# A Resource Management Approach to Web Browser Security

**Jun Li**

**University of Oregon**

**Carlos III Univ of Madrid**

**Institute IMDEA Networks**

**Dongting Yu, Luke Maurer**

**University of Oregon**

ICNC, 1/31/2012

UNIVERSITY OF OREGON

# Problem Statement

- A web browser is no longer a client for downloading and rending *static* content from web sites.
- It has become a common environment shared by *principals* from different origins.
  - Every principal can be a frame with JavaScript, or a plug-in
- However, there is no resource management of these principals: a principal can access resources of other principals.

# Examples

- A malicious frame can cause another frame to navigate a phishing site.
- A malicious gadget in a mashup site (e.g., iGoogle) can replace another benign gadget with a spoofed one.
- A malicious web site www.malicious.com can invoke the browser to send a request to another web site www.honest.com in the name of the user, effectively impersonating the user by "hijacking" the browser.

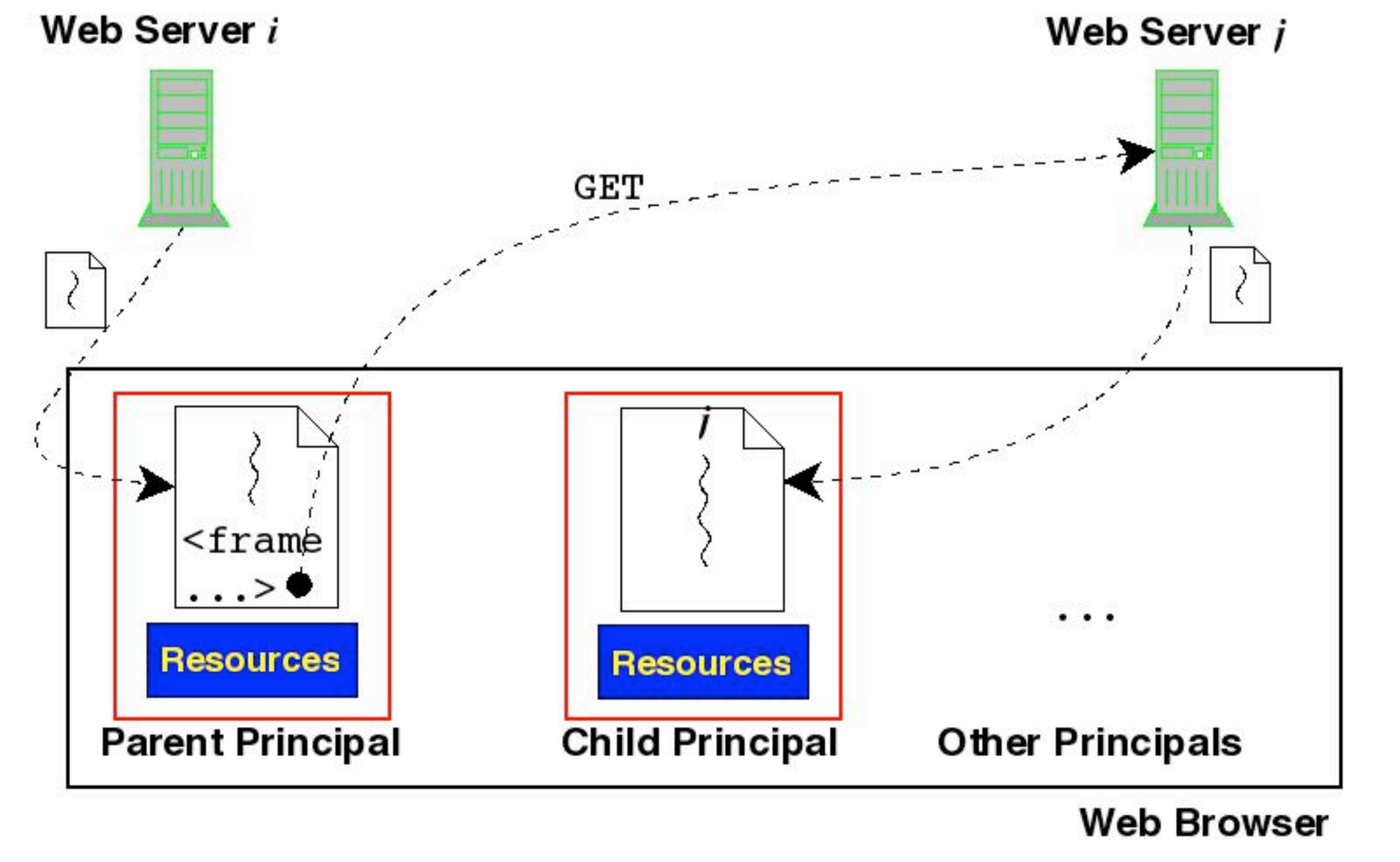
# In Contrast ...

