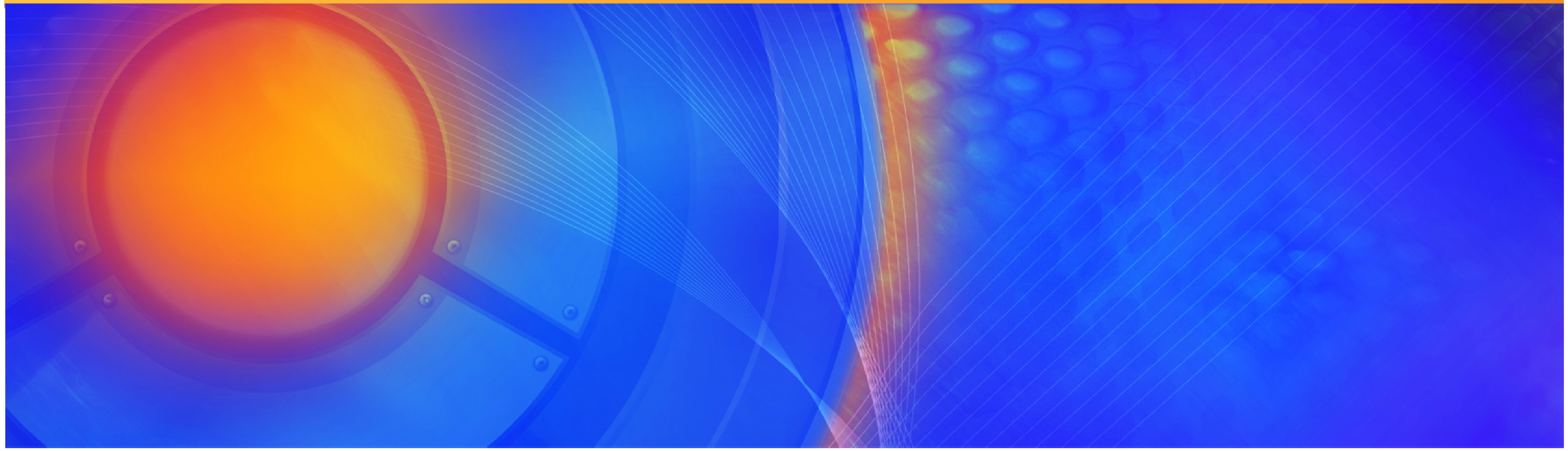
FORTIFY®



Hybrid 2.0 – In search of the holy grail…

A Talk for OWASP BeNeLux

by

Roger Thornton

Founder/CTO Fortify Software Inc

Before we Begin:

- Expectations
- Objectives
- Agenda

About Your Presenter

- 22 years of Engineering ("building stuff") in the Silicon Valley
  - Semiconductors
  - Operating Systems
  - Development Tools
  - Brokerage / E-Commerce
- The Last 6 years working on Securing that Stuff
  - Founder & CTO of Fortify Software

A Simple, Reasonable, Question….

If I run software, am I putting my business, data, customers or even life on earth at risk?

If so, how serious is the threat?

