Targeted process injection – Linux

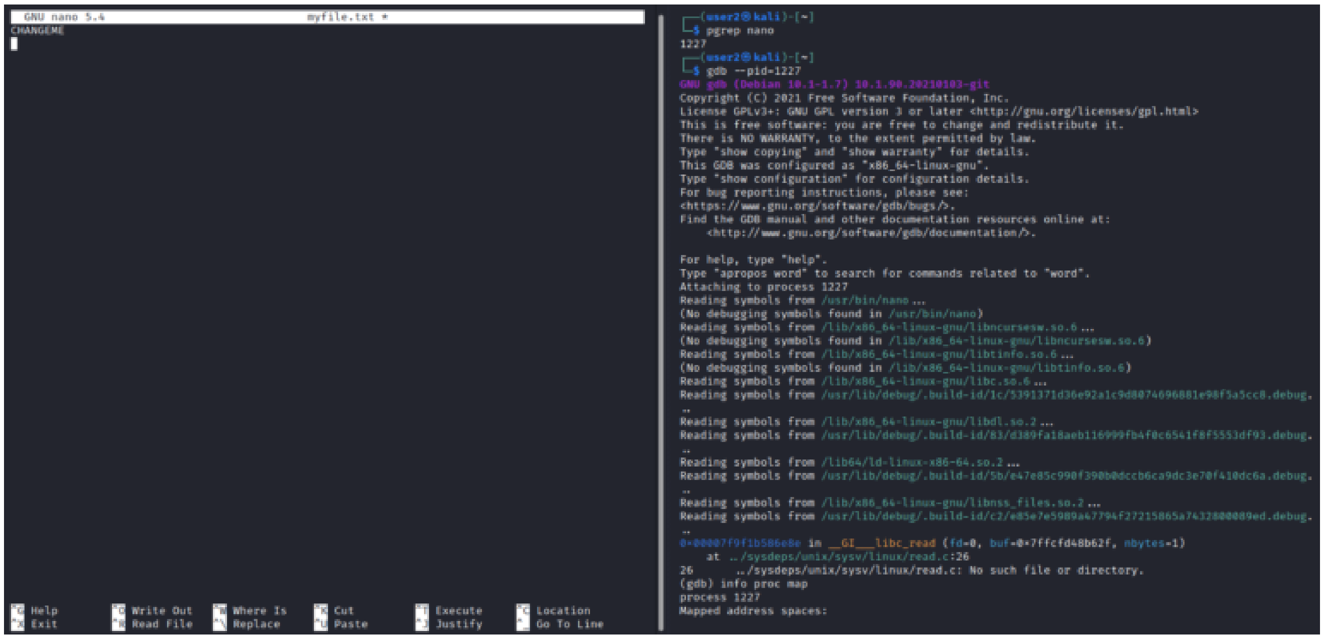
cybersecuritynotepad.com/index.php/2022/02/20/targeted-process-injection-linux

20 February 2022

In this post I will cover briefly how to modify the memory of running process in real-time. The example chosen for this demonstration is simple. The main purpose is to show the approach rather than make it look sophisticated. However, you can take it as far as you wish, depending on your objectives. In future posts, I will cover additional scenarios and also will add more weaponised examples.

One important question that needs answering is why would somebody want to directly inject information/data into the running process. Imagine the following scenario, post compromise you have SSH access to the Linux server. At this stage you can escalate privileges or stay on the system as low privileged user. For privilege escalation traditional methods such as misconfigured permissions, cron jobs, unpatched software and other issues could be used. Those techniques and methods are well known to offensive, defensive and forensic teams. As soon as you follow the usual script, your presence will be detected and the access may be removed. In those types of scenarios, you can be creative and that's when this technique may come in handy.

For the demo, I used the "nano" text editor. The user "user2" who is a standard system user with limited privileges opened a text file "myfile.txt" and started typing text into the file just like a typical user would.

