James E. Molini, CISSP, CSSLP
Microsoft
Member, (ISC)² Advisory Board of the Americas
jmolini@microsoft.com
http://www.codeguard.org/blog

Software Development: The Next Security Frontier

The Changing Landscape of Security

- De-perimiterization of networks places more burden on the security of individual machines and applications
- Malware increased by 200-300% over the past year
- More incidents of data loss could result in greater government oversight and regulation
  - 38 out of 50 states in US have now enacted breach disclosure laws
- 2008 (ISC)² Global Information Security Workforce Study (GISWS) report found significant costs result from data breaches
  - **US $50 to $200 per record lost** (not including reputation damage and loss of trust)

Software Vulnerabilities:
Opening the Door to Criminals

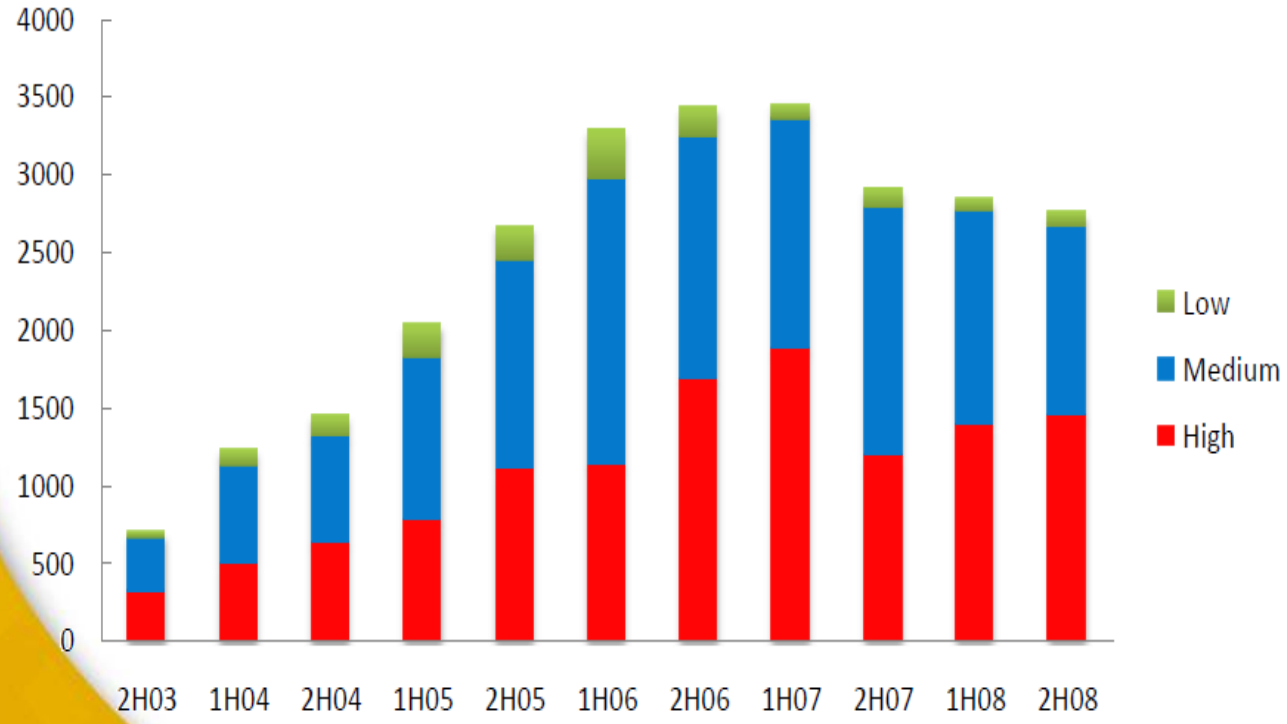
- XSS Attacks (Ongoing)
  - Cross Site Scripting (XSS) is becoming the new "buffer overflow"
  - In 2007, XSS accounted for 80% of documented vulnerabilities
  - OWASP site recommends proper web site coding practices

