Rick van Rein

Published

Thu 07 February 2019

←Home

KXOVER, the design of a protocol

// under kerberos cryptography architecture protocol tls

Much of the work in our project centers around open protocols. We use as much of what we find as is, but new ideas sometimes call for new protocols, and the design of these is a bit of a roller-coaster ride. KXOVER is an interesting example.

Although we are supportive of everyday protocols, the basis of our security is Kerberos. We have good reasons for that:

- Highly suitable for central identiy management
- Single sign-on makes proper security simple for its users
- Integrated in much modern software, directly or via SASL
- Extremely efficient integration with TLS possible

Users under a Kerberos realm need to login once a day, and after this security is not even noticed. It works for all applications using all protocols, and tickets for use of services are quietly arranged when needed, based on the daily login. This kind of system makes it doable to have one very difficult password for all work done. *What we plan to do with KXOVER is to make this mechanism usable across the entire Internet.*

Kerberos is normally used within an enclosed security realm. All users and all machines are assumed to exist in that same realm. This alone would make it unfit for use on the Internet at large, but there are a few extensions that can help a client in one realm to use services in another.

Kerberos can link independently managed realms (for instance, in the commercial product Active Directory this forms a forest) but it is currently based on manual agreement of keys between operators. To simplify matters, it is possible to link through intermediate realms, but these would then be able to decode and intervene in all Kerberos traffic.

This is not secure as a mechanism for the Internet, where middle men are unknown and so they cannot always be trusted. Even more importantly, what is still missing is a mechanism for impromptu crossover between realms, namely when a client in one realm wants to contact a server anywhere on the Internet.

We believe that the linkage between realms can be made automatically without endangering security. What we need is a facility for public key crypto, and validation of keys from hitherto unknown realms, which is precisely what DANE can do.

First stab: PKINIT

There is a mechanism for public-key crypto already embedded in Kerberos, and it is called PKINIT. It is also known as Smart Card Login on Windows systems. The smart card on one and, and the Active Directory server on the other, engage in a public-key based exchange during which they establish a secret for the duration of a Kerberos session, which usually grants access to services for one day.

PKINIT cannot be used literally as a KXOVER protocol. For one, we had to be more stringent because the original definition is too loose to be completely secure and sufficiently modern. But we also needed a few changes that would be incompatible.

Second attempt: KX-OFFER with Tailored Crypto

We therefore designed our own Kerberos messages, clearly distinguishable from any existing ones, and used those to send a KX-OFFER (Kerberos Crossover Offer) from the client's realm controller to the service's, and another KX-OFFER is sent back. The composition of the two KX-OFFER messages allows the derivation of a shared key that can be used as the basis for the crossover. Again, DANE is used to confirm the certificate used by a remote realm, even if it is formally not described for our protocol.

Clients do not need any changes. There is a widely implemented form for server referrals that hints the client to switch to another realm, and contact its realm controller for a service that could not be resolved in the client's own realm. Only the realm controller needs to be updated, which is a great advantage to the KX-OFFER mechanism.

Third charm: KX-OFFER over TLS

We ended up designing the cryptography for KX-OFFER, and were deeply involved in the precise structures when we analysed that TLS-KDH is a Post Quantum mechanism and KXOVER is not. The move to a Post Quantum mechanism is still difficult, and not just a matter of changing an algorithm; not all structures will move over; specifically Diffie-Hellman style key agreement may be different, if it exists at all. Were we going to preview all that in our design?

This was when we started looking into TLS, even if it may be quite a pull on resources, as a basic layer over which to exchange the two KX-OFFER messages, but without the tailor-made cryptographic facilities. TLS will be maintained and will certainly be updated with Post Quantum technology at some point.

As it turns out, a little-used specification exists for Kerberos over TLS which requests STARTTLS through a TCP extension mechanism. It may be designed for reasons for privacy, but can be used as a protected mechanism between realms too. What's more, a standardised label in DNS can be used to inquire if TLS is an option at all for a realm.

The idea of running simpler KX-OFFER messages over a TLS connection between realms clearly integrates quite naturally with existing standards, which is always a good sign. Also useful is that DANE is formally defined for TLS, and our previous attempts sort-of freely used it for other uses.

Our realms will now setup a TLS connection initiated by the client realm, and both client and server will present a certificate. This is still close to the PKIX certificate with just two small changes:

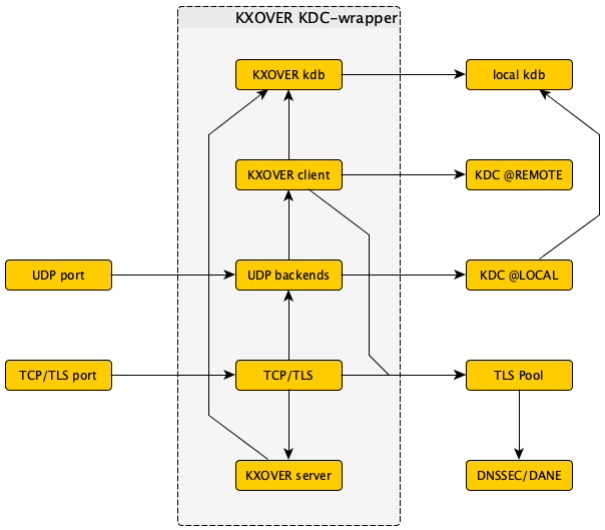
1. The certificate's Subject field is now defined, and must hold a `commonName` set to the host name of the realm controller; this is commonly used in TLS solutions to validate a remote peer's name.
2. The Subject Alternative Name lists `krbtgt/REALM@REALM` identities. We shall insist on the name-type `NT-SRV-INST` and the actual occurrence of the realm that we expect to be our remote peer.

Interestingly, TLS solutions that integrate DANE can do a lot of the work of this new approach to KXOVER. All we need to add is securely link from the realm name to the server names of the realm controllers. We will indeed use our own TLS Pool to separate the realm's private key from the KXOVER code.

Finally, the reliance on TLS means that we need not be concerned about the key exchange to derive the shared key between the client and service realms anymore; there is a facility built into TLS that exports the same shared key material from a pseudo-random number generator seeded with the master secret and some other things that we include. The master secret is the pivot of TLS security, and any cipher suite worth its salt will fight for it to be highly unpredictable for outsiders — yet have the same value for the TLS endpoints.

Simplicity is almost always a sign of a good design :)

KXOVER as a Bump in the Wire



The new design can be used as a front-end to a realm controller. When TCP/TLS and UDP pass through it to the actual realm controller, an answer stating that a name was not found can be intervened on, by looking up a server host name's realm with a _kerberos TXT query and then performing realm crossover with the remote realm. Once a key is setup, it is stored in a database with crossover keys that the realm controller can use, and the request is again sent to the KDC.

The only behaviour required inside the KDC to make this work is that it can see that it should use a crossover key. One way is through the same `_kerberos TXT` queries to detect an unknown realm and to conclude that a crossover key from the client's realm to the service's

should be looked up in a database written by the KXOVER service.

Again, simplicity is a good sign!

Conclusion

Protocol design is hard labour! It is often needed to tear up ideas that we defended with vigour in an earlier phase, and start from scratch. The iterations that this goes through tends to be useful, and it has brought us closer to the goal of KXOVER, which has always been the same:

Single sign-on with a local identity to the entire Internet.

This, we are making happen. And we are not cheating by making it web-only — we can simply step down from Kerberos to any web-only mechanism. And as with our TLS cipher suite, the work is extremely efficient, because an established crossover key between realms can be recycled at a very slow pace, but intermediately put to good use by all clients and services in the respective realms. The only (slow) public-key crypto involved would be for the crossover key, whereas the day-to-day uses by clients and services is founded on the (fast) key management mechanisms of Kerberos.