Securing the Application Layer

Sean Malone, CISSP, CCNA, CEH, CHFI
sean.malone@coalfiresystems.com

Institute of Internal Auditors        April 10, 2012
WHY
- do we need application security?

WHAT
- security concepts and principles should be considered?

HOW
- do we integrate security into our business?
Why do we Need Security? Hacking Pays!

- Hacking continues to mature in sophistication but relies on three core vulnerabilities for the majority of success:
  - Default system credentials or shared credentials
  - Applications vulnerable to SQL injection
  - Weak or miss-configured application, network or system access controls
- Web application attack vectors have increased every year
- Internal applications are vulnerable to both internal and external attack vectors
- A combination of Hacking and Malware continue to be the biggest threats and attacks – Over 90% of compromised records in 2010 from this vector
- The vast majority of vulnerabilities exploited were old and should have been mitigated
Hacking… It’s getting easier
Growing Threat Surface

Exponential Growth of IDs
Identity and access management challenging

Number of Digital IDs

- B2E
- B2C
- B2B
- Internet
- client/server

Mainframe
- Pre-1980s
- 1980s
- 1990s
- 2000s

Crime On The Rise

Attacks Getting More Sophisticated
Traditional defenses are inadequate

National Interest
Personal Gain
Personal Fame
Curiosity

Largest segment by $ spent on defense
Largest area by $ lost
Fastest growing segment

Examples:
- Spyware
- Rootkits
- Application attacks
- Phishing/Social engineering

Exponential Growth of IDs

- User
- GUI
- Applications
- Drivers
- O/S
- Hardware
- Physical

Script-Kiddy
Amateur
Expert
Specialist

Internet

Phishing/Social engineering

Author
Trespasser
Thief
Spy

Largest area by volume

B2E
B2C
B2B

Growing Threat Surface
Threats Are Expanding Their Playbook

- No longer effective to be a “one trick pony”
- Multiple threat vectors employed
- Organizational and customer profiling over time
- Profile information shared among threats
- Attack and collect “low value” data to build threat vectors to valuable data
- Service Providers are part of the mix
Security Challenges

New Threats and Increased Sophistication
- New forms of malware
- Profit-based cybercrime

Rising Internal Attacks
- 75 percent of companies report insiders responsible for breaches
- Growing headcount reductions increase risk of intentional intellectual property theft

Shrinking IT Budgets
- 2010 IT spending flat to negative
- More than 2 million jobs shed in 2010

Costly Data Breach
- Average cost of data breach is $202/record or $6.6 million per breach

Need for Increased Access and Sharing
- 70 percent of workforce will connect to corporate networks via mobile devices within four years

Increased Regulatory Pressure
- Requirements for automated controls and new privacy regulations (PCI, HIPAA, SB1386, and others)

Increase Productivity
Grow Sales
Mitigate Risks
Retain Customers
Adapt to Change
WHY
do we need application security?

WHAT
security concepts and principles should be considered?

HOW
do we integrate security into our business?
Current Practices

Generally three approaches to addressing security in application development:

1. Find and Fix

   Use app vulnerability scanning/pen testing on production apps.

2. Defend and Defer

   Web app firewalls or application-level proxies used to reduce or defer the need to address security in design and implementation by architects and developers.

3. Secure at the Source

   Integrating security throughout the software development lifecycle is the only real option to ensure we address security and control requirements effectively, thus increasing the elimination of security vulnerabilities before applications are deployed.
<table>
<thead>
<tr>
<th><strong>Best Practice Recommended Approach</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>An Integrated Approach</strong></td>
</tr>
<tr>
<td><strong>Be Deliberate</strong></td>
</tr>
<tr>
<td><strong>Think Strategically</strong></td>
</tr>
<tr>
<td><strong>Methodology Agnostic</strong></td>
</tr>
<tr>
<td><strong>Example</strong></td>
</tr>
</tbody>
</table>
Hurdles to an Integrated Security Approach

Cost
It’s difficult to persuade management to invest in proactive and strategic software security measures because of the time to produce a ROI.

Resistance to Change
Development organizations often resist changes to existing processes because developers are under tremendous time-to-market pressures.

Inefficiencies
Development organizations view these additional strategic security tasks as impediments to completing overall project and business objectives.

Lack of Training
Development organizations have not been trained in application security nor to see the merit of security design practices.

