

Cursory Assessment of the Oculus Quest 2

Michael E. Mahan
Louisiana Tech University
mem061@latech.edu

Abstract

To examine and classify possible attack vectors in the space of the Oculus Quest 2, a wide-scale methodology is applied. This methodology uses aspects of the device which include cursory information gleaned from publicly available documents, as well as more practical methods like hardware part analysis. The presented findings form a wide surface of potential vulnerability vectors, as well as generally outline the technical aspects of the Quest 2.

1. Introduction

In 2021 the global virtual reality market in was valued at \$30.7 billion, the expected increase to \$296.9 billion by 2024 [1]. There have been 14.02 million headsets of various types sold up until 2020, including standalone and tethered varieties [2]. There are a multitude of fields attempting to use VR to augment and/or add to existing infrastructure: entertainment, aerospace and defense, gaming, healthcare, education, and even manufacturing.

In many of these fields, VR is being used in training, allowing students and the general workforce to engage in training simulations with higher fidelity and pedagogical effectiveness than previously available [3].

In other fields, it's being used as an overlay, an enhancement on top of existing job structure. MR (mixed reality) and AR (augmented reality) allow one to engage with this reality and the simulated one at the same time, a feat of which has been depicted in Hollywood for decades, now less fiction, more familiar [4].

VR is evolving the state of play: the methods by which people can engage with both art and multiplayer oriented games. It is now possible to visit a simulated fantasy world and connect with peers to battle for virtual glory, from the comfort of home, in

an incredibly engrossing and immersive simulated VR environment.

The social perspective is also notable. One can meditate on a mountain peak in VR: software for guided meditations in various environments has been made, and in some social VR social spaces, people hold actual globally interconnected sessions. There are communal movie nights in VR social space programs like VRChat and NeosVR.

There's a project to develop the first real VR Browser, Vircadia, which takes the idea of what Tim Berners Lee did so many years ago with the World Wide Web, and adapts it to the VR interface [5].

And finally, in a case of self-experimentation, a Youtuber who goes by the moniker of Disrupt, spent a week in VR. With the headset he used, he was able to pipe a low-quality stream of his own environment back into the headset screens, when he wasn't engaged with some sort of simulation experience [6]. The experience may be an interesting for psychological analysis, the reported effects of long-term immersion, and subsequent exiting of a VR space.

Because of this level of burgeoning popularity, and breadth of use cases, it becomes necessary to begin asking questions which are relevant to security, as well as digital forensics. Any new technology offers to increase the size and scope of both fields. Security, because any device, given clever exploitation, might act as a vector for cybercrime. Forensics, because not only are headsets and VR equipment useful for interacting with digital reality more effectively, but they are also pathways through which digital reality might be fed through into this one, providing forensically useful evidence.

On the security side, there is a notable lack of user authentication with both standalone and tethered headsets. This lack makes tying usage data to specific users more difficult. There are a few headsets which implement the option for a pattern lock, but most do not require a lock to operate.

The processing of the environment around a user may also be a desirable vector of attack. The ability to access data denoting a user's actions, placement,

and general environment may not always be as inaccessible or limited. This can leave users open to spying/voyeuristic attacks.

Bad actors might also have an incentive to compromise headsets to track user activity. There already exist many methods of tracking and stalking with other computing devices, for many standalone headsets this may also be possible.

For a forensics perspective, the interest is in the data that might be sitting on a particular headset. There are many chips and controllers on each, but standalone headsets are particularly interesting, due to their nature of holding not only output hardware, but also permanent storage.

Headsets engage in an array of sensor data processing, be it motion-sensing, spacial detection via cameras on the headset, or microphones. It would be forensically useful to be able to tell if a user had specific patterns of movement within a headset, and if that could be tied to usage.

The interest of this paper is to outline compositional aspects of the Oculus Quest 2, in order to lay groundwork for later vulnerability assessments, vulnerabilities being vectors by which a cyberattack or forensic extraction might occur.

2. Related work

Yarramreddy and co. spent time looking into network based artifacts generated by various social VR applications [7]. They found that the headsets used did in fact store local debugging logs which were useful in cross analysis with recorded network traffic. They were able to determine system information such as host OS, GPU, and even the exact headset model. They were also able to ascertain time-stamped events from the user.

