CS390S, Week 1: Introduction to Secure Programming

Pascal Meunier, Ph.D., M.Sc., CISSP
January 10, 2007
Developed thanks to support and contributions from Symantec Corporation, support from the NSF SFS Capacity Building Program (Award Number 0113725) and the Purdue e-Enterprise Center

Copyright (2004) Purdue Research Foundation. All rights reserved.
Learning Objectives

- Be able to define and use common words and concepts relating to secure programming
- Be able to name, tell apart and utilize secure programming standards and resources
- Understand why emphasis on secure programming in the first place is better than relying on patch management
Contents

- Problem statement
- Definitions
- Resources
- Why education in secure programming is better than patches
Axiom: Costs of Insecurity Increase with Time

- More complex, networked, hosted software
  - Web applications (e.g., Google Apps)
- Delayed security costs more to produce
  - Early security (design) costs less than late security (patches, announcements, identity theft, blackmail and ransoms, consultants)
- Business risks increase
  - New and more severe laws (liability)
  - Competitors with better security may be preferred
Software Security Defined

- Reliable software does what it is supposed to do.
- Secure software does that... and nothing else.
  - Ivan Arce
- How do we get there?
Dubious Propositions

- “What works in my day-to-day life should also work for the software I make”
- “Blocking known bad things from happening is good enough”
- “We’ll just keep that secret”
- “We can always patch it later”
Vulnerability

- The consequence of a flaw in a system, that allows a policy to be violated
- Example policy: the content on a web site is restricted to authenticated users.
- Flaw: the web site relies on JavaScript to be executed on the client browser for access control
- Vulnerability: JavaScript needs to be disabled
- An abundance of vulnerable sites exist
Exploit

- An exploit is the act of exercising a vulnerability
  - Exploit: Disable JavaScript and access the vulnerable website

- Also used to refer to an actual program, binary or script that automates an attack
Latent Vulnerabilities

- Sometimes, a vulnerability can be protected by a change that leaves the vulnerable code in place:
  - some change external to the application (firewall)
  - a configuration change (disabling an option)
  - a code wrapper that blocks exploit attempts

- A vulnerability that is not exploitable at the moment is a latent vulnerability
Potential Vulnerabilities

- Bad practices, quality defects and other flaws that could result in vulnerabilities in a different code context are potential vulnerabilities.
Exploitability

- Difficult to establish whether a vulnerability is exploitable, latent or potential in complex systems.
- A latent or potential vulnerability can become exploitable when:
  - The software is used in a different context sometime after its design.
  - The configuration is changed.
  - The code is changed.
  - Someone thinks of something you didn't.

- Is a memory leak exploitable?
  - It depends!
Exposure

- An exposure is an information leak that may assist an attacker.

- Examples:
  - Software identification and version number released when connecting to a service, which may be used to select the most effective attack.
  - When web pages display SQL error messages
  - When an IT person is having trouble (e.g., with their firewall) and posts questions to public mailing lists with their company's email address
  - Sun's "tar" utility disclosed part of the password file
Exercise

- Identify other examples of information leaks that may assist an attacker
Exercise Sample Answers

- Identify other examples of information leaks that may assist an attacker
  - Finger may release information about who is online, e.g., administrators
  - Source code leaks (especially if the code contains vulnerabilities)
  - Directory listings
  - Wireless networks that broadcast their existence
A security objective is a high-level description of what the program or system must accomplish.

- Federal regulations drive many of these objectives
  - HIPAA (Health Insurance Portability and Accountability Act)
  - etc...
- Examples:
  - all money transfers must be legal
  - the system must pass EAL4 Common Criteria certification
Security Requirements

- **Functional**
  - Logs
  - Access control
  - Intrusion detection

- **Assurance**
  - Methodology used to create the software
  - Programmer training (secure programming)
  - Source code audit for vulnerabilities
  - Tools used to find vulnerabilities
  - Architecture providing security advantages
Policies specify which activities, states and processes are allowed.

- Examples:
  - All users must be authenticated
  - Money transfers can only be requested by the account owner

Also refers to security models that specify rules

Famous policies (e.g., see Bishop 2002):
- The Bell-LaPadula confidentiality model
- The Biba integrity model
- The Clark-Wilson integrity model
Resources: Why?

