wingolog

- subscribe
- related
  - the merry month of ma
  - meta data
  - a wingolog user's manual
  - lakewards
  - 10 years of wingolog
  - meta: per-tag feeds
  - on the new posix
  - metablog
  - biting the hand that feeds
  - I'm telling you

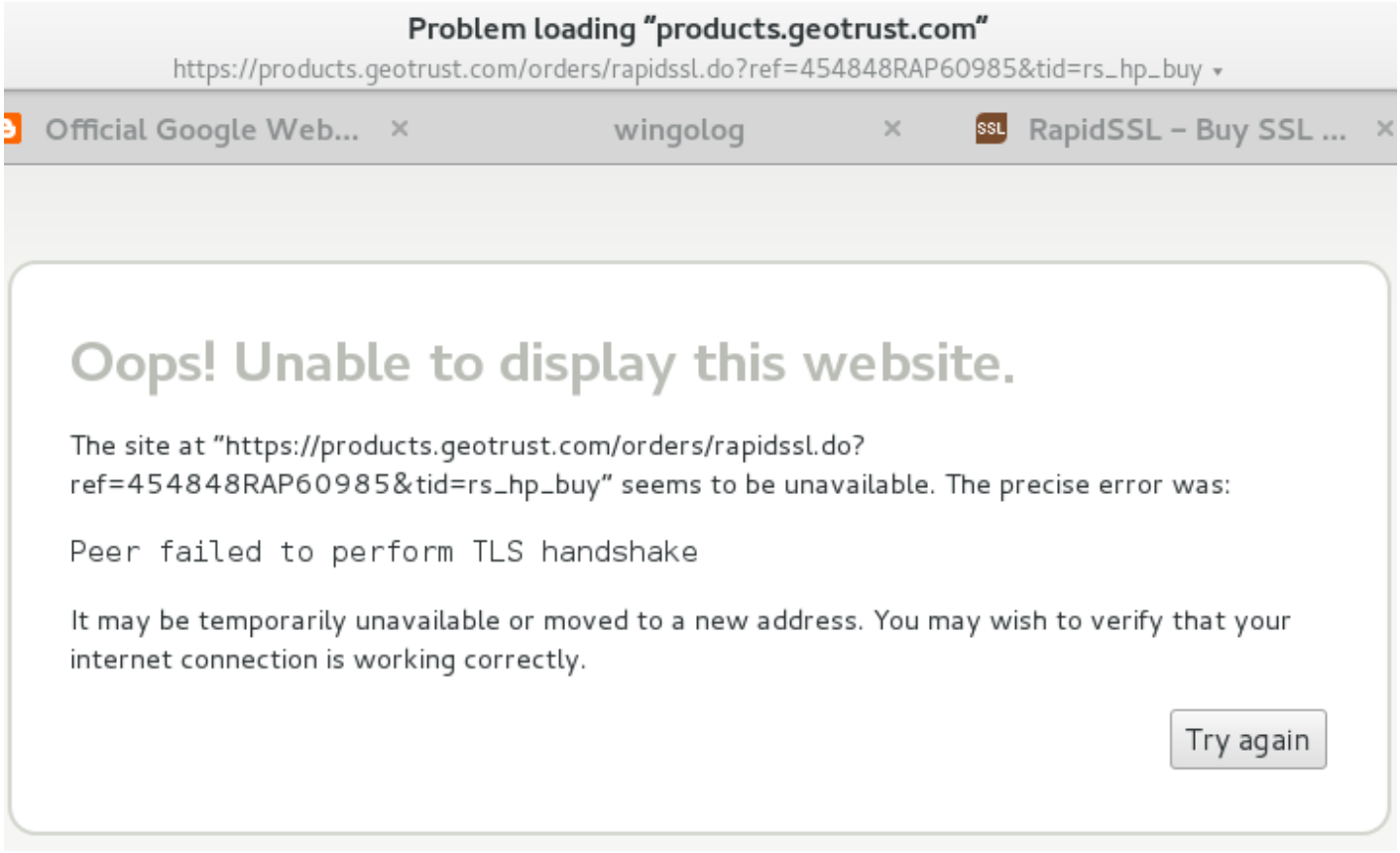
ffs ssl

17 October 2014 2:33 PM (ssl | tls | ffs | meta | https | http | hsts | pki | networking)

I just set up ~~SSL~~TLS on my web site. Everything can be had via `https://wingolog.org/`, and things appear to work. However the process of transitioning even a simple web site to SSL is so clownshoes bad that it's amazing anyone ever does it. So here's an incomplete list of things that can go wrong when you set up TLS on a web site.

