Immersive Virtual Reality Attacks and the Human Joystick

Peter Casey, Ibrahim Baggili and Ananya Yarramreddy

Abstract—This is one of the first accounts for the security analysis of consumer immersive Virtual Reality (VR) systems. This work breaks new ground, coins new terms, and constructs proof of concept implementations of attacks related to immersive VR. Our work used the two most widely adopted immersive VR systems, the HTC Vive, and the Oculus Rift. More specifically, we were able to create attacks that can potentially disorient users, turn their Head Mounted Display (HMD) camera on without their knowledge, overlay images in their field of vision, and modify VR environmental factors that force them into hitting physical objects and walls. Finally, we illustrate through a human participant deception study the success of being able to exploit VR systems to control immersed users and move them to a location in physical space without their knowledge. We term this the Human Joystick Attack. We conclude our work with future research directions and ways to enhance the security of these systems.

1 Preamble

Virtual Reality (VR) has found itself into our lives. While the excitement and buzz surrounding VR, and specifically immersive VR continues to increase, it is imperative that the scientific community pays attention to the security of these platforms. While it may be obvious to security researchers that immersive VR needs to be secure, prior work has not examined it in a systemic way while exploring the impact immersion has on manipulating people. Our work does not only coin and hypothesizes potential VR attacks, but also implements them. Furthermore, we illustrate through a human participant deception study that we are indeed capable of moving VR users in a physical space, to a location of our liking, without their knowledge or consent. Our hope is that this primary work catalyzes future research efforts in this domain, especially since immersive VR bears psychological and physiological implications on its users.

2 Introduction

VR is rapidly becoming a household item providing a platform for a wide variety of applications. Currently, VR is highly used in gaming, socialization, rehabilitation and job training, but there many other applications [1]. VR technology has become more accessible at an affordable price point, yet its utility is not fully realized. There is an anticipation that VR will become ubiquitous. Due to improvements in graphics technology, mobile VR and price reduction, VR has solidified its presence in the consumer market [2].

In 2017, the International Data Cooperation estimated that $11.4 billion investments were made in VR equipment [3]. Forecasts for spending over the next four years calls for a 100% increase in spending every year. Two key factors in driving the growth of the VR market are the consumer's desire for immersion and the novelty of 3D user interfaces [2]. Small scale portable VR implementations such as the Google Cardboard offers an affordable VR environment to compatible smart phones. A simple *cardboard* headset can be purchased for about $15 to house a user's phone. Acting as the VR system, the phone using Google's software will render two stereoscopic views allowing portability and availability. Other options for mobile VR include the Google Daydream and Samsung's GearVR, which is the most popular at 4.51 million units sold in 2016 [4]. Fully immersive systems which include features such as motion controllers and Infrared tracking are also becoming increasingly popular. Among the most common are the Playstation VR (PSVR), Oculus Rift, and the HTC Vive. The fully immersive nature of these systems provides a unique experience. The common thing amongst all VR experience is their ability to create Virtual Environments (VEs).

VEs attempt to create a replica of an object or experience in a way for a human to understand [5]. Fully immersing someone into VR has shown to influence their behaviors and invoke sensations experienced in reality. These sensations are in part a result of *presence*, defined as a participant feeling as though they are "physically present within the computer-mediated environment" [6]. For example, in a study by [7], participants were presented with a small ledge and a simulated cliff in VR, eliciting a fear of heights and demonstrating the participants' presence. The current generation of VR systems completely encapsulate the user's vision. The complete removal of real-world distractions further increases the user's presence [8]. The perception of presence and other familiar sensations may create a false sense of security and lend to the user being susceptible to harm.

A VE allows the developer to fully dictate the encounters a user will experience. With content becoming more realistic, psychologists have found use for immersion therapy [5]. The ability to illicit responses from subjects in VEs allows psychologists to treat phobias. The first phobia exposure treatment progressed from displaying images of spiders to grabbing a virtual spider and its toy counterpart [9]. Again, the sense of presence allowed researchers to evoke fear. Other applications such as entertainment and training can have similar effects where users *forget* they are in a VE. Being that content creators can induce such strong responses and VR systems are becoming commonplace, an evaluation of the security of VR systems is prudent.

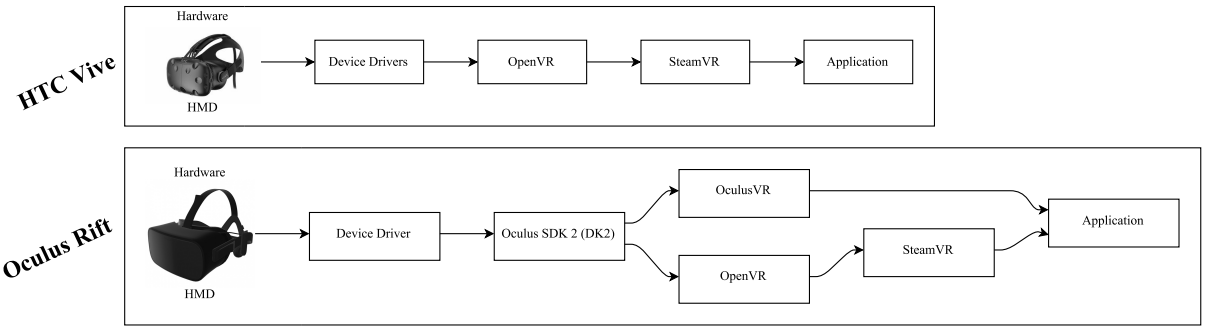

Fig. 1: Hardware and Software Abstraction Layers of the HTC Vive and Oculus Rift

The Internet in its infancy was much like VR, in that the excitement of such a new technology can cause some potential downfalls to be overlooked. Simply achieving a working product is a challenge in its own right. The creators of the Internet could not predict how it could be used maliciously [10]. As VR systems become ubiquitous, they will face the same scrutiny and malicious users will exploit them. For example, a recent study by [11] identified that children immersed in VR for 20 minutes showed deteriorated balance immediately after. We contend that VR players can be manipulated and may be physically compromised making VR users an assailable population. Although, research has been conducted regarding security and privacy of Mixed Reality (MR) systems [12], we found no applied work specific to vulnerabilities presented by room-scale immersive VR. Therefore, we deem it necessary to explore novel attack vectors in VR.

Our work contributes the following:

- We coin the following attacks: *Immersive Attack*, *Chaperone Attack*, *Overlay Attack*, *Disorientation Attack*, *Human Joystick Attack* and finally *The Man-In-The-Room Attack*.
- We implement proof of concept *Immersive Attacks* using OpenVR against the HTC Vive and Oculus Rift and discuss ways to mitigate them.
- We successfully implement the *Human Joystick Attack* and provide evidence of participant manipulation through a human participant deception study.
- We catalyze a research agenda for the security of VR systems.
- We provide the source code developed upon request and the vetting of the requesting party.

The rest of the paper is organized as follows. First, we share background information and related work in Section 3. The devices and software used in our work are discussed in Section 4 followed by attack implementations in Section 5. Results of testing the Chaperone, Disorientation, Overlay, and Camera Attacks testing are shared in Section 6. In Section 7, we share a human subject experiment testing the *Human Joystick Attack* and discuss our findings and attack mitigations in Section 8. We discuss new and similarly novel attacks in Section 9. Finally, future work is identified and concluding remarks are made in Sections 10 and 11 respectively.

3 Background Information

3.1 Virtual Reality Components

Due to the characteristics of full immersion discussed in Section 2, our work focused on immersive VR systems; the HTC Vive and the Oculus Rift. These systems provide a realistic virtual experience through a sophisticated tracking system allowing user movements to be replicated in the VE in real-time. The Oculus Rift Head Mounted Device (HMD) has a series of Light Emitting Diodes (LED)s that are tracked by a small camera, known as the Constellation Tracking System [13]. This is supplemented by the Adjacent Reality Tracker (ART), which features an Inertial Measurement Unit (IMU), magnetometer, accelerometer, and gyroscope [14]. The HTC Vive reverses these roles by placing the tracking sensors on the headset and emitting an Infrared (IR) beam from the base station or *lighthouses*. The lighthouses begin the tracking cycle with a synchronizing pulse, followed by two perpendicular sweeps of IR. The sensors measure the time between the pulse and the sweeps to determine their angle to the lighthouse [15]. The Vive also incorporates an IMU to fill the gaps in tracking due to obstruction.

