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

Web Programs are Unsafe

Malicious Extensions

Browser Exploits
[Reis 06, Cox 06]

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

CSRF Attacks
[Johns 06]

AJAX Worms
[Grossman 06]

Symptoms of four fundamental problems

Need safe architectural principles

Cache Timing Attacks
[Felten 00]

Browser Crashes

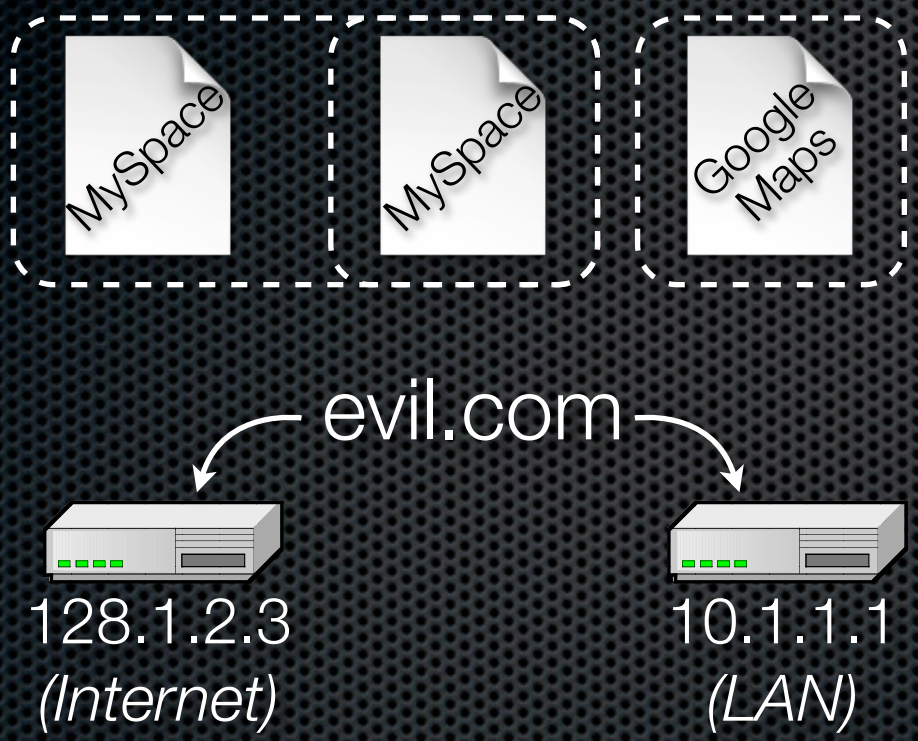
DNS Rebinding Attacks
[Jackson 07]

Drive-By Downloads
[Moshchuk 07]

Visited Link History
[Jackson 06]

