Make My Day – Just Run A Web Scanner

Countering the faults of typical web scanners through bytecode injection

Toshinari Kureha, Fortify Software

Agenda

- Problems With Black Box Testing
  - Approaches To Finding Security Issues
  - 4 Problems With Black Box Testing
- Solution:WhiteBox Testing With ByteCode Injection
  - The Solution
  - Demo Of Solution
  - Building The Solution
- Q&A

Current Practice

Current Practice

How Do You Find Security Issues?

- Looking at architectural / design documents
- Looking at the source code
  - Static Analysis
- Looking at a running application
  - Dynamic Analysis

Current Practice

- Dynamic Analysis
  - Testing & Analysis Of Running Application
    - Find Input
    - Fuzz Input
    - Analyze Response
  - Commercial Web Scanners
    - Cenzic
    - SPIDynamics
    - Watchfire

Current Practice

Most People Use Web Scanners Because…

- Easy To Run
- Fast To Run
- “Someone Told Me To”

Dynamic Analysis Demo

Web Scanner Review

- Good
  - Found Real Vulnerabilities
  - Was Easy To Run
- “Did I Do A Good Job?”

Question 1: How Thorough Was My Test?

- Do You Know How Much Of Your Application Was Tested?

Question 1: How Thorough Was My Test?

- How Much Of The Application Do You Think You Tested?

Truth About Thoroughness

- We ran a “Version 7.0 Scanner” on the following:

Application	EMMA Code Coverage Tool	Web Source
HacmeBooks	**34% classes** **12% blocks** **14% lines**	**30.5%**
JCVS Web	**45% classes** **19% blocks** **22% lines**	**31.2%**
Java PetStore 2	**70% classes** **20% blocks** **23% lines**	**18%**

Web Scanner Review

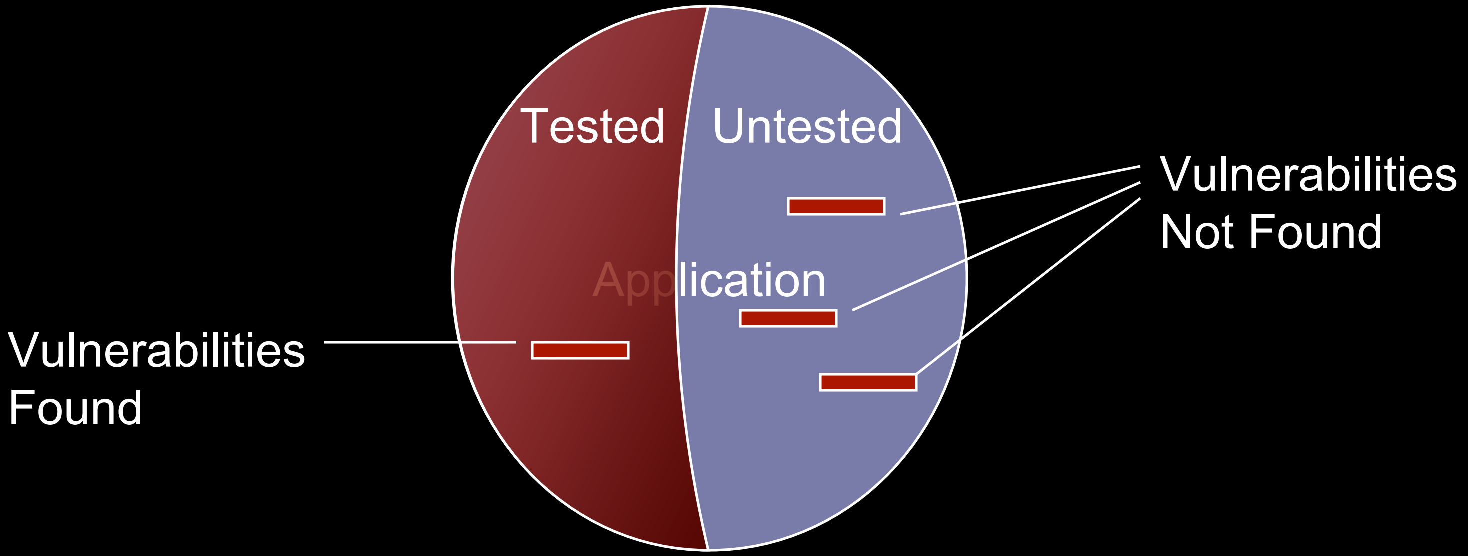
- Good
  - Found Real Vulnerabilities
  - Was Easy To Run
- Bad
  - How Thorough Was My Test?
    - No Way To Tell, And Actual Coverage Is Often Low

Question 2: Did I Find All Vulnerabilities?

- 3 Ways To Fail
  - Didn’t Test
  - Tested – But Couldn’t Conclude
  - Can’t Test

Question 2: Did I Find All Vulnerabilities?

1. Didn't Test
   - If The Web Scanner Didn't Even Reach That Area, It Cannot Test!



Question 2: Did I Find All Vulnerabilities?

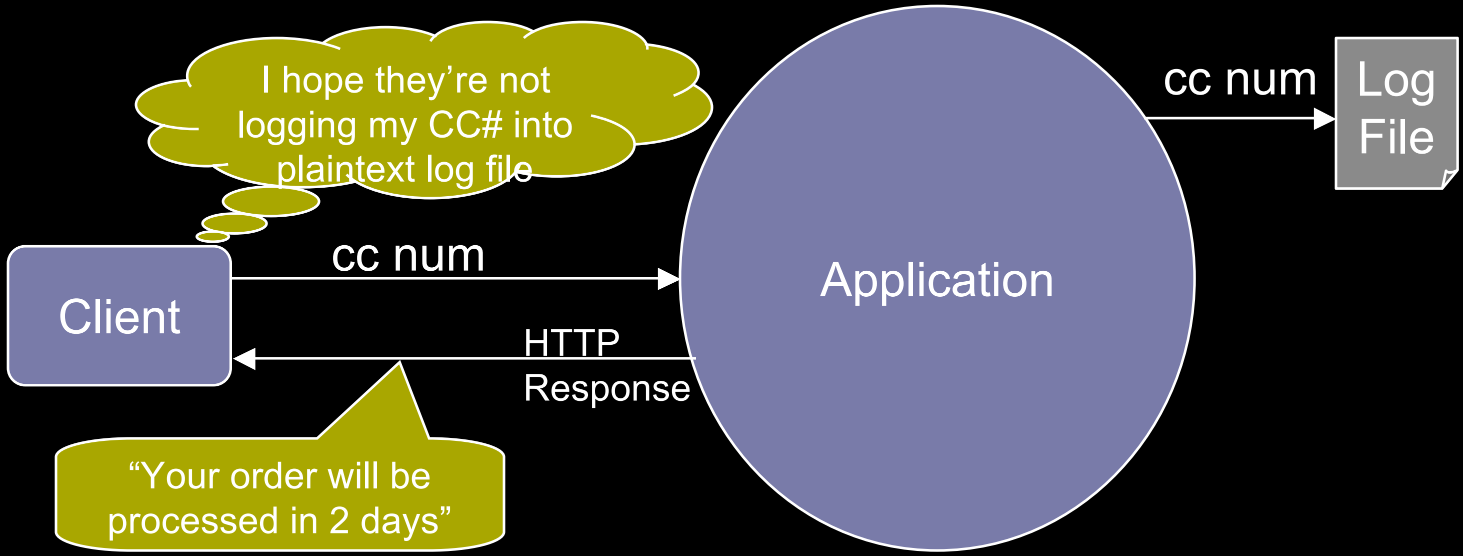
2. Tested, But Couldn't Conclude
   - Certain Classes Of Vulnerabilities <u>Sometimes</u> Can Be Detected Through HTTP Response
     - SQL Injection
     - Command Injection
     - LDAP Injection

