

Inception Attacks: Immersive Hijacking in Virtual Reality Systems

Zhuolin Yang, Cathy Yuanchen Li, Arman Bhalla, Ben Y. Zhao, Haitao Zheng
Department of Computer Science, University of Chicago

ABSTRACT

Recent advances in virtual reality (VR) system provide fully immersive interactions that connect users with online resources, applications, and each other. Yet these immersive interfaces can make it easier for users to fall prey to a new type of security attacks. We introduce the *inception attack*, where an attacker controls and manipulates a user’s interaction with their VR environment and applications, by trapping them inside a malicious VR application that masquerades as the full VR system. Once trapped in an “inception VR layer”, all of the user’s interactions with remote servers, network applications, and other VR users can be recorded or modified without their knowledge. This enables traditional attacks (recording passwords and modifying user actions in flight), as well as VR interaction attacks, where (with generative AI tools) two VR users interacting can experience two dramatically different conversations.

In this paper, we introduce inception attacks and their design, and describe our implementation that works on all Meta Quest VR headsets. Our implementation of inception attacks includes a cloned version of the Meta Quest browser that can modify data as it’s displayed to the user, and alter user input en route to the server (e.g. modify amount of \$ transferred in a banking session). Our implementation also includes a cloned VRChat app, where an attacker can eavesdrop and modify live audio between two VR users. We then conduct a study on users with a range of VR experiences, execute the inception attack during their session, and debrief them about their experiences. Only 37% of users noticed the momentary visual “glitch” when the inception attack began, and all but 1 user attributed it to imperfections in the VR platform. Finally, we consider and discuss efficacy and tradeoffs for a wide range of potential inception defenses.

1 INTRODUCTION

Recent advances in Virtual Reality (VR) hardware and software are poised to change the way we interact with the world and each other. VR headsets have the potential to deliver users a deeply immersive experience comparable to reality itself. They also serve as a way to bridge vast distances, facilitating enhanced social and (remote) workplace interactions through the use of personalized avatars or digital representations of us.

The flip side of these immersive capabilities is that when misused, VR systems can facilitate security attacks with far more severe consequences than traditional attacks [39]. In particular, VR systems can use immersive sensory input to manipulate the users into a false sense of comfort, misleading them to leak private and sensitive information (e.g. authentication credentials to financial accounts) or to trust what they see/hear (e.g. gestures, movements, and conversations).

For example, consider the following scenarios:

Scenario 1. *Alice spends 4 days a week working remotely out of her home office via her employer’s new VR application. Each morning, she puts on her headset and opens the app to start her daily routine. Today, she notices that the intranet server for her employer companyX is taking an extra long time to recognize her corporate password in VR. She makes a mental note to reset her home WiFi router. Elsewhere, Carl sits and watches his VR headset as it finishes recording Alice’s login sequence.*

Scenario 2. *Alice takes off her VR headset in frustration, after the latest argument with her partner Madison during their normal VR date night. Madison had been increasingly distant all week, and actually ended their relationship tonight. Elsewhere, Carl turns off his generative voice interface and watches in satisfaction as a confused Madison storms off.*

Both scenarios described above result from a single attack, where an attacker (e.g. “Carl”) compromises the integrity of the users’ VR system, and inserts his own software in between the user and their VR layer. Under this attack, users (e.g. “Alice”) are no longer interacting directly with their expected VR counterparts (e.g. “CompanyX’s servers, Madison”). Instead, users are actually operating in Carl’s VR layer, interacting with their VR counterparts through a level of indirection, and these interactions are being monitored, recorded, and modified real-time by Carl. Given VR’s immersive properties, Carl is able to replicate the user’s normal interactions with network servers (similar to a man-in-the-middle attack) and other users in his own “inception VR layer,” and users are unable to distinguish which VR layer they are in. Hence, we call this the *inception attack*¹, and provide a high-level illustration in Figure 1.

Inception attacks are very powerful, because of two properties. First, they can be extremely difficult to detect, because VR apps are often designed to approximate the real world and avoid indicators and prompts that characterize traditional computer applications. Thus a user currently has no way of authenticating if any part of their immersive experience (visual or auditory) comes from a particular VR app (*VR authentication*). Second, an inception attacker can take near total control of the VR experience, not only eavesdropping and recording data, but also altering information and experiences for the target user in real time (*VR interaction confidentiality and integrity*). Not only can an attacker silently observe a user’s interactions with a VR provider, recording input, passwords, or other sensitive data, but they can also hijack and replace entire social interactions. For example, in the aforementioned Scenario 2, if Alice becomes a target of an inception attack, she can engage in a one-on-one social interaction with Madison in VR, where both users only hear and see what the attacker wants them to see and

arXiv, March 2024.

© 2024 Copyright held by the owner/author(s).

This is the author’s version of the work. It is posted here for your personal use. Not for redistribution.

2024-03-12 01:11. Page 1 of 1–15.

¹The inception attack is inspired by both its namesake, the 2010 Christopher Nolan film *Inception*, and movies with similar attacks, including the 2018 Steven Spielberg film *Ready Player One*.

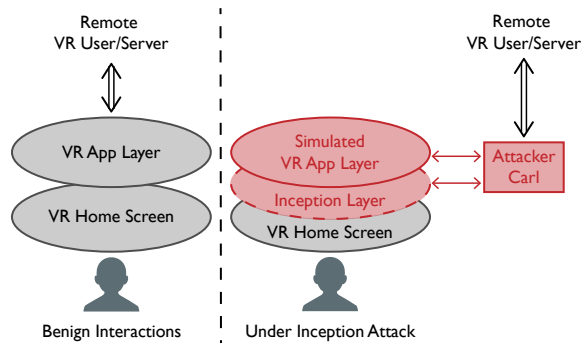


Figure 1: Inception Attacks: A user thinks they are interacting directly with a VR app launched from the VR home screen, when they are in fact running a simulated VR app inside the attacker’s inception layer.

hear. This is made even easier with the use of generative AI tools that can replicate human voices and generate visual deepfakes.

Inception attacks can be launched on VR systems today by exploiting any of a number of vulnerabilities. In some cases, vulnerabilities are basic issues already addressed in traditional computing applications, yet overlooked in VR systems. In other cases, traditional authentication mechanisms are made more challenging due to the nature of VR applications and interfaces.

In this paper, we identify a broad class of inception attacks and their dangers to current VR systems and architectures. We discuss multiple launch points for inception attacks on current VR platforms, describe and evaluate (via user study) an example of an inception attack on Meta Quest Pro, and analyze a range of security improvements necessary to limit or prevent inception attacks. To the best of our knowledge, we are the first to identify, implement, and evaluate this new attack.

Our work makes the following key contributions.

- We introduce inception attacks, and describe multiple attack vectors that facilitate the successful launching of inception attacks.
- We implement a full inception attack on Meta Quest VR headsets, capable of hijacking a user’s headset and home environment, replicating and mimicking benign applications (Quest Browser and VRChat), eavesdropping on user inputs, and altering a variety of user experiences and user inputs (e.g. modifying user financial transactions and modifying live user-to-user audio in VRChat).
- We perform an IRB-approved, in-person VR user study with 27 participants. In this deception study, participants interact with applications as their session is hijacked by an inception attack. Roughly 1/3 of users noticed the glitch as their session was hijacked, and all but 1 attributed the glitch to benign system performance issues.
- We provide a thorough analysis of a range of potential defense measures, and comment on their efficacy and usability tradeoffs against inception attacks.

2 BACKGROUND AND RELATED WORK

As background, we briefly describe how VR systems support a diverse set of applications, and existing attacks against VR systems.

2.1 VR for Non-Gaming Applications

Initially intended for gaming, VR has already made significant impacts on a diverse spectrum of industry sectors, from architecture and design [36], education [25, 46], finance [7], social chat and date [18, 63], healthcare [17, 27, 53, 56], to everyday business operations [24]. For these sectors, VR is particularly appealing because it not only eliminates geographical barriers for social and professional interactions, but also delivers unmatched realism and immersion that boost awareness and operational efficiency.

To support a diverse range of applications, modern VR headsets are equipped with enhanced computing and networking capabilities. These include GPUs to support local 3D rendering, and wireless networking modules to ensure smooth connections with devices and servers. Customized VR systems and tools like Meta Quest for Business [34] and SideQuest [51] also allow professionals and individuals to install and manage private apps outside of official app stores, and for users to find and install non-store content on their VR devices [5].

2.2 Existing Attacks in VR

We now summarize existing attacks against VR systems.

Malware. Like all other computing devices [60], VR devices are susceptible to malware. Attackers can introduce malware directly onto VR headsets by malicious apps, or infect the devices connected to headsets, which then install malware onto these headsets. Some privacy attacks have assumed a successful injection of malware to gain access to VR users’ motion data [28, 29, 54].

Privacy Attacks. To deliver immersive experiences, VR headsets employ multiple sensors to continuously gather data on user activities, giving rise to privacy attacks. A rich collection of existing works [39] have shown that records of user motion, collected by VR headsets, can be used to reveal private information such as identity, age, gender, ethnicity, emotion, and health conditions (e.g. [20, 40, 72]). Recent works have also exploited ways to eavesdrop on individual VR app sessions. The man-in-the-room attack [62] infiltrated a chat room app called Bigscreen, so that unauthorized attackers can join a *private* virtual chat room as an invisible user and thus observe the behaviors of users in this private environment. Others show that the avatar’s movements observed in a shared virtual environment can be used to recover keystroke content [70] or identity [16, 47].

Perceptual Manipulation Attacks. A recent user study [9] highlights that if attackers are able to modify the screen feed of a mixed reality (MR) headset, they can introduce physical harm to users, e.g. creating physical collisions or inducing motion sickness to its user. Similarly, another work [8] shows that if attackers can obtain write privileges on VR system configuration files, they can shift the virtual environment to deceive users into adjusting their physical locations [8]. One can also develop a malicious application that shifts users’ virtual locations when they use the app [57].

