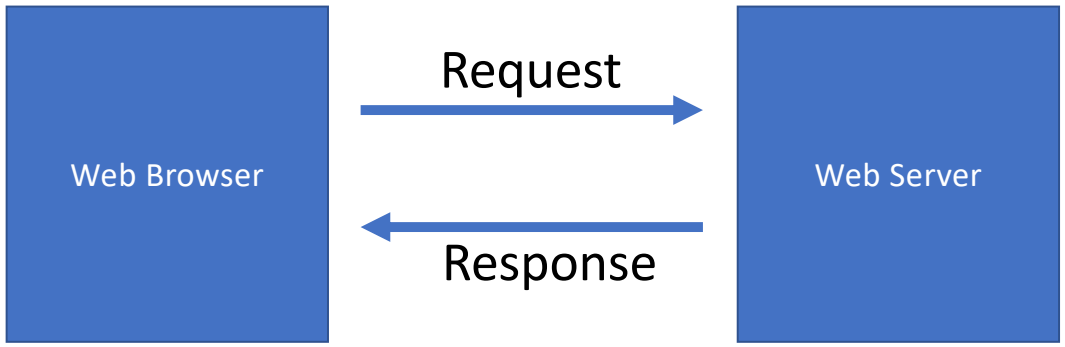
CSCE 465: HTTP Security Headers

Instructor: Abner Mendoza

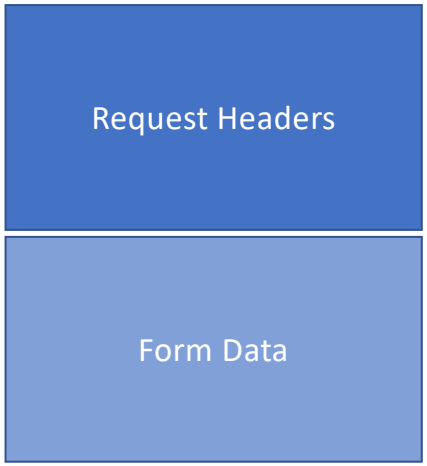
Background: HTTP

- Transport Layer protocol for Web Requests and Responses



Background: Requests

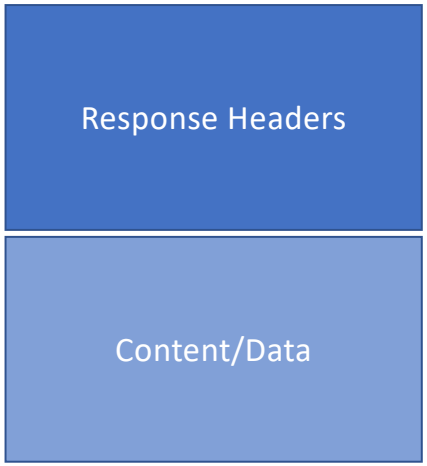
- Example: Request to www.facebook.com



GET / HTTP/1.1
Host: www.facebook.com
Connection: keep-alive
Cache-Control: max-age=0
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_13_3)
AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/64.0.3282.140 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Cookie: enduserid=23orqijfas0dfjasd0;