The improvement in tracking technology has brought forth room-scale VR experiences. Room-scale adds another dimension to the experience allowing VR users to walk about freely in a play area. Being that the player's vision is entirely encapsulated by the HMD, there must be a safeguard to protect the player from obstacles (walls). Both systems implement similar solutions involving the user tracing the boundary of the room prior to playing using a controller. The Vive collision avoidance safeguard is referred to as the Chaperone and is the target of our proof of concept attack, while the Oculus Rift uses the term *Guardian*. One can think of these as *VR fake translucent walls* that appear during a VR experience when a user approaches a real wall.

3.2 OpenVR - The Attack Layer

OpenVR was developed by Valve Software for SteamVR devices with the intent to allow developers to design applications without having to rely on vendor specific Software Development Kits (SDK). This is accomplished by providing virtual functions in which the compatible SDK can be matched to the initialized system [16]. Figure 1 shows how OpenVR acts as a wrapper for the Oculus Rift to allow compatibility for developers between VR systems. Although distribution is currently limited to the HTC Vive and Oculus

TABLE 1: Virtual Reality Devices

HTC Vive				Oculus Rift			
Device	**VID**	**PID**	**Firmware**	**Device**	**VID**	**PID**	**Firmware**
Hub	N/A	N/A	N/A	Rift	2833	0031	708/b1ae4f61ae
Hub Controller	0424	274D	N/A	Rift Audio	2833	0330	708/b1ae4f61ae
Bluetooth	N/A	N/A	211	Sensor x3	2833	0211	178/e9c7e04064ed1bd7a089
Watchman Board	28DE	2000	1462663157	Left Touch			f3c65f7a5f
Camera	0BB4	2C87	8590262295	Right Touch			f3c65f7a5f
Audio Device	0D8C	0012	3				
Main Board	0BB4	2C87	1.6				
Wireless Receiver 1	28DE	2101	C638F6E4EF				
Wireless Receiver 2	28DE	2101	90538B7D13				
Base Stations			436				

Rift, at the time of writing, Microsoft had also announced their VR headsets will be Steam compatible [17], meaning they will likely be adapted for OpenVR, since Steam is based on OpenVR. Vulnerabilities found in OpenVR are likely to affect all compatible devices. The attacks described in our work are implemented on the software interface level; OpenVR. Although we suspect there are other security weaknesses at the hardware layer (Section 10) and specific applications, targeting OpenVR offers a broad attack surface.

SteamVR is an application that manages the VR hardware and provides an interface to launch applications. The HTC Vive relies largely on the services that SteamVR provides such as the Chaperone. Steam provides compatibility for the Oculus Rift by acting as a wrapper to the Oculus Rift Manager. This provides redundancy of services in the case of collision avoidance. The services provided specifically by the Oculus Rift Manager were not targeted in our proof of concept attacks, yet the function of the Guardian and Oculus Manager are similar enough that they would require an equal security evaluation. In our work, all references to the Chaperone are specifically regarding the SteamVR generated collision avoidance system.

4 Apparatus

This section outlines the system used in our testing in Table 2 and the VR devices used in Table 1. It is important to note that these are off-the-shelf devices, and even though the granular details are presented, all that was needed to conduct this work was a gaming computer, the HTC Vive, and the Oculus Rift. Of course, we also constructed our own tools (see Section 5.3).

TABLE 2: System Details

Device	**Details**
Processor	Intel Core i7-6700 CPU
System Type:	64-bit OS, x64 based processor
Graphics Card	NVDIA GeForce GTx 1070
Manufacturer	iBUYPOWER
Installed Memory (RAM)	8.00 GB

Application	**Version**
Steam	1508273419
SteamVR	1507941678
Oculus	1.19.0.456194

5 Discovery and Attack Implementation

The security analysis of the VR systems was performed with several main objectives. Our study sought to answer the following questions:

- *Can a VR user's safety be compromised?*
- *Can a VR user be disoriented?*
- *Can a VR user be manipulated?*

To gain a preliminary understanding of the systems, we first conducted forensic artifact collection. To identify artifacts of potential value, a system report was generated using SteamVR. This yielded the locations of the boundary data, default and current system settings, executable path location, among many other features. Stored in plain-text, with no integrity checks in place, this was immediately deemed a vulnerability. Manual inspection of Steam, SteamVR and application artifacts was conducted for completeness.

Familiarization of OpenVR was accomplished by examining the API documentation. Small scale development efforts lead to the discovery of lack of access control to VR resources. Unvetted applications were found to be able to access and modify the VE. The inability to regulate and authenticate accesses to the VR system was also deemed a vulnerability.

5.1 Adversial Model

During the research and discovery phase we adopted a strong adversarial model; placing no limitations on an adversary's capabilities. This was done to facilitate a robust security analysis. Because our scope is limited to the VR systems and their applications, we assumed the adversary has compromised the target machine in some fashion, allowing for user-land post-exploitation. On the other hand, the adversarial goals were guided by hypotheses, specifically related to Immersive VR. Aligned with our hypotheses and indicative of a successful attack, we impose several other secondary adversarial objectives:

- The target VR user would have no immediate indication that their privacy is being compromised.
- The attack does not interrupt ongoing VR applications or indicate that malicious activities were occurring, *unless that was the objective of the attack.*
- No action would be necessary from the VR user for the attack to take place.

Our findings and requirements for attack implementation (following sections) allow for the adversarial requirements to be further refined. Attacks which modify the VE require the ability to initiate instances of OpenVR applications, or meaningful write privileges on VR system configuration files. Configuration file availability alone is not a requirement but decreases attack sophistication and offers several other worthwhile points of control. The ability to affect an ongoing user experience also requires the ability to spawn OpenVR instances, whereas solely modifying configuration files would preclude real-time attacks. Furthermore, attacks which affect the VR scene or request information from the run-time also require OpenVR instances. Generally speaking, the adversary requires the capability to modify the safety boundary or room-scale configuration.

5.2 Classification of Immersive Attacks

We define an **Immersive Attack** to be any attack that targets the unique properties of Immersive VR and associated immersed user vulnerabilities. An Immersive attack results in a VE that has been maliciously modified, as to cause physical or mental harm and/or disrupt the user. We classify Immersive attacks by the attack's outcome. We coin below new terms for specific implementations of Immersive attacks:

- **Chaperone Attack:** Any attack that modifies the VE boundaries (VR walls). This could be used to make a virtual space appear smaller or larger to an immersed user or may be used to prevent the *Chaperone* from helping users identify their real world boundaries (real walls) during an Immersive session.
- **Disorientation Attack:** Any attack that is used to elicit a sense of dizziness and confusion from an immersed VR user.
- **Human Joystick Attack:** Any attack that is used to control an immersed user's *physical* movement to a predefined physical location without the user's knowledge. In our work, we implemented this by manipulating a user's Virtual Environment.
- **Overlay Attack:** Any attack that *overlays* unwanted images / video / content on a player's VR view. The player will have no option to remove the content. This attack includes persistent images as well as content that remains fixed in virtual space.

5.3 Tools

Manipulation of the Steam configuration files were essential for the *Chaperone* and *Disorientation* Attacks. For the purpose of parsing and modifying the configuration files, we utilized Python 3.6, specifically the *JSON* module.

To replicate the scenario and demonstrate that a remote hacker can implement attacks, we incorporated a reverse shell capable of remotely executing attack payloads. To collect images captured from the device's front facing HMD from the HTC Vive, we utilized a User Datagram Protocol (UDP) stream constructed using the Simple and Fast Multimedia Library (SFML) API [18].

5.4 Scenario

To provide some context to the reader, we preface the attack implementation with a generic scenario (Figure 2) in which these threats may be employed. Our assumption regarding payload delivery relies on an initial compromise of the target machine and the presence of a VR system with the appropriate OpenVR drivers (default provided by SteamVR and Oculus). Any number of exploits or social engineering tactics that could result in user land payload execution would suffice for this stage. As per Section 5.3, we developed a command and control (C2) listener specifically for these attacks, however any post exploitation tool, such as meterpreter, will work. Once executed our C2 is used to manipulate the necessary configuration files and initiate instances of OpenVR. These minimal applications will forward the attackers command to the VR runtime. Because our OpenVR applications are independent, the ongoing VR application can continue to execute uninterrupted. Finally, any extracted information is relayed back to the attacker.