3 INCEPTION ATTACKS

In the context of VR systems, we define an **Inception Attack** as:

an attack where the attacker controls and manipulates the user's interaction with their VR environment, by trapping the user inside a single, malicious VR application that masquerades as the full VR system.

An inception attack inserts the attacker between the user and any external entities, by creating a simulation of the VR system's *home environment* and *applications* inside the malicious app and trapping the user inside. While the user thinks they are interacting normally with different VR applications, they are in fact interacting within a simulated world, where everything they see and hear has been intercepted, relayed, and possibly altered by the attacker.

When a user is under an inception attack, all of their input (voice, motion, gestures, keystrokes) and VR app output (virtual browsers, audio/video from remote servers, interactions with other VR users via avatars) can all be eavesdropped, recorded, or modified in real-time by an attacker. For example, in Scenario 1 (discussed in §1), an attacker can compromise any user authentication scheme within an inception attack, because everything seen by the user (prompts, virtual keyboard, challenges and responses) can be captured by the attacker. Similarly in Scenario 2 (§1), an attacker can intercept and alter interactions between users, altering audio and visuals (possibly augmented with generative AI tools) to create altercations or spread false information.

At a high level, the inception attack is analogous to a “man-in-the-middle” (MITM) attack on a network, applied to the VR context. Like MITM attacks, inception attacks can be launched by taking advantage of a variety of security vulnerabilities. In addition, inception attacks can come in a variety of variants, depending on the ultimate goal (e.g. eavesdropping or altering long-term VR interactions). In the rest of this section, we discuss different threat models and vulnerabilities that attackers can leverage to launch inception attacks. We then describe different varieties of inception attacks that can be launched under those threat models, including attack variants that run locally on the headset and/or over the network, and those that run in the background or as foreground VR apps.

3.1 Threat Models and Points of Entry

Inception attacks can be launched from multiple points of entry. Here, we consider *two* different threat models where the adversary has decreasing levels of privileges in the target VR system. For each threat model, we describe how an attacker can potentially achieve this access level.

Threat Model 1. The adversary has the highest privileges (i.e., root access) to the target VR system. An attacker can obtain root access using a variety of traditional attack vectors, including physical access/hacking, privilege escalation, remote attacks, or traditional jailbreaking methods. Root access allows the attacker to control what the user sees or hears by directly tapping into the device's display and audio channels. But an easier alternative is to install a malicious app with a hidden inception layer inside and run it when turning on the headset. The user in their VR home screen might notice a flicker before they are moved into the inception layer.

We note that the current generation VR systems lack many of the security protection mechanisms available in operating systems

for desktop or mobile computing systems. More specifically, VR systems have no secure bootloader or user authentication to prevent an adversary from gaining control of the system. For example, open-source VR system drivers [38, 43] are available for existing VR devices such as Oculus Rift [68] and HTC Vive [67].

Threat Model 2. The adversary has no root access but is able to run a malicious app that includes the inception attack hidden inside (i.e., “an inception app”). This can happen in two ways.

First, many VR systems (e.g., Meta Quest 2/3/Pro, VIVE Focus 3 and PICO 4) support sideloading of VR apps for enhanced usability [44, 51, 61]. Sideloading allows VR headsets to install and run apps not from the official AppStore [69, 73], which can be initiated from a remote server connected to a VR headset that has sideloading enabled. This functionality is available on existing VR systems including Meta Quest 2, 3 and Pro [32]. Thus, a remote attacker can install an inception app on a target headset that mimics a legitimate app.

Second, for headsets that do not support sideloading of apps, the adversary can embed their inception component into an otherwise benign app, e.g. a *weather app*, and publish it to the AppStore. For VR systems with strong security enforcement like the newly released Apple Vision Pro [3], disabling sideloading does not prevent the target user from installing and running seemingly benign inception apps. In this case, any user who downloads the app from the App Store can become a target of the inception attacks when the hidden functionality is triggered.

As described above, an inception attack just needs to run an app with a hidden inception layer on the target's headset, and that can be done by exploiting a number of diverse security vulnerabilities. This emphasizes the difficulty of preventing these attacks entirely, and suggests that a multi-faceted defense strategy is necessary, i.e., one that combines hardening of multiple attack surfaces and active detection of ongoing inception attacks. We discuss a range of these defensive approaches and their implications in §7.

3.2 Attack Variants

While the inception attack is defined by its ability to hijack a user's immersive VR experience, it can actually be implemented in two very different designs.

Local Background Mode with Root Access. Using a more powerful threat model (e.g. Threat Model 1) where the attacker has root access on the target headset, an inception attack has near total control over the user experience. A root-level attacker can run the attack process (along with necessary computation and storage) as a local background process on the headset, and directly tap into the headset display and audio channels to alter what the user sees and hears. More importantly, changes to the user experience would be transparent and implemented in real-time in a near-seamless way.

Foreground Mode. In an alternative weaker threat model (Threat Models 2), the attacker has limited control over the headset, but is able to either run the inception app without alerting the user or trick the user into running the inception app themselves. This is the most practical and realistic threat model, and is the primary threat model we assume in the rest of this paper. With few if any

exceptions, a stronger threat model (e.g. root access) can perform any attack or technique described in this weaker threat model.

This scenario has several implications on the rest of the attack. First, lack of root access means that the inception attack will likely be limited to running as a single foreground app process. Computation, storage and access to device components on the local headset will be limited.

To address this challenge, the local inception app can leverage assistance from an external server, where any additional computation (e.g. alternation/generation of human voices or text) and storage will reside. Furthermore, the attack server runs a copy of the legitimate VR app “on behalf of” the target user to interact with their VR counterparts, by either connecting (via USB) to a separate VR headset or running a headset emulator. As such, the attack server acts as a MITM and relays data (with any desired alternation) between the user and their VR counterparts.

In this approach, the attack performance is restricted by the network performance. For instance, our existing network bandwidth is sufficient to transfer conversation audios but insufficient to support transmission of visual inputs in frames. While 90 fps is the most commonly used screen display setting on a VR system [66], in our experiments, we observe an average fps=32 when transferring frames (screen recordings of another Quest Pro headset) to a Meta Quest Pro headset through a 5GHz WiFi network.

4 IMPLEMENTATION ON META VR HEADSETS

We present an implementation of the inception attacks on today’s Meta Quest headsets. Our implementation is applicable to all three versions of the Meta Quest headsets, i.e., Quest 2, 3 and Pro. Later in §6 we describe a user study where we execute the attack in real-time during study sessions and evaluate its effectiveness in deceiving users.

4.1 Attack Overview

This attack implementation follows the Threat Model 2 discussed in §3.1 and runs in the foreground mode. In this implementation, a remote attacker is able to inject and activate the inception app without alerting the user. Specifically, the target user has a clean headset free of any malicious elements. The target puts on the headset, which by default connects the WiFi network to allow usual functionalities, e.g. system updates and connections with peripheral devices. A remote attacker obtains the network connection with the target’s headset, quietly injects the inception app and a spy script, and leaves the network. The spy script automatically activates the inception attack on the headset at an opportune moment, thereby taking complete control over the target’s interactions with the VR system.

The attack implementation includes four components:

- **Bootstrapping** – Through a brief network connectivity, the attacker connects with the target’s headset to run a set of shell scripts via Android Debug Bridge (ADB). These scripts run silently in the background and collect configuration information of the headset, such as configuration files of the home environment and the list of applications on the headset. We discuss the details on establishing the connection with the headset and executing the scripts in §4.3.



Figure 2: An example home environment with 3D background on a Meta Quest Pro headset. Credit: A screenshot taken inside the Pro headset.

- **Simulating home environment** – Using information collected during bootstrapping, the attacker builds a precise replica of the target’s home environment. To avoid suspicion, this replica mimics not only the 3D background and the exact menu (see Figure 2 for an illustrative example), but also the immersive interactions. Furthermore, the replica should support both official environments published by the VR platform (i.e., Meta) and custom-made ones [64]. We discuss efficient methods for replication in §4.4.
- **Simulating individual apps** – Since users can only access applications via the home environment, our inception attack can take full control of interactions between the user and individual applications by building application replicas and placing them onto the simulated home environment. The exact replication methods depend on the application and attack goals. We discuss the key methodology in §4.5 and present the detailed implementations for two apps in §5.
- **Activating inception** – The attacker packs the simulated home environment and apps as an “inception app” and injects it into the target headset via networked ADB access. Also, a spy script, injected together with the app, runs in the background to monitor system events, and activates the inception attack when the user signals the system to exit the current application, which should return them to the home environment. Rather than executing this request as expected, the spy script intercepts and destroys the signal, terminates the application, and initiates the inception app. This smoothly transitions the user into the simulated home environment, thereby starting the inception process. The decision to activate inception during the application exit moment is deliberate, as it presents a natural disruption in the user’s immersive experience, thereby mitigating suspicion. We discuss the detailed implementation in §4.6.

4.2 Preliminaries: Meta Quest VR

Before delving into implementation details, we first describe the workflow and features of the Meta Quest VR system. Meta Quest headsets run on a modified Android operation system. After putting on the VR headset, the user will see the 3D home environment (e.g. Figure 2) and a menu panel to select applications and configure system settings. The user clicks on an app button to enter the app,

and returns to the home environment after exiting the app, typically by pressing the “home” button on the right controller.

A substantial number² of Meta Quest users opt to enable developer mode to unlock functionalities such as sideloading a diverse set of applications, adjusting headset resolution, or capturing screen content. Under developer mode, Meta Quest handsets support the use of Android Debug Bridge (ADB) to accomplish the aforementioned functionalities.

Android Debug Bridge (ADB). ADB is a versatile command-line tool included in the Android Software Development Kit (SDK). With ADB, individuals equipped with a computing machine (e.g. PC) can connect with Android devices and perform a range of tasks, including screen recording, data transfer, debugging, and app installation (i.e., sideloading). Once a connection is established, users/machines gain access to the on-device OS shell, enabling them to execute command lines and backend scripts. These include installing and running scripts, installing or updating apps, deleting data, reading device settings, transmitting data, and more.

