KNOWLEDGE consulting group KCG

Information assurance services with expertise in Cyber Threat Analysis and FISMA Compliancy

Lost in Translation: Understanding the Hacker Mindset

11710 Plaza America Dr., Ste 520, Reston, VA 20190

www.KnowledgeCG.com

Voice: 703.467.2000 Fax: 703.547.0322

Agenda

- Introductions
- Secure SDLC Approach
- Penetration Testing: Real-world Examples
- Lessons Learned
- Open Q&A

Speaker Bios

Paul Nguyen

Paul Nguyen, CISSP, CISA, CGEIT, currently serves as the Vice President of Cyber Solutions for Knowledge Consulting Group where he advises federal Chief Information Officers (CIO) and CISOs on various cybersecurity issues. He is also a former software developer in a CMM Level 5 organization, former CISO of the U.S. Court Services and Supervision Agency, and attack/exploitation practitioner with renowned firms such as @stake and Neohapsis. Mr. Nguyen holds a Bachelors and Masters degrees from Carnegie Mellon University.

Ryan Stinson

Ryan Stinson is the lead for Cyber Assessment Services at Knowledge Consulting Group. The KCG Cyber Assessment Group provides penetration testing, code reviews, secure architecture assessments, web application security testing, and vulnerability research. Mr. Stinson holds a Bachelor of Science in Computer Science, as well as certifications as a Certified Information Systems Security Professional (CISSP), GIAC Certified Incident Handler and GIAC Certified Penetration Tester.

Secure SDLC Approach

Where to Begin?

Before we even get into development of an application, one must rationalize the various frameworks, standards, best practices, software development methodology, and any drivers that may impact the security posture of the system. The security approach should adapt to the software development methodology chosen and ensure the proper checks and balances are in place.

FIPS 199	NIST SP 800-53 rev. 3	NIST Risk Management Framework
OWASP Top 10	Top 25 Common Weakness Enumeration	DISA Secure Technical Implementation Guide
DoD 8500.2	ICD 503	.....

Software Dev. Methodology

- RAD
- Waterfall
- Prototyping
- RUP
- Agile
- ...

Top 25 CWE and OWASP Top 10 – Deeper Look

Understanding the Top 25 and OWASP Top 10 provides a basis for understanding how malicious attackers think. Permutations of the vulnerabilities will vary from platform to platform (e.g. .NET, J2EE, AJAX, etc...). The goal is to provide context for engineers and developers to understand the various mistakes that can be avoided (I've made the same mistakes as a reformed developer!).

Rank	ID	Name
[1]	CWE-79	Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')
[2]	CWE-89	Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection')
[3]	CWE-120	Buffer Copy without Checking Size of Input ('Classic Buffer Overflow')
[4]	CWE-352	Cross-Site Request Forgery (CSRF)
[5]	CWE-285	Improper Access Control (Authorization)
[6]	CWE-807	Reliance on Untrusted Inputs in a Security Decision
[7]	CWE-22	Improper Limitation of a Pathname to a Restricted Directory ('Path Traversal')
[8]	CWE-434	Unrestricted Upload of File with Dangerous Type
[9]	CWE-78	Improper Neutralization of Special Elements used in an OS Command ('OS Command Injection')
[10]	CWE-311	Missing Encryption of Sensitive Data
[11]	CWE-798	Use of Hard-coded Credentials
[12]	CWE-805	Buffer Access with Incorrect Length Value
[13]	CWE-98	Improper Control of Filename for Include/Require Statement in PHP Program ('PHP File Inclusion')
[14]	CWE-129	Improper Validation of Array Index
[15]	CWE-754	Improper Check for Unusual or Exceptional Conditions
[16]	CWE-209	Information Exposure Through an Error Message
[17]	CWE-190	Integer Overflow or Wraparound
[18]	CWE-131	Incorrect Calculation of Buffer Size
[19]	CWE-306	Missing Authentication for Critical Function
[20]	CWE-494	Download of Code Without Integrity Check
[21]	CWE-732	Incorrect Permission Assignment for Critical Resource
[22]	CWE-770	Allocation of Resources Without Limits or Throttling
[23]	CWE-601	URL Redirection to Untrusted Site ('Open Redirect')
[24]	CWE-327	Use of a Broken or Risky Cryptographic Algorithm
[25]	CWE-362	Race Condition

