Preparing for the Cross Site Request Forgery Defense

Chuck Willis
chuck.willis@mandiant.com

Black Hat DC 2008
February 20, 2008

MANDIANT™
INTELLIGENT INFORMATION SECURITY

About Me

- Principal Consultant with MANDIANT in Alexandria, VA
  - Full spectrum information security company:
    - Commercial and Government Services
    - Public and Private Training Courses
    - Forensic and Incident Response Products
  - Services include Application Security, Network Security, Incident Response, Computer Forensics, Research and Development
  - Free Software releases include Red Curtain, Web Historian, First Response
  - Product available: MANDIANT Intelligent Response

Agenda

- Scenario
- What is Cross Site Request Forgery?
- How do CSRFs relate to investigations and forensics?
- CSRF Case Studies and Live Demos
- Scope of CSRF Vulnerabilities
- How to detect or rule out CSRF during a forensic exam
- How to detect and prevent CSRF in a web application

Scenario

MANDIANT™
INTELLIGENT INFORMATION SECURITY

Scenario

- Examining a user’s computer for evidence of “kitty” (as in cat) pornography and you find:
  - Google searches for “kitty pr0n”
  - Flikr searches for “kitty”
  - Images in web cache of cats in compromising positions
  - Pages in the web cache and browser history for sites like “www.kittyandme.com”

Scenario

- Continue looking and find more things (via the cache or via a subpoena):
  - Netflix queue has movies like:
    - Garfield: A Tail of Two Kitties
    - Hello Kitty's Paradise
    - Cat on a Hot Tin Roof
  - Posts to online forums describing “love” for cats

Scenario

- Question: Based on this evidence can you determine that the user was actively seeking or knowingly possessing cat porn?

- Answer: Not necessarily – all the evidence above could have been placed by a web application vulnerability known as Cross Site Request Forgery (CSRF)

My Experience

- I have not seen use of Cross Site Request Forgeries of the nature described in this presentation during investigations

- However:
  - It is possible that they are being used in some cases in this way
  - More importantly, this issue could be brought up as part of a person’s defense

What is CSRF?

What's in a name?

- Cross Site Request Forgery (CSRF) is the most common name for a web application security issue also known as:
  - Cross Site Reference Forgery (CSRF)
  - XSRF (similar to XSS acronym for Cross Site Scripting)
  - "Sea Surf"
  - Session Riding
  - One-Click Attack (Microsoft's terminology)
  - Hostile Linking
  - A type of Confused Deputy attack

