

Security Development Lifecycle: Applications and Infrastructure

October 2004

Speaker: Himanshu Dwivedi

@stake



Where Security & Business IntersectSM

Agenda

- **Challenges facing application and infrastructure security today**
- **Rationale behind the lifecycle approach to security**
- **Benefits of integrated security in application and infrastructure lifecycles**
- **@stake Solutions**

Challenges: Mission Impossible?

- You're responsible for the security of a *live* mission critical (and revenue generating) application
- A security assessment just found significant vulnerabilities
- You own application but don't know where to start
- You need to plug the holes fast

What to do?

Challenges: Application Development

- **Security is being considered too late during the genesis of an application.**
- **@stake Hoover study:**
 - 70% of security defects were design flaws
 - Nearly half of those defects could have been fixed *inexpensively* in the design phase
- **More dependencies result in hard-to-secure applications:**
 - Third-party tools and libraries
 - Application servers
 - Infrastructure components
- **Required collaboration between infrastructure and application teams isn't happening.**
- **Assessment is a driver for change, but only infrastructure components are being assessed.**

Challenges: Infrastructure

- **The importance of patch management**
 - “Worm-centric” and “Microsoft-centric”
 - Exposes organizational inefficiencies
- **Dissolving network perimeters**
 - VPN and extranet/partner networks
 - XML and SOAP
- **Dissolving application boundaries**
 - Active Directory / LDAP
 - DNS
- **Complexity of attacks on infrastructure increasing**
 - Routing protocols
 - Security products

Questions: Lifecycle Approach

- **Where does security fit into the organization?**
- **When does it make sense to think about security?**
- **How do you avoid making the same mistakes in the future?**

