Fixing SixXS's SSL Cert Issue

I like SixXS, I don't like their SSL cert provider.

1. Intro

I use SixXS (https://www.sixxs.net) to get a free IPv6 tunnel and subnet. It's a great service, I've had very little downtime, and they work with the easy-to-use AICCU tunnel manager. For some more information, please see the Ubuntu Wiki (https://wiki.ubuntu.com/IPv6).

However, they've chosen to use CACert as their SSL certificate provider. Note the lack of a free advertising link. This is because CACert has done a pretty poor job at getting into browsers which means that they either aren't putting forth the effort I'd like to see to pass their audits (they do seem to be actively pursuing Mozilla), or they've failed the security audit (I don't see Chrome nor IE on their list of inclusions). Ubuntu is currently not supporting them because they issue unassured and assured certs from the same server - essentially munging together low security free stuff on the same server as their high security offerings. On the plus side, many flavors of Linux do support them so they can't be all that bad.

I prefer using StartSSL (https://startssl.com/) for my free SSL cert (https://startssl.com/) because they do have support in most major browsers, with only BlackBerry being a holdout (they support Android, FireFox, Chrome, IE, Opera, and some Apple browsers). They've also got decent pricing (https://startssl.com/?app=40) for more advanced stuff but for home use their free service is more than sufficient. You can see Installing SSL on Apache2 Under Ubuntu (https://www.ebower.com/docs/apache-ssl) on this site for information on how I've configured Apache to use a StartSSL (https://startssl.com/) cert..

OK, unpaid advertising section over (because I do need one CACert link below). CACert is annoying but some people you care about use it. What do you do about it?

2. Solutions

There are a few solutions that are viable, and several that aren't. Each has its own risks and rewards as I'll describe below.

2.1. Boycott the Website

An extreme example is to boycott sites that use CACert. I don't agree with this method because, frankly, it's not a huge deal on the sites that use CACert. If someone steals my SixXS login because they're able to break into CACert and get SixXS' data what's the worst that will happen? I'd be stuck with IPv4 connectivity and then I'd need to contact SixXS and explain what happened - but chances are an issue of this magnitude would be a very high-profile event. If a financial institution I used was using CACert I would be much more concerned, but for something that's not mission critical a self-signed cert is probably fine.

2.2. Ignore the Error

The most common solution is to just ignore the error and continue. You should take a look at the reason why the error is being thrown up each time and examine the cert, but we know that's unlikely to happen. The benefit here is you get to maintain the same level of integrity that your browser normally comes with but the security downside is that it's much easier to fall victim not only to a spoof of the website in question, but also by becoming desensitized to the error message you become more vulnerable to spoofing of other websites. Plus it's pretty darn annoying.

2.3. Create an Exception

Some browsers allow you to create an exception for individual sites. This is probably the best solution in that it allows you to trust SixXS, but not necessarily the rest of the CACert wild west. Unfortunately Chrome is not one of those browsers (under Linux) and I'm a bit too lazy so I went with the last option.

2.4. Add CACert to Your Browser

Finally, you can just add CACert to your browser as a trusted CA. The good part about this is that any CACert-provided website will now appear correctly. The bad part is that there may be a valid reason why CACert is not trusted by many browsers.

Imagine that I'm a hacker. I send out an email with CuteKitty.exe attached and while people marvel at the little kitten crawling around on your screen I edit your `c:windowssystem32etchosts` file to say that www.amazon.com should point to my IP address. You're a Linux user? Maybe I'll put some malware into a trusted repository (http://it.slashdot.org/story/10/06/13/0046256/Backdoor-Found-In-UnrealIRCd-Source-Archive) and have it go undetected for months. Malware targets an operating system, and with 95% of the desktop market Windows is the best target for trojans but don't fool yourself into thinking that a Mac or Linux is all the protection you need.

What happens now? Well, you go to www.amazon.com and it looks like Amazon, but it prompt you for your login. And then it asks to verify your credit card. And then it wants a backup card. And maybe if you type your SSN it will help authorize the card.

But you're a savvy user (or this is the not-to-distant future where every page of every website has sweet, sweet SSL encryption), you noticed that there's no lock symbol on your browser, it's not SSL enabled. If you go to https://www.amazon.com/ you'll be presented with a big red screen saying these are not the droids, er, websites you're look for. This is because the cert you'll get from my malware-enabled Amazon site won't match the databases your browser is told to look up.

Now let's say I start up my own Certificate Authority. There's nothing that prevents me from generating a certificate with all the proper credentials for www.amazon.com. If I get a popular website to use my certs by offering them for free I can get a lot of people to add my CA into their browsers. Now when they visit my malware version of Amazon the certificate tells them to ask me if it's really Amazon, and I'll say "of course it is, silly!" Obviously, as soon as I pull the trigger on the malware the jig is up - but not before I get thousands of credit cards and SSNs.

Now, CACert is in all likelihood not playing a really long con in order to steal your credit card information - if they are they need to take a long hard look at their business model. However, until browsers successfully audit them we have no idea why CACert isn't accepted. It may simply be that they have razor-thin margins and can't afford the audits, or it could be that there are gaping security holes that could allow certificate spoofing. The problem is that I've got no idea why Microsoft, Google, and Mozilla have all shunned them, but I do know that I trust their audits more than my own.

But life is about compromise. Sure, adding CACert as a certificate authority to my browser is a security risk. But realistically as near as I can tell CACert is fine for day-to-day stuff and it makes my life easier. I chose to ignore my own advice and install CACert as a CA and I still think my browser is safe enough from attacks.

CACert has instructions (http://wiki.cacert.org/BrowserClients#Linux) for doing this, but to summarize them simply run the following:

```
sudo apt-get install libnss3-tools
cd /tmp
wget -O cacert-root.crt "http://www.cacert.org/certs/root.crt"
wget -O cacert-class3.crt "http://www.cacert.org/certs/class3.crt"
certutil -d sql:$HOME/.pki/nssdb -A -t TC -n "CAcert.org" -i cacert-root.crt
certutil -d sql:$HOME/.pki/nssdb -A -t TC -n "CAcert.org Class 3" -i cacert-class3.crt
```

The first command installs the certificate management tools, the **wget** commands download the certs, and the **certutil** commands install them into your cert database. Now restart Chrome (you probably need to shutdown all windows) and head to SixXS (https://www.sixxs.net) to test it out.