MPhil Thesis

November 1, 2021

Title:

Cheat detection and security in video games

MPhil Thesis

By

Md Rashedul Hassan

Student ID: 100805742

Under the supervision of

Professor Dr. Konstantinos Markantonakis

ISG, RHUL

Contents

List of Figures

1 Introduction

Video games are a source of entertainment for a wide range of people, starting from children to adults. There exist different types and genres of game in the current industry, and these games are available in multiple mediums. To go more in-depth, a typical video game can be played in a hand-held device, such as the Nintendo DS or on a TV connected console, such as the PlayStation 5. The software of the game can exist in multiple formats, ranging from a hardware chipset, CD/DVD or simply an application package downloaded from the internet.

Retro arcade-based video games contained the video game software within a chipset or the read-only memory (ROM) itself. These consoles or devices are generally air-gapped and have no connection to the internet or game developers. Once these games are manufactured and sent out to the consumer, they are permanent. The permanent nature of the game is not an issue when the game is played purely for entertainment purposes; however when there is a competitive element to the game, this could yield unfair results. The permanent nature of the game would mean that, should there be a gaming competition on that device which could potentially award the winner with a lump sum of money or gift items, the security of the device or the game would come into question.

Since the game is stored in ROM, any kind of security patch or update is out of the question, other than replacing the whole device itself. This is a potential security flaw.

Beyond retro games, such as the games played on PlayStation or Xbox, the game consoles are usually connected to the internet, and the game developers have more flexibility in patching security loopholes with intermittent updates. This methodology does solve the problem to a large degree compared the games stored in ROM.

As we start discussing the variety of mediums of video games in different types of devices, e.g. retro arcade, hand-held devices such as an iPhone or even PlayStation, we can see the video game part can be misconstrued to be sometimes referred to as firmware (ROM) based or just application based (PlayStation/Xbox).

In this thesis, we focus on the security of video games. However, the domain of video games can be vast. It will be beyond the scope of this thesis to discuss all the variants of video games. Therefore, we will begin by trying to categorise where the video game falls into - software or a firmware, or both - and then pick one of the smaller domain to discuss in detail. To achieve this, we study the taxonomy of firmware and shortlist the branches so we can target a smaller portion of the game industry and move on to discuss the security and vulnerability of that domain.

1.1 Report Organisation

The objective of the thesis is to represent the discussion on key topics pursued throughout the MPhil period.

The first chapter will discuss the organisation of the thesis. The second chapter will start by providing a detailed analysis of firmware to justify the requirement of a taxonomy. The third chapter will begin by analysing and evaluating security mechanisms provided in video games.

The rest of the chapter will be based on this study, where we pick a smaller domain from the taxonomy to discuss the security of video games in much more detail.

We will start by discussing video games, starting from history and a background of the game industry. We will proceed towards the discussion of attacks and piracy of intellectual property (IP).

Following this, chapter 4 will discuss Digital Rights Management (DRM) in video games. The chapter will give an overview of the past techniques and current mechanisms for piracy protection. Next, chapter 5 will discuss how cheating is done in video games. The impact and the magnitude in real world will inform the reader about the importance of this study. Following this chapter, we will discuss potential work that could be undertaken in the near future.

1.2 Motivation

The motivation of this study arose from the international gaming competitions that are hosted worldwide. Many companies host tournaments that are played by the best gamers in the world. Most of these world events involve a large amount of cash money, usually in millions of USD. A few notable companies hosting world events includes Electronic Arts (EA), Valve, World Cyber Games (WCG) and many more. Professional gamers participate in these events and win cash money; it is also a very prestigious honour for the country they represent. Most of the winners end up being a gaming celebrity.

Gaining any small unfair advantage can be extremely valuable and can give the competitors an advantage. One of the ways the games can be exploited is via taking advantage of the weaknesses or limitations of the platform. On the client side, where the game is being launched or played, there are multiple ways a gamer can exploit these advantages. These methods are discussed in further detail later on.

Out of all the popular gaming tournaments and games, this study will mostly focus on Valves' world tournament, where the game 'Dota 2' is played.

1.3 Statement of objectives

The thesis will focus on short-listing the big domain of video games, where we will pick a smaller domain to study the security vulnerability of the video game clients. These will entail:

- An introduction to firmware and software.
- A discussion of the work done by others in this domain.
- A proposed taxonomy on firmware.
- A detailed analysis and justification of the proposed taxonomy.
- An introduction of the current video game market.

- A discussion on the history and evolution of different types of games and attacks.
- An introduction to DRM techniques and how they apply to protecting video games.
- A discussion on popular gaming clients and analysis of their limitations and weaknesses.
- An analysis of cheating in video games.
- A discussion on the work carried out on current cheating techniques.
- A discussion on future approach for the study.

2 Firmware and Software

This chapter will begin with discussion on firmware, providing an overview of the current domain. The discussion leads to a background analysis of firmware and an in-depth analysis on the taxonomy of firmware.

We will start with a brief introduction and continue by discussing the specific aspects of this term. This chapter will also discuss in detail specific branches and sections of the terminology and provide a taxonomy. We will also discuss the classifications using examples and justification for placing the items into certain categories.

2.1 Introduction

Firmware is a very broad term when it comes to everyday usage of the terminology. In the scientific community, the term is used very broadly and most of the time the usage depends on its context. In computing, firmware could be computer software that provides low-level computer instructions to control the hardware. It can also mean the operating system of a specific device. For the Internet of Things (IoT), firmware could mean a complete operating system for the full device that performs all kinds of control and functionality. Firmware in general could also mean a simple piece of software. The everyday usage of this scientific terminology is vague. In this chapter we discuss the classification of this terminology in a qualitative matter. The results of this study have been accepted for publication in ICEBE 2016 [1].

2.2 Background and requirement

Embedded devices exist at large in our ecosystem of computers and IoT devices. Devices are getting smarter every day. The core underlying logic and the functionality are being updated from day to day. Embedded systems comprise the majority of our global network substrate. Smart appliances and IoT devices are very common in our everyday household items. Using stock components, these devices are equipped and programmed to perform

certain actions. They are all equipped with certain capabilities that allow them to interact with other general-purpose computers.

The work undertaken in this study draws a line between the diverse classifications of firmware. We provide a modularised, better understanding of their properties, functionalities and security. We have examined the use of the terminology over a period of years and found that firmware is used in numerous situations in a range of contexts. In this chapter, we propose a taxonomy of firmware and provide a detailed analysis to classify the field to understand it better.

2.3 Related work

In this section, we discuss previous work on firmware and trends over the years. The section is shared in our publication but has also been included in this section for reference and connection to the taxonomy of firmware to exhibit a detailed picture of the classifications.

We start by discussing the changes over the years. More discussion on trends follows this section.

Year: 1967 - 1970 The usage of the word 'firmware' began in 1967. It was first mentioned by Opler [2]. Work conducted by Opler discusses the software portion that is saved in a ROM, providing specific guidelines on how the software should behave. From 1967 to 1971, there were no significant work done on firmware. In 1971, Barsamian and Dec mentioned firmware as programmed instructions stored in special ROM or read-write (RW) control stores [3].

Year: 1971 - 1993 There was no major update from the year 1971 till 1993. In 1993,

Mange et al. used the term firmware to visualise the concept of transforming software logic into hardware instruction sets [4]. The idea was used to provide a bridge between the hardware and software and vice versa. The software exhibited the idea of driving the hardware to do a specific task. The author did not refer to the software itself as the firmware;

however, he indicated the transformation process to be the firmware, since it was bridging the gap for establishing a communication platform.

Year: 1993 - 2012 According to work done by Chen [5], the software itself has been called firmware. From 1993 to 2012, much work has been done [6, 7, 8, 9, 10, 11, 12, 13] on firmware; however, the idea was addressed according to the context in discussion. It ranged from an "in-between" layer to "software", and nobody highlighted its position.

Early storage of firmware was done in a ROM or permanent storage media. The practice of writing firmware in the ROM continues today. In 1971, firmware was quoted as "micro-instructions that could be saved in the ROM mostly to be used for read-only purposes" [3]. However, the convention did not remain stagnant and limited to ROM. The 1990s and beyond introduced the rise of integrated circuitry (IC). Instruction sets in IC were hard-coded and installed in devices. Flash storage was introduced simultaneously. This introduction provided more speed and convenience for the developers and manufacturers. Re-writing the flash storage became easier. Unlike the ICs', which had to be manually removed to update the programs, flash storage did not require any removal and it was reused to overwrite existing firmware.

Upon introduction of flash storage, the year 1993 referred to firmware as "a way to convert hardware instructions into software instructions" [4]. As discussed in the previous section, the idea of a bridge was being highlighted. It has also been referred as a type of a software loosely referring as "a form of software" known as firmware [14].

The idea was not limited to software alone, but also focused on the area of hardware. Work done by Schubert [15] described firmware as a 'chip-set'. The connection or the 'layer' that resides between software and hardware is typically referred to as 'firmware', as identified in 2009 as the interface between the two [6]. Beyond the 1990s, flash storage was widely used, but the usage of ROM was still in existence. In 2010, the firmware stored in ROM was referred to as a defined 'functionality' that accomplishes a set of functions or jobs [9]. Following the work, further research in 2012 on firmware redefined it as being software stored in certain memories [10].

Firmware has evolved to represent any programmable content of a hardware device; it does not only represent software or binary code. It also exemplifies machine code for a processor, but also configurations and data for application-specific integrated circuits (ASICs), programmable logic devices and others. On the domain of processors, the firmware is being referred to as being "complicated logic which is stored as micro-instructions" [3]. In an embedded system, Cesário described firmware as programs that are stored in hardware memories such as ROMs. The work deems the firmware to be just 'hardware" [14].

In work carried out in aviation and dealing with software engineering challenges, it was quoted "It is sometimes not clear what constitutes a processor, for example, because so much specialised electronics is involved. Similarly, software is sometimes in read-only memories called firmware rather than software" [7].

It can be clearly seen the understanding has varied over the years, where firmware was referred to as dedicated software and/or hardware-specific instruction set. The ways the term has been defined was in context to the application or device under discussion.

2.4 Taxonomy and its justification

While most of the material in this section is also present in the publication, the below section also includes extra examples and justifications that we could not include in the publication due to word and page limitations.

We need to establish a distinct understanding of what a video games can consist of. Thus, it is vital for our video game security study to find out how many categories of video game there are. Within those categories, we will simplify our scope. This will allow us to conduct a qualitative analysis and present the flaws of that particular arena.

In this section, we will start by expanding on the landscape of firmware. Then, we will follow it by proposing a taxonomy structure of firmware. We will end with a subsequent section briefly discussing the rationale behind the categorisation. The diagram of our taxonomy is illustrated in Figure 1.

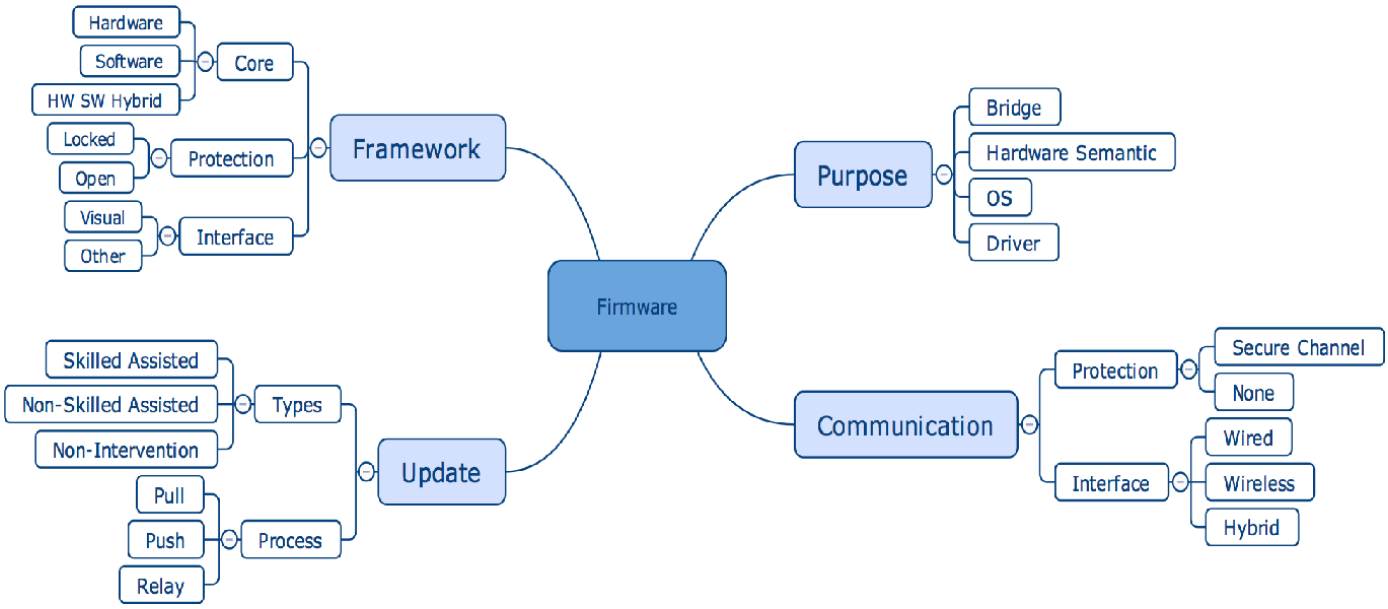

Figure 1: Taxonomy of firmware

2.5 Framework

The first family is named 'Framework' and highlights the basic structure and characteristics of firmware. This family tree has been further dissected into three branches, namely: 'Core', 'Protection' and 'Interface'.

2.5.1 Core

We bisect the firmware into multiple parts. The rationale behind naming this section 'Core' is to represent the major functionalities of the firmware. This branch consists of three more sub-children: 'Hardware', 'Software' and 'HW SW Hybrid'.

Hardware We categorise 'hardware' where information to control a device runs on a printed circuit board or a signal converter. An example of this classification would be the microchip inside a watch, table lamp, or toy cars [16]. The circuitry uses electricity to convert electrical energy into different forms of energy. In the case of a watch, it is converted to mechanical and kinetic energy which helps with the movement of dials. In case of a lamp, the electrical energy is converted to light and heat energy. The range of attacks

within this domain are limited, since they involve manipulating the hardware components and are usually referred to as hacks rather than attacks [17].

Software We classify 'software' as the kind of firmware where the programs are stored programmatically inside a storage media e.g. ROM, EPROM, EEPROM etc. These are programs that have their existence in a digital format rather than a series of electrical signals. Examples would include drivers for a printer [18], visual firmware for customising a mouse [19], etc. In work done by Maskiewicz et al. [20], a Logitech G600 mouse was programmed by writing custom software to achieve a file transfer task.

This is a typical branch of the taxonomy where our use-case of video game domain could be targeted. This 'software' group could consist of video game clients where the client can be vulnerable to attacks. Ideally, retro arcade video games should not fall under this category and our target group of PlayStation games would fall here. The category where retro arcade games would fall is discussed in the next paragraph of 'HW SW Hybrid'.

Attacks in this domain usually involves manipulating the code to achieve extra benefits. It has been shown in work by Yanlin et al. [21] that it is possible to reverse engineer the firmware of a network card and manipulate it. Using a package known as interactive disassembler (IDA), it is possible to alter the function calls within a software program.

HW SW Hybrid The device family which involves both characteristics of hardware and software are enlisted in this category. If we consider BIOS, the chip set itself has a physical existence, whereas it is not categorised as a hardware because there is a software program within the chip. On the contrary, labelling it as mere software is not applicable, since it is a chip which has physical existence and does not require power to store the information. Another example for these criteria includes the set-top TV boxes. They convert direct button press signals into machine readable signals [8]. Furthermore, they have their storage installed within their device that stores the firmware content, and updating that content can be done by connecting a cable and flashing it [22].

Retro arcade games fall into this category. The video game part of the console is stored

in a chipset and doing an online update is not possible. One of the reasons would be that the information stored in the chipset is not writable. However, the most important part is that most retro arcade games are air-gapped and not connected to any network. As such, as discussed previously, updating these would require a whole replacement of the units. More on firmware update is discussed in Section 2.8.

There are attacks targeted to BIOS. Work done by Wojtczuk and Alexander [23] showed a BIOS attack by manipulating several checksums. They used a BIOS building tool followed by patching to compensate the checksum. Other work has shown how data can be hidden in BIOS chips. BIOS has a capacity of storing 128k to 512k of data, and this storage can be used by criminals [24]. Criminals could use this feature to manipulate a target computer to cause damage or create data-loss in an organisation or a specific person. Fortunately, the modern implementations of BIOS [25] have changed and stronger security primitives within BIOS have been introduced with secure boot and unified extensible firmware interface (UEFI) [26].

