- Memory Analysis
- BREW Rewrite
- Generate Code

Codesurfer
- Build SDG
- Browse

Clients
- Detect Malicious Code
- Detect Buffer Overrun
- Build Program Specification

Generated Binary
Dynamic Buffer Overflow Detection

Binary

IDA Pro
- Parse Binary
- Build CFGs

Connector
- Memory Analysis

BREW
- Rewrite
- Generate Code

Codesurfer
- Build SDG
- Browse

Clients
- Detect Malicious Code
- Detect Buffer Overrun
- Build Program Specification

Generated Binary
Malicious Code Detection

- Binary
- IDA Pro
  - Parse Binary
  - Build CFGs
- Connector
  - Memory Analysis
  - BREW
    - Rewrite
  - Generate Code
- Codesurfer
  - Build SDG
  - Browse
- Clients
  - Detect Malicious Code
  - Detect Buffer Overrun
  - Build Program Specification
- Generated Binary

Feb 2005, ONR Review
Static Buffer Overflow Detection

Binary → IDA Pro
- Parse Binary
- Build CFGs

Connector
- Memory Analysis
- Build SDG

BREW
- Rewrite
- Generate Code

Codesurfer
- Parse C
- Build SDG
- Browse

Source Code

Clients
- Detect Malicious Code
- Detect Buffer Overrun
- Build Program Specification

Generated Binary
Team

• Somesh Jha
  - Analysis of malicious code, intrusion detection, verification of security protocols, and trust management

• Bart Miller
  - Distributed computing, kernel instrumentation, intrusion detection

• Tom Reps
  - Static-analysis techniques, trust management, and model checking
Six Graduate Students

- Gogul Balakrishnan
- Mihai Christodorescu (US citizen)
- Vinod Ganapathy
- Jon Giffin (US citizen)
- Shai Rubin (Prelim)
- Hao Wang (US citizen)
- Louis Kruger heavily interacts with our group

Summary
- Three US citizens
  - All are Ph.D. students and have passed their qualifiers
  - Three students have passed their prelims
Problem: Cross-Domain Authorization (Hao Wang)

Q1: Should Alice be allowed to access R in domain UW?
Q2: If so, prove it!

Centralized Solution

- Assume a centralized authority
- Does not deal with privacy concerns
Solution: Distributed Certificate-Chain Discovery Using WPDS

- Developed a distributed algorithm to solve the cross-domain authorization problem
  - Apply techniques from a well-studied domain (WPDS) to a new problem domain (SPKI/SDSI)
- Addressed shortcomings of existing approaches
  - Distributed algorithm \( \Rightarrow \) No need for centralized authority
  - Preserve users’ privacy
- Implemented a prototype
  - Scalable—tested in a simulated environment with up to 1,600 certificates
Model-Based Intrusion Detection (Jon Giffin)

- Detect deviations from model of normal system-call execution behavior
- Context-sensitive data-flow analysis for system-call argument recovery
Environment-Sensitive Models

- Program execution often depends upon data values in execution environment
- Environment-sensitive models restrict allowed execution given environment values
Discovering API-Level Exploits (Vinod Ganapathy)

- **Key concept**: Exploit-finding is different because of need to **model low-level details**.
- **Benefits**: Improved vulnerability-detection.
  - Exploitable vulnerabilities versus **false alarms**.
  - Capability to find **variants** of exploits.
- **Case study**: Format-string exploit-finding tool.
  - Finds exploits against **real-world** applications.
Security Testing

1. **NIDS**
   - Problem: Find an attack instance that eludes a NIDS.
   - Solution: Attack generation using natural deduction.
     - Shai Rubin · Somesh Jha · Bart Miller

2. **Virus scanners**
   - Problem: Generate virus sample that evades AV tool.
   - Solution: Guided attack generation using oracle access.
     - Mihai Christodorescu · Somesh Jha
Challenges in Static Analysis of x86 Executables
(Gogul Balakrishnan)

- Distinguishing between code and data
- Identifying variables

- Identifying parameters
- Resolving indirect jumps
- Resolving indirect calls
- Identifying may-aliases

Binary

IDA Pro
  Parse Binary
  Build CFGs

Connector
  Value-set Analysis

CodeSurfer
  Build SDG
  Browse

Security Analyzers
Decompiler
Binary Rewriter
User Scripts
Challenges in Static Analysis of x86 Executables (Gogul Balakrishnan)

- Distinguishing between code and data
- Identifying variables
- Identifying parameters
- Resolving indirect jumps
- Resolving indirect calls
- Identifying may-aliases

Binary

IDA Pro
  - Parse Binary
  - Build CFGs

Connector
  - Value-set Analysis

CodeSurfer
  - Build SDG
  - Browse

Security Analyzers
Decompiler
Binary Rewriter
User Scripts
• **Research Papers**
  - 18 papers accepted in major conferences (USENIX Security, Oakland, CCS, NDSS, CSFW, ISSTA, ICSE)
  - 3 best paper awards
  - > 10 related publications

• PIs served on several program committees and reviewed for several journals

• See the overview document for details
• Developed a significant infrastructure for analyzing and rewriting x86 binaries
  - Collaboration with GrammaTech
• Applicable to several research problems
  - Identifying buffer overruns
  - Malicious code detection
  - Protection, event logging, remediation..
• Created many technology-transfer and collaborative opportunities
• **WiSA infrastructure**
  - Discovering buffer overruns
  - Malicious-code detection
  - Constructing models for intrusion detection
  - Many more under development ...

• **SandboX86**
  - Sandbox applications using a security policy
  - Discovering spyware features in unknown applications

• **Our analysis techniques do not require access to source code**
  - Can be readily applied to COTS software

• **Reduces risk of deploying COTS**
• **GrammaTech (GT)** an important vehicle for technology transfer

• **GT -> UW**
  - GT implemented an important piece of the architecture

• **UW -> GT**
  - Value-set analysis (**Gogul**)  
  - BREW infrastructure (**Jon, Mihai, and Hao**)  
  - Buffer-overrun-detection tool (**Vinod**)
• Starting to explore collaborative opportunities with Sandia National Laboratories
  - System Assessment and Research Center
• Doug Ghormley from Sandia came and gave a talk
• Louis Kruger (UW) is a summer intern at Sandia
  - Working on using BREW for “classified” applications
  - Will give a talk about this
Contact Information

- **Prof. S. Jha**
  - email: jha@cs.wisc.edu
- **Prof. B. Miller**
  - email: bart@cs.wisc.edu
- **Prof. T. Reps**
  - email: reps@cs.wisc.edu

- **Computer Sciences Dept.**
  - 1210 West Dayton Street
  - Madison, WI 53706

Project home page
http://www.cs.wisc.edu/wisa