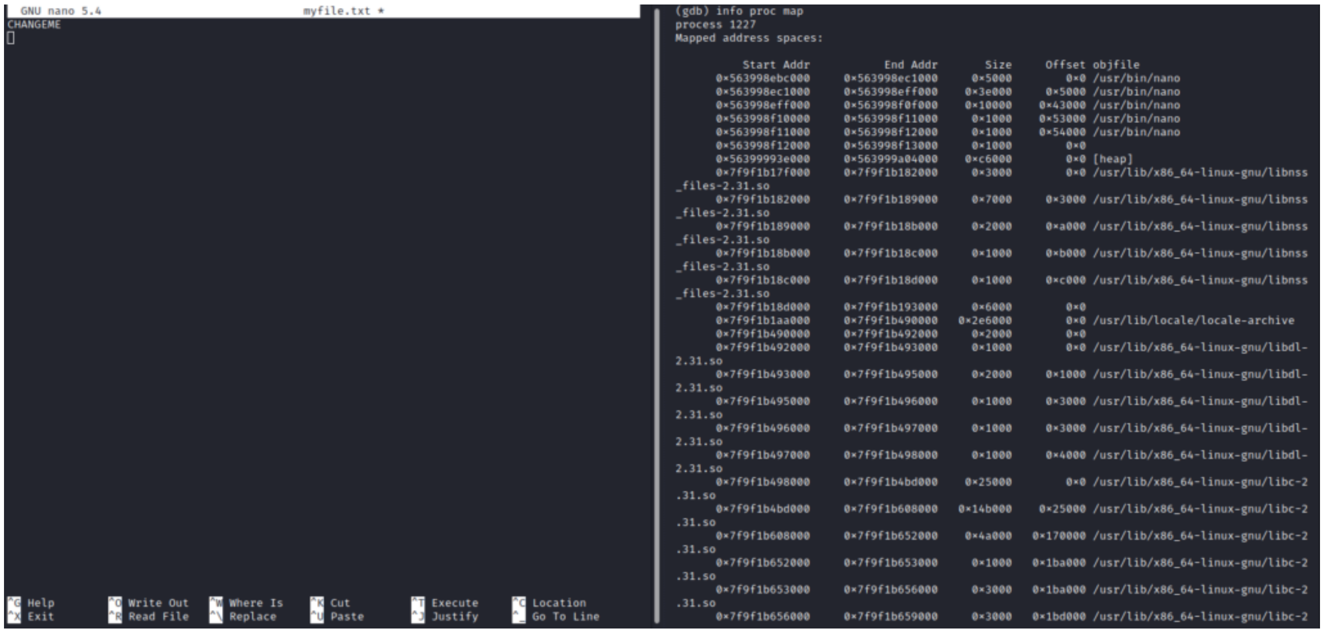


```
  GNU nano 5.4                        myfile.txt *
CHANGEME


^G Help       ^O Write Out  ^W Where Is   ^K Cut        ^T Execute    ^C Location
^X Exit       ^R Read File  ^\ Replace    ^U Paste      ^J Justify    ^_ Go To Line
```

```
┌──(user2㉿kali)-[~]
└─$ pgrep nano
1227
┌──(user2㉿kali)-[~]
└─$ gdb --pid=1227
GNU gdb (Debian 10.1-1.7) 10.1.90.20210103-git
Copyright (C) 2021 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
Type "show copying" and "show warranty" for details.
This GDB was configured as "x86_64-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<https://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
    <http://www.gnu.org/software/gdb/documentation/>.

For help, type "help".
Type "apropos word" to search for commands related to "word".
Attaching to process 1227
Reading symbols from /usr/bin/nano...
(No debugging symbols found in /usr/bin/nano)
Reading symbols from /lib/x86_64-linux-gnu/libncursesw.so.6...
(No debugging symbols found in /lib/x86_64-linux-gnu/libncursesw.so.6)
Reading symbols from /lib/x86_64-linux-gnu/libtinfo.so.6...
(No debugging symbols found in /lib/x86_64-linux-gnu/libtinfo.so.6)
Reading symbols from /lib/x86_64-linux-gnu/libc.so.6...
Reading symbols from /usr/lib/debug/.build-id/1c/5391371d36e92a1c9d8074696881e98f5a5cc8.debug.
..
Reading symbols from /lib/x86_64-linux-gnu/libdl.so.2...
Reading symbols from /usr/lib/debug/.build-id/83/d389fa18aeb116999fb4f0c6541f8f5553df93.debug.
..
Reading symbols from /lib64/ld-linux-x86-64.so.2...
Reading symbols from /usr/lib/debug/.build-id/5b/e47e85c990f390b0dccb6ca9dc3e70f410dc6a.debug.
..
Reading symbols from /lib/x86_64-linux-gnu/libnss_files.so.2...
Reading symbols from /usr/lib/debug/.build-id/c2/e85e7e5989a47794f27215865a743280089ed.debug.
..
0x00007f9f1b586e8e in __GI___libc_read (fd=0, buf=0x7ffcfd48b62f, nbytes=1)
    at ../sysdeps/unix/sysv/linux/read.c:26
26      ../sysdeps/unix/sysv/linux/read.c: No such file or directory.
(gdb) info proc map
process 1227
Mapped address spaces:
```

As you can see the text “CHANGEME” text appears in the left terminal and in the right terminal the process 1227 is attached to the “gdb” debugger.

