BRNO UNIVERSITY OF TECHNOLOGY
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

FACULTY OF INFORMATION TECHNOLOGY
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

DEPARTMENT OF INFORMATION SYSTEMS
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

SECURITY ANALYSIS OF IMMERSIVE VIRTUAL REALITY AND ITS IMPLICATIONS

BEZPEČNOSTNÍ ANALÝZA VIRTUÁLNÍ REALITY A JEJÍ DOPADY

MASTER'S THESIS
DIPLOMOVÁ PRÁCE

AUTHOR Bc. MARTIN VONDRÁČEK
AUTOR PRÁCE

SUPERVISOR Ing. JAN PLUSKAL
VEDOUCÍ PRÁCE

BRNO 2019

Brno University of Technology
Faculty of Information Technology

Department of Information Systems (DIFS) Academic year 2018/2019

Master's Thesis Specification

22158

Student:	**Vondráček Martin, Bc.**
Programme:	Information Technology Field of study: Information Technology Security
Title:	**Security Analysis of Immersive Virtual Reality and Its Implications**
Category:	Security

Assignment:

1. Deliver a brief overview of currently available and popular VR applications intended for communication and social activities.
2. Study technologies used in the field of such VR applications. Focus also on security mechanisms and possible vulnerabilities.
3. Get familiar with methods concerning forensic analysis and penetration testing. Outline selected tools suitable for this case of VR applications from the view of reverse engineering, penetration testing, and forensic analysis.
4. Carry out an analysis of selected VR application, while documenting applied approaches and utilized tools.
5. Evaluate outcomes of this work, discuss impact in the field of VR applications.

Recommended literature:

1. Yarramreddy, Ananya & Gromkowski, Peter & Baggili, Ibrahim. (2018). Forensic Analysis of Immersive Virtual Reality Social Applications: A Primary Account. 10.1109/SPW.2018.00034.
2. Oriyano, Sean-Philip. Penetration Testing Essentials. (2016). Somerset: John Wiley & Sons, Incorporated.
3. Callegati, W. Cerroni, and M. Ramilli. Man-in-the-Middle Attack to the HTTPS Protocol. In: Security Privacy, IEEE (Jan. 2009), pp. 78-81.

Requirements for the semestral defence:

- Items 1, 2, 3.

Detailed formal requirements can be found at http://www.fit.vutbr.cz/info/szz/

Supervisor:	**Pluskal Jan, Ing.**
Consultant:	Baggili Ibrahim, University of New Haven
Head of Department:	Kolář Dušan, doc. Dr. Ing.
Beginning of work:	November 1, 2018
Submission deadline:	May 22, 2019
Approval date:	November 19, 2018

Abstract

Immersive virtual reality is currently used not only for entertainment but also for work and social interaction where user's privacy and confidentiality of the information has a high priority. Unfortunately, security measures applied by software vendors are often not sufficient. This thesis delivers an extensive security analysis of a popular VR application Bigscreen which has more than 500,000 users. Techniques of network traffic analysis, penetration testing, reverse engineering, and even application crippling were utilised. Research led to a discovery of critical vulnerabilities directly exposing the privacy of the users and allowing the attacker to take full control of a victim's computer. Found security flaws allowed distribution of malware and creation of a botnet using a computer worm spreading in virtual environments. A novel VR cyber attack Man-in-the-Room was implemented. Furthermore, a security vulnerability in the Unity engine was discovered. Carried out responsible disclosure has helped to mitigate the risks for more than half a million Bigscreen users and all affected Unity applications worldwide.

Abstrakt

Virtuální realita je v současné době využívána nejen pro zábavu, ale i pro práci a sociální interakci, kde má soukromí a důvěrnost informací vysokou prioritu. Avšak bohužel, bezpečnostní opatření uplatňovaná dodavateli softwaru často nejsou dostačující. Tato práce přináší rozsáhlou bezpečnostní analýzu populární aplikace Bigscreen pro virtuální realitu, která má více než 500 000 uživatelů. Byly využity techniky analýzy síťového provozu, penetračního testování, reverzního inženýrství a dokonce i metody pro application crippling. Výzkum vedl k odhalení kritických zranitelností, které přímo narušovaly soukromí uživatelů a umožnily útočníkovi plně převzít kontrolu nad počítačem oběti. Nalezené bezpečnostní chyby umožnily distribuci škodlivého softwaru a vytvoření botnetu pomocí počítačového červa šířícího se ve virtuálních prostředích. Byl vytvořen nový kybernetický útok ve virtální realitě nazvaný Man-in-the-Room. Dále byla objevena bezpečnostní chyba v Unity engine. Zodpovědné nahlášení objevených chyb pomohlo zmírnit rizika pro více než půl milionu uživatelů aplikace Bigscreen a uživatele všech dotčených aplikací v Unity po celém světě.

Keywords

Immersive Virtual Reality, Man-in-the-Room Attack, Responsible Disclosure, Bigscreen, Unity, Forensic Analysis, Security Analysis, Network Traffic Analysis, Penetration Testing, Reverse Engineering, Application Patching, Application Crippling, Augmented Reality, Mixed Reality, HTC Vive, Oculus Rift, Altspace VR, Rec Room, Facebook Spaces

Klíčová slova

Immersive Virtual Reality, Man-in-the-Room Attack, Responsible Disclosure, Bigscreen, Unity, Forenzní analýza, Bezpečnostní analýza, Analýza síťového provozu, Penetrační testování, Reverzní inženýrství, Application Patching, Application Crippling, Rozšířená realita, Smíšená realita, HTC Vive, Oculus Rift, Altspace VR, Rec Room, Facebook Spaces

Reference

VONDRÁČEK, Martin. *Security Analysis of Immersive Virtual Reality and Its Implications.* Brno, 2019. Master's thesis. Brno University of Technology, Faculty of Information Technology. Supervisor Ing. Jan Pluskal

Rozšířený abstrakt

Virtuální realita je rozšiřující se oblast ve světě počítačových technologií, která nachází stále více a více oblastí využití. Přestože počátky virtuální reality můžeme najít už v 70. letech minulého století, extrémní nárůst výkonu a velmi přijatelné ceny dělají tuto technologii nyní dostupnou i pro běžného uživatele. Virtuální realita dnes slouží nejen k zábavě, ale využívá se k průmyslovým aplikacím, výzkumu a vývoji, ale také k sociální interakci. Právě sociální a komunikační aplikace pro virtuální realitu nachází využití jak pro volnočasové setkávání, tak i například pro firemní jednání jako forma telekonference. Zejména v rámci komunikačního využití má mít zajištění soukromí a důvěrnosti informací vysokou prioritu.

Řada dodavatelů software usiluje o dodání svých produktů na trh v nejkratší možné době. Avšak bohužel, v tomto kompetitivním prostředí vývojářské společnosti často nevěnují dostatečnou pozornost zabezpečení svých produktů. Produkty jsou často vydávány už jako alfa a beta verze zatímco jsou stále ve vývoji. Tato situace dává více prostoru bezpečnostním slabinám, které jsou zavedeny během průběžného vývoje software a které zatím nebyly odhaleny důslednou bezpečností analýzou.

Tato práce se věnuje oblasti bezpečnosti technologií a aplikací pro virtuální realitu se sociálním využitím. Práce přináší systematickou bezpečnostní analýzu populární aplikace Bigscreen pro virtuální realitu, která má více než 500000 uživatelů. Tato analýza navazuje na předchozí výzkumy na University of New Haven v Connecticutu v USA, kde autor této práce působil jako *Visiting Scholar.* Oblast virtuální reality má rostoucí význam pro kybernetickou bezpečnost a forenzní analýzu.

Technologie využívané v rámci virtuální reality spadají do širokého souboru témat. Aby bylo možné prověřit bezpečnost daných aplikací, je nutná detailní znalost použitých technologií. Při bezpečnostní analýze jsou využívány také znalosti vývoje software a zejména povědomí o častých chybách, které vedou k výskytu bezpečnostních zranitelností.

Práce v úvodní části přináší přehled populárních sociálních aplikací pro virtuální realitu. Následně se autor zaměřil na studium systémů pro virtuální realitu. V rámci toho práce dále přináší přehled možností vývoje zmiňovaných aplikací a identifikuje často využívané technologie. Práce v této části shrnuje aktuální výzkumy z oblasti bezpečnosti a forenzní analýzy systémů pro virtuální realitu. Výzkumné projekty tak poukazují na závažná bezpečnostní rizika.

V následujícím kroku tato práce prezentuje přípravu na bezpečnostní a forenzní analýzu aplikace. Jsou zde například shrnuty základní informace, definovány pojmy a nastudovány relevantní metodiky. Na základě identifikování technologií, které jsou často použity v aplikacích pro virtuální realitu, byly nastudovány a připraveny nástroje pro zachytávání a následnou analýzu síťového provozu, nástroje pro reverzní inženýrství C# a .NET software, také nástroje pro deobfuskaci a analýzu zdrojového kódu v jazyce JavaScript.

Následně bylo v rámci tohoto výzkumu přistoupeno k samotné analýze. Byly definovány její cíle a nastavena metodika rozdělující postup do několika fází a charakterizující jednotlivé úkony. Pro vybranou aplikaci Bigscreen byly vytvořeny testovací scénáře pro analýzu a testování. Fáze analýzy zahrnovaly počáteční pasivní průzkum cíle, přípravu laboratoře a vybraných dříve nastudovaných nástrojů, analýzu síťového provozu, penetrační testování z pohledu sítě a reverzní inženýrství samotné aplikace Bigscreen. Následující fáze se zaměřily na vytvoření ukázkových útoků, které demonstrují závažnost objevených zranitelností a začlenění útoků do jednotného nástroje pro srozumitelné vysvětlení dopadů těchto

zranitelností. Objevené chyby a implementované útoky byly následně otestovány s využitím zmíněných scénářů.

V rámci tohoto výzkumu se autorovi podařilo odhalit kritické bezpečnostní zranitelnosti v aplikaci Bigscreen a v platformě Unity. Bezpečnostní chyby umožňovaly závažným způsobem narušit soukromí uživatelů, ale i dokonce kompletně ovládnout jejich počítač. Slabiny v zabezpečení umožňovaly vytvoření ukázkového botnetu napadených aplikací, které byly ovládány z útočníkova *command and control* serveru. Autorovi práce se podařilo vytvořit počítačového červa, který se šířil mezi uživateli při setkání ve virtuálním prostoru. Autor dále zjistil, jak realizovat nový kybernetický útok nazvaný *Man-in-the-Room*, který byl teoreticky definován předchozími výzkumy. Při útoku *Man-in-the-Room* se útočník dokázal připojit do soukromých uzavřených místností a pohybovat se ve virtuálním prostoru s ostatními účastníky a být přitom neviditelný. Tento útok tak dramaticky narušuje soukromí uživatelů. Nalezené zranitelnosti byly zodpovědně nahlášeny v rámci procesu *responsible disclosure* společnostem Bigscreen a Unity Technologies. V rámci komunikace byly společnostem předány také doporučení, jak napravit uvedené chyby a jak se chybám vyvarovat v budoucnu. Obě společnosti zareagovaly a provedly nápravná opatření.

Výsledky tohoto výzkumu si získaly pozornost řady světových i českých médií a autor práce se dále věnoval popularizaci bezpečnosti virtuální reality. Výzkum byl prezentován na Excel@FIT 2019, kde autor obdržel *Ocenění odborné komise*, *Cenu Jiřího Kunovského (Ocenění veřejnosti)* a *Ocenění společnosti Škoda Auto*.

Tento výzkum se tedy zabýval bezpečností technologií a aplikací pro virtuální realitu. Autorovi se podařilo nejen realizovat nové útoky na virtuální realitu, ale hlavně zmírnit bezpečnostní rizika pro vice než půl milionu uživatelů aplikace Bigscreen a pro uživatele všech aplikací zasažených zranitelností v platformě Unity.

Security Analysis of Immersive Virtual Reality and Its Implications

Declaration

Hereby I declare that this master's thesis was prepared as an original author's work under the supervision of Ing. Pluskal with the supplementary information provided by expert consultant Ibrahim Baggili, Ph.D. All the relevant information sources, which were used during preparation of this thesis, are properly cited and included in the list of references.

. . . .
Martin Vondráček
May 22, 2019

Acknowledgements

I would like to thank Ing. Jan Pluskal and Ibrahim Baggili, Ph.D. for the great opportunity to work on this very interesting research under their supervision. I would like to thank my colleague Peter Casey for assisting me with this project. I deeply appreciate their help and professional advice.

I am very grateful to Dr Baggili for accepting me for an individual research stay at the University of New Haven. It was an invaluable experience to be part of the University of New Haven Cyber Forensics Research & Education Group.

Contents

Chapter 1

Introduction

This thesis is focused on the area of immersive virtual reality applications. The aim of the research is to deliver an extensive analysis of Virtual Reality (VR) applications from the cybersecurity and forensic point of view. A successful realisation of this research will bring an evaluation of a current state of security risks for users of widely adopted consumer VR systems.

VR is a growing field in the computer technology world which finds more and more areas of application. Although origins of the virtual reality can be dated to 1970s, today's extreme performance improvements and very acceptable costs bring this technology to the average user. Many software & hardware vendors are striving to release their products to the market as soon as possible. Unfortunately, in this highly competitive environment, companies often do not pay enough attention to the security of their products. Products are often being released as alpha and beta versions while they are still under development. This also creates more space for security vulnerabilities which are introduced during continuous software development and which yet have not been revealed by extensive security analysis. Importance of this topic is supported by the fact that VR applications are used not only for entertainment but also for work and social interaction where are high concerns of loss of confidentiality. This work delivers an extensive and novel security analysis in the field of VR social applications which has a growing importance for cybersecurity and forensic research.

Another purpose of this thesis is to gather and structure background knowledge required for a proper understanding of the security analysis. Technologies utilised in VR applications intended for social interactions fall under a wide range of computer technology topics. In order to responsibly evaluate the security of such applications and to be able to discover possible vulnerabilities, it is essential to understand examined technologies in detail. This includes not only individual development knowledge, but particularly specific shortcoming and weaknesses of given technologies, tools, and approaches. During penetration testing, it is also important to keep in mind the most common mistakes that software developers do when working with certain frameworks and programming languages.

The presented research was carried out at the University of New Haven[1] (CT, USA) where Dr Ibrahim Baggili is a principal investigator of a funded grant. Martin Vondráček joined the University of New Haven Cyber Forensics Research & Education Group (UNHcFREG)[2]

[1]http://www.newhaven.edu/
[2]https://www.unhcfreg.com/

as a visiting scholar for this project. Dr Ibrahim Baggili and Peter Casey have kindly provided Vondráček with software & hardware resources and background information for the research. Vondráček was assisted by Casey and advised by Baggili. This thesis presents the technical work of Vondráček unless explicitly stated otherwise.

This thesis achieves the following:

- Implement the *Man-in-the-Room* Attack.
- Implement the first Proof of Concept (PoC) VR Worm & Botnet.
- Conduct a deep security evaluation of a widely used immersive VR social application.

Structure of this thesis is organised as follows. Chapter 2 presents a brief overview of VR applications massively used for virtual social interactions. The chapter covers Facebook Spaces (section 2.1), AltspaceVR (section 2.2), Rec Room (section 2.3), and Bigscreen (section 2.4), where the biggest emphasis is laid on Bigscreen. Following chapter 3 introduces the field of immersive virtual reality systems and virtual environments. Section 3.3 of this chapter emphasises security concerns of Virtual Environments (VEs) and also highlights recently discovered vulnerabilities which put users at risk. It, for example, points to demonstrated novel attacks which have the potential to hurt user of the attacked VR system physically. Chapter 4 is dedicated to cybersecurity and forensic analysis methodologies which are relevant for communication and social VR applications. This chapter also delivers a detailed summary of available tools, which are feasible for individual phases of the cybersecurity and forensic analysis. The main practical contribution of this research is presented in chapter 5, which follows carried out security analysis of the Bigscreen application. Content of chapter 5 corresponds to a proper penetration testing with vulnerability disclosure and exemplary exploits of discovered vulnerabilities. Possible mitigations and suggestions for software developers (and software development companies) are outlined in section 5.15. Section 5.16 describes how the process of *Responsible Disclosure* was handled by UNHcFREG and by contacted companies.

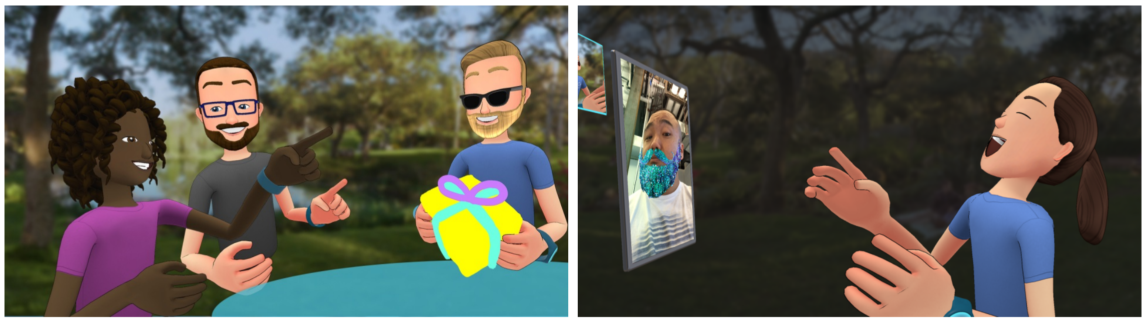
Chapter 2

Selected Virtual Reality Social Applications

This chapter delivers a brief overview of popular VR applications intended for communication and social activities. Applications are selected according to the previous research carried out by *UNHcFREG*[1] (Yarramreddy et al. 2018). Following text describes VR applications from the average user's perspective. Therefore, this chapter aims to introduce the reader to the field of VR applications and does not represent the reconnaissance phase of consequent security analysis, which is in chapter 5.

2.1 Facebook Spaces

Facebook Spaces[2] can be considered a VR extension to the Facebook social network. It allows users to join VE and interact with up to 3 other friends using Oculus Rift or HTC Vive. This use case is presented in fig. 2.1a.



(a) Meeting with friends (b) Facebook Messenger in VE

Figure 2.1: Facebook Spaces is a VR extension for the social network. (Franklin 2017)

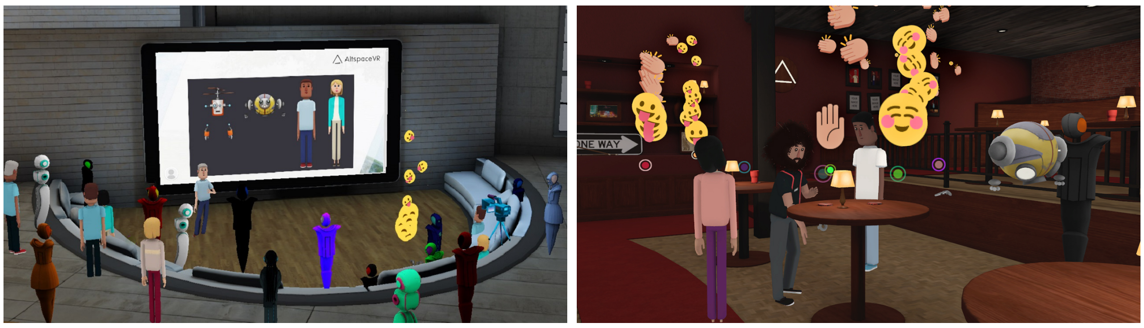
VE of Facebook Spaces allows immersed users to view photos and videos from Facebook. Users can create and then customise their avatar based on their profile photos (Franklin

[1] https://www.unhcfreg.com/
[2] https://www.facebook.com/spaces/

2017). The application also has functionality for drawing 3D objects. Social interactions in Facebook Spaces consist of sharing microphone audio, various hand gestures and video calls using Facebook's Messenger application (Facebook Inc. 2019), as shown in fig. 2.1.

2.2 AltspaceVR

AltspaceVR^{TM} is a VR communication platform (fig. 2.2). It is developed by the company *AltspaceVR, Inc.*, which was founded in 2013, and an open beta version of the application was released to the public in 2015. The company was acquired by Microsoft in 2017 (Eadicicco 2017). The VR application is compatible with Oculus Rift, HTC Vive, Samsung Gear VR, and Daydream by Google. It also supports a mode without VR hardware for Android phones, Windows and Mac (AltspaceVR Inc. 2019).



(a) Virtual presentation (b) Virtual social interactions

Figure 2.2: AltspaceVR is focused mainly on communication between users and social events (AltspaceVR Inc. 2019)

The application is mainly intended for social activities of users, who represented by a selected avatar, share VE with others. Users can also share their microphone audio, web browser, presentations (fig. 2.2a) or images. They are free to create their own public or private virtual events such as sports parties, movie screenings, and mini-games. The company further encourages organising events[3] in VR, including yoga classes, language meet-ups and open mic nights (Pullen 2016).

2.3 Rec Room

Rec Room® is a VR online multiplayer game environment and social application (fig. 2.3). It is a cross-platform software compatible with Oculus Rift, HTC Vive, PlayStation VR, and Windows VR. Rec Room is developed by *Against Gravity*[4] company which was founded in 2016; the application was released in the same year. However, the application is still in Early Access on Steam[5].

Rec Room's players can customise their avatar, who is shown as a simplified floating model of head, torso, and hands. Entertainment activities are considered the primary use of Rec

[3]https://account.altvr.com/events/featured
[4]https://www.againstgrav.com/
[5]https://store.steampowered.com/app/471710/Rec_Room/

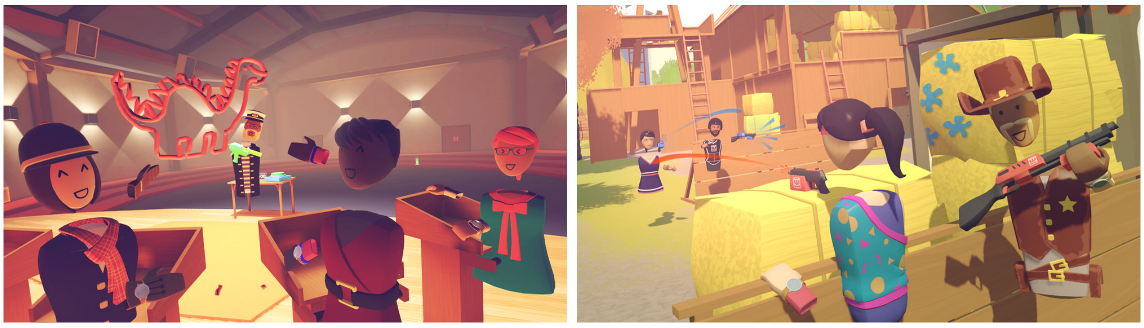
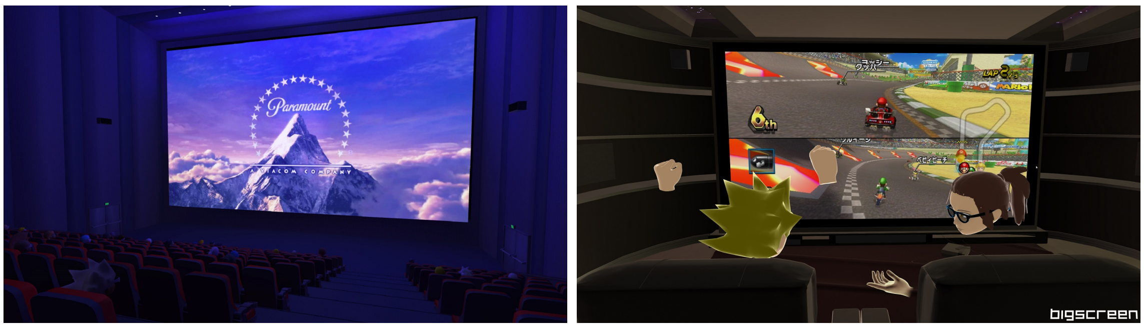

(a) Virtual 3D charades game (b) Virtual paintball

Figure 2.3: Prevailing entertainment use cases of the Rec Room application (Against Gravity Corp. 2019[a])

Room. There are several predefined multiplayer games like paintball (fig. 2.3b), laser tag, disc gold and 3D charades (fig. 2.3a) (Against Gravity Corp. 2019[a]). However, players are able to build their own rooms, events, and mini-games, which further expand the application's possibilities. Against Gravity Corp. (2019b) announced that Rec Room had been installed on over 1 million VR configurations (January 2019). Rec Room's social features emerge mainly in the communal lobby—the locker room—where users can freely meet and hang out with others in VE (Metz 2017). Rec Room also supports microphone audio sharing and creating private rooms.

2.4 Bigscreen

Bigscreen[6] is a VR telepresence platform for social activities. It is intended not only for leisure (entertainment) activities like playing computer games, watching movies and hanging out (fig. 2.4) but also for productivity, work, meetings and collaboration (fig. 2.5). User's of this application, while immersed in a shared VE, can still use their computers in VR via Head-mounted Display (HMD).



(a) Virtual cinema (b) Virtual gaming

Figure 2.4: Entertainment use cases of the Bigscreen application (Bigscreen, Inc. 2018)

Bigscreen application is developed by *Bigscreen, Inc.* which was founded in November 2014. Bigscreen's beta launch was in March 2016, and it became one of the most popular VR

[6]https://bigscreenvr.com/

applications for social activities. It is available for Windows 7, 8.1, 10 operating system through Oculus Home, Steam, Microsoft Store. Shankar (2017b), company's Chief Executive Officer (CEO), has announced \$14 million funding and the application claims to have more than 500, 000 users, according to Bigscreen's press kit (Bigscreen, Inc. 2018). Recent job offers from Bigscreen company even claim it has nearly 1 million users[7]. Concerning only Steam version of Bigscreen, analytic tools like SteamSpy[8] and SteamDB[9] estimate between 200, 000 and 500, 000 owners on Steam. To this date (January 2019), the application has 92 % positive out of 1, 001 reviews on Steam[10], 90 % three or more star reviews out of 1, 767 on Oculus[11].

Concerning social interaction, Bigscreen offers ways to share computer screen, computer audio, microphone audio, text messages, and 3D drawings. Each user can configure their own avatar as their representation in VE. The application also features hand gestures and simulated eye contact (Shankar 2017b). Users can set up VEs in Bigscreen as public or private rooms. According to Bigscreen's in-app Code of Conduct (appendix B), private rooms are invite-only, and they utilise encrypted peer-to-peer communication. This means that private rooms might be used by users for exchanging confidential information.

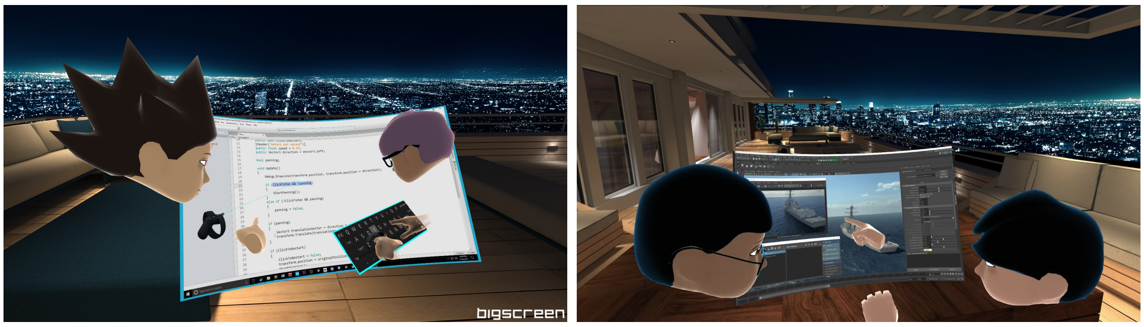

(a) Virtual programming (b) Virtual collaboration

Figure 2.5: Productivity use cases of the Bigscreen application (Bigscreen, Inc. 2018)

The most straightforward use case of Bigscreen application is a VR cinema. A group of friends can create and join their own cinema hall in VE, and one of the users can project their computer screen to *the big screen*. All participants can then see and hear the movie played on *the big screen* and also see and hear each other in a cinema hall. According to Shankar (2017a), the Bigscreen company further partnered with film studios like *Paramount Pictures* to organise VR film screening events, as shown in fig. 2.4a.

Users are able to create customised virtual screens and attach them to their computers. This functionality is used for VR Local Area Network (LAN) parties in terms of computer gaming and entertainment (fig. 2.4b). Users can also play non-VR games inside Bigscreen's VE (Jagneaux 2016). It is even possible to connect gaming consoles like Nintendo Switch[12] to a computer and play games in VR (Williams 2017).

[7]https://jobs.lever.co/bigscreenvr/9f851df7-bbd6-4a6b-8439-156db62a2f39
[8]https://steamspy.com/app/457550
[9]https://steamdb.info/app/457550/graphs/
[10]https://store.steampowered.com/app/457550/Bigscreen_Beta/
[11]https://www.oculus.com/experiences/rift/1018613041536358
[12]https://www.nintendo.com/switch/

2.5 Summary

This chapter introduced popular VR applications selected based on previous forensic research by Yarramreddy et al. (2018). Facebook Spaces application (section 2.1) takes Facebook content into VE and enables VR video calls via Facebook's Messenger. AltspaceVR (section 2.2) is a communication platform with extensive social features and solid hardware support. On the other hand, Rec Room (section 2.3) provides some social features, but it's main focus is on entertainment with mini-games and user-built VEs. Finally, Bigscreen application (section 2.4) is focused on communication and social interaction, but also with productivity and collaboration in mind. Bigscreen can be even used for a meeting during work. It also allows to share several multimedia channels, and it has more than $500,000$ users.

