Privacy and Security
Constraints for Code Contributions

Rodrigo Andrade
Advisor: Paulo Borba
Collaborative software development

Task A

Task B

Task C

code contribution

Some contributions might carelessly or maliciously introduce privacy and security violations
Sensitive information flowing to such code contributions

password
email
location
- Scenarios-
First scenario
Checking developers contributions

Naive developer

Gitblit open issues
Task: improving the Gitblit code that deals with Java Servlets

```java
class RootPage {
    void loginUser(UserModel user) {
        if (user != null) { ... 
        auth().setCookie(req, resp, user);
    }
    } ... 
}
```
Violating privacy properties

Sensitive user information unnecessarily flows to Cookie Mechanism code contribution
SHA-1 algorithm
https://hashkiller.co.uk/sha1-decrypter.aspx
Manually reviewing source code

+142 additions
- 142 deletions

This task requires understanding:
- Gitblit source code
- Gitblit sensitive information and their flows
JOANA IFC analysis

Manually analyze 408 bytecode instructions to label:

+ auth().setCookie(req, resp, user);

We need to manually analyze and label 141 additions plus sensitive information!
Second scenario
Checking the most recent Blojsom version

Logger
File

Naive developer
Leaking password

```java
boolean isAuthorized(HttpServletRequest req) {
    boolean result = false;
    ...
    password = realmAuth.substring(pos + 1);
    result = _blog.checkAuthorization(username, password);
    if (!result) {
        _logger.info("Unable to authenticate user [" + username + "] w/password [" + password + "]");
    }
    return result;
}
```
Violating privacy properties

Sensitive user information

password

mistakenly flows to

Logger code
Password leaks to log files

Unable to authenticate user [ admin ] w/password [ 123 ]
Manually reviewing source code

• Most recent Blojsom version
  • 58 Logger references
  • 340 Logger methods calls
• Who introduced the violation?
• Where?
JOANA IFC analysis

Manually label Logger methods

Manually label sensitive information
Third scenario
Checking ScribeJava contributions
### Class OAuth1AccessToken

```java
class OAuth1AccessToken {
    + String toString() {
        + return "OAuthAccessToken{" +
        + "oauthtoken=" + getToken() +
        + ",oauthtokensecret=" + getTokenSecret() + "}";
    }

    ...
}
```
Violating privacy properties

Sensitive user information mistakenly flows to Token secret

Code contribution
Manually reviewing contribution

- Code contribution
  - 33 files
  - 385 additions
  - 174 deletions
- Who introduced the violation?
- Where?
JOANA IFC analysis

Manually label Logger methods

Manually label sensitive information
Fourth scenario
Checking developers' tasks

Untrustworthy developer
Task: Improving authentication mechanisms

- boolean isPub = "true".equals(getProperty("isPublic"));
- String token = isPub ? null : TokenCookie.getToken();

...
Violating privacy properties

Sensitive user information

mistakenly flows to

Untrustworthy developer code contribution
Private user spreadsheets might be accessed without proper permission
Manually reviewing source code

• 813 commits
  • 1,027,084 additions
  • 589,028 deletions
JOANA IFC Analysis

• Manually
  • Labeling 813 commit code
  • Compile 813 versions
Manually reviewing code contribution

• Necessary in many cases
• Expensive, error-prone, and time-consuming
Information flow control static analysis tools

Restricted to work on certain technical domains

OR

Require manual specification of the sensitive information and code contribution
Lack of generality

These tools implicitly specify privacy and security constraints relevant only to their technical domain.
Time-consuming and error-prone

Require developers to manually specify sensitive information and code contributions
Time-consuming and error-prone

Require developers to manually specify sensitive information and code contributions
Time-consuming and error-prone

Require developers to 
manually specify sensitive information and code contributions
- Salvum -
Salvum allows developers to declaratively express constraints to protect confidentiality and integrity of sensitive information from specific code contributions.
Backward

Salvum

constraint A  ✓
constraint B  ❌

A \textbf{no}set \texttt{User\{login\}}

\texttt{User\{login\} noflow B}

Specialist
Forward

Developer

Specialist

A

B

Salvum

A \textbf{nosed} \texttt{User\{login\}}

\texttt{User\{login\} nostream \texttt{B}}

constraint A

\checkmark

constraint B

\times
General idea of our tool – First step

Specialist

Software project

password location e-mail...