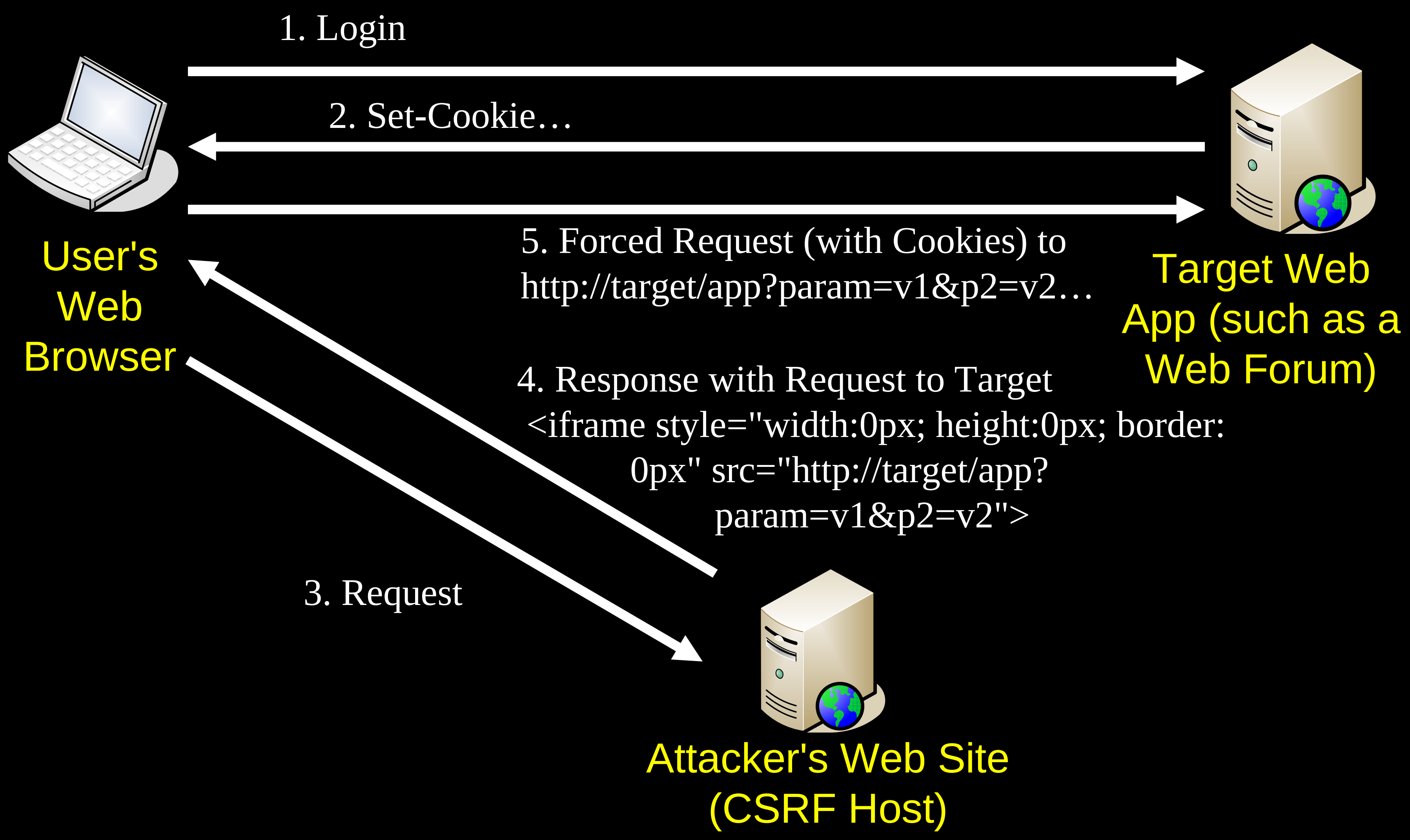
CSRF vs XSS

- Despite the similar sounding names, Cross Site Request Forgeries (CSRF) and Cross Site Scripting (XSS) refer to completely different issues which require entirely different protection mechanisms

CSRF Defined

- CSRF is an exploitation of the HTTP protocol's feature that a web page can include HTML elements that will cause the browser to make a request to any other web site
- Like all HTTP transactions, the submission to the second site will include the user’s session information (usually cookies) if they have an established session
- Regardless of if the user has a session with the second site, elements of the second site will be loaded in the victim's browser and can appear in the cache and history
- CSRF can occur on either an HTTP GET or a POST

Simple GET CSRF In Action


1. Login
2. Set-Cookie…
5. Forced Request (with Cookies) to
http://target/app?param=v1&p2=v2…
User's
Web
Browser
Target Web
App (such as a
Web Forum)
4. Response with Request to Target
<iframe style="width:0px; height:0px; border:
0px" src="http://target/app?
param=v1&p2=v2">
3. Request
Attacker's Web Site
(CSRF Host)

GET CSRF

- The simplest way to create a GET request is with an HTML Image tag, such as:

  ```
  <img src="http://target/app?
  param=v1&p2=v2">
  ```

- But, an image tag will only retrieve the specific URL listed (not any referenced images, scripts, etc) so another method is to use a "hidden inline frame":

  ```
  <iframe style="width:0px;
  height:0px; border: 0px"
  src="http://target/app?
  param=v1&p2=v2">
  ```

