Interactive Code Reviews

Use ‘Manual IAST’ for Effective CR

Tamir Shavro
Head of Seeker R&D, Synopsys

OWASP Israel, September 2016

SYNOPSYS®
Silicon to Software™

Speaker

- Head of Seeker R&D at Synopsys
- 18 years of XP in the Software & Security Fields
- Hands-on Pen-Tester XP
- IDF Intelligence Corps, Tech Unit

Agenda

- Background & Motivation
- Core Idea of the Solution
- Implementation Steps
- Live Demo
- Pros & Cons
- Q&A

Background & Motivation

What’s wrong with current review process?

Background

What’s wrong with current review process?

- Many ‘Too-s’
  - Too much code, too short timeframe
  - Too many attack vectors
  - Too many entry points / pages / parameters
  - Too many new frameworks / third party components
  - Too often, too complex to follow and understand

Motivation

What if I could tell you where to look…

- Don’t spend time chasing ghosts
  HINT: no LDAP activity → LDAP Injection goes off the list
- Make new frameworks transparent by looking at the provider level
- Focus only on relevant code sections
- Order of magnitude improvement of value for $$$

Core Idea of the Solution

Empowering the Reviewer with Runtime Technology

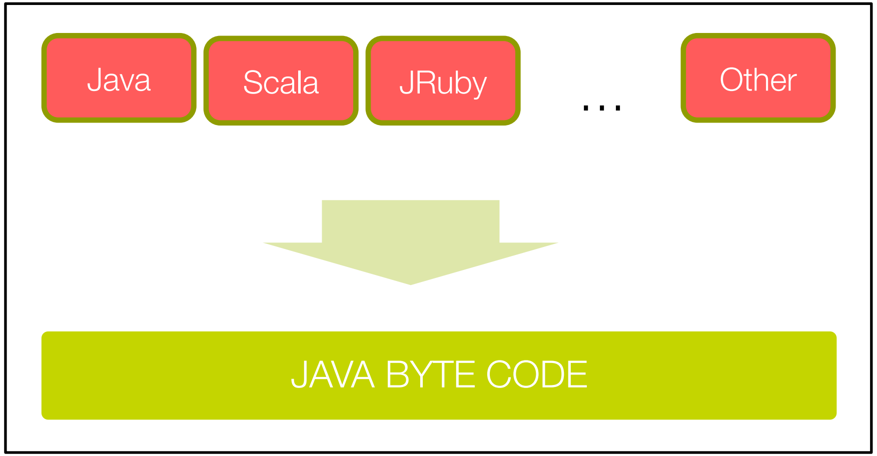
Core Idea of the Solution

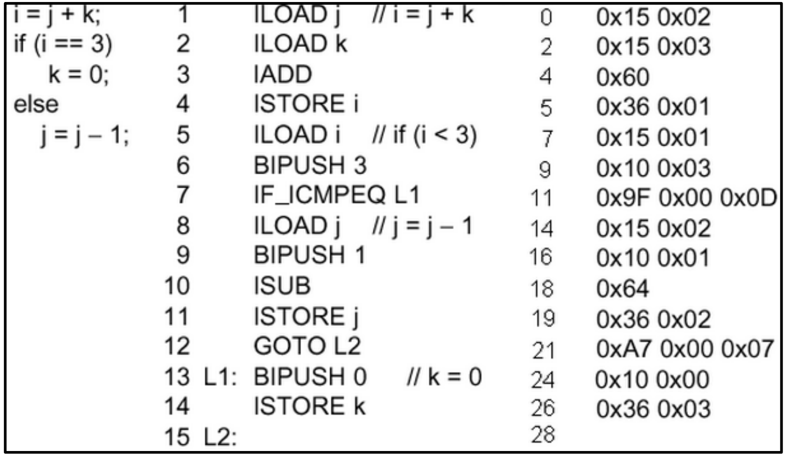
What is Runtime/Interactive Technology?

- Runtime information could be monitored easily by using existing technologies
  - Live debugging techniques at provider level
  - On-the-fly instrumentation/profiling techniques