2.5.2 Protection

This sub-tree discusses the security mechanism that is available in any kind of firmware. It has been further elaborated into two more sub-children, namely:

Locked When describing firmware, how the firmware itself is utilised or stored in a medium is a very important aspect. Firmware can be stored in a ROM [27], but not always, since it can also be stored in a hard disk drive, EPROM flash drive or solid-state drive (SSD). The protection of the storage is classified as an attribute. Some firmware has a protective mechanism to restrain unauthorised modification or alteration, whereas other firmware does not have any protection. By 'Locked' we refer to the family of firmware which is digitally signed by the manufacturer [28] and has their default administrator login credentials encrypted.

Attacks involved in this domain include the successful 'jailbreaking' [29] of multiple iOS versions. iOS jailbreaking removes the restrictions of the Apple's iOS operating sys-

tem using software exploits. This allows the locked restriction of downloading additional applications, music and themes that are not available through the official AppStore [30].

Open Situations where the username and password to the administrator access portal of the firmware is not cryptographically protected or written in plain text falls into our branch of 'Open'. It has been discovered that manufacturers [11] do not take necessary precaution to encrypt the default administrator login credentials within the firmware. Rather, they use specific company-title [11] and other fixed-place decimal value data linked with the firmware that can be easily brute-forced to retrieve the credentials. These types of devices are termed as 'Open' in our taxonomy. For example, in a TP-Link TD-W8951ND V4 ADSL router, it was observed that the username and password were stored in plain text when a simple man-in-the-middle attack was performed using Cain [31]. Furthermore, it has also been reported in specific software packages - e.g. Deploy Studio [32], which is used to image and deploy Mac computers to manage workstations and servers - that the firmware passwords are stored in the log in plain texts. In addition to these, the telecommunication giant EE's BrightBox routers login page stores their customers credentials in a plain text format [33].

2.5.3 Interface

The interface of a piece of firmware is placed as a part of the firmware's framework capabilities. It represents two different types of firmware that consist of an interface to allow configuration; there is another set of firmware that comes with predefined characteristic and functionality that does not allow any customisation and thus only behaves according to the manufacturers' pre-set instruction sets.

Visual Ideally, the interface makes way for configuring the firmware. It also allows the control of the device. The BIOS of a computer machine has been considered as an example. The BIOS is a type of firmware that gives the user an option to customise the settings of a computer [34]. The interface used is a graphical user interface (GUI) which allows

navigation via a keyboard that would allow the user to tweak settings, e.g. a primary or secondary boot device.

Another example would include when a user of a computer would want to enable a firmware password to restrict the machine from booting from another hard drive. In an Apple Macintosh, where there is a specific GUI that enables the usage of a firmware password [28]. The presence of an interface enables the option to set up a password. The interface allows the possibility to set up a password, and without it the user will not be able to set it up. Only the manufacturer would be able to do so.

The visual category has been established to also represent messaging output. In a telephone handset, the display that shows the digits input into the set can also be referred as the visual interface. More examples would include tuning the thermostat in a home, where the current temperature and other settings are displayed as a message or information to the user.

None The family of firmware that does not exhibit a configurable interface is included in this section. The example clarifies the members of these sub-children. The hardware content within a central processing unit (CPU) consists of processors [35], RAM [36] and hard disk drive [37] to name a few. All of these devices have firmware within them that provides them with instruction on how to operate specific mechanical parts of that device. For example, the firmware within the hard disk [38] drive controls the disk rotation and read-write speed. It also contains information regarding how the allocation tables are managed. Configuring the firmware is not possible, since the data is already set up by the manufacturer and there is no interactive interface to tweak and configure the settings.

There have been attacks on some of these components, even though they lack a visual interface.

Kaspersky researchers discovered that the hard drive firmware could be compromised [37]. They showed that subverting the firmware allows the attacker to create invisible storage space to hide data from the system. The data is not erased even when the hard drive is formatted. This allows the attacker to retrieve the data at a later date.

The above section of our taxonomy discussed the basic structure and characteristics of firmware. Within the limited scope of this domain, we tried to explain the different array of firmware that exists in the industry and highlight their respective positions.

2.6 Purpose

This category has been formed to fulfil our goal of categorising different firmware according to their functional capabilities.

2.6.1 Bridge

It is known as an intermediary that allows the connection from the product itself to its operator. An example is given to depict the usability of bridge.

One example is the firmware of a battery: when the operating system of the computer or a phone asks for the percentage left or the life cycle or capacity of the battery, the firmware within the battery replies with that information [39]. In this situation, the firmware is acting as a bridge providing a connection medium to establish workflow between the software and the hardware counterpart. Another example would include the mechanism in a touch screen device in a supermarket till. The firmware within the touch screen converts the mechanical touch inputs into electrical output to be read by the system [40]. The software or 'firmware' here aids as a bridge connecting the input to convert into outputs.

2.6.2 Hardware semantic

A module of hardware that has micro-instructions [41] embedded into the chipset falls into this category. It is a block originally created by the manufacturer and has the logic within it to make the hardware work accordingly. An example would include an electric razor by Philips SensoTouch. The power button is synchronised with the motor in a chipset. When the power button is pressed, the logical instructions within the semiconductor chip instruct the motor head to start rotating, hence achieving the task and making the device useful [42]. A similar example would be the usage of a kitchen toaster. It has the same

architecture, where the button is programmed to push the bread and initiate the heating of the filament. This branch overlaps with our 'HW SW Hybrid branch' within 'framework'. However, we have still placed 'Hardware Semantic' here, since it is a specific quality of the firmware and cannot be ignored when it comes to categorise a purpose of the firmware.

2.6.3 Operating System

The operating system, e.g. iOS, Android, Windows Embedded acts as firmware to operate the embedded device or a phone or kiosk. The firmware allows the device hardware to be used to produce a convenient output. i.e. processing a touch or processing a swipe of the card in a kiosk. Operating system is closely related to a bridge. The difference comes in its behaviour: the operating system provides a user interface to interact with the phone, making the buttons on the phone do specific task, making the touch inputs process relevant output, or allowing the ability to respond to force-touch [43]. When an iPhone software fails to launch, the device needs to be reset to factory settings. The process is known as recovery mode [44]. However, the technical terminology is referred to as device firmware upgrade (DFU) [45]. To update the operating system of iPhone, an iPod software firmware file is required [46]. This can be obtained manually and injected into iTunes, or can be automatically downloaded by iTunes itself. Once the software binary is acquired, iTunes can initiate installation of the latest firmware onto the device.

2.6.4 Driver

When there is hardware, there needs to be specific type of instruction set provided by the manufacturer to make the device work desirably in another environment. For example, for the printer HP LaserJet 4200 to be identified properly by a computer or a network, there is a need for a driver program that would help accomplish the effective communication [47]. In addition, when an Apple computer is used to run a different operating system, the hardware should be given proper instruction to behave according to the new environment. An example of this is the Bootcamp software from Apple, which makes the Macintosh hardware

compatible with the Microsoft Windows operating system, allowing all the peripherals to work perfectly [48].

2.7 Communication

The firmware requires transfer of data within other devices and with the outside world. We have referred the process of data transfer as 'Communication'. This branch consists of two more sub-children that entail the functionality of our communication.

2.7.1 Protection

We refer to the ways the communication is monitored or allowed. Some communications are followed with cryptographic security mechanisms, whilst others are not. Thus, the sub-children of this branch represent:

Secure Channel The firmware in a Logitech G600 mouse uses Rivest, Shamir, and Adelman (RSA) encryption [20] to communicate with the device. Gathering data from the mouse and transferring the data to the operating system of the computer is done via a secure channel communication. The data communicated are not in plain text. This allows a secured medium for communication that prohibits interception of data and adds more security.

The importance of security varies from use cases. Different firmware has different levels of security. When the application is a sensitive issue, for example in any medical equipment that risks the well-being of the patient, security is a very important criterion. One example is the pacemaker within a patient's heart; the firmware of any planted device should have security since it is vulnerable [12] to malicious attacks [49] which could compromise a human life.

There has been attacks reported that falls into our categorisation within a secured channel. As Maskiewicz et al. [20] reported in their work, the firmware of the mouse can be compromised to write custom malicious code executing other tasks. They have written their own code to copy a file while the mouse was in work. The study shows that the

firmware in peripheral devices is not secure. In other work done by [50], they discussed the manipulation of a network card which was using flawed firmware that an attacker may subvert remotely by sending packets on the network to the adapter.

Other However, there are cases such as the Nikon camera D3100 or other entertainment devices, e.g. music players, where the firmware could be easily manipulated to enable hidden features are termed as 'none' or 'other' in the protection category.

The Canon IXUS30 (SD200) camera's firmware can be easily modified [51] to attain pictures of higher resolution, taking RAW images, shooting self-capture images, time-lapse, etc. The firmware modification allows all these, and the user may not have to buy a later model to get those features. The tweak has more usage. In an experiment to capture images of the Earth, the camera was modified to capture pictures throughout the whole duration of the experiment [52]. In addition to this, previous work [49, 53, 54] also discusses lacking proper user authentication.

2.7.2 Interface

The branch has been dissected further to categorise different mediums of communication that take place within a firmware. Typically, three interfaces are proposed.

Wired Firmware updates in the automotive industry are carried out using a wired interface [55]. It can also be used to connect to the diagnostics port to retrieve information about the vehicle. The firmware diagnosis and update process is done off-board, by connecting (hardwired) a diagnosis tool with the on-board network and performing firmware updates[56].

Wireless Other ways of communication include performing the whole data transfer over a wireless connection e.g. WiFi, 4G, Bluetooth, infrared, etc. For example, in the automotive industry the update to the firmware can be patched via over-the-air (OTA) [57, 58]. The wireless mechanism is not only limited to automotive domain, in a mobile phone OTA,

updates of the firmware of a mobile telephone handset can also be done [59].

Hybrid However, some devices are not only limited to these two types of communication. It is not possible to categorise them into the aforementioned wired and wireless category. So we open up a new category named 'Hybrid'. A smart-card is an example of this category. The smart-card can operate two ways, as a chip-and-pin mechanism and also in a contactless mechanism [60]. The example of smart card cannot necessarily be categorised as having a 'wired' interface nor it can be categorised as having a 'wireless' interface. It has the ability of performing both interfaces, and thus we have next sub-child: 'hybrid'.

2.8 Update

This branch consists of two more sub-children that represent the ways the update of the firmware is conducted.

As discussed earlier, when firmware used to reside on a ROM, it was difficult to apply updates. However, with recent improvements in technology, as of 2013, most firmware can be updated [61], but still there are risks. Upon a failed update, the whole device runs the risk of being termed as 'bricked' where the update procedure is deemed to have destroyed the device [62].

Applying firmware updates on critical embedded systems can be cumbersome and daunting [13]: however, there are different mechanisms that accommodate varying update mechanisms and techniques. As depicted in Figure 1, the 'Update' branch is dissected into two further sub-children: 'Types' discusses the security mechanism of the updates and 'Process' the ways the updates take place.

2.8.1 Types

Types refer to the possible ways the update of a firmware could be performed. Types has been broken down into three simplistic branches that discuss the possible ways a user can have their firmware updated to get a newer version or revert back to a different version.

Skilled This paragraph refers to the installation of firmware that a normal user would not be able to perform by themselves, e.g. to perform an update of the BMW 6 series automobile's dashboard navigation system, the user needs to bring the car to BMW dealership to apply an update. Here, the technician will use one of the communication mechanisms as discussed earlier to perform the update.

In addition to this example, retro arcade video games fall into this category, where the average gamer would not be able to update the video game themselves. A skilled person would be required to replace the whole game module from the arcade and have the new video game installed.

Non-Skilled Any sort of update that does not require special assistance or an unavailable toolset is referred to as a non-skilled update. Regarding our video game example of PlayStation 5, if a specific video game has an update available, all the user would need to do is click an update button using their game controller and the update would start. This does not require any specific skillset.

Furthermore, if we consider the situation where the phone's firmware needs to be updated, it can be easily done via the operating system's interface. Phone manufacturers have readily made this resource available to everyone for them to be able to perform it without any skilled assistance. For example, any 'Android' user could update their phone software by themselves clicking appropriate buttons from the user interface.

Non-intervention This type of update refers to those planned updates that are done automatically on a specific date, time or a period. As an example, when a new software update is available in a Windows machine, if the user has agreed to apply the updates, they will be downloaded and installed automatically onto the system [63]. Devices today are increasingly equipped with WiFi components and the availability of WiFi hotspots has improved. This means that devices can take advantage of free and fast connections to download web content, e.g. pre-fetch web pages for offline use, update RSS feeds, or download podcasts, new e-mails or updates for web widgets [63].

For example, when a user has set up an 'auto-update' in their machine, the update will take place without any further input. It should carry on at a prescribed time. The situation is also true when a user turns on the auto-update feature, e.g. Microsoft Windows Update.

2.8.2 Process

When an update is due, the manufacturer may choose to distribute the latest update to all the devices under its ecosystem. The update option could be prompted, or if previously recorded, it can initiate on its own.

Consecutively, in a situation where the update has failed, the machine may automatically request the update to be performed again [64].

Pull To categorise the situation when updates are sent or received, we classified a 'Pull' sub-child. The purpose of this sub-child is to include the kind of updates that take place upon a request. When a new version of the firmware is being released by the manufacturer, the consumer might not be notified straight away. This could be due to design issues of the manufacturer or server traffic once a new version is release. To obtain these types of update materials, a user would have to query the network asking for new update. If an update is available, the firmware would then be downloaded or installed. For example, when a firmware update of nVidia GTX 650M graphics card driver is released, the notification does not pop up reminding the user to update it. However, when the user voluntarily prompts 'Check for Updates', the new firmware version will appear [65]. Obtaining this kind of update is referred as the 'Pull' mechanism in our tree.

Push In situations when new information or data needs to be sent to the subscribers or customers, one of the ways used by companies or manufacturers is to dispatch or release the data. The data from the central network are intended to reach the product as a part of an update [66]. For example, in a Programmable Logic Controller in an industry, when it is needed to provide changes to the current instruction set [67], the whole firmware can be patched in via a push update.

Relay In a very specific case, a resource does not necessarily have to be obtained via a push or pull mechanism. A resource can sometimes be available via a third party. We have termed this type of procedure as 'Relay'. The source is obtained from multiple media, e.g. a ROM, DVD or flash storage, and then applied to the target media. For example, when a user attempts to update the firmware of an Apple Watch Sport, the user needs to obtain the firmware update via the iPhone rather than the Apple Watch itself [68]. The Apple Watch does not have the mechanism or capability to download the new firmware, and relies on another device to apply the update. To make this update possible, the iPhone has to transfer or 'relay' the data via Bluetooth to complete the update [69].

2.9 Summary

In this chapter, we discussed the details of a proposed taxonomy on firmware. The differences between software and firmware have been discussed in detail and the need for a taxonomy was mentioned.

A detailed analysis on the background of firmware and the work undertaken by other researchers was provided. Patterns of usage over the years were discussed. Conclusion of the discussion lead to the development of a taxonomy for firmware.

Furthermore, the proposed taxonomy of firmware was accepted in a publication, and the rest of the text discussed the branches of the taxonomy in detail together with examples.

The work done here will allow us to have a deeper understanding of how the different variants of video games can be categorised. This will enable us to pick a smaller domain from the aforementioned branches and continue our work, which will be discussed in the next few chapters.

3 Video Games and Security

This chapter will discuss security in video games. This section will start with a brief history of the evolution of gaming devices. Throughout this section, work done by other researchers will also be discussed. The first section will conclude with discussion on the evolution of attacks and piracy. The second section will continue the discussion on piracy. This section will start with an introduction on DRM and continue to discuss industry standard solutions. This chapter discusses and analyses the limitations of the few popular anti-piracy techniques currently employed by the gaming companies e.g. SecuROM, SafeDisk, Denuvo and Steam.

The final chapter will provide an analysis on cheating and discuss the work done in this domain by other scientists.

3.1 Introduction

For the younger generations in society, computer games may turn out to be a source of entertainment [70]. Besides the fun factor, over the decades, computer application seems to have a bigger impact. The number of people that understand that computer games have been a driving force when it comes to the development of the IT infrastructure is much fewer. In early 1970s, it was a major funding source for the silicon industry because the gaming industry provided good business. The demand for computer games meant that the manufacturers could produce hardware at a cheaper price. This included the manufacturing of monitors and any other hardware that was required for gaming.

The statistics at present have not changed much [71]: the gaming industry still leads when it comes to manufacturing computer hardware, e.g. graphics cards, game pads, monitors, etc. The gaming industry is still a billion-dollar market. The gaming market generated $9.4 billion in sales for the first time in 2001, competing with the US box-office market, which made almost $9 billion [72].

With emphasis on good-quality graphics and better and interesting gameplay, the com-

puter gaming industry has had a major impact over decades. It has risen so much, the online gaming tends to be one of the most popular applications on the internet.

