HTTP Strict Transport Security Performance: Is There An Issue? Does the Performance Working Group Have Recommendations for Tuning SSL/TLS For Internet2-Class Traffic?

Joe St Sauver, Ph.D.
joe@internet2.edu or joe@uoregon.edu
Manager, InCommon Certificate Program and
Manager, Internet2 Nationwide Security Programs

http://pages.uoregon.edu/joe/hsts-perf/

Today's Question For The Performance WG

- If sites deploy HTTP Strict Transport Security, how much of a performance hit, if any, will sites experience? Will there be:

  -- increased session setup time delay?
  -- latency issues for interactive sessions using HSTS?
  (see www.semicomplete.com/blog/geekery/ssl-latency.html)
  -- decreased total throughput for bulk transfers?
  -- limits to the number of concurrent sessions?
  -- other performance issues?
  -- do we/should we care?

  Should sites tune HSTS-using servers? If so, what resources, and how? (e.g., see http://httpd.apache.org/docs/2.2/mod/mod_ssl.html#sslsessioncache for one potential item)

What About Web Server Configurations?

- What's the recommended/most cost efficient hardware for an web server load that may now be dominated by SSL traffic?
- What's better, Apache or something else, like nginx? (seemingly minor things such as choice of cipher can have a big impact, see for example http://matt.io/technobabble/hivemind_devops_alert:_nginx_does_not_suck_at_ssl/ur)
- What about thinking out of the box, such as trying GPUs to accelerate SSL performance? See http://shader.kaist.edu/sslshader/ (not clear if this software is currently publicly available)
- Are there other issues in this area where we should be providing tuning guidance to server operators using HSTS (or simply doing SSL traffic over Internet2?)

A Terrific Article: “Overclocking SSL”

- One highly recommended article: “Overclocking SSL,” http://www.imperialviolet.org/2010/06/25/overclocking-ssl.html
- As mentioned in that, are y’all paying attention to Google’s SPDY initiative? See http://www.chromium.org/spdy
  -- **“Next Protocol Negotiation”** (see http://tools.ietf.org/html/draft-agl-tls-nextprotoneg-02)
  -- **“False Start” protocol** (There’s an expired draft, see http://tools.ietf.org/id/draft-bmoeller-tls-falsestart-00.html)
  -- **“Snap Start” protocol** (expired draft: http://tools.ietf.org/html/draft-agl-tls-snapstart-00)
- See also "More Bandwidth Doesn't Matter (Much)" and "An Argument for Changing Slow Start", as are linked from the primary SPDY page

Background Material on HSTS

Certificates: They're Not Just For Critical Web Content Anymore

- For a long time, most sites only deployed certs for critical content, leaving the vast majority of routine web traffic flowing over the network unencrypted.
- Why? The usual answers (valid or not) were all or some of:
  -- we do encrypt login info, that's the only real worry, right?
  -- no secrets are being served; users don't care; why bother?
  -- why buy expensive certificates when they aren't needed?
  -- it's a hassle obtaining, installing and maintaining certs
  -- we don't want to have to accept the "performance hit" associated with doing encryption and decryption on traffic
  -- debugging problems will be harder if the traffic is encrypted
  -- encrypted traffic can't be cached or proxied
  -- incoming encrypted traffic can't be scanned for malware
  -- let me look into that/I'm too busy/I'll do it "real soon now"

Then, The World Encountered Firesheep...

- If you're not familiar with Firesheep, see http://codebutler.com/firesheep (24 October 2010)
- Firesheep is an application that does a nice job of demonstrating that encrypting just the user's login session is not enough (at least if a web site relies on cookies for authentication and access control): even if an attacker couldn't capture your username and password, they could still capture an unencrypted cookie, and after that, the attacker would then have full control of your account.
- This wasn't a new vulnerability, but creation of Firesheep made it apparent to everyone that this was a practical (rather than theoretical) worry.
- The only real solutions? Don't rely on cookies to carry critical security information -- **or** encrypt everything with https.