Background: Response

- Example: Response from www.facebook.com

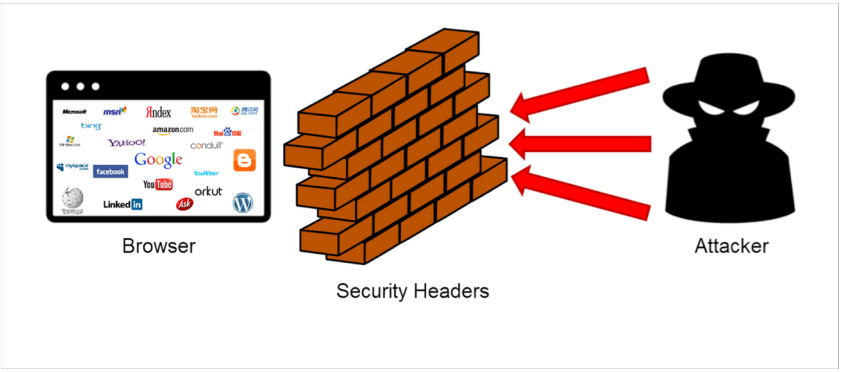


HTTP/1.1 200 OK
cache-control: private, no-cache, no-store, must-revalidate
content-encoding: br
content-security-policy: default-src * data: blob:;script-src *.facebook.com *.fbcdn.net *.facebook.net *.google-analytics.com *.virtualearth.net *.google.com 127.0.0.1:* *.spotilocal.com:* 'unsafe-inline' 'unsafe-eval' fbstatic-a.akamaihd.net
content-type: text/html; charset=UTF-8
strict-transport-security: max-age=15552000; preload
vary: Accept-Encoding
x-content-type-options: nosniff
x-frame-options: DENY
x-xss-protection: 0

HTTP Security Headers

- Special HTTP Headers in Responses from Server that instruct the Client to implement certain security controls

- Used to Protect against some common web attacks, such as Cross-Site Scripting, Click Jacking, etc.

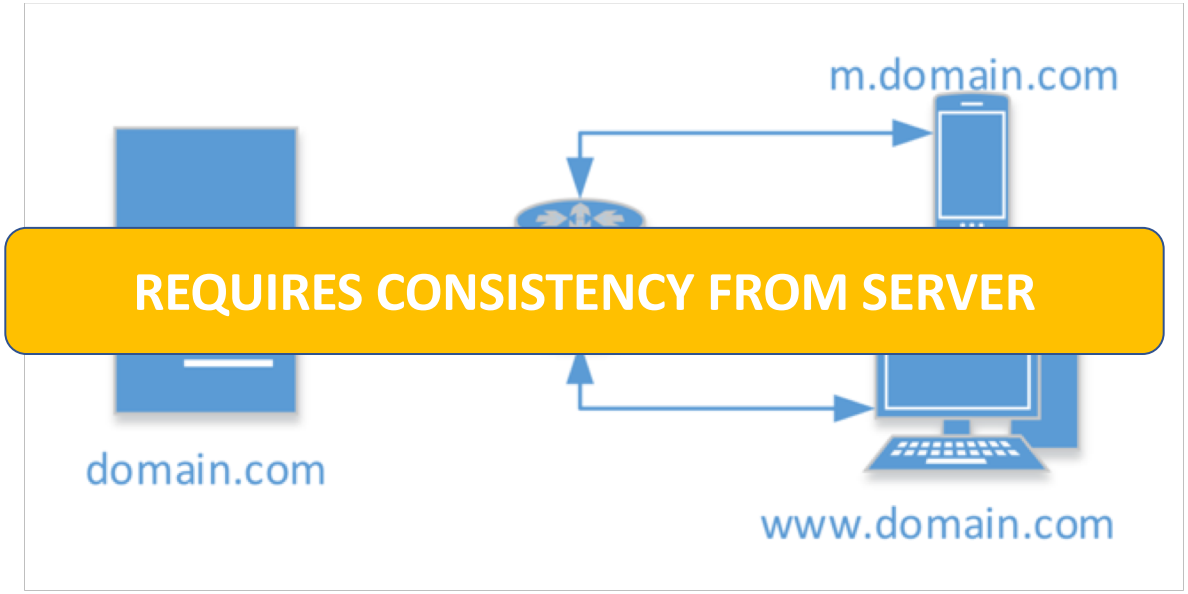


HTTP Headers: Fragile Requirements

- HTTP Headers merely **instruct** the browsers to enable or disable certain security controls.
- Requires close **coordination and consistency** between browser and server for **all user agents**.
- Server must send **properly configured** headers to the browser in responses.
- Browser must appropriately implement security controls.

Mobile and Desktop Views

- Server sends mobile or desktop view depending on the client form factor.
- Mobile and Non-Mobile views of a website shares server-side content.