With ADB, there are two ways for a machine to communicate with a VR headset: direct USB connection and wireless TCP connection. The direct access requires physical access to the headset. The wireless access requires the machine to be on the same wireless network as the Meta Quest headset. Beyond these two methods, Meta Quest also supports ADB proxy over SSH, another useful tool to enable remote access to Android devices (including Meta Quest headsets) *without* the need to be on the same wireless network [11].

Next, we will discuss how our implementation leverages ADB (and ADB proxy) to deploy inception attacks in practice.

4.3 Bootstrapping

As discussed in §4.1, during this phase the attacker aims to connect to the target headset and run shell scripts in the background to collect essential configuration data. This information is then used to replicate the target’s home environment and installed applications with high precision. Our attack implementation achieves this via ADB.

Obtaining Remote ADB Access. Assuming that the target’s headset is connected to a wireless network, a remote attacker has two methods to obtain ADB access to the headset.

- **TCP Connection** – This method operates under the assumption that the attacker has gained access to the same local wireless network as the target’s headset. For example, this could occur if the target is using a public WiFi network, or if the attacker is within the same institute or company as the target. Using tools like Nmap [41], the attacker scans the wireless network to find Meta Quest headsets with open TCP ports and then requests access. By default, the Meta Quest’s wireless ADB listens to TCP port 5555 and requires no credential-based authentication. Only a pop-up window appears on the headset screen, asking for permission. The pop-up states “Allow USB debugging?” and displays the request device’s RSA fingerprint, which lacks relevance to identifying the device type and owner [48]. Consequently, the user is unable to tell the source of the access request, instead perceiving it as

²For example, SideQuest is a platform for sideloading apps on Meta Quest headsets. It has 2.2 million active users monthly [51]. By default, these users enable the developer mode to allow sideloading.

one of the headset’s peripheral devices and granting the access. Furthermore, since this access request is designed to be accepted by default, it is easy for users to simply *confirm* the acceptance without considering the implications.

- **SSH Connection** – This method assumes the attacker does not have access to the local wireless network of the headset. Instead, the attacker aims to gain remote ADB access via ADB proxy. To do so, the attacker first needs to trick the target into installing a “helper” app on the target’s PC that connects to the target’s headset. For instance, this could occur if the attacker publishes a free app and advertises it as a file-transporting app between a headset and a PC. Yet the real purpose of the app is to establish a remote connection between the target’s PC and the attacker’s PC, which is now indirectly connected to the headset. This remote connection (i.e., not in the same local network) can be established via ADB proxy with SSH [11]. Furthermore, these actions occur without any notification to the user, including the absence of a pop-up window seeking permission.
- **Additional Social Engineering** – With brief physical access to the target’s headset, the attacker can further streamline the first method of remote ADB access. For instance, when an attacker’s accomplice briefly wears the target’s VR headset, the attacker can remotely request ADB access while the accomplice selects “always allow from this computer” in the pop-up window. This grants the attacker’s machine permanent remote ADB access to the headset, preventing the pop-up window from appearing again.

Collecting Headset Configurations. After obtaining the remote ADB access, the attacker has command-line access to the headset’s OS shell and can now command the headset without the target’s awareness. In particular, the attacker runs a sequence of commands to obtain two configuration information from the headset: (1) the specific 3D background used by the target’s home environment, and (2) the list of installed apps on the headset.

Since all the available 3D backgrounds are stored as APK files on the headset, the attacker just needs to identify the exact APK file used by the current home environment. This can be done by:

```
adb logcat
```

which returns a log of system activities including the name of background APK currently in operation. Given the APK name, the attacker can then locate the file on the headset and transfer it to the attacker’s machine:

```
adb shell pm path APK_name
adb pull path_to_APK
```

The typical size of these APK files varies between 20 and 100MB. Using a standard WiFi connection of 10Mbps, it takes 2-10 seconds to pull an APK file.

Alternatively, a more intelligent attacker first determines whether the APK in use comes from a public source, and if so, avoids the file transfer completely by downloading it from somewhere else. For instance, the attacker can determine whether the current background is one of those issued by Meta,

```
adb logcat | grep "com.oculus.environment.prod"
```

or downloaded from popular online services like [52],

```
adb logcat | grep "com.environment"
```

Finally, to find a list of the apps, the attacker uses the following:

```
adb shell cmd package list packages
```

which returns a list of installed apps on the headset. Knowing the app names, the attacker can locate their button images, from either local app APK files on the headset or online sources like Meta App Store. Also, the attacker can easily access the state information, such as recently used applications, via the ADB command `dumpsys`.

4.4 Replicating the Home Environment

Given the configuration data collected during the bootstrapping phase, the attacker builds a replica of the target’s home environment, packages it as an inception app, and injects it into the headset, again using the remote ADB access. Here the challenge is how to efficiently replicate all crucial, immersive aspects of the home environment at a high precision.

In our implementation, we focus on replicating four key aspects of the home environment: (1) a 3D background with a menu, (2) displaying, calling, and exiting apps, (3) monitoring and reacting to user inputs, and (4) configuring device settings, such as Internet access. Next, we discuss detailed implementation for each.

Replicating Visual Content. Recall that during bootstrapping, the attacker obtains the background APK file, the list of installed apps and their button images, and recent history of app usage, as well as other state information like WiFi and Bluetooth connections. The attacker can replicate the exact 3D background chosen by the target, by unpacking the corresponding APK file. Similarly, the attacker can replicate the application library window, where the app buttons are placed in a grid-like layout (see Figure 3), and add more state information such as history of applications and WiFi/Bluetooth connections. Finally, since the configurations of the cursor, pointer, and menu bar are fixed, they can be replicated with high precision.

Replicating App Calling/Exiting. Upon detecting that the user clicks an app button (discussed below), the inception app will “transition” the user to the inception version of the selected app. This transition is straightforward since the inception app includes these inception versions (see discussion in §4.5).

Furthermore, when exiting an app (e.g. by pressing the “home” button), the inception attack needs to “return” the user to the simulated home environment instead of the actual home. This is done by installing a spy script (together with the inception app) via ADB, which runs in the background to monitor and intercept the exit signal³. Upon detecting this signal, the spy script halts (or kills) subsequent activities and transitions the user to the simulated home environment. The same action is used to activate the inception attack. We will list the detailed implementation in §4.6.

Monitoring and Reacting to User Inputs. Enabling users to engage with virtual objects is vital for any VR experience. Meta has addressed this need by releasing the Interaction SDK [45], allowing developers to seamlessly integrate user interaction into their VR experiences. These include recognizing and reacting to user movements and inputs via controllers and/or hands. Leveraging

³The spy script can detect the “pressing home button” signal sent by the handheld controller, and other exit signals (e.g. “pressing virtual exit button on the app”) that change the system foreground activity.

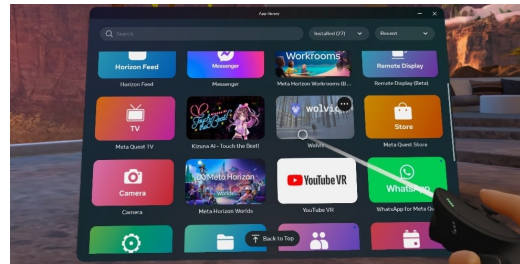


Figure 3: The application library window in VR home. The user is selecting an application called wolvic. Credit: This screenshot is taken inside the Meta Quest Pro VR home.

the SDK, the attacker can easily replicate the interaction function in the home environment.

Enabling System Configuration. Key configuration elements include WiFi/internet access [14], Bluetooth peripheral device access [35], and audio [31]. Access to all of these elements is crucial for achieving full immersion, and therefore, is automatically granted to all applications at runtime without requiring explicit user permission. The inception app can easily replicate the process and result of user configuration.

Removing App Splash Screen. One notable distinction between an app and the native home environment is that apps typically feature a splash screen at launch (e.g. all apps created with Unity include one by default), which is not the case for the home environment. When packaging the inception app, the attacker can easily eliminate the splash screen. They have two options: either acquire a Unity Pro or Enterprise plan (\$2000 per year), which offers the removal feature, or utilize the free Unity Asset Bundle Extractor to adjust the application parameters [26]. We use the latter to successfully remove the splash screen.

Injecting the Inception App. In our current implementation, the final inception app has a file size of around 700MB. The transfer (via WiFi) and installation over ADB take about 1 minute in our experiments. And this process occurs discreetly, without the user’s knowledge. Later in §6, we confirm this observation in our user study – despite conducting the file transfer and app installation within each participant’s session (while they are wearing the headset), they remained oblivious to these actions, exhibiting no signs of suspicion.

4.5 Replicating Applications

After taking control of the home environment, the attacker can choose to hijack the interaction between the target and any given app. To do so, the attacker needs to “replicate” the app in order to either eavesdrop on the target’s actions and/or modify the target’s input into the app (e.g. text, speech, gesture) or the app’s output to the target (e.g. screen content, audio). For each app, the attacker has three replication options.

Option 1: Direct App Call (no inception). For applications that the attacker does not wish to hijack/control, there is no need to replicate the app. The attacker only needs to instruct the simulated home environment to call the original applications (installed on the headset) when the target chooses to open them. This functionality

(i.e., an app sends the user to another app) is already supported by Android as one of its most important features [15] with ready-to-use plugins [4]. Thus the implementation is straightforward. Finally, as discussed in §4.4, when exiting an app, our implementation ensures that the user returns to the simulated home environment.

Option 2: Replicating App GUI. If the attacker only wants to eavesdrop on the target’s credentials on an app, they only need to replicate the login page of the app. After the target inputs their credential, the inception ends the replica and loads the original app to the foreground. Note that if the original app includes a splash screen during loading, this will lead to an extra splash screen that may raise suspicions. We can mitigate this by first displaying an app crashing popup and then loading the original app. Given the common occurrence of Meta Quest apps crashing during loading, this approach helps prevent user suspicions.

