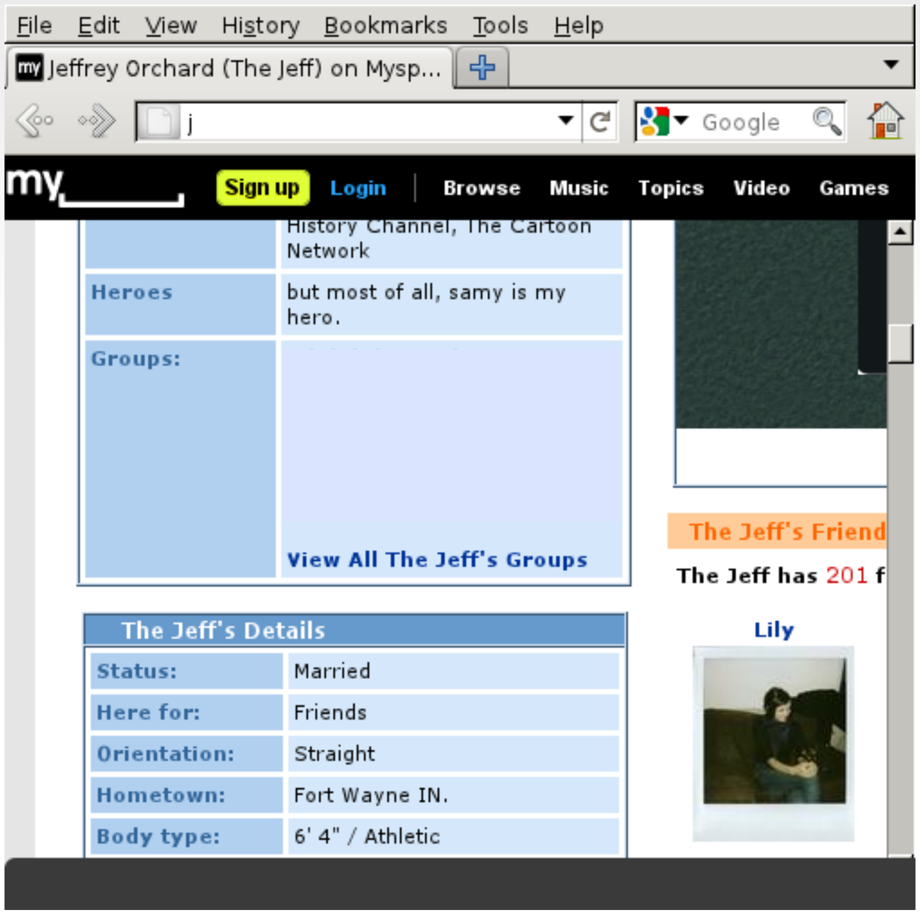
Towards Client Side HTML Security Policies

Joel Weinberger, Adam Barth, Dawn Song

MySpace



File Edit View History Bookmarks Tools Help
Jeffrey Orchard (The Jeff) on Mysp...
Google
Sign up
Login
Browse
Music
Topics
Video
Games