5.5 Chaperone Attack

Through manual system examination, we discovered a JSON file that stores the details of the room setup, Table 3, 1-2. Each room setup or *universe* contained the 3D geometric data representing the boundaries of the room. We created a tool to parse and modify the boundaries in the *Chaperone* artifact. This artifact portrays a security weakness where critical safety feature data (physical wall boundaries) is stored in cleartext, unencrypted, and is easily accessible.

We begin the attack by modifying the JSON configuration file. The HTC Vive and the Oculus Rift create separate directories containing the artifacts. If the file manipulations happen while the user is immersed into a VR environment, this modification will have no effect until the VR system is restarted. To combat this challenge, we initialize an OpenVR instance as a background application (Listing 1, line 1). This application type does not have access to the renderer and will not start SteamVR but can send commands to the Steam processes. We call for Steam to reload the configuration file from the disk (Listing 1, line 2). This loads the data into our process's working copy of the Chaperone. We then commit our copy to the system, as the active Chaperone (Listing 1, line 3). This change occurs seamlessly during VR immersion. If a VR user was not conscious of the Chaperone, this would likely go unnoticed and does not affect rendering. Physical harm could result from an immersed user's confidence in bounds that are no longer in effect.

The attack was implemented due to the ease of access to the Chaperone data and its simplicity, however, it should be noted that the Chaperone may be modified entirely in OpenVR. The function `SetWorkingCollisionBoundsInfo` can be used to modify the boundaries of the working copy of the Chaperone. Calling `CommitWorkingCopy` will then apply the changes.

If an attacker's intent is to simply hide the collision boundaries the opacity attribute can be set to transparent or `forceVisible` to `false`.

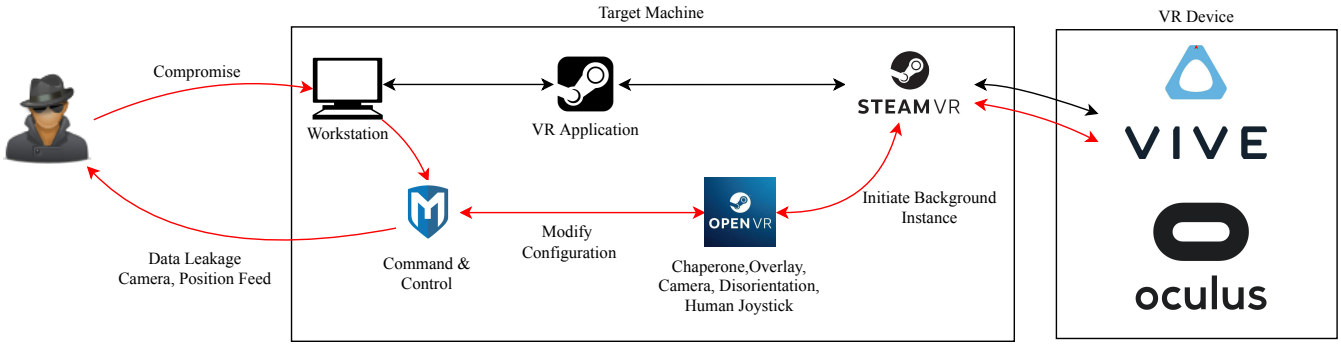

Fig. 2: A potential scenario for Immersive attacks.

TABLE 3: Files Modified in Attacks

	File Path	Contents
1	\Steam\config\chaperone_info.vrchap	Vive Chaperone Coordinates, Translation, Yaw, Time, Play Area, UniverseID
2	\Steam\config\oculus\driver_oculus.vrchap	Rift Chaperone Coordinates, Translation, Yaw, Time, Play Area, UniverseID
3	\Steam\config\steamvr.vrsetting	Camera Enabled, Chaperone Opacity/Color, Lighthouse Bluetooth, etc

Listing 1 : Chaperone Attack

```
1 m_pHMD = vr::VR_Init(&eError, vr::VRApplication_Background);
2 vr::VRChaperoneSetup()->ReloadFromDisk(vr::EChaperoneConfigFile_Live);
3 vr::VRChaperoneSetup()->CommitWorkingCopy(vr::EChaperoneConfigFile_Live);
```

5.6 Disorientation Attack

The *Disorientation Attack* is similar to the Chaperone Attack in that the overall procedure is the same. This attack adjusts the VR immersed user's location and rotation in the virtual play-space. When users are subject to visual motion cues in the absence of physical motion, Visually Induced Motion Sickness (VIMS) can occur [19]. By applying the translation and rotation vigorously and randomly we replicated a sea-sick sensation.

Utilizing our Python JSON parsing script we modified the *Chaperone* configuration file (Table 3, 1-2). Two *universe* traits control the players orientation: yaw and translation. Adjusting the translation will cause lateral movements while yaw will affect the direction the user faces. Making small adjustments and calling for the *Chaperone* reload we can take control of the player's orientation.

Being that our implemented attack employs the same configuration files as the Chaperone Attack, the extent of our success for the Oculus Rift was similar to the Chaperone Attack in the HTC Vive. When the Oculus Rift launched from Steam, we were able to apply the translations and rotations.

We discuss driver level attacks in Section 10, however, we note that through OpenVR custom drivers it is possible to map the tracking solution from one device to another [20], [21]. For example, rapid and disorienting motions can be caused by replacing the tracking solution of the HMD with the tracking solution of the controller. This implementation of the attack reduces latency by using the systems tracking against itself and resulting in a more fluid disruption.

5.7 Human Joystick Attack

Our implementation of the *Human Joystick Attack* is similar to the Disorientation Attack in all aspects except for the rate and control at which it is conducted. However, the effect the attack has on the immersed VR user is distinct enough to warrant separate classification. The attack attempts to lead the player to an attacker defined direction and location by compounding imperceivable VE translation. The gradual shift in the VE aims to cause the player to readjust their location to the new virtual center point.

With the intent to guide and control the VR user's movements without their knowledge, we must first remove any possible indication of foul play. Continuous translation in any direction will cause the player to pass over the Chaperone, thus in most cases, we begin the attack by disabling or expanding the Chaperone. As we will discuss in Section 5.9, it is possible for the attacker to access both the screen data and the front facing camera. This could inform the attacker of necessary information to make a decision regarding the direction they intend to move the player and the optimal time to apply the translation.

We then apply the translation towards the desired direction incrementally in an amount small enough to be imperceptible to the immersed VR user. This is repeated until the player reaches an obstacle or the desired physical endpoint. In some instances, it is necessary to utilize a change in rotation, however, this is application (VR experience) dependent. Since all other attacks are technical in nature, we ran a human participant deception study, to examine the effectiveness of our implemented Human Joystick Attack, which is discussed in Section 7.

Listing 2 : Overlay Attack

```
1 m_pHMD = vr::VR_Init(&eError, vr::VRApplication_Overlay);
2 vr::VROverlay()->CreateOverlay(oKey.c_str(), "Overlay", &oHandle);
3 vr::HmdMatrix34_t current_pos;
4 vr::VRChaperoneSetup()->GetWorkingStandingZeroPoseToRawTrackingPose(&current_pose);
5 current_pos.m[2][3] = -1;
6 vr::VROverlay()->SetOverlayTransformTrackedDeviceRelative(oHandle,
7       vr::k_unTrackedDeviceIndex_Hmd, &currrent_pos);
8 vr::VROverlay()->SetOverlayFromFile(oHandle, img_path);
9 vr::VROverlay()->ShowOverlay(oHandle);
```

5.8 Overlay Attack

An overlay is a two-dimensional image that is projected onto the rendered screen [22]. This feature is intended to supplement an Immersive VR scene and will not cause an interrupt. This allows overlays to be used regardless of the running VR application. Unlimited overlays can be created, however, only one high definition overlay can be rendered. Overlays are typically used for application menus, information display, and the dashboard.

