SearchFinancialSecurity.com
INFORMATION SECURITY
FINANCIAL
INFORMATION SECURITY DECISIONS

Software Security for Financial Services

Meeting the New PCI Application Security Requirements

Compliance, PCI, and Beyond

Diana Kelley
SecurityCurve

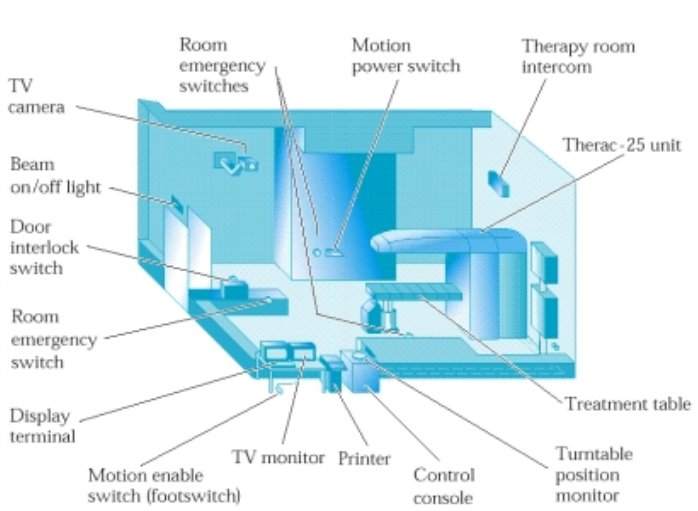
Agenda

- Software Security Matters
- Software Security and PCI
- The PA-DSS
- Building Security In
- Testing Tools

What do These have in Common?

2005 Toyota Prius

Therac-25 Radiation Therapy Machine



Miele G885 SC Dishwasher

Software

- **The dishwasher . . . was rendered useless after a power outage. Its software got knocked out."**

  http://www.baselinemag.com/print_article/0,3668,a=35839,00.asp

- **"Prius hybrids dogged by software. . . stall or shut down at highway speeds"**

  http://money.cnn.com/2005/05/16/Autos/prius_computer/index.htm?cnn=yes

- **Six known accidents involved massive overdoses by the Therac-25 -- with resultant deaths and serious injuries."**

  http://courses.cs.vt.edu/~cs3604/lib/Therac_25/Therac_1.html

Financial Services Software

- **Chemical Bank ATM Incident**
  - "…a *single line in an updated computer program* . . . caused the bank to process every withdrawal and transfer at its automated teller machines twice. Thus a person who took *$100 from a cash machine had $200 deducted*, although the receipt only indicated a withdrawal of $100."

http://query.nytimes.com/gst/fullpage.html?res=9B00E7D7173BF93BA25751C0A962958260

Financial Services Software

- **Royal Bank of Canada Software Error**
  - "After a *software upgrade went badly* awry last week, the holders of some *10 million accounts* at the bank had to *wait days in some cases for deposits to be credited* or prearranged payments to be completed."
  - …"the problems started with *a routine programming update* by the information technology staff. . . . the new software was written in-house"

http://query.nytimes.com/gst/fullpage.html?res=9A06E4D81131F934A35755C0A9629C8B63

Financial Services Compliance

- **FFIEC**
  - Authentication Guidance
- **GLBA**
  - Third party access to data
- **SEC 17a-4**
  - Record keeping and archiving for trades
- **SOX**
  - For FIs that are publicly traded
- **….and PCI**

Software Security and PCI

- **Requirement 6 – "Develop and maintain secure systems and applications"**
  - Patching
  - Configuration
  - Development lifecycle
  - Testing
  - Production

Sub-requirement 6.3

- **Develop software applications based on industry best practices and incorporate information security throughout the software development life cycle.**

  - 6.3.1 Testing of all security patches
  - 6.3.2 Separate development, test, and production environments
  - 6.3.3 Separation of duties between development, test, and production
  - 6.3.4 Live PANs are not used for testing or development
  - 6.3.5 Removal of test data and accounts before production
  - 6.3.6 Removal of custom application accounts, usernames, and passwords
  - 6.3.7 Review of custom code prior to release to production or customers

Text edited from original source: PCI Data Security Standard, https://www.pcisecuritystandards.org/pdfs/pci_dss_v1-1.pdf

