# Finding Security Errors in Java Applications Using Lightweight Static Analysis

Benjamin Livshits
Computer Science Lab
Stanford University

# Vulnerability Research Focus

- Static analysis for vulnerability detection
- Until recently, a large portion of server-side software was written in C/C++
- Vulnerabilities come from poor language and API design:
  - Buffer overruns
  - Format string violations
- More profound:
  - Time-of-check-time-of-use errors (TOCTOU)

# Security Errors in Java are Emerging

- Situation is changing…
- More and more Web-based applications are written in Java
- Web-based applications are good vulnerability targets
- New categories of errors in this domain

SQL Injections

HTTP response splitting

LDAP injection

Cross-site scripting

Bad session stores

Forceful browsing

# Finding Errors with Static Analysis

- Our approach:
  - Static Analysis has been proven useful for finding security errors in C programs
  - Apply to Java to find new categories of errors
- What we did:
  - Created user-friendly code analysis tools
  - Based on Eclipse, an open-source Java IDE
  - Easy to run on your own code
  - Focused on two types of errors so far
    - Bad session stores
    - SQL injections
  - We look at these two error patterns next…

# Focus on Two Error Patterns

## Bad session store

```
Object o = …
HttpSession s = …
s.setAttribute("name", o);
```

- A common pattern in servlets leading to errors
- HttpSessions need to be saved to disk
- Object *o* must implement `java.io.Serializable`
- Bad API design
- Can lead to crashes and DOS attacks

## SQL injection

```
String query =
   request.getParameter("name");
java.sql.Statement stmt = …
stmt.executeQuery(query);
```

- Unchecked input passed to backend database
- Carefully crafted input containing SQL will be interpreted by database
- Can be used by the malicious user to
  - read unauthorized info,
  - delete data,
  - even execute commands,
  - etc.

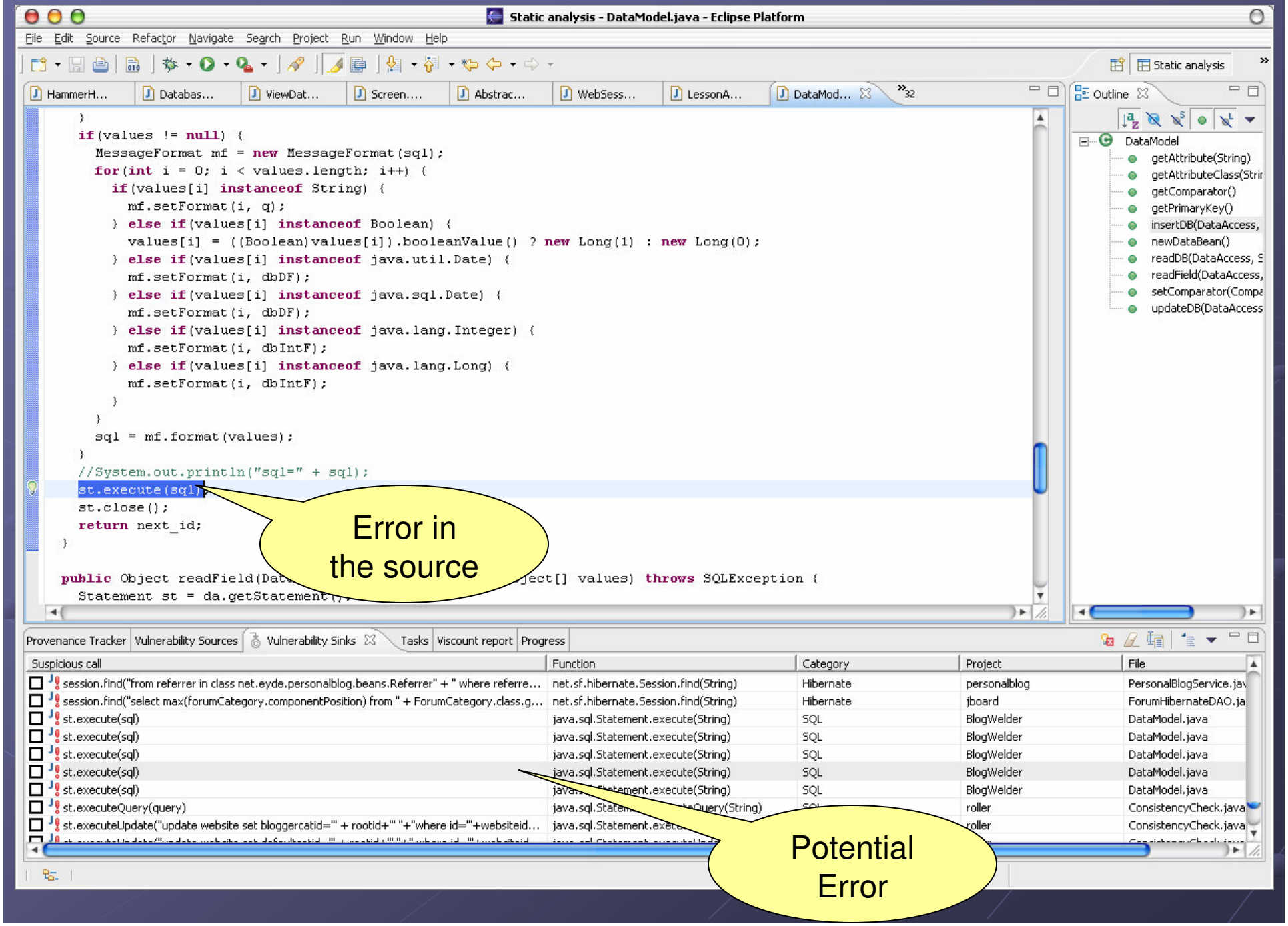
# Our Tools…

Bad session stores

- Look at the type of the 2nd argument of `setAttribute`:
  - `setAttribute(…, expr);`
- Do a type check for **`expr`** that don't implement `java.io.Serializable`
- Report errors