HTTP Strict Transport Security (HSTS)

- What we really need is a way for sites to declare that ALL traffic for their domain MUST be sent via https, and ONLY via https.
- If we just wanted to ENCOURAGE use of https on a site, a formal protocol isn't absolutely necessary. Any site could simply decide to start using https to secure the web pages on their site, and voila, it could be done. However, it's easy for a user to accidentally request a page via http (instead of https), or for a web programmer to mistakenly link to an unencrypted local web page rather than an encrypted one.
- Fortunately, HSTS provides a way to say that a site MUST use https only. See: "HTTP Strict Transport Security (HSTS)," Hodges (Paypal), Jackson (CMU) & Barth (Google), tools.ietf.org/html/draft-ietf-websec-strict-transport-sec-02 (expires February 6th, 2012)

Enabling HSTS

- Enabling HSTS on a web site that uses Apache is pretty easily done, see the description at "Adding HTTP Strict Transport Security (HSTS) to Apache Virtual Hosts," http://linux.dashexamples.com/2011/08/adding-http-strict-transport-security-hsts-to-apache-virtual-host/
- The one required additional HSTS header must be sent via an https: page – that header will be ignored if it is sent via an unencrypted http page. As a result of that requirement, and also due to limited browser support, OWASP emphasizes use of a 301 permanent redirect instead: www.owasp.org/index.php/HTTP_Strict_Transport_Security
  (note that while approach work with any browser, it doesn't rule out use of self-signed certs or other HSTS corner cases)
- That OWASP page also provides pointers to recipes for enabling HSTS on IIS, NGINX, and other web servers, too.

Non-Performance Issues To Be Aware Of When Performance Testing TLS: a) Browser Support

- To be candid about one disappointing point: browser support for HSTS is not currently really where I'd like it to be, and that's a shame, because browsers play a key role in recognizing and enforcing use of the HSTS protocol.
- ***The good news?*** HSTS *is* at least currently enabled in recent versions of Firefox and Chrome (e.g., see for example http://www.chromium.org/sts). Another point: even if you don't use a browser that support HSTS, browser non-support of HSTS shouldn't actively break anything.
- ***The bad news?*** There <u>are</u> major/important browsers that don't currently have support for HSTS: Internet Explorer, Safari, and Opera do <u>not</u> have support for HSTS at this time. (Likewise, I don't believe that HSTS has made it into many mobile device web browsers). Talk to your vendors!

b) Name-Based Virtual Hosting and https Usage

- One other consideration: when it comes to regular (non-https) hosting, many sites use name-based virtual hosting. In name-based virtual hosting, dozens or even hundreds of domains may get hosted on a single shared IP (e.g., see for example http://httpd.apache.org/docs/2.2/vhosts/name-based.html)
- Traditionally, secure web sites needed IP-based hosting, with each secure web site residing on a dedicated address. If you have lots of secure web sites, doing IP-based hosting could rapidly deplete your pool of available addresses.
- Server Name Indication ("SNI") eliminates that requirment if you're running a current secure web server and browser. See http://wiki.apache.org/httpd/NameBasedSSLVHostsWithSNI
- *Caution:* Some browsers on some operating systems do not have support for SNI. Current versions of Firefox are generally SNI-safe on most all current operating systems.

c) Mixed Scripting and Mixed Display Issues

- While you're tightening things up and promoting use of https everywhere, you may particularly want to note the problem of "mixed scripting," where an https page loads a script, cascading style sheet or plugin resource over an insecure (http, instead of https page).
- Also bad: when an https page loads an image, iframe or font over http, a related if somewhat less serious problem that's sometimes called "mixed display".
- A nice summary posting on this issue is available at "Trying to End Mixed Scripting Vulnerabilities," June 16, 2011, http://googleonlinesecurity.blogspot.com/2011/06/trying-to-end-mixed-scripting.html (the comments to that post bring up some interesting examples of prominent sites that apparently have issues in this regard)