Sub-requirement 6.3

- **Develop software applications based on industry best practices and incorporate information security throughout the software development life cycle.**
  - 6.3.1 Testing of all security patches
  - 6.3.2 Separate development, test, and production environments
  - 6.3.3 Separation of duties between development, test, and production
  - 6.3.4 Live PANs are not used for testing or development
  - 6.3.5 Removal of test data and accounts before production
  - 6.3.6 Removal of custom application accounts, usernames, and passwords
  - 6.3.7 Review of custom code prior to release to production or customers

Text edited from original source: PCI Data Security Standard, https://www.pcisecuritystandards.org/pdfs/pci_dss_v1-1.pdf

Sub-requirement 6.3

- **Develop software applications based on industry best practices and incorporate information security throughout the software development life cycle.**

  - 6.3.1 Testing of all security patches
  - 6.3.2 Separate development, test, and production environments
  - 6.3.3 Separation of duties between development, test, and production
  - 6.3.4 Live PANs are not used for testing or development
  - 6.3.5 Removal of test data and accounts before production
  - 6.3.6 Removal of custom application accounts, usernames, and passwords
  - 6.3.7 Review of custom code prior to release to production or customers

Text edited from original source: PCI Data Security Standard, https://www.pcisecuritystandards.org/pdfs/pci_dss_v1-1.pdf

Sub-requirement 6.5

- **Develop all web applications based on secure coding guidelines such as the Open Web Application Security Project (OWASP) guidelines. Review custom application code to identify coding vulnerabilities.**

  - 6.5.1 Unvalidated input
  - 6.5.2 Broken access control (for example, malicious use of user IDs)
  - 6.5.3 Broken authentication and session management (use of account credentials and session
  - cookies)
  - 6.5.4 Cross-site scripting (XSS) attacks
  - 6.5.5 Buffer overflows
  - 6.5.6 Injection flaws (for example, structured query language (SQL) injection)
  - 6.5.7 Improper error handling
  - 6.5.8 Insecure storage
  - 6.5.9 Denial of service
  - 6.5.10 Insecure configuration management

Text edited from original source: PCI Data Security Standard, https://www.pcisecuritystandards.org/pdfs/pci_dss_v1-1.pdf

Sub-requirement 6.5

- **Develop all web applications based on secure coding guidelines such as the Open Web Application Security Project (OWASP) guidelines. Review custom application code to identify coding vulnerabilities.**

  - 6.5.1 Unvalidated input
  - 6.5.2 Broken access control (for example, malicious use of user IDs)
  - 6.5.3 Broken authentication and session management (use of account credentials and session
  - cookies)
  - 6.5.4 Cross-site scripting (XSS) attacks
  - 6.5.5 Buffer overflows
  - 6.5.6 Injection flaws (for example, structured query language (SQL) injection)
  - 6.5.7 Improper error handling
  - 6.5.8 Insecure storage
  - 6.5.9 Denial of service
  - 6.5.10 Insecure configuration management

Text edited from original source: PCI Data Security Standard, https://www.pcisecuritystandards.org/pdfs/pci_dss_v1-1.pdf

Quick Example – SQL Injection

- **Ability to show orders from a table in a SQL DB**
  - Correct Usage
    - User enters in Name field = Kenny
    - Result

```
SELECT * FROM OrdersTable WHERE CustomerName =
'Kenny'
```

  - Exploit Usage
    - Attacker enters Name and SQL Command
      - Semi colon triggers end of query begins a new one
    - Result

```
SELECT * FROM OrdersTable WHERE CustomerName =
  'Kenny';drop table OrdersTable--'
```

  - What happens to the Orders table?

SQL Injection in the News

- **April 2008 - nihaorr1**
  - Infected upwards of 100,000 web pages (per the Register)
    - 500,000 per Slashdot
  - Used SQL injection to infect databases
  - Legitimate users (at legitimate but infected sites) were redirected to the attacker site
  - And infected by drive-by malware/Trojan if vulnerable

Sub-requirement 6.6

- **Ensure that all web-facing applications are protected against known attacks by applying either of the following methods:**
  - Having all custom application code reviewed for common vulnerabilities by an organization that specializes in application security
  - Installing an application layer firewall in front of web-facing applications.