An overlay, like any virtual object, can be absolutely or relatively positioned. The overlay can be transformed to match the location of any tracked device. By linking the translation of the overlay to the HMD or VR controllers, an attacker can ensure the malicious images will be persistently visible.

There are no built-in VR application features that allow users to force close an overlay. This combined with the ability to call an overlay from any application makes this particularly prone to misuse. Once the attack has been executed, the only way for the player to close the overlay is to restart the application.

This feature is provided by Valve Software as a convenient means to render images, however, aspects of overlays lend to potentially malicious behavior. Reference code for this attack is provided in Listing 2. We begin the attack by initializing the type to $VRApplication\_Overlay$ (line 1). This will cause SteamVR to boot if not already running. We then create the overlay and provide a key and handle for referencing (line 2). The default overlay visibility is set to hidden. This attack will display an image directly in front of the user and encapsulate the majority of the VR view. To accomplish this, we first capture the origin of the play space in line 4. We then apply a translation to move the image 1 meter in front of the player's field of view (line 5). We then link our transformation with the tracking of the HMD (line 6). Finally, we load our image and set the overlay to visible (lines 8-9).

Strategically replacing this attacks playload with a commonly used VR application could be a means to deliver advertisements or completely blocking game play as a form of ransomware. We were able to accomplish this by replacing the path for a SteamVR tool with our payload. *We ask the reader to consider the psychological effects of overlaying disturbing images by an attacker during Immersive experiences*.

5.9 Camera Stream and Tracking Exfiltration

Although not unique to immersion (thus not included in Section 5.2), the following attack further exploits the permissive nature of the VR system. With the advent of inside-out tracking, cameras may be a requirement for room-scale immersion [23]. Furthermore, a wide variety of information can be extracted from the system to provide an attacker with information not visible to the camera. This, in conjunction with the resulting high impact we include the following proof-of-concept attack.

The HTC Vive is unique from the Oculus Rift in that the HMD has a front facing camera. This presents another attack surface. Note that SteamVR does offer support to enable and disable the camera. Disabling this feature will prevent the camera from initializing and providing a video stream. *This was the only attempt to restrict permissions to services we observed*.

Similar to the Chaperone configuration file, SteamVR stores configuration settings in an unencrypted JSON file. We modify the file containing general settings (Table 3, 3), adding the attribute `camera: {enableCamera: True}`. Conversely, the absence of the attribute disables the camera. For the change to take effect, SteamVR must be rebooted. OpenVR provides the functionality to shut the system down in the function `vr::VR_Shutdown()`. Finally the system can be rebooted by temporarily initializing the system in a mode other than background, such as `vr::VRApplication_Overlay` or `vr::VRApplication_Scene`. This allows an attacker to access the camera regardless of the state the system is in.

Accessing the camera does not produce a rendered image nor require a specific scene, therefore we can initialize the attack as an OpenVR background process (Listing 3, line 1). Again, this allows the process to influence the VR system without interrupting the current scene application. Being that there is no indication that the camera is powered on, the user would otherwise be unaware of the attack. We request access to the video stream which will activate the camera (Listing 3, line 4). Note, if the attack is the first client to call for video services, the first few frames may be delayed due to camera spin-up and auto exposure [24].

Calling `GetvideoStreamBuffer` copies the frame and header into the specified buffer (Listing 3, line 5). We recommend receiving the header prior to the frame to ensure an appropriate amount of memory has been allocated.

Utilizing the UDP stream discussed in Section 5.3, we were able to export the camera's video stream. OpenVR also has the support to capture many other data points. In the same manner we can export the tracking solutions, providing further incite to the target's physical behavior or environment. Reconstructing the exfiltrated tracking information, we are able to monitor the user's movements in real time.

Listing 3 : Camera Attack

```
1  vr::IVRSystem *m_pHMD = NULL;
2  vr::IVRTrackedCamera *camVr;
3  m_pHMD = vr::VR_Init(&eError, vr::VRApplication_Background);
4  camVr->AcquireVideoStreamingService(vr::k_unTrackedDeviceIndex_Hmd, &m_hCamera);
5  camVr->GetVideoStreamFrameBuffer(m_hCamera, vr::VRTrackedCameraFrameType_Undistorted,
6        m_pCameraFrameBuffer, m_nCameraFrameBufferSize, &frameHeader, sizeof(frameHeader));
```

6 Testing and Results

In this section we discuss our findings for each attack on both the HTC Vive and the Oculus Rift.

6.1 Chaperone Attack Results

Our proof of concept attack was successful against all tested OpenVR and SteamVR applications for the HTC Vive. As SteamVR is the sole manager of the collision bounds for the HTC Vive, targeting the Chaperone configuration file affected all applications. We found the method of modifying the artifact had the advantage of reliability over the faster, Chaperone working-copy method accomplished entirely in OpenVR. This is due to the working-copy failing to commit depending on the state of the HMD. Additionally, rapid modifications caused the Chaperone to enter an erroneous state, preventing further Chaperone commits. We found that if the system was idle or the proximity sensor did not detect the headset being worn, the Chaperone would also be inactive. In contrast by modifying the configuration file, our changes would remain, and the next time the Chaperone is loaded our changes would take effect.

As discussed in Section 3.1, when launched via Steam, the Oculus Rift will inherit both the Guardian and the Chaperone. Our implementation will influence only the Steam generated Chaperone. Upon startup, SteamVR will load the Guardian boundary information and create a JSON file containing the *universe*, similar to the SteamVR generated room setup. Being that our attack targets the Steam generated file, this implementation will not alter the *Guardian*. We expect that the Oculus Rift produces an artifact containing this information and could be modified in a similar manner.

We found that expanding the boundaries beyond what the player was capable of reaching served as a better method of hiding the Chaperone. In some cases, reducing the Chaperone's opacity did not affect all safety features. If enabled, the front-facing camera would continue to activate as the player approached an obstacle. Additionally, some Chaperone modes would continue to display the outline on the floor, a parameter which can also be modified.

6.2 Disorientation Attack and Human Joystick Attack Results

The technical success rate of these attacks were identical to that of the Chaperone Attack being that we targeted the same artifacts. We found that in both systems, the VR user's orientation could be manipulated. By targeting the Steam artifacts attack success on the HTC Vive was expected, however OculusVR maintains an independent room configuration and orientation. When the Rift is initialized in Steam, the rooms orientation is retrieved from Oculus and stored in the Chaperone configuration file (Table 3, 3). This file is then referred to by all applications launched through SteamVR. Changes to this artifact will reflect upon all Steam applications but will not be inherited by the Oculus room configuration. The ability to change the user's location and orientation from SteamVR generated configuration file suggests that Oculus entirely relinquishes its room-scale configuration. OpenVR is likely utilizing functions within the Oculus SDK, suggesting these attacks may also be conducted solely through the Oculus SDK. Despite being a *closed* system, the Oculus Rift still allows third party managers access to some of its features.

We found the effectiveness of the Disorientation attack to be related to the speed and magnitude of the artificial motion. Surprisingly we observed that slower fluid sine waves produced the sea-sick sensation and degraded balance more rapidly than breakneck fluctuations. The fluidity of the attack was improved through smaller and more frequent updates, paired with interprocess communication synchronizing manipulation and commits. There may be a point of magnitude at which the user will reject any perceived artificial motion. Further testing is required to validate this relationship.

6.3 Overlay Attack Results

In all applications tested, we were successful at populating an image on the screen. Additionally, we identified no means to remove the image while the user is immersed. Hiding the payload process from the victim will further conceal the responsible program [25]. This attack was also successful against the Oculus Rift, however, OculusVR does check for third-party applications. By default, the Oculus Rift Manager will not allow applications that did not originate from the Oculus store to launch. This permission must be granted to utilize Steam software and its applications. Given that some content is only found on Steam or from other developers, many users will have allowed this feature. Once the payload application is executed it will appear in the victim's Oculus library reducing transparency. Further development and testing are required to avoid this user induced permission-based model. In contrast, the Chaperone, Disorientation, and Human Joystick attacks were not flagged, as they did not produce a rendered product.