OWASP Top 10 – 2010 (New)
A1 – Injection
A2 – Cross-Site Scripting (XSS)
A3 – Broken Authentication and Session Management
A4 – Insecure Direct Object References
A5 – Cross-Site Request Forgery (CSRF)
A6 – Security Misconfiguration (NEW)
A7 – Insecure Cryptographic Storage
A8 – Failure to Restrict URL Access
A9 – Insufficient Transport Layer Protection
A10 – Unvalidated Redirects and Forwards (NEW)

Threats to Your Applications

Area	Threats/Attacks
Input Validation	Buffer overflows; Cross-Site Scripting; SQL Injection; Cannonicalization attacks
Authentication	Network eavesdropping; Brute force attacks; Dictionary attacks; Cookie replay attacks; Credential theft
Authorization	Elevation of privilege; Disclosure of confidential data; Data tampering; Luring attacks
Configuration Management	Unauthorized access to administration interfaces; Unauthorized access to configuration stores; Retrieval of clear text; configuration secrets; No individual accountability; Over privileged process and service accounts
Sensitive Data	Access sensitive data in storage; Network eavesdropping; Information Disclosure
Session Management	Session Hijacking; Session Replay; Man in the Middle
Cryptography	Poor key generation or key management; weak or custom encryption
Parameter Manipulation	Query string manipulation; Form field manipulation; Cookie manipulation; HTTP header manipulation
Exception Management	System or Application Details Are Revealed; Denial of service
Auditing and Logging	User denies performing an operation; Attacker exploits an application without trace; Attacker covers his tracks

Secure Development Lifecycle

SDL struggles:

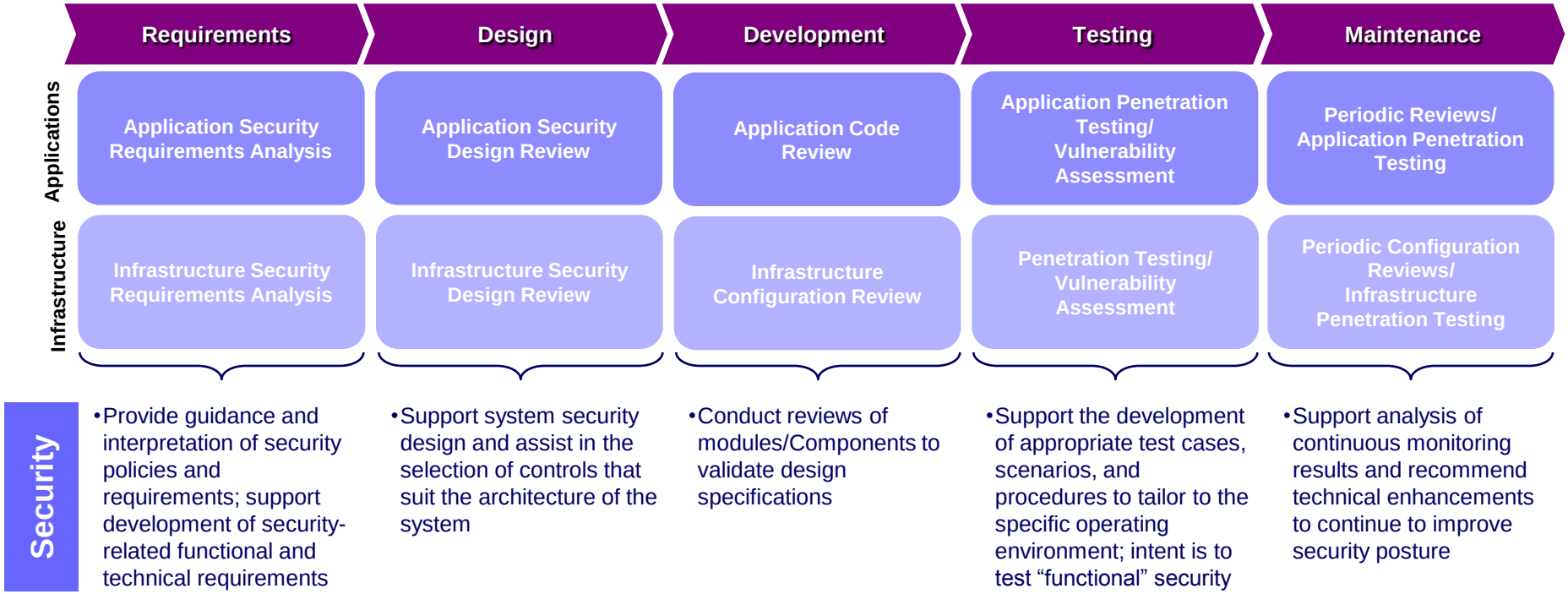
- How do we validate application security throughout the development process
- How do I incorporate security into our development process
- How do I measure externally developed applications adhere to a similar process

