

## A Practical Guide to Securing the SDLC

**Kathleen Koh**  
*Rational Solution Architect*  
*IBM Software Group, Rational*  
[kohslka@sg.ibm.com](mailto:kohslka@sg.ibm.com)

# Innovate2010

The Rational Software Conference

Let's **build** a smarter planet.

24 August 2010 Mandarin Orchard, Singapore



## Application Security is more visible...



**Combating Data Theft**



**Securing Outsourcing**



**Supporting Compliance**



**Securing the SDLC**

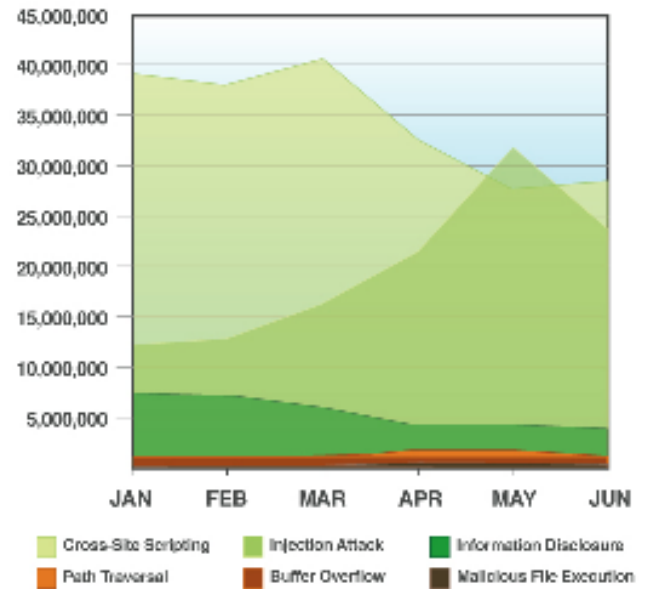
### Managers & Developers Are Being Asked Difficult Questions

- What regulations and standards are required?
  - ▶ PCI, HIPAA, FISMA
- What confidential data is at risk?
- What risk threshold is tolerable?

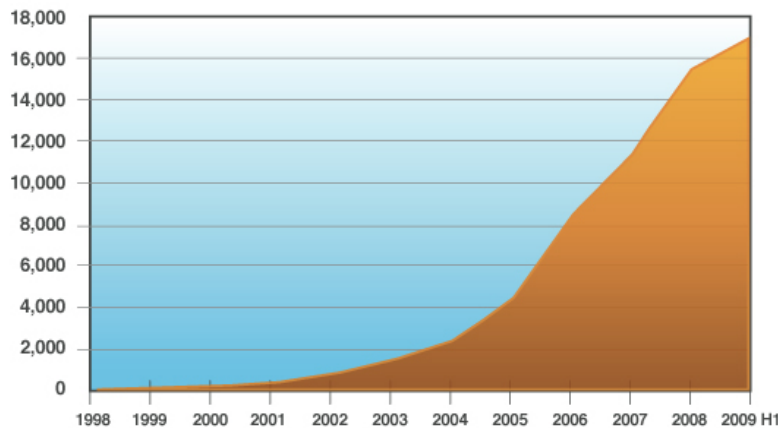
# Web App Vulnerabilities Continue to Dominate

- ▶ **49%** of all vulnerabilities are Web application vulnerabilities
- ▶ SQL injection and Cross-Site Scripting are neck-and-neck in a race for the top spot
- ▶ **90%** of injection attacks are attributed to SQL-related attacks
- ▶ Automated toolkits continue to flourish in 2009
- ▶ SQL injection attacks continue to grow up **50%** in Q1 2009 vs. Q4 2008 and nearly doubling in Q2 vs. Q1

Web Application Attacks by Category

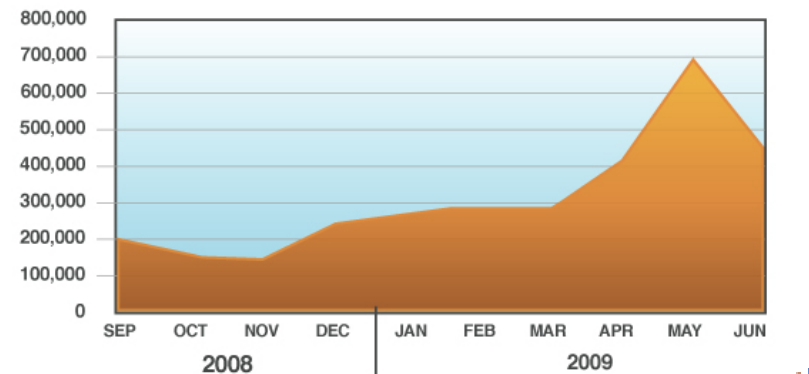


Vulnerability Disclosures Affecting Web Applications  
Cumulative, year over year

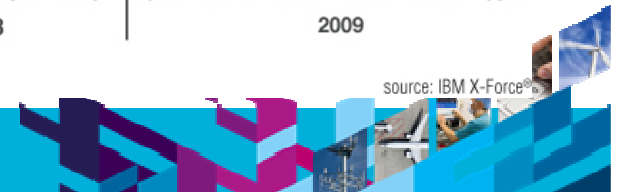


source: IBM X-Force®

SQL Injection Attacks  
Average Daily Attacks by Month



source: IBM X-Force®



# Mapping from OWASP 2007 to 2010 Top 10

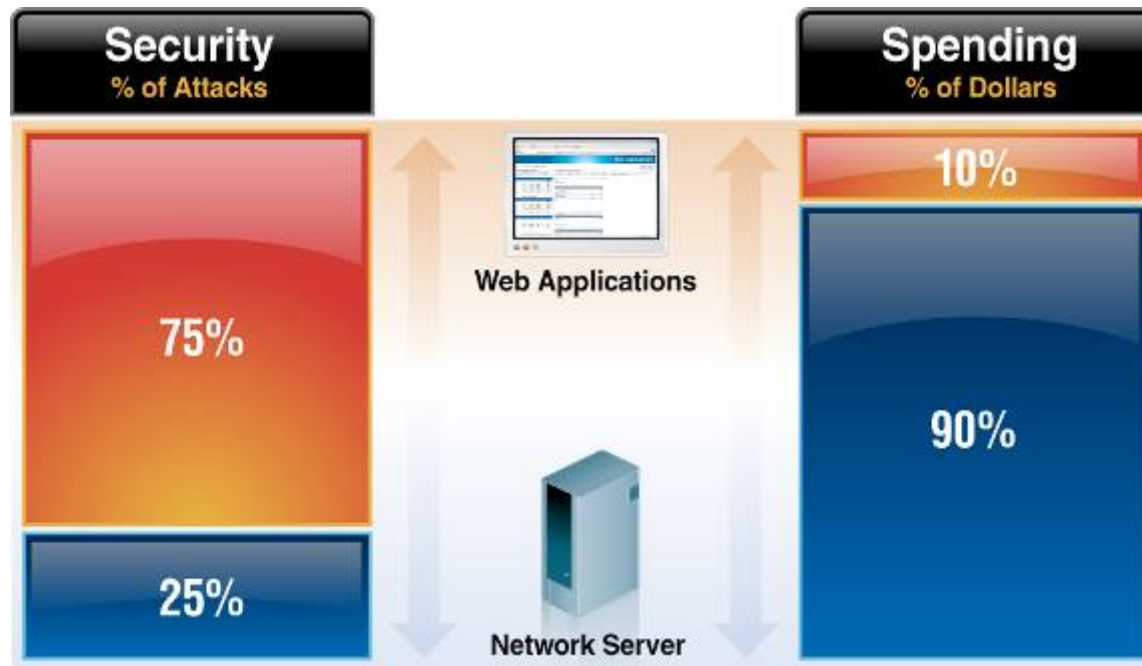
OWASP Top 10 – 2007 (Previous)	OWASP Top 10 – 2010 (New)
A2 – Injection Flaws	↑ A1 – Injection
A1 – Cross Site Scripting (XSS)	↓ A2 – Cross Site Scripting (XSS)
A7 – Broken Authentication and Session Management	↑ A3 – Broken Authentication and Session Management
A4 – Insecure Direct Object Reference	= A4 – Insecure Direct Object References
A5 – Cross Site Request Forgery (CSRF)	= A5 – Cross Site Request Forgery (CSRF)
<was T10 2004 A10 – Insecure Configuration Management>	+ A6 – Security Misconfiguration (NEW)
A8 – Insecure Cryptographic Storage	↑ A7 – Insecure Cryptographic Storage
A10 – Failure to Restrict URL Access	↑ A8 – Failure to Restrict URL Access
A9 – Insecure Communications	= A9 – Insufficient Transport Layer Protection
<not in T10 2007>	+ A10 – Unvalidated Redirects and Forwards (NEW)
A3 – Malicious File Execution	- <dropped from T10 2010>
A6 – Information Leakage and Improper Error Handling	- <dropped from T10 2010>

Source: [http://www.owasp.org/index.php/Top\\_10](http://www.owasp.org/index.php/Top_10)



# Web App Vulnerabilities Continue to Dominate

## Security and Spending are Unbalanced



*“The cleanup cost for fixing a bug in a homegrown Web application ranges anywhere from \$400 to \$4,000 to repair, depending on the vulnerability and the way it's fixed.”*

-Darkreading.com



# Cost is a Significant Driver

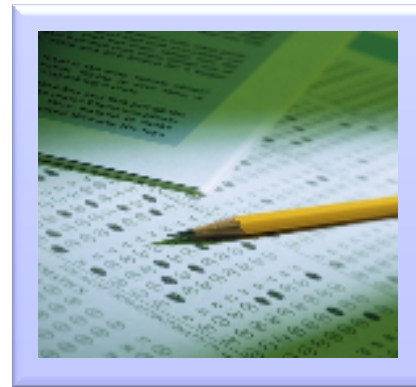
80% of development costs are spent identifying and correcting defects!\*



During the coding phase  
**\$80/defect**



During the Build Phase  
**\$240/defect**



During the QA/Testing phase:  
**\$960/defect**



Once Released as a product :  
**\$7,600/defect + Law suits, loss of customer trust, damage to brand**

The increasing costs of fixing a defect...

