F4F: Taint Analysis of Framework-based Web Applications

Manu Sridharan\textsuperscript{1} Shay Artzi\textsuperscript{1} Marco Pistoia\textsuperscript{1} Sal Guarnieri\textsuperscript{2} Omer Tripp\textsuperscript{2,3} Ryan Berg\textsuperscript{2}

\textsuperscript{1}IBM Research\textsuperscript{2} IBM Software Group\textsuperscript{3} Tel-Aviv University

OOPSLA 2011
Web Application Security
Web Application Security

T.J. Maxx theft believed largest hack ever

Sony Pictures hacked by Lulz Security, 1,000,000 passwords claimed stolen (update)

PBS Website Hacked With Fake News
Web Application Security

T.J. Maxx theft believed largest hack ever

Sony Pictures hacked by Lulz Security, 1,000,000 passwords claimed stolen (update)

PBS Website Hacked With Fake News

Lady Gaga's website hacked by SwagSec cyber attackers
Taint Analysis for Security

Untrusted data (request params)

Sensitive operation (DB query)

Key: accurate tracking of data flow
Web Application Frameworks

- Spring
- ASP.net
- RAILS
Web Application Frameworks

- Bad for taint analysis!
- Data flow via reflection, “interpreting” config files
Existing Approaches

- Only analyzing the code **not effective**
  - Analyze reflection: many false positives
    - Can’t see config info!
  - Ignore reflection: miss many issues
- Baked-in framework support **doesn’t scale**
  - Tens of frameworks for Java alone
  - Different versions, customizations, ...
A Framework for Frameworks (F4F)
A Framework for Frameworks (F4F)
A Framework for Frameworks (F4F)

config. info → Framework handlers

- Spring Handler
- ASP.NET Handler

application → Taint analysis → vulnerabilities
A Framework for Frameworks (F4F)

config. info

Framework handlers
- Spring Handler
- ASP.NET Handler
- ...

Web Application Framework Language (WAFL) specification

Taint analysis

application

vulnerabilities
A Framework for Frameworks (F4F)

- Spring Handler
- ASP.NET Handler
- Framework handlers

- Web Application Framework Language (WAFL) specification

- Taint analysis
  - application
  - config. info

- Easy to support new frameworks
  - vulnerabilities
A Framework for Frameworks (F4F)

- Spring Handler
- ASP.NET Handler

Framework handlers

config. info

application

Taint analysis

Web Application Framework Language (WAFL) specification

vulnerabilities

Easy to support new frameworks

Expressive, yet easy to integrate
A Framework for Frameworks (F4F)

Key to precision

config. info

Framework handlers

Spring Handler

ASP.NET Handler

Web Application Framework Language (WAFL) specification

application

Taint analysis

vulnerabilities

Easy to support new frameworks

Expressive, yet easy to integrate
Example: edit profile
<table>
<thead>
<tr>
<th>Code</th>
<th>Configuration</th>
</tr>
</thead>
</table>

Example: edit profile
Example: edit profile

Code

```java
// for user data
class UserForm {
    String firstName, lastName;
    // getters and setters...
}
```

Configuration
Example: edit profile

Code

// for user data
class UserForm {
    String firstName, lastName;
    // getters and setters...
}

// updates profile
class UserAction implements IAction {
    String exec(HttpRequest req, Object form) {
        UserForm uf = (UserForm) form;
        updateDB(uf);
        ...
    }
}

Configuration
Example: edit profile

Code

```java
// for user data
class UserForm {
    String firstName, lastName;
    // getters and setters...
}

// updates profile
class UserAction implements IAction {
    String exec(HttpRequest req, Object form) {
        UserForm uf = (UserForm) form;
        updateDB(uf);
        ...
    }
}
```

Configuration

```
<action url="/edit"
    type="UserAction"
    formtype="UserForm">
</action>
```

**In English**: When "/edit" is visited, create a UserForm object (reflection), set its fields using request data (reflection), and pass it to UserAction.exec() (reflection).
fun entrypoint UserAction_entry(request) {
    UserForm f = new UserForm();
    f.setFirstName(request.getParam("firstName"));
    f.setLastName(request.getParam("lastName"));
    (new UserAction()).exec(request, f);
}
fun entrypoint UserAction_entry(request) {
  UserForm f = new UserForm();
  f.setFirstName(request.getParam("firstName"));
  f.setLastName(request.getParam("lastName"));
  (new UserAction()).exec(request, f);
}

- **Simple structure**: no branches, loops, etc.
- Eases integration with analysis engine
- Taint analysis usually flow insensitive anyway
- Based on **both app code and config info**
<table>
<thead>
<tr>
<th>Code</th>
<th>Configuration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example: show profile</td>
<td></td>
</tr>
</tbody>
</table>
Example: show profile

