OPA: Language Support for a Sane, Safe and Secure Web

David Rajchenbach-Teller
Head of R&D
MLstate
David.Teller@mlstate.com
twitter.com/mlstate
+33 1 55 43 76 55

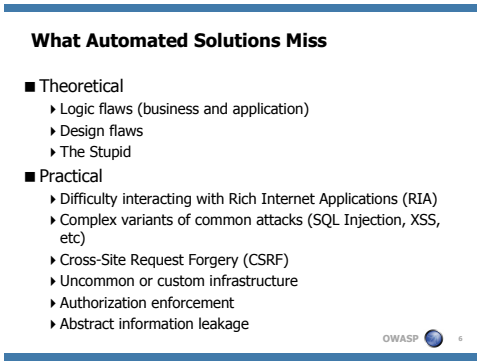
OWASP
June 24th, 2010

The OWASP Foundation
http://www.owasp.org

What Automated Solutions Miss

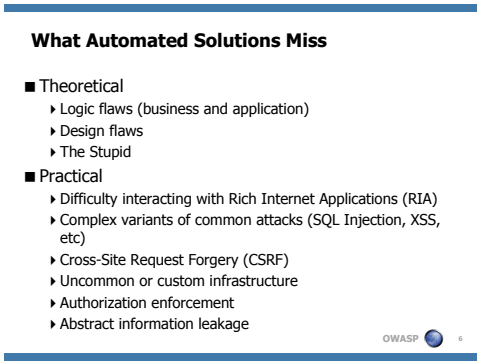
- Theoretical
  - Logic flaws (business and application)
  - Design flaws
  - The Stupid
- Practical
  - Difficulty interacting with Rich Internet Applications (RIA)
  - Complex variants of common attacks (SQL Injection, XSS, etc)
  - Cross-Site Request Forgery (CSRF)
  - Uncommon or custom infrastructure
  - Authorization enforcement
  - Abstract information leakage

What it's all about



"Automated vs. Manual: You can't filter The Stupid"
Charles Henderson, David Byrne
AppSec DC 2009 + 2010

What it's all about



"Automated vs. Manual: You can't filter The Stupid"
Charles Henderson, David Byrne
AppSec DC 2009 + 2010

"Oh, yeah?"
The MLstate team
AppSec Research 2010

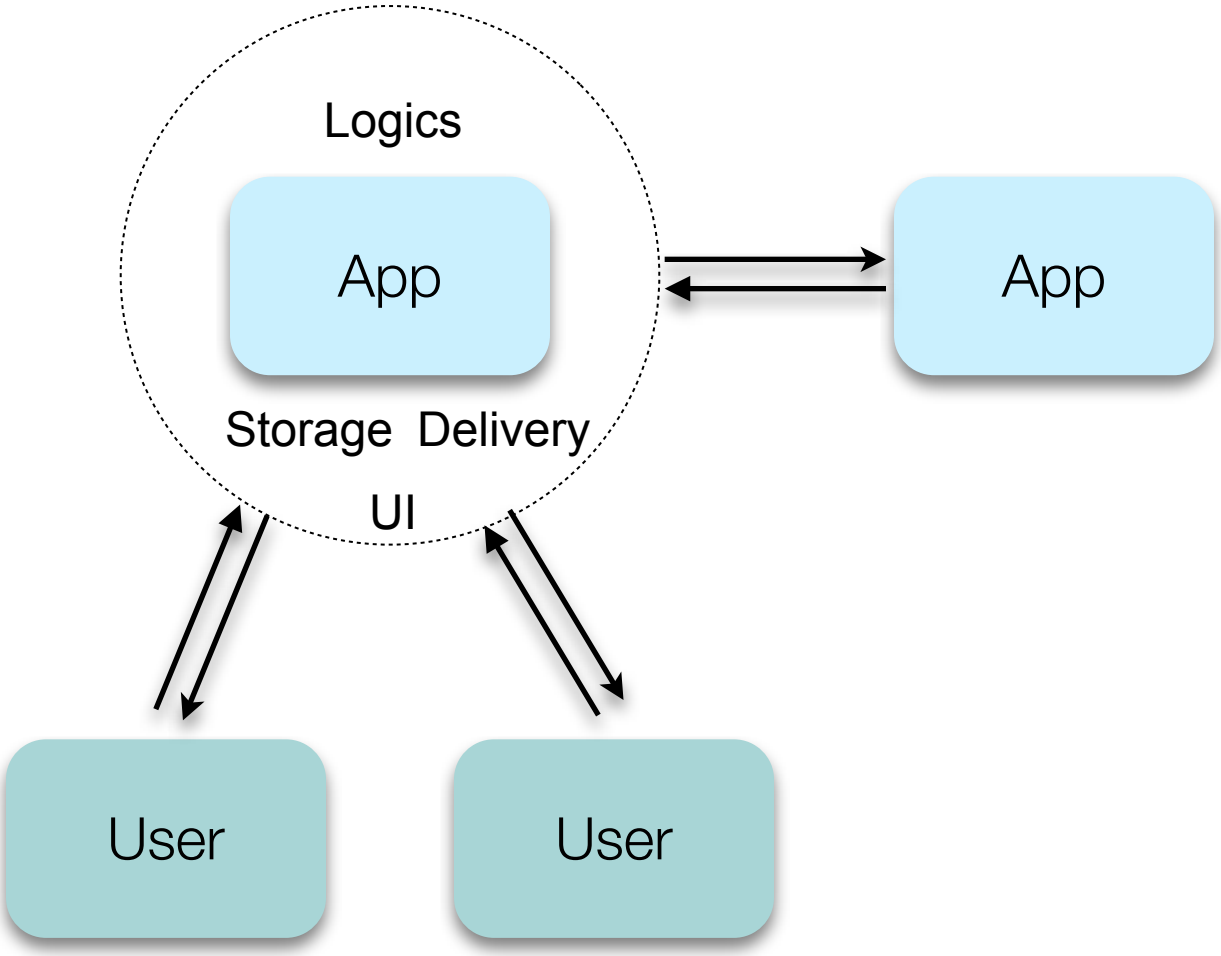
OPA: Language Support for a Sane, Safe and Secure Web

- General approach
- Keeping in the ~~Smart~~ Useful
- Filtering out the ~~Stupid~~ Fragile
- Lessons and Future