Being able also to tie inter-user and user-environment interactions, was a forensically relevant discovery. Using all this information collectively, it becomes possible to reconstruct a series of relatively detailed events, which could be used as evidence.

Following directly after, the Casey group spent time ascertaining if the HTC Vive headset volatile memory was of forensic relevance, and feasible capture [8]. They were primarily involved with analyzing the data available from the OpenVR framework, which is a generalized compatibility layer created by Valve and distributed through Steam.

Integrating with an existing volatile memory examination software, Volatility Framework, they created a plugin dubbed Vivedump. Through this, it is possible to reconstruct a virtual environment from stored data. The virtual environment is a mapping of

the headset user, not in virtual space, but in the room which they physically reside. The data points for the headset, controllers, and trackers, as well as the dimensions of the physical space were made apparent in this visualization. This is a large amount of fine detail. Effectively an explorable 3D render of the user and their physical space.

With Odeleye and Co.'s research, rendering a device effectively useless through cyberattacks was the goal [9].

It is well noted in the video game space that even normal first-person visual displays on a standard monitor can induce motion sickness in a user. This is likewise true in a VR headset, but in this context, it is termed VR sickness.

Odeleye's team considered that, one primary aspect of this is in fluctuations of framerate. They then determined two attack vectors by which framerate could be manipulated. The first is directly from the perspective of the GPU, which if overwhelmed, will crash the headset, but not prior to dropping frames, and experiencing screen tears. Both symptoms can be visually disorienting for a user, possibly inducing VR sickness prior to system crash.

The team also determined that they could perform a network-based attack, by flooding a headset with ICMP pings. This did work to overwhelm system processing speed, and as consequence, framerate issues did occur. Like before, this situation might induce VR sickness in a user.

A hobbyist developer, Ruotsalainen, created a plugin for a popular streaming software, OBS [10]. This plugin allows for the streaming of live headset footage from an application built with OpenVR API, using a feature of from the library called a "mirror texture" [11]. This is a shared rendering from a Direct3D device instance, which can then be called on by disparate system programs, such as the aforementioned plugin [12]. It is relatively simple, but effective and highlights how easily accessible render data is in OpenVR applications. Render data may be of interest, for cyberstalking or for forensic records.

3. Methodology

3.1. Framing the stack

To determine methodology, I examined my own understanding of the up and down stack of what a conglomeration of hardware and software like the Quest 2 is. I also used an understanding of common vectors that I have seen used over the years of following many reverse engineering projects. The

goal of this methodology being: with the widest breadth, identify the most feasible points for a vulnerability.

At the highest level, we can consider reverse engineering to be an exercise in many forms of both logic and knowledge acquisition. A Quest 2 is an amalgam of many varieties of information, made to work in concert. Just like any other concert, the result will have imperfections, however small.

There is no random incidence in the act of creating. Every aspect follows as an effect of intention and assumption. Intention can be traced: stated outright in myriad forms by documentation, code, and other types of written information. Assumption comes from the inescapable human tendency towards convention. There is also, always, negative space. That which exists will only ever be carved from the limited space of the possible, by the intention and assumption of creators. This is complementary information, the overlooked. Here lay vulnerabilities and all manner of bugs.

When we look at the act of designing and creating a product like the Oculus Quest 2, we can form a list of knowledge areas that may contain relevant information, and then proceed to chip away at forming a grander architectural blueprint. Like a nonogram, the shape of a device's design can be filled in one point at a time, in an interlocking and consequentially oriented manner.

3.2. Pre-stack

Before approaching the main stack, we can examine cursory aspects which give us items to look for later: legal documents, press releases, and other generally public information.

Legal documents, such as a Terms of Service or Privacy Policy, may indicate key data outputs and restrictions on a system, without one needing to do much beyond reading and noting things down. This is important, because if we know what kinds of data outflow to be on the lookout for, we can more easily determine the nature of functions and data when probing the live device.

It is not uncommon that data is grouped in consistent locations, so in finding data we expect, we may also trace its path to find other data, which for our purposes may be of greater value. It is also not uncommon for third parties to be involved in the data collection process, many Privacy Policies may even list specific vendors, who may have specific network facing APIs which they use.