SQL Injections

- Identify all sources of user information
- Identify all sinks where sensitive data can flow
- Filter out sinks that take constant strings
- Help to follow data from *sources* to *sinks*
- Report errors


Static analysis - DataModel.java - Eclipse Platform
File Edit Source Refactor Navigate Search Project Run Window Help
Static analysis
HammerH... Databas... ViewDat... Screen.... Abstrac... WebSess... LessonA... DataMod...
Outline
DataModel
getAttribute(String)
getAttributeClass(Strir
getComparator()
getPrimaryKey()
insertDB(DataAccess,
newDataBean()
readDB(DataAccess, S
readField(DataAccess,
setComparator(Compa
updateDB(DataAccess
}
if(values != null) {
MessageFormat mf = new MessageFormat(sql);
for(int i = 0; i < values.length; i++) {
if(values[i] instanceof String) {
mf.setFormat(i, q);
} else if(values[i] instanceof Boolean) {
values[i] = ((Boolean)values[i]).booleanValue() ? new Long(1) : new Long(0);
} else if(values[i] instanceof java.util.Date) {
mf.setFormat(i, dbDF);
} else if(values[i] instanceof java.sql.Date) {
mf.setFormat(i, dbDF);
} else if(values[i] instanceof java.lang.Integer) {
mf.setFormat(i, dbIntF);
} else if(values[i] instanceof java.lang.Long) {
mf.setFormat(i, dbIntF);
}
}
sql = mf.format(values);
}
//System.out.println("sql=" + sql);
st.execute(sql);
st.close();
return next_id;
}
public Object readField(Dat ject[] values) throws SQLException {
Statement st = da.getStatement();
Error in the source
Provenance Tracker Vulnerability Sources Vulnerability Sinks Tasks Viscount report Progress
Suspicious call
Function
Category
Project
File
session.find("from referrer in class net.eyde.personalblog.beans.Referrer" + " where referre...
net.sf.hibernate.Session.find(String)
Hibernate
personalblog
PersonalBlogService.jav
session.find("select max(forumCategory.componentPosition) from " + ForumCategory.class.g...
net.sf.hibernate.Session.find(String)
Hibernate
jboard
ForumHibernateDAO.ja
st.execute(sql)
java.sql.Statement.execute(String)
SQL
BlogWelder
DataModel.java
st.executeQuery(query)
roller
ConsistencyCheck.java
st.executeUpdate("update website set bloggercatid='" + rootid+"' "+"where id='"+websiteid...
roller
ConsistencyCheck.java
Potential Error

# Benchmarks

- 10 Web-based applications
- Widely deployed and vulnerable to attacks
- Most blogging tools
- Quite large – 10s of KLOC
- Rely on *very large* J2EE libs

| Benchmark | LOC | Classes |
|---|---|---|
| mapleblog | 2,156 | 36 |
| personalblog | 2,317 | 38 |
| blueblog | 4,142 | 38 |
| blogwelder | 4,901 | 33 |
| javablog | 5,184 | 79 |
| snipsnap | 9,671 | 1,331 |
| blojsom | 14,382 | 30 |
| jboard | 17,368 | 138 |
| pebble | 30,319 | 169 |
| roller | 47,044 | 267 |
| | | |
| **Total** | **137 K** | **2,159** |

# Results for Bad Session Stores

- Found 14 errors
- 8 false posititives
- 37% false pos rate
- Why false positives?
  - Declared types are too wide
  - Can improve with better type info from pointer analysis

| Benchmark | All | Bad | Errors | False pos. |
|---|---|---|---|---|
| mapleblog | 5 | 5 | 3 | 2 |
| personalblog | 2 | 0 | 0 | 0 |
| blueblog | 0 | 0 | 0 | 0 |
| blogwelder | 3 | 3 | 3 | 0 |
| javablog | 10 | 0 | 0 | 0 |
| snipsnap | 28 | 12 | 7 | 5 |
| blojsom | 0 | 0 | 0 | 0 |
| jboard | 1 | 0 | 0 | 0 |
| pebble | 2 | 1 | 1 | 0 |
| roller | 24 | 1 | 0 | 1 |
| | | | | |
| **Total** | **75** | **22** | **14** | **8** |

# Results for SQL Injections

- Found 6 errors
- Can find "low-hanging" errors
- Easy when sources and sinks are "close"
- Often they are very far apart
- **Many require more elaborate analysis**

| | | All | Unsafe | |
|---|---|---|---|---|
| **Benchmark** | **Sources** | **sinks** | **sinks** | **Errors** |
| mapleblog | 8 | 16 | 16 | 1 |
| personalblog | 29 | 35 | 27 | 1 |
| blueblog | 6 | 1 | 1 | 0 |
| blogwelder | 115 | 24 | 24 | 0 |
| javablog | 12 | 42 | 38 | 0 |
| snipsnap | 195 | 33 | 33 | 1 |
| blojsom | 12 | 1 | 1 | 0 |
| jboard | 3 | 18 | 17 | 3 |
| pebble | 109 | 1 | 1 | 0 |
| roller | 81 | 45 | 30 | 0 |
| | | | | |
| Total | 560 | 216 | 188 | 6 |

# Summary

- Created lightweight interactive tools for finding security errors in Java
- Found a total of 20 errors
- However, there are
  - false positives and
  - “unknowns” – potential errors our tools can’t address
- Conclusion:
  - Our tools are good for finding simpler errors
  - Hard errors often require a stronger analysis of data propagation
  - Working on a pointer analysis-based approach