Next we identify the area where the text is in the process memory. In this case it is on the heap.



```
  GNU nano 5.4                    myfile.txt *
CHANGEME
```

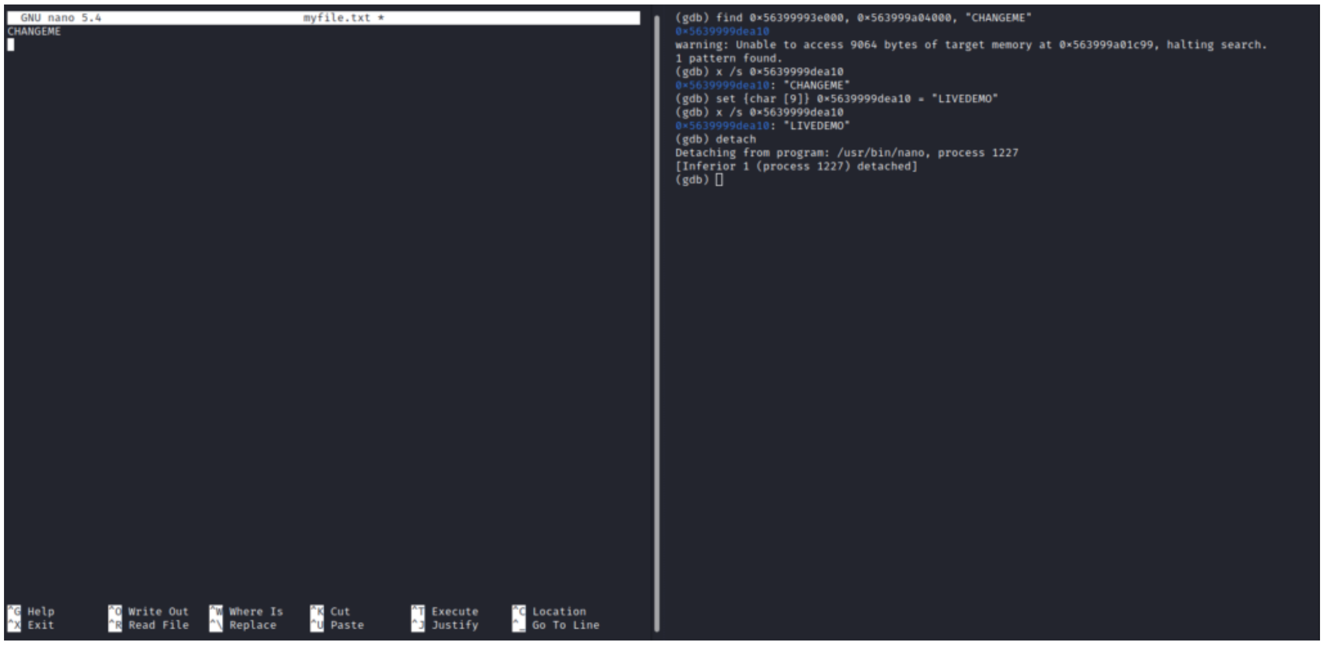
```
(gdb) info proc map
process 1227
Mapped address spaces:

          Start Addr           End Addr       Size     Offset objfile
      0x563998ebc000     0x563998ec1000     0x5000        0x0 /usr/bin/nano
      0x563998ec1000     0x563998eff000    0x3e000     0x5000 /usr/bin/nano
      0x563998eff000     0x563998f0f000    0x10000    0x43000 /usr/bin/nano
      0x563998f10000     0x563998f11000     0x1000    0x53000 /usr/bin/nano
      0x563998f11000     0x563998f12000     0x1000    0x54000 /usr/bin/nano
      0x563998f12000     0x563998f13000     0x1000        0x0
      0x56399993e000     0x563999a04000    0xc6000        0x0 [heap]
      0x7f9f1b17f000     0x7f9f1b182000     0x3000        0x0 /usr/lib/x86_64-linux-gnu/libnss
_files-2.31.so
      0x7f9f1b182000     0x7f9f1b189000     0x7000     0x3000 /usr/lib/x86_64-linux-gnu/libnss
_files-2.31.so
      0x7f9f1b189000     0x7f9f1b18b000     0x2000     0xa000 /usr/lib/x86_64-linux-gnu/libnss
_files-2.31.so
      0x7f9f1b18b000     0x7f9f1b18c000     0x1000     0xb000 /usr/lib/x86_64-linux-gnu/libnss
_files-2.31.so
      0x7f9f1b18c000     0x7f9f1b18d000     0x1000     0xc000 /usr/lib/x86_64-linux-gnu/libnss
