Developing Secure Software in the Age of Advanced Persistent Threats

ERIC BAIZE
EMC Corporation

DAVE MARTIN
EMC Corporation
Our Job: Keep our Employer out of the Headlines

**Product Security Group**

**The Journal**

Vendor [ABC] issues an emergency patch for its flagship product and urges customers to apply it without delay to address an actively exploited vulnerability.

**Product impact on customers risk**

**IT Security Organization**

**The Journal**

Company [ABC] admits to losing sensitive information following a security breach in its corporate network.

**Security impact on enterprise risk**
March 2011: A breach on RSA’s Infrastructure leads to Customer Risk

The Journal

“RSA urges customers to take immediate steps to strengthen their SecurID implementations …

… following the detection of a sophisticated cyber attack in progress being mounted against RSA”

APTs are Redefining Product Security. How?
Traditional Approach to Product Security
Security Groups in High-Tech Organizations

Product Security Group

Product security assurance programs
(Vulnerability response, Security Development Lifecycle)

Product impact on customers risk

IT Security Organization

Internal security and protection programs
(Employees, systems & IP protection and risk management)

Security impact on enterprise risk

Product Security Group

IT Security Organization

EMC²

RSACONFERENCE EUROPE 2012

Product security assurance programs
(Vulnerability response, Security Development Lifecycle)

Assume the customer environment is compromised

Minimize risks introduced by products into the customer environment

- Build attack resistant products
- Document products for secure deployment
- Efficiently handle security vulnerabilities and security patches
Product Security Development Lifecycle
Focuses on Software Vulnerabilities

Sec. Dev. Lifecycle
✓ Training
✓ Requirements
✓ Threat modeling
✓ Code analysis
✓ Security testing
✓ Documentation
✓ Assessment
✓ Vulnerability response

PRODUCT SECURITY POLICY & Related Standards

Design
✓ Authentication & access control
✓ Logging
✓ Network security
✓ Cryptography and key management
✓ Serviceability
✓ Secure design principles

Implementation
✓ Input validation
✓ Injection protection
✓ Failing securely
✓ Web and C/C++ coding standards
✓ Handling secrets
✓ Secure Build operations
✓ Code signing

PRODUCT RISK (4 levels)
- Critical: Requires executive sign-off
- High: Requires remediation in next release
- Medium: Requires monitoring
- Low

Gap assessment as part of product engineering process
The Changing Landscape
Characteristics of advanced threats

- Single minded, determined and innovative
- Target individuals over systems
- Through reconnaissance will understand your processes, people & systems better than us
- Will exploit ANY weakness
- Countermeasures increase sophistication
- Custom malware, NOT detectable by signatures
- Are not in a hurry will take as long as it takes
- Goal is long term & persistent access
- The perimeter has shifted, all systems now exist in a hostile environment
Evolution of IT Products Creates New Attack Vectors

- Mainframe
- PC / Client-Server
- Web
- Cloud Computing: Technology providers host customer applications & data

- Network Security
- Perimeter Security
- System Security
Implications
Attacks Against Technology Providers Are Impacting Customers

- **March 2011:** “RSA urges customers to take immediate steps to strengthen their SecurID implementations following the detection of a sophisticated cyber attack in progress being mounted against RSA.”

- **April 2011:** “Microsoft issues an update to all supported versions of Windows after Comodo issues fraudulent digital certificates as a result of an attack.”

- **January 2012:** “Symantec recommends disabling the pcAnywhere product as a result of a theft of source code”

- **July 2012:** “Data breach at Yahoo results in disclosure of 400,000 user names and passwords”
Advanced Threats are Often Undetected

94% of companies learn they have been compromised from a third party such as law enforcement.

The median length of time an organization has been compromised before they find out is **416 days**.

Source: Mandiant M-Trends (2012)
Assume You Are Compromised …

“Consider that no organization is impenetrable. Assume that your organization might already be compromised and go from there.”

Security for Business Innovation Council (August 2011)
Fighting APTs: Layered Defense, Intelligent Monitoring and Governance

① Layered defense

② Intelligent monitoring and analytics

③ Strong Governance

Logs & Events

Early detection
Technology Providers Need to Adapt their Product Security Strategy

- Create an integrated governance model
- Build intelligent monitoring into products
- Design layered defense in products
The New Face of Product Security
Technology Providers Need to Adapt their Product Security Strategy

Product Security Group

Build intelligent monitoring into products

Create an integrated governance model

Build intelligent monitoring into products

Design layered defense in products

IT Security Organization
Rethinking Product Security Assuming the Customer and its Supply Chain are Compromised

Assume every system across the supply chain is compromised

Minimize risks introduced by products into the customer environment
- Develop secure software

- Secure product delivery & hosted services
- Secure the product development environment
- Secure the supply chain
Expanding the Security Development Lifecycle into Product Operations

Product Security Group

IT Security Organization

Product Operations: Hosting, Engineering systems, Manufacturing

Product Governance

Drive standard adoption as part of our Security Development Lifecycle

Collaborate on standards for:
- Source code management
- Anti-counterfeiting
- Cloud / Hosting