6.4 Camera Attack Results

At the time of writing, the Oculus Rift did not have a front facing camera thus, our results are solely for the HTC Vive. In all tested VR applications, we were able to initialize and extract the video stream buffer. In fact, an active scene was not necessary. The state of the VR system was a factor, however, and in all instances, we were able to produce

conditions to access the camera. Without knowledge of the state of the system, the two courses of action described in Section 5.9 alleviated a handle request for the camera returning `NULL`. The first, indicated by a failed request for he HMD is remedied by powering on the system. To ensure the system is active without interrupting a current application we can initialize a short-lived application as an overlay. As per our assumptions, this will ensure that if the system is in use there will be no effect. Second, the permission associated with the camera being enabled via the configuration file granted the needed access. The only instance in which the player would be disrupted is the case where the system is in use and the camera is disabled. Inducing a system restart may raise an alarm to the victim.

If further information is required by the attacker as to the type of hardware available, OpenVR provides this information using the `IVRSettings` class [24].

7 Human Joystick Attack Experiment

To determine if the Human Joystick Attack is capable of manipulating an immersed user's location without their knowledge, we designed and conducted a deception study utilizing the HTC Vive. In the experiment, users played immersive arcade style games in VR while the Human Joystick Attack attempted to navigate the player to a predetermined location. The attack's success is measured by the player reaching a physical destination created by the attacker and the immersed VR user reporting no awareness to the attack or unusual movement. We hypothesized that immersed players will follow the VE.

7.1 Methodology

The Human Joystick deception study was conducted in the following phases:

- **Experimental Design:** We designed the experiment as well as the post-immersion survey instrument. The experiment was a deception study, as participants were recruited under the premise that they would be playing VR games. Participants were not told that we would manipulate their VE to influence their physical movements until they were debriefed at the end of each experimental run.
- **Tool Creation:** In this phase we developed software to track and visualize the participant's location and the translation of the VE.
- **IRB Approval:** We submitted an IRB request and received approval for the deception study.
- **Participant Recruitment:** We recruited participants by sending e-mails, posting posters, and finally by e-mailing the gaming club members. Participants were selected with no regards to gender, age, or experience with VR.
- **Experimental Runs:** The participant first signed the consent forms, and the first phase of the experimental run was used to familiarize the participant with the HTC Vive. They were then asked to play the Arcade style game, where data was recorded using our created tools. Participants were finally asked to complete a post-immersion survey and were debriefed of the nature of the deception.

7.2 Experimental Design

To afford maximal play space along the axis of attack, the experiment started with the VE offset from the center of the physical room as shown in Figure 3. The Chaperone was disabled to prevent participants from receiving visual cues regarding the physical room (3.4mx3.4m) and the workstation positioned such that the HTC Vive's tether allowed access to all areas of the room. We defined a location for the participant to reach signifying attack success as 1.9 meters in front of the participant along the Y-axis with a radius of 20 centimeters. The distance to the destination was determined sufficient from observing a participant's typical *territory* for the games to be tested. This ensured participants would not travel to the destination as a result of uninfluenced gameplay, and in most games, this was discouraged by virtual walls and boundaries. Games were selected on the basis that they could be quickly understood, player skill would not greatly influence the experience, and the average game cycle would last between 5-15 minutes.

Participants were provided with a quick HTC Vive tutorial and were allowed to familiarize themselves with the system. When they started a VR game, we started collecting HMD and VE location data using our developed tools at a frequency of $20x/second$. Prior to commencing the attack, participants were allowed to play for at least one game cycle. This allowed us to confirm that each individual player territory did not extend to the destination.

The Human Joystick Attack was then administered by shifting the VE along the Y-axis at a rate of 0.01 $meters/second$[1]. The total attack translation equaled the distance to the destination finishing with the VE centered on the destination. Data collection continued for a maximum of 15 minutes or until the participant was halted due to proximity to physical room boundaries. Participants completed a post-immersion survey which asked them about their in-game (immersed) awareness to the attack.

7.3 Observations and Results

In this section we present our findings for each game tested shown in Table 4. We tested $(n = 5)$ games and recruited $(n = 64)$ participants. Although the purpose of the experiment was to investigate the success of the Human Joystick Attack, we tested the attack against several styles of games to explore the external validity of the constructed attack. We note that this is not a comprehensive analysis of this attack type, against all application types, but we observed trends in gameplay that lend to the success of the attack.

TABLE 4: Success Rate per Game

	Game	R/U	R/A	F/U	F/A	N
1	Longbow	26	1	0	0	27
2	Xortex	11	0	0	1	12
3	Slingshot	6	0	0	0	6
4	Surgeon Simulator	8	0	1	0	9
5	Guns'n'Stories	5	0	5	0	10
	Total	56/64	1/64	6/64	1/64	

R: Reached Destination U: Unaware of the attack
F: Failed to reach the Destination A: Aware of the attack

1. Demonstration can be viewed at https://www.youtube.com/watch?v=iyK94jFuniM

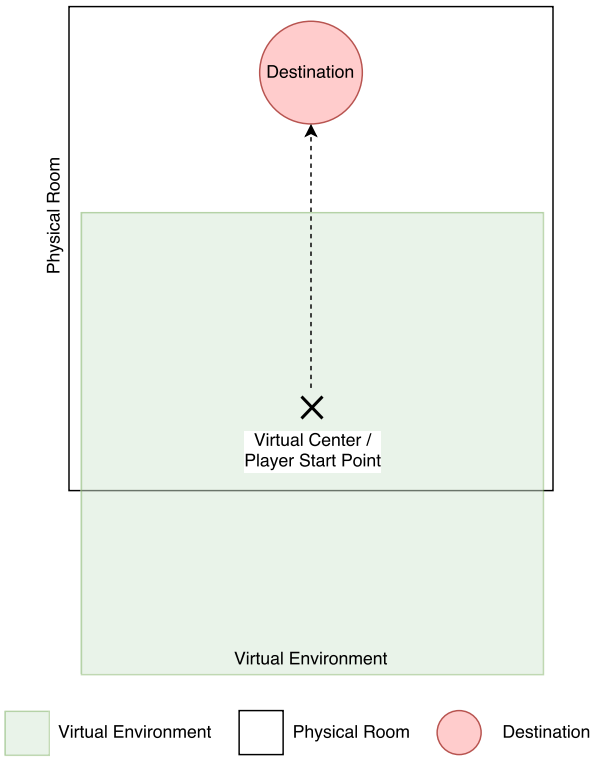

Fig. 3: Experimental Design for the Human Joystick Experiment

Games 1-4 were common in that the player was either required to move to a given location in order to interact with a virtual object or given an advantage based on location. These games provided strong motivation for the player to remain centered in the VE and accordingly we achieved a 94.4% success rate of being able to move an immersed VR user to our predetermined location.

Game 5, Guns'n'Stories was selected on the basis that the player was not forced to interact with a static virtual object and could be played standing only. For this game we observed a 50% success rate. The participant reaction to the attack varied in this game, likely in part due to their style of play. As this game did not explicitly provide motivation for the player to remain centered in the VE, participants that did not respond allowed the VE drift away from them. In contrast, participants who chose to interact with scenery responded in a similar fashion to Games 1-4. A third and peculiar response was observed in three participants; they did not interact with static virtual objects yet tended to maintain their relative location to nearby objects. This suggests that some players, regardless of gameplay requirements, will subconsciously self-correct to the translation. Although the attack's success was not on par with Games 1-4, this finding suggests that some victims will be susceptible to the attack regardless of the application.

Participants who responded to the attack $(56/64)$, reported being surprised of their location at the conclusion of the VR immersion. Figure 4 shows example paths participants took in the VE, which remain centered in the playspace. This is a visualization of what the player experienced in regard to the VE. The player's steady territory in the VE is demonstrative of the player's ignorance to the translation as they perceive their overall location to be static. The contrast in the path in the VE to the physical path indicates that the forward movement was a result of the Human Joystick Attack.

Thirteen participants were reoriented upon colliding with a physical wall at the destination, however, reported prior to the collision they were unaware they had traveled significantly from the starting location. Of the two participants that were aware of the changes to the VE, one noticed the translation and complied while the other simply did not feel comfortable moving in VR thus observing the full $1.9m$ translation. The success of this attack will be dependent on the victim's ability and willingness to move as the VE translation is simply attempting to lead the player to the destination. We did not vary the rate at which the attack was carried out, however, further investigation into the relationship between the rate and magnitude of the attack versus awareness may lead to an improved success rate.