Core Idea of the Solution

Basic Byte-Code Debugging Explained



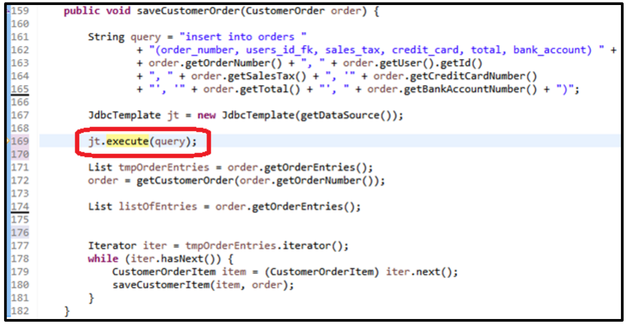


```
i = j + k;      1        ILOAD j    // i = j + k     0    0x15 0x02
if (i == 3)     2        ILOAD k                     2    0x15 0x03
    k = 0;      3        IADD                        4    0x60
else            4        ISTORE i                    5    0x36 0x01
   j = j – 1;   5        ILOAD i    // if (i < 3)    7    0x15 0x01
                6        BIPUSH 3                    9    0x10 0x03
                7        IF_ICMPEQ L1                11   0x9F 0x00 0x0D
                8        ILOAD j    // j = j – 1     14   0x15 0x02
                9        BIPUSH 1                    16   0x10 0x01
                10       ISUB                        18   0x64
                11       ISTORE j                    19   0x36 0x02
                12       GOTO L2                     21   0xA7 0x00 0x07
                13  L1:  BIPUSH 0     // k = 0       24   0x10 0x00
                14       ISTORE k                    26   0x36 0x03
                15  L2:                              28
```

Core Idea of the Solution

Debugging at Provider Level Explained

Tested Application

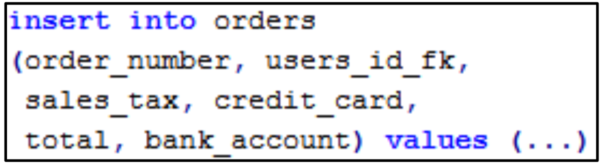


```java
    public void saveCustomerOrder(CustomerOrder order) {

        String query = "insert into orders "
                + "(order_number, users_id_fk, sales_tax, credit_card, total, bank_account) " +
                + order.getOrderNumber() + ", " + order.getUser().getId()
                + ", " + order.getSalesTax() + ", '" + order.getCreditCardNumber()
                + "', '" + order.getTotal() + "', " + order.getBankAccountNumber() + ")";

        JdbcTemplate jt = new JdbcTemplate(getDataSource());

        jt.execute(query);

        List tmpOrderEntries = order.getOrderEntries();
        order = getCustomerOrder(order.getOrderNumber());

        List listOfEntries = order.getOrderEntries();


        Iterator iter = tmpOrderEntries.iterator();
        while (iter.hasNext()) {
            CustomerOrderItem item = (CustomerOrderItem) iter.next();
            saveCustomerItem(item, order);
        }
    }
```

Provider Level Breakpoints

com.mysql.jdbc.Statement.executeQuery(..)

Runtime Data

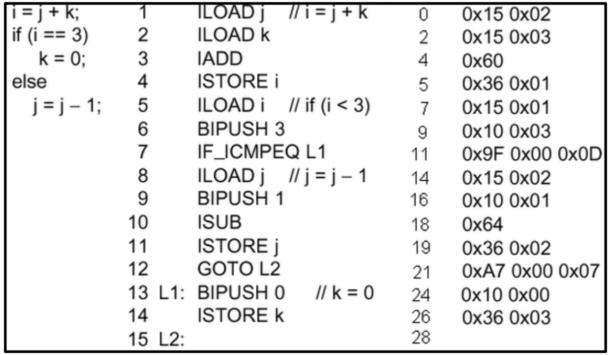


```sql
insert into orders
(order_number, users_id_fk,
 sales_tax, credit_card,
 total, bank_account) values (...)
```