Chapter 3

Technologies in the Field of Immersive Virtual Reality

Following sections provide relevant background information in the field of immersive virtual reality. This chapter is also focused on technologies currently massively used in popular VR social applications. Many of the technologies and approaches are not exclusive to VR but can be seen in various types of communication or visualisation software ranging from entertainment to productivity areas. The information outlined in this chapter was continually updated as new technologies were encountered during consequent security analysis (chapter 5). Therefore, the purpose of this chapter is to set baseline knowledge to understand further chapters of this thesis fully.

VR is computing, simulating, tracking, and visualising system utilized for Human Computer Interaction (HCI). Its aim is to immerse a user into a computer-simulated world fully. This way, user's senses are stimulated by effectors of computer system leading to user's separation from reality. In order to make VE realistic, VR system needs to offer some form of known interactivity. This means that the user can affect the virtual world and objects within it and that they are able to do so in a similar way to the real world. LaValle (2017, p. 227) explains that virtual interaction mechanisms which have physical world counterparts adhere to the *Universal Simulation Principle*. Most VR systems are able to stimulate user's vision and hearing. User can move in real space which results in movement in the virtual space. Interaction with virtual objects is often achieved by physical controllers held by the user or by tracking user's pose and gestures. VR offers an intuitive interaction with virtual objects while it also allows developers of VEs to for example set different laws of physics. This was demonstrated by Ivan Poupyrev et al. (1996) with VR interaction technique that let users grow their arm to reach distant objects in VE.

Although it might seem that VR is a technology of approximately last decade, its roots can be tracked far to the history of 1970s. One of the first devices that could be considered VR HMD was presented by Sutherland (1968). It was able to track HMD in 3D space using mechanical head position sensor and continuous wave ultrasonic head position sensor. This experimental device was able to project 3D objects in form of a wire frame models using two miniature Cathode-ray Tube (CRT) displays (Sutherland 1968).

3.1 Augmented Reality and Mixed Reality

Other similar areas of HCI are Augmented Reality (AR) and Mixed Reality (MR). AR is characteristic by using additional information visualised to the user in a way that it augments the real perception of the situation. Main motivation during the beginning of AR research was the vision that the performance of human operators for remote machine-manipulated tasks could be improved by computer-generated guides. This introduced for example virtual fixtures which extend user's knowledge about perceived scene with sensory information (Rosenberg 1992; Rosenberg 1993). Concept of AR is therefore to let user interact with the real world, but enhance their perception with additional information.

On the other hand, MR systems do not aim to just enhance the perception of real world, but they focus on merging virtual and real environments into one. In MR, virtual objects are inserted (registered) into user's perception of a real scene and they are able to interact with objects from real world.

Bowman et al. (2005, pp. 391–403) provide a survey of various classes of AR interfaces. *3D data browsing* applications place 3D data as virtual object to the real world space. In this case, virtual objects are registered to specified positions or attached to real world objects. For example, State et al. (1996) presented real-time AR system for ultrasound-guided needle biopsy[1]. In this case, the system was able to merge live ultrasound data, geometric models, and stereo images of the patient in HMD. Beran (2004) focused on possible use of AR in videoconferencing systems. This project aimed to separate image of a user from processed video, map image on model, and render it. User of this system would wear HMD and rendered videoconference participants would be placed on tracked paper cards on user's desk. Inserted objects can also represent text notes and multimedia annotations (Bowman et al. 2005, p. 392). Rekimoto et al. (1995) focused on Human Real World Interaction (HRWI) when they presented *NaviCam*. This portable device with camera was able to detect coloured identifiers in real world and then display additional situation sensitive information on its video-see-through display. Experimental applications of *NaviCam* included annotating museum exhibitions, enhancing real world calendar with personal schedule, and collaboration tool where a remote instructor guides a user in a workplace (Rekimoto et al. 1995, pp. 31–33).

With *augmented surfaces* approach, virtual objects are registered only to a limited area of a selected surface, for example a laboratory desk. *Augmented surfaces* can use simplified visualisation techniques like image projection instead of HMD. Interaction with such surfaces can be realised with computer vision methods for tracking real world objects, special physical handles, and hand gestures in order to make the system intuitive (Bowman et al. 2005, p. 395). Wellner (1993) proposed *The DigitalDesk* where computer screen was projected to user's desk, cameras were tracking user's fingers and pencils, and image recognition was able to read paper documents to process their contents in the computer.

Tangible interfaces cover area of a border between AR and MR. Virtual objects are registered to their physical containers and interaction with these real world containers results in interaction with respective virtual objects (Bowman et al. 2005, p. 398). This leads to design of general purpose physical handles which can be moved in 3D space. Individual gestures can be then assigned to specific actions with selected virtual object. Billinghurst

[1]medical examination of tissue from a living body

et al. (2000) used HMD with a camera and physical cards, each card was marked with a special symbol. Computer vision detected card with its symbol and the HMD then displayed virtual 3D object placed on the physical card. I. Poupyrev et al. (2002) presented *Tiles* AR tangible interface which allowed full 3D spatial interaction with virtual object. The system maps virtual objects to physical cards (tiles) detected using computer vision as described earlier. *Tiles* system was developed for rapid prototyping of aircraft control panels, but introduced concept of *data tiles*, *operator cards*, and *menu tiles* is application-independent (I. Poupyrev et al. 2002, p. 46; Bowman et al. 2005, p. 399).

Interesting and recent research in the field of MR includes for example work by Bambušek (2018), who focused on improving spatial visualisation in a human-robot collaborative workspace. Implemented MR system enables user to see robot's programs and also guides the user during programming instructions for the robot. As explained by Mjartan et al. (2018), VR is already utilised for training and education of employees, now.

3.2 Software and Hardware Development

Many currently available and widely used consumer VR software platforms and hardware equipment systems are in detail described by Brestič (2018, pp. 4–10), Zouhar (2017, pp. 7–18), and Valenta (2018, pp. 5–10). These works cover systems such as Oculus Rift, HTC Vive, SONY PlayStation VR, Google Cardboard, Google Daydream View, and Samsung Gear VR. The process of developing a VR application in Unity game engine using C#, SteamVR, and Virtual Reality Toolkit (VRTK) is presented by mentioned authors as well. While Valenta (2018) introduces an interesting incorporation of Artificial Neural Network (ANN) for VR interaction mechanisms, Brestič (2018) proposes and demonstrates a novel and feasible way of utilizing a limited real world space for a realistic movement over much larger VE, and efforts of Zouhar (2017) were aimed mainly in the area of 3D models, textures, animations and interactive VEs.

Furthermore, Novotný (2017, pp. 18–25) not only developed an interactive VR application using Unity game engine and VRTK, but also outlined unusual use cases for VR such as virtual therapy, education, and training (Novotný 2017, pp. 13–14). Cronin (2018) reported how VR is nowadays used for training players of National Football League (NFL) in USA.

It is important to point out, that development of VR applications is not limited to technologies for desktop software. Surprisingly, modern web technologies like Hypertext Markup Language (HTML), Cascading Style Sheets (CSS), and extensions of ECMAScript (or JavaScript (JS)) are sometimes used for User Interface (UI) layer of the application. This can be achieved with libraries like for example *Coherent UI* from *Coherent Labs*[2]. It is also common to find use of other web-related technologies like *WebSockets* for full-duplex communication and *WebRTC* for real-time multimedia transport. *WebRTC* is usually accompanied by Interactive Connectivity Establishment (ICE), Session Traversal Utilities for NAT (STUN), and Traversal Using Relays around NAT (TURN).

[2] https://coherent-labs.com/

3.3 Security Concerns

The field of VR applications is a fast-paced and growing field in computer business. As with almost every modern technology, many software vendors and developers are striving for releasing their products to the market as soon as possible. Unfortunately, with this harsh environment and tight budgets, companies often do not pay enough attention to the security of their products. Products are often being released as alpha and beta versions while they are still under development. This also creates more space for security vulnerabilities which are introduced during continuous software development and which yet have not been revealed by extensive security analysis. Importance of this topic is supported by the fact that VR applications are used not only for entertainment, but also for work and social interaction where are high concerns about loss of confidentiality.

New and modern VR applications used for social interactions which are released while still in development have a high chance of containing unresolved bugs. The bugs can represent security vulnerabilities which when exploited can place users at risk. Concerning social applications, the most critical risk is considered loss of confidentiality (privacy).

The biggest advantage of VR could also be its biggest security weakness. VR is based around the point that users are fully immersed into virtual world and that they ideally have no perception of the real world around them. In a case of insufficient security of a VR system or any other component in the utilized computer system, the threat actors could aim to silently gain control over victim's VE (Chmiel 2019).

Security researchers have recently (November 2018) expressed warnings about analysed vulnerabilities in consumer VR systems. The team of UNHcFREG has devised and performed several attacks which exploit found vulnerabilities (Hennessey 2018). The *overlay attack* is able to block user's HMD sight. This can be considered an example VR ransomware rendering the VR system practically unusable. With the *disorientation attack* researchers were able to disturb the VR movement and tracking system making the users disoriented. The *chaperone attack* consists of operations which remove safety boundaries of the VR system causing the user to easily hit walls and objects in the real world. Casey et al. (2019a) combined these attacks together which resulted in a much more disturbing VR attack—the *Human Joystick Attack*[3]. With this attack, the researchers were able to slowly move the victim (the user of the VR system under attack) in the real world, but without the victim noticing any movement in space. This attack can practically gain control over the unsuspecting user (Ng 2018). Attacks like these slowly erase the boundary between real and virtual world in terms of computer security, because computer hackers and other threat actors could manage to physically hurt their victims (Yurong 2018).

Forensic analysis of most widely adopted consumer VR applications and platforms conducted by Yarramreddy et al. (2018) explored ways of reconstructing evidence from system's network traffic and also from the host computers. As a result, this research found out and mapped significant amount of forensically feasible information regarding user identities and usage details. Researchers focused on artifacts generated by the HTC Vive and the Oculus Rift systems. With the information extracted from the network and from the host computer, it is possible to reconstruct a timeline of events and interactions between individual VR systems and participants.

[3]Human Joystick Immersive Virtual Reality Attack https://www.youtube.com/watch?v=iyK94jFuniM

Consequent research carried by the team of UNHcFREG has discovered a security vulnerability which led to serious violation of user's privacy. The team has identified a front-facing camera on HMD of HTC Vive as a potential attack surface. Researchers were able to influence the system without interrupting current VR application. The camera was started and images of user's real-world surroundings were captured and exfiltrated from the host computer over the network to the attacking device. Using similar approach, researchers Casey et al. (2019a, p. 6) also managed to force capture and exfiltrate screenshots of user's computer. Meyer-Lee et al. (2018) demonstrated an attack against a widely used Mobile application for AR, which allowed the researchers to identify the location of a user interacting with AR content. This attack was developed based on findings from network traffic analysis of the AR application. Ling et al. (2019) investigated security and privacy issues of VR. Their research presented vision-based and motion sensor-based attacks which potential attacker could utilise to infer keystrokes in a VE. Casey et al. (2019b) also performed memory forensic analysis of VR systems which gave details about used devices and room setup.

Demonstrated attacks show that modern VR systems are vulnerable to losing each critical part of the Confidentiality, Integrity, Availability (CIA) triad. Main advice that Siskind et al. (2018) give to users of VR systems is to be on alert, not to download cracked software, to find a reputable source for download such as official application stores, and to make sure that host system is always up to date.

3.4 Summary

Main purpose of this chapter was to introduce technologies which are used in the area of VR applications and to reference relevant sources for further reading. At first, the concept of VE was described and it was subsequently pointed out that the idea of VR can be dated back to 1970s, when Sutherland (1968) presented one of the first HMDs. Rosenberg (1992) shown that incorporation of VE in form of VR, AR, or MR can significantly improve one's performance in various tasks. VEs are now used even for more efficient sports training (Cronin 2018). This chapter further identified key software technologies and tools used during development of VR applications. Interesting research and development works were referenced as well. Section 3.3 delivered a review of current security concerns in the field of VR. This section also focused on recently published security and forensic research in this area.

Chapter 4

Security & Forensic Analysis Focused on Virtual Reality Applications

This chapter aims to deliver background knowledge concerning penetration testing, computer security, and related forensic analysis. Information presented in this chapter will reflect focus of this research on the area of VR social applications, which are outlined in chapter 2. VR social applications in their nature are for interaction between multiple users (multiplayer), which is realised over computer network. This implies that such applications utilise computer network technologies for sharing multimedia (microphone audio, computer audio, camera video, computer screen video). Users of VR application participating together in one VE can also be illustrated as a distributed computer system where proper synchronisation of users' actions and synchronisation of VE itself is essential. Therefore, it is anticipated that applications' network traffic contains valuable traces of identities of participating users, their activities in VE, and most importantly users' social interaction and communication. As applications like Bigscreen (previously described in section 2.4) offer use cases for remote workplaces and meetings, acquiring users' communication can be a subject for privacy concerns.

Chapter 3 outlined, among other things, software technologies used in VR applications. It also included recent security concerns (section 3.3). Penetration testing is described in section 4.1. Study of various available methodologies and approaches suitable for forensic analysis and penetration testing is available in section 4.2. Overview of tools feasible for the analysis of identified technologies relevant for VR applications is delivered in section 4.3, section 4.4, and section 4.5.

Before moving forward to security analysis, it is essential to define basic terminology such as the CIA triad (Oriyano 2016, pp. 4–5):

Confidentiality
: Concept of confidentiality describes security efforts to make sure that private information is available only to the participants with adequate permissions and not to anyone else.

Integrity

Security efforts preventing unauthorised changes to the data aim to maintain the integrity of handled information. Preserving integrity can be also described as making sure that format, accuracy, trustworthiness, and interpretation of data remains the same between information producer & consumer or message sender & receiver.

Availability

Concept of system's availability ensures that data and system components are available to the users when requested. Good availability is often achieved by resource redundancy and backup systems.

4.1 Penetration Testing

One of the possible methods of security analysis is penetration testing. It can be described as a process of evaluating security of a system (not only computer-based) by conducting a controlled attack against it. Penetration testing is able to identify system's weak points and vulnerabilities. Upon discovering a vulnerability, penetration testers focus on ways how to exploit it in order to reach system's valuable assets. Result is a penetration testing report summarising exploited vulnerabilities.

As Hrozek (2010, pp. 7–8), Němec (2011, pp. 24–26), and Vrána (2011, p. 15) explain, penetration tests can be categorised from several views. During *White Box* testing, the tester has full knowledge of the targeted system. This approach includes for example availability of application's source codes and information about topology of the infrastructure. On the other hand, during *Black Box* testing, the tester is provided just with specification of the target. Special type is the *Gray Box* testing, where the tester has some, but limited information about the target. Another classification of the testing is based on possible damage caused to the evaluated system. *Destructive* tests are allowed to modify target environment to the extent that some recovery actions are required after testing. Although *non-destructive* penetration test aims to attack the system, it takes care not to disrupt running system. Therefore, such test does not violate availability and integrity or critical components, but it still analyses vulnerabilities and possible exploits. Last classification of penetration tests discussed here is based on the origin of attacks. *External* test starts outside of the evaluated system and aims to break through security barriers to the system. *Internal* test starts already inside the system. Hrozek (2010, p. 7) highlights that external tests are often combined with black box testing whereas internal tests are often carried out as white box or grey box tests.

4.2 Methodologies

There are several security analysis and penetration testing methodologies suitable for the scope of VR social applications communicating over computer networks. The Open Source Security Testing Methodology Manual (OSSTMM) guides how to perform a security test (Herzog et al. 2010). This methodology covers many aspects of security ranging from human and physical security to security of data networks. Herzog et al. (2010, pp. 36–37) also define more types of security tests like for example *Reversal* test. During this controlled test the attacker has full knowledge of the target, but the tested system or organisation has no prior information about the attack. This is also known as *Red Team Exercise*. *Data Networks Security Testing* from OSSTMM could be feasible for the security analysis of network communication and network infrastructure of VR applications. Unfortunately, some guidelines defined by Herzog et al. (2010, pp. 167–183) might be too broad and detailed for the purpose of this research.

SP 800-115 published by National Institute of Standards and Technology (NIST) covers information security assessment methodology (Scarfone et al. 2008). This document for example effectively summarises *Target Vulnerability Validation Techniques* (Scarfone et al. 2008, pp. 5-1–5-7) and defines basic penetration testing methodology consisting of four stages (Scarfone et al. 2008, pp. 5-3):

1. planning,
2. discovery,
3. attack,
4. reporting.

Scarfone et al. (2008, A-1–A-2) also highlights individual security testing techniques and suitable tools, which might be highly beneficial for this research. *SP 800-115* further describes *Post-Testing Activities*. Alsaadi et al. (2018) performed penetration testing of network communication system which was structured into following steps:

1. experimental setup (hardware specification, environment setup),
2. vulnerability scanning,
3. port scanning,
4. Man-in-the-Middle attack,
5. network protocol analysis.

Dorai et al. (2018), among other things, developed a Forensic Evidence Acquisition and Analysis System (FEAAS) and utilised methodology consisting of following phases:

1. scenario creation,
2. data acquisition,
3. data analysis,
4. inference engine construction,
5. report engine construction.

Zhang et al. (2017) delivered a forensic analysis evaluating security and privacy of Android vault applications which are intended for protecting highly sensitive and personal data of users. During this project, researchers have divided testing procedures into following phases:

1. scenario creation,
2. data acquisition,
3. case analysis,
4. tool development.

Haigh et al. (2019) worked on a forensic & security analysis of several Android cryptowallet applications. The researchers also carried out artifact analysis of information stored by examined applications. They also implemented a PoC malware which exploits discovered vulnerabilities. Utilised methodology consisted of following steps, where *creating transactions* can be interpreted as *scenarios*:

1. application setup,
2. data acquisition,
3. creating transactions,
4. analysis,
5. manipulation of an application.

Vrána (2011) focused on penetration testing of web applications, especially on discovery of Denial of Service (DoS) vulnerabilities. The thesis describes main steps of real attacks to web applications as follows:

1. performing reconnaissance,
2. scanning and enumeration,
3. gaining access,
4. escalation of privileges,
5. maintaining access,
6. covering tracks.

Vrána (2011) further emphasises that penetration testing needs to be properly documented and reproducible compared to real attack. Andreu (2006) summarises that many formalised penetration testing methodologies share a common pattern: *discovery*, *attack planning*, *attack*, *remediation*. He strongly recommends using following loose methodology and adapting it to needs of current analysis, (Andreu 2006):

1. discovery,
2. analysis of discovery,
3. launch attack simulation on target,
4. analysis of attack results,

5. documentation of results,
6. presentation of results,
7. remediation.

Overview of methodologies can be concluded with mention of Open Web Application Security Project (OWASP), which is a not-for-profit organisation with goal of improving security of software. Organisation's projects include *OWASP Testing Guide v4* which offers a detailed penetration testing framework and a web application penetration testing methodology (Meucci et al. 2014). Another project *OWASP Top 10 2017* summarises the most critical security flaws of web applications (Stock et al. 2017).

4.3 Selected Tools for Network Traffic Analysis

Security analysis of network traffic focuses mainly on the confidentiality of transmitted information. Callegati et al. (2009, pp. 78–81) and Prowell et al. (2010, pp. 101–120) have described in detail how Man-in-the-Middle (MitM) attack can be carried out by an attacker for eavesdropping on victim's confidential communication. MitM attack can be also used to intercept secured network communication. "The *MitM* refers to the situation, where the attacker's device is located in the network topology between two participants of the communication. The attacker then acts as an intermediary and the network traffic is routed through the attacking device." (Vondráček et al. 2018a, p. 65) Based on this principle, realisation of MitM attack often requires tampering topology of victim's network. Widely used network technologies like for example Dynamic Host Configuration Protocol (DHCP) (Droms 1997), Address Resolution Protocol (ARP) (Plummer 1982), Neighbor Discovery Protocol (NDP) (Narten et al. 2007), and Domain Name System (DNS) (Mockapetris 1987) suffer from weaknesses which can be exploited for MitM attack. Vondráček (2016, pp. 13–17) and Vondráček (2016, p. 20) delivered an overview of attacks focused on changing network topology such as *DHCP Spoofing*, *ARP Spoofing*, *IPv6 Neighbor Spoofing*, and *DNS Spoofing*. Vondráček (2016, pp. 17–18) also outlines available tools which can execute mentioned attacks. "A successful realization of this kind of attack allows not only to eavesdrop on all the victim's network traffic but also to spoof his communication [. . .]" (Vondráček et al. 2018b, p. 207)

Ability to decrypt otherwise encrypted traffic is extremely useful during security analysis for finding out possible information leak and for reverse engineering of application's own communication protocols. The power of MitM to spoof network messages can be utilised during security analysis for examining Bigscreen's proper validation of received data and application's behaviour based on various network messages. MitM can be also used for *replay attacks*.

There is a wide variety of tools suitable for this phase. Following points briefly describe few selected tools suitable for network traffic analysis. Please note this list is definitely not exhaustive as that is not the main purpose of this work. Furthermore, tools ranging from network sniffing to penetration testing summarised by NIST can be utilised (Scarfone et al. 2008, A-1).

1. **Scapy**[1] is a packet manipulation tool and *Python* package. It is able to capture communication from network, but also manipulate network traffic in capture files. *Scapy* can edit, forge, and also directly send analysed packets. Its interface allows for building custom packet structures and it can send invalid network frames, too.

2. The **mitmproxy**[2] can be used for interception, inspection, modification, and replay of network communication (Cortesi et al. 2010). It focuses on Hypertext Transfer Protocol (HTTP), Hypertext Transfer Protocol Secure (HTTPS), WebSockets (WS), Secure WebSockets (WSS), and generally on protocols that use Secure Sockets Layer (SSL)/Transport Layer Security (TLS). It features a flexible scripting interface for *Python*. Main benefit of *mitmproxy* for the purpose of this research is its ability to capture routed traffic and log used SSL/TLS master keys. This allows other software to decrypt and analyse communication.

3. **Framework for Man-In-The-Middle Attacks (MITMf)**[3] focuses on interception of network traffic. "[...] *MITMf* is especially important for included *HTTP* and *DNS* servers. It also offers various plugins which can be used during the attack. An available plugin called *Spoof* provides functionality to perform several attacks aimed at traffic redirection such as ARP Spoofing and DHCP Spoofing [...]" (Vondráček 2016, p. 18) However, it is currently (2019) not actively maintained and refers to **bettercap**[4] as a more feature complete tool.

4. **wifimitm**[5] is a fully automated Command-line Interface (CLI) tool and a *Python* package for realisation of MitM attack on Wireless Local Area Networks (WLANs). It incorporates and manages controlled execution of several penetration testing tools (e.g. MITMf, *Wifiphisher*[6], *Aircrack-ng suite*[7], ...).

5. **SSLsplit**[8] is a penetration testing tool which is capable of executing MitM attack against SSL/TLS. When a client (victim) tries to create an encrypted connection to a server, connection is redirected to *SSLsplit*, which is located between client and server. The tool is able to generate forged certificates and establish connection with client (victim) and another connection with server. This allows *SSLsplit* to log all transmitted data (Marušic 2016, pp. 13–14)

6. **Wireshark**[9] is a network protocol analyser focused on the view on individual network frames. It can directly capture communication from many different types of interfaces. It supports not only network traffic analysis of saved capture files, but also analysis during live capture. *Wireshark* also supports traffic decryption of several protocols including SSL/TLS. Its main strength is in enormous amount of protocol dissectors and in a possibility to write own dissector plugins if needed. *Wireshark* can be considered a traditional and almost essential tool for network protocol analysis.

[1] https://scapy.net/
[2] https://mitmproxy.org/
[3] https://github.com/byt3bl33d3r/MITMf
[4] https://www.bettercap.org/
[5] https://mvondracek.github.io/wifimitm/
[6] https://wifiphisher.org/
[7] https://www.aircrack-ng.org/
[8] https://www.roe.ch/SSLsplit
[9] https://www.wireshark.org/

7. **Netfox Detective**[10] is a forensic analysis tool which focuses on extraction of evidence from captured network traffic (Pluskal et al. 2015). *Netfox Detective* targets more on traffic analysis from the view of individual network conversations and transmitted data objects than on individual frames. However, examination of frames is also possible. This tool has a wide range of parsers and visualisations for different application protocols. *Netfox Detective* is able to decrypt SSL/TLS, too. One of its advantages is that it implements advanced techniques for reconstruction of incomplete network data as presented by Matoušek et al. (2015).

8. **NetworkMiner**[11] is a network forensic analysis tool which is also focused mainly on transmitted data objects and network artifacts. It is able to perform Operating System (OS) fingerprinting based on the analysed traffic. *NetworkMiner* can be used to directly capture from network interface or it can load content of capture files. *NetworkMiner*'s significant advantage is in its ease of extracting files and certificates from the captured traffic. It also supports decapsulation of several protocols like for example Generic Routing Encapsulation (GRE)Hanks et al. 1994. It is available as free or professional edition, where free edition has a limited set of features, but it is still very useful for quick overview of captured traffic.

4.4 Selected Tools for Reverse Engineering Focused on C# and *.NET*

Chapter 3 identified software tools and technologies widely used for development of popular VR applications. These include C# programming language and generally *.NET* environment. Many applications utilise Unity game engine and OpenVR, SteamVR, or OculusVR for software abstraction of a VR equipment. VRTK was presented as a highly beneficial library for VR developers.

Knowledge of application's internal logic and communication protocols can significantly improve conducted security and forensic analysis. Furthermore, ability to alter the logic and then employ patched version of the application with appropriately changed behaviour can help to discover additional vulnerabilities. For these reasons, this section is focused on reverse engineering of VR applications.

Following points deliver an overview of selected tools for reverse engineering, which are focused on C# language and *.NET* environment:

1. The **de4dot** is a *.NET* assembly deobfuscator and unpacker. There are 19 obfuscators/packers supported[12] by de4dot at the moment (January 2019). It utilizes *dnlib* for manipulation with assemblies.

   The dnlib is a *.NET* library intended for reading and writing *.NET* assemblies. It is developed by Github user *0xd4d* under MIT licence and it is available in the form of a NuGet package[13] or directly from its repository[14].

[10] http://netfox.fit.vutbr.cz/About.en.html
[11] https://www.netresec.com/?page=Networkminer
[12] https://github.com/0xd4d/de4dot#supported-obfuscatorspackers
[13] https://www.nuget.org/packages/dnlib/
[14] https://github.com/0xd4d/dnlib

The tool is available as a standalone CLI application, but also in the form of plugins for other tools like *Reflexil* and *JustDecompile*, which are outlined later in this section. The de4dot is also developed by GitHub user *0xd4d* and contributors under GNU General Public License (GPL) version 3 and it can be downloaded from its repository[15].

2. The **Reflexil** is a *.NET* assembly editor which is built with *Mono.Cecil*[16] (Lebreton 2017). Cecil is a library which provides ways to analyse, modify and generate *.NET* assemblies by manipulating Common Intermediate Language (CIL) source code. Lebreton (2013) illustrated that Reflexil also supports direct C# and Visual Basic .NET (VB.NET) code injection. The editor utilises *de4dot* for deobfuscation and unpacking of assemblies.

   Reflexil is available in form of a plugin for several *.NET* decompilers. It is developed by Sebastien Lebreton under GPL version 3 licence and it can be downloaded from its repository[17]. Graphical User Interface (GUI) of Reflexil plugin is shown in appendix C and appendix E.

3. **ILSpy** is a *.NET* assembly decompiler engine and a GUI application for decompiling and browsing assemblies. This engine is often used as a plugin or extension for other tools focused on reverse engineering and *.NET* decompilation. ILSpy is able to decompile assemblies and show resulting source code in form of C#, CIL, or a combination of CIL with C#. This engine also supports decompilation of Binary Application Markup Language (BAML) files into Extensible Application Markup Language (XAML) format. BAML is a binary-compiled form of XAML (*WPF Globalization and Localization Overview* 2017). Functionality of ILSpy can be extended with plugins, for example with the *Reflexil* assembly editor.

   ILSpy is developed by *SharpDevelop Team*, it is distributed under the MIT license and can be downloaded from its repository[18]. Latest ILSpy release is version 4.0.0.4521 (January 2019). However, latest Reflexil 2.2 is not compatible with ILSpy 4, only with ILSpy 3.2. View of the ILSpy GUI shows appendix C.