Traditional Systems Development Lifecycle

Requirements

Design

Implement

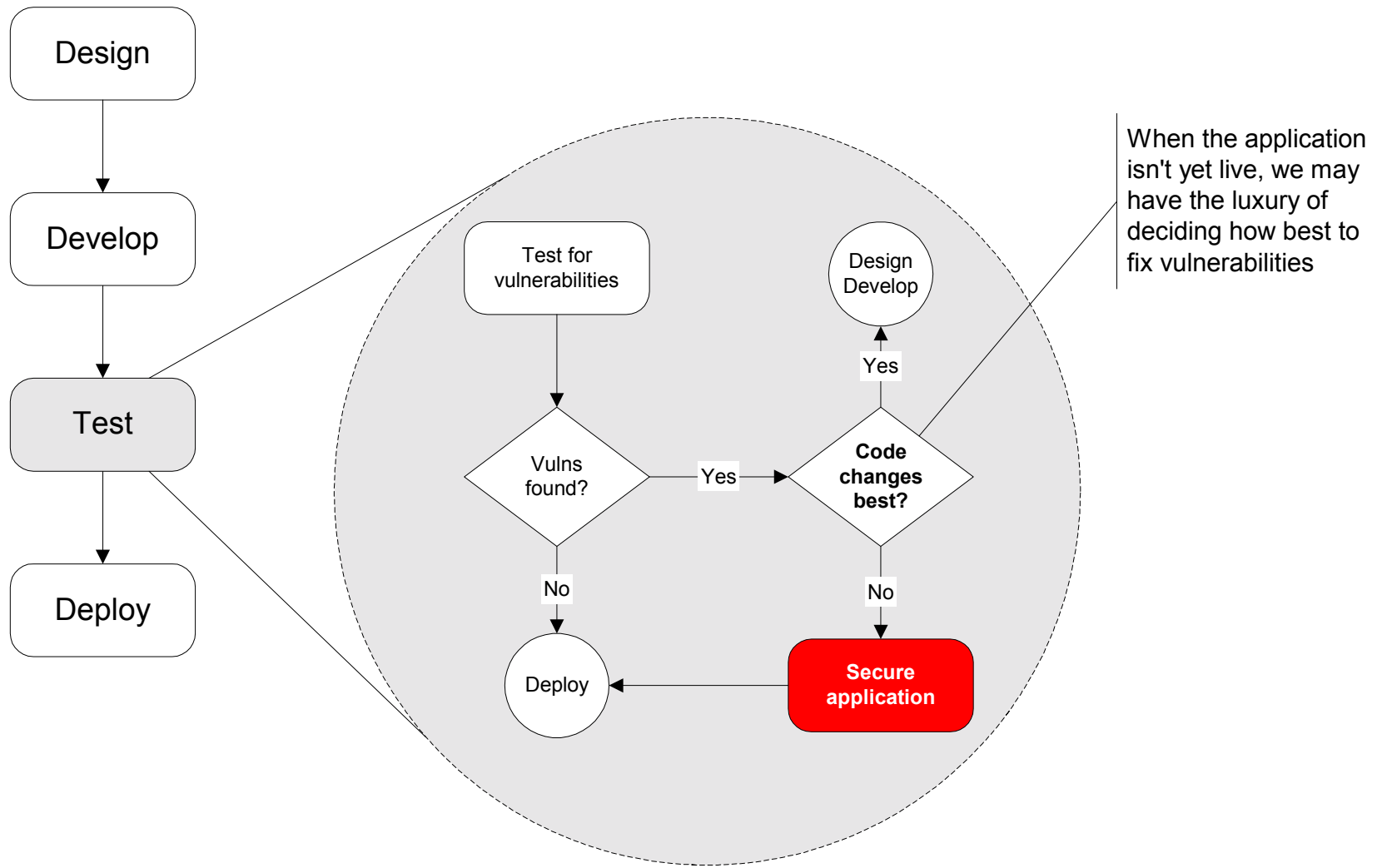
Test

Deploy

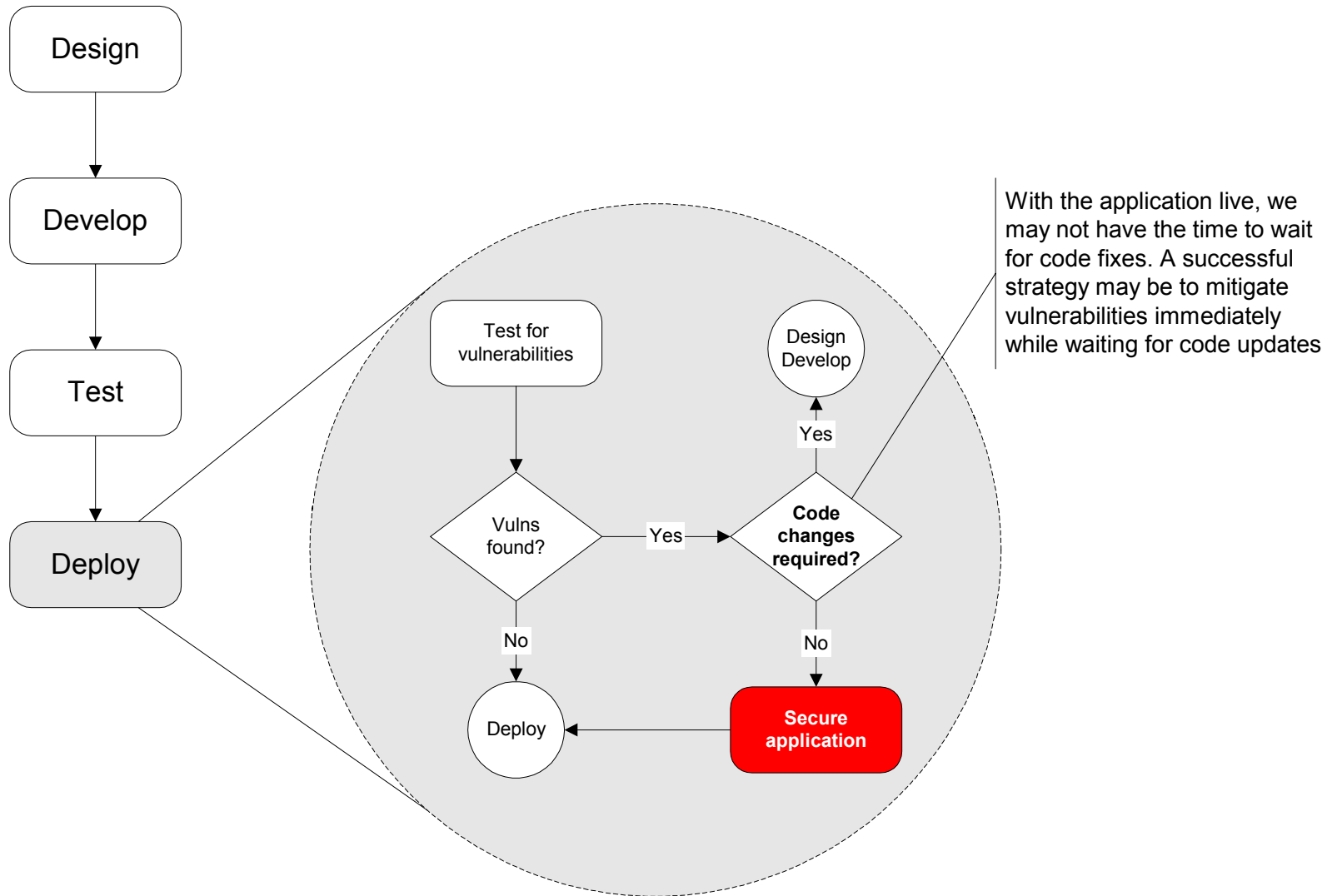
Maintain

- **Traditional waterfall lifecycle model**
- **Has designated entry and exit points**
- **Can be reentrant or repeatable**
- **Can take lessons learned from one project and apply to the next**
- **Applicable to software and infrastructure**
- **An excellent blueprint for architecting security**

When Are Vulnerabilities Mitigated? (Optimistic)



When Are Vulnerabilities Mitigated? (Realistic)