SDL program capability:

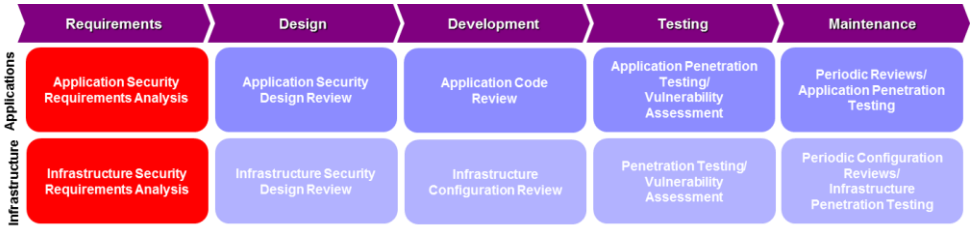
- Do you have the fundamental framework to incorporate security
- What is are the required phases to address security from initial design through deployment
- What additional skill set development is required
- What additional processes can accommodate our current requirements
- What does a mature SDL process consist of, and what will suit our organization

Putting the SDL into Action

System Development Life Cycle Support



Requirements Analysis

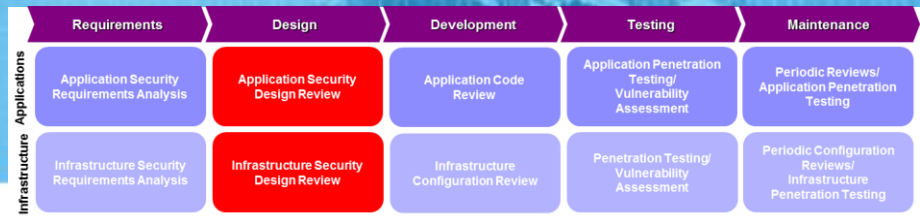


Key Considerations

- Understand the threat environment and potential pitfalls (Top 25 CWE/OWASP)
- FIPS 199 Categorization
- NIST 800-53 Baseline Requirements: biggest challenge comes from interpreting the requirements within the context of the architecture
- Review your agency's policies, standards, and Enterprise Architecture to ensure proper alignment

Security's Role

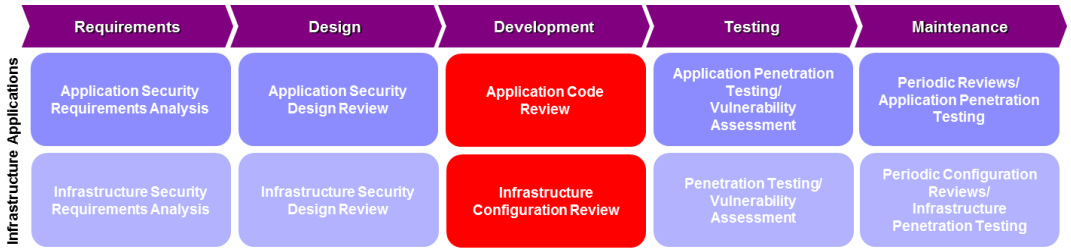
- Conduct threat modeling prior to requirements phase (understand what threat scenarios you're concerned about)
- Providing enough interpretation and expertise in defining the proper security requirements (understanding the malicious mindset to prevent the mistakes)
- Injecting the proper security requirements within the context of the architecture