- For insight into how vulnerabilities get tracked
- For situational awareness
  - Be ready to answer queries from customers who also saw that information
  - Get notification of vulnerabilities pertinent to your product
    - As a backup (should be rare)
    - The situation where developers learn first about a vulnerability through public sources should be covered in an organization's policy
Resources (Cont.)

- To proactively prevent vulnerabilities in your product by being informed about vulnerabilities in other products
  - Learn from other people's mistakes
- For reference
- For additional sources on best programming and software engineering practices
  - So you can grow and learn more about secure programming on your own
  - For other examples and ideas
• MITRE

  ▪ CVE (cve.mitre.org)
    – Common Vulnerability and Exposures
    – Unique identifier for a vulnerability
  ▪ CWE (cwe.mitre.org)
    – Common Weakness Enumeration
      ✴ Relevant CWE identifiers will be provided in this course
  ▪ CCE (cce.mitre.org)
    – Common Configuration Enumeration
      ✴ Some configurations are more vulnerable or provide exposures. What can you tweak (a.k.a. “harden”)?
Exercise

- Point your browser to cve.mitre.org
- What is the number of the first vulnerability in 2007?
  - Make sure to type "2007-0001" with the correct number of zeros!
- What operating system was involved in the first vulnerability of 2007?
- What stage is it in?
- Search for vulnerabilities in products from a company you know
  - Look at the entries returned, and the CVE web site FAQs.
NIST

- National Institute of Standards and Technologies
- Publishes many security guides and standards
- NVD: National Vulnerability Database
  - Keeps track of which version of which product is affected by a CVE
- CVSS: Common Vulnerability Scoring System
  - How severe is a vulnerability?
NIST Security Documents

  
  - SP 800-101 Guidelines on Cell Phone Forensics, May 2007
  - SP 800-64 Security Considerations in the Information System Development Life Cycle, October 2003

And many others...
CVSS

- Score from 0-10
- Three groups of scoring criteria:
  - Base
  - Temporal
  - Environmental
- Payment Card Industry (PCI) Security Standards Council
  - Vendors must be “compliant” based on CVSS scores of known vulnerabilities
  - Companies can certify vendor compliance (scan services)
Exercise

- Do a search for vulnerabilities in Adobe Acrobat reader in the NVD
  - How many entries are there?
  - What is their CVSS score?
  - How did the latest vulnerability happen?

- Go to the statistics section of the NVD.
  - How many vulnerabilities have been found every year?
  - Is there a trend?
CERT

- Computer Emergency Response Team
  - You can report vulnerabilities to them
- Secure coding issues
  - News and presentations
  - Software (libraries)
  - http://www.cert.org/secure-coding/
- Secure Coding web site
  - “secure coding standards”
  - http://www.securecoding.cert.org/
US-CERT

- http://www.us-cert.gov

- Department of Homeland Security
  - "Technical Security Alerts"
  - "Non-technical Security Alerts"
    - e.g., "Understanding Firewalls", like a "Firewalls for dummies"
  - Security Bulletins
  - Security Tips
  - BuildSecurityIn: Best practices
    - https://buildsecurityin.us-cert.gov/daisy/bsi/home.html
Other Resources

- OWASP
  - Webgoat, a web application that is vulnerable by design
- Security Focus
- Secunia
- SANS
  - SPSA: The Secure Programming Skills Assessment
    - http://www.sans-ssi.org/
Flaws get discovered by different people at different times

- Birthday: how many people can discover the same flaw in the same period of time?

You don't know how many flaws a given software product may have, and if others have discovered them

But... You may have assurances that it was designed and implemented properly by qualified (educated) people, using tools, methods and processes that have historically yielded few defects

- The goal of this course is education
Risks to Confidentiality, Integrity and Availability

- Confidentiality is threatened when information can be revealed in violation of a policy.
  - Examples: eavesdropping and inadequate access control.
- Integrity is threatened when information can be manipulated by an attacker.
  - Example: "man-in-the-middle" attack.
- Availability is threatened when a resource can be disabled or made unavailable.
Example

- An FTP server is read-only. If passwords are sent in clear text, what is threatened if transmissions are captured?
  - Confidentiality of the passwords
    - Confidentiality of the documents on the FTP server
    - Confidentiality, Integrity and Availability of other resources that use the same password!
People Want Software That:

- Is produced with security assurance
- Lowers security risks
  - To comply with laws mandating low security risks
    - HIPAA
    - GLBA (Gramm-Leach Bliley)
    - FERPA
  - To protect trade secrets and other valuable company information
- Has fewer maintenance headaches (patching) and costs
- Protects their reputation
Exercise

- Identify risks that would cause you to stop using a product. Be specific.
Exercise Example Answers

- Frequencies of vulnerabilities and patches
- Absence of patches (or slow turnover) for known issues
- Severity of vulnerabilities
- Criticality of the application
- Unreliability of patches
  - Patches that break previous fixes
  - Patches that are incompatible with other software
  - Downtime while applying patches
- Unreliable file systems (non-journaled)
Your motivation as a participant in software development

- How important is quality?
  - Quality assurance is inclusive of secure programming techniques

- How much design?
  - Information assurance happens by design

- How risk-averse?
  - Security problems in your projects and code can hurt your reputation as well as your employer's
Patches

- Flaws pose risks until they have been fixed everywhere in the field
  - Very difficult to do correctly and swiftly

- Reasons to rely on patching
  - Attempt to delay or avoid immediate costs and opportunity costs
    - Hope that flaws are never discovered
  - Incompetence
    - We have no idea if this architecture/design/software could be vulnerable in these conditions, and we did nothing to find out
Cost of Patching

- Cost of evaluating vulnerability claims
- Cost of patch development and testing
- Cost of patch notification and download system
- NIST recommendation on applying patches (s.p. 800-40)
  - Patch and Vulnerability Group (customer's cost)
    - test patches
    - notify administrators
    - monitor application of patches by system administrators
- Vulnerability scanning to verify or enforce
Question

- Patches incur costs to?
  b) the vendor
  c) the customer
  d) both
Question

- Patches incur costs to?
  b) the vendor
  c) the customer
  d) both
Cost of Patching vs Preventing Flaws

- Flaws in the Software Development Lifecycle
  (assuming for the purposes of this discussion that this division is correct and unambiguous, which it is not)
  - Design and architecture
  - Implementation
  - Operations
- Fixing security bugs with a patch costs 60 times more than catching them at design time*

If it costs $100,000 to issue each security patch, approximately how much could have been saved by correcting the problem at design time?

b) $9,800

c) $98,000

d) $980
Question

- If it costs $100,000 to issue each security patch, approximately how much could have been saved by correcting the problem at design time?
  
b) $9,800
  
c) $98,000
  
d) $980

Note: It isn’t possible to catch all security flaws at design time.
Current Results

- Several thousand vulnerabilities reported each year
- About 50% of vulnerabilities are commonly repeated mistakes
- About 25% of vulnerabilities could be avoided by applying secure design principles at design time
- Need new methods
- Patches created using the same development methods that created the buggy software, are likely buggy themselves!
Question

- Approximately what percentage of documented vulnerabilities are common repeated mistakes?
  b) 25%
  c) 50%
  d) 75%
Approximately what percentage of documented vulnerabilities are common repeated mistakes?

b) 25%
c) 50%
d) 75%
How much money does a developer for a large software project typically save a company when catching and fixing a vulnerability during development instead of patching?

b) $1,000
c) $10,000
d) $100,000

A vulnerability fixed before release may take one hour, compared to weeks of several people's time to fix it after release.
Answer

- c) $100,000

- Without counting
  - 1) costs to customers
    - Especially if revenue-generating activities are interrupted!
  - 2) intangible costs
Patches Don't Work Well

- Vendors can take a long time to issue patches
  - Risk: Malicious researchers can find the vulnerability independently
    - More and more likely to happen due to the "birthday effect"

- People are late applying patches
  - Afraid of breaking working mission critical things
  - Requires time, testing
  - Difficult to verify compliance in a large federated enterprise

- Patches give excellent information to attackers

- Patches don't repair exploited systems
Today's Challenge

- Vendors, Academia and Government need as many people trained in secure programming as possible
  - That means you ;)
- Race between "good" and "bad" guys in finding vulnerabilities first isn't good
  - Vulnerability auctions: incentive for vulnerabilities to stay secret, and secret vulnerabilities tend to never get fixed
- Need to produce software with as few security issues as possible right from the start
Next Week:

- Principles of secure programming
- Software assurance primer
Motivation and Definitions: End