Option 3: Full Replication via API Calls. This option aims to realize the full hijacking attack where the user’s interactions with their VR counterparts are eavesdropped, recorded, or modified in real-time by an attacker. One may raise doubts about the feasibility (or cost) of fully replicating a VR app, particularly since attackers generally lack access to the source code. Instead, we demonstrate that replication can be efficiently accomplished by focusing on replicating the app GUI and API calls. This is because many VR applications function by interfacing with servers via APIs, i.e., the application constantly sends API requests to access the web server, fetching data to present on the user’s headset. App developers prefer this approach because it not only streamlines app development, saving time and effort, but also significantly enhances app functionality and compatibility across platforms [23].

In our attack, the app replication method depends on whether the API is public or private. An API is considered public when it is accessible to all users [1], and private if access is restricted to authenticated parties such as app developers but not attackers.

- **Public API** – The replication includes three simple tasks. The attacker first clones the app’s GUI and the standard user interaction, similar to the process of replicating the home environment in §4.4. Next, the replica calls the public APIs to communicate with the legitimate app’s server to fetch data, using the target’s credential obtained via the replicated GUI. After receiving the server data, the replica app integrates them onto the GUI and displays the content on the target’s headset.

Using the replica app, the attacker can now (1) eavesdrop on communications between the target and the server; (2) change the target’s input to the GUI, use the modified input to call the API and thus the server; (3) change the server’s output data (e.g. visual content, audio) and display the modified data on the target’s headset. In this way, the attacker hijacks the entire app session.

- **Private API** – In this case the replication is more complicated due to the lack of API access. Yet we show an efficient replication is feasible when the attacker uses a separate VR headset and a computer, which are connected via USB. Specifically, the app replica on the target’s headset implements the app GUI and user interaction (like the above), from which it obtains the target’s credentials for the app, and forwards them to the attacker’s PC and thus the headset using standard network connections. The attacker’s headset runs

the original app using the target’s credentials to interact with the server. In this case, the attacker’s PC/headset combination acts like a man-in-the-middle between the target’s headset/app and the server, allowing the attacker to manipulate the interaction between the two.

Later in §5, we describe the detailed replications of two apps based on public and private APIs, respectively.

Finally, the attacker packs both the replicated home environment and replicated apps into a single “inception app”. The file size depends on the specific configurations of the home environment and apps being replicated. In our implementation, we find that the replication of the home environment is the dominating factor.

4.6 Entering Inception

Now the inception app is installed on the target headset via (remote) ADB access, the remaining task is to start the inception session. As explained in §4.1, the optimal moment to stealthily transition the user into the inception is upon exiting a 3D app, just before they return to the home environment. In this case, the inception brings them back to the simulated home environment without raising any suspicion.

To do so, the attacker injects a spy script (together with the inception app), which runs continuously in the background. The script monitors the headset activity and user inputs using ADB commands such as `dumpsys` and `getevent`. In particular, the script can detect the signal of app exits using `getevent`, marked by an action of pressing the “home” button on the right controller⁴. Upon detecting such signal, the script proceeds to halt (or kill) subsequent activities (detected via `dumpsys`) and activates the inception app. We list the complete script below.

```
-----  
#!/system/bin/sh  
  
getevent -l | awk '  
  /EV_KEY      KEY_FORWARD      UP/ {  
    system("am force-stop `dumpsys activity |  
      grep top-activity |  
      grep -o 'com.*/' |  
      tr -d '/'"')  
    system("am start -n inception_app")  
  }  
-----
```

5 SPECIFIC APP REPLICATIONS AND ATTACK INSTANCES

In this section, we present two applications that we have replicated in our attack implementation, and use them to demonstrate the feasibility and impact of the inception attacks. The two applications are Meta Quest Browser [33], which uses public APIs to communicate with web servers, and VRChat [63], which uses private APIs.

5.1 Meta Quest Browser

Meta Quest Browser is the built-in browser for Meta Quest headsets, offering users an immersive web browsing experience. With this

⁴Other types of exit signals (e.g. “pressing virtual exit button on the app”) can be detected via monitoring changes in the system foreground activity using a spy script.

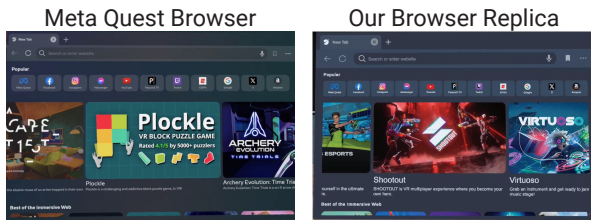


Figure 4: A side by side comparison of the screenshots of the Meta Quest Browser and our replica.

app, users can simultaneously navigate multiple websites through multiple windows, enjoy streaming videos on a captivating theater-sized screen, and engage with social media platforms seamlessly. It also enables web developers to build attractive web content offering immersive VR experiences.

We build a replica of the browser app in Unity [58]. Since the app uses public APIs to communicate with web servers (e.g. HTTP requests to access websites), our replication effort mainly focuses on (1) replicating the browser GUI, (2) displaying the website content, and (3) detecting and responding to user inputs.

For (1) we clone the browser GUI using Unity UI [59]. Specifically, we use a Canvas object as our 2D browser window. The Canvas object defines an area for displaying UI elements. Inside the defined area, the attacker can arrange and display texts, images, buttons, toggles, webpages and more. Unity UI also handles registering user interactions. For instance, each button element has `OnClick` function which allows the attacker to define the effect of clicking this button.

For (2) we leverage the Android WebView library [13, 65]. In particular, the function `loadUrl(the_URL)` fetches the content of a website with address `the_URL`. We then display this content at the corresponding location on the above Canvas object.

For (3), we use the Meta Interaction SDK [45], an API for apps to access user inputs at runtime. Using this SDK, we configure the replica app to monitor and log user’s clicks, drags, and keystrokes on each webpage and their exact locations, and respond to these actions by updating the GUI content and/or sending API calls to web servers to fetch desired contents.

Figure 4 plots a side-by-side comparison of the screenshots of the Meta Quest Browser and our replica. We see that the replica closely resembles the real version. As a reference for the attack cost, we build the replica in less than a day.

Inception Attack Scenarios. Using this browser replica, we demonstrate three attack instances, where the attacker is able to eavesdrop on user login credentials, modify user input to websites, and modify website content displayed on the headset, respectively, all in real time. We use these examples to demonstrate the practicality and simplicity of the attack.

(1) Eavesdropping on Target’s Credentials – When the target uses Meta Quest Browser to access sensitive accounts such as banking, corporate, medical, and emails, the attacker can intercept and log private information, including credentials entered by the target. This is because, using the interaction SDK, the replica app can accurately monitor cursor movements, record keystrokes, capture button presses, and track headset motions. It also has complete

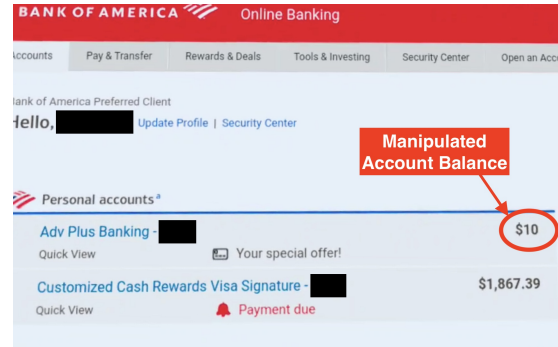


Figure 5: In a banking scenario, the bank server sends the correct banking account balance to the target in VR. However, this balance is altered by an inception attack to \$10 before it gets displayed to the target.

knowledge of the website’s content. Consequently, the attacker can accurately extract user inputs to specific web entries.

(2) Manipulating Server Output (e.g. bank balance) – The replica app can display a modified version of the website content fetched from the server. Consider a typical example of online banking sessions. When the target uses the replica browser to access a bank website, the browser first collects their credentials from the replicated GUI and sends the credentials to the bank server via HTTP requests. After verifying the login credentials, the bank server returns the target’s account information to the headset, including the account balances. While all of these network communications are encrypted using SSL handshakes, the content to be displayed on the headset is encrypted using a key provided by the replica browser during the handshake. Thus the replica browser can decrypt and obtain the raw contents, and can modify them before displaying them on the target’s headset.

Figure 5 shows a screenshot of the target’s headset display, where we altered their Bank of America account balance to \$10. The data alteration is done by executing the following JS code in the replica browser via [65]:

```
document.getElementsByClassName(
  'balanceValue TL_NPI_L1'
)[0].innerHTML = '$10';
```

Here we note that the attacker has complete control over the replica browser, allowing them to execute any arbitrary code at will.

(3) Manipulating Target’s Input (e.g. transaction amount) Similarly, the attacker can also modify the content of user inputs on the replica, and use the modified content to form API calls to web servers. These API calls typically utilize plain text and numerical values within the parameter fields (e.g. HTML format).

Here is an example. Figure 6 demonstrates a scenario where the victim is doing an online transaction via Zelle [71], a digital payment service owned by Bank of America. The victim first initializes a \$1 transaction by filling out the web form and clicking “Continue transfer” (see Figure 6a). Before the web form is submitted to the server via an HTTP request, the attacker altered the transaction amount to \$5 in the web form by the following JS code:

```
document.getElementById('btnModalSave')
  .addEventListener('mouseover', () => {
```

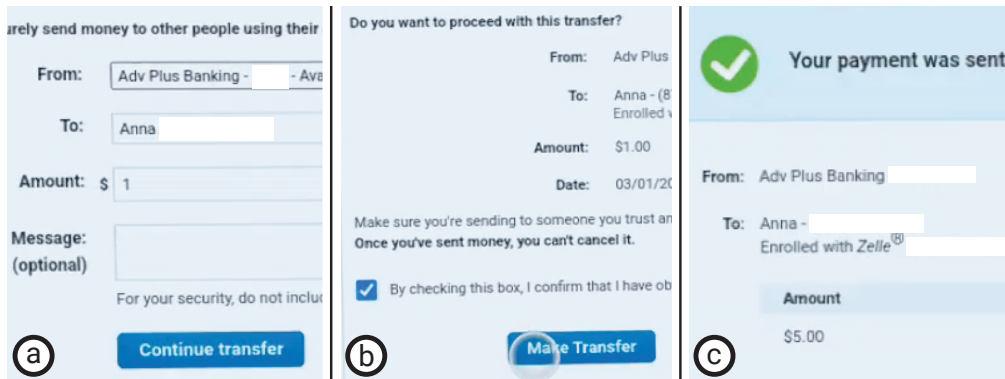



Figure 6: In a transaction scenario, the victim’s amount input is altered by an attacker before submitting to the bank server. (a) The victim initializes a \$1 transaction by filling out the web form. The attacker secretly alters the transaction amount to \$5 in the form before sending it to the server. (b) The victim is then taken to a confirmation page to finish the transaction. The attacker sets the amount value to \$1.00 on the confirmation page to avoid any suspicion from the victim. (c) The actual transaction amount is altered to \$5.

```
document.getElementById('txtAmount').value = '5';
});
```

After submitting the form, the victim is taken to a confirmation page (see Figure 6b) where they need to finish the transaction by clicking “Make transfer”. To prevent the victim from noticing a modified transaction amount has been submitted, the attacker alters the amount value displayed on the confirmation page via the JS code below:

```
document.getElementById('transfer-amount')
.innerHTML = '$1.00';
```

As shown in Figure 6c, the actual transaction made is \$5 according to the final record page. Note that this page can also be modified by the attacker in a practical attack. We leave it this way to demonstrate the success of the attack.