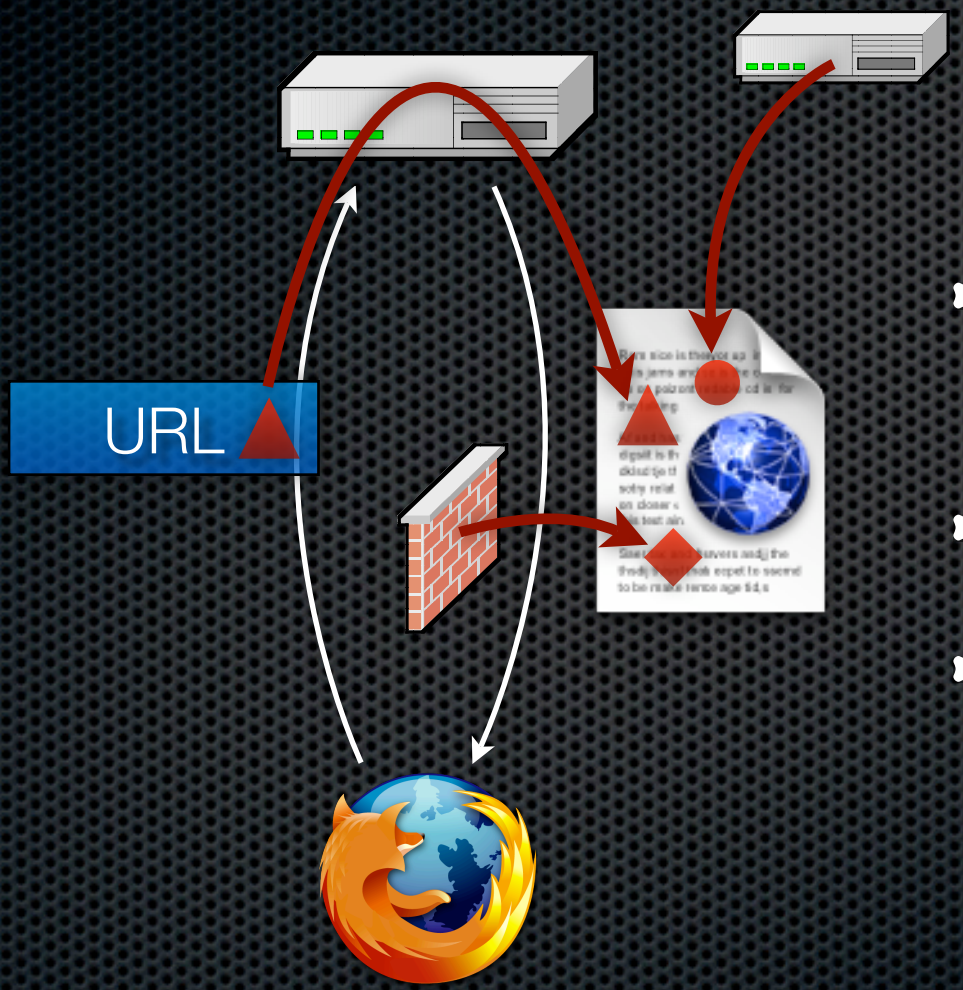
Fundamental Problems

1 Can't identify program boundaries



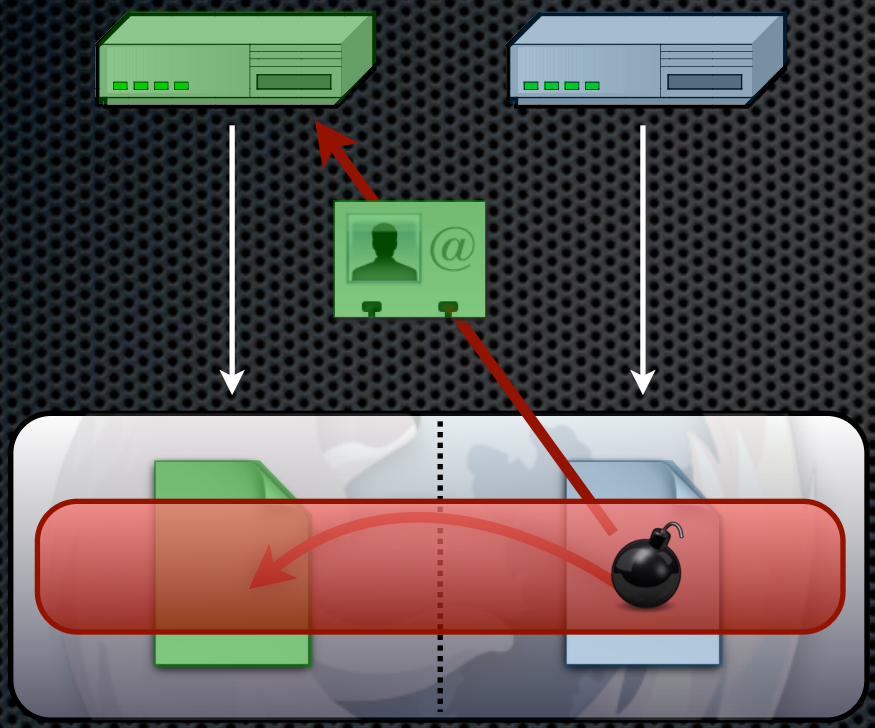
- Which pages can talk?
- **Same Origin Policy**
- Flawed approach:
  - Too narrow
  - Too broad
  - Easily compromised

