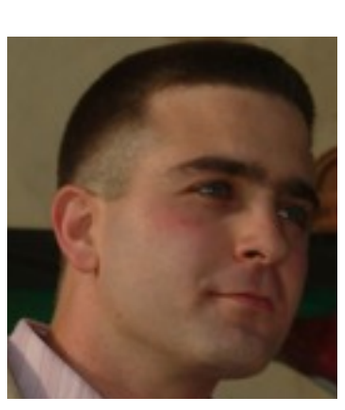
F4F: Taint Analysis of Framework-based Web Applications








Manu Sridharan[1] Shay Artzi[1] Marco Pistoia[1] Sal Guarnieri[2] Omer Tripp[2,3] Ryan Berg[2]

[1]IBM Research [2]IBM Software Group [3]Tel-Aviv University

OOPSLA 2011

Web Application Security

Web Application Security

T.J. maxx®

Security on msnbc.com

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

Security on msnbc.com

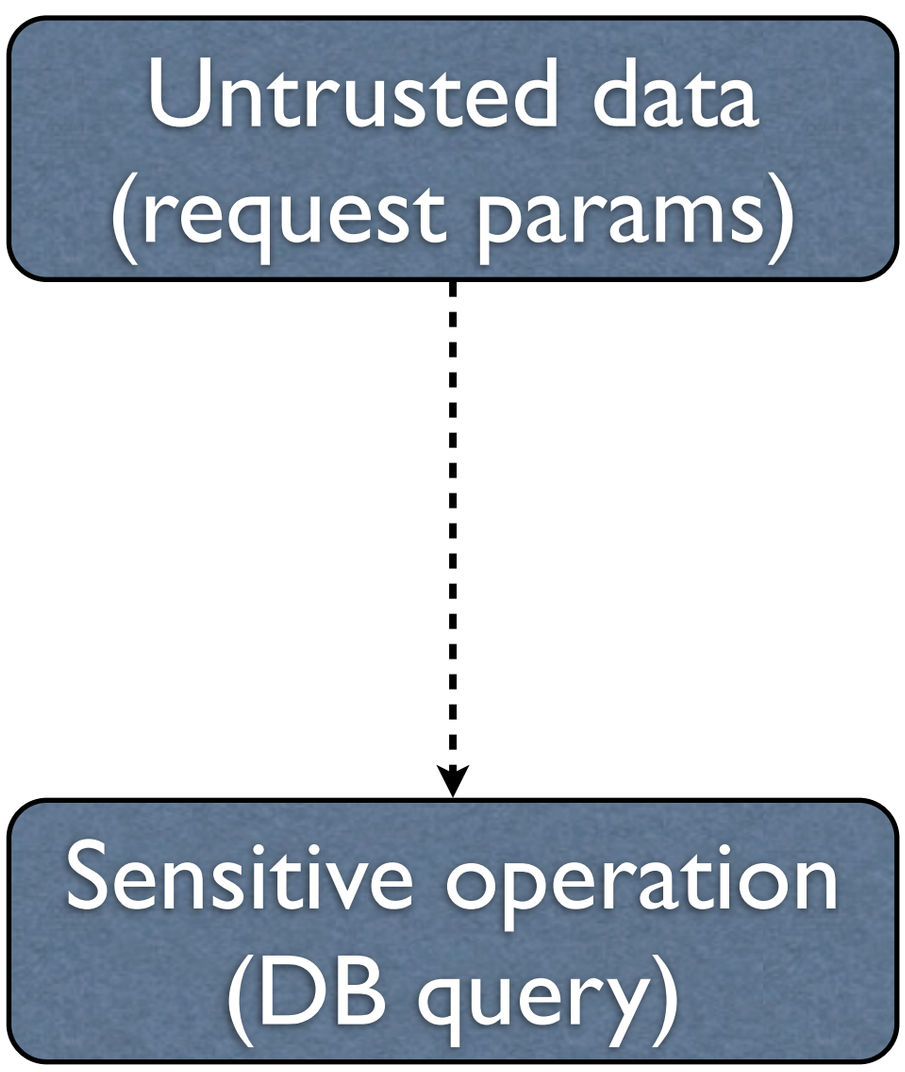
T.J. Maxx theft believed largest hack ever