\*National Institute of Standards & Technology

Source: GBS Industry standard study

Defect cost derived in assuming it takes 8 hrs to find, fix and repair a defect when found in code and unit test.

Defect FFR cost for other phases calculated by using the multiplier on a blended rate of \$80/hr.



# Headlines - Rate and Cost of Breaches Steadily Increasing

## South Korean Government Websites are Attacked by Suspected Cyber Attack

Submitted by Jimmy Peterson on 2010, July 8 - 17:19 Technology Sector Featured TNM South Korea

### Defeat Man-in-the-Browser

Free whitepaper for Banks & FIs on defeating MITB malware

### On-line Trading

Plus500®: worldwide stock trading. No Fees, Free demo account!

Ads by Google



On Tuesday, [South Korean](#) officials uncovered that suspected [cyber attacks](#) had led to the complete shutdown of websites belonging to South Korean main government firms, banks and Internet sites, in a rapid outpouring that is cited to be associated with similar attacks witnessed in the U. S.

Ahn Jeon-eun, a spokeswoman at Korea

[WashingtonPost.com](#) > Technology > Special Reports > Cyber-Security

## More than 75,000 computer systems hacked in one of largest cyber attacks, security firm says

By [Felix Welton](#)  
Washington Post Staff Writer  
Thursday, February 18, 2010

More than 75,000 computer systems at nearly 2,500 companies in the United States and around the world have been hacked in what appears to be one of the largest and most sophisticated attacks by cyber criminals observed to date, according to a northern Virginia security firm.

The attack, which began in late 2008 and was discovered last month, targeted proprietary corporate data, e-mails, credit-card transaction data and login credentials at companies in the health and technology industries in 196 countries, according to Herndon-based NetWitness.

News of the attack follows reports last month that the computer networks at Google and more than 30 other large financial, energy, defense, technology and media firms had been compromised. [Google](#) said the attack on its system originated in China.

TOOL BOX

[Home](#) | [Print](#) | [E-mail](#)  
[Yahoo! Buzz](#)  
[Google Analytics](#) **TRY OUR TRACKING FREE FOR 30 DAYS!**

---

COMMENT

193

Comments | [View All](#)

COMMENTS ARE CLOSED

---

WIKIS BLOGGING powered by [sphere](#)

> [Links to this article](#)

---

Sponsored Links

[Bank of America's Mortgage](#)  
[Want To Learn More About Buying A Home?](#)  
[Start With Our Free Guide](#)  
[myhome.bankofamerica.com](#)

---

[I Had High Blood Pressure](#)  
[New Guidelines to 120/80](#) [and out how I did!](#)

THE BUSINESS VALUE OF TECHNOLOGY

---

NEWS
BLOGS
SOFTWARE
SECURITY
HARDWARE
MOBILITY
WINDOWS
INTERNET

Privacy > Attacks/Breaches > Vulnerabilities > Application Security > End User/Client Security > Endpoint > Storage Security > Encryption > Security Software > All Security Stories > Security Blog > Security Clearance

---

E-mail this page
Print this page
BOOKMARK
Facebook
Twitter
LinkedIn
StumbleUpon

---

### Data Loss Costing Companies \$6.6 Million Per Breach

Customers, it seems, lose faith in organizations that can't keep data safe and take their business elsewhere, a Ponemon Institute survey found.

By [Thomas Claburn](#)  
InformationWeek  
February 3, 2010 04:00 AM

The total average cost of a data breach last year reached \$202 per record, a 21% increase since 2007, a study published Monday revealed.

The study was conducted by the Ponemon Institute, a privacy and data protection research group, and 114% of data encryption vendors' sales based on the costs incurred by 43 organizations that have suffered data breaches.

WHITE PAPER: [Lost Laptops!](#)  
Protect your employees' laptops while they are on the road.  
[Download Now](#)

More Security Insights
White Papers

> The Web Hacking Incident Database 1011 - Attack

Protect The Business Enable Access

---

ATTACKS / BREACHES
VULNERABILITIES
APPLICATION

SECURITY MANAGEMENT
STORAGE SECURITY
ENCRYPT

---

E-mail this page
Print this page
BOOKMARK
Facebook
Twitter
LinkedIn

---

## Small Business: The New Black In Cybercrime Targets

Enticed by poor defenses of mom-and-pop shops, hackers turn away from hardened defenses of banks and large enterprises



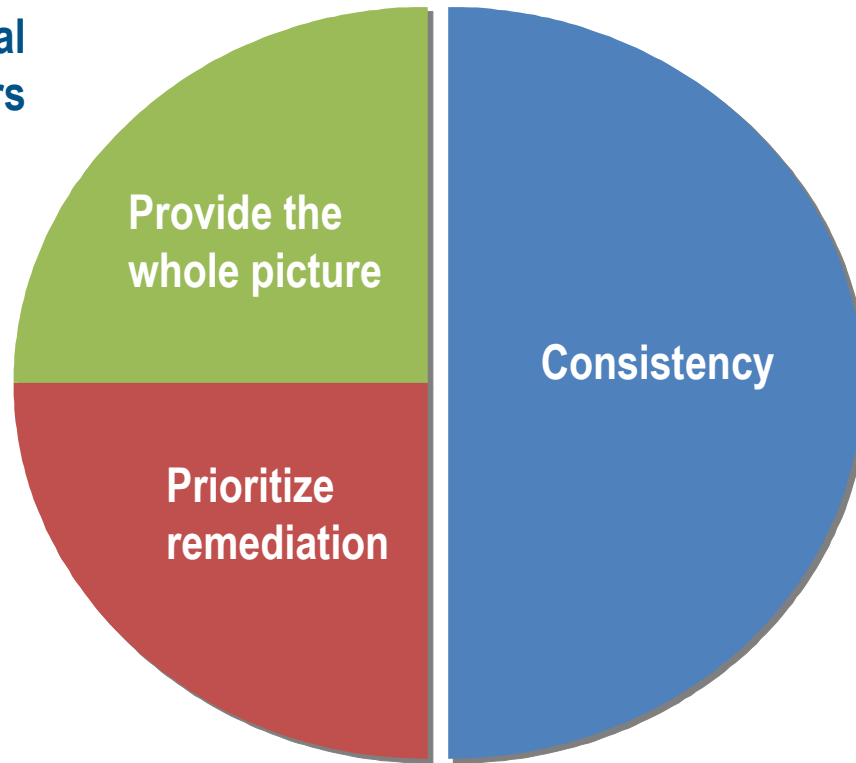
## Is this really necessary?

