

Web Application Security

OWASP

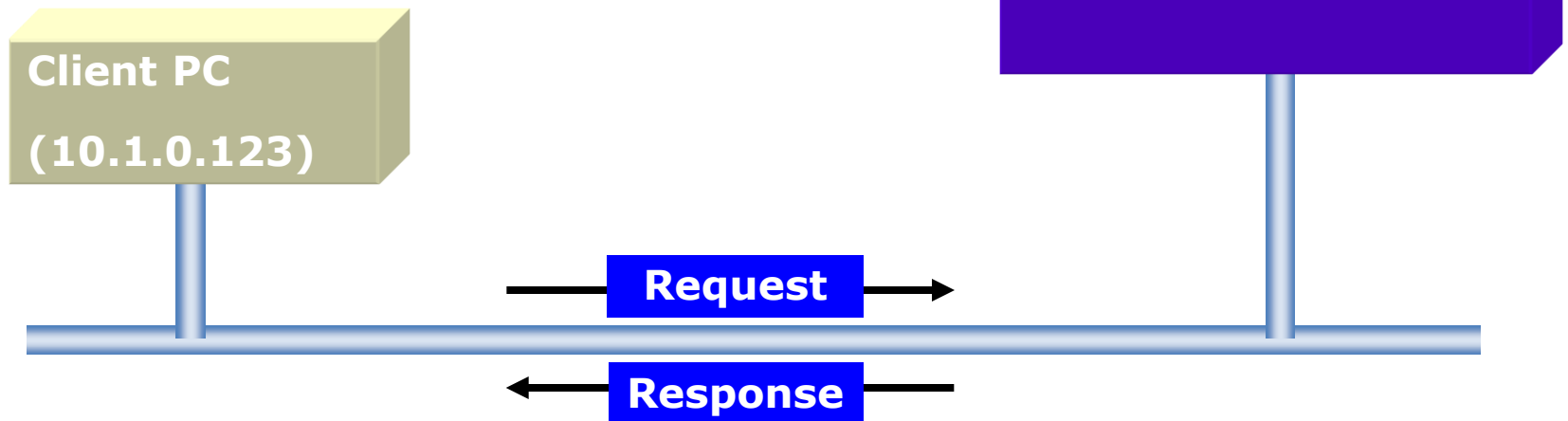
Overview

- Background
- Web app vulnerabilities
- Securing web apps

HTTP

- Hypertext Transfer Protocol

- “Hypertext Transfer Protocol (HTTP) is a communications protocol for the transfer of information on intranets and the World Wide Web. Its original purpose was to provide a way to publish and retrieve hypertext pages over the Internet.”
- <http://en.wikipedia.org/wiki/HTTP>



HTTP Request - GET

- Form data encoded in the URL
- Most common HTTP method used on the web
- Should be used to retrieve information, not for actions that have side-effects

HTTP Request - GET



GET <http://www.mysite.com/kgsearch/search.php?catid=1> HTTP/1.1

Host: www.mysite.com

User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.8.1.13) Gecko/20080311
Firefox/2.0.0.13

Accept:

text/xml,application/xml,application/xhtml+xml,text/html;q=0.9,text/plain;q=0.8,image/png,*/*;q=0.5

Accept-Language: en-us,en;q=0.5

Accept-Encoding: gzip,deflate

Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7

Keep-Alive: 300

Connection: keep-alive

Referer: http://www.mysite.com/

HTTP Requests - POST

- Data is included in the body of the request.
- Should be used for any action that has side-effects
 - Storing/updating data, ordering a product, etc...

HTTP Requests - POST



POST <http://www.mysite.com/kgsearch/search.php> HTTP/1.1

Host: www.mysite.com

User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.8.1.13) Gecko/20080311 Firefox/2.0.0.13

Accept: text/xml,application/xml,application/xhtml+xml,text/html;q=0.9,text/plain;q=0.8,image/png,*/*;q=0.5

Accept-Language: en-us,en;q=0.5

Accept-Encoding: gzip,deflate

Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7

Keep-Alive: 300

Connection: keep-alive

Referer: http://www.mysite.com/

catid=1

GET v. POST Security

- There information contained in parameters can tell a user a lot about how your application works
- GET parameters are easily visible in the address bar
- POST parameters are hidden from the *average* user
 - Users can still view source code
 - Users can still view the packets
 - Users can still intercept & modify web requests