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



Key: accurate tracking of data flow

Web Application Frameworks

Web Application Frameworks

- **Bad for taint analysis!**
  - Data flow via reflection, "interpreting" config files

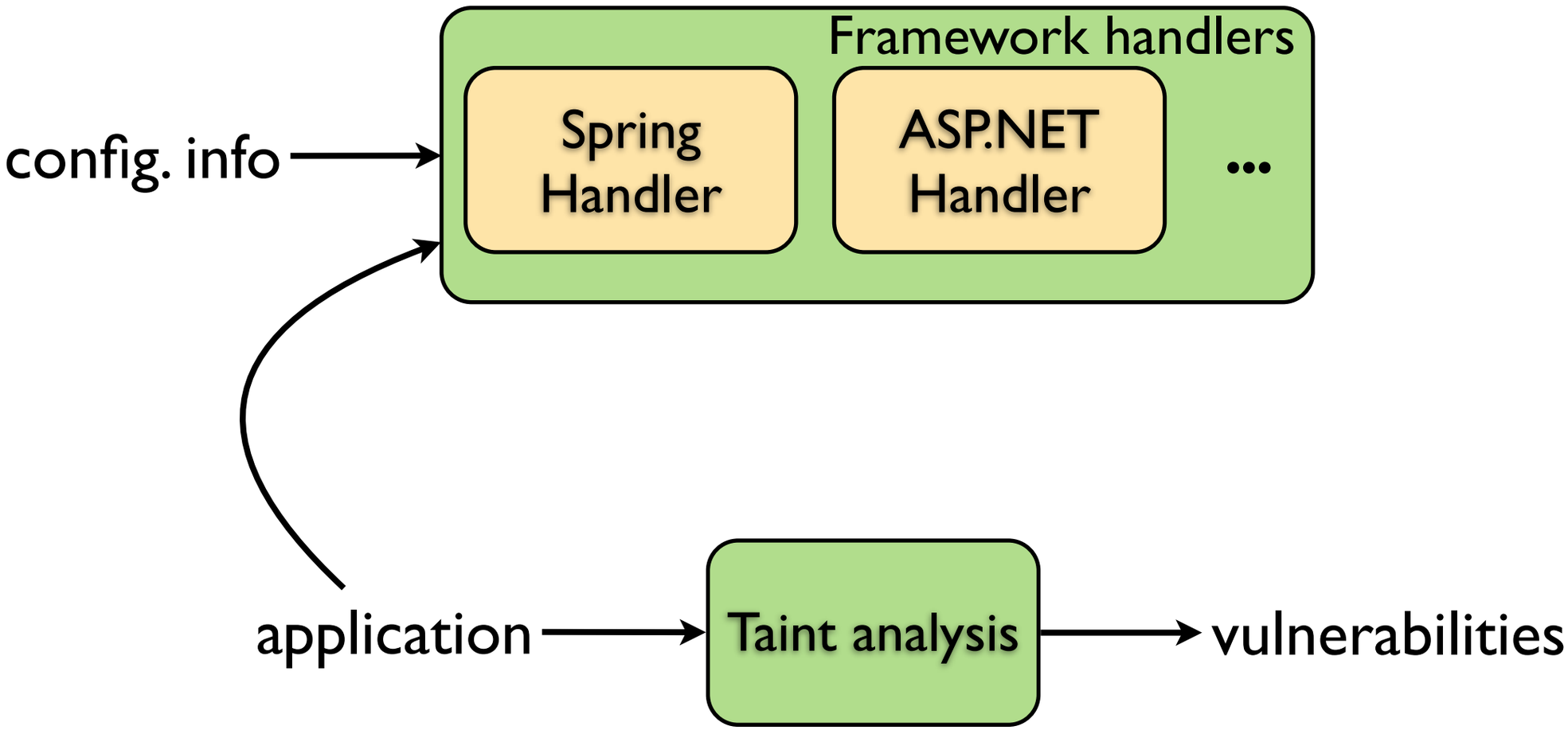
Existing Approaches

- 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, ...

