TLS for MySQL at Large Scale

Jaime Crespo
Things we are *NOT* going to talk about:

- Security and encryption fundamentals
- “At rest” encryption
- Best practices for web/HTTP encryption
- How perfectly and good we are- we made mistakes and we will present them to you
Things we *ARE* going to talk about:

- “On the wire” encryption
- Focused on for large scale web applications
- Operational/DBA point of view
- Feature requests for MySQL/MariaDB developers
- Failures that can serve as lessons learned for other ops
Why deploying TLS for MySQL?

- Privacy and security over cost - we aim for full stack encryption
- Known, documented security threads
- Compliance with modern security standards; getting modern authentication methods
TLS Myths

- TLS is slow
- TLS doesn’t work at scale
- TLS is not needed on a private network/for databases
- TLS is hard - it is not, it is mostly an operational challenge
TLS on MySQL is easy

* Latest MySQL versions even do this for you automatically
Thank you!
TLS for MySQL at Large Scale

Jaime Crespo
”The greatest failure, teacher is”

-- Yoda. Star Wars: The Last Jedi
We rushed to production.

- We were going to activate a second datacenter for the first time - people on top wanted encryption rolled in ASAP.
- We setup some initial configuration with some test certificates.
- We ended up working 3 times as much: first when we set them up, again to remove it and setup it again.
- Resources were limited: 1 full time employee (which were already in charge of all MySQL maintenance and firefighting); no external resources.
We didn’t have proper orchestration in place

- TLS at internal storage treated like rolling public HTTPS - different use case and problems
- We didn’t have a proper certificate manager service
- Older OpenSSL version had frequent security problems
- Every time OpenSSL or MySQL had to be upgraded, we had to restart the daemon
- If the change was incompatible (e.g. CA update), you had to sync client/server and master/replicas
Server support was poor

- MySQL/MariaDB older version (5.5) had problems with modern ciphers/protocols
- Only OpenSSL-linked servers had proper modern TLS support (>=1.2)
- OpenSSL was not GPL-compatible
- We had to deploy our own package (wmf-mariadb, wmf-mysql)
Client and 3rd party support was poor

- Client libraries also had to be upgraded(linked) to OpenSSL
- Some problems with clients (Mono/Sharp) silently enabling TLS for “MySQL as a service” products
- Most issues related to TLSv1.2 support
- Old client connectors (PHP5) incompatibilities
- ProxySQL did not support TLSv1.2
- Colleagues report mysql cli “no longer works”
Successes and things we did right

- We rolled TLS at first opt-in- This allowed easy rollback. We defaulted to TLS enabled, though.
- Communicated the change to fellow ops
- Organization support
- We went for replication channel and administration encryption first- indetectable overhead due to almost no reconnections
- We went for TLSv1.2 from the beginning (2015)
- 100% coverage is not rushed- we can wait for CA, licensing and client library support
- **Same-DC, non-SSL:**
  - 0.001132071018219 s/conn
  - 0.00024072647094727 s/query
- **Same-DC, SSL:**
  - 0.057012629508972 s/conn
  - 0.00025907039642334 s/query
- **Cross-DC, non-SSL:**
  - 0.1113884806633 s/conn
  - 0.036313643455505 s/query
- **Cross-DC, SSL:**
  - 0.22943157196045 s/conn
  - 0.036422135829926 sec/query
- **Local ProxySQL+Cross-DC, non-SSL:**
  - 0.0002328896522522 s/conn
  - 0.036425504684448 s/query
MySQL community wishlist

- Easier certificate/TLS library handling from the servers (#81461, #75404, #83758)
- Proper TLS 1.2+ support from connectors/clients/middleware (e.g. ProxySQL #1247)
- Proper OpenSSL 1.1+ support (#83814, #12811)
- Sharing more tests/metrics/performance benchmarks
Pending work for us

- Setup persistent connections (not only for TLS, but also for active-active cross-dc requests)
- Enable TLS also for regular connections
- Better monitoring (certificate expiration)
- Enforce TLS at grant level
- Roll in modern authentication (sha256)
Thank you!