As participants indeed tended to follow the virtual center-point, we observed two major mechanisms that resulted in them correcting for the VE translation. The Xortex, Surgeon Simulator and Guns'n'Stories games made player frequently rotate. This caused participants to make many small steps in which they may not notice they had moved forward. Participants that confidently moved about in virtual space and actively engaged in the game tended to closely reflect the translation in comparison to participants that were timid in their movements. Figure 4, Xortex, Surgeon Simulator and Guns'n'Stories paths show the player gradually adjusting to the translation as a result of frequent rotations.

Although the game Longbow involved rotating, this proved to be dependent on the participant's style of play. Participants that did not readily move their feet tended to respond to the translation with similarity to the Longbow and Slingshot paths in Figure 4. The correction to the VE translation primarily took place through a large movement forward followed by an incomplete reciprocal movement. Participants reported feeling as though they had to reach farther than normal, however, after completing the motion felt as though they had returned to their original position.

The degree of the participants' movements may be representative of their prior experience with VR and comfort and confidence moving about in the VE is reflective of a participant's familiarity with VR. To determine if prior experience influenced the rate at which participants responded to the attack, we compared the time it took participants to reach the destination for ones that reported prior experience against those who had not. Participants that reported prior VR use (M = 183.0, SD = 62.9) showed no difference in time to reach the destination compared to those who had not (M = 181.6, SD = 42.0), $t(53)$ = -0.10, p = 0.919. The p value indicates there is a 91.9% probability that the two groups are statistically similar.

As the rate of the attack remained constant throughout the experiment, these preliminary results point to the hypothesis that the attack is successful regardless of a player's experience level, although more testing is needed to validate that claim. The mean time to the destination further reinforces our original hypothesis given the rate of translation

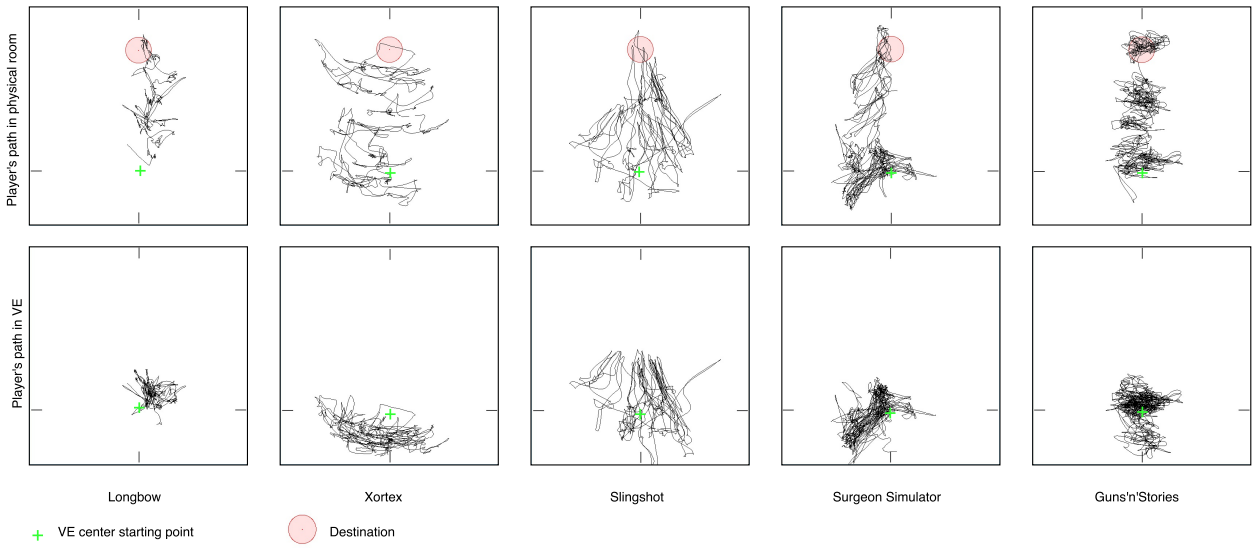

Fig. 4: Player path in the physical room compared to the path in the Virtual Environment. The path in virtual environment depicts how the player perceives their movements due their relationship to the virtual objects around them, while the physical path shows the movements the player truly made.

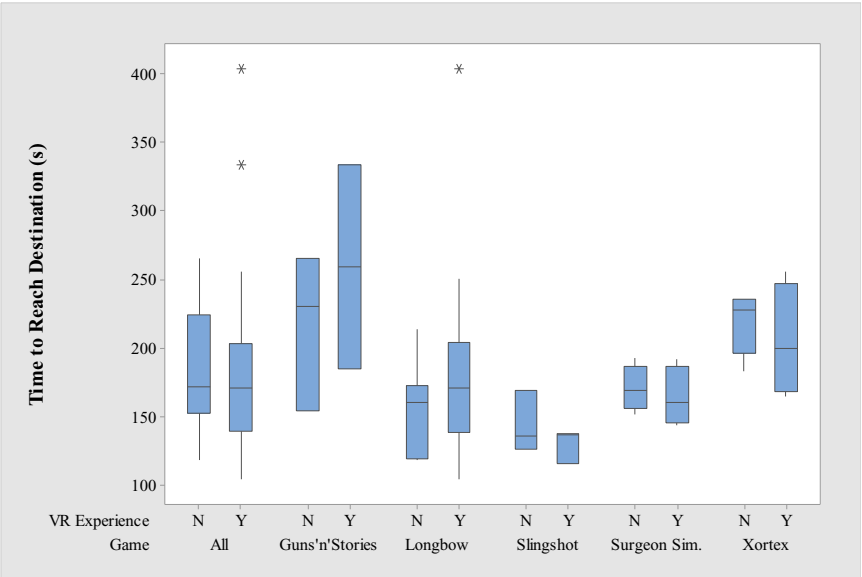

Fig. 5: Time to reach the destination grouped by VR experience. Mean times are comparable between players with VR experience and those without. Trend holds within games and overall.

$(0.01m/s)$ and distance $(1.9m)$. An ideal reaction to the attack would take the participant $170 - 190s$ due to the radius of our destination $(20cm)$ compared to observed combined mean time to destination of $182.4s$. The overall and per game similarity of times for both groups are shown in Figure 5. Comparing the times per application may reveal a relationship with player responsiveness, however, this initial study's limitation on sample size precluded this analysis.

The games serve two functions in facilitating the Human Joystick Attack. We hypothesize that constant movements dilute small shifts in the VE, and the game's objective captivates the user. This may provide the necessary distraction to prevent the attack from being perceived. Secondly, the game provides the motivation for the user to move. Often, players are forced to comply with the attack in order to continue to play. The large forward movement described above was occasionally a result of the participant reaching for a virtual object or interacting with a menu item necessary for the player to carry on. As a result, games that have strong ties to a particular space in the VE will obligate the player to move. In such a case, the success of the attack relies more so on remaining clandestine. On the other hand, games that do not provide advantage to player location must depend on the player's tenancy to maintain their relative position. The games used in this study represent a small portion of the many types of applications for immersive VR, many of which intend for the player to remain seated. As each

style of application varies in the level of distraction and motivation, further investigation may yield a relationship between the degree of immersion and susceptibility to the attack.

8 Discussion and Mitigations

The technical universal success for each our attacks lends to the attack layer, OpenVR. With reasonable certainty, we can conclude that all OpenVR compatible systems are susceptible to these types of attacks. Irrespective of the hardware interface, we contend that all aspects of VR systems need to be protected. The consequence of malicious behavior is amplified by the physical and physiological implications of immersion. Although we specifically targeted OpenVR, the potential exploits identified in this study are applicable to all VR systems. Demonstrated by our proof-of-concept attacks, VR security and safety features require further development and protection.

Given that all fully immersive VR systems will require some form of collision detection, a comprehensive evaluation of these safety features should be prioritized. Though we agree there is some benefit in the user manually adjusting the boundaries of his/her play area, access to this data needs to be guarded. SteamVR storing these artifacts in plaintext provided us with a major vulnerability. We recommend encrypting this data and restricting its access to the creating individual and service. Because the integrity of safety features must be maintained, while also manageable, we suggest applying the Clark-Wilson integrity model. Where transactions must be made and validated through the VR runtime.

