Cheating and Virtual Crimes in Massively Multiplayer Online Games

Rahul Joshi

Technical Report
RHUL-MA-2008-06
15 January 2008

Royal Holloway
University of London

Department of Mathematics
Roal Holloway, University of London
Egham, Surrey TW20 0EX, England
`http://www.rhul.ac.uk/mathematics/techreports`

Cheating and Virtual Crimes in Massively Multiplayer Online Games

Rahul Joshi

Supervisor: Andreas Fuchsberger

Submitted as part of the requirements for the award of the MSc in Information Security at Royal Holloway College, University of London.

I declare that this assignment is all my own work and that I have acknowledged all quotations from the published and unpublished works of other people. I declare that I have also read the statements on plagiarism in Section 1 of the Regulations Governing Examination and Assessment Offences and in accordance with it I submit this project report as my own work.

Table of contents

1 Executive Summary

Massively Multiplayer Online Games (MMOG) have become extremely popular since the birth of the Internet, with many millions of players playing games such as Poker and World of Warcraft. However, they do not seem to be well understood, and academic research into them has been limited. This project explains the nature of MMOG, and the relationship between MMOG and information security. This project discusses the problem of cheating in MMOG – it explains what cheating is, how it occurs, and how information security can be used to prevent it. The nature of virtual economies in MMOG is discussed, and the virtual crimes that have affected MMOG along with preventative measures are examined.

2 Introduction

The growth of the Internet over the last decade has led to the emergence of a multi billion dollar online gaming industry. A huge variety of games can be played on the Internet: chess, first person shooters, and casino games, such as poker and blackjack. Aside from casino games, Massively Multiplayer Online Games (**MMOG**) have become a driving force in the rapid growth of the online gaming industry. MMOG are games that are hosted on servers and played by hundreds or thousands of players using the Internet. MMOG allow players to collaborate with, or pit their wits against, thousands of other players located across the globe. The use of the Internet to play games has added an extra dimension to gameplay. Huge online gaming communities have emerged from MMOG, where players can interact with each other and form friendships.

Research has shown that being part of such a community forms a major part of the enjoyment for some players [HN06]. There are a number of different MMOG genres, for example first person shooters such as World War II Online: Battleground Europe (www.battlegroundeurope.com) or real time strategy games such as ShatteredGalaxy (www.sgalaxy.com). By far the most popular genre is **MMORPG** (Massively Multiplayer Online Role Playing Games). Blizzard Entertainment, publishers of the popular **World of Warcraft** (WoW) game, claim to have over 8 million subscribers [Bl07].

MMORPG involve players adopting the role of a fictional character who resides in a virtual fantasy world, designed and hosted by the game provider. This world exists and evolves on a permanent basis, even when the player is away from the game. By controlling the actions of their character and roaming the fantasy world, players earn money and develop skills through the trading of items or completion of certain tasks, such as the killing of opponents. Certain tasks may only be completed via co-operation with other players, which serves to develop the community spirit. As touched on earlier, most MMORPG use a client-server architecture. The software that generates and maintains the virtual world is written by the game publisher and runs continuously on a server. Players connect to the server via client software provided to them in return for a fee. Some MMORPG, such as WoW require payment of a monthly subscription to play.

This revenue stream helps publishers cover the maintenance and hosting costs of such a large scale operation.

2.1 Project objectives

The objectives of this project are to:

- Examine the problems of cheating and virtual crimes in MMOG, paying particular attention to MMORPG
- Describe the various forms that cheating and virtual crimes take, and examine their implications
- Show how information security can be used to prevent cheating and virtual crimes
- Analyse and evaluate real-world approaches taken by game publishers to these problems

Although many people see them as just games, with little impact on the real world, I believe this attitude to be misguided. Items earned through gameplay, such as weapons, are bought for hard currency through websites, in order for players to increase the power of their characters, and thus progress through the game at a faster pace. Trading of game items for real money has led to criminal gangs seeking to exploit MMORPG and the players who play them. Such interlinking between the virtual world of an MMORPG and the real world is a feature not seen before in computer games, and is unique to MMORPG.

The academic research into the relationships between MMOG and information security has been limited thus far. With increasing numbers of people playing MMOG, and the number of threats growing, it is important to analyse the security concerns that they present.

3 How an MMORPG works

WoW is the most popular MMORPG, with its makers, Blizzard, claiming to have 8 million subscribers worldwide. A look into how the game works will be useful for future discussions. To start playing the game, players must purchase the game software, and install this on their computer. Using the unique key provided with the installation CD, an online game account is created. Players must also pay a monthly subscription fee to continue playing. Successful completion of this phase allows a player to create a character and choose a faction to whom the character will belong. Two factions exist in World of Warcraft, and the purpose of joining a faction is to "make you feel like a member of one enormous team, while at the same time setting up the other faction as an enemy" [Wa07].

The character chosen resides within a virtual gameworld. Players explore the virtual world's landscape, where opponents such as monsters are encountered and have to be fought. Players move through the game's levels by gaining experience. Experience is gathered by killing opponents, exploring new locations and completing "quests". A quest is a special task, such as the delivering of an item to a given person. Quests are given to players by computer characters (also know as non-player characters). Completion of quests results in rewards, such as money, potions or weapons. Some quests are designed to require co-operation with other players in order to be completed.

A large component of WoW and other MMORPG is the player community. As the game has thousands of players, features have been created to enable in-game socialising. These include creating friends lists and online chat facilities. This enables players to team up with others to complete quests. The result of these in-game socialising features has led to the development of a strong player community and spirit within WoW as well as many other MMORPG.

Screenshot of the popular WoW MMORPG

You dance with Keeg Gibn.
You dance with Keeg Gibn.
You dance with Keeg Gibn.
[1. General] Beryl: get down tonight?
Dunmore says: join in folks
[1. General] FourInchTool: Considering we're dwarves, little is right.
Damador bursts into dance.
Dunmore bursts into dance.

TigerLibra gains Devotion Aura.
BeoWolven gains Devotion Aura.
Devotion Aura fades from BeoWolven.
Devotion Aura fades from BaleogFierce.
Devotion Aura fades from TigerLibra.
Devotion Aura fades from Erebuni.
Devotion Aura fades from Brunhilde.
Devotion Aura fades from BaleogFierce

Damador
Dwarf Paladin
Level 8

3.1 MMOG Architecture

Typically, MMOG use a client-server architecture. The clients are the player's machines. They provide an interface to players in order to play the game, and pass commands issued by the player to the server. The server is owned by the game publisher and is responsible for receiving these commands, updating the game state according to these commands, and sending these updates to clients. The server must track and update all entities in the game constantly; entities are dynamic objects like monsters. This is a hugely computationally intensive task, with so many players constantly changing entities. To provide a view of the game to the client, the client possesses a graphical engine which uses a cache of the game state stored on the client and any updates it receives from the server to render a graphical representation of the virtual world.

With the number of subscribers to WoW, it is impossible for everyone to play together in the same place in the virtual world. To ensure players do not play in the same place and therefore overload the server, causing lags in gameplay, "realms" are created by the game publishers. Realms are clusters of servers in which a few thousand players play the game. Players are tied to a realm, and cannot interact with players in other realms. Realms are exact copies of the game world, and support different types of gameplay. For example, some realms support Player versus Player gameplay (where players can be attacked and attack others), and others support Player versus Environment, where players do not play against other human players, just against the virtual gameworld.

4 Cheating in Online Games

4.1 The problem of cheating

A serious issue facing the developers of MMORPG is the issue of player cheating. Cheating by players giving them an unfair advantage over others results in the degradation of the gaming experience for legitimate players. This problem is exacerbated if no adequate methods are found to detect and sanction cheating players. This will encourage more players to cheat as the chance of being detected or punished is negligible. Evidence has shown these legitimate players will quit the game, thus reducing the numbers of subscribers and the publisher's revenues, and will affect the long term reputation of the game. This could have serious ramifications for the industry as a whole. Estimates have put industry revenues at $8bn a year, and growing. Cheating represents a serious threat to this. If cheating becomes rampant in MMOG, the reduction in player numbers will affect the whole industry. The detection and prevention of cheating thus represents a serious challenge to game publishers.

Comment [R1]: Facts needed on nos of players who quit cos of cheating

4.2 Defining cheating

Defining what is meant by the term cheating when applied to MMOG is an important and somewhat unclear issue. It is vital for game publishers to provide a precise definition of the term in their service agreements to their subscribers, to ensure that both sides are clear on what constitutes illegal behaviour, and so disputes cannot occur later. However the definition will vary between games and their publishers. For example, some games allow play to be scripted (by using a macro to repeat a keyboard sequence for example) whilst others forbid this practice entirely. Another complication of providing a precise definition is that it is sometimes extremely hard to tell the difference between a player who is cheating and a player who is using clever tactics or possesses exceptional ability. Yan [Ya02] demonstrates the difficulty of this distinction by using the example of "camping" behaviour in first person shooter games. Camping involves a player hiding in a particular location (such as a hole or a dark corner) in the game waiting for other players to pass by, and then kills them. The location in which they hide is often hard for other players to spot at first glance, and is very hard (if not impossible) for the camper to be killed by other players. Camping spoils the game for many players, and as such is regarded by some as cheating. However, as Yan points out, in war games, hiding in locations to kill opponents is simply a simulation of real combat, and so camping can be considered legitimate behaviour. This serves to underline the point that the definition of cheating varies between games, and even between players. Due to the problems discussed above, there is not a universally accepted definition of cheating when applied to MMOG.

Yan [Ya05] believes cheating is "any behaviour that a player uses to gain an advantage or achieve a target in an online game is cheating if, according to the game rules or at the discretion of the game operator the advantage is unfair to his peer players or the target is one that he is not supposed to achieve".

I believe this is a sufficient definition as it covers situations where a player may not have cheated another player, but instead has cheated his way up a leaderboard, or has used a cheating exploit to defeat a computer controlled opponent. It also encompasses the fact that the definition of cheating varies between games. I will use Yan's definition of cheating in my subsequent discussions.

4.3 Motivations for cheating

Traditionally, cheats in computer games were placed in the game code by the game's developers to reward players. They could be activated by entering a special code whilst the game was running. They would give players privileges such as moving them up some levels, giving them unlimited lives or inflating their statistics in order to make them stronger.

Developers of MMORPG are careful not to include cheat codes in the game code. This is because players who find the code will gain a significant advantage over those who don't. It can also leave the developers open to accusations of cheating themselves – if they play the game and know (and use) the cheat code then this is clearly unfair on other players. Instead, cheating has taken a different form in MMORPG. The primary motivation for cheating in MMORPG stems from the concept of virtual economies that exist in these games, such as Entropia Universe and EverQuest. Items earned through gameplay, such as weapons and gold, are sold for real money on websites. For example, the currency in Entropia Universe, Project Entropia Dollars (PED), could be bought and redeemed for real-world money at a rate of 10 PED for $1 (US).

Online marketplaces such as IGE (www.ige.com) exist in which players can buy and sell gold for a number of different MMORPG. This may seem strange: how can items which don't exist in the real world be bought for money? However, this concept may be easier to understand when you consider the fact that these items are earned over many hours of play. Other players, who wish to possess these items but do not want to invest the time, are willing to pay money to own them. Virtual economies have added an extra dimension to gameplay – money can be earnt through playing MMORPG. This financial reward is a powerful incentive for players to look for ways to cheat to accumulate virtual property which they can then sell on. I believe this is the principal driver behind cheating in MMORPG.

Another reason why some players cheat is to cause "grief" to other players. These types of players are known unsurprisingly as "griefers". Their sole purpose in playing the game is to harass other players. They claim they do so in order to make the game interesting after completing the game themselves. Others "grief" simply to cause misery to other players. Some "griefing" behaviour is within the game rules (such as insulting players using the online chat facility), but griefers also exploit cheating exploits, for example, to steal player's virtual items.

4.4 Types of cheating

There are many different ways of cheating in online games. Some methods only occur in MMORPG, others only in other online games such as poker or chess. Yan, who has written several papers on cheating in online games, has classified cheating into 15 different categories [Ya05]. This was a thorough description and encompassed all the types of cheating I could find whilst conducting my research. This section will explain his definitions, with examples I have found of each type, as this will provide an excellent foundation for understanding the diversity of cheating.

Yan defines the following categories:

1. Cheating by Exploiting Misplaced trust

The game client supplied to the player in order to play the game performs a number of functions - it passes commands to the server, receives commands from the server and contains a graphical engine to render a graphical representation of the game on the player's screen. The game client is a very attractive target for a cheater who has the ability to modify the client using a disassembler. It can be reverse engineered passively, as it is in the cheater's possession. One of the ways in which a client has been modified is to perform cheats known as wallhacks and maphacks. Maphacks typically occur in real time strategy games. In these games, a player is given a map that shows him the areas/locations he controls, and no others. A wallhack is the first person shooter version of a maphack. This cheat allows players to see through walls – a huge advantage in such a game. Servers send information to the clients, and it is up to the client program to hide information that should not be seen by the player. Maphacks and wallhacks can be performed by modifying the graphical engine in the game client to initialise map control values differently, thus allowing a player to see parts of the map he shouldn't. A detailed explanation of how map and wall hacks are achieved is discussed later.

Comment [R2]: Move to MMOG cheating section

[Ya05] believes that this form of cheating is due to misplaced trust – too much trust is placed on the client side, and because of the possibility that the client may be a cheater, this trust is misplaced.

2. Cheating by collusion

This form of cheating occurs between players who combine to attempt to gain an unfair advantage over others. An example of this was Blizzard's StarCraft game. Two players would take turns to win and lose to each other. As points would be awarded for a win, the players would be able to advance up the leaderboard. Both players would thus be able to climb up the leaderboard, event though the results of each game had been pre-determined.

3. Cheating by Abusing the Game procedure

Flaws in a game's operating procedures or rules can also be exploited by cheaters. [Ya05] states that this type of cheating requires no technical sophistication. An example of this is escaping: a cheater disconnects himself from the game when he is about to lose. This behaviour is common in games where wins and losses are recorded on a player's account. To prevent the loss from being recorded, a player will disconnect from the game. This type of cheating was noticed on Battle.Net, an online gaming service provided by Blizzard which allows players to initiate multiplayer games. Battle.Net records the wins and loss of every game in Warcraft 3. Hackers took advantage of this fact and used a disconnect hack to disconnect from the server legally (within the rules set by Battle.Net and Warcraft 3) without having Battle.Net record the loss. Recently, Blizzard has deleted the accounts of all players using this cheat.

4. Cheating relating to virtual assets

When a player purchases a virtual item from a site such as IGE, the player selling the item agrees to meet the buyer at a pre-defined location in the game. This form of cheating can occur when the seller never turns up to deliver the item to the buyer.

5. Cheating by exploiting machine intelligence

This type of cheating relates to the ability of a player to use artificial intelligence in some online games. For example, in a game of chess against a human opponent, a cheater may decide his next move by simulating the game on a computer chess program and then using the chess machine's artificial intelligence to choose his next move.

6. Cheating by modifying client infrastructure

In this type of cheating, there is no modification of game programs, or data on the client side. Instead, client infrastructure, such as device drivers or the operating system, is tampered with. For example, one way to perform a maphack is to modify the computer's graphics card driver. By doing this, it is possible for a cheater to make walls transparent and gain an advantage over other players.

7. Cheating by denying service to peer players

A cheater may delay the responses from an opponent in a real time game by flooding his network connection.

8. Timing cheating

An example of timing cheating is the "look-ahead" cheat. A cheating player can delay his own move until he knows all his opponents' moves, thereby gaining a big advantage of his opponents.

9. Cheating by compromising passwords

By stealing another player's password a cheater can gain full access to that player's character in the game. This gives him possession of that character's virtual assets, which he can choose to sell. A large scale case of password stealing occurred in China where 44 people were arrested after stealing almost $90,000 of virtual items from players whose passwords they had stolen. [Pl06]

10. Cheating by exploiting lack of secrecy

Packet sniffing programs allow users to examine, modify, send or block packets to and from their computers. Such programs, such as tcpdump, can be used by a cheater to their advantage. Packets which affect a player negatively, such as a loss of life, may be blocked. Packets can be replayed (such as the shooting of an enemy player, especially when ammunition is low).

11. Cheating by exploiting a lack of authentication

Massively multiplayer **First Person Shooter** (FPS) games such as Quake allow players to host the game on their own servers, which can be customised according to how they want the game to be played. Some of these player-hosted servers allow any players to connect to them (some of these servers are just created for the hoster's friends to connect to). Player hosted servers give rise to the possibility that a cheater could create his own game server and harvest valid user's login credentials. This is more powerful than the password stealing form of cheating, as the cheater can harvest many users details in one go

12. Cheating by exploiting a bug or loophole

This type of cheating involves exploiting a bug or loophole in the game. There is no modification of client code or data. An example of such a bug occurred in the game Habitat. A programming error meant that players were able to sell virtual items to a pawn shop for more than they paid to buy them from a vending machine. By purchasing from the vending machine and selling in the pawn shop, some players became overnight millionaires!

13. Cheating by compromising game servers

Game server programs and their configurations can be altered by a cheater to their advantage, for example to give certain players, such as the server administrators friends, an advantage over others.

14. Cheating related to Internal misuse

Game server administrators and other company insiders have access to servers and databases in which the game and its data is stored. This gives them power to modify this data. For example, according to [AP06], in a case in China in 2006, three employees from Shanda Interactive, who supply the MMOG Legend of Mir II, were found guilty of embezzlement relating to virtual assets. An employee at the company who was responsible for maintaining and updating the game database created a large number of copies of some rare game items. These rare game items were then sold to players for a huge profit. Although they were caught and players who bought the items of them had them confiscated, this example shows how insiders are able to use their position to cheat or help others to cheat.

15. Cheating by social engineering

Cheaters use “phishing” techniques, such as sending forged emails pretending to be from the game provider, asking players for their account details. Many game providers provide detailed guidelines to their players to ensure they are not duped by such tricks.

Yan’s fifteen categories represent a thorough description of the types of cheating that exist in all genres of online games. They are useful in order to show how diverse the methods used to cheat can be. The grouping of cheats into categories helps in a number of ways. It helps to provide a systematic and structured way to look at online cheating. By doing this it can enable information security experts who may have a limited knowledge of online gaming to understand the threats posed by cheating, and thus enable them to propose countermeasures to cheating.

5 How cheats work

5.1 Client side cheating

For players to have a realistic and enjoyable gaming experience, any actions a player performs should appear seamless. There should not be a lag or delay when a player issues a command – when a player issues a command, they should see the results of the action immediately. If there was a delay between issuing a command and seeing the result this would lead to a poor gaming experience. For lags not to occur would require the server to take the client's commands, process them, update the game state accordingly and transfer the new state to the player instantaneously. Unfortunately, this is not possible. The reason is that the Internet is used as the communications medium. The Internet is not designed to allow the game state to be transferred to all clients in parallel. [FC05] studied the traffic that is generated by online games, such as Quake and Half-Life. They found that game traffic consists of large, periodic bursts of short packets, which they state is fundamental in all highly interactive online games due to their nature. They state that "short packets are required for low latency, while highly periodic traffic allows for the game to provide uniform interactivity amongst all of its clients. Unfortunately for games, network devices are not necessarily designed for this type of traffic."

To overcome this problem, the client must share some of the work with the server. This is accomplished by allowing the client software to store and manage some of the game's state. Allowing the game client to store some of the game state means that it is a very attractive target for cheaters. Typically, the client stores information about the landscape and map, as well as the location of opponents who may not currently be visible to the player.

Hacking the game client

By reverse engineering the game client, a cheater is able to gain access to the data it hides from the player. This data may contain locations of other players or items that the player cannot yet see. Possessing this information gives a cheater a significant advantage over other players. There are a number of tools available to a cheater to help him compromise the game client.

Decompilers

Decompilers allow binary code to be decompiled into human readable source code. By decompiling the game client executable file, an attacker can see the source code, variable names and programmer comments. Knowing this information can give an attacker an insight into how the code works. Decompilers make the code a lot easier to read and understand. Decompilers are not 100% accurate in the source code they generate, but they can help significantly.

Disassemblers

A disassembler is used to transform binary code into assembly language. Assembly language can be understood more easily than binary code, and by tracing through the disassembled code allows a cheater to understand how the program works.

Once the attacker understands how the code works, he can make changes to the values of important variables which could give him more ammunition, for example. He may also be able to read the values of variables designed to be hidden – thus leaking sensitive information that is used to provide an advantage over other players.

Debuggers

Debuggers are an extremely useful tool – they allow a cheater to step through the game client code whilst the program is running. This allows them to determine the state of the program and its variables during program execution. Values of variables can also be changed during execution, and breakpoints can be inserted into the code to allow execution to stop at the breakpoint. Using the variety of tools a debugger contains can allow a cheater to discover the inner workings of the game client.

DLL Injection

A Dynamic Link Library (DLL) is Microsoft's implementation of the shared library concept, and is called by an executable program at run time when it needs to access library functions. DLL injection is a mechanism for introducing new code into an executable, and is a common way for attackers to get game clients to execute code of their choice, thus allowing them to develop and run cheating exploits. There are two types of DLL injection: static and dynamic. Static DLL injection occurs before any code is executed, when an attacker inserts a jump instruction in a file that points to the address of his code that he wants to be executed. Dynamic DLL injection occurs after a program has been executed. When a program has been executed, a process is created in the operating system. Dynamic injection occurs when an attacker attempts to load his code into the process' memory space.

5.2 Data modification

Client data modification

Any data that is sent to the game client by the server is susceptible to access or modification by a cheater. Much of this data is designed to be hidden from the player by the game client software. Game client software displays information to the player through the user interface. The user interface only displays to the player information they are authorised to see, which is only a fraction of the information that the client actually possesses.

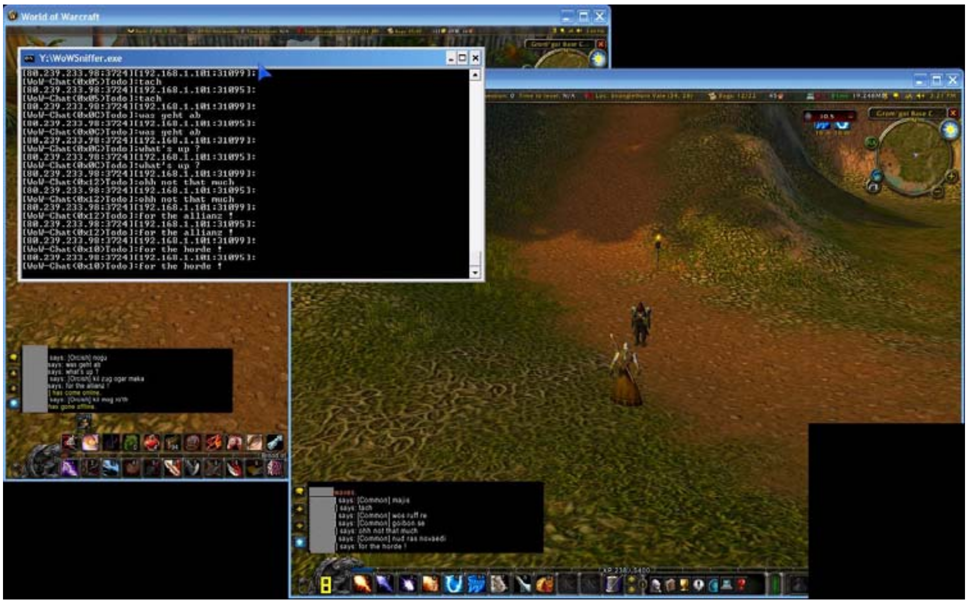
McGraw and Hoglund [MH2007] use an example of a magic potion, which can be drunk by a player to give them extra powers. If this potion's strength is more than 100, the player receives an extra bonus. The code to check if the potion's strength is greater than 100 is in the client software, and so is the variable that stores the potion's strength. Therefore, these are liable to modification by a cheater. By using a debugger, when the program is running, the variable that contains the potion's strength can be changed to be greater than 100, and a bonus gained. Finding the location within the program of where this variable is stored is not a trivial task. However skilled use of disassemblers and debuggers simplify the task considerably.

Network data

To communicate with the game client, the game server sends packets of data to the client across a network connection, and vice versa. As this data passes along the Internet, there is no guarantee of the route it takes, leaving the data vulnerable to sniffing and modification. Cheaters can examine these packets, and packets which may harm the cheater can be blocked (such as being attacked by an opponent), whilst others which have positive effects (such as the drinking of a magic potion) can be replayed. This interferes with the integrity of these packets, and hence the game itself.

An example of packet sniffing in practice is WoWSniffer. This program has been written for WoW which enables a cheater to sniff chat messages over the network. Chat messages are a WoW "instant messaging" service allowing players to communicate with each other whilst playing. WoWSniffer allows messages exchanged between rival players to be read. This can lead to vital information being exposed to a cheater.

Screenshot of WoWSniffer



```
[80.239.233.98:3724][192.168.1.101:31099]:
[WoW-Chat(0x05)Todo]:tach
[80.239.233.98:3724][192.168.1.101:31095]:
[WoW-Chat(0x05)Todo]:tach
[80.239.233.98:3724][192.168.1.101:31099]:
[WoW-Chat(0x0C)Todo]:was geht ab
[80.239.233.98:3724][192.168.1.101:31095]:
[WoW-Chat(0x0C)Todo]:was geht ab
[80.239.233.98:3724][192.168.1.101:31099]:
[WoW-Chat(0x0C)Todo]:what's up ?
[80.239.233.98:3724][192.168.1.101:31095]:
[WoW-Chat(0x0C)Todo]:what's up ?
[80.239.233.98:3724][192.168.1.101:31099]:
[WoW-Chat(0x12)Todo]:ohh not that much
[80.239.233.98:3724][192.168.1.101:31095]:
[WoW-Chat(0x12)Todo]:ohh not that much
[80.239.233.98:3724][192.168.1.101:31099]:
[WoW-Chat(0x12)Todo]:for the allianz !
[80.239.233.98:3724][192.168.1.101:31095]:
[WoW-Chat(0x12)Todo]:for the allianz !
[80.239.233.98:3724][192.168.1.101:31099]:
[WoW-Chat(0x10)Todo]:for the horde !
[80.239.233.98:3724][192.168.1.101:31095]:
[WoW-Chat(0x10)Todo]:for the horde !
```

[redacted] says: [Orcish] nogu
says: was geht ab
says: what's up ?
says: [Orcish] ki zag ogar maka
says: for the allianz !
] has come online.
says: [Orcish] ki mog ro'th
has gone offline.

[redacted] waves.
says: [Common] majis
] says: tach
says: [Common] wos ruff re
says: [Common] goibon se
] says: ohh not that much
says: [Common] nud ras novaedi
] says: for the horde !

Hardware data modification

High quality graphics are a vital part of any MMOG. They provide for a quality gaming experience and a colourful and realistic looking virtual world. Many MMOG use the OpenGL specification to create their graphics. OpenGL is a specification for a cross platform, cross-language API for writing applications that produce 3-D computer graphics. OpenGL is implemented on a graphics card by a driver, and hence the rendering of graphics is done on the graphics card itself. Games instruct the driver what to draw on the screen such as walls, and their location.

Hacking OpenGL to create an aimbots

Aimbots are cheats that allow players to automatically aim and fire their weapon at opponents. This is regarded as cheating as it gives a player superhuman killing abilities. Aimbots work by detecting the 3-D coordinates of an opponent, then automatically calculating where and at what angle the weapon should be positioned in order to take the best possible shot at an opponent. As the game instructs the OpenGL video library on the graphics card what to draw on the screen, the 3-D coordinates of an opponent can be intercepted from the communication between the game and the OpenGL library, and can be used to create an aimbot.

API level hacking

OpenGL and Direct3D are publicly available specifications that define the format which graphical data must be presented to the graphics card for rendering. They define API's that developers must use for rendering in game graphics. Several forms of cheating can be accomplished by replacing a DLL which deals with graphical rendering in the game client with a Trojanised version. This Trojan DLL has the ability to intercept and alter any rendering function call. This can be used to make enemies appear in fluorescent colours, for example. This was the method used by the notorious XQZ cheat for Counter Strike. This cheat was an aimbot, with an optional wallhack feature. It relied on replacing the OpenGL DLL file with a hacked version, so that the rendering of any objects on the screen could be modified, either to make them transparent for use in a wallhack, or to find their coordinates which could then be used by an aimbot.

Hardware level hacking

It is the job of the graphics card itself to convert the data it receives into objects on the display. To do this, the graphics card is supplied with the locations of all objects that need to be rendered. As mentioned before, this data conforms to a standard such as OpenGL or Direct3D, as detailed above. However, the graphics cards themselves possess a video hardware standard. For example, all graphics cards produced by NVIDIA use their own documented format for video information. This means that at some point in the graphics rendering process, there will be data that will be in a known, documented format which is used by the graphics card. Getting at this data will allow a cheater to find out coordinates of objects, allowing him to aim and kill them with 100% accuracy.

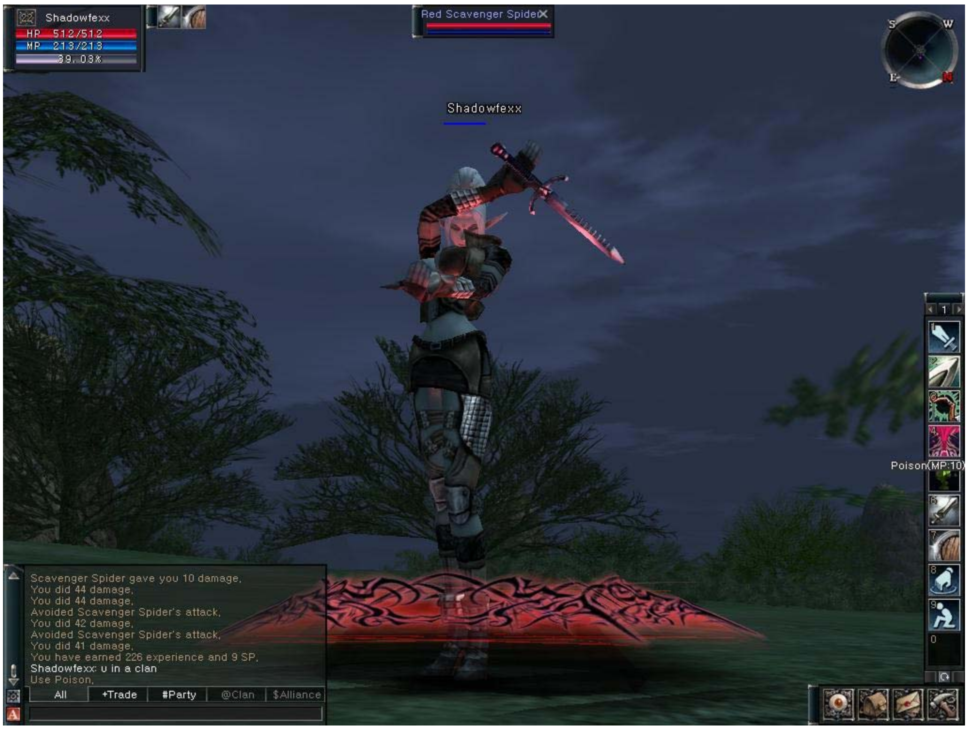
Sensitive data which should be kept from players is sent to, stored in, and used by an MMOG game client resident on a player's machine. This requires the game publishers trusting players to not attempt to gain access to this data, as well as taking measures to ensure cheaters cannot get at this data. If this data is leaked, it can lead to cheating.

6 MMORPG specific bugs

Some of Yan's cheating types are uncommon or not used at all in MMORPG, such as cheating by using artificial intelligence. This section will focus in detail on the types of cheating most common in MMORPG, and attempt to explain the relationships between information security and cheating.

A MMORPG is hosted on a realm (or bank) of servers which communicate with client machines that belong to players. The role of the client is to receive commands from the player and communicate them to the server, using the Internet as the communications medium. The client software provides the player with a view of the game, allowing them to interact and issue commands.

Screenshot of the popular Lineage 2 MMORPG



```
Scavenger Spider gave you 10 damage,
You did 44 damage,
You did 44 damage,
Avoided Scavenger Spider's attack,
You did 42 damage,
Avoided Scavenger Spider's attack,
You did 41 damage,
You have earned 226 experience and 9 SP,
Shadowfexx: u in a clan
Use Poison,
```

When a player performs actions on his game client, the game server takes these actions and updates the game in response to these actions. In other words, the state of the game is changed – from the player's view, the landscape may change, opponent's locations and actions also change.

To support the vast number of players playing MMORPG, games are often hosted across a number of servers (realms). This prevents a single server from being overloaded by client connections. In addition to multiple servers, MMORPG systems will also contain a number of databases, to hold information such as player account and virtual item data. With thousands of client processes running on a server, maintaining and updating the state of all these players playing in real time across the Internet becomes a headache.

6.1 Race conditions

According to [Wh04], "a race condition occurs when a program doesn't work as it is supposed to because of an unexpected ordering of events that produces contention over the same resource."

With MMORPG there are a huge number of client processes running on a server concurrently, and these processes will each be running a number of different threads. If these processes and threads share any resources, there is a chance they may interfere with each other – this is how a race condition may occur. In a MMORPG server, the game state is determined by the data being used. This data may be in memory, databases and other secondary storage. Changing an MMORPG state, involving the access and manipulation of this data by thousands of client processes, is an environment that is extremely susceptible to race conditions.

McGraw and Hoglund [MH07] describe race conditions and other state problems (discussed later) as the primary source of bugs in online games. They describe the problem using the example of MMORPG databases. Transactions which involve multiple databases they believe are vulnerable to race conditions. In WoW for example, different parts of the virtual world are stored on different servers. When a player wants to move to another part of the world, he may be transferred from one server to another. McGraw and Hoglund believe that events that cause these transfers (such as moving from one dungeon to another) are well tested during the software development process, and work correctly when a player does what they are supposed to do. However they suggest that by performing actions not expected by the programmers, cheaters can get unexpected results. They give the example of WoW, in which several "item duping" bugs exist around the entrances to dungeons – each dungeon is handled on a specific server, and when a player enters a dungeon, data is transferred from one server to another – raising the possibility of a race condition. Certain dungeons are more popular than others, meaning that server can become overloaded, leading to latency. This latency can be exploited to create an item duping bug in the following way. (The following was an actual WoW bug). Two players collude to duplicate gold; Player 1 gives Player 2 a quantity of gold outside the entrance to the Maruadon instance. Player 1 goes into the instance, which because of the heavy load, will not allow him to enter, and will throw him back out. After being kicked out, the gold Player 1 gave to Player 2 has not been deducted from him, but has been added to Player 2.

As the exact sequence of events and how WoW is coded is not known, only a likely scenario as to how this occurred can be proposed. The main problem is that trading gold is not an atomic operation. Rather than the gold being debited in one step, several steps are involved. This means that an attacker can insert actions of his choice in between these steps of the gold debiting operation. One such action is the entering of the laggy Maruadon instance. This delays or interrupts the trading operation, causing the debiting of gold from Player 1 to be delayed. Therefore the gold debiting operation takes longer than the gold crediting operation to Player 2. When the player cannot enter the instance because of the excess load, the gold debiting operation is cancelled and causes him to be reset to the state he was in before entering the instance. However as the gold crediting operation to Player 2 is much quicker and has already been completed, Player 2 keeps the gold.

6.2 Server lagging

A key element of the gold duping bug detailed above and other item duping bugs is lagging of the server. If a server is lagged, or delayed, it takes longer to perform operations. This exacerbates a race condition – it gives more opportunity for a cheater to perform actions such as gold duping, increasing the likelihood of a successful race condition attack.

Botnets for game lagging

One of the most effective ways to cause lag on a server is to use a botnet. A botnet is a network of compromised computers controlled by an attacker. A botnet can be used to send a vast number of packets to a given game server, causing lags as the server deals with the extra load it faces. Other techniques such as logging into a server multiple times with the same account can also be used to cause delays on the server. Multiple logins with the same account are not allowed, but processing these multiple logins uses computational power and thus increases computational delays.

6.3 State changing bugs

In many MMORPG, certain activities can cause a character to change into a different state. For example, in WoW, casting a spell called "Aspect of the Beast" allows a player to take on the characteristics of a beast, allowing them to become untrackable. Again, performing spells in ways in which the game developers don't expect can lead to bugs. An example is enabling Aspect of the Cheetah spell before joining a battleground, then straight away enabling Aspect of the Hawk – this can lead a player to gaining both aspects simultaneously. The crux of this problem is that with MMORPG there are many different states which interact, and certain unexpected combinations can lead to players getting an advantage over others. Such combinations were not envisaged by the game's developers, hence performing these combinations can cause unexpected behaviour.

6.4 Bots

Accumulating virtual items and moving up levels in MMORPG can be a long and monotonous process for many players. This has led to the growth of bots – programs that play the game on behalf of a player. In FPS, aimbots can automatically aim and fire at opponents. In MMORPG, bots are commonly used to automate the boring parts of the game. There is an additional incentive for the use of bots in MMORPG – virtual assets. Due to the fact that virtual assets can be traded for currency, it can become financially lucrative for players to accumulate as many virtual assets as possible. By creating bots which "farm" virtual assets, this goal can be achieved. Although bots can are used in other MMOG such as FPS, their use is widespread in MMORPG.

Using a bot provides a clear advantage to a player. Bots can run forever, without getting bored or tired like a human. The use of bots has become widespread in MMORPG, with a continuous "arms race" between bot developers and game publishers. Game publishers have taken drastic measures to counter bots. The publishers of the Lineage MMORPG, employ 150 game "minders" who monitor the game for bot use, and then issue bans to players they find who are using bots. According to [Si06], it is reported that 500,000 Lineage accounts had been suspended between 2004 and 2006 due to bot activity.

Macros

A macro is a set of commands (keystrokes and/or mouse clicks), that can be written to automate a task. In MMORPG, opponents such as monsters appear at certain locations in the game at periodic intervals. A macro can be written to get a character to stand in a location and automatically kill monsters when they appear. There are many websites in which macros similar in function to the above can be obtained.

How Macros work

Macros work by issuing commands using the user interface provided by the game client. Macros send commands to the client that appear to be normal keystrokes, mouse clicks or other game messages.

Controlling keystrokes

By controlling keystrokes, a character can be moved around, and keyboard shortcuts which lead to quests and other events can be generated. It is simple to generate keyboard events – programming languages contain keyboard API's which can be called to generate the keystream sequence desired.

Controlling mouse events

Mouse clicks and movements can also be simulated using standard API functions, giving the impression that a player is actually playing.

Pixel sampling

By sampling the colours of pixels on the screen, some vital game information can be obtained. An excellent example of this is determining character health. A character's health bar is usually located in a fixed place on the screen. By sampling pixel colours from this area, the health of the character can be judged. This is typically done by sampling the rightmost, centre and leftmost pixels on the health bar. If a character has little health left, the bar will usually be red in colour, whereas full health is indicated by green pixels.

Other measures, such as opponent's health (useful when in a battle) can also be obtained by pixel sampling.

Looting

Once an opponent is killed, it can be looted, to gain any valuable items it may possess. The loot can be taken by right clicking on the loot dialog box that comes up when an opponent is killed. As it is not known exactly where this dialog will appear, looting can be automated by getting the mouse to right click in a number of different locations on the screen.

Generic bot automation tools

There are a number of different programs and engines available for players to create their own macros.

AC Tool

AC Tool is a free macro creation tool that can be used in a number of MMORPG. It can be used to list a sequence of keystrokes and mouse clicks, save them, and then send them to the game at a later time, thus allowing simple tasks to be automated.

Inner Space

Inner Space is a platform developed by Lavish Software that allows the development of programs that can be used to perform a multitude of tasks within a game. The following quote from [La07] neatly summarises Inner Space's capabilities:

"Inner Space is a layer between Windows and games. Within this layer, programs written for Inner Space can safely perform nearly any task inside the game, with access to both standard Windows API and libraries, as well as the systems built into Inner Space. Inner Space provides systems for automation (scripting), input emulation, customizable user interfaces, bindable input (hotkeys), dual monitor support, and that's just scratching the surface!"

A detailed discussion of the architecture of Inner Space is beyond the scope of this paper. Simply, Inner Space allows programs to be developed in the .NET language. MMORPG are launched with Inner Space attached, which inserts the layer between the game and Windows. The introduction of this layer allows interactions with the game to be programmatically controlled. This means that programs can be created that allow key strokes and certain tasks within the game to be automated. A number of such programs have been developed using Inner Space for several different MMORPG. These programs perform tasks such as automatic fighting of monsters, or collection of gold. Websites, such as www.ismods.com, exist to allow players to upload scripts and programs for a variety of different MMORPG. Other useful cheating utilities exist, such as a program that automatically performs searches on a well known WoW cheat website whilst the game is running, to allow the player to find the latest exploits.

Custom made automation tools

MMORPG specific tools have been written by some developers to automate play in certain games. Examples are Glider for World of Warcraft and MacroQuest2 for Sony's EverQuest II game. [Mm07] describes Glider as "A tool that plays your World of Warcraft character for you, the way you want it. It grinds, it loots, it skins, it heals, it even farms soul shards... without you."

With respect to the 15 categories of cheating defined by Yan, I believe use of third party programs for cheating is a type of cheating in itself and should be added to Yan's list. Although it could be classified as cheating relating to virtual assets, I have shown that external programs are not solely used for this purpose.

Hiding bots from detection

Kernel resident bots

As alluded to earlier, cheaters are trying to develop bots that are harder to detect. One of the ways this is done is by creating a bot that resides in the kernel. To detect a kernel resident bot, anti cheating tools would need to scan the kernel to detect the bot. This would require the anti cheat tool to possess special access privileges. Another obstacle is that a kernel resident bot has full access to the kernel - this means that it can manipulate any function call the anti cheating tool makes, and therefore can hide itself from detection. This kind of behaviour is basically the same as a rootkit, and leads to the conclusion that a bot which acts like a rootkit would be very hard to detect.

Comment [R3]: How to detect these bots?? P.181 Hog

6.5 Features of cheating in MMORPG

The very nature of MMORPG gives rise to particular features of cheating not seen in other online games. One aspect of this is that bugs which lead to cheats are often kept secret by the people who discover them. The reason for this is simple: these cheats can lead to financial reward, and if other players learn of the bug and try it for themselves, the more likely it is that the game operators will learn of it. They will then fix the bug, cutting off any further opportunities for exploitation.

The way gameplay works in some MMORPG can also have an effect on the nature of cheating. For example, in WoW, players must choose between two factions when they enter the game. If a clan finds a cheat which allows them to kill opponents easily, they are highly unlikely to pass this onto members of the other clan. Not passing the cheat on increases the power of their clan. Thus certain cheats may only be propagated to certain groups within a MMORPG.

6.6 Conclusions

There are some forms of cheating that are more prevalent in MMORPG than other MMOG. MMORPG often involve the repetition of certain tasks, which has led to cheaters developing bots that automate boring parts of the game as well as working to accumulate virtual assets. Tools which help the creation of bots exist, and rootkit like techniques are being used to hide bots from detection.

The sheer complexity of MMORPG – they are huge distributed systems – leads to the possibility of race conditions and state conditions which are unexpected by the developers. If players can find these conditions it can lead to them gaining an advantage over other players – in other words, it leads to cheating.

7 Cheating prevention methods

There are a number of different techniques employed to cheat in MMOG. The most popular method is to use bots to automate gameplay with the aim of accumulating virtual items which can be sold later.

Other methods seek to exploit weaknesses inherent in massively distributed systems such as MMOG, for example the storage of sensitive data on the client's machine, and race conditions. The next section will discuss some methods that can be used to mitigate cheating.

Information security has a vital role to play in preventing cheating in MMORPG. MMORPG are basically massively distributed systems. This means the lessons learned here can be applied to the development of other massively distributed systems in the future. Such systems could include company web applications that are used by all the employees of global organisations, who may be located across the world.

7.1 Client side cheating

A game client installed on a player's machine is vulnerable to reverse engineering by an attacker. By reverse engineering the game client code, the inner workings of the code can be scrutinised and bugs which may exist in the code can be found and exploited. There are several methods that can be used to prevent client side cheating.

Preventing data exposure

Game clients come as executable files, and with the help of a disassembler and debugger, an attacker can attempt to reverse engineer the game executable. Measures must be taken to ensure that reverse engineering is made as difficult as possible for an attacker. If an attacker can successfully reverse engineer the game client, he can begin to identify any vulnerabilities, which if they exist and the attacker can exploit them, can lead to cheating. Values of sensitive variables can be found and tweaked. An example of this is a buffer overflow vulnerability discovered in the PunkBuster remote web administration tool discussed later. Another reason to make reverse engineering as hard as possible is to protect intellectual property rights. If code is reverse engineered, its algorithms can be studied and imitated by a competitor. This can lead to a firm losing its advantage over its rivals.

Packing the executable

Packers compress and encrypt the contents of executable files. Packers also perform some anti-disassembly and anti-debugging techniques. These techniques can include removing all comments inserted into the code by programmers. Programmers often leave debugging comments in their code, which can provide valuable clues to reverse engineers. A packer modifies the contents of the executable on disk. When the program is run it is decompressed in memory, and starts at a section that contains the decompression and decryption code.

Using a packer means the executable on disk cannot be disassembled without being first unpacked, as it is in a compressed and encrypted state.

Packer weaknesses

A number of universal packers exist, and due to their widespread use, people have created unpackers for them. Unpackers decompress and decrypt packed code, reversing the transformations of the packer. It is important for game developers to make sure they do not use a universal packer to pack their code. If a universal packer was being used, it could be unpacked easily using the corresponding unpacker, making the original use of the packer meaningless.

This has lead to the development of custom packers, which are designed to pack individual executables. Using a custom packer introduces an element of security through obscurity. Custom packers are not exposed to wide peer review. This leaves the risk that software bugs, ill-thought out use of encryption, and design errors which exist in packers, may not be found. Consequently, this can lead some packers to give a false sense of security.

Packers possess one inherent weakness which can be exploited to defeat them – reading the program from memory. When the program is executed, it is decompressed in memory. Therefore an unpacked version of the program is in memory and can be analysed. Analysing the program in memory is a more complicated task, but one that can be achieved.

Anti debugging measures

Using a debugger allows an attacker to step through code line by line and examine the values of variables as he does so. Debuggers run by attaching themselves to the client program, so detection involves checking the target system to see if a debugger is present.

Detection by timing

This method is based upon the game program reading the system time at defined intervals. If the time between samples is above a pre-defined threshold, the program assumes it is being debugged. This is a reasonable assumption as debuggers pause and slow down the execution of a program. This measure is crude, and is easy to defeat, by simply avoiding pausing the program and avoiding any activity that may considerably slow the game down.

Code obfuscation

This is process of "converting a program into an equivalent one making reverse engineering uneconomical" [Co98].

Obfuscation is designed to make source code hard to understand and read. There are a number of different techniques that can be used to accomplish this. Variable names can be changed to symbols, such as the underscore (_) symbol. Control loops are also modified to make them harder to understand. The problem with obfuscation methods is that although they make things harder to understand, a determined attacker will eventually figure out how the code works.

Preventing access to game client data

There are two ways in which a cheater can gain access to game data – either by intercepting packets sent from the game server to the client, or by accessing data stored by the game client itself or in memory whilst the game client is running. If a player can get access to data, he can perform cheats such as maphacks.

One way to prevent the leakage of sensitive game data is to use encryption. If packets sent from the server to the game client are encrypted, an attacker would have to break the encryption scheme to obtain the data. However, the encryption scheme used cannot not be computationally intensive, as this would increase the load on the server when it encrypts each packet. As hundreds of thousands of packets are sent simultaneously, this would increase the load on the server greatly, causing packet delays, which would ruin the player's gaming experience.

A problem with using encryption to protect game data is that the client must know how to decrypt the data sent to it. This means the decryption key must be sent to the client, or stored in it. If the key is sent to the client, it can be sniffed by an attacker, as it is sent in the clear. If the key is stored in the client, if the client program is reverse engineered, the key may be exposed, meaning that all the data encrypted with that key can be decrypted.

Sensitive game values that are stored in the client and in memory when the program is running should be encrypted. Otherwise they can be read simply by using a debugger. However this suffers from the same problem that the decryption key must be stored or sent to the client program.

Reducing information on the client

With the risk that any information stored on the client may be exposed, an alternative approach would be to reduce the amount of information stored on the client. This could mean that the game server handles all graphical rendering and then pushes this to the clients. This means that sensitive values, such as the locations of enemies or special weapons, are not sent to the client at all. This solution would require a re-design of the game's software and architecture, which would be an expensive process. Another significant obstacle is that the load on the server becomes too great. Constantly updating each client's game state and sending this back to all connected clients is a difficult task. Doing so would undoubtedly lead to laggy gaming experiences for players, resulting in end user dissatisfaction with the game.

7.2 Detecting game client integrity violations

Attackers will modify the game client to their own advantage. This can be done by either replacing a file in the game client with their own version, or by DLL injection.

A file that has been modified violates the integrity of its data. A modified file can be detected using a modification detection code (MDC), a hash function. By calculating an MDC on each legitimate (untampered with) game file and then comparing with MDC's calculated on game files on the client's machines, if any MDC does not match, this indicates the file has been tampered with. An advantage of using hash functions as MDC's is that they are easy to compute, therefore there is little computational overhead in computing and sending hashes of files on the player's machine to a server for checking. Hash functions are second pre-image resistant, which means an attack in which an attacker creates a file in such a way that the file he creates possesses the same MDC as a legitimate game file, is very hard to perform.

7.3 Preventing game bugs

Designing secure software

The importance of creating secure bug free game code cannot be underestimated. MMORPG are played by hundreds of thousands of players, and with this number of players, the number of threats that the software is exposed is huge. MMORPG represent a challenge which is unlike other software projects. MMORPG are massively distributed systems, with hundreds and thousands of users playing simultaneously. Using the Internet to connect to the game increases the security risk, as malicious users can gain access to the game easily. Sharing game state with so many users concurrently just augments the complexity. These factors serve to make MMORPG one of the hardest types of software to produce. This means that security cannot just be bolted on as an afterthought – this increases costs and complexity. Therefore security must be considered from the very beginning.

Awareness of security should be extended to people involved at all stages of the software development process – developers, testers, architects and managers. This raises awareness and increases knowledge throughout the organisation. Having a development process in which security is an integral part offers several benefits. Bugs are found and fixed earlier, which reduces costs and increases the quality of the end product. Player confidence in the game is increased as there are fewer bugs.

The software development process

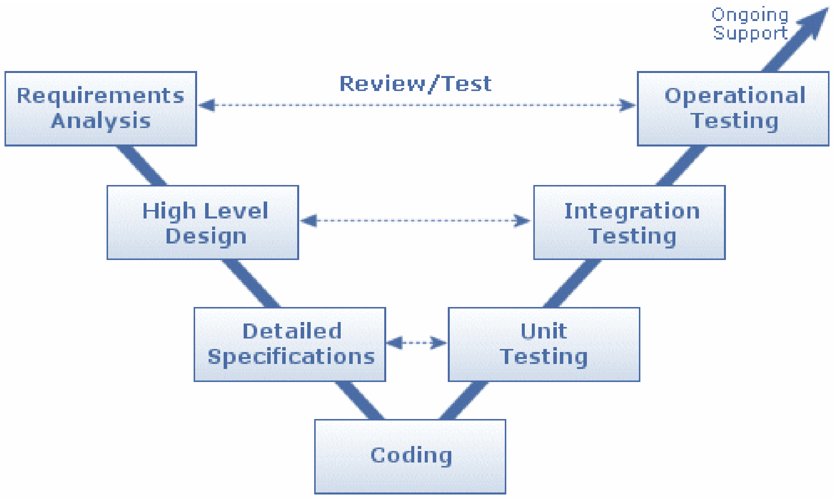


Diagram taken from www.uksh.com/about/software-development-life-...

The design phase of the software development process must consider threats to the security of the system. If it does not, the design may be insecure, resulting in a system that has many vulnerabilities. Considering security from this stage means the system can be designed to address any threats it faces. Threats to the system can be modelled by documenting "abuse cases" – behaviour which is designed to break the system, and also by undertaking a risk analysis.

The results of these analyses are fed into the system design stage, which after completion may be reviewed by security experts. After thorough review of the design, the system is coded. Mistakes during the programming phase can lead to bugs later. Therefore programmers should be educated about how to write secure code, and secure coding guidelines should be defined to ensure a uniform approach. By educating developers, bugs caused by malformed inputs, for example, should be reduced - reducing the opportunity for cheaters to exploit them later.

The testing phase of MMORPG development plays a vital role. It is at this stage that potential cheats are discovered. Testers should not just focus on standard "use cases" or functional testing of the system. They should try and do unexpected things in the game to see its response. As discussed previously, state bugs in which unexpected interactions cause unusual behaviour are a prime example. With so many players, the chance of a bug caused by a player doing something the developers did not expect becomes a real possibility. Therefore testers have an integral role to play in finding such bugs. In MMORPG these are the bugs that can lead to item duping, and thus to rampant cheating.

7.4 Preventing race conditions

Race conditions can be prevented by careful design and coding of game software. Preventing race conditions involves ensuring that any thread which is manipulating data does so exclusively. This can be done by making sure the thread obtains a "lock" on a resource, which means the thread has exclusive rights to access or modify the resource. Ensuring that threads obtain the right locks in the right order, and preventing "deadlocks" in which threads may get stuck waiting for each other to release a locked resource, is not an easy task. With the complexity and multitude of resources and processes in an MMORPG, it becomes an even more complicated task. Therefore careful design, coding and testing is vital to ensure the number of race conditions is minimised.

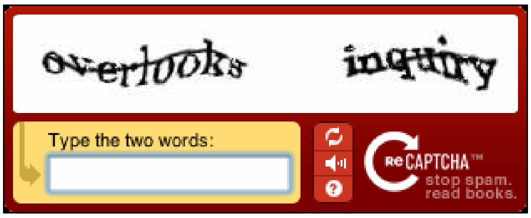
7.5 Bot detection

CAPTCHAS – Reverse Turing tests

The goal of bot detection is to see if a human is playing the game or not. One of the simplest ways to do this is to use a challenge-response test. A challenge-response test for exactly this purpose is the CAPTCHA – an acronym for "Completely Automated Public Turing test to tell Computers and Humans Apart".

The principle behind CAPTCHA's is simple. A server generates and sends a test to the user's computer. The user's computer is not able to solve the test on its own – solving it requires a human. The test usually requires the user to type the letters of a distorted image.

Example of a CAPTCHA



CAPTCHA's can be used to detect MMORPG bots by having the game server send a CAPTCHA to random players. If the test is not answered correctly within a certain time period, then it can be assumed that the player is using a bot. The response must be given within a certain time period to ensure liveness of the player. This is to ensure that the CAPTCHA is not circumvented in the following way: the CAPTCHA is sent to the player, who has a bot running. The player comes back a few hours later and finds the CAPTCHA on his screen. He then responds to the CAPTCHA and is not considered a cheat. If a requirement that the CAPTCHA must be completed within two minutes (for example) is implemented the opportunity for this type of circumvention is greatly reduced. There is still a possibility that a player who is using a bot happens to see the CAPTCHA within the two minutes of it being issued, stops the bot, and responds to the

CAPTCHA correctly. Periodic sending of CAPTCHAs to player's machines will help to catch bots which evade detection first time round.

As with any challenge response system, the challenges that are issued should not be repeated or predictable. If they are, cheaters will realise this and write programs which are able to predict and generate responses to CAPTCHA's automatically, allowing them to use bots undetected.

Detecting mouse and keyboard movements

When a bot is used, there are no keyboard presses or mouse movements – everything is scripted. When a human plays, keys are pressed and the mouse is moved. This distinction forms the basis of a method that could be used to detect bots. If key presses, mouse movements and clicks could be recorded somehow and sent to the server, proof that a human was playing would be established.

Hardware signing

This measure is based upon an idea by Wu-Chang Feng of Portland State University [Fe07]. It is based upon new hardware - secure mice and keyboards. It revolves around a mouse, when it moves, or is clicked, producing data that says it has moved or been clicked. For the moving of a mouse, this could be the co-ordinates of its movement, for example. The mouse needs to possess a unique private cryptographic key. This key could be used by the mouse to produce a digital signature on its movement or click data.

The idea is to use "hardware signing" to prevent bot use. The game client could be configured to pass digitally signed data about mouse clicks/moves or keyboard presses by players to a monitoring server hosted by the game publisher. A public key certificate, provided by the mouse manufacturer, could then be used by the publisher to verify the digital signature on this "movement" data. A unique identifier would need to be possessed by the mouse and keyboard. This must be done as keyboards and mice are portable, and can be moved to other computers. This allows the mouse to be associated with another player at a future time.

The monitoring server would store the player's id, keyboard/mouse ids and the digitally signed movement data messages from the mouse and keyboard. If over a certain period, there are no digitally signed messages for a given player, it can be assumed that they are using a bot, as bots do not generate mouse clicks or key presses. This period must not be too short – it is possible that a player has paused the game temporarily, so mouse and keyboard movements would not be expected.

Using a digital signature provides data origin authentication which is necessary in order to be sure that the movement came from that particular device. It also provides data integrity, which means that if any modification is made to the movement data it will be detected.

To address privacy concerns, it is obvious that the data signed by the keyboard should not reveal any personal information. Therefore only data that a key has been pressed (and not the particular key that was pressed) should be revealed.

Problems with hardware signing

This method would require new hardware devices capable of generating information and signing it. These devices are likely to be costly, and hence only a small percentage of the game's players will obtain them. If details of such a cheat detection scheme were made public, cheaters would definitely not buy these devices! Game makers could force players to buy them by not allowing them to play the game without them, but this could be a public relations disaster if they were seen to be forcing players into this.

Player monitoring

This approach involves scanning the running processes of players' machines to see if they are running known bot programs. This could work by taking a certain number of bytes in memory from running processes, hashing them, then sending the hashes to a detection server. The detection server would have precomputed and stored hashes of the same data from these known cheating programs, and if a hash obtained from a player's machine matches a stored hash, that player is known to be running a bot program.

This approach requires the bot program to be known before it can be detected. Thus effort is required to find new exploits. Cheating exploits are often shared between closed user groups, so obtaining bot code is not a simple task.

One advantage of this approach is its simplicity. Computation of hashes is relatively fast, and simple comparison is needed. Scanning a player's machine to find known cheats is easier than defining IDS type models to detect bots (discussed later). Detection using player monitoring means cheats can be found and banned quickly, keeping player confidence that the game is "clean". Player monitoring is used in the Warden, Blizzard's anti cheating solution, and PunkBuster, a standalone cheating detection tool, discussed later.

Player reporting of cheats

Some MMOG include facilities for players to report gamers who they suspect are cheating. Eve Online allows players to file "petitions" in which they report suspicious player behaviour to the game administrators. The administrators then use game logs to investigate the player's actions to see if the petition is justified. A similar mechanism can be used by honest players to report bugs to the developers.

It can be argued that this method places too much reliance on players to report cheaters. Players must be prepared to spend their own time filing a petition. It also relies on players not abusing the system by maliciously reporting others. Players may be tempted to report an opponent who they may bear a grudge against, even though they may not even be indulging in cheating behaviour. This would lead to the game administrators wasting time investigating a false complaint.

IDS type approaches

The principles upon which anomaly based Intrusion Detection Systems (IDS) are based can also be applied to detection of MMORPG cheats. Anomaly based IDS works by observing and collecting data on legitimate user behaviour, and then using statistical tests on observed user behaviour to determine whether the behaviour is legitimate.

One such measure that has been researched is the timing of client commands in MMORPG. This was proposed in research conducted by Chen, Jiang et al [CJ06]. Client commands are, for example, attacking a monster or casting a spell. Humans initiate client commands by mouse clicks and/or keyboard strokes. They distinguish this with the behaviour of bots, in which triggering of client commands is decided by programming logic in the bot program. They postulate that the design of when to issue the next command is important as it can lead to major discrepancies in traffic patterns between bots and humans, and these patterns can be analysed to detect bots.

They propose two possible measures that can be used to detect bots by traffic analysis – command timing and traffic burstiness.

Command timing

This refers to the time difference between a client packet departure and the most recent server packet arrival time. [CJ06] found that some of the bots they examined, after receiving a server packet, use a scheduled intentional delay time before issuing their client commands, leading to regularity in response times. Other bots would respond extremely promptly (within 10ms) to a server command. As a result they created a bot identification scheme based on prompt responses and regularity in response times.

Traffic burstiness

Traffic burstiness is the variability of the number of packets sent in successive periods, and indicates how traffic fluctuates over time. Chen, Jiang et al hypothesise that as a result of the bots periodicity (scheduling a delay before issuing commands), they exhibit smoother traffic patterns compared to player traffic. Bots take action based on server packets, which are periodic for smooth screen updates. As a result this periodicity filters through into bot traffic. Therefore they were able to find lower traffic burstiness around times when the game state was updated.

Results of their research

Chen, Jiang et al collected traffic data on MMORPG traffic which was generated by bots and human players alike. Using the command timing and traffic burstiness approaches, Chen, Jiang et al achieved correct decision rates (ratio that the client type of the trace was correctly determined) higher than 95% and false negative rates (fraction of bots judged to be human players) less than 5%, with an input size of more than 10,000 packets.

They propose a conservative approach to bot detection which states that a traffic stream can be considered to come from a bot only if the command timing and traffic burstiness tests agree. By doing this, they reduced the false positive rate to zero, and achieved a 90% correct decision rate, within an input size of 10,000 packets.

Attacks against this method

Chen, Jiang et al acknowledge that a bot which adds random delays to the release time of its commands renders the command timing scheme ineffective, as this method relies on the regularity of bot reactions. They reason that the traffic burstiness scheme is immune to random delay attacks because bots take action in response to packets from the server which are periodic in nature. Random delays do not affect the burstiness of the traffic, just when that burst occurs.

The bursty nature of player generated traffic can be simulated by a bot, by turning itself "on and off". This means that traffic is not sent in one go, but rather in bursts. The authors argue using this "on and off" behaviour in a bot reduces its effectiveness, as it may be idle for long periods of time – defeating the purpose of the bot playing constantly while the player is away.

Machine learning of player reactions

Another approach that can be used to detect aimbots is to learn player reaction times. Player reaction times to opponents in FPS can be recorded. These measurements could include the time it takes for a player to kill an opponent who suddenly appears. Measurements should be taken from the best players, as this will give indications as to the quickest possible human reaction times. This measurement can be used for a threshold detection mechanism. If a player during gameplay is recorded as having a reaction time quicker than the threshold level, that player can be flagged and assumed to be using an aimbot.

I believe IDS-like approaches represent a good way to detect bots. Bots can be distinguished from players in terms of playing ability (for example aiming and killing opponents) and in the way bots respond to server commands. The research undertaken by Chen, Jiang et al represents a promising start to the use of IDS type measures to detect bots, but there is still a lot of work to be done before this can be used as a bot detection mechanism by game publishers.

8 Anti-cheating tools

There are a number of approaches game publishers have undertaken to combat online cheating. Software manufacturers have designed standalone anti-cheating tools, such as PunkBuster, which work in conjunction with many different MMOG. Game publishers such as Blizzard have also designed proprietary anti-cheating tools, such as the Warden, which are designed for a particular MMORPG - the Warden is designed specifically for World of Warcraft.

8.1 PunkBuster

Tools which fall into this category include **PunkBuster** (PB), an anti cheating tool designed by Even Balance. PB is based on a client/server architecture. The client runs on the player's machine whilst they are playing the game, and the server software runs on the game publisher's server. PB operates in a similar manner to anti-virus tools: it scans the player's computer whilst they are playing the game, looking for known cheats and exploits. Periodic status reports are sent to the PB server, and if suspicious behaviour is detected, a violation is raised at the server, which can cause a player to be removed from the game. To ensure that the latest exploits are caught, PB includes an auto update feature which allows the client to connect to a master server and download the latest version of PB without interrupting gameplay.

How PB works

There are three types of cheating that PB aims to detect: modified client variables, modified game files and use of known cheating exploits.

Detecting modified variables

Due to the proprietary nature of PB, the details on how it operates have been taken from EvenBalance's official documentation, and precise implementation details are unable to be provided as they are not revealed by EvenBalance.

PB examines player's computers for modified client variables (known as cvar's). Cvars are variables that are modified by cheaters to change the way things are displayed (i.e.

to perform wallhacks). Each game server which hosts PB has its own configuration file, that allows game server administrators to define the permitted values for cvars for clients using that server. Detection of irregular cvars occurs by the sending of regular, encrypted status reports from the PB client to the server. These cvars are then compared against the permitted cvar values, and any deviation can lead to a player being banned for cheating.

Detecting modified game files

To detect if a player has tampered with their game files, PB computes an MD5 hash of certain key files (chosen by the PB server admin) contained in the game installation directory. The MD5 hash acts as a modification detection code, or checksum. The PB server stores MD5 checksums computed on the game's files. The stored hashes are compared with the ones taken from the player's computer. If they differ, it can be deduced that the player has tampered with his game files. A warning or ban can then be issued to the player.

Detecting known cheating exploits

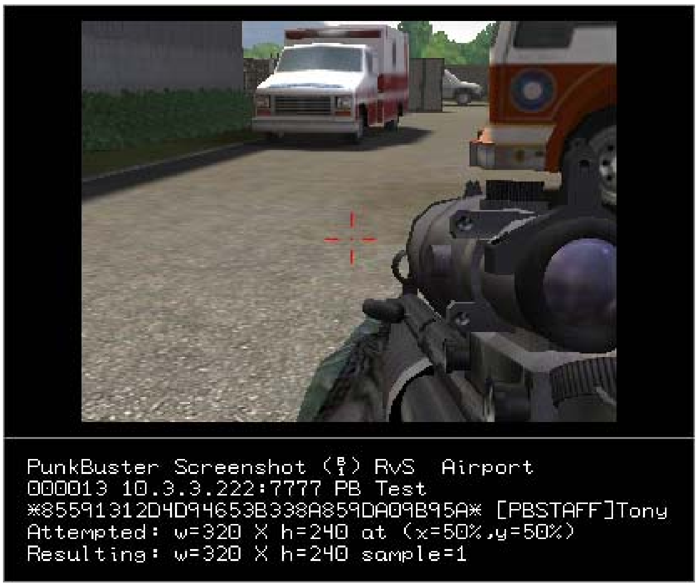
To detect known exploits or 3rd party programs that are used to cheat, PB performs random memory scans on the player's machine in order to see if the player is running any known cheat programs on their machine.

Screenshots

PB also has a facility which allows administrators to request screenshots of currently connected players. These screenshots are then transmitted and stored on the PB server. Screenshots can provide evidence of player cheating, such as use of wallhacks. The ability to take screenshots of player's screens raises privacy issues. Apart from capturing a screenshot of the game, does an admin need to know what else a player has on their screen? They may be able to find out a player's bank account details if they are using an Internet banking service, for example. What is to stop a rogue admin repeatedly taking screenshots from a player's screen, and gaining sensitive personal information?

To address such concerns, PB has two different safeguards. PB limits how many times a PB server can request a screenshot, to prevent server administrators from taking screenshots one after another. To stop confidential non game information being leaked, if a screenshot is taken but the player has minimised the game, the screenshot will be blank. Hence only screenshots of the game only can be taken, not of any other windows, such as web browsers.

Example of a screenshot produced by PB



```
PunkBuster Screenshot (?) RvS  Airport
000013 10.3.3.222:7777 PB Test
*85591312D4D94653B338A859DA09B95A* [PBSTAFF]Tony
Attempted: w=320 X h=240 at (x=50%,y=50%)
Resulting: w=320 X h=240 sample=1
```

PB use of digital signatures

Another concern is the manipulation of screen shots by administrators to ban players for malicious reasons. Administrators could swap screenshots between players, or even doctor them using image editing tools to concoct evidence of cheating. To prevent this from happening, EvenBalance claim a digital signature is computed on the screenshot before it is stored in a log file on the server. They also claim that a digital signature is also calculated and stored with the log file itself. The name of the log file along with the digital signatures are sent to all players.

Using a digital signature should prevent several different threats to screenshots and logs. Alteration of logs or screenshots after they have been created can be detected, as the digital signature sent to the player will not match a signature calculated on the modified log file. By signing them provides data origin authentication – players can be assured that they came from the PB server.

After reading in detail the information on EvenBalance's website about how they use digital signatures, it became apparent to me that they were in fact not using digital signatures at all to protect their log files. [Ev06] states that "We have for download, on our website a free utility called pbmd5 that will compute the Digital Signature (also known as MD5) of any file."

MD5 cannot be used on its own to compute a digital signature, and stating that it can is incorrect. MD5 is a hash function, not a digital signature scheme. By using MD5 to protect their log files only provides them with weak data integrity. The reason why they only provide weak data integrity is because there is nothing to stop someone with access to the logs from changing the log files contents, then computing a new MD5 hash value for the log file, and storing it. When someone wants to verify the integrity of the log file, they recompute this hash, which will match. Therefore even though the file has been altered, the person verifying the files will believe they have not been tampered with as the MD5 they compute on the file matches the one stored for the file.

Log files

The way PB transmits information about log files to players is in my opinion, unsatisfactory. At the end of the game session, the PB server reports the filename of the log files and their MD5 checksum (digital signature in EvenBalance's language) to players. To capture this information for future use, EvenBalance's website recommends "bringing down the game console and then taking a screenshot of your playing screen while that information is displayed". I believe this method places far too much reliance on the player to record this information. What happens if a player simply forgets? The following situation becomes possible. As a player has no record of the checksums of the log files, a malicious admin could modify the logs to show that a player has used a known cheating tool last time they played the game. The admin could then recompute the checksum, and store it with the log file. This could then be used as evidence to ban that player from the game. The banned player then has no redress – he cannot argue with the checksum on the log file as it computes correctly and he has no evidence that the log file has been modified.

The use of MD5 to provide data integrity for the log files is also a cause for concern. MD5 has shown to be vulnerable to a collision attack. In 2005, a Chinese cryptographer, Xiaoyun Wang, demonstrated a method in which collisions could be found for MD5 in as little as 15 minutes. Using MD5 opens up the possibility of an attack where an attacker creates a number of minor modifications to a log file. The birthday paradox tells us that collisions in MD5 hash functions are found with complexity 2^{64}. With Wang's collision attack, collisions in MD5 can be found with much less complexity: 2^{39}. Due to the fact that collisions in MD5 can be found with less complexity, the following attack becomes possible. This attack is based on one conducted by Daum and Lucks. Based on Wang's work in finding collisions in MD5, Daum and Lucks "implemented an attack which found random collisions in MD5. It took just a few hours on a customary PC" [DL05]. Using this method, they were able to create two postscript files that produced identical MD5 hashes. The original postscript file contained a letter of recommendation for an employee, while the other contained a letter that granted security clearance. I believe a similar approach can be used against the PB log files. If an authentic log file is taken then a number of possible log files can be created. After 2^{39} different log files have been created, there is a greater than 50% chance that one of these log files will produce the same MD5 checksum as the original log file. The log file the admin has produced may show that a player has been banned for violating game rules, for instance. The admin may then replace the original log file with his version. The admin can then ban the player from the game, and can claim to have evidence to back up the claim that the player has been cheating. However this evidence has been fabricated. The admin may then even confiscate the banned player's virtual assets and then sell them. If such behaviour was to occur, this would have a severely detrimental effect on the game operator's business. Trust that legitimate players had in the operator would be eroded, and the negative effect on the operator's reputation would inevitably lead to players quitting the game.

Screenshots and MD5

As PB uses MD5 checksums to verify the integrity of screenshots it captures from players, a collision attack can also theoretically be carried out to produce a screenshot that has an identical MD5 checksum to an authentic screenshot stored on the server. A screenshot could then be produced to show evidence of a player indulging in cheating. Screenshots could be changed fairly easily using image editing tools such as Paint Shop Pro.

Remote logging facility

EvenBalance is in the process of offering players and server administrators the chance to remotely log their information. This involves screenshots and logs being stored in password protected files on EvenBalance's servers. If EvenBalance continue to use MD5 hashes and do not instead use digital signatures based on public key cryptography with supporting public key certificates, there could be problems. One problem with relying solely on MD5 checksums is the fact that they provide no data origin authentication. With remote logging, there is nothing to stop an attacker from creating a screenshot or a log file, computing an MD5 checksum and then simply storing it in the user's file. Such an attack could be prevented by using digital signatures. EvenBalance could sign the screenshots and log files with a signing key, and players could then use the corresponding public verification key and public key certificate to verify EvenBalance's signature. This would provide the log files and screenshots with data origin authentication, data integrity and non-repudiation. Log files may be used by administrators to check if there has been any cheating behaviour on their servers. If digital signatures are not used to protect the log files, cheaters would be able to delete evidence of cheating behaviour from the log files, compute a new checksum and thus avoid detection completely. Therefore I believe PB must use digital signatures to protect log files in their new remote logging facility.

Attacks on PB

As a powerful anti-cheating tool that is used in a number of MMOG, PB is an obvious target for attackers who aim to circumvent it. Circumventing PB usually involves one of two things: using an exploit or cheat that PB cannot detect, or attacking PB itself.

As mentioned before, cheats are propagated through limited membership, private cheating groups. PB does not use a heuristic approach to cheat detection - it relies on memory scanning of player's machines to detect cheats, therefore the cheat must be known before it can be detected. Due to the private nature of cheating groups, discovering cheats is not always an easy task.

Attacks on PB itself are a much harder task to accomplish. PB checks to ensure that the files it uses have not been altered, which makes replacement of PB files detectable. PB also uses "hardware bans" to deter would be cheaters. A player receives a hardware ban if memory scans show that a cheat that is known to circumvent or disrupt PB's normal operation or its facilities is activated.

Hardware bans

Players caught cheating on a PB enabled server would be banned from that game server by the admin. They would then go to another game server and attempt to cheat there until they are caught. This was not an ideal situation, as cheaters were just transferred from one game server to another rather than being removed from all game servers permanently. To address this, PB introduced the concept of hardware banning. This involves the creation of a GUID (Globally Unique Identifier) to uniquely identify each cheat.

Tools can automatically record information about cheating players, including the GUID, and other information such as the cheater's IP address and any known game aliases. These tools then automatically feed this information into a Master Ban list. PB servers can be configured to automatically update their own banning lists with this master ban list. This system greatly reduces the overhead for server administrators, as they no longer have to rely on banning cheaters manually. By automatically updating servers with banning lists means that cheaters will only have limited opportunity to switch to playing on servers they have not been banned on, as the server's banning lists will quickly be updated. This system allows cheaters to be banned from all PB enabled servers quickly and effectively.

GUID computation

The GUID is used to identify a cheat, and is computed by using unique hardware identifiers and the game installation CD key as inputs to a hash function. Such hardware identifiers include the serial number of the hard drive on the cheater's machine, and the MAC address of the network interface card (if one exists). By using a hash function means that no serial numbers for individual computers are stored. For a cheater to overcome a hardware ban, they would have to change their hard drive, and other hardware components. (It is not known exactly which hardware identifiers are used to compute the GUID, but each hardware device that is used to compute the GUID would have to be replaced). Although this may represent a degree of "security through obscurity" by not revealing what hardware components are used to compute the GUID, changing many hardware items is an expensive and prohibitive process for many cheaters, and as such the process of overcoming a hardware ban is a costly process. This represents a deterrent to players who are wishing to cheat.

However the practice of hardware banning does raise some issues. For example, if a computer is purchased second hand, it may have a hardware ban caused by a previous owner. This means that the new owner will not be able to play any MMOG that use PB. EvenBalance states that hardware GUID bans are permanent and cannot be reversed, which makes this situation worse for an innocent player, as there is no redress.

Vulnerabilities in PB itself

As stated before, PB is an attractive target for players wishing to cheat. Would be attackers can attempt to find vulnerabilities by reverse engineering the PB code. Skilled attackers can examine this code to see how it works, and if there are any ways to attack or circumvent it. Avenues for attacks would include looking for buffer overflow vulnerabilities. These occur when an attacker constructs an input which the code does not handle correctly, and allows an attacker to execute his own code. A buffer overflow vulnerability was found to exist in PB's WebTool utility. This utility is an HTTP Server that allows administrators to remotely administer their PB Server via a web browser.

This tool requires a key parameter and a password to authenticate the user. However, the length of the key parameter is not checked before it is copied to a buffer, and if a long key is entered, the buffer can overflow and lead to remote arbitrary code execution by an attacker, allowing him to gain control over the PB server. Detailed information about this vulnerability can be found at http://www.symantec.com/avcenter/attack_sigs/s21892.html

This could lead to a whole host of potential attacks: the PB server could be disabled, and the cheater could then connect to the game server and run cheating exploits without fear of being caught. Log files could be deleted to remove any trace of illegitimate activity. There is no documented evidence of such an attack taking place, but it underlines the importance of ensuring there are no bugs in anti-cheating programs. It is imperative that makers of anti cheating tools test them thoroughly to ensure there are no input validation errors. As attackers can find these bugs by disassembling the executable code, obfuscating the code by renaming variables, removing code comments and making program logic harder to follow all serve to make a would-be attacker's task tougher and more time consuming. However, obfuscation will not stop an attacker who is determined enough.

PB Summary

PB uses the concept of player monitoring to detect MMOG cheats. By scanning a player's machine and verifying the integrity of game files and cvar's, PB hopes to catch cheaters. EvenBalance's idea to use hardware bans to permanently ban cheaters from all PB supported MMOG is an excellent idea which should stop the problem of cheaters jumping from one MMOG to another.

I am concerned about PB's use of MD5 as the hash function to verify the integrity of files, as MD5 has shown to be vulnerable to collision attacks. Also of concern is EvenBalance's belief that MD5 can be used to produce a digital signature. This, in my view, represents an alarming lack of understanding of basic cryptography. I have shown that not using digital signatures correctly could have adverse consequences on PB's remote logging facility.

8.2 The Warden – Blizzard's approach to cheating

The Warden is Blizzard's anti cheating tool that is used in a number of their games, including WoW. The Warden was reverse engineered by Greg Hoglund in 2005, which exposed how it functioned. Much of the subsequent discussion is based on Hoglund's work.

What the Warden does

Hoglund [Ho05] states:

"The Warden is downloaded on the fly from Blizzard's servers, and it runs about every 15 seconds. It reads information from every DLL loaded in the 'World of Warcraft' executable process space. The Warden then uses the GetWindowTextA function to read the window text in the titlebar of every window. These are windows that are not in the WoW process, but any program running on your computer. Once these strings are obtained, they are passed through a hashing function and compared against a list of 'banning hashes' - if you match something in their list, I suspect you will get banned. Next, Warden opens every process running on your computer. When each program is opened, Warden then calls ReadProcessMemory and reads a series of addresses - usually in the 0x0040xxxx or 0x0041xxxx range - this is the range that most executable programs on Windows will place their code. Warden reads about 10-20 bytes for each test, and again hashes this and compares against a list of banning hashes. These tests are clearly designed to detect known 3rd party cheating programs, such as WoW Glider. Every process is read from in this way."

Privacy issues surrounding the Warden

The functions of the Warden raise several privacy issues. Reading the name of each window that is currently open on the player's machine can expose sensitive personal information. To quote Hoglund's experience:

"I watched the Warden sniff down the email addresses of people I was communicating with on MSN, the URL of several websites that I had open at the time, and the names of all my running programs, including those that were minimized or in the toolbar. These strings can easily contain social security numbers or credit card numbers, for example, if I have Microsoft Excel or Quickbooks open with my personal finances at the time.”

These strings are passed through a one-way hash function, which produces a digest which is hard to reverse. By “hard to reverse” this means that it would be computationally infeasible for Blizzard to try and obtain the original strings from the hashed values. However, I do not believe hashing these strings is a strong enough safeguard for personal data. The reason for this is that the Warden itself could easily be modified to not hash these strings, and simply transmit them to Blizzard in cleartext form. This would lead to the leak of hundreds of thousands of players’ sensitive personal data. It is not an easy task to ensure this does not happen. The Warden is Blizzard’s proprietary software and would have to be reverse engineered by an outsider to determine how it works. Blizzard could make reverse engineering extremely hard by obfuscating their code, for example by encrypting it. This would make determining exactly what the Warden was doing a tough task. This is exacerbated because the Warden is downloaded on the fly from Blizzard’s servers while the game is running. This raises the possibility that the Warden code could be changed to capture sensitive strings in cleartext form for a period of time (by not hashing them), then be changed back to hash these strings at a later time. Even if this was done for a period of a few hours, the amount of personal data collected would be huge, as the Warden runs every 15 seconds and WoW has thousands of players playing at any one time.

I also do not believe that collecting data about which windows are currently open is a useful exercise. Presumably the purpose of collecting these strings is to see if a player is running a cheating program that is not currently on Blizzard’s banning list. If the program was on the banning list, it could be detected by Blizzard by comparing the hash of the first 10-20 bytes of the cheating program to the hashes of known cheating programs. In my opinion, by capturing the window text of each toolbar window Blizzard hope to catch cheaters who are “obviously” using cheating programs – i.e. if one of the windows is named WoWCheatTool, then this player will be assumed to be cheating and will be banned. I think this is a facile approach. People who write cheating programs will naturally be aware of the Warden and how it functions. So one of the first things they will do is to make sure the window text of the cheat tool in the toolbar is given a name that has no cheating connotations, rendering the collection of window text data useless.

The Warden has been classified as spyware by some, including the Electronic Frontier Foundation (EFF) who brandished it as spyware and said its use constituted "a massive invasion of privacy".

The Anti Spyware coalition defines spyware as:

“Technologies deployed without appropriate user consent and/or implemented in ways that impair user control over:

Material changes that affect their user experience, privacy, or system security;

Use of their system resources, including what programs are installed on their computers; and/or collection, use, and distribution of their personal or other sensitive information.”

The Warden does collect sensitive data and can affect a user’s privacy. However, it is not deployed without the user’s consent - in the WoW license agreement, there is a “consent to monitor” clause which states that the game may monitor a player’s memory to detect “unauthorised third party” programs. The full text of the license agreement can be obtained here: http://www.worldofwarcraft.com/legal/eula.html.

Critics such as the EFF argue that this information is “buried in license agreements which few people read”, but this is the case with almost all types of software. Players who are concerned about privacy issues will read license agreements in detail and choose whether or not to proceed with installation. Others will simply accept these terms as they wish to play the game and are not concerned about any privacy issues.

Warden summary

The Warden, like PB, is based upon monitoring of player’s machines to ensure they are cheating. This is an efficient and fast process which ensures players are not using known cheating tools.

The Warden has aroused controversy in some quarters regarding the private nature of some of the information it captures from player’s machines. I do have some concerns regarding the possibility that the Warden could be modified to capture sensitive data in the clear, but I feel the issue of collection of sensitive data depends on an individual’s trust in Blizzard. If a player believes Blizzard’s word and thinks they are trustworthy and will not capture sensitive personal data in cleartext, they will feel comfortable with the Warden’s actions. However it must also be noted that privacy concerns are not a major issue for most players. Many players in fact welcome the use of the Warden, as it helps to catch cheaters who impair the gaming experience of fair players. Fair gameplay is more important to these gamers than any potential privacy issues.

9 Novel cheating detection mechanisms

Game statistics

Game publishers could gather statistics about elements of their games in order to help with detecting cheats. This is similar to the concept of threshold detection used by some anomaly based intrusion detection systems, in which certain behavioural attributes, such as the number of times a file is accessed, is recorded, and if it exceeds a certain threshold value an alarm is raised.

In a FPS game, statistics on each player's total number of shots fired and the number of hits registered can be used to calculate the percentage of successful shots a player fires. This statistic could be used to detect aimbots. Aimbots give players "superhuman" aiming abilities, allowing them to automatically shoot directly at a target. Any player using an aimbot is likely to have a very high shot success percentage, perhaps 95% or more. If such a statistic is calculated any players who have a shot success percentage of 95% or over could be flagged as a potential cheat.

Economic statistics

A novel way of detecting that gold duping bugs are being exploited by cheats is to examine economic statistics from the virtual world. When gold duping occurs, the supply of gold in the virtual world increases, and this consequently reduces the virtual money to real money exchange rate. Websites exist which track the exchange rate between real money and virtual gold, such as GameUSD (www.gameusd.com). The currency devaluation may also be detected by seeing whether the price of gold on gold selling websites (such as IGE) has decreased. Everquest suffered from a gold duping bug which caused hyperinflation in its virtual economy. The impact that gold duping and the resultant hyperinflation have on an MMORPG can be catastrophic. Hyperinflation causes the price of virtual items to rise astronomically, with basic items costing huge sums of gold. The high price of basic items means that players who are new to the game and possess little gold, cannot afford to buy anything, making it impossible for them to progress in the game. Analysing economic statistics to detect any devaluation of game currency can be used to detect the symptoms of a gold duping bug, but the statistics that are used must be reliable.

10 Virtual crime

10.1 Virtual economies

The phenomenon of virtual economies was first discussed by Castronova [Ca02]. His research focused on measuring in economic terms the value of virtual economies. He studied the Everquest MMORPG and found that the Gross Domestic Product (GDP) of the Everquest virtual world was larger than Bulgaria's. This may seem completely bizarre – how can a computer game world have a GDP greater than a country with approximately 7 million people?

Everquest players take on a profession within the game, and once they have acquired sufficient skill in their profession they can produce virtual items which can be traded with other players. For example, iron ore can be smelted into iron, then a sword made out of it, which can be sold. This mirrors economic activity in the real world, and leads to "virtual world GDP".

However the relationship between virtual and real economies runs deeper than this. Players of MMORPG invest considerable time and effort in order to accumulate virtual items and move up levels. To use an age-old adage, "time is money". Virtual game items possess real economic value – players who don't wish to spend hours playing the game in order to get virtual items or progress levels, can buy these for "real" money. For example, ILevelU http://www.ilevelu.com will increase a player's level from level 1 to 50 in the Lineage II MMORPG for a price of $325. There have been cases of virtual items being sold for thousands of dollars – an island in the Project Entropia game was sold for $26,500! It has been estimated that worldwide annual sales of virtual items exceeded $800m in 2005 – so virtual item trading is big business and will only continue to grow as MMORPG grow.

Making money from MMORPG

In some MMORPG, virtual item trade is encouraged and supported. A good example of this is Second Life (http://secondlife.com). The makers of Second Life, Linden Labs, allow any virtual items created by a player to belong to that player. This allows players to trade and profit from virtual items as they please. Linden Labs' approach to virtual items is in contrast to other game developers, who explicitly state in their End User Licence Agreements that virtual property belongs to them.

Many game publishers have attempted to ban virtual item trading in their games, as many legitimate players believe it gives an unfair advantage to richer players. After much pressure from game publishers, eBay, the largest trader of online game articles, agreed to ban virtual item trading in 2007. However, this has not stopped virtual item trading from flourishing in other places.

Much of virtual item trade is done through middlemen, with Internet Gaming Entertainment (IGE), being the largest. IGE has an agreement with Blizzard to sell gold and levelling facilities for WoW. IGE has around 100 suppliers – experienced gamers who accumulate virtual items and sell them to IGE. IGE's customers are players wishing to buy virtual items or move up levels. Lots of other less legitimate middlemen websites appear and disappear with regularity. These websites are less scrupulous about the source of their virtual items. Using middlemen makes it easy for players to indulge in what is known as Real Money Trade (RMT) – making real money through playing MMORPG and selling the virtual items accumulated to the middleman in return for cash.

The potential for RMT through playing MMORPG has attracted the attention of people eager to make a fast buck. This has led to criminals looking to exploit MMORPG and their players. This section will focus on how criminals attempt to profit from virtual economies, and how information security can help to prevent this.

10.2 Stealing virtual property

Rather than spending many hours playing the game trying to accumulate virtual items, why not steal them off someone else? The easiest way to do this is to capture someone's online identity, by stealing their User Id and password. Once these are obtained, that player's character can be stripped of their all possessions which can then be transferred to the thief and sold.

How identities are stolen

Phishing

[We07] describes phishing as “the act of sending an e-mail to a user falsely claiming to be an established legitimate enterprise in an attempt to scam the user into surrendering private information that will be used for identity theft.”

[Hu05] gives an example of a phishing email used to steal account details from players of the Eve MMORPG , shown in full below.

From: Eve Team "suspension@eve-online.com" Date: September 3, 2005 10:20:26 PM EDT To: hunterd@wharton.upenn.edu Subject: Limited Account Access - Eve-online

We are contacting you because on 2 Sep 2005 our Account Review Team
identified some unusual activity in your account. In accordance
with Eve's User Agreement and to ensure that your account has not
been compromised, access to your account was limited. Your account
access will remain limited until this issue has been resolved.

To secure your account and quickly restore full access, we require
you to login in you account .This process is mandatory, and if not
completed within the nearest time your account may be subject for
suspension or will be banned

To securely confirm your Eve-Online information please click on the
link bellow:

[Redacted to avoid mistaken logins]

We encourage you to log in and perform the steps necessary to
restore your account access as soon as possible. Allowing your
account access to remain limited for an extended period of time may
result in further limitations on the use of your account and
possible account closure.

Thank you for using Eve-online!
The Eve Team

The email contains what seems to be a perfectly valid request, and looks extremely authentic. The from: address in the email has been spoofed to make it appear that it comes from the makers, Eve. Clicking on the link led to the following URL: www.portatildirecto.com/secure.eve-online.com/login.htm which was not a website run by Eve online. This URL contained a faked login screen in which the user's credentials were entered. These were then stolen by the phisher and used to gain access to the player's account.

Shoulder surfing

Many MMORPG players, especially in Asia, play in Internet cafes. According to [CS04], there have been cases of Internet cafes installing hidden cameras to monitor players when they entered their login credentials. This is a more sophisticated form of traditional "shoulder surfing" which occurs when someone observes a user entering their credentials.

Malware

Malware, such as Trojans and keyloggers are used by attackers to steal player's credentials. [CS04] note the case of twenty players having their credentials stolen in an Internet café after the owner of the café had installed keyloggers on the machines.

Trojans have been specifically created to obtain credentials for MMORPG. An example is Infostealer.Lingling.B which tries to obtain the user's ID, password and WoW server name, and then sends this information to a website. See http://www.symantec.com/security_response/writeup.jsp?docid=2007-030914-4716-99&tabid=2 for further information. More sophisticated Trojans, such as InfoStealer.Multigame, exist that work to steal credentials for a number of different MMORPG. To make information useful to the attacker, this Trojan sends the player's level and amount of virtual money to the attacker, allowing them to focus on player's accounts with large amounts of virtual money.

A group of Korean hackers were arrested after stealing 50,000 user ids by installing Trojans on websites they had compromised. By visiting these websites users would inadvertently be infected with the Trojan. According to [So05], one of the perpetrators is estimated to have made $150,000 from the scheme.

These examples serve to illustrate the popularity and potency of malware as a method to steal MMORPG player's credentials. Players can be infected with malware without their knowledge, and have their credentials stolen from under their nose.

10.3 Other virtual crimes

Virtual item trading problems

This sub-section discusses a few scams that have been used by people indulging in virtual item trade. Sellers in unsanctioned auctions would "sell" an account to a player, receive payment, and immediately change the password on the account and contact the customer service department to regain control of the account. The player who paid the seller was left empty-handed.

Another scam involves winning bidders who would be "stood up" by the seller who would not appear to complete the transaction. With unsanctioned auctions, there is no accountability for sellers to actually deliver what they claim to be selling.

In unsanctioned auctions of items, what is bid on is not always what is received. A dishonest seller might offer one sword for sale, but after the exchange of money takes place, a player receives a completely different one - not the one that was bid on.

Farming – sweatshops

Due to the ease at which virtual items can be traded for real money, there is a clear incentive for people to attempt to accumulate as many virtual items as they can. This has led to the rather alarming growth of "virtual sweatshops". These are gangs of people employed specifically to play MMORPG for often little money, with the sole purpose of farming as many virtual assets as possible. Some of this farming is done for RMT, to supply middlemen with gold. Other sweatshops are run for the purpose of providing gold and levelling facilities for players, whilst for some profit is the goal.

The sweatshops which provide levelling use their employees to take the player's character and play the game for them until a higher level, (specified by the player when he purchases the levelling) is reached. This is done by playing the character constantly for a fixed period to increase the player's level or amount of gold as soon as possible.

Game publishers and players frown upon virtual sweatshops - although playing the game 24 hours a day, 7 days a week is not a violation of the rules, virtual sweatshops can spoil enjoyment for other gamers. Sweatshops which provide levelling and gold facilities are considered unfair by legitimate players, as a player's financial position can be used to advance them through a game. There are also concerns of exploitation of people who work in virtual sweatshops.

Virtual mugging

This term was first coined when a Chinese man was arrested in 2005 on suspicion of using a bot to carry out a series of muggings in the Lineage II MMORPG. He used a bot to beat up and rob characters, and then sold their possessions for real money. There have also been reports of virtual extortionists who threaten weaker players into handing over virtual or real protection money to avoid negative consequences.

10.4 Conclusions

Bruce Schneier [Sc05] noted that "every form of theft and fraud in the real world will eventually be duplicated in cyberspace", and from the examples I have described, he may be right. Fraud in virtual item trading represents a serious problem for game publishers. Players who lose confidence in virtual item trading may quit the game as they may not have the time to progress through the game, and cannot buy virtual items to help them progress, due to fraud. Due to RMT, the incentive to steal player accounts is large, which has led to a number of security threats to MMORPG players, such as phishing and malware. These threats erode player confidence as they may not want to become targets of online criminals. It is vital for game publishers to address these issues to make sure that these threats do not spiral out of control and player confidence is maintained. The next section discusses some measures that might be used to achieve this.

11 Virtual crime prevention

This section focuses on how information security measures can be used to help prevent virtual crimes.

11.1 Preventing Identity Theft

Players log into MMORPG using a username and password authentication mechanism. This is a traditionally weak authentication mechanism, for a number of reasons. WoW gives players the following advice when choosing passwords:

- **Do not share the account information with anyone. This includes both the User Name and password. Most compromised accounts turn out to be the work of friends, family, guild members, or others that you "thought" you could trust.**
- **Do not write your password down. If you must, then do not keep it where others can find it. Places such as your wallet, under your keyboard, or with your World of Warcraft CDs are not good places to store it.**
- **Choose a secure password. Your password should be difficult to guess, but easy for you to remember. Here are some tips to make your password more secure:**
- **Be certain to use a combination of numbers and letters (a mix of upper and lower case letters is recommended).**
- **Avoid using words found in any dictionary or names of friends, family, pets, etc.**
- **Avoid using your character or account name in your password.**
- **Never share the "Secret Question" and answer you defined when creating your account with anyone.**

This is basic password advice, but there is no enforcement of it in WoW. There is no requirement for a player to choose a password which is a combination of letters and numbers; they can choose any password they like, as long as it is greater than 6 characters long.

Many users choose weak, easy to guess passwords – passwords that are short in length and common words or names and hence are very easily guessed by attackers. Password based authentication mechanisms are also vulnerable to Trojan programs, which can be designed to steal passwords. I believe that using a username and password authentication mechanism, with no enforcement of password strength, is a wholly inadequate way to protect player accounts. Although Blizzard emphasise that it is the user's responsibility to choose a strong password, when a user's account is compromised, according to [Wo07] "It takes an extensive amount of time for Blizzard's Customer Support Staff to investigate and attempt to resolve the consequences of an account compromise".

I believe that by using a stronger authentication mechanism, Blizzard could save time and money investigating account compromises, as stronger means of authentication should reduce the instances of account compromise. A few alternative authentication mechanisms are discussed below.

Smart cards

Smart cards are tamper resistant cards which possess microprocessors and memory that can be used for secure storage. Smartcards could be used in conjunction with a password to provide two factor authentication for MMORPG players. The smartcard could contain the user's account number, and a default password, loaded by the game publisher. This default password is also emailed to the user. When the user first wishes to play the game, they must insert the smartcard into a card reader, and enter the default password. This ensures that the correct person is in possession of the smartcard. Once authenticated, they are free to choose their own password.

Whenever the user wants to play the game, the user must insert the smartcard into a card reader, plus enter their password. If the player's smartcard is lost or stolen, they can report this to game support, and their account can be suspended until a replacement is sent. If an attacker wishes to compromise the player's account, he must physically obtain the smartcard as well as guess the password. This makes compromising a player's account harder, as he has two barriers to cross to gain access to the account.

There are obvious problems with this method. The first is cost. This method is far more costly than password based authentication, as smartcards have to be bought and distributed, as well as replaced in the event of theft or loss. This could be mitigated by making players liable for replacement costs. For this scheme to work, players would need to possess smart card readers, which come at a price. This can be prohibitive for many players, who may decide to switch to another MMORPG which does not require smart cards. Many players play in Internet cafes, and if their Internet café does not have any smart card readers, they will be unable to play.

One time passwords

This method involves the use of password generators, such as an RSA SecurID token. The password generator generates dynamic passwords, which change periodically. The generator is synchronised with the MMORPG server. When a player wishes to log in, they must enter their User ID and password, as well as the password currently displayed on their password generator. By using a dynamic password, if passwords are obtained by a Trojan they are of limited value, as the password is different every time a user logs in, hence the term one-time. This prevents old passwords being re-used. Passwords cannot be guessed or brute forced either, as the password generator produces random, unpredictable passwords.

This method requires players to possess a password generator token, which could be included in the price of purchasing the game. It is also a portable solution, as the player can simply take the password generator anywhere they like and be authenticated. It does not rely on Internet cafes to possess any special equipment, unlike the smart card approach.

Password policies

Policies that force the user to change their password periodically could also be enforced, for example by not allowing them to logon to the game until they have changed their password. Better enforcement of a user's password choice is another method that can be used to strengthen account security. This can be done simply and quickly by including validation code in the software that checks that the user's chosen password has a mixture of letters and numbers, and is of a minimum length. Although this method can help users to choose stronger passwords, it does not overcome the inherent weaknesses of a password authentication mechanism.

User awareness

Educating players about the need to ensure they have strong passwords can go a long way to preventing stolen player accounts. If players are made aware of the need to have strong passwords, and how to choose them, they are likely to choose stronger passwords. Advice can be give on how to choose strong, but easy to remember passwords. Research by [YB00] has shown that a good method of choosing strong but easily recalled passwords is to use passphrases, which are passwords formed by choosing certain features of well known phrases. For example, if the first letter of each word of the phrase "the quick brown fox jumps over the lazy dog" is chosen, the password becomes "tqbfjotld" – an easy to remember but hard to guess password.

Online scanning

A mechanism that can be used to prevent malware from stealing user ids and passwords could be to provide a scanning facility from within the MMORPG, maybe as part of the game client. For example, when a player starts an MMORPG, the client could scan the player's machine for any keyloggers, Trojans, viruses or worms, in the same way anti-virus software operates. I believe this is a good and viable solution to malware. By performing an automatic scan, players without anti-virus software, or with little knowledge about malware, can be alerted to the presence of malware on their machines, without having to do anything whatsoever. This is of great benefit to players who play in Internet cafes – if someone has compromised their machines then players are forewarned.

The privacy issues of the MMORPG scanning the entire contents of a player's machine may pose a problem for some players. However, anti cheating tools, such as the Warden, are more invasive, as they send data about processes and memory to a remote server. In the case of malware scanning, there should be no transfer of data – the scan should take place on the player's machine and there should be no transfer of data. Another difference is that malware scanning is designed to help players, and most players will realise this.

Phishing prevention

The most effective way to stop players becoming victims of phishing attacks is to educate and make them aware of how phishers operate. There are a number of pieces of advice which can be given:

1. Encouraging players to learn to recognize fraudulent phishing emails – emails that are almost always not personalised, asking for user id's and passwords, being aware of unexpected messages that require urgent action to avoid "imminent problems" with an account.
2. Warning players not to click on links in e-mails from unknown senders. Instead players should only navigate to an address that is known to be authentic or a support number that is listed on the publisher's website.
3. Security updates for browsers should be downloaded regularly. This will help prevent potential hackers from gaining access to information by exploiting known security issues.

MMORPG publishers state that they never ask for a player's password either in email or telephone communication, and emphasise this fact on their websites. If players can assimilate this knowledge the instances of phishing to obtain player's passwords should be reduced. An approach which could be used is to periodically display a popup during game play with a message stating (for example) "Blizzard never asks for user id and password in correspondence – any communication which does is NOT from Blizzard".

Although this may annoy some, it should help to get the message across to other less aware players.

11.2 Virtual item trading schemes

Many MMORPG publishers have tried to use litigation to close down middlemen websites which trade virtual items, with limited success. Despite some players disliking RMT as giving an unfair advantage, the fact remains that there is a huge demand to purchase virtual items.

Using middlemen websites has led to a number of problems such as non-delivery of items and virtual sweatshops, discussed previously. This section discusses a scheme, StationExchange, that has been implemented by Sony for their MMORPG Everquest, which I believe can be used as a model for safe virtual item trading.

StationExchange

[So07] define Station Exchange as "the official Sony Online Entertainment auction service that provides players a secure method of buying and selling the right to use in game coins, items and characters in accordance with SOE's license agreement, rules and guidelines. This service provides players a way to take part in this growing secondary market in a more secure manner."