Ways to force a GET request in HTML

- GET requests can be elicited using:
  - Image: <img src="">
  - Script: <script src="">
  - Link: <a href="">
  - Background Image
  - Cascading Style Sheet
  - Page Icon
  - Frame (Inline or traditional)
  - Prefetch Link
  - Pop-Up / Pop-Under browser window
  - Applet / Flash Code / ActiveX Control (<object>, <embed> and/or <applet> tag)

Types of CSRF Hosts

- An attacker does not need to lure the victim to his or her own web server to create a CSRF
- Other places to host a CSRF:
  - Online Forum (often allow a user to link to an image as an avatar or as an attachment)
  - HTML Email
  - Photo Gallery
  - Wiki
  - Blog
  - Online Auctions and E-Commerce Sites
  - …
  - Pretty much any site that allows for posting anything like HTML
- The CSRF could be hosted on the target server itself

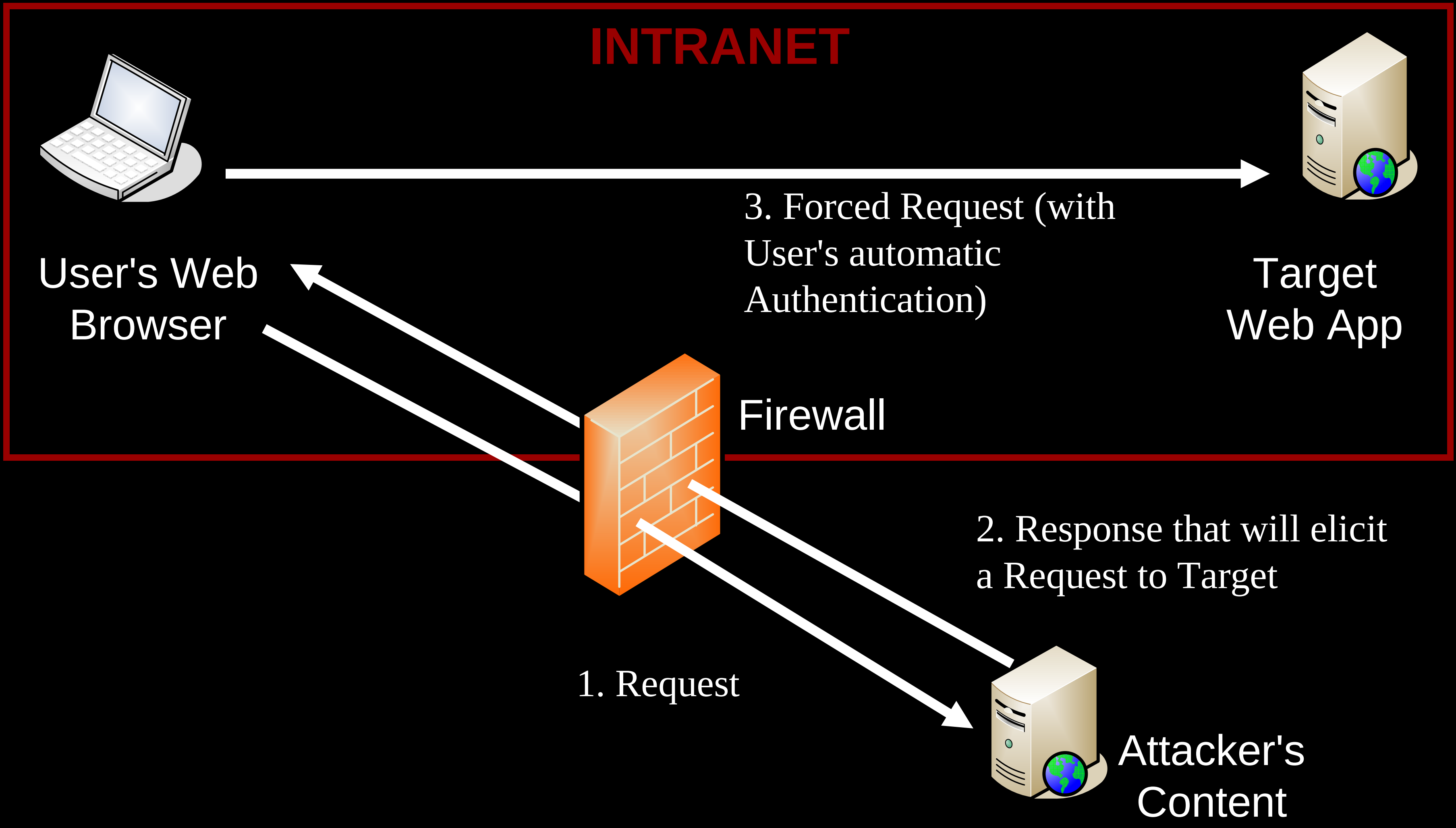
Session Mechanisms

- Most web applications store session identifiers in a cookie, which makes them vulnerable to CSRF
- Other session mechanisms are also vulnerable to CSRF:
  - HTTP Basic Authentication
  - HTTP Digest Authentication
  - Integrated Windows Authentication (NTLM or Kerberos)
- Session tracking in the URL or query parameters is not vulnerable to CSRF

CSRF on an Intranet

- One of the troubling aspects of CSRF is that the attacker does not even need to be able to access the target application
- All traffic to the application comes from the victim user, so as long as he or she can access the application, the CSRF can be performed
- CSRF can be particularly devastating against Intranet applications:
  - Often use Windows Integrated authentication - no login required!
  - Often have poor access controls and logging

CSRF on an Intranet


INTRANET
