

Application Security

**Attackers Won't Stop at the Firewall
(Why should you?)**

**Kenneth H. Newman, CISM
Certified Information Security Manager**



Disclaimer: Don't believe a word I say...

Who am I?



- 10 years in information security (in financials)
 - Deutsche Bank
 - Citigroup
 - Bank of Hawaii
- Leading a variety of security functions
 - Risk Management and Consulting
 - Engineering (Security Products)
 - Operations (Account Administration and Monitoring)
 - Education and Awareness
- Prior PC and network management work

Credits



- @Stake
 - Secure coding training
- E&Y
 - Application penetration tests
- Cenzic
 - Security Q&A (fault injection) testing

What is 'Application Security'?



- It includes many concepts:
 - Authentication
 - Identity Management
 - Single Sign On
 - Authorization
 - Encryption
 - Logging and Monitoring
 - Administration
 - Configuration Management
 - Change Management
 - Penetration Testing
- We will focus on three main areas:
 - Crafting a framework to define requirements
 - Introducing controls to allow compliance
 - Providing awareness to enhance understanding

What is a Bad Application?



- Exposes private data to unauthenticated users
- Crashes or halts after unexpected inputs
 - Denial of service
- Allows someone to inject and execute code
- Allows access to privileged commands
- Leaves transient information behind

For Example



- ZoneAlarm personal firewall (overwriting memory)
- McAfee ePolicy Orchestrator (arbitrary code execution)
- Password Safe open-source tool (clear text in memory)
- Microsoft (six months after Slammer...)
 - MSBlast.RPC worm exploits recent RPC vulnerability
- Linux and *BSD kernel audit presented at DC11
 - Hundreds of common flaws found in what should be secure code

Why Aren't Applications Secure?



- Extremely short development cycles
 - Security is seen as a delay; not a 'value add'
- Lack of training/awareness for developers
 - Not expected to understand security
- Network security limitations
 - Firewalls blind to application level traffic
 - Network IDS detective not preventative
- Growth of pervasive technology
 - Web Services 'transparent' by design