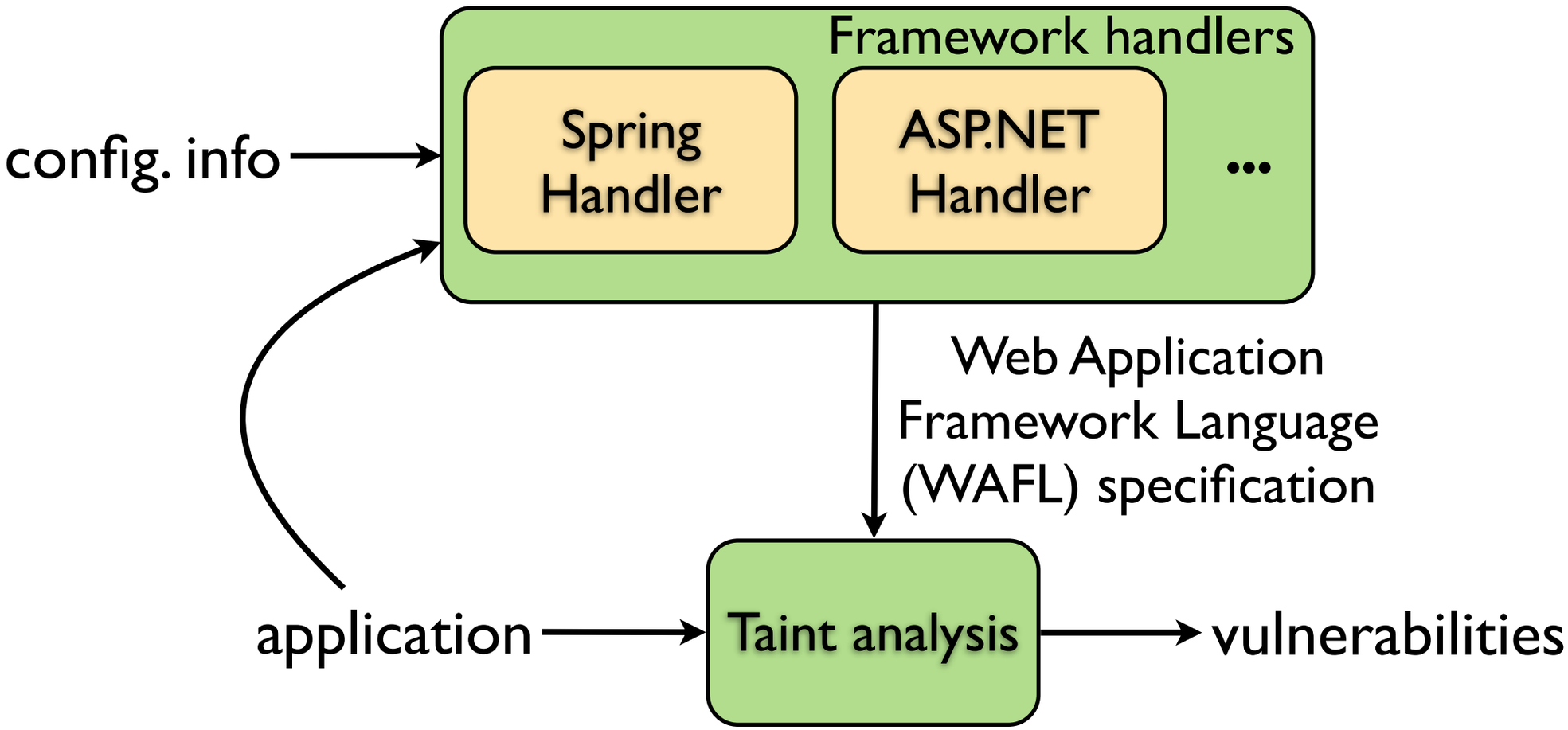
A Framework for Frameworks (F4F)

A Framework for Frameworks (F4F)