4. The **JetBrains dotPeek** is a *.NET* decompiler and assembly browser. It is based on a decompiler incorporated in *ReSharper*[19], but it is available as a free standalone tool. The dotPeek is able to decompile a *.NET* assemblies into C# or CIL. Decompiled source code can be exported as a project for *Microsoft Visual Studio* Integrated Development Environment (IDE). Additionally, dotPeek can decompile XAML UI definitions. Another useful feature of dotPeek is the ability to analyse and decompile *.NET* assemblies of currently running processes. It can also create an assembly dependency diagram with references between selected assemblies. This tool is developed by JetBrains s.r.o.[20] and can be downloaded from its website[21]. Screenshot of dotPeek's GUI is presented in appendix D.

[15] https://github.com/0xd4d/de4dot
[16] https://github.com/jbevain/cecil/
[17] https://github.com/sailro/Reflexil
[18] https://github.com/icsharpcode/ILSpy or https://ilspy.net
[19] https://www.jetbrains.com/resharper/
[20] https://www.jetbrains.com/company/
[21] https://www.jetbrains.com/decompiler/

5. The **Progress® Telerik® JustDecompile** serves as a decompiler focused on *.NET* software framework. It is able to decompile and analyse assemblies. Resulting source code can be exported to development projects. The tool itself does not offer source code editing, however an *Assembly Editor Plugin* for JustDecompile is available. This plugin is a ported version of Reflexil the .NET assembly editor. Unfortunately, the *Assembly Editor Plugin* can edit CIL, but direct C# source code editing functionality is limited. Another JustDecompile's plugin provides a wrapper for *de4dot*—*.NET* deobfuscator and unpacker. Screenshot of the JustDecompile tool with ported version of Reflexil is presented in appendix E.

   The tool is developed by *Progress Software Corporation* and it uses *JustDecompile Engine* which is licensed under the Apache License version 2.0. The JustDecompile can be downloaded from its homepage[22].

6. The **dnSpy** provides a wide range of features for *.NET* development and reverse engineering. It can be used as a *.NET* decompiler, debugger, and assembly editor. It is able to debug *.NET Framework*, *.NET Core* and *Unity* game assemblies—original source code is not required. Upon examining an assembly, all its metadata can also be edited, it can also read obfuscated assemblies. When dnSpy successfully decompiles an assembly, it is able to export the source code to a project for further development or reverse engineering. However, a feature that proved itself a very useful one, was the ability to directly edit the source code and recompile it in place. The tool has UI similar to *Microsoft Visual Studio* which makes it very easy to use. Screenshot of dnSpy's main window is shown in appendix F.

   It is developed mainly in C# language by GitHub users known as *0xd4d* and *yck1509* (*Ki*) and other contributors. The dnSpy utilises *ILSpy* decompiler engine and the *Roslyn* compiler. The tool is licensed under GPL version 3 and can be obtained from its repository[23].

4.5 Selected Tools for Deobfuscation and Analysis of JavaScript

Chapter 3 helped to identify that VR applications can use web technologies like JS, HTML, and CSS for their UI. Minification, packing, and obfuscation of JS source code can sometimes be seen in web applications.

Minification aims at reformatting the source code to a much smaller version while maintaining the same functionality. It can be achieved by removing all comments and unnecessary whitespaces, new lines, and delimiters. Although minification makes the source code less readable, its purpose is solely to compress the code while still keeping it executable without any decompression preprocessing. Some JS minifiers also rename variables and optimise blocks of the code, however, this functionality is on the border between minification and obfuscation. Notable JS minifiers are outlined in appendix G.

On the other side of this field are tools called source code **formatters and beautifiers**. These tools are able to rearrange code structures according to a set of rules or code style

[22]https://www.telerik.com/products/decompiler.aspx
[23]https://github.com/0xd4d/dnSpy

guides. Source code formatters are often incorporated into IDEs and source code editors, as for example the *Prettier*[24] formatter. Tools like *JSLint*[25] and *JSHint*[26] inspect JS source code for violations of specified coding rules.

Obfuscation is a process of transforming source code structures so that the functionality stays unchanged, but the code is extremely hard to read by human. Obfuscation often includes techniques which make even debugging or any other analysis of the code very complicated. Some tools not only obfuscate the original source code, but also mix it with code blocks which have no real functionality just to make the resulting code more unintelligible. The benefit of obfuscation tools is that it is significantly harder to reverse engineer the software or even know its intentions & programming logic before executing it. Software developers therefore tend to use obfuscation in attempts to protect application's proprietary source code. Similarly, malware developers can use obfuscation in an effort to prevent detection of their malicious software. However, it is important to keep in mind that obfuscation does not make reverse engineering impossible, but just significantly harder. It is considered *security through obscurity*. According to Montfort (2008), programming methods related to source code obfuscation started to appear in the beginning of the 1980s. And as Mateas et al. (2005) interestingly explains, there have been many contests on writing obfuscated software in the computer science community. Notable tools for JS obfuscation are outlined in appendix G:

Another method of processing JS source code utilises tools called **packers**. They are similar to minifiers and obfuscators. Packers often preprocess their input with minification techniques. Packers aim to reduce the size of the software by remapping (encoding) its source code. Packed source code usually cannot be directly executed, but needs to be unpacked (evaluated) first and then it can be executed. This means that packed code is also hard to read and analyse. Main difference between obfuscators and packers is that packed source code can be usually easily unpacked to a more readable form with *unpacking* tools which are often incorporated in formatters and beautifiers. Popular packer tool is the *Packer*[27] by Dean Edwards which encodes the source code using *Base62* encoding.

Many JS formatters offer some features for deobfuscation, but unfortunately with varying quality of the resulting source code. Following points summarise few selected **tools for reverse engineering and deobfuscation of Javascript language**:

1. The **JS Beautifier**[28] serves as a JS formatter, but includes detection of packed source code and feasible unpacker, too. *JS Beautifier* also has deobfuscation features. It is able to unescape printable characters which are written using hexadecimal or unicode escape sequence.

2. **JSNice**[29] is an online tool for statistical renaming, type inference and deobfuscation of JS code. It also incorporates detectors of JS packers and is able to unpack such code. *JSNice* predicts and renames minified names of variables and parameters based on its statistical model[30].

[24] https://prettier.io/
[25] https://www.jslint.com/
[26] https://jshint.com/
[27] http://dean.edwards.name/packer/
[28] https://github.com/beautify-web/js-beautify or https://beautifier.io/
[29] http://jsnice.org/
[30] http://www.nice2predict.org/

3. The **Packer**[31], which is mainly intended for packing JS code, can be also used for unpacking. Unlike *JS Beautifier*, the *Packer* is able to repeatedly unpack code which was packed multiple times. Unfortunately, it has troubles unpacking code that was packed, combined with another source code and then packed again.
4. The **de4js**[32] is a JS deobfuscator and unpacker. Its big advantage is that it incorporated several different unpackers. The tool supports automatic mode, when it repeatedly tries its unpacking and deobfuscating techniques. This means that the user does not have to select the right unpacker and that the tool is able to unpack code which was packed for multiple times.
5. **ESDeobfuscate**[33] is a PoC JS Abstract Syntax Tree (AST) deobfuscator based on partial evaluation. Using this approach, functions and expressions which are evaluated to constant values are replaced in the source code with given constant values.
6. **JStillery**[34] is a JS formatter and deobfuscator which utilises approach of partial evaluation of source code (Paola 2015). It was initially developed for analysis of JS malware samples and for Web Application Firewall (WAF) analysis of Cross-site Scripting (XSS) payloads.
7. The **dCode Javascript Unobfuscator**[35] is available as online interface for deobfuscation and formatting of JS source code. It can convert escaped ASCII sequences back into directly written symbols, but other deobfuscation functionality is quite limited.

As there are numerous different obfuscation methods and JS malware *„in the wild“* comes up with many more variants, it is expected that the analysts must use multiple reverse engineering tools and sometimes even develop their own with custom deobfuscation logic (Heyes 2016; Hung 2017; Palladino 2012).

4.6 Summary

This chapter summarised background knowledge in terms of forensic and security analysis of VR applications. Section 4.1 focused on different approaches to penetration testing. Various methodologies useful for this research were outlined in section 4.2. This included for example OSSTMM, *SP 800-115*, *OWASP Testing Guide v4*. Subsequent section 4.3 delivered an overview of tools useful for interception, decryption, and analysis of network traffic. For example, *mitmproxy* be utilised to intercept application's network traffic. *Wireshark* is able to analyse captured network traffic from the perspective of individual frames, whereas *Netfox Detective* and *NetworkMiner* focus more on network conversations and data objects. Next section 4.4 is a more detailed study of available tools for Reverse Engineering (RE) of *C#/.NET* (e.g. very powerful tool *ndSpy*). And section 4.5 covered areas of JS minification, formatters & beautifiers, numerous methods of obfuscation, and RE & deobfuscation.

[31] http://dean.edwards.name/packer/
[32] https://github.com/lelinhtinh/de4js or https://lelinhtinh.github.io/de4js/
[33] https://github.com/m1el/esdeobfuscate
[34] https://github.com/mindedsecurity/jstillery/
[35] https://www.dcode.fr/javascript-unobfuscator

Chapter 5

Analysis of the Bigscreen Application

This chapter delivers an extensive security analysis of the Bigscreen VR application, which was briefly described in section 2.4. This chapter also builds upon knowledge from chapter 3. Methodologies, approaches, and tools were selected according to chapter 4. This chapter starts with definition of research questions in section 5.1 and definition of selected methodology for the analysis in section 5.2. Next section 5.3 provides reasoning for selection of the Bigscreen application. Scenarios which are used during the analysis and testing are described in section 5.4. Subsequently, the practical part of the analysis was started in section 5.5 and continued in individual phases (section 5.6, section 5.7, section 5.8, section 5.9). With identified security weaknesses, the research focused on development of exemplary exploits (section 5.10) and their incorporation into single attacking tool (section 5.11). Testing is covered in section 5.12 and findings are then outlined in section 5.13. During this analysis, a way how to perform and implement novel Man-in-the-Room (MitR) attack was discovered, this is explained in section 5.14. Section 5.15 then concludes this chapter with mitigations & suggestions.

5.1 Research Questions

This work set up several research goals that this forensic and security analysis aims to achieve. This helped to formulate research questions as follows:

1. Are popular social VR applications properly secured and do they maintain CIA?
2. Can first PoC of novel MitR attack be implemented in current VR applications, as it was anticipated by research of Casey et al. (2019a) and Yarramreddy et al. (2018)?

5.2 Methodology

The study of numerous approaches, phases, and methodologies concerning forensic and security analysis delivered in section 4.2 helped to identify a common pattern. As suggested by Andreu (2006), this analysis was not too strictly following general penetration testing methodologies. Final methodology for the analysis was strongly inspired by structures mentioned in section 4.2, but the phases were adjusted to better fit current case of VR applications. Defined phases therefore cover analysis of forensic artifatcs, security analysis including penetration testing which has some tasks similar to real attack, development of a security tool for demonstrating findings, and evaluation of the results. Based on the classification of penetration testing approaches outlined in section 4.1, it is possible to describe this analysis as follows:

- *external* origin of the analysis,
- *black box* penetration testing,
- limited to *non-destructive* attacks to the production environment,
- considering *ethical* actions,
- carried out in a *controlled environment.*

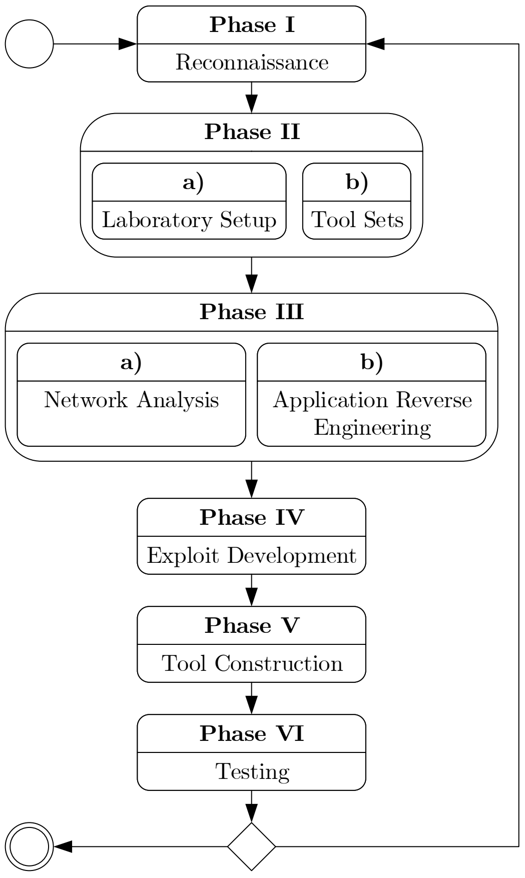


Figure 5.1: Security analysis of the Bigscreen application was divided into several phases. Individual phases were inspired by studied methodologies in section 4.2 and are described in section 5.2. The whole process was iterative. This figure represents a state diagram of the phases.

Phases of carried out analysis are illustrated in fig. 5.1. The first phase *Reconnaissance* focuses on gathering publicly available information about Target of Evaluation (TOE) and its context. This phase also includes initial *passive* and *non-destructive* examination of TOE. After gathering information, it is possible to proceed to preparation. This step consists of two phases *Laboratory Setup* and *Tool Sets*. Previous knowledge of VR technologies and results of reconnaissance were utilised to effectively prepare laboratory equipment (section 5.6) and prepare useful tools section 5.7. Tools were selected from the survey in section 4.3, section 4.4, and section 4.5. With controlled testing environment prepared, practical part of the analysis could begin with phases *Network Analysis* (section 5.8) and *Application Reverse Engineering* (section 5.9). These steps identified vulnerabilities in the Bigscreen application and in the Unity engine. Next phase *Exploit Development* (section 5.10) focused on demonstration of impact of discovered vulnerabilities, while phase *Tool Construction* (section 5.11) documents development of demonstrative attacking tool which incorporated created exploits. Last phase of the analysis *Testing* (section 5.12) evaluated feasibility of attacks and tested functions of developed demonstrative attacking tool. The whole forensic and security analysis was iterative.

5.3 Target of Evaluation Selection

The assumption for security analysis is following. New and modern immersive virtual reality applications used for social interactions which are released while still in development have a high chance of containing unresolved bugs. These bugs can represent security vulnerabilities which when exploited can place users at risk. Concerning social applications, the most critical risk is considered loss of confidentiality (privacy).

Based on the stated assumption and a recent forensic analysis by Casey et al. (2019a) and Yarramreddy et al. (2018), Bigscreen application was chosen as TOE for extensive security analysis. The Bigscreen application is described in section 2.4. The expectations were that should there be any security vulnerabilities found, responsible disclosure can mitigate the risks for mentioned *more than* $500,000$ *users*.

5.4 Scenarios

As practically demonstrated by Zhang et al. (2017), Dorai et al. (2018), and Haigh et al. (2019), clearly defined scenarios significantly help to maintain systematic approach during the analysis and testing. Therefore, this analysis defines several scenarios as well. These were later used for network traffic capture in section 5.8, RE of Bigscreen's communication protocol and mapping of individual network messages to their meaning in application's context. Implemented attacks exploiting discovered vulnerabilities and developed attacking tool were also evaluated based on these scenarios (section 5.12). Defined scenarios include Alice and Bob who are two legitimate users of the Bigscreen application. Then there are attackers called Mallory and Trudy. Names are assigned according to a common naming convention for scenarios and do not refer to real persons. Main elements participating in scenarios are illustrated in fig. 5.2.

Passive stay in the lobby Legitimate user Alice starts the Bigscreen application and enters the lobby which is the main screen of the application. Alice waits until the application shows a list of all public rooms. Alice waits passively in the lobby for several seconds and then she decides to terminate the application.

Created public room Alice opens the application and she is presented with a list of public rooms in the lobby. Alice creates & joins her own room, which is configured as public. She spends several seconds in the VR room. After that she leaves her room and terminates the Bigscreen application.

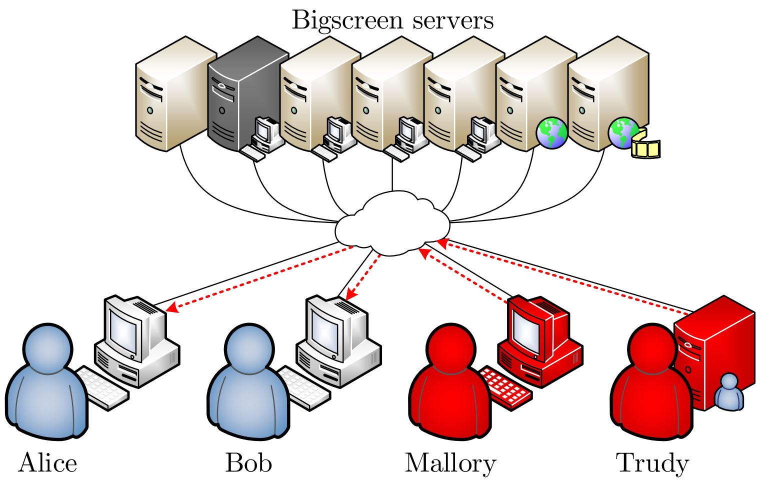


Figure 5.2: Basic scenario for attacking the Bigscreen application. Alice and Bob are legitimate users of the application, each in a different location. Mallory is an attacker with maliciously patched Bigscreen application. Trudy is an attacker with developed Command & Control (C&C) server capable of attacking Bigscreen users and controlling created botnet. Mallory and Trudy aim at users of the application and do not attack Bigscreen servers.

Created private room This scenario is similar to previous one. Alice starts the application, enters the lobby and then she creates & joins her room. The room is this time configured as private. Alice similarly spends several seconds in her private room and then she leaves the room and terminates the Bigscreen application.

Private meeting This scenario involves both Alice and Bob. Alice starts the application and waits in the lobby for few seconds. Bob starts his application and enters the lobby, too. Alice creates & joins her private room. Alice wants Bob to join her private room. In order to do so, she sends him her confidential private room ID. This room ID is sent using some separate and secured channel. Bob joins the room using received room ID. They both interact in VR room for several seconds and also exchange few chat messages. Later both users leave the room and terminate their Bigscreen applications.

Transition between rooms Alice starts the Bigscreen application, enters the lobby, list of public rooms is loaded, and she creates & joins her public room. Bob then does the same procedure which means that he also creates & joins his public room. Alice spends some time in her public room alone. She then leaves her room, enters lobby, and joins Bob's room which is public. They similarly interact in VR room and after few seconds they leave the room and they both terminate their applications.

5.5 Phase I – Reconnaissance

The Bigscreen application was briefly introduced in section 2.4, however, this phase focuses on reconnaissance in order to gather information useful for subsequent phases of the analysis.

The VE of Bigscreen is based on the concept of a virtual room, where users can create and join VR rooms to interact with others. VR movie theatres allow selected user to stream video from their computer for other room participants to watch. Room participants can enable sharing their computer screens, computer audio, and microphone audio inside the room. Users can further use 3D markers for drawing in a 3D space and they can send text messages (chat), too. Each participant has their name and avatar—visualisation of their head, torso, and hands in VR space (fig. 5.3).

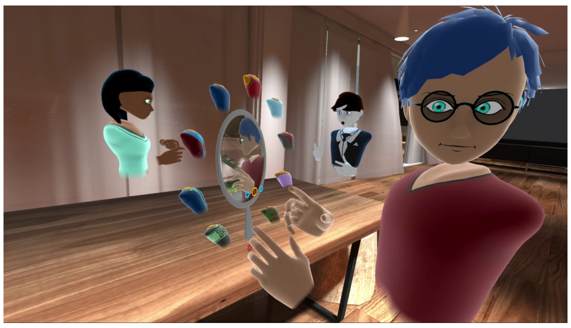


Figure 5.3: Users of the Bigscreen application are in VR room represented by their 3D avatar. Bigscreen, Inc. 2018

Each created room has a name, designated category (Chat, Movies, Games, Productivity, NSFW), and an optional text description. Room admin can choose room's 3D model and textures from a wide variety of environments ranging from campfire, balcony, bedroom, theatre, to even space panoramas. Every room can be created as either *4-player Room*, or *Big Room*, which determines maximum number of players and screen sharing rules. Room can exist as *public*, or a *private/invite-only*. Any user can join any public room and they are listed on application's main screen, as illustrated in appendix A. Code of Conduct displayed in the application (appendix B) states that inappropriate behaviour in public space is penalised by permanent bans from public rooms. This Code of Conduct further states that private rooms are not moderated and that even the developers themselves can not find out what is happening inside such private rooms, because the communication is peer-to-peer and encrypted. Room's admin can control its *locks*—limitations of room participants for streaming and 3D drawing. Upon creation of a room, it is assigned a unique *Room ID* in form of 8 alphanumeric characters presented like for example `room-9hckep83`. Private rooms can be joined using their confidential *Room ID*, no further authorisation is required. Public rooms can also be joined by *Room ID*, but since all public rooms are listed on main screen, their *Room ID* is not considered confidential. However, shall the private *Room ID* be revealed to an unauthorised threat actor, they can join given private room.

Previous text provided an overview of Bigscreen from the user's point of view. The next step is to focus on publicly available information that could reveal inner parts of the Bigscreen system. Therefore, currently opened software engineering job offers from the Bigscreen company[1] were analysed. The company is looking for *Unity3D* developers experienced with *C#* and *C/C++*. Next positions are for web developers experienced with *Node.js* and *AFrame*, *WebVR*, or *WebGL*. It also mentions technologies like HTML, CSS, JS, *jQuery*, *React*, *Redis*, and *WebRTC*. Another offer for the video streaming and networking specialist further describes that Bigscreen's codebase is focused on *C/C++*, *C#*, and *Java*. This offer also explains that Bigscreen's peer-to-peer networking uses *WebRTC* and backend servers utilise *Node.js*. Bigscreen's blog also contains information about application's updates and posts about their development[2].

Yarramreddy et al. (2018) identified several artifacts left on computer's hard drive by the Bigscreen application. For example, application's debug logs have confirmed that ICE and STUN are utilised for peer-to-peer data transfer. This research also identified that analysed version of the application used *WebSocket* protocol for full-duplex communication with Bigscreen's servers. However, previous research from December 2017 examined Bigscreen Beta at version *0.23.0* (Yarramreddy et al. 2018, p. 3), whereas this thesis presents analysis finished in November 2018 which was focused on Bigscreen Beta version *0.34.0*. Therefore it is important to keep in mind that Bigscreen's codebase and protocols have changed since version *0.23.0*.

The reconnaissance also examined that application's data are stored on computer's hard disk in two main locations. Location `AppData\LocalLow\Bigscreen, Inc_\Bigscreen` contains files with information about application's crashes and analytics. It also contains interesting `output_log.txt`, `cookies.dat`, and *localstorage* files of *SQLite* Relational Database Management System (RDBMS). The *localstorage* has keys like for example: `uuid`, `my-room-private`, `my-room-name`, `hotkey-screenshot`, and `banned`. Second identified location at `Steam\steamapps\common\Bigscreen` holds application's executable, libraries, and data files. Folder `Bigscreen_Data` still contains generated `output_log.txt` and `output_log2.txt` files, as mentioned by Yarramreddy et al. (2018, p. 5). Further examination of the mentioned directory structure revealed that `Bigscreen_Data\uiresources` folder contained not only Bigscreen's UI files, but also testing and development example files for UI. Based on these artifacts, Bigscreen application apparently utilises *Coherent UI*/*Coherent GT* for realisation of UI layer which is implemented using HTML, CSS, and JS. With this approach, application's UI elements are controlled by JS in a limited built-in web browser environment. However, individual logic parts of UI layer use bindings to the application's core layer.

This reconnaissance phase resulted in an overview of technologies which are present in Bigscreen's codebase. The Bigscreen application is implemented using Unity engine, which has really broad userbase (table 5.1). This gives a set of expected technologies and protocols, so that subsequent penetration testing can focus on searching for their weak points and vulnerabilities. This phase also provided valuable insight into the application's usage and logic.

[1] `https://bigscreenvr.com/careers/`
[2] `https://blog.bigscreenvr.com/`

Table 5.1: Userbase

Software	Reach
Bigscreen Beta	over 500,000 users[3]
Unity	3,000,000,000 devices[4]

5.6 Phase IIa – Laboratory Setup

Experiments during this research were carried out in the same controlled VR laboratory environment as previous works by (Yarramreddy et al. 2018; Casey et al. 2019a). Laboratory workstations had *Steam* and *Oculus* installed. Details of the hardware and software equipment are available in table 5.4, table 5.3, table 5.2. User accounts specially for the purpose of testing in the laboratory were created. Both computers were connected to the same LAN and then to the Internet (fig. 5.4). This setup was used for initial experiments with the Bigscreen application according to scenarios defined in section 5.4. However, subsequent experiments and analysis required changing network topology.

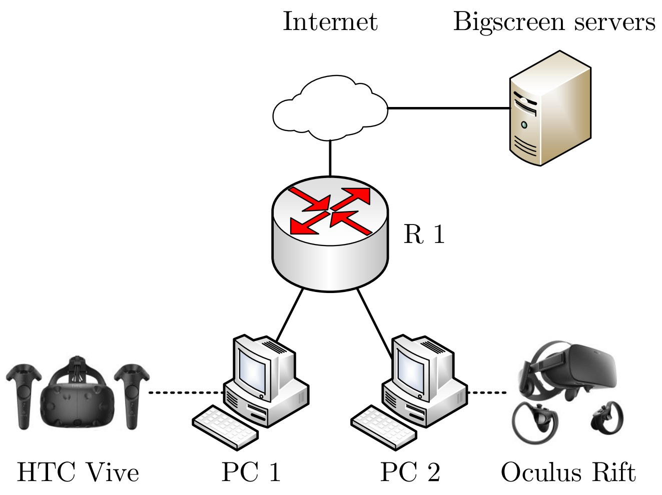


Figure 5.4: Laboratory setup for initial experiments based on scenarios defined insection 5.4

Table 5.2: Applications for VR used during the analysis. *Table is based on Yarramreddy et al. (2018) and Casey et al. (2019a), but updated.*

Application	Version
Bigscreen Beta	0.34.0
Oculus App	1.36.0.215623
Steam	1549129917
SteamVR	1.2.10

[3] https://bigscreenvr.com/press/
[4] https://unity3d.com/public-relations

Table 5.3: Hardware equipment for VR applications which is available in the laboratory and which was used during the analysis. *Table is based on Yarramreddy et al. (2018) and Casey et al. (2019a), but updated.*

Device	**Component**	**Firmware**
Vive	Headset Vive MV HTC	1462663157
	Base HTC V2-XD/XE	436
	Base HTC V2-XD/XE	436
	Controller MV HTC (x2)	1533720215
Rift	Headset Rift	709/b1ae4f61ae
	Sensor (x2)	178/e9c7e04064ed1bd7a089
	Left Touch	f3c65f7a5f
	Right Touch	f3c65f7a5f

Table 5.4: System details of computers in the laboratory. *Table is based on Yarramreddy et al. (2018) and Casey et al. (2019a), but updated.*

Device	**Details**
Processor	Intel Core i7-6700 CPU
System Type:	64-bit OS, x64 processor
Graphics Card	NVDIA GeForce GTx 1070
Manufacturer	iBUYPOWER
Installed Memory (RAM)	8.00 GB
Operating System	Windows 10 (10.0.0.17134)

5.7 Phase IIb – Tool Sets

This phase focused on preparation of tools for following phases. The tools are intended for use inside the controlled laboratory environment, which was prepared in section 5.6.

Section 5.5 pointed out several technologies that the analysis most probably needs to deal with. Tools outlined in section 4.3, section 4.4, and section 4.5 were prepared for following *Phase III* which includes *Network Analysis* (section 5.8) and *Application Reverse Engineering* section 5.9. IDE and development tools were also prepared in this step in case it would be necessary to write own tools or automate existing software.

5.8 Phase IIIa – Network Analysis

This section outlines a security and forensic analysis of Bigscreen's network traffic and a penetration testing of the application from the network side. **However, this text directly presents only successful paths and results**.

MitM Attack to HTTPS & WSS Initial network traffic analysis was carried out in the laboratory LAN topology from fig. 5.4. The scenario of the experiment included usual use of Bigscreen application like creating own rooms, joining other rooms, and interacting with other room participants (section 5.4). The *Wireshark* was initially used to capture the traffic. Even though the traffic capture contained some unencrypted Representational

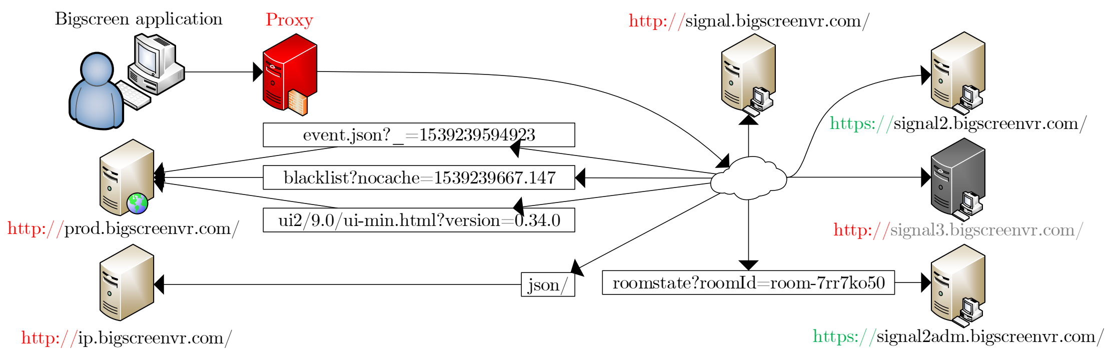

Bigscreen application
Proxy
http://signal.bigscreenvr.com/
https://signal2.bigscreenvr.com/
event.json?_=1539239594923
blacklist?nocache=1539239667.147
ui2/9.0/ui-min.html?version=0.34.0
http://prod.bigscreenvr.com/
http://signal3.bigscreenvr.com/
json/
roomstate?roomId=room-7rr7ko50
http://ip.bigscreenvr.com/
https://signal2adm.bigscreenvr.com/

Figure 5.5: Mapped network infrastructure of Bigscreen servers.