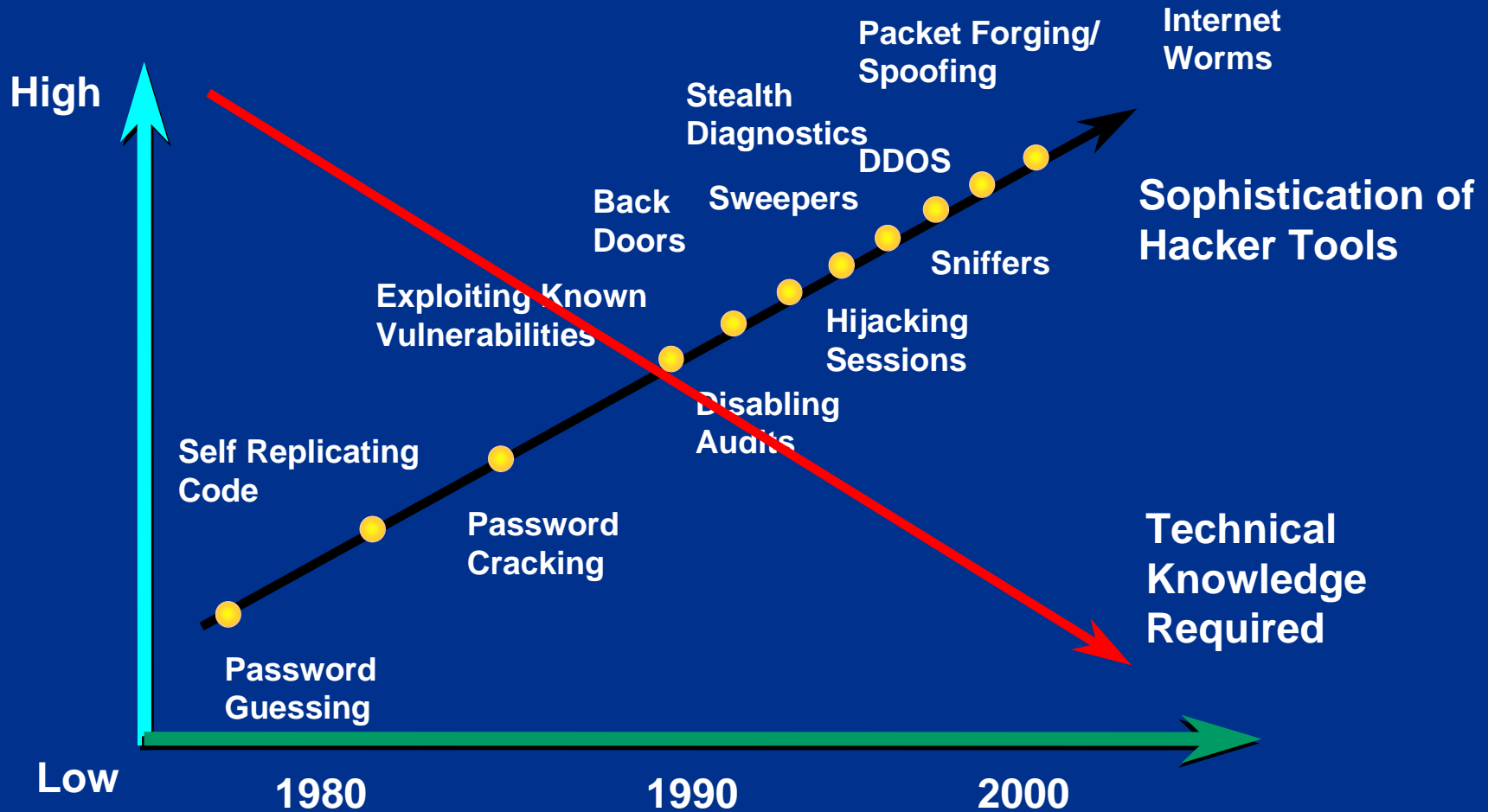


Why Else?



- Development tools do not prevent simple bugs
 - Buffer overflows, etc.
- Q&A Testing methods do not address security
- Customers will pay for bad software

Why is This an Issue?



Graph: Cisco, Inc.

Who's Trying to Change Things?



Microsoft

- Trustworthy computing initiative
- Secure coding awareness and training internally
- Risk assessment services and security certification

• Regulators

- State of California privacy legislation SB 1386 (7/1/03)
 - Legal obligation to inform clients of breaches
- SEC 'SOX' (6/15/04+)
 - Ensuring financial statements are accurate
- Of course, HIPAA, GLBA, and the Basel II Accord

Presentation Overview



- Application Security Framework
- Application Security Integration
- Application Security Awareness

Application Security Framework



- Assumption: ‘Hard boiled’ Infrastructure
 - Standard, repeatable regular and hardened builds
 - Software patch and configuration management
 - N-Tier DMZ firewall implementation
 - Network intrusion monitoring (beyond detection)
 - Host event monitoring and log management
 - Standard services – WebSSO, Kerberos, PKI, etc.
 - Golden hosts for system support
 - Standard utility and hosting platforms
 - Anti-virus and malicious code prevention tools

Application Security Framework



- Policy
 - Ratified document highlighting key principles
 - Defined roles and responsibilities for compliance
- Standards
 - Generic controls for applications development
 - Links to application & change management standards
 - When to introduce...
 - Basic vs. strong authentication
 - ACLs vs. roles-based authorization
 - Logging vs. monitoring for auditing

Application Security Framework



- Standards
 - Isolation vs. encryption for confidentiality
 - Hashes vs. digital signatures for integrity
 - A combination of the above for non-repudiation
- Guidelines and ‘Key Ops’
 - Configuration & implementation of specific technologies
 - Administration and operational requirements

Application Security Framework



- Technology Catalog (Security Architecture)
 - Compliant product ratings and recommendations
 - Code samples and libraries – crypto, authentication, etc.
 - Roadmap for evolving technologies – biometrics, etc.
- Risk Management
 - A methodology for application assessment
 - For Example: Risk Exposure = Magnitude * Probability
 - Identify and document technical and operational risk
 - Define appropriate control levels
 - Tools and methods to measure compliance

