Same Origin Policy

- Code can only access data from the same origin
  - Origin: (scheme, domain, port) triple
  - E.g. (http, cseweb.ucsd.edu, 80)

- Typical context: JavaScript executing in browser
  - Main page loads iframe element with content from another site
  - JavaScript from main page cannot access data inside iframe

- Why do we need the same origin policy?
Same Origin Policy

- **Want:** Combine user-specific content from distinct sites
  - iframes, XMLHttpRequest, embedded Java, embedded Flash
- User does not trust sites with others’ data
- Sites (origins) do not trust each other
- Sites must be isolated from each other
SOP as Multilevel Security

- All content (code and data) carries its own label (origin)
  - Can change one’s origin to less specific domain only
- Can write via HTTP request to any origin
- Can read only from same origin
- Can get and set cookie only for same origin

**What is SOP protecting?**
- Secrecy? Integrity? Availability?
Domain Hierarchy

document can change its own origin to less specific domain
SOP as Multilevel Security

- What if Web used BLP model?
- Labels (origins) don’t form a lattice
  - No label that contains both ucsd.edu and google.com
  - No way to specify more than one origin in content
- Must declassify to send to another origin
  - Both sender and receiver must declassify to common domain
From the Aether to the Ethernet

Yossef Oren and Angelos D. Keromytis
USENIX Security 2014
Smart TV

- Can watch broadcast TV
- Can watch streaming content
  - Netflix, YouTube, Hulu, etc.
- Can browse Web

TV-era non-interactive content
Web-era interactive content
HbbTV

- Web content delivered in broadcast television stream
  - HTML, JavaScript
- Content overlaid on broadcast
- User can interact with content
"Tout le monde aime la Chine"

Les invités du jour :
- Pascal Blanchard : Historien, spécialiste du fait colonial.
- Jean-Paul Tchang : Spécialiste de l’économie chinoise.
- Dorian Malovic : Chef du service Asie au quotidien La Croix.

HbbTV

- JavaScript application starts executing automatically when user starts watching channel
- Application can access content from broadcast stream or Internet (via broadband)
HbbTV Security

- Broadcast stream is not authenticated
  - Authentication would require extensive non-existent PKI
- No way to disable application
HbbTV Security

- Broadcast and broadband content together in one place
- Need a security policy to restrict interaction
  - Can broadband content access broadcast content?
  - Can broadcast content access broadband content?
  - Can broadband content access other broadband content?
- HbbTV uses same origin policy
- Broadcast content explicitly specifies its own origin
HbbTV SOP

- Broadcast content explicitly specifies its own origin
- Broadcast application can say “I am facebook.com” and TV browser will allow it to access facebook.com content
- This completely nullifies same origin policy
- Huge security problem
  - Exploitation requires unscrupulous broadcaster
  - Anyone with TV-band transmitter
IFRAMEs
Iframes

- Frame contains content from another origin
  - Origin’s cookies sent when retrieving content
- Framed content cannot be accessed by enclosing document (same origin policy)
- Displayed to user in style specified by enclosing page
- User can interact with framed content
Iframe Security

- Same origin protects direct access by enclosing page
- But enclosing page controls layout and presentation
  - Page position and CSS
- Potential for enclosing page to manipulate presentation of framed content to mislead user
Clickjacking

- **Clickjacking**: Inducing user to click an element of page having side effects that the user does not realize
- Using iframes: obscure UI element in frame so user does not understand what they are clicking on
  - Can also make iframe completely transparent
Likejacking

Prove you’re human, click on “blue”

[Image of two boxes, one blue and one white]
Likejacking
Clickjacking Defenses

- Don’t allow transparent iframes?
  - How transparent is transparent?
  - Can magnify and show 1 pixel of content.

- Don’t allow resizing iframes?
  - Load lots of identical frames

- Don’t allow frames?
  - Web will stop working
Clickjacking Defense

- **Framebusting**: go to framed page ("busting out" of frame)
- Could also prevent content from rendering in iframe
- Preferred defense against clickjacking
- **Simple fragment**
  
  http://en.wikipedia.org/wiki/Framekiller

  ```javascript
  <script type="text/javascript">
  if(top != self) top.location.replace(location);
  </script>
Clickjacking defense straightforward once you know it
But why is clickjacking a problem in the first place?
Who is to blame for it?
How It Happened

- Initially Web pages were just static content
- Persistent sessions allowed interactive sites
- Original frames (using `<frame>`) allowed split-pane sites
  - Very ugly; will not show screenshot example
  - User still clearly interacting with original site
- Iframes: site embedded in page
  - A better, much less ugly frame
How It Happened

- **Original security assumption:** users can see site with which they are interacting

- A series of small feature additions erode assumption
  - Each apparently innocent

- Where did we go wrong?

- How do we prevent it?