5.2 VRChat

Previously in §1, we described an inception scenario where Alice and Madison’s conversations during VR date nights are hijacked by an attacker Carl. Next, we describe an implementation of this attack on Meta Quest devices by replicating the VRChat app [63], where the attacker can listen to and modify live audio communications between Alice and Madison. An illustration is shown in Figure 7.

VRChat is the leading online social platform for VR users, enabling them to engage with each other through personalized avatars. According to [12], VRChat consistently maintains an active player base of 20 million every month throughout 2023. Within VRChat’s immersive virtual environments, participants engage in conversation, gaming, and romantic encounters like date nights [18]. To safeguard player privacy, VRChat employs a private API for seamless communication between players and servers.

We replicate VRChat with the help of a laptop (Macbook Pro 2019) and an additional Meta Quest headset connected to the laptop via USB (as discussed in §4.5), both under the attacker’s control. We hereby refer to the combo as the attacker server, which acts as a man-in-the-middle between the target and the VRChat server.

Specifically, the replica app on the target’s headset implements the GUI of VRChat, from which it captures the target’s credentials

(for their VRChat account). Furthermore, the replica app captures the target’s live speech and motion, and streams these data to the attacker server. The attacker’s server runs a legitimate VRChat app on its headset using the target’s credentials, which communicates with the VRChat server. The attacker server also captures its VR headset’s screen display and audio stream, and livestreams them to the replica app on the target’s headset, which then displays both on the headset. In this way, the attacker server is a man-in-the-middle between the target’s headset and the VRChat server. More importantly, it has access to the raw visual and audio data and can modify either of them in real time to hijack the VRChat session.

The key tasks in this implementation include (1) cloning the app GUI and user interaction, (2) live streaming data between the target’s headset and the attacker server, and (3) modifying the visual and/or audio data in real-time. For (1), we follow the same process discussed earlier to build the GUI in Unity. For (2) we set up a streaming server on the laptop in Unity, using the FMETP STREAM library [37] designed for live streaming between devices. We also configure the replica app on the target’s headset to communicate with the attacker server using WebSocket via the STREAM library.

For (3), the task of modifying visual or audio data in real time is more challenging. We illustrate an immediate solution for audio editing. Let Alice be the target. In the target-to-server direction, Alice’s headset records her speech and live streams it to the attacker’s laptop. The laptop plays the received speech (or the modified version) next to the attacker’s VR headset, which runs the legitimate VRChat app. As such, Alice is “speaking” to the legitimate VRChat app, with some delay. In the server-to-target direction, the VRChat server sends Madison’s speech to the attacker’s VRChat app. The attacker’s laptop records this speech via USB connection to the attacker’s headset and live streams it (or the modified version) to the target’s headset, so that Alice hears Madison’s speech. In both directions, the attacker’s laptop processes both Alice and Madison’s speech and can replace them with some pre-recorded ones, e.g. replacing “yes” with “no”. We also note that the attacker’s laptop can also employ real-time speech recognition and synthesis tools to produce high-quality speech of any content.

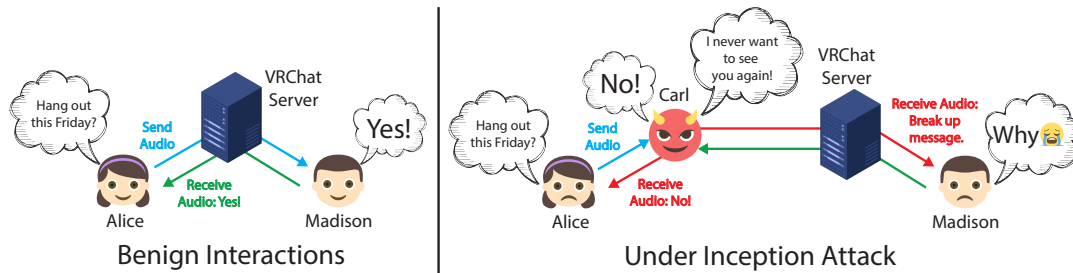


Figure 7: Under the inception attack, the attacker Carl acts as a man-in-the-middle between Alice and Madison’s VRChat session. Consequently, Carl is able to alter the real-time audio transmission. For example, Alice will hear Madison answering “No” to her question while Madison is actually asking “why” in reply to the fake break up message sent by the attacker.

Figure 7 plots the comparison between a benign VRChat session between Alice and Madison, and a version when Alice is under the inception attack. Here the attacker Carl is able to modify Madison’s answer to Alice’s question in real time, using a pre-recorded speech of Madison (e.g. those recorded from prior sessions).

For this proof-of-concept implementation, we also measure the additional delay introduced by the inception attack. The primary factor is the communication between the attack server (Carl) and Alice’s headset across the network, marked by the red line in Figure 7. Comparing to this delay, the delay between the attack server and Carl’s headset, connected via USB, is minimal, as is the processing delay on the attack server. In the setup where the attack server is connected to Alice’s headset via a 5GHz WiFi network, the extra delay is between 0.4 and 0.6 seconds for both directions.

5.3 Discussion

The attack instances (§5.1-5.2) clearly highlight the practicality, simplicity of execution, and the extent of damage caused by the inception attack. With nearly full control over the application process in real time, the inception attacker possesses the capability to execute diverse, arbitrary malicious actions, enabling tracking of user behaviors and manipulation of their sensory input/output and immersive experiences. As a result, the potential attack vectors are virtually limitless.

On-Device vs. Over-Network Control. Our examples also show that, depending on the application being targeted, the inception attack can be controlled by a process on the target’s headset (§5.1) or remotely over the network by a server (§5.2).

The on-device option is constrained by the computational capabilities of the VR headset, which remain notably inferior compared to computers/servers. As such, today’s Meta Quest headsets are unable to run any generative AI tools.

The over-network option is largely limited by network bandwidth, particularly when the attack involves transmitting visual content between the target’s device and the attack server for recognition and modification. This is because, to offer immersiveness, VR headsets demand much higher resolutions and frame rates than current computer or tablet screens. For example, requiring a minimum frame rate of 90 frames per second (fps) is crucial to mitigate motion sickness, a significant increase over the 60fps typically used for computer screens.

6 EVALUATION: USER STUDY

We perform a user study to evaluate the efficacy of our inception attacks in deceiving users. Our focus is to determine whether users can detect differences, if any, between the real VR environments/apps and the simulated ones created by our inception attacks, thereby prompting suspicion and vigilance when they interact with the simulated environments. Our user study was evaluated and approved by our local IRB board.

6.1 Study Setup

Design. To evaluate the efficacy of our inception attacks in a typical everyday setting, we withhold information about the attacks from the participants at the beginning of the study. We inform the participants that the study aims to investigate their experiences when interacting with VR applications. After they explore various VR applications, we debrief the participants with a detailed explanation of the true objectives and conduct the interview. No personal data of the participants is retained post-study.

Participants. We recruited 27 participants from our institution (P1-27), including 5 females, 21 males, and 1 participant who chose not to disclose. The participants range in age from 20 to 30 years old and have varying degrees of familiarity and experience with VR devices. Among them, 3 participants are experts who use VR devices on a daily basis (P1-3); 6 participants are professional users, regularly using VR devices on a weekly basis (P4-9); 10 participants are knowledgeable users who have had several experiences using VR devices (P10-19); and 8 “entry-level” participants have no prior experience (P20-27). Our study lasts roughly 30 minutes. Each participant is compensated with \$20.

Procedure. The study is held inside a large room, and involves one participant at a time. We provide each participant with a Meta Quest Pro headset to immerse themselves in their VR journey. Upon putting on the headset, they find themselves in a legitimate VR home environment that is *custom-made*. By opting for a custom-made home environment rather than Meta’s default home environment, we evaluate our inception design using a more challenging scenario where the attacker lacks any prior knowledge of the target’s home environment. Within the VR home, we guide participants through operations of headset controls and interactions, allowing them to navigate freely until they feel comfortable with the system.

Next, we invite the participant to explore VR applications and rate their experiences. This task is divided into two parts. In Part I, we ask the participant to access the application library from the VR home and choose an application they wish to explore. The library offers a diverse array of 14 applications, covering office, social networking, gaming, entertainment, and web browsing. Examples include Horizon Workrooms, Horizon Worlds, Beat Saber, YouTube VR and Meta Quest Browser. After approximately 5 minutes, we ask the participant to exit the application, which brings them back to the VR home. Here the participant has the option to explore one more app, or go directly to open the Meta Quest Browser app and rate their experiences by completing a Google form. Here the Google form has a question asking for the participant’s institutional ID number⁵. After submitting the form, the participant exits the app and returns to the VR home.