You search "how to set up https" on the Googs and click the first link. It takes you here which tells you how to use StartSSL, which *generates the key in your browser*. Whoops, your private key is now known to another server on this internet! Why do people even recommend this? It's the worst of the worst of Javascript crypto.

OK so you decide to pay for a certificate, assuming that will be better, and because who knows what's going on with StartSSL. You've heard of RapidSSL so you go to rapidssl.com. WTF their price is 49 dollars for a stupid certificate? Your domain name was only 10 dollars, and domain name resolution is an actual ongoing service, unlike certificate issuance that just happens one time. You can't believe it so you click through to the prices to see, and you get this:



Whatttttttttt

OK so I'm using Epiphany on Debian and I think that uses the system root CA list which is different from what Chrome or Firefox do but Jesus this is shaking my faith in the internet if I can't connect to an SSL certificate provider over SSL.

You remember hearing something on Twitter about cheaper certs, and oh ho ho, it's rapidsslonline.com, not just RapidSSL. WTF. OK. It turns out Geotrust and RapidSSL and Verisign are all owned by Symantec anyway. So you go and you pay. Paying is the first thing you have to do on rapidsslonline, before anything else happens. Welp, cross your fingers and take out your credit card, cause SSLanta Clause is coming to town.

Recall, distantly, that SSL has private keys and public keys. To create an SSL certificate you have to generate a key on your local machine, which is your private key. That key shouldn't leave your control -- that's why the DigitalOcean page is so bogus. The certification authority (CA) then needs to receive your public key and then return it signed. You don't know how to do this, because who does? So you Google and copy and paste command line snippets from a website. Whoops!

Hey neat it didn't delete your home directory, cool. Let's assume that your local machine isn't rooted and that your server isn't rooted and that your hosting provider isn't rooted, because that would invalidate everything. Oh what so the NSA and the five eyes have an ongoing program to root servers? Um, well, water under the bridge I guess. Let's make a key. You google "generate ssl key" and this is the first result.

```
# openssl genrsa -des3 -out foo.key 1024
```

Whoops, you just made a 1024-bit key! I don't know if those are even accepted by CAs any more. Happily if you leave off the 1024, it defaults to 2048 bits, which I guess is good.

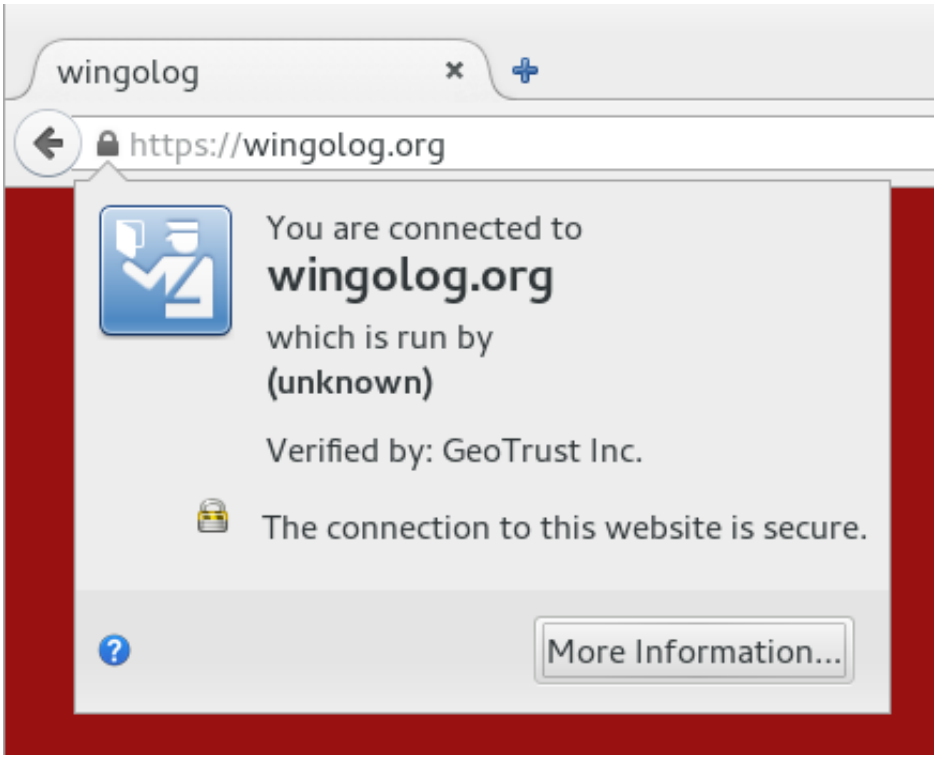
Also you just made a key with a password on it (that's the `-des3` part). This is eminently pointless. In order to use your key, your web server will need the decrypted key, which means it will need the password to the key. Adding a password does nothing for you. If you lost your private key but you did

