[Insider Series]

Wayk Den Certificate Authority

VALIDATION CAN NOW BE DONE AUTOMATICALLY

At Devolutions, we like to innovate by making complex security accessible to everybody. In Wayk Now 2.3, we introduced a feature that appears small on the surface but required a lot of work under the hood: the Wayk Den certificate authority. For most users, the only noticeable difference is that when making a peer-to-peer connection through the Wayk Den, the certificate window should no longer be shown because validation can now be done automatically.

This is a big deal because a lot of products on the market either do not use certificates or use them without enforcing proper validation. The reason is simple: when a certificate cannot be validated, the user should normally inspect and validate it manually. However, explaining to end users how an X.509 certificate works can prove to be quite tricky, leading to bad security practices like ignoring certificate validation errors to avoid support calls from confused users.

Quick Security Primer

To provide a satisfying explanation of how this works, some basic knowledge of security concepts is required. This topic can be difficult to grasp even for a technical audience, which is why we should start with a quick review of the relevant concepts, starting with HTTPS.

- **HTTPS:** the secure version of HTTP, the main protocol used for the Web, relies on TLS for its security.

- **Transport Layer Security (TLS):** an industry standard that defines how to provide encrypted communication using known encryption ciphers and key exchange algorithms.

- **Secure Socket Layer (SSL):** the name that was used in previous versions of TLS.

- **X.509:** the standard that defines the format for certificates used in TLS. In practice, the certificates are often called SSL/TLS certificates, and many different file formats for X.509 certificates exist.

- **Public key cryptography:** a cryptosystem using a key pair formed by a public and a private key. The public key can be shared and is used to encrypt data, but only the owner of the private key can decrypt it afterwards. The private key can also be used to produce a cryptographic signature that can be validated using the associated public key.

- **Public key infrastructure (PKI):** an infrastructure built on top of public key cryptography meant to structure how keys can be distributed and trusted. The most common form of a public key infrastructure is an X.509 certificate authority.

- **Domain Name System (DNS):** the naming system and protocol used to provide name mapping between human-readable hostnames such as google.com and the corresponding IP address on the network.

- **Self-signed certificate:** a certificate signed by itself instead of a third-party certificate authority, making it unsuitable for automatic validation.

Let’s start at the beginning, assembling the whole system piece by piece:

- Public key cryptography makes it possible to encrypt and send data to someone without having to send the decryption key alongside, ensuring that only the owner of the private key can decrypt messages.

- In order to encrypt data to be sent, the sender needs to acquire the public key from the receiver. This public key is normally sent in clear text over the network, where it could be substituted by an attacker.

- Since the public key is not known ahead of time, the sender cannot tell the difference between the intended public key and the attacker’s public key. In the case of a man-in-the-middle (MITM) attack, the public key is substituted by another one for which the private key is known, making decryption possible.

- A third-party creates a key pair and uses it to make a cryptographic signature of the intended public key of the receiver. The public key of this third-party is installed on the sender and marked as trusted. When the sender inspects the public key sent by the receiver, it checks that the associated cryptographic signature was made by the trusted third-party key pair.

- Public keys are stored and distributed inside X.509 certificates along with other information such as the subject name, issuer name, and valid dates. Trusted third-parties are called certificate authorities and issue certificates corresponding to specific domain names after verifying ownership. Websites can request certificates for their own domain names like google.com from any of the certificate authorities trusted by major browsers.

To summarize, certificate authorities are built entirely on the trust that they issue only certificates for domain names truly owned by the one making the request. When a browser loads google.com, it first resolves the corresponding IP address using DNS, and proceeds to make a TLS connection. During the TLS handshake, the client receives the X.509 certificate from the server, and checks the following:

- The certificate was signed by a trusted certificate authority.
- The certificate name corresponds to the name of the server (google.com).
- The certificate is not currently expired.

If any of these checks fail, most browsers show a very scary warning and prevent access to the website because communication could potentially be intercepted and decrypted.

Peer-to-Peer Certificate Validation

In the previous section, we have seen that one important part of certificate validation is the matching of the certificate name and the domain name of the website. This works well in the case of a public website like google.com, where the hostname can be resolved by any computer connected to the internet. Certificate authorities are not restricted to websites, and it is possible to create private certificate authorities for other purposes such as remote desktop on a local network. For instance, it is possible to configure a certificate authority in a Windows domain and use group policies to deploy the proper certificates on all machines on the network. In this case, certificate validation uses the name of the machine on the local network and the private certificate authority, and no certificate window is shown when connecting with RDP.

Automatic validation of certificates becomes problematic for the following reasons:

- The certificate authority is not trusted on the machine initiating the connection. The certificate authority certificates used for websites are usually preinstalled and trusted on all machines, but a private certificate authority requires a trusted distribution method for all machines that need it.