Code contributions
General idea of our tool – Second step

Specialist writes

User{password} noflow A

B noset User{password}
General idea of our tool – Third step

Enforcing constraints

Source code
General idea of our tool – Fourth step

constraint A ✓
constraint B ×

Illegal flow from sensitive information to a line of a class at commit hash
General idea of our tool – Fifth step

Illegal flow from sensitive information to a line of a class at commit hash

Specialist
Language Specification
Constructs

Sensitive information ➡️ noflow ➡️ Code contribution

Code contribution ➡️ noset ➡️ Sensitive Information

Information ➡️ flow ➡️ Code contribution

Code contribution ➡️ set ➡️ Information
Code contributions - Explicitly

{c | c.message.contains("issueID")}

{5odle482}

{c | c.author("AuthorID")}
Code contributions - Implicitly

Contribution $\equiv$ Set of commits we want to analyze
Code contributions - Operations listing

```java
{Logger.error(), Logger.info(), Logger.warn()}

{Cookie.setCookie(), File.write()}
```
Examples
Specifying information that cannot flow to code contributions

UserModel{password, locality} noflow Contribution

sensitive information
code contribution

commits
Specifying information that cannot flow to Logger methods

sensitive information

UserModel{password,locality} noflow Log

where Log = {Logger.error(), Logger.info()}

Dangerous logger method
Specifying information that cannot flow to Bob commits

**UserModel**: \{password, locality\} \textbf{noflow} UntrustworthyBob

\textbf{where} UntrustworthyBob = \{c \mid c\text{.author("Bob")}\}

**Sensitive information**                      **Code contribution**

![Diagram of code contribution and sensitive information flow]

```java
import java.util.concurrent.atomic.AtomicInteger;
import java.util.regex.Pattern;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
```
Specifying information that cannot be altered by Bob commits

code contribution

UntrustworthyBob \textbf{noiset} UserModel\{password, locality\}

\textbf{where} UntrustworthyBob = \{c \mid c\text{.author\("Bob")}\}

sensitive information

commits
Specifying information that cannot flow to a certain commit

Sensitive information: 

UserModel `{password, email, location}` noflow ServletsTask

where ServletsTask = `{efdb2b3d0}`

Code contribution:
noflow and noset are permissive in nature
Specifying information that can **flow** to code contributions

**public information**

`UserModel {login, nickname, countrCode} flow PullRequest`

**code contribution**

`where PullRequest = {c | c.message.contains("#921")}`
Specifying information that can be altered by code contributions

```
UpdateUserTask set UserModel {login, nickname, countrCode}
where UpdateUserTask = {c | c.message.contains("task87")}
```
Constraints might be too large

UserModel{login, nickname, countryCode},
HttpServletRequest{httpRequest, httpSession},
AuthenticationType{authType}, HttpServletRequest{httpResponse},
GitBlitWebSession{session}
...

**flow** PullRequest

**where** PullRequest = \{c | c.message.contains("#921")\}
Positive and negative primitives

- noflow
  noset
  Specified information cannot flow or be changed by code contribution

+ flow
  set
  Only specified information can flow or be changed by code contribution
Grammar

<contribution_expression> ::= <contribution_spec> |
<contribution_expression> "&&" <contribution_expression> |
<contribution_expression> "||" <contribution_expression> |
"!" <contribution_expression>

{c | c.message.contains("task87") && c | c.author("Bob")}
Salvum implementation: automatically enforcing constraints
Steps
1- Preprocessing – operations listing

User{password} noflow A

Constraints

Source code

output mapping

[Main: [11, 12]], [Repository: [6, 16]]
1 - Preprocessing – Code contributions

output mapping

[RootPage: [27, 28, 60, 71]], [Mailing: [64, 65]]
2 – Generating SDG

Compiled classes (bytecode)

