Architectural Principles for Safe Web Programs
Charlie Reis, Steve Gribble, Hank Levy
University of Washington
HotNets VI - November 14, 2007
The Shift to Web Programs

- Browsers were built to render HTML
- We’ve tacked on **active code**
- Now running **web programs** within the browser
- Browser must act like OS

- JavaScript
- Flash
- Silverlight
Web Programs are Unsafe

**Malicious Extensions**

**XSS Attacks**
[Jim 07, Jovanovic 06, Kirda 06, Ismail 04, Huang 03/04]

**Browser Exploits**
[Reis 06, Cox 06]

**AJAX Worms**
[Grossman 06]

**Cache Timing Attacks**
[Johns 06]

**DNS Rebinding Attacks**
[Jackson 07]

**Drive-By Downloads**
[Moshchuk 07]

**Need safe architectural principles**

Symptoms of four fundamental problems

**Visited Link History**
[Jackson 06]
Can’t identify program boundaries

- Which pages can talk?
- Same Origin Policy
- Flawed approach:
  - Too narrow
  - Too broad
  - Easily compromised

- 128.1.2.3 (Internet)
- 10.1.1.1 (LAN)

 evil.com

MySpace
MySpace
Google Maps
Can’t prevent unwanted code

- Scripts injected via user input (XSS)
- Scripts injected in-flight
- Pages request data via code files
Can’t isolate programs in browser

- Side channels
- Can abuse credentials of other sites (CSRF)
- Failures, resource contention
Can’t apply uniform policies

- Each content type has its own security model
- No restrictions on browser extensions
- Can’t reason about a web program’s abilities
Summary of Problems

1. Can’t identify **program boundaries**
2. Can’t prevent **unwanted code**
3. Can’t **isolate programs** in browser
4. Can’t apply **uniform policies**
Architectural Principles
Principles for Web Programs

- Browsers don’t know what a program is
- Support web programs as first class entities
  - Must improve both program definitions and browsers
Program Boundaries

- New abstractions:
  - Web program
  - Program instance
- Must explicitly assign resources to programs
Specifying Boundaries

- One solution: **key pair**
  - Author holds private key
  - Web program consists of signed pages
  - No PKI required
Authorized Code

- Explicitly authorize all code in a web program
- Whitelists (e.g., BEEP)
- Code restrictions (e.g., MashupOS)
- Safe data requests (e.g., JSONRequest)

**Whitelist:**
364700f3feb405918a881...
34e523012ce91e1e6956b...

**Sandbox**

```html
<script>...
</script>
```
Program Isolation

- **Privacy:**
  - Partition storage between *programs*
  - Isolate credentials between *instances*

- **Robustness:**
  - OS process for each *instance*
Security Policies

- Sandbox content types & browser extensions
- Interposition layer
- Extensible policies
Conclusion

- **Web programs** need first class support

  - Explicit boundaries, authorized code, isolation, uniform policies

  - Improve program definitions and browsers

  - Web can be a safe platform

http://www.cs.washington.edu/homes/creis/