1 Subverting the Fingerprinting Implementation

One direct way of attacking our fingerprinting implementation is to subvert the calls to `clock_gettime`. In this sense, an attacker with access to the program executable can identify the `clock_gettime` call-sites by inspecting the assembly code:

```
$ objdump -d main

# wrapper around the C library clock_gettime
00000000000036d0 <clock_gettime@plt>:
    36d0:  f3 0f 1e fa           endbr64
    36d4:  f2 ff 25 65 e7 00 00  bnd jmp *0xe765(%rip) # 11e40 <clock_gettime@GLIBC_2 .17>
    36db:  0f 1f 44 00 00        nopl   0x0(%rax,%rax,1)

# non-trivial wrapper around the previous wrapper
0000000000004446 <_Z14_clock_gettimeiP8timespec>:
    4446:  f3 0f 1e fa           endbr64
    444a:  55                    push   %rbp
    444b:  48 89 e5              mov    %rsp,%rbp
    444e:  48 83 ec 20           sub    $0x20,%rsp
    4452:  89 7d ec              mov    %edi,-0x14(%rbp)
    4455:  48 89 75 e0           mov    %rsi,-0x20(%rbp)
    4459:  0f ae f0              mfence
    445c:  90                    nop
    445d:  48 8b 55 e0           mov    -0x20(%rbp),%rdx
    4461:  8b 45 ec              mov    -0x14(%rbp),%eax
    4464:  48 89 d6              mov    %rdx,%rsi
    4467:  89 c7                 mov    %eax,%edi
    4469:  e8 62 f2 ff ff        call   36d0 <clock_gettime@plt>
    446e:  89 45 fc              mov    %eax,-0x4(%rbp)
    4471:  0f ae f0              mfence
    4474:  90                    nop
    4475:  8b 45 fc              mov    -0x4(%rbp),%eax
    4478:  c9                    leave
    4479:  c3                    ret

# fingerprint timing routine
    4c3c:  e8 05 f8 ff ff        call   4446 <_Z14_clock_gettimeiP8timespec>
    ...
    4c6b:  e8 d6 f7 ff ff        call   4446 <_Z14_clock_gettimeiP8timespec>
```

Here, it is worth noting that line `4469` calls the C library `clock_gettime` wrapper, whose result (supposedly stored in the `eax` register), is moved onto the stack in line `446e`. If, instead of writing the contents of `eax` onto the stack, the attacker can write the value 0, they will generate a reproducible null fingerprint. One way to achieve this is to replace the instruction

```
    446e:  89 45 fc              mov    %eax,-0x4(%rbp)
```

with

```
    446e:  c7 45 fc 00 00 00 00  movl   $0x0,-0x4(%rbp)
```

However, the new instruction has a different width, which could modify the address of further instructions. To avoid this, the new instruction (which poisons the stack with null values) can instead replace the call into the C library's `clock_gettime`, which has the same width. Then, to avoid the null

value being overwritten by a garbage value stored in `eax`,

```
    446e:  89 45 fc             mov    %eax,-0x4(%rbp)
```

can be replaced with

```
    4475:  8b 45 fc             mov    -0x4(%rbp),%eax
```

This effectively behaves as a no-op, since it is repeated a few instructions later. The result of making these two changes should be a reproducible null-fingerprint generator. To install these changes, the attacker can find the instructions of interest in the binary file:

```
$ hexedit main

00004448  1E FA 55 48 89 E5 48 83 EC 20 89 7D EC 48 89 75 E0 0F AE F0 ..UH..H.. .}.H.u....
0000445C  90 48 8B 55 E0 8B 45 EC 48 89 D6 89 C7 E8 62 F2 FF FF 89 45 .H.U..E.H.....b....E
00004470  FC 0F AE F0 90 8B 45 FC C9 C3 F3 0F 1E FA 55 48 89 E5 53 48 ......E.......UH..SH
00004484  83 EC 48 48 89 7D B8 48 89 75 B0 64 48 8B 04 25 28 00 00 00 ..HH.}.H.u.dH..%(...
...
000036B0  F3 0F 1E FA F2 FF 25 75 E7 00 00 0F 1F 44 00 00 F3 0F 1E FA ......%u.....D......
000036C4  F2 FF 25 6D E7 00 00 0F 1F 44 00 00 F3 0F 1E FA F2 FF 25 65 ..%m.....D........%e
000036D8  E7 00 00 0F  1F 44 00 00 F3 0F 1E FA F2 FF 25 5D E7 00 00 0F .....D........%]....
000036EC  1F 44 00 00 F3 0F 1E FA F2 FF 25 55 E7 00 00 0F 1F 44 00 00 .D........%U.....D..
```

and replace them with the new instructions

```
00004448  1E FA 55 48 89 E5 48 83 EC 20 89 7D EC 48 89 75 E0 0F AE F0 ..UH..H.. .}.H.u....
0000445C  90 48 8B 55 E0 8B 45 EC 48 89 D6 89 C7 E8 62 F2 FF FF 8B 45 .H.U..E.H.....b....E
00004470  FC 0F AE F0 90 8B 45 FC C9 C3 F3 0F 1E FA 55 48 89 E5 53 48 ......E.......UH..SH
00004484  83 EC 48 48 89 7D B8 48 89 75 B0 64 48 8B 04 25 28 00 00 00 ..HH.}.H.u.dH..%(...
...
000036B0  F3 0F 1E FA F2 FF 25 75 E7 00 00 0F 1F 44 00 00 F3 0F 1E FA ......%u.....D......
000036C4  F2 FF 25 6D E7 00 00 0F 1F 44 00 00 F3 0F 1E FA C7 45 FC 00 ..%m.....D.......E..
000036D8  00 00 00 0F  1F 44 00 00 F3 0F 1E FA F2 FF 25 5D E7 00 00 0F .....D........%]....
000036EC  1F 44 00 00 F3 0F 1E FA F2 FF 25 55 E7 00 00 0F 1F 44 00 00 .D........%U.....D..
```

The resulting binary generates a null-fingerprint with consistent 2000/2000 match scores.

```
$ ./main fingerprint
$ ./main fingerprint -cmp
2000 (+42 -66)/2000
fingerprint match
```