- A modern operating system can separate processes cleanly, and every process has its own logical address space.
- A process can access a system resource only if it is explicitly made available.

# A Resource Management Approach

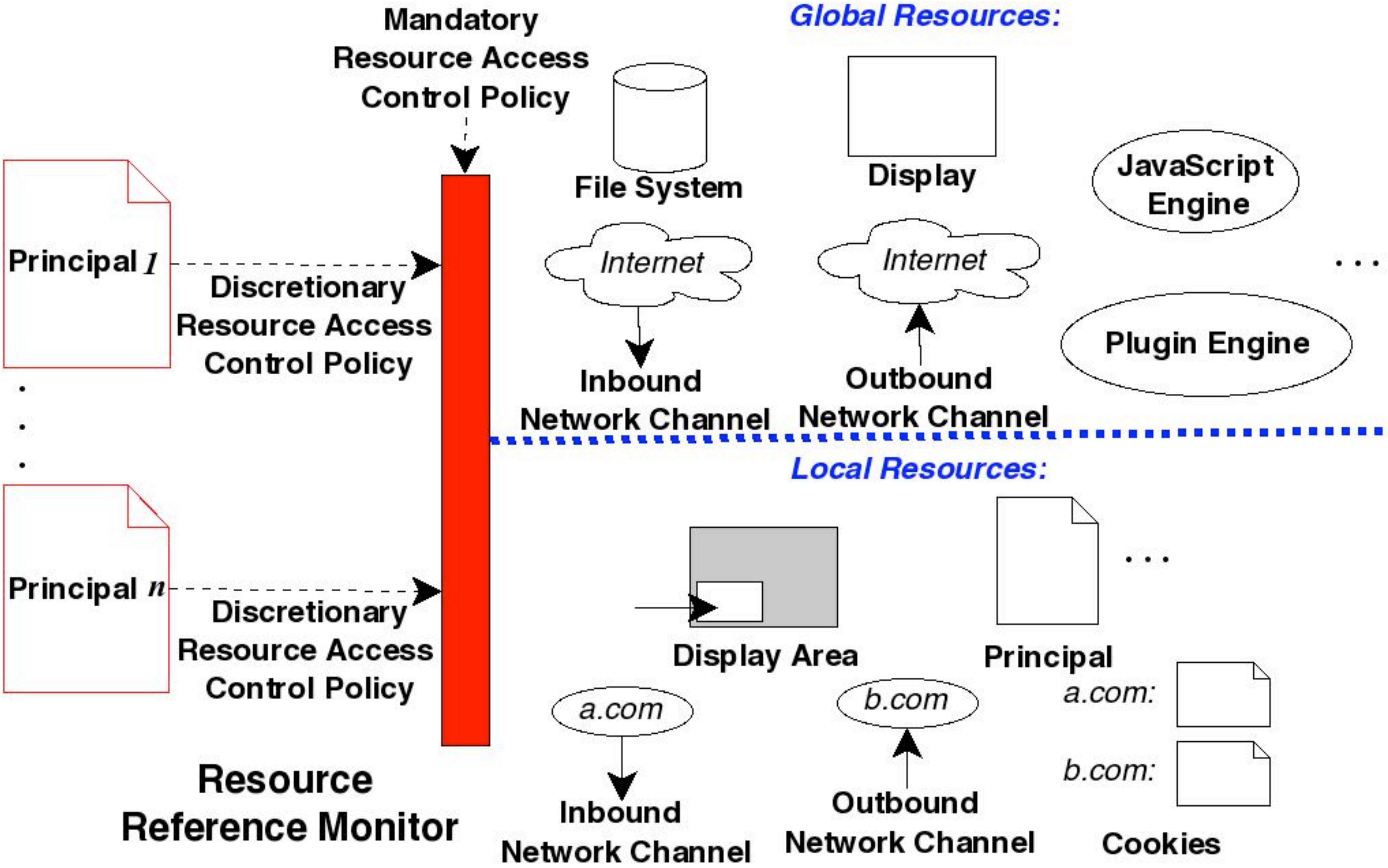
- Just as an OS is a resource allocator for processes, a web browser should also be a resource allocator for its principals.
- Every principal must be isolated and protected from each other.
- The web browser must support a *reference monitor* concept to systematically enforce resource access control and protect inter-principal interaction and communication.

