Traffic Dump, Certifier and JA3

Oath:

About Me

Zeyuan Yu @dyrock

Been working on ATS (mostly plugin works) for about a year.

Traffic Dump

Purpose

- Part of the replay testing
- Captures traffic through ATS and logs in a JSON file
    - Schema here: trafficserver/tests/tools/lib/replay_schema.json
- Samples a fraction of all sessions
- Data recorded contains:
    - Timestamps
    - The four headers
    - The protocol stack for the user agent
    - The transaction count for the outbound session
    - The content block size

Implementation

- TS_HTTP_SSN_START_HOOK:
  - Set up a continuation to record this session and write beginning to file via AIO
  - TS_HTTP_TXN_CLOSE_HOOK:
    - Collect headers from ua request, proxy request, server response, and proxy response, generate JSON format output and schedule AIO write
- TS_HTTP_SSN_CLOSE_HOOK:
  - Clear continuation and write closing to file via AIO
- TS_AIO_EVENT_DONE:
  - If session closed, clean up and close file.

Configuration

- --logdir <path_to_dump> directory for dump files
- --sample <N> sampling ratio
- traffic_ctl plugin msg traffi_dump.sample <N>

Sanitizer

- Python script to do post-processing on collected replay files
- Sanitizes sensitive data (e.g. metadata, ip address)
- Concatenates multiple session files into one
    - Traffic Dump dumps data on a per session basis while the replay format allows for multiple sessions in one file
- Filters out undesired transactions
    - Specific methods
    - Malformed responses (i.e. missing critical fields)

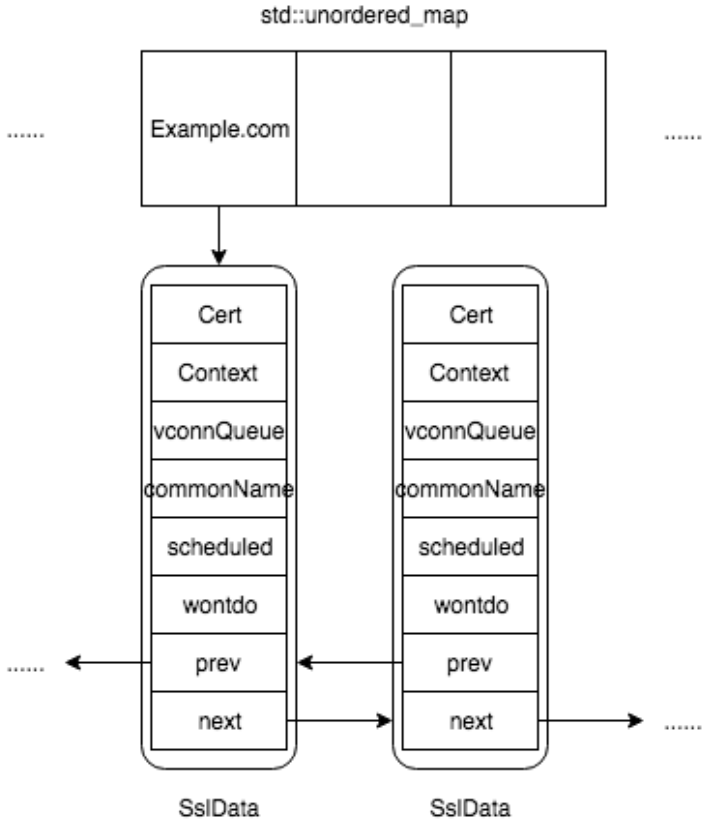
Certifier

Purpose

- Load SSL certificates from file storage on demand.
- Manage the number of certificates loaded in memory.
- Generate SSL certificates on demand with a provided root ca certificate.

Implementation

- SSL certs and contexts are managed by a map between common name and its SslData node in linked list
- Lookup and easy insertion/removal
- Most recently used node will be moved to head
- Remove tail node once the number of certificates in memory reaches the limit



Implementation

- TS_SSL_CERT_HOOK:
  - Look up SNI name of incoming HTTPS request and set up the SSL context if a valid one exists.
  - Schedule a thread to retrieve certificate from disk (or generate using SNI), or put current connection into the queue if such a thread is already scheduled.
- The retriever thread:
  - Load the file from disk. If none found and dynamic generation is enabled, try to generate a certificate and write to disk.
  - Use the certificate to set up all queued SSL connections with correct context and re-enable them.

