REG

By: Nicole Gaehle, Omar Salih, Katrina Clark, Will Smith, and Rebecca Craine

FindBugs

FindBugs - javaissues

File Edit View Navigation Designation Help

Class name Filter: Filter

Group bugs by: Category | Bug Kind | Bug Pattern | ↔ | Bug Rank | De

Bugs (306)

- Correctness (14)
- Bad practice (30)
- Experimental (4)
- Internationalization (8)
- Malicious code vulnerability (38)
- Multithreaded correctness (46)
- Performance (61)
- Dodgy code (105)

No cloud selected | Enable cloud plugin...

0 reviewed bugs / 0 total reviews

View in browser

Find | Next | Previous

http://findbugs.sourceforge.net

UNIVERSITY OF MARYLAND

Tool: FindBugs

- FindBugs uses *static analysis* to inspect Java bytecode for occurrences of bug patterns. Static analysis means that FindBugs can find bugs by simply inspecting a program's code: executing the program is not necessary.
- FindBugs looks for bugs in Java programs. It is based on the concept of *bug patterns*. A bug pattern is a code idiom that is often an error. Bug patterns arise for a variety of reasons:
  - Difficult language features
  - Misunderstood API methods
  - Misunderstood invariants when code is modified during maintenance
  - Garden variety mistakes: typos, use of the wrong boolean operator
- Matched the following CWEs:
  - 89
  - 395
  - 502
  - 821

Java Decompiler

JD-GUI

File Edit Navigation Search Help

secondassignment3806-20170312T205643Z-001.zip

- secondassignment3806
  - build/classes
    - umsl/edu
      - account.class
      - atm.class
        - atm
    - _netbeans_automatic_build
    - _netbeans_update_resources
  - nbproject
  - src/umsl/edu

Search

Search string (* = any string, ? = any character):

*atm

Search For: Type, Constructor, String Constant, Field, Method

Limit To: Declarations, References

4 matching entries:

- secondassignment3806-20170312T205643Z-001.zip
  - secondassignment3806
    - build/classes/umsl/edu
      - atm.class
    - nbproject
    - src/umsl/edu

Open Cancel

atm.class

```
package umsl.edu;

import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import java.io.PrintStream;
import java.util.Scanner;

public class atm
{
  account[] ac = new account[3];
  boolean accountsLoaded = false;

  public void loadaccts()
    throws IOException
  {
    try
    {
      FileInputStream inStream = new FileInputStream("file1.out");
      ObjectInputStream is = new ObjectInputStream(inStream);
      this.ac = ((account[])is.readObject());
      is.close();
      this.accountsLoaded = true;
    }
    catch (Exception ioe)
    {
      System.out.println("No file found");
    }
  }

  public void createAcct()
  {
    account[] ac = new account[3];
    for (int i = 0; i < 3; i++) {
      ac[i] = new account(100.0D);
    }
  }
```

Tool: Java Decompiler

- The java decompiler aims to develop tools in order to decompile and analyze Java 5 “byte code” and later versions.
- It has four different options in which you can download to utilize the tool:
  - JD-Core
  - JD-GUI
  - JD-Eclipse
  - JD-IntelliJ
  -
- JD-GUI:
  - - a standalone graphical utility that displays Java source codes of “.class” files. Ability to browse the reconstructed source code with the JD-GUI for instant access to methods and fields.
- Matched the following CWEs:
  - 360
  - 484
  - 545

14 CWEs on Java

1. CWE-395: Use of NullPointerException Catch to Detect NULL Pointer Dereference**
2. CWE-360: Trust of System Event Data**
3. CWE-821: Incorrect Synchronization**
4. CWE-484: Omitted Break Statement in Switch**
5. CWE-89: Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection')**
6. CWE-545: Use of Dynamic Class Loading**
7. CWE-502: Deserialization of Untrusted Data**
8. CWE-362: Concurrent Execution using Shared Resource with Improper Synchronization ('Race Condition')
9. CWE-543: Use of Singleton Pattern Without Synchronization in a Multithreaded Context
10. CWE-103: Struts: Incomplete validate() Method Definition
11. CWE-497: Exposure of System Data to an Unauthorized Control Sphere
12. CWE-299: Improper Check for Certificate Revocation
13. CWE-496: Public Data Assigned to Private Array-Typed Field
14. CWE-767 Access to Critical Private Variable via Public Method

**Notes Exploits Shown in Presentation

CWEs: How to Exploit

CWE-395: Use of NullPointerException Catch to Detect NULL Pointer Dereference

- NullPointerException has the potential to be malicious since a programmer should not be running into this code.
- This is only acceptable if the code is part of a test harness that supplies unexpected input to the classes under test.
- It is a bad practice to catch NullPointerException due to the fact that a program explicitly throws a NullPointerException to signal an error condition.