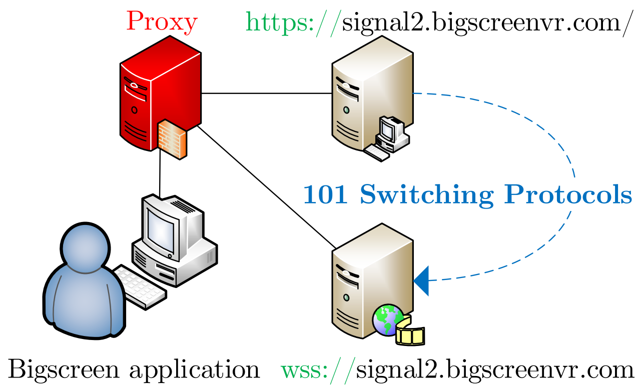

Proxy
https://signal2.bigscreenvr.com/
101 Switching Protocols
Bigscreen application
wss://signal2.bigscreenvr.com

Figure 5.6: WSS connection for the signaling channel is established by first requesting an HTTPS endpoint and then switching to WSS protocol.

State Transfer (REST) Application Programming Interface (API) calls via HTTP, most of the traffic generated by the application was encrypted.

A connection through created proxy was therefore established for intercepting encrypted communication (fig. 5.5). The *mitmproxy* tool was used for this step (section 4.3). Own Certificate Authority (CA) was configured and the laboratory computers with the application trusted it. Both communication channels (i.e. application-proxy and proxy-servers) were still encrypted, but the proxy was able to log SSL/TLS master keys. This way, it was possible to decrypt HTTPS and WSS traffic. Except for HTTP, HTTPS, WSS flows, the traffic also included still encrypted Datagram Transport Layer Security (DTLS) channels.

Infrastructure Mapping Throughout the network analysis phase, the application's network communications were monitored which allowed to create a map of Bigscreen's network infrastructure (fig. 5.5).

REST API Analysis By analysing ongoing traffic between the application and its servers, it was possible to gain knowledge of individual REST endpoints. The `/json` at `ip.bigscreenvr.com` provides the client application with their Internet Protocol (IP) address and geolocation information. The `/roomstate` endpoint at `signal2adm` subdomain serves information about any room specified by its *Room ID*. The endpoint at `signal2` subdomain switches from the HTTPS to the WSS protocol, which is then used as a connection point for the signaling channel (fig. 5.6). API at subdomain `signal3` is out of order. Subdomain `prod` has three endpoints: `/event.json` provides information about active community events, `/blacklist` replies with a list of banned users, and `/ui2/9.0/ui-min.html` serves the main UI file which then requests the others.

Following weaknesses were detected. The `/roomstate` endpoint can be used to find out whether room (public or private) with given *Room ID* exists. Therefore, private *Room IDs* could be potentially brute forced as this endpoint apparently has *no request limits*. Endpoints at subdomains `prod`, `signal`, and `ip` are used without authentication and without encryption. Endpoints at `signal2` and `signal2adm` use encrypted communication, but without authentication. Another finding is that the Bigscreen application updates its UI by downloading it from the server using HTTP. Should the attacker get into the MitM position for the victim, they can spoof the UI files during download. Banlist including *uuid*, *reason* and *username* is publicly available.

UI Layer Analysis Next task was to focus on the analysis of UI files which were being downloaded by the application from the server. With the knowledge of file hosting at `prod` subdomain, it was not necessary to intercept the UI files during download, but it was possible to download them manually from respective Uniform Resource Locators (URLs). The UI files represent JS environment which communicates with Bigscreen's C# core layer locally via JS-C# function bindings. This UI layer also communicates with Bigscreen's servers using WSS. Obtained JS source code was obfuscated and minified. After extensive analysis and with a significant effort, RE of key parts was achieved.

With progressing knowledge of Bigscreen's UI layer, it was possible to send control messages to the signaling server from the JS UI environment. However, prior to that, it was essential to find out how to spoof the Unity version number for checks in the UI logic. It was also discovered how to enable UI's debug mode.

During analysis of JS-C# bindings, a suspicious behaviour of JS function bound from the UI layer to the core C# layer was encountered. Calling `Unity.openLink(url)` in JS UI layer calls `engine.call(„OpenLink", url)` which is bound from JS to C# to the method `Assets.Scripts.Helpers.UrlWrapper.OpenLink(string url);` which calls Unity scripting API method `Application.OpenURL(url);`. Detailed analysis of its behaviour led to discovery of a critical security vulnerability in the Unity scripting API. This method *OpenURL* is dangerously capable of running programs, opening folders and files on the host computer. This method can be also used to automatically download and execute any payload (e.g. malware) on the host computer.

Signaling Protocol Reverse Engineering Next task was focused on WSS communication (signaling channel). This communication was examined from two points of view: decrypting proxy on the network (fig. 5.7), reverse engineered UI layer. Analysis of transported binary messages resulted in a finding that Bigscreen application utilises *MessagePack* for serialisation of JavaScript Object Notation (JSON) signaling messages into binary for

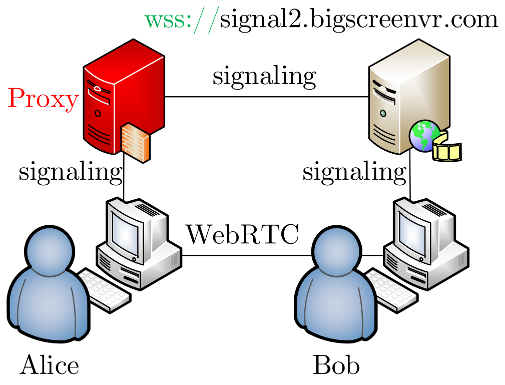


Figure 5.7: WSS traffic used for Bigscreen's signaling channel was successfully decrypted with MitM attack executed between the application and the signaling server.

transmission over the network. By monitoring and study, it was possible to reverse engineer Bigscreen's signaling protocol and assign meaning to its individual messages.

This signaling channel is used to manage VR rooms. It is also used for transport of ICE & STUN information and Session Description Protocol (SDP) messages to create peer-to-peer *WebRTC* connections. After successful negotiation, *WebRTC* audio, video, and data channels are created; they are established over secure DTLS (fig. 5.7). The signaling protocol is therefore used for example to poll the server for a list of latest rooms, join room & create room (signaling group), kick any user from the room, set room settings, establish *WebRTC* peer connections, etc.

WSS communication is encrypted, but it is used without authentication and mainly without authorisation mechanisms. That means that encrypted, unauthenticated, and unauthorized message is able to manage any room selected by *Room ID*. This includes changing settings and kicking users. Next step was to understand the structure in which the Bigscreen application transferred necessary elements of SDP messages. After several systematic experiments, structure of transmitted messages concerning peer-to-peer connections was correctly interpreted.

Penetration Testing of the UI Layer from the Network Side Complete RE of UI layer was achieved. Its software logic and flow of data were analysed. When the Bigscreen application is started, the first screen is the lobby which shows a detailed list of public rooms. Data for this list is retrieved from the signaling server with `room-latest` message. Details of individual rooms are requested from `/roomstate` endpoint at `signal2adm` subdomain. When a user joins a VR room, the application retrieves information about other room participants and displays them in the UI.

As the UI layer is implemented with web technologies, it is potentially vulnerable to attacks normally deployed against web applications (e.g. XSS). A user can change their name in the application's settings. Input field uses validation to allow only letters and numbers in the username. However, from the view of attack from the network side, it was discovered that it is possible to set any value as username and send a specially crafted message to the victim using the signaling channel. To remind, signaling channel uses encryption, but neither authentication nor authorisation. The attacker can set arbitrary values to room name, room description, and room category, too. The attacker can, therefore, send a signaling message to the signaling server and the message is forwarded to room participants or users in application's lobby, as illustrated in fig. 5.9. The application does not perform proper sanitization of data received through encrypted signaling channel from the signaling server. *The Bigscreen application naively trusts the Bigscreen signaling server.* Therefore, XSS in room participant name, room description, room category, and room name was discovered.

5.9 Phase IIIb – Application Reverse Engineering

As explained earlier, there are RE tools focused on C# and *.NET* available. Some of the studied tools (section 4.4) can try to decompile a given application and export its source code into a development project. The *dnSpy* is able to decompile application or library into C# code and this tool offers very useful feature—direct adjustments of C# code and recompilation in place.

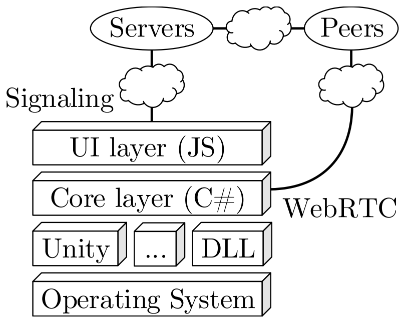


Figure 5.8: Application reverse engineering phase (section 5.9) allowed to understand inner logic and structure of the Bigscreen application, which consists of several layers.

Application Crippling The Bigscreen application loads numerous libraries —Dynamic Link Library (DLL). In this task, decompilation of these libraries back into C# source code was managed. This lead to exploration of the core C# layer of the application (fig. 5.8). This progress was very valuable for a deeper understanding of the application's inner logic including the system of JS-C# bindings on the border between the core C# layer and UI layer. It further allowed to study how the application downloads the UI files from the servers. The core C# layer is also responsible for broadcasting audio & video streams and sharing the state of user's VR avatar and location in the current VR room.

Important point is that these DLLs are loaded without integrity checks. This makes space for possible unauthorised patching of the Bigscreen application (i.e. Application Crippling).

5.10 Phase IV – Exploit Development

Several weak points and vulnerabilities were identified during previous phases of the analysis. It is very important to know the severity of identified issues and possible impact in case of their exploitation. For this reason, attention was in this phase paid to find out ways how malicious hackers or other threat actors could possibly abuse the Bigscreen application and put its users at risk.

Room State Polling Bigscreen server at `signal2adm` subdomain offers the `/roomstate` endpoint. Apparently, it has no request limits. Peter Casey developed a brute forcing script that could search for private *Room ID*s. However, the size of the target state space makes this exploit impractical.

Automated Room Creation Because the signaling channel lacks authentication and authorisation methods, the attacker could implement an automated script that would continuously request signaling server to allocate resources for new rooms (signaling group). This could potentially lead to a DoS attack, but this security analysis is non-destructive (section 5.2).

Kicking All Users from All Public Rooms This is another exploit which is possible due to a lack of authentication and authorisation in the signaling channel. Forged signaling message can kick any user from any room. Only required information is *Room ID* and identification of a given user. This is a possible DoS attack.

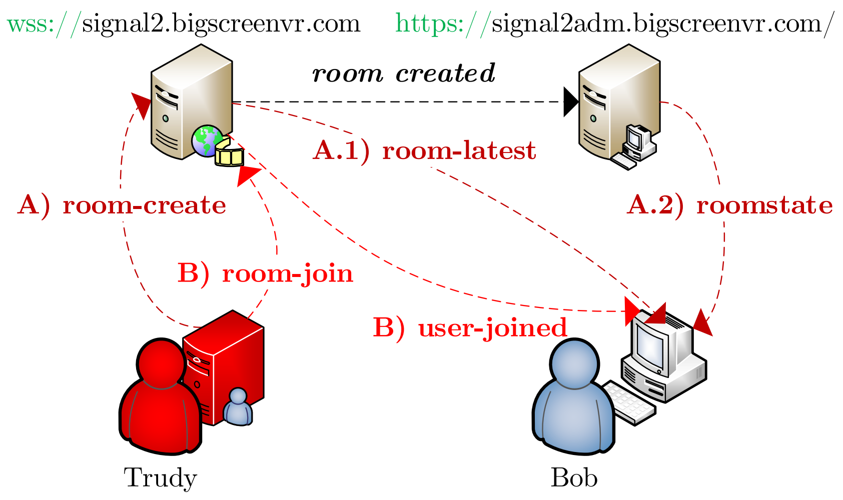


Figure 5.9: Two possible paths of XSS attack over the network against Bigscreen users. On path A, the attacker creates a new public room with payload in room name, room description, or room category (`room-create`). Bob requests list of all public rooms which causes XSS in application of all users in the lobby (`room-latest`). Victim can also request details about selected room which also delivers the XSS payload (`roomstate`). On path B, the attacker sets payload as username and joins Bob's room (`room-join`). As soon as the attacker joins the room, XSS is executed in Bob's application (`user-joined`).

Remote JS Code Execution Using XSS through Signaling Channel The attacker can create a new VR room with XSS payload in room name, room description, and room category (path A in fig. 5.9). The XSS is executed on all users of the Bigscreen application worldwide who are in that moment in the application's lobby. The attacker can insert XSS payload into his username. In this case, the XSS payload is executed by all room participants in any room that the attacker joins (path B in fig. 5.9). XSS attack gains complete control over the UI layer of the victim's application.

Eavesdropping Victim's Computer Screen, Computer Audio, and Microphone Audio Mentioned XSS attack is also able to take control over victim's multimedia sharing. A PoC WebRTC application (fig. 5.10), that was developed during this phase, was able to connect to a legitimate Bigscreen application. With this exploit, the victim would unknowingly send their multimedia to the attacker.

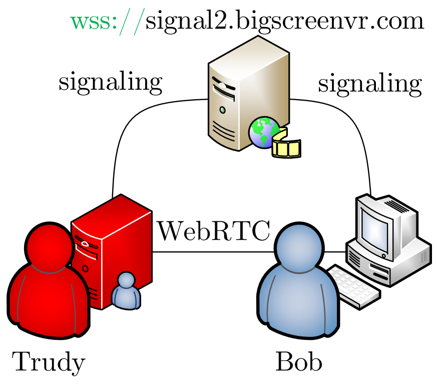


Figure 5.10: Phase *Exploit Development*, among other things, lead to development of a PoC application, which is compatible with the Bigscreen multimedia streaming system.

JS Worm Spreading through the Whole Bigscreen Community Changes applied to the environment due to the XSS attack will persist until reset or modification by the user. This allows victims of an XSS attack to further propagate the payload even in the absence of the initial attacker. An attacker could modify the victim's name to also include an XSS payload, resulting in any future contact with other users to disseminate the payload. Users who observe the attacker's username and execute the script will also modify their username, further circulating the attack (fig. 5.11). JS source code in listing 5.1 illustrates the principle of the worm.

```
function worm(){
  /* payload here */;
  NAME='<sc'+'ript>'+worm.toString()+
  ';worm();</sc'+'ript>John';
};
worm();
```

Listing 5.1: Example of self-replicating XSS payload in variable `NAME` which is used for the VR worm.

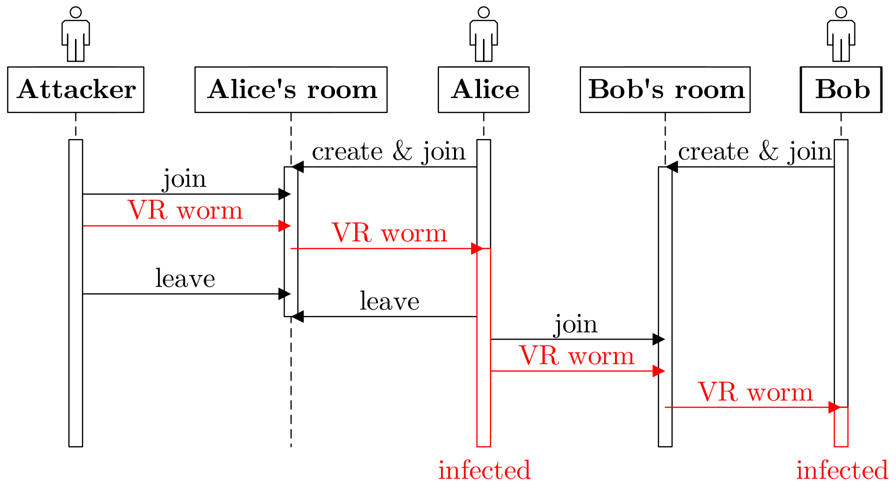


Figure 5.11: Sequence diagram illustrating propagation of VR worm infection between users. Alice was initially infected by attacker in her room. When Alice and Bob met in Bob's room, the worm duplicated and infected Bob as well. This way, the VR worm spreads during contact in VEs like disease in real life.

Discovery of Private Rooms The XSS attack with payload capable of overriding, while maintaining original functionality, the JS function `joinRoomWithId` in the UI layer can force the victim to leak ID of a created room. Room ID is sent to the attacker's C&C server.

Botnet of Infected Applications Controlled from C&C Server As already mentioned, an XSS attack can gain complete control over the UI layer of the victim's application. The XSS attack can spread between victim's using JS worm. Therefore, such demonstrative attack was created. When the victim gets infected by the worm, it becomes a zombie, announces itself to the C&C Server, and awaits commands. With this exploit, it is possible to create a botnet of computers of the whole Bigscreen community and control them from the attacker's C&C Server. Presented exploits were wrapped up into developed PoC C&C server to show the severity of the findings (appendix H). Design and implementation of this tool is in section 5.11.

Download & Execute Malware on Victim's Computer Discovered critical security flaw in Unity scripting API can be exploited to run programs, open folders and files on the victim's computer. It can also force the victim's computer to download and execute malware. Individual steps of this attack are illustrated in fig. 5.12. Listing 5.2 illustrates few ways how this flaw can be exploited.

```
openLink('cmd');
openLink('C:\');
openLink('http://example.com/malware.exe');
```

Listing 5.2: Example of exploiting `openLink` function inside Bigscreen's JS UI, which subsequently calls `Application.OpenURL` method from Unity engine.

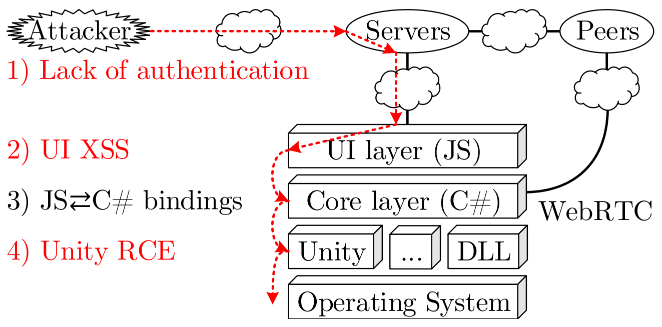


Figure 5.12: Diagram of developed exploit to download and execute malware on victim's computer. Bigscreen application consists of several layers and communicates over network with its servers and peer users in the room. Attacker sends payload to servers which distribute it to users. Payload arrives at UI layer where it causes XSS attack. The attack propagates through bindings from UI to core layer. Application core then calls method from Unity with malicious payload which causes Remote Code Execution (RCE). Attack escapes from the application, downloads malware and executes it.

Man-in-the-Room Attack Combination of several discovered flaws allowed implementation of novel cyber attack in the VEs, which is described separately in section 5.14.

5.11 Phase V – Tool Construction

Previous phase of the analysis which was focused on *network analysis* (section 5.8) and *application reverse engineering* (section 5.9) identified several weak points and vulnerabilities in the communication and in the Bigscreen application itself. Subsequent phase *exploit development* (section 5.10) strove for answering what is the severity of individual vulnerabilities and what could be the real impact of the attacks.

As outlined in section 5.10, numerous exploits and attacks were developed. Exploit which leads to establishing a botnet of infected applications was also presented. Some of the attacks depend on others, e.g. private room needs to be discovered before it can be eavesdropped. In order to demonstrate the findings understandably and to the fullest, it was decided to incorporate invented attacks into a single easy-to-use attacking tool. The tool is able to execute individual attacks against Bigscreen users. It also works as a C&C server which controls the entire botnet of infected Bigscreen applications.

Design of the Functionality

The C&C server's main goal is to be used to easily demonstrate critical impact of the attacks and therefore critical impact of the findings. From its user's (attacker's) perspective, it most importantly offers a simple and consistent UI. This tool acts as a dashboard with all relevant information and controls available.

In order to execute numerous developed attacks, the tool needs to be able to use API of official Bigscreen servers and to communicate using the official signaling channel (fig. 5.5). The Bigscreen's signaling protocol which uses WSS (fig. 5.7) was previously decrypted and then reverse engineered. Signaling communication is also a path for subsequent attacks (fig. 5.9). The tool also needs to request `roomstate` over HTTPS (fig. 5.5). The Bigscreen multimedia streaming system utilises *WebRTC*, therefore the tool also needs functionality to establish *WebRTC* channels (fig. 5.10).

The dashboard's desired functionality is not only monitoring of all public rooms, but also discovery & monitoring of private rooms. The tool then offers ways of eavesdropping on conversations in individual monitored rooms. The attacker can attack any room and take control over the Bigscreen application of any room participant.

One of the developed exploits is, to the best author's knowledge, first PoC VR worm. The tool therefore offers functionality to release such worm infection. Each infected user becomes a zombie which is part of the created botnet. The tool monitors zombies and maintains connection with them in order to control them. The attacker can take control over the individual zombies and send them commands. As already mentioned, a vulnerability which allows for escape from the Bigscreen application and leads to execution of commands and malware delivery on the victim's system was discovered. The C&C server inevitably includes this attack functionality, too.

Because the C&C server establishes and maintains botnet connections to its zombies, it needs some simple communication protocol for controlling zombies and receiving feedback. The tool also needs storage for malware is which distributed during one of the attacks. Technological requirements defined for implementation of desired C&C tool are summarised in table 5.5.

Table 5.5: Technological requirements for the C&C server

Requirement	**Reason**
GUI	Easy-to-use dashboard
Video & audio	Eavesdropping on victim's microphone audio, computer audio, and computer screen
HTTP & HTTPS	Interaction with Bigscreen servers
WSS	Communication using Bigscreen's signaling protocol
WebRTC	Receipt of P2P multimedia streaming
C&C protocol	Control and monitoring of zombies in botnet
File hosting	Malware distribution to victims

Implementation of the Command & Control Dashboard

The implementation process was iterative and several prototypes have been discarded during the development. This text presents only the final version of the implemented solution. Responsibilities of the dashboard were divided into several components, as illustrated in fig. 5.13.

The dashboard is implemented using web technologies (i.e. JS, HTML, CSS). Modern web browser environment offers convenient support for required technologies like *WebRTC*, WSS, and HTTP & HTTPS. Modern browsers also support video and audio elements. It is also good to note that decision to implement the dashboard using web technologies brought the benefit of fast prototyping.

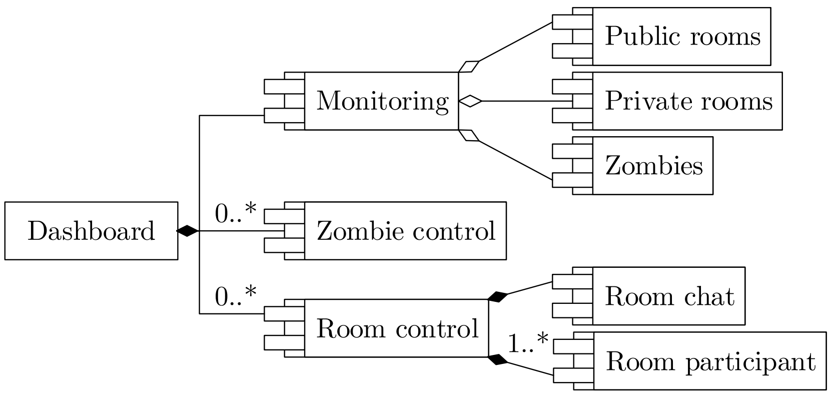


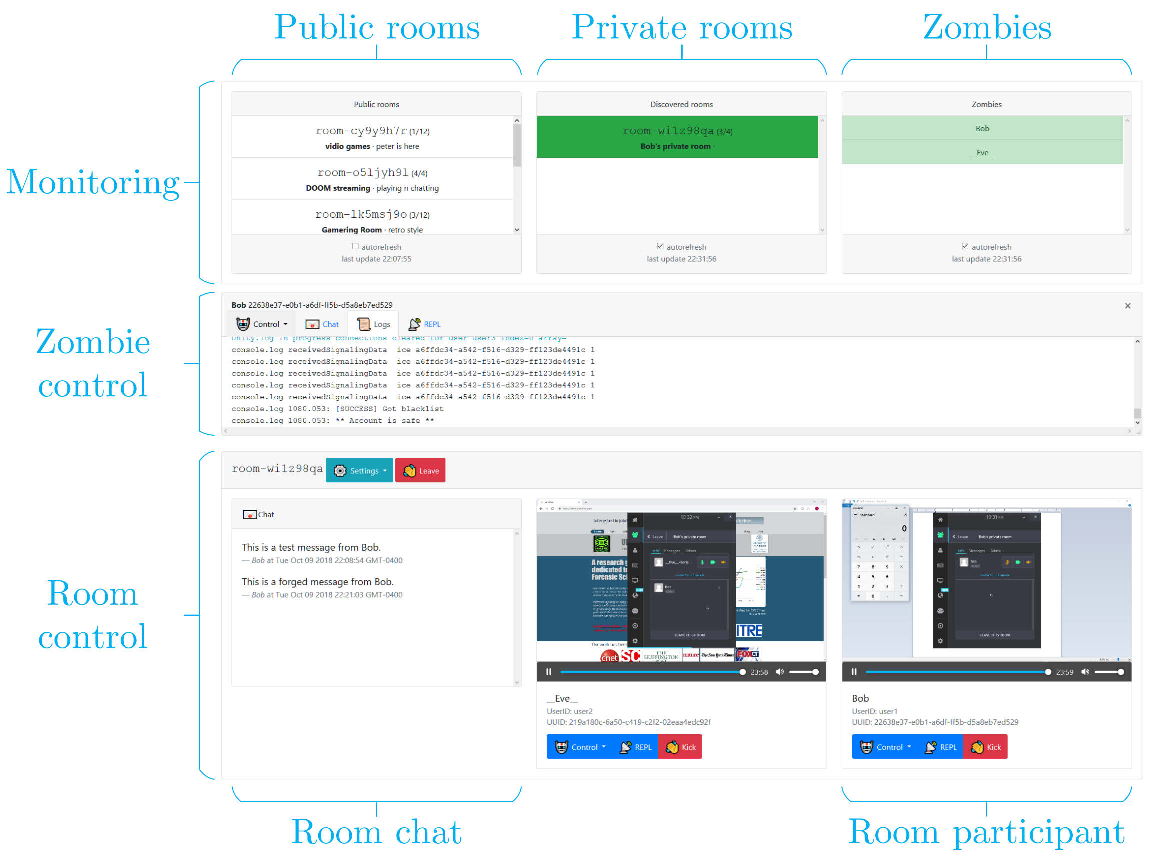
Figure 5.13: Component diagram of the C&C's UI (dashboard) elements. As illustrated, dashboard can control multiple zombies and rooms. Visualisation of these elements is in fig. 5.14.

Attacking the lobby The attacker can easily attack everyone in the Bigscreen's lobby which is done according to path A in fig. 5.9. The attacker can limit the attack only to specific user and it is also possible to choose whether perform a single attack or release replicating VR worm as illustrated in fig. 5.11. Attack to the lobby is managed by instance of `LobbyPoisoner`. The form for controlling the attack is shown in fig. 5.15

Monitoring of public rooms Signaling protocol offers ways to poll server for a list of all public rooms. Because the signaling protocol is not authenticated, this weak point was exploited. Developed `PublicRoomsMonitor` is responsible for creation of specially crafted messages and communication with Bigscreen's signaling server. The `PublicRoomsMonitor` utilises WSS communication. Instance of `PublicRoomsMonitor` is able to periodically poll for latest rooms. Received messages are parsed and list of all public rooms is updated in the dashboard. The UI element for monitoring public rooms is represented by instance of `PublicRoomsMonitorUI`. Each room is represented by list item with room ID, participant count & capacity, room name, and room description (top left part of fig. 5.14).

Monitoring of private rooms Private rooms are monitored in a similar way as public rooms. However, the tool needs to obtain confidential private room ID in order to monitor given room. As soon as the confidential room ID is received, private room is discovered. Monitoring is managed by instance of `DiscoveredRoomsMonitor`. Bigscreen *roomstate* server does not require authentication, as well. Therefore, `DiscoveredRoomsMonitor` is able to periodically poll the server for information about private rooms based on their ID. The monitor utilises HTTPS communication via XMLHttpRequest (XHR). Similarly, the UI element is represented by instance of `DiscoveredRoomsMonitorUI`. This monitor is further able to detect when given room no longer exists. Active rooms are marked with green colour and lost rooms with grey (top central part of fig. 5.14).

