Secure Software Programming and Vulnerability Analysis

Christopher Kruegel chris@auto.tuwien.ac.at
http://www.auto.tuwien.ac.at/~chris

Administrative Issues

- Mode
  - lectures and paper discussions
  - small programming assignments (but only a few)
  - written final (end of June)
- Dates
  - Monday 2pm. - 3pm. and Wednesday 5pm. - 6pm.
  - HS 13 Ernst Melan
  - no class next Monday (26.04.04)
- Slides and News
  - available under http://www.auto.tuwien.ac.at/~chris

Topics

- Introduction
- Linux features and TCP/IP tutorial
- Architectural issues
  - interface design and privilege separation
  - input validation
  - race conditions
  - denial of service
- Implementation issues
  - stack overflow
  - heap overflow
  - miscellaneous problems (e.g., format strings, integer overflows)
  - source code auditing tools
- Operational issues
  - system management and patching

Introduction

Overview

- Security issues at various stages of application life-cycle
  - mistakes, vulnerabilities, and exploits
  - avoidance, detection, and defense
- Architecture
  - security considerations when designing the application
- Implementation
  - security considerations when writing the application
- Operation
  - security considerations when the application is in production

Overview

- Architecture and design
  - validation of requirements (building the right model)
  - verification of design (building the model right)
- Common problems
  - authentication and privileges
    - session reply
    - principle of least privilege
  - communication protocol design
    - sniffing, man-in-the-middle
    - session killing, hijacking
  - parallelism and resource access
    - race conditions
  - denial of service

Overview

- Implementation
  - verification of implementation
  - classic vulnerabilities (often programming language specific)
- Common problems
  - buffer overflows
    - static (stack) overflows
    - dynamic (heap) overflows
  - input validation
    - URL encoding
    - document root escape
    - SQL injection
  - back doors

Overview

- Operation
  - decisions made after software is deployed
  - often not under developer’s control
- Common problems
  - denial of service (DOS)
    - network DOS
    - distributed DOS, zombies
  - administration problems
    - weak passwords
    - password cracking
    - unsafe defaults

Terminology

- What is an attack?
  - no easy answer, it depends
- Security Policy
  - The framework within which an organization establishes needed levels of information security to achieve the desired integrity, confidentiality, and availability goals. A policy is a statement of information values, protection responsibilities, and organization commitment for a system.

    (US Congressional Office of Technology)
  - A set of guidelines defining what you want to protect and what you want to allow at your site.

Terminology

- What you want to protect?
  - defines assets
- What are the goals of your protection efforts?
  - Integrity
    - property that data has not been altered or destroyed in an unauthorized manner
  - Confidentiality
    - property that information is not made available or disclosed to unauthorized individuals, entities or processes
  - Availability
    - property of being accessible and useable upon demand by an authorized entity

Terminology

- What do you want to protect against?
  - threat model
  - risk analysis
- Different security policies
  - bank answers questions different than home user
- Attack
  - any maliciously intended act against a system or a population of systems
  - any action that violates a given security policy

Insecure Software

or, why good people write bad code

- Technical factors
  - complexity of task
- Economic factors
  - deadlines
  - insufficient funding
- Human factors
  - mental models
  - social factors

Technical Factors

- Complexity
  - algorithmic complexity
  - parallel processes, threads
  - multi-user
  - indeterminism
- Composition
  - incorrect assumptions
  - surprising interactions
  - example: rlogin -l -froot
- Changes
  - consequences are hard to predict
  - example: Sun tarballs

Economic Factors

- Production pressure
  - not enough time
  - not enough manpower for testing
- Security is not a feature
  - just secure enough
- Open-source vs. closed-source debate
  - open-source is peer-reviewed
  - closed-source is written by professionals
- Legacy software

Human Factors

- Poor risk assessment
  - invisible enemy
- Mental models
  - only check for errors that are understood
  - assume software is used for a specific task
    example: mouse driver exploit

Improvement

- Tools
  - detect mistakes and vulnerabilities
  - support programmer
  - formal verification
- Standards and metrics
  - hold vendors accountable
  - allow for comparison between products
- Education
  - that’s why we are here