In Part II, we ask the participant to repeat the Part I process by exploring two additional applications before accessing the Meta Quest Browser app to respond to the Google form. What the participant does not know is that, upon exiting the app at the end of Part I, an adversary initiates the inception attack on their headset. Consequently, instead of returning to the authentic VR home, the participant enters the simulated VR home generated by the inception attack as Part II begins. Similarly, the Meta Quest Browser app used in Part II is a malicious replica crafted by the adversary (see §5.1), enabling them to record the participant’s responses to the Google form, including their institutional ID number.

Debriefing Interview. After the above VR session, a member of our research team conducts a debriefing with the participant, disclosing details about the inception attack and its occurrence within the session. Next, the researcher interviews the participant, posing the following two questions:

- (Q1) Did you observe any suspicious or unusual occurrences during the inception attack?
(Q2) Did you experience any hesitation when disclosing your personal information to the Browser app in Part II?

6.2 Results

We start from summarizing the key observations after analyzing participant responses before and after the debriefing.

- Prior to our debriefing, only one out of the 27 participants, who is a VR expert, voiced some suspicion during the inception attack phase. And all 27 participants proceeded without hesitation when prompted to enter their institutional ID in the Google form.
- After disclosing the true intent of the study, 10 out of the 27 participants recalled experiencing something unusual or “off” during the inception phase. These 10 participants include 2 experts, 3 professional, 4 knowledgeable, and 1 entry-level VR user. Interestingly, 9 out of these 10 participants attributed the discrepancies they observed to system glitches, which did not concern them at all. Only one expert (P3) stated that “I would likely investigate the matter if it weren’t for the user study.” Finally, for the remaining 17 participants, the inception attack came as a complete surprise since they did not notice any difference between Part I and II.

⁵The ID is immediately discarded after the current study.

	Exiting Application			
	Beat Saber	Meta Horizons	Bigscreen	Rec Room
No attack	8.10 ± 0.68	7.55 ± 0.74	8.30 ± 0.64	8.10 ± 0.68
Inception	9.41 ± 0.64	8.98 ± 0.73	9.62 ± 0.68	9.69 ± 0.72

Table 1: Measured loading time (seconds) of VR home environment, in terms of mean and std, for both no attack and inception attack scenarios. This small difference only occurs during the initial activation of the inception attack. After that, there is no notable difference between the two.

Next, we present a more detailed analysis of participant responses. We begin with the types of discrepancies they recalled, and then delve into the underlying reasons for the absence of suspicion (Q1) and hesitation (Q2).

Observed Discrepancies. Ten of our participants recalled four types of unusual or “off” details.

- P2, P9, P10, and P26 recalled that a system pop-up was missing during the inception phase. In the normal phase, when pressing the “home” button on the handheld to exit an app, a pop-up will appear and ask the user to confirm the action of exiting the app.
- P3, P7, and P13 recalled that the status of recent applications is not displayed in the VR home during the inception phase⁶. In particular, P3 stated that “[The fact that] it didn’t memorize the previous states that I have [really bothers me].”
- P9 and P14 reported that the controllers are a bit different. P14 noted that “I think the [cursor] beam [of the controllers] is shorter [compared to the legitimate one].”
- P5, P9, and P12 noticed that the loading time of the “VR home” during inception is slightly longer.

We believe that the first three types of discrepancies can be effectively addressed by deploying enhanced attack implementations, where the attacker diligently replicates all crucial aspects of the VR environments with greater precision. For instance, the system pop-up is essentially a 2D app, which the attacker can easily replicate, and the attacker can easily access the state information (i.e., recent applications) via an ADB command: `dumpsys` during bootstrapping.

The loading time discrepancy is caused by the difference in how VR loads the home environment versus initializing an application. Initiating a 3D application requires decompressing assets and loading the 3D scene, which may take extra time. To gain deeper insights, we measure loading times in both normal and inception scenarios, as the duration from the user’s initiation of exiting an application to the appearance of the home environment. Here we vary the application being closed and perform 10 trials per application, while ensuring a consistent 100% battery charge throughout. Table 1 reports the mean and standard deviation of the loading time (in seconds), which vary across trials and applications. The inception attack results in an average increase of only 1.5 seconds. Considering the inherent variability in VR performance over time, such deviations may not trigger suspicion among users. This observation is further supported by the feedback we gathered from user responses (discussed next).

⁶Indeed, the simulated VR home used to conduct this user study did not display the status of recent applications because, at the time, we did not perceive it as necessary or significant.

We note that this time discrepancy only occurs when activating the inception attack, because loading the inception app takes longer than loading the original home environment. After that, the perceived VR home loading time is nearly identical to the no attack scenario (since the app assets and scenes are already loaded).

Reasons for No Suspicion. We study the reasons behind the lack of suspicion raised by our participants during the inception attack phase. In particular, for all 10 participants who noticed unusual patterns, including 2 experts and 3 professionals, none voiced any suspicion except one expert VR user (P3). Our analysis yielded two primary explanations for both highly experienced and less experienced VR users.

- **Accustomed to fluctuating VR operations and glitches.** Four participants (P1, P2, P4, and P5) who are highly experienced VR users stated that they are accustomed to fluctuations and glitches in VR operations, and tend to perceive any unusual patterns as standard system variations or glitches. Thus upon noticing unusual behaviors, they raised no suspicions. Additionally, P1 elaborated that the frequent updates to VR systems contribute to the difficulty in recognizing (ab)normal operations – “*Even if I notice some differences [in the inception], I wouldn’t think they are fake. I will be like [the company] probably did some [system] updates.*”
- **Harmless discrepancies.** Less experienced users may not have a comprehensive grasp of what defines a legitimate VR system, but this does not prevent them from perceiving differences between the inception phase and the normal phase. Interestingly, they do not think that these observed variations as alarming or suspicious in any way. For example, P13 noticed that the status of recent applications is missing but stated that “*I don’t think it was suspicious or anything.*” P14 recalled that the controller cursor is a bit different, but raised no suspicion because “*the interaction still works like when I want to click something, it still works.*”

Reasons for Disclosing Personal Information. All participants confirmed that the malicious copies of the Meta web browser app and the Google form page closely resembled the authentic ones utilized in Part 1. Also, the expert user (P3)’s concern stemmed from observations made while navigating the replicated VR home environment. Next, we inquire about the rationale behind participants’ decision to disclose personal information (i.e., their institutional ID number), which yields two noteworthy reasons.

- **Trusting enterprise app.** Many have developed trust in reputable enterprises and believe that the products offered by these companies are safe to use. In our study, some participants expressed confidence in the safety of using Google Forms on the Meta Quest Browser, considering that both products are supplied by well-established enterprises. For example, “*I mean the browser seems legitimate. It was again the Meta browser and [...] in the particular Google form so it did not seem like it was an attempt to get my data.*” (P24).
- **Repetitive behaviors.** People tend to autopilot with momentums and habits when it comes to repetitive activities. For instance, we usually key-in the password to unlock our smartphones without hesitation. In our study, each participant responded to a Google form twice, first on a legitimate browser and then on a malicious

replica. Several participants mentioned that they thought the second time was the same as before so they just repeated what they did without hesitation. “*I feel like I was going through the same route as before [...] so I do not assume [...] that would steal my identity.*” (P14).

Key Takeaways. The results of our study demonstrate the initial feasibility and effectiveness of our inception attacks, which successfully deceived 26 out of 27 participants. Notably, even highly experienced users, who interact with VR devices on a daily/weekly basis, were susceptible. Our study also reveals that it is challenging for individuals to resist inception attacks. First, the inherent volatility and glitches within today’s VR systems make it highly challenging to detect minor discrepancies or raise suspicions. Furthermore, the trust placed in applications developed by reputable enterprises and the habits formed through prior use of legitimate applications also encourage users to maintain their usual operations during the inception attacks. We need more systematic approaches to defend against such attacks.

7 POTENTIAL DEFENSES

We discuss potential defenses against the inception attack and their efficacy and cost tradeoffs. Similar to traditional MITM networking attacks, we also classify our consideration of potential defenses based on way of implementation and location of solution [6, 10]. We list defenses via prevention (§7.1), attack detection (§7.2), and hardware-based mitigation (§7.3). Ultimately, a combination of these defensive strategies would provide the best coverage of attack surfaces (force an attacker to bypass multiple mechanisms), and give the most robust results (§7.4).

7.1 Defenses via Prevention

Preventing installation. First, we consider defenses that prevent the inception app from being installed on the target headset.

- **Adding secure authentication to networking ports.** Requiring stronger authentication on networking ports would limit the installation of inception attacks through unauthorized remote connections. However, inception app installation is still feasible via two other methods: (temporary) physical access to the headset and AppStore download.
- **Disabling sideloading on headsets.** Inception is conveniently executed through sideloading; without it, the attacker would need alternative ways to inject the inception app, e.g. packaging and publishing it on AppStore inside a benign app. With an Android-based OS, Meta Quest supports sideloading via ADB, which is an essential feature for individuals and enterprises, supporting device configuration, screen recording, app development, testing, access to apps not on AppStore, and business workflows (see §4.2). Considering these use cases and the extensive user base [51], fully disabling sideloading could be very challenging.

To improve security, VR systems may mandate informative tutorials on the security risks associated with sideloading, or reduce the need for enabling sideloading and ADB access by migrating important use cases to more controlled environments. While these approaches are realistic, their protection against inception attacks relies completely on user choices, making them less reliable. In

contrast, Apple's visionOS currently does not support sideloading, but attackers might exploit device-to-device communication features such as the Apple Wireless Direct Link (AWDL) [55].

- **Safe bootloader and secure enclave.** A secure enclave is a processor isolated from the main application processor, storing cryptographic keys and keeping them inaccessible from the rest of the system. During startup, the secure enclave verifies the bootloader, operating system kernel, and privileged processes, thereby protecting the integrity of the entire boot process. Approaches like these would prevent the Inception app from installing as part of the OS or running on the headset. Safe bootloader is quite powerful, since it disables the attack even if the adversary gains physical access to the headset and tries to launch the inception directly. Unfortunately, the capacity of secure enclave is limited, often used to store keys and biometric data, and is far from scaling to the entire OS [21, 22, 49]. The limitation of this approach is in the complexity and high cost of its implementation [2, 30].