Small bits of information are often present in the news. Intentionally, as press releases or conference talks. Patents and copyrights are also novel sources.

Documentation from regulatory agencies like the FCC (Federal Communications Commission) or USPTO (United States Patent and Trademark Office) may also provide photos of internal workings, and wireless compliances. Papers may have been published, and there may even be third party discussions, informed speculation, or leaks of info on forums.

Finally, we can search sites like LinkedIn and use web searches to determine team members for a device, and from there, the kinds of technologies those persons are familiar with, which ones they recently added to their resumes, and therefore may have injected into the product. We can also look at public information from the same people, see what commits they've made to public repositories, what questions they might have asked on technical forums like StackExchange or certain subreddits. These pieces may relate to their influence on the project that we're interested in.

3.3. Hardware

At the bottom of the stack, we have hardware, and hardware specification. These two things don't always necessarily line up one to one, but the hardware is a realized version of the specification. This is useful, because while hardware itself must be probed and prodded to elicit information, finding direct specification documents shortcuts to pure knowledge. There are a few approaches for gleaning info from a device.

When exposed, some internal components will have a signifier or label. These labels can then be used to search for specification documents, 'spec sheets' in common parlance. Spec sheets for chips will indicate intimate details of a component: inputs, outputs, and internal machinations. If a component has a variety of sub-models, the spec sheet will indicate how they vary, and very often how to discern between variations. This can reveal unknown functionality. Much like any other program, the arrangement of modular functionality on a circuit board tells of a logical flow. Logical flow which can be exposed, and possibly modified or circumnavigated.

It is additionally much harder to patch out physical vulnerabilities, a notable example being the original models of the Nintendo Switch, whose boot sequence has an opening which researchers were able to use as a vulnerability, exploiting it for custom firmware insertion [13]. The later models of the Switch do not have this physical issue, but the older models always will. Because of this patching difficulty, physical vulnerabilities are very valuable

for long-term system compromises, despite the efforts and resources involved to find them.

There are also debug interfaces such as JTAG or SWD which may be in use, as well as test pads on the physical board. It is not uncommon for a manufacturer to neglect deactivating these components. Debug interfaces can be used to simulate and record input signals, as well as do initial firmware dumping.

3.4. Firmware

As mentioned, there exist methods for firmware dumping, which is the process of extracting the firmware data from a chip on a board. Once a firmware dump is acquired, tools like QEMU and binwalk can be used to emulate the original system or traverse the filesystem and assembly code. Firmware size and nature can be determined from chip analysis, most easily from spec sheets [14].

The purpose of obtaining firmware is that, in analysis, we can observe low level interactions, such as partitioning schemes and boot load order. Partitioning schemes are useful for data extraction and analysis, and boot order can later be analyzed for possible entry points at which a vulnerability occurs. Encryption keys or other types of hardcoded authentication may also be found. These types of authentications might be used to push modified firmware later, or to request fresh firmware data from the remote servers which serve Oculus devices updates. We could also ostensibly modify the firmware and re-flash onto the device this modified code.

Use cases for modified firmware include circumventing hardware locks, or restrictions on functionality, as well as turning off unwanted functions, such as those that might be sending out telemetry data.

3.5. Communications

Communications protocols have their own stack, the OSI model. Communications protocols are very often open specification. This means intimate knowledge of how they should be implemented can be compared to data capture (much like how hardware specification and hardware may be compared), and from there we can compare how they are implemented, possibly identifying misconfigurations. There are also known vulnerabilities in spec for older protocols and known circumvention methods.

3.6. Operating system

Modernly, few highly complex devices, if any, run on entirely in-house developed code. This means, that even at an operating system level, we can look for common, known vulnerabilities. The most popular OS in the world is Android, which uses the Linux kernel at its core. Though in the instance of the Quest 2 the Android version is modified in-house, it should still have significant overlap with both Android 10 and more generally Linux kernel-related vulnerabilities.

3.7. Applications

Applications can be native to a default install, or something added to the system by the user. In either case, application code can be used to garner user data, gain elevated system access, or generally cause the user trouble with the system.