- What's being done isn't working
  - Albert Einstein - "Insanity: doing the same thing over and over again and expecting different results"
- Security isn't always included
  - ▶ Software Engineer vs. Software Security Engineer
  - ▶ Coding guidelines vs. *secure* coding guidelines
- "Build Security In" – sounds deceptively simple
  - ▶ Starts with training. This class is a good place to start.
  - ▶ Requires a commitment to change. If we agree that what we are doing isn't working than it should be obvious that we need to change what we are doing.
- Policy is not a four letter word
  - ▶ Requirements, Requirements, Requirements
  - ▶ Developers need security requirements, if not given don't assume, demand.
- Have a plan before you need one



# Follow the path to Secure Coding

Large-scale design flaws typically trump individual coding errors



Create consistent processes, policies, and a culture of improved security

Developers must identify all vulnerabilities in the code, then remediate the greatest risks first

# Sometimes the answers can **only** be found in the source code

- Does the application enforce or even use appropriate access controls?
- In what ways and in what places does the application attempt to connect to the network?
- Is there malicious code or back doors in your applications?
- Can user inputs or outputs can corrupt your system ?
- Is customer credit card information encrypted?
- Is sensitive data being stored outside of your database?

```

/**
 * Gets the instructions attribute of the sqlInjection object
 * @return The instructions value
 */
protected String getInstructions(WebSession s)
{
    String instructions = "Enter your account number to revi
    return ( instructions );
}

/**
 * Gets the menuItem attribute of the DatabaseFieldScreen objec
 * @return The menuItem value
 */
protected Element getMenuItem()
{
    return makeMenuItem( EEN,
}

/**
 * Gets the ranking attribute
 * @return The ranking
 */
protected Integer getRanking()
{
    return new Integer
}

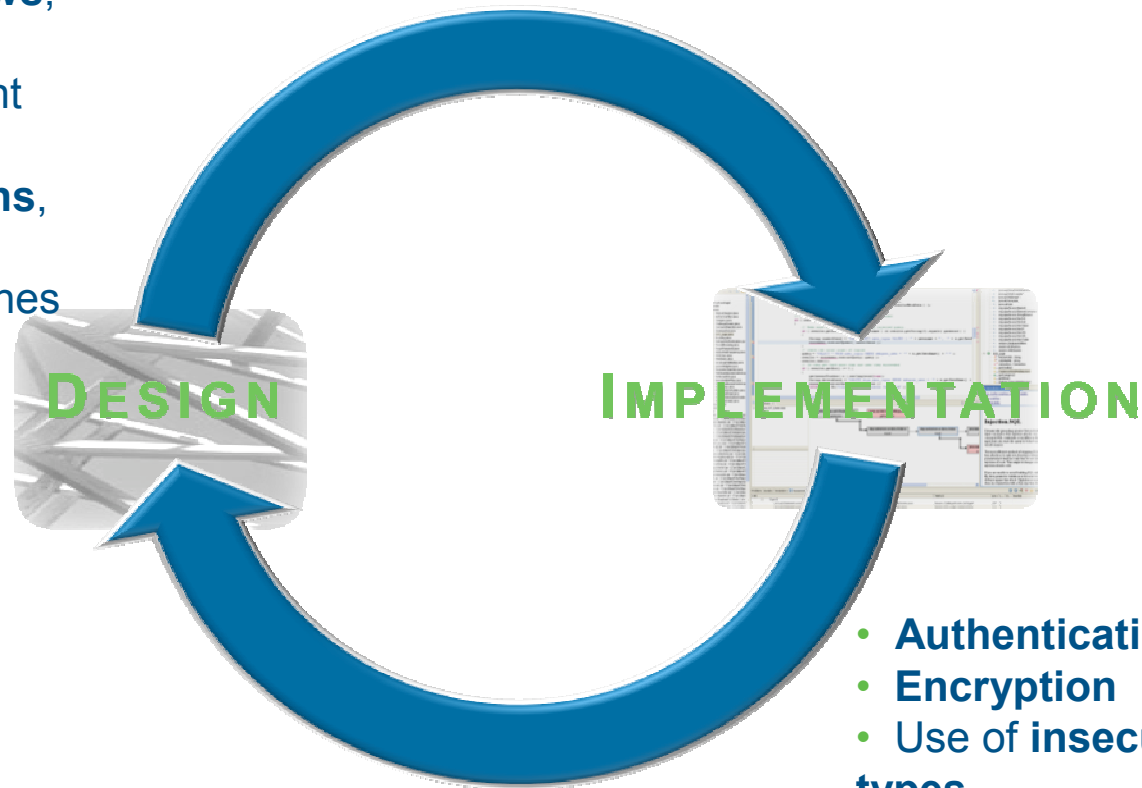
/**
 * Gets the title attribute
 * @return The title value
 */
public String getTitle()
{
    return ( "How to Perform SQL Injection" );
}

/**
 * Constructor for the DatabaseFieldScreen object
 * @param s Description of the Parameter
 */
public void start( WebSession s )
{
    try
    {
        setup( s );
        if ( connection == null )
    }
}
    
```



## Where to look for vulnerabilities

- **Buffer overflows**, result from mismanagement of memory
- **Race conditions**, result from call timing mismatches

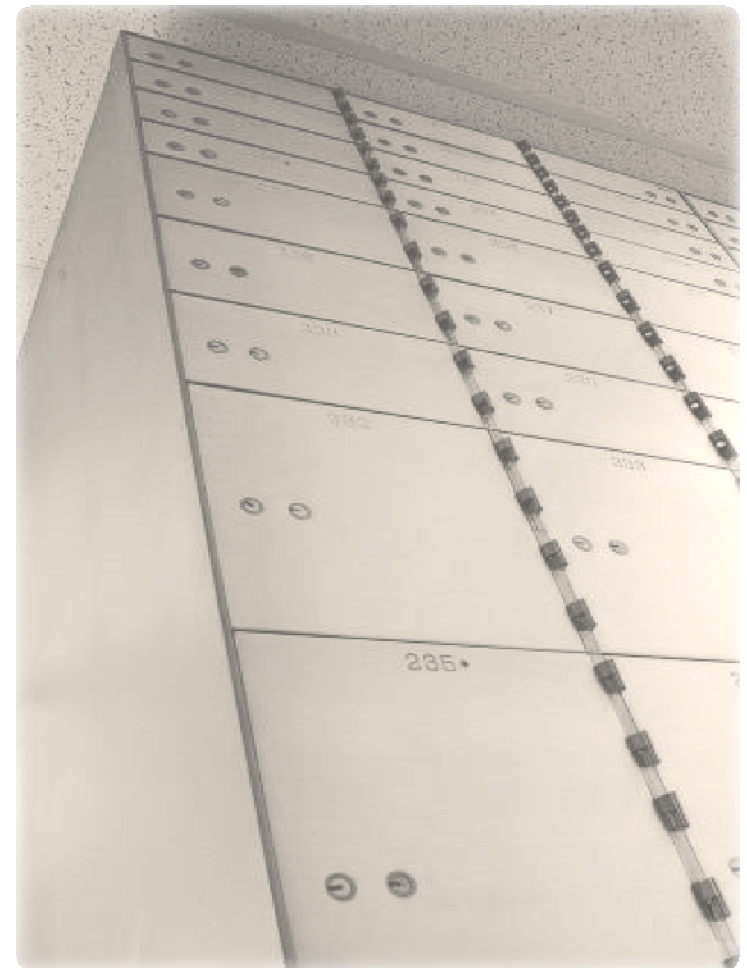


- **Authentication**
- **Encryption**
- **Use of insecure external code types**
- **Validation of data input and application output**

## How to look for vulnerabilities

- Manual Code Review
  - ▶ Time-consuming, expensive, error prone
- Penetration Testing
  - ▶ Useful but can only discern a small sub-set possible errors
- Automated Testing Tools

*“The most effective approach is to integrate source code vulnerability scanners into the application development, integration and test process.” (Gartner)*



## An Interesting Article – The Trustworthy Computing Security Development Lifecycle

From Microsoft's SDL <http://msdn.microsoft.com/en-us/library/ms995349.aspx>

**“However, one finding will come as no surprise to long-time security researchers: penetration testing is not the way to achieve security. Penetration testing is an element of the Final Security Review (FSR) for a major software release, but product team activities throughout the entire lifecycle focus on threat modeling, code reviews, the use of automated tools, and fuzz testing rather than penetration testing. The latter measures are much more thorough in preventing or removing security bugs than the classic ad hoc penetration testing.”**



## That wasn't really helpful

- It is much more effective to look at the places in the SDLC that you can reduce risk.
  - ▶ Requirements
  - ▶ Design
  - ▶ Implementation
  - ▶ Test
  - ▶ Deployment



# Requirements

- Identifying security requirements are an integral part of the software design process, and the most neglected
    - Plan for appropriate resources to support the product team's schedule
    - Include security milestones and exit criteria that is based on project size, complexity and risk.
    - Maximize software security while minimizing disruption to plans and schedule
  - Just as good project requirements requires use cases, good security requirements require abuse cases
    - Security Goals, Challenges and plans must be reflected in requirement planning documents
    - Industry standards need to comply
  - Must be able to identify all potential assets at risk and outline the required and acceptable mitigation requirements.
- 
- ❌ Example of a bad requirement:
    - All sensitive data needs to be encrypted
  - ✅ Example of a better requirement:
    - All sensitive data needs to be encrypted both in transit and at rest using no less than 256 bit AES encryption, see addendum A for the list of items that are considered sensitive for this application.