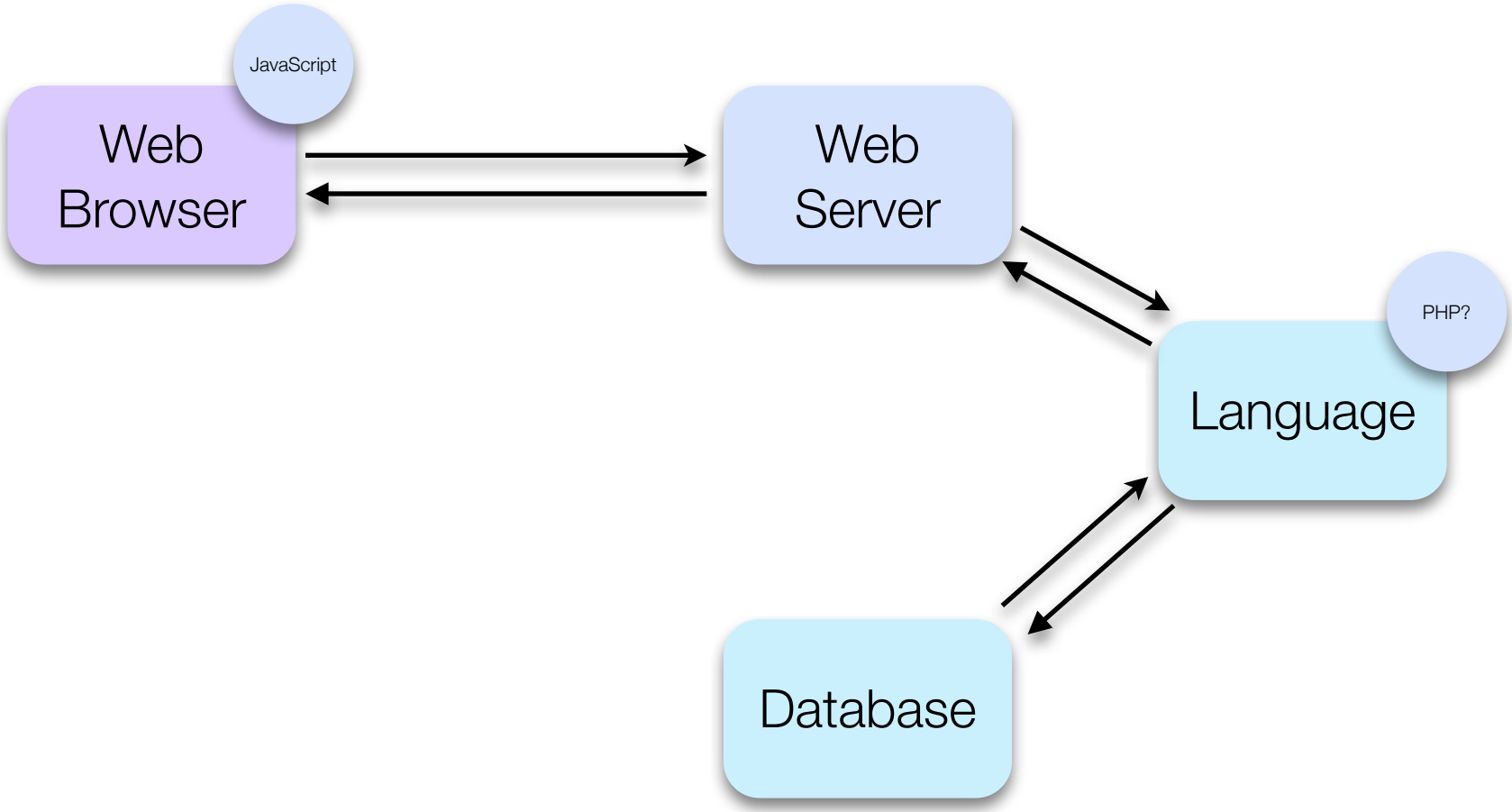
OPA: Language Support for a Sane, Safe and Secure Web

- General approach
- Keeping in the ~~Smart~~ Useful
- Filtering out the ~~Stupid~~ Fragile
- Lessons and Future

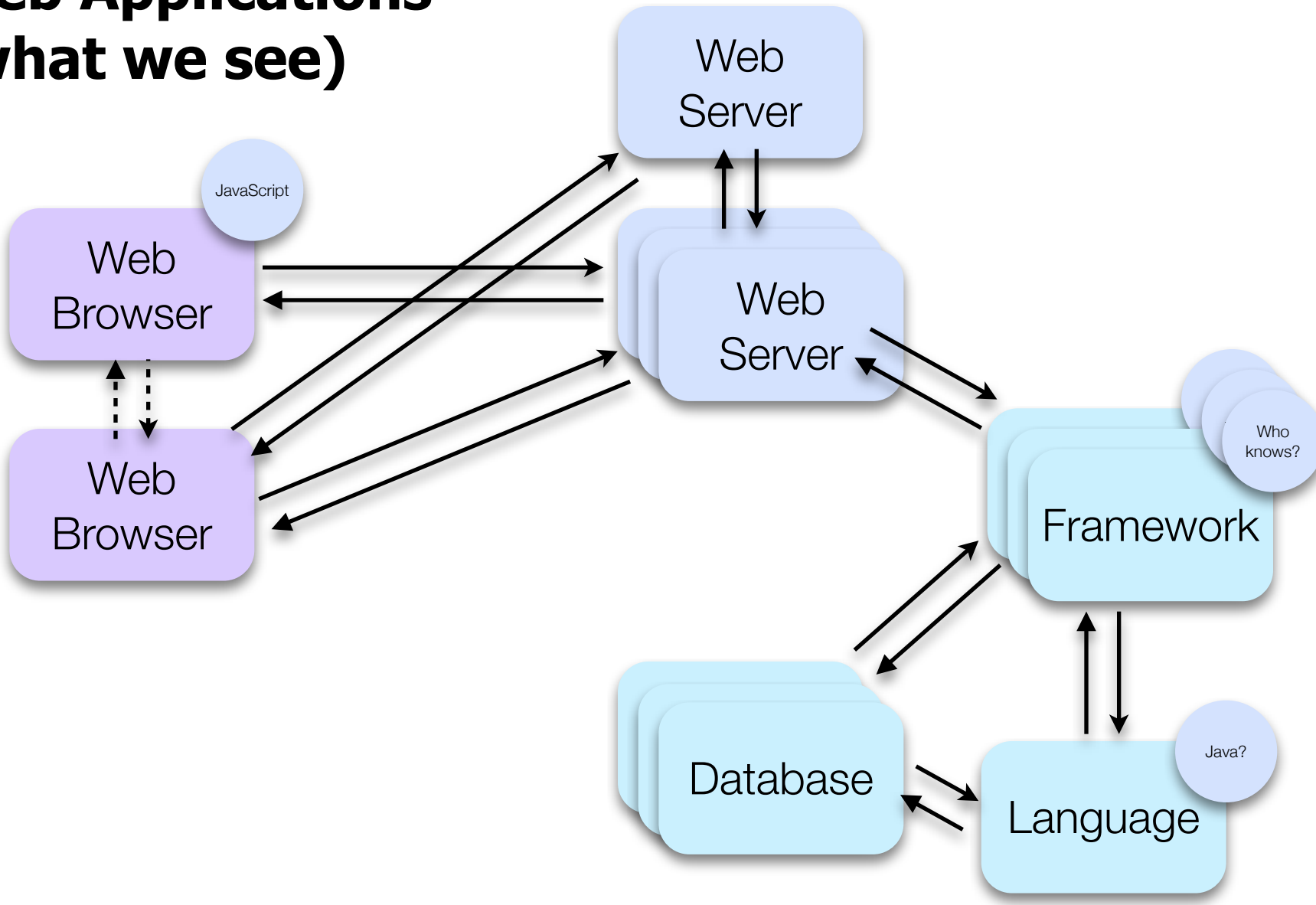
Web Applications (high-level design)



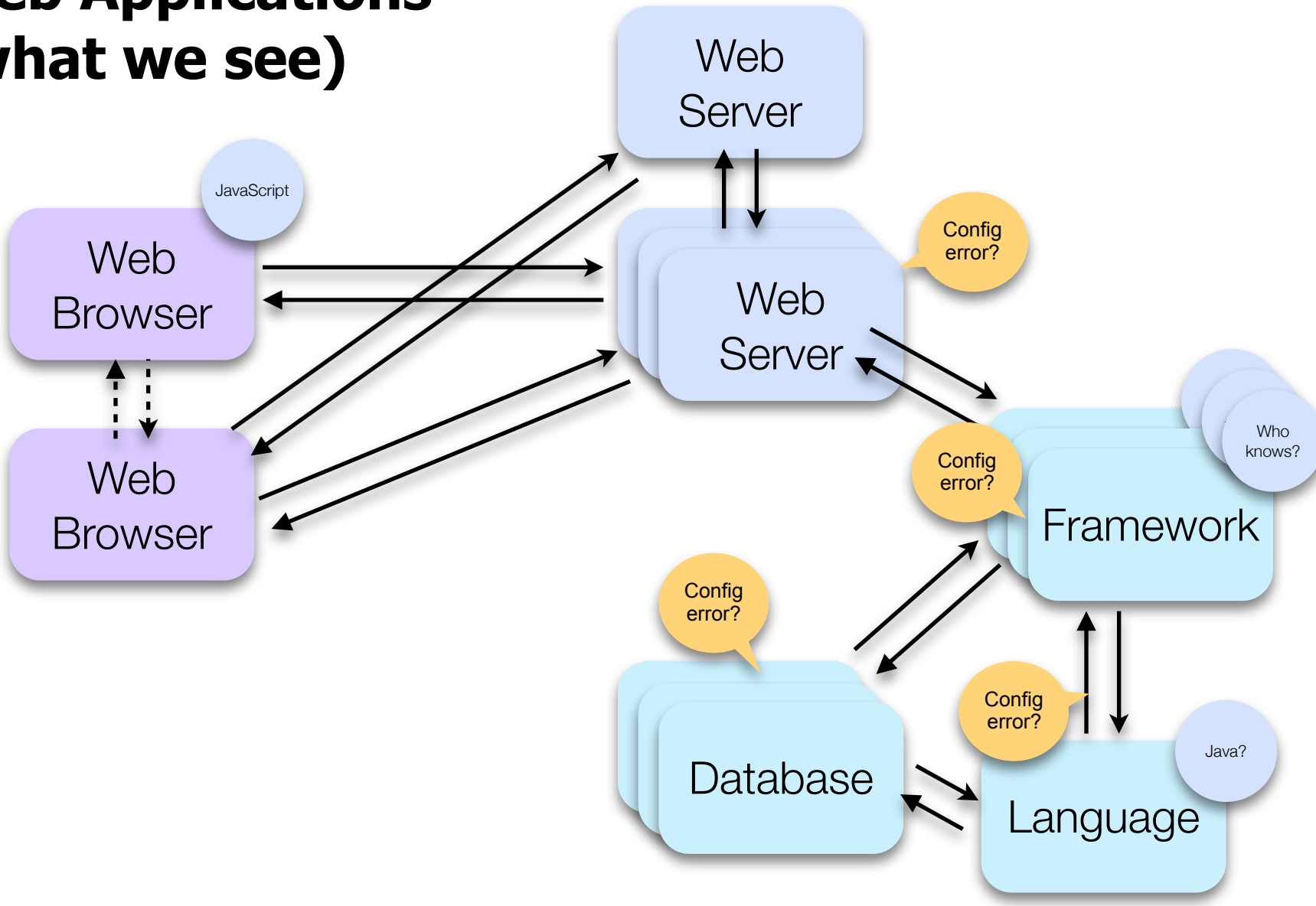
Web Applications (what we see)