3.2 Brief history

The idea of gaming to begin with was a part of automation software. It can be traced back to 1884, when Charles Babbage discussed how a chess game can be played automatically [73]. The current gaming phenomena that exist at present with the use of modern computers initially began in the 1940s. In 1940s, John Von Neumann and Oskar Morgenstern were experimenting with the idea of general theory of games. They went on to devise their min-max algorithm and applied this theoretical idea to a game of chess [74].

Computer intelligence was mostly the earliest research means for modern computer games. In the research field, the game of chess was an excellent choice for study. There are a great many scientists who excelled in the study of computer chess, and used this game as a means for their research. To name a few, in 1950 Claude Shannon made a publication [75] on a computer chess game. In the work done by Shannon, he mostly discussed strategies for the game of chess. He also discussed anticipation of chess moves that could be made to determine the future positions of a chess piece. Even though he never developed a computer chess-playing game, to date, most of the chess games that have been developed follow Shannon's ideas. In 1953, Alan Turing published a work automating chess strategy.

3.2.1 Early games: Mainframe computers

The first computer game to have an impact was a military simulation. This was written by Bob Chapman at the Rand Air defence lab in 1952 [76]. Apart from the military base games, in 1954 a student from the University of Michigan developed a game of pool. At MIT in 1962, Space War was developed by Russell et al. [77]. They developed it on the PDP-1. During that time, the PDP-1 had an interactive cathode ray tube (CRT) monitor which engaged the human on a physical level. The game was very interesting because it had human-machine interaction, the sort we have at present. Development of these games

was very expensive in terms of resources, because most of these games were developed in mainframe computers, and apparently these resources were available to researchers and computer scientists only. The general population had no idea about these developments, and it was a luxury to be playing games in a mainframe computer.

3.2.2 Arcade games

Nolan Bushnell in 1971 [78] developed a machine that accepted coins and only executed certain programs. He was using a computer which would only launch a game. His first development was a game called 'Computer Space'. In terms of computer game and coin operation, this was the first to have a video output. It was not a text-based terminal type text game. To follow up with this game, Bushnell went on to make another successful game known as Pong.

In 1996, Bushnell [79] went on making the publication about his Computer Space game. He mentioned the Computer Space was not supposed to be a computer game. Rather, it was supposed to be an electronic game based on a hardware state machine. Using logic gates and counters, this was merely a state machine, but it gathered popularity unintentionally. Arcade games were the norm until 1971, when microprocessors were not yet abundant. Although it is quite difficult to say when the first computer arcade game was developed, Nintendo started selling a coined version of the game from 1978.

3.2.3 Console games

Historically console games were graphics simulations which were mostly created by computer researchers to assist their research. The commercial aspect rose when Nolan Bushnel [79] started the idea of gaming. Since it was very expensive to play at home, the commercial aspect of the console games was not appealing. The development of console games allowed the games to be played at home, which made it hihgly appealing, as it proved to be an alternative entertainment device in addition to the TV. The console games were connected to TV and only required usage of a small piece of hardware. 'Odyssey' was the first of its

kind. This was marketed by Magnavox and invented by Ralph Baer in 1971 [80]. These consoles did not have an interchangeable cartridge and relied on factory-installed games. It was not until 1976 that replaceable cartridges were introduced, which allowed the users to play different games within one console. Nintendo [81] made its mark with their eight-bit NES console in 1980. The famous PlayStation console was released by Sony in 1995. In 2001, to stay in the competition, Microsoft entered the market with their production of the Xbox. To date, Microsoft and Sony has invested a huge sum of money and they are competing the market to gain customers. These two gaming giants have had continuous success with their products, so much that they have already acquired a huge loyal customer base. The latest, as of 2021, is that console games for Microsoft are known as Xbox Series X and for Sony as PlayStation 5. These consoles are equipped with quad core processors which are capable of producing very smooth and realistic graphics. Console gaming is a very popular market for the gaming industry and has inspired more and more developers to keep on creating games and market them commercially.

3.2.4 PC Games

In 1976, Apple I and TRS-80 were the host for the first personal computer games. They were initially developed by hackers mostly for entertainment purposes . Steve Wozniak developed a game called 'Breakout' [79]: this was specifically designed for Atari and it played a significant role when Wozniak went on to create Apple II.

The Apple II machine dominated the first computer game market. Their competitors became the computer giant IBM. Both Apple and IBM had good success when it came to competing with Linux. Linux was already a popular operating system. However, when it came to personal gaming, people preferred PC gaming over Linux. The PC games were mostly known as MS-DOS and MS Windows Games.

3.2.5 Online games

Early computer games were developed to be played by one person. Even though it is still a popular genre in gaming, it can be boring for some gamers, as they keep on playing with virtual opponents. The virtual opponents are mostly artificial intelligence (AI) developed by the developer [82]. Most of the time, the moves or the gameplay made by the AI can be predictable and may make the game uninteresting. Multiplayer games were not popular when gaming was constricted to mainframe games or console games. It was with the rise of internet connectivity that people become interested in better communication and data transfer. This influenced the game developers to create better versions of multiplayer games that use internet connectivity or network connection [83]. With the online games, games are no longer restricted to one-player games, and the number of players that can join the single game could vary from two to even 200 players. The massive multiplayer game 'Eve' can host 200 human players in one single session of the game. There are even different genres for the multiplayer games, e.g. multiplayer online battle arena (MOBA), role-playing game (RPG), first person shooter (FPS), sporting simulation, etc. These games are played over the internet and the hardware could be anything ranging from a mobile device to a personal computer powered by a microprocessor.

In 1979 at Essex University in England, Roy Trubshaw and Richard Bartle developed the first online computer game, known as the MUD1 [84]. It was very popular and supported multiplayer mode, which inspired them to make sequels of this version.

Gaming has evolved so much from 1979, that it has become a major sporting event. The gaming giant also known as 'Valve' [85] has hosted a gaming competition every year since 2000. In 2018, the total prize money for one computer game competition was $25 million. These days, the gaming industry is becoming a multibillion-dollar business. One of the main reasons is that people like interacting with real human players compared to an automated AI bot. They are even willing to pay a subscription fee or a one-off fee to purchase an online service. EverQuest, developed by Sony, gives the company a revenue of $5 million dollars every month only from their subscription fee. This single game gives

a 40% profit margin for the company. Other online games known as Ultima online, World of Warcraft, Diablo and Eve, have had similar success. The subscription fee for World of Warcraft [86] started from as little as $2 a month has have increased to $20 a month with its increasing popularity.

At present, there is also a different variety of game developed by certain companies that offer the gamers to play for free [85]. There is no fee to play the game. However if the user chooses, they can spend money to buy cosmetics that increase their gaming visuals to a different level. Even though there is no advantage the gamer gains by purchasing these cosmetics or add-on items, it has still proved to be very popular; it gives gamers bragging rights and lets them show off quality amidst their peers. In the free-to-play game known as Dota 2 [87], a single cosmetic item could cost from $1 to $1000. Surprisingly, people are buying them, which makes the free-to-play game a continuous success for the developers and an attraction for a bigger crowd due to its no-fee policy.

3.3 Evolution of attacks and piracy

The first PC games entered the commercial market in floppy disks, cartridges, cassettes etc. From 1980s, the commercial industry has been attacked by piracy. Hacking copyrighted materials and pirating expensive movies and games have become hobbies or real jobs for certain kinds of people. Society and the industry have had little success counteracting these hackers. Hacking the game source code, better known as modelling the game environment, has become a playground for the gamers. Where a studio or the developers are busy or constrained with resources to support the existing customers with support, the hacking community has taken the edge to extend the current version of the games to a better playable platform. At times, when a game patch or bug fix has become essential and the limited resources of the company could take them a couple of months to provide a fix, the hacking community has earned their popularity by providing that fix for the consumers in a couple of hours' time. These fixes are also provided for free and seem to have gained consumers' trust. Any illicit modifications the hackers make whilst patching or modifying

the game is very difficult to identify, and can provide a huge security threat. They could knowingly install keyloggers or trojans that could steal valuable information and send it to the hackers. The user community has very little awareness about this and tends to care less as long as they are receiving the patches or fixes on time and for free.

3.3.1 Motives

Apart from the incentive for the hackers to pirate a game, they also have other incentives. Sometimes, hackers also take it as bragging rights to pirate a new release so they become very renowned within the community. This could get gain them extra money or fame that are useful when striving for new job offers in the black market.

3.3.2 Illegal distribution of games

One of the biggest reasons for pirated distribution of games is to obtain a copy of an expensive game for free. Over the years, many different protection mechanics have been employed by the industry to tackle this. These have ranged from floppy disks or CD/DVDs having special sectors to prevent illegal copying, but with the recent technological advances, it has become difficult to prevent these attacks. In the early 1980s, a brand-new game bought in DVD would require the insertion of the physical disc into the disc drive, which will check for specific sector to be present or not to verify the genuineness of the purchase. However, these protection mechanisms are not at all applicable with the development of specific software known as Daemon tools. Daemon tools create a virtual copy of the disc and let the operating system think the user has actually bought a genuine copy. The software bypasses all the system calls made to the CD or DVD and imitates these calls at a very low level to bypass this protection.

3.3.3 Illicit usage and modification scenarios

Hacking or piracy of the game is also known as cracking. When a game is known to be cracked, it tends to have fewer protective measures. A game could come with a pre-existing

license and require a new copy to be purchased when it wants to be played on a different machine. These protections are bypassed by editing the game code. To tackle this, code obfuscation is employed by many companies, but most small companies do not tend to use this, since it could require more production time or a bigger budget.

Call of Duty Modifications in games are also known as mods. In one of the popular franchise developed by EA, the game 'Call of Duty' can be modded to not only play it for free, but also to gain an unfair advantage. Hackers can use a status injector during an online version of the game which shows a player's false availability. These false representations make it easier for gamers to get better ranks more easily whilst tarnishing the image or statistics of the genuine player.

There are other modifications made 'for' the games without editing or modifying the client itself; for example computer scripts can be written for a game which would allow the users in a shooting game to shoot other users easily. This is also known as 'Auto-aim bot' What this bot essentially does is automatically locate a player and point the mouse or controller cursor on top of the enemy object. This makes it easier to account for human error or human reaction times.

Diablo III In the section on game-modding, the game 'Diablo 3' is reported to have the highest amount of modding. The gaming scenario involves a target to be destroyed by a group of players. Players begin the game at the preliminary level, where their initial damage is from 2 to 100, and progressively their damage could increase up to 500. As they keep progressing in the game, the attack damage tends to increase in a controlled manner. Hackers change the game code in the local client to have their damages increased to 1,000,000 instead of 100 ,which allows them to finish the game much earlier than other teams. These hacks or cheats are not detected by the leaderboard server. As a result, the modders ranked highly in the leaderboard and get nominated for special prizes or free entry to prestigious events. This illicit usage of scripts has hindered the gameplay and annoyed the legitimate users playing the game. This has also made the gaming competitions less fun

and led to poor quality matchups. An overall effect has ultimately deterred gamers from playing this game and made them avoid future purchases from this franchise.

Pokémon Another famous title of the gaming industry is known as 'Pokémon'. Special hacking software known as PKX editor allows the user to edit the game's inventory items. The objective of the Pokémon game is to trade cards having special abilities to build a personal fighter. Accumulation of abilities is hard to acquire and usually expensive. A modded Pokémon can make the game very unfair, as it will be very easy to win against a legitimate Pokémon character having basic abilities. This hack or mod is very annoying for gamers in the online portal where they compete with other real players for prize money and fame. The developers of the game are aware of this modification and have placed a Pokémon validation server, known as PokéBank ,where all trading must go. However, hackers have found a way to bypass this server and have their Pokémon with special abilities approved.

A famous scenario in the Pokémon gaming industry is when Ray Rizzo, the gamer who has been crowned Pokémon champion three times, was caught cheating using modified characters. The PokéBank failed to detect this change, but later the gamer was flagged by the user community who detected an anomaly in Rizzo's Pokémon's animation. Ray was using a Pokémon with a modified animation timer that allowed him to execute more attack events while reducing the time to execute a complete attack animation. This yielded him attacking three to four times faster compared to the time taken for a single attack. Rizzo's modification technique proved how easy it was to counteract the PokéBank and how difficult it can be to detect such small detail change, i.e. attack animation time.

4 Digital Rights Management (DRM)

4.1 Introduction

Computer games developers or development studios continuously go through the ill effects of unlawful theft and data spillage which are caused by customers or gamers. This happens not long after the official release or the publication of a computer game or piece of software. The fight to get rid of these illicit hackers or consumers destroying revenue and their IP is a constant struggle. Systems to avoid or in any event delay this illicit marketing or publishing are DRM or copy protection mechanisms. The lengths to which these techniques or protection are applied comes with a trade-off cost for the end users. In any case, these systems are exceptionally disliked, on the grounds that they confine the clients in playing the games and also request a high overhead in the organisation efforts of the gaming company or the studio.

The management of the striving companies does not end with these mediocre protection mechanisms. The techniques to overcome the digital copyright protections are continuously being broken, and the proof is the continuous delivery of 'patches' from the gaming company or the software company. These 'patches' are deemed to bring in a better experience and mend the existing security mechanisms that are in play.

4.2 Inspiration: Video game piracy

The market for computer games has been quickly developing and these days outflanks the deals for different sorts of mixed media, e.g. music charts, box-office sales for movies. Subsequently, video games fall into this genre and they can be copied effortlessly. This activity can be observed some time before the official market release of the product. During the phase of release, where a game studio releases their 'beta' or 'demo' version as a taster for the consumers, the piracy attacks are seen from there on. The computer game market is especially helpless against theft like this. The time which the hackers take to pirate the

full and final version is within a couple of weeks to months. The initial few weeks upon a product release is a significant time for the developers or the company to win the market. Furthermore, the leaked pre-releases can also have a major impact on the sales. This could hinder the future of a company or the people employed there, since the production of 'big titles' could cost up to several million dollars.

Very often, a so-called 'cracked' version of a video game wrongfully appears on the internet as free downloads not long after its official market release. These cracked versions are the ones which would carry a implemented bypass mechanism for the digital copyright technologies that may have been integrated by the developers. This release of 'cracked' version demonstrates the copyright techniques are a volatile medium of security. Crytek computer game 'Crysis 2' was an unmistakable example: it was renowned as the most pirated game of the year 2011. The number of illicit downloads for this version of the game recorded only from the software BitTorrent was around 4,000,000. The title: 'Crysis 2', is a very renowned franchise from the gaming studio EA, who usually have budgets of around several million dollars for their production of every single title. The loss of revenue is hindering the development of the gaming industry.

There is a strong plea for better copyright policies for the development of video games, which has forced a large number of studios or video game developers to strategise their development and marketing policy.

4.3 DRM and copyrighting

The usual methods for piracy prevention were confined to certain copyright protections, e.g. valid serial number or licence numbers. However, these days, with the aid of better data connectivity and availability of internet, the technologies have been extended to integrate online support. This includes online activation of a license key rather than a hard-coded license key activation on the back-cover of the game disc, which could easily be hacked or tackled by 'cracking' software like CheatEngine [88]. The online activation system was used on a wider scale with the release of Ubisoft's 'Assassin's Creed 2' (2009/2010) and

Activision Blizzard's 'Diablo 3' (2012).

With the constant evolving of better techniques, it has been seen these technologies can also be difficult for the developers. The stronger the DRM, the more resources are extinguished by the customer service to provide support for the end-users to resolve their complaints. In addition to these, the usage of online DRM servers also meant a security hazard due to the constant DoS (denial-of-service) attacks made on the online services. At times, these servers had to be 'taken off' for maintenance, and this creates chaos and confusion within the gaming community, bringing deterioration of brand values. Some of these attacks have been very successful, and millions of valuable and personal user information and privacy details have been tarnished. The most renowned attack on the online servers is the Sony PlayStation hack in 2011. Millions of credit card details, personal addresses and user's passwords and other sensitive information were hacked.

In spite of the vulnerable the distributors or the developers need to depend on DRM because of limited choices or alternative security approaches.

4.3.1 Browser-embedded games

The market of the browser based games is still developing and has not reached notable popularity; this is mostly due to the limited capabilities of a browser. However, security on these browsers can be established by internet protocols. Since hardly any competitive games worthy of a decent amount of prize money are played on this platform, the platform is therefore at a lower risk compared to others.

4.3.2 Free-to-play games

Free-to-play games are also known as F2P. This is a very good mechanism to counteract copyright piracy issues. The idea of this copyright mechanism is to provide the game free of charge but provide the gamer with lucrative offers in game. The user can then purchase specific contents or have access to exclusive offers and deals. The add-ons on a free-to-play game are usually very cheap, but there are plenty to choose from. In general, the marketing

model is aimed to increase popularity rather than profit from the title sales.

4.3.3 Crowd funded games

Often, a gaming studio runs out of resources or does not have adequate resources to complete a project. Some developers could reach out to their loyal fan base for help. The idea is to ask for financial help from the projected future users, giving them a sneak peak of how the actual product will look and allowing them exclusive rights. One example is 'Kickstarter'. This is a funding platform for projects that are funded by the general population. This methodology of crowd funding is still a new phenomenon in the world of game development.