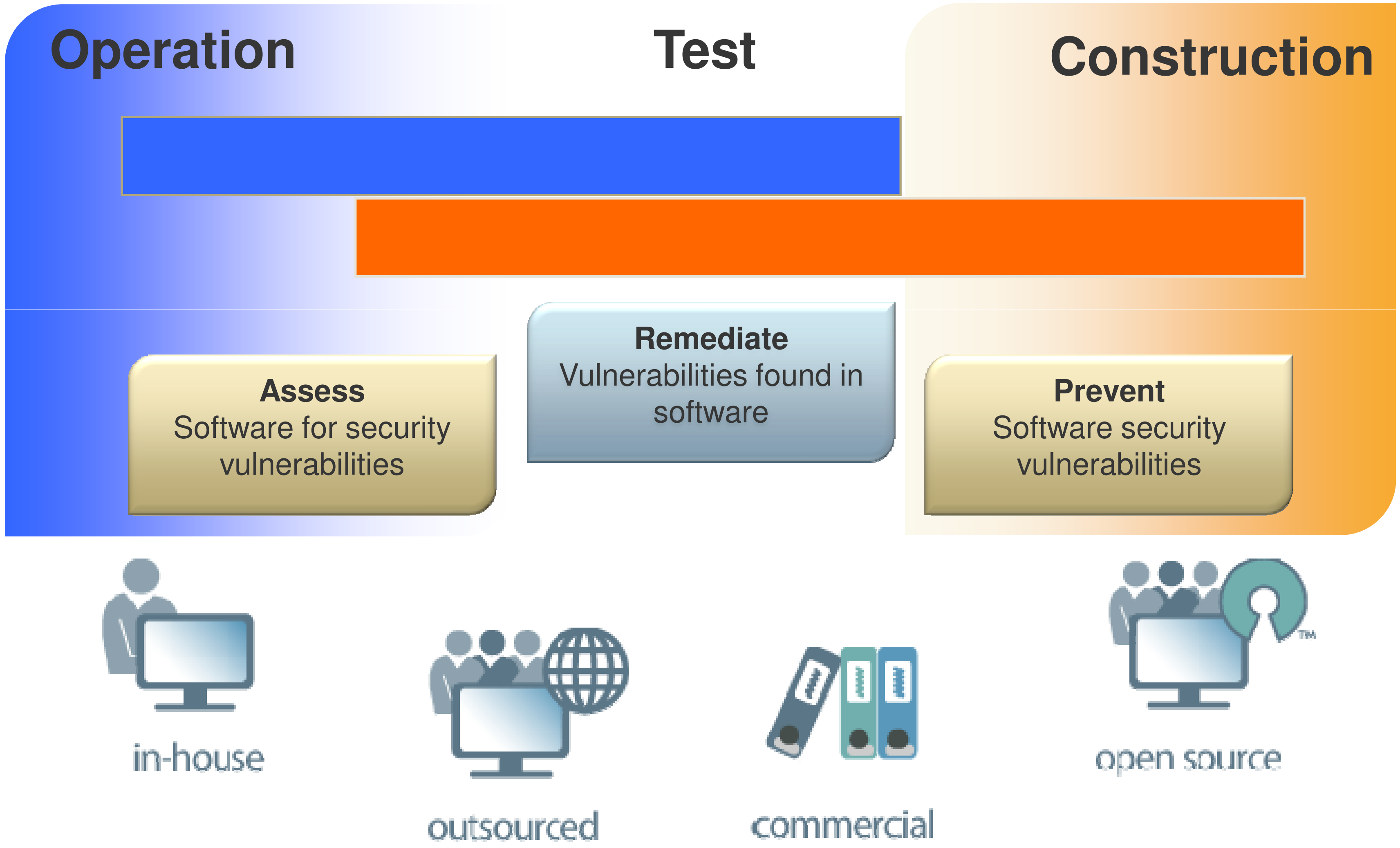
Unfortunately not so simple to answer…

Three Basic Approaches

Hire an expert Ethical Hacking	Automate Hacking Black Box / Penetration Test	Analyze the Software Static and Dynamic Analysis
Exactly what the bad guy does..	Cheap and easy way to find the most obvious issues	Look for root cause issues from the "inside out" – the code
• Hard to know if your "experts" are as good as the bad guy • Prohibitively expensive to do on a regular basis • No advantage over the bad guys	• "Badness-ometer" limitations and issues • Automated crawler and web traffic analysis can yield	• Requires intimate access to the software • Requires programming knowledge and expertise • Exploitability information is not present as with other two.
• Identifies the result – not the root cause	• Identifies the result – not the root cause	• Identifies the root cause not the result

Software Security Assurance (SSA)

The management & prevention of security risks in software



Operation
Test
Construction
Assess Software for security vulnerabilities
Remediate Vulnerabilities found in software
Prevent Software security vulnerabilities
in-house
outsourced
commercial
open source

Software Security Maturity

Risk Awareness	**Risk Reduction**	**Prevention**
Vulnerability Assessment	Analysis & Remediation	Secure SDL & Software
Proving the problem or meeting a basic regulatory requirement	Fixing security issues uncovered from assessments	Secure the development and procurement lifecycle avoiding issues altogether
• An info-sec project • Generates awareness & support security initiatives • Consulting, PenTesting & some manual code review	• Info-sec driven project with development support • Forces a rework of code • "Inside-out" Static and Dynamic Analysis required	• Info-sec-sponsored Development-led project • Requires significant organizational buy-in • Requires more than a point solution
Recurring cost that does not "fix" anything	*Lowering risk but costs too high*	*Minimizing business risk systematically*