# Resource Allocation Model



# Resource Management Framework

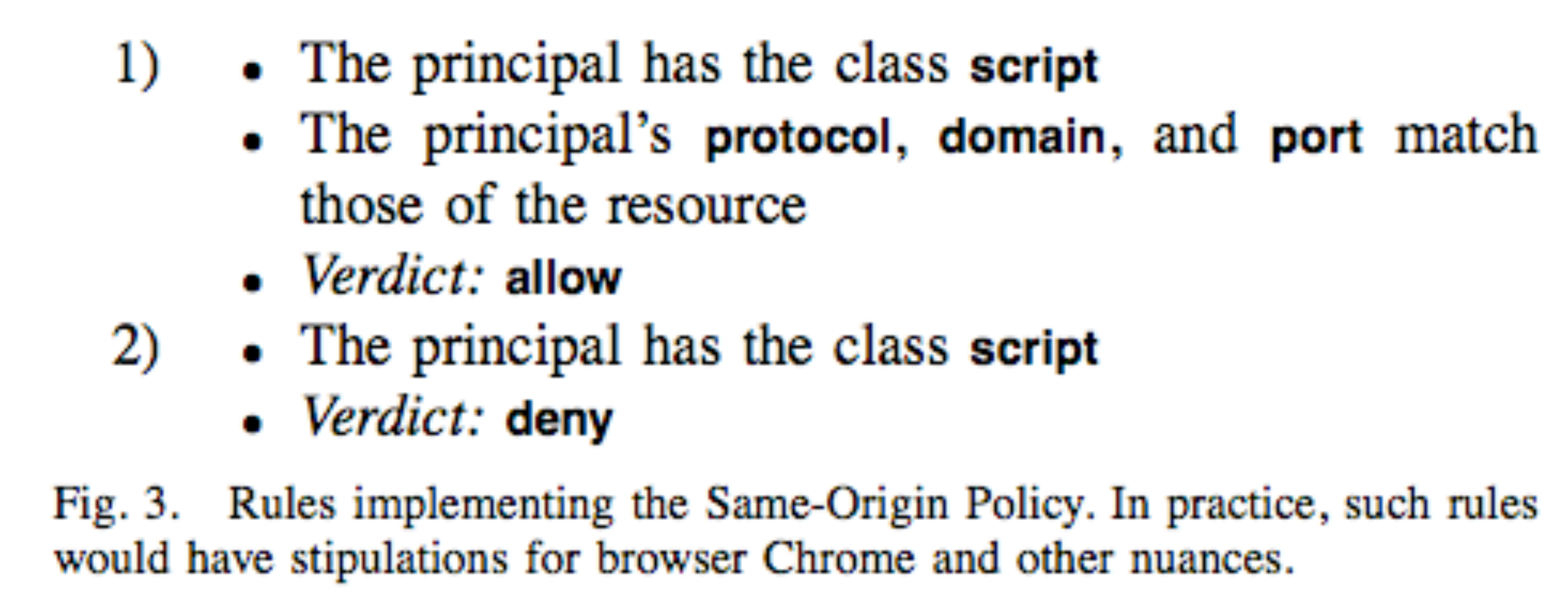
- Mandatory access control at the web browser level
  - What to allow, what not, under what conditions
  - Applies to all principals based on a browser's configuration
- Discretionary access control specified by relevant principals
  - How other principals (e.g. a child principal) can access its resources


Mandatory
Resource Access
Control Policy
Global Resources:
File System
Display
JavaScript Engine
Principal 1
Discretionary
Resource Access
Control Policy
Internet
Internet
Inbound
Network Channel
Outbound
Network Channel
Plugin Engine
. . .
Local Resources:
. . .
Principal n
Discretionary
Resource Access
Control Policy
Display Area
Principal
a.com
b.com
a.com:
b.com:
Inbound
Network Channel
Outbound
Network Channel
Cookies
Resource
Reference Monitor

# Resource Policy Language Design

- Three types of objects: principal, resource, and action.
- Rules: Whether a principal can take specific actions on a resource.
- Properties of a resource: class, type, protocol, domain, port, path, document, parent.
- Properties of an action: class, protocol, security.