Monitoring of zombies in botnet Because the C&C server can also control botnet of all infected Bigscreen applications, it is important to have an overview of such zombies



Public rooms

room-cy9y9h7r (1/12)
vidio games · peter is here

room-o5ljyh91 (4/4)
DOOM streaming · playing n chatting

room-lk5msj9o (3/12)
Gamering Room · retro style

☐ autorefresh
last update 22:07:55

Discovered rooms

room-wi1z98qa (3/4)
Bob's private room ·

☑ autorefresh
last update 22:31:56

Zombies

Bob

__Eve__

☑ autorefresh
last update 22:31:56

Bob 22638e37-e0b1-a6df-ff5b-d5a8eb7ed529

```
unity.log in progress connections cleared for user user3 index=0 array=
console.log receivedSignalingData  ice a6ffdc34-a542-f516-d329-ff123de4491c 1
console.log receivedSignalingData  ice a6ffdc34-a542-f516-d329-ff123de4491c 1
console.log receivedSignalingData  ice a6ffdc34-a542-f516-d329-ff123de4491c 1
console.log receivedSignalingData  ice a6ffdc34-a542-f516-d329-ff123de4491c 1
console.log receivedSignalingData  ice a6ffdc34-a542-f516-d329-ff123de4491c 1
console.log 1080.053: [SUCCESS] Got blacklist
console.log 1080.053: ** Account is safe **
```

room-wi1z98qa

Chat

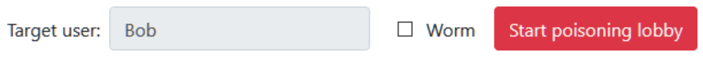
This is a test message from Bob.
— *Bob* at Tue Oct 09 2018 22:08:54 GMT-0400

This is a forged message from Bob.
— *Bob* at Tue Oct 09 2018 22:21:03 GMT-0400

__Eve__
UserID: user2
UUID: 219a180c-6a50-c419-c2f2-02eaa4edc92f

Bob
UserID: user1
UUID: 22638e37-e0b1-a6df-ff5b-d5a8eb7ed529

Figure 5.14: Visualisation of the main parts of the C&C tool. Component diagram is presented in fig. 5.13. *Zombie control* and *Room control* can be opened and closed for each controlled zombie and room. Each room can have multiple *Room participants*.

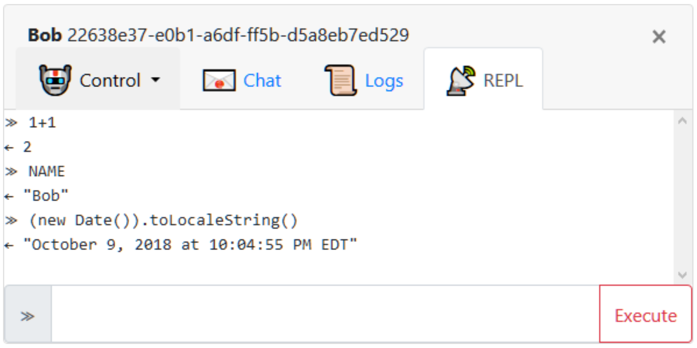


Target user: Bob ☐ Worm

Figure 5.15: Bigscreen's lobby can be attacked using the form in top left part of the dashboard (appendix H). Resized for better readability.

(infected applications). Monitoring of zombies is managed by instance of `ZombiesMonitor`. It utilises developed C&C protocol for communication with zombies. Visual element is represented by instance of `ZombiesMonitorUI`. The monitor is also able to periodically poll the zombies. Active zombies are marked with green colour and the ones with lost connection are marked with grey (top right part of fig. 5.14).

Controlling zombies Each zombie which is part of the botnet can be controlled using corresponding *Zombie control* (central part of fig. 5.14). This control panel can be opened by clicking on selected zombie in a list of monitored zombies. The panel is closed by clicking on given zombie in a list again or by close button in top right corner of the panel. In case a connection to selected zombie was lost, corresponding *Zombie control* panel becomes read-only. The control panel shows zombie's (victim user) name and `uuid`. Each zombie can be controlled using *control menu* with prepared attacks, but also using Read-eval-print loop (REPL) environment (fig. 5.16). The panel consists of the *control menu* and following tabs: Chat, Logs, and REPL. Zombie's chat messages and application logs are continuously eavesdropped even after changing rooms. Chat tab shows chat messages, Logs tab contains log records. The REPL tab is an interactive remote JS shell where the attacker can submit commands and see results from the zombie (similar to CLI usage). Controlling zombies is done using the C&C protocol. Visual representation of *Zombie control* is managed by instance of `ZombieControlUI`. Each controlled zombie has a corresponding `Zombie` instance.

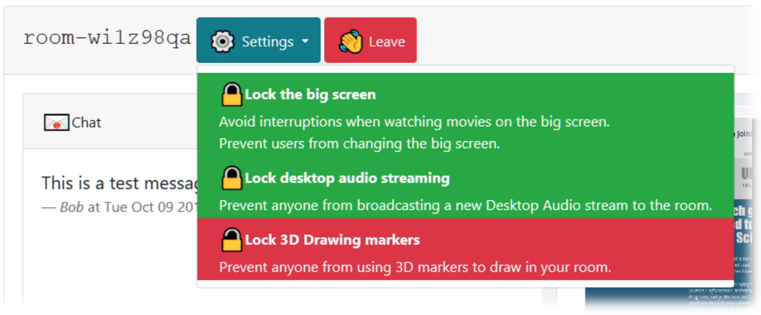


```
≫ 1+1
← 2
≫ NAME
← "Bob"
≫ (new Date()).toLocaleString()
← "October 9, 2018 at 10:04:55 PM EDT"
```

Figure 5.16: Attacker can also send commands to zombie directly using REPL tab in *Zombie control* (central part of fig. 5.14). Resized for better readability.

Controlling rooms The attacker can join and control active rooms with public or discovered room ID. Eavesdropping on conversations in the room and controlling the room can be done using the *Room control* (bottom part of fig. 5.14). Dashboard can control multiple rooms at the same time. When the attacker submits selected room ID to the eavesdropping form (room ID input above zombie monitor in top right part of appendix H), the attacking tool invisibly connects to the room and the *Room control* panel is opened. By clicking on the red Leave button, the attacking tool disconnects from the eavesdropped room and corresponding *Room control* panel closes.

The panel header shows room ID of the controlled room. The Settings button opens *Room settings* menu which shows current state of VR locks (room settings) and the attacker can switch these locks (fig. 5.17). Once the attacker joins a room, chat messages of room participants are eavesdropped and shown in *Room chat* panel (bottom left part of fig. 5.14). Each chat message has its sender, date and time shown.



This is a test messag
— *Bob* at Tue Oct 09 20

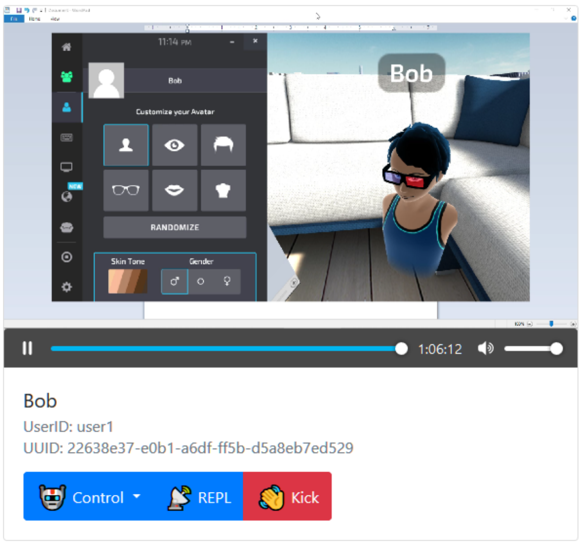
Lock the big screen
Avoid interruptions when watching movies on the big screen.
Prevent users from changing the big screen.

Lock desktop audio streaming
Prevent anyone from broadcasting a new Desktop Audio stream to the room.

Lock 3D Drawing markers
Prevent anyone from using 3D markers to draw in your room.

Figure 5.17: *Room settings* menu in *Room control* panel (bottom part of fig. 5.14) allows the attacker to switch VR locks, see table I.4.

Next part of the UI for controlling rooms is *Room participant* panel (bottom right part of fig. 5.14), shown in detail in fig. 5.18. The panel shows information about associated victim (user name, user ID, `uuid`). The attacker can play victim's audio and video streams. Due to the lack of authentication in the signaling channel, the attacker can kick selected user out of the room back to Bigscreen's lobby just by pressing the red Kick button (table I.4). The panel also offers REPL environment for controlling the victim similar to the one in fig. 5.16.



Bob
UserID: user1
UUID: 22638e37-e0b1-a6df-ff5b-d5a8eb7ed529

Figure 5.18: *Room participant* panel inside the *Room control* panel (bottom part of fig. 5.14) offers ways to control selected victim and to eavesdrop on audio and video streams.

Clicking on the Control button opens the *Control menu* which offers a variety of prepared attacks, as shown in fig. 5.19. Most of the attacks available from the *Control menu* correspond to exploits summarised in table I.3 and table I.6. The attacker can for example force victim to download malware from selected URL and then execute the malware or any other command/program. Another interesting attack is phishing, which is illustrated in fig. 5.20. The phishing attack is not related to RCE in Unity.

The functionality of *Room control* panel is managed by instance of `Room`. The `Room` instance is capable of communicating with Bigscreen signaling server (WSS) and with `roomstate` server (HTTPS via XHR). It can also establish *WebRTC* P2P connection with room participants, this is a responsibility of developed `Connection`. The instance of `Connection` creates *WebRTC* communication compatible with Bigscreen application. Visual element of the *Room control* panel is instance of `RoomUI`, which also manages visual elements of the *Room chat* panel. Instance of `RoomParticipantUI` is responsible for visual representation *Room participant* panel. It is associated with `Identity` instance of the victim and `Room` instance. Visual elements of the REPL environment (fig. 5.16) is managed by instance of `Repl`.

Implementation of the Command & Control Protocol

As explained earlier, discovered flaws allow for taking control over infected Bigscreen applications which then become zombies of created botnet. In order to effectively control zombies

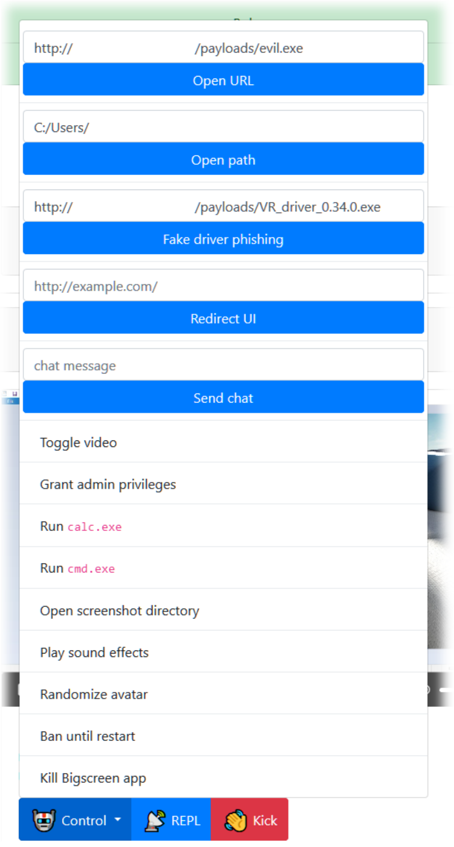


Figure 5.19: *Control menu* gives the attacker ability to execute various attacks against selected victim/zombie. The menu is available from *Room participant* panel (fig. 5.18) and similar menu can be opened from the *Zombie control* panel (fig. 5.16). Overview of individual visual components is in fig. 5.14.

from the C&C server, custom C&C communication protocol was developed. This way, the attacker can conveniently use *control menu* (fig. 5.19) and corresponding commands are transmitted to selected zombie. The C&C protocol was developed with following goals in mind:

- C&C server can send commands to individual zombies.
- Zombies can send results of commands back to the C&C server.
- C&C server can detect whether selected zombie is active or whether the connection was lost.
- Zombie can send additional information to the C&C server (e.g. private room ID, chat messages, log records).

Mechanism for controlling infected Bigscreen applications is based on a combination of XSS vulnerability in application's UI layer and a lack of authentication in Bigscreen's signaling protocol (fig. 5.9). The XSS payload can therefore control UI layer and part of the core application using existing JS to C# bindings. The UI layer offers a limited browser-like environment for JS. In order to continuously control the application, it was essential to find a way how to establish a continuous communication channel to the C&C server from the reachable context of the XSS. Because JS environment of the UI layer supports WS, this technology was utilised for full-duplex communication. Both the C&C server and zombie can send messages asynchronously. It is important to note, that the dashboard of C&C server is implemented with web technologies and runs in a browser. Unfortunately, WS technology is based on client-server model, where WS in a browser is a client. This means that connection between C&C dashboard and zombie is P2P. P2P communication could be achieved via another *WebRTC* connection, but that would further require its dedicated signaling communication, which would lead again to WS. However, it is possible to connect peers with WS using a central *relay server*. Each peer connects as a client to the relay server which then forwards messages from one peer to another, as shown in fig. 5.21.

Data structures transmitted in messages are serialised using JSON. Each message object has information about a `type` of the carried message. Most of the messages use `uuid` to identify associated zombie. Brief overview of messages of developed C&C protocol is in table 5.6.

Table 5.6: Overview of messages of developed custom C&C protocol

Type	**Description**
`dashboard-register`	C&C connected to relay server, connection is marked as C&C and messages for C&C are forwarded to this connection.
`zombie-register`	New zombie announces itself to C&C. Messages for this zombie are forwarded to this connection.
`zombie-cmd`	C&C gives command to a zombie.
`zombie-result`	Zombie responds to C&C with result of command.
`zombie-ping`	C&C monitors whether zombie is active.
`zombie-pong`	Zombie responds to C&C that it is active.
`room-discovered`	Zombie leaks private room ID to C&C.
`chat`	Zombie leaks chat message to C&C.
`log`	Zombie leaks log record to C&C.

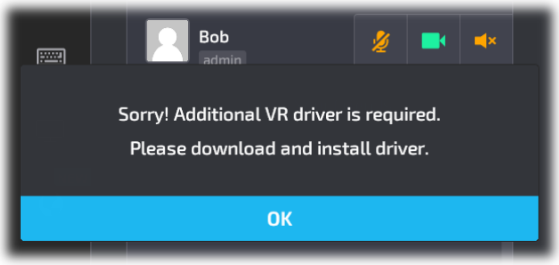

Sorry! Additional VR driver is required.
Please download and install driver.

Figure 5.20: *Control menu* also allows the attacker to execute phishing attack. Victim's Bigscreen application shows modal window asking the victim to install some driver (malware). Clicking OK button downloads the malware. This phishing is not related to RCE in Unity.

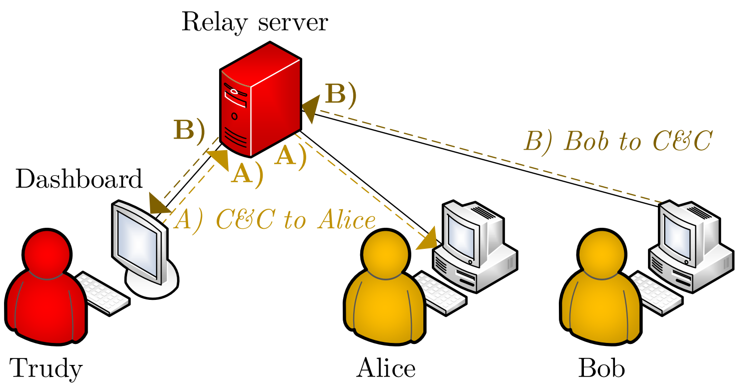


Figure 5.21: Network diagram of developed relay server. Trudy uses C&C dashboard, which is connected to the relay server using WS as a client. Alice and Bob are both zombies already. Zombies are connected to the relay server using WS as clients. Relay server forwards messages. Path A shows message sent from C&C to Alice, path B shows message sent from Bob to C&C.

Implementation of the Relay Server and the File Hosting Server

The relay server (fig. 5.21) was implemented in *Node.js*[5]. Simple web file server developed in *Node.js* was used for file hosting. The file server is used to host demonstrative malware which is then downloaded to victim's computer (table I.6).

In order to demonstrate possible malware outbreak, a program which upon execution renders computer screen black and shows logo of UNHcFREG as ASCII art was developed. The program also shows red warning which states that computer could have been hacked due to a security flaw. It is implemented in C language. This demonstrative program does not cause any other harm to infected testing computer. *It is important to emphasise, that experiments during testing phase were carried out in a controlled laboratory environment.* It is used only to demonstrate that discovered attack against Bigscreen application can download and execute malware.

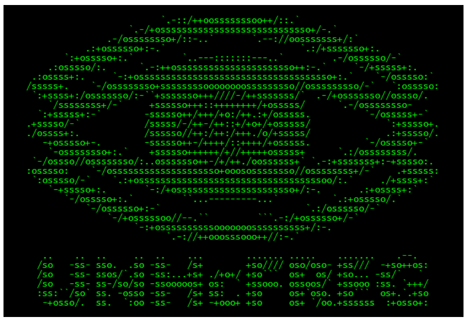


Figure 5.22: Testing that C&C can force victim to download and execute malware was done with developed demonstrative program. The program turns computer screen back and shows logo of UNHcFREG with warning that computer could have been hacked.

[5] https://nodejs.org/

Installation, Configuration, and Usage

As mentioned earlier, the C&C dashboard is implemented using web technologies and is intended to be used directly in a web browser environment. The relay server and file hosting server are both implemented in *Node.js*. Both servers are currently implemented in a single script. But it would be easy to separate them if needed. Dependencies of servers are defined in `package.json`. Demonstrative program which is used as placeholder for malware for distribution is represented by `evil.c` file in `payloads` folder next to server script.

In order to install the attacking tool, a user needs to run `npm install` command in `relay/` directory. This installs dependencies of the server script. As the next step, it is advised to compile malware payload located in `relay/payloads/` using a C compiler.

The C&C dashboard and relay server both need to be configured after installation. A user needs to make sure that `relayWebSocketServerUrl` and `webServerUrl` configuration variables of the dashboard (`dashboard/index.html`) correspond to URLs of relay and file hosting servers (`relay/index.js`).

In order to start the whole C&C server a user (an attacker) should start relay and file hosting server by executing `node ./index.js` command in `relay/` directory. After that, it is possible to open the dashboard by opening `dashboard/index.html` file in a browser. Usage of individual elements of the dashboard was explained in section 5.11.

5.12 Phase VI – Testing

Testing of individual developed attacks (which exploit discovered vulnerabilities) was carried out according to previously defined scenarios for this security analysis (section 5.4 and fig. 5.2). Goals of the testing were mainly to:

- Validate success rate of individual attacks/exploits.
- Validate correct implementation of developed attacks in the C&C server.
- Validate feasibility of developed C&C protocol for controlling zombies.
- Describe critical impact of discovered vulnerabilities in *real-world* scenarios.

This phase also focused on testing attack functionality incorporated in the developed PoC C&C server which can control botnet of infected Bigscreen applications via developed C&C protocol. These scenarios describe normal behaviour of legitimate users. Except for the separate phishing attack (fig. 5.20), all *tested attacks require no action from the victim.*

C&C server setup procedure Each test case is started with following steps. The attacker starts the relay server (fig. 5.21) and opens dashboard (appendix H) which connects to the relay server using `dashboard-register` message (table 5.6). The dashboard connects to Bigscreen signaling servers and obtains list of public rooms for monitoring (fig. 5.14). The attacker ensures that testing malware is correctly prepared and available from the web file hosting server (fig. 5.22). Test cases for the scenarios were evaluated as follows and results are summarised in table 5.7.

Passive stay in the lobby The attacker Trudy performs already described *C&C server setup procedure*. By pressing the red *Start poisoning lobby* button (fig. 5.15), the C&C server sends special signaling messages to Bigscreen signaling server which creates a public room with XSS payload hidden in the room name. This corresponds to path A in fig. 5.9. According to the scenario, Alice downloads list of all public rooms. XSS payload is executed and Alice becomes a zombie in the botnet. Alice appears in zombie monitor in dashboard and Trudy opens *Zombie control*. Trudy forces Alice to download prepared testing malware and then forces Alice to execute it. Malware took control of Alice's computer (fig. 5.22). **The attack was successful**, Alice was hacked and all she did was just opening Bigscreen application.

Created public room The attack starts with *C&C server setup procedure*. Attacker Trudy has an overview of all public rooms in the dashboard. When Alice creates & joins her public room, the room appears in Trudy's dashboard. Trudy selects Alice's room and connects to it for eavesdropping using the dashboard. Alice thinks she is alone in the room. Trudy uses *control menu* to stealthily toggle Alice's video sharing. Trudy can see screen of Alice's computer now. She can take control of Alice's Bigscreen application and also download & execute malware on Alice's computer. Alice's has no suspicion that Trudy can see her screen. **Eavesdropping (attack) was successful.**

Created private room This test starts with *C&C server setup procedure*, too. The attacker Trudy starts attacking the lobby by pressing the red *Start poisoning lobby* button (fig. 5.15). As explained earlier, lobby is attacked according to path A in fig. 5.9. As soon as Alice starts the application and lobby loads list of public rooms, she is attacked and her Bigscreen application becomes a zombie in a botnet. Alice creates & joins private room, but because she is zombie already, her application is automatically forced to leak confidential private room ID to Trudy's C&C server. The room ID is sent using `room-discovered` message of the C&C protocol (table 5.6). Alice's private room has just been discovered and it appears in monitor of private rooms in Trudy's dashboard. Trudy selects Alice's private room and connects to it for eavesdropping. Trudy toggles Alice's video sharing as well. Even though Alice created private room and she thinks she is alone in a secure room, Trudy can now see screen of Alice's computer. Trudy can take control of Alice's Bigscreen application and distribute malware, too. **The attack was successful.**

Private meeting **This scenario further tests the novel MitR attack.** Trudy performs *C&C server setup procedure*. This test scenario includes another malicious actor called Mallory. Mallory uses the patched (Application Crippling) version of the Bigscreen application (fig. 5.24). Attackers Mallory and Trudy can communicate and coordinate the attack. However, this test scenario does not require Trudy and Mallory to be different people, one attacker could easily use the dashboard of C&C server and at the same time use the patched Bigscreen application. For clarity purposes, this test is described with both Trudy and Mallory. Trudy starts attacking the lobby using the *Start poisoning lobby* button (fig. 5.15). Alice starts the Bigscreen application, the lobby is opened, list of public rooms is loaded, Alice is attacked and becomes a zombie. Trudy can see Alice in a list of zombies. Trudy stops attacking the lobby. Alice creates & joins private room, room ID is leaked to Trudy. As described in the scenario, Alice gives Bob room ID and he joins Alice's private room. Trudy selects Alice's private room from list of discovered private rooms in the dashboard and connects to it for eavesdropping. Trudy can now control both Alice and Bob, Trudy can also toggle their video sharing & see their screens. She can distribute malware at this point. As Alice and Bob exchange chat messages, Trudy can see the messages in *Room chat* panel (bottom left part of fig. 5.14). Trudy can also spoof chat messages, for example impersonate Bob and write messages in his name. However, the attackers want to see inside the VE of the VR room. Trudy shares obtained confidential private room ID with Mallory. Mallory joins Alice's room as invisible user. Alice and Bob have no idea that Mallory is with them in their private room. Mallory can move in virtual space, hear, and see everything that is happening in the room. This way, Mallory can literally look over their shoulders. **This attack including MitR attack was successful.**

Transition between rooms This test is focused on the worm attacking lobby and spreading infection from one victim to another. Attacker Trudy executes *C&C server setup procedure*. Trudy starts attacking the lobby with VR worm. Worm infection was during testing limited to testing users Alice and Bob located in the laboratory. As Alice starts the application, she is infected with the replicating worm and becomes zombie. Trudy stops attacking the lobby. Alice creates & joins her new public room. Bob creates & joins his public room. Alice leaves her room and joins Bob's public room. *As soon as Alice meets Bob in virtual space, the VR worm duplicates and infects Bob.* This procedure is roughly illustrated in fig. 5.11. Bob is now zombie, too. He also propagates the infection. Trudy can now see both Alice and Bob in list of zombies. Trudy can see that Alice's room no longer exists and that Alice and Bob are both in Bob's room. Trudy can eavesdrop on any room that Alice or Bob visit. From this point on, Trudy can take control of every infected victim that Alice or Bob meet in VEs while they carry the worm infection. Trudy can distribute malware to all these affected computers. **This attack including VR worm propagation was successful.**

Table 5.7: Attack testing results based on initial scenarios

Scenario	Test result
Passive stay in the lobby	Attack successful
Created public room	Attack successful
Created private room	Attack successful
Private meeting	Attack successful
Transition between rooms	Attack successful

5.13 Findings

Results of carried out security analysis are summarised in appendix I. Table I.1 points out weak points of individual connections that the Bigscreen application establishes over the network. Table I.2 summarises main discovered vulnerabilities. Individual developed exploits and attacks are sorted according to the vulnerability they are based on (table I.3, table I.4, and table I.5). Advanced attacks require a combination of several discovered security flaws, as presented in table I.6.

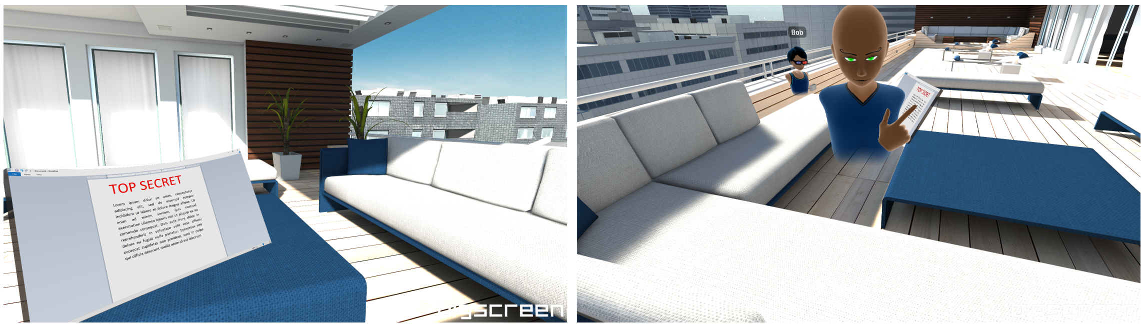


Figure 5.23: Novel Man-in-the-Room attack. Figure on the left shows the view of the user Bob, figure on the right shows attacker Mallory who is invisible while in the room with Bob. The attacker uses malicious version of the application, as shown in fig. 5.24.

The analysis further discovered additional information which can be classified as *miscellaneous findings* with low severity compared to other findings. These include for example: development information leak (API key and SteamIDs & Oculus IDs of developers) in JS UI source code, development sample files leak in application's directory, and a publicly available blacklist containing information like `uuid`, ban reason and username.

5.14 Novel Man-in-the-Room Attack

Because the Bigscreen application uses DLLs without integrity checking (table I.2), it was possible to change the source code of selected libraries (patch) and the Bigscreen application still used these libraries. This allowed changing selected behaviour. Developed PoC patched Bigscreen application was able to connect with legitimate Bigscreen applications. This also obtained complete control over one end of audio/video/microphone/data streams (fig. 5.24). Furthermore, previously discovered way how to join private VR rooms was combined with this attack. A series of steps which also allowed to hide attacker's presence from UI of others using XSS payloads was created. Implemented MitR attack exploited multiple discovered attacks (table I.6) in order to achieve invisibility in VR & UI and to join even private rooms. This attack led to complete invisibility in selected VR room. Victims do not have any information about the attacker being in their room. The attacker can see victims in VR, see screens of their computers, and hear their audio/microphone (fig. 5.23). Dr Baggili has named this novel privacy violation technique in VR a *Man-in-the-Room Attack*.

5.15 Mitigations & Suggestions

During this analysis, several security vulnerabilities in the Bigscreen application and in the Unity platform were discovered. This section outlines mitigations and suggestions, which were expressed in the *responsible disclosure report*, so that the identified issues may be remedied (section 5.16).

