Item 31
git_comments:

1. * * This method must be implemented when you hook HOOK_TXN_CLOSE
2. *< This hook will be fired after send response headers, only for TransactionPlugins::registerHook()!.
3. reset the txn arg to prevent use-after-free

git_commits:

1. **summary:** Add TXN_CLOSE hook to CPPAPI TransactionPlugin (#6800)
   message: Add TXN_CLOSE hook to CPPAPI TransactionPlugin (#6800) * Add TXN_CLOSE hook to CPPAPI TransactionPlugin * Clean up TransactionPlugin object and associated Continuation in txn_close * Address review comments * More review comments (cherry picked from commit 34b57fccb40ef711ce2e6b31042c96efc74c0ecc)

github_issues:

github_issues_comments:

github_pulls:

1. **title:** Add TXN_CLOSE hook to CPPAPI TransactionPlugin
   body: Email to dev@ ``` We recently ran into a core dump issue due to one of our TransactionPlugin's continuations being called back with TXN_CLOSE event. The current CPPAPI's TransactionPlugin does not expect a call back on TXN_CLOSE, and asserts on it. This is because the expectation seems to be that all the continuations for TransactionPlugins attached to the Transaction are destroyed in the global TXN_CLOSE hook. However, this only works based on the plugin table in the Transaction object, which gets populated based on `Transaction::addPlugin()`. Turns out, some of our plugins do not call `Transaction::addPlugin()` and just create a `TransactionPlugin()` and register hooks to it resulting in them getting called back for TXN_CLOSE event as well. Long story short, it didn't particularly seem unreasonable to add support to TXN_CLOSE event in TransactionPlugin, since the current API does not seem to mandate calling `Transaction.addPlugin()`. https://github.com/apache/trafficserver/pull/6800 Please let me know if there are any concerns/comments. ```` Here's the stack trace ```(gdb) bt #0 0x00002aabdd0cc1d7 in raise () from /lib64/libc.so.6 #1 0x00002aabdd0cda08 in abort () from /lib64/libc.so.6 #2 0x00002aabdd0c5146 in __assert_fail_base () from /lib64/libc.so.6 #3 0x00002aabdd0c51f2 in __assert_fail () from /lib64/libc.so.6 #4 0x00002aabfb6223cf in invokePluginForEvent (event=468845984, ats_txn_handle=0x2aac6e63a070, plugin=0x2aac92d6d4e0) at utils_internal.cc:148 #5 atscppapi::utils::internal::invokePluginForEvent (plugin=plugin@entry=0x2aac92d6d4e0, ats_txn_handle=ats_txn_handle@entry=0x2aac6e63a070, event=event@entry=TS_EVENT_HTTP_TXN_CLOSE) at utils_internal.cc:222 #6 0x00002aabfb624310 in (anonymous namespace)::handleTransactionPluginEvents (cont=0x2aad52c4ce00, event=TS_EVENT_HTTP_TXN_CLOSE, edata=0x2aac6e63a070) at TransactionPlugin.cc:57 #7 0x00000000004c4833 in INKContInternal::handle_event (this=0x2aad52c4ce00, event=60012, edata=0x2aac6e63a070) at InkAPI.cc:1012 #8 0x00000000005b5b1f in HttpSM::state_api_callout (this=this@entry=0x2aac6e63a070, event=event@entry=60000, data=data@entry=0x0) at HttpSM.cc:1638 #9 0x00000000005bca82 in HttpSM::state_api_callback (this=this@entry=0x2aac6e63a070, event=event@entry=60000, data=data@entry=0x0) at HttpSM.cc:1507 #10 0x00000000004d9a69 in TSHttpTxnReenable (txnp=0x2aac6e63a070, event=TS_EVENT_HTTP_CONTINUE) at InkAPI.cc:5714 #11 0x00002aabfbea9e41 in AtsPluginUtils::creqCleanup (cont=0x2aabfb2fc3a0, event=TS_EVENT_HTTP_TXN_CLOSE, edata=0x2aac6e63a070) at ClientRequest.cc:76 #12 0x00000000004c4833 