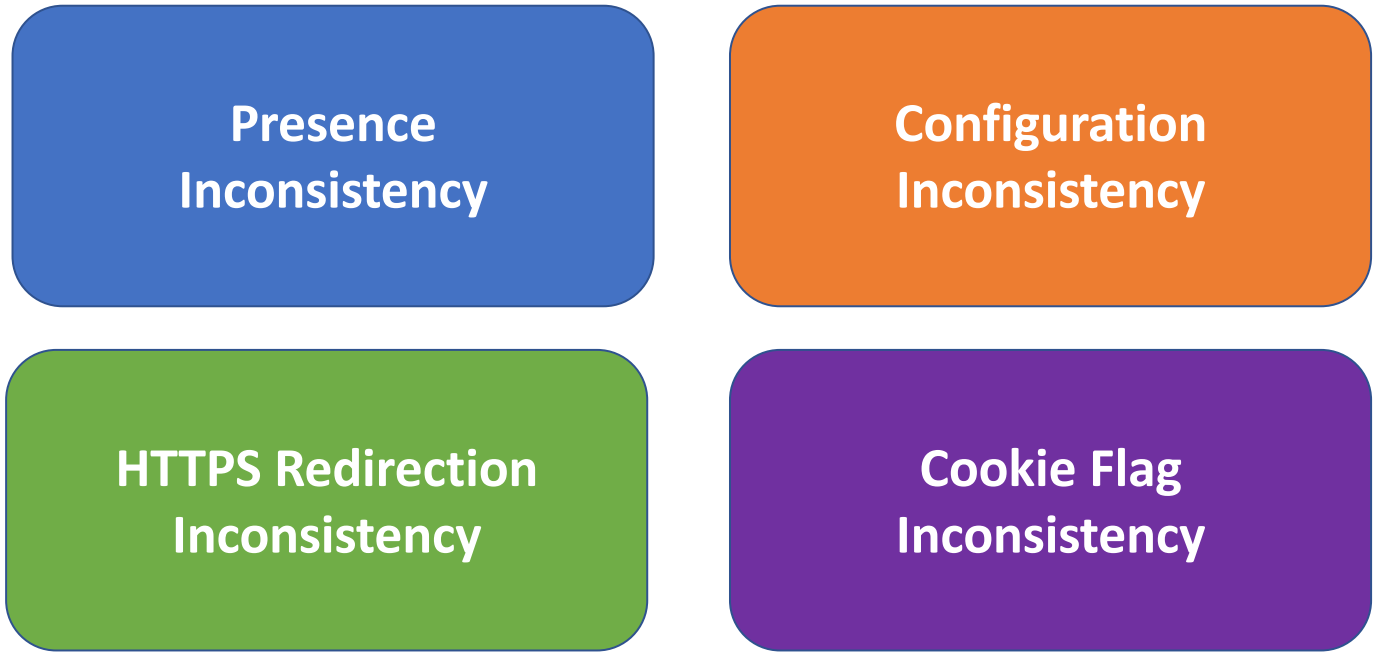
Consistency for Mobile Web

- Mobile-Optimized websites must still consistently include the same security header controls
- Difficult to coordinate if mobile and desktop sites are on different servers
- One subtle inconsistency can lead to interesting attack possibilities due to shared server resources

Our Research

- Large-Scale Analysis of the top 70,000 websites to compare inconsistency of HTTP Security Headers between mobile and desktop clients.
- Analysis of the Security Impact of HTTP Header Inconsistencies.
- Real World case studies on top websites such as Netflix.com

Inconsistency Analysis: Mobile vs Desktop



List of HTTP Headers

Headers	**Example Attacks Mitigated**
Set-Cookie	Session hijacking, Cookie stealing.
X-Frame-Options	Clickjacking
Access-Control-Allow-Origin	Cross-site access
X-XSS-Protection	Cross-site scripting
Strict-Transport-Security	Man-in-the-middle
X-Content-Type-Options	MIME sniffing.
Content-Security-Policy	XSS, CSRF

Security Impact Analysis

Threat Model

- Attacker can analyze and compare inconsistencies for a chosen website. Devises a list of attack preconditions.
- Attacker can act as an active network attacker, but cannot decrypt or modify HTTPS traffic.
- Attacker can act as a legitimate web user, and can also replay cookies from other users.