User's Web Browser
3. Forced Request (with User's automatic Authentication)
Target Web App
Firewall
2. Response that will elicit a Request to Target
1. Request
Attacker's Content

CSRFs in Investigations

- During investigations and forensics, we are concerned with CSRF for two reasons:
  - Server side state changes (the normal motivation for preventing CSRF in web application security)
  - Effects of CSRF on the client web browser and the client's web traffic
    - Causes sites to be visited without the user's knowledge
    - Causes items to be written into the user's web cache
    - Can cause URLs to be added to the browser history (depends on circumstances and browser)

CSRF Case Studies

MANDIANT™
INTELLIGENT INFORMATION SECURITY

Netflix CSRF Case Study

- In September 2006, security researcher Dave Ferguson notified Netflix of a variety of CSRF issues with their site
- Dave publicly released information about the issues in October 2006 after the most significant issues were addressed

Netflix CSRF Case Study

- Like most Internet applications, Netflix uses cookies to store session information
- Netflix used GET requests to handle pretty much all its user input
- For example, the "Add" button to add a movie to your rental queue was a simple link to a URL like:

```
http://www.netflix.com/AddToQueue?
  movieid=12345678
```

Messing with the Queue

- Dave recognized this fertile ground for CSRF
- He crafted HTML to add a movie to the user's queue:

```
<img src = "http://www.netflix.com/
 AddToQueue?movieid=12345678">
```

- He also recognized that he could move the new movie to the top of the user's queue (after a short delay in JavaScript):

```
<img src = "http://www.netflix.com/
 MoveToTop?movieid=12345678
 &fromq=true">
```