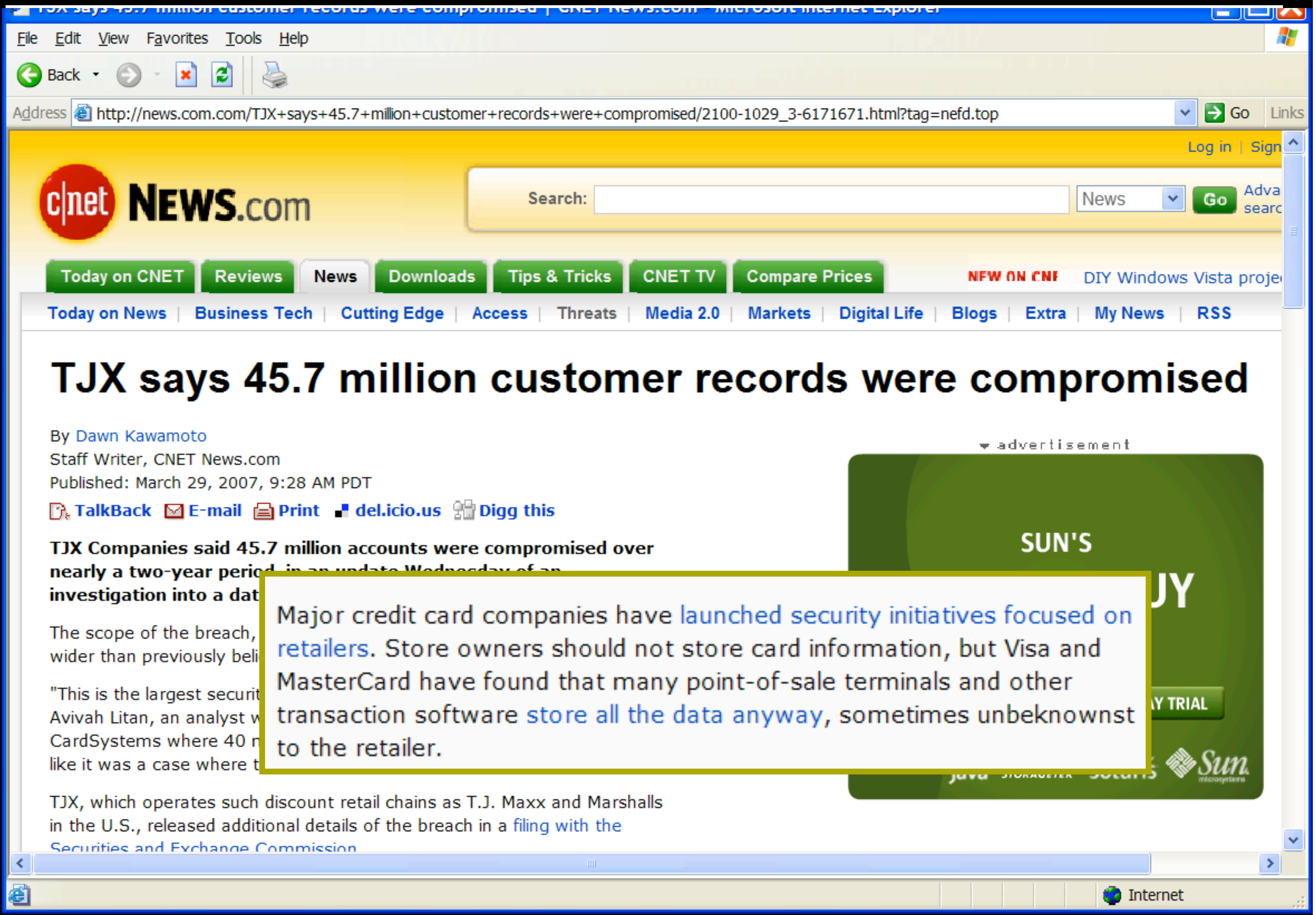
```
public void doGet(HttpServletRequest req, HttpServletResponse res)
        throws ServletException, IOException
    {
                ServletOutputStream out = res.getOutputStream();
                String user = req.getParameter("user");
                if(user != null) {
                        try {
                                String[] args = { "/bin/sh", "-c", "finger " + user };
                                Process p = Runtime.getRuntime().exec(args);
                                BufferedReader fingdata = new BufferedReader(new
InputStreamReader(p.getInputStream()));
                                String line;
                                while((line = fingdata.readLine()) != null)
                                        out.println(line);
                                p.waitFor();
                        } catch(Exception e) {
                                throw new ServletException(e);
                        }
                } else {
                        out.println("specify a user");
                }
      …
```

```
public void doGet(HttpServletRequest req, HttpServletResponse res)
        throws ServletException, IOException
    {
                ServletOutputStream out = res.getOutputStream();
                String user = req.getParameter("user");
                if(user != null) {
                        try {
                                String[] args = { "/bin/sh", "-c", “sendMail.sh " + user };
                                Process p = Runtime.getRuntime().exec(args);
                                p.waitFor();
                        } catch(Exception e) {
                                e.printStackTrace(System.err);
                        }
                        out.println(“Thank you note was sent”);
                } else {
                        out.println("specify a user");
                }
      …
```

Question 2: Did I Find All Vulnerabilities?

3. Can't Test
   - Some Vulnerabilities Have No Manifestation In Http Response


I hope they're not logging my CC# into plaintext log file
Client
cc num
HTTP Response
"Your order will be processed in 2 days"
Application
cc num
Log File


TJX says 45.7 million customer records were compromised | CNET News.com - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back
Address http://news.com.com/TJX+says+45.7+million+customer+records+were+compromised/2100-1029_3-6171671.html?tag=nefd.top
Go Links
Log in | Sign
cnet NEWS.com
Search: News Go Adva searc
Today on CNET Reviews News Downloads Tips & Tricks CNET TV Compare Prices
NEW ON CNE DIY Windows Vista proje
Today on News | Business Tech | Cutting Edge | Access | Threats | Media 2.0 | Markets | Digital Life | Blogs | Extra | My News | RSS
advertisement
SUN'S
Y TRIAL
Sun
Internet

TJX says 45.7 million customer records were compromised

By Dawn Kawamoto
Staff Writer, CNET News.com
Published: March 29, 2007, 9:28 AM PDT

TalkBack E-mail Print del.icio.us Digg this

TJX Companies said 45.7 million accounts were compromised over nearly a two-year perio
investigation into a dat

The scope of the breach,
wider than previously beli

"This is the largest securit
Avivah Litan, an analyst w
CardSystems where 40 n
like it was a case where t

TJX, which operates such discount retail chains as T.J. Maxx and Marshalls in the U.S., released additional details of the breach in a filing with the Securities and Exchange Commission

Major credit card companies have launched security initiatives focused on retailers. Store owners should not store card information, but Visa and MasterCard have found that many point-of-sale terminals and other transaction software store all the data anyway, sometimes unbeknownst to the retailer.