The Challenge

Immediate Problem

Existing Legacy Applications

immediate

Assessment & remediation of existing software

compliance

Compliance & Regulatory Requirements

Systemic Problem

Software Procurement & Development Cycle

Define
Design
Code
Test
Monitor

Prevention of the introduction of new risk

in-house

outsourced

commercial

open source

Benefits of a “Hybrid” Approach

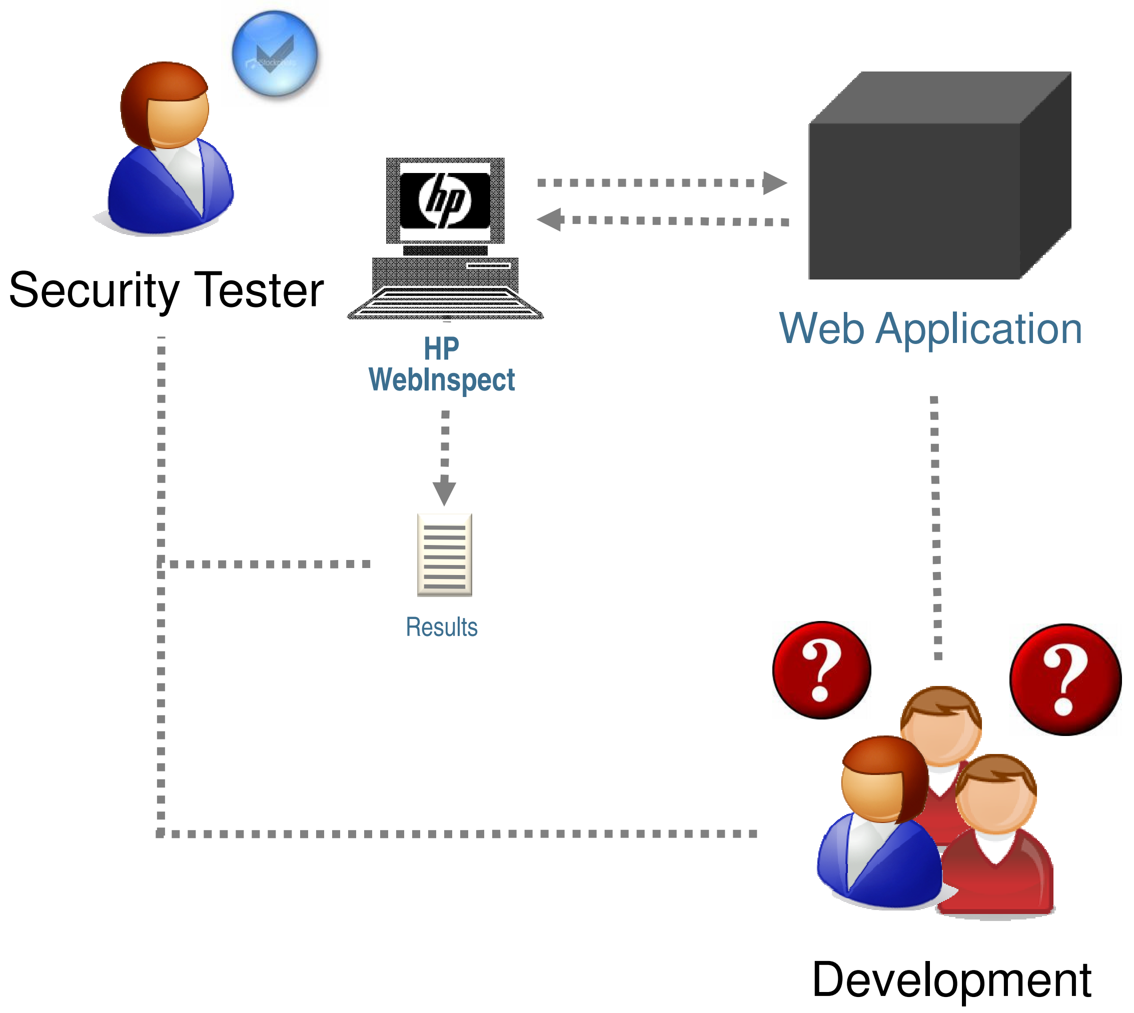
- A seamless flow from Assessment to Prevention
  - *Facilitates growth in maturity from assessment to prevention*
- Combined benefits at Testing phase - “Remediation Gap”
  - Application Testing & Software Analysis:
    - Rapid identification of high priority issues (DAST)
    - Precise description of root cause vulnerability in code (SAST)
- **Reduced time and costs to remediate vulnerabilities**
  - ✓ By mapping each security issue to root cause in source code
    - **Developers understand security findings – faster fixes**
    - **Security findings are more accurate – less research**
    - **Security findings are more comprehensive – less rework**

Reduced time to fix

Reduced false positives

Less conflict between security and development

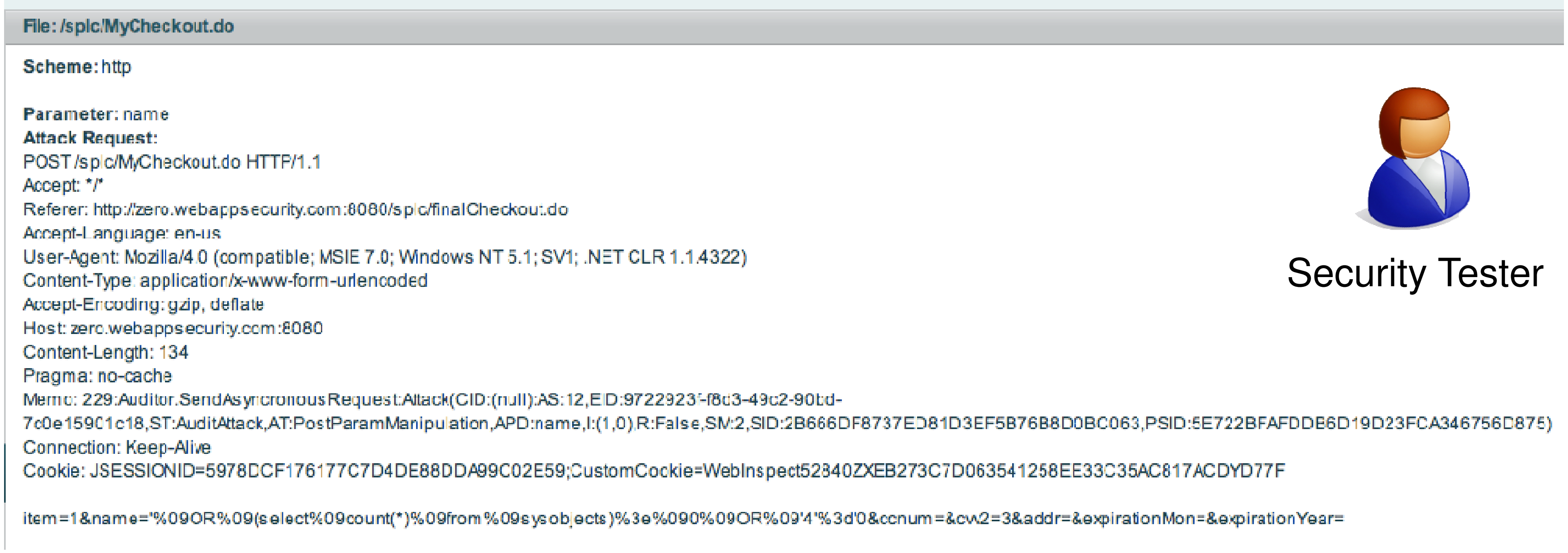
Dynamic Application Security Testing "Black Box"



