Wt - Bug #9106

Wt with websockets can leak WebRequests and file descriptors at session exit

09/24/2021 07:57 PM - Bruce Toll

Status:	Review	**Start date:**	09/24/2021
Priority:	Normal	**Due date:**	
Assignee:	Roel Standaert	**% Done:**	0%
Category:		**Estimated time:**	0.00 hour
Target version:	4.10.0		

Description

NOTE: This is an issue that I have been tracking outside of Redmine and believe still exists in Wt 4.5.0. It was originally written-up in two parts, as my understanding of the issue evolved. I've included both parts as "Initial Report" and "Updated Report", since they cover multiple ways to reproduce/test the issue. The most interesting material, though, is probably in the "Updated Report" along with the updated patch file: **0004-Fix-webrequest-leak-with-websockets-due-to-race.patch**.

Initial Report

There appears to be a race in github Wt 4.0.5-15-g3b1778d6 for Wt apps with websockets enabled that can result in the "leak" of WebRequests and associated file descriptors until exit. The issue is timing-dependent and appears to be due to a race between the websocket flush() in the WebSession destructor and the final websocket write.

This issue can occur if a Wt app has output pending when quit() is called, e.g. if an application displays a WDialog, prior to quit(), as part of an application-level session timeout.

The race condition results in a debug log entry of "WtReply::send(): still busy sending... ignoring", although it will not be seen with the logging defaults. One way to "fix" the leak issue is to call the base class "Reply::send()" before returning from WtReply::send() when this error is encountered. However, it does not seem like the cleanest approach and it might have other side-effects.

An alternative solution (proposed below) is to ensure that the WebSession is kept alive until the final websocket write is complete. It is more complex and requires careful review.

Attached are three files:

1. 0001-Modified-hello-example-for-test-of-webrequest-leak.patch. This is a modified version of hello with an idle timeout that can be used to reproduce the issue. It also buffers the LOGGER output, as it can be difficult to reproduce the race condition with normal logging enabled. To use it:

    1. Apply the patch and build a debug version of Wt with DEBUG and WT_DEBUG_JS.
    2. Start the hello application: .../hello.wt -c wt_config_less_debug.xml --docroot . --http-address 0.0.0.0 --http-port 8080 --deploy-path=/test
    3. Start chromium browser (firefox also works, but chromium seems to hit the issue more reliably)
    4. Create a number of sessions, e.g. 40, with: for i in {1..40}; do chromium http://localhost:8080/test; done
    5. On a second session, try: lsof -p (PID of hello.wt) | grep -c localhost: After a few seconds, this should report 42.
    6. Within three minutes, the sessions in chromium should timeout. Try: lsof -p (PID of hello.wt) | grep -c localhost: If it reports more than 2, some WebRequests and file descriptors have leaked. This happens reliably on a 6 core NixOS 18.09 (linux) system.
    7. Kill the hello.wt application (from another session if you are using tee to capture output, otherwise the output will be lost). There should be "*WebRequest: took xxxx ms*" entries after the "*Shutdown: stopping web server*" message.

2. 0002-TEST-ONLY.-Force-webrequest-leak.patch. This patch is similar to the proposed fix, but has the opposite effect, causing the webrequest leak to happen for every WebSession. I found it handy for testing and to provide additional insight on the issue. It can be applied on top of 0001-Modified-hello-example...

3. 0003-Fix-webrequest-leak-with-websockets-due-to-race.patch. The goal of this patch is to extend the lifetime of the websession until the last write completes. For testing, it can be applied on top of 0001-Modified-hello-example.... I did not test it extensively. If used, it would benefit from a careful review -- especially with respect to thread safety.

Updated Report

As a follow-up, I've been looking into a closely related issue where a window unload also leaks a WebRequest and file descriptor. This issue can be demonstrated with an unmodified version of the widgetgallery example (Wt 4.0.5-15-g3b1778d6, boost 1.67) under the following conditions:

1. Limited CPU. I could only replicate this issue with one or two cpus (and the default 10 threads), e.g. using "taskset -c 0". However, by modifying the application to have a delay, e.g. inserting a "sleep(1)", I could also replicate the issue with additional CPUs.