Web Scanner Review

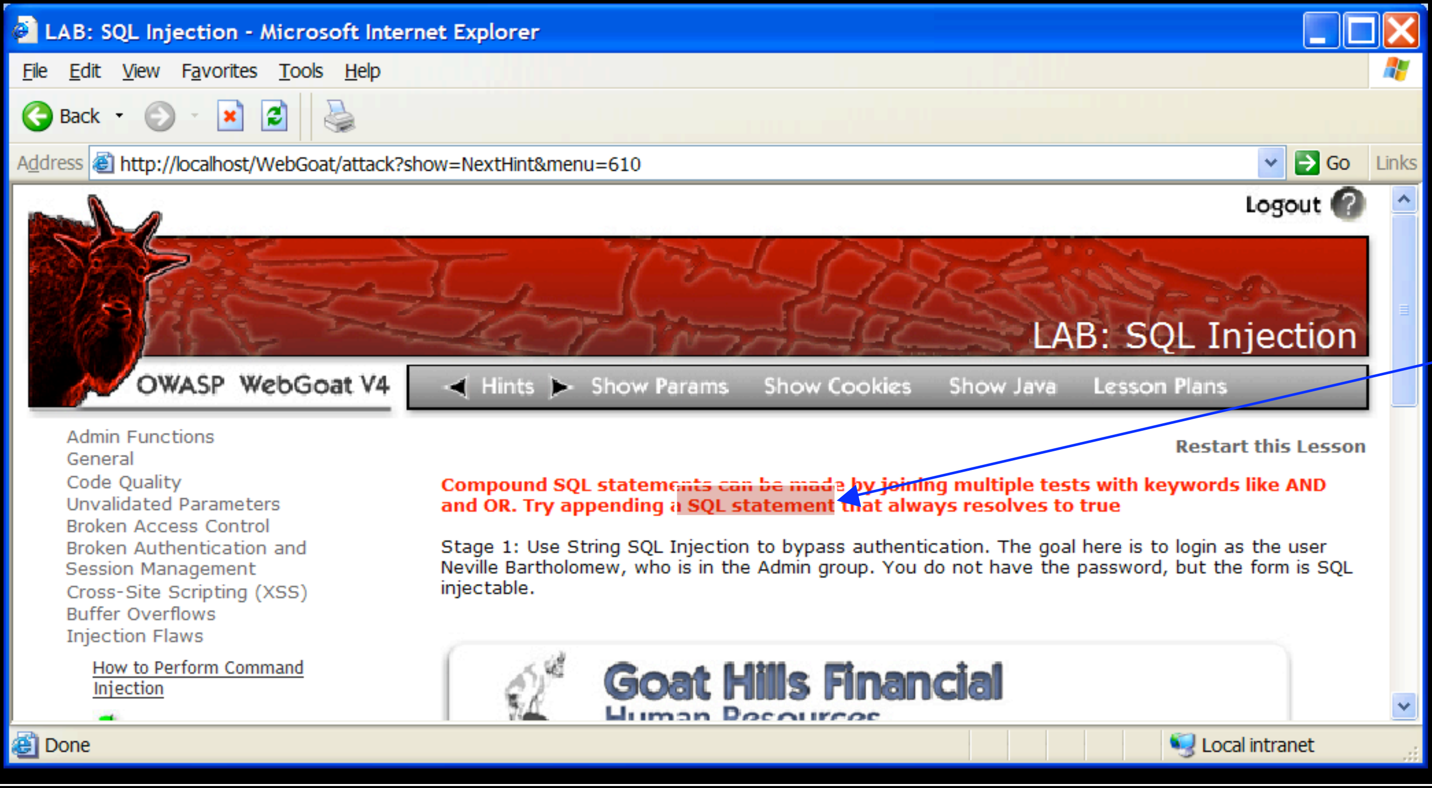
- Good
  - Found Real Vulnerabilities
  - Was Easy To Run
- Bad
  - How Thorough Was My Test?
    - No Way To Tell, And Actual Coverage Is Often Low
  - Did I Find All My Vulnerabilities?
    - Didn’t Test, Tested But Couldn’t Conclude, Can’t Test

Question 3: Are All The Results Reported True?

- No Method Is Perfect
- Under What Circumstances Do Web Scanners Report False Positives?
  - Matching Signature On A Valid Page
  - Matching Behavior On A Valid Page

Question 3: Are All The Results Reported True?

- Matching Signature On A Valid Page



LAB: SQL Injection - Microsoft Internet Explorer

File Edit View Favorites Tools Help

Address http://localhost/WebGoat/attack?show=NextHint&menu=610

Logout

LAB: SQL Injection

OWASP WebGoat V4

Hints | Show Params | Show Cookies | Show Java | Lesson Plans

Admin Functions
General
Code Quality
Unvalidated Parameters
Broken Access Control
Broken Authentication and Session Management
Cross-Site Scripting (XSS)
Buffer Overflows
Injection Flaws

How to Perform Command Injection

Restart this Lesson

Compound SQL statements can be made by joining multiple tests with keywords like AND and OR. Try appending a SQL statement that always resolves to true

Stage 1: Use String SQL Injection to bypass authentication. The goal here is to login as the user Neville Bartholomew, who is in the Admin group. You do not have the password, but the form is SQL injectable.

Goat Hills Financial
Human Resources

Done

Local intranet

Question 3: Are All The Results Reported True?

- Matching Behavior On A Valid Page
  - "To determine if the application is vulnerable to SQL injection, try <u>injecting an extra true condition</u> into the WHERE clause… and if this query also <u>returns the same</u> …, then the application is susceptible to <u>SQL injection</u>" (from paper on Blind SQL Injection)
- E.g.
  - http://www.server.com/getCC.jsp?id=5
    - select ccnum from table where id='5'
  - http://www.server.com/getCC.jsp?id=5' AND '1'='1
    - select ccnum from table where id='5' AND '1'='1'

Question 3: Are All The Results Reported True?

- E.g.
  - http://www.server.com/getCC.jsp?id=5
    - select ccnum from table where id='5'
    - Response:
      - "No match found" (No one with id "5")
  - http://www.server.com/getCC.jsp?id=5' AND '1'='1
    - select ccnum from table where id='5\' AND \'1\'=\'1'
    - Response
      - "No match found" (No one with id "5' AND '1'='1")
        - All single quotes were escaped.
- According To The Algorithm ("inject a true clause and look for same response"), This Is SQL Injection Vulnerability!

Web Scanner Review

- Good
  - Found Real Vulnerabilities
  - Was Easy To Run
- Bad
  - How Thorough Was My Test?
    - No Way To Tell, And Actual Coverage Is Often Low
  - Did I Find All My Vulnerabilities?
    - Didn’t Test, Tested But Couldn’t Conclude, Can’t Test
  - Are All The Results Reported True?
    - Susceptible To False Signature & Behavior Matching

Question 4: How Do I Fix The Problem?

- Security Issues Must Be Fixed In Source Code
- Information Given
  - URL
  - Parameter
  - General Vulnerability Description
  - HTTP Request/Response
- But Where In My Source Code Should I Look At?

Question 4: How Do I Fix The Problem?