Design Considerations

Area	Guidelines
Input Validation	Don't trust input; validate input: length, range, format and type; constrain, reject, sanitize input
Authentication	Use strong password policies; Don't store credentials; Encrypt communication channels to secure authentication tokens; use HTTPs only with Forms cookies
Authorization	Use least privilege accounts; Consider granularity of access; Enforce separation of privileges
Configuration Management	Use least privileged service accounts; Don't store credentials in plaintext; Use strong authentication and authorization on administrative interfaces; Don't use the LSA; avoid storing sensitive information in the web space
Sensitive Data	Don't store secrets in software; Enforce separation of privileges; Encrypt sensitive data over the wire; Secure the channel
Session Management	Partition site by anonymous, identified and authenticated; reduce the timout; avoid storing sensitive data in Session; Secure the channel
Parameter Manipulation	Don't trust fields the client can manipulate (Query string, Form fields, Cookie values, HTTP headers)
Exception Management	Use structured exception handling (try-catch); Only catch and wrap exceptions if the operation adds value/information; Don't reveal sensitive system or app info; Don't log private data (passwords ... etc.)
Cryptography	Don't roll your own; XOR is not encryption; RNGCryptoServiceProvider for random numbers; Avoid key management (use DPAPI); Cycle your keys
Auditing and Logging	identify malign or malicious behavior; know your baseline (what does good traffic look like); instrument to expose behavior that can be watched (the big mistake here is typically app instrumentation is completely missing)

Development



Identify Source Code to be Reviewed

- Choose a threat path. Start with highest risk.
- If critical processing modules identified, review those modules first.
- Identify source code modules for components on the threat path
- Identify source code for utility classes and functions used by modules on the threat path

Review code for known classes of problems

- Arbitrary code injection: buffer overflows and format string attacks
- Race conditions
- Inadequate privilege checking
- Canonicalization issues
- Error handling
- Information leakage
- Script injection
- Cryptographic implementation errors

Code Review Toolkit

- Source Code Analyzers provide developers with the ability to conduct unit testing for security vulns
- Examples: Fortify, Ounce Labs, Parasoft…and even open source

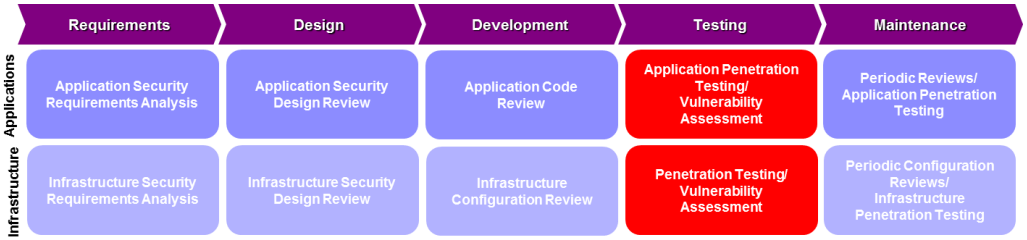
Stack Overflow example

The size of the string copied into a buffer exceeds its size
What happens when argv[1] has a length greater than 7?

```
#include <string.h>   // for strcpy
int main(int argc, char *argv[])
{
        char buff[8];
        strcpy( buff, argv[1] );
       return 1;
}
```

Buffers on heap can be exploited as well.

Testing



Key Considerations

- Choose a comprehensive approach to include all components:
  - Web App Pen. Testing
  - Infrastructure Pen. Testing
  - SOA Application Pen Testing
- Testing should occur as early as possible to provide time for remediation
- Make sure the rules are established for the testing (put a Rules of Engagement in place)

Security's Role

- Simulation of threat vectors and attack scenarios identified during threat modeling
- Identification of attacks that could compromise the system or data
- Documentation of the vulnerabilities and risks
- Providing recommendations to support remediation

Penetration Testing: Real-World Examples

Overview of the Application Penetration Test

System Information

- 4-tier J2EE Internet-facing Web Application
  - Apache JBOSS
  - Windows Server 2008
  - MS SQL Server 2008
- Contains highly sensitive information related to personally identifiablee information (PII)
- Development timeframe was 6 months start to finish
- Leveraged a COTS product as the core with custom development to meet the agency's functional requirements

Our Objectives

- Provide security subject matter expertise to support all phases of the SDLC and integrate into the development project plan
- Obtain an Authorization to Operate (ATO) for the new system as required under FISMA
- Ensure risks are identified as early in the SDLC as possible to minimize the cost of potential vulnerabilities later in the SDLC
- Conduct final Security Assessment using Application Penetration Testing techniques and infrastructure vulnerability assessment

