Proposal

Removal of tx_hosts, rx_hosts and conn_uids from files.log and unrolling files.log such that each entry has a single, optional, connection uid and connection identifier.

Background

Zeek currently has the capability to associate multiple connections with a single file observed on a network. This is reflected in files.log by the fields conn_ids, rx_hosts and tx_hosts that are each of type set. While the default TSV format looks harmless for the common case (rendered as a single string), the JSON representation of sets are rendered as arrays. Further, files.log does not have the typical uid and c$id fields that most other logs have, representing a special case that can cause confusion and complications working with this log.

Sampling real-world networks, the ratio of files.log entries that have more than a single connection associated is marginal. On four university networks, out of 3.8 mio, 2.6 mio, 4.0 mio, and 1.4 mio files.log entries, the ratio of those with |conn_uids| > 0 were 0.03%, 0.05%, 0.01% and 0.08%. For these, the source field was consistently HTTP, pointing at HTTP range requests done from one client for the same file over different connections. Separately, ata was collected from a financial institution network with a ratio of 0.02% of 16mio entries and a health care provider network with 4 (!) out of 17.2mio file log entries having multiple conn_uids.

So, while there are real-world scenarios where the current logic applies, it makes the files.log more difficult to work with in the common case:

- Downstream storage systems need to account for the array type or naively convert into a string / IP when consuming TSV.
- Zeek scripts using Files::Info in log filter hooks or stream policies can not re-use c$id as for any other log. They need to iterate through rx_hosts and tx_hosts.
- More generally, filter languages for Zeek logs need special casing for files.log due to non-existence of the typical c$id fields. A basic example would be running `jq` on files.log searching for a given IP. Applies to custom filtering languages or SIEM queries.
- None of the existing Zeek testing baselines contain multiple entries in conn_uids, rx_hosts or tx_hosts. It's not very well tested.
- For what it's worth: Suricata does not have this logic for its eve.json fileinfo entries.
- files.log is not cluster aware: Different workers seeing different connections transferring pieces of the same file all create individual files.log entries with identical fuids. Essentially,files.log is partly unrolled already today: If all conn_uids for a given files.log are of interest, an analyst needs to aggregate by fuid and concatenate the conn_uids.

Specifically the last point and the complexity (and limitations) involved to properly make the aggregation cluster aware as well as the low ratio of files.log entries it applies to, leads us to propose removing the existing set fields and unrolling files.log. Correctly dealing with the special case doesn't become more complicated (and in fact explicit), while the common case becomes much simpler.

Fallout

Unrolling files.log so that tx_hosts, rx_hosts and conn_uids consistently only have a single entry shouldn't cause major issues. This is essentially the common case and fuid duplication happens in a Zeek cluster setup when two workers see different connections for the same file already today.

Removal of tx_hosts, rx_hosts and conn_uids will affect Zeek scripts relying on these, data pipelines and any analyst/SIEM workflows downstream. This is major, specifically since these fields have been around since 2013. However, working with files.log should become easier in the long term to justify these changes, still.

Deprecation Approach

The new unrolling mechanism will need to be hard-coded, with no way to revert back to the old approach unfortunately.

The removal of the tx_hosts, rx_hosts and conn_uids fields from Files::Info will be the new default, but the Zeek distribution will provide a policy script that re-adds these fields back in. Loading this script will emit a deprecation warning. The script will exist until and including v6.0. In v6.1, this transition mechanism will be removed. This is rather invasive/aggressive in that reversal is opt-in, but it promotes awareness better. It's unclear if opt-in removal of the fields, but warning by default, would be easier for users to adapt and change. There will probably be some complaints one way or the other, and this way we hear them earlier as deprecation warnings may go unnoticed longer than removed fields.

Future

- With this change in place, we wil be able to reduce the C++ and Zeek script implementation complexity to stop tracking multiple connections for files (i.e. removal of protocol specific get_file_handle()). Every File object would at most have a single connection.
- We will be able to start including information about partial downloads / range-requests within files.log. It is currently not explicitly shown (outside of guessing from seen_bytes, missed_bytes and total_bytes) that a transfer only requested parts of a file.

Implementation

The following branch contains a draft implementation of above proposal for Zeek 5.1:
https://github.com/zeek/zeek/tree/topic/awelzel/files-log-unrolling

Appendix: Protocols

Known protocols/scenarios where a file may be transferred over multiple connections:

- Bittorrent - not implemented in Zeek (?)
- Concurrent HTTP range-requests (seen as Microsoft's Windows-Delivery-Optimization)
- SMB allows reads and writes with offsets, potentially allowing concurrent downloads
- Apparently FTP when using REST (RESTART) command - not implemented in Zeek (?)