Example Language: **Java**

```
try {
  mysteryMethod();
} catch (NullPointerException npe) {

}
```

https://cwe.mitre.org/data/definitions/395.html
https://www.owasp.org/index.php/Catch_NullPointerException

Example of CWE-395: Use of NullPointerException Catch to Detect NULL Pointer Dereference

FindBugs - javaissues

File Edit View Navigation Designation Help

Class name filter: Filter

Group bugs by: Category | Bug Kind | Bug Pattern | ↔ | Bug Rank | De

- Database resource not closed on all paths (1)
- Incorrect definition of Serializable class (1)
- Incorrect use of finalizers (2)
- Null pointer dereference (1)
  - equals() method does not check for null argument (1)
    - equals(Object) does not check for null argument
- Serializable class with no Version ID (5)

No cloud selected — Enable cloud plugin...

ClientXid in org.apache.derby.client — View in browser

```
1 Unable to find source
```

Find | Next | Previous

equals(Object) does not check for null argument
In <Unknown>
In method org.apache.derby.client.ClientXid.equals(Object)

equals() method does not check for null argument

This implementation of equals(Object) violates the contract defined by java.lang.Object.equals() because it does not check for null being passed as the argument. All equals() methods should return false if passed a null value.

Bug kind and pattern: NP - NP_EQUALS_SHOULD_HANDLE_NULL_ARGUMENT

http://findbugs.sourceforge.net

UNIVERSITY OF MARYLAND

CWEs: How to Exploit

CWE-360: Trust of System Event Data

- This can simply be malicious to the end-user because events cannot be trusted.
- Events often does not have any type of authentication framework to allow them to be verified from a trusted source.
- One should not trust the system-event information because if commands are executed based on trust then they could potentially take actions based on a spoofed identity.

https://cwe.mitre.org/data/definitions/360.html

Example Language: **Java**

```
public void actionPerformed(ActionEvent e) {
  if (e.getSource() == button) {
    System.out.println("print out secret information");
  }
}
```

Example of CWE 360: Trust of System Event Data

JD-GUI

File Edit Navigation Search Help

gooeybuttercake.zip

- gooeybuttercake
  - build/classes
    - umsl/edu
      - buttontest.class
        - buttontest
          - fancyButton : JButton
          - plainButton : JButton
          - actionPerformed(ActionEvent) : void
          - buttontest()
          - main(String[]) : void

buttontest.class

```
private JButton plainButton;
private JButton fancyButton;

public buttontest()
{
  super("Testing Buttons");

  Container mycontainer = getContentPane();
  mycontainer.setLayout(new FlowLayout());

  this.plainButton = new JButton("Plain Button");
  mycontainer.add(this.plainButton);

  Icon bug1 = new ImageIcon("bug1.gif");
  Icon bug2 = new ImageIcon("bug2.gif");
  this.fancyButton = new JButton("Fancy Button", bug1);
  this.fancyButton.setRolloverIcon(bug2);
  mycontainer.add(this.fancyButton);

  this.fancyButton.addActionListener(this);
  this.plainButton.addActionListener(this);
  setSize(275, 100);
  setVisible(true);
}

public void actionPerformed(ActionEvent event)
{
  JOptionPane.showMessageDialog(null, "You pressed: " + event

    .getActionCommand());
}

public static void main(String[] args)
{
  buttontest application = new buttontest();
  application.setDefaultCloseOperation(3);
}
```

Search

Search string (* = any string, ? = any character):

*action

Search For: Type, Constructor, String Constant, Field, Method

Limit To: Declarations, References

2 matching entries:

- gooeybuttercake.zip
  - gooeybuttercake
    - build/classes/umsl/edu
      - buttontest.class
    - src/umsl/edu

Open Cancel

CWEs: How to Exploit

CWE-821: Incorrect Synchronization

- The software utilizes a shared resource in a concurrent manner but it does not correctly synchronize access to the resource.
- It modifies and reads the applications data and alters the execution logic. Basically it affects integrity and confidentiality.
- It can be malicious if the attacker can influence the source code the end-user is sharing.