Preventing inception launch. If the inception app is successfully installed on the headset, some defensive techniques could help prevent it from executing on the headset.

- **Kiosk mode.** In the enterprise setting, the use of kiosk mode can restrict the set of apps that the user can interact with, which would prevent the inception app from launching. But kiosk mode significantly limits the flexibility of enterprise SaaS and increases operational costs. We also note two additional caveats: 1) some enterprises allow employees to use personal headsets for work, and the defense cannot be implemented in this case; 2) the machine that manages the kiosk mode is not immune to the attack: if compromised, the entire system would fall under control by the attacker.
- **App certificates.** To the VR system, the inception attack app is just like any other custom VR app, without requiring extra verification. Therefore, the validation process needs to be enhanced for all apps. If app certificates are enforced, the attacker would need to bypass the validation to run inception, by getting a certificate or exploiting flaws. For example, Android apps try to increase custom certificate validation logic security and block insecure applications, but research suggests that the system is still vulnerable [42].

Prevent inception attacks from calling other apps. If the attacker already has the inception app running on the headset, defenses may increase the cost of the inception attack by preventing it from replicating individual apps by calling them directly on the local headset.

- **Disabling app calls by non-system apps.** Disabling app transitions would prevent the inception app from opening other apps or their sub-pages. This means the inception attack cannot clone individual apps via direct app call, *i.e.*, option 1 or 2 discussed in §4.5. Yet the attack can always take option 3 in §4.5 to replicate an app, which results in a higher cost. On the other hand, disabling these app calls presents significant practical drawbacks – mobile development ecosystems, including VR, support application calls by other apps because it is crucial for apps to initiate other apps and fork processes, especially for multi-scene apps. Disabling this feature increases development costs and reduces the usability of many legitimate apps, thus making it an undesirable choice.

- **Validating authenticity of app calls.** Strong client authentication is one of the standard ways to prevent MITM attacks. Headsets can add similar validations to authenticate app communications. Nevertheless, certificate validation and source authentication are still not immune to MITM attacks [19, 50] and will not completely defend against the inception attack. Yet it would increase the attack cost, by eliminating the app replication option 1 and 2.
- **Uni-processing for 3D VR apps.** Due to performance limitations, Meta Quest only supports a single 3D environment at a time and stops the previous 3D app when launching a new one. If the inception app chooses to directly call a 3D app (*i.e.*, using app replication option 1 or 2 in §4.5), the inception app itself will stop. However, this has little impact on the attack since the spy script will reactivate the inception app when the current 3D app exits.

Prevent user access to OS shell. The inception attack runs shell scripts to detect when the user exits an app and then activates the inception app, and to collect configuration information of the headset to replicate the home environment at a high precision. Disabling user access to the OS shell blocks an attacker from these scripts, which would reduce the effectiveness and stealthiness of the inception attack. However, doing so also disables legitimate headset users from communicating with the OS, *e.g.* the user can no longer run customized processes, kill processes, or change system settings.

7.2 Defenses via Anomaly Detection

If the attack bypasses prevention methods and the inception app executes on the headset, detection can mitigate harm, for example, by quitting all apps or restarting the device when the attack is detected (either done by the system automatically or by prompting the user to do so). The inception app would be constantly calling other apps, which can leave traces in control flow and performance. We discuss two directions and consider their limitations.

Control flow monitoring. When app calling is supported for non-system apps, it enables the inception app to call other apps to implement low-cost app replication (*i.e.*, option 1 and 2). A detection mechanism can exploit app behaviors, as the inception app potentially raises suspicion with more frequent and diverse app calling than legitimate apps. However, apps have complex and user-driven flows, which poses challenges to benchmarking a benign control flow measurement. Also, as VR integrates more into workflows, app calling between legitimate apps may become more common, making the distinction harder to detect. Hence, detection mechanisms relying on control flow may generate many false alarms.

Performance profiling. Performance statistics like delay and power consumption may increase when the inception app is running. Measuring these metrics may provide information that aids detection. The system could monitor parameters like CPU/GPU usage, memory access patterns, system calls, API calls, etc., to establish a baseline, and alert if performance deviates significantly. However, with the unpredictable nature of user behavior, these metrics are inherently noisy and the detection mechanism is likely inaccurate. Moreover, the attacker's ability to adapt can introduce further challenges, as the inception app could be carefully crafted

to mimic the performance profile of benign apps, making detection more intractable.

Educating users. Users might notice subtle anomalies but dismiss them as bugs or glitches (see our user study in §6). Inception may be detected if users are aware of this form of attack and are alert to minor changes in appearance or experience. However, user detection of cyber attacks tends to be challenging and unreliable in general. Specifically for VR, with users accustomed to imperfect VR systems, this defense is unlikely to be effective.

7.3 Defenses via Hardware

We also consider hardware defenses, which provide an offline layer of security independent of potentially compromised software. We discuss a few strategies as follows.

Regular headset restarts. Restarting the headset will kill the process running the inception app and make the attack dormant until it is launched again. Although it does not remove the inception app, it mitigates potential harm by reducing the active time of the attack. A downside of this defense is significant interruptions on user experience, since users prefer to resume their progress from the last time they took off the headset.

Regular hard resets. Hard-resetting the headset wipes the device and hence completely removes the inception app and spy scripts. The adversary would need to inject the inception app again to attack the user. We note that this is quite an extreme defense, since wiping the device significantly disrupts user experience.

7.4 Combining Defenses

While development of VR hardware and software platforms is still in its early days, it is clear that they are missing many of the security mechanisms we take for granted on traditional network applications. A strong defense against inception attacks ultimately requires a combination of tools for prevention, detection and mitigation. Our discussion above includes a long list of security improvements that can be used to build parts of a multifaceted defense against inception attacks, each of which will require careful consideration of tradeoffs between security and usability/performance.

While each VR platform/vendor will no doubt make their own decisions with respect to security and protective measures, we suggest considering the following five steps.

- Disable sideloading
- Enforce app certificates
- Disable app calls by non-system apps / validate app calls
- Encrypt network traffic
- Regular headset restarts

8 CONCLUSION

In this paper, we introduced and described *inception attacks*, a powerful class of attacks feasible on the most popular virtual reality systems today. We described an implementation that eavesdrops and modifies everything that the user sees or hears, but also everything sent by the user to VR apps. The result is a wide array of personalized misinformation attacks, from misrepresenting a user's bank balances and changing value of financial transactions, to VR chat

apps that modify interactions with other users, so that two parties experience two completely different versions of the same conversation. Finally, results of our user studies validated the potency of these attacks in real world settings.

Looking forward, we believe there is still enough time to design and implement multiple security measures to dramatically reduce both the expected proliferation of these attacks as well as the damage they inflict. But the clock is ticking. Each new generation of VR hardware will bring increasing computational power, which will in turn enable more powerful inception attacks (e.g. an attacker replacing a VR user with seamless, real-time injection of a generative AI version of their voice). VR platforms and developers need to act now to not only improve security on VR systems, but also work to educate their users about the potential security risks they face on these platforms.

REFERENCES

- [1] 2024. Public APIs. <https://github.com/public-apis/public-apis>
- [2] Anjuna. 2023. Secure Enclaves: The Powerful Way to Make Data Secure by Default. <https://www.anjuna.io/resources/what-is-a-secure-enclave>
- [3] Apple. 2024. Vision Pro. <https://www.apple.com/apple-vision-pro/>
- [4] AstricStore. 2024. App Launcher. <https://assetstore.unity.com/packages/tools/integration/app-launcher-20454>
- [5] Harry Baker. 2021. SideQuest for Oculus Quest: Everything You Need To Know. <https://www.uploadvr.com/everything-you-need-to-know-sidequest>
- [6] Bharat Bhushan, G. Sahoo, and Amit Kumar Rai. 2017. Man-in-the-middle attack in wireless and computer networking – A review. In *Proc. of ICACCA*.
- [7] Abraham G. Campbell, Thomas Holz, John A. Cosgrove, Mike Harlick, and Tadhg O'Sullivan. 2019. Uses of Virtual Reality for Communication in Financial Services: A Case Study on Comparing Different Telepresence Interfaces: Virtual Reality Compared to Video Conferencing. *LNNS* (2019).
- [8] Peter Casey, Ibrahim Baggili, and Ananya Yarramreddy. 2021. Immersive Virtual Reality Attacks and the Human Joystick. *IEEE Trans. Dependable Secure Comput.* 18, 2 (2021), 550–562.
- [9] Kaiming Cheng, Jeffery F. Tian, Tadayoshi Kohno, and Franziska Roesner. 2023. Exploring User Reactions and Mental Models Towards Perceptual Manipulation Attacks in Mixed Reality. In *Proc. of USENIX Security*.
- [10] Mauro Conti, Nicola Dragoni, and Viktor Lesyk. 2016. A Survey of Man In The Middle Attacks. *IEEE Commun. Surv. Tutor* 18, 3 (2016), 2027–2051.
- [11] Paulo Costa. 2024. ADB Proxy. <https://github.com/paulo-raca/adb-proxy>
- [12] Player Counter. 2024. VRChat Player Count And Statistics 2023. <https://playercounter.com/vrchat/>
- [13] Android Developers. 2024. Build web apps in WebView. <https://developer.android.com/develop/ui/views/layout/webapps/webview>
- [14] Android Developers. 2024. Connect to the network. <https://developer.android.com/develop/connectivity/network-ops/connecting>
- [15] Android Developers. 2024. Sending the user to another app. <https://developer.android.com/training/basics/intents/sending>
- [16] Brandon Falk, Yan Meng, Yuxia Zhan, and Haojin Zhu. 2021. POSTER: ReAvatar: Virtual Reality De-anonymization Attack Through Correlating Movement Signatures. In *Proc. of CCS*.
- [17] Adeel Faruki et al. 2019. Virtual reality as an adjunct to anesthesia in the operating room. *Trials* 20 (2019), 782.
- [18] flirtual. 2024. The first VR dating app. <https://flirtu.al>
- [19] Martin Georgiev, Subodh Iyengar, Suman Jana, Rishita Anubhai, Dan Boneh, and Vitaly Shmatikov. 2012. The Most Dangerous Code in the World: Validating SSL Certificates in Non-Browser Software. In *Proc. of CCS*.
- [20] Sindhu Reddy Kalathur Gopal, Diksha Shukla, James David Wheelock, and Nitesh Saxena. 2023. Hidden Reality: Caution, Your Hand Gesture Inputs in the Immersive Virtual World are Visible to All!. In *Proc. of USENIX Security*.
- [21] Christian Göttel, Pascal Felber, and Valerio Schiavoni. 2019. Developing Secure Services for IoT with OP-TEE: A First Look at Performance and Usability. In *Proc. of DAIS*.
- [22] Saulius Griškėnas. 2023. Secure enclaves: The key to data security. <https://nordvpn.com/blog/secure-enclave/>
- [23] IBM. 2024. What is an application programming interface (API)? <https://www.ibm.com/topics/api>
- [24] MIT Technology Review Insights. 2023. Augmenting the realities of work. <https://www.technologyreview.com/2023/11/29/1083726/augmenting-the-realities-of-work/>

