GDB enhancement for OP-TEE

Background

As of today it's easy to debug a Trusted Application (TA) under these circumstances

- Build OP-TEE with CFG_TA_ASLR=n
- Debugging a single TA
- Secure world UART enabled

This also means that it is very cumbersome to try to debug multiple TA's running at the same time or debugging a single TA when ASLR is enabled.

How to debug a TA in GDB as of today

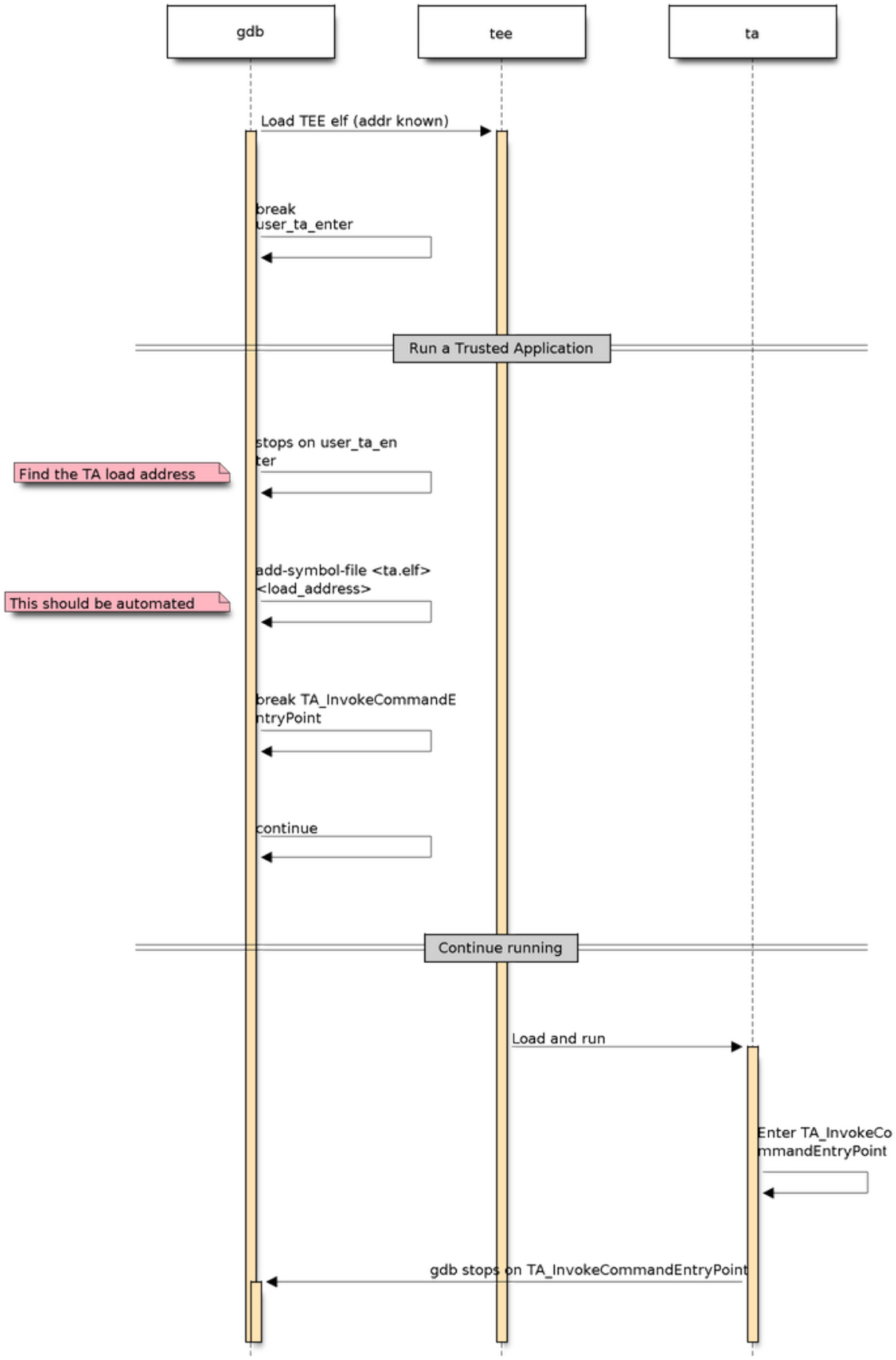
1. Build with **CFG_TA_ASLR=n** (OP-TEE build instructions).
2. Boot up everything, run your TA once, and on secure side log pay attention to the load address (a line like this … example from running **optee_hello_world**)
   D/LD: ldelf:169 ELF (8aaaf200-2450-11e4-abe2-0002a5d5c51b) at 0x113000
3. Restart it all, boot again and then in gdb load the TA ELF:
   add-symbol-file <path/to/ta.elf> <load_address>
   I.e., can be something like
   add-symbol-file
   <project-root>/out-br/build/optee_examples-1.0/hello_world/ta/out/8aaaf200-2450-11e4-abe2-0002a5d5c51b.elf 0x113020
   The **20** there is because the ".text" segment starts at offset 20, so that is added to what we found in the secure side log on step 2.

How do we envision debugging with ASLR enabled (or multiple TA's)?

At the sequence diagram below (scroll down past the empty stuff) we have tried to illustrate our thinking. It consists of these steps:

- Load the TEE ELF first and set a breakpoint in one of the functions (user_ta_enter is a good candidate) that are close to dealing with the load address for TA's.
- Using gdb python script try to automate finding the TA load address.
- When address is found load that TA at correct address, in gdb python that is done with something like this (for details look here):
  gdb.execute("add-symbol-file {}/{} {}".format(OPTEE_PROJ_PATH, ta, TA_LOAD_ADDR))

- Set the breakpoint in function TA_InvokeCommandEntryPoint in the TA we’re about to load.
- Continue running and if everything goes well we hit the breakpoint.


gdb
tee
ta
Load TEE elf (addr known)
break
user_ta_enter
Run a Trusted Application
stops on user_ta_en
ter
Find the TA load address
add-symbol-file <ta.elf>
<load_address>
This should be automated
break TA_InvokeCommandE
ntryPoint
continue
Continue running
Load and run
Enter TA_InvokeCo
mmandEntryPoint
gdb stops on TA_InvokeCommandEntryPoint