4. Data/Analysis

4.1. Pre-stack

There are multiple and different sets of legal notices, depending on which type of account a user creates, due to the acquisition of Oculus by Meta in 2014. Users of accounts strictly tied to Oculus are provided the Oculus Privacy Policy and Oculus Terms of Service. Users of Facebook accounts are provided both Facebook and ‘Supplemental’ Oculus variants of the Terms of Service and Data Policies.

The Supplemental Oculus Data Policy was particularly notable, it mentions that third party apps have access to user data items, including friend lists [15]. There will likely then be a section of the SDK (Software Development Kit) for developers to use to request this data directly.

The policy also mentions that hand tracking, and spacial data is sent out to the company’s own servers. User voice recordings are saved and shared with ‘vendors and service providers’ to improve speech recognition systems. All this data must be exported across the internet. There is then a later opportunity to attempt correlation with captured packet data.

On a blog hosted by Qualcomm, a semiconductor manufacturer, is a post regarding the usage of their own product, the Qualcomm Snapdragon XR2, within the Oculus Quest 2 [16]. This gives the more general idea of what to expect in terms of the CPU/GPU on the Quest 2 and was one of the primary sources of information on the component in section 4.2.

This discovery led to an attempt to source more documentation on the XR2, of which only a cursory overlook from a product overview page was available [17]. From this page it can be discerned that the voice recognition service provided by the Quest is likely operating on the Always On voice activation model, constantly parsing environmental noise from user commands.

The public repository for the original Quest kernel is available on GitHub, underneath Meta's 'facebookincubator' organization. The Quest 2's custom kernel is also present, as one of the branches in this same repository.

By comparing git commit messages with the version numbering available in the Quest's settings, it was confirmed that the current version numbering in the headset settings did indeed match the number sequence in the latest commit in the repo. From here it's assumable that the custom Android build present in this public repo is, at least, the general OS level code in the release builds, in its modified form. This would make sense as a necessary step for compliance with the licensing involved with the Linux kernel and Android OS.

Through the GitHub repo, I was able to find four connected user accounts and a pair of bot accounts authoring and making commits.

The bots are an interesting method of obfuscation. The first bot seems to be centralizing internally authored code, the latter is committing these mass patches to the repository. The name of the authoring bot is interesting: 'SandcastleBuildBot'. Possibilities for this naming scheme led to the conclusions of either: it is a simple name based on an underlying technology it's using, or it is a code name that would only be understood internally.

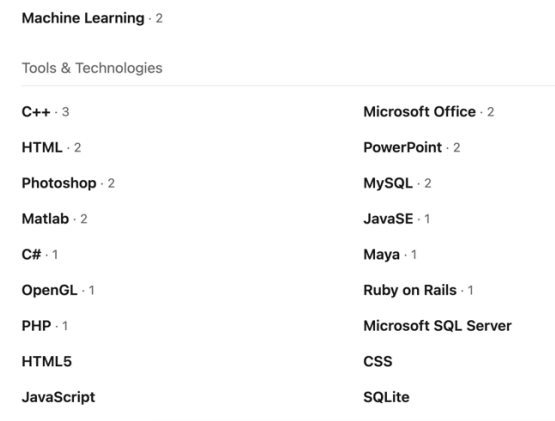
Using web searches for programming tool usage of 'sandcastle' leads immediately to an older MSDN style documentation generator, however, that does not seem to be the use here. Searching the term specifically on GitHub, there are several repositories, but one that may be a lead is sandcastle-buildroot. It is a variation on Buildroot. Buildroot is a cross-platform compiling tool for embedded Linux implementations [18]. Since the Quest 2 is using an embedded/ARM implementation of Linux, it would not be entirely out of scope for the development team to be using an implementation of Buildroot for CI (continuous integration) reasons.

It is perhaps worth looking for any artifacts that Buildroot generates, and seeing if they're in the '.gitignore' file of the repo, or actually present. Of note, in other Meta repositories, there were authoring bots with nonsensical names, as well as names that related to other types of tools, such as codemod

(which is a large scale codebase refactoring tool) [19]. This lends credence to the theory that the sandcastle-buildbot may indeed be named after sandcastle-buildroot, but again does not confirm it.