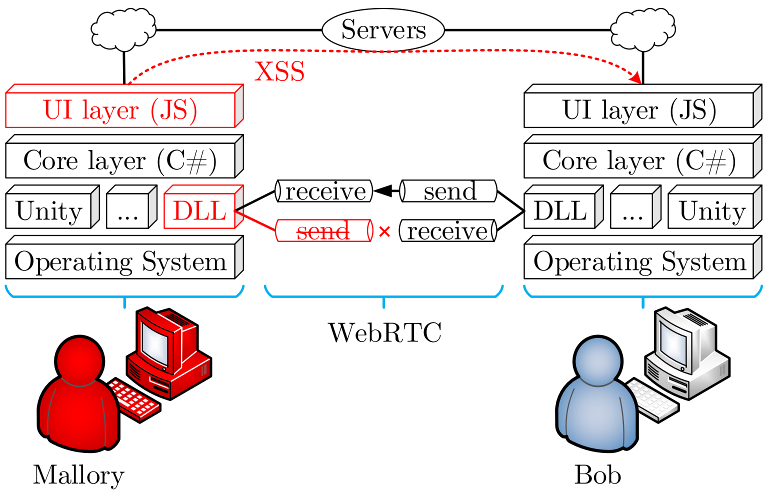


Figure 5.24: The attacker Mallory uses patched (application crippling) version of the application which does not send VR state to other room participants and which also uses XSS payloads to hide traces of Mallory's presence in the room.

Bigscreen

Individual discovered security flaws were caused by shortcomings & vulnerabilities in authentication, authorisation, encryption, data sanitization, integrity checking, or by a critical security vulnerability in 3rd party software (Unity engine). Individual flaws with smaller impact were chained together resulting in attacks with critical impact. Therefore, it is suggested to address the following.

1. **Safe data manipulation and proper data sanitization** Because the application's UI is implemented with web technologies, it inherits security risks from the area of web applications. Several injection points for XSS existed due to unsafe HTML manipulation. It is recommended to use safe data manipulation and proper data sanitization methods at all times. It is also recommended to periodically check use of methods which can directly create and manipulate HTML without sanitizing data. One of the suggested solutions to this issue is to use some templating engine which would offer automatic escaping of data. Today's templating engines also often take care of *context-aware escaping*.

2. **Secure authentication & authorisation** Both administrative activities and private rooms should have secure authentication & authorisation to determine the validity of requests. In order to join a private room, all that is now required is the private Room ID. It is recommended to introduce user accounts and proper authentication & authorisation.

3. **Cautious handling of insecure API** It is suggested to cautiously handle insecure API, especially proper sanitization of `url` parameter of the `Application.OpenURL` method from the Unity Scripting API.

4. **Encrypted communication** It is also strongly suggested to use HTTPS communication instead of unencrypted HTTP.

5. **Integrity checking** It is further suggested to ensure that the application and it's dependencies have not been modified. Methods like DLL integrity checking would be beneficial in this approach.

6. **Enforcing VR state sharing** The application should monitor and enforce that all room participants correctly share information about their avatar and position in VE.

7. **Brute force protection** The Bigscreen's server infrastructure should utilise brute force protection by for example enforcing limits on the number and frequency of requests made to the room status servers.

8. **Confidential blacklist** It is recommended to change the way of checking username against blacklist, because the whole blacklist should not be publicly available.

9. **Removing development relics** Some of the debugging functionality and testing files aided in the investigation. It is recommended to remove development relics and functionality unnecessary for production software.

Unity Scripting API

The *responsible disclosure* expressed concerns about the ability of the `Application.OpenURL` method to run commands/programs and open directories/files on host systems (without Uniform Resource Identifier (URI) scheme). The *responsible disclosure* considered such functionality to be a severe security vulnerability. It is suggested to implement parameter validations inside this API, which would prevent this issue.

It is reasonable for `Application.OpenURL` method to support various types of URLs. However, some URI schemes might be unexpected for a developer. Therefore, it is suggested to consider support of some unexpected URI schemes. In a case that support for schemes like for example `search-ms`, File Transfer Protocol (FTP) and Server Message Block (SMB) is expected, one of following is suggested:

- Updating documentation with warning that developer has to conduct proper sanitization of parameter `string url` and also, warning about possible consequences would be very helpful.

- Updating `Application.OpenURL` method so that developers have to provide the second parameter in form of a URI scheme whitelist for a given method call.

5.16 Responsible Disclosure

Details about all discovered flaws, demonstrative exploits, and implemented attacks together with mitigations & suggestions were summarised in *responsible disclosure reports* by Casey and Vondráček. Dr Baggili then led the communication with Bigscreen and Unity Technologies.

After the *responsible disclosure*, the Bigscreen company understood that reported vulnerabilities had really critical impact on the Bigscreen application and its more than $500,000$

users. Vulnerability disclosure timeline was agreed with UNHcFREG. Results of this research were not public until contacted companies took action concerning reported security vulnerabilities. The Bigscreen application had an extensive update in which reported vulnerabilities were addressed. Bigscreen's CEO Darshan Shankar then published statement, which included following.

"[...] thank you to security researchers Ibrahim Baggili, Peter Casey, Martin Vondráček from the University of New Haven for helping us discover several vulnerabilities with Bigscreen's servers and streaming systems. These bugs have been fully patched in this update." (Shankar 2019)

In reaction to the *responsible disclosure*, Unity Technologies company decided to address their issue by updating documentation with warnings as was suggested. Initial documentation of this dangerous method contained only the following.

"Opens the url in a browser. In the editor or standalone player this will open a new page in the default browser with the url. It will also bring the browser application to the front." (Unity Documentation 2018.3-002V[6])

After the *responsible disclosure*, the documentation was greatly expanded and now includes several warnings like for example the following.

"[...] the OpenURL command can be unexpectedly powerful [...] you must be extremely careful that you do not provide a string to this function which could possibly be maliciously crafted or modified by a 3rd party [...] consider this method to have similar security implications as an eval type function [...]" (Unity Documentation 2019.1-002V[7])

[6]https://docs.unity3d.com/2018.3/Documentation/ScriptReference/Application.OpenURL.html
[7]https://docs.unity3d.com/2019.1/Documentation/ScriptReference/Application.OpenURL.html

Chapter 6

Conclusion

The research in the area of VR security was started at the University of New Haven (CT, USA) by Yarramreddy et al. (2018), Casey et al. (2019a), and Casey et al. (2019b) from UNHcFREG. The researchers carried out forensic analysis which examined feasible artifacts and behaviour of popular VR social applications. They also conducted security analysis and developed attacks related to immersive VR. This Vondráček's work follows these researches.

This thesis delivered security analysis focused on the Bigscreen application intended for communication and social activities in VR. Bigscreen was chosen because of its popularity and userbase which includes more than $500,000$ users (table 5.1). The application is implemented with use of Unity engine, which can be found on $3,000,000,000$ devices worldwide (table 5.1). This thesis also offers a systematic and documented approach to penetration testing and security analysis focused on VR communication applications. The analysis discovered several security flaws in the Bigscreen application and also a security vulnerability in the Unity engine. The author developed exemplary exploits and then wrapped up implemented attacks into a PoC C&C server. The security flaws allowed a potential attacker to create a botnet of vulnerable Bigscreen applications, create a VR worm spreading via contact in VEs, and deliver malware potentially to all Bigscreen users. This analysis also found a way how to perform a novel attack, named by Dr Baggili *Man-in-the-Room*, that lets the attacker eavesdrop in locked private VR rooms while being invisible to other users.

These flaws and attacks were tested and the outcomes of the analysis were summarised (section 5.13). Relevant companies were notified about discovered flaws in their software according to *responsible disclosure* process (section 5.16) so that the problems could be remedied. Provided the size of userbase, it is clear that the potential impact on users of these technologies — should they have been exploited — could have been dramatic. Both companies used the findings for improving the security of their technologies and protection of the users.

Because the responsible disclosure reports and this thesis include *mitigations & suggestions* (section 5.15), it is beneficial also for other software developers who want to be aware of common development mistakes which can lead to such critical security vulnerabilities.

This research also aimed at raising public awareness about security of VR technologies and popular applications. Impact of this research is supported by the fact, that it earned international attention via more than 80 news sites, electronic and printed magazines, podcasts, radios, etc. Some of them are listed in appendix J.

The author of this thesis also directed efforts to raise public awareness and popularising VR security by communication with media. His direct efforts caused that media in the Czech Republic published 18 articles, posts, and interviews (May 2019), as summarised in appendix K. Additional popularisation and media coverage is currently planned. He's also author of two informal and popularising articles about this VR security analysis (appendix L).

The author presented this thesis at the Faculty of Information Technology Brno University of Technology during students' conference Excel@FIT in April 2019. He received three awards for his work: 1) Award of the Expert Panel, 2) Award of Jiří Kunovský (Award of the Public), 3) Award of the Škoda Auto company. The author also agreed on presenting his work during the faculty's event for the public in May 2019.

To conclude, this research pointed out the risks related to security of VR technologies. It not only developed new immersive attacks, but most importantly helped to mitigate the security risks for more than half a million Bigscreen users and all affected Unity applications worldwide.

Bibliography

[1] Against Gravity Corp. *Rec Room®*. Online. Against Gravity. URL: https://www.againstgrav.com/rec-room/ (visited on 01/14/2019).

[2] Against Gravity Corp. *Rec Room in 2018*. Online. Against Gravity, Jan. 2019. URL: https://www.againstgrav.com/2018inreview/ (visited on 01/14/2019).

[3] H. H. Alsaadi, M. Aldwairi, et al. "Penetration and Security of OpenSSH Remote Secure Shell Service on Raspberry Pi 2". In: *2018 9th IFIP International Conference on New Technologies, Mobility and Security (NTMS)*. Feb. 2018, pp. 1–5. DOI: 10.1109/NTMS.2018.8328710.

[4] AltspaceVR Inc. *Press Kit*. Online. AltspaceVR. URL: https://altvr.com/news/ (visited on 01/14/2019).

[5] Andres Andreu. *Professional Pen Testing for Web Applications*. eng. Indianapolis: Wiley, 2006. ISBN: 978-0-471-78966-6.

[6] Daniel Bambušek. "User Interface for ARTable and Microsoft Hololens". MA thesis. Brno, CZ: Brno University of Technology, Faculty of Information Technology, 2018. URL: http://www.fit.vutbr.cz/study/DP/DP.php?id=20106.

[7] Vítězslav Beran. "Augmented Multi-user Communication System". In: *Proceedings of the Working Conference on Advanced Visual Interfaces*. AVI '04. Gallipoli, Italy: ACM, 2004, pp. 257–260. ISBN: 1-58113-867-9. DOI: 10.1145/989863.989907.

[8] Bigscreen, Inc. *Press Kit*. Online. Bigscreen, Oct. 2018. URL: https://bigscreenvr.com/press/ (visited on 01/14/2019).

[9] M. Billinghurst, I. Poupyrev, et al. "Mixing realities in Shared Space: an augmented reality interface for collaborative computing". In: *2000 IEEE International Conference on Multimedia and Expo. ICME2000. Proceedings. Latest Advances in the Fast Changing World of Multimedia (Cat. No.00TH8532)*. Vol. 3. July 2000, 1641–1644 vol.3. DOI: 10.1109/ICME.2000.871085.

[10] Doug Bowman, Ernst Kruijff, et al. *3D User Interfaces: Theory and Practice*. eng. Pearson Education, Inc., 2005. ISBN: 0-201-75867-9. URL: https://books.google.cz/books?id=JYzmCkf7yNcC (visited on 03/10/2019).

[11] Tomáš Brestič. "Effective Use of Limited Game Space". Czech. MA thesis. Brno, CZ: Brno University of Technology, Faculty of Information Technology, 2018. URL: http://www.fit.vutbr.cz/study/DP/DP.php?id=21255.

[12] F. Callegati, W. Cerroni, and M. Ramilli. "Man-in-the-Middle Attack to the HTTPS Protocol". In: *Security Privacy, IEEE* (Jan. 2009), pp. 78–81. ISSN: 1540-7993. DOI: 10.1109/MSP.2009.12.

[13] Peter Casey, Ibrahim Baggili, and Ananya Yarramreddy. "Immersive Virtual Reality Attacks and the Human Joystick". In: *IEEE Transactions on Dependable and Secure Computing* (2019), pp. 1–1. ISSN: 1545-5971. DOI: 10.1109/TDSC.2019.2907942.

[14] Peter Casey, Rebecca Lindsay-Decusati, et al. "Inception: Virtual space in memory space in real spacee – Memory forensics of immersive Virtual Reality with the HTC Vive". In: *Digital Investigation* (July 2019). DOI: 10.1016/j.diin.2019.04.007.

[15] Renee Chmiel. *University of New Haven Professor Abe Baggili Warns of Vulnerabilities in Virtual Reality Gaming Systems.* Online. University of New Haven, Charger Nation News, The Charger Blog, Jan. 2019. URL: https://www.newhaven.edu/news/blog/2019/virtual-reality-gaming-systems.php (visited on 01/14/2019).

[16] Aldo Cortesi, Maximilian Hils, et al. *mitmproxy: A free and open source interactive HTTPS proxy.* [Version 4.0]. 2010. URL: https://mitmproxy.org/.

[17] Courtney Cronin. *How more than 2,500 virtual reality reps helped transform Case Keenum's game.* Online. ESPN, NFL Nation, Jan. 2018. URL: http://www.espn.com/blog/nflnation/post/_/id/265904/virtual-reality-played-a-role-in-transforming-case-keenums-game (visited on 01/14/2019).

[18] Gokila Dorai, Shiva Houshmand, and Ibrahim Baggili. "I Know What You Did Last Summer: Your Smart Home Internet of Things and Your iPhone Forensically Ratting You Out". In: *Proceedings of the 13th International Conference on Availability, Reliability and Security.* ARES 2018. Hamburg, Germany: ACM, 2018, 49:1–49:10. ISBN: 978-1-4503-6448-5. DOI: 10.1145/3230833.3232814.

[19] Ralph Droms. *Dynamic Host Configuration Protocol.* RFC 2131. RFC Editor, Mar. 1997. URL: http://www.rfc-editor.org/rfc/rfc2131.txt.

[20] Lisa Eadicicco. *Microsoft Just Bought a Virtual Reality Company to Challenge Facebook.* Online. Time, Oct. 2017. URL: http://time.com/4967092/microsoft-altspacevr-virtual-reality/ (visited on 01/14/2019).

[21] Facebook Inc. *Facebook VR Apps.* Online. Facebook Help Center. URL: https://www.facebook.com/help/145986952600577?locale=en_US (visited on 01/14/2019).

[22] Rachel Franklin. *Facebook Spaces: A New Way To Connect With Friends In VR.* Online. Facebook Newsroom, Apr. 2017. URL: https://newsroom.fb.com/news/2017/04/facebook-spaces/ (visited on 01/14/2019).

[23] Trevor Haigh, Frank Breitinger, and Ibrahim Baggili. "If I Had a Million Cryptos: Cryptowallet Application Analysis and a Trojan Proof-of-Concept". In: *Digital Forensics and Cyber Crime.* Ed. by Frank Breitinger and Ibrahim Baggili. Cham: Springer International Publishing, 2019, pp. 45–65. ISBN: 978-3-030-05487-8.

[24] S. Hanks, T. Li, et al. *Generic Routing Encapsulation (GRE).* RFC 1701. RFC Editor, Oct. 1994.

[25] Jackie Hennessey. *Cybersecurity and Forensics Work Makes National Impact.* Online. University of New Haven, Charger Nation News, The Charger Blog, Oct. 2018. URL: http://www.newhaven.edu/news/blog/2018/virtual-reality-research.php (visited on 01/14/2019).

[26] Pete Herzog and Marta Barceló. *The Open Source Security Testing Methodology Manual.* 3.02. Online. Institute for Security and Open Methodologies (ISECOM). Dec. 2010, pp. 1–209. URL: http://www.isecom.org/mirror/OSSTMM.3.pdf (visited on 01/14/2019).

[27] Gareth Heyes. *Executing non-alphanumeric JavaScript without parenthesis.* Online. PortSwigger Web Security Blog, PortSwigger Ltd., July 2016. URL: https://portswigger.net/blog/executing-non-alphanumeric-javascript-without-parenthesis (visited on 05/20/2019).

[28] Jakub Hrozek. "Penetration Testing of an Open-Source Software". Czech. MA thesis. Brno, CZ: Brno University of Technology, Faculty of Information Technology, 2010. URL: http://www.fit.vutbr.cz/study/DP/DP.php?id=9336.

[29] Nguyen Viet Hung. *Breaking down one of the coolest JavaScript obfuscators.* Online. The Z – Medium, Dec. 2017. URL: https://medium.com/the-z/breaking-down-one-of-the-coolest-javascript-obfuscators-15b234f768c (visited on 05/20/2019).

[30] David Jagneaux. *'Bigscreen' Is Hosting The First Virtual LAN Party in VR Tonight Featuring 'Rocket League'.* Online. UploadVR, Nov. 2016. URL: https://uploadvr.com/bigscreen-lan-party-rocket-league/ (visited on 01/14/2019).

[31] Steven M. LaValle. *Virtual Reality.* Online. University of Illinois: Cambridge University Press, 2017. URL: http://vr.cs.uiuc.edu/ (visited on 01/14/2019).

[32] Sebastien Lebreton. *Assembly Manipulation and C# / VB.NET Code Injection.* Online. CodeProject, Apr. 2013. URL: https://www.codeproject.com/Articles/20565/Assembly-Manipulation-and-C-VB-NET-Code-Injection (visited on 01/14/2019).

[33] Sebastien Lebreton. *Reflexil, The .NET Assembly Editor.* Online. July 2017. URL: http://reflexil.net/ (visited on 01/14/2019).

[34] Zhen Ling, Zupei Li, et al. "I Know What You Enter on Gear VR". In: *Proceedings of IEEE Conference on Communications and Network Security (CNS).* Washington, D.C., USA, June 2019.

[35] Marek Marušic. "Automatization of MitM Attack for SSL/TLS Decryption". Czech. Bachelor's thesis. Brno, CZ: Brno University of Technology, Faculty of Information Technology, 2016. URL: http://www.fit.vutbr.cz/study/DP/BP.php?id=18593.

[36] Michael Mateas and Nick Montfort. "A box, darkly: Obfuscation, weird languages, and code aesthetics". In: *Proceedings of the 6th Digital Arts and Culture Conference, IT University of Copenhagen.* 2005, pp. 144–153.

[37] Petr Matoušek, Jan Pluskal, et al. "Advanced Techniques for Reconstruction of Incomplete Network Data". In: *Digital Forensics and Cyber Crime.* Ed. by Joshua I. James and Frank Breitinger. Cham: Springer International Publishing, 2015, pp. 69–84. ISBN: 978-3-319-25512-5.

[38] Rachel Metz. *Virtual Reality's Missing Element: Other People*. Online. MIT Technology Review, June 2017. URL: https://www.technologyreview.com/s/607956/virtual-realitys-missing-element-other-people/ (visited on 01/14/2019).

[39] Matteo Meucci, Andrew Muller, et al. *OWASP Testing Guide 4.0 - Release*. Online. The OWASP Foundation, Sept. 2014. URL: https://www.owasp.org/index.php/OWASP_Testing_Project (visited on 05/19/2019).

[40] G. Meyer-Lee, J. Shang, and J. Wu. "Location-leaking through Network Traffic in Mobile Augmented Reality Applications". In: *2018 IEEE 37th International Performance Computing and Communications Conference (IPCCC)*. Nov. 2018, pp. 1–8. DOI: 10.1109/PCCC.2018.8711065.

[41] Michal Mjartan and Vladimir Bronis. "Virtuální realita přináší nové možnosti pro zaškolování i proškolování zaměstnanců". In: *IT SYSTEMS* 9 (Sept. 2018). Online, pp. 46–47. URL: https://www.systemonline.cz/it-asset-management/virtualni-realita-prinasi-nove-moznosti-zaskolovani.htm (visited on 05/21/2019).

[42] P. Mockapetris. *Domain names - concepts and facilities*. STD 13. RFC Editor, Nov. 1987. URL: http://www.rfc-editor.org/rfc/rfc1034.txt.

[43] Nick Montfort. "Obfuscated code". In: *Software Studies: A Lexicon* (2008), pp. 193–199.

[44] T. Narten, E. Nordmark, et al. *Neighbor Discovery for IP version 6 (IPv6)*. RFC 4861. RFC Editor, Sept. 2007. URL: http://www.rfc-editor.org/rfc/rfc4861.txt.

[45] Václav Němec. "Penetration Testing Application for Data Validation Flaws Based Web Vulnerabilities". Czech. Bachelor's thesis. Brno, CZ: Brno University of Technology, Faculty of Information Technology, 2011. URL: http://www.fit.vutbr.cz/study/DP/BP.php?id=11541.

[46] Alfred Ng. *VR systems Oculus Rift, HTC Vive may be vulnerable to hacks*. Online. CBS Interactive Inc. CNET, Wearable Tech, Apr. 2018. URL: https://www.cnet.com/news/hack-a-vr-system-lead-a-player-astray-yes-say-researchers/ (visited on 01/14/2019).

[47] Miroslav Novotný. "VR Interactive Application". Czech. MA thesis. Brno, CZ: Brno University of Technology, Faculty of Information Technology, 2017. URL: http://www.fit.vutbr.cz/study/DP/DP.php?id=20032.

[48] Sean-Philip Oriyano. *Penetration Testing Essentials*. Online. Wiley, 2016. ISBN: 9781119235309. URL: https://books.google.cz/books?id=mJIgCwAAQBAJ (visited on 01/14/2019).

[49] Patricio Palladino. *Brainfuck beware: JavaScript is after you!* Online. Blog Patricio Palladino, Aug. 2012. URL: http://patriciopalladino.com/blog/2012/08/09/non-alphanumeric-javascript.html (visited on 05/20/2019).

[50] Stefano Di Paola. *Advanced JS Deobfuscation Via AST and Partial Evaluation (Google Talk WrapUp).* Online. Minded Security Blog, Oct. 2015. URL: https://blog.mindedsecurity.com/2015/10/advanced-js-deobfuscation-via-ast-and.html (visited on 05/20/2019).

[51] David C. Plummer. *Ethernet Address Resolution Protocol: Or converting network protocol addresses to 48.bit Ethernet address for transmission on Ethernet hardware.* STD 37. RFC Editor, Nov. 1982. URL: http://www.rfc-editor.org/rfc/rfc826.txt.

[52] Jan Pluskal, Petr Matoušek, et al. "Netfox Detective: A tool for advanced network forensics analysis". In: *Proceedings of Security and Protection of Information (SPI) 2015.* Brno, CZ: Brno University of Defence, 2015, pp. 147–163. ISBN: 978-80-7231-997-8. URL: http://www.fit.vutbr.cz/research/view_pub.php?id=10863.

[53] I. Poupyrev, D. S. Tan, et al. "Developing a generic augmented-reality interface". In: *Computer* 35.3 (Mar. 2002), pp. 44–50. ISSN: 0018-9162. DOI: 10.1109/2.989929.

[54] Ivan Poupyrev, Mark Billinghurst, et al. "The Go-go Interaction Technique: Non-linear Mapping for Direct Manipulation in VR". In: *Proceedings of the 9th Annual ACM Symposium on User Interface Software and Technology.* UIST '96. Seattle, Washington, USA: ACM, 1996, pp. 79–80. ISBN: 0-89791-798-7. DOI: 10.1145/237091.237102.

[55] Stacy Prowell, Rob Kraus, and Mike Borkin. "CHAPTER 6 - Man-in-the-Middle". In: *Seven Deadliest Network Attacks.* Ed. by Stacy Prowell, Rob Kraus, and Mike Borkin. Boston: Syngress, 2010, pp. 101–120. ISBN: 978-1-59749-549-3. DOI: 10.1016/B978-1-59749-549-3.00006-7.

[56] John Patrick Pullen. *This Startup Is Solving Virtual Reality's Biggest Problem.* Online. Time, Jan. 2016. URL: http://time.com/4185359/altspace-virtual-reality/ (visited on 01/14/2019).

[57] Jun Rekimoto and Katashi Nagao. "The World Through the Computer: Computer Augmented Interaction with Real World Environments". In: *Proceedings of the 8th Annual ACM Symposium on User Interface and Software Technology.* UIST '95. Pittsburgh, Pennsylvania, USA: ACM, 1995, pp. 29–36. ISBN: 0-89791-709-X. DOI: 10.1145/215585.215639.

[58] Louis B. Rosenberg. "The Use of Virtual Fixtures as Perceptual Overlays to Enhance Operator Performance in Remote Environments". In: (Sept. 1992). Online, p. 52. URL: https://apps.dtic.mil/dtic/tr/fulltext/u2/a292450.pdf (visited on 01/14/2019).

[59] Louis B. Rosenberg. "Virtual fixtures: Perceptual tools for telerobotic manipulation". In: *Proceedings of IEEE Virtual Reality Annual International Symposium.* Sept. 1993, pp. 76–82. DOI: 10.1109/VRAIS.1993.380795.

[60] Karen Scarfone, Murugiah Souppaya, et al. *SP 800-115: Technical Guide to Information Security Testing and Assessment.* National Institute of Standards and Technology. Sept. 2008.

[61] Darshan Shankar. *Bigscreen brings movie nights to VR with the debut of Top Gun in 3D*. Online. Bigscreen Blog, Dec. 2017. URL: https://blog.bigscreenvr.com/bigscreen-brings-movie-nights-to-vr-with-the-debut-of-top-gun-in-3d-df785a4e9764 (visited on 01/14/2019).

[62] Darshan Shankar. *Bigscreen raises $11 Million in Series A financing led by True Ventures*. Online. Bigscreen Blog, Oct. 2017. URL: https://blog.bigscreenvr.com/bigscreen-raises-11-million-in-series-a-financing-led-by-true-ventures-a2d49da40e (visited on 01/14/2019).

[63] Darshan Shankar. *The Bigscreen Beta „2019 Update“ is now live!* Online. Bigscreen Blog, Feb. 2019. URL: https://blog.bigscreenvr.com/the-bigscreen-beta-2019-update-is-now-live-a68ab6ceb506 (visited on 05/21/2019).

[64] Geoff Siskind and Bruce Snel. *Virtually Vulnerable*. Online. Hackable? McAfee, Nov. 2018. URL: https://hackablepodcast.com/episodes/virtually-vulnerable (visited on 01/14/2019).

[65] Andrei State, Mark A. Livingston, et al. “Technologies for Augmented Reality Systems: Realizing Ultrasound-guided Needle Biopsies”. In: *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques*. SIGGRAPH ’96. New York, NY, USA: ACM, 1996, pp. 439–446. ISBN: 0-89791-746-4. DOI: 10.1145/237170.237283.

[66] Andrew van der Stock, Brian Glas, et al. *OWASP Top 10 – 2017*. Online. The OWASP Foundation. 2017. URL: https://www.owasp.org/index.php/top10 (visited on 05/19/2019).

[67] Ivan E. Sutherland. “A Head-mounted Three Dimensional Display”. In: *Proceedings of the December 9-11, 1968, Fall Joint Computer Conference, Part I*. AFIPS ’68 (Fall, part I). San Francisco, California: ACM, 1968, pp. 757–764. DOI: 10.1145/1476589.1476686.

[68] Marek Valenta. “VR Interactive Application”. Czech. MA thesis. Brno, CZ: Brno University of Technology, Faculty of Information Technology, 2018. URL: http://www.fit.vutbr.cz/study/DP/DP.php?id=21178.

[69] Martin Vondráček. “Automation of MitM Attack on WiFi Networks”. Bachelor’s thesis. Brno, CZ: Brno University of Technology, Faculty of Information Technology, 2016. URL: http://www.fit.vutbr.cz/study/DP/BP.php?id=18596 (visited on 03/21/2019).

[70] Martin Vondráček, Jan Pluskal, and Ondřej Ryšavý. “Automated Man-in-the-Middle Attack Against Wi-Fi Networks”. In: *Journal of Digital Forensics, Security and Law* 13.1 (2018), pp. 59–80. DOI: 10.15394/jdfsl.2018.1495.

[71] Martin Vondráček, Jan Pluskal, and Ondřej Ryšavý. “Automation of MitM Attack on Wi-Fi Networks”. In: *Digital Forensics and Cyber Crime*. Ed. by Petr Matoušek and Martin Schmiedecker. Cham: Springer International Publishing, 2018, pp. 207–220. ISBN: 978-3-319-73697-6.

[72] Jaroslav Vrána. “Penetration Testing Application for DoS Based Web Vulnerabilities”. Czech. MA thesis. Brno, CZ: Brno University of Technology, Faculty of Information Technology, 2011. URL: http://www.fit.vutbr.cz/study/DP/DP.php?id=11542.

[73] Pierre Wellner. “Interacting with Paper on the DigitalDesk”. In: *Commun. ACM* 36.7 (July 1993), pp. 87–96. ISSN: 0001-0782. DOI: 10.1145/159544.159630.

[74] Owen Williams. *We figured out how to play Nintendo Switch in VR and it’s amazing.* Online. Bigscreen Blog, Nov. 2017. URL: https://blog.bigscreenvr.com/we-figured-out-how-to-play-nintendo-switch-in-vr-and-its-amazing-d2de9638bd19 (visited on 01/14/2019).

[75] *WPF Globalization and Localization Overview.* Online. Microsoft Docs, Mar. 2017. URL: https://docs.microsoft.com/en-us/dotnet/framework/wpf/advanced/wpf-globalization-and-localization-overview (visited on 01/14/2019).

[76] A. Yarramreddy, P. Gromkowski, and I. Baggili. “Forensic Analysis of Immersive Virtual Reality Social Applications: A Primary Account”. In: *2018 IEEE Security and Privacy Workshops (SPW).* May 2018, pp. 186–196. DOI: 10.1109/SPW.2018.00034.