## Security Requirements Categories and Formulate end to end security architecture

- Auditing and Logging
- Authentication and Authorization
- Session Management
- Input validation and output encoding
- Exception Management
- Cryptography and Integrity
- Data at rest
- Data in motion
- Configuration Management

*IBM's Secure Engineering Framework:*

<http://www.redbooks.ibm.com/redpieces/abstracts/redp4641.html?Open>





## Design: Policy Definition

- It is important that organizations begin to formalize secure coding guidelines.
- Avoid the temptation to “grade” an organization, development manager, or individual contributor’s, ability to deliver secure code without letting them know what is on the test.
- Policy, in the case of security requirements, is to remove ambiguity as much as possible.
- Examples
  - ▶ New development projects using C/C++ must avoid the use of all following api’s: gets(), strcpy(),unbounded use the printf and sprintf family of calls etc.
  - ▶ All data transferred from web clients that contain customer specific information must be transported using SSL, and if any personal information is stored using cookies the entire application needs to be SSL enabled.



## What Details: Threat Modeling

- Threat modeling is an important aspect in developing good security requirements as well as designing good mitigation strategies
- Aspects of threat modeling should occur in several phases of the SDLC
  - ▶ During Requirements
    - Phase 1: Identifying assets at risk and business objectives
    - Phase 2: Generate use and abuse cases and assign an impact for each case
    - Phase 3: Determine the probability of compromise and rank the risks
  - ▶ During Design
    - Phase 4: Identify components responsible for controlling access to and from assets identified in Phase 1.
    - Phase 5: Identify the threats posed by Phase 2 and Phase 3 against the components outlined in Phase 4.
  - ▶ During Implementation & Test
    - Phase 6: Review application to identify weaknesses against the threats identified in Phase 5 about and review mitigation and remediation efforts.



## Design: Security Design Review

- The security design review is a critical step in the SDLC. The primary goal of this step is to verify that the policies identified in the requirements and phases 1-6 of the threat modeling exercise have the appropriate mitigation strategies identified in the application architecture.
- Identify any gaps, this may include identifying new threats.
- This should be done as early in the process as possible, for an agile development process every feature iteration that impacts security as identified by the requirements needs to perform this step.



## Application Vulnerability Assessment

- Think of this as the verification step. This is to verify that all policy requirements and threats have the appropriate mitigation in the final product.
- This also enables unintended or new threats to have another chance of being found prior to deployment.
- Leverage tools as much as possible to reduce costs.



# What To Look For: The Checklist

- ✓ Security-related functions
- ✓ Input/Output validation and encoding errors
- ✓ Error handling and logging vulnerabilities
- ✓ Insecure Components
- ✓ Coding errors

*“Detecting and correcting security vulnerabilities early in the application development life cycle, prior to deployment and operations, results in significant risk and cost reduction.”*

Gartner

```

/**
 * Gets the instructions attribute of the DatabaseFieldScreen object
 * @return The instructions
 */
protected String getInstructions()
{
    String instruction = null;
    return (instruction);
}

/**
 * Gets the menuItem attribute of the DatabaseFieldScreen object
 * @return The menuItem
 */
protected Element getMenuItems()
{
    return makeMenuItem( "SQL Injection", "Screen.SCREEN",
    );
}

/**
 * Gets the ranking attribute of the DatabaseFieldScreen object
 * @return The ranking value
 */
protected Integer getRanking()
{
    return new Integer( 70 );
}

/**
 * Gets the title attribute of the DatabaseFieldScreen object
 * @return The title value
 */
public String setTitle()
{
    return ( "How to Perform SQL Injection" );
}

/**
 * Constructor for the DatabaseFieldScreen object
 * @param s Description of the Parameter
 */
public void start( WebSession s )
{
    try
    {
        setup( s );
        if ( connection == null )
        {
    
```



# Security-related Functions



Weak or Nonstandard Cryptography

Non-Secure Network Communications

Access Control Vulnerabilities

- ✓ MD5 is no longer considered secure for highly sensitive and business critical applications, SHA1 is also considered broken though no practical attacks have been identified.

*“Microsoft is banning certain cryptographic functions from new computer code, citing increasingly sophisticated attack”, <http://www.eweek.com/article2/0,1759,1859751,00.asp>*

- ✓ The following example was from a content management systems password reset function.

```
/**
 * Generates a random 10 characters password.
 *
 * @return the generated password.
 */
public static synchronized String generate()
{
    return Long.toString(Math.abs(random.nextLong()) % MAX_RANDOM_LENGTH, Character.MAX_RADIX);
}
```

---

*The biggest failure in encryption is not often the algorithm used but more often than not it is the failure to properly identify what data needs to be encrypted and making sure that the appropriate encryption is always utilized.*

---



# Input/Output Validation and Encoding Errors



- SQL Injection Vulnerabilities
- Cross-Site Scripting Vulnerabilities
- OS Injection Vulnerabilities
- Custom Cookie/Hidden Field Manipulation

Have we not learned to **NEVER** trust the user, all input needs to be validated?

What is the problem with the code below?

```

public void doGet( HttpServletRequest req, HttpServletResponse res )
    throws IOException, ServletException
{
    String pageName = getParameter("pageName") != null ? "" :
getParameter("pageName");
    log.info("Request for page: "+pageName);
    String forward = "/" + pageName + "?" + req.getQueryString();
    RequestDispatcher disp = req.getRequestDispatcher(forward);
    disp.forward( req, res );
}
    
```

---

**It is not all about SQL Injection and XSS (though those are still a huge problem).**

---



# Error Handling & Logging Vulnerabilities



**Insecure Error Handling**  
**Insecure or Inadequate Logging**

**Consider the following code example:**

```
public void doPost( HttpServletRequest req, HttpServletResponse res )
    throws IOException, ServletException
{
    RequestDispatcher disp = null;
    String user = getParameter("user") !=null ? "" : getParameter("user");
    String pwd = getParameter("pwd") !=null ? "" : getParameter("pwd");
    if(!validUser(user,pwd)) {
        log.warn("Invalid login received from: " + user + " password:" +pwd);
        disp = req.getRequestDispatcher("/jsp/invalidLogin.jsp");
    } else {
        log.info("Successful login attempt from: " + user);
        disp = req.getRequestDispatcher("/jsp/loginSuccess.jsp");
    }
    disp.forward( req, res );
}
```

---

**There really are two major issues with logging:**

- 1. Lacking a consistent logging framework.**
  - 2. Logging the wrong data or breaking company policy and regulations (think: PCI)**
- 





# Insecure Components



**Unsafe Native Methods**  
**Unsupported API**  
**Improper Use of 3rd Party Application Frameworks**

Developers need to understand where the utilities provided by the framework begin and end when related to security. Consider the following code from a .NET web application.

```

<head>
  <title>Registration Form Please Sign-In</title>
</head>
<%String loader = Request.Params["loader"]; %>
<body onload = "<%=loader%>">
...
</body>
    
```

Even if you have Microsoft's page validation enabled (the default) you are still vulnerable.


As we focus our efforts to fix the low hanging fruit, the attacks are moving to the application layer.

There are many undocumented APIs that exist as public interfaces in the JDK or the .NET framework

Many of these interfaces may bypass internal member data validation that if used directly could crash the JVM (or lead to more serious vulnerabilities <http://www.blackhat.com/presentations/win-usa-03/bh-win-03-schoenfeld.pdf>)



# Coding Errors



- Buffer Overflow Vulnerabilities
- Format String Vulnerabilities
- Denial of Service Errors
- Race Conditions

Use of native libraries (System.loadLibrary, [DllImport]) can also expose your web application to this more traditional style of attack.

## What's wrong with this code?

```
protected void doGet(HttpServletRequest request, HttpServletResponse response) {
    InputStreamReader inStr = new
    InputStreamReader(request.getInputStream());
    BufferedReader in = new BufferedReader(inStr);
    while(in.readLine() != null) {
        //process the request
        ...
    }
}
```

---

*Most modern day web applications are immune to the more traditional “overflow” style of attacks, but anytime the user is able to control data that reaches an internal system the possibility exists.*

---

[http://documents.iss.net/whitepapers/IBM\\_X-Force\\_WP\\_final.pdf](http://documents.iss.net/whitepapers/IBM_X-Force_WP_final.pdf)



## Follow The Path: The Checklist



- **Security-related functions**

- Weak or Nonstandard Cryptography
- Non-Secure Network Communications
- Application Configuration Vulnerabilities
- Access Control Vulnerabilities



- **Input/Output validation and encoding errors**

- SQL Injection Vulnerabilities
- Cross-Site Scripting Vulnerabilities
- OS Injection Vulnerabilities
- Custom Cookie/Hidden Field Manipulation



- **Error handling and logging vulnerabilities**

- Insecure Error Handling
- Insecure or Inadequate Logging



- **Insecure Components**

- Unsafe Native Methods
- Unsupported Methods
- Improper use of 3rd Party Application Frameworks



- **Coding errors**

- Buffer Overflow Vulnerabilities
- Format String Vulnerabilities
- Denial of Service Errors
- Race Conditions