PCI Data Security Standard, https://www.pcisecuritystandards.org/pdfs/pci_dss_v1-1.pdf

Sub-requirement 6.6

- **Clarifications were issued in February 2008**
  - Reiterates that this is an either/or requirement
    - Do not need to implement both
- **For Code Review**
  - Manual review of application source code
  - Proper use of automated application source code analyzer (scanning) tools
  - Manual web application security vulnerability assessment
  - Proper use of automated web application security vulnerability assessment (scanning) tools

Information Supplement: Requirement 6.6
Code Reviews and Application Firewalls Clarified
https://www.pcisecuritystandards.org/pdfs/infosupp_6_6_applicationfirewalls_codereviews.pdf

Sub-requirement 6.6 Clarifications

- **For Code Review**
  - Internal resources can perform the scan/assessment
  - Must be organizationally separate from the management of the application being tested

Information Supplement: Requirement 6.6
Code Reviews and Application Firewalls Clarified
https://www.pcisecuritystandards.org/pdfs/infosupp_6_6_applicationfirewalls_codereviews.pdf

More information on scanning and assessment tools coming up in next section!

Sub-requirement 6.6 Clarifications

- **Application Firewall**
  - = Web Application Firewall (WAF)
    - Not an application-layer firewall
  - What makes a WAF acceptable for PCI?
    - Meet all applicable PCI DSS requirements
    - React appropriately to threats
    - Inspect web application input and respond
    - Prevent data leakage
    - Enforce both positive and negative security models
    - Inspect Hypertext Markup Language (HTML), Dynamic HTML (DHTML), and Cascading Style Sheets (CSS)

  For more additional information please review:

Information Supplement: Requirement 6.6
Code Reviews and Application Firewalls Clarified
https://www.pcisecuritystandards.org/pdfs/infosupp_6_6_applicationfirewalls_codereviews.pdf

Sub-requirement 6.6 Clarifications

- **Example WAF Vendors**
  - **Barracuda (NetContinuum)**
  - **Breach Security**
  - **Check Point Web Intelligence**
  - **Cisco ASA**
  - **Citrix**
  - **F5**
  - **Imperva**
  - **Protegrity**

Requirement 6 Recommendations

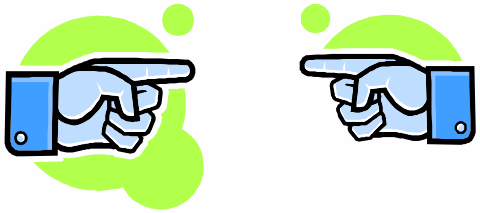
- **Have change control procedures**
  - Impact statements, signoff for changes, and backout procedures
- **Have a process for identifying new vulnerabilities**
- **Test production changes**
  - And educate testers!
- **Have a documented software development lifecycle**
- **Have separate personnel and environments for production and test**
- **Code review or firewall consider:**
  - Time constraints
  - Code availability

PA-DSS

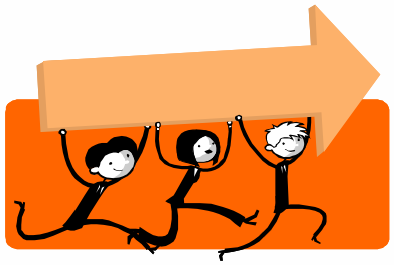
- **Payment Application Data Security Standard**
  - Started life out as VISA's PABP
  - Guidelines on creating secure credit card payment systems
- **Applies to payment applications that are**
  - Developed and sold to more than one customer
  - Not resident in standalone (dumb) terminals if they meet basic standards
    - Not connected to rest of the network
    - No SAD is stored
- **For buyers**
  - Look for point of sale systems that have received PA-DSS (or PABP) certification
- **For developers**
  - Review the PA-DSS for additional in-depth guidance when creating secure payment systems