Web Sites

- **No applications**
- **Static pages**
- **Hard coded links**

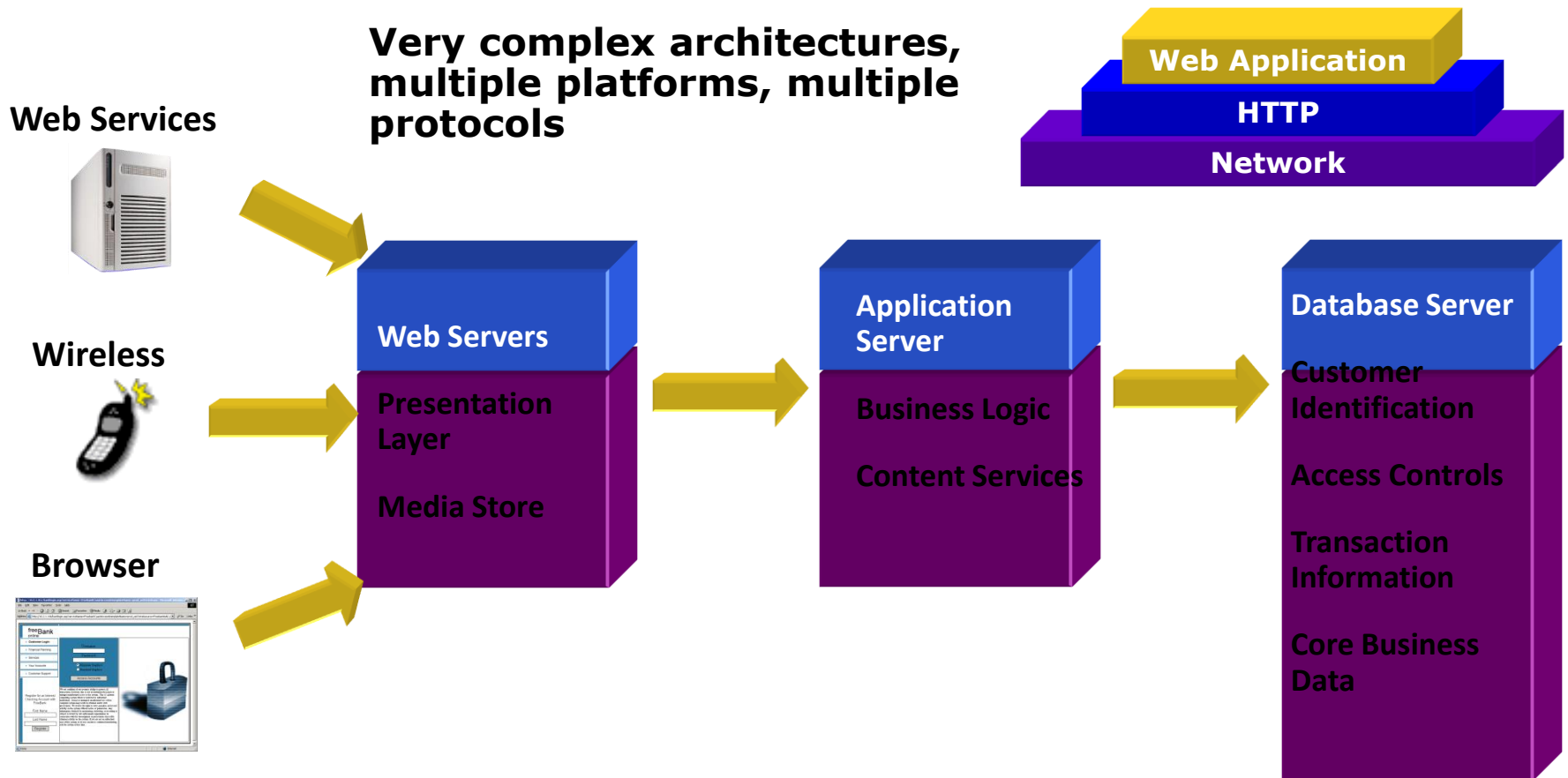
Browser



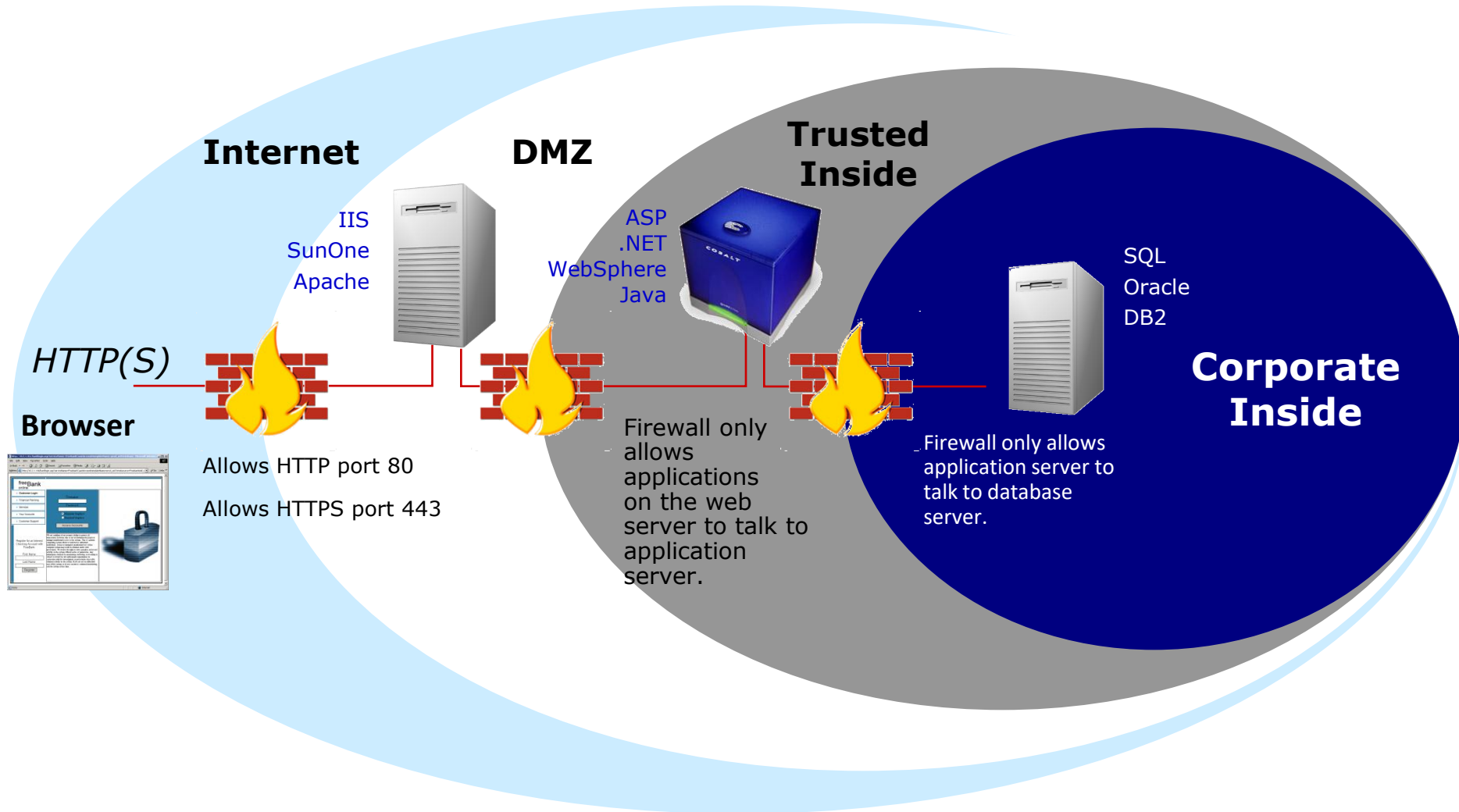
Web Server



Web Applications



Web Applications Breach the Perimeter

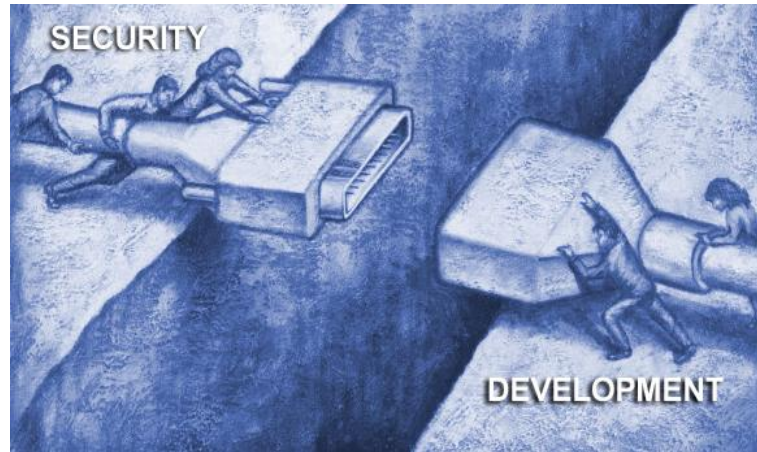