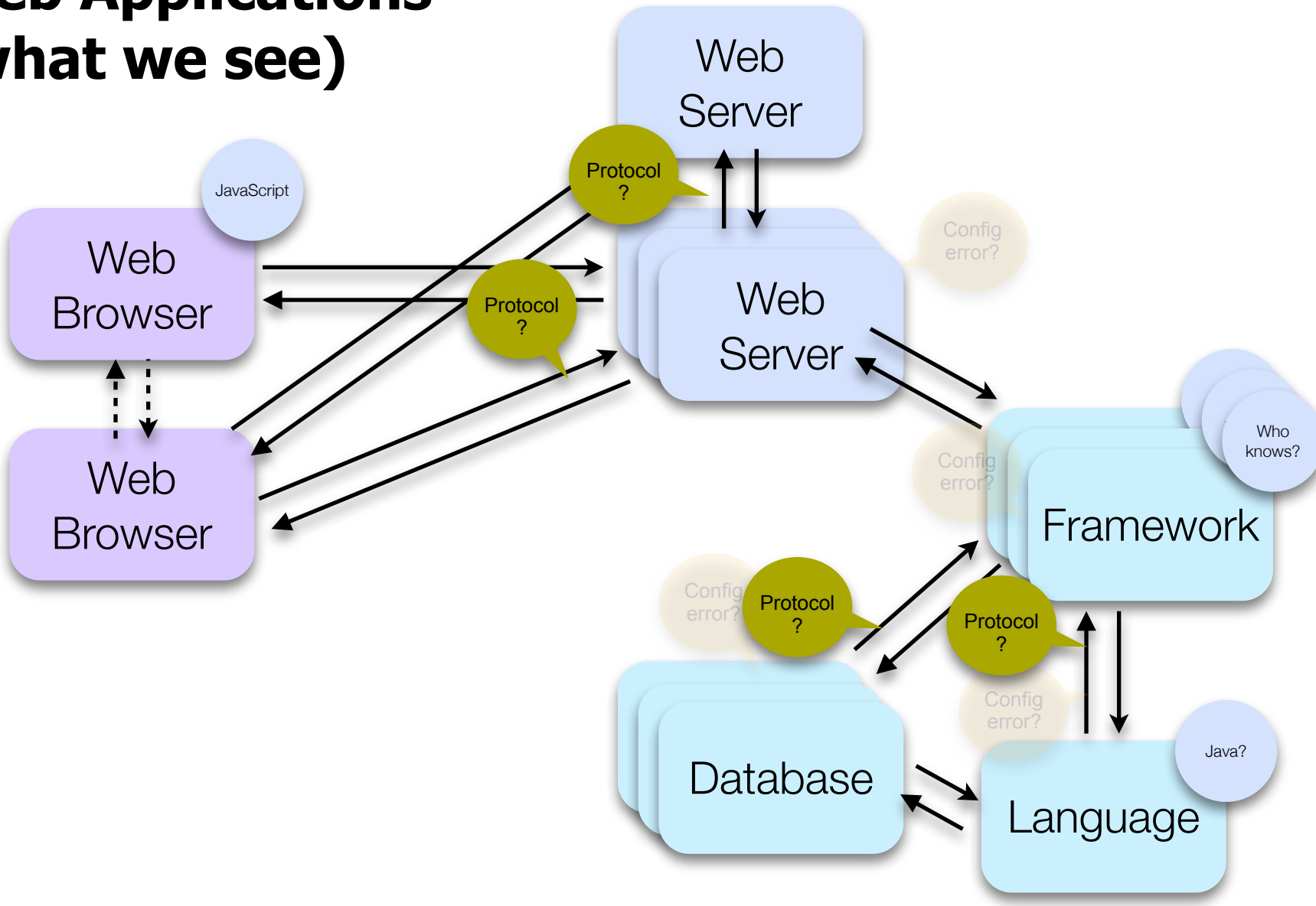
Web Applications (what we see)



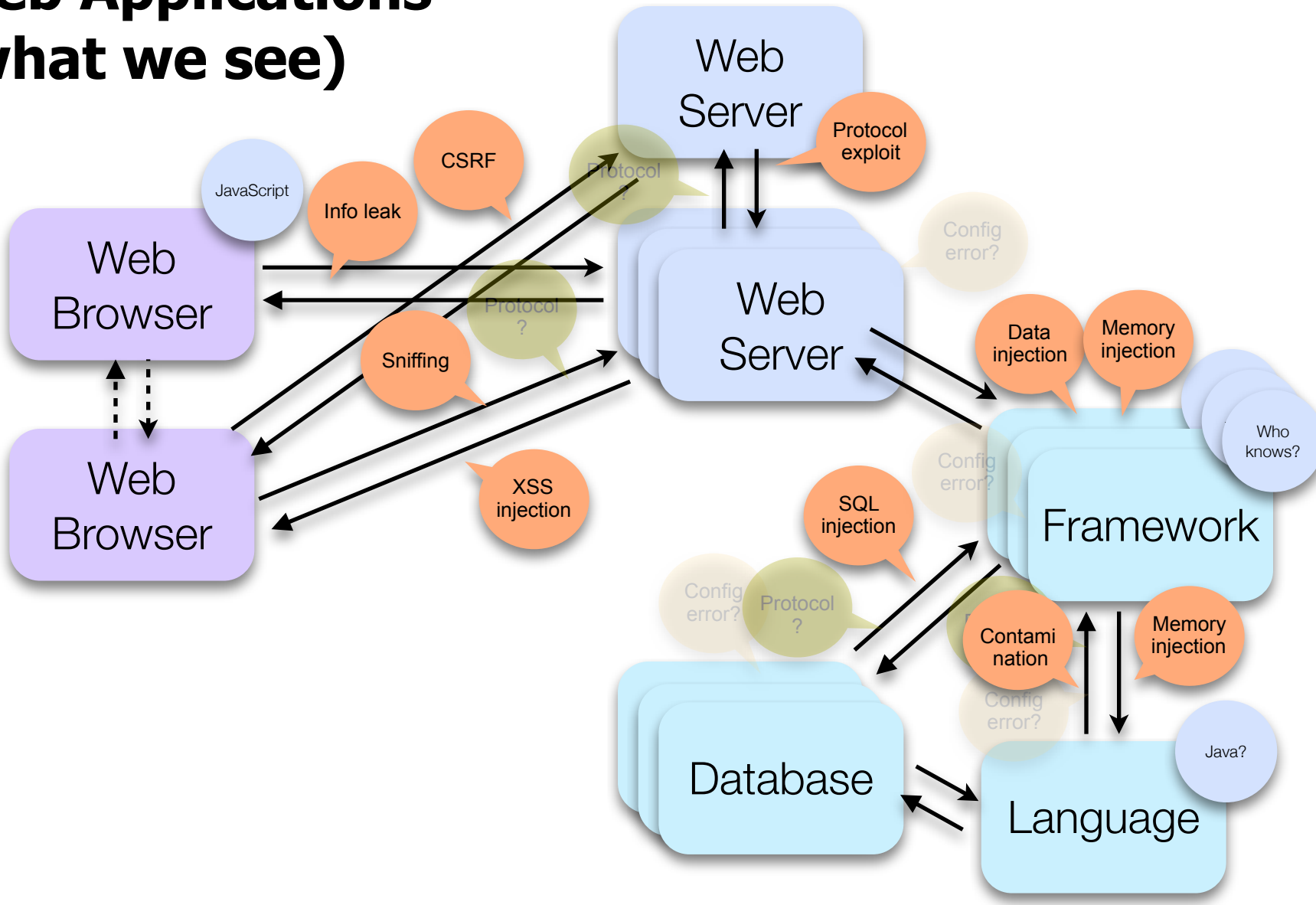
Web Applications (what we see)



