- start gdbserver on target

```
root@granite2v8:~# gdbserver localhost:2345 /usr/bin/jbigtest
Process /usr/bin/jbigtest created; pid = 427
gdbserver: Unable to determine the number of hardware watchpoints available.
gdbserver: Unable to determine the number of hardware breakpoints available.
Listening on port 2345
```

- start cgdb on host

```
cgdb -d aarch64-poky-linux-gdb jbigtest
```

- in gdb

```
(gdb) target remote 10.38.52.191:2345
Remote debugging using 10.38.52.191:2345
warning: Unable to find dynamic linker breakpoint function.
GDB will be unable to debug shared library initializers
and track explicitly loaded dynamic code.
0x0000007fb7fd3d00 in ?? ()
(gdb) b main
Breakpoint 1 at 0x401878: file jbig_test.c, line 129.
(gdb) c
Continuing.
warning: Could not load shared library symbols for 7 libraries, e.g. /lib64/libpthread.so.0.
Use the "info sharedlibrary" command to see the complete listing.
Do you need "set solib-search-path" or "set sysroot"?

Breakpoint 1, main (argc=1, argv=0x7ffffffcc8) at jbig_test.c:129
(gdb)
```

让被调试application运行到main()，这样其用到的.so就都已经被载入了(在gdbserver最初始连接到application时，其用到的.so还没有机会没载入，因为dynamic link loader都还没工作呢)

```
00400000-00405000 r-xp 00000000 b3:22 5104                               /usr/bin/jbigtest
00414000-00415000 rw-p 00004000 b3:22 5104                               /usr/bin/jbigtest
00415000-00416000 rw-p 00000000 00:00 0                                  [heap]
7fb7fd3000-7fb7fef000 r-xp 00000000 b3:22 101                            /lib64/ld-2.21.so
7fb7ffc000-7fb7ffd000 r--p 00000000 00:00 0                              [vvar]
7fb7ffd000-7fb7ffe000 r-xp 00000000 00:00 0                              [vdso]
7fb7ffe000-7fb8001000 rw-p 0001b000 b3:22 101                            /lib64/ld-2.21.so
7ffffdf000-8000000000 rw-p 00000000 00:00 0                              [stack]
```

当用户态的第一条指令运行时，从上面的memory mapping可看到只有 `jbigtest` 本身和dynamic linking loader(/lib64/ld-2.21.so)被载入了。其他的.so都要依赖ld-2.21.so来载入，而这时ld-2.21.so还没有机会运行呢！

