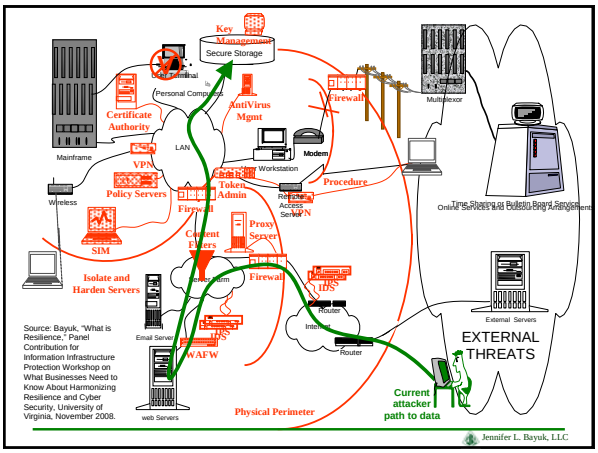
Securing Web Applications

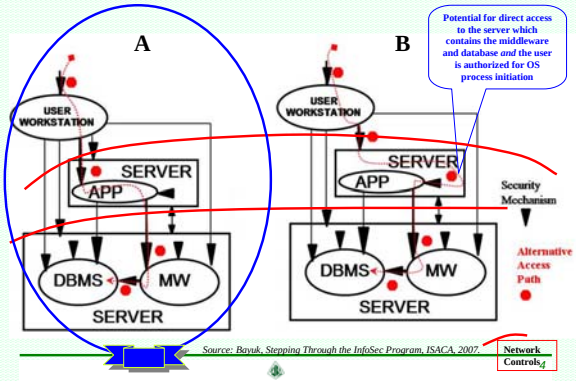
Jennifer L. Bayuk
jennifer@bayuk.com
www.bayuk.com



Web Application Security Roadmap

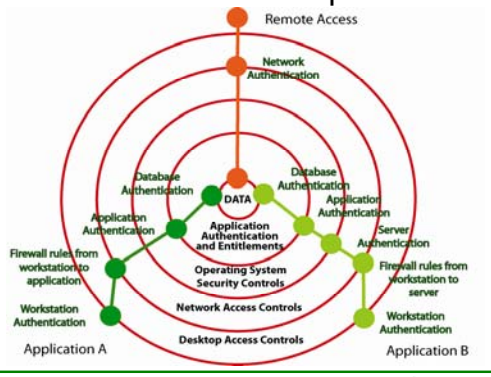
- Architecture
- Engineering
- Software Development
- Operation

Architecture Alternatives

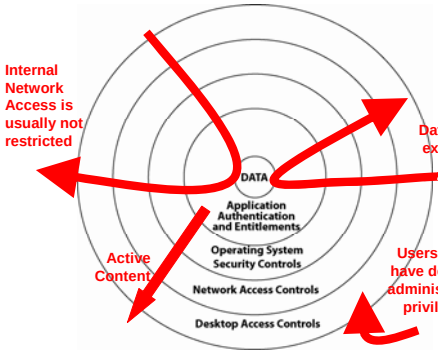


Defense in Depth

Defense in Depth



Layers are Porous



ISF Recommendation

There should be an enterprise-wide process for implementing coherent and consistent security services (eg identity services, authentication services and cryptographic services) and establishing common user and application programming interfaces (APIs).

Excellence in Simplicity

Source: The Standard of Good Practice for Information Security, Security Management, 4.6.1, Information Security Forum, 2007. www.securityforum.org

A Practical Approach

Defense in Depth

- Focus on Prevention, Detection, Recovery
  - Reusable Components
    - Security Model
      - Implementation Requirements

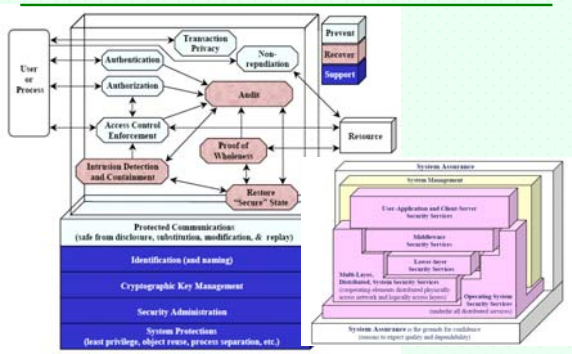
Triad Mining

App A: Control:	Data	Operating System	Application	Network	Workstation
Prevention	DBMS login	Password repository on local server	Password repository on local server	Handheld authentication Session-based filters	Centralized Identity and Access Mgmt
Detection	Logs on servers	File integrity checking	Operations log monitoring	Link and utilization monitoring	Intrusion detection and prevention
Recovery	Backup tapes	Cold standby	Cold standby	Redundant alternative routing	Automated imaging