2 Can’t prevent unwanted code



- Scripts injected via user input (XSS)
- Scripts injected in-flight
- Pages request data via code files

3 Can’t isolate programs in browser



- Side channels
- Can abuse credentials of other sites (CSRF)
- Failures, resource contention

4 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

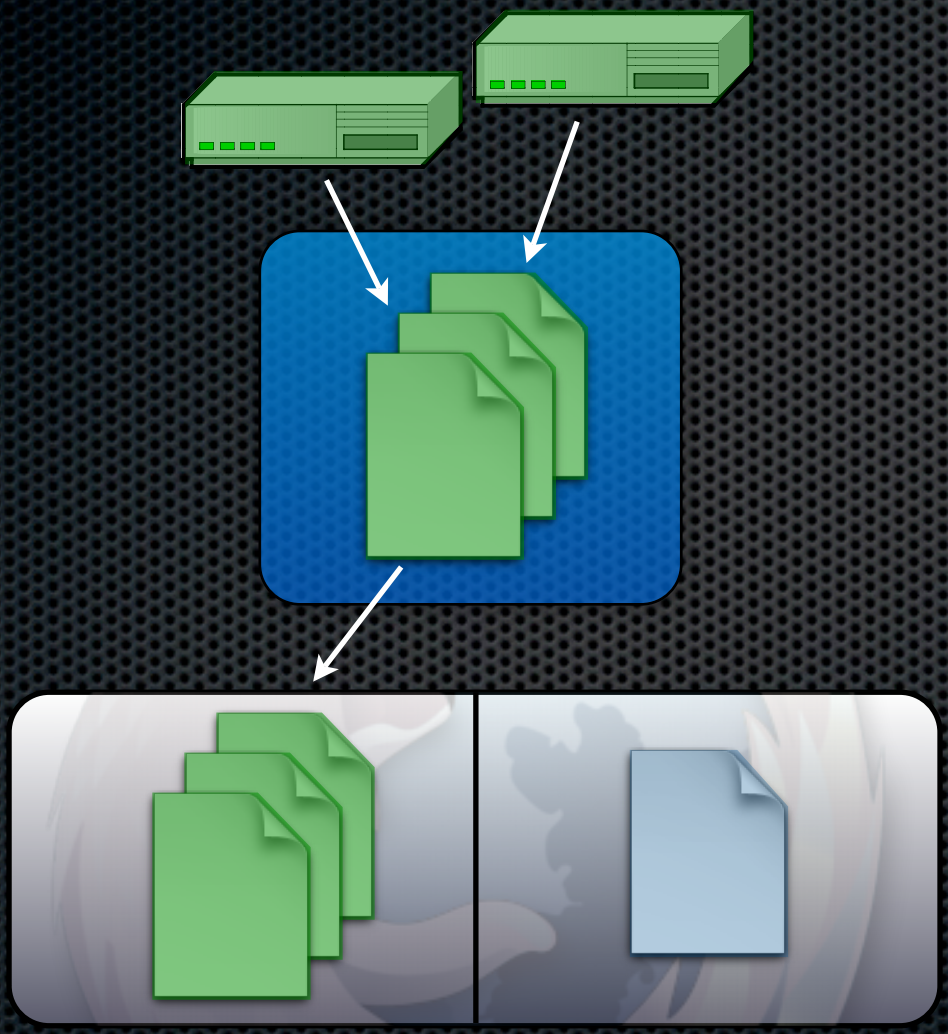
1. Can't identify **program boundaries**
2. Can't prevent **unwanted code**
3. Can't **isolate programs** in browser
4. Can't apply **uniform policies**