Entry-points

System Dependence Graph
3- Labeling: confidentiality

```java
class UserModel {
    String password;
    String email;
    String location;
    ...
}
```

```java
class RootPage {
    void loginUser(UserModel user) {
        if (user != null)
            +auth().setCookie(req, resp, user);
    }
}
```
3 - Labeling: integrity

```java
class UserModel {
    String password;
    String email;
    String location;
    ...
}
```

```java
class RootPage {
    void loginUser( UserModel user) {
        if (user != null)
            +auth().setCookie(req, resp, user);
    }
}
```

Low label: Integer, String

High label: None
4 - IFC analysis

High label

Low label

Low label

Low label

Low label
5 - Process Results

constraint A ✔
constraint B ✗

Illegal flow from UserModel.password, UserModel.email, and UserModel.location to RootPage:27 at commit hsf354ks
- Evaluation -
Goal, Question, and Metric (GQM)

Evaluate Salvum regarding privacy and security violations for code contributions

Detecting violations:
- NVW
- NV
- NvC
- NrC

Reducing effort:
- SA
- PA
JOANA precision, recall, and accuracy

How well?

TP  FN  FP  TN
Q1 – Can Salvum detect code contribution violations of sensitive information?

Number of violations warnings (NVW)

- constraint A
- constraint B

Number of violations (NV)

- constraint A
- constraint B
Assessing **highly active and well-supported** software projects

<table>
<thead>
<tr>
<th>Project</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gitblit</td>
</tr>
<tr>
<td>Open Refine</td>
</tr>
<tr>
<td>Voldemort</td>
</tr>
<tr>
<td>ScribeJava</td>
</tr>
<tr>
<td>Solo</td>
</tr>
<tr>
<td>HikariCP</td>
</tr>
<tr>
<td>Apache Kafka</td>
</tr>
<tr>
<td>Teammates</td>
</tr>
<tr>
<td>Crawler4J</td>
</tr>
</tbody>
</table>
Assessment procedures

1. Define constraints
2. Run our tool
3. Report issue
4. Check the rate of acceptance
Gitblit – Policy

User password, locality, token, access information, and permission cannot flow to code contributions
Gitblit - Constraints

ChangePasswordPage {password, confirmPassword} ... noflow Contribution
## Answering Q1

<table>
<thead>
<tr>
<th>Project</th>
<th>Versions</th>
<th>NVW</th>
<th>NV</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gitblit</td>
<td>49</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>Open Refine</td>
<td>10</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Voldemort</td>
<td>52</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>ScribeJava</td>
<td>27</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>Solo</td>
<td>10</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>HikariCP</td>
<td>10</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Apache Kafka</td>
<td>43</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Teammates</td>
<td>10</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Crawler4J</td>
<td>10</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>221</strong></td>
<td><strong>6</strong></td>
<td><strong>5</strong></td>
</tr>
</tbody>
</table>
Gitblit violation example

Illegal flow from UserModel.password to RootPage.loginUser() at line 284 in commit efdb2b3d

```java
app().authentication().setCookie(request, response, user);
```
Assessing **low active and poorly-supported** software projects

<table>
<thead>
<tr>
<th>Project</th>
</tr>
</thead>
<tbody>
<tr>
<td>Blojsom</td>
</tr>
<tr>
<td>PersonalBlog</td>
</tr>
<tr>
<td>GridSphere</td>
</tr>
<tr>
<td>SnipSnap</td>
</tr>
<tr>
<td>Lutece</td>
</tr>
</tbody>
</table>
Answering Q1

<table>
<thead>
<tr>
<th>Project</th>
<th>NVW</th>
<th>NV</th>
</tr>
</thead>
<tbody>
<tr>
<td>Blojsom</td>
<td>34</td>
<td>33</td>
</tr>
<tr>
<td>Personalblog</td>
<td>44</td>
<td>43</td>
</tr>
<tr>
<td>GridSphere</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>SnipSnap</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Lutece</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>=</td>
<td>86</td>
<td>80</td>
</tr>
</tbody>
</table>
Salvum can detect proper violations of sensitive information and code contributions

Why just a few violations regarding well-supported projects?
Q1.1 – Do developers solve violations before code contribution merging on Github?