## Where?

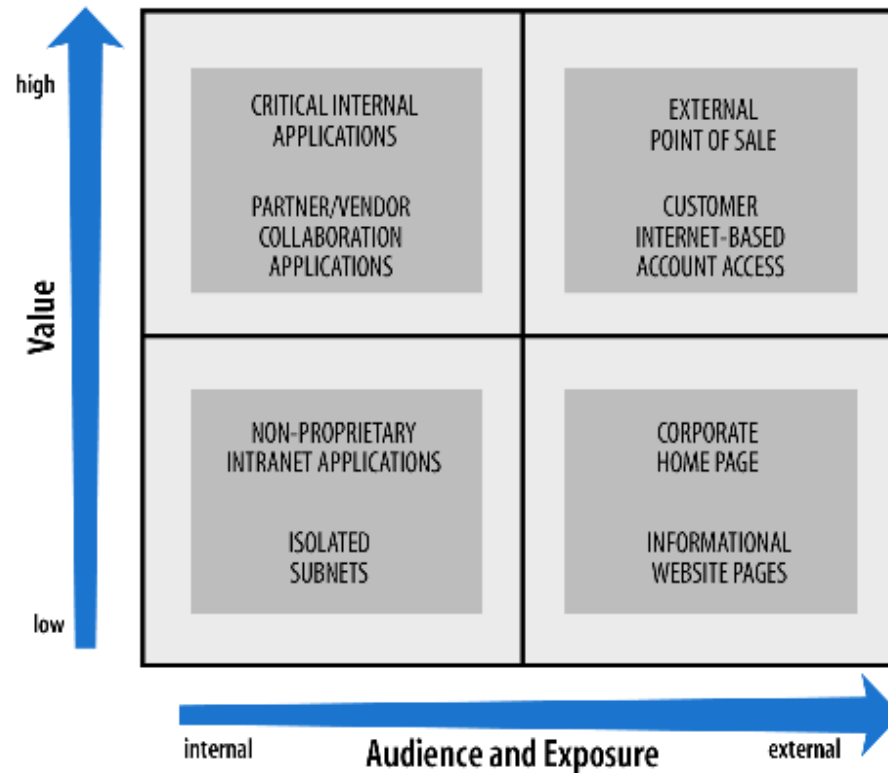
- **Baking security into requirements:** gathering security requirements/needs, abuse cases, and threat modeling
- **Baking security into design:** security design patterns, security reviews and threat modeling
- **Baking security into development:** secure coding guidelines, tools, and audit
- **Baking security into testing:** negative testing, thinking like the bad guy and “red teams”
- **Baking security into deployment:** secure deployment guidelines, secure update mechanisms (patching) and much, much more!



# When?

- As often as is practical
  - Prioritize the most critical applications
  - Separate legacy from new development
  - Customer facing vs. internal

FIGURE 1: VALUE AND EXPOSURE

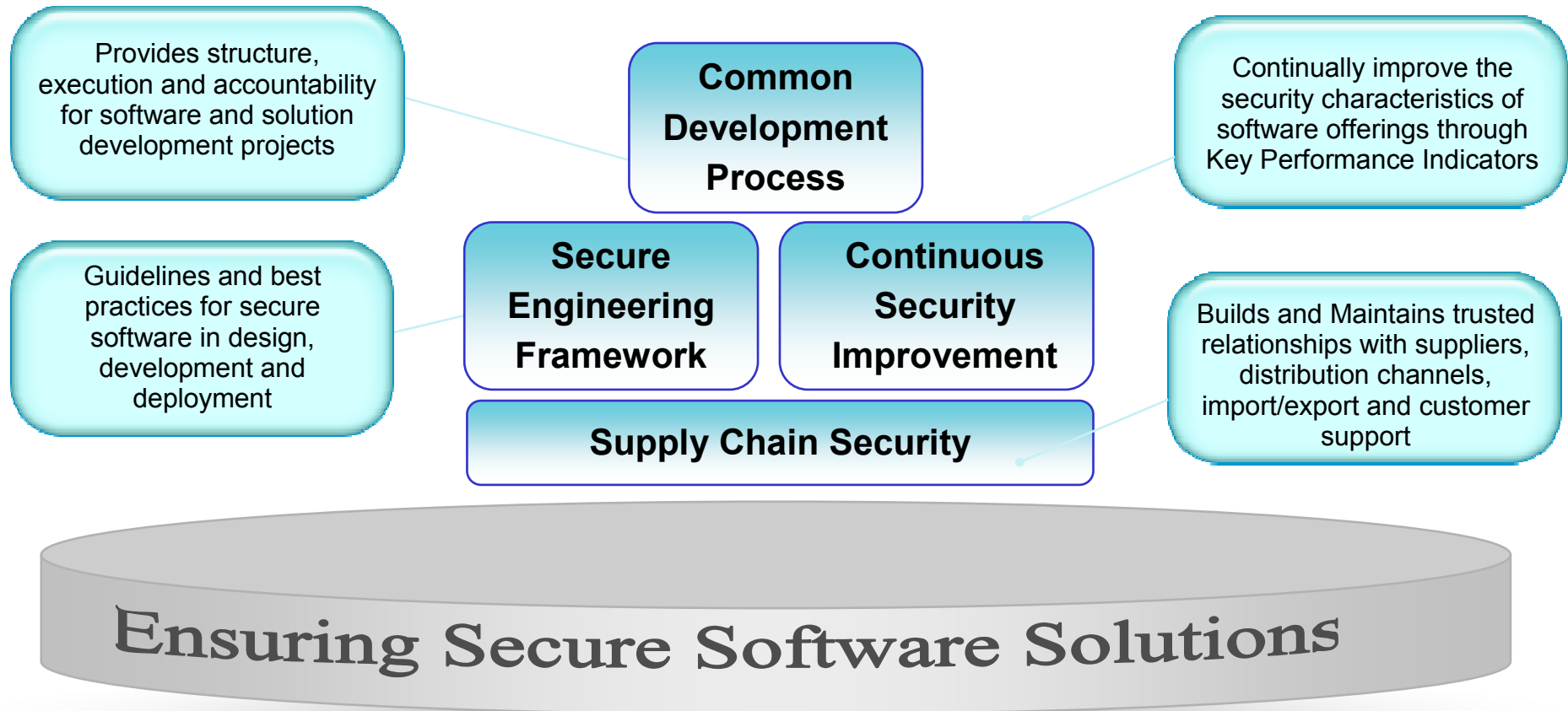


## How: Objectives for Practical Security

- Improve *existing* development process, not create new one
- Maximize security impact of personnel and technologies
- Use models as initial framework and tailor to individual organization
- Select model with consideration for future requirements



# IBM Secure Engineering Initiative



Link to Security Engineering Framework:

<http://www.redbooks.ibm.com/redpieces/abstracts/redp4641.html?Open>

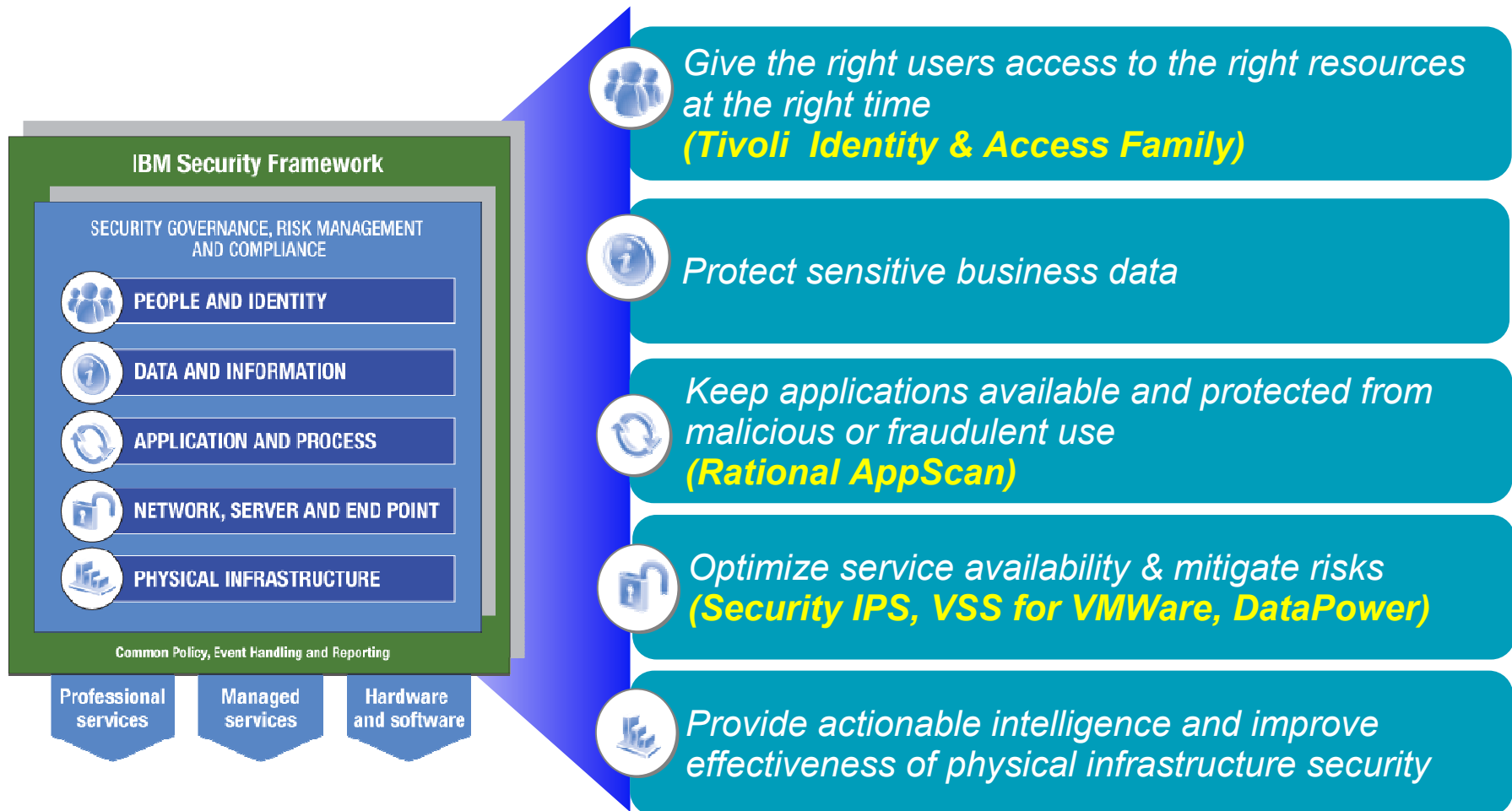


SecureDevelopment Framework						
Inception		Elaboration		Construction	Transition	
Secure Development Framework Goal						
	<b>1</b> <b>Predict Business Risk</b>		<b>2</b> <b>Minimize Attack Surface</b>		<b>3</b> <b>Code Securely</b>	<b>5</b> <b>Final Review</b>
	<b>1.1.</b> Engage Security Expert <b>1.2.</b> Determine Predictive Threat Index <b>1.2.1.</b> High <sup>1</sup> <b>1.2.2.</b> Medium <sup>2</sup> <b>1.2.3.</b> Low <b>1.3.</b> Map user requirements to security requirements <b>1.4.</b> Determine authorization requirements <b>1.5.</b> Identify key internal/external compliance objectives <b>1.6.</b> Define application security test process & deliverables <b>1.7.</b> Adjust project plan to include security resources <b>1.8.</b> Contract secure code review (5.1) <b>1.9.</b> Contract manual pen test (5.2)		<b>2.1.</b> Architecture Review <b>2.2.</b> Identify secure design techniques <b>2.3.</b> Identify certified components responsible for security functions <b>2.4.</b> Document attack surface <b>2.5.</b> Create threat modeling document <b>2.6.</b> Review/modify security requirements <b>2.7.</b> Identify components for Secure Code Review <b>2.8.</b> Define secure integration with external systems <b>2.9.</b> Define security test requirements <b>2.10.</b> Determine authorization requirements model <b>2.11.</b> Update Security Master Test Plan <b>2.12.</b> Update test schedule and budget		<b>3.</b> Code / Build <b>3.1.</b> Certified components <b>3.2.</b> Static Code Assessment <b>3.3.</b> Dynamic Application Assessment <b>4.</b> Test / Verify <b>4.1.</b> Peer Code Review <b>4.2.</b> Static Code Assessment <b>4.3.</b> Dynamic Application Assessment	<b>5.</b> Final Security Review <b>5.1.</b> Secure Code Review <b>5.2.</b> Manual penetration testing <b>5.3.</b> Static Code Analysis <b>5.4.</b> Dynamic Application Assessment <b>5.5.</b> Review of all bugs for possible security vulnerabilities <b>5.6.</b> Review threat model for possible late developing threats
	<b>Activities</b>		<b>Deliverables</b>		<b>Tools</b>	
<ul style="list-style-type: none"> <li>- Security Expert assigned</li> <li>- Predictive Threat Index</li> <li>- Preliminary security requirements defined</li> <li>- Security test strategy</li> </ul>		<ul style="list-style-type: none"> <li>- Architecture Review</li> <li>- Minimized application attack surface</li> <li>- Application security test roles</li> <li>- Threat Model</li> <li>- Security requirements in well defined components</li> <li>- Application security test plans</li> <li>- Certified components identified</li> </ul>		<ul style="list-style-type: none"> <li>- Working application</li> </ul>		<ul style="list-style-type: none"> <li>- Problems, defects, enhancements logged</li> <li>- Detailed test results</li> <li>- Validated requirements</li> <li>- Updated test results in centralized location</li> <li>- Certification</li> </ul>
<ul style="list-style-type: none"> <li>- Security Knowledge Portal</li> <li>- Security Consultant</li> <li>- Design Review Checklist</li> <li>- Requirements Software</li> <li>- Predictive Threat Index calculator</li> </ul>		<ul style="list-style-type: none"> <li>- Security Knowledge Portal</li> <li>- Architectural Review Checklist</li> <li>- Threat Modeling Software</li> <li>- Platform dependent coding checklist</li> <li>- Certified Components</li> </ul>		<ul style="list-style-type: none"> <li>- Security Knowledge Portal</li> <li>- Static Code Analyzer</li> <li>- Dynamic Application Analysis Tool</li> <li>- Certified Components</li> <li>- Security Development Guidelines</li> </ul>		<ul style="list-style-type: none"> <li>- Security Knowledge Portal</li> <li>- Security Auditor</li> <li>- Static Code Analyzer</li> <li>- Dynamic Application Analysis Tool</li> <li>- Final Review Checklist</li> </ul>





# IBM Security Framework includes integrated solutions for Web Application Security



# Secure Code Development and Vulnerability Management – IBM Rational® AppScan®

- **A market leader for Web application vulnerability scanning**
  - ▶ A leader in numerous industry “bake offs”
- **Automatically scans Web applications for vulnerabilities**
  - ▶ SQL Injection
  - ▶ Cross-site Scripting
- **Provides clear recommendations on how to remediate identified vulnerabilities**
- **Scans Web sites for embedded malware**
  - ▶ Protect your Web site from distributing the next Conficker to every Web site visitor
  - ▶ Powered by the IBM Internet Security Systems™ X-Force® malware prevention system



# Enabling the Operationalization of Security Testing

Address Web Application Vulnerabilities in three ways:

1

**Enable Security Specialists**

- AppScan® Standard
- AppScan Enterprise

2

**Embed Security into Development**

- AppScan Source
- AppScan Tester

3

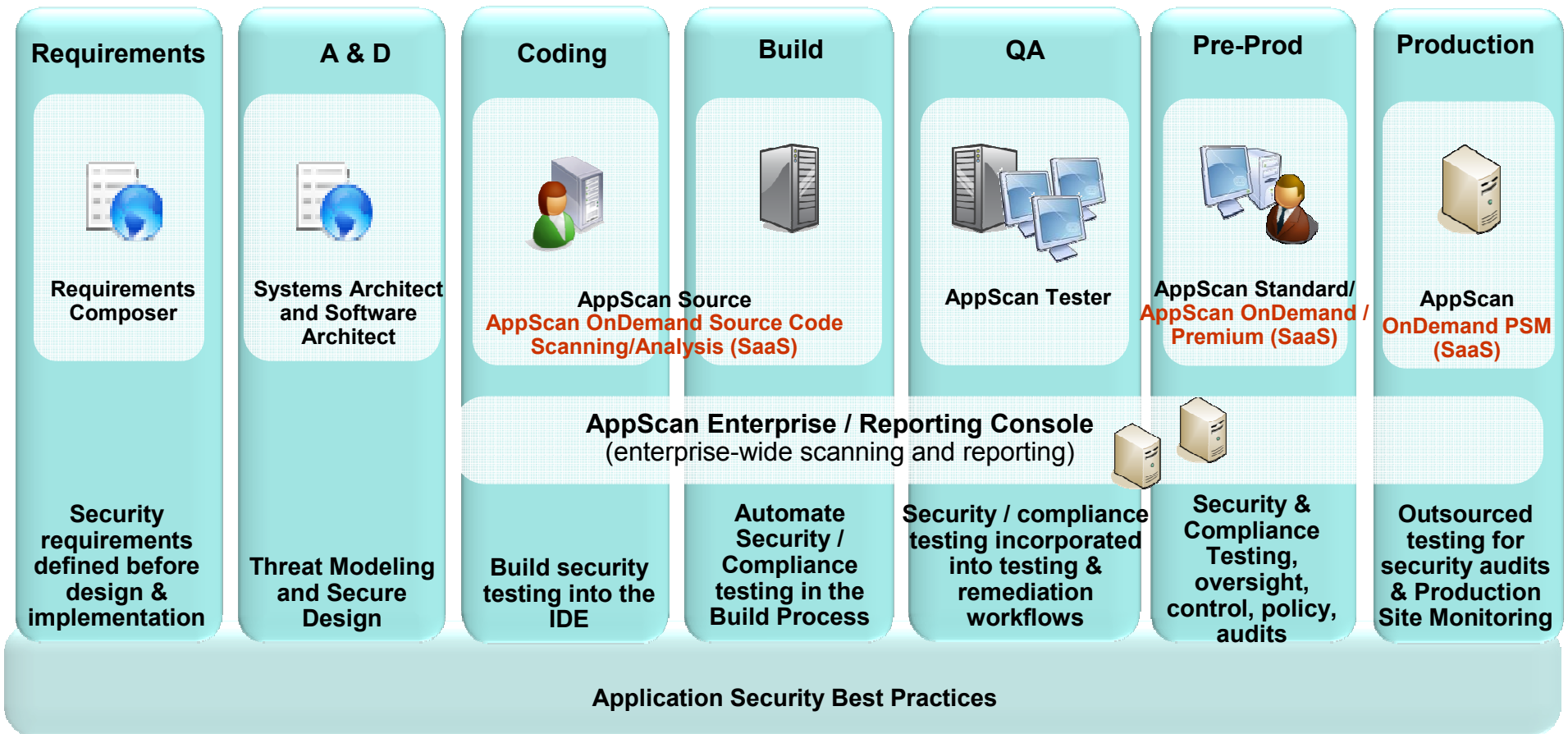
**Outsource Security Testing**

- AppScan OnDemand
- AppScan Security Consulting

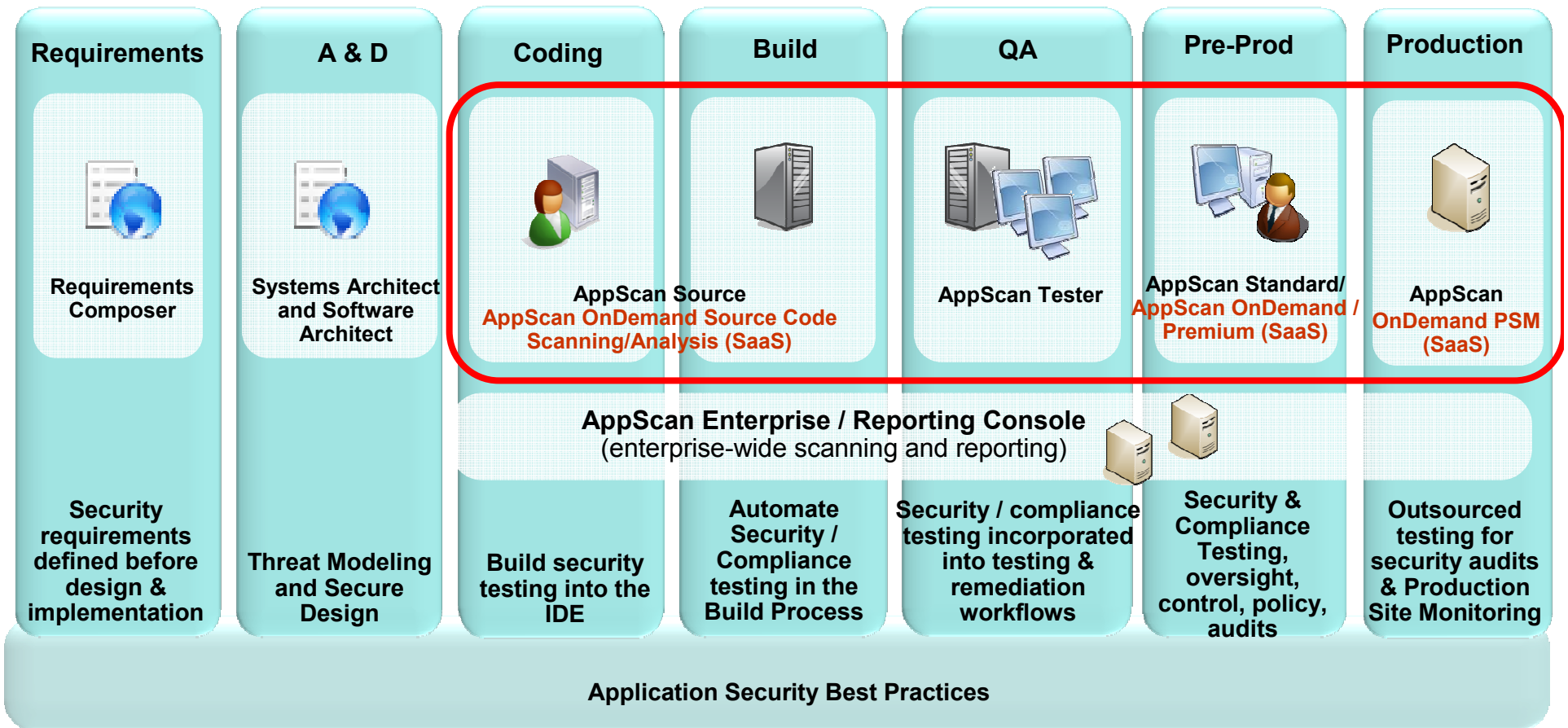
**Control, Monitor, Collaborate and Report Web Application Security Testing**  
(AppScan Reporting Console)