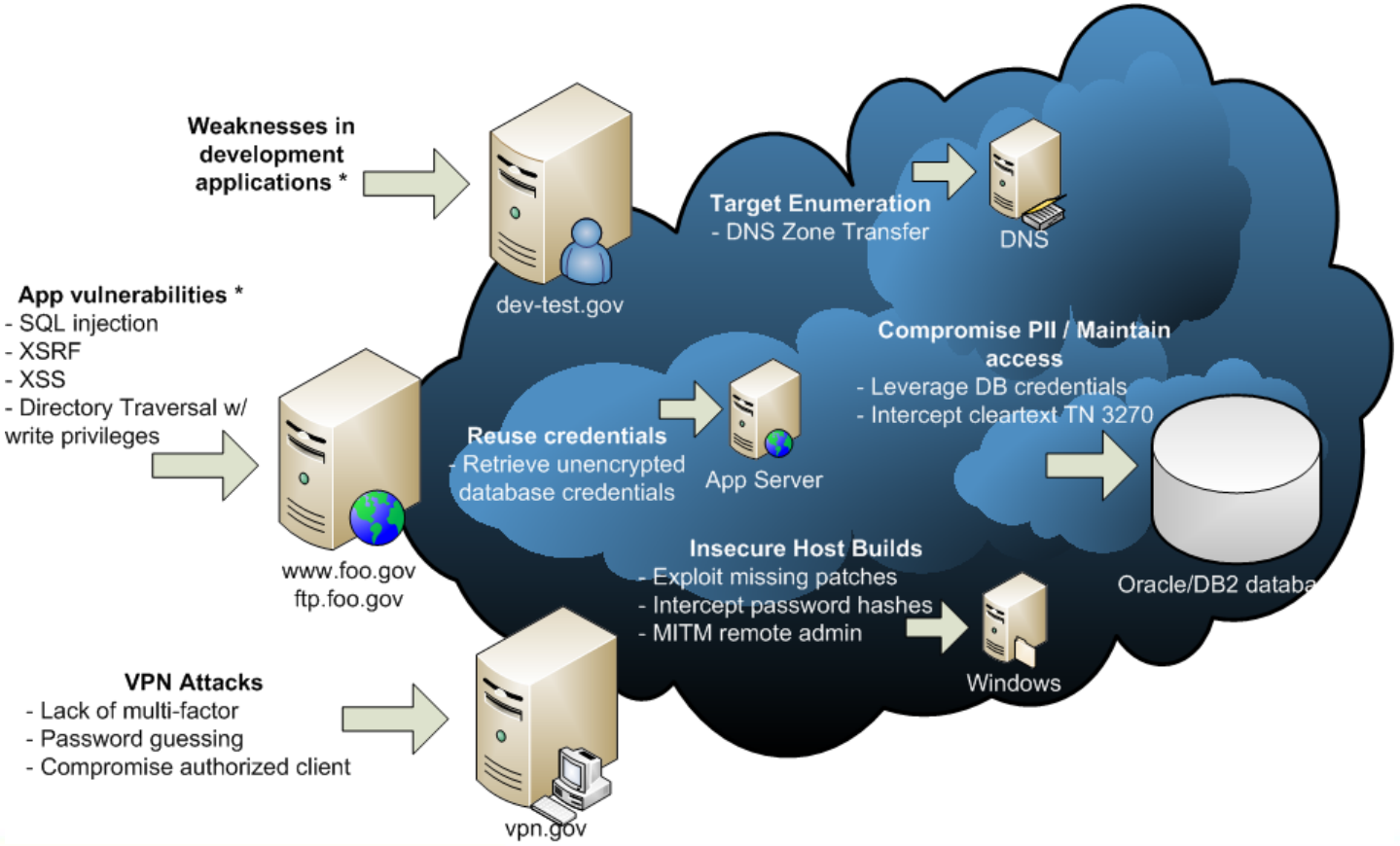
Anatomy of an attack

The attack strategy used by professional white hat and black hat purposes follows five significant and noteworthy stages, culminating in the discovery and possible removal of sensitive data, including PII.