- [25] Lasse Jensen and Flemming Konradson. 2018. A review of the use of virtual reality head-mounted displays in education and training. *Education and Information Technologies* 23 (07 2018), 1–15.
- [26] kiraio moe. 2024. Remove Unity Splash Screen. <https://github.com/kiraio-moe/remove-unity-splash-screen>
- [27] Katleho Limakatso. 2023. How Virtual and Augmented Reality Are Changing Healthcare. <https://healthnews.com/news/virtual-and-augmented-reality-boom-in-healthcare/>
- [28] Zhen Ling, Zupeli Li, Chen Chen, Junzhou Luo, Wei Yu, and Xinwen Fu. 2019. I Know What You Enter on Gear VR. In *Proc. of IEEE CNS*.
- [29] Shiqing Luo, Xinyu Hu, and Zhisheng Yan. 2022. HoloLogger: Keystroke Inference on Mixed Reality Head Mounted Displays. In *Proc. of IEEE VR*.
- [30] Jâmes Ménétrey, Aeneas Grüter, Peterson Yuhala, Julius Oeftiger, Pascal Felber, Marcelo Pasin, and Valerio Schiavoni. 2024. A Holistic Approach for Trustworthy Distributed Systems with WebAssembly and TEEs. In *Proc. of OPODIS*.
- [31] Meta. 2024. Build Immersive Audio Experiences with Audio SDK. <https://developer.oculus.com/blog/build-immersive-audio-experiences-meta-quest-sdk/>
- [32] Meta. 2024. Expand your world with Meta Quest. <https://www.meta.com/quest/>
- [33] Meta. 2024. Meta Quest Browser. <https://www.meta.com/experiences/1916519981771802/>
- [34] Meta. 2024. Meta Quest for Business. <https://forwork.meta.com/quest/business-subscription/>
- [35] Meta. 2024. Tracked keyboards for Meta Quest. <https://www.meta.com/help/quest/articles/headsets-and-accessories/meta-quest-accessories/tracked-keyboards-meta-quest/>
- [36] Meta. 2024. WHY CREATE IN VR?: Increase the pace of innovation. <https://forwork.meta.com/vr-creativity-design>
- [37] Frozen Mist. 2024. FMETP STREAM. <https://frozenmist.com/docs/apis/fmetp-stream/>
- [38] Monado. 2024. Monado - OpenXR Runtime. <https://monado.dev>
- [39] Gonzalo Munilla Garrido, Vivek Nair, and Dawn Song. 2024. SoK: Data Privacy in Virtual Reality. *Proc. of PETS 2024*, 1 (2024), 21–40.
- [40] Vivek Nair, Gonzalo Munilla Garrido, Dawn Song, and James O'Brien. 2023. Exploring the Privacy Risks of Adversarial VR Game Design. *Proc. of PETS 2023*, 4 (2023), 238–256.
- [41] Nmap. 2024. NMAP.ORG. <https://nmap.org>
- [42] Marten Oltrogge, Nicolas Huaman, Sabrina Amft, Yasemin Acar, Michael Backes, and Sascha Fahl. 2021. Why Eve and Mallory Still Love Android: Revisiting TLS (In)Security in Android Applications. In *Proc. of USENIX Security*.
- [43] OpenHMD. 2024. Welcome to OpenHMD.net. <http://www.openhmd.net>
- [44] Thierry Pul. 2024. How to Easily Sideload a VR App (.apk) to Pico Headsets. <https://headjack.io/knowledge-base/how-to-sideload-a-vr-app-to-pico-headsets/>
- [45] Meta Quest. 2024. Interaction SDK Overview. <https://developer.oculus.com/documentation/unity/unity-isdk-interaction-sdk-overview/>
- [46] Jaziar Radianti, Tim A. Majchrzak, Jennifer Fromm, and Isabell Wohlgenannt. 2020. A systematic review of immersive virtual reality applications for higher education: Design elements, lessons learned, and research agenda. *Computers & Education* 147 (2020), 103778.
- [47] Mohd Sabra, Nisha Vinayaga Sureshkanth, Ari Sharma, Anindya Maiti, and Murtuza Jadliwala. 2023. Exploiting Out-of-band Motion Sensor Data to De-anonymize Virtual Reality Users. *CoRR abs/2301.09041* (2023).
- [48] Joachim Schuster. 2024. Check the computer's RSA key fingerprint. <https://joachimschuster.de/posts/debug-on-device-rsa-fingerprint/>
- [49] Apple Platform Security. 2021. Secure Enclave. <https://support.apple.com/guide/security/secure-enclave-sec59b0b31ff/1/web/1>
- [50] Maliheh Shirvanian and Nitesh Saxena. 2014. Wiretapping via Mimicry: Short Voice Imitation Man-in-the-Middle Attacks on Crypto Phones. In *Proc. of CCS*.
- [51] SideQuest. 2022. SideQuest Turns 3: New Features & 2.2 Million Monthly Active Users. <https://www.uploadvr.com/sidequest-turns-3-sponsored/>
- [52] SideQuest. 2024. Custom Home. <https://sidequestvr.com/apps/customhome/0/rating>
- [53] Bill Siwicki. 2023. What the metaverse and virtual reality can contribute to healthcare. <https://www.healthcareitnews.com/news/what-metaverse-and-virtual-reality-can-contribute-healthcare>
- [54] Carter Slocum, Yicheng Zhang, Nael Abu-Ghazaleh, and Jiasi Chen. 2023. Going through the motions: AR/VR keylogging from user head motions. In *Proc. of USENIX Security*.
- [55] Milan Stute, Sashank Narain, Alex Mariotto, Alexander Heinrich, David Kreitschmann, Guevara Noubir, and Matthias Hollick. 2019. A Billion Open Interfaces for Eve and Mallory: MitM, DoS, and Tracking Attacks on iOS and macOS Through Apple Wireless Direct Link. In *Proc. of USENIX Security*.
- [56] Maki Sugimoto. 2022. *Cloud XR (Extended Reality: Virtual Reality, Augmented Reality, Mixed Reality) and 5G Mobile Communication System for Medical Image-Guided Holographic Surgery and Telemedicine*. 381–387.
- [57] Wen-Jie Tseng, Elise Bonnal, Mark McGill, Mohamed Khamis, Eric Lecolinet, Samuel Huron, and Jan Gugenheimer. 2022. The Dark Side of Perceptual Manipulations in Virtual Reality. In *Proc. of CHI*.
- [58] Unity. 2024. Real-Time Development Platform. <https://unity.com>
- [59] Unity. 2024. Unity UI: Unity User Interface. <https://docs.unity3d.com/Packages/com.unity.ugui@1.0/manual/index.html>
- [60] Verizon. 2023. 2023 Mobile Security Index white paper. (11 2023).
- [61] VIVE. 2024. Installing APK files on the headset. https://www.vive.com/us/support/focus3/category_howto/installing-apk-on-headset.html
- [62] Martin Vondráček, Ibrahim Baggili, and Peter Casey. 2022. Rise of the Metaverse's Immersive Virtual Reality Malware and the Man-in-the-Room Attack & Defenses. *Computers & Security* (09 2022).
- [63] VRChat. 2024. Create, Share, Play. <https://hello.vrchat.com>
- [64] VRVOYAGING. 2022. How to use a custom home environment in VR. <https://www.vrvoyaging.com/how-to-use-a-custom-home-environment-in-vr>
- [65] Vuplex. 2024. 3D WebView: the ultimate cross-platform web browser for Unity. <https://developer.vuplex.com/webview/overview>
- [66] Jialin Wang, Rongkai Shi, Wenxuan Zheng, Weijie Xie, Dominic Kao, and Hai-Ning Liang. 2023. Effect of Frame Rate on User Experience, Performance, and Simulator Sickness in Virtual Reality. *IEEE Trans. Vis. Comput. Graph.* 29, 5 (2023), 2478–2488.
- [67] Wikipedia. 2024. HTC Vive. https://en.wikipedia.org/wiki/HTC_Vive
- [68] Wikipedia. 2024. Oculus Rift. https://en.wikipedia.org/wiki/Oculus_Rift
- [69] Wikipedia. 2024. Sideload. <https://en.wikipedia.org/wiki/Sideload>
- [70] Zhuolin Yang, Zain Sarwar, Iris Hwang, Ronik Bhaskar, Ben Y. Zhao, and Haitao Zheng. 2024. Can Virtual Reality Protect Users from Keystroke Inference Attacks?
- [71] Zelle. 2024. How to send money with Zelle. <https://www.zellepay.com>
- [72] Tianfang Zhang et al. 2023. FaceReader: Unobtrusively Mining Vital Signs and Vital Sign Embedded Sensitive Info via AR/VR Motion Sensors. In *Proc. of CCS*.
- [73] ZIMPERIUM. 2024. Sideload. <https://www.zimperium.com/glossary/sideload/>