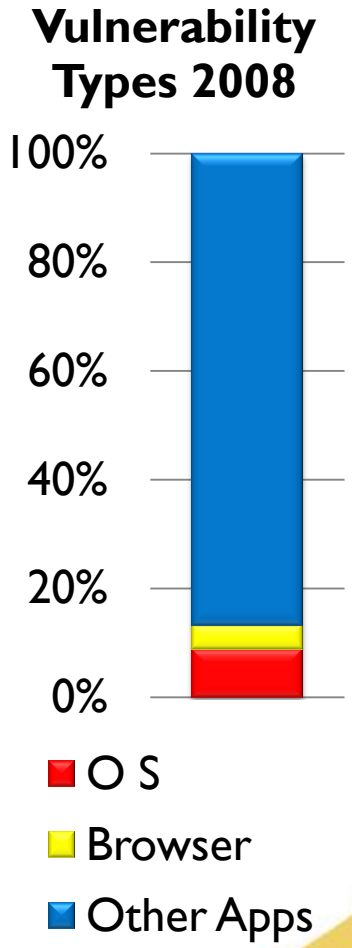
- SQL Injection Attacks (Ongoing)
  - Recently several security sites were attacked using this technique
  - Data entry fields on websites are loaded with SQL commands
  - Bypasses the firewall and many web gateways
  - Input validation reduces the exposure from this attack (see OWASP Notes on SQL Injection)

- Recent worms exploit patching latency
  - Conficker worm released 1 month after the patch from Microsoft
  - This exposes a flaw in patch management practices

Vulnerability Rates 2003 - 2008

Figure 1. Industry-wide vulnerability disclosures by CVSSv2 severity, by half-year, 1H03–2H08





Taken from Microsoft Security Intelligence Report: Volume 6. Jul – Dec 2008

What Is Software Security?

- Security is a distinct property of a software system or application. It is composed of Confidentiality, Integrity, Availability, Authenticity, and other related attributes*.

- Software Security vs. Secure Software
  - Secure software can be delivered by rigorously applying all the techniques of a software security plan
- Software Security vs. Secure Coding
  - Secure coding is one aspect of an overall software security plan
- Software Security vs. Software Quality
  - High quality software can also be insecure (e.g. medical device software)
  - Security requires specialized skills

*Definition derived from description provided in Software Assurance BoK from DHS.

Can't We Just Learn How to Write Secure Code?

- Many eyeballs won't solve the security problem. (e.g. recent DNS bug took 10 years to discover)
- If you have seen a SQL Injection attack, or a Buffer Overflow consider this:
  - The IDE should have enforced ASLR for all production code
  - The requirements document should have specified composition parameters for all incoming data
  - The design should have defined an input validator that rejected input which failed to meet specifications
  - The code should have called validation routines for all external data requests
  - The tester should have included injection attacks and fuzzing as part of the testing process

Can Secure Systems Really Prevent Intrusions?

- Two Firewalls. Two manufacturers. Two development methodologies.
- One was based on a Trusted OS & Security Development Lifecycle.
- One was not.

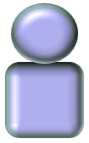
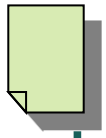
	1999	2000	2001	2002	Total
Firewall-A	3	15	10	4	**32**
Firewall-B	0	0	0	0	**0**

- Vulnerabilities listed by US Natl. Vulnerability Database: 1999-2002

To perform your own search, visit: http://web.nvd.nist.gov/view/vuln/statistics?execution=e2s1

COMMON ELEMENTS OF A SOFTWARE SECURITY PROGRAM

Overview

- Security must become an integral process throughout the SDLC
- Software security requires:
  1) Technology – Layered and maintained
  2) Process -- targeted and structured
  3) People -- trained and qualified (first line of defense and organization's most critical asset)
- Employ  people,  process, &  technology to get the job done.

Secure Software Concepts

- Confidentiality, Integrity, Availability Authentication, Authorization, and Auditing
- Security Design Principles
- Risk Management (e.g., vulnerabilities, threats and controls)
- Regulations, Privacy, and Compliance
- Software Architecture (e.g., layers)
- Software Development Methodologies
- Legal (e.g., Copyright, IP and trademark)
- Standards (e.g., ISO 2700x, OWASP)
- Security Models (e.g., Bell-LaPadula, Clark-Wilson & Brewer-Nash)
- Trusted Computing (e.g., TPM, TCB)
- Acquisition (e.g., contracts, SLAs and specifications)

Getting Started

- Training and Awareness
  - Start with basic concepts
  - Train developers and testers first
  - When you train – train well.
- Appoint or hire a Security Lead
  - Becomes local authority on software security
  - Coordinates security activities and drive SDL
  - Establishes risk management process
  - Consider certification for security specialists

Advice:
~ RoleSeparation() { …}

- Security process decisions must be conscious risk decisions.
- Do not let the development group separate security off into another segment of the process.
- Every subsystem team and every developer must have a coherent understanding of the security design
- Tuning and testing teams must understand security requirements