- Incomplete Vulnerability Report -> Bad Fixes
- Report:
  - Injecting “AAAAA…..AAAAA” Caused Application To Crash
- Solution By Developers:

```
….
if (input.equals(“AAAAA…..AAAAA”))
    return;
…..
```

Web Scanner Review

- Good
  - Found Real Vulnerabilities
  - Was Easy To Run

- Bad
  - How Thorough Was My Test?
    - No Way To Tell, And Actual Coverage Is Often Low
  - Did I Find All My Vulnerabilities?
    - Didn’t Test, Tested But Couldn’t Conclude, Can’t Test
  - Are All The Results Reported True?
    - Susceptible To Signature & Behavior Matching
  - How Do I Fix The Problem?
    - No Source Code / Root Cause Information

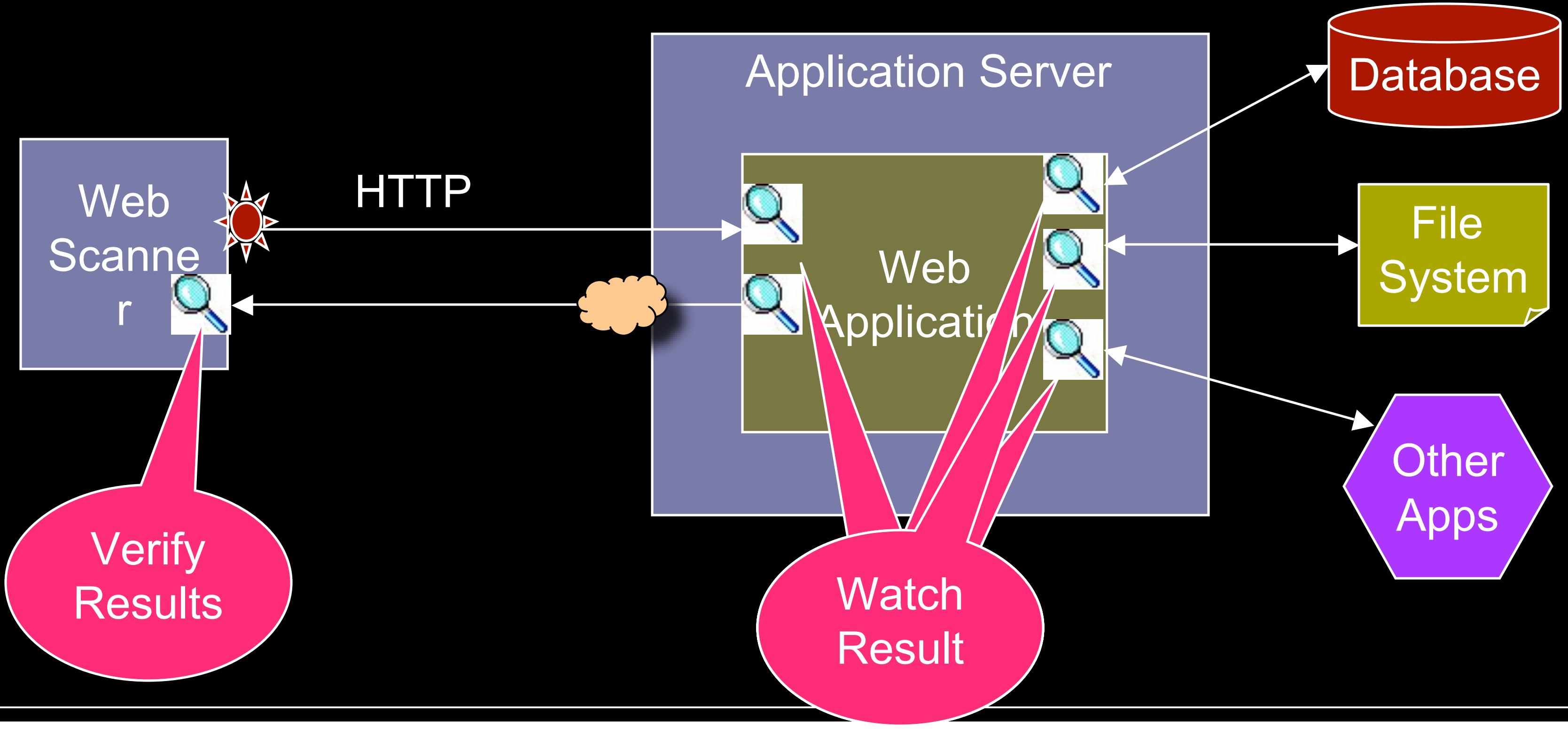
Attacking The Problems

White Box Testing With
Bytecode Injection

Agenda

- Problems With Black Box Testing
  - Approaches To Finding Security Issues
  - 4 Problems With Black Box Testing
- Solution:WhiteBox Testing With ByteCode Injection
  - The Solution
  - Demo Of Solution
  - Building The Solution
- Q&A

Review… and Proposal



How Will Monitors Solve The Problems?

- How Thorough Was My Test?
- Did I Find All My Vulnerabilities?
- Are All The Results Reported True?
- How Do I Fix The Problem?

- Monitors Inside Will Tell Which Parts Was Hit
- Monitors Inside Detects More Vulnerabilities
- Very Low False Positive By Looking At Source Of Vulnerabilities
- Monitors Inside Can Give Root Cause Information

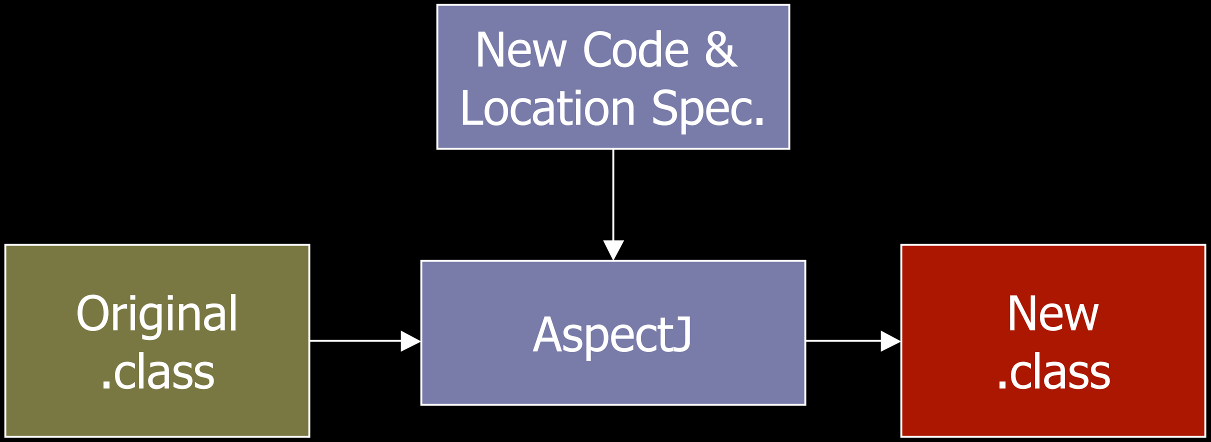
How To Build The Solution

- How Do You Inject The Monitors Inside The Application?
- Where Do You Inject The Monitors Inside The Application?
- What Should The Monitors Do Inside The Application?

How Do You Inject The Monitors?