The whole idea behind copyright and piracy protection is to provide a reasonable solution that provides better security while costing less of effort and money and also retaining customers. If these features are available in a specific copyright protection mechanism, this could influence potential developers and more studios to invest more and have a turnaround profit from the sale of the product.

4.4 Industry Standardised DRM solutions

In the next section, we will be discussing industry solutions for DRMs. We will provide a brief overview of the solutions that are accepted in the industry and the test cases where each mechanism is being used to achieve copyright protection.

4.4.1 SecuROM

Sony Digital Audio Disc Corporation, also known as Sony DADC, has developed a copyright protection known as SecuROM [89]. The objective of this copy protection is to prevent piracy in computer games. Sony does this by preventing unauthorised copying of the game and also making it difficult to reverse engineer the software. SecuROM is one of the most common DRM mechanisms currently used in the market. Many major companies, such as Atari, EA, Ubisoft, Microsoft, Sega, Konamy, Capcom, Lucas Arts, Eidos, etc., rely on this.

This copy protection is a very popular mechanism for computer games running commercially under Microsoft Windows.

SecuROM uses the ideology of manipulating the hardware to provide better protection mechanisms for the discs. The disc protection method in later versions of SecuROM is known as a data position measurement, which can also be used in conjunction with online activation DRM.

To differentiate between a copied material versus an original copy: (version 4.6) used by Sony utilises the q-channel of the disc. Within the disc, nine specific locations of the q-channel are broken by purpose, which acts as a flag or a marker. This is done using a vendor-specific key which can calculate and determine the exact locations. A calculation function is then injected, which will verify later if the disc is a genuine or a fake copy. The function is triggered when the game is launched: if the function is unable to read the specific nine broken sections when the game is being played, it returns a 'True' value. This 'True' value is a token of genuine authenticity confirmation. On the contrary, if the function is able to read any of the specified locations, the resulting verdict would be negative or 'False' and would mean, the user has a fake copy.

This mechanism by Sony was later enhanced to increase more durable protection (version 4.7) via the usage of a new scheme called 'Data Density Management'. A specific pattern used by Sony which degrades over time is the main feature used in this new technology. This pattern is much stronger, as it has the ability to reconstruct via high-precision time measurements. Compared to nine locations with the previous version, this version makes use of 72 locations to calculate density discrepancies which are spread out all over the disc.

In later versions, Sony also included a trigger function. This function always runs in the background and keeps checking if any of the protection mechanisms were removed. If a fault is detected, the function will trigger a negative value and disrupt the running of the game. This disruption is also known as 'crash' or temporary suspension of the running of the game. This can be particularly annoying for gamers, as it would persist in a fake copy. This disruption is expected to intentionally destroy the gaming experience.

However ,different franchises or different game companies may choose to hide the fact that the user is actually playing a fake copy and exhibit different gameplay tactics to stop the user from enjoying the game. Such an example is demonstrated by the game called 'Serious Sam'. The developers have integrated a specific mechanism and instead of crashing or quitting the game, the game-world gives the user an extremely difficult challenge that the user will not be able to accomplish at all. For example, when the trigger function has verified an unauthenticated copy, the game takes the user to fight an impossible scorpion that kills the gamer instantly, taking him to the 'Game Over' screen.

A similar but different mechanism is used by the game 'Batman Arkham Asylum'. The developers strip away the users inventory that are required to get to the next level, thus preventing them from proceeding with the game.

4.4.2 SafeDisk

Developed by Microvision Corporation, SafeDisk [90] aims to prohibit physical copying of DVDs and inhibit reverse engineering technologies. SafeDisk uses cryptography together with hardware protection of disc sectors. The technology behind SafeDisk implements the usage of digital signature. Every single time the game is launched, the authenticator will verify the digital signature on the disc. During replication of the disc, the digital signature will be applied. It is designed so it is difficult to obtain the digital signature from the optical disc drive. As a result, for every iteration of running the game, the gamer would have to reinsert the game disc.

With the rise of software like Daemon Tools, the first versions of SafeDisk were easy to overcome. Since Daemon Tools can create a virtual disc drive within the operating system and imitate the system calls, SafeDisk had to be improved.

Later versions of SafeDisk would refuse to execute if they were able to detect the presence of image-copying software such as Daemon Tools already installed in the system.

In version 1, SafeDisk started using encryption mechanisms to protect critical information about the game in an .ICD file extension. This file is a prerequisite to run any game

under SafeDisk, as the optical drive will also have an executable that would decrypt the ICD file to parse the information.

However, hackers learned to easily decrypt the ICD file and manufactured the famous 'crack.exe' for pirated games that allowed video games to be run without requiring a physical copy.

In version 2, SafeDisk overcame the 'crack.exe' issue via adding an extra layer of errors from sector 822-10255. This caused difficulty in copying and introduced greater chances during the copy to fail with errors. From version 2.5 to 2.9, SafeDisk implemented the introduction of weak sectors to cause copy-failures within the buffers.

In version 3, digital signature usage was re-introduced. The game loader executable was encrypted with a key to generate a digital signature which was integrated with the physical disc itself. The size of the digital signature was variable depending on the strength of the cryptographic encryption employed (3 MB-20 MB).

4.4.3 Denuvo

The team members behind Denuvo [91] were once a part of Sony DADC DigitalWorks. Denuvo was formed from ex-Sony members and provides an anti-tampering technology and DRM for software and games. The core working mechanism of Denuvo is not revealed. However, early reports suggested the anti-tampering technology used by Denuvo continuously encrypts and decrypts itself so that it is impossible to crack.

Official documentation of the company did not confirm any of the findings, since the idea was to keep the anti-tampering technology classified and hidden to provide better security. However, the Denuvo games require an online activation when there is a change in a hardware. Additionally, the number of times this change could be done was controlled by Denuvo, limiting it to four hardware changes every 24 hours.

Denuvo caused a stir in the market with their strong technology and raised the rumour of their technology of being secure compared to other DRM technologies in the market, but the developers had been modest and realistic with their security.

According to Digital Spy, it used to take a maximum of 20 days to remove the security protection of an average game within the hacking community. Upon initiation of Denuvo, the game "Lord of the Fallen" took a record number of 273 days to crack. This stirred rumours in the hacking community about Denuvo becoming the de-facto standard for DRM protection. As no system is un-breakable, statistics for Denuvo were by far the best the development community had in early 2014. To quote some figures, Denuvo was focused and developed with the top franchises in consideration e.g. the renowned FIFA sequels from EA sports. FIFA 14 used SafeDisc and was 'cracked' within 21 days of its release date, but FIFA 15, which used Denuvo, took 132 days upon release to be 'cracked' by the hacking community.

The development strategy of Denuvo was highly confidential and it was causing difficulties for the famous Chinese teams of hackers - Warez, PirateBay and more. Success of Denuvo relied on the fact that it was an anti-tampering solution that secured the DRM.

The developers of Denuvo state Denuvo prevents reverse engineering and debugging of the game. Denuvo was first hacked by a Chinese team named 3DM during late 2014 after a vigorous amount of effort and sophisticated techniques. The first failure of Denuvo was the game 'Dragon Age: Inquition'. The hacking community released information about Denuvo and referred to it as a box-key mechanism. 3DM stated that, Denuvo is a box which stores the game content along with its DRM and locks the box in a virtual environment. The lock is an encryption and the key for the lock is generated uniquely per machine. This meant that every single game had to be 'cracked' individually, adding more time to the calendar days. 3DM also added, "Denuvo uses 64-bit encryption that gets it's key from several variable from the hardware itself, it will need specific cryptographic keys". The technology behind Denuvo meant no single key could be used in a wide-spread manner to unlock a game anymore.

It has also been learnt that Denuvo generates a unique .exe file itself from the encrypted version of the game using a set of variables from the host machine. Nowadays, the hardware configuration of the machine can vary greatly, depending on subtle details, e.g. number of processor cores, bus speed of RAM, processor type, CPU clock speed, graphics card vendor

company, cache size, etc. Using all this information, Denuvo generates its own key using a set of variables and to break this lock would mean the hacking community had to work with a bigger number of permutations and combinations. The key generated by Denuvo using this combination is converted into cryptographic keys that are targeted to work on a specific machine, thus making sure the same set of keys will not work on a different hardware.

It is an extremely prestigious endeavour to be able to 'crack' a difficult DRM within the hacking community, and Denuvo performed exceptionally to beat the hackers. Denuvo ensured the reliability of 'cracks' and limited its failures greatly. Once a single machine was 'cracked', it did not mean the full game was compromised as a whole, which added extra efforts for the pirating community. The development of 'serial key generators' by developing unique algorithms did not help the hacking community at all.

The Chinese hacker 3DM, who was the first to break Denuvo, reached out to the user community to collect data and establish a pattern to better understand Denuvo's protection. He published a small program online and shared within his loyal fan base to participate in data collection to break Denuvo faster. His tool would collect specific hardware information as discussed earlier, e.g. CPU cores, vendor information, cache size, etc. Efforts to break Denuvo are difficult, but not impossible. The developers have been successful in making it difficult for the hackers, thus ensuring a small grace period to do business with the release of new titles.

4.4.4 Steam Client

Steam [92] is a gaming client developed by Valve [93] that provides a single roof for the gamers to purchase their games without having to worry about any specific hardware or ownership. With the constant struggle of having to own a physical copy of a game and having to carry or store it, Steam's digital solution has proved to be a very popular solution amongst the gaming community. Steam is more like a platform having extremely sophisticated DRM techniques embedded in them, which provides a bridge between the

Figure 2: Steam games are not bound to machines but are tied to user accounts

development community and the consumer base. Owning a game via Steam ensured the users do not have to worry about getting licenses to be able to play on different hardware. It also meant less manufacturing and customer support costs for the developers.

Steam has its own security mechanism, which will be discussed in this section. It provides a seamless and easy-to-use interface for the gamers and peace of mind for the development community. Once a user purchases or installs a game from the Steam market, the game is linked to the user's Steam account rather than the hardware the user intends to use. This has also given this a new terminology and is widely known as 'Steamworks CEG'. This also means that manufactures can spend less resources worrying about hardware binding and can also attract a bigger customer base who intend to play on multiple gaming platforms e.g. PlayStation, Xbox, PC, etc. Steam pursuing the digital download pathway also ensured that users did not have to own their personal physical retail copies of the game, and this also mean that the games are never lost or damaged due to natural causes, e.g. bad sectors, broken disc, lost or stolen, etc.

The idea behind Steam's marketing is a better understanding of the security market. Rather than focusing on developing a game itself, they worked on providing a platform that developers and consumers can trust. Initially, Steam started with three gaming titles - 'TeamFortress', 'Counter Strike' and 'Half-life'. However, to date their popularity has taken

them to store over 5,000 games and millions of user accounts.

Steam encrypts games, allowing developers to submit games to the Steam client without having to run the risk of being pirated and leaked before release date. The core mechanism behind Steam's protection mechanism is known by 'Custom Executable Generation' (CEG). It creates a unique .exe for every single user, linking it to their database without any hardware bindings. This ensures that users are not limited to play their purchase from a specific set machine; they have more freedom and scalability upon making a single digital purchase.

The developers of Steam have built a Steam 3 DRMS server which allows gaming studios to submit their video game. Upon a successful purchase or an install made by the user, the Steam engine will generate a set of metadata, uniquely identify it with its Steam 3 DRMS server and link it with the specified Steam account. The CEG technology of Steam ensured the user has more freedom and is not tied to a specific set of hardware; instead, the game is tied to the Steam account.

Steam enforces its own policy when a user chooses to install this client on their machine. It will force the user to connect to the internet upon any update released by the developer. This feature ensures that the security is kept up-to-date and the revenue loss is minimised. Upon a preliminary installation, a new set of CEG is generated and the user do not need to have internet connection for future offline sessions. However, once an update is released or when the user intends to play the specific game on a different hardware [94], the internet connection is a requirement and a new set of CEG is generated. The above also applies when a new update is being released.

The way it works is depicted in Figure 2: instead of conventional DRM techniques where most developers tie their license key to a specific hardware, Steam works differently. We have tested ourselves that Microsoft Windows 10 Professional is known to have their license key tied to the motherboard MAC address. As a result, when we tried to install a new graphics card, Windows prompted the user to update the license key again. We have also tested with removing other hardware e.g. hard disk, DVD-ROM drive from the motherboard; however, on those occasions Windows 10 Pro DRM did not require us to apply our license key again. The whole process can be inconvenient from the end-user point of view,

since we own the copy of Windows. Swapping out a graphics card and asking us to verify the license can be annoying, since most of the time the license key is not available to hand.

Steam protection mechanism skips this inconvenience and works by tying the copy of game to the user account itself rather than the hardware the gamer intends to run it on. As a result, the gamer is able to play the game on multiple devices without being required to verify their game copy every time. As discussed above, every time the user launches the game via Steam on a specific device, Steam generates a unique executable that can only be run on the specific machine the user requested the game to be run on. For example, if the user wanted to play the game on their Mac, they could launch Steam and play the game. However, if someone were to copy the executables and try it on a different Mac with the same user account logged in, it will not work. The user would have to login to Steam server and launch the game via Steam interface; thus requiring Steam to re-generate the unique executable for that specific machine. This protection mechanism makes Steam very portable and quite lucrative to the customers and the potential developers.

The Steam community has multiple layers of protection embedded in theitm. CEG is one of them. The Steam DRM is a scheme operated by Steamworks Digital Rights Management (Steam DRM) which builds a wrapper on a game executable. It provides protection to the game via deterring users to reverse engineer or debug the game.

The very first version of Steam DRM did not carry any encryption. It used a specific set of data to generate metadata to authenticate the user to allow them to play the game. However, from 2008, with the release of version 1.5, Steam started adding digital signatures which are verifiable during launch time. This prevented users from launching any unowned game not tied to their specific account. Encryption was introduced in a cipher which used cipher block chaining (CBC) mode. From version 2 onwards, Steam DRM had major improvements. The DLL library of Steam DRM is encrypted and is loaded using its own API rather than using shared Microsoft's Windows API. This ensured encapsulation and also meant that the activity of the Steam DRM is not written on the disc which could be reverse engineered.

This version of Steam DRM comprised of a three-stage procedure. Stage one decrypts

the required files, mostly configuration files. The configuration files were encrypted using the odd cipher CBC, and they contained a pointer which could locate game information, application ID and specific details about the latter stages of the process.

In the second stage of this process, the outputs from stage one are used mostly as parameters for stage three. However, the second stage revolves around obfuscating this information or the output from stage one before executing stage three.

Third stage is decrypting the DLL to give away the control for loading the game. It should be noted that in stage two, the developers have introduced obfuscation, which may appear the data to be very small, 5-10 kilobytes.

However, it carries real data at different offsets, making it harder to distinguish between a real and a fake set of data. In addition to this, to obtain the offset embedded in the DLL, they would have to be decrypted first. The whole process ensures the DLL file is unique to every piece of hardware rather than limiting itself to a single game. This is because the offsets are very random and used on a variety of .exe due to the pre-configured CEG execution. Steam DLL also uses Public Key signatures, and this is verified at execution time.

The protection mechanism is not limited to pointers only. Steam DRM encrypts game code using AES-256. Version 3 enhanced the Steam DRM into a new level via merging both stages one and two, enhancing better performance. Not to mention, Steam also requires the client to be run first before allowing the game to be executed. This also adds to the protection scheme.

Limitations Limitations of Steam mostly comprise of bypassing the server instead of qualifying Steam as it shortcomings or flaws.

Usually, Steam is the housing wrapper which hosts the games. Once the games are launched, a specific game can have its own set of gaming servers. Since Steam is acting as a wrapper which provides a secure environment to launch the game, they do not have any control over how the game is being played. For example, if Steam hosts Dota 2 game within their boundaries, they will provide all the necessary set of protections to stop illicit users from starting a tampered version of the game.

The way Dota 2 and most other games work is that once the game is launched either via Steam or any other gaming client, the gaming server or the Dota 2 game has their own set of local servers that they use to host gaming sessions. These could be for many specific purposes. Mostly, game servers do it, to host gamers who would like to play a specific game mode or when the gamers are in a specific geographic location, e.g. gamers in Asia are most likely to be hosted on a specific server to avoid latency or connection issues. The limitation of Steam comes in here, where the gamers could hack the specific game-mode servers and have their own set of rule-set. This does not violate Steam architecture but acts as more of a 'bypass' mechanism and mostly relies on the detection ability system of the game-mode server.

The protection mechanism of Steam is very popular within the online multiplayer community. Purchasing a game via Steam means that the users connect to Steam's user server first and then log in to the specified game server. The limitation we discussed here mostly summarises the fact that if users download or purchase a game via Steam's portal but connect to a private server of their own creation, they can control the game in their own way and have added advantage.

