State of the SSL Onion

Susan Hinrichs – ATS Summit Fall 2015

OpenSSL 1.0.1 vs 1.0.2

- ATS runs against 1.0.2
  - How many folks are running with openssl 1.0.2?
- Feature drivers for adoption
  - Support multiple certificate chains per context
    - RSA and ECDSA deployments
  - ALPN Support

Alternatives to OpenSSL

- Libraries that are consistent with openssl API
  - BoringSSL: Google's fork of openssl 1.0.2
    - Zwoop got this to work with ATS. Requires a few relatively minor changes in ATS.
  - LibreSSL: Forked from openssl in 2014.
- Libraries that are not consistent with openssl API
  - WolfSSL (formerly CyaSSL)
    - Small, targeting RTOS space.
  - s2n: From Amazon. Launched this summer. Replacement for libssl not libcrypto.
    - No locks or mutexes. Small code base.
    - More work due to API change, but might be interesting to test against. More likely to yield performance improvements

ATS performance limits with OpenSSL

- Benchmarks by Bryan Call showed lock contention to be limiting factor
  - “One outstanding issue was lock contention in the OpenSSL library. This shows up as a spinlock in the kernel and I traced it back to a pthread mutex lock that is being call from SSL_get_error() and ssl3_accept():”
  - We call SSL_get_error() all the time since we are non-blocking and run into NEED_READ/NEED_WRITE a lot
- Other issues?

Scaling Certificate Issues

- Seen by Steven Feltner loading over 10,000 certificates
- RedHat version of 1.0.1e has issues (TS-3554)
  - Loading lots (1000's of certificates) takes over a minute. Runs the system out of memory.
  - Build own version of openssl 1.0.1 (or 1.0.2) loading takes a few seconds. Memory use is minimal.
- Fix qsort to use median of three. (TS-3867)
  - Original version would hit pessimal sort and seg fault.
- Scaling problems with getaddrinfo and many IP addresses assigned to an interface. Bugs filed for Centos and RedHat

SSL Certificate Reload

- TS-3960 Reload based on content changes
  - Brian Geffon

Client Certificate Support

- Only global controls
  - Can only specify one client certificate to be supplied to all origin servers
  - Can only specify one set of requirements for requiring certificates from clients
- Seems like there ought to be use-cases that require different client certs for different origin servers
  - Question seems to be coming up more often.
  - Should this be a plugin-specific solution?
    - What hook? Should we add a cert selection callback for the ATS to origin connection?
  - Or is there a common use case that requires finer granularity client certificate specification?

SSL Session Ticket Keys

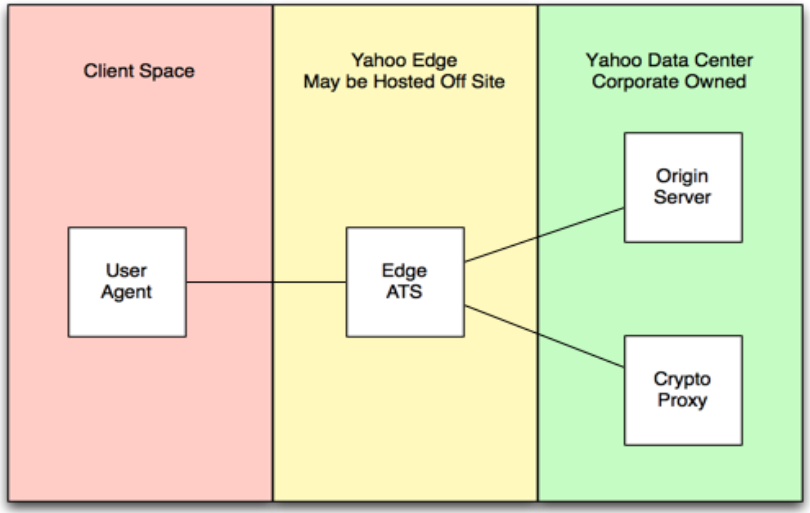
- Configuration simplification
  - Currently specify ticket keys and ticket support enable/disable on per domain basis in ssl_multicert.config
    - Bugs on disabling (TS-3742)
  - However, common case want to specify ticket keys and enable/disable ticket support globally.
    - TS-3528 - Create Global session ticket disable and discussion on IRC/mailing list.
  - Provide PluginAPI to support more complex scenarios.
- TS-3570 - Implement ATS ticket support to origin server

SSL Negotiation Refinement

- Use Cases for finer grained decision making about what attributes are available during initial session negotiation
  - Degree of certificate verification
  - Set of protocols we are willing to accept (via NPN/ALPN)
  - Set of ciphers to negotiation
- Can implement most (all?) of this via plugins.
  - Are there standard cases to be set up via static configuration
  - Add to ssl_multi_cert.config? Or another configuration paradigm?

CryptoProxy Experiments

- Move both private key storage and computation with private keys to a CryptoProxy
  - Extension of a standard crypto card scenario



Implementation

- Implement OpenSSL engine that takes over RSA operations
  - Uses default except for the private key encrypt and private key decrypt options
  - Implement network protocol to CryptoProxy for the private key operations
  - Need to do something similar for the ECDSA crypto family

ATS Implementation Issues

- SSL_accept blocks during the private key operations.
  - Including non-trivial communication delays will cause thread to block until communication returns.
  - Unlike Certificate loading, there is not a nice place to pause control and resume
- We have a dev build that spawns worker threads to do the SSL Handshake.
  - After handshake send event back to regular worker thread
  - A hassle to track operations the require the Ethread local storage (e.g. logging).
- Currently have a fixed size communication pool
  - Need to beef that up to be more resilient

No Free Lunch

- The cost of proxying the private key operation is dominated by the the RTT between ATS and the Proxy

Scenario	Average RTT between ATS and CryptoProxy	Average time to fetch small cached item
Standard Openssl Authentication	NA	74 ms
Best Case	1 ms	72 ms
Near Pod	10 ms	83 ms
Far Pod	50 ms	123 ms
Mini Pod	100 ms	173 ms

What is the real cost? When is it useful?

- From yesterday's talk, the expected "average" impact
  - For proxy_RTT = 100ms and RTT = 50ms
    - Average connection overhead increases from 32.35ms to 43.55ms
  - If we get ATS to avoid blocking working thread during handshake, the additional handshake overhead should be a constant addition to the transaction time.
- Consider adding Keyless Edge Access Point in Elbonia

	Client RTT	Proxy RTT	Ave Conn. Overhead	Ave. Time Fetch Small Cached Obj
Orig Edge	100	N/A	64.7	164.7
Elbonian Edge	50	100	43.55	93.55

Questions?

YAHOO!