Design Criteria for the Lifecycle Approach

Requirements

Design

Implement

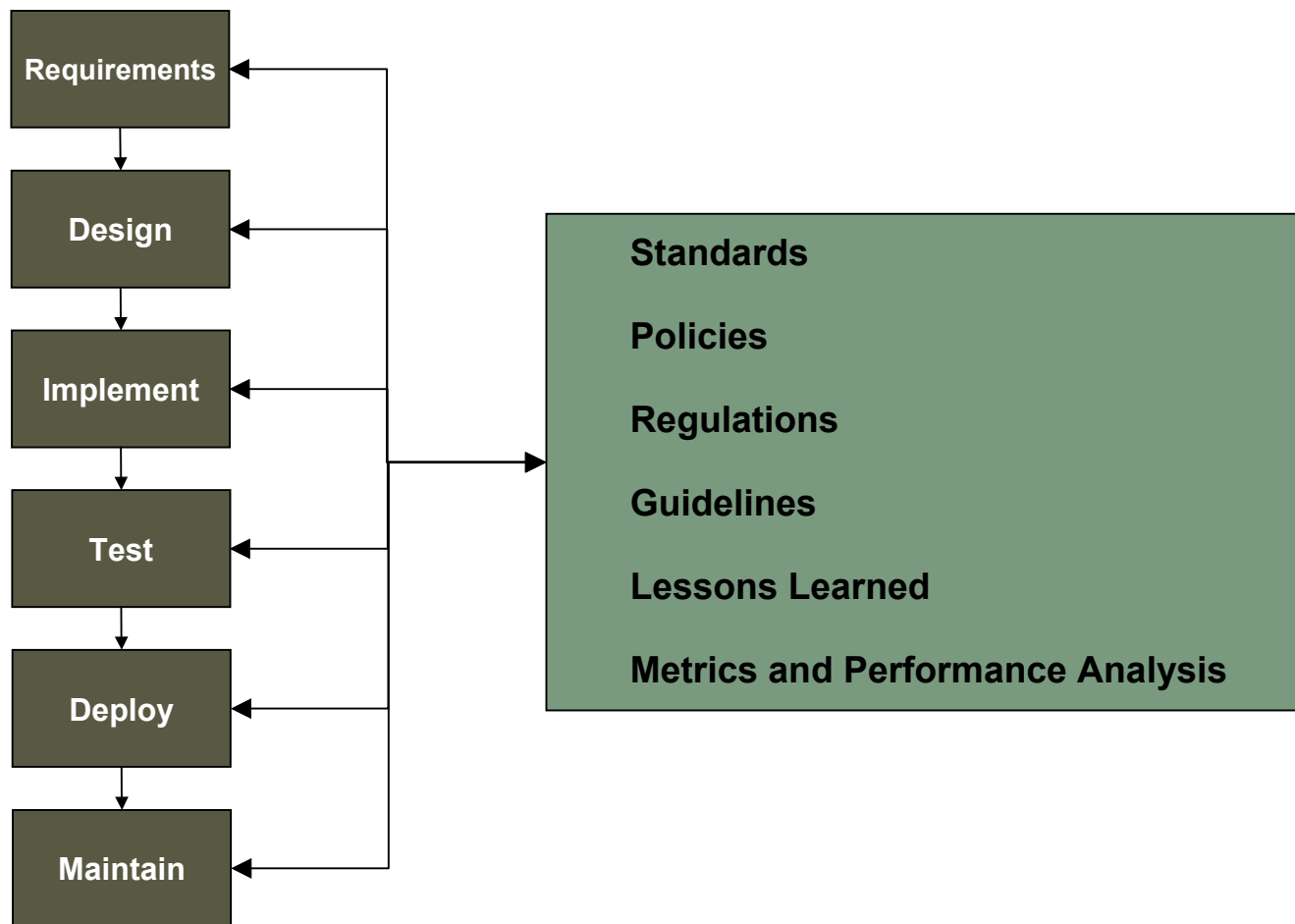
Test

Deploy

Maintain

- **Make security persistent across:**
 - Lifecycle
 - Organization
- **Minimize organizational and process impact**
 - Leverage current processes and people
 - Use well-known vocabulary and tools
- **Provide high level framework and intermediate steps**
 - Where am I going?
 - How do I start?

@stake Secure Development Lifecycle



Lifecycle Prerequisites... and Deliverables!

- **Policy**

- Information Classification
- Business Risk Profiling and Analysis
- Regulatory and Industry Requirements

- **Standards**

- Coding practices
- Strategic technology initiatives
- Standardization on platforms, encryption, physical and environmental requirements

Requirements Analysis

Requirements

Design

Implement

Test

Deploy

Maintain

- **Application**

- Authentication and authorization
- Data flow analysis
- Data classification and storage
- Application administration
- Interaction with legacy and backend systems
- Change management

- **Infrastructure**

- Availability and performance
- Network Management and Monitoring
- Redundancy
- Remote Access
- Anti-virus and IDS
- Backup
- Firewalls
- External Network Connections
- Incident Response Guidelines

Design Analysis

Requirements

Design

Implement

Test

Deploy

Maintain

- **Application**

- Threat Model
- Input Data Types
- Security Use Cases
- Security Architecture

- **Infrastructure**

- Physical Security
- Application Use Cases to define system requirements and sizing
- Authentication Services
- Naming and Directory Services
- Performance baselines for availability requirements

Development / Implementation

Requirements

Design

Implement

Test

Deploy

Maintain

- **Application**

- Coding Standards
- Centralized Security Modules
- Security Configuration and Interfaces
- Evaluate Known Security Vulnerabilities
- User Presentation-layer Security Design and Review
- Middleware Security Design and Review
- Database Security Design and Review

- **Infrastructure**

- Hardware and Operating Environment Standards
- Centralized Security Audit, Authentication, Access Control Review and Implementation
- Secure Network Device Configuration Design and Review
- Secure Operating System Configuration Design and Review
- Evaluate Known Security Vulnerabilities