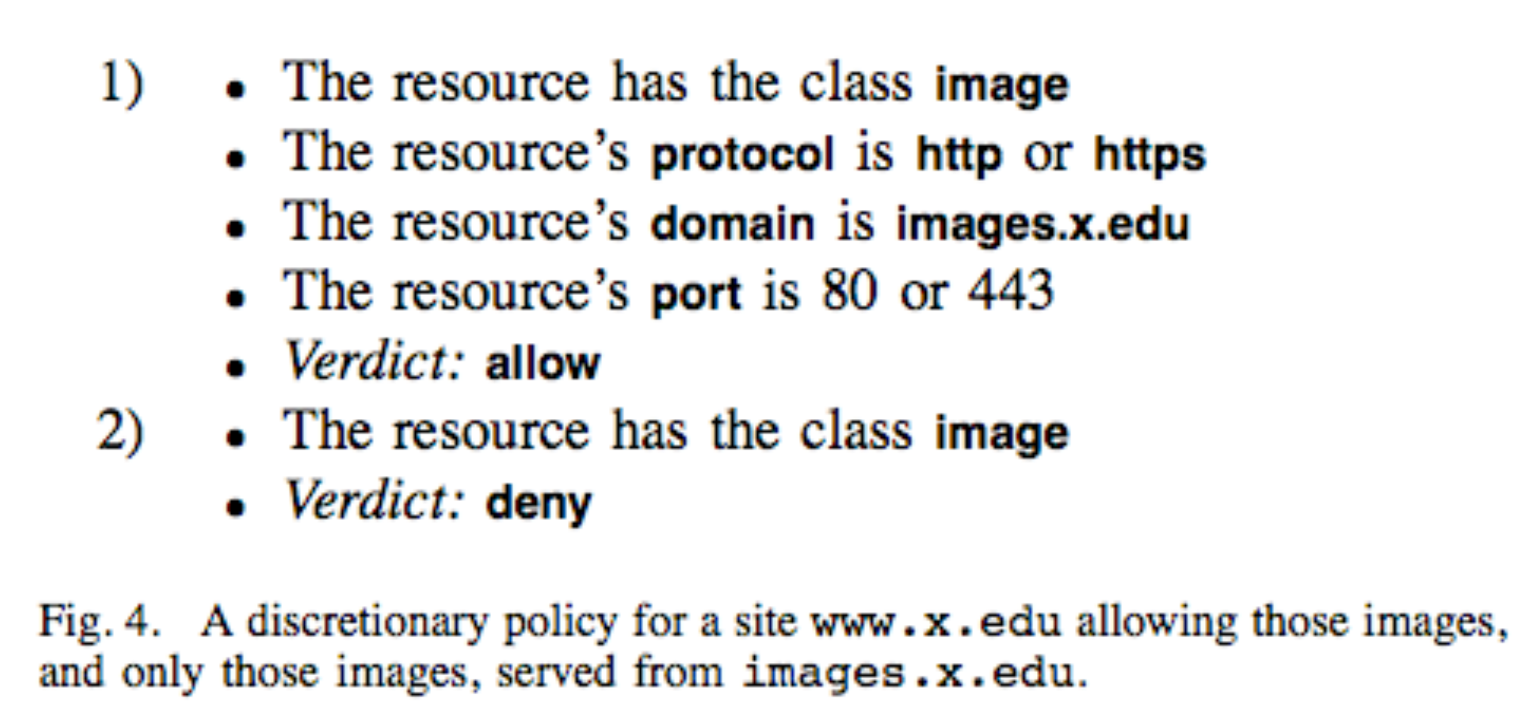
# Examples



1)
- The principal has the class **script**
- The principal's **protocol**, **domain**, and **port** match those of the resource
- *Verdict:* **allow**

2)
- The principal has the class **script**
- *Verdict:* **deny**

Fig. 3. Rules implementing the Same-Origin Policy. In practice, such rules would have stipulations for browser Chrome and other nuances.



