THE DEPUTIES ARE STILL CONFUSED

RICH LUNDEEN

Hi my name is Rich

- I have a twitter @webstersprodigy
- I have a website http://webstersprodigy.net

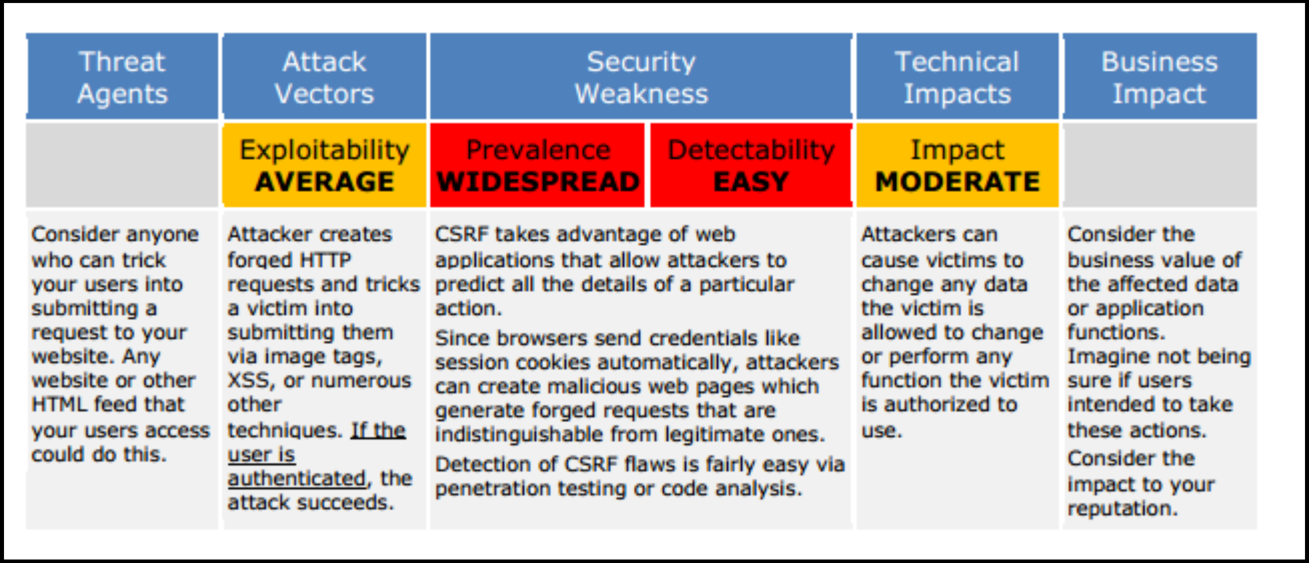
What is the same origin policy?

- Simple answer: content from one website should not (usually) be able to access or modify content on another website
  - Even with frames, tabs, etc.
  - A lot of web vulnerabilities happen when websites inadvertently allow cross site access
- Crypto Rule #1 – never invent your own
- Does this rule apply to all security?
- Unfortunately, this is easier said than done... (for crypto too)

Between the browser tabs

- Advanced CSRF Attacks
  - Forcing cookies
  - OAuth
  - Other interesting issues
- Clickjacking
  - BeEf clickjacking module
  - X-FRAME-OPTIONs Edge Cases

CSRF: Detectability Easy

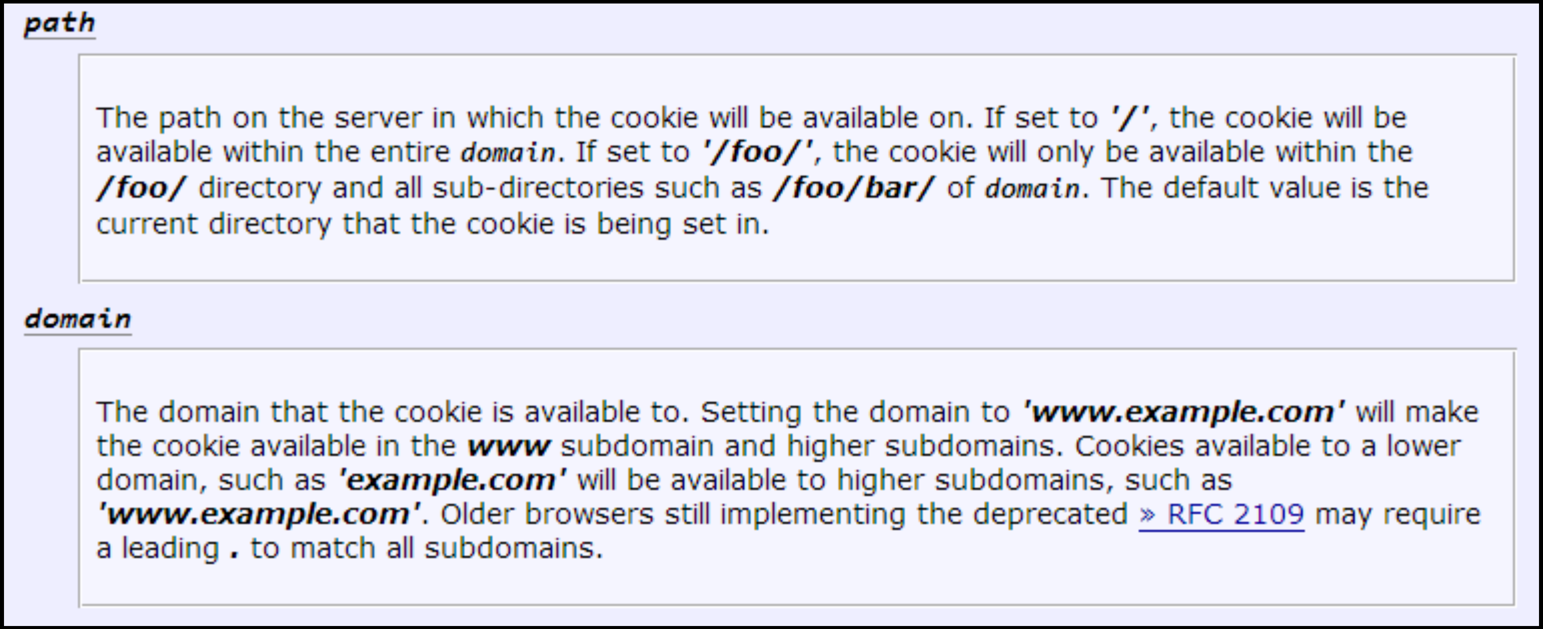


Threat Agents	Attack Vectors	Security Weakness		Technical Impacts	Business Impact
	Exploitability **AVERAGE**	Prevalence **WIDESPREAD**	Detectability **EASY**	Impact **MODERATE**	
Consider anyone who can trick your users into submitting a request to your website. Any website or other HTML feed that your users access could do this.	Attacker creates forged HTTP requests and tricks a victim into submitting them via image tags, XSS, or numerous other techniques. If the user is authenticated, the attack succeeds.	CSRF takes advantage of web applications that allow attackers to predict all the details of a particular action. Since browsers send credentials like session cookies automatically, attackers can create malicious web pages which generate forged requests that are indistinguishable from legitimate ones. Detection of CSRF flaws is fairly easy via penetration testing or code analysis.		Attackers can cause victims to change any data the victim is allowed to change or perform any function the victim is authorized to use.	Consider the business value of the affected data or application functions. Imagine not being sure if users intended to take these actions. Consider the impact to your reputation.

Cookie Forcing CSRF

- There are tons of quirks to the same origin policy
- It's possible to GET or POST to any domain (basis for traditional CSRF)
- Lesser known: writing cookies is often much easier than reading them

Recap: Writing Cookies



path

The path on the server in which the cookie will be available on. If set to ***'/'***, the cookie will be available within the entire ***domain***. If set to ***'/foo/'***, the cookie will only be available within the ***/foo/*** directory and all sub-directories such as ***/foo/bar/*** of ***domain***. The default value is the current directory that the cookie is being set in.

domain

The domain that the cookie is available to. Setting the domain to ***'www.example.com'*** will make the cookie available in the ***www*** subdomain and higher subdomains. Cookies available to a lower domain, such as ***'example.com'*** will be available to higher subdomains, such as ***'www.example.com'***. Older browsers still implementing the deprecated » RFC 2109 may require a leading **.** to match all subdomains.

