Who is Afraid of Cookies?

by Asaf Gery

Email: asaf.gery@gmail.com
Phone: +972-4-9995014
Mobile: +972-54-2215733

Outline

- Introduction
- Innocent Uses of Cookies
- Cookies Mechanism
- Not So Innocent Uses of Cookies
- Malicious Uses of Cookies and Defense Techniques

Introduction

What is a Cookie?

- Cookie is data stored by the browser on behalf of a web server
- A browser sends Cookies with every request to the corresponding web server*

Innocent Uses of Cookies

Innocent Uses of Cookies

- Personalization
- Navigation
- Shopping Carts
- Google Analytics
- Facebook Connect
- etc.

Cookies Mechanism

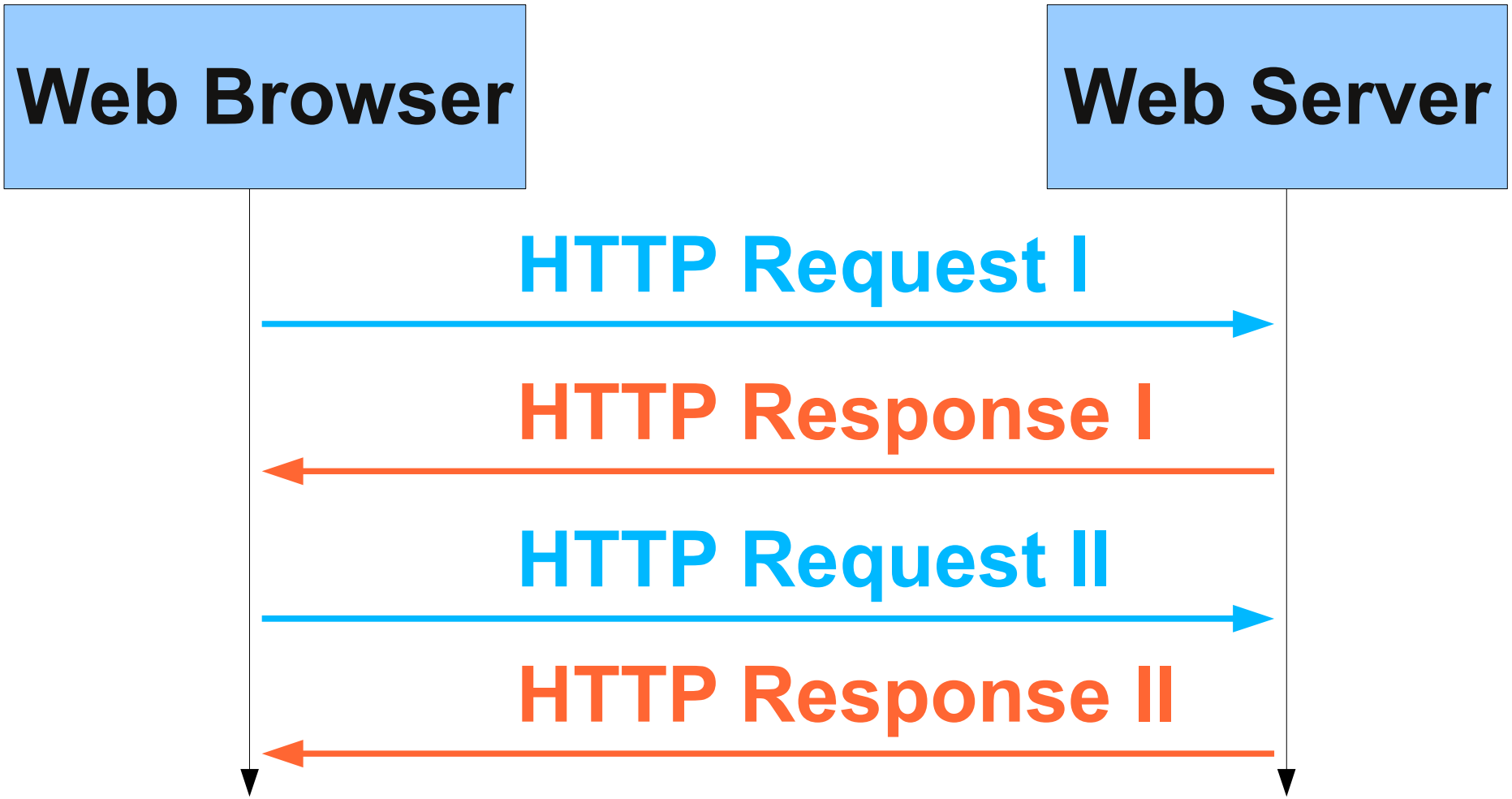
Cookies Under the Hood

- Cookies are set by a web server using an HTTP Header or by a browser using JavaScript
- In order to understand the mechanism, we should have a basic understanding of HTTP – Hyper Text Transfer Protocol