Number of violations before merging code contributions (NvC)
Answering Q1.1

We analyzed discussions of 198 pull requests

The NvC metric is zero
Q1.2 – Are unmerged code contributions on Github related to violations of sensitive information?

Number of unmerged code contributions containing violations (NrC)
Answering Q1.2

We analyzed code contributions of 45 pull requests

The NrC metric is zero
Q2 – Can Salvum reduce the effort to find violations of specified constraints in a set of code contributions?

Source lines of code to analyze (SA)

Project versions to analyze (PA)
Answering Q2

<table>
<thead>
<tr>
<th>Project</th>
<th>PA without Salvum</th>
<th>PA with Salvum</th>
<th>Percentage of total</th>
<th>SA without Salvum</th>
<th>SA with Salvum</th>
<th>Percentage of total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gitblit</td>
<td>49</td>
<td>3</td>
<td>6.12%</td>
<td>6012</td>
<td>32</td>
<td>0.53%</td>
</tr>
<tr>
<td>Open Refine</td>
<td>10</td>
<td>0</td>
<td>0%</td>
<td>578</td>
<td>0</td>
<td>0%</td>
</tr>
<tr>
<td>Voldemort</td>
<td>52</td>
<td>0</td>
<td>0%</td>
<td>5040</td>
<td>0</td>
<td>0%</td>
</tr>
<tr>
<td>ScribeJava</td>
<td>27</td>
<td>1</td>
<td>3.7%</td>
<td>4496</td>
<td>4</td>
<td>0.08%</td>
</tr>
<tr>
<td>Solo</td>
<td>10</td>
<td>0</td>
<td>0%</td>
<td>952</td>
<td>0</td>
<td>0%</td>
</tr>
<tr>
<td>HikariCP</td>
<td>10</td>
<td>0</td>
<td>0%</td>
<td>216</td>
<td>0</td>
<td>0%</td>
</tr>
<tr>
<td>Apache Kafka</td>
<td>43</td>
<td>0</td>
<td>0%</td>
<td>3402</td>
<td>0</td>
<td>0%</td>
</tr>
<tr>
<td>Teammates</td>
<td>10</td>
<td>0</td>
<td>0%</td>
<td>1863</td>
<td>0</td>
<td>0%</td>
</tr>
<tr>
<td>Crawler4J</td>
<td>10</td>
<td>1</td>
<td>10%</td>
<td>494</td>
<td>22</td>
<td>4.45%</td>
</tr>
<tr>
<td>Total</td>
<td>221</td>
<td>5</td>
<td>-</td>
<td>23053</td>
<td>58</td>
<td>-</td>
</tr>
</tbody>
</table>
Salvum reduces the effort to find violations of the specified constraints
Q3 – How does JOANA perform when being applied to IFC issues regarding precision, recall, and accuracy of its analysis?

- Number of true-positives (TP)
- Number of false-negatives (FN)
- Number of false-positives (FP)
- Number of true-negatives (TN)
JOANA Evaluation