Open Source Intel	Public Accessibility	Privilege Escalation	Data-centric Exploitation	Data Exfiltration
<u>Identification of potential client-side targets</u> • Spear phishing preparation • Review of social network profiles • Email address harvesting <u>Identification of publicly accessible services</u> • Service fingerprinting • Public vulnerability research • Network profiling	• Development environment exploitation • Application layer vulnerabilities • VPN password guessing • Malicious authorized users • Authorized client compromise	• Exploitation of internal vulnerabilities • Network spidering • Expand access	• Database exploitation • Credential Intercept • Man-in-the-middle attacks • Mainframe attacks • Credential Reuse	• Data harvesting • Removal of data from the interior network • Transmit externally via SMTP, HTTPS

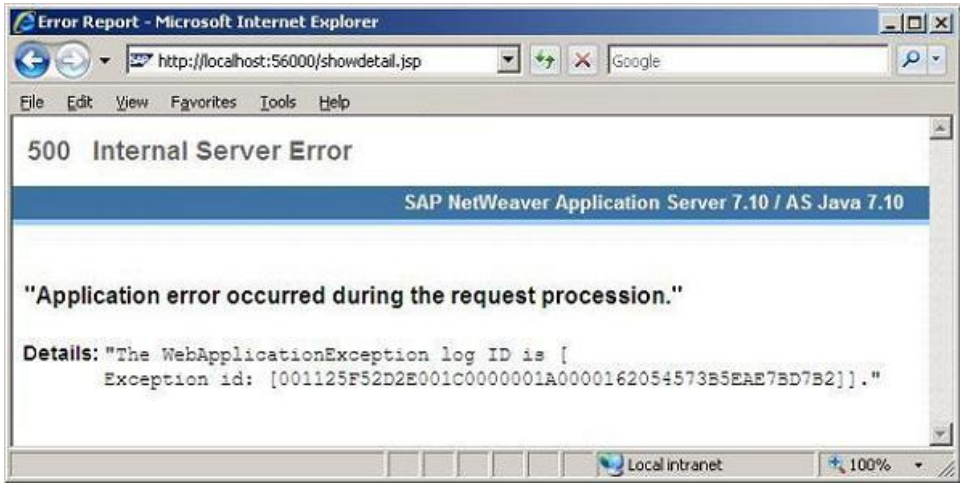
Anatomy of an attack

The attack strategy used by professional white hat and black hat purposes follows five significant stages, culminating in the discovery and possible removal of sensitive data, including PII.



Input Validation – Am I properly handling user input?

What if I inject *"'" or 1=1;--* into the "Search" box…



Something isn't right…looks like insufficient scrubbing of my input and definitely wasn't expecting it

Top 25 CWE	OWASP
CWE-732: Incorrect Permission Assignment for Critical Resource	Category A6 - Security Misconfiguration or Category A4 - Insecure Direct Object References

Error Handling – Giving Us Way Too Much Information

Building on the last step…we're getting SQL fragments. Sign of a SQL Injection…

```
        at org.apache.tomcat.util.net.JIoEndpoint$Worker.run(JIoEndpoint.java:447)
        at java.lang.Thread.run(Thread.java:619)
Caused by: javax.faces.el.EvaluationException: javax.ejb.EJBTransactionRolledbackException: org.hibernate.hql.ast.QuerySyntaxException:
unexpected token: like near line 1, column 60 [select u.username from org.jboss.seam.example.booking.User like u where u.username=:el1]
        at javax.faces.component.MethodBindingMethodExpressionAdapter.invoke(MethodBindingMethodExpressionAdapter.java:102)
        at com.sun.faces.application.ActionListenerImpl.processAction(ActionListenerImpl.java:102)
        ... 51 more
Caused by: javax.ejb.EJBTransactionRolledbackException: org.hibernate.hql.ast.QuerySyntaxException: unexpected token: like near line 1,
column 60 [select u.username from org.jboss.seam.example.booking.User like u where u.username=:el1]
        at org.jboss.ejb3.tx.Ejb3TxPolicy.handleInCallerTx(Ejb3TxPolicy.java:115)
        at org.jboss.aspects.tx.TxPolicy.invokeInCallerTx(TxPolicy.java:130)
        at org.jboss.aspects.tx.TxInterceptor$Required.invoke(TxInterceptor.java:194)
        at org.jboss.aop.joinpoint.MethodInvocation.invokeNext(MethodInvocation.java:102)
        at org.jboss.aspects.tx.TxPropagationInterceptor.invoke(TxPropagationInterceptor.java:76)
        at org.jboss.aop.joinpoint.MethodInvocation.invokeNext(MethodInvocation.java:102)
        at org.jboss.ejb3.tx.NullInterceptor.invoke(NullInterceptor.java:42)
        at org.jboss.aop.joinpoint.MethodInvocation.invokeNext(MethodInvocation.java:102)
        at org.jboss.ejb3.security.RoleBasedAuthorizationInterceptorv2.invoke(RoleBasedAuthorizationInterceptorv2.java:201)
        at org.jboss.aop.joinpoint.MethodInvocation.invokeNext(MethodInvocation.java:102)
        at org.jboss.ejb3.security.Ejb3AuthenticationInterceptorv2.invoke(Ejb3AuthenticationInterceptorv2.java:186)
        at org.jboss.aop.joinpoint.MethodInvocation.invokeNext(MethodInvocation.java:102)
        at org.jboss.ejb3.ENCPropagationInterceptor.invoke(ENCPropagationInterceptor.java:41)
        at org.jboss.aop.joinpoint.MethodInvocation.invokeNext(MethodInvocation.java:102)
        at org.jboss.ejb3.BlockContainerShutdownInterceptor.invoke(BlockContainerShutdownInterceptor.java:67)
        at org.jboss.aop.joinpoint.MethodInvocation.invokeNext(MethodInvocation.java:102)
        at org.jboss.aspects.currentinvocation.CurrentInvocationInterceptor.invoke(CurrentInvocationInterceptor.java:67)
        at org.jboss.aop.joinpoint.MethodInvocation.invokeNext(MethodInvocation.java:102)
        at org.jboss.ejb3.session.SessionSpecContainer.invoke(SessionSpecContainer.java:176)
        at org.jboss.ejb3.session.SessionSpecContainer.invoke(SessionSpecContainer.java:216)
        at org.jboss.ejb3.proxy.impl.handler.session.SessionProxyInvocationHandlerBase.invoke(Se
)
        at org.jboss.ejb3.proxy.impl.handler.session.SessionProxyInvocationHandlerBase.invoke(Se
)
        at $Proxy464.register(Unknown Source)
```