have it password-protected, you're still toast: the available encryption cyphers are meant to be fast, not hard to break. Any serious attacker will crack it directly. And if they have access to your private key in the first place, encrypted or not, you're probably toast already.

OK. So let's say you make your key, and make what's called the "~~CRT~~CSR", to ask for the cert.

```
# openssl req -new -key foo.key -out foo.csr
```

Now you're presented with a bunch of pointless-looking questions like your country code and your "organization". Seems pointless, right? Well now I have to live with this confidence-inspiring dialog, because I left off the organization:



Don't mess up, kids! But wait there's more. You send in your CSR, finally figure out how to receive mail for `hostmaster@yourdomain.org` because that's what "verification" means (not, god forbid, control of the actual web site), and you get back a certificate. Now the fun starts!

How are you actually going to serve SSL? The truly paranoid use an out-of-process SSL terminator. Seems legit except if you do that you lose any kind of indication about what IP is connecting to your HTTP server. You can use a more HTTP-oriented terminator like bud but then you have to mess with X-Forwarded-For headers and you only get them on the first request of a connection. You could just enable mod_ssl on your Apache, but that code is terrifying, and do you really want to be running Apache anyway?

In my case I ended up switching over to nginx, which has a startlingly underspecified configuration language, but for which the Debian defaults are actually not bad. So you uncomment that part of the configuration, cross your fingers, Google a bit to remind yourself how systemd works, and restart the web server. Haich Tee Tee Pee Ess ahoy! But did you remember to disable the NULL authentication method? How can you test it? What about the NULL encryption method? These are actual things that are configured into OpenSSL, and specified by standards. (What is the use of a secure communications standard that does not provide any guarantee worth speaking of?) So you google, copy and paste some inscrutable incantation into your config, turn them off. Great, now you are a dilettante tweaking your encryption parameters, I hope you feel like a fool because I sure do.

Except things are still broken if you allow RC4! So you better make sure you disable RC4, which

incidentally is exactly the opposite of the advice that people were giving out three years ago.

OK, so you took your certificate that you got from the CA and your private key and mashed them into place and it seems the web browser works. Thing is though, the key that signs your certificate is possibly not in the actual root set of signing keys that browsers use to verify the key validity. If you put just your key on the web site without the "intermediate CA", then things probably work but browsers will make an additional request to get the intermediate CA's key, slowing down everything. So you have to *concatenate the text files with your key and the one with the intermediate CA's key*. They look the same, just a bunch of numbers, but don't get them in the wrong order because apparently the internet says that won't work!

But don't put in too many keys either! In this image we have a cert for `jsbin.com` with one intermediate CA:

And here is the same but with an a different root that signed the GeoTrust Global CA certificate. Apparently there was a time in which the GeoTrust cert hadn't been added to all of the root sets yet, and it might not hurt to include them all:

Thing is, the first one shows up "green" in Chrome (yay), but the second one shows problems ("outdated security settings" etc etc etc). Why? Because the link from Equifax to Geotrust uses a SHA-1 signature, and apparently that's not a good idea any more. Good times? (Poor Remy last night was doing some basic science on the internet to bring you these results.)

Or is Chrome denying you the green because it was RapidSSL that signed your certificate with SHA-1 and not SHA-256? It won't tell you! So you Google and apply snakeoil and beg your CA to reissue your cert, hopefully they don't charge for that, and eventually all is well. Chrome gives you the green.

Or does it? Probably not, if you're switching from a web site that is also available over HTTP. Probably you have some images or CSS or Javascript that's being loaded over HTTP. You fix your web site to have scheme-relative URLs (like `//wingolog.org/` instead of `http://wingolog.org/`), and make sure that your software can deal with it all (I had to patch Guile :P). Update all the old blog posts! Edit all the HTMLs! And finally, green! You're golden!