Stanford SecuriBench Micro

JOANA
### Answering Q3

<table>
<thead>
<tr>
<th>Test Case Group</th>
<th>TP</th>
<th>TP + FN</th>
<th>FP</th>
<th>FN</th>
<th>TN</th>
<th>Recall</th>
<th>Precision</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aliasing</td>
<td>12</td>
<td>12</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>100%</td>
<td>100%</td>
<td>100%</td>
</tr>
<tr>
<td>Arrays</td>
<td>9</td>
<td>9</td>
<td>5</td>
<td>0</td>
<td>1</td>
<td>100%</td>
<td>64%</td>
<td>62%</td>
</tr>
<tr>
<td>Collections</td>
<td>14</td>
<td>14</td>
<td>6</td>
<td>0</td>
<td>1</td>
<td>100%</td>
<td>70%</td>
<td>71%</td>
</tr>
<tr>
<td>Datastructure</td>
<td>5</td>
<td>5</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>100%</td>
<td>62%</td>
<td>62%</td>
</tr>
<tr>
<td>Factories</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>100%</td>
<td>50%</td>
<td>50%</td>
</tr>
<tr>
<td>Inter</td>
<td>16</td>
<td>16</td>
<td>8</td>
<td>0</td>
<td>3</td>
<td>100%</td>
<td>66%</td>
<td>70%</td>
</tr>
<tr>
<td>Pred</td>
<td>5</td>
<td>5</td>
<td>3</td>
<td>0</td>
<td>1</td>
<td>100%</td>
<td>62%</td>
<td>66%</td>
</tr>
<tr>
<td>Reflection</td>
<td>3</td>
<td>4</td>
<td>6</td>
<td>1</td>
<td>0</td>
<td>75%</td>
<td>33%</td>
<td>30%</td>
</tr>
<tr>
<td>Session</td>
<td>3</td>
<td>3</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>100%</td>
<td>75%</td>
<td>75%</td>
</tr>
<tr>
<td>Strong Updates</td>
<td>1</td>
<td>1</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>100%</td>
<td>20%</td>
<td>20%</td>
</tr>
</tbody>
</table>
Example of Strong update Test Group

```java
class Outter{
    class Widget {
        String value = null;
    }

    void m() {
        String name = "secret";
        Widget w = new Widget();
        w.value = name;
        w.value = "abc";
        PrintWriter writer = resp.getWriter();
        writer.println(w.value); /* OK */
    }
}
```
Threats to validity

Metrics

Manual reviewing pull requests

Constraints

Project versions

Violations
Related Work

Manual code review

OWASP

CWE
Related work

Automated Analysis
Related work

Language-based Information flow

Joe-E: A Security-Oriented Subset of Java
Adrian Mettler  David Wagner  Tyler Close
University of California, Berkeley  waterken.org  tyler.close@gmail.com

A Language for Automatically Enforcing Privacy Policies
Jean Yang  Kuat Yessenov  Armando Solar-Lezama
MIT CSAIL  {jeanyang, kuat, asolar} @csail.mit.edu

Access Control in Feature-Oriented Programming
Sven Apel, Sergiy Kolesnikov, Jörg Liebig, Christian Kästner, Martin Kuhlemann, Thomas Leich

Exploring and Enforcing Security Guarantees via Program Dependence Graphs
Andrew Johnson  Lucas Waye
MIT Lincoln Laboratory and Harvard University, USA  lwaye, sdmooe, chong@seas.harvard.edu

检查器框架
Future work

• Enforcing constraints for features
• Customizing SDGs
• Android Support
• Declassification
Conclusion

Collaborative software development:
- Task A
- Task B
- Task C

These contributions might carelessly or maliciously introduce privacy and security violations.

Task: implementing cookie mechanism
```java
class RootPage {
    void loginUser(UserModel user) {
        if (user != null) {
            auth().setCookie(req, resp, user);
        }
    }
}
```

The developer did not notice that `UserModel` specifies sensitive user information.

Specifying information that cannot flow to code contributions:
- Sensitive information
- Code contribution
- `UserModel(password, locality)` noflow Contribution

Goal, Question, and Metric (GQM):
- Evaluate `Salum` regarding privacy and security violations for code contributions
- Detecting violations
- Reducing effort

<table>
<thead>
<tr>
<th>Project</th>
<th>Versions</th>
<th>NVW</th>
<th>NV</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gitblit</td>
<td>49</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Open Refine</td>
<td>10</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Voidemor1</td>
<td>52</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>ScribeJava</td>
<td>27</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Solo</td>
<td>10</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>HikariCP</td>
<td>10</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Apache Kafka</td>
<td>43</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Teammates</td>
<td>10</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Crawler4J</td>
<td>10</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Total</td>
<td>221</td>
<td>6</td>
<td>5</td>
</tr>
</tbody>
</table>

Answering Q1
Privacy and Security
Constraints for Code Contributions

Rodrigo Andrade
Advisor: Paulo Borba