Testing

Requirements

Design

Implement

Test

Deploy

Maintain

- **Application**

- Security Bug Tracking
- Data Validation
- Session Management
- Access Control
- Cryptography
- Third-party Components

- **Infrastructure**

- System Vulnerability Scanning
- Network Vulnerability Scanning
- Redundancy/Failover Testing
- Security Fail-safe Posture Verification

Deployment

Requirements

Design

Implement

Test

Deploy

Maintain

- **Application**

- Secure Management Procedures
- Key management and cryptographic deployment requirements
- Production access control and authentication deployment
- Removal of test and debug functionality

- **Infrastructure**

- Monitoring Network and System Deployment, including integrity, availability, and performance
- Validation of production environments
- Actualization of monitoring requirements
- System baselining and integrity checksums

Maintenance

Requirements

Design

Implement

Test

Deploy

Maintain

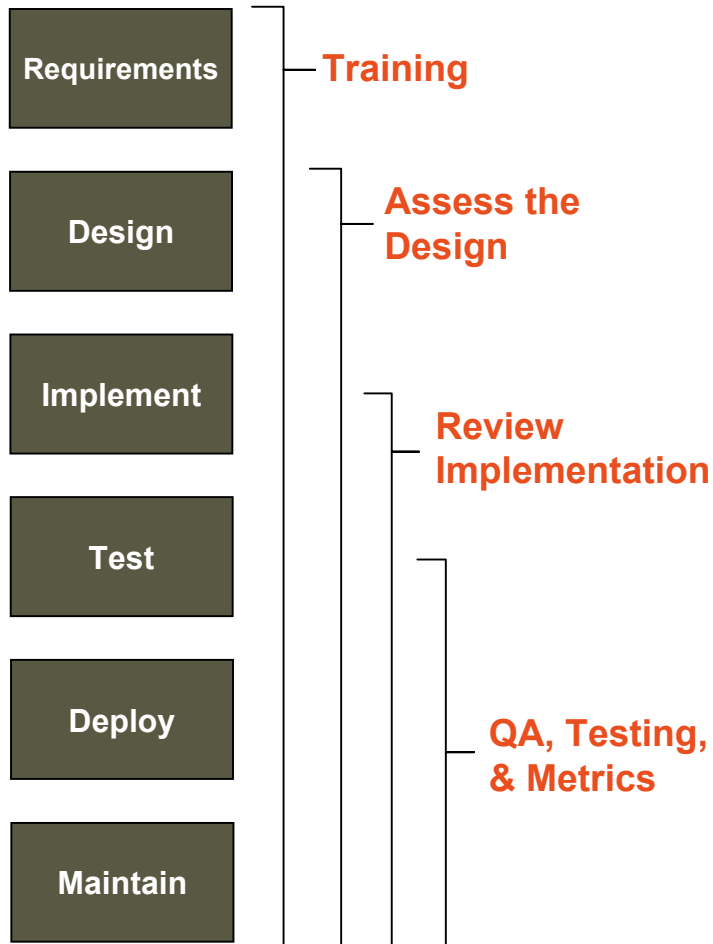
- **Application**

- Change management and control
- Development access into production environment
- Readiness and staging environment maintenance and validation of new releases
- Remediation of security flaws

- **Infrastructure**

- Log Analysis and Alerting
- Security Incident Details and Reporting
- Patch Identification, Maintenance, and Verification

Continuous Security Improvement



- **@stake engagements can cover all phases:**
 - Training
 - Architecture Assessment - Application & Network
 - Code Review and Secure Build Design
 - Vulnerability Assessment
- **Each phase has specific security exit criteria**
- **Training ensures that IT staff learn from the vulnerabilities uncovered**
- **Metrics enable the organization to track its progress and continuously improve**
- **The result: as time goes on, fewer assessments are needed, and are reserved for key applications and infrastructure deployments**

Questions?

Himanshu Dwivedi
hdwivedi@stake.com

Gus Gougas
ggougas@stake.com