4.4.5 Watermarking

In the domain of multimedia and modern art, a seamless solution is given by digital watermarking protection [95]. Watermarking specific copies of the specific user with an appropriate opaque marker has become an industry standard to reduce illicit distribution. Videogames are also seen and considered a subset of this multimedia domain. However, it is very difficult to provide similar protection, as it is done from movies and other digital arts. This is due to the fact a videogame may consist of using manuals, graphical images, videos, audio content, 2D and 3D models and many more. To provide watermarking in every single stage is a very challenging task.

Despite it being a challenging task, the motivation to pursue more and more secure algorithms has not stopped.

5 Cheating in video games

5.1 Introduction

Gaming is a multibillion dollar entertainment industry that manufactures products for a very large population throughout the globe. 'Fun' feature was the primary focus when the games were designed initially; however, with the growth of competitive gaming, security has become a very important issue. The fun feature is diminished when illicit players or gamers poisons the gaming environment with 'cheats', which lets them establish a dominating and unfair advantage over honest gamers.

This chapter discusses the current game cheating techniques deployed in the industry. It discusses the possible branches of cheating and elaborates on the planning and research of a specific branch on the cheating.

As per our study, we will only be focusing on the 'software' part of the video games as the full domain of video game can be quite vast. In addition to this, there is much more stake in this domain of the video game instead of the retro-based games. The frequency of competitions that involve a lump sum money are mostly hosted on the 'software-based' video games. Therefore, we have decided to cover this domain.

5.2 Cheats

We identify all cheating forms known to us, as they have occurred or might occur in video games. We also briefly discuss some general properties of these cheats. Our initial task is to find the known forms and shortlist them to identify a reasonable target to work throughout the next few years. We identified sixteen common cheating forms in this document. While continuing our study on game cheating, however, we have seen the need to refine this list, since some cheating forms are identical to others, and having a lower number reduces ambiguity.

5.3 Threat model

The main motivation of this work to understand how massively multiplayer online games (MMOGs) actually allow large groups of people to play a single online game simultaneously, with fair gameplay. In addition to that, we ask what are the problems that are still faced by the players playing this game. We can never rely on a client machine, since it is not under supervision by any human referees and there are numerous chances of cheating; the machine can be altered and made to perform to user's accord. Our motivation for the study is to set up an infrastructure on the client which may allow us to measure the integrity of the client prior to running a game. With extra and added assurance set on a client side, the mechanisms of cheating and unfair plays are expected to fall.

5.3.1 Background

In MMOGs, developers of the games have to make sure the gaming experience is kept interesting to keep players engaged in the game and make sure they return. However, with incremental updates focusing on new features, very often the security mechanisms of these gaming platforms are overlooked. Companies tend to invest more on their active users and the security concerns are not prioritised unless there is a stake in their revenue.

In a client-server architecture, most states are stored on the server. Upon receiving information from the client, the server processes it at its end based on the server local state and time. In these situations, the server has the most control, since updates and information processing are compared with the data that are stored on the server despite clients' sending falsified information.

Despite the secure infrastructure of client-server architecture, it does not stand as a foolproof protection architecture. A number of vulnerabilities exist. Servers cannot properly detect time-cheats, packet modification, etc. More of these kinds of threat are detailed in a taxonomy proposed by Yan and Randall [96]. There is also another taxonomy proposed by Ki et al. [97], but this taxonomy mostly deals with organisation in a hierarchical manner, e.g. client, server, network, environment, etc., rather than classifying the types of cheats.

Different cheating techniques have been implemented to hack into games [98] and gain unfair advantage. A few of them include:

1. *AimBots*, this is where a client is modified by an external running program that aims the mouse cross-hair automatically to focus on the enemy. This reduces the reaction time and makes the gameplay easier.

2. *WallHacks*, this is where the video graphics driver is tweaked to display hidden contents across the wall which other legit players cannot see.

3. *MapHacks*, this is where the local client obtains information about a map they are not subscribed to.

4. *Speed and movement hacks*, where through a variety of mechanisms, a client enables their avatar to move more quickly and in ways not intended by the game publisher.

Some countermeasures have been proposed already in the academia. They are focused on the elimination of cheating in networked games. ORTS [99] is a security approach for client-server real-time strategy (RTS) games. The server stores and updates all the important and crucial state changes. It also calculates the minimum distribution of state information that the client needs to render the gaming view.

Fung [100] proposes detecting movement cheats such as speed-hacks by the inclusion of a trusted third party, which will validate the legality of each move reported by participants, rather than simply accepting those moves without verification.

Gamers' behaviour is also a major attribute when the topic of cheating is discussed. Validation of the input stream for virtual environments [101] allows detection of player skill augmentation cheats, such as 'AimBots'. RET [102] is a scheme for profiling gamers activities, then observing an activity stream of a control and measuring if the outputs could have been generated by the same player. Chen et al. [5] analysed 'Quake 2' traces to show that computer intelligence (bots) can be differentiated from human players 98% of the time with a 12-minute trace. This phenomenon can be used to separate the usage of bots in MMOGs and increase fairness.

Laurens et al. [103] profile the behaviour of gamers to differentiate between a cheating character and a fair gamer. They show an example where 'WallHacks' cheats are detected by examining the fraction of time a client aims at an occluded opponent. Players with 'WallHacks' can see opponents who should be occluded by walls, and aim at those opponents more often than non-hacked clients. In a similar vein, DeLap et al. [104] propose a scheme for performing runtime validation of transactions in games, flagging validation errors upon receiving a falsified data from the player performance statistics. For example, an avatar which earned a huge amount of gold per unit time would have their transaction histories audited using this scheme.

Aggarwal et al. [105] took a step away from server-based virtual environment security by empowering proxies. These proxies mostly share the security load. Their authoritative proxies can also cache the game state and validate some client state changes on behalf of the central server.

Some virtual environment security solutions are intended to work with peer communities. They mimic some of the properties of client-server architectures by electing participants into privileged roles, and trusting those participants to behave correctly.

The public server approach [106] allows massive multiplayer online (MMO) gamers to set up their own game servers. Centrally issued certificates protect critical states - such as distribution of valuable virtual property - to prevent replication cheats. Additional calculations are performed to provide evidence that player-owned servers are obeying the virtual environment rules, rather than distributing disproportionate amounts of virtual resources. This is a promising phenomenon, but this architecture has scalability issues in terms of the PKI used as evidence and proof of ownership.

Kabus et al. [107] present a variety of mechanisms for detecting cheating, such as electing auditors to verify event streams provided by participants, and using trusted computing bases to protect data streams and executable code.

Some solutions can work in peer-to-peer environments as well as hybrid and client-server environments.

Chambers et al. propose a round-based bit commitment protocol [106]. 'Warcraft III'

[86] adversaries periodically send hashes of their moves to each other, then exchange full move logs after game completion [108]. When the full logs are received, each player can validate that the sequence of moves provided corresponds to the hashes received during play, and that the movement sequences are valid.

5.4 Attack scope

Listed below are the 16 different types of cheating techniques.

5.4.1 Cheating by modifying Client infrastructure

With complete control over the client, cheaters can hide their presence by modifying the operating system, or disabling or spoofing anti-cheat software [109]. They can also remove the cheat from the system just before anti-cheat software runs. On top of these techniques, the adversary can employ memory tampering, anti-debugging, obfuscation, and evasion techniques that make the gaming client untrusted.

Analysis In [110] the authors propose the use of tamper-resistant hardware to detect cheaters. They [111] provide an empirical study of online game cheating. On the attack side, in [112], is presented "The Supervisor, a kernel-level rootkit made specifically to bypass The Warden, Blizzard Entertainments anti-cheating technology". In the Defcon talk "So Many Ways to Slap A Yo-Ho: Xploiting Yoville and Facebook for Fun and Profit", the authors showed how to cheat on the Yoville game. It has been discovered amongst all these papers, the most popular [88] tool to tamper with game memory is called 'Cheat Engine'.

5.4.2 Cheating by Collusion

Collusion cheating has also been widely seen in online card games, e.g. online poker. Online poker is a game played between a number of persons ranging from two to eight. Unlike chess, in which all pieces are on the board and known to each side, poker is a game with hidden information. Each player knows only the subset of cards they have access to. How-

ever, by illicitly exchanging card information over the telephone, instant messenger or the like, collusive cheaters can gain huge advantages over honest players.

Analysis Reiter and Rubin [113] has shown how they collect users in a group and call 'crowd' to browse the web anonymously. To enter the group, a user contacts the server to gather more information about other members. A user that is interested in a web page forwards the request randomly to another member in the group. Upon receiving the request from another member, a random choice is being made to either forward the page to another member or process it themselves. Later, the server provides same pathway to create a seamless interaction to hide the fact that it has been routed to other members and has reached the intended client.

In my point of view, the work done was done long time ago and its relevance to the gaming world is not adequate enough.

Work done by Freedman and Morris [114] focused on creating an anonymised layer. It provides anonymous effort IP service and is transparent to applications. Their system uses layered encryption and fixed-length messages and obfuscates traffic to make sure there are no attack in transit.

The work done seemed reasonable; however implementation details and the success rate of the paper was not mentioned, and taking the idea on board would not lead to further work.

5.4.3 Cheating by Escaping

This form of cheating may be carried out without any technical expertise. A cheater simply abuses the operating procedure of a game. One common case that we have observed in many online games is escaping: a cheater disconnects themselves from the game system when they are going to lose. In cases, disconnection results in a match being not scored and a legitimate player loses.

Analysis Ideas connecting to this domain are quite vague. Several ideas suggested unplugging the connection with the ethernet port, thus switching off the machine. However, due to the network connection issues in consideration, it is difficult to distinguish between an intentional disconnection and a genuine network flaw.

5.4.4 Cheating related to virtual assets

Virtual characters and items acquired in online games can be traded for real money. A cheater might order a virtual item, receive real money for the item, but never deliver it as agreed. Alternatively, the cheater might get hold of another's personal profile by illicitly gaining access.

Analysis It has been thought that the idea of Blockchains could be implemented in this scenario. There is a study done by 'Hunter Coin' [115] which establishes the idea. However, the work done is for a turn-based game and discussed limitations of real-time games. This is because of the time that is needed for a block to be added on the chain, and to provide a seamless experience, the gamer might be frustrated to wait that long.

However, the idea could be extrapolated to provide attestation for items or valuables within the inventory that does not require a real-time collaboration.

5.4.5 Cheating by exploiting machine intelligence

Using a robot to gain advantage is a new form of cheating. For example, the advancement of computer chess research has produced many programs that can compete with human players at the highest level. When playing chess online, a cheater can replicate their moves on a strong chess machine and reproduce the results against their opponent.

Analysis There are multiple number of studies done [116, 117, 118, 119] on this domain. Different machine-learning techniques have been put into use to achieve the best results. Usage of WEKA was used in most cases. The papers did not specifically specify which MLA (Machine Language Algorithm) suited best in the situation, rather they did an experimental

analysis and showed the results of a specific algorithm e.g. Naive Bayes, SVM (Support Vector Machine) to be the best algorithms.

In [120], authors discussed about behavioural analysis to detect cheaters in Online First Person Shooter games. The game was developed using Unity3D game engine and it had client-server architecture. The game logs were stored in the server side. Data were analysed using implemented machine learning classifiers in WEKA.

In [121], party play log analysis, work done included with establishment of threshold levels for the activities that may be performed by game bots. Based on this, they built a knowledge base of detection rules. The rules are generic.

Basically party play was used to analyse user behaviour. Other work involved users moving or aiming. They went on discussing, party bots will have different patterns when it comes to fighting or moving. Human players may not stick to their style of play, and over a period of time the patterns reflect the usage of bots.

5.4.6 Cheating by modifying external software

Without modifying game programs, configurations or data on the client side, a player can cheat by modifying the device drivers in his operating system. For example, he can modify a graphics driver to make a wall transparent so that he can see through the wall, locating other players who are supposed to be hidden behind the wall.

Analysis There is a real-time map-hack study being proposed in a very good conference, S&P [122]. Authors discuss a generic tool, Kartograph which they have made - it lifts the fog of war in online real-time strategy games by snooping on the memory used by the game. Their main idea is based on the fact; they share the game state amongst the other players using a distributed mechanism.

More focus to be had on this paper and see if it can be improved or the idea taking forward.

5.4.7 Cheating by denying service to peer players

A cheater can gain advantages by denying service to his peers. For example, a cheater could delay the responses from his opponent by flooding his network connection. This may allow the cheater to gain extra lives and specific game servers may allow this considering network flaws.

Analysis No study is available in the academia to analyse in detail.

5.4.8 Timing cheat

In some real-time online games, a cheating player can delay his own move until he knows all the opponents moves, and thus gain a huge advantage. This look-ahead cheat is one kind of timing cheating.

Analysis No study is available in the academia to analyse in detail.

5.4.9 Cheating by compromising passwords

A password is often the key to much of or all the data and authorization that a player has in an online game system. By compromising a password, a cheater can have access to the data and authorization that the victim has in the game system

Analysis No study is available in the academia to analyse in detail.

5.4.10 Cheating due to lack of secrecy

When communication packets are exchanged in plain text format, one can cheat by eavesdropping on packets and manipulating them to gain advantage.

Analysis No study is available in the academia to analyse in detail.

5.4.11 Cheating by exploiting lack of authentication

If there is no proper mechanism for authenticating a game server to clients, a cheater can collect many ID-password pairs of legitimate players by setting up a bogus game server. Similarly, if there is not a proper mechanism authenticating a client, a cheater can also exploit this to gain advantages. For example, in Asia pacific, where the usage of game cafes are abundant, people tend to login to public computers and their details are easily captured.

Analysis No study is available in the academia to analyse in detail.

5.4.12 Cheating by exploiting bug or loophole

This form of cheating exploits a bug or loophole in-game programs or the game design itself, without involving any modification of game code or data. Once discovered, such a bug/loophole will give knowledgeable players a major advantage. Sometimes, this type of cheating is debatable in the gaming community since the bug can also be referred as a feature in-game.

Analysis No study is available in the academia to analyse in detail.

5.4.13 Cheating by compromising game servers

A cheater can tamper with game server programs or change their configurations once he has obtained access to the game host systems.

Analysis No study is available in the academia to analyse in detail.

5.4.14 Cheating by internal misuse

A game operator usually has the privileges of a system administrator. It is easy for an insider an employee of the game operator to abuse this privilege. For example, the admin could easily increase the number of coins in any account without others noticing.

Analysis No study is available in the academia to analyse in detail.

5.4.15 Cheating by social engineering

Often cheaters attempt to trick a player into believing something attractive or annoying has happened to him and that as a result his ID and password are needed. It is also known as phishing-scam

Analysis No study is available in the academia to analyse in detail.

5.4.16 Cheating by impersonation

A gamer could hire another gamer to compete or play on behalf of him. If it is an online gaming competition, the other users will not be able to understand who is actually playing the game.

Analysis No study is available in the academia to analyse in detail.

5.5 Objective

Following the types mentioned above, our main objective is to better distinguish and categorise the types of cheats which manipulates the client. This would help disintegrate the broad challenge of tackling protection against cheating and allow modularisation to provide solutions.

6 Future approach

This chapter discusses about the possible future work that could be completed based on the research carried out. We start by discussing an analysis on cheating detection scheme and how this can be provide a baseline to carry out the tasks in the future. With the nature of thesis being an MPhil instead of a PhD, the outlines are discussed which can provide a guideline for the forthcoming future.

6.1 Cheating detection

Detection of cheating varies widely. The accuracy of the detection is of paramount importance, since professional players can be deemed to be cheating whilst in reality they are just very good players with very high activity rates and tremendous reflexes and reactions.

However, specific companies have created their algorithm to protect their platform to try and ensure adequate measurements to provide a means of fairness. Annoying traits of these companies are documentation. These companies are not keen on releasing any information or whatsoever about how they operate their cheat detection engines. The famous one being valve anti-cheating (VAC) [123] adopted by the gaming giant Steam. More cheat detection engines follows, Warden [124], Punkbuster [125].

Detection techniques dealing with cheating in P2P games can be classified such as Game Log Verification which is an analysis technique to find anomaly or detect bots, in other words, a common technique is that all actions are audited and verified for security breaches [126]. One of the forms of verification that can detect cheaters is comparing hash messages of future updates with the actual updates. For example, at the start of the game, each player, it is necessary to replace the hash value of the initial state of the game, at the end of the game, each player exchanges all the operations he issued and then to confirm the validity of the hash value and the state or operations to simulate the game again [126].

But for specific hacks like memory exploits and code injections, there are no specific go to methods adopted by the hacking community. There is ‘CheatEngine’ available to the

gamers, which has a memory traversal software together with decrypting mechanisms.