https://www.pcisecuritystandards.org/pdfs/pci_pa-dss_faqs.pdf

Building Security In

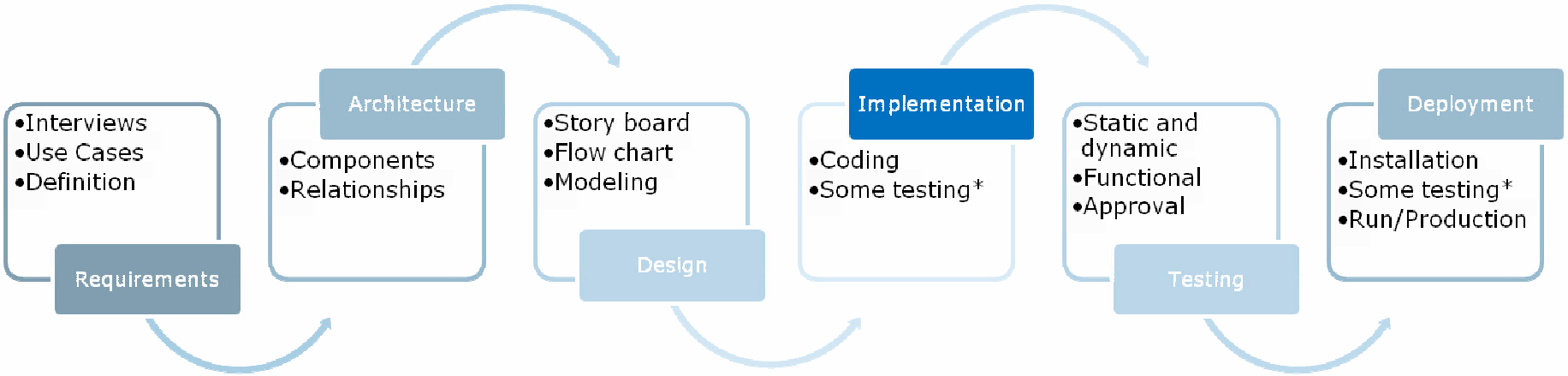


- **Check assumptions**
  - And leave finger pointing at the door
- **Is a team effort**
  - "It Takes a Village"