Challenges

-Visibility to "root cause"…
- It is called "Black Box"
- 1 Issue may be indicative of many
- Multiple issues may trace back to one problem

-Communicating to developers
- URLs and hacking technique vs. code errors
- Validating behavior (FP)



File: /splc/MyCheckout.do

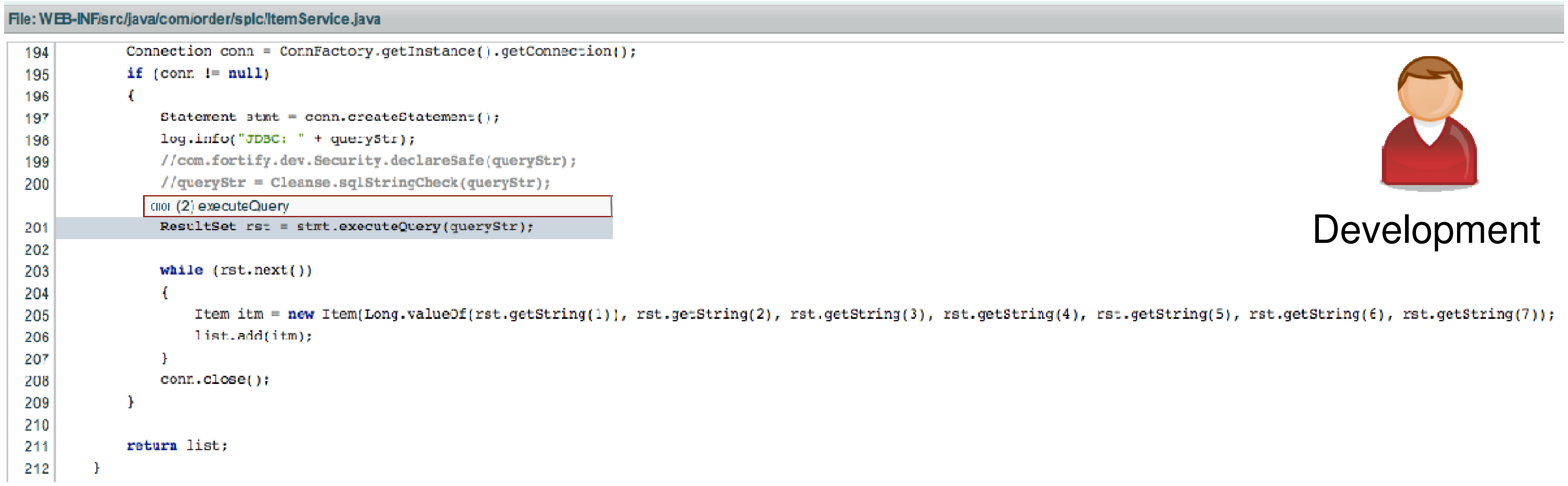
Scheme: http

Parameter: name
Attack Request:

```
POST /splc/MyCheckout.do HTTP/1.1
Accept: */*
Referer: http://zero.webappsecurity.com:8080/splc/finalCheckout.do
Accept-Language: en-us
User-Agent: Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 5.1; SV1; .NET CLR 1.1.4322)
Content-Type: application/x-www-form-urlencoded
Accept-Encoding: gzip, deflate
Host: zero.webappsecurity.com:8080
Content-Length: 134
Pragma: no-cache
Memo: 229:Auditor.SendAsyncronousRequest:Attack(CID:(null):AS:12,EID:97229231-f8c3-49c2-90bd-
7c0e15901c18,ST:AuditAttack,AT:PostParamManipulation,APD:name,I:(1,0),R:False,SM:2,SID:2B666DF8737ED81D3EF5B76B8D0BC063,PSID:5E722BFAFDDE6D19D23FCA346756D875)
Connection: Keep-Alive
Cookie: JSESSIONID=5978DCF176177C7D4DE88DDA99C02E59;CustomCookie=WebInspect52840ZXEB273C7D063541258EE33C35AC817ACDYD77F

item=1&name='%09OR%09(select%09count(*)%09from%09sysobjects)%3e%090%09OR%09'4'%3d'0&ccnum=&cvv2=3&addr=&expirationMon=&expirationYear=
```