_files-2.31.so
      0x7f9f1b18d000     0x7f9f1b193000     0x6000        0x0
      0x7f9f1b1aa000     0x7f9f1b490000   0x2e6000        0x0 /usr/lib/locale/locale-archive
      0x7f9f1b490000     0x7f9f1b492000     0x2000        0x0
      0x7f9f1b492000     0x7f9f1b493000     0x1000        0x0 /usr/lib/x86_64-linux-gnu/libdl-
2.31.so
      0x7f9f1b493000     0x7f9f1b495000     0x2000     0x1000 /usr/lib/x86_64-linux-gnu/libdl-
2.31.so
      0x7f9f1b495000     0x7f9f1b496000     0x1000     0x3000 /usr/lib/x86_64-linux-gnu/libdl-
2.31.so
      0x7f9f1b496000     0x7f9f1b497000     0x1000     0x3000 /usr/lib/x86_64-linux-gnu/libdl-
2.31.so
      0x7f9f1b497000     0x7f9f1b498000     0x1000     0x4000 /usr/lib/x86_64-linux-gnu/libdl-
2.31.so
      0x7f9f1b498000     0x7f9f1b4bd000    0x25000        0x0 /usr/lib/x86_64-linux-gnu/libc-2
.31.so
      0x7f9f1b4bd000     0x7f9f1b608000   0x14b000    0x25000 /usr/lib/x86_64-linux-gnu/libc-2
.31.so
      0x7f9f1b608000     0x7f9f1b652000    0x4a000   0x170000 /usr/lib/x86_64-linux-gnu/libc-2
.31.so
      0x7f9f1b652000     0x7f9f1b653000     0x1000   0x1ba000 /usr/lib/x86_64-linux-gnu/libc-2
.31.so
      0x7f9f1b653000     0x7f9f1b656000     0x3000   0x1ba000 /usr/lib/x86_64-linux-gnu/libc-2
.31.so
      0x7f9f1b656000     0x7f9f1b659000     0x3000   0x1bd000 /usr/lib/x86_64-linux-gnu/libc-2
```