in INKContInternal::handle_event (this=0x2aabfb2fc3a0, event=60012, edata=0x2aac6e63a070) at InkAPI.cc:1012 #13 0x00000000005b5b1f in HttpSM::state_api_callout (this=this@entry=0x2aac6e63a070, event=event@entry=60000, data=data@entry=0x0) at HttpSM.cc:1638 #14 0x00000000005bca82 in HttpSM::state_api_callback (this=this@entry=0x2aac6e63a070, event=event@entry=60000, data=data@entry=0x0) at HttpSM.cc:1507 #15 0x00000000004d9a69 in TSHttpTxnReenable (txnp=0x2aac6e63a070, event=TS_EVENT_HTTP_CONTINUE) at InkAPI.cc:5714 #16 0x00002aabfb621e25 in (anonymous namespace)::handleTransactionEvents (cont=<optimized out>, event=<optimized out>, edata=0x2aac6e63a070) at utils_internal.cc:95 #17 0x00000000004c4833 in INKContInternal::handle_event (this=0x2aabfb2fcf40, event=60012, edata=0x2aac6e63a070) at InkAPI.cc:1012 #18 0x00000000005b5b1f in HttpSM::state_api_callout (this=0x2aac6e63a070, event=<optimized out>, data=<optimized out>) at HttpSM.cc:1638 #19 0x00000000005bbeba in HttpSM::kill_this (this=this@entry=0x2aac6e63a070) at HttpSM.cc:7039 #20 0x00000000005bcd80 in HttpSM::main_handler (this=0x2aac6e63a070, event=2301, data=0x2aac6e63b400) at HttpSM.cc:2958 #21 0x00000000005fee8d in handleEvent (data=0x2aac6e63b400, event=2301, this=<optimized out>) at ../../iocore/eventsystem/I_Continuation.h:153 #22 HttpTunnel::main_handler (this=0x2aac6e63b400, event=<optimized out>, data=<optimized out>) at HttpTunnel.cc:1642 #23 0x000000000076da7e in handleEvent (data=0x2aac0aed7e98, event=103, this=<optimized out>) at ../../iocore/eventsystem/I_Continuation.h:153 #24 write_signal_and_update (vc=0x2aac0aed7d00, event=103) at UnixNetVConnection.cc:179 #25 write_signal_done (vc=0x2aac0aed7d00, nh=0x2aabe160be00, event=103) at UnixNetVConnection.cc:224 #26 write_to_net_io (nh=nh@entry=0x2aabe160be00, vc=0x2aac0aed7d00, thread=<optimized out>) at UnixNetVConnection.cc:564 #27 0x000000000076ddd8 in write_to_net (nh=nh@entry=0x2aabe160be00, vc=vc@entry=0x2aac0aed7d00, thread=<optimized out>) at UnixNetVConnection.cc:421 #28 0x000000000075dba4 in NetHandler::mainNetEvent (this=0x2aabe160be00, event=<optimized out>, e=<optimized out>) at UnixNet.cc:530 #29 0x000000000078f3ef in handleEvent (data=0x2aabdf7fd8c0, event=5, this=<optimized out>) at I_Continuation.h:153 #30 process_event (calling_code=5, e=0x2aabdf7fd8c0, this=0x2aabe1608000) at UnixEThread.cc:152 #31 EThread::execute (this=0x2aabe1608000) at UnixEThread.cc:279 #32 0x000000000078dffa in spawn_thread_internal (a=0x2aabde99d6d0) at Thread.cc:86 #33 0x00002aabdc0f4dc5 in start_thread () from /lib64/libpthread.so.0 #34 0x00002aabdd18e76d in clone () from /lib64/libc.so.6 ```

github_pulls_comments:

1. In the state dump, in frame #4, it seems to be showing the parameters in the reverse order from what they are declared in the function: https://github.com/apache/trafficserver/blob/bf097d4289a3eaa3759a58fb19e48381b89ce32f/src/tscpp/api/utils_internal.cc#L113 The event parameter is not TS_EVENT_HTTP_TXN_CLOSE (60012). When the TXN_CLOSE hook is triggered, the CPPAPI destroys the instance of your custom class that is derived from TransactionPlugin. So you are supposed to put anything you want to run on that hook in the destructor of that class. Because continuations on hooks run in arbitrary order, I think your change could cause a use after free.