Netflix CSRF Unresolved

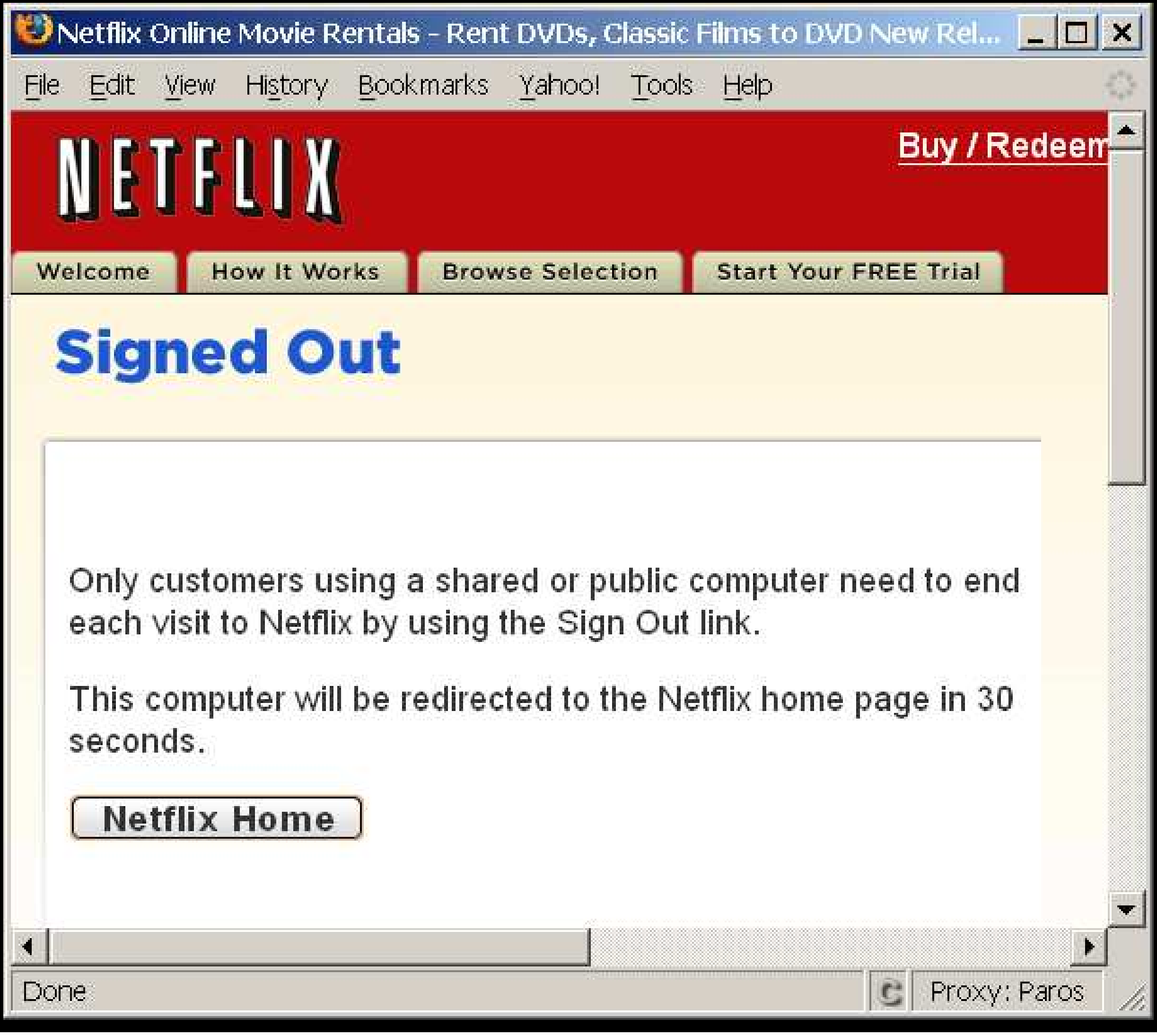
- As of February 2008 (17 months later), this issue has not been resolved and Netflix users are still vulnerable to CSRF in the form:

```
<img src = "http://www.netflix.com/
  AddToQueue?movieid=12345678">
```

Netflix CSRF Demo

MANDIANT™
INTELLIGENT INFORMATION SECURITY

Netflix CSRF Unresolved


Netflix Online Movie Rentals - Rent DVDs, Classic Films to DVD New Rel...
File Edit View History Bookmarks Yahoo! Tools Help
NETFLIX
Buy / Redeem
Welcome
How It Works
Browse Selection
Start Your FREE Trial
Netflix Home
Done
Proxy: Paros

Signed Out

Only customers using a shared or public computer need to end each visit to Netflix by using the Sign Out link.