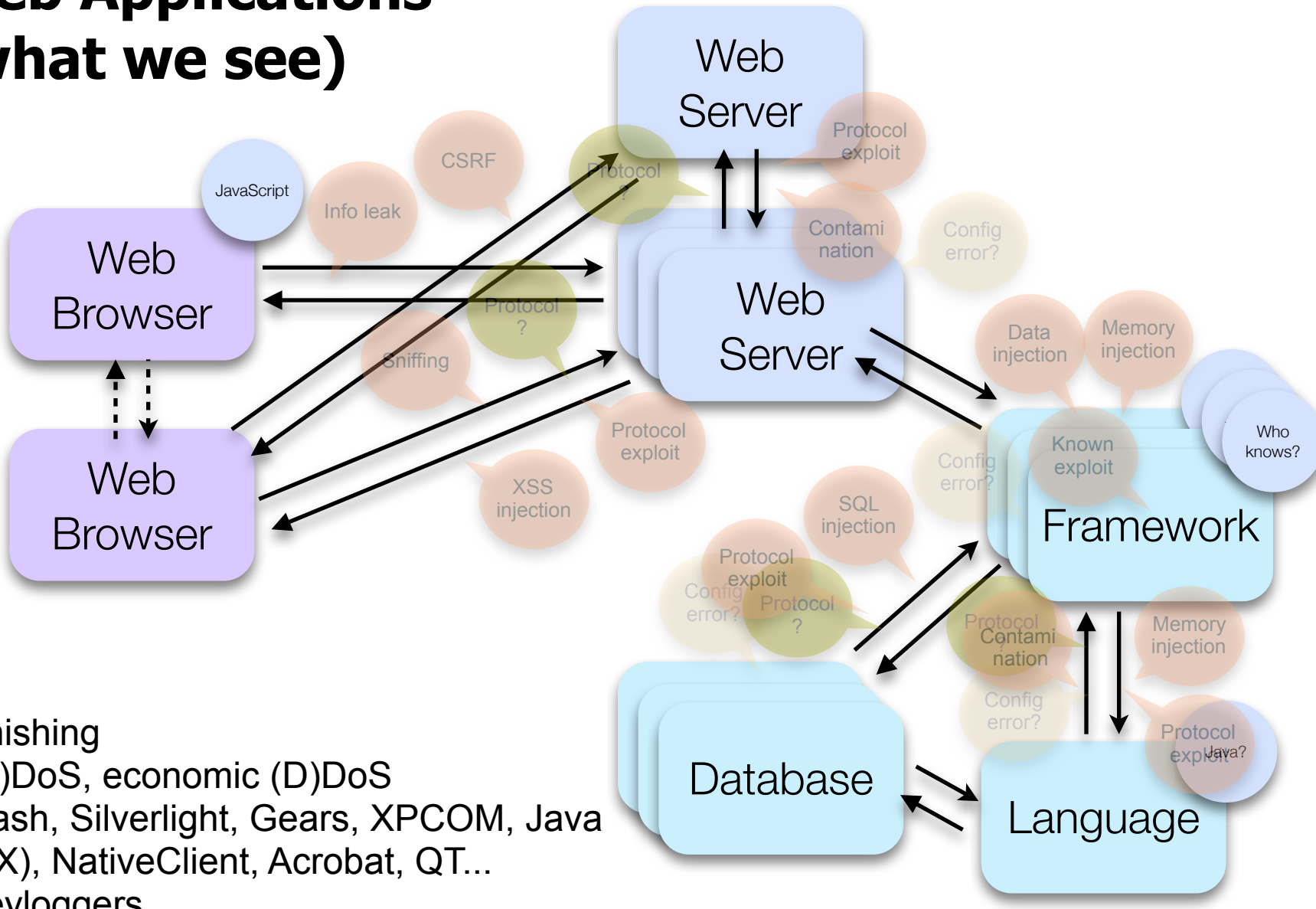
Web Applications (what we see)



Web Applications (what we see)



Web Applications (what we see)



+ Phishing
+ (D)DoS, economic (D)DoS
+ Flash, Silverlight, Gears, XPCOM, Java (FX), NativeClient, Acrobat, QT...
+ Keyloggers
+ ...

The general idea

The general idea

"If you can't solve the problem, change the problem."
Ferengi Rule of Acquisition #408

The general idea

"If you can't solve the problem, change the problem."
Ferengi Rule of Acquisition #408

"What if the problem is that Joe User is stupid?"
(Joe Developer)

The general idea

"If you can't solve the problem, change the problem."
Ferengi Rule of Acquisition #408

"What if the problem is that Joe User is stupid?"
(Joe Developer)

"It's not him, it's you. You should design your tool so that Joe can't do anything stupid."
Ferengi Rule of Acquisition #408, contd.

The general idea

"If you can't solve the problem, change the problem."
Ferengi Rule of Acquisition #408

"What if the problem is that Joe User is stupid?"
(Joe Developer)

"It's not him, it's you. You should design your tool so that Joe can't do anything stupid."
Ferengi Rule of Acquisition #408, contd.

"What if the problem is that Joe Developer is stupid?"
(Joe Meta-Developer)

The general idea

"If you can't solve the problem, change the problem."
Ferengi Rule of Acquisition #408

"What if the problem is that Joe User is stupid?"
(Joe Developer)