Or not! Because if you left on SSLv3 support you're still broken! Also, TLSv1.0, which is actually greater than SSLv3 for no good reason, also has problems; and then TLS1.1 also has problems, so you better stick with just TLSv1.2. Except, except, older Android phones don't support TLSv1.2, and neither does the Googlebot, so you don't get the rankings boost you were going for in the first place. So you upgrade your phone because that's a thing you want to do with your evenings, and send snarky tweets into the ether about scumbag google wanting to promote HTTPS but not supporting the latest TLS version.

So finally, finally, you have a web site that offers HTTPS and HTTP access. You're good right? Except no! (Catching on to the pattern?) Because what happens is that people just type in web addresses to their URL bars like "google.com" and leave off the HTTP, because why type those stupid things. So you arrange for `http://www.wobsite.com` to redirect `https://www.wobsite.com` for users that have visited the HTTPS site. Except no! Because any network attacker can simply strip the redirection from the HTTP site.

The "solution" for this is called HTTP Strict Transport Security, or HSTS. Once a visitor visits your HTTPS site, the server sends a response that tells the browser never to fetch HTTP from this site. Except that doesn't work the first time you go to a web site! So if you're Google, you friggin add your

name to a static list in the browser. EXCEPT EVEN THEN watch out for the Delorean.

And what if instead they go to `wobsite.com` instead of the `www.wobsite.com` that you configured? Well, better enable HSTS for the whole site, but to do anything useful with such a web request you'll need a wildcard certificate to handle the multiple URLs, and those run like 150 bucks a year, for a one-bit change. Or, just get more single-domain certs and tack them onto your cert, using the precision tool `cat`, but don't do too many, because if you do you will overflow the initial congestion window of the TCP connection and you'll have to wait for an ACK on your certificate before you can actually exchange keys. Don't know what that means? Better look it up and be an expert, or your wobsite's going to be slow!

If your security goals are more modest, as they probably are, then you could get burned the other way: you could enable HSTS, something could go wrong with your site (an expired certificate perhaps), and then people couldn't access your site at all, even if they have no security needs, because HTTP is turned off.

Now you start to add secure features to your web app, safe with the idea you have SSL. But better not forget to mark your cookies as secure, otherwise they could be leaked in the clear, and better not forget that your website might also be served over HTTP. And better check up on when your cert expires, and better have a plan for embedded browsers that don't have useful feedback to the user about certificate status, and what about your CA's audit trail, and better stay on top of the new developments in security! Did you read it? Did you read it? Did you read it?

It's a wonder anything works. Indeed I wonder if anything does.

27 responses

1. *Fedor Indutny* says:
   17 October 2014 2:46 PM

   Good post, thank you!

   So, on the matter of wildcards, multiple certs, and CWND. You could configure multiple contexts in bud, each for a different domain name (using TLS protocol SNI extension, which works in the most of the browsers).

   In case of such configuration, only relevant subset of certs will be sent to the client. Keeping you with the safe ServerHello packet size.

2. *Derek* says:
   17 October 2014 3:36 PM

   You forgot to mention how ECC is totally hacked by NSA and they will be reading all your traffics if you use ECDH. I really liked your post.

   On the other hand, a strong password on your private key should not be crackable, even with "fast" ciphers like AES and 3DES. Yes, it needs to be in-the-clear for the browser to use it. Perhaps you'd like to add a level of insanity to the pile with an HSM (Hardware Security Module) like the TPM or a smartcard? Configuring OpenSSL engines is something everyone does, no?

   Self-sign your certificate and be a hipster; claim you didn't want their traffic anyway, if they're scared off by someone refusing to kowtow to the CA racket.

3. *Nicholas* says:
   17 October 2014 5:18 PM

Great post, very entertaining and brightened my similarly long Friday.

Thanks!

4. *Curt* says:
17 October 2014 5:24 PM

Like your blog! Glad to know I'm not the only one that rode the SSL Merry-Go-Round! One thing to mention, the Certificate Authorities will be discontinuing the issuing of wildcard keypairs in the near future.

5. *Emanuele Aina* says:
17 October 2014 5:57 PM

I'm currently experimenting with https://www.cloudflare.com/ssl which may sidestep much of the pain described, let's see how it goes.

6. *Yousef Ourabi* says:
17 October 2014 6:17 PM

Yep - getting this right is hard.

Couple of points about the article.