The Overlay and Camera Attacks exhibited the permissive construction of the architecture. During our testing, we utilized in excess of five additional malicious applications running in the background of the scene. Again, SteamVR made apparent the lack of application control and OculusVR failed to check for renderless processes. OpenVR offers an API interface for application management [24], however, the success of the Overlay Attack demonstrated the underutilization of application isolation. Ungoverned applications led to loss of control of the VR system and data leakage. Windowing, or limiting VR application access to their respective environment may suit to prevent malicious actors from interrupting valid applications [26]. Thus, we recommend that VR application managers limits access to vetted programs and require user approval for third-party applications to a greater extent than found. Although outside of the scope of this paper, Steam applications were found to lack integrity checks. We recommend instituting application signing to prevent unwarranted modifications.

An alternative solution, *Arya* is a framework to implement policies that govern AR reality output [27]. This policy management should be incorporated into existing immersive VR systems.

The Camera Attack made apparent the lack of permission-based access. Though Steam allowed the user to restrict the camera's usage, the permission attribute was stored unencrypted and was easily accessible. This transparency allows an attacker to overcome camera restrictions. As VR continues to spread, so will its functionality and features. For example, Android has implemented permission-based security with Access Control Lists (ACL) [28]. A similar approach with the special features of VR systems, limiting their use to specified applications may prevent malicious third-party software from abusing their access. We recommend access control be expanded to authenticate all applications accessing VR resources. Many of which can be validated should application signing from trusted developers be instituted.

9 Related Work

AR and VR both require special considerations when handling sensitive user information. Tracking and processing of sensor information differs from traditional mobile devices which may not need prolonged access to such data streams [29]. Applications which utilize MR input may be unintentionally granted access to unfiltered video streams. [30] proposes fine grained permission and abstraction of the AR object recognizer to relieve applications need for direct access. Content sharing within the VE also requires consideration; SecSpace presents a model for privacy permissions based on physical space [31].

HMD's provide a separation of visual channels, where interactions with authentication mechanism's (PIN pads, and gestures) can be more difficult to detect by an onlooker [32]. AR headsets are a prime candidate for visual cryptography. Image pairs that while separate are meaningless can be overlaid using Google Glass decoding the underlying message [33], [34]. Overlaying randomized keys onto a physical keyboard can further disguise input and prevent shoulder surfing [35]. Conversely, *candid interaction* with AR may grant onlookers context into users' computing activities [36].

Securing the input and output of AR systems has been investigated by [26], [27], [29], where the system runtime institutes policies which govern acceptable application behavior. [37] investigates the security of AR specific web browsers and the challenges associated with supporting AR content. The unique capabilities of MR systems can be leveraged to improve authentication procedures. [38] incorporates Android's facial recognition into handshake protocol of connecting devices.

10 Future Work

Although our attacks are focused at the software interface layer, throughout our analysis, we identified other areas of vulnerabilities. OpenVR provides the framework to develop drivers for additional hardware [20]. This includes overloading the drivers already in place for the Vive system and creating virtual controllers [21]. Menu operations and interactive overlays largely receive input from a controller's pointer and with a malicious virtual controller, an adversary can remotely control the player's computer system.

A popular VR application such as Virtual Desktop, which by default loads on start-up, allows users access to their desktop in VR [39]. Should a virtual controller be present, the attacker may gain complete access to the client's computer. A virtual controller has the advantage

of transparency, where a 3D model representation of the controller would not be specified nor rendered. It should be noted that the installation of the drivers would make delivery and execution of the payload increasingly complex.

The HTC Vive has a unique tracking system that utilizes *lighthouses*. These lighthouses use IR to provide absolute positioning for the tracked devices [40]. To reduce the number of necessary sensors required, the tracking solution is supplemented by relative tracking provided by an IMU [15]. Tracking solely based on IMU data alone is not very accurate but is useful to fill in the gaps when sensors are obstructed. We suspect that disabling absolute tracking will cause the tracking solution to drift from the true player's location. Further testing is required, however, the *Lighthouse Redox* HTC Vive reverse engineering project, reversed the Bluetooth Low Energy (LE) communication to wake up and set sleep timeout [41].

The lighthouses have their own set of configuration and log files that contain location and normal information. These files are also stored in plaintext JSON format. These artifacts may present an additional avenue of attack. It is likely that the disorientation attack could be carried out by manipulating the locations of the lighthouses themselves. The approach we used attacked a static user defined *universe*, established during the room setup. Being that the lighthouses self-configure, this may not be as feasible. Furthermore, if the lighthouses detect that they have been disturbed they will immediately power down the rotors [15]. Further testing is required to determine if IMU drifting could be induced by manipulating these artifacts.

Social applications allow immersed VR users to congregate and share content in VR. Applications such as Big Screen Beta, go as far to share a user's computer screen, gestures, and audio [42]. Our preliminary network layer analysis has shown that much of this information is passed unencrypted. We suspect that a new form of the Man-In-The-Middle (MITM) Attack may allow an attacker to join a chat room and extract information and even possibly manipulate the client's environment [43]. We coin this the *Man-In-The-Room* (MITR) Attack.

Despite the name, OpenVR is currently not entirely open source. Valve is yet to release the source code, however, the API acts as a wrapper for backend drivers and user applications inherit the OpenVR Dynamic Linked Library (DLL). Analysis of the `openvr_api.dll` with a tool such as Interactive Disassembler *(IDA) Pro* may uncover further vulnerabilities [44]. In our initial investigation we were able to identify the memory locations storing the tracking solutions for tracked devices. Disabling the tracking updates and manipulating the present solution could allow for the adversary to control the victim's controllers,

11 Conclusion

With any emerging technology, securing data and protecting the user may present new challenges, however proven techniques and best practices must not be overlooked. The permissive nature of the tested VR systems is a testament to the lack of implemented security and privacy measures. A deeper evaluation of the vulnerabilities unique to VR and Immersion attacks is warranted for the safety and privacy of its users. As development out paces security, the broad range of applications for VR will continue to provide a comparable breadth of vulnerabilities capable of exploiting the physical and psychological ramifications of immersion.

Acknowledgements

This material is based upon work supported by the National Science Foundation under Grant No. 1748950. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

References

[1] H. B. et al, "Profiles in innovation, virtual and augmented reality, understanding the race for the next computing platform," *Equity Research*, January 13 2016.
[2] C. W. Mark Beccue, "Virtual reality for consumer markets," *Tractica*, vol. 4Q, 2016.
[3] M. T. Michael Shirer. (2017, August 07) Worldwide spending on augmented and virtual reality expected to double or more every year through 2021, according to idc. [Online]. Available: https://www.idc.com/getdoc.jsp?containerId=prUS42959717
[4] J. Durbin. (2017, January 17) Super data report: 6.3 million virtual reality headsets shipped in 2016. [Online]. Available: https://uploadvr.com/report-6-3-million-virtual-reality-headsets-shipped-2016/
[5] G. Riva, "Virtual reality: an experiential tool for clinical psychology," *British Journal of Guidance & Counselling*, vol. 37, no. 3, pp. 337–345, 2009.
[6] M. V. Sanchez-Vives and M. Slater, "From presence to consciousness through virtual reality," *Nature Reviews Neuroscience*, vol. 6, no. 4, pp. 332–339, 2005.
[7] Meehan, Michael, Insko, Brent, M. Whitton, and F. P. Brooks Jr, "Physiological measures of presence in stressful virtual environments," *ACM Transactions on Graphics (TOG)*, vol. 21, no. 3, pp. 645–652, 2002.
[8] Y. Wang, K. Otitoju, T. Liu, S. Kim, and D. A. Bowman, "Evaluating the effects of real world distraction on user performance in virtual environments," in *Proceedings of the ACM symposium on Virtual reality software and technology*. ACM, 2006, pp. 19–26.
[9] Carlin, A. S, Hoffman, H. G, and S. Weghorst, "Virtual reality and tactile augmentation in the treatment of spider phobia: a case report," *Behaviour research and therapy*, vol. 35, no. 2, pp. 153–158, 1997.
[10] D. C. Craig Timberg, "Net of insecurity, a flaw in the design," May 30 2015.
[11] F. M. Robin Mckie, "Virtual reality headsets could put childrens health at risk," https://www.theguardian.com/technology/2017/oct/28/virtual-reality-headset-children-cognitive-problems, last accessed 2017-11-02.
[12] J. A. de Guzman, K. Thilakarathna, and A. Seneviratne, "Security and privacy approaches in mixed reality: A literature survey," *arXiv preprint arXiv:1802.05797*, 2018.
[13] D. Nield, "How oculus rift works: Everything you need to know about the vr sensation," March 29 2016, HowOculusRiftworks: EverythingyouneedtoknowabouttheVRsensation, last-accessed 2017-11-03.
[14] N. Patel and D. Cober. Adjacent reality.
[15] E. Williams. (2016, December 21) Alan yates: Why valves lighthouse can't work. [Online]. Available: https://hackaday.com/2016/12/21/alan-yates-why-valves-lighthouse-cant-work/
[16] Valve, "Api documentation," https://github.com/ValveSoftware/openvr/wiki/API-Documentation, last accessed 2017-10-26.
[17] Road to VR, "Steamvr will support microsoft vr headsets," https://www.roadtovr.com/windows-vr-headsets-mixed-reality-support-steamvr/, last accessed 2017-10-26.
[18] SFML, "Documentation of sfml 2.4.2," https://www.sfml-dev.org/documentation/2.4.2/2.4.2/classsf_1_1UdpSocket.php, last accessed 2017-10-26.