Why Web Application Vulnerabilities Occur

The Web Application Security Gap

Security Professionals Don't Know The Applications

"As a Network Security Professional, I don't know how my companies web applications are supposed to work so I deploy a protective solution...but don't know if it's protecting what it's supposed to."



Application Developers and QA Professionals Don't Know Security

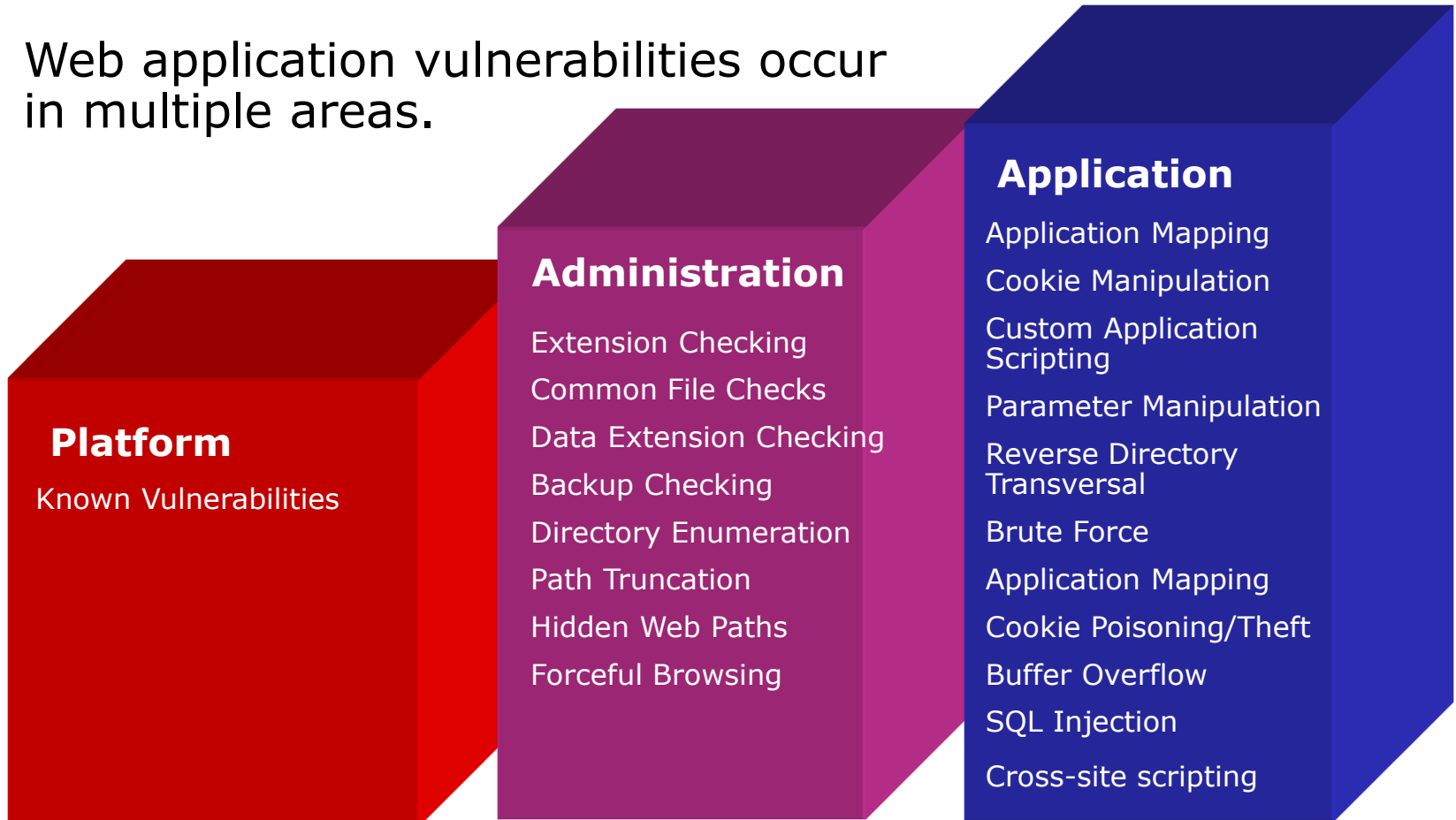
"As an Application Developer, I can build great features and functions while meeting deadlines, but I don't know how to develop my web application with security as a feature."