- login the target

  > ssh 10.38.52.191 -l root

- get the memory map of debugged application

```
root@granite2v8:~# ps | grep 427
  427 root      3156 t    /usr/bin/jbigtest
root@granite2v8:~# cat /proc/427/maps
00400000-00405000 r-xp 00000000 b3:22 5104                               /usr/bin/jbigtest
00414000-00415000 rw-p 00004000 b3:22 5104                               /usr/bin/jbigtest
00415000-00416000 rw-p 00000000 00:00 0                                  [heap]
7fb7d0c000-7fb7e3c000 r-xp 00000000 b3:22 21                             /lib64/libc-2.21.so
7fb7e3c000-7fb7e4b000 ---p 00130000 b3:22 21                             /lib64/libc-2.21.so
7fb7e4b000-7fb7e4f000 r--p 0012f000 b3:22 21                             /lib64/libc-2.21.so
7fb7e4f000-7fb7e51000 rw-p 00133000 b3:22 21                             /lib64/libc-2.21.so
7fb7e51000-7fb7e55000 rw-p 00000000 00:00 0
7fb7e55000-7fb7e5e000 r-xp 00000000 b3:22 3735                           /usr/lib64/libjbig.so.1.0
7fb7e5e000-7fb7e6e000 ---p 00009000 b3:22 3735                           /usr/lib64/libjbig.so.1.0
7fb7e6e000-7fb7e6f000 rw-p 00009000 b3:22 3735                           /usr/lib64/libjbig.so.1.0
7fb7e6f000-7fb7e8f000 rw-p 00000000 00:00 0
7fb7e8f000-7fb7e9e000 r-xp 00000000 b3:22 4878                           /usr/lib64/libdmaalloc.so.1.0
7fb7e9e000-7fb7ead000 ---p 0000f000 b3:22 4878                           /usr/lib64/libdmaalloc.so.1.0
7fb7ead000-7fb7eaf000 rw-p 0000e000 b3:22 4878                           /usr/lib64/libdmaalloc.so.1.0
7fb7eaf000-7fb7eee000 rw-p 00000000 00:00 0
7fb7eee000-7fb7f7f000 r-xp 00000000 b3:22 79                             /lib64/libm-2.21.so
7fb7f7f000-7fb7f8f000 ---p 00091000 b3:22 79                             /lib64/libm-2.21.so
7fb7f8f000-7fb7f90000 r--p 00091000 b3:22 79                             /lib64/libm-2.21.so
7fb7f90000-7fb7f91000 rw-p 00092000 b3:22 79                             /lib64/libm-2.21.so
7fb7f91000-7fb7f97000 r-xp 00000000 b3:22 57                             /lib64/librt-2.21.so
7fb7f97000-7fb7fa6000 ---p 00006000 b3:22 57                             /lib64/librt-2.21.so
7fb7fa6000-7fb7fa7000 r--p 00005000 b3:22 57                             /lib64/librt-2.21.so
7fb7fa7000-7fb7fa8000 rw-p 00006000 b3:22 57                             /lib64/librt-2.21.so
7fb7fa8000-7fb7fbe000 r-xp 00000000 b3:22 83                             /lib64/libpthread-2.21.so
7fb7fbe000-7fb7fcd000 ---p 00016000 b3:22 83                             /lib64/libpthread-2.21.so
7fb7fcd000-7fb7fce000 r--p 00015000 b3:22 83                             /li
```

```
b64/libpthread-2.21.so
7fb7fce000-7fb7fcf000 rw-p 00016000 b3:22 83                           /lib64/libpthread-2.21.so
7fb7fcf000-7fb7fd3000 rw-p 00000000 00:00 0
7fb7fd3000-7fb7fef000 r-xp 00000000 b3:22 101                          /lib64/ld-2.21.so
7fb7ff4000-7fb7ff7000 rw-p 00000000 00:00 0
7fb7ffb000-7fb7ffc000 rw-p 00000000 00:00 0
7fb7ffc000-7fb7ffd000 r--p 00000000 00:00 0                            [vvar]
7fb7ffd000-7fb7ffe000 r-xp 00000000 00:00 0                            [vdso]
7fb7ffe000-7fb7fff000 r--p 0001b000 b3:22 101                          /lib64/ld-2.21.so
7fb7fff000-7fb8001000 rw-p 0001c000 b3:22 101                          /lib64/ld-2.21.so
7ffffdf000-8000000000 rw-p 00000000 00:00 0                            [stack]
```

```
7fb7e55000-7fb7e5e000 r-xp 00000000 b3:22 3735                         /usr/lib64/libjbig.so.1.0 ①
7fb7e5e000-7fb7e6e000 ---p 00009000 b3:22 3735                         /usr/lib64/libjbig.so.1.0 ②
7fb7e6e000-7fb7e6f000 rw-p 00009000 b3:22 3735                         /usr/lib64/libjbig.so.1.0 ③
```

这是libjbig.so.1.0的memory mapping.
①显然是code segment
②可能是起保护作用的隔离带(64K, non-readable, non-writable, non-executable)
③是data segment

The code segment of libjbig.so.1.0 is loaded in `0x7fb7e55000`

- get the .text section from libjbig.so.1.0

```
$ aarch64-poky-linux-objdump -h libjbig.so.1.0 | grep .text
 10 .text         00005054  0000000000002760  0000000000002760  00002760  2**2
```

The .text section offset is `0x2760`

- calculate the loaded address of libjbig.so.1.0

section	Size	VMA	LMA	File Offset
.text	00005054	0000000000002760	0000000000002760	00002760

可以计算出libjbig.so.1.0中实际的.text section所载入的virtual address
code segment virtual address + .text VMA =
0x7fb7e55000 + 0x2760 = `0x7fb7e57760`

- load symbol

```
(gdb) add-symbol-file ~/work/current/ccsgit/driver/jbig-codec/jbig-codec-app/libjbig.so.1.0 0x7fb7e57760
add symbol table from file "/home/walterzh/work/current/ccsgit/driver/jbig-codec/jbig-codec-app/libjbig.so.1.0" at
        .text_addr = 0x7fb7e57760
(y or n) y
Reading symbols from /home/walterzh/work/current/ccsgit/driver/jbig-codec/jbig-codec-app/libjbig.so.1.0...done.
```

- check symbol is OK

> (gdb) disassemble jbig_init

从汇编代码看应该是对的。如果地址错了，看到的是完全无意义的指令。

- set breakpoint in .so file

```
(gdb) break jbig_init
Breakpoint 2 at 0x7fb7e5aabc: file mrvl_jbig.c, line 105.
```

可看到gdb已经找到 `jbig_init()` 的source code了

- everything is OK

```
(gdb) c
Continuing.

Breakpoint 2, jbig_init () at mrvl_jbig.c:105
(gdb) list
100     #ifdef HAVE_UNIT_TEST
101         int unit_test_res;
102     #endif
103
104
105         ASSERT(jbig_block_config_ptr == NULL);                          // has CdmaInit() already been called?
106         jbig_block_config_ptr = jbig_platform_get_config();
107         ASSERT(jbig_block_config_ptr != NULL);
108
109         jbig_init_rtos(jbig_block_config_ptr);
```