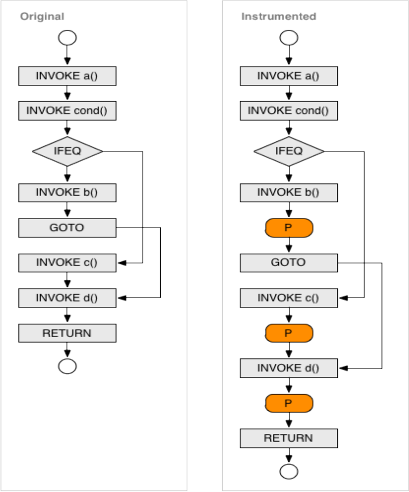
SYNOPSYS®

Core Idea of the Solution

On-the-fly Instrumentation Explained



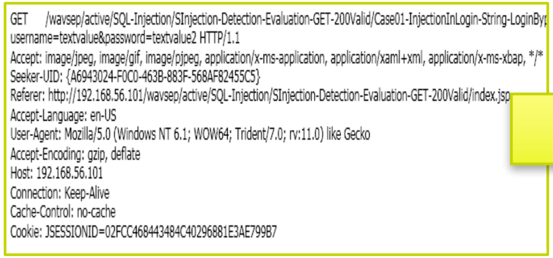
```
i = j + k;       1      ILOAD j    // i = j + k     0    0x15 0x02
if (i == 3)      2      ILOAD k                     2    0x15 0x03
    k = 0;       3      IADD                        4    0x60
else             4      ISTORE i                    5    0x36 0x01
   j = j – 1;    5      ILOAD i    // if (i < 3)    7    0x15 0x01
                 6      BIPUSH 3                    9    0x10 0x03
                 7      IF_ICMPEQ L1               11    0x9F 0x00 0x0D
                 8      ILOAD j    // j = j – 1    14    0x15 0x02
                 9      BIPUSH 1                   16    0x10 0x01
                10      ISUB                       18    0x64
                11      ISTORE j                   19    0x36 0x02
                12      GOTO L2                    21    0xA7 0x00 0x07
                13  L1: BIPUSH 0     // k = 0      24    0x10 0x00
                14      ISTORE k                   26    0x36 0x03
                15  L2:                            28
```



Core Idea of the Solution

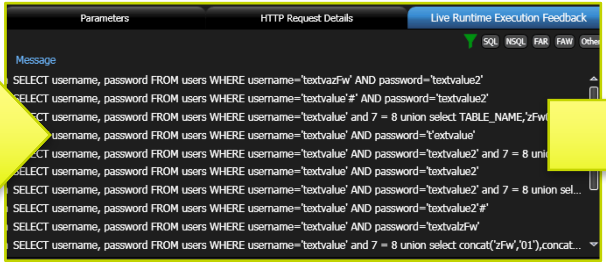
So…What's in it for us?

HTTP Request



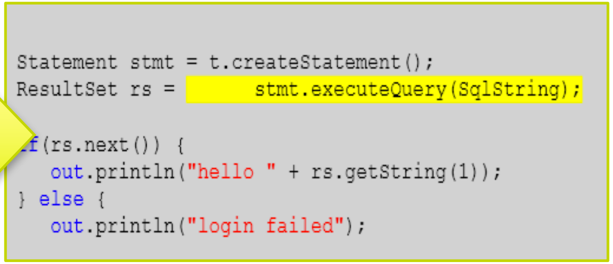
```
GET     /wavsep/active/SQL-Injection/SInjection-Detection-Evaluation-GET-200Valid/Case01-InjectionInLogin-String-LoginByp
username=textvalue&password=textvalue2 HTTP/1.1
Accept: image/jpeg, image/gif, image/pjpeg, application/x-ms-application, application/xaml+xml, application/x-ms-xbap, */*
Seeker-UID: {A6943024-F0C0-463B-883F-568AF82455C5}
Referer: http://192.168.56.101/wavsep/active/SQL-Injection/SInjection-Detection-Evaluation-GET-200Valid/index.jsp
Accept-Language: en-US
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; Trident/7.0; rv:11.0) like Gecko
Accept-Encoding: gzip, deflate
Host: 192.168.56.101
Connection: Keep-Alive
Cache-Control: no-cache
Cookie: JSESSIONID=02FCC468443484C40296881E3AE799B7
```

Runtime Data



```
Message
SELECT username, password FROM users WHERE username='textvazFw' AND password='textvalue2'
SELECT username, password FROM users WHERE username='textvalue'#' AND password='textvalue2'
CT username, password FROM users WHERE username='textvalue' and 7 = 8 union select TABLE_NAME,'zFw
username, password FROM users WHERE username='textvalue' AND password='t'extvalue'
CT username, password FROM users WHERE username='textvalue' AND password='textvalue2' and 7 = 8 uni
SELECT username, password FROM users WHERE username='textvalue' AND password='textvalue2'
SELECT username, password FROM users WHERE username='textvalue' AND password='textvalue2' and 7 = 8 union sel...
SELECT username, password FROM users WHERE username='textvalue' AND password='textvalue2'#'
SELECT username, password FROM users WHERE username='textvalue' AND password='textvalzFw'
SELECT username, password FROM users WHERE username='textvalue' and 7 = 8 union select concat('zFw','01'),concat...
```

Line of Code



```
Statement stmt = t.createStatement();
ResultSet rs =       stmt.executeQuery(SqlString);

f(rs.next()) {
   out.println("hello " + rs.getString(1));
} else {
   out.println("login failed");
```