Web Application Vulnerabilities

- Technical Vulnerabilities
 - Result of insecure programming techniques
 - Mitigation requires code changes
 - Detectable by scanners
 - [http://example/order.asp?item=<script>alert\('p0wned'\)</script>&price=300.00](http://example/order.asp?item=<script>alert('p0wned')</script>&price=300.00)
- Logical Vulnerabilities
 - Result of insecure program logic
 - Most often due to poor decisions regarding trust
 - Mitigation often requires design/architecture changes
 - Detection often requires humans to understand the context
 - <http://example/order.asp?item=toaster&price=30.00>

Web Application Vulnerabilities

Web application vulnerabilities occur in multiple areas.



Web Application Vulnerabilities

Platform:

- Known vulnerabilities can be exploited immediately with a minimum amount of skill or experience – “script kiddies”
- Most easily defensible of all web vulnerabilities
- MUST have streamlined patching procedures



Web Application Vulnerabilities

Administration

Extension Checking

Common File Checks

Data Extension
Checking

Backup Checking

Directory
Enumeration

Path Truncation

Hidden Web Paths

Forceful Browsing

Administration:

- Less easily corrected than known issues
- Require increased awareness
- More than just configuration, must be aware of security flaws in actual content
- Remnant files can reveal applications and versions in use
- Backup files can reveal source code and database connection strings

Web Application Vulnerabilities

Application

Application Mapping
Cookie Manipulation
Custom Application Scripting
Parameter Manipulation
Reverse Directory Transversal
Brute Force
Application Mapping
Cookie Poisoning/Theft
Buffer Overflow
SQL Injection
Cross-site scripting

Application Programming:

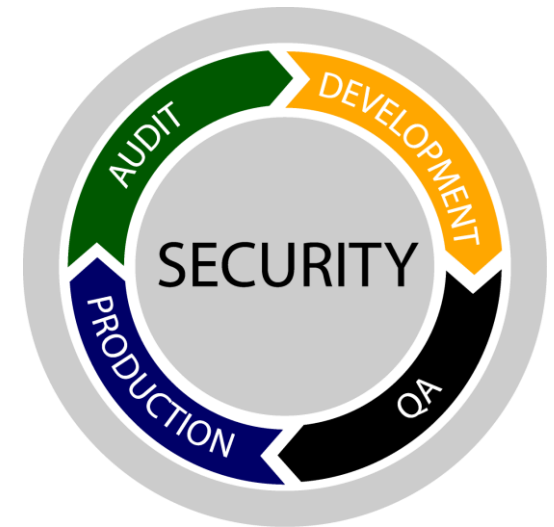
- Common coding techniques do not necessarily include security
- Input is assumed to be valid, but not tested
- Unexamined input from a browser can inject scripts into page for replay against later visitors
- Unhandled error messages reveal application and database structures
- Unchecked database calls can be 'piggybacked' with a hacker's own database call, giving direct access to business data through a web browser

How to Secure Web Applications

- Incorporate security into the lifecycle
 - Apply information security principles to all software development efforts
- Educate
 - Issue awareness, Training, etc...

How to Secure Web Applications

- Incorporating security into lifecycle
 - Integrate security into application requirements
 - Including information security professionals in software architecture/design review
 - Security APIs & libraries (e.g. ESAPI, Validator, etc.) when possible
 - Threat modeling
 - Web application vulnerability assessment tools



How to Secure Web Applications

Educate

- Developers – Software security best practices
- Testers – Methods for identifying vulnerabilities
- Security Professionals – Software development, Software coding best practices
- Executives, System Owners, etc. –
Understanding the risk and why they should be concerned