- Problem: How Do You Put The Monitors Into The Application?
- Assumption: You Do Not Have Source Code, Only Deployed Java / .NET Application
- Solution: Bytecode Weaving
  - AspectJ for Java
  - AspectDNG for .NET

How Does Bytecode Weaving Work?



Similar process for .NET

How Does Bytecode Weaving Work?

```
List getStuff(String id) {
  List list = new ArrayList();
  try {
    String sql = “select stuff from
    mytable where id=‘” + id + “’”;
    JDBCstmt.executeQuery(sql);
  } catch (Exception ex) {
    log.log(ex);
  }
  return list;
}
```

Before “executeQuery()” Call “MyLibrary.doCheck()”

```
List getStuff(String id) {
  List list = new ArrayList();
  try {
    String sql = “select stuff from
    mytable where id=‘” + id + “’”;
    MyLibrary.doCheck(sql);
    JDBCstmt.executeQuery(sql);
  } catch (Exception ex) {
    log.log(ex);
  }
  return list;
}
```

Bytecode Injection Demo

Applying Byte-Code Injection To Enhance Security Testing

- How Do You Inject The Monitors Inside The Application?
- Where Do You Inject The Monitors Inside The Application?
- What Should The Monitors Do Inside The Application?

Where Do You Inject The Monitors?

- All Web Inputs (My Web Scan Should Hit All Of Them)
  - request.getParameter, form.getBean
- All Inputs (Not All Inputs Are Web)
  - socket.getInputStream.read
- All “Sinks” (All Security Critical Functions)
  - Statement.executeQuery(String)
  - (FileOutputStream|FileWriter).write(byte[])
  - …

Applying Byte-Code Injection To Enhance Security Testing

- How Do You Inject The Monitors Inside The Application?
- Where Do You Inject The Monitors Inside The Application?
- What Should The Monitors Do Inside The Application?

What Should The Monitors Do?

- Report Whether The Monitor Was Hit
- Analyze The Content Of the Call For Security Issues
- Report Code-Level Information About Where The Monitor Got Triggered

What Should The Monitors Do?

```
aspect SQLInjection {
        pointcut sqlExec(String sql):call(ResultSet Statement.executeQuery(String))
                && args(sql);
        before(String sql) : sqlExec(sql) { ch          ection(sql, thisJoinPoint); }
        void checkInjection(String sql, J        oint thisJoinPoint){
                System.out.println("HIT:" +
                                thisJoinPoint.getSourceLocation().getFileName() +
                thisJoinPoint.getSourceLocation().getLine());
                if (count(sql, '\'')%2 == 1) {
                        Syste   out.println("*** SQ
being executed as fo          " + sql);
                }
```

1) Report whether API was hit or not

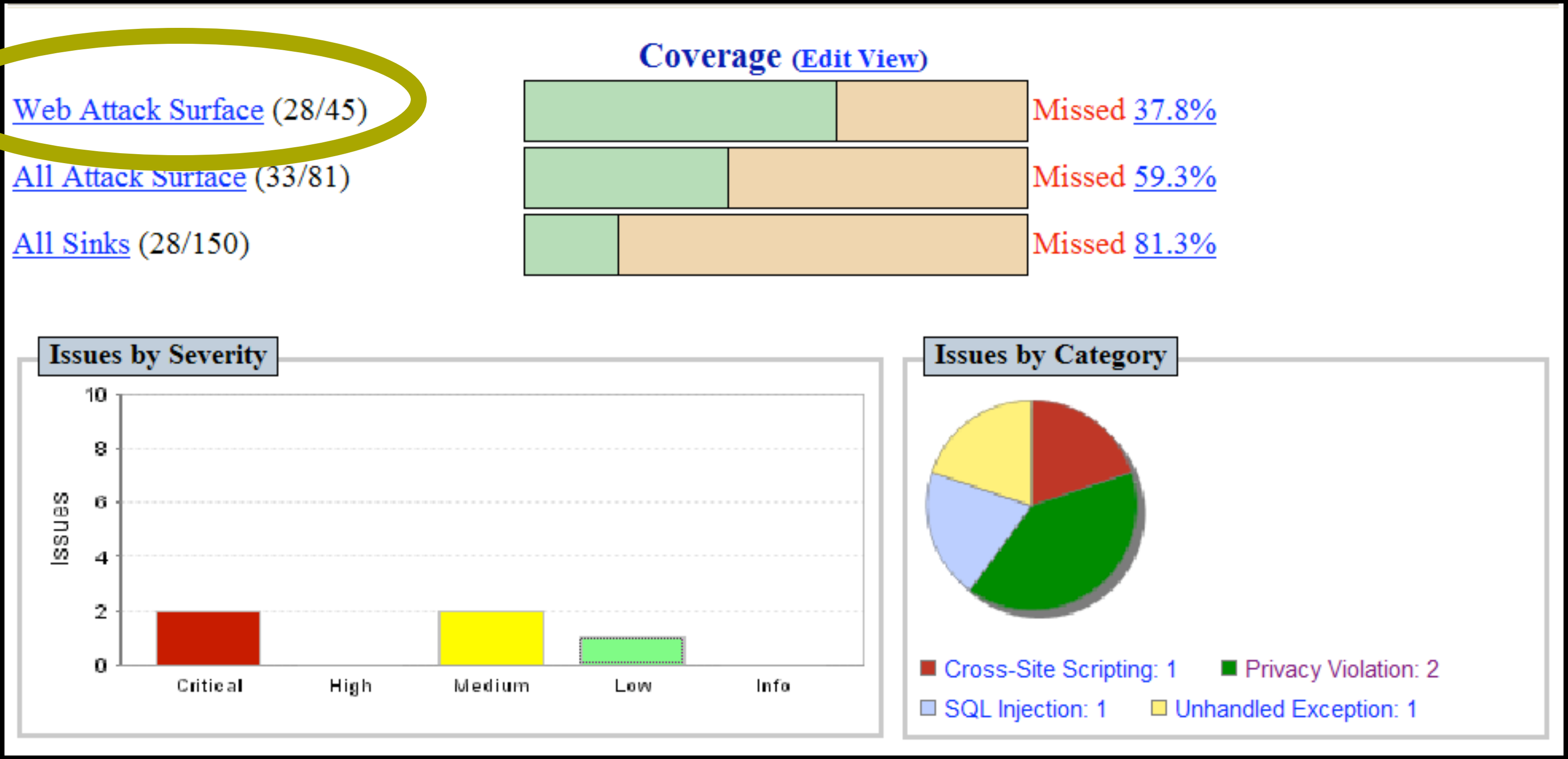
2) Analyze The Content Of The API Call

3) Report Code-Level Information

Proof Of Concept

- Running The Custom Solution

With Additional Work on UI


Coverage (Edit View)

Web Attack Surface (28/45)		Missed 37.8%
All Attack Surface (33/81)		Missed 59.3%
All Sinks (28/150)		Missed 81.3%

