How to get brand URLs in Google AMP cache by extending TLS for Web Package

Google has successfully rushed a solution with an implementation scheme that will display your brand URLs in your page’s Google AMP cache search results. If your business depends on Google AMP search results, you may want to consult the following guide to implementing the Web Package bundle with new Signed Exchanges. This entails either a vendor making it easy for you, or changing your TLS cryptography to match.

Be aware that Google's announcement is somewhat misleading: "Instant-loading AMP pages from your own domain." Instant loading AMP pages in this context are definitely not from your own domain but from Google's, of course. The new feature enables you to cryptographically sign a Web Package so that recent mobile versions of Google's Chrome browser will essentially trick users by swapping the true location from Google over to your domain name. Everyone wins, right? However, other browser makers are wary of this development, not without reason.

Google has a "privacy preserving" policy that extends to "instant loading" AMP when they pre-render your AMP in search results. They "delay" network events to your origin server. They'll serve assets from Google's cache and you won't receive the slightest signal your page rendered anywhere until the first click might fetch certain page resources from your server. We think that's an unfortunate but necessary concession to be able to conduct business in an era of Google search dominance.

Cons

Through the use of Signed Exchanges Google can mislead Chrome users into thinking they're located on your website with that click from search results, when in fact they're not. A second click is required before browsers perform the necessary HTTP verb "GET" to load a page from your origin server, and only then will you see a hit register in your own server logs. We're confident Google will register your ranking impressions and cache-served clicks in Google Search Console. You just have to trust it…

Security Hazard

Fooling users into thinking they are at a location when they're not is a hazardous practice with a lengthy history of

abuse. From domain squatters owning slight misspellings of popular website destinations, to phishing scam sites that fool users into giving up sensitive information, fooling users into thinking they’re at a location when they’re not is dangerous. It is for this reason other browser makers are skeptical of Google, and Mozilla has labeled Signed Exchanges in its first incarnation a security hazard.

Mozilla TLS Hazard Warning for Signed Exchanges

Pros

On the plus side, it’s nice to see there’s a concept for decentralizing web content so that Web Packages can reside for up to 90 days served from cache server reserves outside areas where there is ample networking. This enables, for example, content that is up to 90 days fresh to be stored in places as far flung as high-altitude mountain ranges, across the tundra, in distant forests, and possibly even outer space where adventurers and scientists can consume it.

Implementation in a Nutshell

The best implementation details so far can be found from video taken at the stage at #AMPConf 2019. The footage demonstrates the ease of use of Google’s new product with Cloudflare, an early adopter vendor offering free Signed Exchange certs with

a simple interface. Setting up Cloudflare requires you to edit DNS settings to point to Cloudflare's Name Servers, which reverse proxy back to your origin host. If you're able to do that, then you won't need to meddle with TLS (Transport Layer Security).

Implementing Signed Exchange Yourself

To implement the technology yourself, you'll need to change your TLS Certificate and renew every 90 days. If you're familiar with how to do this beyond clicking a "Let's Encrypt" automation service, this may be the option you opt for. DigiCert is currently the only CA (Certificate Authority) offering Signed Exchange certificates, though we imagine others will begin to appear.

DigiCert

If you don't already use Digicert, you'll need to open an account and have the ability to run a shell terminal process on your host for crafting the ECC (Elliptic Curve Cryptology) keypair and the server CSR (Certificate Signing Request). These steps are required for TLS to ensure your private key is valid for encrypting/decrypting communications associated with your domain name. DigiCert has instructions for both Apache and Microsoft.

With Microsoft, you're basically using Windows for the process so you can follow the DigiCert blog post instructions pretty easily. With Apache, on the other hand, understanding the work can be more difficult.

Instructions for Apache

To start, you'll need to keep your DigiCert order form session

handy, and generate both a private key and a certificate signing request (CSR). The output will be .key and .csr files respectively, and you'll need to make sure you have sudo permissions for switching to root on the host for this process. Therefore, you'll want to be familiar with the Unix/Linux shell environment before proceeding.