This computer will be redirected to the Netflix home page in 30 seconds.

MANDIANT™

CSRF in Search Engines

- Search engine queries are almost always done with GET requests
- This makes them easily vulnerable to GET CSRF requests

CSRF in Google

- A normal Google search URL looks like:

```
http://www.google.com/search?hl=en&q=
  cat+pics&btnG=Google+Search
```

- All that is need to execute a CSRF to Google is the HTML:

```
<iframe style="width:0px; height:0px;
  border: 0px"
  src="http://www.google.com/search?hl
  =en&q=cat+pics&btnG=Google+Search">
```

CSRF in Google

- A CSRF forced search to Google will:
  - Show up in the user's cache
  - Possibly show up in the user's web browser's history
  - In Firefox, cause first link to be pre-fetched and added to the cache
  - Show up in the user's search history if they have enabled that feature with Google
  - Probably be stored in Google's internal databases

MANDIANT™

Google CSRF Demo

MANDIANT™
INTELLIGENT INFORMATION SECURITY

Scope of CSRF Vulnerabilities

MANDIANT™
INTELLIGENT INFORMATION SECURITY

Scope of CSRF Vulnerabilities

"In fact, if you have not taken specific steps to mitigate the risk of CSRF attacks, your applications are most likely vulnerable."

- Chris Shiflett in 2004
  http://shiflett.org/articles/cross-site-request-forgeries

Scope of CSRF Vulnerabilities

"No statistics, but the general consensus is just about every piece of sensitive website functionality is vulnerable [to Cross Site Request Forgery]."

- Jeremiah Grossman and T.C. Niedzialkowski in 2006
http://www.whitehatsec.com/home/resources/presentations/files/javascript_malware.pdf

Scope of CSRF Vulnerabilities

"Cross-Site Request Forgery (aka CSRF or XSRF) is a dangerous vulnerability present in just about every website."

- Jeremiah Grossman in 2006
http://jeremiahgrossman.blogspot.com/2006/09/csrf-sleeping-giant.html

Scope of CSRF Vulnerabilities

"Cross site request forgery is not a new attack, but is simple and devastating…."

"This vulnerability is extremely widespread…."

"All web application frameworks are vulnerable to CSRF. "

- OWASP Top Ten 2007
http://www.owasp.org/index.php/Top_10_2007-A5

How to detect or rule out CSRF during an investigation

MANDIANT™
INTELLIGENT INFORMATION SECURITY

How to detect or rule out CSRF

- Look for pages that forced the requests in the cache
  - Page(s) will not be in the cache if they have been marked "no cache":
    - By the server in HTTP headers
    - In the HTML itself using <meta equiv> tags
  - Some CSRF hosts will not allow the attacker to control caching of the page(s)
  - Be aware of encodings of the target URL and the enclosing tag

URL Encodings / Obfuscation

- Parameters in a URL may be URL encoded: http://www.google.com/search?%68%6C=%65%6E&%71=%63%61%74%2B%70%69%63%73&%62%74%6E%47=%47%6F%6F%67%6C%65%2B%53%65%61%72%63%68
- Hostname in a URL may be replaced by IP address in:
  - Standard dotted format (64.233.169.103)
  - DWord Format (1089055079)
  - Hex Format (0x40.0xe9.0xa9.0x67)
  - Octal Format (0100.0351.0251.0147)

HTML Tag Encodings / Obfuscation

- HTML tags may be encoded in a variety of manners similar to a Cross Site Scripting attack

```
<iframe/randomtext ... >

<<<iframe ... >

<IMG """><iframe ... >">

perl -e 'print "<ifr\0ame...>";'

<script>document.write(unescape('%3C%
 69...%22%3E'));</script>
```

- See RSnake’s Cross Site Scripting (XSS) Cheat Sheet at http://ha.ckers.org/xss.html