Of the four user accounts, only two led to accounts elsewhere on the internet. The other two accounts were dead ends: their usernames did not resolve to any other sites on any indexed web searches, nor did their account pages lead to any other useful information. The two accounts that did lead elsewhere were linkable to LinkedIn users who also happened to work at Meta, on VR projects.

Through this account Figure 1 was acquired, a list of skills presented by this user account.



Machine Learning · 2	
Tools & Technologies	
C++ · 3	Microsoft Office · 2
HTML · 2	PowerPoint · 2
Photoshop · 2	MySQL · 2
Matlab · 2	JavaSE · 1
C# · 1	Maya · 1
OpenGL · 1	Ruby on Rails · 1
PHP · 1	Microsoft SQL Server
HTML5	CSS
JavaScript	SQLite

Figure 1. Sampling of first user's skillset

The other of the two GitHub leads also had a LinkedIn page, and the most recent job listing, as seen in Figure 2, does confirm that they are employed towards working on the Quest 2, as well as various other pieces of the VR development at Meta. They have extensive Android development experience, including being on the development team for Cyanogen Mod (which was a highly popular Android fork).

Software Engineer, Facebook Reality Labs
San Francisco Bay Area · 264 connections

Figure 2. Second user's job title and location

A skim of web search results led to finding a particular account of interest, a user whose Meta work entry speaks of having been a lead creator behind the Quest 2 (see Figure 3). This user's page lists several major technologies which are likely used in the technology stack of the Quest. They seem to



Software Engineer, AR/VR (Oculus)

Meta

Apr 2020 - Present · 1 year 8 months

Zurich, Switzerland

If you have used a Quest 2, you have already interacted with my work. :)

Currently, working across multiple teams — research, SLAM, manufacturing, product design, sensors, OS and Firmware — to accurately model, parameterize and deploy highly accurate sensor models for “inside out tracking” to millions of devices.

Also bringing up a framework and devising metrics to automatically validate how close these sensor models (calibration) are to reality.

Figure 3. Third user's current listed position

touch what is most of the stack is made of and are seemingly some form of architectural lead.

The FCC hosts a number of publicly facing documents through their web services, and a site called FCCID.io provides a method for reviewing these documents [20]. I searched for submissions from Oculus (which was acquired by Facebook in 2014) and Facebook Technologies. I was able to locate images of the Quest line components. Some of the images provided label capture for chip components from headset internals. These labels are listed in Table 1 and were used for later data retrieval in section 4.2.

4.2. Hardware

I sought out teardowns of the headset, and read through official documents that listed, at least generally, the used hardware specs. Using two teardowns I could find images from; I was able to identify many of the major components of the board [21, 22]. A dedicated tracking chip, a dedicated audio processor, the main flash storage, a Bluetooth low energy module, additional memory, and core processor. These were manufactured by TKD, Cmedia, SanDisk, Nordic, and SKhynix, Qualcomm. Details on chip information are listed in Table 1.

I was able to locate specification sheets for some of these components. The audio, additional memory, and core processor documents were locked behind either paywalls or having to contact the manufacturers directly to request access, a co-requirement being that the request comes from a verifiable company.

I also assessed the physical I/O interfaces, as these have been used to varying degrees in other projects. As an inventory, the HMD has 2 microphones, 2 speakers, a 3.5mm headphone jack, a USB-C port, 2 eye displays, and 4 cameras. It also has a light sensor which sleeps/wakes the headset based on light exposure.

4.3. Firmware

This is a very low-level analysis of the system logic and is a little beyond my experience level/available tools/inclination towards doing a teardown on a system I had one physical copy of.

4.4. Communications

The Quest 2 has both Wi-Fi 6 and Bluetooth 5.0 capabilities. There are a few configuration methods for both, however Bluetooth capabilities are often enough locked to what has been configured by the OEM (Original Equipment Manufacturer). This is potentially good or bad in searching for a vulnerability: poorly configured Bluetooth defaults are by their nature much more vulnerable, but configured properly, modern Bluetooth can be quite secure.

Wi-Fi security is at the general mercy of the host network and is much more variable in natural settings. For the sake of examination, there can be exploitation of the fact that network keys will be known to devices besides the Oculus, for packet sniffing purposes.