1)
- The resource has the class **image**
- The resource's **protocol** is **http** or **https**
- The resource's **domain** is **images.x.edu**
- The resource's **port** is 80 or 443
- *Verdict:* **allow**

2)
- The resource has the class **image**
- *Verdict:* **deny**

Fig. 4. A discretionary policy for a site `www.x.edu` allowing those images, and only those images, served from `images.x.edu`.

# Security Effectiveness

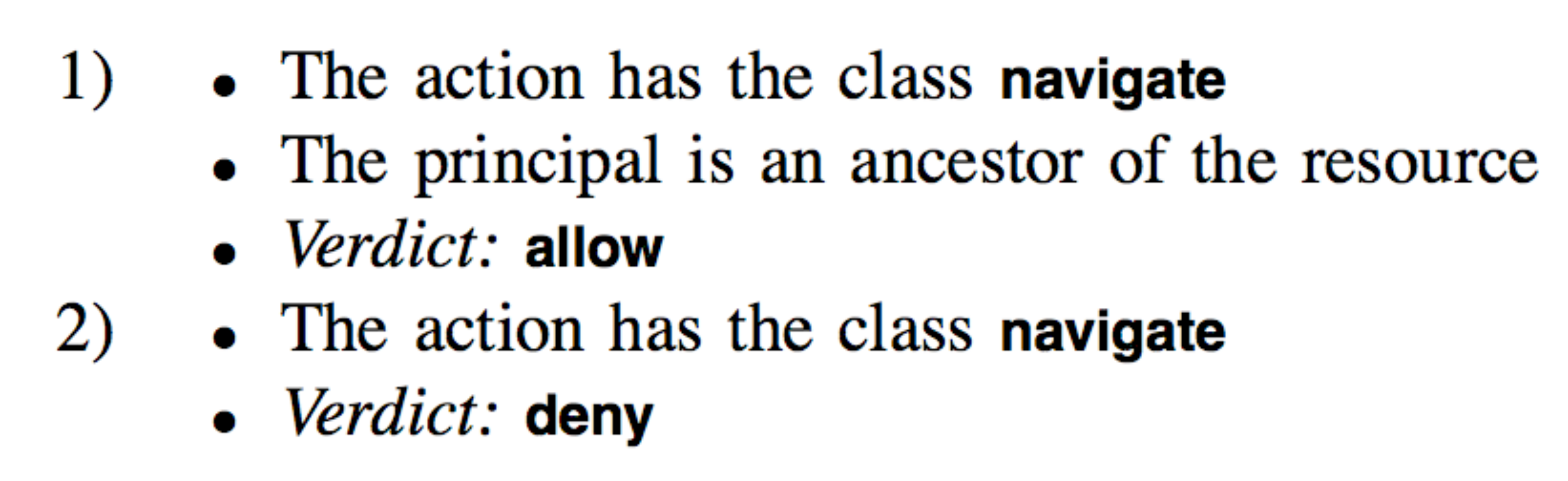
- With the resource management framework in place, all we need is to specify robust security policies to secure web-based activities
- Application examples to web-based attacks:
  - Frame Hijacking
  - Cross-Site Request Forgery (CSRF)
  - DNS Rebinding Attack

# Frame Hijacking

- A frame often contains sub-frames (Google maps, ads, Flickr albums, etc.) from different sources
- One frame can direct another frame to load its content from an arbitrary URL: the *navigate* action
- Dangerous if a malicious frame sends another frame to a phishing site, or replace a gadget with a malicious one

# Frame Hijacking Prevention via Resource Access Control

- A frame is both a principal and a resource.
- Following [BJM 2008], we can specify a policy such that a frame can only navigate its descendants.



1) 
- The action has the class **navigate**
- The principal is an ancestor of the resource
- *Verdict:* **allow**

2) 
- The action has the class **navigate**
- *Verdict:* **deny**

*[BJM2008] A. Barth, C. Jackson, and J. Mitchell, “Securing frame communication in browsers,” in Proc. of the USENIX Security Symposium, 2008, pp. 17–30.*

# Conclusions

- A web browser is not a static content viewer, but a common environment shared by multiple principals from different origins.
- A web browser should be a resource allocator to secure principals from one another and secure web operations.
- We proposed a resource management framework that is general enough for various browsers to implement.

Thank you!

Questions?

Email: lijun@cs.uoregon.edu

UNIVERSITY OF OREGON