Top 25 CWE	OWASP
CWE-209: Information Exposure Through an Error Message	Category A6 - Security Misconfiguration
CWE-89: Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection')	Category A1 - Injection

SQL Injection – Can I get to the data I want?

Let's test how far I can get with this SQL Injection and test the limits…

- Couple things here...the application logic from the search box kept interpreting spaces as separate search terms.
- To pull this off we had to create a SQL query with no spaces in it…
- How'd I accomplish that…I used SQL comment characters ***/**/***

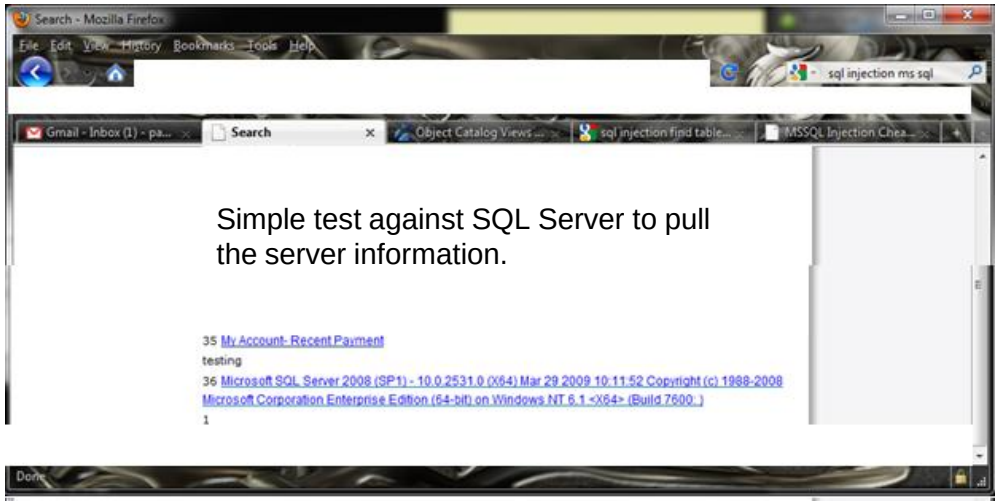
Here's the injection string…

'UNION/**/ALL/**/SELECT/**/1,36,@@version,'1','1','1',1,'1';--

Look mom…no spaces!

- In fairness, they used PreparedStatements which means…variables passed as arguments to prepared statements will automatically be escaped by the JDBC driver.
- However, all queries should be parameterized and not passed directly to the query which is why this was possible:

*PreparedStatement prepStmt = con.prepareStatement("SELECT * FROM user WHERE userId = '+strUserName+'");*



Top 25 CWE	OWASP
CWE-209: Information Exposure Through an Error Message	Category A6 - Security Misconfiguration

Access Control – How deep do I go?