We are only interested in the heap because this is where the text is that the user “user2” wrote within “nano” text editor.

```
heap start address 0x56399993e000
heap end address 0x563999a04000
```

When searching memory it is never easy to find specific information without some trial and error approach. In this case, after spending a while examining the memory I knew immediately where I should look for the desired information. I simplified the process for you so that you can see the results rather than trying to find the needle in the haystack. As you can see now, I found the memory address where the string “CHANGEME” sits and replaced it with the string “LIVEDEMO”. After completing that task I detached the process 1227 from the debugger.

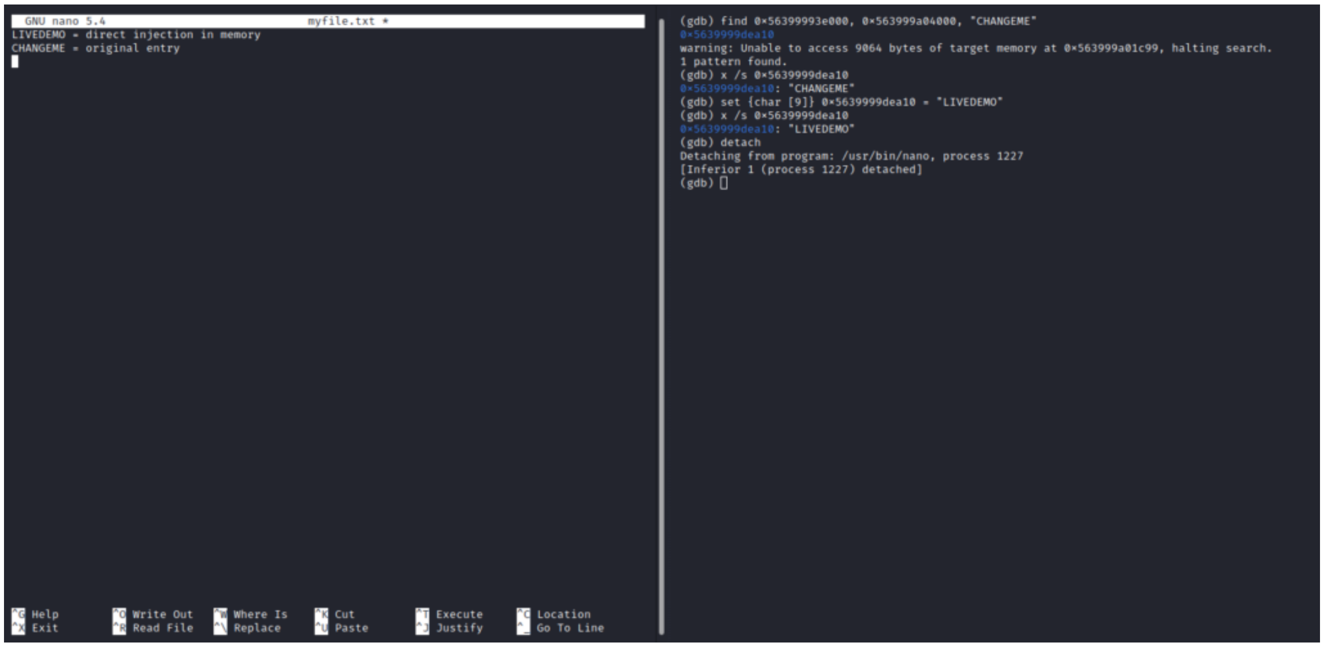
```
0x5639999dea10 = CHANGEME (string)
```



```
  GNU nano 5.4                          myfile.txt *
CHANGEME
█

^G Help        ^O Write Out   ^W Where Is    ^K Cut         ^T Execute     ^C Location
^X Exit        ^R Read File   ^\ Replace     ^U Paste       ^J Justify     ^_ Go To Line
```

```
(gdb) find 0x56399993e000, 0x563999a04000, "CHANGEME"
0x5639999dea10
warning: Unable to access 9064 bytes of target memory at 0x563999a01c99, halting search.
1 pattern found.
(gdb) x /s 0x5639999dea10
0x5639999dea10: "CHANGEME"
(gdb) set {char [9]} 0x5639999dea10 = "LIVEDEMO"
(gdb) x /s 0x5639999dea10
0x5639999dea10: "LIVEDEMO"
(gdb) detach
Detaching from program: /usr/bin/nano, process 1227
[Inferior 1 (process 1227) detached]
(gdb) 
```

Now you can see above that in the left terminal, the "CHANGEME" string is still there as before. It looks to the user "user2" as if nothing has changed because the text has not been refreshed. However, in memory the change has been made and, if the user saved the file or even copied the first line "CHANGEME" into the clipboard they would get "LIVEDEMO" string when pasting from the clipboard or saving the file "myfile.txt". I have added short descriptions to the left terminal so that you can see what happened.