https://cwe.mitre.org/data/definitions/821.html

Example of CWE-821: Incorrect Synchronization

FindBugs - javaissues

File Edit View Navigation Designation Help

Class name filter: Filter

Group bugs by: Category | Bug Kind | Bug Pattern | ↔ | Bug Rank | De

- Multithreaded correctness (46)
  - Inconsistent synchronization (17)
    - Inconsistent synchronization (17)
      - Inconsistent synchronization of Connection.availableFo
      - Inconsistent synchronization of Connection.xaState_; l
      - Inconsistent synchronization of Sqlca.connection_; locl

No cloud selected | Enable cloud plugin...

Connection in org.apache.derby.client.am | View in browser

1 Unable to find source

Find | Next | Previous

Inconsistent synchronization of Connection.availableForReuse_; locke
In <Unknown>
Field org.apache.derby.client.am.Connection.availableForReuse_
Synchronized 83% of the time

Inconsistent synchronization

The fields of this class appear to be accessed inconsistently with respect to synchronization. This bug report indicates that the bug pattern detector judged that
The class contains a mix of locked and unlocked accesses,
The class is **not** annotated as javax.annotation.concurrent.NotThreadSafe,
At least one locked access was performed by one of the class's own methods, and
The number of unsynchronized field accesses (reads and writes) was no more than one third of all accesses, with writes being weighed twice as high as reads
A typical bug matching this bug pattern is forgetting to synchronize one of the methods in a class that is intended to be thread-safe.
You can select the nodes labeled "Unsynchronized access" to show the code locations where the detector believed that a field was accessed without synchronization.
Note that there are various sources of inaccuracy in this detector; for example, the detector cannot statically detect all situations in which a lock is held. Also, even when the detector is accurate in distinguishing locked vs. unlocked accesses, the code in question may still be correct.

Bug kind and pattern: IS - IS2_INCONSISTENT_SYNC

http://findbugs.sourceforge.net

UNIVERSITY OF MARYLAND

CWEs: How to Exploit

CWE-484: Omitted Break Statement in Switch

- Write a switch statement
- Don't include a switch or break statement between cases.
- This will cause the code to execute on multiple conditions instead of one which can cause malicious code to be inserted

Bad Code:

```
switch (myVariable) {
  case 1:
    foo();
    break;
  case 2:  // Both 'doSomething()' and 'doSomethingElse()' will be executed. Is it on purpose ?
    doSomething();
  default:
    doSomethingElse();
    break;
}
```

Good Code:

```
switch (myVariable) {
  case 1:
    foo();
    break;
  case 2:
    doSomething();
    break;
  default:
    doSomethingElse();
    break;
}
```

http://cwe.mitre.org/data/definitions/484.html

Example of CWE-484: Omitted Break Statement in Switch

Payroll.class - Java Decompiler

File Edit Navigation Search Help

AndrewNicoleASS1-20170314T003255Z-001.zip

- AndrewNicoleASS1
  - build/classes
    - andrewnicoleass1
      - CommissionEmployee.class
      - Employee.class
      - HourlyEmployee.class
      - Payroll.class
        - Payroll
      - SalaryEmployee.class

Search

Search string (* = any string, ? = any character):

*c

Search For: Type, Constructor, String Constant, Field, Method

Limit To: Declarations, References

17 matching entries:

- AndrewNicoleASS1-20170314T003255Z-001.zip
  - AndrewNicoleASS1
    - build/classes/andrewnicoleass1
      - CommissionEmployee.class
      - Employee.class
      - HourlyEmployee.class
      - Payroll.class
      - SalaryEmployee.class

Open Cancel

Payroll.class

```
    }
    this.empLoaded = true;
    System.out.println("Employees have been loaded. ");
  }

  public void populateEmployees()
  {
    boolean empLoaded = false;
    if (empLoaded == true) {
      System.out.println("The emplyee has already been loaded.");
    } else {
      for (int i = 0; i < this.empArray.length; i++)
      {
        Scanner sc = new Scanner(System.in);
        System.out.println("What kind of employee are you? ");
        System.out.println("1 - Hourly ");
        System.out.println("2- Salary ");
        System.out.println("3- Commission ");
        int input = sc.nextInt();
        switch (input)
        {
        case 1:
          this.empArray[i] = new HourlyEmployee();
          break;
        case 2:
          this.empArray[i] = new SalaryEmployee();
          break;
        case 3:
          this.empArray[i] = new CommissionEmployee();
        }
        empLoaded = true;
      }
    }
  }

  public void SaveEmployees()
  {
    try
    {
```