Some reference: Sze Chuen Tan

Recap: Writing Cookies

- From pr.bank.com we can set a cookie with
  - name: csrf_token
  - value: is_swear_this_is_a_nonce
  - domain: .bank.com
- secure.bank.com would now receive the cookie

Recap: Writing Cookies

Can https://secure.bank.com differentiate between cookies it sets vs. cookies set from http://pr.bank.com?

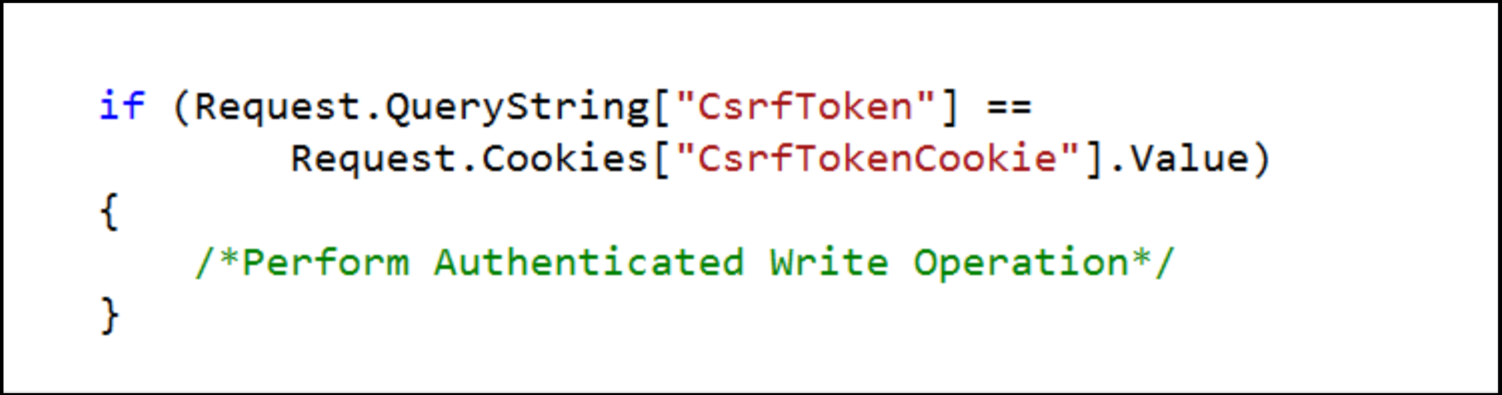
Recap: Writing Cookies

- Web frameworks most often (almost always) take the first cookie value when multiple cookies are given with the same name
- http://securebank.com can overwrite cookies for https://securebank.com (no duplicate cookies)
- All browsers have a limit to the number of cookies in the cookie jar
- It's common to add or modify cookies based on the DOM or request (cookie injection)

Recap: Writing Cookies

- To drill this in, it's often possible to write cookies, even though reading them is hard:
  - XSS in a neighbor domain
  - MiTM (usually even with HSTS)
  - Cookie injection

Double Submit Cookies



```
if (Request.QueryString["CsrfToken"] ==
        Request.Cookies["CsrfTokenCookie"].Value)
{
    /*Perform Authenticated Write Operation*/
}
```

Cookies Apply to other CSRF Things!

- What is the CSRF token tied to?
  - The CSRF token must be tied to something unique, or one user can replay another user's information
  - This is usually a session cookie, or sometimes (worse) a static piece of information like a userID
- What if the framework ties the CSRF token to the default sessionID, but then custom auth is used?
- This is most common with 'custom auth' or 'stateless' apps

.NET MVC CSRF Protection

- This is very good
- It checks:
  - sessionToken is correct
  - The cookie is tied to the POST parameter
  - The token is tied to the user
  - The user is properly logged in
  - An expiration
- But...
- Where does the user/session come from???

.NET MVC CSRF Protection

- MVC CSRF protection works fine by default.
  - The information is derived from the sessionID cookie automatically
  - The sessionID cookie is used to track users by default
- What if you auth another way?

.NET MVC CSRF Protection

demo

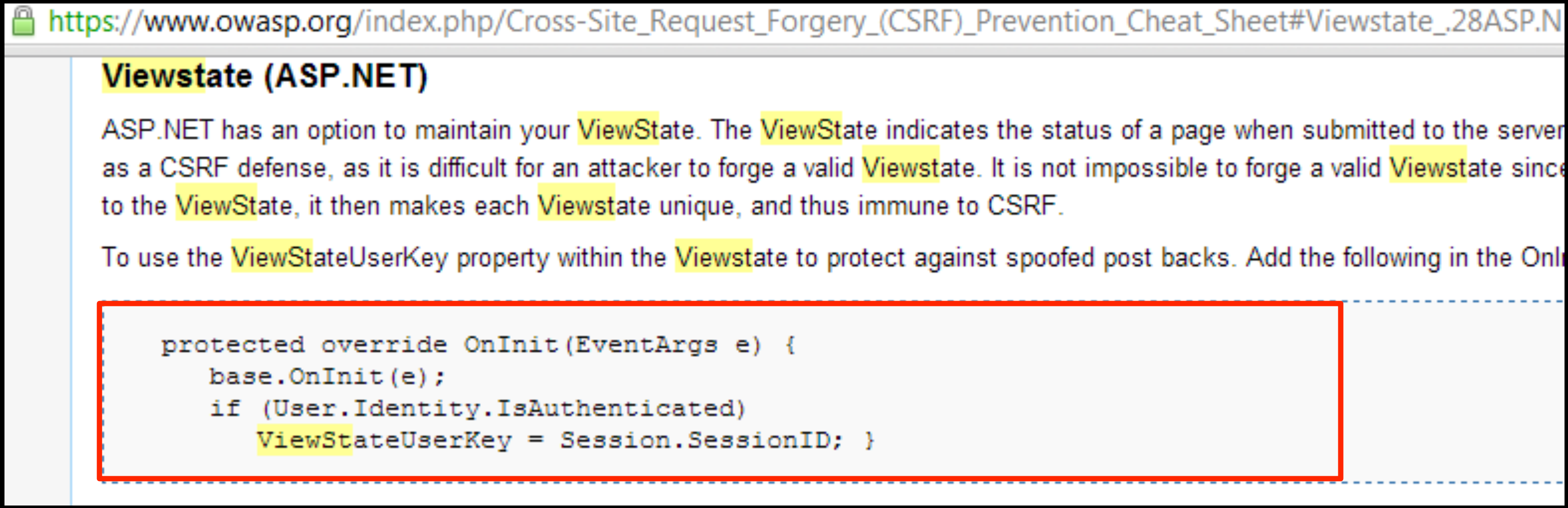
Generically, what can we learn from this?