The Oculus line of products in general also have a communication method called Link, which is a

Table 1. Listing of chip information discovered

Manufacturer	Chip Model	Chip identifier	Chip usage	Documentation acquired
TKD	ICM-42688-P	I428P	6-axis spacial tracking	Spec sheet
Cmedia	CM7104	CM7104	Audio processing	Partial from online catalog
SanDisk	iNAND 8521	SDINDDH4-64G	NAND flash storage	Spec sheet
Nordic	nRF52832	N52832 QFAAE0 2022IM	Bluetooth LE	Spec sheet
SK hynix	H9HKNNNEBMAVAR-NEH	H9HKNNNEBMAV ARNEH 003A	BGA memory	none
Qualcomm	Snapdragon XR2	undiscovered	ARM processor	Partial from Qualcomm

physical tether to a host computer, or AirLink, which is possibly the same base application, but run over wireless. These services both allow for the streaming of video data direct from the headset to the receiving device. The hope is that perhaps the streamed data is not encrypted beyond that which the Wi-Fi host's standard is, or even TLS. On a personal network, one could use a variety of tools to access this data, if so. Tools would include Wireshark, and SSLstrip.

4.5. Operating system

Android 10 is both 2 versions below the latest iteration, and like all Android versions, built from the Linux kernel. This means that there is the potential for core vulnerabilities to propagate up from both the kernel level and the original unmodified Android OS. Within a kernel level, there are built-in utilities, permissions, and core libraries, as well as general system architectural choices.

Within the broader OS, we have more architectural choices, as well as native applications. Native applications like a web browser, which depending on browser engine implementation, may have recorded vulnerabilities. With systems like IOS, in the past the web browser has been an attack vector for obtaining elevated permissions, because Safari is a system application with elevated privileges.

4.6 Applications

Finally, consider developed applications. There are two major marketplaces for the Quest 2: the Oculus marketplace, which is officially supported, and SideQuest, which is a 3rd party repository of

applications not available on the Oculus marketplace for various reasons.

SideQuest is a GUI-based application for sideloading applications onto the Quest series of devices. The term 'sideloading' refers to loading an application onto an Android device through ADB (Android Debug Bridge, a standard tool for Android development).

Applications could be created and then sideloaded onto the device through ADB or they could in theory be hosted on the SideQuest or even Oculus marketplaces.

The SDK documentation may not reveal complete functionality, but SDK source code analysis almost certainly will, if available. By exploiting issues with the SDK, custom coded applications could be made to perform attacks, some simply targeting the underlying system, some oriented around attacking a device user, through theft or even possibly to render the headset unusable. Evidence of a previous vulnerability present in version 30 or lower of the system, allowed background applications to access user motion data, was based around the exploiting the SDK's vulnerability [23].

5. Future work

While many technical aspects have been determined, there are is a lack of deep analysis on them. Airlink and Link traffic could be analyzed, and firmware could be dumped and analyzed. Network traffic capture at device initialization, and at several points of the setup process would possibly reveal a user's environment or physical data. The teardown photos were ultimately limited in fidelity and suffered from low or poor resolution on certain chips,

resulting in an inability to determine and document both form and function thereof. A quality teardown and documentation of all discernible parts would be useful to fill in these gaps.

There is room for analysis of the GitHub repository. Examining extent of what is present, but with the assumption that most of the custom applications (as well as other in-house developed aspects of the system) will not be in this repository, because as a company, if Meta does not have to publicly host specific pieces of source, then they will not.

6. Conclusion

Many openings remain to search through in future work. Digging deeper into this analysis and trying to gain data on some of the more immediately approachable areas previously mentioned would be a reasonable target for vulnerability research.