Code

class UserAction {
    String exec(HttpRequest req,
                Object form) {
        UserForm uf = (UserForm) form;
        updateDB(uf);
        // store for use by view
        req.setAttribute("user",uf);
        // return view name
        return "showuser";
    }
}

Configuration
Example: show profile

**Code**

```java
class UserAction {
    String exec(HttpRequest req,
                Object form) {
        UserForm uf = (UserForm) form;
        updateDB(uf);
        // store for use by view
        req.setAttribute("user", uf);
        // return view name
        return "showuser";
    }
}
// view code in ShowUser.jsp
<% UserForm uf =
    (UserForm)req.getAttribute("user"); %>
<p><%= uf.getLastName(); %>,
    <%= uf.getFirstName(); %></p>
```
Example: show profile

Code

class UserAction {
    String exec(HttpRequest req, 
        Object form) {
        UserForm uf = (UserForm) form;
        updateDB(uf);
        // store for use by view
        req.setAttribute("user", uf);
        // return view name
        return "showuser";
    }
}

// view code in ShowUser.jsp
<% UserForm uf =
    (UserForm)req.getAttribute("user"); %>
<p><%= uf.getLastName(); %>,
   <%= uf.getFirstName(); %></p>

Configuration

<forward name="showuser"
    path="/ShowUser.jsp" />

In English: If exec() method returns "showuser", display "/ShowUser.jsp" (reflection).
WAFL: call replacements and globals
// model flow through “user” request attribute

global request_user;

replaceCall 'req.setAttribute("user", uf)' in exec() {
    request_user = uf;
}

replaceCall 'uf = req.getAttribute("user")' in ShowUser.jsp {
    uf = request_user;
}
WAFL: call replacements and globals

// model flow through “user” request attribute
global request_user;
replaceCall ‘req.setAttribute(“user”, uf)” in exec() {
    request_user = uf;
}
replaceCall ‘uf = req.getAttribute(“user”)’ in ShowUser.jsp {
    uf = request_user;
}
fun entrypoint UserAction_entry(request) {
    // model forward to (compiled) ShowUser.jsp
    ShowUser_jsp.render(request);
}
WAFL: call replacements and globals

```java
// model flow through “user” request attribute
global request_user;
replaceCall 'req.setAttribute("user", uf)' in exec() {
    request_user = uf;
}
replaceCall 'uf = req.getAttribute("user")' in ShowUser.jsp {
    uf = request_user;
}
fun entrypoint UserAction_entry(request) { ...
    // model forward to (compiled) ShowUser.jsp
    ShowUser_jsp.render(request);
}
```

- Globals help modeling flows across disparate modules
- Only replace calls
- Expressive enough in our experience
- Significantly eases integration
Building WAFL generators

genSynthMethod(<action> a, SynthMethod m):
  type := a.attr("type"), formtype := a.attr("formtype")
  // populate form object
  m.addLocal("f", formtype)
  for each setter method setProp() in formtype:
    m.add(‘f.setProp(request.getParam("prop"))’)
  // invoke action
  m.addLocal("a", type)
  m.add(‘a.exec(request,f)’)
  // handle forwards to views
  for each string s returned by type.exec():
    f := <forward> element with name s
    url := f.attr("path")
    m.add(invokeURLEntry(url))
Building WAFL generators

- Only lightweight analysis needed
  - class hierarchy, local data-flow analysis
  - little redundancy with taint analysis
- Others can build generators!
  - Sales engineer built EJB handler using API
Taint Analysis Integration

- Generate code for synthetic methods
- Pass on entrypoint list
- Call replacements via `modified dispatch`

```java
replaceCall 'req.setAttribute("user", uf)' in exec() {
  request_user = uf;
}
```

Method matching signature
```
static void repl1(Request r, String s, UserForm uf) {
  request_user = uf;
}
```

Replacement info
```
dispatch 'req.setAttribute("user", uf)'
in exec() to repl1()
```
Evaluation

- Nine framework-based web apps
- Frameworks: Struts, Spring, Tiles, EL
- Ran state-of-the-art taint analysis with and without F4F
2.10X (1.1X-14.9X) more issues with F4F
Exploitability

Most issues require inspection
Related Work

- Spec languages for “environment”, e.g.
  - effects of Java native methods
  - how OS kernel invokes drivers
- MERLIN (Livshits et al., PLDI’09)
- TAJ (Tripp et al., PLDI’09)
Conclusions

- F4F helps to tame web frameworks
  - Separate framework handlers, analyzing config files
  - Expressive, consumable spec language (WAFL)
- Future work: apply to other domains, DSL for WAFL generators