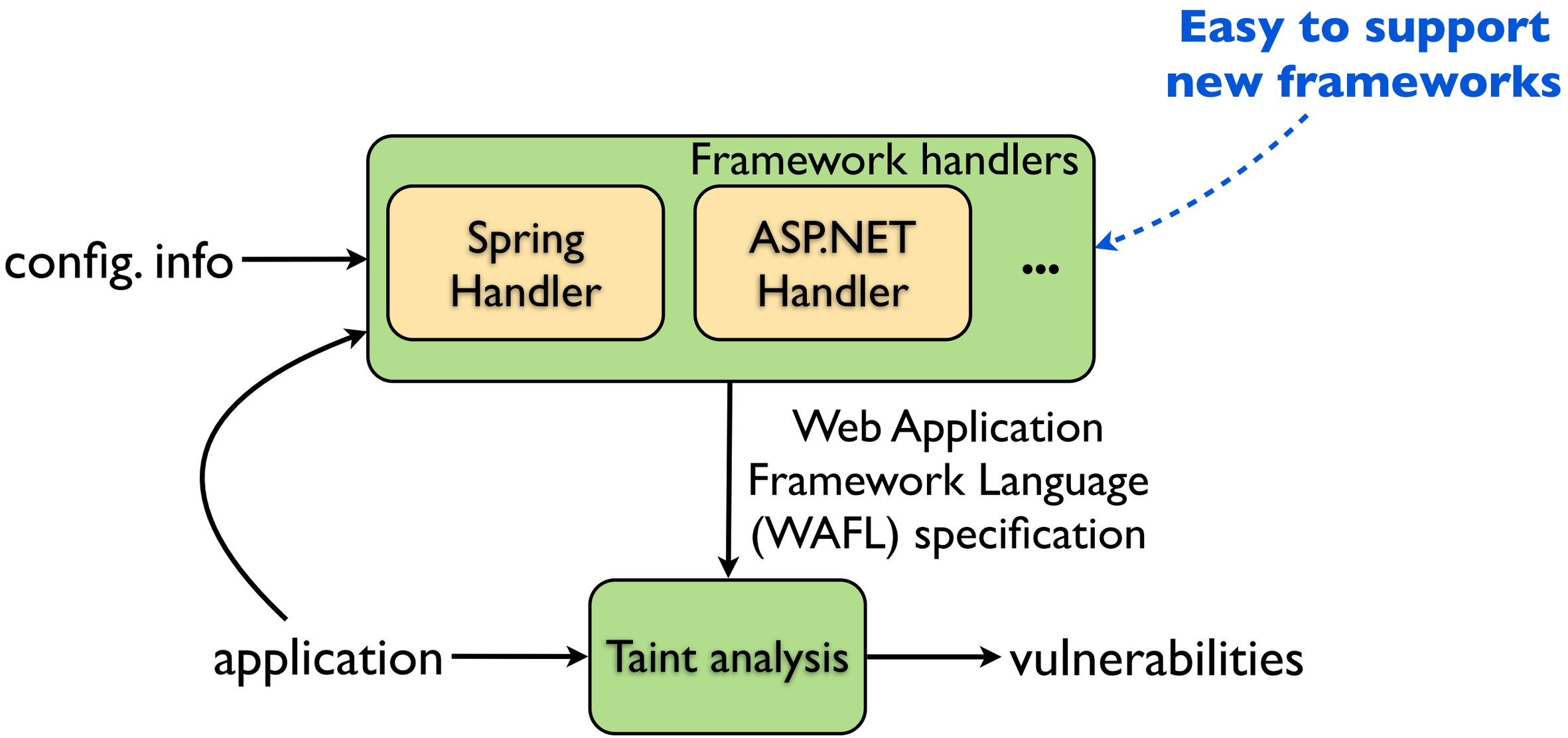
A Framework for Frameworks (F4F)