7. References

- [1] BCG, "Augmented (AR), virtual reality (VR), and mixed reality (MR) market size worldwide from 2021 to 2024 (in billion U.S. dollars).", ed. bcg.com: BCG, 2021.
- [2] T. Alsop, "Virtual reality (VR) headset unit sales worldwide from 2019 to 2024 ", ed. arinsider.co: AR Insider, 2021.
- [3] H. G. Richard Mayne, "Virtual reality for teaching and learning in crime scene investigation," *Science & Justice*, vol. 60, no. 5, pp. 466-472, 2020. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S135030620300927>.
- [4] N. o. 41, "What really is the difference between AR / MR / VR / XR ?", vol. 2021, ed. Medium.com: Medium, 2018.
- [5] Various, "Vircadia Docs," 2021. [Online]. Available: <https://docs.vircadia.com/>
- [6] Disrupt, YouTube.com. I spent a week in a VR headset, here's what happened. (March 7, 2019). Accessed November 11 2021. [Video]. Available: <https://youtu.be/BGRY14znFxY>.
- [7] A. Yarramreddy, P. Gromkowski, and I. Baggili, "Forensic Analysis of Immersive Virtual Reality Social Applications: A Primary Account," 2018 2018: IEEE, doi: 10.1109/spw.2018.00034. [Online]. Available: <https://dx.doi.org/10.1109/spw.2018.00034>
- [8] P. Casey, R. Lindsay-Decusati, I. Baggili, and F. Breitinger, "Inception: Virtual Space in Memory Space in Real Space—Memory Forensics of Immersive Virtual Reality with the HTC Vive," *Digital Investigation*, vol. 29, pp. S13-S21, 2019.
- [9] B. Odeleye, G. Loukas, R. Heartfield, and F. Spyridonis, "Detecting framerate-oriented cyber attacks on user experience in virtual reality," 2021.
- [10] *OBS-OpenVR-Input-Plugin*. (2020). GitHub, GitHub.com. Accessed: November 11, 2021. [Online]. Available: <https://github.com/baffler/OBS-OpenVR-Input-Plugin>
- [11] *OpenVR SDK*. (2016). GitHub, GitHub.com. Accessed: November 11, 2021. [Online]. Available: <https://github.com/ValveSoftware/openvr/blame/master/headers/openvr.h#L3427>
- [12] D. B. Steve White, Michael Satran "Introduction to a Device in Direct3D 11," *Direct3D 11 Graphics* docs.microsoft.com: Microsoft, 2021. [Online]. Available: <https://docs.microsoft.com/en-us/windows/win32/direct3d11/overviews-direct3d-11-devices-intro>
- [13] G. T. H. G. Roussel-Tarbouriech, N. Menard, T. True, and T. Vi, "Methodically defeating nintendo switch security," *arXiv preprint arXiv:1905.07643*, 2019.
- [14] A. Useche, "Intro to Hardware Hacking - Dumping your First Firmware," April 15, 2019. [Online]. Available: <https://blog.nvisium.com/intro-to-hardware-hacking-dumping-your-first-firmware>
- [15] Facebook. "Supplemental Oculus Data Policy." <https://www.oculus.com/legal/privacy-policy/> (accessed).
- [16] H. Swart, "Oculus Quest 2: How Snapdragon XR2 powers the next generation of VR," ed. OnQ Blog: Qualcomm, 2020.
- [17] Qualcomm. "Snapdragon XR2 5G Platform - Overview." <https://www.qualcomm.com/products/snapdragon-xr2-5g-platform> (accessed).
- [18] Developers, "About Buildroot," *The Buildroot user manual* buildroot.org, 2021. [Online]. Available: https://buildroot.org/downloads/manual/manual.html#_about_buildroot
- [19] *codemod*. (2019). GitHub, GitHub.com. Accessed: November 12, 2021. [Online]. Available: <https://github.com/facebookarchive/codemod>
- [20] FCC. "Facebook Technologies, LLC - FCC Wireless Applications." FCCID. <https://fccid.io/2AG0Z> (accessed November 13, 2021).
- [21] @GOROman. "Oculus Quest 2 Teardown by @GOROman." imgur. <https://imgur.com/a/45QtW3u> (accessed November 13, 2021).
- [22] sctanf and Contributors. "Oculus Quest 2 Disassembly." iFixit. <https://www.ifixit.com/Guide/Oculus+Quest+2+Disassembly/139759> (accessed November 13, 2021, 2021).
- [23] Oculus, "Oculus Release Notes," 2021. [Online]. Available: <https://support.oculus.com/release-notes/>.