History Channel, The Cartoon Network

Heroes	but most of all, samy is my hero.
Groups:	View All The Jeff's Groups

The Jeff's Friend

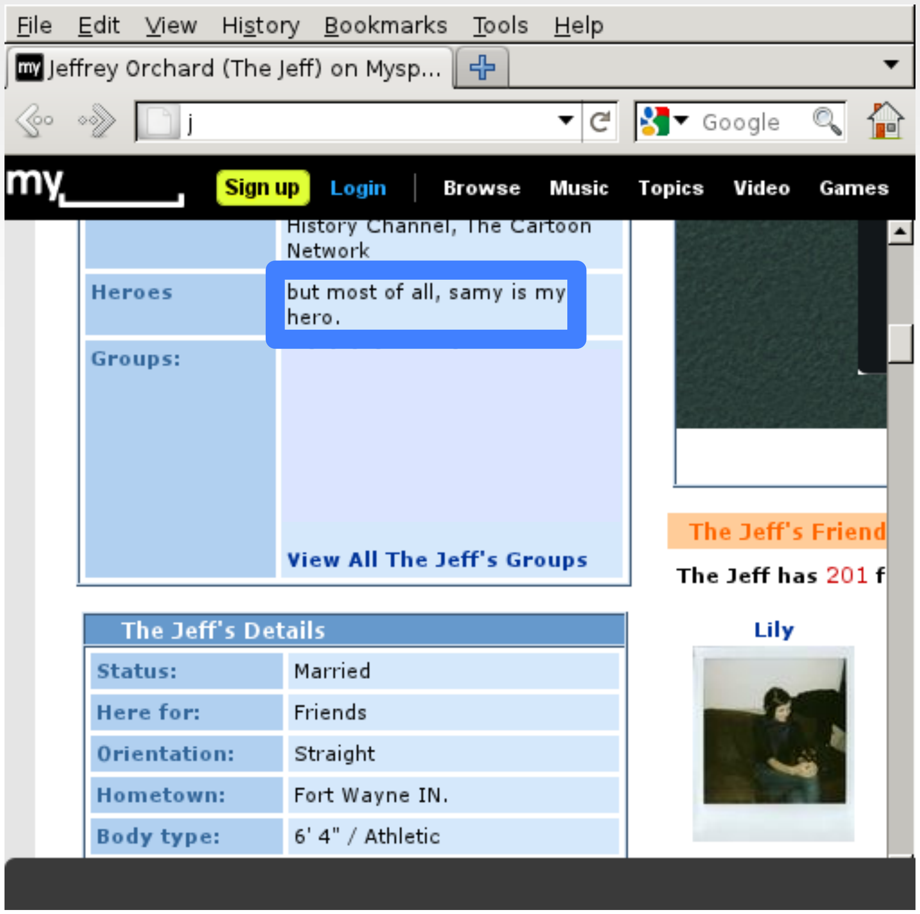
The Jeff has 201 f

Lily

The Jeff's Details

Status:	Married
Here for:	Friends
Orientation:	Straight
Hometown:	Fort Wayne IN.
Body type:	6' 4" / Athletic

Samy Worm


File Edit View History Bookmarks Tools Help
Jeffrey Orchard (The Jeff) on Mysp...
Google
Sign up
Login
Browse
Music
Topics
Video
Games

	History Channel, The Cartoon Network
Heroes	but most of all, samy is my hero.
Groups:	View All The Jeff's Groups

The Jeff's Details

Status:	Married
Here for:	Friends
Orientation:	Straight
Hometown:	Fort Wayne IN.
Body type:	6' 4" / Athletic

The Jeff's Friend

The Jeff has 201 f

Lily

Content Injection

The insertion of untrusted data, structure, or code into an application

Key Points

- Explicit policies form a compelling, unique point in the content injection protection design space
- The current trade-offs in explicit policy systems make none of the current systems completely viable
- Explicit policies are the way forward, but we need new system designs

Content Injection

```
<html>
     <h1>Forum Post #1</h1>
     <div>
     This is the content of the post.
     </div>
</html>
```

Content Injection

```
<html>
     <h1>Forum Post #1</h1>
     <div>
     <script>alert(document.cookie);</script>
     </div>
</html>
```