Application Security Framework



Document	Sample	Example
Policy	Co. Information Security Policy	Protect the Co.'s information assets
Standard	Application Software Security Standard	Applications must authenticate users
Guideline	Payment System Security Guideline	Further verification required to release a payment
Key Ops	Payment Application Key Ops	How users are assigned tokens
Catalog	Two factor authentication	RSA SecureID card
Risk Mgmt	Assessment Tool	Following Slide

Application Security Framework



- Risk assessment for payment application
 - Confidentiality: **HIGH** (competitive information)
 - Integrity: **HIGH** (high dollar transactions)
 - Availability: **HIGH** (processing within 60 seconds)
 - Data Criticality: **HIGH** (any application down time unacceptable)
 - Data Sensitivity: **HIGH** (private client data)
 - Review Period: 12 months
 - Assessment: "Clients are allowed to enter or view payment data with a user ID and password. However, authorizing the release of funds is validated through the use of an additional token."

Application Security Integration



- Security, like success, is a journey not a destination
- Security is a core component and not 'pixie dust'
- Consider security at the start and all along the way
 - Requirements, design, development, Q&A, and maintenance
- Requirements phase
 - Include security section in document (based on the framework)
 - Budget time and resources now for security throughout project
 - Make it clear to the project team that security is a success factor

Application Security Integration



- Design phase
 - Security components should be part of the application architecture
 - How the application will 'plug into' standard security services
 - What custom security elements might be required
 - Think 'end-to-end' to limit security design errors
 - Begin threat modeling the application architecture
- Threat modeling (Schneier's Attack Trees)
 - For all application components and data flows
 - Start as soon as application architecture and use cases in place
 - Opportunity to begin to document security (risk) due diligence

Application Security Integration



- Threat modeling
 - Along each data flow 'route', identify...
 - Standard, available controls
 - SSL, ID, password, etc.
 - Classes of users
 - Remote, local, administrative, etc.
 - Entries and boundaries (where data crosses privilege levels)
 - Processes, system, machines, etc.
 - Address 'routes' with the greatest 'exposure' first
 - List what each component does
 - Input, processing, and output

Application Security Integration



- Threat modeling
 - Address components along 'routes' with the greatest access first
 - Files, databases, processes, credentials, etc.
 - Review specific controls (if any)
 - Determine if they are sufficient
 - Classify remaining vulnerabilities
 - **STRIDE**: **S**poofing, **T**ampering, **R**epudiation, **I**nformation disclosure, **D**OS, **E**levation (privilege)
 - Create the 'attack trees' to see where vulnerabilities multiply
 - Risk Exposure = Magnitude (how bad) * Probability (how easy)

Application Security Integration



- Threat modeling
 - Focus assessment on high magnitude/high probability first
 - Weigh cost to address against risk (where the magic happens)
 - Make appropriate design changes (easier prior to development)
 - It's OK to tolerate some risk like credit risk, market risk, etc.
- Development phase
 - Implement all security design decisions and requirements
 - Plan to physically and logically segregate elements
 - Data - by criticality - in backend or internal DMZ database
 - Business logic in middle DMZ application/transaction server

Application Security Integration



- Development phase
 - Physically and logically segregate elements
 - User interface in an external DMZ front end server
 - Each server's system, application, and data components
 - No excessive access to underlying resources (least privilege)
 - Never run under privileged accounts
 - Take steps to prevent common coding errors (awareness)
 - Overflows and format strings
 - Script and SQL injection
 - Race conditions
 - Resource leaks leading to DOS

Application Security Integration



- Development phase
 - Control input and output
 - VALIDATE ALL INPUT (based on an 'expected list')
 - Rationalize all output
 - Never use "Password incorrect"
 - Limit debugging messages , comments, field names, etc.
 - Use thorough error and exception handling
 - Closely monitor the use of system resources
 - Components should 'fail safe' protecting assets
 - With consideration for business operating requirements