Browsers verify SSL certificates (OCSP). This is an ongoing service that has a direct impact on latency - so SSL *is* an ongoing service very much like DNS. However, most people don't realize this.

Also you send in a CSR - certificate signing request - not CRT (which is usually short-hand for certificate).

Also it gets worse - A recent OpenSSL vulnerability would still allow SSLv3 even if it was configured with "no-ssl3": https://www.openssl.org/news/secadv_20141015.txt

This is why I built https://snitch.io - security and SSL secured sites in particular are moving targets and not "fire and forget". You really need an external process monitoring and auditing your secured site.

7. *geeknik* says:
17 October 2014 6:23 PM

I find that Mozilla has some great information on setting up and configuring TLS.

https://wiki.mozilla.org/Security/Server_Side_TLS

8. *Michael Buckbee* says:
17 October 2014 6:26 PM

It's even worse than that as even if all the above is actually correct and done perfectly that you may still not get the "green" depending on the browser as the browser makers are moving to showing domain validated (via email) certificates as gray and only EV (extended validation) certs as green.

Things to note: EV certs are much more expensive. You cannot get a cert that is both EV and a Wildcard, single domains only.

Screenshots of how the various browsers show Domain vs EV certs at:

https://www.expeditedssl.com/pages/visual-security-browser-ssl-icons-and-design

9. *lifeforms* says:
17 October 2014 7:08 PM

You forgot that you probably generated SHA1 certificates, which will be deprecated soon in favor of SHA2. Time to re-issue all your certificates!

And, that finally as thanks for all your effort, you then got compromised to hell and back through Heartbleed!

SSL, who doesn't love it?

10. *Nick* says:
17 October 2014 7:18 PM

Please don't make the fundamental mistake of assuming that SSL is actually trustworthy. Trusted, yes, but trust-worthy, no.

11. *Andrew* says:
17 October 2014 8:04 PM

I'm using latest Chrome and your site is using AES_128_CBC which means it isn't getting forward secrecy. You need to reorder your chosen ciphers!

12. *harrybuttle* says:
17 October 2014 8:11 PM

Last week i went through some of the same, i wish this was written 10 days ago! To start things off i got a free cert from startssl.

Some points (and links to other sites, hope it's ok, i found them useful):

* Startssl doesn't generate the key in the browser, their web app sends the key password to their server and gets back the generated key.

* Their key generation can be skipped and you can use your really private key.

* They include a subdomain of your choice (usually www) in the free certificate, so at least that one is covered.

* nginx supports sni too, in case that's needed. But old versions of android browser (at least up to gingerbread) don't support it. To quickly check if the browser supports sni: https://sni.velox.ch/

* There are some wildcard certificates for less than 100$: http://webdesign.about.com/od/ssl/tp/cheapest-ssl-certificates.htm and with some caveats some very cheap non-wildcard too.

* This is a fresher and better community tutorial dealing only with openssl (es. passwordless key, example of csr info filling...)
https://www.digitalocean.com/community/tutorials/openssl-essentials-working-with-ssl-certificates-private-keys-and-csrs

* It may be important to generate the csr with the -sha256 openssl option, to avoid the deprecated

sha1.

* I found out the hard way the need to concatenate the intermediate between the key and the certificate: browsers worked fine, but my apps complained.

* I should really try out sslstrip and see how effective it can be... certificate pinning will help but not on first connection.

13. *Gert van Dijk* says:
17 October 2014 8:12 PM

"StartSSL, which generates the key in your browser"
You misunderstood that part! You're generating an SSL Client Certificate in your browser to authenticate with StartCom on their control panel. You should generate a private key and CSR (SHA-2!) on your server and submit only the CSR to StartCom.

14. *You Are Incorrect* says:
17 October 2014 8:21 PM

StartSSL doesn't generate your private key on the server side. You are completely incorrect about this and your misinformation is borderline libellous!

Look up the "keygen" element in the HTML5 spec, to see what really occurs (hint: the key is generated by your actual browser, and saved locally).

15. *Mathias Bynens* says:
17 October 2014 8:58 PM

> You fix your web site to have scheme-relative URLs (like `//wingolog.org/` instead of `http://wingolog.org/`),

This is actually an anti-pattern. When a document is available over HTTPS, there's no reason not to include the `https://` scheme and make sure that version is used at all times.

16. *Paula* says:
17 October 2014 9:01 PM

the "which is run by (unknown)" is not due to missing the oranization field. any non EV certificate would have said the same thing in Firefox. latest versions of FF don't have this issue.