- Where is this most common?
  - Custom auth with standard web framework
- Test methodology
  - Much easier to test than exploit (but CSRF will break your heart)
  - Figure out how the parameter nonce is tied to a cookie, and replace the values between users
- Exploit
  - Again: MiTM, cookie injection, neighbor XSS (in the demo we used neighbor XSS)

Let’s look at other Frameworks

- Does this only apply to .NET MVC? Of course not.
- Most languages/frameworks tie CSRF mitigations to the default session
- The cookie tossing CSRF issue is most common when using custom authentication

Forms .NET


https://www.owasp.org/index.php/Cross-Site_Request_Forgery_(CSRF)_Prevention_Cheat_Sheet#Viewstate_.28ASP.N

Viewstate (ASP.NET)

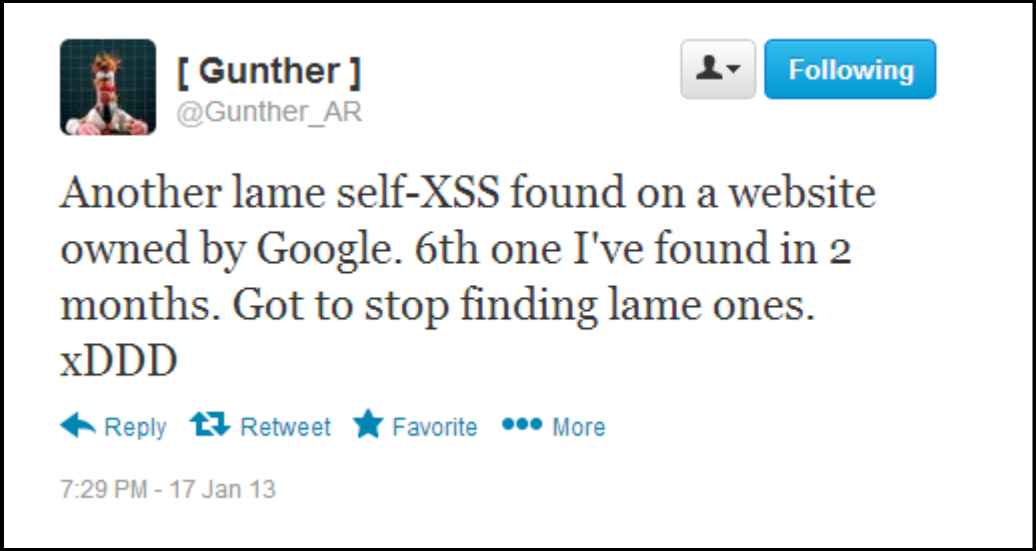
ASP.NET has an option to maintain your ViewState. The ViewState indicates the status of a page when submitted to the server
as a CSRF defense, as it is difficult for an attacker to forge a valid Viewstate. It is not impossible to forge a valid Viewstate since
to the ViewState, it then makes each Viewstate unique, and thus immune to CSRF.

To use the ViewStateUserKey property within the Viewstate to protect against spoofed post backs. Add the following in the Onl

```
protected override OnInit(EventArgs e) {
   base.OnInit(e);
   if (User.Identity.IsAuthenticated)
      ViewStateUserKey = Session.SessionID; }
```

“Non-Exploitable” XSS

- I see this a lot
- But remember… we can frequently write cookies


[Gunther]
@Gunther_AR

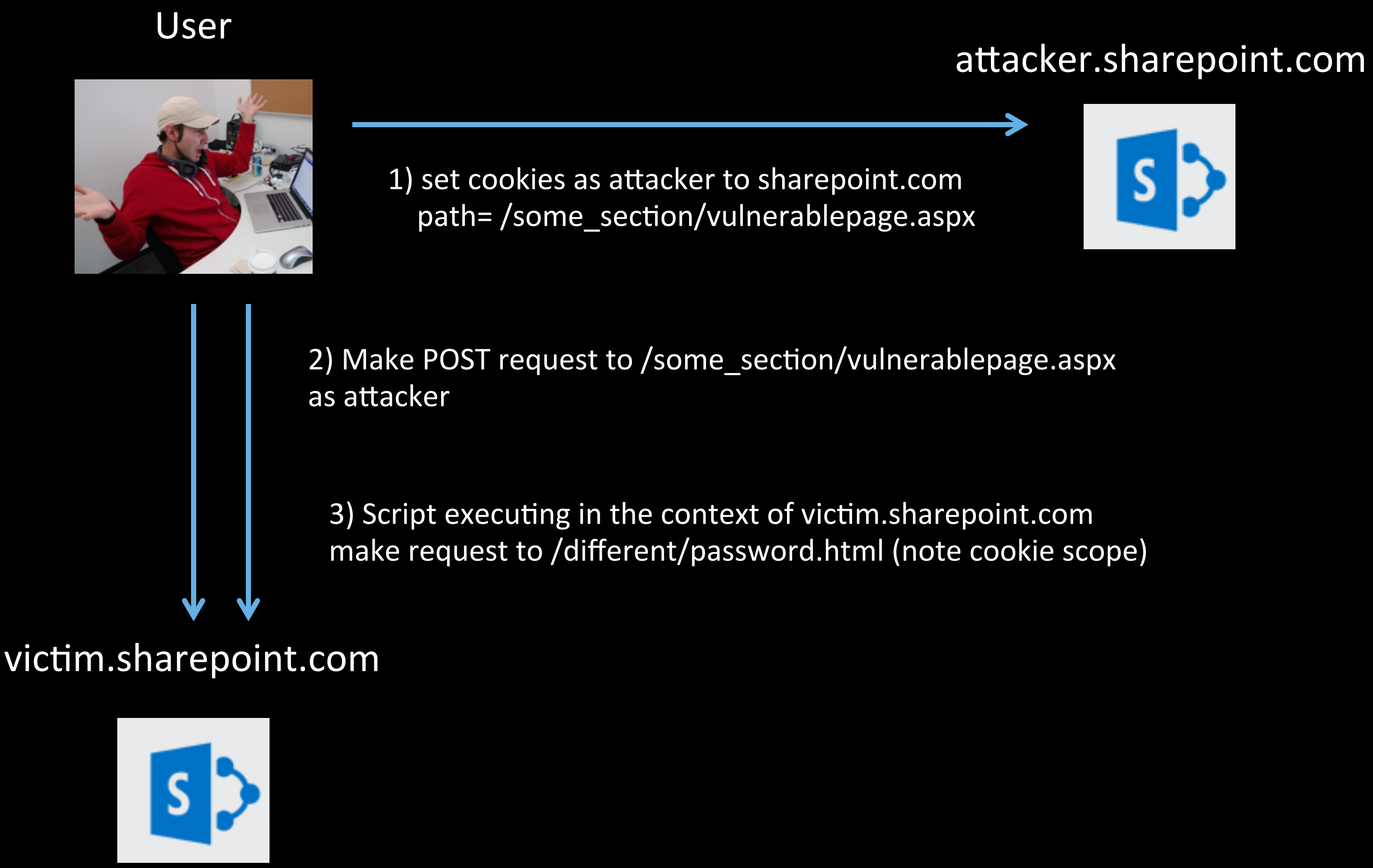
Another lame self-XSS found on a website owned by Google. 6th one I've found in 2 months. Got to stop finding lame ones. xDDD

7:29 PM - 17 Jan 13

“Non-Exploitable” XSS example

- Say an XSS exists in a CSRF protected POST request: http://customer.sharepoint.com/some_section/vulnerablepage.aspx
- How could we exploit this?
- SharePoint disclaimer:
  - This could equally apply to other places where we have cookie tossing
  - SharePoint is a good/easy example, because by design you have script execution in your separate domain attacker.sharepoint.com

self-xss in xxx.sharepoint.com/some_section/vulnerablepage.aspx


User

attacker.sharepoint.com

1) set cookies as attacker to sharepoint.com
path= /some_section/vulnerablepage.aspx