Building on the last step I know a couple of things:

- The “Search” box is available without authentication yet I still have access to SQL Server system tables…
- Indication the access control model is broken
- This function should not have read/write/update/delete access to any tables outside of the table it needs and even then should only be read access
- Common mistake is to think hackers will never get that far back to even touch the database

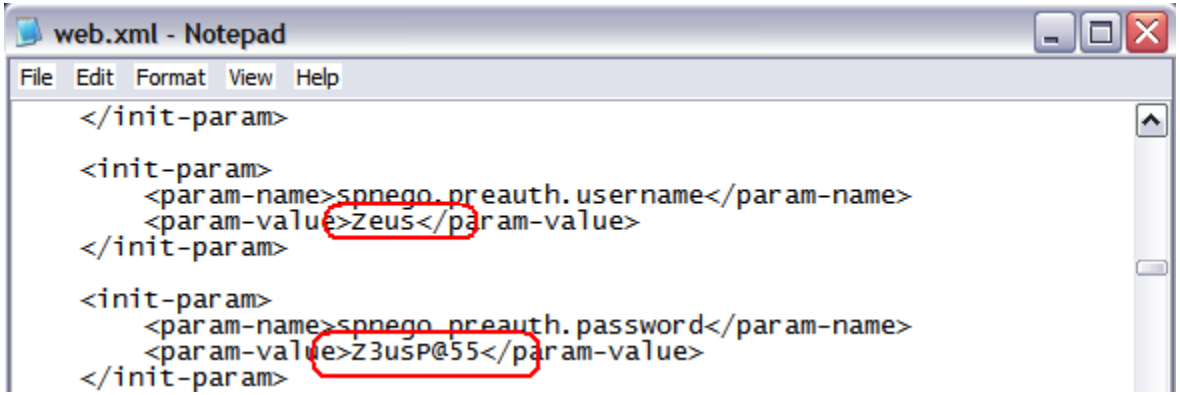
Top 25 CWE	OWASP
CWE-732: Incorrect Permission Assignment for Critical Resource	Category A6 - Security Misconfiguration or Category A4 - Insecure Direct Object References

Local File Inclusion

Top 25 CWE	OWASP
CWE-98: Improper Control of Filename for Include	Category A8 – Failure to Restrict URL Access

Clear-text Passwords – But it's only on my internal network…

Looks like we can traverse to the web.xml and pull out cleartext passwords…



Top 25 CWE	OWASP
CWE-311: Missing Encryption of Sensitive Information	Category A8 - Insecure Cryptographic Storage

Full Compromise! – There goes my data…

The injection string…**'UNION/**/ALL/**/SELECT/**/1, ssn, ssn,null,null, first_name,1,null/**/FROM/**/peoples_PII;--**

Here's Johnny!!! We used test data to simulate the SSN.

- In the end, I want to make off with the information and all I had to do was dump it to the "Search" results page and I'm home free…
- Went right through an HTTPS port without even being seen…every firewall has those ports open (although an application firewall may have caught this)

Lessons Learned

What'd we learn from this experience…

- You can't possibly predict every possible permutation of a vulnerability
- This exploitation scenario was possible due to multiple vulnerabilities tied together
- There was clear intent to implement security but somewhere in the middle…things got "Lost in Translation" between security and the development team
- Where do we go from here…
  - Better education to help developers (I used to be one) understand the potential threat scenarios and attack paths and have the "hacker mindset"
  - Application security continues to evolve with the technology but the general principles remain the same
  - We (security and IT) must continually work together and improve our processes and technologies to mitigate these risks (I promise security is not there to be a complete PITA…)

Contact Us

Paul Nguyen, Vice President, Cyber Solutions
Paul.nguyen@knowledgecg.com
Phone: (703) 467-2000 x108
Mobile: (571) 722-7083

Ryan Stinson, Lead Assessment Engineer
Ryan.stinson@knowledgecg.com
Phone: (703) 467-2000
Mobile: (703) 600-9388

Knowledge Consulting Group (KCG)
11710 Plaza America Dr., Ste 520
Reston, VA 20190

Voice: 703.467.2000
Fax: 703.547.0322
http://www.KnowledgeCG.com