# Security in the SDLC View

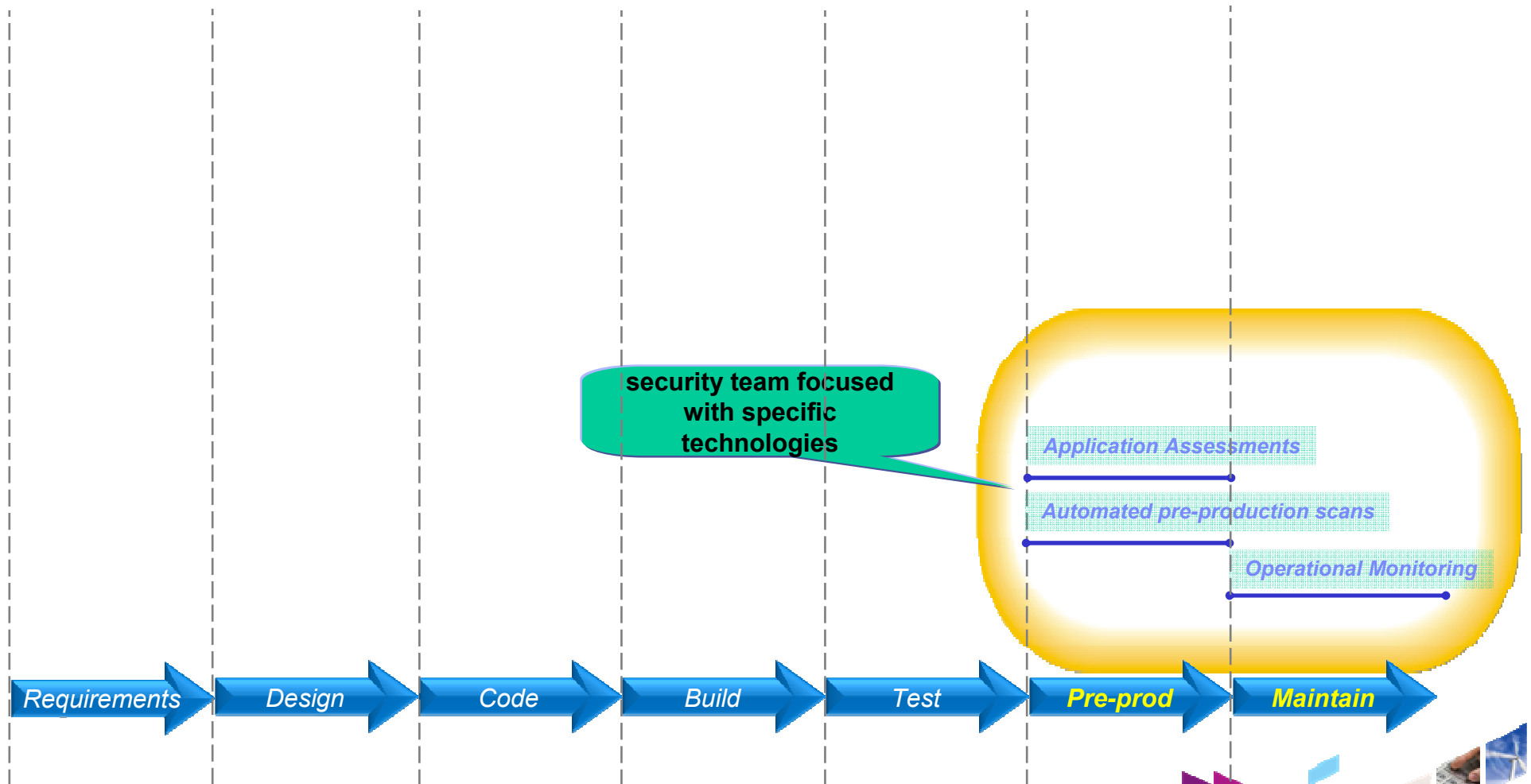


# Security in the SDLC View



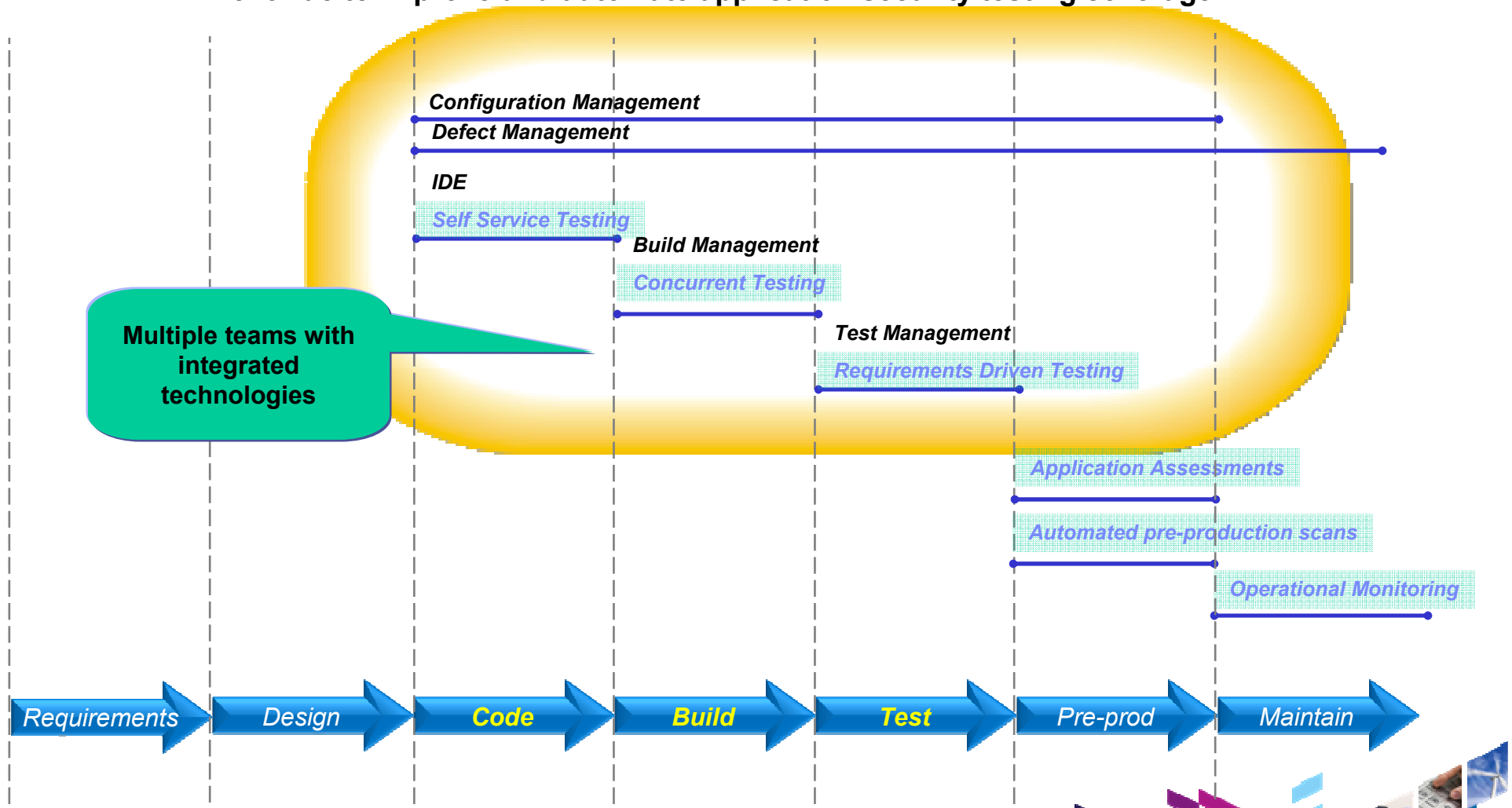
# Security in the SDLC View

Initial improvement and automation of application security testing coverage



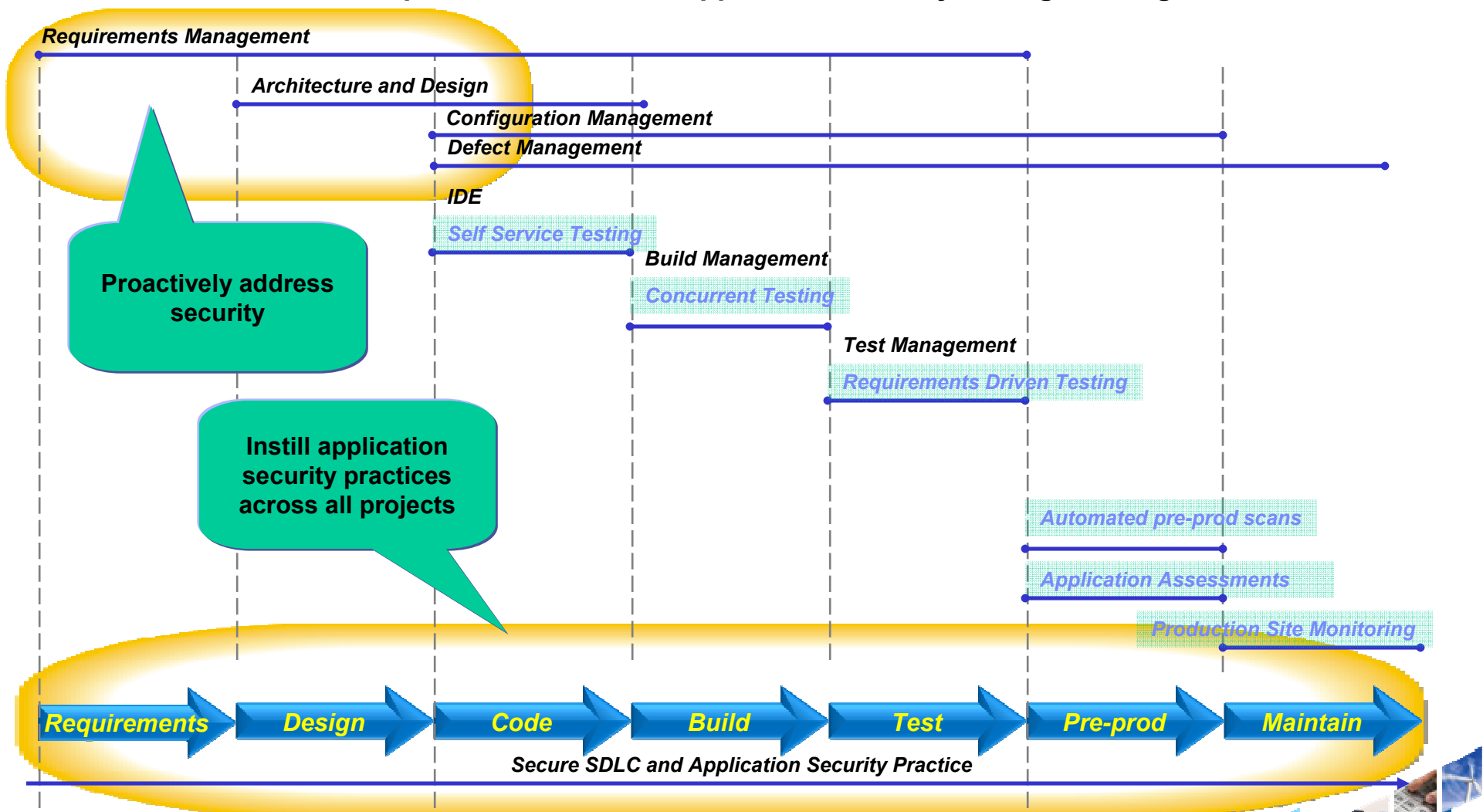
# Security in the SDLC View

Improve and automate collaboration of security issues  
Continue to Improve and automate application security testing coverage



# Security in the SDLC View

Improve and automate collaboration of security issues  
 Continue to Improve and automate application security testing coverage





# Best Practices and Business Case Formulation

- Best Practices

- ▶ Establish a policy and internal standards for software security
- ▶ Build software security into the SDLC (including sourcing decisions)
- ▶ Establish policies and procedures for continual vulnerability detection and remediation
- ▶ Automate where possible for coverage, accuracy, efficiency and trending
- ▶ Establish a realistic Plan B. And C.

- Business Case

- ▶ Money savings via catching vulnerabilities and other security flaws earlier (well before they reach production)
- ▶ Money and reputational savings when NERC fines for failure to comply with minimum vulnerability identification, mitigation and remediation processes are avoided
- ▶ Utility's and their customers' savings when major outages are prevented via responsible cyber security practices



# For More Information...

To learn more about using IBM's secure engineering framework

<http://www.redbooks.ibm.com/redpieces/abstracts/redp4641.html?Open>



# Questions





**IBM**

**Thank You**