2) Make POST request to /some_section/vulnerablepage.aspx
as attacker

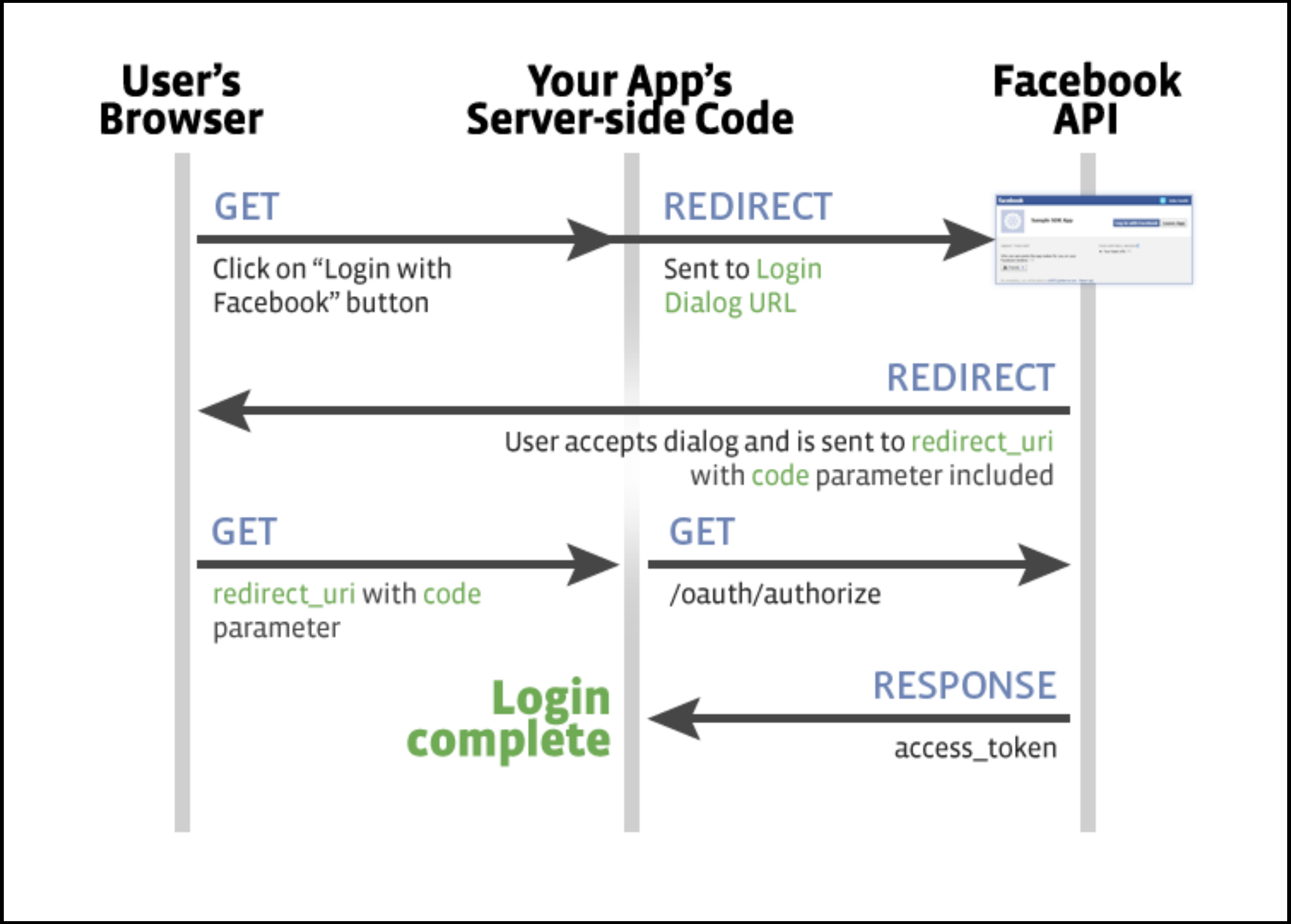
3) Script executing in the context of victim.sharepoint.com
make request to /different/password.html (note cookie scope)

victim.sharepoint.com

Single Sign On

- e.g. NTLM, Kerberos, Basic, etc.
  - But mostly NTLM with extended protection or Kerberos, since the others have worse problems
- It should be obvious that this is so easy to get wrong.
- By it's nature, SSO auth is separate from cookies, but out-of-box CSRF mitigations must use cookies

OAuth2 and OpenID

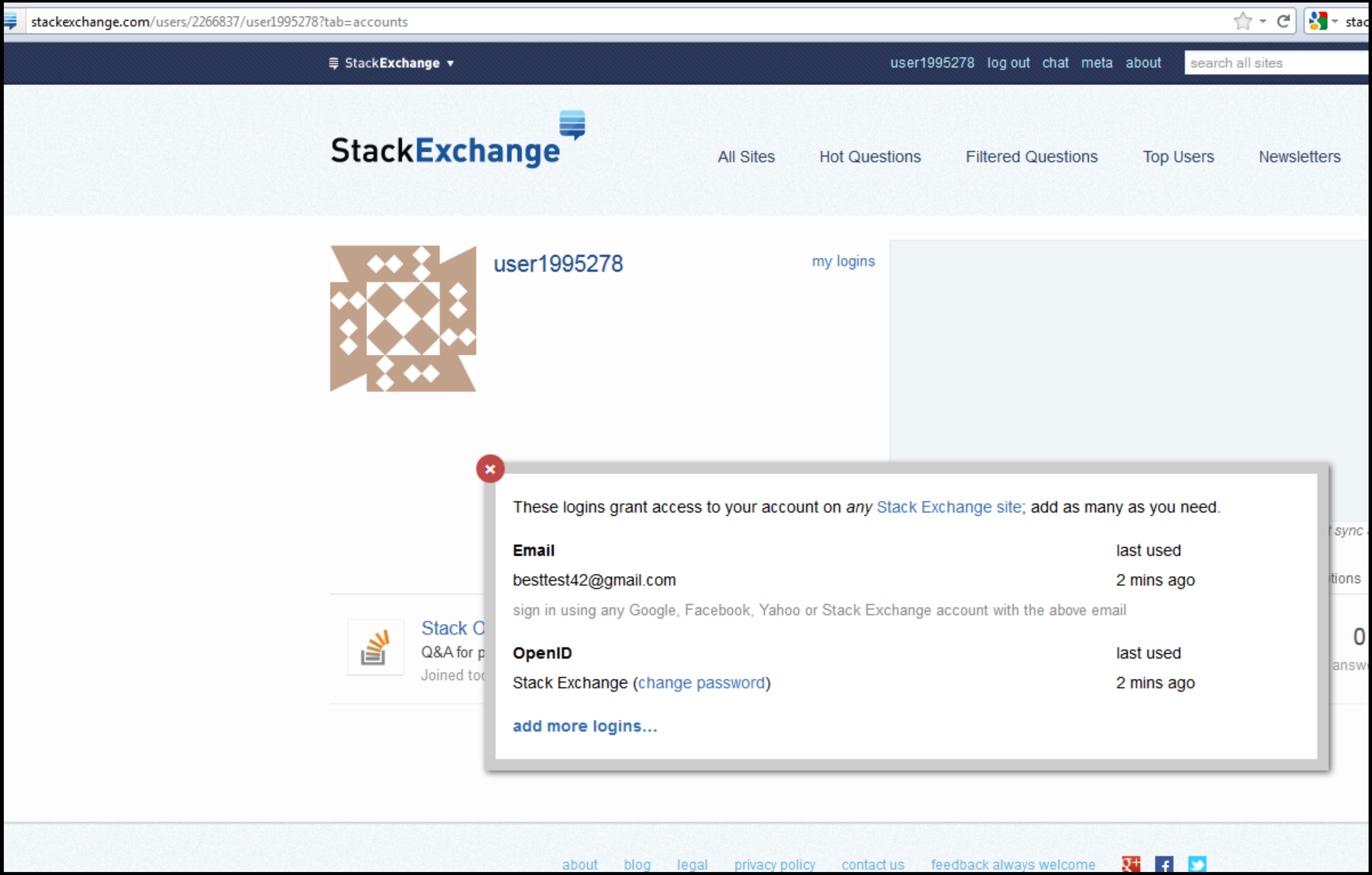

User's Browser
Your App's Server-side Code
Facebook API
GET
Click on "Login with Facebook" button
REDIRECT
Sent to Login Dialog URL
REDIRECT
User accepts dialog and is sent to redirect_uri with code parameter included
GET
redirect_uri with code parameter
GET
/oauth/authorize
RESPONSE
access_token
Login complete

Facebook Login Diagram

OAuth2

- What's the impact of CSRF here?
  - http://stephensclafani.com/2011/04/06/oauth-2-0-csrf-vulnerability/
  - http://sso-analysis.org/
- CSRF Mitigations are covered in the spec itself
- "state" parameter should be used
  - Non guessable value
  - User agent's authenticated state
  - Kept in a location accessible only to the client (i.e. cookies, protected by the same-origin policy)

Tying Accounts Together



stackexchange.com/users/2266837/user1995278?tab=accounts

These logins grant access to your account on *any* Stack Exchange site; add as many as you need.