Security Tester



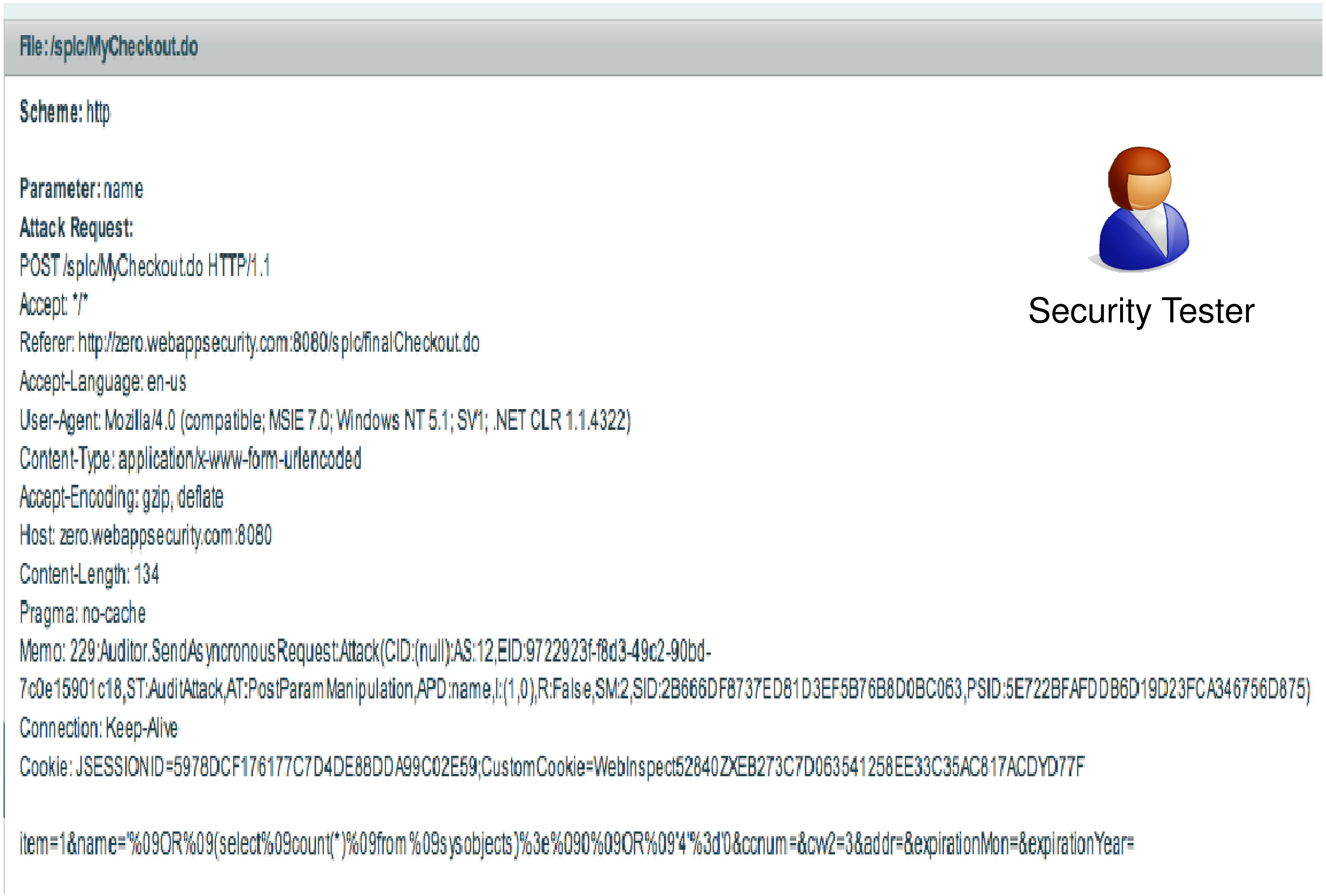
File: WEB-INF/src/java/com/order/splc/ItemService.java

```
        Connection conn = ConnFactory.getInstance().getConnection();
        if (conn != null)
        {
            Statement stmt = conn.createStatement();
            log.info("JDBC: " + queryStr);
            //com.fortify.dev.Security.declareSafe(queryStr);
            //queryStr = Cleanse.sqlStringCheck(queryStr);
            (2) executeQuery
            ResultSet rst = stmt.executeQuery(queryStr);

            while (rst.next())
            {
                Item itm = new Item(Long.valueOf(rst.getString(1)), rst.getString(2), rst.getString(3), rst.getString(4), rst.getString(5), rst.getString(6), rst.getString(7));
                list.add(itm);
            }
            conn.close();
        }

        return list;
    }
```