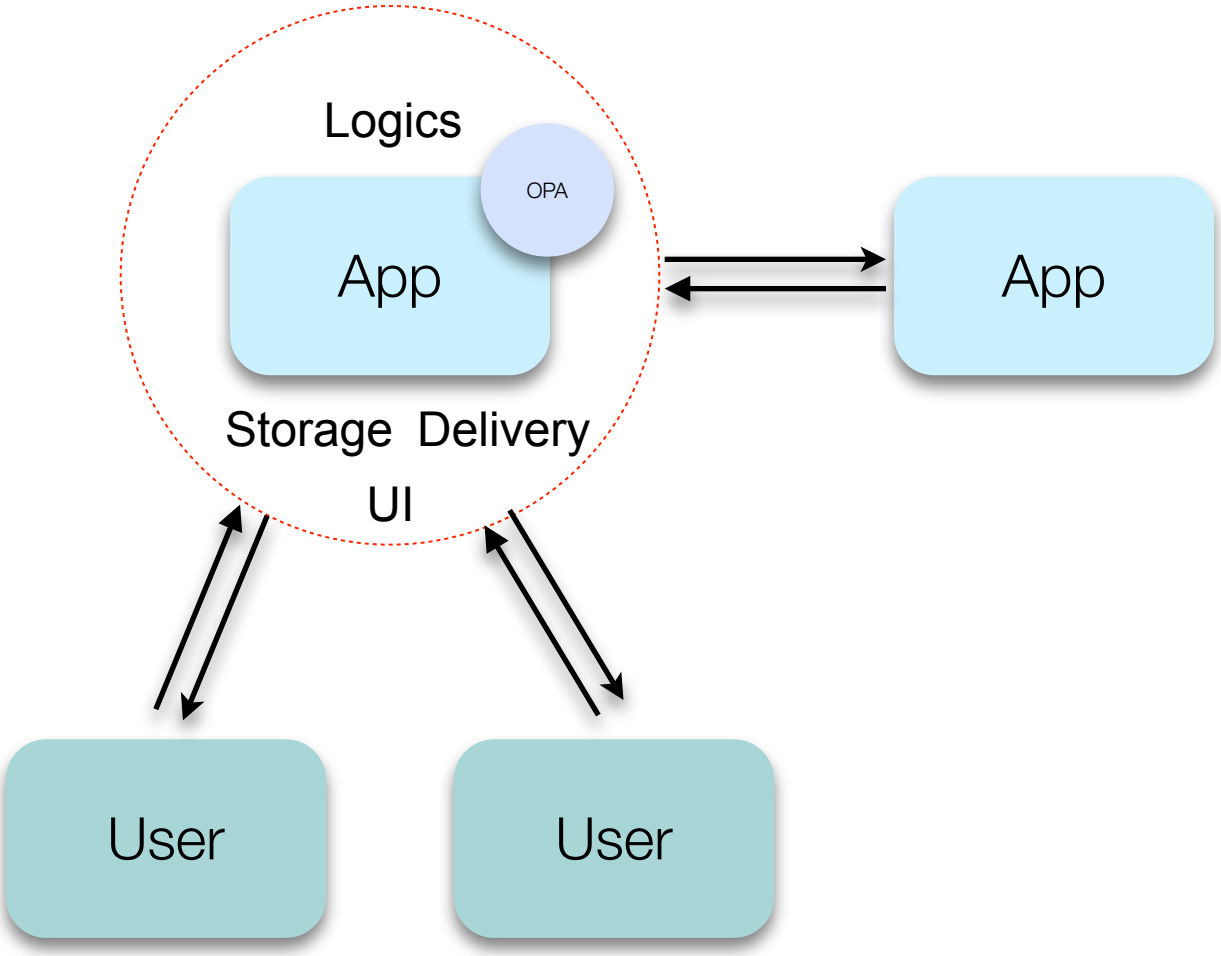
"It's not him, it's you. You should design your tool so that Joe can't do anything stupid."
Ferengi Rule of Acquisition #408, contd.

"What if the problem is that Joe Developer is stupid?"
(Joe Meta-Developer)

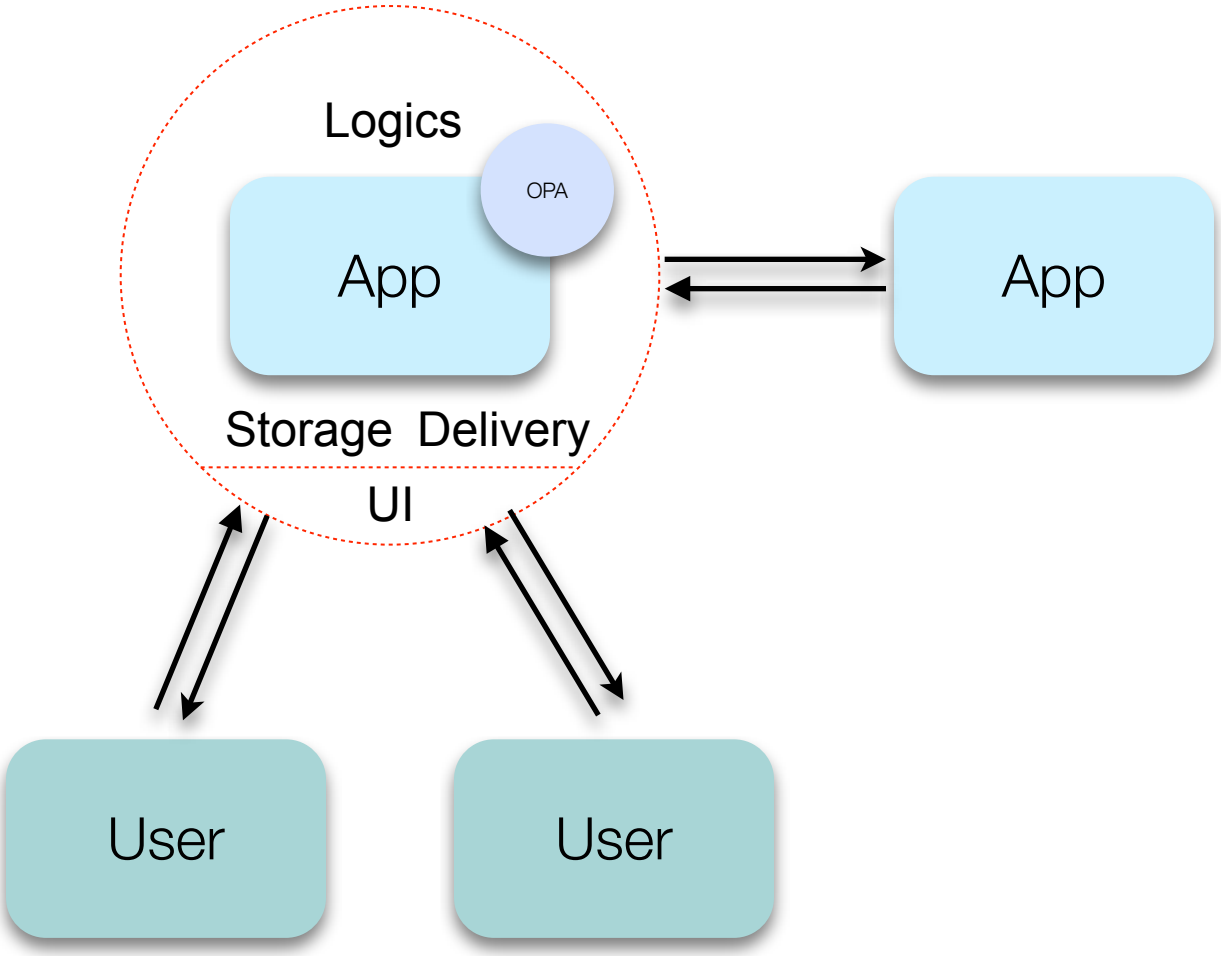
"Haven't I answered that question already?"
Ferengi Rule of Acquisition #408, contd.

"...oh, and don't forget to make sure that Joe can still use the tools!"
Ferengi Rule of Acquisition #408, contd.

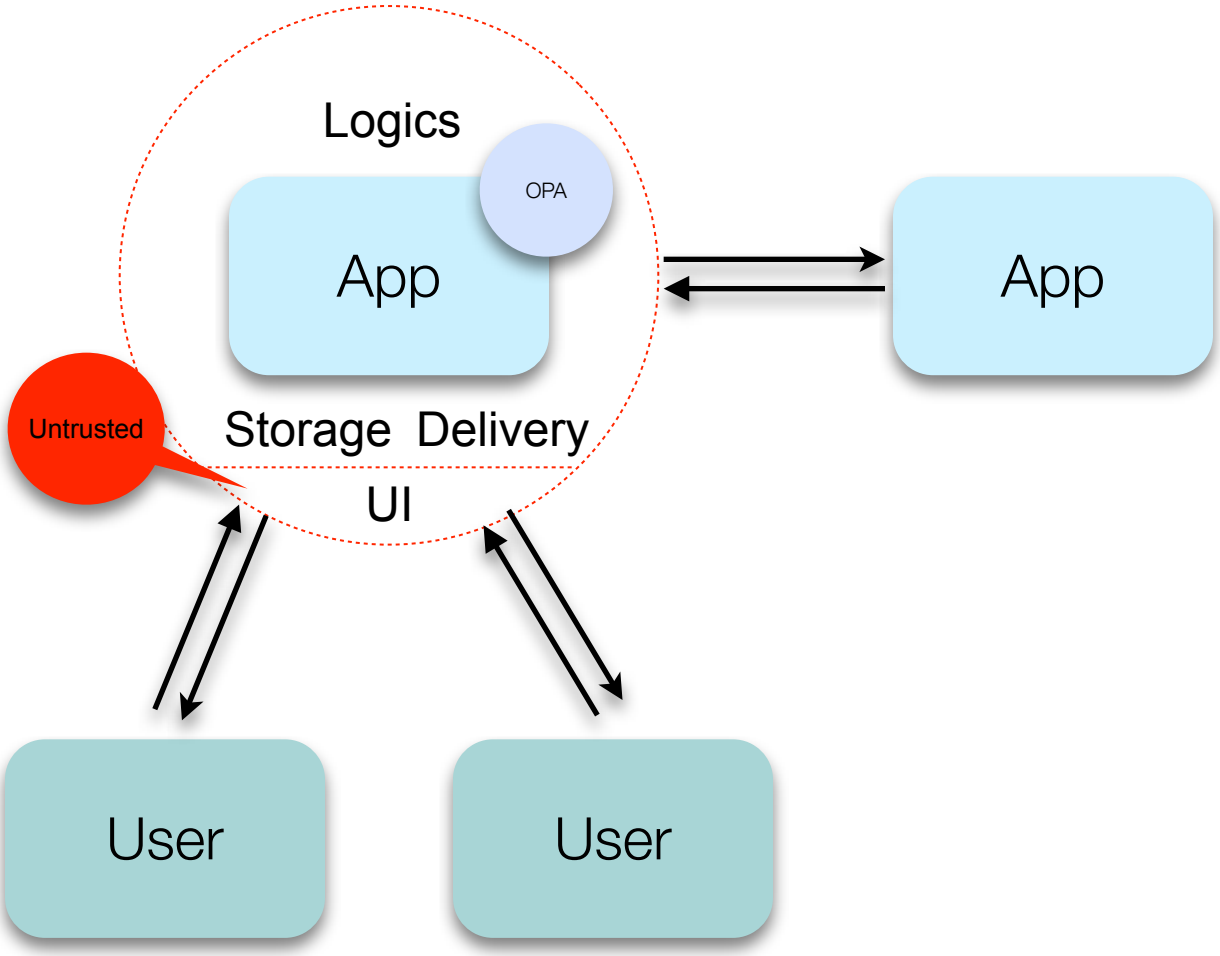
Web applications (with OPA)



Web applications (with OPA)



Web applications (with OPA)



General OPA design

General OPA design

- Clean-slate design
  - Based on formal methods
  - Safe languages from the bottom up

General OPA design

- Clean-slate design
  - Based on formal methods
  - Safe languages from the bottom up

- One language for the whole application
  - Glue & checks generated automatically
  - No impedance mismatch

General OPA design

- Clean-slate design
  - Based on formal methods
  - Safe languages from the bottom up
- One language for the whole application
  - Glue & checks generated automatically
  - No impedance mismatch
- "Just" a distributed system
  - In which not all principals are trusted
  - And communications use web standards
  - Security is (mostly) automatic

General OPA design

- Clean-slate design
  - Based on formal methods
  - Safe languages from the bottom up
- One language for the whole application
  - Glue & checks generated automatically
  - No impedance mismatch
- "Just" a distributed system
  - In which not all principals are trusted
  - And communications use web standards
  - Security is (mostly) automatic

Joe doesn't need to know

Simpler than LAMP

Joe doesn't need to know

OPA: Language Support for a Sane, Safe and Secure Web

- General approach
- Keeping in the ~~Smart~~ Useful
- Filtering out the ~~Stupid~~ Fragile
- Lessons and Future

OPA: Language Support for a Sane, Safe and Secure Web

- General approach
- Keeping in the ~~Smart~~ Useful
- Filtering out the ~~Stupid~~ Fragile
- Lessons and Future

Hello, web

Hello, web

```
server = one_page_server("Hello, web", -> <>Hello, web</>)
```

Complete web application.

1) Compile
`opa hello.opa`

2) Run
`./hello.exe`

3) Test
browse `http://localhost:8080`

URL shortener (22 loc)

```
db /abbrevs: intmap(string)
db /abbrevs[_] = "/"

make_shorter(url:string):string =
  key = Db.fresh_key(/abbrevs)
  do /abbrevs[key] <- url
  "{key}"
_ = @server(make_shorter)

do_shorten(_) =
exec([#shortened <- make_shorter(Page.getVal(#origin))])

urls = parser
| "/" dest=Rule.integer ->
    Resource.redirection_page("Please wait a second",
           <div class="loading">(loading...)</>,{address_moved},0,/abbrevs[dest])
| .* ->
    Resource.html("MLstate redirector",
    <>Please enter a URL you wish to shorten<br/>
    <input id="origin"/><button onclick={do_shorten}>Shorten</button>
    <div id="shortened"></div>
    </>)

server = simple_server(urls)
```

URL shortener (22 loc)

```
db /abbrevs: intmap(string)
db /abbrevs[_] = "/"

make_shorter(url:string):string =
  key = Db.fresh_key(/abbrevs)
  do /abbrevs[key] <- url
  "{key}"
_ = @server(make_shorter)

do_shorten(_) =
exec([#shortened <- make_shorter(Page.getVal(#origin))])

urls = parser
| "/" dest=Rule.integer ->
    Resource.redirection_page("Please wait a second",
          <div class="loading">(loading...)</>,{address_moved},0,/abbrevs[dest])
| .* ->
    Resource.html("MLstate redirector",
    <>Please enter a URL you wish to shorten<br/>
    <input id="origin"/><button onclick={do_shorten}>Shorten</button>
    <div id="shortened"></div>
    </>)

server = simple_server(urls)
```

Database

Control

UI

Nice web chat

```
type mess = {id: string; message: string}
room = Network.empty():Network.network(mess)
connect(callback) = Network.add(Session.make_callback(callback), room)
make_broadcaster(id) = _ -> Network.broadcast({~id message=Page.get_value(#entry)}, room)

styled_page(title, body) = (
  Resource.full_page(title, body,
  <link rel="stylesheet" type="text/css" href="resources/css.css" />, {success}, [])
)


user_update(x:mess) = (
  line = <div class="opa-line">
    <div class="opa-wrap"><div class="opa-user">{x.id}:</div>
    <div class="opa-message">{x.message}</div></div>
  </div>
  do exec([#show +<- line ])
  Page.set_scroll_top(#show, Page.get_height(#show)+Page.get_scroll_top(#show))
)


start(connexion) = (
   id = String.sub(0, 8, Server.get_user(connexion) ? "Unknown")
   broadcast = make_broadcaster(id)
   styled_page("Chat",  //Display
     <script onload={_ -> connect()}/>
     <div id="header"><div id="logo"></div></div>
     <div id="show"></div>
     <input id="entry"/>
     <div class="opa-button" onclick={broadcast}>Send!</div>
   )

)

urls      = parser .* -> start
resources = @static_include_directory("resources")
server    = Server.make(Resource.add_auto_server(resources, urls))
```

Nice web chat

```
type mess = {id: string; message: string}
room = Network.empty():Network.network(mess)
connect(callback) = Network.add(Session.make_callback(callback), room)
make_broadcaster(id) = _ -> Network.broadcast({~id message=Page.get_value(#entry)}, room)

styled_page(title, body) = (
  Resource.full_page(title, body,
  <link rel="stylesheet" type="text/css" href="resources/css.css" />, {success}, [])
)


user_update(x:mess) = (
  line = <div class="opa-line">
    <div class="opa-wrap"><div class="opa-user">{x.id}:</div>
    <div class="opa-message">{x.message}</div></div>
  </div>
  do exec([#show +<- line ])
  Page.set_scroll_top(#show, Page.get_height(#show)+Page.get_scroll_top(#show))
)


start(connexion) = (
   id = String.sub(0, 8, Server.get_user(connexion) ? "Unknown")
   broadcast = make_broadcaster(id)
   styled_page("Chat",  //Display
     <script onload={_ -> connect()}/>
     <div id="header"><div id="logo"></div></div>
     <div id="show"></div>
     <input id="entry"/>
     <div class="opa-button" onclick={broadcast}>Send!</div>
   )

)

urls      = parser .* -> start
resources = @static_include_directory("resources")
server    = Server.make(Resource.add_auto_server(resources, urls))
```

UI

Nice web chat

```
type mess = {id: string; message: string}
room = Network.empty():Network.network(mess)
connect(callback) = Network.add(Session.make_callback(callback), room)
make_broadcaster(id) = _ -> Network.broadcast({~id message=Page.get_value(#entry)}, room)

styled_page(title, body) = (
  Resource.full_page(title, body,
  <link rel="stylesheet" type="text/css" href="resources/css.css" />, {success}, [])
)


user_update(x:mess) = (
  line = <div class="opa-line">
    <div class="opa-wrap"><div class="opa-user">{x.id}:</div>
    <div class="opa-message">{x.message}</div></div>
  </div>
  do exec([#show +<- line ])
  Page.set_scroll_top(#show, Page.get_height(#show)+Page.get_scroll_top(#show))
)


start(connexion) = (
   id = String.sub(0, 8, Server.get_user(connexion) ? "Unknown")
   broadcast = make_broadcaster(id)
   styled_page("Chat",  //Display
     <script onload={_ -> connect()}/>
     <div id="header"><div id="logo"></div></div>
     <div id="show"></div>
     <input id="entry"/>
     <div class="opa-button" onclick={broadcast}>Send!</div>
   )

)

urls      = parser .* -> start
resources = @static_include_directory("resources")
server    = Server.make(Resource.add_auto_server(resources, urls))
```

Real-time web

UI

Nice web chat

```
type mess = {id: string; message: string}
room = Network.empty():Network.network(mess)
connect(callback) = Network.add(Session.make_callback(callback), room)
make_broadcaster(id) = _ -> Network.broadcast({~id message=Page.get_value(#entry)}, room)

styled_page(title, body) = (
  Resource.full_page(title, body,
  <link rel="stylesheet" type="text/css" href="resources/css.css" />, {success}, [])
)


user_update(x:mess) = (
  line = <div class="opa-line">
    <div class="opa-wrap"><div class="opa-user">{x.id}:</div>
    <div class="opa-message">{x.message}</div></div>
  </div>
  do exec([#show +<- line ])
  Page.set_scroll_top(#show, Page.get_height(#show)+Page.get_scroll_top(#show))
)


start(connexion) = (
   id = String.sub(0, 8, Server.get_user(connexion) ? "Unknown")
   broadcast = make_broadcaster(id)
   styled_page("Chat",  //Display
     <script onload={_ -> connect()}/>
     <div id="header"><div id="logo"></div></div>
     <div id="show"></div>
     <input id="entry"/>
     <div class="opa-button" onclick={broadcast}>Send!</div>
   )

)

urls      = parser .* -> start
resources = @static_include_directory("resources")
server    = Server.make(Resource.add_auto_server(resources, urls))
```

Real-time web

UI

URLs

There's more

- Games
- Productivity tools
- Development tools
- Security applications
- e-Commerce applications
- …

Things we can do (and check!)

Things we can do (and check!)

Programmable security policies

Ajax SOAP Complex database

...

Comet XML processing

Fine-grained sessions Time-unlimited rollback

Distribution

Multitasking

Higher-order database

User interface Concurrent transactions

WSDL

Higher-order programming

Data history

Capabilities

OPA: Language Support for a Sane, Safe and Secure Web

- General approach
- Keeping in the ~~Smart~~ Useful
- Filtering out the ~~Stupid~~ Fragile
- Lessons and Future

OPA: Language Support for a Sane, Safe and Secure Web

- General approach
- Keeping in the ~~Smart~~ Useful
- Filtering out the ~~Stupid~~ Fragile
- Lessons and Future

Transparent protections

```
type mess = {id: string; message: string}
room = Network.empty():Network.network(mess)
connect(callback) = Network.add(Session.make_callback(callback), room)
make_broadcaster(id) = _ -> Network.broadcast({~id message=Page.get_value(#entry)}, room)

styled_page(title, body) = (
  Resource.full_page(title, body,
  <link rel="stylesheet" type="text/css" href="resources/css.css" />, {success}, [])
)


user_update(x:mess) = (
  line = <div class="opa-line">
    <div class="opa-wrap"><div class="opa-user">{x.id}:</div>
    <div class="opa-message">{x.message}</div></div>
  </div>
  do exec([#show +<- line ])
  Page.set_scroll_top(Page.get_height(#show)+Page.get_scroll_top(#show), #show)
)


start(connexion) = (
   id = String.sub(0, 8, Server.get_user(connexion) ? "Unknown")
   broadcast = make_broadcaster(id)
   styled_page("Chat",  //Display
     <script onload={_ -> connect()}/>
     <div id="header"><div id="logo"></div></div>
     <div id="show"></div>
     <input id="entry"/>
     <div class="opa-button" onclick={broadcast}>Send!</div>
   )
)

urls      = parser .* -> start
resources = @static_include_directory("resources")
server    = Server.make(Resource.add_auto_server(resources, urls))
```

Transparent protections

```
type mess = {id: string; message: string}
room = Network.empty():Network.network(mess)
connect(callback) = Network.add(Session.make_callback(callback), room)
make_broadcaster(id) = _ -> Network.broadcast({~id message=Page.get_value(#entry)}, room)

styled_page(title, body) = (
  Resource.full_page(title, body,
  <link rel="stylesheet" type="text/css" href="resources/css.css" />, {success}, [])
)

user_update(x:mess) = (
  line = <div class="opa-line">
    <div class="opa-wrap"><div class="opa-user">{x.id}:</div>
    <div class="opa-message">{x.message}</div></div>
  </div>
  do exec([#show +<- line ])
  Page.set_scroll_top(#show, Page.get_height(#show)+Page.get_scroll_top(#show))
)

  id = String.sub(0, 8, Server.get_user(connexion) ? "Unknown")
  broadcast = make_broadcaster(id)
  styled_page("Chat",  //Display
    <script onload={_ -> connect()}/>
    <div id="header"><div id="logo"></div></div>
    <div id="show"></div>
    <input id="entry"/>
    <div class="opa-button" onclick={broadcast}>Send!</div>
  )
)

urls      = parser .* -> start
resources = @static_include_directory("resources")
server    = Server.make(Resource.add_auto_server(resources, urls))
```

All insertions are checked

Transparent protections

```
type mess = {id: string; message: string}
room = Network.empty():Network.network(mess)
connect(callback) = Network.add(Session.make_callback(callback), room)
make_broadcaster(id) = _ -> Network.broadcast({~id message=Page.get_value(#entry)}, room)

styled_page(title, body) = (
  Resource.full_page(title, body,
  <link rel="stylesheet" type="text/css" href="resources/css.css" />, {success}, [])
)


user_update(x:mess) = (
  line = <div class="opa-line">
    <div class="opa-wrap"><div class="opa-user">{x.id}:</div>
    <div class="opa-message">{x.message}</div></div>
  </div>
  do exec([#show +<- line ])
  Page.set_scroll_top(Page.get_height(#show)+Page.get_scroll_top(#show), #show)
)


start(connexion) = (
   id = String.sub(0, 8, Server.get_user(connexion) ? "Unknown")
   broadcast = make_broadcaster(id)
   styled_page("Chat",  //Display
     <script onload={_ -> connect()}/>
     <div id="header"><div id="logo"></div></div>
     <div id="show"></div>
     <input id="entry"/>
     <div class="opa-button" onclick={broadcast}>Send!</div>
   )
)

urls      = parser .* -> start
resources = @static_include_directory("resources")
server    = Server.make(Resource.add_auto_server(resources, urls))
```

Transparent protections

All communications are checked

```
type mess = {id
room = Network
                                                    (mess)
make_broadcaster(id) = _ -> Network.broadcast({~id message=Page.get_value(#entry)}, room)

styled_page(title, body) = (
  Resource.full_page(title, body,
  <link rel="stylesheet" type="text/css" href="resources/css.css" />, {success}, [])
)


user_update(x:mess) = (
  line = <div class="opa-line">
    <div class="opa-wrap"><div class="opa-user">{x.id}:</div>
    <div class="opa-message">{x.message}</div></div>
  </div>
  do exec([#show +<- line ])
  Page.set_scroll_top(Page.get_height(#show)+Page.get_scroll_top(#show), #show)
)


start(connexion) = (
   id = String.sub(0, 8, Server.get_user(connexion) ? "Unknown")
   broadcast = make_broadcaster(id)
   styled_page("Chat",  //Display
     <script onload={_ -> connect()}/>
     <div id="header"><div id="logo"></div></div>
     <div id="show"></div>
     <input id="entry"/>
     <div class="opa-button" onclick={broadcast}>Send!</div>
   )
)

urls      = parser .* -> start
resources = @static_include_directory("resources")
server    = Server.make(Resource.add_auto_server(resources, urls))
```

Transparent protections

```
type mess = {id: string; message: string}
room = Network.empty():Network.network(mess)
connect(callback) = Network.add(Session.make_callback(callback), room)
make_broadcaster(id) = _ -> Network.broadcast({~id message=Page.get_value(#entry)}, room)

styled_page(title, body) = (
  Resource.full_page(title, body,
  <link rel="stylesheet" type="text/css" href="resources/css.css" />, {success}, [])
)


user_update(x:mess) = (
  line = <div class="opa-line">
    <div class="opa-wrap"><div class="opa-user">{x.id}:</div>
    <div class="opa-message">{x.message}</div></div>
  </div>
  do exec([#show +<- line ])
  Page.set_scroll_top(Page.get_height(#show)+Page.get_scroll_top(#show), #show)
)


start(connexion) = (
   id = String.sub(0, 8, Server.get_user(connexion) ? "Unknown")
   broadcast = make_broadcaster(id)
   styled_page("Chat",  //Display
     <script onload={_ -> connect()}/>
     <div id="header"><div id="logo"></div></div>
     <div id="show"></div>
     <input id="entry"/>
     <div class="opa-button" onclick={broadcast}>Send!</div>
   )
)

urls      = parser .* -> start
resources = @static_include_directory("resources")
server    = Server.make(Resource.add_auto_server(resources, urls))
```

Transparent protections

```
type mess = {id: string; message: string}
room = Network.empty():Network.network(mess)
connect(callback) = Network.add(Session.make_callback(callback), room)
make_broadcaster(id) = _ -> Network.broadcast({~id message=Page.get_value(#entry)}, room)

styled_page(title, body) = (
  Resource.full_page(title, body,
  <link rel="stylesheet" type="text/css" href="resources/css.css" />, {success}, [])
)


user_update(x:mess) = (
  line = <div class="opa-line">
    <div class="opa-wrap"><div class="opa-user">{x.id}:</div>
    <div class="opa-message">{x.message}</div></div>
  </div>
  do exec([#show +<- line ])
  Page.set_scroll_top(Page.get_height(#show)+Page.get_scroll_top(#show), #show)
)


start(connexion) = (
   id = String.sub(0, 8, Server.get_user(connexion) ? "Unknown")
   broadcast = make_broadcaster(id)
```

```
styled_page("Chat",  //Display
  <script onload={_ -> connect()} />
  <div id="header"><div id="logo">
  <div id="show"></div>
  <input id="entry"/>
  <div class="opa-button" onclick={broadcast}>Send!</div>
)
```

Per-user capability?*

```
server     = Server.make(Resource.add_auto_server(resources, urls))
```

Transparent protections

```
type mess = {id: string; message: string}
room = Network.empty():Network.network(mess)
connect(callback) = Network.add(Session.make_callback(callback), room)
make_broadcaster(id) = _ -> Network.broadcast({~id message=Page.get_value(#entry)}, room)

styled_page(title, body) = (
  Resource.full_page(title, body,
  <link rel="stylesheet" type="text/css" href="resources/css.css" />, {success}, [])
)


user_update(x:mess) = (
  line = <div class="opa-line">
    <div class="opa-wrap"><div class="opa-user">{x.id}:</div>
    <div class="opa-message">{x.message}</div></div>
  </div>
  do exec([#show +<- line ])
  Page.set_scroll_top(Page.get_height(#show)+Page.get_scroll_top(#show), #show)
)


start(connexion) = (
   id = String.sub(0, 8, Server.get_user(connexion) ? "Unknown")
   broadcast = make_broadcaster(id)
   styled_page("Chat",  //Display
     <script onload={_ -> connect()}/>
     <div id="header"><div id="logo"></div></div>
     <div id="show"></div>
     <input id="entry"/>
     <div class="opa-button" onclick={broadcast}>Send!</div>
   )
)

urls      = parser .* -> start
resources = @static_include_directory("resources")
server    = Server.make(Resource.add_auto_server(resources, urls))
```

More

A1-Injection	A6-Security Misconfiguration
A2-Cross Site Scripting (XSS)	A7-Insecure Cryptographic Storage
A3-Broken Authentication and Session Management	A8-Failure to Restrict URL Access
A4-Insecure Direct Object References	A9-Insufficient Transport Layer Protection
A5-Cross Site Request Forgery (CSRF)	A10-Unvalidated Redirects and Forwards

More

A1-Injection	A6-Security Misconfiguration
A2-Cross Site Scripting (XSS)	A7-Insecure Cryptographic Storage
A3-Broken Authentication and Session Management	A8-Failure to Restrict URL Access
A4-Insecure Direct Object References	A9-Insufficient Transport Layer Protection
A5-Cross Site Request Forgery (CSRF)	A10-Unvalidated Redirects and Forwards

More

A1-Injection	A6-Security Misconfiguration
A2-Cross Site Scripting (XSS)	A7-Insecure Cryptographic Storage
A3-Broken Authentication and Session Management	A8-Failure to Restrict URL Access
A4-Insecure Direct Object References	A9-Insufficient Transport Layer Protection
A5-Cross Site Request Forgery (CSRF)	A10-Unvalidated Redirects and Forwards

What Automated Solutions Miss

- Theoretical
  - Logic flaws (business and application)
  - Design flaws
  - The Stupid
- Practical
  - Difficulty interacting with Rich Internet Applications (RIA)
  - Complex variants of common attacks (SQL Injection, XSS, etc)
  - Cross-Site Request Forgery (CSRF)
  - Uncommon or custom infrastructure
  - Authorization enforcement
  - Abstract information leakage

What Automated Solutions Miss

other

- Theoretical
  - Logic flaws (business and application) (in progress)
  - Design flaws
  - The Stupid (what we do best)
- Practical
  - ~~Difficulty interacting with Rich Internet Applications~~ (RIA) (been there, done that)
  - Complex variants of common attacks (~~SQL Injection~~, ~~XSS~~, etc) (been there, done that)
  - Cross-Site Request Forgery (CSRF) (in progress)
  - ~~Uncommon or custom infrastructure~~ (abstract&specify them away!)
  - ~~Authorization enforcement~~ (been there, done that)
  - ~~Abstract information leakage~~ (been there, done that)

OPA: Language Support for a Sane, Safe and Secure Web

- General approach
- Keeping in the ~~Smart~~ Useful
- Filtering out the ~~Stupid~~ Fragile
- Lessons and Future

OPA: Language Support for a Sane, Safe and Secure Web

- General approach
- Keeping in the ~~Smart~~ Useful
- Filtering out the ~~Stupid~~ Fragile
- Lessons and Future

Theoretical foundations

- Building on 30+ years of research in formal methods & language theory.
- Key for precise analysis.
- Key for precise optimizations.

False positives

- False positives are annoying but not that bad -- as long as they are predictable and there's a workaround.
- Experienced developers make safety/security mistakes quite often.
- False positives often later reveal themselves as **true** positives.

Future

- Eliminate CSRF.
- Extend per-application security policy.
- Extend per-library/per-data safety policy.
- Plenty of additional features!
- Hiring you?

That’s all, folks!

http://www.mlstate.com

That’s all, folks!

Thanks and apologies to David Byrne & Charles Anderson
for hijacking their slides

http://www.mlstate.com