Compliance != Security
Compliance as a driver typically results in easier-to-implement security measures but does not necessarily lead to truly integrated and more effective application security initiatives.

Lack of Incentives
Development staff are not provided incentive to “care about security” nor is a measurement of security defined and measured.

Key Takeaway:
Through regulatory change, examples of data breaches in the industry, and a significant change in the amount of malicious activity being seen across all sectors, securing the firm’s information is seen as a priority from the top of the organization.
Some Research on Costs and ROI

The 1:10:100 Ratio

Traditional research shows the following ratios:

1. 10-fold increase in the cost to fix issues during the testing/validation phase versus during design/implementation.
2. Up to 100-fold increase in cost to fix issues following deployment versus during design/development.

Note: These ratios do not include any damages caused by insecure software related to card brand costs, state privacy litigation/settlement costs, etc.

Recent Studies

A recent study by Aberdeen Group shows a 4.0-times return on annual investment in “preventive” application security showing that successful prevention (i.e. “secure at the source”) outweighs proactive detection and defense.

A Forrester Research study shows that a coordinated approach to application security drives a positive ROI. Additional positive ROI can be driven by highly integrated and aligned development and testing/validation processes.

Quantify the business value ROI of integrated security by recognizing both cost avoidance and cost savings in the ROI equation:

\[
\frac{\text{AppSecCostsAvoided}}{(\text{AnnualInvestmentInAppSec}) + (\text{AppSecCostsNotAvoided})}
\]
Integrated Security – How To Do It?

Overview of integrated Security Development Lifecycle

1. The Microsoft SDL is one of the most mature in the industry today, and we’ll use it as an example to highlight how to build security components into your lifecycle.

For more info on MS SDL: http://www.microsoft.com/security/sdl/default.aspx

2. MS SDL has a proven track record: By embedding security and privacy throughout its software development lifecycle, MS has reduced its total cost of development and significantly improved overall quality.
   a. 45% reduction in vulnerabilities in Vista vs. XP
   b. 91% reduction in SQL Server 2005 vs. 2000
   c. 35% reduction in vulnerabilities of IE7 vs. IE6

3. Organizations new to a security development process may use the “Simplified Implementation of the SDL” and the “SDL Optimization Model” as a starting point.

Integrating Security in the Dev Lifecycle

Microsoft SDL – An Overview

1. The Microsoft SDL will serve as an example SDLC and includes the following phases:
   a. Core Training (considered a pre-SDL activity)
   b. Requirements
   c. Design
   d. Implementation
   e. Verification
   f. Release
   g. Response
Microsoft considers training a pre-SDL activity.

1. All team members (not just developers) should be made aware of security issues and the integrated approach to security taken by the vendor.
2. Provide secure coding technique training to your development staff based upon the specific development codebase being used. Leverage available training resources (SANS, CWE, OWASP, Etc.).
3. Adequate training prepares all team members to contribute to the security lifecycle.
4. Training materials and sessions should be refreshed and delivered as threat landscape or secure development lifecycle processes change.

**Essential Take-Away**
Prepare and train your staff to actively participate in the secure development lifecycle.
Establish functional and security/privacy requirements.

1. Define the data assets at risk and the security, privacy or compliance issues which apply.
2. Analyze, define and agree upon a minimum set of security requirements for the application.
3. Define the acceptable level of security quality. Use quality gates and/or bug bars to define the severity thresholds for security vulnerabilities.
4. Perform an application security risk assessment to identify functional aspects of the application that require deeper review.
5. Ensure controls are defined to safeguard the most valuable sensitive data/assets.
6. Define an on-going vulnerability discovery and management program.

**Essential Take-Away**

Know your users; know the data in scope of the app; know your risk tolerance; define the “security bar” for the development process.
Design

Define and document your security architecture.

1. Functional specifications accurately and completely describe the intended use of features or functions and describe how to deploy the feature or function in a secure fashion.
2. Analyze the attack surface and develop the threat model (i.e. perform an application security risk assessment).
3. Enumerate threats against the application to map out the attack surface.
4. Consider the data that has to be safeguarded and how it travels through and is protected (in transit and/or storage) by the application.
5. For each threat, define the mitigation strategy and add to the requirements list.
6. Application development teams should have a structured process to consider, document and discuss security implications of various design options.