CWEs: How to Exploit

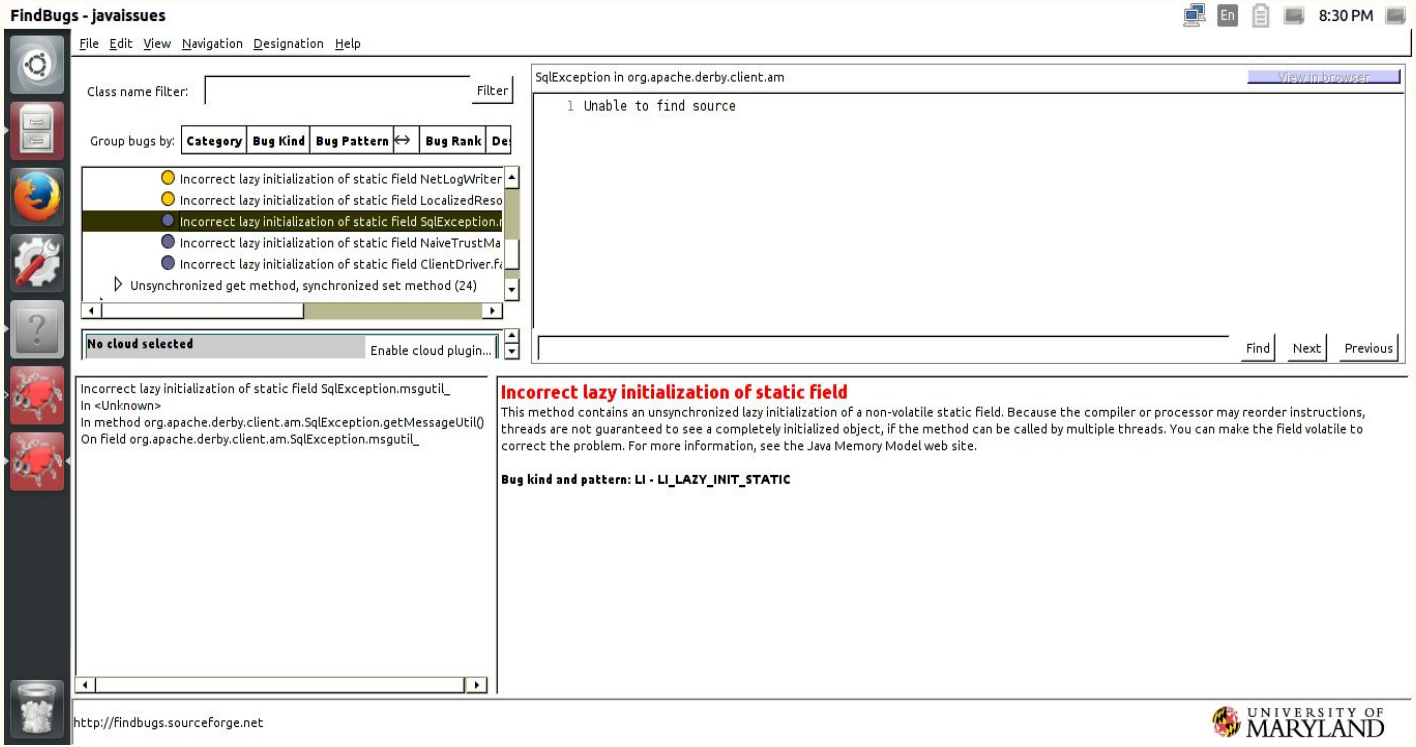
CWE-89: Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection')

- SQL injection used to modify a web site to serve malicious code
- Can bypass the requirement to only return items owned by authenticated user
- Shell command execution in MS SQL

http://cwe.mitre.org/data/definitions/89.html

```
'; exec master..xp_cmdshell 'dir' --
```

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



CWEs: How to Exploit

CWE-545: Use of Dynamic Class Loading

- Dynamically loaded code has the potential to be malicious.
- When it can be introduced in the architecture and design and implementation.
- Avoid the use of class loading as it complicates code analysis.