[77] Dale Yurong. *Beware! Virtual reality games are susceptible to hacking, say experts.* Online. ABC Inc. KFSN-TV Fresno. Technology, Dec. 2018. URL: https://abc30.com/technology/beware-virtual-reality-games-are-susceptible-to-hacking-say-experts/4850104/ (visited on 01/14/2019).

[78] Xiaolu Zhang, Ibrahim Baggili, and Frank Breitinger. “Breaking into the vault: Privacy, security and forensic analysis of Android vault applications”. In: *Computers & Security* 70 (2017), pp. 516–531. ISSN: 0167-4048. DOI: 10.1016/j.cose.2017.07.011.

[79] Marek Zouhar. “Creation and Demonstration of Assets for VR Application”. Czech. MA thesis. Brno, CZ: Brno University of Technology, Faculty of Information Technology, 2017. URL: http://www.fit.vutbr.cz/study/DP/DP.php?id=20131.

Acronyms

A

ANN Artificial Neural Network.
API Application Programming Interface.
AR Augmented Reality.
ARP Address Resolution Protocol.
ASCII American Standard Code for Information Interchange.
AST Abstract Syntax Tree.

B

BAML Binary Application Markup Language.

C

C&C Command & Control.
CA Certificate Authority.
CEO Chief Executive Officer.
CIA Confidentiality, Integrity, Availability.
CIL Common Intermediate Language.
CLI Command-line Interface.
CRT Cathode-ray Tube.
CSS Cascading Style Sheets.

D

DHCP Dynamic Host Configuration Protocol.
DLL Dynamic Link Library.
DNS Domain Name System.
DoS Denial of Service.
DTLS Datagram Transport Layer Security.

F

FEAAS Forensic Evidence Acquisition and Analysis System.
FTP File Transfer Protocol.

G

GPL GNU General Public License.
GRE Generic Routing Encapsulation.

GUI Graphical User Interface.

H

HCI Human Computer Interaction.
HMD Head-mounted Display.
HRWI Human Real World Interaction.
HTML Hypertext Markup Language.
HTTP Hypertext Transfer Protocol.
HTTPS Hypertext Transfer Protocol Secure.

I

ICE Interactive Connectivity Establishment.
IDE Integrated Development Environment.
IP Internet Protocol.

J

JS JavaScript.
JSON JavaScript Object Notation.

L

LAN Local Area Network.

M

MIT Massachusetts Institute of Technology.
MitM Man-in-the-Middle.
MITMf Framework for Man-In-The-Middle Attacks.
MitR Man-in-the-Room.
MR Mixed Reality.

N

NDP Neighbor Discovery Protocol.
NFL National Football League.
NIST National Institute of Standards and Technology.

O

OS Operating System.
OSSTMM The Open Source Security Testing Methodology Manual.
OWASP Open Web Application Security Project.

P

P2P Peer to Peer.
PoC Proof of Concept.

R

RCE Remote Code Execution.
RDBMS Relational Database Management System.
RE Reverse Engineering.
REPL Read-eval-print loop.
REST Representational State Transfer.

S

SDP Session Description Protocol.
SMB Server Message Block.
SSL Secure Sockets Layer.
STUN Session Traversal Utilities for NAT.

T

TLS Transport Layer Security.
TOE Target of Evaluation.
TURN Traversal Using Relays around NAT.

U

UI User Interface.
UNHcFREG University of New Haven Cyber Forensics Research & Education Group.
URI Uniform Resource Identifier.
URL Uniform Resource Locator.

V

VB.NET Visual Basic .NET.
VE Virtual Environment.
VR Virtual Reality.
VRTK Virtual Reality Toolkit.

W

WAF Web Application Firewall.
WLAN Wireless Local Area Network.
WS WebSockets.
WSS Secure WebSockets.

X

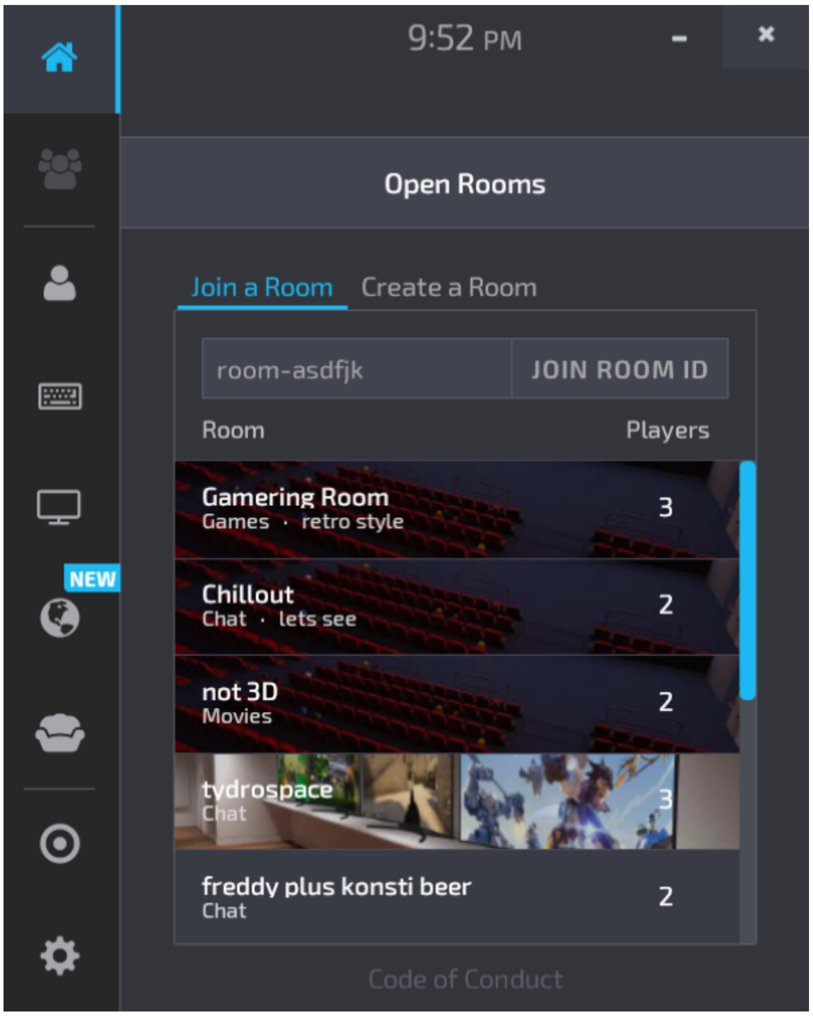
XAML Extensible Application Markup Language.
XHR XMLHttpRequest.
XSS Cross-site Scripting.

Appendices

List of Appendices

Appendix A

Bigscreen – Lobby

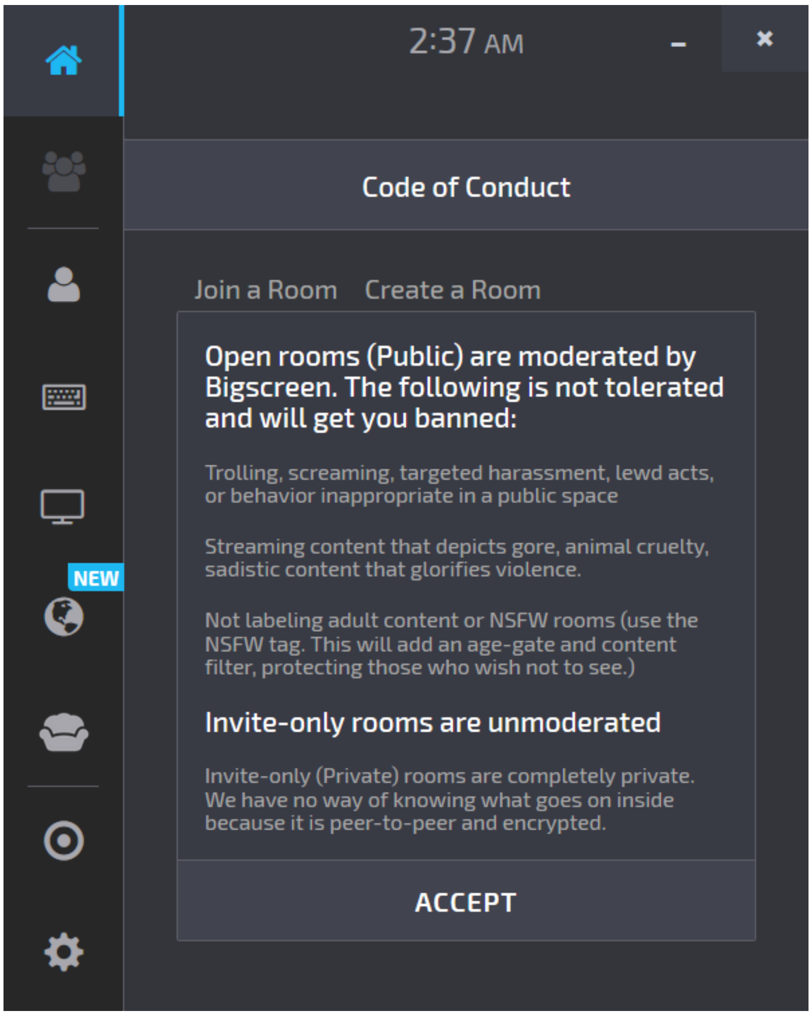


Room	Players
Gamering Room Games · retro style	3
Chillout Chat · lets see	2
not 3D Movies	2
tydrospace Chat	3
freddy plus konsti beer Chat	2

Figure A.1: Main screen of the Bigscreen application shows the lobby (list of all public rooms).

Appendix B

Bigscreen – Code of Conduct



Open rooms (Public) are moderated by Bigscreen. The following is not tolerated and will get you banned:

Trolling, screaming, targeted harassment, lewd acts, or behavior inappropriate in a public space

Streaming content that depicts gore, animal cruelty, sadistic content that glorifies violence.

Not labeling adult content or NSFW rooms (use the NSFW tag. This will add an age-gate and content filter, protecting those who wish not to see.)

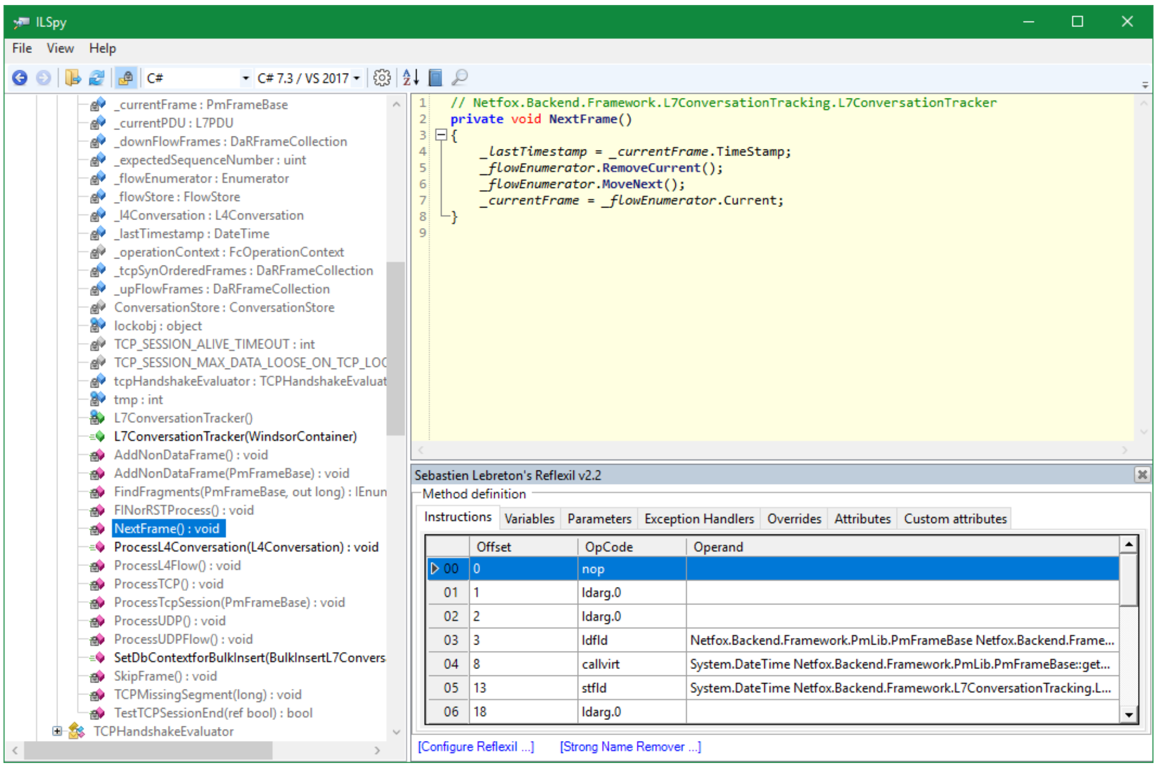
Invite-only rooms are unmoderated

Invite-only (Private) rooms are completely private. We have no way of knowing what goes on inside because it is peer-to-peer and encrypted.

Figure B.1: Bigscreen's Code of Conduct as available in Steam version of the application in October 2018. The text states that private rooms are invite-only and they utilise encrypted peer-to-peer communication.

Appendix C

ILSpy



```
_currentFrame : PmFrameBase
_currentPDU : L7PDU
_downFlowFrames : DaRFrameCollection
_expectedSequenceNumber : uint
_flowEnumerator : Enumerator
_flowStore : FlowStore
_l4Conversation : L4Conversation
_lastTimestamp : DateTime
_operationContext : FcOperationContext
_tcpSynOrderedFrames : DaRFrameCollection
_upFlowFrames : DaRFrameCollection
ConversationStore : ConversationStore
lockobj : object
TCP_SESSION_ALIVE_TIMEOUT : int
TCP_SESSION_MAX_DATA_LOOSE_ON_TCP_LOC
tcpHandshakeEvaluator : TCPHandshakeEvaluat
tmp : int
L7ConversationTracker()
L7ConversationTracker(WindsorContainer)
AddNonDataFrame() : void
AddNonDataFrame(PmFrameBase) : void
FindFragments(PmFrameBase, out long) : IEnun
FINorRSTProcess() : void
NextFrame() : void
ProcessL4Conversation(L4Conversation) : void
ProcessL4Flow() : void
ProcessTCP() : void
ProcessTcpSession(PmFrameBase) : void
ProcessUDP() : void
ProcessUDPFlow() : void
SetDbContextforBulkInsert(BulkInsertL7Convers
SkipFrame() : void
TCPMissingSegment(long) : void
TestTCPSessionEnd(ref bool) : bool
TCPHandshakeEvaluator
```

```csharp
// Netfox.Backend.Framework.L7ConversationTracking.L7ConversationTracker
private void NextFrame()
{
    _lastTimestamp = _currentFrame.TimeStamp;
    _flowEnumerator.RemoveCurrent();
    _flowEnumerator.MoveNext();
    _currentFrame = _flowEnumerator.Current;
}
```

Sebastien Lebreton's Reflexil v2.2

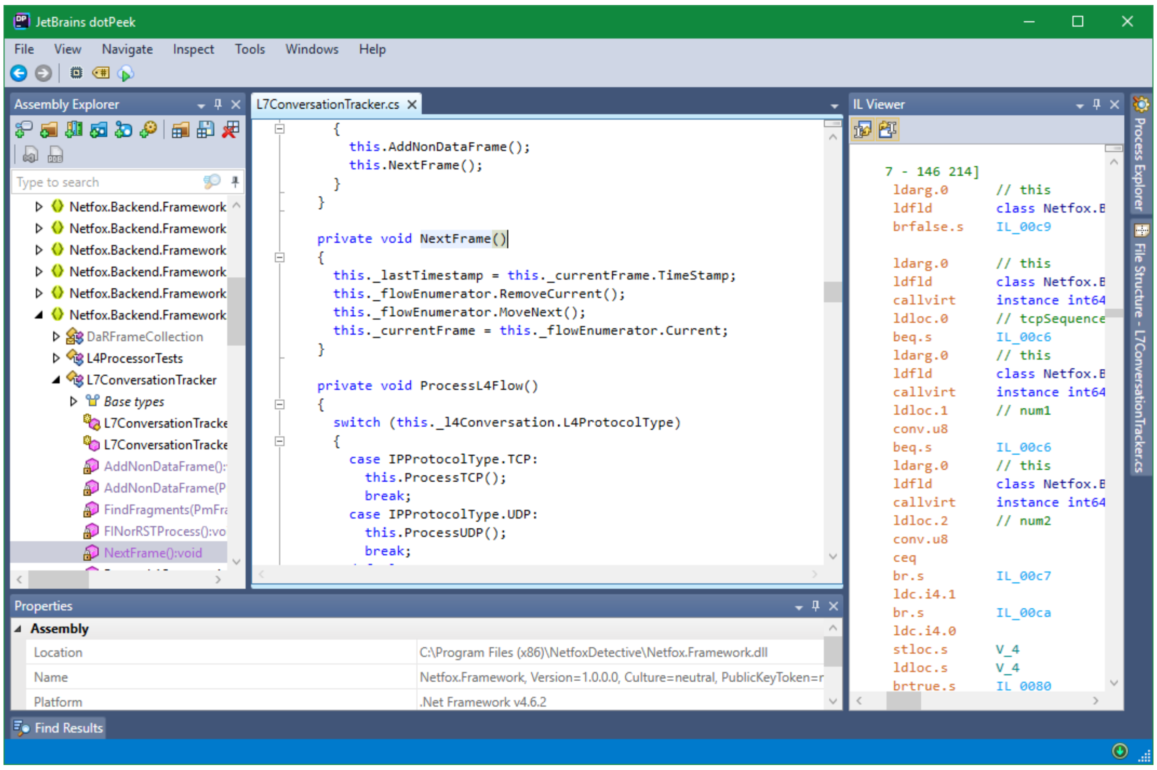
Method definition

	Offset	OpCode	Operand
00	0	nop	
01	1	ldarg.0	
02	2	ldarg.0	
03	3	ldfld	Netfox.Backend.Framework.PmLib.PmFrameBase Netfox.Backend.Frame...
04	8	callvirt	System.DateTime Netfox.Backend.Framework.PmLib.PmFrameBase::get...
05	13	stfld	System.DateTime Netfox.Backend.Framework.L7ConversationTracking.L...
06	18	ldarg.0	

Figure C.1: GUI of ILSpy 3.2.0.3856 application with enabled Reflexil 2.2 plugin.

Appendix D

JetBrains dotPeek



```
      {
        this.AddNonDataFrame();
        this.NextFrame();
      }
    }

    private void NextFrame()
    {
      this._lastTimestamp = this._currentFrame.TimeStamp;
      this._flowEnumerator.RemoveCurrent();
      this._flowEnumerator.MoveNext();
      this._currentFrame = this._flowEnumerator.Current;
    }

    private void ProcessL4Flow()
    {
      switch (this._l4Conversation.L4ProtocolType)
      {
        case IPProtocolType.TCP:
          this.ProcessTCP();
          break;
        case IPProtocolType.UDP:
          this.ProcessUDP();
          break;
```

```
7 - 146 214]
 ldarg.0      // this
 ldfld        class Netfox.B
 brfalse.s    IL_00c9

 ldarg.0      // this
 ldfld        class Netfox.B
 callvirt     instance int64
 ldloc.0      // tcpSequence
 beq.s        IL_00c6
 ldarg.0      // this
 ldfld        class Netfox.B
 callvirt     instance int64
 ldloc.1      // num1
 conv.u8
 beq.s        IL_00c6
 ldarg.0      // this
 ldfld        class Netfox.B
 callvirt     instance int64
 ldloc.2      // num2
 conv.u8
 ceq
 br.s         IL_00c7
 ldc.i4.1
 br.s         IL_00ca
 ldc.i4.0
 stloc.s      V_4
 ldloc.s      V_4
 brtrue.s     IL_0080
```

Properties

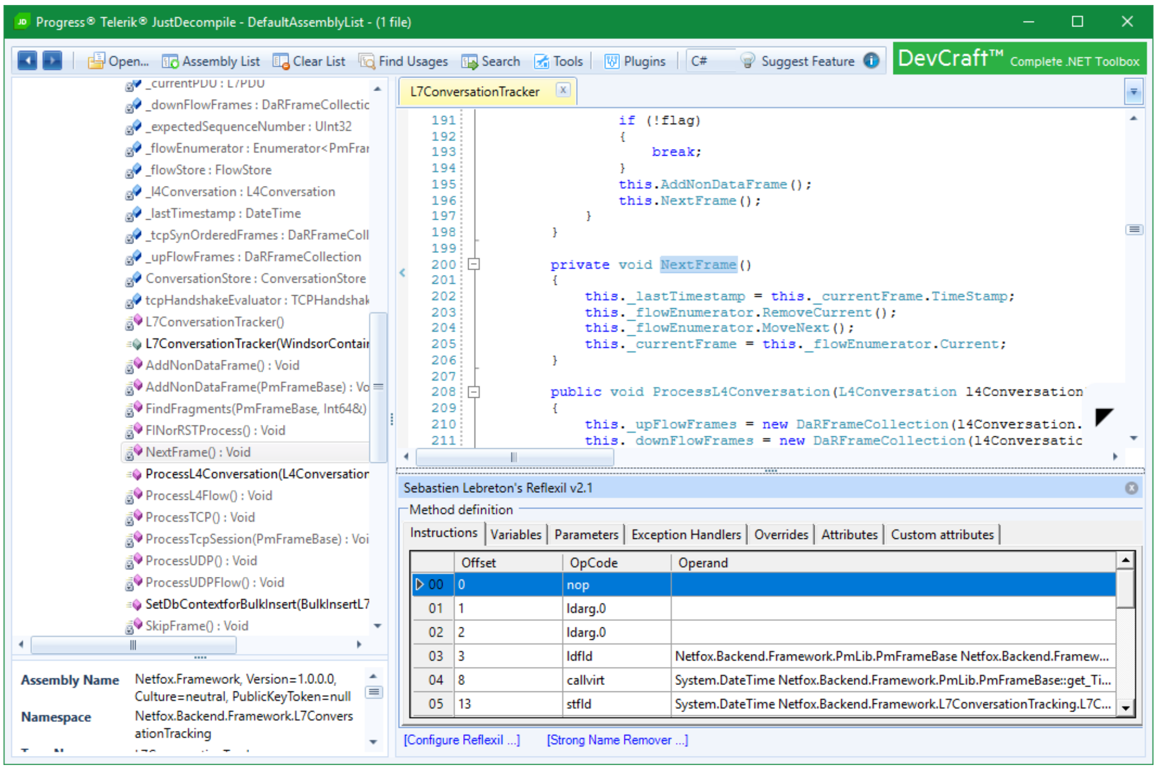
Assembly

Location	C:\Program Files (x86)\NetfoxDetective\Netfox.Framework.dll
Name	Netfox.Framework, Version=1.0.0.0, Culture=neutral, PublicKeyToken=r
Platform	.Net Framework v4.6.2

Figure D.1: JetBrains dotPeek 2018.2.3 shows C# source code of a decompiled assembly.

Appendix E

Progress Telerik JustDecompile



```
_currentPDU : L7PDU
_downFlowFrames : DaRFrameCollectic
_expectedSequenceNumber : UInt32
_flowEnumerator : Enumerator<PmFrar
_flowStore : FlowStore
_l4Conversation : L4Conversation
_lastTimestamp : DateTime
_tcpSynOrderedFrames : DaRFrameColl
_upFlowFrames : DaRFrameCollection
ConversationStore : ConversationStore
tcpHandshakeEvaluator : TCPHandshak
L7ConversationTracker()
L7ConversationTracker(WindsorContair
AddNonDataFrame() : Void
AddNonDataFrame(PmFrameBase) : Vo
FindFragments(PmFrameBase, Int64&)
FINorRSTProcess() : Void
NextFrame() : Void
ProcessL4Conversation(L4Conversatior
ProcessL4Flow() : Void
ProcessTCP() : Void
ProcessTcpSession(PmFrameBase) : Voi
ProcessUDP() : Void
ProcessUDPFlow() : Void
SetDbContextforBulkInsert(BulkInsertL7
SkipFrame() : Void
```

Assembly Name	Netfox.Framework, Version=1.0.0.0, Culture=neutral, PublicKeyToken=null
Namespace	Netfox.Backend.Framework.L7ConversationTracking

```
                if (!flag)
                {
                    break;
                }
                this.AddNonDataFrame();
                this.NextFrame();
            }
        }

        private void NextFrame()
        {
            this._lastTimestamp = this._currentFrame.TimeStamp;
            this._flowEnumerator.RemoveCurrent();
            this._flowEnumerator.MoveNext();
            this._currentFrame = this._flowEnumerator.Current;
        }

        public void ProcessL4Conversation(L4Conversation l4Conversation
        {
            this._upFlowFrames = new DaRFrameCollection(l4Conversation.
            this._downFlowFrames = new DaRFrameCollection(l4Conversatic
```

Sebastien Lebreton's Reflexil v2.1

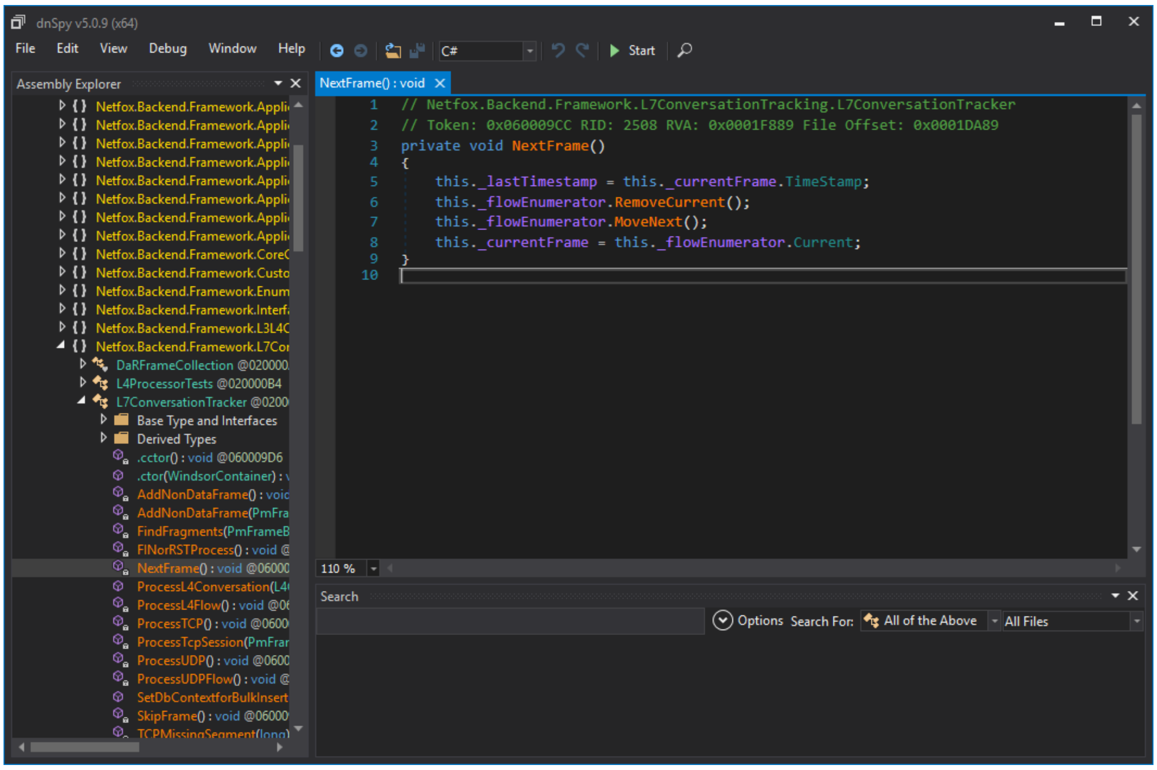
Method definition

	Offset	OpCode	Operand
00	0	nop	
01	1	ldarg.0	
02	2	ldarg.0	
03	3	ldfld	Netfox.Backend.Framework.PmLib.PmFrameBase Netfox.Backend.Framew...
04	8	callvirt	System.DateTime Netfox.Backend.Framework.PmLib.PmFrameBase::get_Ti...
05	13	stfld	System.DateTime Netfox.Backend.Framework.L7ConversationTracking.L7C...

Figure E.1: View of a C# assembly which was decompiled using *Progress Telerik JustDecompile* version 2018.2.803.0 with *Assembly Editor Plugin* (bottom right panel) enabled.