**Essential Take-Away**

Functional specs must be accurate and complete. Threat modeling is critical to create mitigation strategies and test plans to validate a secure implementation.
Application Risk Assessment / Threat Modeling

THREAT MODELING == APPLICATION RISK ASSESSMENT

1. Formalize your threat modeling process....it must be repeatable.
2. High level threat model process:
   a. Diagram the application flows including Processes, Data Stores, Data In Transit and Trust Boundaries.
   b. With the application diagram, identify and enumerate the vulnerabilities, threats and risks. Think maliciously. Ask the question, “What could possibly go wrong?”. Make it a team/brainstorming activity. Use the Diagram and a threat model like STRIDE, as follows:
      i. Spoofing – authentication impersonation
      ii. Tampering – modification of data undetected
      iii. Repudiation – ensure that an action can be traced to a specific user
      iv. Information Disclosure – obtaining info without authorization
      v. Denial of Service – overload or crash a service (downtime)
      vi. Elevation of Privileges – obtain access above that which is authorized
3. Threat mitigation strategies must now be defined for each threat in the model.
4. Threat modeling must be used as input to test cases....test cases must be defined for the threats defined in the model.

Threat Modeling – An Example

Threat Analysis & Modeling – An Example from Star Wars

**Asset** – The resource of value and worthy of protection/security.

**Vulnerability** – An exploitable weakness in the system.

**Risk** – The likelihood that a threat will harm the asset.

**Threat** – An event or activity that will cause harm to the asset.

**Countermeasure** – Reduces risk without changing design.

Poor or incomplete threat modeling can lead to devastating consequences.
Implementation

Define approved tools, libraries and common controls and enforce their use to implement the security plan.

1. Standardize on approved tools and libraries and use the latest versions which have completed the security review process.
2. All functions and APIs must be reviewed and analyzed for security risk; those determined to be unacceptable should be prohibited from use.
3. Code is reviewed according to firm-standard controls & processes and acceptable use of all tools, libraries or common controls. Unacceptable functions and APIs should be removed and replaced with a secure alternative.
4. Code reviews occur for all major changes to the application using a skilled resource independent of the original developer of the code. Independence of the review is key.
5. Static code analysis and initial penetration testing occur during this phase.

**Essential Take-Away**
Standardized tools enable an on-going security posture. Independent (internal or external) and skilled review is critical. Leverage peer review, static code analysis and penetration testing for broad analysis coverage and defense in depth.
Verification/Validation/Testing

Validate the application against the requirements.

1. Define your security testing plan to verify specific design is implemented according to plan.
2. Develop test plans including the threat model analysis from the design phase to ensure that each enumerated threat is adequately tested.
3. Use run-time and internal code scanning tools to ensure that functionality works as designed and security vulnerabilities have been adequately implemented, especially for sensitive projects.
4. Use fuzzing by deliberately introducing malformed and/or random data to induce failure.
5. Ensure “code coverage” by using debuggers or analysis tools to determine if any code is never executed. Remove or re-test as necessary.
6. Imperative that the verification/validation process be performed by independent (internal or external) and skilled/trained resources.

**Essential Take-Away**

Test planning should be based upon the Threat Model. Independent run-time testing in a pre-production environment is an essential pre-release activity.
Release

Prepare the software for production deployment and develop your incident response plan.

1. Final review of all security-related activities should be performed on the application prior to approval and release.
2. Review your incident response plan including the appropriate resources to act as first points of contact in the event of a security emergency (should include development, marketing, communications and management staff).
3. Define how discovered vulnerabilities will be reported; who’s responsible for managing the process; what performance/quality goals should be targeted.
4. Use a defined model (DREAD, e.g.) to assess the criticality of a discovered vulnerability.
5. Define user training, if applicable, to include training the users on how to use the application in a secure manner. Training should include how to recognize and report security issues in production.

**Essential Take-Away**

Incident response planning is a critical activity prior to production rollout.
Response

Execute the incident response plan as necessary.

1. If a security incident or data breach has (or may have) happened, report it immediately to ensure the correct response plan is initiated to mitigate potential damage to the firm.
2. Ensure incidents feed back into the SDL process, with a particular focus on reviewing/updating the threat model and test cases.
3. Application updates and security patches must undergo the same amount of development rigor as that used during the full security development process.
4. Implement and track your on-going vulnerability discovery and management program.