17. *Michael Catanzaro* says:
17 October 2014 9:01 PM

https://bugs.debian.org/cgi-bin/bugreport.cgi?bug=743339 is not the cause of your TLS issue with GeoTrust, but it's known to completely break sites signed by GeoTrust certs for Debian users. So add that to your bucket of things to complain about....

18. *paul* says:
17 October 2014 9:34 PM

Generating a key in your browser does NOT upload the private key to the server. It only uploads the public key, while saving the private key to local storage in the browser configuration. It's all very clever. The purpose is for client-side authentication of browsers, not generating server certificates. Unfortunately, in part because of the cumbersome technical implementation, client side certificates in browsers never caught on, so we still use passwords.

19. *hrb* says:
17 October 2014 9:41 PM

Stopped reading when you completely missed how StartSSL lets you generate your own key and submit an CSR.

20. *Nick Desaulniers* says:
17 October 2014 9:47 PM

Everyone's all excited for Service Workers; that they will deliver us from our offline-web-app-UX sins, except me. Their requirement that they only be available to sites with HTTPS (to prevent cache poisoning), and your blog post about how user friendly setting up TLS is, make me quite doubtful that Service Workers will see wide enough adoption.

21. *Tim* says:
17 October 2014 10:21 PM

FWIW, SSLs.com is cheaper plus has completely idiot-proof line-by-line cut/paste setup instructions.

But, yeah.

22. *Kai* says:
17 October 2014 11:10 PM

I think for now we are better with TOR hidden services even if they don't have nice names and you also need to tell them to others though a secure channel, but all the rest is just crap - let's hope for GNUnet.

23. *Jeff Tsay* says:
18 October 2014 0:13 AM

Thanks for this post. I recently went through almost the same process when setting up HTTPS on my site. I got a free Rapid SSL cert with my domain registration from Namecheap, and it worked well, but I had to gather the information from various sites to get it to work. I had forgotten about the secure cookie stuff so after reading your post, I immediately added it. One thing that was helpful to me was this free SSL checker: https://www.ssllabs.com/ssltest/ which gave me piece of mind that everything was setup properly (although it didn't catch the secure cookie stuff).

It is scary that, for most developers, the steps to secure a server involves copying code snippets from random, insecure blogs all over the world.

24. *Sam whited* says:
18 October 2014 1:19 AM

Don't forget that OCSP servers can go down and cause your website to fail verification (not that it matters anyways b/c Chrome doesn't even bother to check for non-EV certs and everything else checks and then ignores it if it can't get an answer). Better enable OCSP stapling to fix this.

25. *wingo* says:
18 October 2014 8:26 AM

Well, that was fun. Some thanks:

to Ivan Ristić, on his pessimism regarding cipher/key exchange blacklists

to Yousef Ourabi, for his comment above; I didn't consider the burden of OCSP on CAs. Also I think Yousef was the only one to recognize the CSR/CRT typo, and for mentioning another bizarre OpenSSL SSLv3 downgrade bug; the idea of a proxy server written in OCaml is getting more attractive by the day.

to evvvvverrrrryyyyone that pointed out that the Firefox "unknown organization" thing is because of EV certificates. EV certificate dogpile!

And a big no-thanks to people that didn't click through to read the StartSSL article on DigitalOcean. I wasn't talking about the client-side cert. There is a screenshot there of your private key in a web browser textarea; I work on browsers for a living and there is no way you could get me to treat that as safe, for all of the reasons on the Matasano JS crypto article.

26. *lanzz* says:
18 October 2014 9:16 AM

That private key is displayed by StartSSL only when they're generating it, because, well, they need to give it to you. That won't happen if you generate your own private key and CSR.

27. *wingo* says:
18 October 2014 9:53 AM

lanzz, no argument there -- I'm told that StartSSL can be used in a secure way, and I might give it a go again one of these days. I just think it's irresponsible to have the option of generating your key in the browser (if that's the case) or on their side (probably better from a crypto perspective as the browser often doesn't have a CSPRNG but totally insecure because in that case they *definitely* have your private key).

Unfortunately, "StartSSL can be used in a secure way" in no way contradicts the fact that the first search result (for me) advises you to use StartSSL in an insecure way. There are many ways for things to go wrong if you don't know what you're about.

Leave a Reply

Name

Mail (will not be published)

Website

What's your favorite number?

Submit Comment