2. > In the state dump, in frame #4, it seems to be showing the parameters in the reverse order from what they are declared in the function: > > https://github.com/apache/trafficserver/blob/bf097d4289a3eaa3759a58fb19e48381b89ce32f/src/tscpp/api/utils_internal.cc#L113 > > > The event parameter is not TS_EVENT_HTTP_TXN_CLOSE (60012). > When the TXN_CLOSE hook is triggered, the CPPAPI destroys the instance of your custom class that is derived from TransactionPlugin. So you are supposed to put anything you want to run on that hook in the destructor of that class. Because continuations on hooks run in arbitrary order, I think your change could cause a use after free. The problem is that cppapi does not have the handle for my custom class in the TransactionPlugin table, because my plugin never called addPlugin() on it. So, cppapi does/can not destroy my plugin object. Does that answer your question or did I misunderstand your point?
3. > Potential use after free I think, and duplicating an existing capability. Can you look at the updated diff? I reset the Transaction object inside Txn arg, which will prevent potential use-after-free and also cleaned up the the transaction plugin object/continuation in txn-clsoe event.
4. > It seems like these changes would be benign to those using the CPPAPI as originally intended. But I can't see how logic that you can't be sure will execute would be useful. The primary motivation for me is to prevent the assert from firing. However, this will also allow other users that want to do stuff in TXN_CLOSE hook without having to put that stuff into the d'tor (which is actually a little bit ugly).
5. > > In the state dump, in frame #4, it seems to be showing the parameters in the reverse order from what they are declared in the function: > > https://github.com/apache/trafficserver/blob/bf097d4289a3eaa3759a58fb19e48381b89ce32f/src/tscpp/api/utils_internal.cc#L113 > > > > The event parameter is not TS_EVENT_HTTP_TXN_CLOSE (60012). ... > Does that answer your question or did I misunderstand your point? What about the parameter reversal?
6. > > > In the state dump, in frame #4, it seems to be showing the parameters in the reverse order from what they are declared in the function: > > > https://github.com/apache/trafficserver/blob/bf097d4289a3eaa3759a58fb19e48381b89ce32f/src/tscpp/api/utils_internal.cc#L113 > > > > > > The event parameter is not TS_EVENT_HTTP_TXN_CLOSE (60012). > > > ... > > > Does that answer your question or did I misunderstand your point? > > What about the parameter reversal? Yeah, that is interesting. Not sure how to explain that, wonder if it's stack corruption.
7. After this line: https://github.com/apache/trafficserver/blob/3376d438b4a6410187e1ddedd87d2e89279ec196/include/tscpp/api/TransactionPlugin.h#L94 I think you should add: ``` * * \note For automatic destruction, you must either register dynamically allocated instances of * classes derived from this class with the the corresponding transaction (with * Transaction::addPlugin()), or register HOOK_TXN_CLOSE (but not both). ```
8. Can this be squashed to a single commit before it gets merged?
9. > Can this be squashed to a single commit before it gets merged? Yes, I'm going to use the github's option to "Squash and Merge". Just need an approval.
10. Cherry-picked to v9.0.x branch.

github_pulls_reviews:

1. But doesn't this mean you could not be sure that your override of handleTxnClose() would actually get run? Because the continuation that runs handleTransactionEvents() may run first. I don't think we ever merged the proposed continuation priority changes. I think thats what you'd need for this to be useful.
2. No, the continuation for my plugin will still run regardless of `handleTransactionEvents()`. That's how the stack trace above resulted. It's like registering CPPAPI twice with the same hook. `handleTransactionEvents()` will have no affect on TransactionPlugins() that never called addPlugin()
3. What if someone called GlobalPlugin::registerHook(HOOK_TXN_CLOSE) ? You should add an assert to block that.
4. OK right.
5. I think it would be good to add a "only for TransactionPlugins::registerHood()!" comment here.
6. I think you should add 'TSAssert(hook_type != Plugin::HOOK_TXN_CLOSE);' after this line: https://github.com/apache/trafficserver/blob/3376d438b4a6410187e1ddedd87d2e89279ec196/src/tscpp/api/GlobalPlugin.cc#L89
7. Ack. Makes sense.
8. Ack.

jira_issues:

jira_issues_comments: