Traffic and Security Analysis Case Study on Virtual Reality Devices: Meta Quest 2

Joey Vongphasouk, *Colorado School of Mines*

***Abstract*—Virtual reality (VR) devices such as the Meta Quest 2 have seen a considerable rise in relevance and usage over the past decade, and have become integrated into many consumer applications and work areas. As a device that takes in and outputs a considerable amount of sensor information, the device can potentially give out user activities and personally identifiable information (PII) similar to an IoT device. In addition, security-breaching technology is becoming more easier to obtain. This paper aims to analyze the traffic and security of the Meta Quest 2 using common and easily accessible methods similar to the ones used for IoT device analysis. It is found that through free network analysis programs such as Wireshark, the output of VR devices can be seen and analyzed to infer accurate user activities. Additionally, further decryption provided by open-source tools can be obtained through a few additional steps and output PII about the user, the device, and its sensors.**

***Index Terms*—WireShark, Network Traffic, Internet of Things, Personally Identifiable Information, Virtual Reality, Domains.**

I. Introduction

THE domain of the Internet of Things (IoT) has been seen to grow at a considerable pace over recent years [1][2]. The devices in IoT utilize sensors and a connected network to achieve effective results. As a result, many fields and activities have utilized sensors connected to the internet. Devices that implement virtual reality (VR) share this aspect in using sensor data to simulate a generated environment. Since VR devices share similar data needs and outputs as IoT devices, security and privacy should be of concern, especially due to the ongoing development of VR devices.

With network analysis and decryption tools becoming more freely available to the wider public, the security of VR devices become increasingly more important. The goal of this paper is to see if information about a user can be gained using the cheap and available tools that a malicious attacker can gain access to.

II. Related Work

The topic of security in VR devices is not a novel concept and has been explored in many different ways before this paper. All of these papers achieved different goals that helped highlight the need for secure VR applications.

A preceding work that utilizes network capture and data analysis is provided in the model of OVRseen [3]. Trimananda et al. have constructed a system that collects network traffic from a given network capture and compares it to the privacy policies of the app obtaining such data. The data is initially provided in the form of .pcapng files in an encrypted format. Then, by decrypting both the transport layer security (TLS) and Unreal/Unity encryption keys by known methods, information such as the application used, PII, and destination of data can be found. The goals of this paper differ from OVRseen where we look to inspect the overall process and level of difficulty in capturing and analyzing VR data. OVRseen's goal looks to showcase privacy and information misuse and uses a complex system to analyze the data.

Previous work has also seen specialized case studies for Development, Security, and Operations (DevSecOps) in VR software. Ersin et al. [4] conducted a case study on security development through the lens of VR programs. The paper looked to give students tasks that consider programs in the context of threat models and security/privacy frameworks. As a result, Ersin et al. gave foundation to student development in DevSecOps and connections between VR and secure development. Our paper differs by focusing on network analysis vulnerabilities and highlighting this process rather than frameworks and threat models to consider.

Additional papers have seen work in general VR security. Stephenson et al. [5] have looked into the authentication of virtual reality and augmented reality (AR) systems and have provided a groundwork of what troubles users and developers in the field. Conducting a survey given to VR and AR users, Stephenson et al. have compiled four properties of user authentication and compared these to both existing and researched solutions. This paper looks at the design flaws of authentication handling resulting in security concerns, whilst our paper aims to address security concerns in network capture and traffic.

Analysis of IoT network traffic shape is also not a novel idea and has been explored many times [6][7]. Apthorpe et al. have looked at traffic data from multiple IoT devices and have been able to make accurate user activity inferences based on device type and traffic patterns sent. Our paper looks to expand this idea into VR devices, which are a combination of multiple sensors and send much more data than the ones used in Apthorpe's paper.

III. Background

A. VR Device Case Study: Meta Quest 2

The VR device used in this study is the Meta Quest 2 (formally the Oculus Quest 2). Developed by Meta, the Meta Quest 2 is a standalone VR headset that comes with a head-mounted display (HMD) and two Touch controllers. The Quest 2 provides six degrees of freedom for the user in translational

Report turned in on December 3, 2024

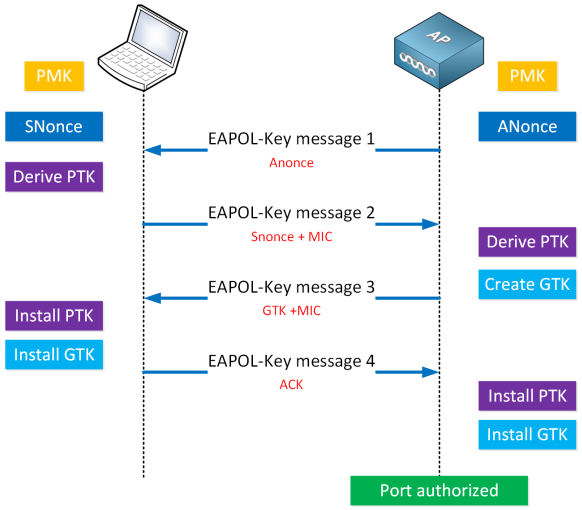

Fig. 1: Four-way handshake for device authentication []

Fig. 2: Raspberry Pi with Kali Linux OS implementation using the ALFA AWUS036ACS network adapter

movement (X, Y, Z) and rotational movement (pitch, yaw, roll) [8]. To track the movement of the controllers, the HMD uses four infrared cameras to track its position in the room as well as the controller positions relative to the HMD. This is done through a blinking light on the controllers, which is sensed from the HMD. Controllers separately use a proprietary protocol to communicate [9]. Additionally, the HMD contains an input microphone, an output speaker, and dual OLED displays. Quest 2 Models further have a face sensor to detect if a user has the HMD equipped.

B. WPA Protocol: 4-way Handshake

A common security protocol widely used in existing networks is Wi-Fi Protected Access (WPA). Developed for the IEEE 802.11 protocol which previously used Wired Equivalent Privacy (WEP), WPA uses multiple keys in a hierarchy to gain secure its contents [10]. For a device to establish a connection to a network using WPA, the device and network access point uses a four-way handshake that authenticates the device and creates all the keys necessary for WPA. The four-way handshake is illustrated in Figure 1.

The handshake uses Extensible Authentication Protocol over LAN (EAPOL) as the transport protocol to communicate between the network access point and the device station [11]. This protocol then allows for the handshake of keys to take place. First, the network access point sends a message to the client device communicating the ANonce random number. The client receives the packet and makes a Pairwise Transient Key and responds by sending a SNonce and message integrity code (MIC). The access point checks MIC values and sends the Group Temporal Key if it finds no problems. Finally, the station sends back an acknowledgment to the access point.

This paper will look to find this transaction in VR devices, and grab such information to derive information sent over the network from the VR device station.

IV. Methodology

The case study begins by looking to implement the project onto as many different systems as possible to illustrate the level of difficulty and accessibility to network traffic data. The experiment was performed on three different systems: Raspberry Pi using Kali Linux operating system (OS), a virtual machine using Kali Linux OS on a Windows machine, and MacOS. Each machine was chosen in regard to popular OS options widely available.

The standalone Kali Linux setup used a Raspberry Pi 4 Model B connected using the ALFA AWUS036ACS network adapter. This adapter was chosen as this adapter had the RealTek RTL8812AU chipset required for monitor mode enabling. Additionally, the Kali Linux OS was chosen as network penetration testing was already widely available to the user at installation and was installed through a MicroSD card preconfigured with the OS Raspberry Pi image. The airmon-ng functionality was then used in conjunction with Wireshark to capture traffic data. The overall setup is shown in Figure 2. Refer to Appendix A for more setup information.

The Windows implementation utilized a Kali Linux image on a virtual machine (VM). The virtual machine was run using the VMWare Workstation Pro software, which became free to use in May 2024. The pre-built image was downloaded from the official Kali Linux website. The Windows network adapter did not have a feasible chipset that supported monitor mode, thus the ALFA AWUS036ACS was used again. After the VM was set up, similar proceedings occurred on this system to capture network data. The overall setup is shown in Figure 3. Refer to Appendix A for more setup information.

The MacOS implementation utilized only Wireshark to capture network data as all other needed components had already been fulfilled. This implementation used a MacBook Pro with Apple M3 chipset ran on MacOS Sequoia 15.1.1. This chipset supported the monitor mode needed for network sniffing, thus no external hardware was needed. Usual methods of enabling monitor mode had been deprecated in the current OS version, thus a workaround was used. The Wireless Diagnostics tool already installed on the machine additionally comes with a network sniffer tool. Activating the tool on arbitrary channels and frequencies provided similar results. Wireshark was then able to capture the network after starting the sniffer tool. The overall setup is shown in Figure 4.

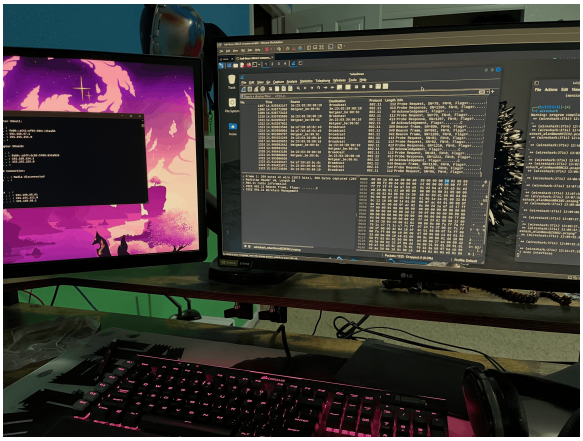

Fig. 3: Windows virtual machine implementation using the ALFA AWUS036ACS network adapter

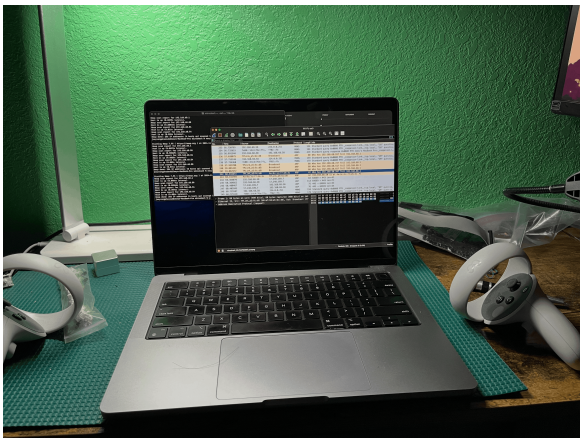

Fig. 4: MacOS Sequoia 15.1.1 implementation

Each implementation utilized a decryption key on the 802.11 protocol. This decryption key followed the wpa-pwd format using the network name and password. The VR headset would be connected to the network and would browse applications such as BeatSaber for around 3 minutes. This process is repeated every capture to obtain the decrypted 802.11 packets. Then, varying filters were used on the packet captures to inspect certain elements of the capture. A "eapol" filter was used to view successful WPA four-way handshakes. A "wlan.addr == b4:17:a8:c7:1e:6d" filter was also used to filter for packets sent to and from the VR device. After network capture, various Python scripts were used to generate statistics and network plots describing the information. Without further tools and complexity, the domain and time of packet capture can easily be obtained and thus plotted and analyzed.

Additionally, the open-source software OVRseen was used to gather more data from the network captures. Specific instructions for using OVRseen are available on its GitHub repository. Testing with the three implementations' datasets proved less effective than those from OVRseen due to the latter's larger and more varied information. Consequently, analysis was conducted using OVRseen's datasets.