Secure Software Requirements

- Policy Decomposition
  - Confidentiality, Integrity, Availability Requirements
  - Authentication, Authorization, and Auditing Requirements
  - Internal and External Requirements
- Identification and Gathering
  - Data Classification
  - Use Cases
  - Abuse Cases (inside and outside adversaries)

Secure Software Requirements: Getting Started

- Create a baseline for security
- Build boilerplate requirements for use in new projects
- Understand how requirements differ for:
  - In-house development
  - Product Development
  - Software Acquisition
- Develop common abuse cases
- Begin Risk Management Process
  - Threat Model Development
  - Feature/Component Risk Analysis

Secure Software Design

- Design Processes
  - Attack surface evaluation, Threat modeling, Control Identification, Control prioritization
- Design Considerations
  - Confidentiality, Integrity, Availability, Authentication, Authorization, and Auditing
  - Security design principles, Interconnectivity, Security management interfaces, Identity management
- Architecture
  - Distributed, Service-oriented, Rich Internet applications, Pervasive computing
  - Integration with existing architectures
  - Software as a Service
- Technologies
  - IAM, Audit, DRM, Flow control (e.g., proxies, firewalls, middleware)
  - Data protection (e.g., DLP, encryption and database security)
  - Computing environment (e.g., programming languages, virtualization, and operating systems
  - Integrity (e.g., code signing)

Secure Software Design: Getting Started

Saltzer & Schroeder: Security Design Principles

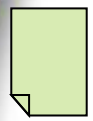
- Economy of mechanism
- Fail Safe Defaults
- Complete Mediation
- Open Design
- Separation of Privilege
- Least Privilege
- Least Common Mechanism
- Psychological acceptability

Secure Coding: Key Concepts

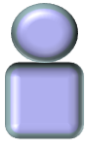
- Declarative versus programmatic security (e.g., bootstrapping, cryptographic agility, and handling configuration parameters)
- Common software vulnerabilities and countermeasures
- Defensive coding practices (e.g., type safe practices, locality, memory management, error handling)
- Exception management
- Configuration management (e.g., source code and versioning)
- Build environment (e.g., build tools)
- Code/Peer review
- Code Analysis (static and dynamic)
- Anti-tampering techniques (e.g., code signing)
- Interface coding (e.g., proper authentication and third party API)

Secure Coding: Getting Started

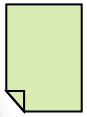
- Never build your own crypto or authentication mechanisms



- Develop a list of banned functions



- Train developers to avoid most common flaws



- Develop with least privilege

Secure Software Testing: Key Concepts

- Testing for Security Quality Assurance
  - Functional Testing (e.g., reliability, logic, performance and scalability)
  - Security Testing (e.g., white box and black box)
  - Environment (e.g., interoperability)
  - Bug tracking (e.g., defects, errors and vulnerabilities)
  - Attack surface validation
- Test types
  - Penetration Testing
  - Fuzzing, Scanning, Simulation Testing (e.g., environment and data)
  - Testing for Failure
  - Cryptographic validation (e.g., environment and data)
- Impact Assessment and Corrective Action
- Standards for software quality assurance (e.g., ISO 9126, SSE-CMM and OSSTMM)
- Regression testing

Secure Software Testing: Getting Started

- Use security testing tools to discover common vulnerabilities.
- Implement static analysis testing for all Internet facing code.
- Add security bug categories to the bug tracking system

Secure Software Acceptance & Deployment: Key Concepts

- Pre-release or pre-deployment
  - Completion Criteria (e.g., documentation, BCP)
  - Risk Acceptance
  - Documentation (e.g., DRP and BCP)
- Post-release
  - Validation and Verification (e.g., Common Criteria)
- Independent testing (e.g., third-party)
- Installation and Deployment
  - Bootstrapping (e.g., key generation, access management)
  - Configuration Management (e.g., elevated privileges, hardening, platform change)

Secure Software Acceptance & Deployment: Getting Started

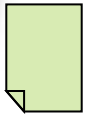
- Develop an official security signoff during release
- Define rules for software security acceptance
- Implement a security documentation standard

Secure Software Operations & Maintenance: Key Concepts

- Operations and Maintenance
  - Monitoring (e.g., Metrics and Audits)
  - Incident Management
  - Problem Management (Root Cause Analysis)
  - Patching
- End of life policies