Potential for Control Re-Use

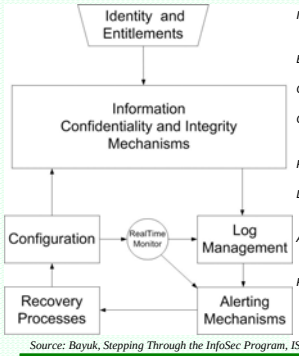
App B: Control:	Data	Operating System	Application	Network	Workstation
Prevention	DBMS login Network isolation	Centralized Identity and Access Mgmt	Centralized Identity and Access Mgmt	Handheld authentication Session-based filters	Centralized Identity and Access Mgmt
Detection	Logs in centralized repository	File integrity checking	Operations log monitoring	Link and utilization monitoring	Intrusion detection and prevention
Recovery	Storage mirroring	Cluster technology	Hot standby	Redundant alternative routing	Automated imaging

NIST Model



Source: NIST800-33: Underlying Technical Models for Information Technology Security

Bayuk Model



Identity Management: authoritative lists of authorized users combined with identity verification capability

Entitlements: map from identities to data-driven business requirements for access control

C&I Mechanisms: access control embedded in business logic

Configuration: security mechanisms such as Single Sign On, Network Isolation, Platform Authorization and Monitoring Configurations

Real Time Monitor: identifies configuration change events that should cause alerts

Log Management: records and detects indications of obvious harm or patterns of suspicious activities

Alerting Mechanisms: initiates automated or manual incident response and (if necessary) recovery

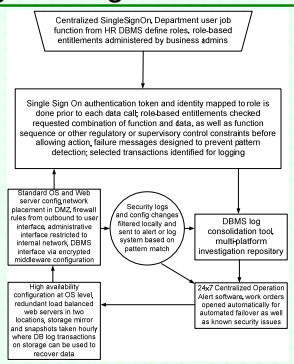
Recovery Processes: methods for maintaining data integrity and availability as well as configuration integrity

Source: Bayuk, Stepping Through the InfoSec Program, ISACA, 2007.

Engineering

Use security model as auditable requirement.

Configuration, configuration, configuration….



The Design/Architecture Review

Objective — *To establish that a system corresponds to security model, and identify configuration parameters in the systems environment required comply.*

Scope — *Network and operating system placement diagrams, as well as detailed technical design documents on system security mechanisms.*

Constraint — *Unknowns or lack of expertise in security mechanisms in externally-built components.*

Approach — *Map architecture onto model. Compare settable parameters of all systems components to known secure configurations.*

Result — *List of issues to address, iterative process.*

Software

- 70% of all vulnerabilities are in application software[1]
- Recently announced 25 known software flaws[2] have been compared to National Quality Forum's "Never Events"[3]
- OWASP Top Ten, Top 25 predecessor which was endorsed by the OCC[4]

[1] Gartner, 2008.
[2] MITRE, 2009.
[3] Charette, R., *The Risk Factor*, IEEE Spectrum Online, January 14, 2009.
[4] Office of the Comptroller of the Currency, 2008.

Top 25 Security Mistakes

Insecure Interaction Between Components

1. Improper Input Validation
2. Improper Encoding or Escaping of Output
3. Failure to Preserve SQL Query Structure
4. Failure to Preserve Web Page Structure
5. Failure to Preserve OS Command Structure
6. Cleartext Transmission of Sensitive Information
7. Cross-Site Request Forgery
8. Race Condition
9. Error Message Information Leak

Risky Resource Management

10. Failure to Constrain Operations within the Bounds of a Memory Buffer
10. External Control of Critical State Data
11. External Control of File Name or Path
13. Untrusted Search Path
14. Failure to Control Generation of Code
15. Download of Code Without Integrity Check
16. Improper Resource Shutdown or Release
17. Improper Initialization
18. Incorrect Calculation

Porous Defenses

19. Improper Access Control
20. Use of a Broken or Risky Cryptographic Algorithm
21. Hard-Coded Password
22. Insecure Permission Assignment for Critical Resource
23. Use of Insufficiently Random Values
24. Execution with Unnecessary Privileges
25. Client-Side Enforcement of Server-Side Security