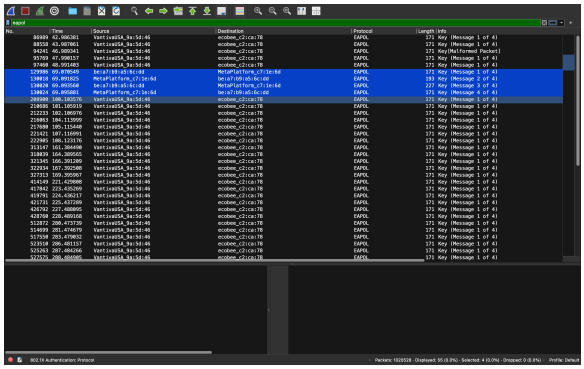

Fig. 5: WPA four-way handshake capture of Quest 2 device

V. Results and Evaluation

Each implementation of the network sniffer was able to capture packets on the provided network through the 802.11 protocol. Additionally, all implementations were able to capture the WPA four-way handshake when the VR device connected to the network. An example capture is shown in Figure 5, showing the connection between the access point represented by the MAC address "be:a7:b9:a5:6c:dd" and the client station represented by the MAC address "MetaPlatform_c7:1e:6d". As a result of the handshake, the network sniffer can decrypt surface-level information regarding the packets sent to and from the device.

After filtering for the VR device, the surface network analysis can be made to grab user information. Figure 6 shows a plot of domains seen in a three-minute capture of the VR device running the BeatSaber application. Additional connections can be made through time of packet capture, which can be seen in Figure 7. This plot used a time bucket of 1 second to capture all data packets sent from a specific domain, and plots such over frequency and time. Utilizing traffic shape analysis, information can be gained such as the VR device start time and application start time, even when PII information has yet to be decrypted.

Using the open-source system provided by OVRseen can also decrypt information in the packet. Running the system on network capture the team had captured was not as effective in providing PII due to the smaller amount of data used and applications tested. However, OVRseen provided a wider and considerably larger set of data and was simulated through the system's post-processing stage after network capture. Figure 8 shows the results of the system's post-processing stage which highlights PII information sent to a domain. The PII includes information such as user ID, device ID, the application ran, and telemetry data. More information can be gathered and summarized in the OVRseen paper.

These results were made using free and open-source tools available online and applicable to all three different OS implementations. More information can be gained about the user's activities both from an extended network shape analysis or deeper usage of the OVRseen system. This illustrates the level of ease with which malicious attackers are able to view information about users through VR devices.

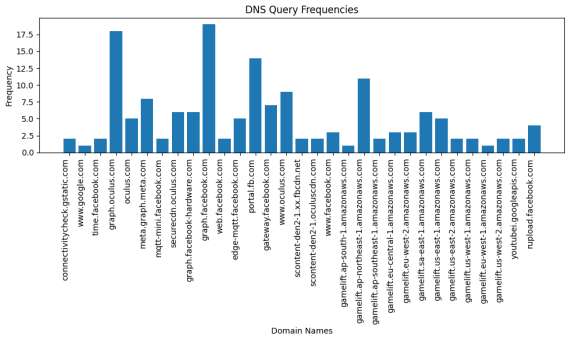

Fig. 6: Domain frequency of network capture packets on BeatSaber over 3 minutes

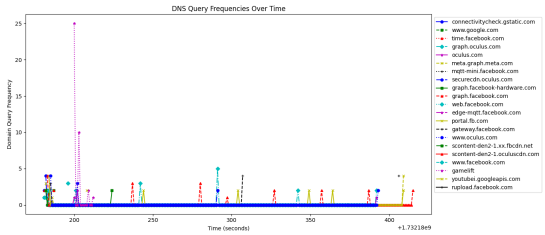

Fig. 7: Domain frequency over time (3 minutes) of network capture packets on BeatSaber

