On the Wasted DevOps Cycles

Munawar Hafiz
munawar@openrefactory.com

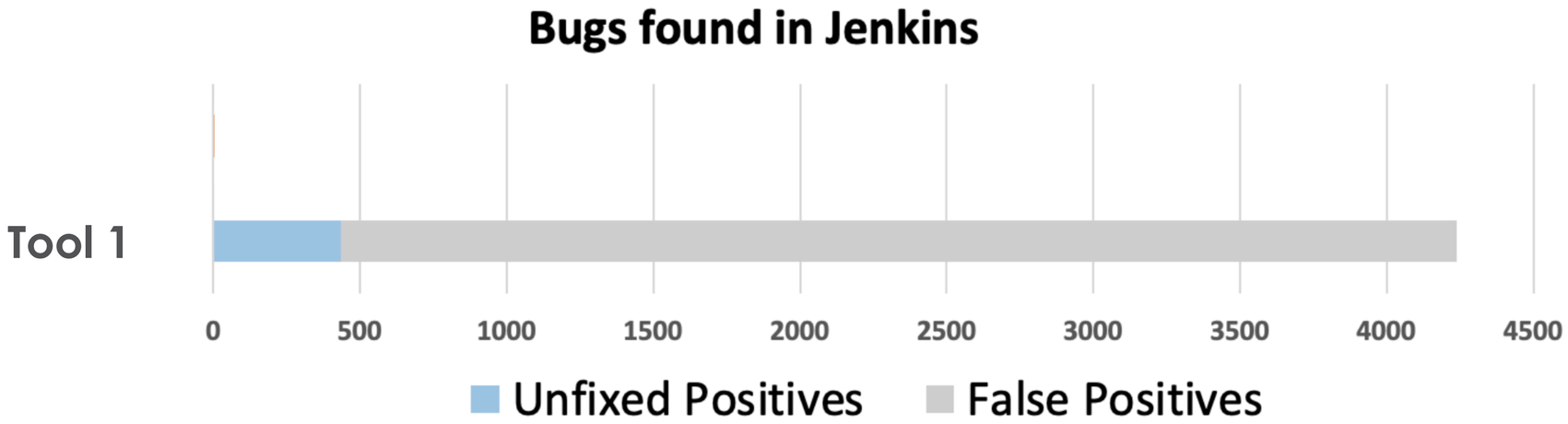
June 16, 2021

A Day In The Life Of A Developer

Project Details:

Application:	Jenkins v2.271
Description:	Most popular automation server
Relevance:	16,000 stars on GitHub; 6,600 forks
Size:	1,743 Java files; 319K LoC
Repo Link:	https://github.com/jenkinsci/jenkins

10,000 Bugs in 300,000 Lines of Stable Code



Tool 1 had 89.6% FP rate only considering the major bugs


The rate at which bugs
are created outpaces the
rate that they are fixed

What DevSecOps Engineers Endure?

* Only focus on high priority bugs. Still 50~70% FP rate.
* Between 15-40% of developer time is wasted because of chasing bugs.
* “Quality” fights happening before release slow things down.
* Use multiple tools since SAST tools miss bugs.

What VPs/CISOs Endure?

* Quality gates not reached; product release delayed.
* For a team of 500 engineers, about 500 man-months saved. Annual savings $1.1M - $3.2M.
* What if a critical bug is missed?

Two Emerging Challenges

Hidden Paths

Polyglot World

Hidden Paths


JAVA

eclipse

web
service

SWING!

JSP

JavaFx

spring

HIBERNATE

LOG4J

JDBC

PRIME FACES

jQuery
write less, do more.

JUnit

Struts 2

Apache
Tomcat

XML

Java
Servlets

Java EE

maven

{JSON}

```
public class Global {
    public static String s;
}

public class Z {
    Thread demo;
    public void foo() {
        MultithreadingDemo runnable = new MultithreadingDemo();
        Global.s = new String();
        demo = new Thread(runnable);
        demo.start();
    }
}

class MultithreadingDemo implements Runnable
{
    public void run()
    {
        System.out.println ("Thread running");
        Global.s.length();
    }
}
```

```
public class Global {
    public static String s;
}

public class Z {
    Thread demo;
    public void foo() {
        MultithreadingDemo runnable = new MultithreadingDemo();
        Global.s = new String();
        demo = new Thread(runnable);
        demo.start();
    }
}

class MultithreadingDemo implements Runnable
{
    public void run()
    {
        System.out.println ("Thread running");
        Global.s.length();
    }
}
```

The call to start () actually calls run()

Safe (No Null Deref) !!