Email	last used
besttest42@gmail.com	2 mins ago

sign in using any Google, Facebook, Yahoo or Stack Exchange account with the above email

OpenID	last used
Stack Exchange (change password)	2 mins ago

add more logins...

about blog legal privacy policy contact us feedback always welcome

Attack Ideas

- The first attack I thought of:
  - Toss cookies into victim (stackoverflow)
  - The cookies used for auth may not be tied to the nonce sent to the identifier
  - Associate the attacker's account with the victim's account and win!
- But there are a lot of cookies for each site
- It turns out there's usually an easier way
  - but the above will probably be a problem for a while

OAuth2 Facebook Attack

- Create an attacker Facebook account
- Grant the accessing application (stackoverflow) permissions to attacker Facebook
- Victim is logged in to stackoverflow
- A malicious site does the following
  - Logs victim in to attacker's Facebook by using CSRF on the Login
  - POSTs to the account association request
  - Attacker Logs out of other sessions

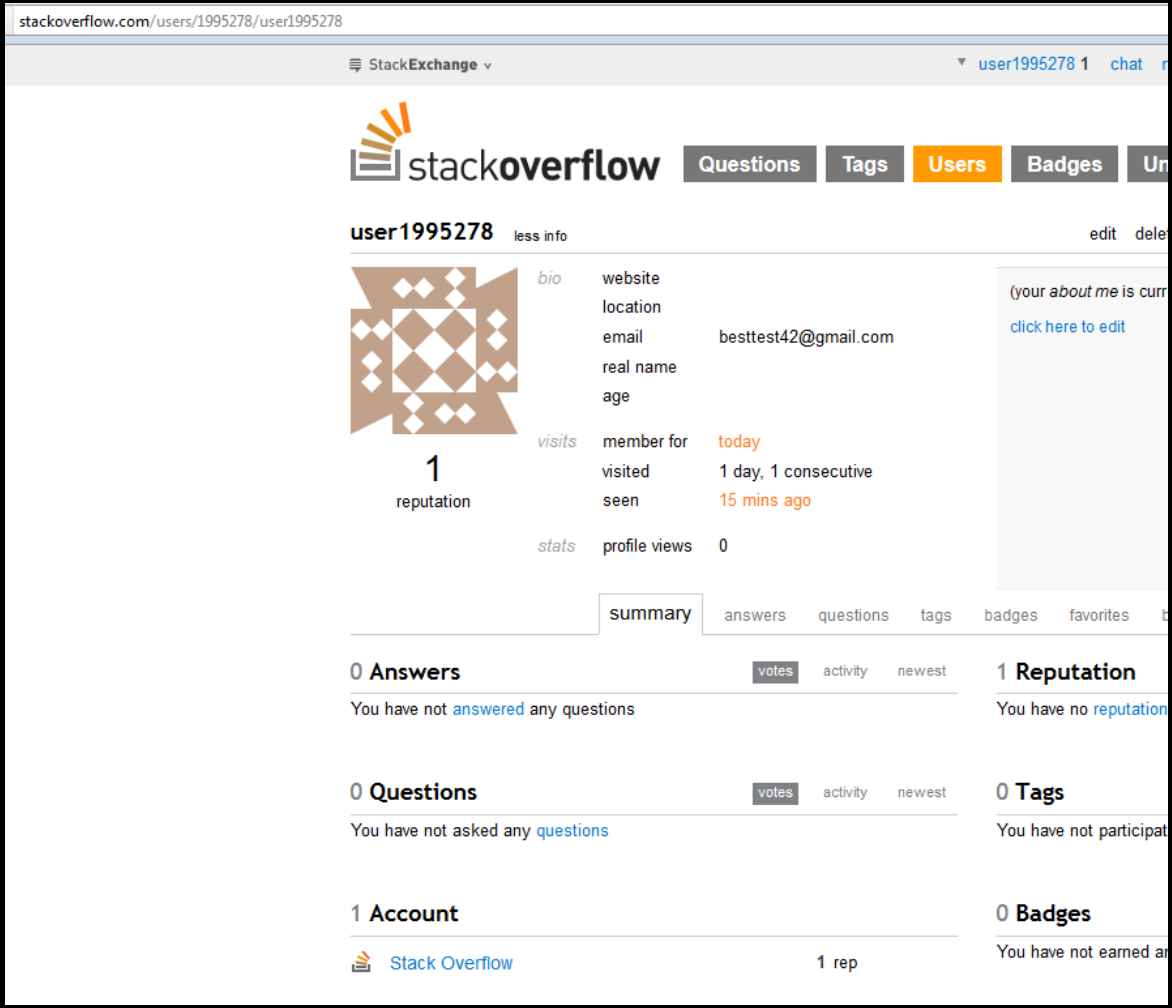
OAuth2 Attack

demo

Logging into an Attacker Account

- To login to Facebook, the referer cannot be set
- There are several ways we can POST cross domain and strip the referer
  - HTTPS -> HTTP (note HTTPS -> HTTPS does send the referer, even cross domain)
  - CORS POST request
  - <meta> refresh to data (kotowicz has a blog post on this)

OAuth2 Attack



stackoverflow.com/users/1995278/user1995278

StackExchange ▾ user1995278 1 chat

stackoverflow Questions Tags Users Badges Un

user1995278 less info edit dele

bio	website	
	location	
	email	besttest42@gmail.com
	real name	
	age	
visits	member for	today
	visited	1 day, 1 consecutive
	seen	15 mins ago
stats	profile views	0