Enterprise Governance

Ensure continuous monitoring as part of the enterprise security management program
Product Governance: Expanding EMC Security Development Lifecycle

PRODUCT SECURITY POLICY & Related Standards

- **Design**
  - Authentication & access control
  - Logging
  - Network security
  - Cryptography and key management
  - Serviceability
  - Secure design principles

- **Implementation**
  - Input validation
  - Injection protection
  - Failing securely
  - Web and C/ C++ coding standards
  - Handling secrets
  - Secure Build operations
  - Code signing

- **Source Code Protection**
- **Counterfeiting Protection**

PRODUCT RISK (4 levels)

- Critical: Requires executive sign-off
- High: Requires remediation in next release
- Medium: Requires monitoring
- Low

Gap assessment as part of product engineering process

Gap assessment as part of product engineering process
Enterprise Governance: Product Security has Become Part of Enterprise GRC Strategy

Supplier risk management
Counterfeiting prevention
Software integrity controls

Source code and hosting infrastructure protection
Software integrity controls

Supply Chain Risk
Product Security Risk
IT Security Risk

EMC Enterprise GRC

Other Business Risks
Technology Providers Need to Adapt their Product Security Strategy

Product Security Group

Create an integrated governance model

Build intelligent monitoring into products

Design layered defense in products

IT Security Organization
Building Attack-aware Software: Add Intelligence to Security Logs

- Leverage threat modeling to dynamically log software abuse
  - Buffer overflow
  - SQL Injections
- Evolve from logging to debug towards logging for detection and alerting
  - Insert anomaly logging in program logic
- Direction: Design software to leverage the enterprise risk ecosystem
  - Reputation, white lists …
Technology Providers Need to Adapt their Product Security Strategy

Product Security Group

- Build intelligent monitoring into products
- Create an integrated governance model
- Design layered defense in products

IT Security Organization

- Build intelligent monitoring into products
- Design layered defense in products
- Create an integrated governance model
Designing APT-Resistant Software: Split-value cryptographic authentication

(1) Passwords are stored split between two servers: Compromising one server does not expose the password

(2) During authentication, claimed passwords are verified without exposing the legitimate password

(3) Random number is regularly refreshed to reduce the windows of time for a successful attack on both servers
Assume the Source Code is Compromised

- No hardcoded secrets
- Accelerate the adoption of a Secure Software Development Lifecycle
  - Threat modeling
  - Code scanning
  - Security Testing
- Account for source code disclosure in threat modeling
- Build integrity control in source code review and protection
- Pay close attention to comments

```python
$secretKey = "London2012";

Avoid unsafe string functions – e.g. `strcpy()`

/*
 * To do:
 *   Add authentication
 */
```
Build Software Integrity Controls

Sourcing & Development
- Source code protection
- Authenticity and integrity control of embedded components
- Backdoor testing and code review
- People, process and supplier controls

Delivery & Execution
- Executable signing
- Malware scanning
- Secure code signing process
- Use of hardware root of trust
- White listing

Reference: “Software Integrity Controls” (June 2010) - Published by SAFECode (www.safecode.org)
Developing Software for the Cloud: Security in Agile

- **Security Stories**
  - Security focused stories
  - Associated security tasks

- **Operational Security Tasks**
  - Targeted at Agile practitioners
  - Conducted on an ongoing basis

- **Advanced Security Tasks**
  - Most advanced security tasks
  - Require guidance from security practitioners

---

Industry Secure Development Practices & CWE

---

EMC²

---

RSA Conference Europe 2012
## Secure Agile Development Example

<table>
<thead>
<tr>
<th>Security-focused story</th>
<th>Backlog task(s)</th>
<th>SAFECCode Fundamental Practice(s)</th>
<th>CWE -ID</th>
</tr>
</thead>
</table>
| As a architect/developer, I want to ensure **AND** As QA, I want to verify correct permission assignment and maintenance for all critical resources | [D/T] When a critical resource is defined or accessed, make sure that the access permissions (programmatic and systemic) to it are left in their most restrictive but useful possible setting.  
[D] Describe correct permissions for the resource in the security configuration guide. | Use least privilege | **CWE-732** |

Source: “*Practical Security Stories and Security Tasks for Agile Development Environment*” (July 2012) - Published by SAFECode (www.safecode.org)
Wrap-Up
Apply: Change Your Software Development Assumptions

Assume every system is compromised

- If you have not done it yet, define a secure software development process and train your developers
- Bridge IT security and software security groups
  - Integrate governance models
- Integrate software integrity controls in your secure software development process
  - Code review for backdoors
  - Verification of source code system security
- Implement a process for controlling integrity and authenticity of external components
  - Start with an inventory
- Implement a secure code signing process
- Build intelligent logging for security, not just for debugging
- Translate your secure software development process in Agile stories
Summary

- Secure product development as grown as an software engineering discipline
- The changing threat landscape and the emergence of cloud are products attack surface
- Technology providers and software development organization need to adapt their secure software development process
  - Change trust assumptions in threat assessment
  - Integrate code integrity controls
  - Develop an integrated governance model
  - Adapt security controls to Agile