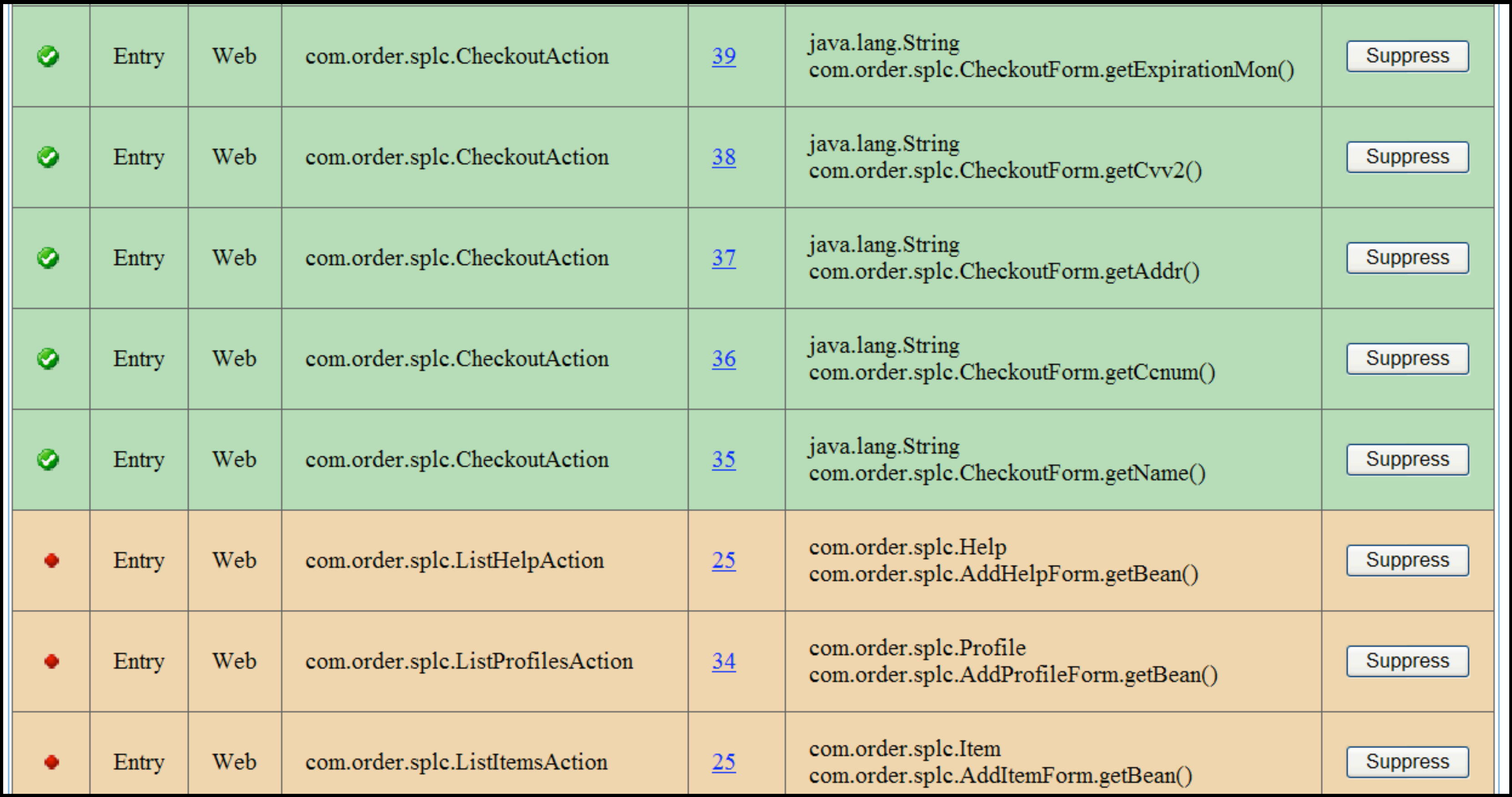
Issues by Severity

Critical	High	Medium	Low	Info
2	0	2	1	0

Issues by Category

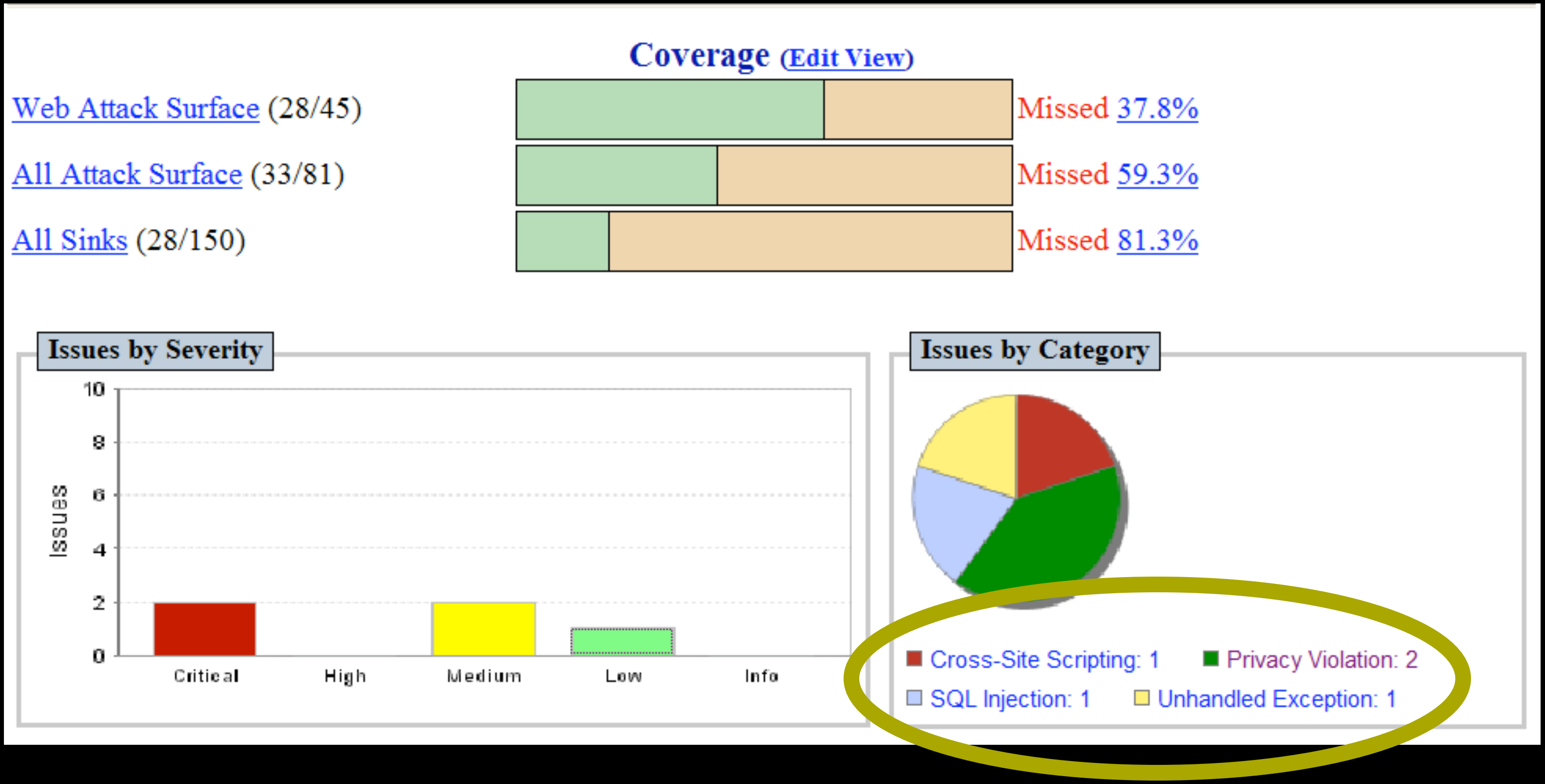
🟥 Cross-Site Scripting: 1
🟩 Privacy Violation: 2
🟦 SQL Injection: 1
🟨 Unhandled Exception: 1