Secure Software Maintenance: Getting Started

- Implement patch security testing and delivery mechanisms



- Develop a Security Response Plan for software vulnerabilities

Security Lifecycle Results

Figure 7. Top 10 browser-based vulnerabilities exploited on computers running Windows XP, 2H08

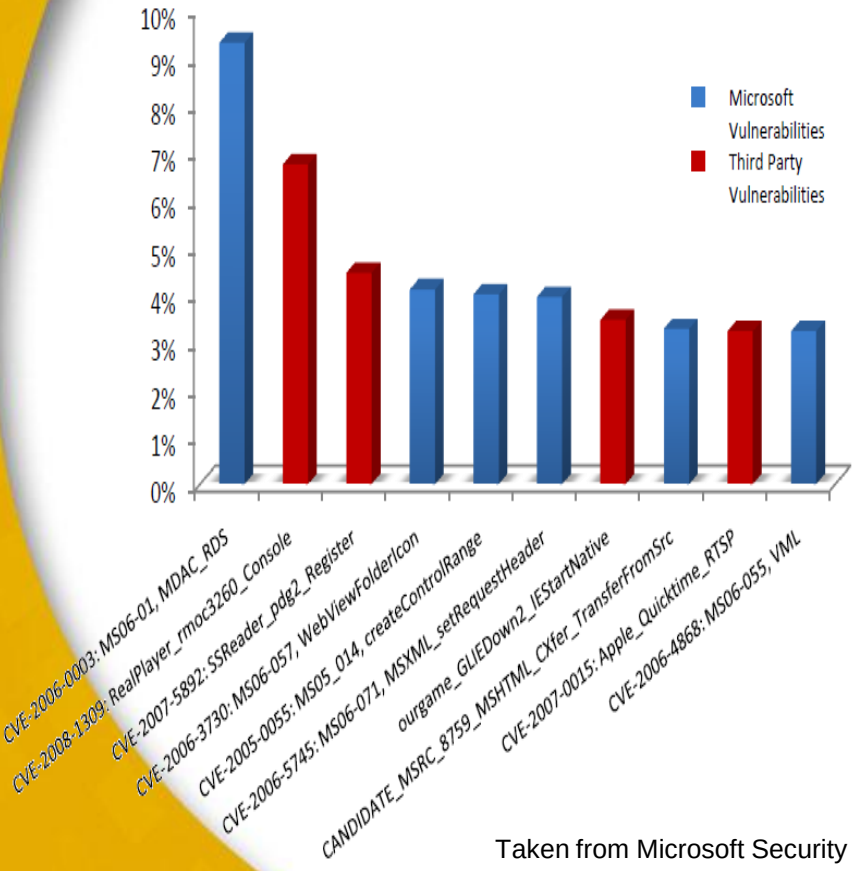
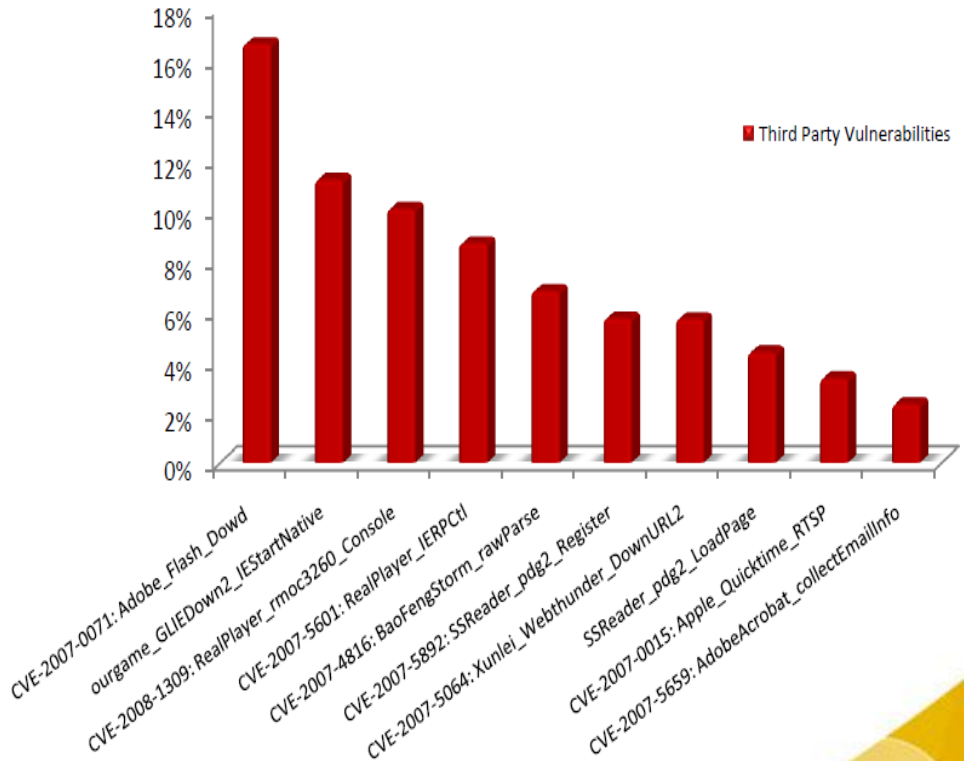


