blogs.akamai.com

Enterprise Security - SSL/TLS Primer Part 2 - Public Key Certificates

4-5 minutes

Join me over the next few posts as I talk about how to provide fast, reliable, and secure applications in the branch while protecting end-users and promoting a transparent and open Internet. In Enterprise Security - SSL/TLS Primer Part 1 - Data Encryption I covered the fundamentals of data encryption. For part two we will cover certificates. Let's start with the basics.

What are certificates?

Remember we said the first phase of the TLS session protocol is the asymmetric phase. We know that asymmetric cryptography has both a private and public key. Certificates are the method through which the public half of the key pair is exchanged. Inside of the certificate is:

- The common name for the site represented by the certificate.
- The public key for the asymmetric key pair.
- Some options for the certificate (not important in this discussion).
- A Certificate Authority (CA) signature.

Notice that last item. This item is the piece of the puzzle that ensures authenticity (ie... that you are talking to who you think

you are). Without it, anyone could generate a certificate for any site on the Internet and assert that they were the real site.

A Certificate Authority is a special third party that both your browser trusts and the site owner did verification with. Think of it this way. Your browser has a list of folks built into it that it trusts to go out and verify that a certificate really belongs to whom it says it belongs to. When Company A, for example, creates a certificate for www.companya.com, they send this certificate to be signed by a trusted third party Certificate Authority, or CA for short.

That CA will call, fax, email, or do a number of other manual verification steps before they sign the certificate. Once they sign the certificate, they are alleging that they have taken the necessary steps to verify that the person who generated that specific www.companya.com certificate really is the Company A.

Thus, when you visit an HTTPS site and a certificate is sent to you in the asymmetric phase of the handshake, your browser extracts the public key and checks the signature on the certificate itself to see if a Certificate Authority that it trusts signed it. If no CAs that the browser trusts signed it, it will give you a warning. If a trusted CA signed it, the browser will give you a lock icon (assuming there are no other problems)!

What are self-signed certificates?

The name is self-describing! Self-signed certificates are certificates that have no explicit CA signer, but rather vouch for themselves. This is an incredibly bad security practice and should be avoided as much as possible. Why?

The first time a browser sees a self-signed certificate, by definition it doesn't have a known CA to compare it to. That guarantees the user will see a warning and have to click through it to see the site. But how does the user know this is a valid site and not just anyone doing a self-signed certificate and intercepting their communications? They don't! And they get in the habit of accepting any self-signed certificate they see, even ones created by real adversaries when that happens.

This practice trains users to click through security warnings. It desensitizes them to real security concerns, removes the protection true CAs give, and sets them up to ultimately be Man in the Middled by adversaries when the time is right.

Join me next time when I will cover Man in the Middle and how it impacts the enterprise.