[19] L. Caroux, L. Le Bigot, and N. Vibert, "Impact of the motion and visual complexity of the background on players' performance in video game-like displays," *Ergonomics*, vol. 56, no. 12, pp. 1863–1876, 2013.
[20] V. Joe Ludwig, "Driver documentation," https://github.com/ValveSoftware/openvr/wiki/Driver-Documentation, last accessed 2017-10-31.
[21] matzman666, "Openvr-inputemulator," https://github.com/matzman666/OpenVR-InputEmulator, last accessed 2017-10-31.
[22] Valve, "Ivroverlay overview," https://github.com/ValveSoftware/openvr/IVROVerlay_Overview, last accessed 2017-10-25.
[23] B. Lang, "Oculus quest hands-on and tech details," 2018, https://www.roadtovr.com/oculus-quest-hands-specs-tech-details-oculus-connect-5/.
[24] Valve, "openvr.h," https://github.com/ValveSoftware/openvr/blob/master/headers/openvr.h, last accessed 2017-10-26.
[25] J. Butler, J. L. Undercoffer, and J. Pinkston, "Hidden processes: the implication for intrusion detection," in *Information Assurance Workshop, 2003. IEEE Systems, Man and Cybernetics Society*. IEEE, 2003, pp. 116–121.
[26] K. Lebeck, T. Kohno, and F. Roesner, "How to safely augment reality: Challenges and directions," in *Proceedings of the 17th International Workshop on Mobile Computing Systems and Applications*. ACM, 2016, pp. 45–50.
[27] K. Lebeck, K. Ruth, T. Kohno, and F. Roesner, "Securing augmented reality output," *IEEE Security & Privacy*, 2017.
[28] D. Barrera, H. G. Kayacik, P. C. van Oorschot, and A. Somayaji, "A methodology for empirical analysis of permission-based security models and its application to android," in *Proceedings of the 17th ACM conference on Computer and communications security*. ACM, 2010, pp. 73–84.
[29] F. Roesner, T. Kohno, and D. Molnar, "Security and privacy for augmented reality systems," *Communications of the ACM*, vol. 57, no. 4, pp. 88–96, 2014.
[30] S. Jana, D. Molnar, A. Moshchuk, A. M. Dunn, B. Livshits, H. J. Wang, and E. Ofek, "Enabling fine-grained permissions for augmented reality applications with recognizers." in *USENIX Security Symposium*, 2013, pp. 415–430.
[31] D. Reilly, M. Salimian, B. MacKay, N. Mathiasen, W. K. Edwards, and J. Franz, "Secspace: prototyping usable privacy and security for mixed reality collaborative environments," in *Proceedings of the 2014 ACM SIGCHI symposium on Engineering interactive computing systems*. ACM, 2014, pp. 273–282.
[32] C. Goerge, M. Khamis, E. von Zezschwitz, M. Burger, H. Schmidt, F. Alt, and H. Hussmann, "Seamless and secure vr: Adapting and evaluating established authentication systems for virtual reality," in *Proceedings of the Network and Distributed System Security Symposium (USEC17). NDSS. DOI: http://dx. doi. org/10.14722/usec*, 2017.
[33] S. J. Andrabi, M. K. Reiter, and C. Sturton, "Usability of augmented reality for revealing secret messages to users but not their devices." in *SOUPS*, vol. 2015, 2015, pp. 89–102.
[34] P. Lantz, B. Johansson, M. Hell, and B. Smeets, "Visual cryptography and obfuscation: A use-case for decrypting and deobfuscating information using augmented reality," in *International Conference on Financial Cryptography and Data Security*. Springer, 2015, pp. 261–273.
[35] A. Maiti, M. Jadliwala, and C. Weber, "Preventing shoulder surfing using randomized augmented reality keyboards," in *Pervasive Computing and Communications Workshops (PerCom Workshops), 2017 IEEE International Conference on*. IEEE, 2017, pp. 630–635.
[36] B. Ens, T. Grossman, F. Anderson, J. Matejka, and G. Fitzmaurice, "Candid interaction: Revealing hidden mobile and wearable computing activities," in *Proceedings of the 28th Annual ACM Symposium on User Interface Software & Technology*. ACM, 2015, pp. 467–476.
[37] R. McPherson, S. Jana, and V. Shmatikov, "No escape from reality: Security and privacy of augmented reality browsers," in *Proceedings of the 24th International Conference on World Wide Web*. International World Wide Web Conferences Steering Committee, 2015, pp. 743–753.
[38] E. Gaebel, N. Zhang, W. Lou, and Y. T. Hou, "Looks good to me: Authentication for augmented reality," in *Proceedings of the 6th International Workshop on Trustworthy Embedded Devices*. ACM, 2016, pp. 57–67.
[39] G. Godin, "Virtual desktop," http://store.steampowered.com/app/382110/Virtual_Desktop/, last accessed 2017-10-31.
[40] D. Coldewey. (2016, Apr 26) Teardown of htc vive highlights the headset's differences from oculus rift. Last updated - 2016-04-27. [Online]. Available: https://techcrunch.com/2016/04/26/teardown-of-htc-vive-highlights-the-headsets-differences-from-oculus-rift/
[41] p. nairol, "Basestation, bluetooth le communications," https://github.com/nairol/LighthouseRedox/blob/master/docs/Base%20Station.md, last accessed 2017-10-31.
[42] B. S. Inc., "bigscreen," bigscreenvr.com, last accessed 11-01-2017.
[43] F. Callegati, W. Cerroni, and M. Ramilli, "Man-in-the-middle attack to the https protocol," *IEEE Security & Privacy*, vol. 7, no. 1, pp. 78–81, 2009.
[44] "IDA pro," https://www.hex-rays.com/products/ida/.

Peter Casey received his B.S. from SUNY Geneseo and is pursuing his M.S. in Computer Science from the University of New Haven. He is a member of the University of New Haven's Cyber Forensics Research and Education Group (UNHcFREG) and Virtual Reality Security Research Laboratory.

Ibrahim Baggili PhD is the Elder Family Endowed Chair of Computer Science, and the founder of UNHcFREG. He is also the founder of the Virtual Reality Security Research Laboratory. He received all of his degrees from the Purdue Polytechnic Institute, and was a researcher at CERIAS.

Ananya Yarramreddy received two Masters, one M.Sc. degree in Computer and Networks from University of North Umbria, UK and M.Sc. degree in Cybersecurity from University of New haven, USA. She is currently a Security Specialist at State of North Carolina. Her career interests cover the designing of Network/Cloud Infrastructure and Vulnerability assessment of networks/cloud and ethical hacking.