There is also a peer based detection mechanism. It is a centralized authority in each region, that is, the super-peer; some peers are elected to be super-peers, which operate both as a server to a set of clients, and as an equal in a network of super-peers. On the other hand, this node can perform most of the security checks, assuming it to have a high reliability [127]. Another hybrid approach is invigilation reputation security (IRS) for purposes of controlling and eliminating game cheaters. This is where communication is handled by the server and update execution is managed by peers. In this approach the server assigns a proxy peer to each peer [128]. Proxy peers are selected at random, and will be assigned on a regular basis. All the executed updates and returning results of the peers are done by the proxy peer. The server is responsible to relay the message between the peer and its proxy. The checks are such that, the server runs a quick revision test to see if the result is returned while if it is possible according to the threshold limit. Conflicting updates and a percentage of updates randomly chosen are then re-executed by a selected monitor peer to verify the results and detect cheating [126]. Moreover, Mobile Guards are used to ensure the integrity of the protection mechanism, which aims at preventing cheating through modification of game client. Mobile guard is a small code segments downloaded from a trusted server, game client will be verified using a checksum and encryption game data [126].

6.2 Objective

Following the types mentioned above, our main objective is to prohibit cheats and restrict a gamer from gaining advantages over honest players.

We plan to reach our goal via following a set of objectives.

- Prepare a shortlisted scope from the Cheat Types
    - Cheating by modifying client infrastructure
    - Cheating by Impostors
- Narrow the scope and work on a specific problem

- Detecting Impostor Cheats

- Follow up on the next shortlisted topic
  - Modifying Client infrastructure
    - AimBots
    - WallHacks
    - MapHacks

6.3 Background: Impostor cheat

There are certain game genre within the gaming industry, MOBA, massively multiplayer online role-playing games (MMORPG), FPS, strategy, simulation, puzzles, etc. The problem scope is being narrowed down to focus one of the major popular genre - MOBA. An application of this domain - Dota 2 [85] has been chosen as the preliminary game application, however the scenario could also be applied to other famous franchises e.g. Starcraft, League of Legends, etc. For the purpose of this study, we have chosen Dota 2. This is due to the fact, we have experience of playing this game over a span of 12 years with over 9,000 hours of game time logged in Steam. This statistic means we have seen the game develop over the years on a first hand basis. The game mechanics of Dota 2 can be exploited easily to implement the impostor cheat. Furthermore, there is evidence to suggest impostor cheating has been successfully executed in a gaming tournament where the prize money was worth tens of thousands of dollars.

Dota 2 developers have created a mode that is aimed at experienced players who want to play in a more competitive environment and know their matchmaking rating (MMR) . Dota 2 matchmaking [129] has always calculated MMR and used it to form matches, established prerequisite values to attend and participate tournaments and develop oneself. The current prize pool for the gaming competition has made the gaming giant Valve to earn 80 M USD [130] this year entirely from one tournament known as ‘TI6’.

Any competitions for this game is hosted by multiple sponsors, companies, start-ups, local organization and more. Big prize money is given out on Valve-hosted competitions where the prize money is usually a very large amount. Valve hosts one major big event known as 'The International' (TI) every year [130] and numerous small events that build up to the TI. Valve has earned 80M USD [131] from their TI6 and have given out a quarter of their earning as prize money to the participants which was a remarkable - 20.6M USD.

Our work done in this domain will concentrate on Valve-only events that are conducted both online and on main-stage key arena. We will mostly focus on the online tournament however we shall also keep the main-stage key arena cheats under consideration. The focus on online is due to the probability of having an impostor attack on an online tournament. Furthermore, before a team is being allowed to participate in a main-stage key arena event, they have to qualify through multiple online tournaments. The initial online tournament begins via tier management system. Players of a certain MMR rating tier will play with similar tier before reaching to main-stage arena where MMR do not matter anymore.

The journey to qualify for a Valve tournament begins with creating a new account with Steam. Steam is the gaming client that helps connect the gamers to the Dota 2 servers. A new gamer starts with a locked MMR rating where the gamer is required to complete approximately one hundred and fifty games to acquire a rating. Each game lasts an average of fifty minutes, played with other nine players.

6.3.1 MMR calculation

Dota 2 uses standard techniques to quantify and track player skill. Each player is assigned an MMR, which is a summary metric that quantifies the gamers skill at Dota 2. After each match, Dota 2 updates MMR based on what happened in that match. In general, when a gamer wins, MMR will go up, and when they lose, MMR will go down. Win/loss is the primary criteria used to update MMR, but individual performance also plays a role, especially when the uncertainty about MMR is high. It is possible for an individual MMR to increase after a loss or decrease after a win, but in general the winning team's average MMR will

increase and the losing team's MMR will decrease.

They also track their uncertainty about MMR. New accounts and those playing in Ranked Matchmaking for the first time will have high uncertainty. Higher uncertainty allows larger adjustments after each match, and lower uncertainty leads to smaller adjustments. Together, the MMR and uncertainty can be interpreted as a probability distribution of performance for the next game: the MMR itself serves as the mean of this distribution and the uncertainty is its standard deviation. If the match outcomes (both the win/loss and individual performance) repeatedly match the server's expectations, the uncertainty tends to decrease until it reaches a floor. A surprising match outcome will tend to cause an increase in uncertainty. This methodology is only for a gamer when he is playing with nine unknown players publicly and is referred as 'solo MMR'. There is another MMR calculation known as 'Party MMR' where the gamer plays with a friend, and the value is calculated similarly.

For the purpose of simplicity, we would like to focus only on games played 'solo' or with nine complete strangers. This is because, the MMR calculation and the game mentality is quite different between each other. Very often, friends will join up over a weekend night to have some fun and enjoy their time rather than playing competitively. This adds extra noise in the study and would require much more sophisticated filtering to pick the games that were played to earn MMR. The 'Party MMR' section is something that can be pursued as an extension to this research work.

In a 'solo' game, players being by creating game accounts. When skilled players create new accounts, they follow a different trajectory than amateur players. Their MMR rises relatively quickly, placing them into the upper left-hand corner of the diagram in the MMR graph as shown in Figure 3, where they will be matched with other players whose skill is high relative to their experience level. However, throughout the Dota 2 journey, the skilled players find their position levelled out upon playing more games.

Measuring success in matchmaking is difficult. Players' appraisals of matchmaking quality are highly correlated with their recent win rate. The developers make design decisions objectively using data. The gaming giant gathers lots of data and uses a statistical tool

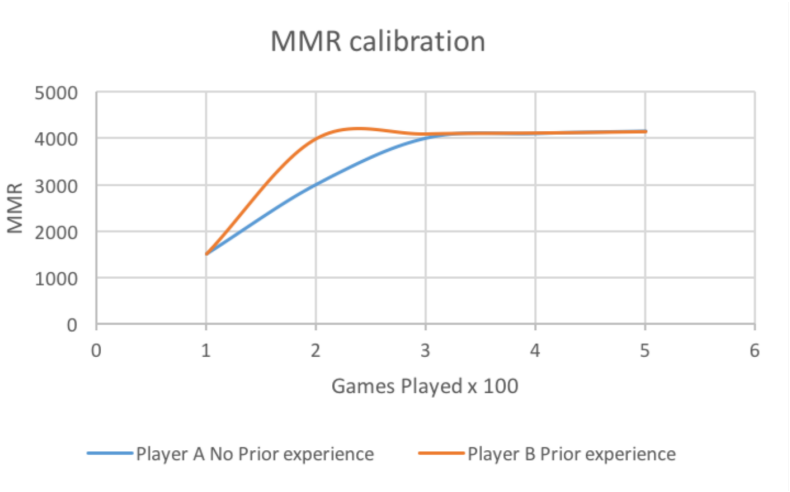

Figure 3: MMR graph

known as logistic regression, which essentially works by trying to create a function that predicts the odds of victory. This function contains several coefficients which determine the MMR bonus given to players in a party. Then they use numerical techniques to solve for the coefficients that produce the function which is most accurately able to predict the match outcome.

6.3.2 Problem Scope

Building up a good profile and playing in the elite league of players (having high MMR) is a challenging task. Higher MMR requires a great deal of gameplay hours. The benefits of having higher MMR values are also rewarding. Players turn out to be gaming celebrities where they stream their playing style in popular streaming websites like YouTube and Twitch. This acts as a very big source of revenue for a lot of gamers as they get to showcase their skilful playing style to the world. Apart from monetary benefits, they come with bragging rights within friends and peers. This also gives a message out to the community, who are actively looking to hire new players in their team roster.

Gaming events are mostly defined of two types: minor events and major events [132]. Minor events [133] are mostly any tournaments where the competitions could happen in a small local region or at best a country-wide event. They tend to vary in prize money

from $100-$20,000. Major events tend to be organised as a broader event, where people from continents will join together to represent their teams. Major events have always been organised by Steam and Valve together and their prize money is in millions of dollars. They usually happen once every year and is famously known as the internationals (TI).

In big major events like 'The International 6' (TI6), the gaming arena would be monitored by human referees. The human referee would monitor cheats and can easily prohibit an impostor by establishing their presence.

Impostor cheating problem arises mostly with minor events where the games are hosted and played online. Players could compete from all parts of the country. To keep the costs low and filter out unskilled players, these events do not happen in a physical venue where there could be human referees. This gives the scope to an impostor to apply some cheats. The lucrative part of applying an impostor cheat is, the champions from these minor events are guaranteed a spot in the major event. This means, the participating team is guaranteed a small sum or a minimum sum from the prize pot of the major event. Therefore, in most semi-major events, just getting to qualify or barely would secure a minimum of thousands of dollars. The prize money for the last place of a semi-major event 'The Manila Major 2015' [134] was 50,000 USD. The total prize pool for the gamers were 3M USD split within 16 teams, with the last team guaranteeing a prize pool of at least 50,000 USD.

If all genuine honest players were to participate, the tournament would be very competitive. 'Wings Gaming' are the proud champion of TI6, winning 10M USD themselves out of the 20M USD prize pool.

In a scenario, if the players of 'Wings Gaming' were to play the qualifiers game, as impostors, let's say for a team called 'IGN gaming', the players would most likely win a spot in the majors [135] for IGN gaming and thus a guaranteed share in the prize pool. This would secure 2 objective. 1) Monetary benefit and 2) For 'Wings gaming', they are guaranteed to face vs easier opponents. Because, 'IGN gaming' is incapable of getting qualified by their own skillset. This also means, a genuine spot for another worthy team is also being occupied.

6.3.3 Motivation

Team VP has been caught [93] in the act of performing an impostor cheat during an ongoing semi-event, 'Summit 5' grand final qualifiers [87]. Suspicions began when one of the players of VP lost internet connection. Later on the match was analysed by the community to raise specific concern about the gameplay behaviour of the player. Later on, to preserve further hearing and investigation, the team succumbed to the cheat themselves and withdrew their participation.

In another major Valve event, it was discovered one of the gamers shared their competition login details and 3 out of 5 games were played entirely by another person. This event had physical existence of human referee and the gamer was still able to cheat. The cheat was later confirmed via rewinding CCTV camera footage.

The news of this new kind of cheating has surfaced in the gaming community and it draws the possibility of more undiscovered cheats gone un-noticed. Competitive gamers expend their life, time, effort and money to prepare for these events. Having unfair plays done via an impostor based cheating makes it extremely difficult for the genuine players to pursue their career. It may also destroy their life and lower their self-esteem and cause social problems.

The current software-based cheat detection is done via Valve's Anti-cheating system (VAC) [123]. However, VAC is unable to flag any of these cheats due to the reason that the entire game is being played fairly but by a different person, with their own consent.

6.3.4 Objective

Following the problems mentioned above, our main objective is to devise an algorithm that will analyse the player's in-game behaviour taking into account of the data of the players previous games. We plan to use a combination of in-game variables to create a pattern that would depict and raise flags to certain users who could be aiming for an illegal competitive advantage.

6.3.5 Solution

Apart from the ability to enrol a team into a competitive tournament, the bragging rights for certain gamers are also expensive. Gamers would pay money to specific online companies who would ramp up their MMR to a higher value so they would be able to gain bragging rights within their peers. Due to the well-designed MMR system of Valve, these gamers would fall back to their designated MMR levels upon spending more hours and we have been able to gather data that these gamers would return to the same companies again to use their service to enhance their MMR more than one time.

This gives us access to good data and bad data. We have also been able to gather genuine legitimate data from 60 players over a period of an average gameplay of 2,000 hours each. There has been previous solutions designed with the usage of Machine learning algorithms [119, 120, 5] that generates success rate upon using specific algorithm Bayes, k-th neighbour, etc. However, these results are tailored to verify a hypothesis. We plan to create a hybrid algorithm that takes into account of behavioural pattern together with machine learning technique which would allow us to detect an impostor.

Our hybrid algorithm would require experimentation with different machine learning algorithm to determine success rates. The game of Dota 2 publishes above 100 in-game parameters as a statistical update upon finishing a game. The target is to cater as many of the variables possible to determine if there is a recurring pattern for an impostor cheater. Figure 4 below represents the in-game data from a specific player 'Carlos Brathwaite'. In-game specific parameters are listed as Gold Per minute (GPM), Kills, death, assist (K/D/A), experience earned per minute (XPM), etc. as depicted in Figure 4. Data discussed here are from one of his latest matches. Figure 5 shows the list of latest games being played by the same gamer. Figure 6 shows the specific clock times of a milestone that has been reached within the game. There is a lot of information available from one single game, which gives very detailed data-set. Comprehending the gaming activity based on this data will allow us to analyse gamers behaviours. The next bit is experimentation and implementation.

Figure 4: Details of one of the games from the latest games played

6.4 Related Work: Impostor cheats

There are multiple types of work done on the domain of behaviour analysis. Work done with the involvement of behavioural biometrics have been done by Al-Khazzar [136]. Three unique virtual worlds are designed and implemented with different 3D environments and avatars, they simulated the different environments of virtual worlds. They were used to identify the users of a virtual world based on their behaviour within these environments. For example, if we look at an example of a game called Sims, the game creates a virtual world with people, where the gamer can choose to do anything with the people. They can make them interact with other people, make them dance, eat food, or even use the washroom. The series of events a gamer can choose will not be the same as the other gamer. i.e. one gamer could create a people in the game and make them eat food all the time, whereas another game could opt in to make the people eat less food. The patterns of how they interact or play the game can reflect a gamers style of play or mentality.

Jain [137] reported that biometric systems based on the usage of fingerprints [138]. It

Figure 5: List of latest games played

Figure 6: Timings of specific gameplay actions

can be used as an authentication to detect the honest user. More work done in this domain include pointer device authentication, user verification via web interfaces and recognising keystrokes. The pointer device based recognition provided a mechanism to use user interactions with a pointer device such as mouse for identification and verification. They also proposed [139], a biometric system based on user interaction with a web page. They propose integrating this biometric feature into a conventional login web page to enhance the security of their system. Most games also use keyboard as the primary input device. Biometric recognition systems based on keystroke dynamics were conducted by [140]. He outlined keyboard biometric features (events) that can be used in keystroke recognition systems, e.g. key-down, up, hold time, delay, etc.

Apart from biometrics analysis, distance measure algorithms have also been used to determine the presence of bots during games. Work has also been done to determine the behaviours of user within virtual worlds. A distance measure algorithm allows a biometric system to compare the newly submitted samples with the samples in the biometric enrolment database. A human user may divert from his pathway to investigate interesting items, whereas a bot may not. Work has also been done user behaviour on virtual worlds. [141] proposed a model to predict user's intentions to return to a virtual world. They suggested that these intentions can be determined based on a state of deep involvement that the users tend to lose the time track.

Other behavioural techniques have been used to distinguish the gamers between bots and humans. Yampolskiy [142] suggested that a behavioural biometric signature can be generated from the strategy of an individual when playing a game of poker. The generated behavioural signature is continuously compared with players' current actions and significant changes in behaviour are reported as security breaches.

Multiple work has been done on analysing a human gamer's behaviour using MLA and techniques. These techniques can be applied to detect the presence of a bot. One of the earliest work involving a bot was done by Thurau et al. [143], where they used different approaches to develop human-like AI agents using the analysis of human player behaviour. This was done by introducing a neural-network-based bot that learnt human behaviour

[144], and dividing players actions into strategies and tactics using Neural Gas Waypoint Learning algorithm to represent the virtual world [143]. They also used Bayesian imitation learning instead of Neural Gas Waypoint Learning algorithm in [145] in order to improve performance [143].

Kuan-Ta Chen et al. [5] provided a bot detection method using manifold learning. Although their method was very accurate, it was based only on the avatar's movement position and thus only worked for detecting a moving bot. Yeung et al. [146] provided a scalable method that uses a dynamic bayesian network (DBN) to detect 'AimBots' in FPS games. Their method provided good results, although if more features were used, better results could have been obtained. Galli et al. [147] used different classifiers to detect cheating in 'Unreal Tournament III'.

6.5 Problem statement and research questions

Focus has shifted more to the side of a specific cheat. The background study of the literature yielded two specific cheats that are discussed in the academia: WallHacks and MapHacks. There are several good papers published on MapHacks in very good conference e.g. S&P.