Development



File: /splc/MyCheckout.do

Scheme: http

Parameter: name

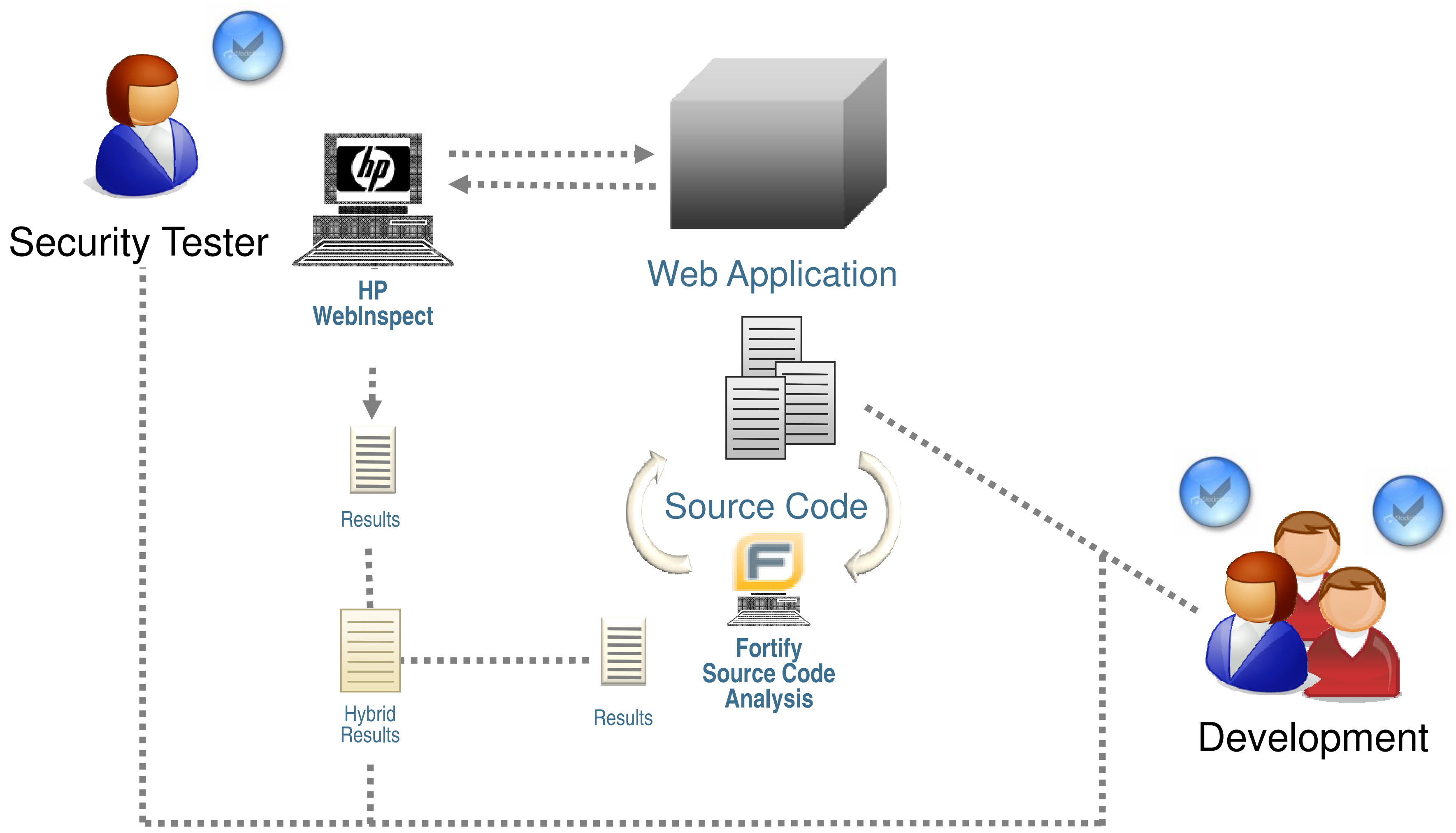
Attack Request:

```
POST /splc/MyCheckout.do HTTP/1.1
Accept: */*
Referer: http://zero.webappsecurity.com:8080/splc/finalCheckout.do
Accept-Language: en-us
User-Agent: Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 5.1; SV1; .NET CLR 1.1.4322)
Content-Type: application/x-www-form-urlencoded
Accept-Encoding: gzip, deflate
Host: zero.webappsecurity.com:8080
Content-Length: 134
Pragma: no-cache
Memo: 229:Auditor.SendAsyncronousRequest:Attack(CID:(null):AS:12,EID:9722923f-f8d3-49c2-90bd-
7c0e15901c18,ST:AuditAttack,AT:PostParamManipulation,APD:name,I:(1,0),R:False,SM:2,SID:2B666DF8737ED81D3EF5B76B8D0BC063,PSID:5E722BFAFDDB6D19D23FCA346756D875)
Connection: Keep-Alive
Cookie: JSESSIONID=5978DCF176177C7D4DE88DDA99C02E59;CustomCookie=WebInspect52840ZXEB273C7D063541258EE33C35AC817ACDYD77F

item=1&name='%09OR%09(select%09count(*)%09from%09sysobjects)%3e%090%09OR%09'4'%3d'0&ccnum=&cvv2=3&addr=&expirationMon=&expirationYear=
```