Configuration

- Dynamic certificate generation:
  - --sign-cert <path_to_signing_cert>
  - --sign-key <path_to_signing_key>
  - --sign-serial <path_to_serial>
- Certificate management:
  - --store <path_to_certs_dir>
  - --max <N>

Setup

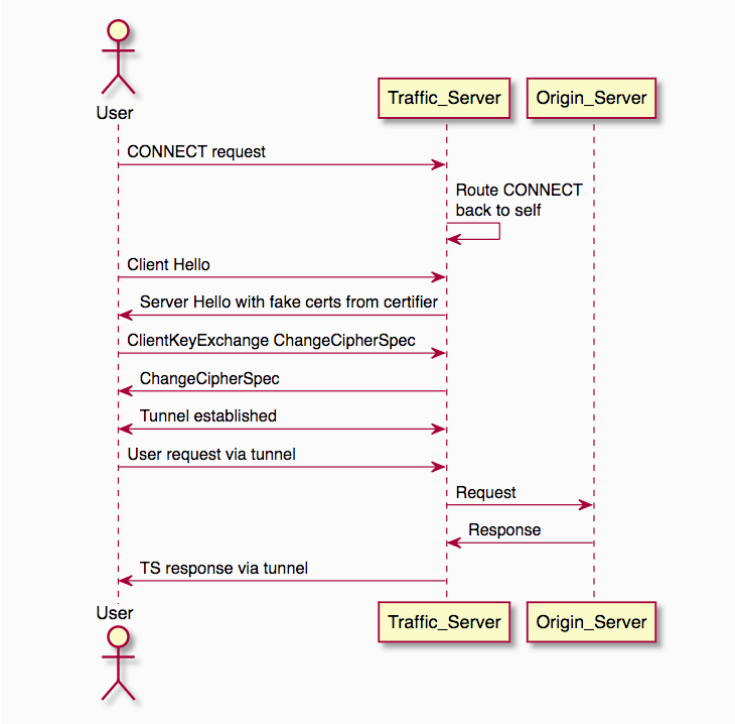
- Cert and key:
  - (In most cases, self-signed) cert and key used as CA, configured to be trusted by clients.
  - No challenge password.
  - e.g. openssl req -newkey rsa:2048 -nodes -keyout ca.key -x509 -days 365 -out ca.cert
- Serial number:
  - A text file containing a valid integer with a trailing new line
  - e.g. 12345

Client Setup

- OS
  - e.g. Linux
  - `sudo cp example.crt /usr/local/share/ca-certificates/example.crt`
  - `sudo update-ca-certificates`
- Browser
  - e.g. Firefox
  - Preferences->Privacy&Security->Certificates->View Certificates->Authorities->Import
- Others
  - e.g. `curl --cacert example.crt https://example.com`

Usage

- Man-In-The-Middle



JA3 Plugin

What is JA3

- Created by John B. Althouse, Jeff Atkinson, and Josh Atkins
- “JA3 is a method for creating SSL/TLS client fingerprints ...”
- Concatenate values in SSL Client Hello packet in order to generate the JA3 string
- MD5 hash the result to produce a 32 character fingerprint
- Malware tend to use the same encryption code/client
- An effective way to detect malicious clients
- More info: https://github.com/salesforce/ja3

Example

- Fields are concatenated in an order:
  - SSLVersion, Cipher, SSLExtension, EllipticCurve, EllipticCurvePointFormat
  - e.g.: 769,47-53-5-10-49161-49162-49171-49172-50-56-19-4,0-10-11,23-24-25,0
- MD5 hashed to a 32 character fingerprint
  - e.g.: ada70206e40642a3e4461f35503241d5

Implementation

- API for TSVConnArg:
  - Virtual connection objects support an array of void * controlled by plugins.
  - Similar behavior to TSHttpTxnArg and TSHttpSsnArg (used to be TSHttpArg)
- TS_SSL_SERVERNAME_HOOK:
  - Grab TLS Client Hello and calculate JA3 string and fingerprint
  - Store it as TSVConnArg with the SSL connection
- TS_HTTP_SEND_REQUEST_HDR_HOOK:
  - If the request is on a SSL connection, add JA3 string and/or fingerprint to the outgoing request headers

OpenSSL Dependencies

- OpenSSL < 1.1.0:
  - Use `init_msg` pointer and `init_num` length in struct SSL to access Client Hello data
- OpenSSL 1.1.1:
  - Use provided APIs to access Client Hello data
  - SSL_client_hello_get0_legacy_versions(), SSL_client_hello_get0_ciphers(), SSL_client_hello_get0_ext(), SSL_client_hello_get1_extensions_present()
- OpenSSL 1.1.0:
  - No APIs for Client Hello and opaque structure doesn't allow direct access
  - Do NOT use OpenSSL 1.1.0

Thank you!