With the narrowed focus of WallHack and MapHack, the initial target is set to work on WallHack cheats. WallHacks are cheats that are employed utilising three entities: 1. Game application, 2. Graphics driver toolkit, 3. Graphics processing API, e.g. OpenGL, Direct X.

The cheat is employed in the industry via multiple mechanisms. Of these mechanisms, three ways are deemed popular and are used in the industry.

Modifying API calls OpenGL is an open source API; games employ the usage of multiple function calls within the API to draw objects. One of the easiest and most popular wall usage is used via these code patch:

```
gl_begin()
gl_Draw(gl_Quads)
```

```
...
gl_end()
```

An attacker can attack the processing of quads, where four parameters are used to draw a quadrilateral. Quadrilaterals are easier to employ in drawing of walls and this is the norm when games initiate wall drawings. The attacker can override the gl_quad calls and make them appear translucent.

There are certain drawbacks with this method, as other game objects could be used via the gl_quad calls and eventually they would start appearing translucent too.

Mesh - wireframe view model Via the usage of graphics card driver, the drivers are tweaked to not draw the full game 3D object. The graphics model is restricted in only the drawing of a wire mesh. This mesh-based drawing allows the gamer to see through walls and other objects. However, the viewing could become very messy to an end user; to help avoid this problem, specific trainer programs are created by hackers. These trainers have a switching on-off option which lets the cheater switch between environments to gain a competitive advantage.

Z- vector OpenGL draw calls are overridden via DLL (dynamic link library) file injection. When a 3D object is drawn, x-y-z components are required to draw the full position of the object. The DLL injection attack allows the z-vector to be not drawn, resulting in a false depth of objects.

The flattening of 3D objects makes it easy for cheaters to detect enemies and gain advantage.

6.6 Potential solution

Three potential good solutions have been considered to date.

Watermarking The simplest of the ideas include watermarking of the three entities that are involved in drawing a graphical object in-game: the client, graphics driver and the API. The development of the game client would bear the source of the solution.

Since the map of a game is pre-loaded and is common knowledge to the game client, the game would be able to render the scenario within itself, embedding specific watermarks. During the runtime of the game, the map or the scenario will not change. The variable entities would be the positioning of the gamers.

The solution lies in comparison. The pre-set solution would be executed by the game client and loaded. Once this step is accomplished, the client would request with specific parameters a screen cast of the current situation of the map that is drawn or displayed in the gamers' screen. The displayed screen should have mapped watermarking. Once the data is received, the watermarked images should be compared for the correct anticipated positioning of the watermarkings. If the marks match, the gamer was not cheating. If the opposite is found, the gamer could be flagged for malicious behaviour and reported.

Assumption The game client is to be trusted.

The solution is unique as the usage of trusted platform module (TPM) chip are not being used here. The usage of TPM provides static attestation and protection that the client has not been tampered with. However, this solution does not involve TPM for reasons:

> TPM chips are not widely available in all machines, so the solution does not carry any inter-dependencies.

> Despite having TPM chips, very few gamers use that chip, so by not involving TPM, we are creating a solution that can be widely adopted.

Integrity of Graphics API Since the API is open source, the codebase of the API is available. If we could attest the code and protect the execution pathways of the API, then WallHacks would be prohibited.

The method would involve having a monitor running on the client that would check

for any changes on the API during runtime. The monitor could be run in an Intel SGX environment. This would ensure runtime attestation on the API.

Graphics card driver Performance analysis The graphics card driver has to be verified for any unauthorised modification. If the graphics card driver is tweaked to not display real 3D objects, the driver would end up drawing only wire-frames. This solution needs to be applied as an addition to the previous solutions.

The graphics card driver needs to be attested prior to running the game, which would ensure the graphics driver is not tampered with. This would allow the client to display legitimate game objects in the screen.

Another method to go beyond this solution is the comparison of effective performance. The idea is to analyse the overhead in the system performance when the cheat is used and when it is not, i.e. storing or measuring the system-calls during the execution of normal world (non-cheating environment) and measuring the number of system-calls when a cheat is being used. Implementing this performance measurement would act as a solution to determine a cheating user. However, a threshold has to be taken into account, since the system may be involved in other performance-incentive tasks.

7 Conclusion

This thesis documented work on video games and focused on client-side security. We initially studied the domain of the video game and refined the area we planned to focus our work on.

The work done on this thesis details the whole MPhil journey. The initial pathway was aimed towards studying the full spectrum of cheating in video games. However, as the research started to take a longer time than expected, we limited our focus to the client side security only.

The security analysis and the findings from the game clients provide multiple opportunities that can be worked on in the near future. Amongst the choices, the impostor cheating study has the highest amount of potential. This is partly due to the fact that any good impostor cheat detection mechanism will be very useful for the gaming community. Furthermore, as discussed in the chapters, there will be fair-play re-established in the gaming community, making the gaming competitions more competitive and much more entertaining.

7.1 Achievements

This document summarises all the work undertaken throughout the journey of the MPhil degree. Analysis of the firmware and software allowed us to taxonomise the scope of firmware. The taxonomy of the firmware would hopefully allow for a greater understanding of the terminology and reduce the confusion between firmware and software. This allowed the study to be focused towards the 'software' part of the video game territory.

From the work carried out in the gaming world, we have achieved a greater understanding of the security in video games. The background analysis discussed the work done in the previous years and the types of cheats that are available and implemented. The background study of cheats over the years has helped us categorise a model of the types of cheats that are implemented in the gaming world. This modularisation of the cheat types helped us understand the scope of the problem of cheating and allowed us to isolate our problem

scope.

The idea of impostor cheating was chosen to be studied in greater depth. To accommodate this study, we have also chosen a MOBA game, Dota 2, to discuss how the cheating of impostors fits into a real-world scenario and provided possible solutions. The access to the volume of Dota 2 game data provided by Steam [92] helped us understand and make progress on the impostor cheating study.

7.2 Potential future works

Future work could lead the work done for impostor cheating to be implemented to derive results for Dota 2, applying and collecting more data. The study could involve collecting more user data and experimenting with the gaming variables. The experiments could be tweaked to enforce a recurring pattern to detect the impostor cheating gamers.

Similar methods could be extended and put to use to try on a different game other than Dota 2, but in the same genre of MOBA. Furthermore, once a pattern has been exhibited that allows a pattern to detect impostor cheaters, this can be further extrapolated to try on different genres of video game, e.g. FPS, strategy, puzzles, etc.

References

[1] R. Hassan, K. Markantonakis, and R. N. Akram, "Can you call the software in your device be firmware?," in *2016 IEEE 13th International Conference on e-Business Engineering (ICEBE)*, pp. 188–195, IEEE, 2016.

[2] H. W. Lawson, "Programming-language-oriented instruction streams," *IEEE Transactions on Computers*, vol. 100, no. 5, pp. 476–485, 1968.

[3] H. Barsamian and A. DeCegama, "Evaluation of hardware-firmware-software tradeoffs with mathematical modeling," in *Proceedings of the May 18-20, 1971, spring joint computer conference*, pp. 151–161, ACM, 1971.

[4] D. Mange, "Teaching firmware as a bridge between hardware and software," *Education, IEEE Transactions on*, vol. 36, pp. 152–157, 1993.

[5] K.-T. Chen, H.-K. K. Pao, and H.-C. Chang, "Game bot identification based on manifold learning," in *Proceedings of the 7th ACM SIGCOMM Workshop on Network and System Support for Games*, pp. 21–26, ACM, 2008.

[6] T. Eisenbarth, R. Koschke, and D. Simon, "Incremental location of combined features for large-scale programs," in *International Conference on Software Maintenance, 2002. Proceedings.*, pp. 273–282, IEEE, 2002.

[7] J. C. Knight, "Software challenges in aviation systems," 2002.

[8] S. Dutta, R. Jensen, and A. Rieckmann, "Viper: A multiprocessor soc for advanced set-top box and digital tv systems," *IEEE Design & Test of Computers*, pp. 21–31, 2001.

[9] K. Kursawe and D. Schellekens, "Flexible mutpms through disembedding.," in *Proceedings of the 2009 ACM Symposium on Information*, pp. 116–124, 01 2009.

[10] L. McMinn and J. Butts, "A firmware verification tool for programmable logic controllers," 2012.

[11] G. Ramesh and R. Umarani, "Data security in local area network based on fast encryption algorithm," 2010.

[12] D. Halperin, T. S. Heydt-Benjamin, B. Ransford, S. S. Clark, B. Defend, W. Morgan, K. Fu, T. Kohno, and W. H. Maisel, "Pacemakers and implantable cardiac defibrillators: Software radio attacks and zero-power defenses," in *2008 IEEE Symposium on Security and Privacy (sp 2008)*, pp. 129–142, IEEE, 2008.

[13] A. Bellissimo, J. Burgess, and K. Fu, "Secure software updates: Disappointments and new challenges.," in *HotSec*, 2006.

[14] W. O. Cesário, D. Lyonnard, G. Nicolescu, Y. Paviot, S. Yoo, A. A. Jerraya, L. Gauthier, and M. Diaz-Nava, "Multiprocessor soc platforms: a component-based design approach," *IEEE Design & Test of Computers*, vol. 19, no. 6, pp. 52–63, 2002.

[15] K.-D. Schubert, "Improvements in functional simulation addressing challenges in large, distributed industry projects," in *Proceedings of the 40th annual Design Automation Conference*, pp. 11–14, ACM, 2003.

[16] S. C. Jacobson, A. W. Moore, and J. M. Ramsey, "Fused quartz substrates for microchip electrophoresis," *Analytical Chemistry*, vol. 67, no. 13, pp. 2059–2063, 1995.

[17] xda devleopers, "Smart watch hack forum," 2011. https://forum.xda-developers.com/smartwatch, [Accessed: 2015-01-22].

[18] A. Cui, M. Costello, and S. J. Stolfo, "When firmware modifications attack: A case study of embedded exploitation," in *NDSS*, 2013.

[19] R. R. Plant, N. Hammond, and T. Whitehouse, "How choice of mouse may affect response timing in psychological studies," *Behavior Research Methods, Instruments, & Computers*, vol. 35, pp. 276–284, 2003.

[20] J. Maskiewicz, B. Ellis, J. Mouradian, and H. Shacham, "Mouse trap: Exploiting firmware updates in usb peripherals," in *8th USENIX Workshop on Offensive Technologies WOOT 14*, 2014.

[21] Y. Li, J. M. McCune, and A. Perrig, "Viper: verifying the integrity of peripherals' firmware," in *Proceedings of the 18th ACM conference on Computer and communications security*, pp. 3–16, ACM, 2011.

[22] Topfield, "Updating the firmware on your topfield," 2015. https://support.icetv.com.au/ [Accessed: 2019-10-19].

[23] R. Wojtczuk and A. Tereshkin, "Attacking intel bios," *BlackHat, Las Vegas, USA*, 2009.

[24] P. Gershteyn, M. Davis, and S. Shenoi, "Forensic analysis of bios chips," 2006.

[25] R. Wilkins and B. Richardson, "Uefi secure boot in modern computer security solutions," 2013.

[26] D. Cooper, W. Polk, A. Regenscheid, and M. Souppaya, "Bios protection guidelines," *NIST Special Publication*, vol. 800, p. 147, 2011.

[27] L. Zhang, S. gang Hao, J. Zheng, Y. an Tan, Q. xin Zhang, and Y. zhang Li, "Descrambling data on solid-state disks by reverse-engineering the firmware," *Digital Investigation*, vol. 12, pp. 77–87, 2015.

[28] Apple, "How to set a firmware password on a mac," 2018. https://support.apple.com/en-us/HT204455 [Accessed: 2019-10-18].

[29] iDB, "How to jailbreak ios," 2018. https://www.idownloadblog.com/jailbreak/ [Accessed: 2019-10-18].

[30] Apple, "Find the apps you love," 2019. https://www.apple.com/uk/ios/app-store/ [Accessed: 2019-10-19].

[31] TheHackerNews, "Some d-link and comba wifi routers leak their passwords in plaintext," 2019. https://thehackernews.com/2019/09/router-password-hacking.html [Accessed: 2019-09-19].

[32] Anonymous, "Firmware passwords in plaintext," 2013. http://www.deploystudio.com/Forums/viewtopic.php?id=4974 [Accessed: 2013-09-17].

[33] J. Scott, "Ee routers vulnerable to incredibly easy hack," 2014. https://www.computerweekly.com/news/2240212815/EE-routers-vulnerable-to-incredibly-easy-hack [Accessed: 2014-01-11].

[34] M. Predko and M. Predko, *PC PhD: Inside PC Interfacing*. McGraw-Hill Professional, 1999.

[35] D. Becker, R. K. Singh, and S. G. Tell, "An engineering environment for hardware/software co-simulation," in *[1992] Proceedings 29th ACM/IEEE Design Automation Conference*, pp. 129–134, IEEE, 1992.

[36] C. Bernard and F. Clermidy, "A low-power vliw processor for 3gpp-lte complex numbers processing," in *2011 Design, Automation & Test in Europe*, pp. 1–6, IEEE, 2011.

[37] Wired, "How the nsa firmware hacking works and why it is so unsettling," 2015. https://www.wired.com/2015/02/nsa-firmware-hacking/ [Accessed: 2015-10-11].

[38] I. Sutherland, G. Davies, and A. Blyth, "Malware and steganography in hard disk firmware," *Journal in computer virology*, vol. 7, pp. 215–219, 2011.

[39] C. Miller, "Battery firmware hacking," *Black Hat USA*, pp. 3–4, 2011.

[40] G. Barrett and R. Omote, "Projected-capacitive touch technology," *Information Display*, vol. 26, pp. 16–21, 2010.

[41] R. G. Ragel, S. Parameswaran, and S. M. Kia, "Micro embedded monitoring for security in application specific instruction-set processors," in *Proceedings of the 2005*

international conference on Compilers, architectures and synthesis for embedded systems, pp. 304–314, ACM, 2005.

[42] Philips, "User manual of philips sensotouch 3d," 2014.

[43] Apple, "Human interface guidelines," 2019. https://developer.apple.com/design/human-interface-guidelines/macos/user-interaction/mouse-and-trackpad/ [Accessed: 2019-10-19].

[44] S. Morrissey and T. Campbell, *IOS forensic analysis for iPhone, iPad, and iPod Touch*. Apress, 2010.

[45] A. Hoog and K. Strzempka, *iPhone and iOS forensics: Investigation, analysis and mobile security for Apple iPhone, iPad and iOS devices*. Elsevier, 2011.

[46] E. Sadun, M. Grothaus, and S. Sande, "Putting your data and media on your ipad," 2011.

[47] Hp, "Hp laserjet - update the firmware," 2019. https://support.hp.com/gb-en/document/c01711356 [Accessed: 2019-10-19].

[48] C. Seibold, *Big Book of Apple Hacks: Tips & Tools for unlocking the power of your Apple devices*. " O'Reilly Media, Inc.", 2008.

[49] S. Hanna, R. Rolles, A. Molina-Markham, P. Poosankam, J. Blocki, K. Fu, and D. Song, "Take two software updates and see me in the morning: The case for software security evaluations of medical devices.," in *HealthSec*, 2011.

[50] L. Duflot, Y.-A. Perez, and B. Morin, "What if you can't trust your network card?," in *International Workshop on Recent Advances in Intrusion Detection*, pp. 378–397, Springer, 2011.

[51] S. Stafford, *Nikon*. Sterling Publishing Company, Inc., 2008.

[52] S. Staff, "Teens put lego man in 'space' (actually stratosphere)," 2012. https://www.space.com/14397-teens-lego-man-space-stratosphere.html [Accessed: 2012-10-19].

[53] A. Aviv, P. Cerny, S. Clark, E. Cronin, G. Shah, M. Sherr, and M. Blaze, "Security evaluation of es&s voting machines and election management system," *USENIX*, 2008.

[54] A. Cui and S. J. Stolfo, "A quantitative analysis of the insecurity of embedded network devices: Results of a wide-area scan," in *Proceedings of the 26th Annual Computer Security Applications Conference*, ACSAC '10, (New York, NY, USA), pp. 97–106, ACM, 2010.

[55] O. Henniger, L. Apvrille, A. Fuchs, Y. Roudier, A. Ruddle, and B. Weyl, "Security requirements for automotive on-board networks," in *2009 9th International Conference on Intelligent Transport Systems Telecommunications,(ITST)*, pp. 641–646, IEEE, 2009.

[56] M. S. Idrees and Y. Roudier, "Computer aided design of a firmware flashing protocol for vehicular on-board networks," 2009.

[57] D. K. Nilsson and U. E. Larson, "Secure firmware updates over the air in intelligent vehicles," in *ICC Workshops-2008 IEEE International Conference on Communications Workshops*, pp. 380–384, IEEE, 2008.

[58] M. Shavit, A. Gryc, and R. Miucic, "Firmware update over the air (fota) for automotive industry," 2007.