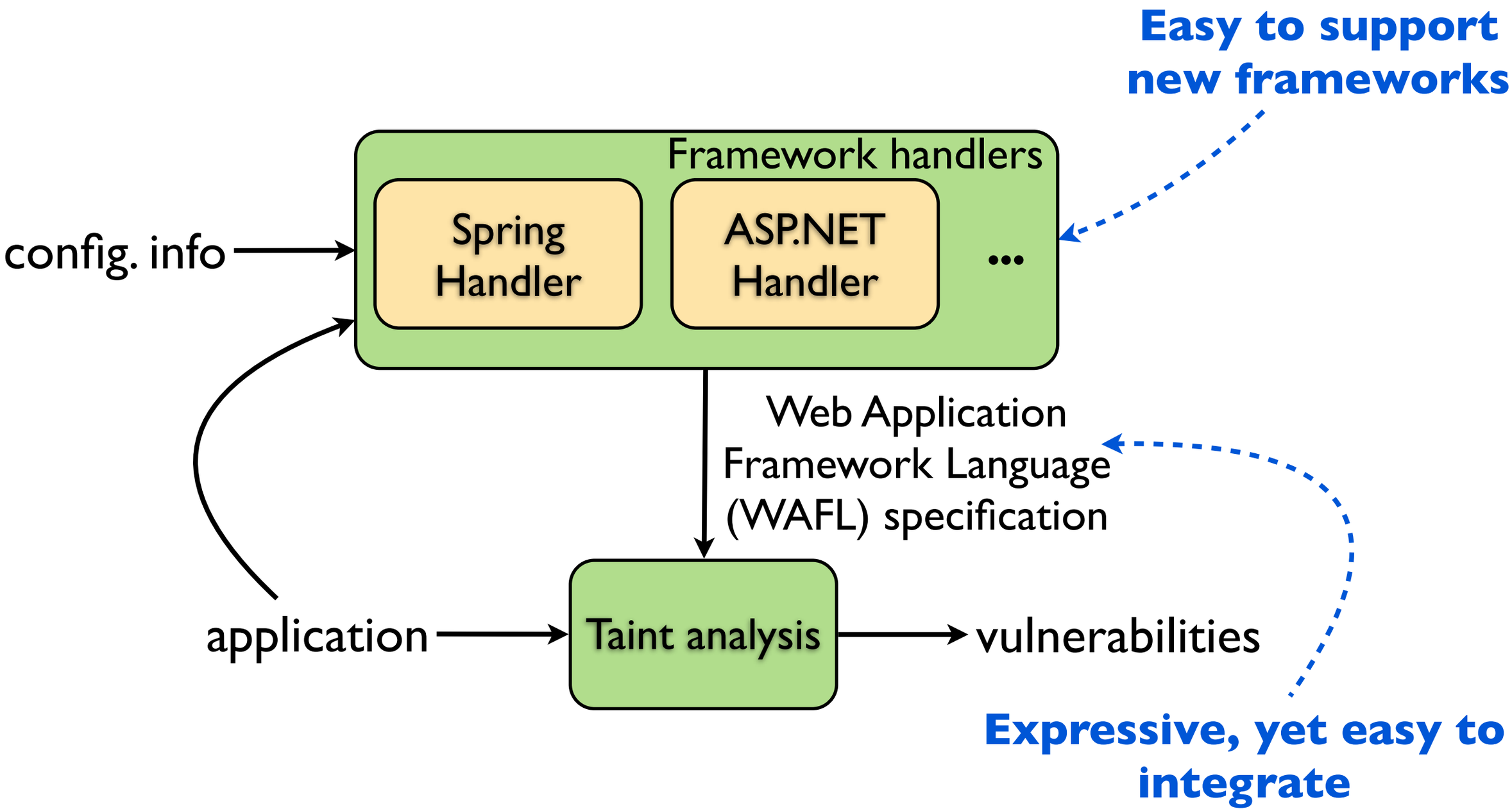
A Framework for Frameworks (F4F)



