## How to fix SSL

# Who's on First?

**We look at some new approaches to certificate verification.**

*By Kurt Seifried*

As you are no doubt aware, some high-profile attacks and break-ins have occurred against several SSL vendors, including Komodo, which signs about a fourth or a fifth of the world's public SSL sites. These attacks come on top of other problems over the years, like SSL certificates being sold to people who have nothing to do with the actual administration of the site in question [1]. All these attacks have one thing in common: They exploit the authenticity properties of certificates.

Currently, if you are in possession of a certificate for say **.mozilla.org* or **.google.com*, you can easily impersonate the secured versions of their sites and harvest user passwords or install malicious software on their systems (e.g., by spoofing *addons.mozilla.org* for Firefox plugins). The bad news is most certificate authorities have the same security issues they had two years ago. The good news is smart people (e.g., the EFF, Moxie Marlinspike, etc.) have been working on solutions to bypass the problems created by certificate authorities (CAs).

Wait a minute. The problems created by certificate authorities? The problem boils down to two main issues: You have to trust the CAs either completely or not at all; there is no in between. And, you have to trust them for … well … forever. In other words, when a CA like Komodo makes mistakes and gets hacked four times, you can either continue to trust them completely or stop trusting them completely (and thus have no way to verify a fifth of the Internet's SSL-enabled websites). So, by and large, users have had to continue to trust CAs like Komodo whether they like it or not. But, what if there were a better way?

## Better Standards

One proposed solution is to set more stringent standards for the issuing of SSL certificates [2]. As the EFF has shown, CAs have not done a very good job of this so far [3]. Great! Except large parts of the standard won't come into effect until 2015 and 2016; until then, you'll still be able to get certificates for sites like *localhost* and *mailserver.internal*. Maybe once these standards have been followed for a few years, they will help, but I wouldn't hold my breath.

## DNSSEC

DNSSEC has gotten a lot of press recently. It's finally been enabled at the DNS root and various TLDs have enabled it; you can even find DNS service providers that support it now. So, why not just shove those web server certificates into the DNS information (e.g., make a text record containing the certificate information or a hash of the certificate you use). Then, when users go to *https://www.example.org/*, they'll be able to verify the certificate from the web server against the information in DNS. In this case, you could even potentially cut out the CAs entirely. If the certificate is in the DNS, it must be legitimate, right?

The problems with this approach are numerous and severe. For one thing, it means you now have to trust the DNS root and TLD root registrars completely, most of which are run by organizations that are not very well known or understood and are beholden to various governments. Also, the state of security among DNS registrars is even worse than with CAs. When you register a domain name like *goog1e.com* (using a one instead of an el), no one is

## KURT SEIFRIED

**Kurt Seifried** is an Information Security Consultant specializing in Linux and networks since 1996. He often wonders how it is that technology works on a large scale but often fails on a small scale.

going to verify that you are indeed Google. Stories of DNS names being stolen through registrar transfers are common. Basically, using DNSSEC to handle the distribution and authentication of SSL certificates results in the same problems that currently exist and doesn't really improve anything.

## Google SSL Pinning

A solution that Google is pushing (and now that Google Chrome has more than a quarter of the web browser market, they can effectively create new security standards) is called SSL pinning [4]. The idea is pretty simple: Websites decide which SSL CAs they will be using and trusting (e.g., VeriSign or StartCom) and register this information with Google – and any other browser vendor supporting pinning. Thus, if someone were to break into Komodo again, for example, and issue a certificate for **.google.com*, the Chrome browser would refuse to accept it, knowing that a **.google.com* certificate should only be signed by VeriSign or StartCom.

This solution has several problems, not the least of which is that if your existing CA issues a certificate in your name to an attacker, it will be seen as legitimate. Additionally, if you want to switch registrars at some point (because your existing one is not secure), you have to change the information with Google and wait for a round of browser updates to take effect.