[59] M. Guven and M. Lorang, "Automated over-the-air firmware update for a wireless phone," 9 2007.

[60] K. Markantonakis *et al.*, *Smart cards, tokens, security and applications*. Springer Science & Business Media, 2007.

[61] M. E. Soper, D. L. Prowse, and S. Mueller, *CompTIA A+ 220-701 and 220-702 Cert Guide*. Pearson Education, 2011.

[62] H. Mansor, K. Markantonakis, R. N. Akram, and K. Mayes, "Don't brick your car: Firmware confidentiality and rollback for vehicles," in *2015 10th International Conference on Availability, Reliability and Security*, pp. 139–148, IEEE, 2015.

[63] E. Vartiainen, V. Roto, and A. Popescu, "Auto-update: a concept for automatic downloading of web content to a mobile device," in *Proceedings of the 4th international conference on mobile technology, applications, and systems and the 1st international symposium on Computer human interaction in mobile technology*, pp. 683–689, ACM, 2007.

[64] O. M. Alliance, "Firmware update management object," *Open Mobile Alliance Ltd., Version*, p. 1, 2006.

[65] Google, "How to update android smartphone," 2015. https://support.google.com/android/answer/7680439?hl=en-GB [Accessed: 2012-10-15].

[66] N. Trigoni, Y. Yao, A. Demers, J. Gehrke, and R. Rajaraman, "Hybrid push-pull query processing for sensor networks," *Informatik 2004, Informatik verbindet, Band 2, Beiträge der 34. Jahrestagung der Gesellschaft für Informatik eV (GI)*, 2004.

[67] Z. Basnight, J. Butts, J. Lopez, and T. Dube, "Firmware modification attacks on programmable logic controllers," *International Journal of Critical Infrastructure Protection*, vol. 6, pp. 76–84, 2013.

[68] Apple, "Updating the software on your apple watch," 2019. https://support.apple.com/en-gb/HT204641 [Accessed: 2019-10-19].

[69] S. Biddle, "The secret history of bluetooth," 2019. https://gizmodo.com/the-secret-history-of-bluetooth-5899082 [Accessed: 2019-10-19].

[70] H. Wang, C. Shen, and U. Ritterfeld, "Enjoyment of digital games: What makes them "seriously" fun?," in *Serious Games*, pp. 47–69, Routledge, 2009.

[71] B. B. C. Timelines, "How british video games became a billion pound industry," 2015.

[72] P. A. de Catalunya, "The video games' industry is bigger than hollywood," 2018. https://lpesports.com/e-sports-news/the-video-games-industry-is-bigger-than-hollywood [Accessed: 2018-06-12].

[73] I. Grattan-Guinness, "Charles babbage as an algorithmic thinker," *IEEE Annals of the History of Computing*, no. 3, pp. 34–48, 1992.

[74] H. W. Kuhn and A. W. Tucker, "John von neumann's work in the theory of games and mathematical economics," *Bulletin of the American Mathematical Society*, vol. 64, no. 3, pp. 100–122, 1958.

[75] C. E. Shannon, "Xxii. programming a computer for playing chess," *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, vol. 41, no. 314, pp. 256–275, 1950.

[76] D. H. Ahl, K. S. Reid-Green, and A. L. Zobrist, "Computer games," 2003.

[77] S. Russell, M. Graetz, and W. Witaenem, "Spacewar," *Computer software*, 1962.

[78] M. J. Wolf, "Inventing space: Toward a taxonomy of on-and off-screen space in video games," *Film Quarterly (ARCHIVE)*, vol. 51, no. 1, p. 11, 1997.

[79] N. Bushnell, S. Bristow, and S. Wozniak, "Breakout," *A video game, Atari Inc*, 1976.

[80] G. Lastowka, "Copyright law and video games: A brief history of an interactive medium," *Available at SSRN 2321424*, 2013.

[81] J. E. Gamble, "Competition in video game consoles: Sony, microsoft, and nintendo battle for supremacy," *McGraw-Hill/Irwin=>?*, 2008.

[82] D. Weibel, B. Wissmath, S. Habegger, Y. Steiner, and R. Groner, "Playing online games against computer-vs. human-controlled opponents: Effects on presence, flow, and enjoyment," *Computers in human behavior*, vol. 24, no. 5, pp. 2274–2291, 2008.

[83] J. Jeff Yan and H.-J. Choi, "Security issues in online games," *The Electronic Library*, vol. 20, no. 2, pp. 125–133, 2002.

[84] R. Trubshaw and R. Bartle, "Mud1," *Essex, United Kingdom*, 1978.

[85] Valve, "Dota 2 players now outnumber World of Warcraft subscribers," 2016. http://www.gamespot.com/articles/dota-2-players-now-outnumber-world-of-warcraft-subscribers/1100-6419431/ [Accessed: 2016-01-20].

[86] Blizzard, "World of warcraft," 2011. http://www.worldofwarcraft.com/pvp/battlegrounds, [Accessed: 2015-03-20].

[87] Valve, "The Frankfurt Major Grand Finals," 2016. http://blog.dota2.com/2015/11/the-frankfurt-major-grand-finals/ [Accessed: 2016-05-20].

[88] D. Byte, "Cheat Engine," 2015. http://www.cheatengine.org/aboutce.php, [Accessed: 2015-09-03.

[89] A. Pedgaonkar and P. M. Bhat, "Rfid based software protection,"

[90] L. D. McMichael, B. B. Khoo, and V. J. Sabella, "Simultaneous tamper-proofing and anti-piracy protection of software," Aug. 7 2012. US Patent 8,239,967.

[91] R. M. Parizi, A. Dehghantanha, K.-K. R. Choo, M. Hammoudeh, and G. Epiphaniou, "Security in online games: Current implementations and challenges," in *Handbook of Big Data and IoT Security*, pp. 367–384, Springer, 2019.

[92] Steam, "Steam engine," 2016. https://support.steampowered.com/, [Accessed: 2016-11-25].

[93] Valve, "The International," 2016. http://www.dota2.com/international/overview/ [Accessed: 2016-07-11].

[94] A. Clements, *Principles of computer hardware*. Oxford University Press, 2006.

[95] R. Ohbuchi, H. Masuda, and M. Aono, "Watermaking three-dimensional polygonal models," in *ACM multimedia*, vol. 97, pp. 261–272, Citeseer, 1997.

[96] J. Yan and B. Randell, "A systematic classification of cheating in online games," in *Proceedings of 4th ACM SIGCOMM workshop on Network and system support for games*, pp. 1–9, ACM, 2005.

[97] J. Ki, J. H. Cheon, J.-U. Kang, and D. Kim, "Taxonomy of online game security," *The Electronic Library*, vol. 22, pp. 65–73, 2004.

[98] G. Armitage, M. Claypool, and P. Branch, *Networking and online games: understanding and engineering multiplayer Internet games*. John Wiley & Sons, 2006.

[99] M. Buro, "Orts: A hack-free rts game environment," in *International Conference on Computers and Games*, pp. 280–291, Springer, 2002.

[100] Y. S. Fung and J. C. S. Lui, *Hack-proof synchronization protocol for multi-player online games*. Springer, 2009.

[101] T. Schluessler, S. Goglin, and E. Johnson, "Is a bot at the controls?: Detecting input data attacks," in *Proceedings of the 6th ACM SIGCOMM workshop on Network and system support for games*, pp. 1–6, ACM, 2007.

[102] K.-T. Chen and L.-W. Hong, "User identification based on game-play activity patterns," in *Proceedings of the 6th ACM SIGCOMM workshop on Network and system support for games*, pp. 7–12, ACM, 2007.

[103] P. Laurens, R. F. Paige, P. J. Brooke, and H. Chivers, "A novel approach to the detection of cheating in multiplayer online games," in *12th IEEE International Conference on Engineering Complex Computer Systems (ICECCS 2007)*, pp. 97–106, IEEE, 2007.

[104] M. DeLap, B. Knutsson, H. Lu, O. Sokolsky, I. Lee, C. Tsarouchis, *et al.*, "Is runtime verification applicable to cheat detection?," in *Proceedings of 3rd ACM SIGCOMM workshop on Network and system support for games*, pp. 134–138, ACM, 2004.

[105] S. Aggarwal, J. Christofoli, S. Mukherjee, and S. Rangarajan, "Authority assignment in distributed multi-player proxy-based games," in *Proceedings of 5th ACM SIGCOMM workshop on Network and system support for games*, p. 5, ACM, 2006.

[106] C. Chambers, W.-c. Feng, and W.-c. Feng, "Towards public server mmos," in *Proceedings of 5th ACM SIGCOMM workshop on Network and system support for games*, p. 3, ACM, 2006.

[107] P. Kabus, W. W. Terpstra, M. Cilia, and A. P. Buchmann, "Addressing cheating in distributed mmogs," in *Proceedings of 4th ACM SIGCOMM workshop on Network and system support for games*, pp. 1–6, ACM, 2005.

[108] Blizzard, "Warden wiki," 2013. http://www.worldofwarcraft.com/, [Accessed: 2015-01-22].

[109] G. Hoglund, "Hacking world of warcraft: An exercise in advanced rootkit design," *Black Hat*, 2006.

[110] W.-c. Feng, E. Kaiser, and T. Schluessler, "Stealth measurements for cheat detection in on-line games," in *Proceedings of the 7th ACM SIGCOMM Workshop on Network and System Support for Games*, pp. 15–20, ACM, 2008.

[111] S. De Paoli and A. Kerr, "The cheating assemblage in mmorpgs: Toward a sociotechnical description of cheating," *Breaking New Ground: Innovation in Games, Play, Practice and Theory. Proceedings of DiGRA 2009*, pp. 1–12, 2009.

[112] G. Hoglund, "Hacking world of warcraft: An exercise in advanced rootkit design," *Black Hat*, 2006.

[113] M. K. Reiter and A. D. Rubin, "Crowds: Anonymity for web transactions," *ACM Transactions on Information and System Security (TISSEC)*, vol. 1, pp. 66–92, 1998.

[114] M. J. Freedman and R. Morris, "Tarzan: A peer-to-peer anonymizing network layer," in *Proceedings of the 9th ACM conference on Computer and communications security*, pp. 193–206, ACM, 2002.

[115] D. Kraft, "Game channels for trustless off-chain interactions in decentralized virtual worlds," *Ledger*, vol. 1, pp. 84–98, 2016.

[116] T. C. Smith and J. Miles, "Continuous and reinforcement learning methods for first-person shooter games," *GSTF Journal on Computing (JoC)*, vol. 1, 2014.

[117] K. Balci and A. A. Salah, "Automatic analysis and identification of verbal aggression and abusive behaviors for online social games," *Computers in Human Behavior*, vol. 53, pp. 517–526, 2015.

[118] N. Alshurafa, J.-A. Eastwood, M. Pourhomayoun, S. Nyamathi, L. Bao, B. Mortazavi, and M. Sarrafzadeh, "Anti-cheating: Detecting self-inflicted and impersonator cheaters for remote health monitoring systems with wearable sensors," in *2014 11th International Conference on Wearable and Implantable Body Sensor Networks*, pp. 92–97, IEEE, 2014.

[119] M. Stevanovic and J. M. Pedersen, "An efficient flow-based botnet detection using supervised machine learning," in *2014 international conference on computing, networking and communications (ICNC)*, pp. 797–801, IEEE, 2014.

[120] H. Alayed, F. Frangoudes, and C. Neuman, "Behavioral-based cheating detection in online first person shooters using machine learning techniques," in *2013 IEEE Conference on Computational Inteligence in Games (CIG)*, pp. 1–8, IEEE, 2013.

[121] A. R. Kang, J. Woo, J. Park, and H. K. Kim, "Online game bot detection based on party-play log analysis," *Computers & Mathematics with Applications*, vol. 65, pp. 1384–1395, 2013.

[122] E. Bursztein, M. Hamburg, J. Lagarenne, and D. Boneh, "Openconflict: Preventing real time map hacks in online games," in *2011 IEEE Symposium on Security and Privacy*, pp. 506–520, IEEE, 2011.

[123] Steam, "Valve anti-cheating ," 2016. https://support.steampowered.com/kb_article.php?ref=7849-RADZ-6869, [Accessed: 2016-11-25].

[124] W. b. hackmag, "Hackmag," 2015. https://hackmag.com/uncategorized/deceiving-blizzard-warden/ [Accessed: 2015-03-13].

[125] E. Balance, "Punkbuster ," 2000. http://evenbalance.com/faq.php [Accessed: 2016-12-15].

[126] A. Yahyavi and B. Kemme, "Peer-to-peer architectures for massively multiplayer online games: A survey," *ACM Computing Surveys (CSUR)*, vol. 46, p. 9, 2013.

[127] B. B. Yang and H. Garcia-Molina, "Designing a super-peer network," in *Proceedings 19th International Conference on Data Engineering (Cat. No. 03CH37405)*, pp. 49–60, IEEE, 2003.

[128] A. Yahyavi, K. Huguenin, J. Gascon-Samson, J. Kienzle, and B. Kemme, "Watchmen: Scalable cheat-resistant support for distributed multi-player online games," in *Distributed Computing Systems (ICDCS), 2013 IEEE 33rd International Conference on*, pp. 134–144, IEEE, 2013.

[129] Valve, "Matchmaking - Dota 2," 2016. http://blog.dota2.com/2013/12/matchmaking/ [Accessed: 2016-10-02].

[130] Valve, "The international," 2016. https://liquipedia.net/dota2/The_International/2016 [Accessed: 2016-01-20].

[131] Valve, "The international - 2011 dota 2 champions," 2011. http://www.dota2.com/international/overview/ [Accessed: 2017-09-10].

[132] Valve, "The frankfurt major grand finals," 2016. https://liquipedia.net/dota2/Frankfurt_Major/2015 [Accessed: 2016-03-11].

[133] A. Esanu, "Virtus.Pro admit they've cheated in the Summit 5 Grand Finals qualifiers ," 2016. http://www.gosugamers.net/dota2/news/35692-virtus-pro-admit-they-ve-cheated-in-the-summit-5-grand-finals-qualifiers, [Accessed: 2016-11-25].

[134] Valve, "Manila Major 2016," 2016. http://dota2.gamepedia.com/Manila_Major_2016, [Accessed: 2016-08-23] .

[135] Valve, "The International - 2011 Dota 2 Champions," 2011. http://cdn.dota2.com/apps/dota2/international2011_static/index.html, [Accessed: 2016-03-20].

[136] A. Al-Khazzar and N. Savage, "Biometric identification using user interactions with virtual worlds," in *2011IEEE 10th International Conference on Trust, Security and Privacy in Computing and Communications*, pp. 517–524, IEEE, 2011.

[137] A. K. Jain, "Biometric recognition: overview and recent advances," in *Iberoamerican Congress on Pattern Recognition*, pp. 13–19, Springer, 2007.

[138] D. Maltoni, D. Maio, A. Jain, and S. Prabhakar, *Handbook of fingerprint recognition*. Springer Science & Business Media, 2009.

[139] H. Gamboa, A. Fred, and A. Jain, "Webbiometrics: User verification via web interaction," in *2007 Biometrics Symposium*, pp. 1–6, IEEE, 2007.

[140] R. D. Labati, A. Genovese, V. Piuri, and F. Scotti, "Wildfire smoke detection using computational intelligence techniques enhanced with synthetic smoke plume generation," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 43, no. 4, pp. 1003–1012, 2013.

[141] L. Goel, N. A. Johnson, I. Junglas, and B. Ives, "From space to place: predicting users' intentions to return to virtual worlds," *MIS quarterly*, vol. 35, pp. 749–772, 2011.

[142] A. A. Mohamed and R. V. Yampolskiy, "Wavelet-based multiscale adaptive lbp with directional statistical features for recognizing artificial faces," *ISRN Machine Vision*, vol. 2012, 2012.

[143] C. Thurau, C. Bauckhage, and G. Sagerer, "Learning human-like movement behavior for computer games," *From Animals to Animats*, vol. 8, pp. 315–323, 2004.

[144] C. Thurau, C. Bauckhage, and G. Sagerer, "Combining self organizing maps and multilayer perceptrons to learn bot-behaviour for a commercial game.," in *GAME-ON*, p. 119, Citeseer, 2003.

[145] C. Thurau, T. Paczian, and C. Bauckhage, "Is bayesian imitation learning the route to believable gamebots," *Proc. GAME-ON North America*, pp. 3–9, 2005.

[146] S. F. Yeung and J. C. S. Lui, "Dynamic bayesian approach for detecting cheats in multiplayer online games," *Multimedia Systems*, vol. 14, pp. 221–236, 2008.

[147] L. Galli, D. Loiacono, L. Cardamone, and P. L. Lanzi, "A cheating detection framework for unreal tournament iii: A machine learning approach," in *2011 IEEE Conference on Computational Intelligence and Games (CIG'11)*, pp. 266–272, IEEE, 2011.