Appendix F

dnSpy



```
Netfox.Backend.Framework.Appli
Netfox.Backend.Framework.Appli
Netfox.Backend.Framework.Appli
Netfox.Backend.Framework.Appli
Netfox.Backend.Framework.Appli
Netfox.Backend.Framework.Appli
Netfox.Backend.Framework.Appli
Netfox.Backend.Framework.Appli
Netfox.Backend.Framework.CoreC
Netfox.Backend.Framework.Custo
Netfox.Backend.Framework.Enum
Netfox.Backend.Framework.Interf
Netfox.Backend.Framework.L3L4C
Netfox.Backend.Framework.L7Co
  DaRFrameCollection @020000
  L4ProcessorTests @020000B4
  L7ConversationTracker @0200
    Base Type and Interfaces
    Derived Types
    .cctor() : void @060009D6
    .ctor(WindsorContainer) :
    AddNonDataFrame() : voi
    AddNonDataFrame(PmFra
    FindFragments(PmFrameB
    FINorRSTProcess() : void @
    NextFrame() : void @06000
    ProcessL4Conversation(L4
    ProcessL4Flow() : void @0
    ProcessTCP() : void @0600
    ProcessTcpSession(PmFrar
    ProcessUDP() : void @0600
    ProcessUDPFlow() : void @
    SetDbContextforBulkInsert
    SkipFrame() : void @06000
    TCPMissingSegment(long)
```

```csharp
// Netfox.Backend.Framework.L7ConversationTracking.L7ConversationTracker
// Token: 0x060009CC RID: 2508 RVA: 0x0001F889 File Offset: 0x0001DA89
private void NextFrame()
{
    this._lastTimestamp = this._currentFrame.TimeStamp;
    this._flowEnumerator.RemoveCurrent();
    this._flowEnumerator.MoveNext();
    this._currentFrame = this._flowEnumerator.Current;
}
```

Figure F.1: Main windows of dnSpy v5.0.9 during reverse engineering of a C# assembly.

Appendix G

Overview of Tools for Minification and Obfuscation of JavaScript

Selected JS minifiers:

1. JSMin,
   `http://www.crockford.com/javascript/jsmin.html`
2. Closure Compiler,
   `https://developers.google.com/closure/compiler/`
3. YUI Compressor,
   `http://yui.github.io/yuicompressor/`
4. UglifyJS,
   `http://lisperator.net/uglifyjs/`
5. Minify,
   `https://www.minifier.org/`
6. babel-minify,
   `https://github.com/babel/minify`
7. Terser,
   `https://github.com/terser-js/terser`
8. Pretty Diff,
   `https://github.com/prettydiff/prettydiff/`

Selected tools for JS obfuscation:

1. JavaScript obfuscator from Timofey Kachalov,
   `https://obfuscator.io/`
2. JavaScript Utility V3,
   `http://jsutility.pjoneil.net/`

3. Javascript Obfuscator from CuteSoft Components Inc.,
   http://www.javascriptobfuscator.com/

4. Free JS Obfuscator,
   http://www.freejsobfuscator.com/

5. _Number described for example by user *baivong*,
   https://jsfiddle.net/ps5anL99/embedded/result,js,html,css/

6. aaencode,
   http://utf-8.jp/public/aaencode.html

7. jjencode,
   http://utf-8.jp/public/jjencode.html

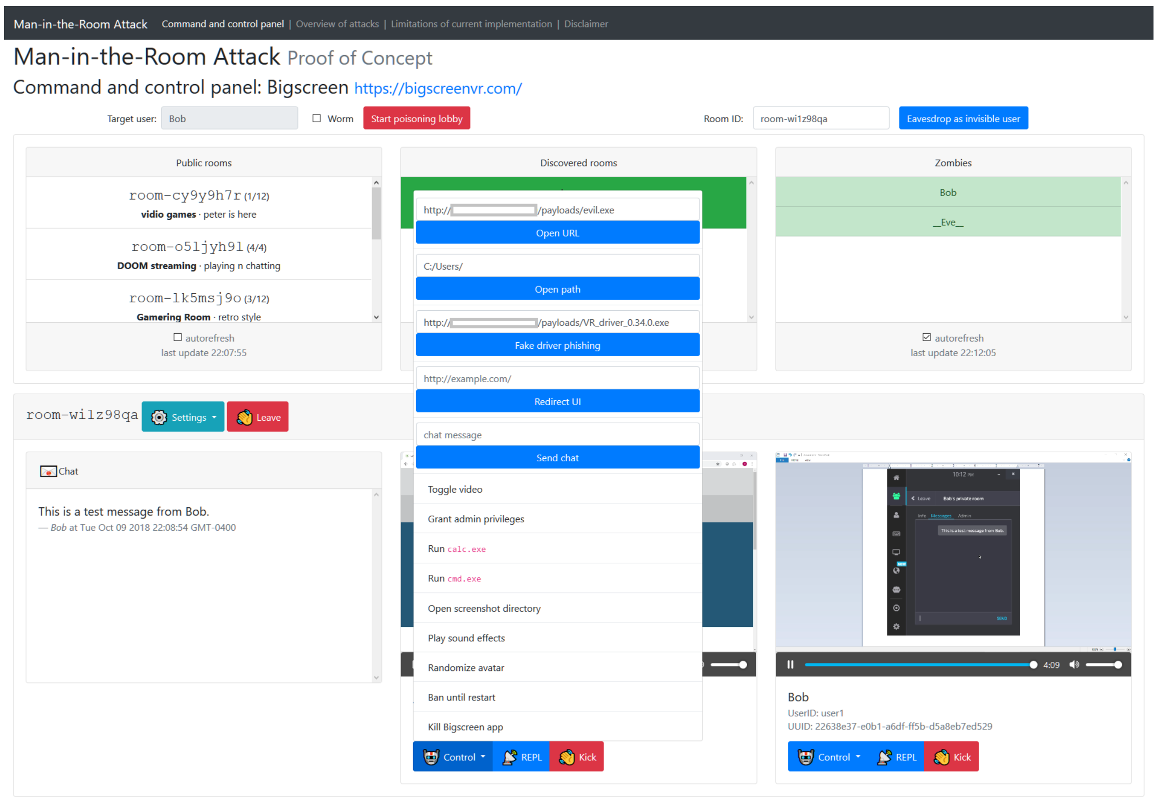
8. JSFuck,
   http://www.jsfuck.com/

9. Hieroglyphy,
   https://github.com/alcuadrado/hieroglyphy

10. WiseLoop PHP Javascript Obfuscator,
    https://wiseloop.com/demo/php-javascript-obfuscator

Appendix H

Proof of Concept Command & Control Server



Man-in-the-Room Attack Proof of Concept

Command and control panel: Bigscreen https://bigscreenvr.com/

Target user: Bob

Room ID: room-wi1z98qa

room-cy9y9h7r (1/12)
vidio games · peter is here

room-o5ljyh9l (4/4)
DOOM streaming · playing n chatting

room-lk5msj9o (3/12)
Gamering Room · retro style

last update 22:07:55

http://[redacted]/payloads/evil.exe

C:/Users/

http://[redacted]/payloads/VR_driver_0.34.0.exe

http://example.com/

chat message

Bob

Eve

last update 22:12:05

room-wi1z98qa

This is a test message from Bob.
— *Bob at Tue Oct 09 2018 22:08:54 GMT-0400*

Bob
UserID: user1
UUID: 22638e37-e0b1-a6df-ff5b-d5a8eb7ed529

Figure H.1: Developed PoC C&C server which can control a botnet of infected Bigscreen applications. This demonstrative attacking tool can also receive victims' multimedia (fig. 5.10). This dashboard is also illustrated in the demonstration video available at the address https://www.youtube.com/watch?v=N_Z3mfzLZME.

Appendix I

Summary of Findings

Table I.1: Network connections

Endpoint	Encryption	Authentication & Authorization
`http://prod.bigscreenvr.com/`	×	×
`http://signal.bigscreenvr.com/`	×	×
`http://ip.bigscreenvr.com/`	×	×
`https://signal3.bigscreenvr.com/`	*Out of order*	
`https://signal2adm.bigscreenvr.com/`	✓	×
`https://signal2.bigscreenvr.com/`	✓	×
`wss://signal2.bigscreenvr.com`	✓	×
P2P WebRTC channels	✓	×

Table I.2: Main vulnerabilities of the application

Vulnerability	Context	Severity
RCE via `Application.OpenURL`	Unity engine	High
XSS in *user name*, *room name*, *room description*, and *room category*	UI layer	High
Information leak via RE of assemblies	Application core	Medium
Information leak via RE of obfuscated and minified JS source code	UI layer	Low
Patching DLLs without integrity check	DLLs	High
Lack of integrity, receiving data without sharing any VR state	WebRTC	High
Lack of authentication, connection from a custom application	WebRTC	Medium
Lack of authentication, connection from a custom application	Signaling channel	High

Table I.3: Developed exploits based on XSS vulnerability

Category	**Attack/Exploit**	**Severity**
Botnet	Control infected applications from C&C server.	High
VR Worm	Spread a worm infection through the whole Bigscreen community.	High
JS RCE	Remotely execute any JS code in the UI layer of Bigscreen.	High
Privacy violation	Discover private rooms.	High
Privacy violation	Eavesdrop computer screen, computer audio, and microphone audio.	High
Privacy violation	Persistently eavesdrop victim's chat, even if they go to another room.	High
Phishing	Ask victim to install "required VR driver". *This phishing is not related to RCE in Unity.*	High
Privacy violation	Toggle video, audio, and microphone sharing.	High
Privacy violation	Change signaling servers of victim's Bigscreen application.	High
DoS	Remotely terminate Bigscreen application.	Medium
Impersonation	Spoof chat messages.	Medium
Privilege escalation	Set selected user as room admin.	Medium
Phishing	Redirect Bigscreen's UI to any webpage.	Medium
Privacy violation	Gather all victim's logs.	Medium
DoS	Ban selected victim until restart.	Low
Miscellaneous	Change victim's avatar.	Low
Miscellaneous	Play sound effects from Bigscreen's UI.	Low

Table I.4: Developed exploits based on the lack of authentication & authorization in the signaling channel

Category	**Attack/Exploit**	**Severity**
DoS	Kick any user from any room.	High
Privilege escalation	Change room's settings (VR locks).	Medium
Resource exhaustion	Automatically create new rooms.	Low

Table I.5: Developed exploits based on the lack of request limits

Category	**Attack/Exploit**	**Severity**
Privacy violation	Enumerate (brute force) room ID of private rooms.	Low

Table I.6: Developed exploits based on the combination of XSS, RCE in Unity, or application patching (application crippling)

Category	**Attack/Exploit**	**UI XSS**	**Unity RCE**	**App. Patch**	**Severity**
Privacy violation	Man-in-the-Room.	✓	×	✓	High
RCE	Download & execute malware.	✓	✓	×	High
RCE	Run any program and open any folder or file.	✓	✓	×	High

Appendix J

Overview of News Covering the Research

1. Bigscreen, Blog,
   https://blog.bigscreenvr.com/the-bigscreen-beta-2019-update-is-now-live-a68ab6ceb506
2. University of New Haven, University News,
   http://www.newhaven.edu/news/releases/2019/discover-vulnerabilities-virtual-reality-app.php
3. University of New Haven Cyber Forensics Research & Education Group, Blog,
   https://www.unhcfreg.com/single-post/2019/02/19/BigScreen-and-Unity-Virtual-Reality-Attacks
4. UNHcFREG YouTube, Man-in-the-Room Attack & Command and Control Server Proof of Concept,
   https://www.youtube.com/watch?v=N_Z3mfzLZME
5. PR Newswire: press release distribution, targeting, monitoring and marketing,
   https://www.prnewswire.com/news-releases/university-of-new-haven-researchers-discover-critical-vulnerabilities-in-popular-virtual-reality-application-300798914.html
6. Yahoo Finance - Business Finance, Stock Market, Quotes, News,
   https://finance.yahoo.com/news/university-haven-researchers-discover-critical-vulnerabilities-popular-virtual-153000147.html
7. Markets Insider: Stock Market News, Realtime Quotes and Charts,
   https://markets.businessinsider.com/news/stocks/university-of-new-haven-researchers-discover-critical-vulnerabilities-in-popular-virtual-reality-application-1027968753
8. finanzen.ch: Realtimekurse | Aktien | Börsenkurse | Börse,
   https://www.finanzen.ch/nachrichten/aktien/university-of-new-haven-researchers-discover-critical-vulnerabilities-in-popular-virtual-reality-application-1027968753

9. KITV Channel 4,
https://www.kitv.com/story/39995438/university-of-new-haven-researchers-discover-critical-vulnerabilities-in-popular-virtual-reality-application

10. Morningstar | Independent Investment Research,
https://www.morningstar.com/news/pr-news-wire/PRNews_20190220DC59728/university-of-new-haven-researchers-discover-critical-vulnerabilities-in-popular-virtual-reality-application.html

11. West Haven Voice,
https://westhavenvoice.com/popular-app-prone-to-hacking-unh/

12. CSO Online,
https://www.csoonline.com/article/3342418/security/meet-the-man-in-the-room-attack-hackers-can-invisibly-eavesdrop-on-bigscreen-vr-users.html

13. PICANTE Today - Hot News Today,
https://picante.today/latest-news/2019/02/21/18976/university-of-new-haven-researchers-discover-critical-vulnerabilities-in-popular-virtual-reality-application/

14. IT Security News - cybersecurity, infosecurity news,
https://www.itsecuritynews.info/meet-the-man-in-the-room-attack-hackers-can-invisibly-eavesdrop-on-bigscreen-vr-users/

15. VRScout - Virtual Reality News and VR Videos,
https://vrscout.com/news/security-flaws-bigscreen-vr-app/

16. TechRistic.com – Technology news, analysis and review,
https://www.techristic.com/researchers-discover-security-flaws-in-bigscreen-vr-app/

17. TerabitWeb Blog – Interesting and Fascinating Technology and Security Information,
https://www.terabitweb.com/2019/02/21/critical-vulnerabilities-in-bigscreen-vr-app-unity-allow-eavesdropping-man-in-the-room-attacks/

18. SC Media | Cybersecurity News, analysis and Product Reviews,
https://www.scmagazine.com/home/security-news/critical-vulnerabilities-in-bigscreen-vr-app-unity-allow-eavesdropping-man-in-the-room-attacks/

19. MRTV - Mixed Reality TV - VR & AR Hardware Reviews,
https://mrtv.co/2019/02/researchers-discover-security-flaws-in-bigscreen-vr-app/

20. ANITH – Cyber Land of Anith Gopal,
https://anith.com/meet-the-man-in-the-room-attack-hackers-can-invisibly-eavesdrop-on-bigscreen-vr-users/

21. Secplicity - Security Simplified, Security Byte,
https://www.secplicity.org/2019/02/22/virtual-reality-vulnerability-bigscreen-security-byte/

22. The Hacker News — Cyber Security and Hacking News Website,
https://thehackernews.com/2019/02/bigscreen-vr-hacking.html

23. Road to VR,
https://www.roadtovr.com/bigscreen-patches-potential-hack-spelled-disaster-platform/

24. PentestTools,
https://pentesttools.net/hacking-virtual-reality-researchers-exploit-popular-bigscreen-vr-app/

25. Security in News,
https://securityinnews.com/2019/02/22/hacking-virtual-reality-researchers-exploit-popular-bigscreen-vr-app/

26. Sean Lister | Just another SysAdmin automating tasks,
https://www.seanlister.co.uk/2019/02/22/hacking-virtual-reality-researchers-exploit-popular-bigscreen-vr-app/

27. Tech News Fix – Your Daily Source of Tech News,
http://www.technewsfix.com/2019/02/22/virtual-reality-vulnerability-bigscreen-security-byte/

28. ETHHack, Hacking & Cyber Security
https://ethhack.com/2019/02/hacking-virtual-reality-researchers-exploit-popular-bigscreen-vr-app/

29. Business Risk Intelligence & Cyberthreat Awareness,
https://brica.de/alerts/alert/public/1248451/meet-the-man-in-the-room-attack-hackers-can-invisibly-eavesdrop-on-bigscreen-vr-users/

30. CoLaBug,
https://www.colabug.com/5540245.html

31. IT Security Guru,
https://www.itsecurityguru.org/2019/02/22/critical-vulnerabilities-in-bigscreen-vr-app/

32. Leiphone.com,
https://www.leiphone.com/news/201902/NRltNgvS7b6jnf3c.html

33. TerabitWeb Blog – Interesting and Fascinating Technology and Security Information,
https://www.terabitweb.com/2019/02/22/bigscreen-vr-hacking-html/

34. Jishuwen,
https://www.jishuwen.com/d/2UNz/zh-tw

35. VRStation – Site Berita VR pertama di Indonesia tentang hardware, software, game, dan industri virtual reality dalam bahasa Indonesia!
https://vrstation.id/2019/02/22/peneliti-temukan-kesalahan-keamanan-di-aplikasi-bigscreen-vr/

36. kknews.cc,
https://kknews.cc/tech/agvznpg.html

37. TrovaPassword, Blog,
https://trovapassword.com/blog/hacking-virtual-reality-researchers-exploit-popular-bigscreen-vr-app/

38. scr3,
http://secure-info3.blogspot.com/2019/02/hacking-virtual-reality-researchers.html

39. TechnoEXPRESS, Security News,
https://www.technoexpress.net/hacking-virtual-reality-researchers-exploit-popular-bigscreen-vr-app.html

40. Responsible Cyber,
https://responsible-cyber.com/2019/02/22/hackers-can-invisibly-eavesdrop-on-bigscreen-vr-users/

41. cloud.tencent.com,
https://cloud.tencent.com/developer/news/397976

42. SpareReality.com - VR Community,
https://sparereality.com/threads/bigscreen-patches-potential-hack-that-could-have-spelled-disaster-for-the-platform.1458/

43. Picture Virtual Reality,
https://pxvr.com/researchers-discover-security-flaws-in-bigscreen-vr-app/

44. Ikanchai.com,
http://www.ikanchai.com/article/20190223/269228.shtml

45. Masterhacks Blog – Un mundo de información a tu alcance,
https://blogs.masterhacks.net/noticias/hacking-y-ciberdelitos/hackean-la-popular-aplicacion-de-realidad-virtual-bigscreen/

46. EthicalHacktTech,
https://ethicalhacktech.wixsite.com/website/post/hacking-virtual-reality-researchers-exploit-popular-big-screen-vr-app

47. Prodefence - Cyber Security Services | Malware & Pentesting,
https://www.prodefence.org/hacking-virtual-reality-researchers-exploit-popular-bigscreen-vr-app/

48. Cyber Security Reviews,
https://cybersecurityreviews.net/2019/02/24/hacking-virtual-reality-researchers-exploit-popular-bigscreen-vr-app/

49. Hack Hex — Updates and Insights on Technology,
https://hackhex.com/security/heres-how-researchers-exploit-virtual-reality-application-4746.html

50. MoguLive, MoguraVR,
https://www.moguravr.com/social-vr-hacking-measures/

51. WinCpr.com,
https://wincpr.com/researchers-exploit-popular-bigscreen-vr-app/

52. LIFARS | Incident Response | Digital Forensics | Penetration Test | Threat Hunting,
https://lifars.com/2019/02/virtual-reality-app-found/-with-critical-vulnerabilities/

53. Bijus.net,
http://www.bijus.net/news/2019/02/25/638574.html

54. Cyber Safe Latest News, Cyber Security, Information Security, IT Security and Hacking News,
https://www.cybersafe.page/hacking-virtual-reality-researchers-exploit-bigscreen-vr-app/

55. Global Cybersec | Ciberseguridad,
https://www.globalcybersec.com/reader.php?p=5596

56. krazy2hack, Hacking Tips & Tricks,
http://krazy2hacking.blogspot.com/2019/02/hacking-virtual-reality-researchers.html

57. Hak4u,
https://hak4u.blogspot.com/2019/02/hacking-virtual-reality-researchers.html

58. Cyber Freakz | Tech News,
https://news.cyberfreakz.com/hacking-virtual-reality-researchers-exploit-popular-bigscreen-vr-app/

59. Static Networks,
https://www.staticnetworks.com/hacking-virtual-reality-researchers-exploit-popular-bigscreen-vr-app/

60. Aktuálně.cz, Ekonomika,
https://zpravy.aktualne.cz/ekonomika/cesky-expert-objevil-novy-typ-kyberutoku-virtualni-realita/r~6d3a12ac3dc111e9ab10ac1f6b220ee8

61. CSC 301 – News and Notes,
http://colleenwrh205.com/blog-4/

62. Phone City Magazine,
http://www.phonecity.com.kh/2019/03/08/bigscreen-vr-hacking/

63. DZone: Programming & DevOps news, tutorials & tools,
https://dzone.com/articles/api-security-weekly-issue-22

64. API Security Articles, News, Vulnerabilities & Best Practices,
https://apisecurity.io/issue-22-sans-swat-list-42crunch-platform-launch/

65. SECURITY magazín,
https://www.securitymagazin.cz/security/agent-nasazeny-do-vasi-virtualni-schuzky-brnensky-student-odhalil-s-kolegy-novy-typ-nebezpecneho-kyberutoku-1404062783.html

66. Český rozhlas Plus,
Interview at time 6 : 25 : 45*.*
https://program.rozhlas.cz/zaznamy#/plus/2019-03-20

67. Český rozhlas, iROZHLAS,
https://www.irozhlas.cz/veda-technologie/technologie/virtualni-realita-moderni-technologie-big-screen-bezpecnostni-chyba-brno-vut_1903240700_och

68. Zprávy z VUT, Nápady a objevy,
https://zvut.cz/napady-objevy/napady-a-objevy-f38103/jake-nebezpeci-hrozi-ve-virtualni-realite-student-z-fit-pomohl-americkym-vedcum-odhalit-nedostatecne-zabezpeceni-d184923

69. ČeskéNoviny.cz,
https://www.ceskenoviny.cz/zpravy/cesky-student-v-usa-pomohl-odhalit-slabiny-virtualni-reality/1746262

70. Česká televize, ČT24
https://ct24.ceskatelevize.cz/veda/2790152-brnensky-student-nasel-chyby-v-programech-pro-virtualni-realitu-pomohl-tak-predejit

71. Aktuálně.cz, Zprávy,
https://zpravy.aktualne.cz/domaci/cesky-student-v-usa-exceloval-pomohl-odhalit-slabiny-virtual/r~66c3e4a060f511e9a305ac1f6b220ee8/

72. Novinky.cz,
https://www.novinky.cz/internet-a-pc/bezpecnost/502778-cesky-student-pomohl-v-usa-odhalit-slabiny-virtualni-reality.html

73. Týden.cz,
https://www.tyden.cz/rubriky/veda/cesky-student-v-usa-pomohl-odhalit-slabiny-virtualni-reality_519780.html

74. Česká škola,
http://www.ceskaskola.cz/2019/04/cesky-student-v-usa-pomohl-odhalit.html

75. ČTUSI.INFO,
http://www.ctusi.info/zpravodajstvi/cestovani/144/cesky_student_v_usa_pomohl_odhalit_slabiny_virtualni_reality/

76. Kečtení.cz, Z domova
http://clanky.kecteni.cz/3206165-cesky-student-v-usa-pomohl-odhalit-slabiny-virtualni-reality

77. Secplicity - Security Simplified, The 443 Podcast,
https://www.secplicity.org/2019/04/22/hacking-vr-with-unhcfreg/

78. IT SECURITY NETWORK NEWS,
https://www.itsec-nn.com/vyzkum-bezpecnostni-chyby-v-aplikacich-pro-virtualni-realitu/

79. Chip,
Article in a printed version of the magazine, 05/2019.
https://www.chip.cz/casopis-chip/05-2019/

80. Dvojklik, Virus Lab,
https://www.dvojklik.cz/tym-vedcu-objevil-zranitelnosti-ve-virtualni-realite-vcetne-noveho-typu-utoku/

81. Kyberbezpečnost,
Vondráček's article "Bezpečnostní slabiny v technologiích pro virtuální realitu ohrozily přes 500 000 počítačů" (Security Weaknesses in Virtual Reality Technologies Have Threatened Over 500,000 Computers) was published in May on this website.
https://www.kyberbezpecnost.cz/?p=27519

82. IT Systems,
Vondráček's article "Bezpečnostní slabiny software pro telekonference a spolupráci ve virtuální realitě ohrozily přes 500 000 počítačů" (Security Weaknesses of Teleconferencing and Collaboration in Virtual Reality Have Threatened Over 500,000 Computers) was accepted for publication in June.
https://www.systemonline.cz/casopis-it-systems/

Appendix K

Selection of News From the Czech Republic Covering the Research

The author of this thesis also directed efforts to raise public awareness and popularising VR security by communication with media. His direct efforts caused that media in the Czech Republic published 18 articles, posts, and interviews (May 2019), as summarised below. Additional popularisation and media coverage is currently planned. He's also author of two informal and popularising articles about this VR security analysis (appendix L). Following points highlight news from the Czech Republic covering this research.

1. Aktuálně.cz, Ekonomika
   `https://zpravy.aktualne.cz/ekonomika/cesky-expert-objevil-novy-typ-kyberutoku-virtualni-realita/r~6d3a12ac3dc111e9ab10ac1f6b220ee8`
2. SECURITY magazín,
   `https://www.securitymagazin.cz/security/agent-nasazeny-do-vasi-virtualni-schuzky-brnensky-student-odhalil-s-kolegy-novy-typ-nebezpecneho-kyberutoku-1404062783.html`
3. Český rozhlas Plus,
   Interview at time $6:25:45$.
   `https://program.rozhlas.cz/zaznamy#/plus/2019-03-20`
4. Český rozhlas, iROZHLAS,
   `https://www.irozhlas.cz/veda-technologie/technologie/virtualni-realita-moderni-technologie-big-screen-bezpecnostni-chyba-brno-vut_1903240700_och`
5. Zprávy z VUT, Nápady a objevy,
   `https://zvut.cz/napady-objevy/napady-a-objevy-f38103/jake-nebezpeci-hrozi-ve-virtualni-realite-student-z-fit-pomohl-americkym-vedcum-odhalit-nedostatecne-zabezpeceni-d184923`
6. ČeskéNoviny.cz,
   `https://www.ceskenoviny.cz/zpravy/cesky-student-v-usa-pomohl-odhalit-slabiny-virtualni-reality/1746262`

7. Česká televize, ČT24
   https://ct24.ceskatelevize.cz/veda/2790152-brnensky-student-nasel-chyby-v-programech-pro-virtualni-realitu-pomohl-tak-predejit

8. Aktuálně.cz, Zprávy,
   https://zpravy.aktualne.cz/domaci/cesky-student-v-usa-exceloval-pomohl-odhalit-slabiny-virtual/r~66c3e4a060f511e9a305ac1f6b220ee8/

9. Novinky.cz,
   https://www.novinky.cz/internet-a-pc/bezpecnost/502778-cesky-student-pomohl-v-usa-odhalit-slabiny-virtualni-reality.html

10. Týden.cz,
    https://www.tyden.cz/rubriky/veda/cesky-student-v-usa-pomohl-odhalit-slabiny-virtualni-reality_519780.html

11. Česká škola,
    http://www.ceskaskola.cz/2019/04/cesky-student-v-usa-pomohl-odhalit.html

12. ČTUSI.INFO,
    http://www.ctusi.info/zpravodajstvi/cestovani/144/cesky_student_v_usa_pomohl_odhalit_slabiny_virtualni_reality/

13. Kečtení.cz, Z domova
    http://clanky.kecteni.cz/3206165-cesky-student-v-usa-pomohl-odhalit-slabiny-virtualni-reality

14. IT SECURITY NETWORK NEWS,
    https://www.itsec-nn.com/vyzkum-bezpecnostni-chyby-v-aplikacich-pro-virtualni-realitu/

15. Chip,
    Article in a printed version of the magazine, 05/2019.
    https://www.chip.cz/casopis-chip/05-2019/

16. Dvojklik, Virus Lab,
    https://www.dvojklik.cz/tym-vedcu-objevil-zranitelnosti-ve-virtualni-realite-vcetne-noveho-typu-utoku/

17. Kyberbezpečnost,
    Vondráček's article "Bezpečnostní slabiny v technologiích pro virtuální realitu ohrozily přes 500 000 počítačů" (Security Weaknesses in Virtual Reality Technologies Have Threatened Over 500,000 Computers) was published in May on this website.
    https://www.kyberbezpecnost.cz/?p=27519

18. IT Systems,
    Vondráček's article "Bezpečnostní slabiny software pro telekonference a spolupráci ve virtuální realitě ohrozily přes 500 000 počítačů" (Security Weaknesses of Teleconferencing and Collaboration in Virtual Reality Have Threatened Over 500,000 Computers) was accepted for publication in June.
    https://www.systemonline.cz/casopis-it-systems/

Appendix L

Author’s Informal and Popularising Articles About the Research

Author of this thesis is also author of following informal and popularising articles which inform about this research.

1. Kyberbezpečnost,
   Vondráček’s article “Bezpečnostní slabiny v technologiích pro virtuální realitu ohrozily přes 500 000 počítačů” (Security Weaknesses in Virtual Reality Technologies Have Threatened Over 500,000 Computers) was published in May on this website.
   https://www.kyberbezpecnost.cz/?p=27519

2. IT Systems,
   Vondráček’s article “Bezpečnostní slabiny software pro telekonference a spolupráci ve virtuální realitě ohrozily přes 500 000 počítačů” (Security Weaknesses of Teleconferencing and Collaboration in Virtual Reality Have Threatened Over 500,000 Computers) was accepted for publication in June.
   https://www.systemonline.cz/casopis-it-systems/