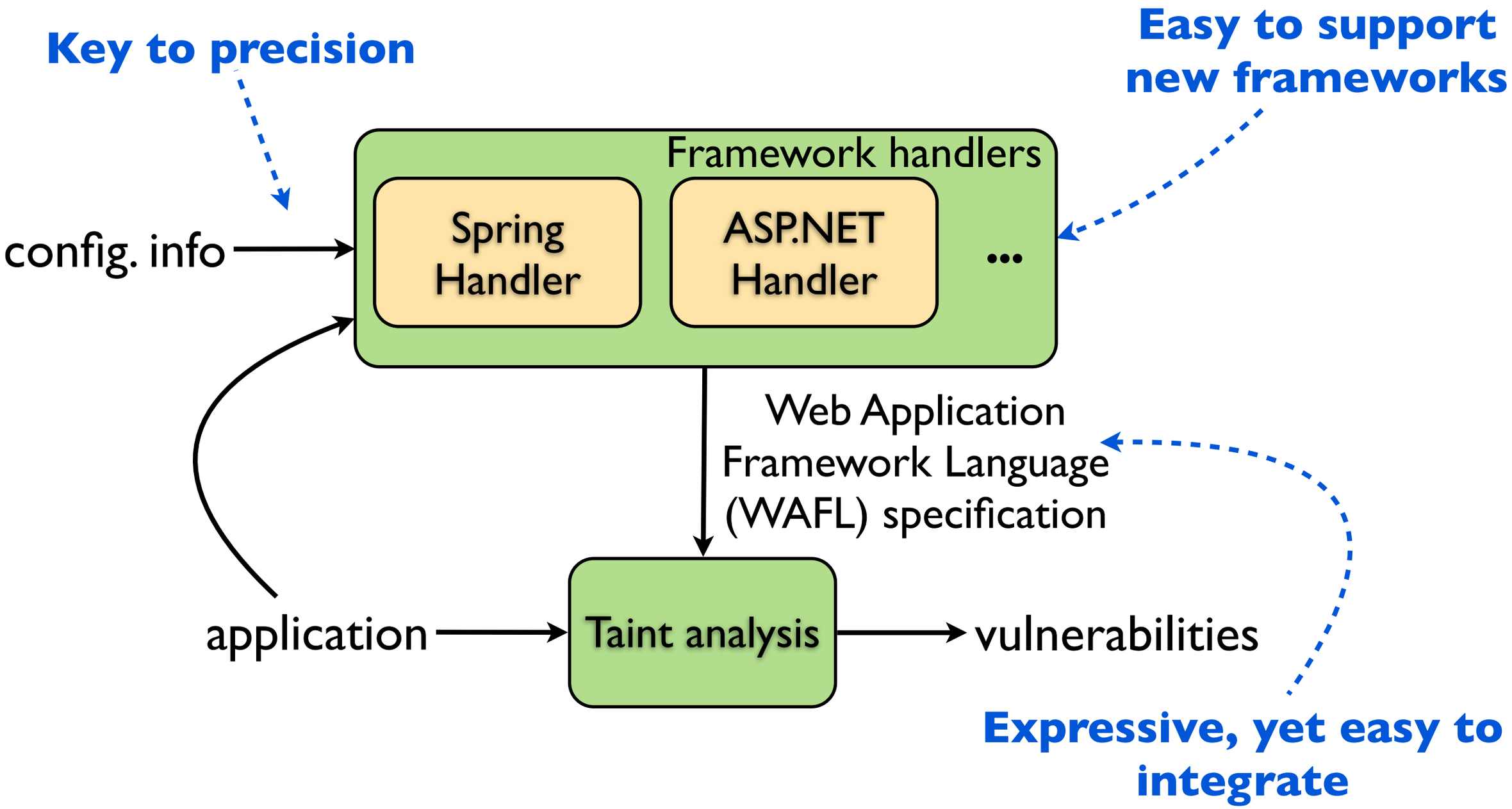
A Framework for Frameworks (F4F)



A Framework for Frameworks (F4F)



A Framework for Frameworks (F4F)



Example: edit profile

Example: edit profile

Code

Configuration

Example: edit profile

Code

```
// 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

```
// 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).