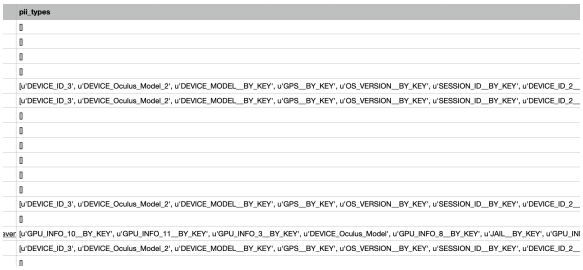

Fig. 8: OVRseen personal identifiable information gain from a domain

VI. Future Work

Network transactions and communications can become largely different across VR devices. Testing such methods across multiple VR devices can give more insight into vulnerabilities in both the individual device and the overall domain. Devices may also come with more or different sensors, such as the Valve Index, which utilizes base stations to calculate telemetry and additional sensors on the controllers to give a more accurate depiction of the simulated world. These devices may give more PII, utilize different encryptions, or send different traffic shape patterns, and should be analyzed on an individual basis.

Additionally, more applications and longer packet captures can be considered for further study. This provides more data points to give more conclusive and accurate results regarding the accuracy of user interaction predictions. Additional applications will also allow for the usage of OVRseen to be replicated to a greater extent and accurately give a better reading on the ease of VR device traffic sniffing.

Another pursuit can be the extension to augmented reality (AR) devices, as such devices pursue similar goals to VR devices. Packet capture and analysis with decryption may produce differing results given the lower computational power on the device itself but may provide similar PII on the user due to its need for cameras. Devices such as the Ray-Ban Meta glasses may also be considered for privacy of information regarding usage of PII.

VII. Conclusion

It is becoming easier for malicious actors to gain access to tools that can breach privacy and security in devices. Online software and open-source research programs can be obtained for free and used in a malicious way to breach encryption. As a result, it becomes important for devices that take in and output sensitive information about the user to be aware of these potential threats.

To better illustrate this point, a collection of implementations utilizing common and free software was used to capture and analyze the network traffic from a VR device. All three tested implementations each using different operating systems had shown the ability to capture network traffic data through Wireshark provided the network adapter had a chipset with monitor mode. By utilizing a decryption key for the network, the four-way handshake of the VR device and the network can be found and subsequent packets to and from the device can be captured. These captured packets show information about user activity through both accessed domain and network traffic shape analysis. Furthermore, with the usage of OVRseen, an open-source system, packets can further decrypt into PII that identifies the user account. This paper demonstrates that user activity and PII can be determined using free online resources.

References

[1] J. M. Kizza, “Internet of Things (IoT): Growth, Challenges, and Security,” in Guide to Computer Network Security, Switzerland: Springer International Publishing AG, 2024, pp. 557–573. doi: 10.1007/978-3-031-47549-8_25.

[2] J. Wang, M. K. Lim, C. Wang, and M.-L. Tseng, “The evolution of the Internet of Things (IoT) over the past 20 years,” Computers industrial engineering, vol. 155, pp. 107174-, 2021, doi: 10.1016/j.cie.2021.107174.

[3] R. Trimananda, H. Le, H. Cui, Janice Tran Ho, A. Shuba, and A. Markopoulou, “OVRseen: Auditing Network Traffic and Privacy Policies in Oculus VR,” arXiv.org, 2021.

[4] Dincelli, Ersin, and Alper Yayla. ”Security and Privacy Implications of Virtual Reality Applications in the Metaverse: A Case of Development, Security, and Operations (DevSecOps).” Journal of Information Systems Education, vol. 35, no. 3, summer 2024, pp. 261+. Gale Academic OneFile, dx.doi.org/10.62273/JMZA1065. Accessed 7 Oct. 2024.