Information Flow in Android Framework

```
public  class  LeakageApp  extends  Activity{

    private  User  user = null;
    protected  void  onRestart (){
        EditText  usernameText = (EditText)findViewById(R.id
            .username);
        EditText  passwordText = (EditText)findViewById(R.id
            .pwdString);
        String  uname = usernameText.toString ();
        String  pwd = passwordText.toString ();
        if(! uname.isEmpty () && !pwd.isEmpty ()) {
            this.user = new  User(uname , pwd);
        }
    }

    ...
```

* Example from FlowDroid paper by Arzt and colleagues

Information Flow in Android Framework

```
public  class  LeakageApp  extends  Activity{

    private  User  user = null;
    protected  void  onRestart (){
        EditText  usernameText = (EditText)findViewById(R.id
            .username);
        EditText  passwordText = (EditText)findViewById(R.id
            .pwdString);
        String  uname = usernameText.toString ();
        String  pwd = passwordText.toString ();
        if(! uname.isEmpty () && !pwd.isEmpty ()) {
            this.user = new  User(uname , pwd);
        }
    }

    ...
```

Password is a sensitive information

* Example from FlowDroid paper by Arzt and colleagues

Information Flow in Android Framework

```
public  class  LeakageApp  extends  Activity{

    private  User  user = null;
    protected  void  onRestart (){
        EditText  usernameText = (EditText)findViewById(R.id
            .username);
        EditText  passwordText = (EditText)findViewById(R.id
            .pwdString);
        String  uname = usernameText.toString ();
        String  pwd = passwordText.toString ();
        if(! uname.isEmpty () && !pwd.isEmpty ()) {
            this.user = new  User(uname , pwd);
        }
    }

    ...
```

Information flows to pwd

* Example from FlowDroid paper by Arzt and colleagues

Information Flow in Android Framework

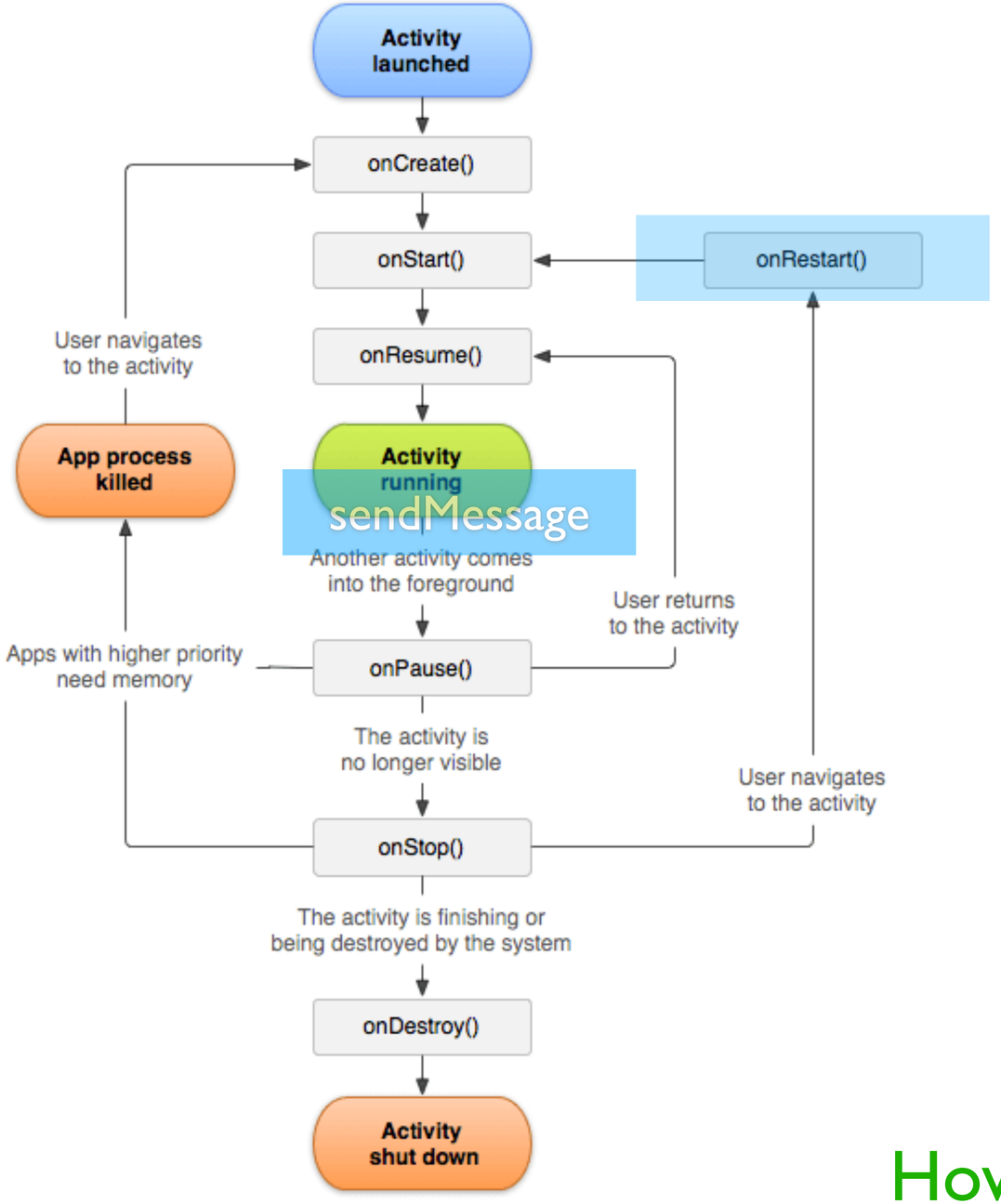
```
public  class  LeakageApp  extends  Activity{

    private  User  user = null;
    protected  void  onRestart (){
        EditText  usernameText = (EditText)findViewById(R.id
            .username);
        EditText  passwordText = (EditText)findViewById(R.id
            .pwdString);
        String  uname = usernameText.toString ();
        String  pwd = passwordText.toString ();
        if(! uname.isEmpty () && !pwd.isEmpty ()) {
            this.user = new  User(uname , pwd);
        }
    }

    ...
```