Coverage



✓	Entry	Web	com.order.splc.CheckoutAction	39	java.lang.String com.order.splc.CheckoutForm.getExpirationMon()	Suppress
✓	Entry	Web	com.order.splc.CheckoutAction	38	java.lang.String com.order.splc.CheckoutForm.getCvv2()	Suppress
✓	Entry	Web	com.order.splc.CheckoutAction	37	java.lang.String com.order.splc.CheckoutForm.getAddr()	Suppress
✓	Entry	Web	com.order.splc.CheckoutAction	36	java.lang.String com.order.splc.CheckoutForm.getCcnum()	Suppress
✓	Entry	Web	com.order.splc.CheckoutAction	35	java.lang.String com.order.splc.CheckoutForm.getName()	Suppress
●	Entry	Web	com.order.splc.ListHelpAction	25	com.order.splc.Help com.order.splc.AddHelpForm.getBean()	Suppress
●	Entry	Web	com.order.splc.ListProfilesAction	34	com.order.splc.Profile com.order.splc.AddProfileForm.getBean()	Suppress
●	Entry	Web	com.order.splc.ListItemsAction	25	com.order.splc.Item com.order.splc.AddItemForm.getBean()	Suppress

With Additional Work on UI



Coverage (Edit View)

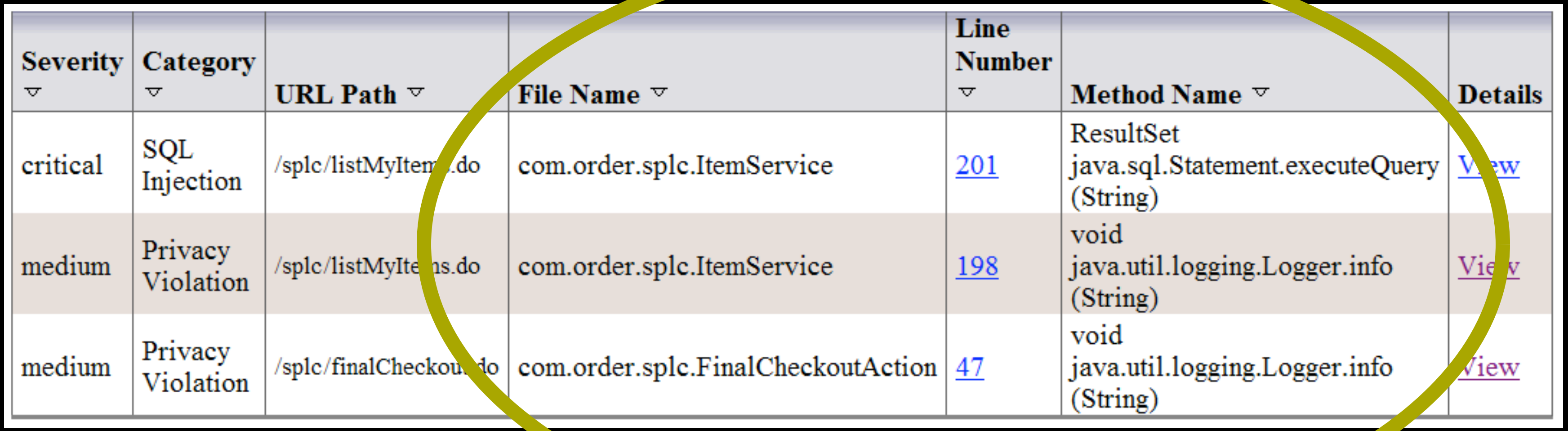
Web Attack Surface (28/45)	Missed 37.8%
All Attack Surface (33/81)	Missed 59.3%
All Sinks (28/150)	Missed 81.3%

Issues by Severity

Issues by Category

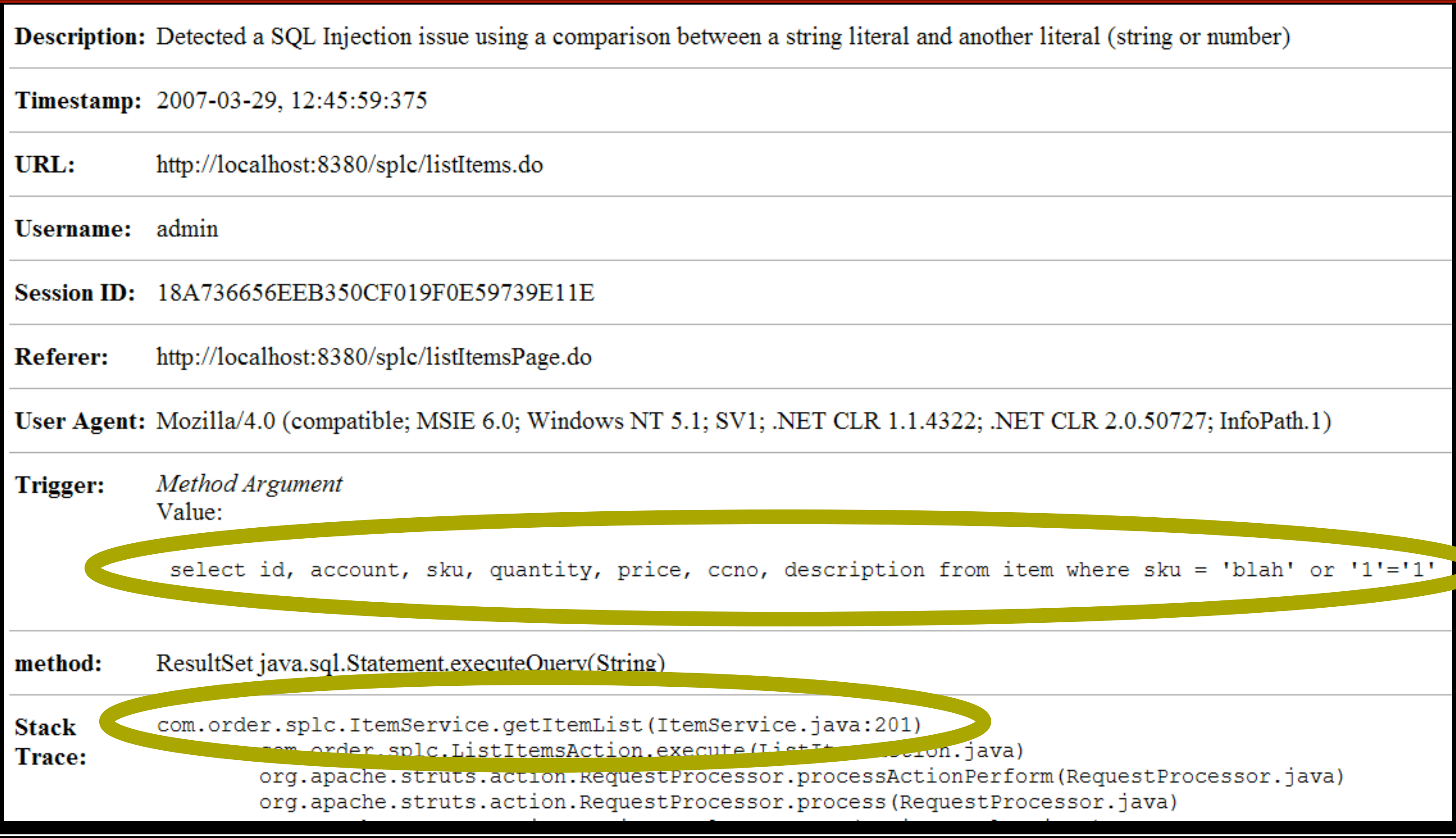
🟥 Cross-Site Scripting: 1
🟩 Privacy Violation: 2
🟦 SQL Injection: 1
🟨 Unhandled Exception: 1