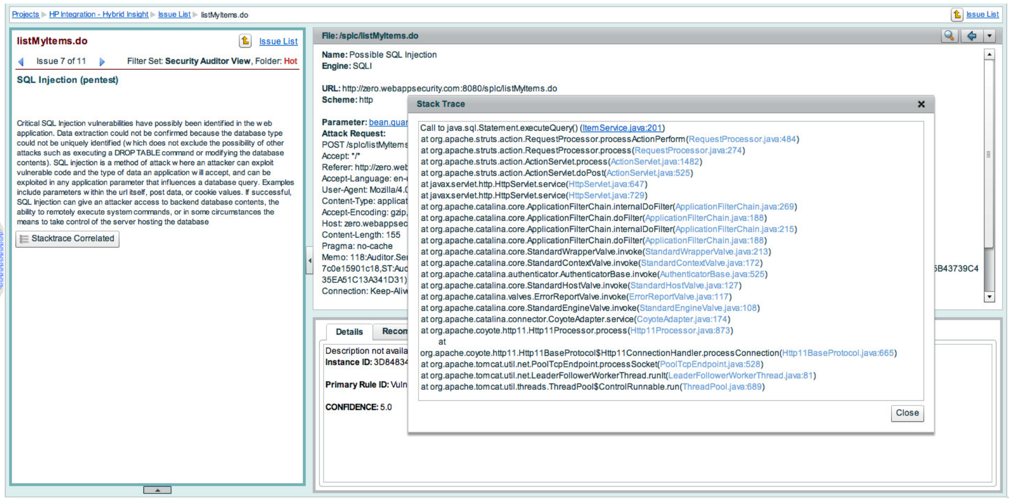
Security Tester

Hybrid Integrated Security Testing



FortifyHybrid Integration Demo

How did we do that?



Projects ▸ HP Integration - Hybrid Insight ▸ Issue List ▸ listMyItems.do

listMyItems.do Issue List

Issue 7 of 11 Filter Set: **Security Auditor View**, Folder: Hot

SQL Injection (pentest)

Critical SQL Injection vulnerabilities have possibly been identified in the w eb application. Data extraction could not be confirmed because the database type could not be uniquely identified (w hich does not exclude the possibility of other attacks such as executing a DROP TABLE command or modifying the database contents). SQL injection is a method of attack w here an attacker can exploit vulnerable code and the type of data an application w ill accept, and can be exploited in any application parameter that influences a database query. Examples include parameters w ithin the url itself, post data, or cookie values. If successful, SQL Injection can give an attacker access to backend database contents, the ability to remotely execute system commands, or in some circumstances the means to take control of the server hosting the database

Stacktrace Correlated

File: /splc/listMyItems.do

Name: Possible SQL Injection
Engine: SQLI

URL: http://zero.webappsecurity.com:8080/splc/listMyItems.do
Scheme: http

Parameter: bean.quar
Attack Request:
```
POST /splc/listMyItems
Accept: */*
Referer: http://zero.web
Accept-Language: en-
User-Agent: Mozilla/4.0
Content-Type: applicat
Accept-Encoding: gzip,
Host: zero.webappsec
Content-Length: 155
Pragma: no-cache
Memo: 118:Auditor.Se
7c0e15901c18,ST:Auc                                          5B43739C4
35EA51C13A341D31)
Connection: Keep-Aliv
```