- The name used to reach the target machine does not match the certificate name. For instance, a connection to the machine named "workstation101" on the local network may work, but when accessed by IP address through an external port forward or network tunnel, the name doesn't match.

- Of course, things works well if all machines are joined to the same domain and a VPN is used when working remotely, such that the names always match. However, in practice, things don't necessarily work so well. In fact, it becomes almost impossible to validate certificates automatically in the case of machines on different networks where no reliable naming system and common certificate authority exists, which is what normally happens in a peer-to-peer connection.

Animal Farm

All protocols are equal, but some are more equal than others. Inside the Wayk Team, we like to give funny names to our projects instead of boring self-descriptive titles. The Call Over Wire (COW) protocol is something we have been working on for quite some time to provide a secure, reliable peer-to-peer network where the different nodes can make Remote Procedure Calls (RPC) with each other in an extensible way. There is a long list of reasons for designing our own protocol, but one of them was to create a certificate authority that could be trusted among the peers connected to the network because it could ensure that the network entity is reliable and the same one used to initiate peer-to-peer connections.

The COW RPC protocol is core to our Wayk Den infrastructure and how the Wayk Now peer-to-peer works. When Wayk Now connects to the Wayk Den, it first makes a secure WebSocket connection to a COW RPC router, resolves the "den" network service and makes a call to obtain a 6-digit ID which will then become its network identity in the COW RPC network. This ID contains digits, but it is in fact just a string: in the future, we will support authenticated identities such as "user@domain.com". Internally, we have our own equivalent of IPv4, DHCP and DNS, all over the WebSocket connection, completely independent from the underlying network on top of which it is built. Since we can ensure that once a name has been assigned to a specific address in the network it cannot be spoofed, we have a reliable naming system inside the COW RPC network.

When a peer-to-peer connection is initiated, the client needs to create a session with the Wayk Den, which will then authorize the connection and contact the server. A server does not accept incoming connections that were not first authorized by the Wayk Den. This is by design: we didn't want to make peer-to-peer connections an open bar; we wanted all connections to be authorized by a central authority for auditing purposes and policy enforcement. We haven't started leveraging this yet, but it will be used when we start offering the Wayk Den on-premises.

Icing on the Certificate

If you are still reading this, first of all, congratulations! As we said earlier, this is not an easy topic, but we have now covered enough to explain how the Wayk Den Certificate Authority truly works.

The Wayk Den service makes use of CFSSL, the Cloudflare PKI toolkit to create a certificate chain: a root certificate authority and an intermediate authority signed by the root. The public portion of this certificate chain is required to perform validation of certificates signed by the intermediate authority, and it is fetched automatically by Wayk Now when it connects to the Wayk Den.

Wayk Now then checks if it already has a valid certificate for its identity in the COW RPC network. If it doesn't, it creates a certificate signing request and sends it to the Wayk Den to receive the final certificate that will be used for server peer-to-peer connections. A certificate signing request is like a half-signed document: a key pair is generated, and the public portion of the certificate is sent to the certificate authority to be signed, but the private key is never sent. The end result is a complete certificate that can be validated against the Wayk Den certificate authority, and a name that matches the one used for peer-to-peer connectivity.

To summarize, here is what happens in a peer-to-peer connection using the Wayk Den certificate authority:

- Alice connects to Wayk Den, obtains the ID “123 000” and fetches the certificate chain. It then sends a certificate signing request for the name “123 000”. Wayk Den verifies that Alice is named “123 000” in the network and signs the certificate.

- Bob connects to Wayk Den, obtains the ID “456 000”, fetches the certificate chain. It then sends a certificate signing request for the name “456 000”. Wayk Den verifies that Bob is named “456 000” in the network and signs the certificate.

- Alice initiates a connection to Bob using the name “456 000” by contacting Wayk Den first, then contacting Bob once authorized. A peer-to-peer connection is established outside of the COW RPC network using NAT traversal protocols (ICE/STUN/TURN).
- When the peer-to-peer connection reaches the TLS handshake, Bob uses the certificate that was requested from the Wayk Den certificate authority. Alice uses the certificate chain that was initially fetched from Wayk Den as trusted chain to validate Bob’s certificate. Alice also checks that the certificate name is “456 000” and that the certificate is not expired.

The TLS handshake is completed without having to show the certificate because it could be automatically validated.

Closing Thoughts

The Wayk Den certificate authority was a lot of work “just to hide certificates” but we feel it was well worth the effort. As far as we know, we are one of the few, if not the only one, to provide such a feature enabling automatic validation of certificates in a peer-to-peer use case.