Switch User (su) to root on Linux

Generate Private Key and CSR Files

It will be necessary to operate as root. Use the "switch user" command (su) as above, with a hyphen, to switch to root before proceding. Find a private and secure working directory, such as your sudo user home equivalent: /home/username-with-sudo. As root, you should have the permissions to access it and work. Generate ECC private key with OpenSSL to output a .key file be sure to replace server with the name of your server:

```
$ openssl ecparam -out server.key -name prime256v1 -genkey
```

Next, you'll need to generate an ECC certificate signing request (CSR). When you issue the command it will ask you questions through a prompt dialog about your location (Country, State or Province, City, Organization) which will end up being what browsers display in certificate detail views. The most important field is the FQDN (Fully Qualified Domain Name). This will bind the certificate for use only with your website address.

Next, you'll need to generate a ECC certificate signing

request (CSR). When you issue the command it will ask you questions through a prompt dialog about your location (Country, State or Province, City, Organization) which will end up being what browsers display in certificate detail views. The most important field is the FQDN (Fully Qualified Domain Name). This will bind the certificate for use only with your website address.

Generate your ECC CSR. Be sure to replace server with the name of your server:

```
$ openssl req -new -key server.key -out server.csr -sha256
```

The output should be a .csr file and you should now have a .key and .csr file for use with the next steps. Tuck these files away securely in your private folder after you've used them for the next steps.

DigiCert Certificate Files

To obtain the .crt files from DigiCert, you'll need to copy the .csr file text and paste it into the correct DigiCert order form. Select Apache from among the options at DigiCert to signify the web server. You might then use the cat utility to copy the .csr text from your terminal to paste. You should get a download that you'll need to transfer (using something like scp or ftp) to your host.

```
$ cat server.csr
```

Apache Configuration

Finally, you're going to need the location and name of your Apache configuration files. We can use grep for this:

```
$ grep -i -r "SSLCertificateFile" /etc/httpd/
```

Apache Config File Locations

Your result may look different. This depiction shows hits on code comments to give you an idea of the location of an example Apache configuration file (/etc/httpd/conf.d/ssl.conf), as well as where you might locate the .crt files from DigiCert (/etc/pki/tls/certs/). If you don’t see where to locate the certificate files, try repeating grep search for “tls” in /etc, or use another search term and location, if needed. The Apache configuration file is more important.

File Location for your Certificate

Place and secure the .crt DigiCert files in a specified location where you’ll be pointing Apache using the Apache configuration file VirtualHost block. In the above case that location is: /etc/pki/tls/certs/server.crt (with another location for a localhost.crt as well). To secure the .crt files once they are in place, make them only readable by root: $ chmod 400 server.key (the depiction below actually shows read/write permissions, but you get the idea).

CRT File Permisions

.

Edit Apache Configuration File

The Apache configuration file will need to point to this location in the VirtualHost block. Vim or Emacs is recommended for this, though you can also use nano if those are unfamiliar. The Apache configuration file is relatively long. You can use vim's search with the forward slash key / in combination with VirtualHost to locate the block.

Edit so that it matches what DigiCert describes in its instruction set, for example the following. Again, don't forget to replace server with the name of your server, and make sure the IP address is correct for your server or set to something like _default_:443:

```
<VirtualHost 192.168.0.1:443>
DocumentRoot /var/www/html
ServerName www.yourdomain.com
SSLEngine on
SSLCertificateFile /etc/pki/tls/certs/server.crt
SSLCertificateKeyFile /etc/pki/tls/certs/server.key
SSLCertificateChainFile /etc/pki/tls/certs/DigiCertCA.crt
</VirtualHost>
```

Save the new configuration, test then stop and restart Apache:

```
$ apachectl configtest
$ apachectl stop
$ apachectl restart
```

(You might need apache2ctl, depending). If these steps are compatible with your current setup, you should be in business. Your setup might be entirely different, however, requiring entirely different Apache configuration settings. For example, if you're running a Ruby application with the Passenger module for Apache, you're going to need very different specifications.

About The Author

Detlef Johnson is Editor at Large for Third Door Media. He writes a column for Search Engine Land entitled “Technical SEO for Developers.” Detlef is one of the original group of pioneering webmasters who established the professional SEO field more than 20 years ago. Since then he has worked for major search engine technology providers, managed programming and marketing teams for Chicago Tribune, and consulted for numerous entities including Fortune 500 companies. Detlef has a strong understanding of Technical SEO and a passion for Web programming. As a noted technology moderator at our SMX conference series, Detlef will continue to promote SEO excellence combined with marketing-programmer features and webmaster tips.
http://feeds.searchengineland.com/~r/searchengineland/~3/DZZmepMFgjo/how-to-get-brand-urls-in-google-amp-cache-by-extending-tls-for-web-package-315640