## SSL Perspectives

So, I'll ignore the whole CA thing for now. I'll assume for a moment that most SSL attacks are typically localized to some degree (e.g., the Iranian attacks against Google users) or at most affect 10 percent of Internet users. If users could all compare notes about the SSL certificates, it would be fairly clear when most users see one thing for several months (the legitimate certificate) and then some start seeing something else (the attacker's new certificate). If you've been to a site off and on for a year, and they still have the same SSL certificate, it's likely the correct one – or else the attacker has been spoofing it perfectly for a long time. The idea behind Perspectives [5] is that when you go to a site, you go to a third party and verify the SSL certificate there as well.

Problems exist with this approach, too, however. Every time you visit an SSL-enabled site, you tell Perspectives that you are going there, so there are some significant privacy issues. Additionally, if a certificate is legitimately changed, it will take some time before it gets reported as legitimate. Finally, Perspectives must periodically visit SSL-enabled sites to be effective, so they won't have 100 percent coverage of public sites and internal sites won't have any coverage.

## Convergence SSL

So, Moxie Marlinspike took the idea of Convergence SSL and extended it [6], solving most of the problems I've mentioned here. The basic idea for Convergence [7] is that anyone can run a notary, and the notary will respond to certificate check requests with a yes or no answer. You can also use more than one notary to validate a certificate, and you can require one, several, or all notaries to agree that the certificate is valid in order for it to be considered trusted. This approach eliminates the all-or-nothing nature of SSL CAs and Perspectives – for example, you can have five notaries configured and require that four agree. So, if one has a false positive or some other glitch, you can still use the site in question.

Another big change is that every time you check a certificate, a copy is sent to the notary. This ensures that private certificates (e.g., internal ones, or those used by captive portals like airports and hotels) can be validated. Of course, this process leads to some significant privacy concerns, which have been addressed by allowing requests to be forwarded through a notary. Essentially, you send the request to notary A who then forwards it to notary B, where the actual check takes place. Notary B doesn't know for whom it is validating the certificate, and notary A doesn't know which site is being asked to be validated. To use Convergence, you'll need a Firefox plugin; unfortunately, Chrome, IE, Safari, and other browsers are not yet supported. However, they will be once plugins or add-ons are created for them.

What I find especially interesting is that notaries can apply different or multiple criteria to validate a certificate. A notary could, for example, simply verify the certificate against the CA, or they could download the certificate at intervals to ensure it doesn't change. More in-depth checks consisting of contacting the site administrators (e.g., the person listed in the certificate or the server responses) could also take place, asking them to verify that the certificate is legitimate.

Extremely high assurance notaries also could be created, requiring business information to ensure that the certificate is legitimate (i.e., what the CAs were supposed to be doing for High-Assurance certificates). Anyone can become a Convergence notary, which I hope will result in a good selection of notaries (e.g., the EFF, Google, and privacy and security groups). The source code is publicly available on GitHub [8] and is written in Python, making it pretty easy to understand.

## Conclusions

I think the situation simply boils down to the fact that when sellers are selling trust, they have a choice. They can either make lots of money or do a good job. And so far, the choice taken has been to make lots of money. Allowing additional parties, especially if you can selectively trust them a little or a lot, is one possible solution to the mess that SSL CAs have created. ■■■

## INFO

[1] "Breach of Trust" by Kurt Seifried, *Linux Magazine*, May, 2010: *http://www.linuxpromagazine.com/Issues/2010/114/BREACH-OF-TRUST/(kategorie)/0*

[2] CA/Browser forum: *http://cabforum.org/*

[3] EFF SSL Observatory: *https://www.eff.org/observatory*

[4] SSL Pinning: *http://www.imperialviolet.org/2011/05/04/pinning.html*

[5] Perspectives Project: *http://perspectives-project.org/*

[6] BlackHat USA 2011: SSL and the Future of Authenticity: *http://www.youtube.com/watch?v=Z7WI2FW2TcA*

[7] Convergence: *http://convergence.io/*

[8] Convergence notary source code: *https://github.com/moxie0/Convergence/wiki/Running-a-Notary*