Attack Scenarios: Presence

- If an attacker knows that one version of a website does not include a certain header, the victim user can be forced to that version
- For example, an attacker can force a victim to an alternative view that does not include clickjacking protection, and then embed that view in a hidden iframe to carry out an attack
- We call this a Chosen User Agent attack

Attack Scenarios: Configuration

- Attacker observes and compares the configuration of a particular security header between two user agents.
- The attacker can redirect users to weakly configured views to carry out an attack.
- For example, if a mobile version of a site uses an improper Content Security Policy configuration, it could be vulnerable to Cross Site Scripting.

Attack Scenarios: HTTPS Redirection

- Attacker leverages the fact that one server response, depending on the user agent, always redirects to HTTPS while another response will not redirect to HTTPS.

- The Non-HTTPS response will be favored for Man-in-the-Middle attack. Data will be sent in clear text.

- For example, an attacker can can force the user to use a non-https view to steal an authentication cookie.

Attack Scenarios: Cookie Flag

- Cookies have special flags (configuration) that define how a cookie can be handled.
- **HTTPOnly** flag designates that it cannot be read by JavaScript.
- **Secure** flag designates that it should only be sent on HTTPS connections.
- An attacker can force a user to a version of the site with a weaker cookie configuration so that the cookie can be stolen.

Large Scale Measurement

Large Scale Data Collection

- Built a custom web crawler to collect HTTP Header data from the top 70,000 websites.
- Crawled using a User-Agent for Mobile and one for Desktop.
- Re-Crawled data **after one year** to compare differences. Third crawl using updated UseAgent string for new browser versions.

Data Analysis Guidelines

1. **Presence.** If a Security Header is present for one user agent, it must be present for all user agents.
2. **Configuration.** Each Security Header must be consistently delivered with the same effective configuration settings regardless of the user agent.
3. **Cookies.** If cookies are shared between mobile and desktop, they must be consistently delivered with the same settings regardless of the user agent.
4. **HTTPS.** If a website redirects to HTTPS, it must do so for all user agents.
5. The presence and configuration settings within and among different artifacts should not cause conflicting policies

Results: Presence Inconsistency

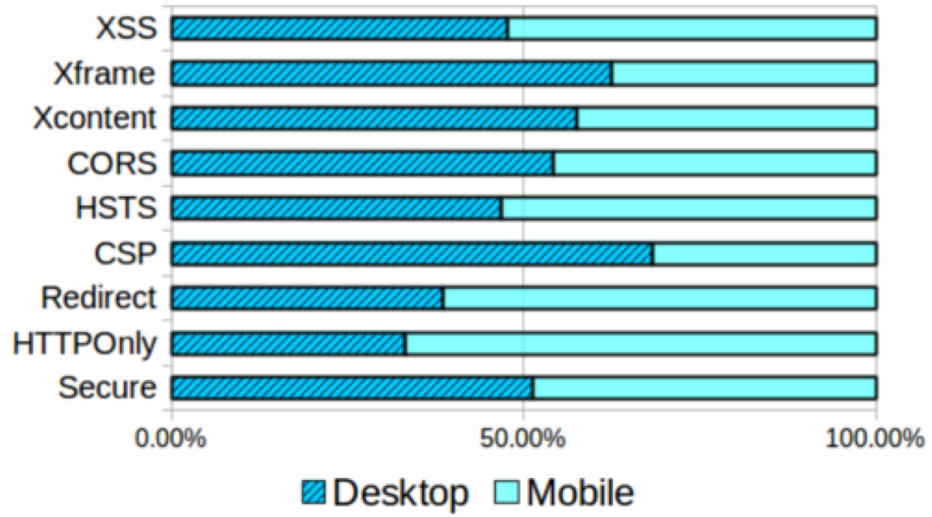


Figure 4: Percentage between desktop and mobile websites with inconsistent presence of HTTP security header

Strong vs Weak Configuration

- Comparing ‘strong’ vs ‘weak’ configuration. The rules in Table 2 define what we consider as ‘strong’ configuration for each header.

Table 2: Heuristics for appraising the Configuration of Security Headers.

