Web-QoS2: Web-browsing Quickly and of course Safely, Too.

Zhenyu Zhou (student)
Duke University
zzy@cs.duke.edu

Theophilus Benson
Duke University
tbenson@cs.duke.edu

With the increasing concern for network security and privacy, adoption of HTTPS has sky-rocket, with over 50% of traffic flows employing HTTPS[1]. Unfortunately, websites implement HTTPS:

Blindly: All Objects are retrieved via HTTPS, regardless of size or sensitivity and in certain cases unnecessarily incurring performance overheads associated with HTTPS. (HTTPS handshake accounts for over 42% of data exchanged[1].)

With harmful consequences: HTTPS prevents network functions, e.g. caches, from inspecting packets and optimizing end-user performance. These can result in a significant increase in the load on ISP link and further increase page load times.

Carelessly: Many sites use HSTS[2] as a mechanism for automatically forcing all traffic to HTTPS. However, over 59% of websites do so incorrectly resulting in security flaws, e.g. redirect to HTTP domain or improperly set the parameter `max-age`. Surprisingly, even the sites that configure HSTS correctly are vulnerable to a number of attacks due to stale policies or modifications of time-stamps[3].

We claim that these limitations, highlight the need for alternative mechanisms for quickly and safely viewing websites: QoS2.

The main insights underlying QoS2 are that: First, websites contain both publicly known information, e.g. java-script, banner images and CSS, which are identical for all users, and relatively personalized content which is unique to subset of users, e.g. pictures or Ads. Second, the only content worth encrypting is the relatively personalized content. Moreover, these objects are often non-cachable, e.g., passwords, or may incur relatively low cache hit-rates, e.g, personal pictures and Ads.

Building on these insights, QoS2 argues for fine-grained identification of data as either public or sensitive, which are then deliver over HTTP and HTTPS respectively.

To illustrate the benefits of QoS2, we manually downloaded, identified and tagged the content from several pages and set our web-servers to serve them over HTTP or HTTPS appropriately. In Table 1, we present the improvements in load time for YouTube under different proxy settings: we observe that a scheme like QoS2 can improve load time by as much as approximately 16% when the HTTP proxy has 1.5ms delay.

Page Loading Time/ms		Delay from origin HTTPS server (in ms)		
		5	25	50
No proxy		888	1070	1420
HTTP proxy Delay (in ms)	1.5	744(16.2%)	892(16.6%)	1190(16.2%)
	5	863(2.8%)	988(7.7%)	1230(13.4%)

Table 1: Page loading time in multiple cases. The first row of result doesn't adopt HTTP. The percentage represents the improving rate.

The main challenge in applying QoS2 lies in ensuring that security is not compromised, namely we must ensure that QoS2 does not leave web sites vulnerable to **Man in the Middle** attacks. By using a combination of HTTP and HTTPS, we leave the content served over HTTP vulnerable to man in the middle attacks. Compromised content can steal information stored in cookies, perform stripping attacks or worst modify the web page's DOM. To counter MITM and similar attacks, QoS2 leverages the following insight:

HTTPS-Control Channel: must be established to support secure communication of meta-data.

Content-Checksums: All content served over HTTP is check-sumed to ensure that the content is not modified. This checksum is served over the control channel.

Active Versus Passive Content: Active content can change the contents of a page, e.g. javascript or CSS, and can be used to perform attacks on user data. Where as passive content can not execute code or modify the page, e.g images or videos. QoS2 argues that only public-passive content should be sent over HTTP, all other should be sent over HTTPS.

References

[1] Naylor, David, et al. "The Cost of the S in HTTPS." Proceedings of the 10th ACM International on Conference on emerging Networking Experiments and Technologies. ACM, 2014.

[2] Hodges, Jeff, Collin Jackson, and Adam Barth. "Http strict transport security (hsts)." URL: http://tools. ietf. org/html/draft-ietf-websec-strict-transport-sec-04 (2012).

[3] Kranch, Michael, and Joseph Bonneau. "Upgrading HTTPS in mid-air: An empirical study of strict transport security and key pinning." (2015).