Details | Recom

Description not availa
Instance ID: 3D84834

Primary Rule ID: Vuln

CONFIDENCE: 5.0

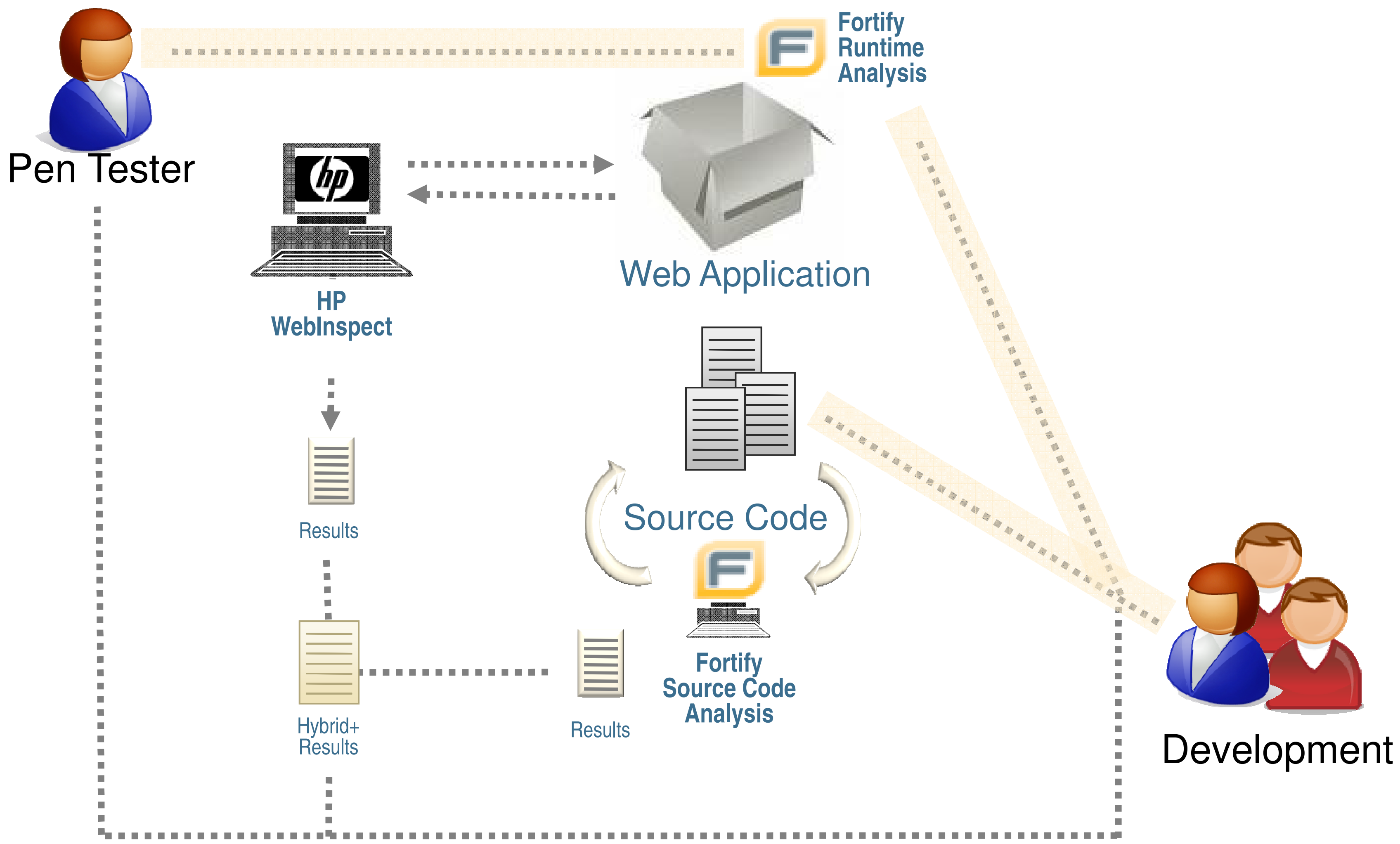
Stack Trace

```
Call to java.sql.Statement.executeQuery() (ItemService.java:201)
at org.apache.struts.action.RequestProcessor.processActionPerform(RequestProcessor.java:484)
at org.apache.struts.action.RequestProcessor.process(RequestProcessor.java:274)
at org.apache.struts.action.ActionServlet.process(ActionServlet.java:1482)
at org.apache.struts.action.ActionServlet.doPost(ActionServlet.java:525)
at javax.servlet.http.HttpServlet.service(HttpServlet.java:647)
at javax.servlet.http.HttpServlet.service(HttpServlet.java:729)
at org.apache.catalina.core.ApplicationFilterChain.internalDoFilter(ApplicationFilterChain.java:269)
at org.apache.catalina.core.ApplicationFilterChain.doFilter(ApplicationFilterChain.java:188)
at org.apache.catalina.core.ApplicationFilterChain.internalDoFilter(ApplicationFilterChain.java:215)
at org.apache.catalina.core.ApplicationFilterChain.doFilter(ApplicationFilterChain.java:188)
at org.apache.catalina.core.StandardWrapperValve.invoke(StandardWrapperValve.java:213)
at org.apache.catalina.core.StandardContextValve.invoke(StandardContextValve.java:172)
at org.apache.catalina.authenticator.AuthenticatorBase.invoke(AuthenticatorBase.java:525)
at org.apache.catalina.core.StandardHostValve.invoke(StandardHostValve.java:127)
at org.apache.catalina.valves.ErrorReportValve.invoke(ErrorReportValve.java:117)
at org.apache.catalina.core.StandardEngineValve.invoke(StandardEngineValve.java:108)
at org.apache.catalina.connector.CoyoteAdapter.service(CoyoteAdapter.java:174)
at org.apache.coyote.http11.Http11Processor.process(Http11Processor.java:873)
    at
org.apache.coyote.http11.Http11BaseProtocol$Http11ConnectionHandler.processConnection(Http11BaseProtocol.java:665)
at org.apache.tomcat.util.net.PoolTcpEndpoint.processSocket(PoolTcpEndpoint.java:528)
at org.apache.tomcat.util.net.LeaderFollowerWorkerThread.runIt(LeaderFollowerWorkerThread.java:81)
at org.apache.tomcat.util.threads.ThreadPool$ControlRunnable.run(ThreadPool.java:689)
```

Close

- “Runtime Data” comes from Runtime Analysis
  - ✓ Today Fortify leverages this to monitor and guard applications
- Fortify Runtime Analysis + WebInspect = Hybrid 2.0
  - Runtime Analysis is required to ensure proper mapping of SAST/DAST results
  - Runtime Analysis allows testers and programmers to see “inside” the app
  - Runtime analysis makes black box testing – white box testing

Introducing Hybrid 2.0



Hybrid 1.0 (2005 Technology – Available since 2006)

Hybrid Aggregation:
The complete set of results

→

Unified management & reporting

Ability to combine SAST and DAST findings for integrated prioritization and reporting.

Hybrid 2.0 (An HP/Fortify **exclusive** advantage)

Hybrid Correlation
The accurate results

Hybrid Insight
The actionable results

→

Reduced time and cost to fix vulnerabilities

Ability to follow test findings “into” the program and the code to see the root cause.

Thank you !