Agenda

- Review: HTTPS and SSL
- Cookies
If you were president Trump, you want to access a website hosted in Russia, you would hope that website is using HTTPS rather than HTTP, why?
If you were president Trump, it’s 3am in the morning, but you can’t sleep because you just want to tweet. However, somehow the twitter app on your Android phone does not work, so you decide to open your browser and type twitter.com in the address bar. twitter.com is smart, and twitter cares about security, so it sends you a 301 response and redirects you to its HTTPS version, so your traffic will then be encrypted. Is there any security problem here?
If you were an FBI agent, you try to monitor the network traffic in and out of the Trump tower. In cybersecurity, what type of attack is this?
If you were Hillary Clinton, you have a private email server, the email server has a web interface which enables you and your staff members access the email server. From security perspective, your IT engineer needs to buy something from a company like GoDaddy or Symantec, or Comodo, so that you are confident you are accessing the right server rather than some malicious server. What does he need to buy?
Subdomain: A subdomain is a domain that is part of a larger domain;

- e.g., www.boisestate.edu and my.boisestate.edu are subdomains of the boisestate.edu domain, which in turn is a subdomain of the edu top-level domain (TLD).

- e.g., www.brunomars.com and store.brunomars.com are subdomains of the brunomars.com domain, which in turn is a subdomain of the com top-level domain (TLD).
In computing, the same-origin policy is an important concept in the web application security model. Under the policy, a web browser permits scripts contained in a first web page to access data in a second web page, but only if both web pages have the same origin.

An origin is defined as the triple protocol, host, port. Two resources are considered to be of the same origin if and only if all these values are exactly the same.
the following table gives an overview of typical outcomes for checks against the URL "http://www.example.com/dir/page.html".

<table>
<thead>
<tr>
<th>Compared URL</th>
<th>Outcome</th>
</tr>
</thead>
<tbody>
<tr>
<td><a href="http://www.example.com/dir/page2.html">http://www.example.com/dir/page2.html</a></td>
<td>Success</td>
</tr>
<tr>
<td><a href="http://www.example.com/dir2/other.html">http://www.example.com/dir2/other.html</a></td>
<td>Success</td>
</tr>
<tr>
<td><a href="http://user:passwd@www.example.com/dir2/other.html">http://user:passwd@www.example.com/dir2/other.html</a></td>
<td>Success</td>
</tr>
<tr>
<td><a href="http://www.example.com:81/dir/other.html">http://www.example.com:81/dir/other.html</a></td>
<td>Failure</td>
</tr>
<tr>
<td><a href="https://www.example.com/dir/other.html">https://www.example.com/dir/other.html</a></td>
<td>Failure</td>
</tr>
<tr>
<td><a href="http://en.example.com/dir/other.html">http://en.example.com/dir/other.html</a></td>
<td>Failure</td>
</tr>
<tr>
<td><a href="http://example.com/dir/other.html">http://example.com/dir/other.html</a></td>
<td>Failure</td>
</tr>
<tr>
<td><a href="http://v2.www.example.com/dir/other.html">http://v2.www.example.com/dir/other.html</a></td>
<td>Failure</td>
</tr>
</tbody>
</table>
The same-origin policy helps protect sites that use authenticated sessions.

The following example illustrates a potential security risk that could arise without the same-origin policy. Assume that a user is visiting a banking website and doesn’t log out. Then, the user goes to another site that has some malicious JavaScript code running in the background that requests data from the banking site. Because the user is still logged in on the banking site, the malicious code could do anything the user could do on the banking site. For example, it could get a list of the user’s last transactions, create a new transaction, etc. This is because the browser can send and receive session cookies to the banking site based on the domain of the banking site.
Cookie Attributes

- Name
- Value
- Domain:
- Path: the Domain and Path attributes define the scope of the cookie.
- Expires/Expiration: states when the cookie should be discarded.
- HostOnly:
- Session: A session cookie, also known as an in-memory cookie or transient cookie, exists only in temporary memory while the user navigates the website. Web browsers normally delete session cookies when the user closes the browser.
- Secure
- HttpOnly
Read/Write Cookies

- Server: using "set-cookie" header
- Client side Javascript: document.cookie
- When sending a request to a server, a web browser includes all unexpired cookies whose domains and paths match the requested URL, excluding those marked as secure from the inclusion in an HTTP request.
Cookie Attributes: Domain and Path

- the Domain and Path attributes define the scope of the cookie.
- e.g., the website maroon5.com cannot set a cookie that has a domain of chase.com because this would allow the maroon5.com website to control the cookies of chase.com.
- if a cookie’s Domain and Path attributes are not specified by the server, they default to the domain and path of the resource that was requested.
Cookie Attribute: HostOnly

- this flag is true when a cookie is set from the server without specifying a domain.
- HostOnly cookie means that the cookie should be handled by the browser only to the same host/server that firstly sent it to the browser, i.e., the request’s host must exactly match the domain of the cookie.
- e.g., if a cookie is set from boisestate.edu without a domain, the cookie will only be sent for requests to boisestate.edu, it will not be sent for requests to my.boisestate.edu, or any other subdomains of boisestate.edu.
- If the server set a cookie with a domain specified (e.g., the boisestate.edu domain), the cookie will be sent for requests to all other subdomains, such as my.boisestate.edu.
Cookie Attribute: Secure

- the browser will only send cookies with the secure flag when the request is going to an HTTPS page. Said in another way, the browser will not send a cookie with the secure flag set over an unencrypted HTTP request.

- By setting the secure flag, the browser will prevent the transmission of a cookie over an unencrypted channel.
Cookie Attribute: HttpOnly

- If the HttpOnly flag is included in the HTTP response header, the cookie cannot be accessed through client side script (like Javascripts). As a result, even if a cross-site scripting (XSS) flaw exists, and a user accidentally accesses a link that exploits this flaw, the browser will not reveal the cookie to a third party.
Cookie Overwriting: If a cookie shares the domain scope with a related domain, it can be directly overwritten by that domain using another cookie with the exactly same name/domain/path. Of particular note, although a secure cookie can only be read by an HTTPS process, it can be written or overwritten by an HTTP request.
Cookie Shadowing: An attacker with the control of a related domain can intentionally shadow a cookie by injecting another one that has the same name, but different domain/path scope. e.g., to shadow a cookie with "value=good; domain=www.example.com; path=/; secure", a related domain evil.example.com can write a cookie with "value=bad; domain=.example.com; path=/home". Later, when browser issues a request to https://www.example.com/home, both cookies match the URL and are included. The cookie header will be "Cookie: value=bad; value=good;". The "good" cookie could be shadowed by the "bad" cookie if a website happens to prefer the value of "bad" over "good".
References

A large portion of the material is adapted from:

- Cookies Lack Integrity: Real-World Implications - Xiaofeng Zheng, Jian Jiang, Jinjin Liang, Haixin Duan, Shuo Chen, Tao Wan, Nicholas Weaver; USENIX Security 2015
- HTTP Cookie - wikipedia  
  https://en.wikipedia.org/wiki/HTTP_cookie
- Same Origin Policy  
  https://en.wikipedia.org/wiki/Same-origin_policy
Backup Slides
Discussion

- You went to Starbucks two weeks ago, you connected to the Starbucks’ wifi and accessed some unencrypted News website, like CNN.com; or you went to Los Angeles for vacation, you spent some time in your hotel room and visited CNN.com using the hotel’s public wifi.

- You do online banking today, at home. Suppose your home network is secure, suppose the online banking is secure - your traffic is over HTTPS, you do not click on any malicious links, you do not access any other websites. You initiate an online banking transaction, can the money goes to an attacker’s account?