Application Security Integration



- Development phase
 - Simplify user interaction
 - Keep controls straightforward and unobtrusive
 - Define appropriate roles to manage user access
 - Disable unnecessary features, functions, add-ons
 - Defaults secure, so user must select 'lower' security
 - Write test plans as code developed and components integrated

Application Security Integration



- Q&A testing phase
 - Make security testing a standard requirement
 - Late or accelerated projects must comply
 - Tests should be both inclusive and exclusive
 - Provide standard tools and configuration for testing
 - Measure compliance to security framework
 - Look for design, development, and implementation weaknesses
 - Offer a testing service as part of a risk management process
 - Partner with trusted 3rd parties if resources or skills are lacking
 - The results of these tests should be fed into awareness program

Application Security Integration



- Implementation/maintenance phase
 - Complete risk assessment (acceptance) process
 - Follow a formal change management process
 - Stay current with security updates and patches
 - Manage security of scripts and 3rd party components
 - Default IDs/passwords, run as root, etc.
 - Appropriate file system ACLs
 - Protect configuration and startup files
 - Ensure controls around management utilities
 - Only authorized SAs/DBAs should access some tools

Application Security Integration



- Implementation/maintenance phase
 - Strong controls around remote management
 - Documented security processes (secure operations)
 - Approvals
 - Administration (dedicated interface)
 - Recertification
 - Monitoring activity and event logs
 - Revalidate security considerations at least annually
 - Or when functionality is enhanced
 - Maintain a complete inventory for all applications

Application Security Awareness



- Prepare standardized, targeted training
- 2, 3, or 4 day courses for application developers
 - Customized: not just c++ 'str' function BO exploits to Java coders
 - Focused: use your own tools and products as reference
 - Examples comparing good and bad code samples
 - Hands on: combine labs, practicals, and demos with lecture
- Half day classes for product managers and architects
 - Explain the importance (value) of this security initiative
 - Demo some examples of problems (ideally in their applications)
 - Present the security strategy and fit to their product roadmaps

Application Security Awareness



- 1 to 2 hour “lunch and learn” sessions for management
 - Focus on the value of integrating security and good coding practices
 - ‘Security is about protecting your most valuable asset – your brand.’
 - Reference the critical policies they should already be familiar with
- Surveys and ‘tests’ to track progress
 - Collect and review feedback and evaluations after each class
 - About a month later send a short quiz: What did you retain?
 - 2 months after send a formal survey
 - What behaviors have you corrected in your work?
 - Where have you improved security in your projects?

Application Security Awareness



- Also think about end user awareness
 - Create materials to show them how to use applications securely
 - Change passwords frequently
 - Never allow applications to remember IDs and passwords
 - Always confirm the last login date and time are correct
 - Encourage users to report application security concerns or problems
 - Include an overview of the Information Security Policy
 - Most effective when delivered directly by the business
 - Collect metrics to report which businesses are doing this effectively

Application Security Awareness



- Policies and even regulations are great, but...
- The value added and cost benefits will really sell this
 - Improved developer awareness can reduce many coding problems
 - Bugs identified earlier can be fixed quickly and cheaply (NIST)
 - Security assessments can be more thorough and efficient
 - Effective use of this framework can reduce implementation times
 - Good controls can also improve reliability and resiliency
 - Simpler solutions can be directly integrated into applications
 - Less operational risk

Application Security Awareness



- Costs savings data
 - Source: National Institute of Standards and Technology (NIST)

Stage	Hours to fix	Frequency
Requirements	1.2	7%
Code/unit test	4.9	42%
Integration	9.5	28%
Beta test	12.1	13%
Post-prod'n	15.3	10%

Application Security Awareness



- Training should include
 - The policies and standards framework
 - Integrates into the development cycle
 - The value proposition
 - What developers are responsible for
 - How the security group can help
 - Design problems you have experienced
 - Not maintaining an 'end-to-end' security context (crypto, etc.)
 - Coding problems you have experienced
 - Cross site scripting, cookie poisoning, privilege escalation, etc.