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*

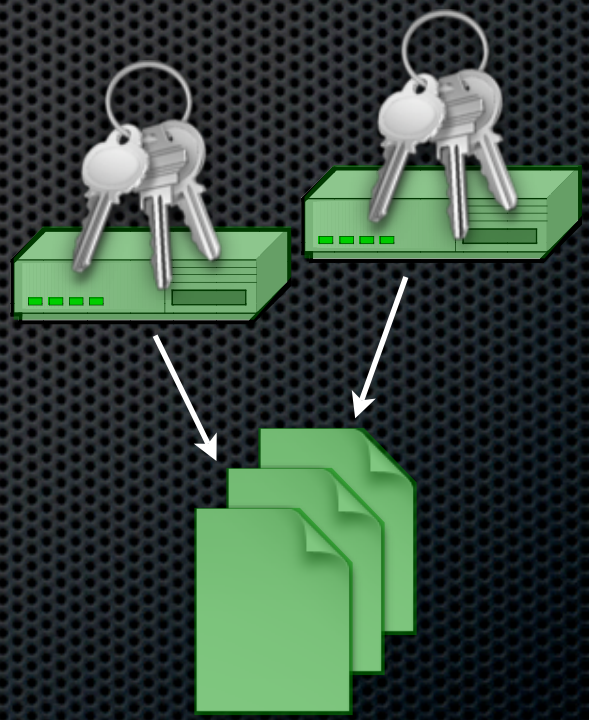
1 Program Boundaries



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

Specifying Boundaries

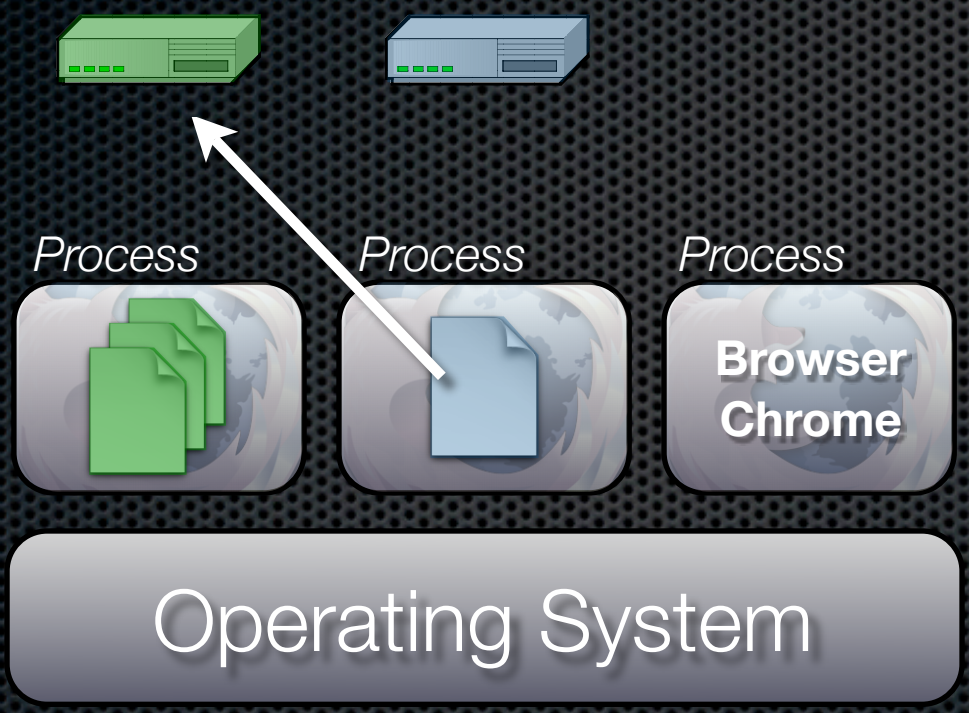
- One solution: **key pair**
  - Author holds private key
  - Web program consists of signed pages
  - No PKI required