Implementation Steps

Empowering the Reviewer with Runtime Technology

Implementation Steps

Step I - Allow Remote Debugging

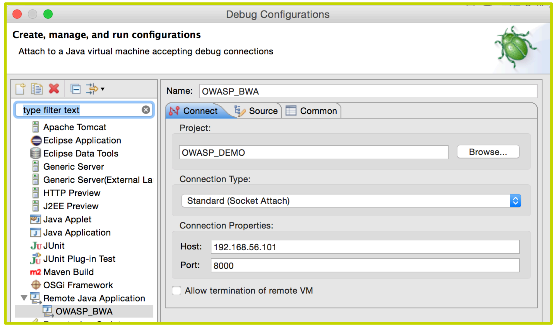
- Alter JVM arguments to allow debugging (same as increasing –Xmx)
- Implemented by adding one line to the startup script of the app

```
JAVA_OPTS="$JAVA_OPTS -agentlib:jdwp=transport=dt_socket,server=y,address=8000,suspend=n"
```

Implementation Steps

Step II – Attach to Target App

- Using your favorite IDE (e.g. eclipse), create a remote connection and import breakpoints file at provider level



Debug Configurations

Create, manage, and run configurations

Attach to a Java virtual machine accepting debug connections

type filter text

- Apache Tomcat
- Eclipse Application
- Eclipse Data Tools
- Generic Server
- Generic Server(External La
- HTTP Preview
- J2EE Preview
- Java Applet
- Java Application
- JUnit
- JUnit Plug-in Test
- Maven Build
- OSGi Framework
- Remote Java Application
  - OWASP_BWA

Name: OWASP_BWA

Connect | Source | Common

Project: OWASP_DEMO (Browse...)

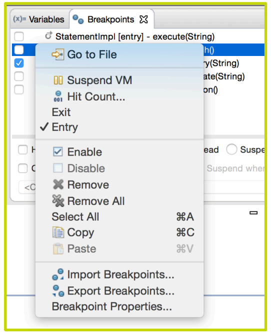
Connection Type: Standard (Socket Attach)

Connection Properties:

Host: 192.168.56.101

Port: 8000

☐ Allow termination of remote VM

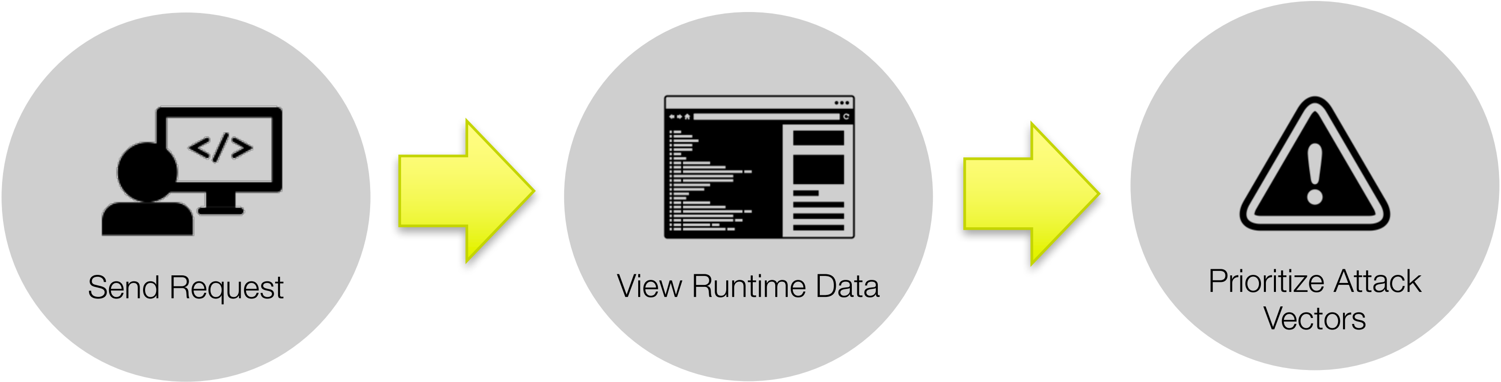


Variables | Breakpoints

StatementImpl [entry] - execute(String)

Implementation Steps

Step III – Use Runtime Data to Focus Your Attacks



Send Request

View Runtime Data

Prioritize Attack Vectors

Live Demo

Use Runtime Data During Manual PT/CR

Pros & Cons

Pros & Cons

Using Runtime During PT/CR

- Pros
  - More value for $$$
  - Makes the PT/CR more effective
  - We’re not chasing ghosts anymore
  - Simple to use
- Cons
  - Access to tested environment needed
  - Need to have the app up & running
  - Might not be possible when testing on production

Pros & Cons

Debugger vs. Profiler

- Debugger
  - Simple to use
  - Great at identifying entry points
  - Might be limited when with heavy traffic apps

- Profiler
  - Harder to fine tune to get relevant data
  - A bit more complex to use
  - Faster than debugger, can handle heavy traffic

Thank You!

Questions?

*Email to get BP pack: tamir.shavro at synopsys (dot) com
mail **title should be**: OWASP BP PACK

SYNOPSYS®
Silicon to Software™