Security Issues Detail



Severity ▽	Category ▽	URL Path ▽	File Name ▽	Line Number ▽	Method Name ▽	Details
critical	SQL Injection	/splc/listMyItems.do	com.order.splc.ItemService	201	ResultSet java.sql.Statement.executeQuery (String)	View
medium	Privacy Violation	/splc/listMyItems.do	com.order.splc.ItemService	198	void java.util.logging.Logger.info (String)	View
medium	Privacy Violation	/splc/finalCheckout.do	com.order.splc.FinalCheckoutAction	47	void java.util.logging.Logger.info (String)	View

Security Issues Detail – SQL Injection



Description: Detected a SQL Injection issue using a comparison between a string literal and another literal (string or number)

Timestamp: 2007-03-29, 12:45:59:375

URL: http://localhost:8380/splc/listItems.do

Username: admin

Session ID: 18A736656EEB350CF019F0E59739E11E

Referer: http://localhost:8380/splc/listItemsPage.do

User Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1; .NET CLR 1.1.4322; .NET CLR 2.0.50727; InfoPath.1)

Trigger: *Method Argument*
Value:

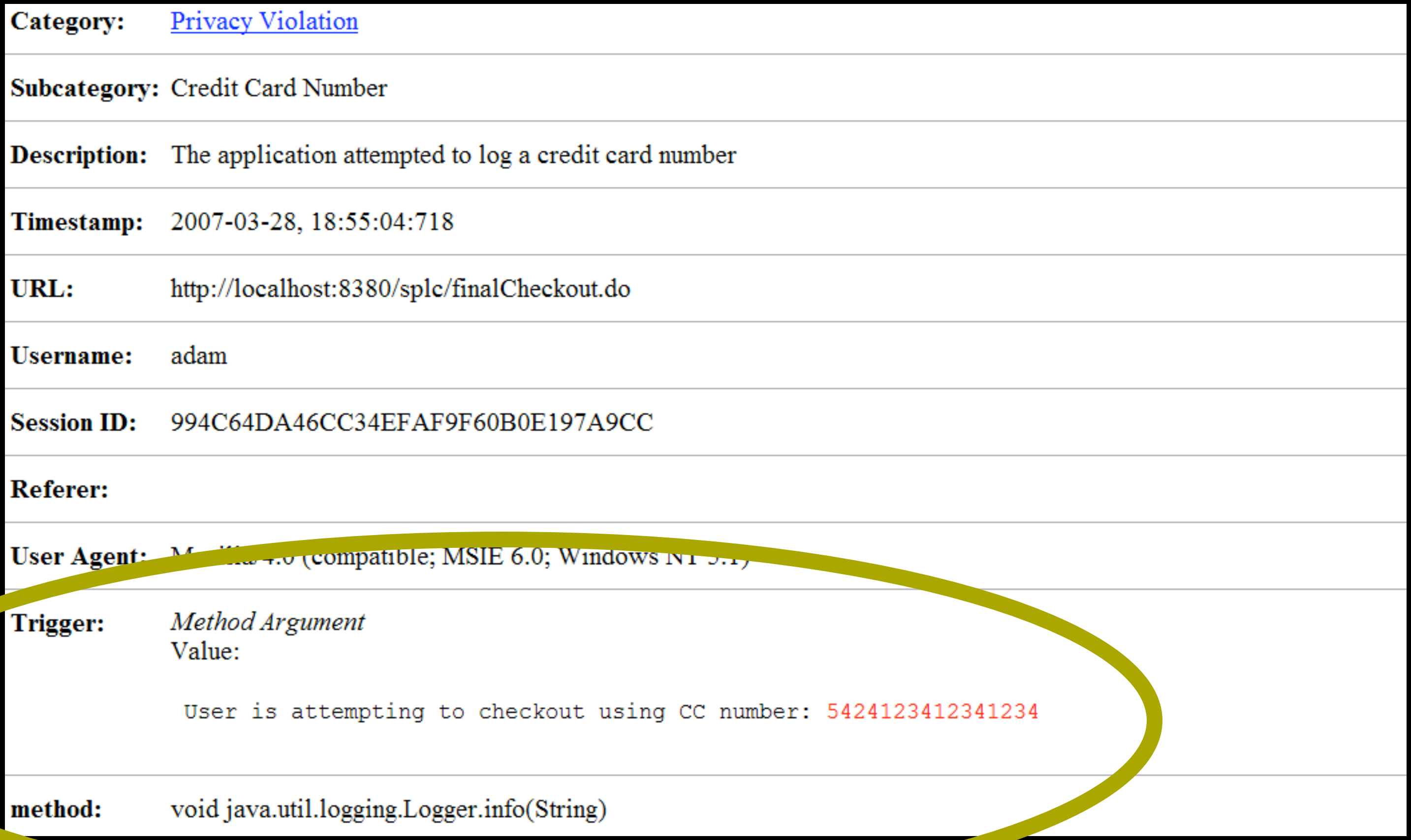
```
select id, account, sku, quantity, price, ccno, description from item where sku = 'blah' or '1'='1'
```

method: ResultSet java.sql.Statement.executeQuery(String)

Stack Trace:

```
com.order.splc.ItemService.getItemList(ItemService.java:201)
        com.order.splc.ListItemsAction.execute(ListItemsAction.java)
        org.apache.struts.action.RequestProcessor.processActionPerform(RequestProcessor.java)
        org.apache.struts.action.RequestProcessor.process(RequestProcessor.java)
```

Security Issue Detail – Privacy Violation



Category:	Privacy Violation
Subcategory:	Credit Card Number
Description:	The application attempted to log a credit card number
Timestamp:	2007-03-28, 18:55:04:718
URL:	http://localhost:8380/splc/finalCheckout.do
Username:	adam
Session ID:	994C64DA46CC34EFAF9F60B0E197A9CC
Referer:	
User Agent:	Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)
Trigger:	*Method Argument* Value: `User is attempting to checkout using CC number: 5424123412341234`
method:	void java.util.logging.Logger.info(String)

Conclusions – Web Scanners

- Good
  - Easy To Use
  - Finding Smoking Gun
- Bad
  - Lack Of Coverage Information
  - False Negative
  - False Positive
  - Lack Of Code-Level / Root Cause Information

Conclusions – White Box Testing

- Bytecode Injection Require Access To Running Application
- In Exchange …
  - Gain Coverage Information
  - Find More Vulnerabilities, More Accurately
  - Determine Root Cause Information

Conclusions – Use Your Advantage

	Attacker	Defender
Time	✦	
Attempts	✦	
Security Knowledge	✦	
Access To Application		✦

Thank You

- Questions?
  - Email: tkureha at fortifysoftware.com