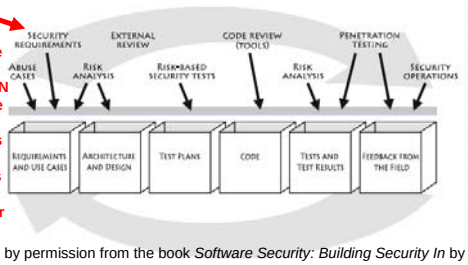
Roadmap to Secure Code

Input Validation and Representation
parse, parse, parse
API Abuse Prevention
use pre and post-conditions
Security Features
close the front door and apply same security to windows
Time and State
maintain authentication across threads, processes, time
Error Handling
fail in safe mode
Code Quality
security is a subset of reliability
Encapsulation
trust no one

See McGraw, Gary, Software Security: Building Security In, also Ott and Fath, "Risk Associated With Web Application Vulnerabilities," ISACA Journal V1 2009.

Secure Software

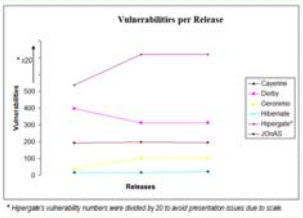
Top 25 list provides guidance in ADDITION to secure coding practices and business require-ments for security.



Adapted by permission from the book *Software Security: Building Security In* by Gary McGraw (Addison-Wesley, 2006) www.swsec.com

See also: *The Security Development Lifecycle* by Howard and Lipner

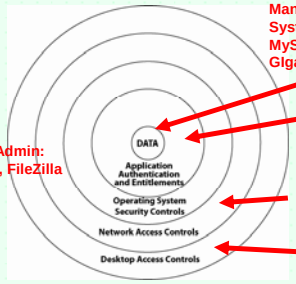
Using Open Source



That they make such a big deal about concentrating on security to the point where they pretty much admit nothing else matters to them

Source: Fortify Open Source Survey, 2008, www.fortify.com.

Open Source Examples



Database Management System: MySQL, GIgaBase

Web and Application Servers: Apache Tomcat Geronimo JBoss Spring Struts

OS Admin: SSH, FileZilla

Network Management Tools: Zenoss

Desktop Components: Ajax, Gnome

See: www.sourceforge.net

Using COTS

- Commercial Off-the-Shelf software should receive the same scrutiny with respect to the security model as any internally built product.
- Lack of access to source code may be mitigated via vendor commitment to secure software practices and increased level of abuse case testing.

CISO's Guide to Software Security

1. SDLC Governance
2. Outsourcing
3. Open Source
4. COTS
5. SaaS
6. Web 2.0

Source: Fortify CISO's Guide Series, 2009, http://www.fortify.com/cisoguides.

Software Security Review Metrics

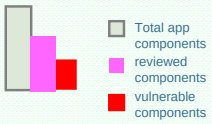
Need to include ALL software components

App 1 = (component 1, component 2, component 3…..)

App 2 = (Inhouse servlet, Apache Webserver, Oracle DBMS)

App 3 = (Inhouse servlet, Solaris Webserver, MySQL DBMS)

App 4 = (Spring servlet, Fuego Workflow Server, Ajax Mashup)



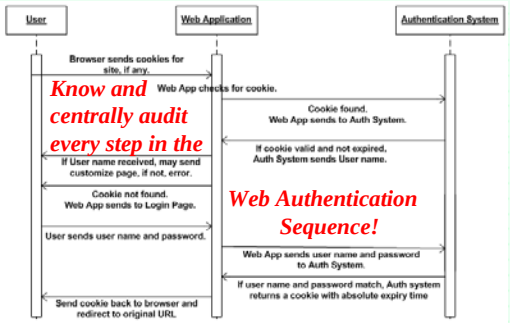
Total app components

reviewed components

vulnerable components

Review strategy may vary where source is not available, but still should be consistently performed.

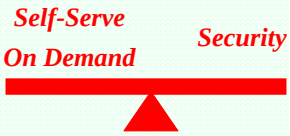
Operation



Know and centrally audit every step in the Web Authentication Sequence!

Web User Administration

- Enrollment
- Login
  - Single Sign-on
- Password Reset
- Entitlements
  - Roles, Groups
- Audit



Questions? Discussion…

Jennifer L. Bayuk
jennifer@bayuk.com
www.bayuk.com