**Essential Take-Away**

Understanding processes and responsibilities for incident response is critical to on-going application support. Incident response activities must feed back into the threat modeling and testing processes of the SDLC.
Process Recommendations

1. Consider security as an integral part of your overall software development lifecycle. Treat software security with a full life-cycle approach, not in a piecemeal fashion.
2. Define security metrics for measuring the performance of development staff and incent developers to take on security responsibilities.
3. Clearly align your corporate business objectives (features and time-to-market) with security objectives.
4. Train your development staff. Re-train your staff as new vulnerabilities continue to be discovered.
5. Demand as much quality of third party code as you do from internally developed code (define stringent acceptance criteria).
7. Stay abreast of emerging technologies (e.g. mobile platforms, cloud computing, etc.) and the net effect these technologies have on your overall security program and strategy.
8. Leverage independent resources (internal or external) for application security validation and checkpoints throughout the lifecycle.

If you know yourself but not your enemy, for every victory gained you will also suffer a defeat. Sun Tzu - Art of War
WHY do we need application security?

WHAT security concepts and principles should be considered?

HOW do we integrate security into our business?
Security Goals: CIA Triad

- **Confidentiality**
  - Only the correct people, programs, and systems are allowed to view the data

- **Integrity**
  - Only the correct people, programs, and systems are allowed to modify the data

- **Availability**
  - The correct people, programs, and systems are always able to access the data when necessary
Security Principles

- Assume that ...
  - The application will be attacked
  - The attacker has full knowledge of the application
  - The attacker has access to the source code
- Every input to the application is potentially malicious
- Think like an attacker: “How could I compromise this application?”
Application Security Concepts

- Identification, Authentication, Authorization
- Input Validation
- Output Encoding
- Cryptographic Controls
- Logging & Auditing
- Information Leakage
- Configuration Management
- Database Security
Identification, Authentication, Authorization

- **Identification**
  - Who do you claim to be?
  - Username or account number

- **Authentication**
  - Prove that you are who you say you are
  - Password, Passphrase, PIN, Etc.

- **Authorization**
  - Now that we know who you are, what should you be able to view, and what should you be able to modify?
  - Access control list (ACL), permission matrix
Input Validation

- All input should be assumed to be malicious
- This includes:
  - URL requests and parameters
  - Form data
  - Cookies
  - Other headers such as referrer, user agent, etc.
  - Uploaded files and filenames
  - Any content from a third-party service (RSS, SMS, etc.)
- Input should be tested using whitelist validation, not blacklist filtering
Output Encoding

- Encode output to neutralize dangerous characters.
- Which characters are dangerous? This varies depending on the context. Encoding methods are different for HTML, JavaScript, SQL, LDAP, and other outputs.
- Use central libraries, such as the .NET Anti-XSS library, or the Java ESAPI library.
Cryptographic Controls

- Do not rely on “security through obscurity”
- Kerckhoffs' principle: A cryptosystem should be secure even if everything about the system, except the key, is public knowledge.
- Use industry-standard algorithms and secure cryptographic libraries
- Encrypt both data-at-rest and data-in-motion
Logging & Auditing

- Discuss logging and auditing requirements with your legal team.
- Log to an external server and restrict access to logs, in order to maintain log integrity.
- Have procedures for regular audit of application logs to detect unauthorized access or suspicious activity.
- No account should be excluded from logging and auditing – how do you catch malicious activity by an administrator?
Information Leakage

- Attackers can use many small pieces of information to construct a better understanding of the application, which assists in future attacks.
- Don’t expose any information unnecessarily. This includes:
  - Technical error messages
  - Stack traces
  - Usernames
  - Full filepaths
- What can an attacker learn based on how the application responds? Is there a difference between how the application responds when a resource does not exist and when a user is not permitted to access a resource?
Configuration Management

- Ensure that all applications have guidelines for secure implementation
- Track changes to all levels: hardware, OS, platform, third-party libraries, etc.
- Keep up-to-date on software patches
- Change all passwords, certificates, and keyfiles when moving an application into production
Database Security

- Minimize the potential damage from unauthorized database access
- Use different roles for different actions, and restrict database users to the only necessary privileges
- NEVER connect an application to the database as the database server administrator
- Restrict all unnecessary functionality on the database server (for example, xp_cmdshell)
WHY
do we need application security?

WHAT
security concepts and principles should be considered?

HOW
do we integrate security into our business?
Questions?
References

- Threat Modeling:
  - https://www.owasp.org/index.php/Application_Threat_Modeling