Information flows to user field

* Example from FlowDroid paper by Arzt and colleagues

```
public  class  LeakageApp  extends  Activity{

    ...

    public  void  sendMessage(View  view){
        if (user == null) {
            return;
        }
        Password  pwd = user.getpwd ();
        String  pwdString = pwd.getPassword ();
        String  temp = "";
        for (char c : pwdString.toCharArray ()) {
            temp  += c + "_";
        }
        String  message = "User: " +
            user.getName () + " | Pwd: " + temp;
        SmsManager  sms = SmsManager.getDefault ();
        sms.sendTextMessage("+1 217 722 1721",
            null , message , null , null);
    }
}
```

Another method that will be called with tainted data by the framework



How the call happens?

```
public  class  LeakageApp  extends  Activity{

    ...

    public  void  sendMessage(View  view){
        if (user == null) {
            return;
        }
        Password  pwd = user.getpwd ();
        String  pwdString = pwd.getPassword ();
        String  temp = "";
        for (char c : pwdString.toCharArray ()) {
            temp  += c + "_";
        }
        String  message = "User: " +
            user.getName () + " | Pwd: " + temp;
        SmsManager  sms = SmsManager.getDefault ();
        sms.sendTextMessage("+1 217 722 1721",
            null , message , null , null);
    }
}
```

Information flows to pwd
from user's field

```
public  class  LeakageApp  extends  Activity{

    ...

    public  void  sendMessage(View  view){
        if (user == null) {
            return;
        }
        Password  pwd = user.getpwd ();
        String  pwdString = pwd.getPassword ();
        String  temp = "";
        for (char c : pwdString.toCharArray ()) {
            temp  += c + "_";
        }
        String  message = "User: " +
            user.getName () + " | Pwd: " + temp;
        SmsManager  sms = SmsManager.getDefault ();
        sms.sendTextMessage("+1 217 722 1721",
            null , message , null , null);
    }
}
```

Information flows to pwdString

```
public  class  LeakageApp  extends  Activity{

    ...

    public  void  sendMessage(View  view){
        if (user == null) {
            return;
        }
        Password  pwd = user.getpwd ();
        String  pwdString = pwd.getPassword ();
        String  temp = "";
        for (char c : pwdString.toCharArray ()) {
            temp  += c + "_";
        }
        String  message = "User: " +
            user.getName () + " | Pwd: " + temp;
        SmsManager  sms = SmsManager.getDefault ();
        sms.sendTextMessage("+1 217 722 1721",
            null , message , null , null);
    }
}
```

Information flows to temp

```
public  class  LeakageApp  extends  Activity{

    ...

    public  void  sendMessage(View  view){
        if (user == null) {
            return;
        }
        Password  pwd = user.getpwd ();
        String  pwdString = pwd.getPassword ();
        String  temp = "";
        for (char c : pwdString.toCharArray ()) {
            temp  += c + "_";
        }
        String  message = "User: " +
            user.getName () + " | Pwd: " + temp;
        SmsManager  sms = SmsManager.getDefault ();
        sms.sendTextMessage("+1 217 722 1721",
            null , message , null , null);
    }
}
```

Information flows to message

```
public  class  LeakageApp  extends  Activity{

    ...

    public  void  sendMessage(View  view){
        if (user == null) {
            return;
        }
        Password  pwd = user.getpwd ();
        String  pwdString = pwd.getPassword ();
        String  temp = "";
        for (char c : pwdString.toCharArray ()) {
            temp  += c + "_";
        }
        String  message = "User: " +
            user.getName () + " | Pwd: " + temp;
        SmsManager  sms = SmsManager.getDefault ();
        sms.sendTextMessage("+1 217 722 1721",
            null , message , null , null);
    }
}
```