[5] S. Stephenson, B. Pal, S. Fan, E. Fernandes, Y. Zhao, and R. Chatterjee, “SoK: Authentication in Augmented and Virtual Reality,” in 2022 IEEE Symposium on Security and Privacy (SP), IEEE, 2022, pp. 267–284. doi: 10.1109/SP46214.2022.9833742.

[6] N. Apthorpe, D. Y. Huang, D. Reisman, A. Narayanan, and N. Feamster, “Keeping the Smart Home Private with Smart(er) IoT Traffic Shaping,” Proceedings on Privacy Enhancing Technologies, vol. 2019, no. 3, pp. 128–148, 2019, doi: 10.2478/popets-2019-0040.

[7] P. Chaudhary, et al. “Network Traffic Analysis using Wireshark,” 2023.

[8] “Oculus Quest (1 2),” Equal Reality, Jul. 27, 2022. https://equalreality.com/courses/facilitator-training/lessons/oculus-quest/ (accessed Dec. 04, 2024).

[9] D. Coldewey, “How Oculus squeezed sophisticated tracking into pipsqueak hardware — TechCrunch,” TechCrunch, Aug. 22, 2019. https://techcrunch.com/2019/08/22/how-oculus-squeezed-sophisticated-tracking-into-pipsqueak-hardware/

[10] R. Molenaar, "WPA and WPA2 4-Way Handshake," NetworkLessons.com, Dec. 12, 2023. https://networklessons.com/wireless/wpa-and-wpa2-4-way-handshake
[11] "EAPoL Protocol — Extensible Authentication Protocol over LAN," vocal.com. https://vocal.com/secure-communication/eapol-extensible-authentication-protocol-over-lan/

Joey Vongphasouk Joey Vongphasouk graduated at the Colorado School of Mines with his Bachelor of Science degree in Computer Science with a focus on Computer Engineering. He is currently pursuing his Master of Science degree at the Colorado School of Mines.

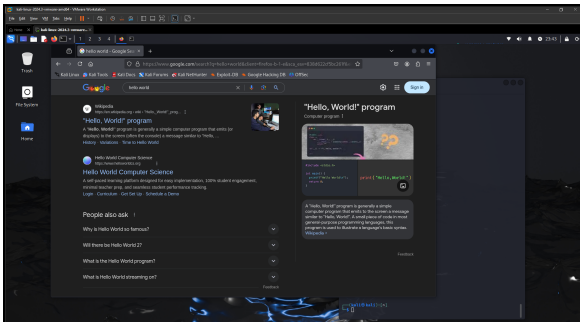

Fig. 9: Working Linux virtual machine with access to network

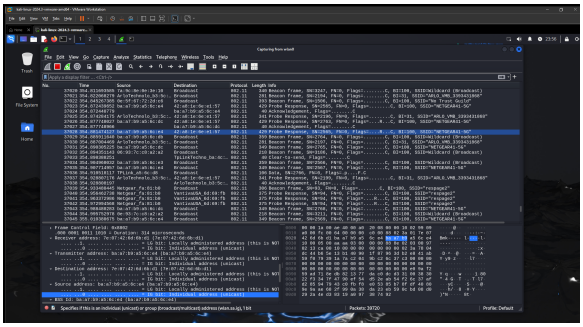

Fig. 10: Wireshark monitor onto 802.11 using a virtual machine

APPENDIX A
HARDWARE SETUP FOR LINUX/WINDOWS VMWARE

- VR Device: Meta Quest 2
- Wireless Adapter: ALFA Network AWUS036ACS (Drivers: github.com/aircrack-ng/rtl8812au)
- Operating System: Kali 2024.3
- Software: Wireshark v4.0, Monitor Mode enabled via iwconfig, VMWare Workstation Pro
- Replication commands from working network, Figure 9:
  - sudo apt-get update
  - sudo apt-get upgrade
  - sudo airmon-ng check kill
  - sudo airmon-ng start wlan0
  - wireshark
- Wireshark is ran on wlan0 with 802.11 network decryption key, Figure 10.