1 reputation

(your *about me* is curr

click here to edit

summary answers questions tags badges favorites

0 Answers votes activity newest

You have not answered any questions

0 Questions votes activity newest

You have not asked any questions

1 Account

Stack Overflow 1 rep

1 Reputation

You have no reputation

0 Tags

You have not participat

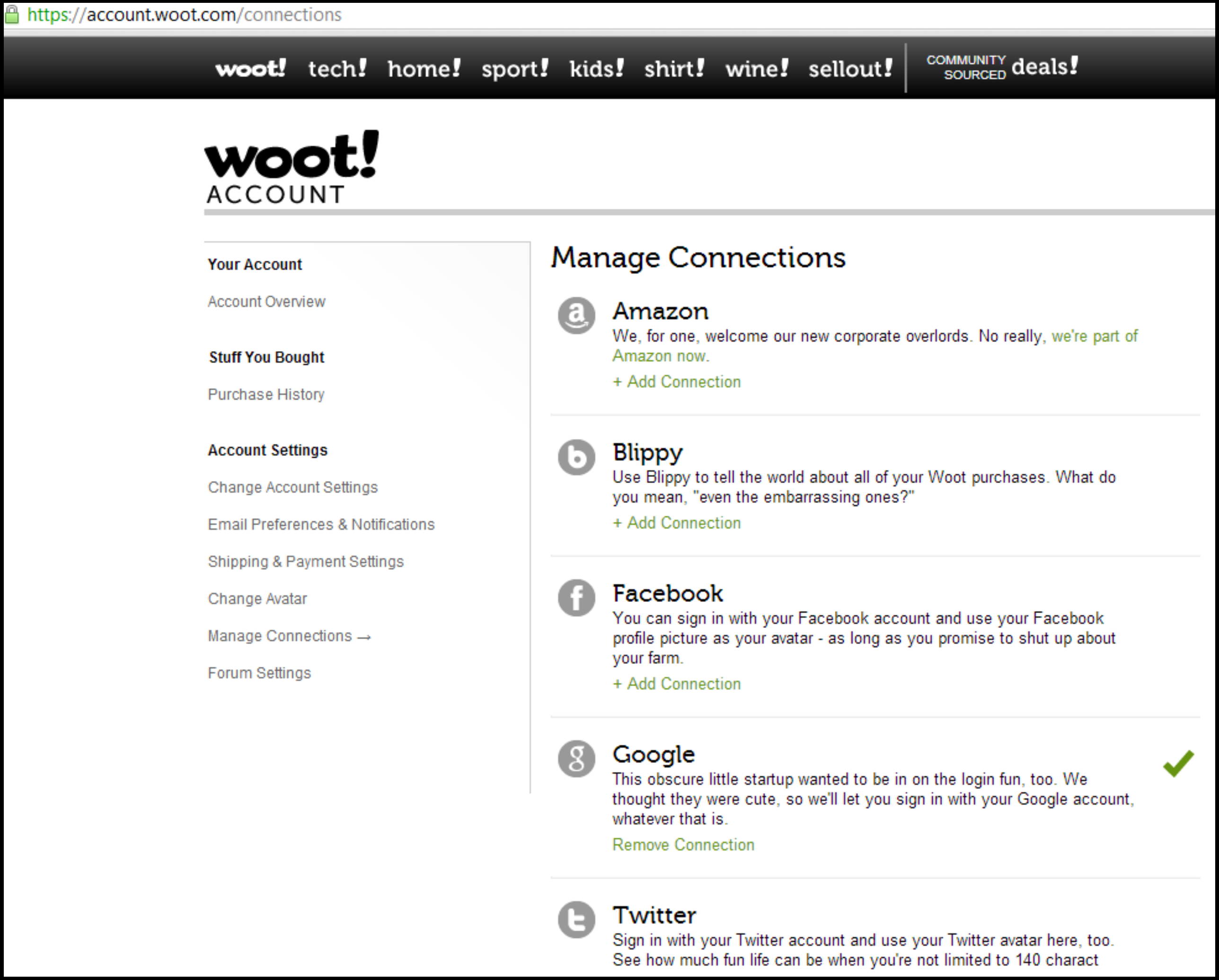
0 Badges

You have not earned ar

stackexchange is just an example

- Is this just stackexchange?
  - This is every application I tested
- …

woot.com



https://account.woot.com/connections

Manage Connections

Amazon
We, for one, welcome our new corporate overlords. No really, we're part of Amazon now.
+ Add Connection

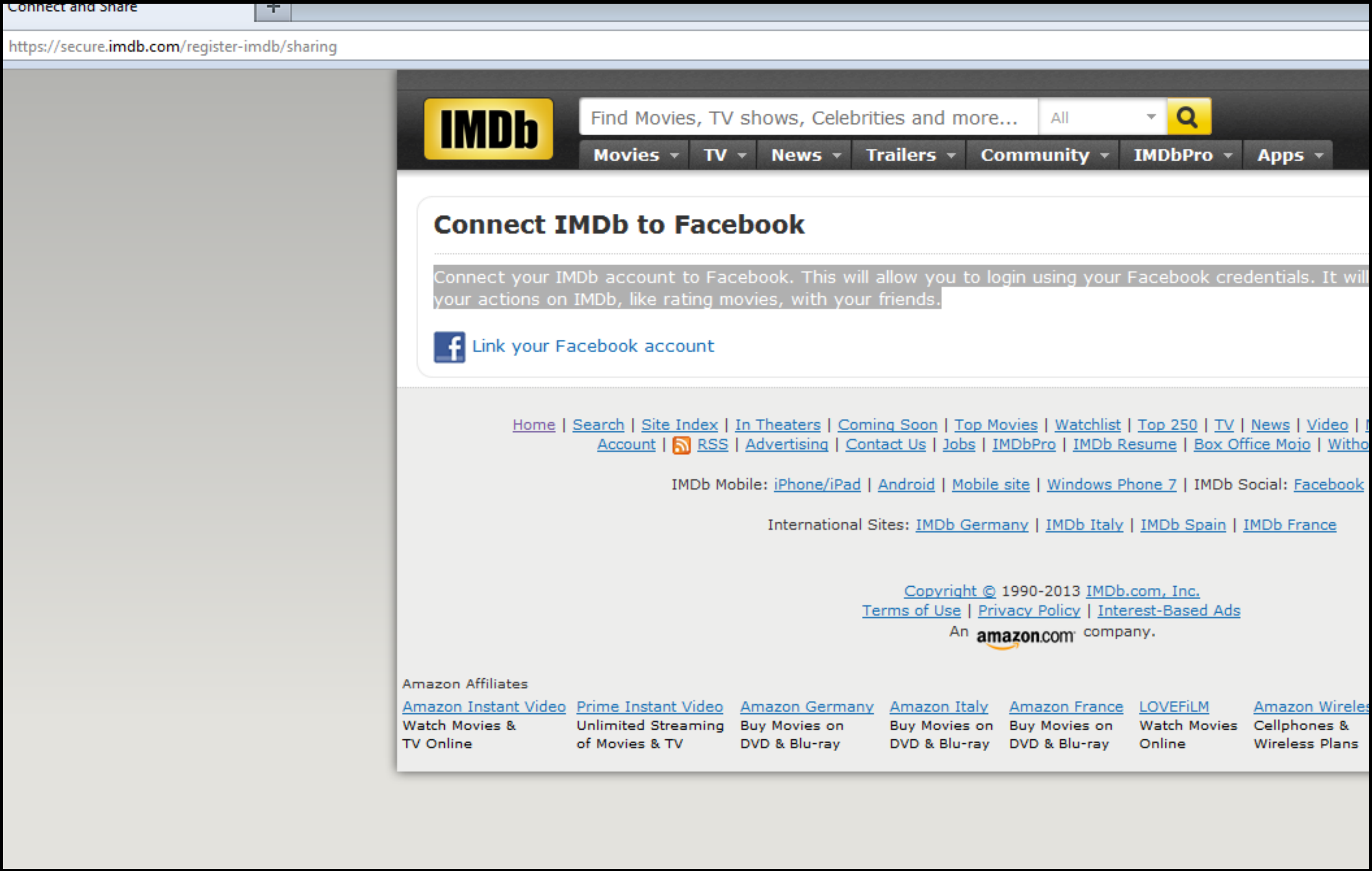
Blippy
Use Blippy to tell the world about all of your Woot purchases. What do you mean, "even the embarrassing ones?"
+ Add Connection

Facebook
You can sign in with your Facebook account and use your Facebook profile picture as your avatar - as long as you promise to shut up about your farm.
+ Add Connection

Google ✓
This obscure little startup wanted to be in on the login fun, too. We thought they were cute, so we'll let you sign in with your Google account, whatever that is.
Remove Connection

Twitter
Sign in with your Twitter account and use your Twitter avatar here, too. See how much fun life can be when you're not limited to 140 charact

imdb.com



Connect and Share

https://secure.imdb.com/register-imdb/sharing

Connect IMDb to Facebook

Connect your IMDb account to Facebook. This will allow you to login using your Facebook credentials. It will
your actions on IMDb, like rating movies, with your friends.