Headers	Rules
X-XSS-Protection (XSS)	set to 1, additional parameter is optional
X-Frame-Options (Xframe)	does not use wild card (*) in allow-from
X-Content-Type-Options (Xcontent)	set to nosniff
Access-Control-Allow-Origin (CORS)	does not use wild card (*)
Strict-Transport-Security (HSTS)	max-age is greater than 999, additional parameter is optional
Content-Security-Policy (CSP)	both script-src and object-src must be present or default-src in their absences. does not contain unsafe-inline or unsafe-eval. does not use wild card (*) in script-src, object-src, or default-src whitelists. does not contain unsafe origin in whitelist [16].

Results: Configuration Inconsistency

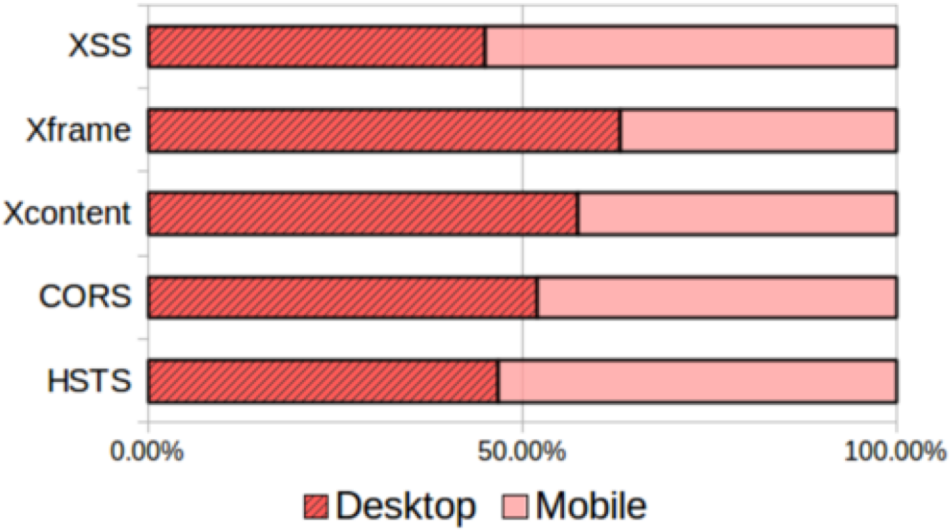


Figure 5: Percentage between desktop and mobile websites with inconsistent strong configuration of HTTP security

Results: Redirection Inconsistency

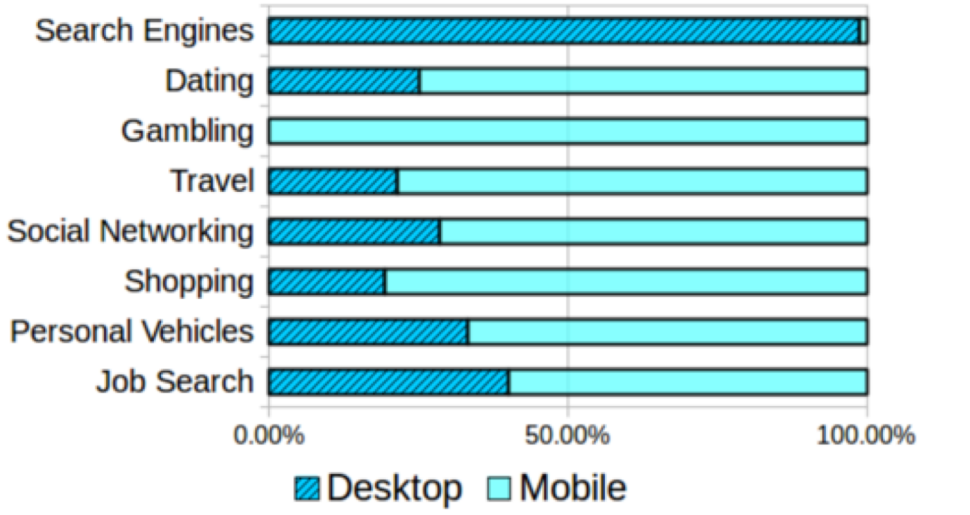
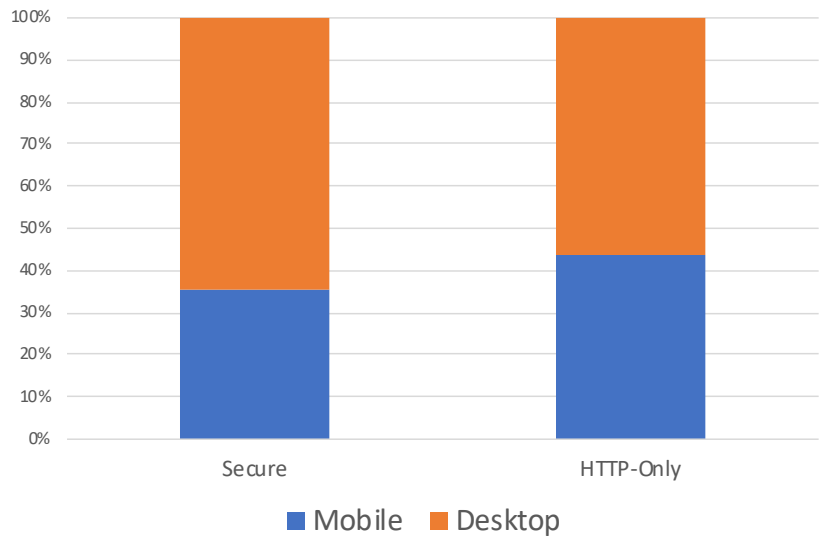