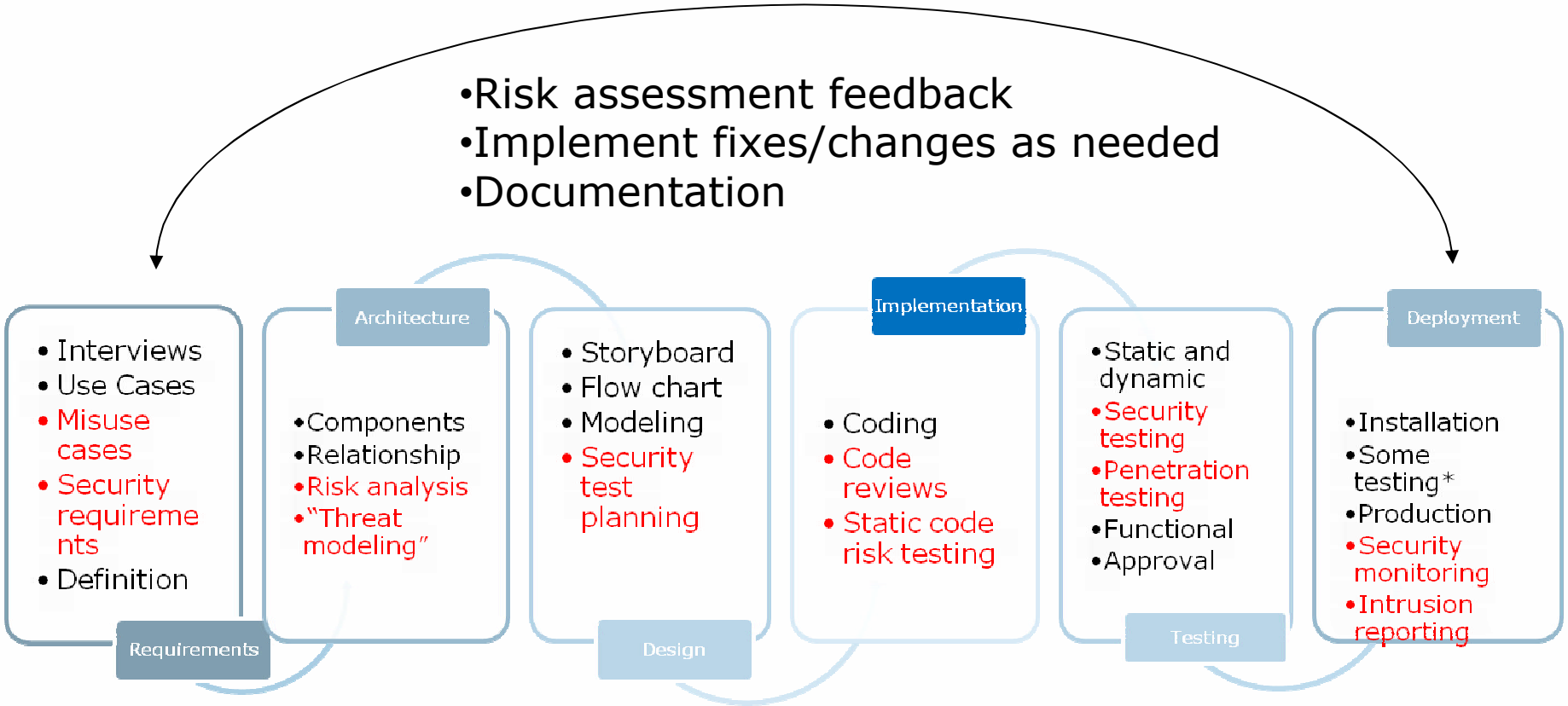


- **Is not the same thing as creating "perfect" code**
  - Unbreakable?
    - Not likely
- **Risk assessment**
  - Balancing the risks and consequences

Software Development Lifecycle



Securing the SDLC



Additional Secure SDLC Resources

- **Comprehensive, Lightweight Application Security Process, (CLASP)**
  - The Open Web Application Security Project (OWASP)

    http://www.owasp.org/index.php/OWASP_CLASP_Project
- **The OWASP Top Ten**
  - And Testing Guide

    http://www.owasp.org/index.php/OWASP_Top_Ten_Project

    http://www.owasp.org/index.php/Category:OWASP_Testing_Project
- **Cigital's TouchPoints**
  - Software Security: Building Security In by Gary McGraw

    http://www.cigital.com/training/touchpoints/

Additional Secure SDLC Resources

- **DHS – Build Security In**

  https://buildsecurityin.us-cert.gov/

  - Top Ten Security Coding Practices

    https://www.securecoding.cert.org/confluence/display/seccode/Top+10+Secure+Coding+Practices

- **Information Assurance Technology Analysis Center (IATAC) SOAR on Software Security Assurance**

  http://iac.dtic.mil/iatac/download/security.pdf

- **Microsoft's Security Development Lifecycle (SDL)**

  - A Look Inside the Security Development Lifecycle at Microsoft

    http://msdn.microsoft.com/msdnmag/issues/05/11/SDL/

  - The Security Development Lifecycle by Michael Howard and Steve Lipner

Tools

- **Static Source Code Analysis**
  - Requires access to source code
  - Can be accomplished before build
  - Manual or
  - Automated
    - For developers (inside the IDE)
    - For auditors/testers (as stand alone)
- **Dynamic**
  - Source code not required
  - Tests the product from the view of the "outsider"
  - Best in conjunction with
    - Skilled testers who can tune the products
    - Manual penetration testing to validate tool findings

Tools

- **Which is better – Static or Dynamic?**
  - a/k/a – Black Box or White Box
  - Tools look at different views of the application
  - Most comprehensive approach is “Grey Box”

Tools - Dashboards

- **Scanning/Testing tools provide dashboards of metrics**
  - Number of flaws
  - Severity of flaws
  - Time to fix
- **Can be grouped by**
  - Application team
  - Application type
- **And used to measure improvement**

Tool Vendors

Static	Dynamic
Fortify	Cenzic
HP/SPI*	HP/SPI
Klocwork	IBM/Watchfire
Ounce Labs	NTObjectives
Veracode* (binary SaaS)	WhiteHat

Final Thoughts

- **FIs create a lot of custom/proprietary code**
- **Compliance is often about the controls**
- **Follow a robust SDLC methodology**
  - Define requirements and hygiene for financial and other sensitive data before implementation
  - Test before production
- **As new mandates arise**
  - Audit – controls may already be in place
  - Adjust - as needed