Application Security Awareness



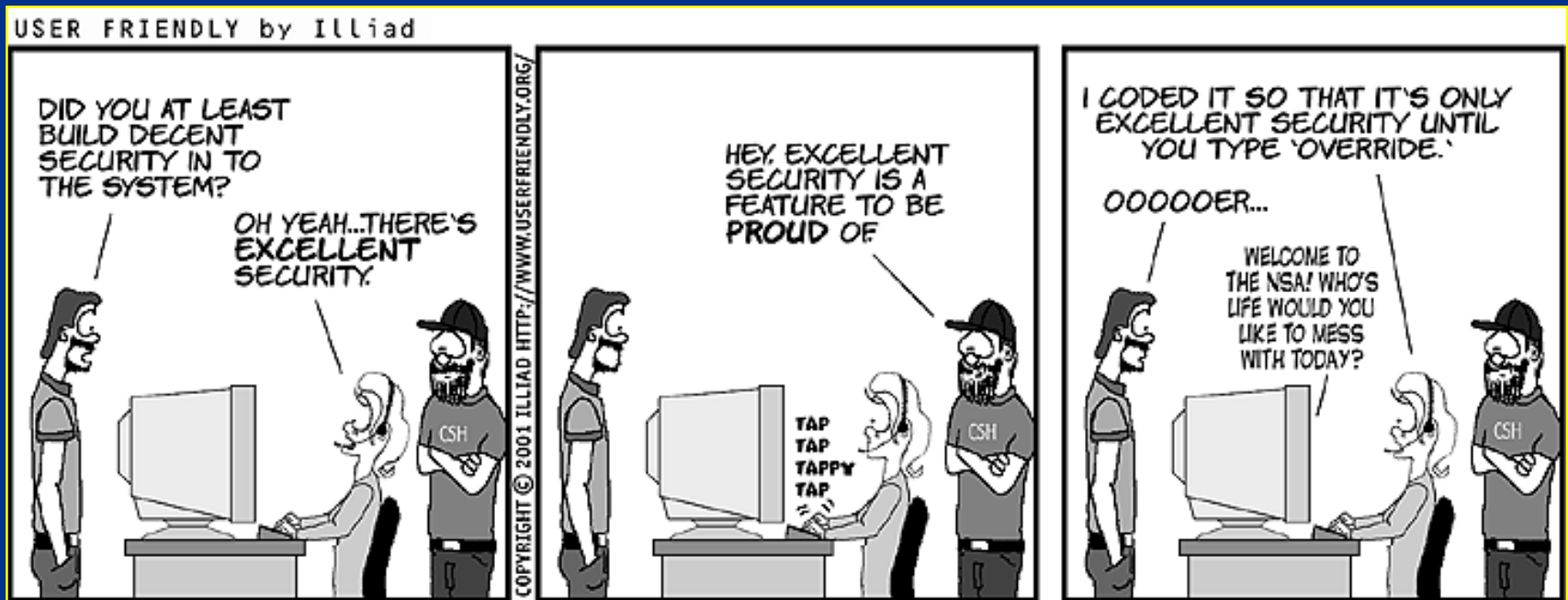
- Training should include
 - Input validation (“location, location, location”)
 - Consistent access controls across every web page
 - Using stored procedures to ‘wrap’ database calls
 - Keeping current with platform security updates, hotfixes, etc.
 - Giving services and application accounts ‘just enough’ privilege
 - Session management
 - Use of Session IDs, cookies, URLs, forms, and hidden fields
 - Preventing brute forcing, replay, or hijacking
 - Never relying on anything from the browser (overhead aside)

Application Security Awareness



- Training should include
 - How to store and use application passwords and encryption keys
 - All actions should produce secure logging and auditing
 - Use of coding and testing tools
 - Static Analysis: compilers and IDEs (Perl taint, etc.)
 - Dynamic Analysis: debuggers (focus on suspected problems)
 - Vulnerability/fault injection: Cenxic Hailstorm
 - Scoping a penetration test (black box, white box, code review)
 - Content checking to manage the integrity of static data

Questions?



Kenneth H. Newman, CISM

Certified Information Security Manager

khn15@columbia.edu