HTTP

- **H**yper **T**ext **T**ransfer **P**rotocol specifies the communication between Browsers and Web Servers
- HTTP is request – response oriented

HTTP Sequence



HTTP Sequence Example

- In the following slides pay attention to the following HTTP Headers: **Referer**, **Set-Cookie** and **Cookie**

HTTP Request I

GET /wiki/Main_Page HTTP/1.1

Host: en.gentoo-wiki.com

Referer: http://en.gentoo-wiki.com/wiki/Framebuffer

HTTP Response I

HTTP/1.1 200 OK
Server: Apache
Last-Modified: Thu, 02 Sep 2010 17:55:00 GMT
Content-Type: text/html; charset=utf-8
Content-Length: 32533
Date: Mon, 27 Dec 2010 18:34:34 GMT
Set-Cookie: show_side_bar=true;Expires=Thu, 22-Mar-2011 18:35:38 GMT

HTTP Request II

GET /img/wiki_g.png HTTP/1.1

Host: en.gentoo-wiki.com

Referer: http://en.gentoo-wiki.com/wiki/Main_Page

Cookie: show_side_bar=true

Third Party Cookies

- Cookies are supposed to be used between browsers and web servers - **1st** and **2nd Parties**
- Web pages embed files (videos, flash, images, css, scripts) from **3rd party sites** such as ad. Networks, Google Analytics etc.

Not So Innocent Uses of Cookies

Not So Innocent Uses of Cookies

- Tracking user's web surfing habits
- Creating user profile by analyzing visited websites

Cookie Attributes

Cookie Attributes

- By setting Cookie attributes a Web Server instructs browsers under which conditions should that Cookie be sent by browsers and how long should it be kept in the browser's cache (memory and hard disk)

Domain Attribute

- **Domain** attribute instructs the browser when to send that Cookie (with which URLs)
- Format: Domain=<*domain-spec*>; where <*domain-spec*> has the same format of Google's advanced search
- More dots (.) means that the Cookie will be associated with less URLs

Domain Attribute

- Examples:

  Set-Cookie: ...;**Domain=.gery.co.il**;... - this Cookie will be sent with every URL whose host name ends with .gery.co.il

  Set-Cookie: ... ;**Domain=mail.gery.co.il**;... - this Cookie will be sent only to URLs whose host name is precisely mail.gery.co.il

Path Attribute

- **Path** attribute compliments **Domain** attribute
- Format: Path=<*path-spec*> where <*path-spec*> is Unix style – with forward slashes (/)
- More slashes means that the Cookie will be associated with less URLs

Path Attribute

- Examples:

  Set-Cookie: ...;**Path=/;**... - this Cookie will be sent with every URL that matches the domain spec (if any), meaning – every path

  Set-Cookie: ... ;**Path=/accounts/a/as/;**... - this Cookie will be sent only to URLs whose path starts with /accounts/a/as/

Expires and Max-Age

- **Expires** specifies Cookie's expiration time in terms of date and and time (e.g. **Expires=Tue, 22-Mar-2011 22:00:00 GMT;**)
- **Max-Age** specifies Cookie's expiration time in terms of secs. from now (e.g. **Max-Age: 3600;** - keep the Cookie for one hour)

Persistent Cookies

- Persistent Cookies are Cookies which have expiration time in the future and therefore are stored in the browser's cache (i.e. on the hard disk)
- Expiration is set using either Expires or Max-Age attribute

Session Cookies

- Cookie whose expiration date or max age is not specified is called **Session Cookie**
- Session Cookies are usually stored on the RAM and deleted as soon as the browser is closed
- The problem: HTTP does not define the life time of an HTTP Session, since it was originally designed as a session-less protocol