2 Authorized Code

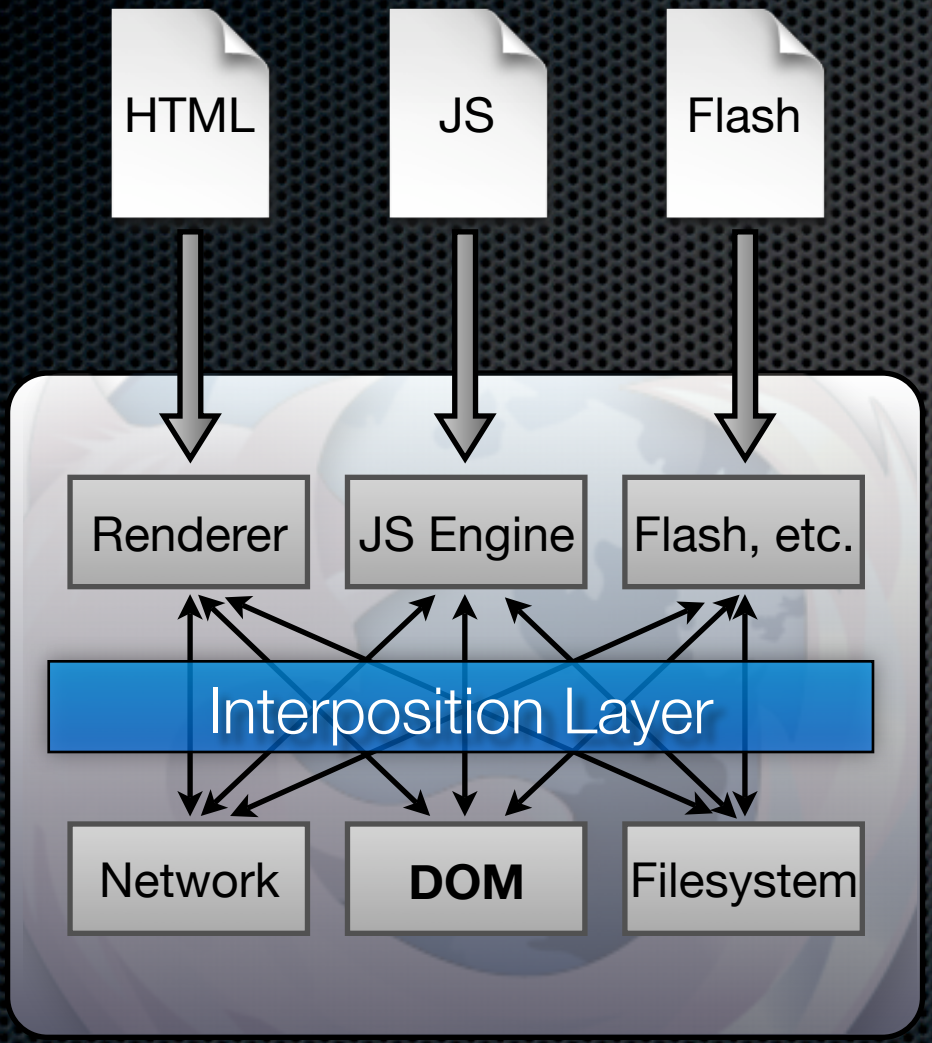
- Explicitly authorize all code in a web program
  - Whitelists *(e.g., BEEP)*
  - Code restrictions *(e.g., MashupOS)*
  - Safe data requests *(e.g., JSONRequest)*

3 Program Isolation



- **Privacy**:
  - Partition storage between *programs*
  - Isolate credentials between *instances*
- **Robustness**:
  - OS process for each *instance*

4 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/