MANDIANT™

How to detect or rule out CSRF

- Look at the web browser's history
  - URLs that have been forced by a CSRF (such as in an IFRAME or an IMG tag) may not appear in the browser history (depends on circumstances and browser)
  - Pages found on disk but not in the history could be an indication of CSRF, but are more likely the result either:
    - Browser history and cache aging differently
    - User clearing the history

How to detect or rule out CSRF

- Construct a timeline
  - Most fruitful way to detect or rule out a Cross Site Request Forgery is to construct a timeline of the user’s activity
  - Merge data from the web browser cache, web browser history, and any other logs that you may have (Proxy, IDS, Firewall, Web Server, etc)
  - Examine items immediately before the activity in question to determine if a CSRF may be involved

How to detect or rule out CSRF

- Look at the list of URLs that were typed into the address bar of the browser
  - This information cannot be forced
  - Not all browsers record a list of URLs that were typed into the browser
  - Users will type in the URL for only a small percentage of sites that they visit

How to detect or rule out CSRF

- Look at items in browser Favorites / Bookmarks
  - This information cannot be forced
  - Users will bookmark only a small percentage of sites that they visit

How to detect or rule out CSRF

- Look for evidence outside of the web browser cache
  - For example, if you are interested in image files, look for image files outside of the cache (indicating that the user intentionally saved them)
  - Will only find things that the user obtained from non-web sources or that the user saved from a web site

How to detect or rule out CSRF

- Determine if the application is vulnerable to CSRF
  - This will only help in investigating a "traditional" CSRF issue where a state change on the server may have been forced
  - There is no way that a web application can prevent CSRFs that only aim to affect the local browser cache and history
  - If the relevant web page of application is not vulnerable to CSRF, then the information in question could not have been forced by a CSRF
  - Next section will detail how to determine if a page on a web application is vulnerable to CSRF

How to detect and prevent CSRF in a web application

MANDIANT™

INTELLIGENT INFORMATION SECURITY

Identifying CSRF

- The key characteristics of a CSRF vulnerability are:
  - Application accepts a request that makes something occur on the server
  - Attacker can determine all the parameters of that request for another user (typically in a CSRF the parameters are fixed for all users)

How not to prevent CSRF

- There are suggestions sometimes made to address CSRF that do not work:
- "Check referrer headers, if the referrer is not from this domain, ignore the request"
  - Referrer headers are not sent with all requests depending on circumstances and the browser in use
  - Some users / browsers never send referrer headers so they will not be able to access the app

How not to prevent CSRF

- "Use POST requests instead of GETs"
  - CSRF with an HTTP POST request is only a tiny bit harder than with an HTTP GET request
- "Limit the duration of sessions"
  - This reduces the window of exposure for your application to CSRF, but does not eliminate it
  - Some applications want / need long sessions

Preventing CSRF

- CSRF cannot be directly prevented in that an application cannot prevent other sites from forcing users to make requests to the application
- CSRF only needs to be prevented in forms that cause a state change in the application - there is no need to worry about CSRF for GET requests on a well designed application
- The key to preventing CSRF is for an application to determine which requests are legitimate and which have been forced by a CSRF before acting on the request

Preventing CSRF

- In order to prevent CSRF, an additional parameter must be added to a request, either on the URL line parameters or in POST parameters
- This is implemented by adding the parameter to a form as a hidden field or to the "action" (location where the form submits)
- This parameter must not be something that the attacker can determine so he or she cannot construct a link or script to execute a CSRF

Preventing CSRF

- The most easiest parameter to use is the session ID
  - Using the session ID to prevent CSRF is sometimes known as "Double Cookie Submission"
  - There are some issues with reusing the session ID for this purpose
  - A better technique is to use a separate unique identifier that is also tied to the user's session

Questions?

Chuck Willis
chuck.willis@mandiant.com

MANDIANT™
INTELLIGENT INFORMATION SECURITY

Preparing for the Cross Site Request Forgery Defense

Chuck Willis
chuck.willis@mandiant.com

Black Hat DC 2008

February 20, 2008

MANDIANT™

INTELLIGENT INFORMATION SECURITY