Figure 6: Percentage between desktop and mobile websites of the top 8 website categories with inconsistent redirection to HTTPS

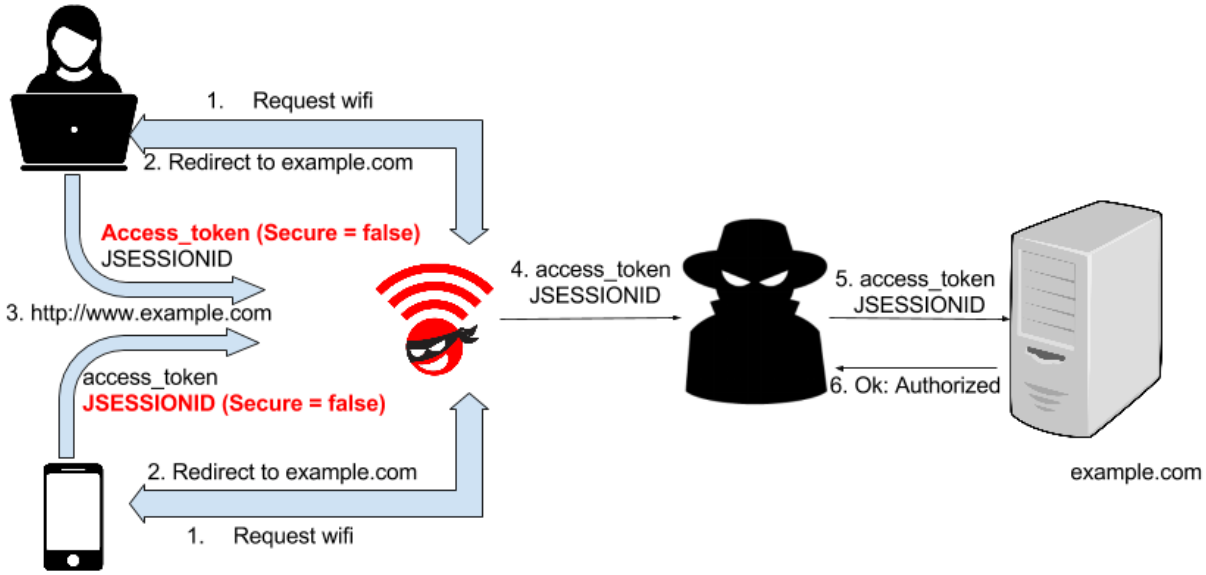
Results: Cookie Flag Inconsistencies



Case Studies

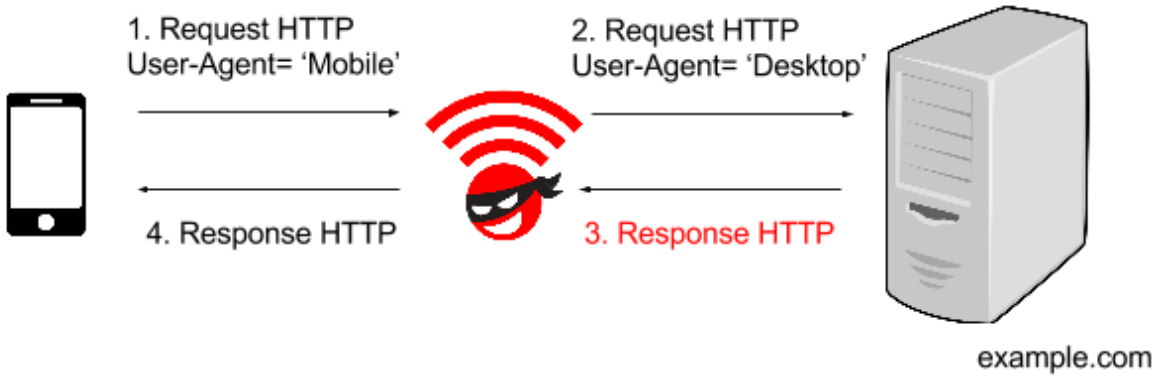
Real World Attacks

- Cross-Platform Information Aggregation on DisneyStore.com



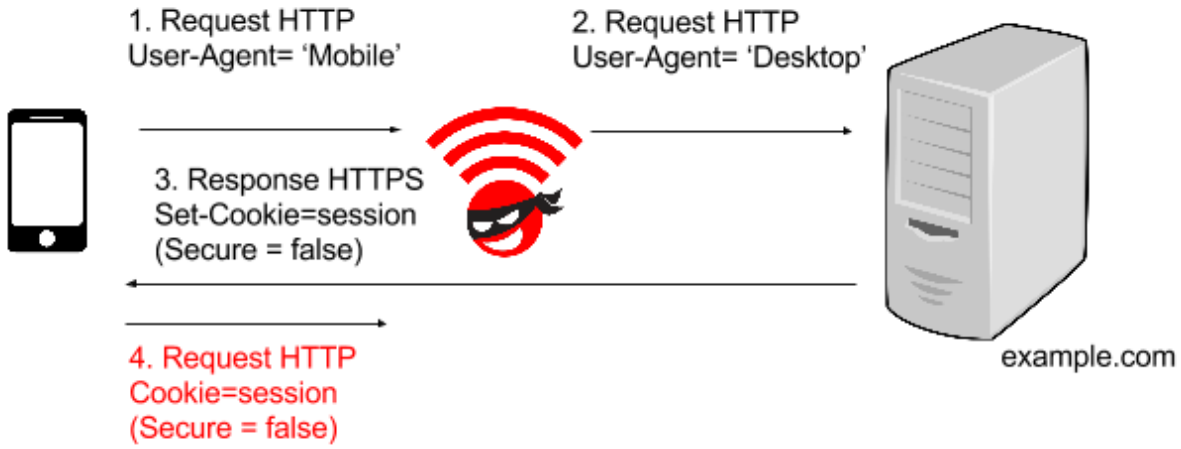
Real World Attacks

- Redirecting to a less secure view on Bet365.com



Real World Attacks

- Stealing Cookies through HTTP unencrypted connections



Some Affected Websites

- Netflix.com, DisneyStore.com, Bet365.com, Google.com
- Acknowledged by Netflix and Google
- These issues are fixed on websites we tested. Netflix worked with us to deploy a fix for a cookie flag inconsistency, and acknowledged us on their website.

Conclusion

- We identified an inconsistency problem with security header configuration for mobile and desktop versions of the same website
- We conducted the first large scale measurement study of these inconsistencies on the top 70,000 websites
- We found that these inconsistencies emerge due to the complexity and flexibility of web deployment to support mobile browsers.
- Highlights need to balance performance and security needs.

Thanks for a Wonderful Semester!

web: success.cse.tamu.edu