Figure 8. Top 10 browser-based vulnerabilities exploited on computers running Windows Vista, 2H08



Taken from Microsoft Security Intelligence Report: Volume 6. Jul – Dec 2008

Download the report at: http://www.microsoft.com/ downloads/details.aspx?FamilyID=aa6e0660-dc24-4930-affd-e33572ccb91f&displaylang=en

The Security Imperative

- We must move from vulnerability management to security engineering
  - Identify security needs up front in the concepts and requirements stage
  - Define adequate security controls in the design stage
  - Code appropriately to reduce security bugs
  - Test for failures in the process
  - Maintain code with security in mind
  - Participate in community activities (e.g. (ISC)^2, OWASP, SANS

What You Can Do

- Get involved in the software development security process now.
- Support one of the active independent organizations that is advocating improvements to the discipline:
  - (ISC)2, ISSA, OWASP, SANS.
- Educate software developers and managers regarding the need for adequate risk management.
- Promote better software security practices throughout the SDLC, including design, testing deployment, and ultimately disposal of software.

For more on software security, visit: http://www.codeguard.org/blog

What is CSSLP^{CM}?

- Certified Secure Software Lifecycle Professional (CSSLP)
- Base credential (no other certification is required as a prerequisite)
- Professional certification program
- Takes a holistic approach to security in the software lifecycle
- Tests candidates competency (KSAs) to significantly mitigate the security concerns
- Purpose
  - Addresses building security throughout the entire software lifecycle – from concept and planning through operations and maintenance, to the ultimate disposal.
  - Provides a credential that speaks to the individual's ability to contribute to the delivery of secure software through the use of standards and best practices.
  - The target professionals for this certification includes all stakeholders involved in the Software Lifecycle.

CSSLP^{CM} Industry Supporters

- Microsoft
- Cisco
- Xerox
- SAFECode
- Symantec
- BASDA
- SANS
- DSCI (NASSCOM)
- SRA International
- ISSA

"As the global dependence on information and communications technology has grown, users have become increasingly concerned over the security of software, especially those in the government, critical infrastructure and enterprise sectors. By offering software professionals a means to increase and validate their knowledge of best practices in securing applications throughout the development lifecycle, (ISC)²'s CSSLP is helping the industry take an important step forward in addressing the 'people' part of the solution."

Paul Kurtz, executive director, SAFECode

CSSLP^{CM} Certification Requirements

By Examination:

- Process
  - The first public exam will be held at the end of June 2009
  - Candidate must submit:
    - Completed examination registration form
    - Proof of 4 years experience in the Software Development Lifecycle (SDLC) or 3 years experience with a one year waiver for 4-year degree or equivalent in an IT related field
    - Pay a Fee of $549 early-bird or $599 standard
  - Candidate must
    - Pass the official (ISC)²® CSSLP certification examination
    - Complete the endorsement process
  - The Associate of (ISC)² Program applies to those who have passed the exam but need to acquire the necessary minimum experience requirements

For more information, please contact:

- Glenn Johnson, (ISC)² , Certification Consultant
  - gjohnson@isc2.org
- Vehbi Tasar, (ISC)² Manager of Professional Programs
  - vtasar@isc2.org

Visit www.isc2.org/csslp

References

- *Secure Software Assurance: A guide to the Common Body of Knowledge to Produce, Acquire, and Sustain Secure Software*, S. Redwine, Ed., US Department of Homeland Security, 2005.
- *The Trustworthy Computing Security Development Lifecycle*, S. Lipner, et al, Microsoft, March 2005. http://msdn.microsoft.com/en-us/library/ms995349.aspx
- OWASP: http://www.owasp.org/index.php/Main_Page
- Microsoft Security Site for Developers: http://msdn.microsoft.com/en-us/security/default.aspx
- Books:
- The Security Development Lifecycle, M. Howard & S. Lipner Microsoft Press, 2006