Secure and HttpOnly

- Secure and HttpOnly attributes allow better security when using cookies:
  - **Secure** specifies a Cookie that will be transferred only via an encrypted channel (HTTPS)
  - **HttpOnly** specifies a Cookie that will not be accessible from Javascript code on the browser side

Malicious Uses of Cookies

Session Hijacking

- Web sites use Cookies to identify user sessions
- By stealing those identity session Cookies an attacker can impersonate the victim
- Web servers have no way telling the difference between a real user and an attacker

Session Hijacking Methods

- Various methods can be used for session hijacking:
  - Network Eavesdropping
  - DNS Cache Poisoning
  - XSS
  - CSRF

Session Hijacking
Using Network Eavesdropping

Network Eavesdropping

- Traffic on a network can be intercepted and read by computers other than its sender and its receiver (particularly over unencrypted open Wi-Fi network)
- FireSheep is an example of using this technique

Network Eavesdropping

- Tools such as tcpdump and WireShark can be used to capture traffic on a network
- This attack can be easily mitigated by using HTTPS and Secure Cookies exclusively

Session Hijacking
Using DNS Cache Poisoning

DNS Cache Poisoning

- CheckPoint's movie
- DNS Cache Poisoning is a more sophisticated attack that takes advantage of the fact that identity session Cookies are usually set in the domain scope

DNS Cache Poisoning

1. A server that is controlled by an attacker pretends to be a member of a domain whose Cookies are to be stolen (e.g. bogus.facebook.com) by poisoning the DNS Cache of the victim

DNS Cache Poisoning

2. A web page created by the attacker which resides on a different server, contains a reference to the bogus server (using an image, css, script, flash, etc)

DNS Cache Poisoning

3. The victim's browser intercepts that server as a member of the impersonated domain (facebook.com in our example) and therefore sends the session cookies with the HTTP request for the image

DNS Cache Poisoning

4. Now the attacker has the victim's session cookies and he/she can use them to impersonate the victim and act on behalf of the victim

- Sometimes, the action can be encoded in the URL of the bogus image, in that case the browser's attempt to load the image will trigger the action automatically

DNS Cache Poisoning

- This attack can be dramatically mitigated by using HTTPS and Secure Cookies exclusively
- HTTPS requires certificate
- Wrong certificate -> browser's warning

Session Hijacking
Using Cross Site Scripting

XSS - Cross Site Scripting

- XSS is a code injection attack
- An attacker injects JavaScript code into a website
- The browser cannot tell the difference between an injected code and a genuine code

XSS - Cross Site Scripting

- Using XSS an attacker can steal the victim's Cookies
- Example: Click here!

Session Hijacking
Using Cross Site Request Forgery

CSRF - Cross Site Request Forgery

- CSRF exploits the trust that a web site has in a user's browser
- In this attack a malicious web site “manipulates” the browser to execute an action on a victim web site by loading a specially crafted image*

CSRF - Cross Site Request Forgery

- Example (very simplified): <img src="http://yourbank.com/withdraw?account=david&amount=9000000&for=moshe">

CSRF - Cross Site Request Forgery

- Prevention is not trivial!

Samy Worm

XSS/CSRF – Samy Worm

- On October 4, 2005, Samy Kamkar released a worm on MySpace that used a combination of XSS and CSRF attacks
- The worm displayed **"but most of all, Samy is my hero"** on victims' profiles
- Within 20 hours, over one million profiles were infected

For Further Reading

HTTP

- RFC 1945 – HTTP 1.0 (May 1996)
- RFC 2068 – HTTP 1.1 (January 1997)
- RFC 2616 – HTTP 1.1 (June 1999, obsoleted RFC 2068)
- All RFCs can be found here - http://www.ietf.org/rfc.html

Wikipedia

- HTTP Cookie - https://secure.wikimedia.org/wikipedia/en/wiki/HTTP_cookie

Netscape Cookie Specification

- Netscape's original Cookie specification (June, 1994) - http://curl.haxx.se/rfc/cookie_spec.html

Cookies RFCs

- RFC 2109 – HTTP State Management Mechanism (February 1997)
- RFC 2965 – HTTP State Management Mechanism (October 2000, obsoleted RFC 2109)
- All RFCs can be found here - http://www.ietf.org/rfc.html

Samy Worm

- Samy's own story - http://namb.la/popular/
- Technical details and the source code - http://namb.la/popular/tech.html

Thank You