https://cwe.mitre.org/data/definitions/545.html

Example of CWE-545 Dynamic Class Loading

JD-GUI

File Edit Navigation Search Help

dynamic-class-loading-example-code.zip

- dynamic-class-loading-example-code
  - src/main
    - java/com/jcg
      - ClassLoaderTest.java
      - JavaClassLoader.java
      - MyClass.java
        - MyClass
    - resources
  - pom.xml

Search

Search string (* = any string, ? = any character):

*cl

Search For: ☐ Type ☐ Constructor ☑ String Constant ☐ Field ☐ Method

Limit To: ☑ Declarations ☑ References

4 matching entries:

- dynamic-class-loading-example-code.zip
  - dynamic-class-loading-example-code
    - src/main
      - java/com/jcg
        - JavaClassLoader.java
        - MyClass.java
      - resources
      - pom.xml

Open Cancel

ClassLoaderTest.java | MyClass.java | JavaClassLoader.java

```java
import java.lang.reflect.Constructor;
import java.lang.reflect.Method;

/**
 * @author ashraf
 *
 */
public class JavaClassLoader extends ClassLoader {

    public void invokeClassMethod(String classBinName, String methodName){

        try {

            // Create a new JavaClassLoader
            ClassLoader classLoader = this.getClass().getClassLoader();

            // Load the target class using its binary name
          Class<?> loadedMyClass = classLoader.loadClass(classBinName);

          System.out.println("Loaded class name: " + loadedMyClass.getName());

          // Create a new instance from the loaded class
          Constructor<?> constructor = loadedMyClass.getConstructor();
          Object myClassObject = constructor.newInstance();

          // Getting the target method from the loaded class and invoke it using its name
          Method method = loadedMyClass.getMethod(methodName);
          System.out.println("Invoked method name: " + method.getName());
          method.invoke(myClassObject);

        } catch (ClassNotFoundException e) {
            e.printStackTrace();
        } catch (Exception e) {
            e.printStackTrace();
        }

    }
}
```

CWEs: How to Exploit

CWE-502: Deserialization of Untrusted Data

- This exploit deserializes untrusted data without sufficiently verifying that the resulting data will be valid.
- Exploit can be introduced during Architecture, Design and Implementation
- To avoid security issues with the data a hash-based message authentication code could be used to ensure that data has not been modified.

https://cwe.mitre.org/data/definitions/502.html

Bad Code

Example Language: **Java**

```
try {
  File file = new File("object.obj");
  ObjectInputStream in = new ObjectInputStream(new FileInputStream(file));
  javax.swing.JButton button = (javax.swing.JButton) in.readObject();
  in.close();
}
```

Good Code

Example Language: **Java**

```
private final void readObject(ObjectInputStream in) throws java.io.IOException {
throw new java.io.IOException("Cannot be deserialized"); }
```

Example of CWE - 502 Deserialization of Untrusted Data

FindBugs - example

5:06 PM

File Edit View Navigation Designation Help

Class name filter: Filter

Group bugs by: Category | Bug Kind | Bug Pattern | ↔ | Bug Rank | De

- Malicious code vulnerability (1)
- Performance (1)
  - Dubious method used (1)
    - Method invokes inefficient new String(String) constructor (1
      - edu.umsl.Student.setID(String) invokes inefficient new
- Dodgy code (7)

No cloud selected — Enable cloud plugin...

View in browser

Find Next Previous

GradStudent is Serializable; consider declaring a serialVersionUID
At GradStudent.java:[lines 12-52]
In class edu.umsl.GradStudent

Class is Serializable, but doesn't define serialVersionUID

This class implements the `Serializable` interface, but does not define a `serialVersionUID` field. A change as simple as adding a reference to a .class object will add synthetic fields to the class, which will unfortunately change the implicit serialVersionUID (e.g., adding a reference to `String.class` will generate a static field `class$java$lang$String`). Also, different source code to bytecode compilers may use different naming conventions for synthetic variables generated for references to class objects or inner classes. To ensure interoperability of Serializable across versions, consider adding an explicit serialVersionUID.

Bug kind and pattern: SnVI - SE_NO_SERIALVERSIONID

http://findbugs.sourceforge.net

UNIVERSITY OF MARYLAND

Conclusion

- Everything has a vulnerability, weakness, or bug that can be exploited.
- BE CAREFUL WITH WHAT YOU DO!!!!!!!!!!
- Questions?