Link your Facebook account

Home | Search | Site Index | In Theaters | Coming Soon | Top Movies | Watchlist | Top 250 | TV | News | Video |
Account | RSS | Advertising | Contact Us | Jobs | IMDbPro | IMDb Resume | Box Office Mojo | Witho

IMDb Mobile: iPhone/iPad | Android | Mobile site | Windows Phone 7 | IMDb Social: Facebook

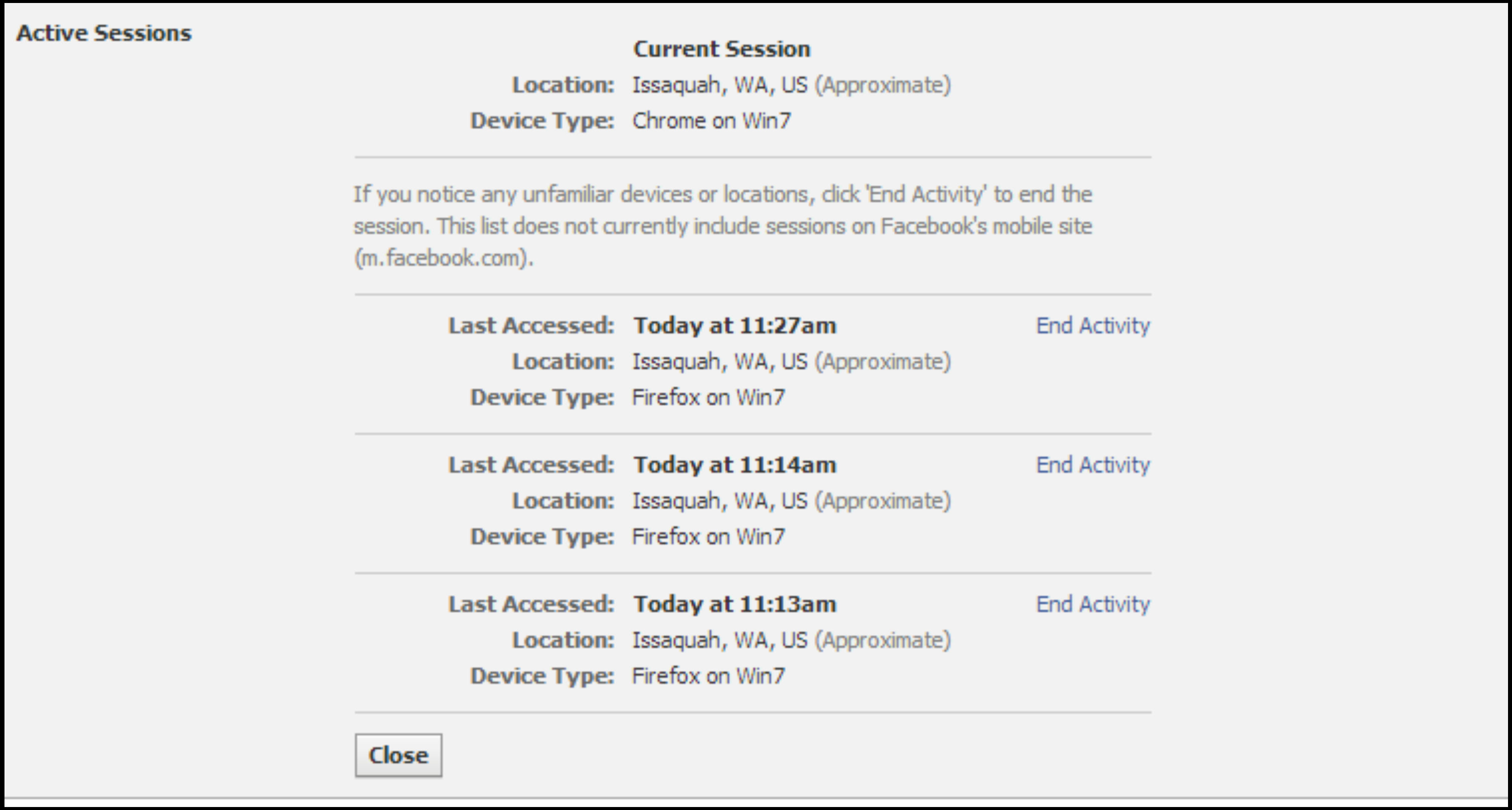
International Sites: IMDb Germany | IMDb Italy | IMDb Spain | IMDb France

Copyright © 1990-2013 IMDb.com, Inc.
Terms of Use | Privacy Policy | Interest-Based Ads
An amazon.com company.

Amazon Affiliates

Amazon Instant Video	Prime Instant Video	Amazon Germany	Amazon Italy	Amazon France	LOVEFiLM	Amazon Wireles
Watch Movies & TV Online	Unlimited Streaming of Movies & TV	Buy Movies on DVD & Blu-ray	Buy Movies on DVD & Blu-ray	Buy Movies on DVD & Blu-ray	Watch Movies Online	Cellphones & Wireless Plans

Logging out of Attacker Account



Active Sessions

Current Session

Location:	Issaquah, WA, US (Approximate)	
Device Type:	Chrome on Win7	

If you notice any unfamiliar devices or locations, click 'End Activity' to end the session. This list does not currently include sessions on Facebook's mobile site (m.facebook.com).

Last Accessed:	**Today at 11:27am**	End Activity
Location:	Issaquah, WA, US (Approximate)	
Device Type:	Firefox on Win7	
Last Accessed:	**Today at 11:14am**	End Activity
Location:	Issaquah, WA, US (Approximate)	
Device Type:	Firefox on Win7	
Last Accessed:	**Today at 11:13am**	End Activity
Location:	Issaquah, WA, US (Approximate)	
Device Type:	Firefox on Win7	

Hiding the CSRF

- Protecting against UI redressing is even in the spec, so just creating a frame isn’t ideal

Attack Rating

- The risk here is large – let's look at that picture again
- Often many ways to login
  - Just ONE of these trusted identifier sites is enough to take over an account FOREVER
  - These can be hidden in the UI
- Once added, you often cannot even remove the logins, or the new account can remove old accounts
- No need to retype your old password!

Attack Rating

- Let’s compare this to a classic XSS in a consumer page without using this?
- If I found an XSS in feedburner.google.com
  - Would this matter for Google accounts? Probably not that much
  - But this is really important for everyone who trusts google.com as an identity provider

How do we fix this?

- Who’s bug is this?
- It can be fixed on the consumer side
  - state parameter properly tied to the sessionID
  - It seems not many people understand this, as not one application I looked at did this
- Can it be fixed on the IDP side?
  - If we make the identity provider login CSRF proof, is this a non-issue?
  - Separate the flow for login versus “associate account”?
  - oauth attack against other id providers

Other Common CSRF Things

- Change the request method and remove the nonce
  - the ispostback problem. set __VIEWSTATE=
  - try submitting CSRF nonce from another user
  - Why not add a CSRF nonce to every request?
- Non-Changing Tokens
  - The demos aren't exciting, but... the fired worker scenario