2. There must be significant socket activity. I've found that opening 100 widgetgallery tags in chromium will typically yield several lost WebRequests and associated file descriptors. I wait until the first client starts to ping (visible as a WebRequest in the log) and then close chromium tabs as quickly as possible with ctrl-w. Note: just closing the browser will not work as the Wt-unloads are not sent.

The above method is cumbersome, so I have tried two other testing approaches that I believe are roughly equivalent:

1. Patch Wt.js schedulePing to send a Wt-unload in addition to (or in place of) the ping request. This will affect all Wt apps using the built library....

2. Use mitmproxy to apply the Wt-unload patch to Wt.js dynamically as it is sent to the client browser. NOTE: I'm using version 4.0.4 of mitmproxy and it does not appear to work with websocket compression.

For the second approach, a WT_DEBUG_JS build of Wt is required, and the procedure is:

1. Start widgetgallery on port 8080 with a single CPU
   taskset -c 0 widgetgallery.wt --no-compression --docroot docroot --approot approot --http-address 0.0.0.0 --http-port 8080 --deploy-path=/test

2. On a second session, use mitmproxy to make the widgetgallery available on port 8081, with Wt.js patch applied:
   mitmproxy --setheaders /.*/Origin/http://localhost --replacements '/~bs ws\.send/ws.send\(\"&signal=ping\"\);/{self.emit(self,"Wt-unload");scheduleUpdate();sendUpdate();}'-p 8081 -m reverse:http://localhost:8080

3. Start an instance of the chromium browser

4. On a third session, open 100 tabs in chromium at http://localhost:8081
   for i in {1..100}; do chromium http://localhost:8080/test; done

5. After a minute or two, all the sessions should exit due to the Wt-unload. Check for lost file descriptors:
   lsof -p $(pidof widgetgallery.wt) | grep -c 'localhost'

6. If the above command results in a value over 2, it indicates a WebRequest / file descriptor leak. If you look at the lsof output, you should see some sockets still in ESTABLISHED state. Killing the chromium browser at this point will switch them to CLOSE_WAIT where they will remain.

An updated patch for your review is attached as **0004-Fix-webrequest-leak-with-websockets-due-to-race.patch**. It works like the earlier patch by keeping the websession alive while a websocket write is outstanding. However, the revised patch takes effect as soon as the websocket is created rather than waiting until the session is being killed.

History

#1 - 09/28/2021 11:05 AM - Roel Standaert

- Target version set to 4.7.0

#2 - 10/19/2021 11:01 AM - Korneel Dumon

- Status changed from New to InProgress

- Assignee set to Korneel Dumon

#3 - 02/08/2022 11:55 AM - Korneel Dumon

- Status changed from InProgress to Review

- Assignee deleted (Korneel Dumon)

#4 - 03/10/2022 04:34 PM - Roel Standaert

- Status changed from Review to InProgress

- Assignee set to Korneel Dumon

- Target version changed from 4.7.0 to 4.8.0

#5 - 07/04/2022 10:10 AM - Roel Standaert

- Status changed from InProgress to Review

- Assignee changed from Korneel Dumon to Roel Standaert

#6 - 07/08/2022 08:55 AM - Roel Standaert

- Target version changed from 4.8.0 to 4.9.0

#7 - 10/06/2022 09:52 AM - Roel Standaert

- Target version changed from 4.9.0 to 4.10.0

Files

0001-Modified-hello-example-for-test-of-webrequest-leak.patch	6.38 KB	09/24/2021	Bruce Toll
0002-TEST-ONLY.-Force-webrequest-leak.patch	5.75 KB	09/24/2021	Bruce Toll
0003-Fix-webrequest-leak-with-websockets-due-to-race.patch	5.75 KB	09/24/2021	Bruce Toll
0004-Fix-webrequest-leak-with-websockets-due-to-race.patch	5.76 KB	09/24/2021	Bruce Toll