Policies

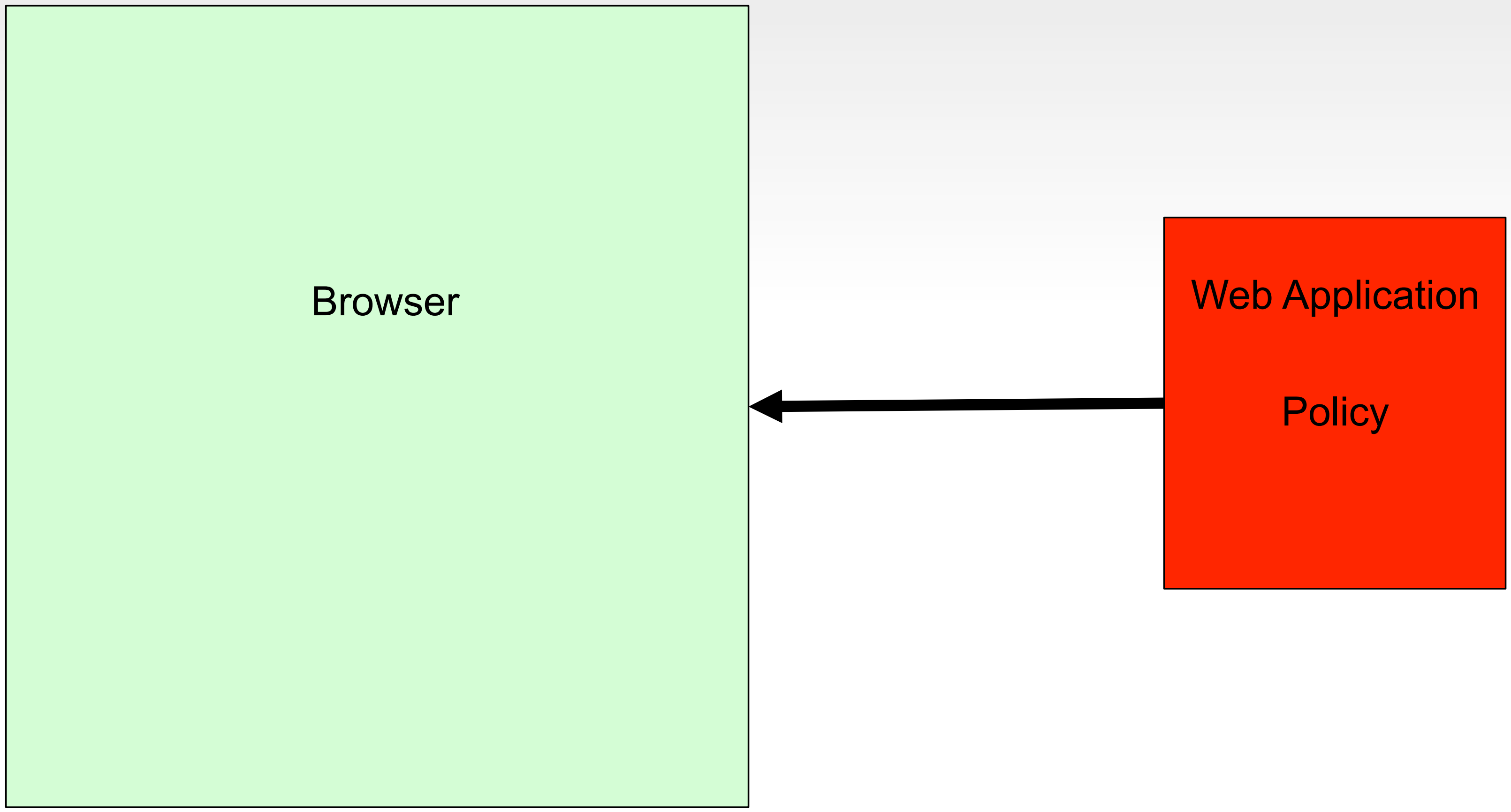
```
    Trusted
<html>
                                         Trusted
     <h1>Forum Post #1</h1>
       Trusted
     <div>
                                                          Untrusted
     <script>alert(document.cookie);</script>

     </div>

</html>
```

Web Application Frameworks

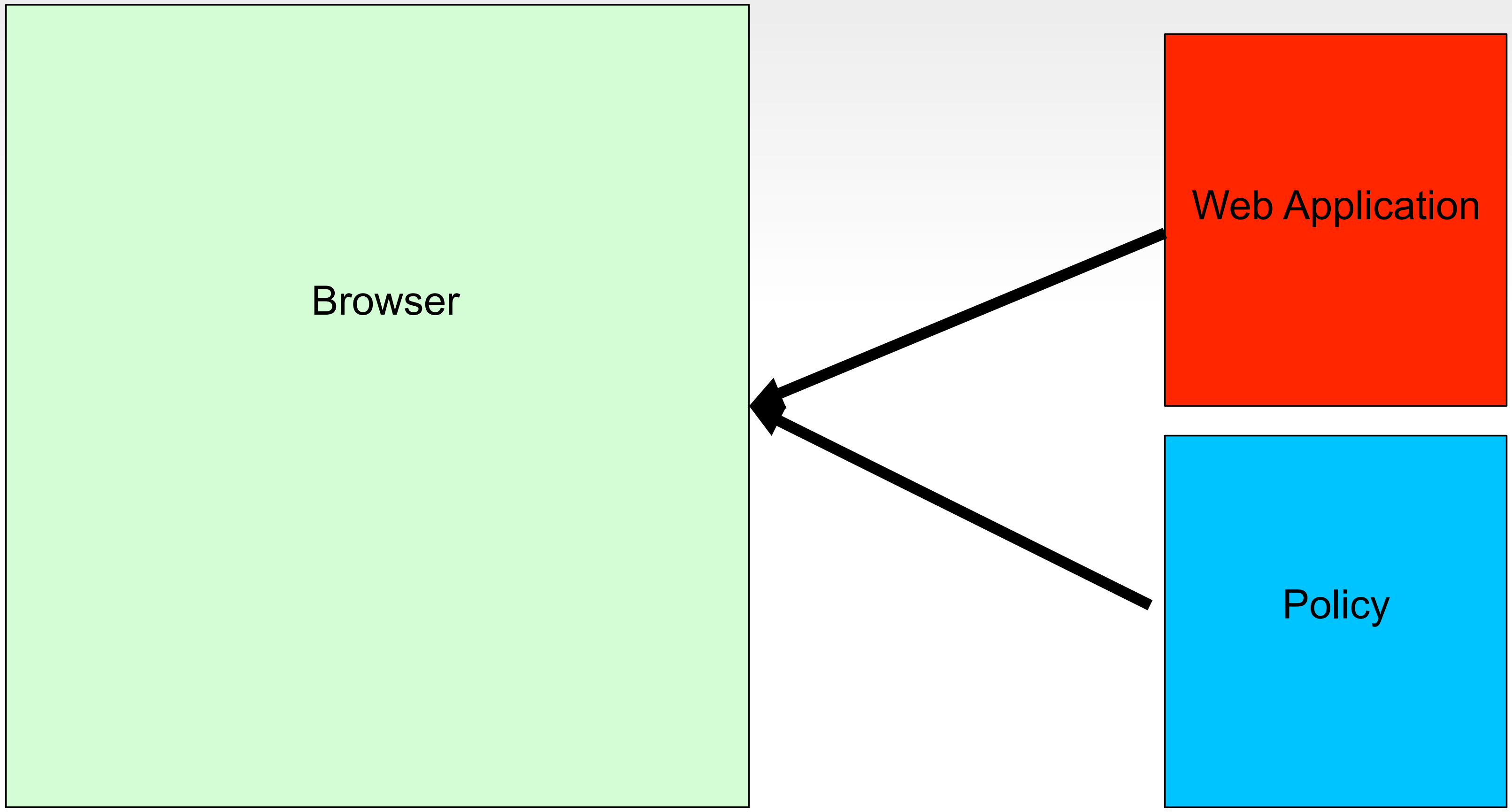
- Systems for writing web applications
- Frameworks provide tools for *sanitizing* content
- Turns out, **sanitization is hard**
  - Shameless plug for our ESORICS 2011 paper:

A Systematic Analysis of XSS Sanitization in Web Application Frameworks

Implicit Policies



Explicit Policies



Key Points

- ✓ Explicit policies form a compelling, unique point in the content injection protection design space

- The current trade-offs in explicit policy systems make none of the current systems completely viable

- Explicit policies are the way forward, but we need new system designs

Explicit Policy Systems

- BEEP
- BLUEPRINT
- Content Security Policy (CSP)

BEEP

- Hashes of allowed scripts
- Performance: good
- Dynamic scripts are **very** hard to get right
- Only XSS

BLUEPRINT

- Structural description of page, enforced by JavaScript library
- Performance: poor
- Does not trust the browser's parser
- Very fine grained granularity

Content Security Policy (CSP)

- Specify allowed behaviors of page
- Performance: ?
- Only handles some content injection
- Coarse grained
  - What is the affect on how applications are written?

Applying CSP to Applications

- How does CSP affect Web applications?
- Apply CSP to Bugzilla and HotCRP
- Measure performance of applications and how the applications were changed

CSP Study

- **Developer effort to retrofit applications to be CSP compatible is large**
- Template variables cannot be used in scripts
- Need to lookup data through JavaScript
- Template logic no longer affects scripts

CSP Study

Bugzilla

Page	No Inline JS	Async JS
index.cgi	14.8%	-3.0%
editsettings.cgi	6.3%	5.1%
enter_bug.cgi	57.6%	44.2%
show_bug.cgi	51.5%	4.0%

HotCRP

Page	No Inline JS	Async JS
index.php	45.3%	37.2%
search.php	52.9%	50.4%
settings.php	23.3%	16.1%
paper.php	61.1%	58.5%
contacts.php	67.8%	35.5%

Key Points

- ✓ Explicit policies form a compelling, unique point in the content injection protection design space
- ✓ The current trade-offs in explicit policy systems make none of the current systems completely viable
- Explicit policies are the way forward, but we need new system designs

Explicit Policies: The Good and the Bad

- Provide a separation policy from application
  - Not doing this makes security hard
- Simple or complex: you choose
- Not good at performance *and* developer usability

Towards the Future

- Policy systems are useful and *should be how we approach content injection*
- CSP has some great properties, but *suffers when applied to current applications*
- How can we *combine features from these different systems?*

Key Points

- ✓ Explicit policies form a compelling, unique point in the content injection protection design space
- ✓ The current trade-offs in explicit policy systems make none of the current systems completely viable
- ✓ Explicit policies are the way forward, but we need new system designs