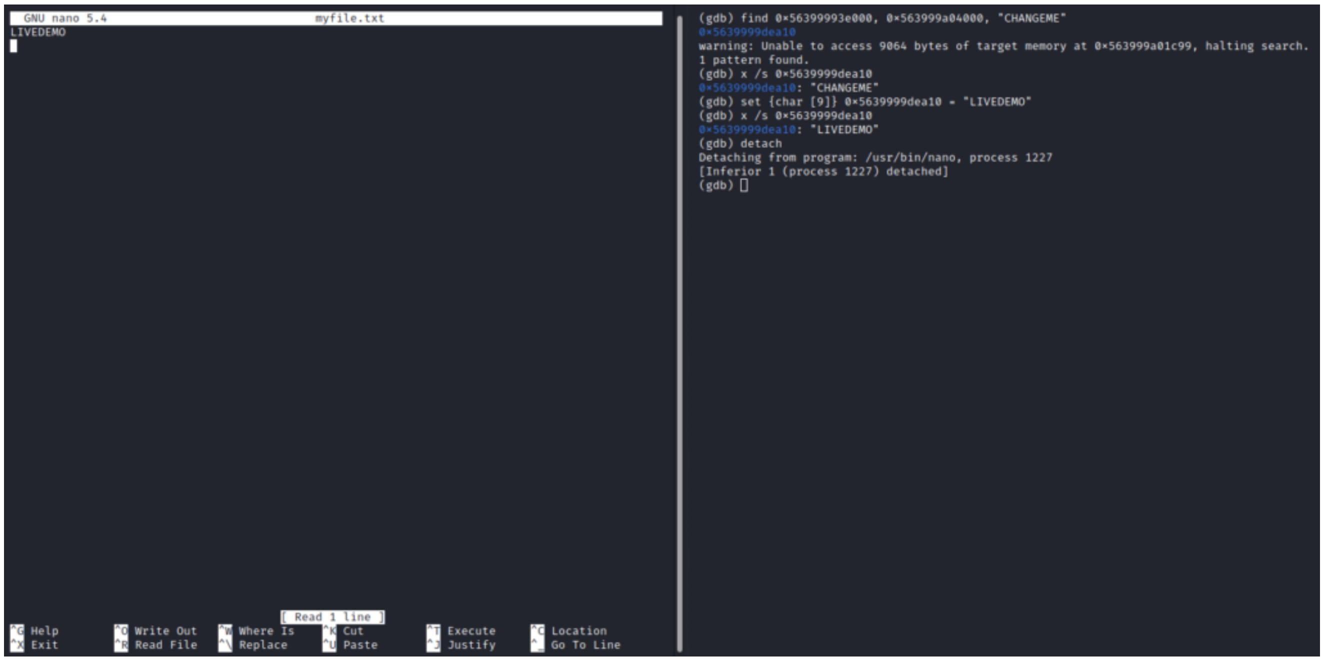
```
  GNU nano 5.4                          myfile.txt *
LIVEDEMO = direct injection in memory
CHANGEME = original entry
█

^G Help        ^O Write Out   ^W Where Is    ^K Cut         ^T Execute     ^C Location
^X Exit        ^R Read File   ^\ Replace     ^U Paste       ^J Justify     ^_ Go To Line
```

```
(gdb) find 0x56399993e000, 0x563999a04000, "CHANGEME"
0x5639999dea10
warning: Unable to access 9064 bytes of target memory at 0x563999a01c99, halting search.
1 pattern found.
(gdb) x /s 0x5639999dea10
0x5639999dea10: "CHANGEME"
(gdb) set {char [9]} 0x5639999dea10 = "LIVEDEMO"
(gdb) x /s 0x5639999dea10
0x5639999dea10: "LIVEDEMO"
(gdb) detach
Detaching from program: /usr/bin/nano, process 1227
[Inferior 1 (process 1227) detached]
(gdb) 
```

The final screenshot shows that even if on user's screen nothing changes, the memory address shown below has been modified:

```
0x5639999dea10
```

The above address no longer stored the string “CHANGEME” but “LIVEDEMO”. After saving the text file and reopening it the user “user2” will see the modified string “LIVEDEMO” confirming that the injection into the live memory within the heap space has been successful.



```
  GNU nano 5.4                     myfile.txt
LIVEDEMO

                       [ Read 1 line ]
^G Help       ^O Write Out  ^W Where Is   ^K Cut        ^T Execute    ^C Location
^X Exit       ^R Read File  ^\ Replace    ^U Paste      ^J Justify    ^_ Go To Line
```

```
(gdb) find 0×56399993e000, 0×563999a04000, "CHANGEME"
0×5639999dea10
warning: Unable to access 9064 bytes of target memory at 0×563999a01c99, halting search.
1 pattern found.
(gdb) x /s 0×5639999dea10
0×5639999dea10: "CHANGEME"
(gdb) set {char [9]} 0×5639999dea10 = "LIVEDEMO"
(gdb) x /s 0×5639999dea10
0×5639999dea10: "LIVEDEMO"
(gdb) detach
Detaching from program: /usr/bin/nano, process 1227
[Inferior 1 (process 1227) detached]
(gdb) 
```

As you can see, it is not that difficult to get it done. In fact, it is easier than you may think. Imagine what would happen, if you changed just one byte in the right place and at the right time to disable security controls. Going one step further, you can execute the code without leaving forensic artifacts, as long as you know what you are doing.

Published
Categorised as Research Tagged debugger, gdb, injection, Linux, memory, process