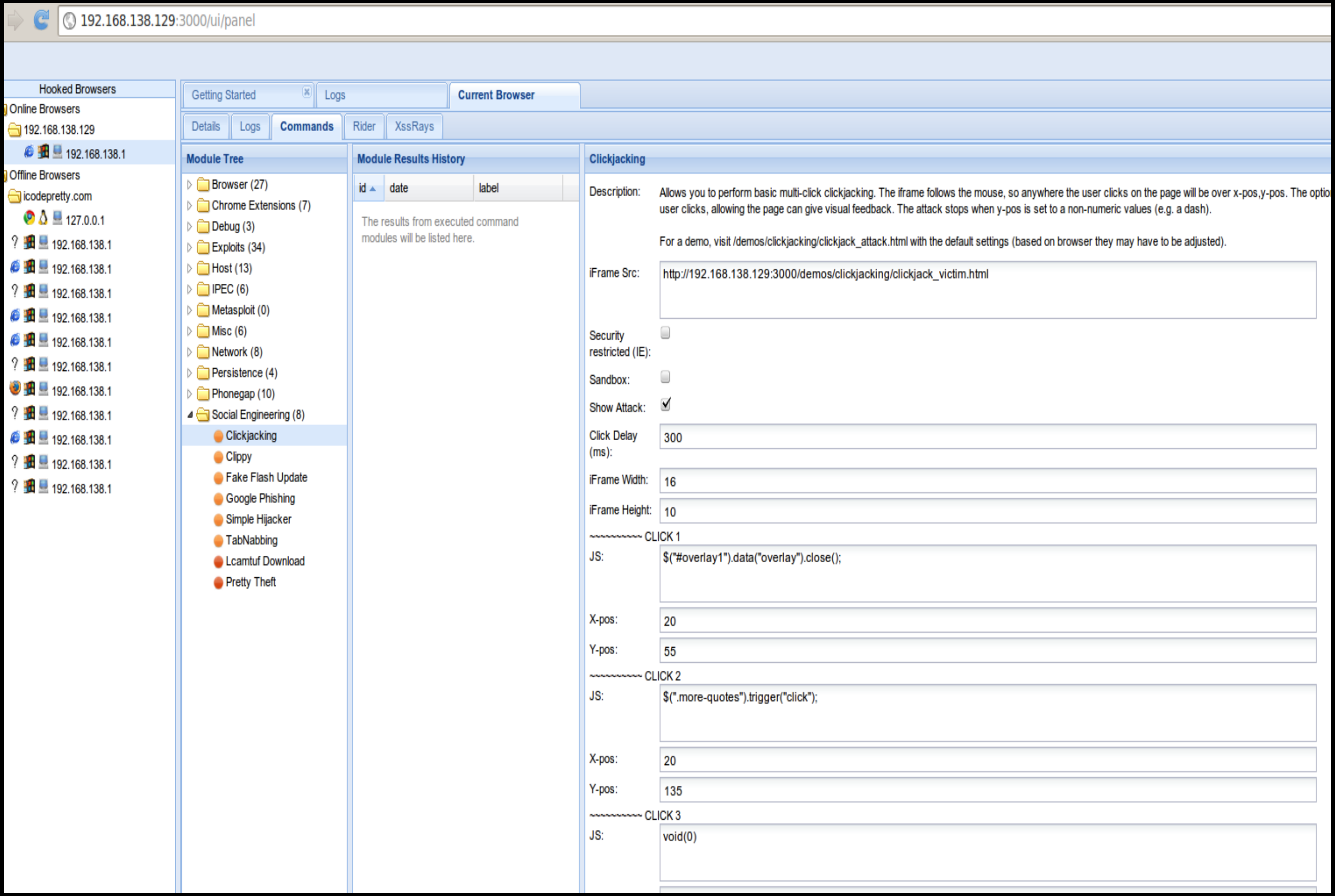
CSRF Mitigations

- Only use POST requests to change state, and all POST requests require an unguessable CSRF token
- CSRF tokens are cryptographically tied to the session ID cookie (which must be tied to auth)
  - This goes for cross domain requests like OAuth too

Whitepaper Content

- Clickjacking
- NTLM Relaying

BeEf Clickjacking Module

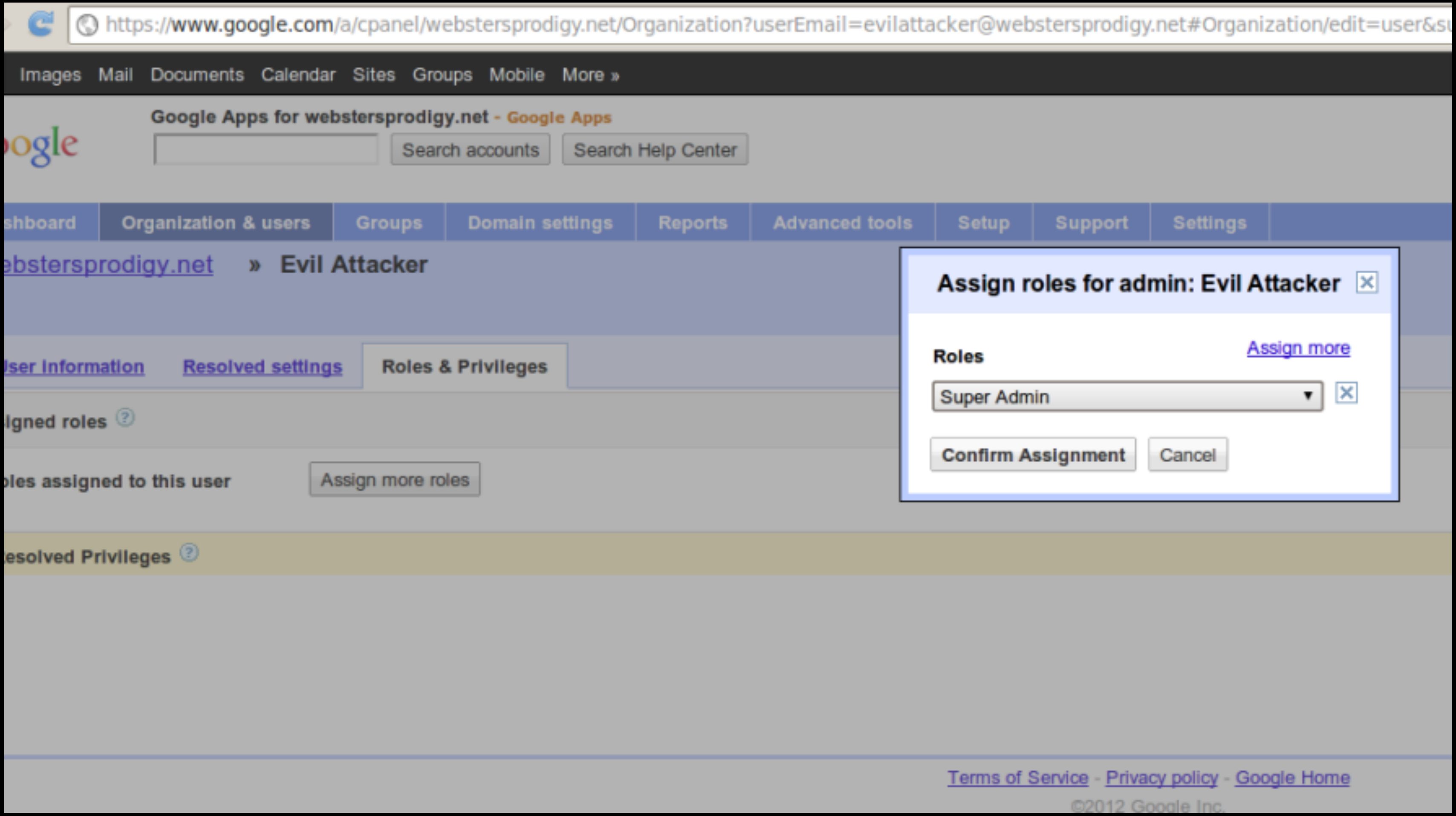


Description: Allows you to perform basic multi-click clickjacking. The iframe follows the mouse, so anywhere the user clicks on the page will be over x-pos,y-pos. The optio user clicks, allowing the page can give visual feedback. The attack stops when y-pos is set to a non-numeric values (e.g. a dash).

For a demo, visit /demos/clickjacking/clickjack_attack.html with the default settings (based on browser they may have to be adjusted).

iFrame Src:	http://192.168.138.129:3000/demos/clickjacking/clickjack_victim.html
Security restricted (IE):	☐
Sandbox:	☐
Show Attack:	✓
Click Delay (ms):	300
iFrame Width:	16
iFrame Height:	10
~~ CLICK 1	
JS:	$("#overlay1").data("overlay").close();
X-pos:	20
Y-pos:	55
~~ CLICK 2	
JS:	$(".more-quotes").trigger("click");
X-pos:	20
Y-pos:	135
~~ CLICK 3	
JS:	void(0)

X-FRAME-OPTIONS Edge Cases


https://www.google.com/a/cpanel/webstersprodigy.net/Organization?userEmail=evilattacker@webstersprodigy.net#Organization/edit=user&s

ebstersprodigy.net » Evil Attacker

igned roles

les assigned to this user

esolved Privileges

Assign roles for admin: Evil Attacker

Roles

Super Admin

©2012 Google Inc.

That’s all!

Please complete the Speaker Feedback Surveys

Here’s my contact info again:

http://webstersprodigy.net

@webstersprodigy

richard.lundeen@gmail.com