Sensitive Information Leaked

Polyglot World

Execution

```java
public class Z3NativeConnector {
    static {
        try {
            System.loadLibrary("ordimacs");
        } catch (UnsatisfiedLinkError ex) {
            System.loadLibrary("libordimacs");
        }
    }

    public native String prepareDimacs(String expr);
}
```

```
JNIEXPORT jstring JNICALL
    Java_com_openrefactory_core_java_pointeranalysis_constraints_Z3NativeConnector
    _prepareDimacs(JNIEnv *env, jobject obj, jstring eStr)
{
  // Issue 925
  // Number of maximum atomic constraints (not unique) we are going to
  // allow in the CNF result got from Z3 solver.
  unsigned MAX_RESULTANT_ATOMIC_CONSTRAINTS_ALLOWED = 900;
```

JNI call from Java to a C++ method

Data Loading

```
public  class  LeakageApp  extends  Activity{

    private  User  user = null;
    protected  void  onRestart (){
        EditText  usernameText = (EditText)findViewById(R.id
            .username);
        EditText  passwordText = (EditText)findViewById(R.id
            .pwdString);
        String  uname = usernameText.toString ();
        String  pwd = passwordText.toString ();
        if(! uname.isEmpty () && !pwd.isEmpty ()) {
            this.user = new  User(uname , pwd);
        }
    }

    ...
```

Data loading from JSON is another form of polyglot program

Shared ID

```
public  class  LeakageApp  extends  Activity{

    ...

    public  void  sendMessage(View  view){
        if (user == null) {
            return;
        }
        Password  pwd = user.getpwd ();
        String  pwdString = pwd.getPassword ();
        String  temp = "";
        for (char c : pwdString.toCharArray ()) {
            temp  += c + "_";
        }
        String  message = "User: " +
            user.getName () + " | Pwd: " + temp;
        SmsManager  sms = SmsManager.getDefault ();
        sms.sendTextMessage("+1 217 722 1721",
            null , message , null , null);
    }
}
```

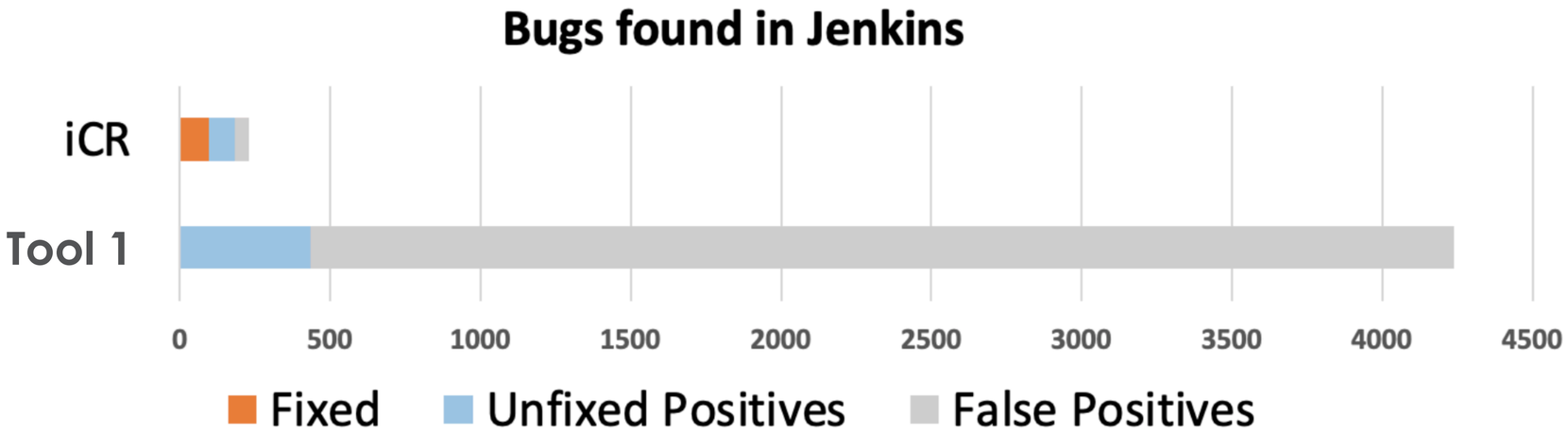
Sharing ID between Java and JSON

Shared ID

```
<aop:aspect id="beforeExample" ref="aBean">

    <aop:before
      pointcut-ref="dataAccessOperation"
      method="doAccessCheck"/>

    ...

</aop:aspect>
```

Sharing ID between Java and JSON

Bugs That Matter



iCR reported real bugs in 80% of the cases, generated fixes in 42% of them

Cloud: https://www.openrefactory.com/introductory-offer/

On Prem: info@openrefactory.com