WAFL: synthetic methods

```
fun entrypoint UserAction_entry(request) {
  UserForm f = new UserForm();
  f.setFirstName(request.getParam(“firstName”));
  f.setLastName(request.getParam(“lastName”));
  (new UserAction()).exec(request, f);
}
```

WAFL: synthetic methods

```
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

Example: show profile

Code

Configuration

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

```
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

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

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

```
// 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

```
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;
}
```

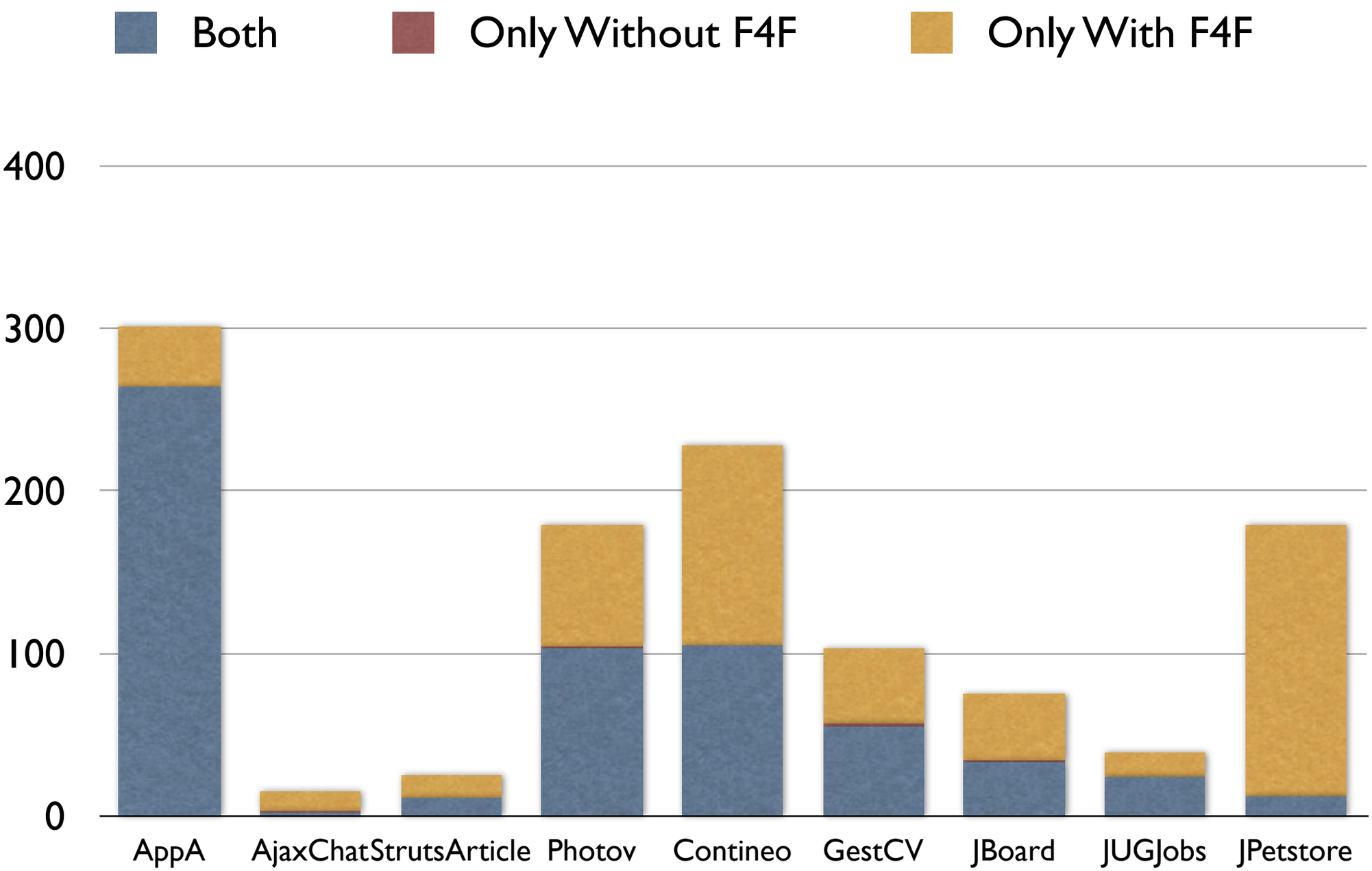
Replacement info

dispatch 'req.setAttribute("user", uf)' in exec() to repl1()

Evaluation

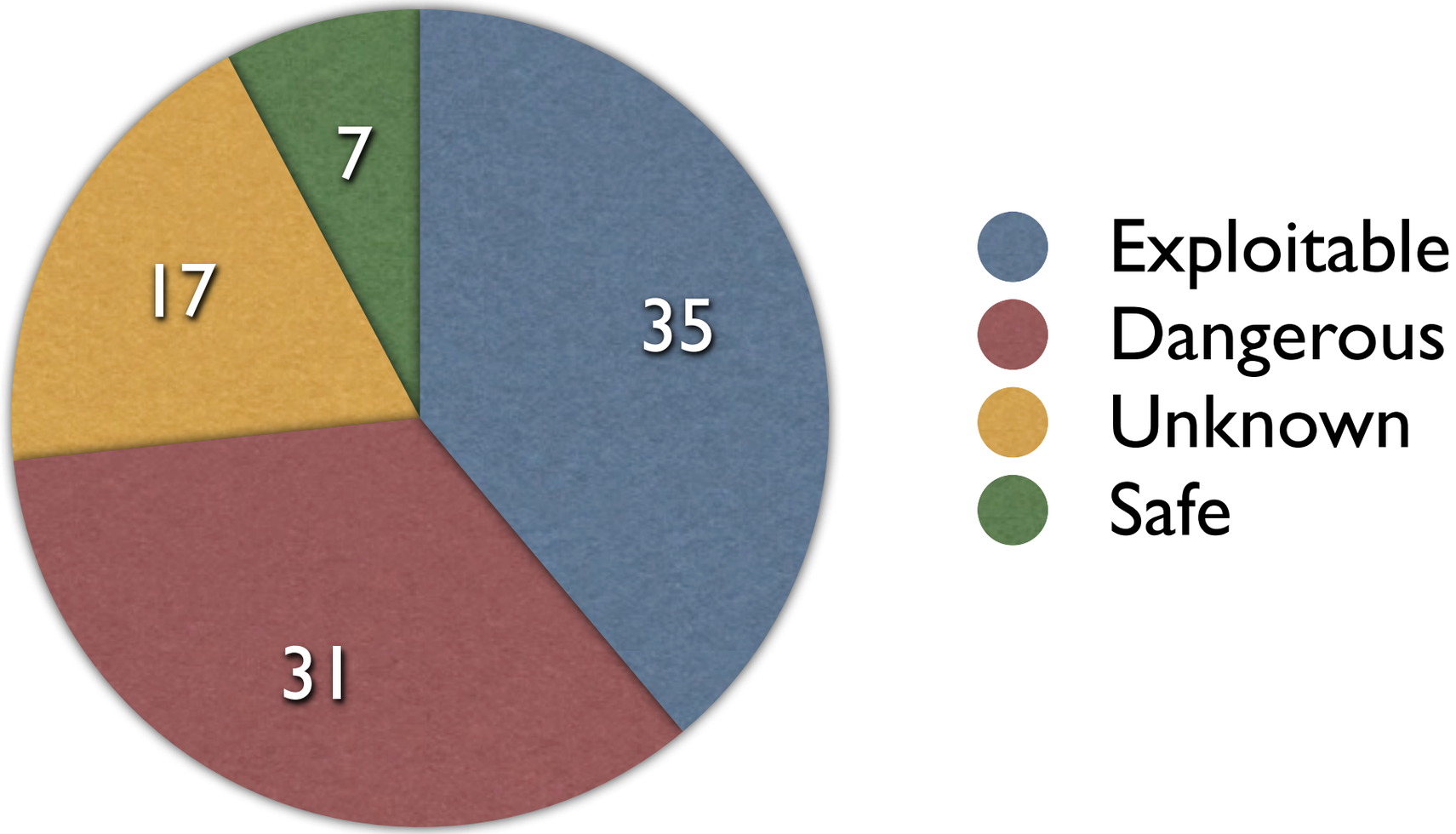
- Nine framework-based web apps
  - Frameworks: Struts, Spring, Tiles, EL
  - See http://bit.ly/F4Fbenchmarks for generated WAFL specs
- Ran state-of-the-art taint analysis with and without F4F

Issues reported



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

Rational. software

AppScan Source Edition 8.0

WALA

T. J. WATSON LIBRARIES FOR ANALYSIS