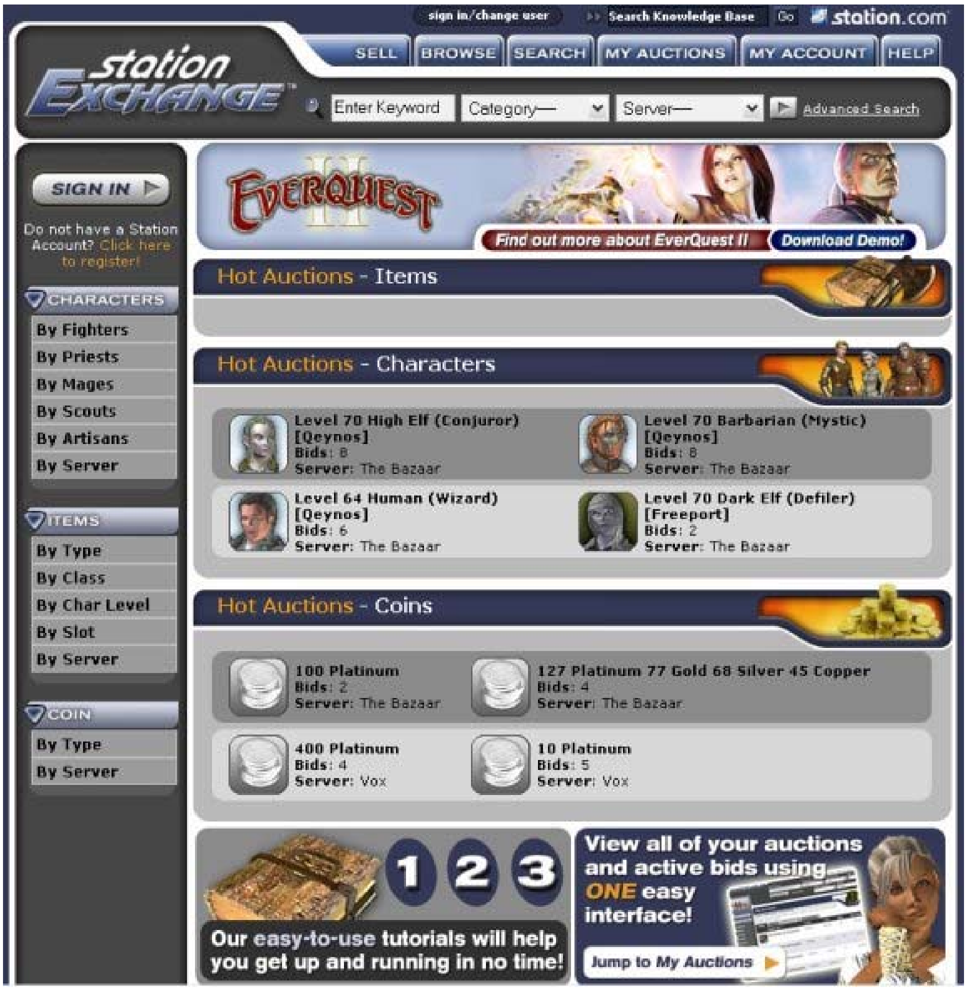
StationExchange was developed in response to the growing number of virtual item frauds that were taking place involving EverQuest players. 40% of Sony's customer service calls were to deal with virtual item disputes. Also, by developing a secure trading environment, Sony could encourage growth in virtual trade, and cash in on the lucrative RMT market themselves.

How it works

When a player wants to sell an item, that item is immediately removed from the game world, and stored in a StationExchange server. It is then listed as an item for auction, and cannot be accessed by the seller. The Station Exchange server will provide statistics on the item or character that is listed so that potential bidders can see exactly what they will be bidding for.

When an auction is won, the successful bidder receives a notification and makes payment, which is conducted via the StationExchange server through PayPal. The item is then transferred to the buyer's game account, and sent to the buyer as an email attachment.

The StationExchange interface



Level 70 High Elf (Conjuror)
[Qeynos]
Bids: 8
Server: The Bazaar

Level 70 Barbarian (Mystic)
[Qeynos]
Bids: 8
Server: The Bazaar

Level 64 Human (Wizard)
[Qeynos]
Bids: 6
Server: The Bazaar

Level 70 Dark Elf (Defiler)
[Freeport]
Bids: 2
Server: The Bazaar

100 Platinum
Bids: 2
Server: The Bazaar

127 Platinum 77 Gold 68 Silver 45 Copper
Bids: 4
Server: The Bazaar

400 Platinum
Bids: 4
Server: Vox

10 Platinum
Bids: 5
Server: Vox

StationExchange acts as a trusted third party (TTP) in the exchange between buyer and seller. StationExchange is trusted as it is owned by the game publisher. This is in contrast to middlemen like IGE. If there was fraudulent behaviour perpetrated by Sony, their reputation would be affected, and so would EverQuest as a result. Middlemen do not have this concern, as most of them are fly-by-night operations. Both buyer and seller have confidence that they will not be defrauded by Sony, as it is in Sony's interest to encourage and secure virtual item trading, as they take a slice of each sale.

As the TTP stores the item until the amount has been paid, there is no chance that the seller can renege on the deal, a problem that occurred in the past. Also, as the item is withdrawn from use when it is offered for sale, it cannot be altered by the seller to reduce its value. For example, if a player account was being sold, it cannot be altered by selling gold to make the character of less value.

Players who do not like the fact that virtual items can be bought and sold do not have to participate – StationExchange is only on two of Sony's Everquest servers, so players who don't wish to trade can play on any of the remaining Everquest servers.

If a TTP based trading scheme is the only way players can trade virtual items, and other virtual item trading websites are shut down, the impact of player identity thefts can be reduced. If criminals who steal players' accounts are only able to sell the items through the game publisher's trading scheme, when a player realises they have been compromised they will be able to inform the publisher and should have their items returned to them. With less trustworthy middlemen there is no redress in this situation, as the middlemen have no formal connection with the game.

Using a TTP based trading scheme should reduce the instances of virtual item trading fraud. Sony have experienced a 30% fall in virtual item trading related calls to their support helplines since adopting StationExchange. I believe using a TTP is a win-win situation for both players and game publishers. Publishers gain financially by promoting trade, and players who wish to trade can do so securely.

11.3 Preventing other virtual item crimes

Virtual mugging

The instances of virtual mugging that have been documented have been performed by bots, which can issue player fight commands quicker than human opponents, and hence can defeat them easily. Therefore bot detection methods, such as scanning of player machines, can be used to detect virtual muggers. Auditing and logging facilities can also help. Important game related information needs to be stored for a period of time so that complaints which arise can be investigated.

Auditing and logging

With the amount of virtual item fraud and theft that occurs in games, it is imperative that game publishers keep track of virtual items – who owns what, who has traded to whom and so on. Daily snapshots of the game databases should be taken and kept for a period of time to help with complaint resolution. This type of information is necessary to investigate and verify a player's complaints. For example, if a player claims they have been mugged by another, then several pieces of evidence are required, such as which player mugged him and which of his possessions were stolen.

This kind of evidence will only be available if there is adequate logging and storage of game data. Associating unique identifiers with each virtual item can help in this respect. If each virtual item that a player possesses, along with their identifiers, are stored in a database, when a mugging occurs before and after snapshots of the databases can be compared to see where the virtual items that belonged to the player have gone.

Auditing and logging are post detection mechanisms. Apart from real time bot detection mechanisms (such as player monitoring approaches discussed previously), information security can do little to prevent virtual muggings.

Measures to prevent virtual sweatshops

It is not easy to catch virtual sweatshops at work. There is only one real tell-tale sign that might be used for detection, which is looking for accounts played constantly.

Monitoring the usage on accounts can reveal this kind of behaviour. An account which is logged in for a few hours a week, which suddenly starts being used 24 hours a day, and then drops back to a few hours a week, is an obvious sign.

A threshold detection mechanism which automatically alerts administrators to accounts being used 24 hours a day can be a useful way to flag suspicious behaviour for further investigation.

Another method that can be used to counter virtual sweatshops is to legitimise virtual item trading and bring it within the auspices of the MMORPG itself, like Sony did with StationExchange. If virtual items are traded through the game, the game publisher has more control over these trades. As virtual sweatshops exist to make money, they will trade frequently. The game publisher can notice sellers who are trading often, and monitor them more closely if they have suspicions. Analysing amounts of certain items being traded may also reveal item duping bugs. To profit from item duping, players need to sell their items in exchange for real money. If unusually large amounts of an item are being traded, this may point to the presence of a duping bug. This was how Sony discovered the presence of a gold duping bug in Everquest – they noticed large amounts of gold being sold on StationExchange. Without bringing trade within the publisher's control, such a trend cannot be spotted.

Bringing virtual item trade within the game should also reduce the number of middlemen. Publisher backed trading schemes offer more security to buyers than middlemen, which should reduce the number of players using middlemen and thus reduce virtual sweatshops who supply the middlemen.

Virtual item trading is big business. Julian Dibbell, author of Play Money: Or, How I Quit My Day Job and Made Millions Trading Virtual Loot, estimates the size of the virtual goods market at $880m annually, and growing. Game publishers have attempted to ban virtual item trading without much success. The financial incentives open to players and middlemen are simply too great, and policing the Internet for virtual item sellers is too great a task. I believe that rather than trying to ban virtual goods trading, game publishers should encourage it by using the approach taken by Sony with

StationExchange. By controlling virtual trade themselves, they can profit from the lucrative market. They should also be able to reduce the instances of virtual fraud that occur with the current way of virtual item trading, and be able to spot and close down virtual sweatshops more easily. I believe game publishers must also take greater steps to reduce identity theft. MMORPG player accounts seem to have become a soft target for criminals. Game publishers must be more proactive in dealing with this problem as it will only continue to grow as the market for virtual goods grows.

12 Conclusion

I hope this project has helped people to appreciate the size and scale of the MMOG industry and the challenges it faces. In particular I hope this project has provided a comprehensive overview of the issues of virtual crime and cheating, and how information security solutions can help to overcome the problems MMOG publishers face.

I believe I have met the objectives I set down in the introduction to this project. This project has explained Yan's fifteen categories of cheating, which represent a thorough classification of the types of cheating that exist in all types of MMOG. Methods used to cheat in MMORPG have been discussed in detail, and measures to prevent these cheats have been described and evaluated. The nature of virtual economies in MMORPG, and virtual crimes that take place in these MMORPG, have been described and countermeasures to prevent virtual crimes have been proposed. I have analysed real-world approaches taken by game publishers to catch cheats, PunkBuster and the Warden. My analysis of PunkBuster was reliant upon information from the developer, EvenBalance, and I would have liked to have found some independent information rather than relying on a sole source. However, due to the proprietary nature of PunkBuster, this was not possible. Likewise, my information about how the Warden operates was obtained from a sole source – Greg Hoglund, who had take time to reverse engineer it. I placed trust in his analysis, and I felt that this was well placed as he is a respected author of a number of software security books.

This project has shown the problems with creating massively distributed systems. The design of such systems is complicated as thousands of different state interactions need to be considered, and the distributed nature of the system can lead to race conditions which may be exploited by attackers. MMOG systems are the forerunners of other very large distributed systems of the future. With the growth of business conducted via the Internet, I believe we will see massively distributed business systems in the future, where thousands of clients systems will allow the thousands of employees of an organisation to interact with a single business system. The lessons learned from MMOG systems can be applied to help the design and development of these distributed systems of the future.

I can see virtual crime, in particular player identity theft, continuing to rise in the near future. I have not seen many initiatives by game publishers to reduce the problem of player identity thefts, and if steps are not taken the problem will inevitably get worse. I think that a new approach by publishers to RMT needs to be adopted. Instead of trying to stop it, they should embrace it, like Linden Labs have done with Second Life. By embracing RMT, publishers can gain financially and increase player confidence in RMT. Players who don't like RMT can be catered for by allowing them to play on non-RMT servers where RMT is not permitted. I believe creation of safe virtual item trading schemes will increase the number of players participating in MMOG as they will see an opportunity to make money through playing games. However this will also serve to increase the threat of players cheating – if money can be made, players will look for ways to cheat that enable them to cash in.

I foresee a continuing arms race between cheaters and game publishers. Increasingly sophisticated approaches to cheating, such as kernel resident bots will become more prevalent as cheaters attempt to avoid detection, and publishers will have to keep up with these advances if they are to keep their games free of cheats and cheaters. It is vital that publishers look to improve their cheating detection mechanisms constantly, as the future of their game depends on it. Games which are pervaded by cheaters will cause disillusionment amongst fair players and cause them to leave the game, which could prove fatal for the publisher's business.

I hope this project has helped to educate people about MMOG - a subject that many people may have heard of, but not had a great deal of understanding about. The MMOG industry has huge potential to expand even further in the coming years; with more people connecting to the Internet, and with better hardware giving an enhanced gaming experience for players, both of which should mean a new crop of players. Information security has a key role to play in ensuring the MMOG industry continues to grow, by ensuring the threats that could hamper its growth are dealt with effectively.

13 References

[AP06] Associated Press: 3 Chinese tried for counterfeiting weapons in online game, *May 2006*, http://newsinfo.inquirer.net/breakingnews/infotech/view_article.php?article_id=19103

[Bl07] Blizzard Entertainment: World of Warcraft passes 8 million subscribers, *January 2007*, http://www.blizzard.com/press/070111.shtml

[Ca02] Castronova, E.: On Virtual Economies, *CESifo Working Paper Series No. 752*, July 2002.

[CJ06] Chen, K-T, Jiang, J-W, Huang, P, Chu,HH, Lei,C-L, Chen,W-C.: Identifying MMORPG bots, *Proceedings of the 2006 ACM SIGCHI international conference on Advances in computer entertainment technology*, Hollywood, California, 2006.

[Co98] Collberg, C, Thomborson, C, Low, D.: A taxonomy of obfuscating transformations, *Technical Report 148, Department of Computer Sciences, The University of Auckland*, July 1997.

[CS04] Chen, YC, Chen, P, Song, R, Korba, L.: Online Gaming and Security Issue – Cases and Countermeasures from Taiwan, *Proceedings of the 2nd Annual Conference on Privacy, Security and Trust*, Fredericton, New Brunswick, Canada, October 2004.

[DL05] Daum, M, Lucks, S.: Attacking Hash Functions by Poisoned Messages "The Story of Alice and her Boss", *June 2005*, http://www.cits.rub.de/MD5Collisions/

[Ev06] EvenBalance: PunkBuster for players, *June 2006*, http://evenbalance.com/publications/wr-pl/index.htm

[FC05] Feng, W, Chang, F, Feng, W, Walpole, J.: A Traffic Characterization of Popular On-line Games, *IEEE/ACM Transactions on Networking Volume 13, Issue 3*, June 2005.

[Fe07] Feng, W.: Got Mips? The need for speed in online games, *June 2007*, http://www.thefengs.com/wuchang/work/cstrike/GotMIPS.pdf

[HN06] Harris, J, Nardi, B.: Strangers and Friends: Collaborative Play in World of Warcraft, *CSCW 2006 Proceedings of the 2006 20th anniversary conference on Computer Supported Cooperative Work*, Banff, Alberta, Canada, November 2006.

[Ho05] Hoglund, G.: 4.5 Million copies of EULA-compliant spyware, *October 2005*, www.rootkit.com/blog.php?newsid=358

[Hu05] Hunter, D.: Virtual world phishing, *September 2005*, http://terranova.blogs.com/terra_nova/2005/09/virtual_world_p.html

[La07] Lavish Software: A guided tour of InnerSpace, *June 2007*, http://www.lavishsoft.com/innerspace/index.php?page=1

[MH07] McGraw, G, Hoglund, G.: Exploiting Online Games: Cheating Massively Distributed Systems, Addison Wesley, 2007.

[Mm07] Glider: An introduction, *June 2007*, http://www.mmoglider.com/

[Pl06] PlayNoEvil Game Security News & Analysis: Virtual item theft ring busted, *December 2006*, http://playnoevil.com/serendipity/index.php?/archives/1051-Virtual-Item-Theft-Ring-Busted.html#extended

[Sc05] Schneier, B.: Schneier on Security, *August 2005*, http://www.schneier.com/blog/archives/2005/08/stealing_imagin.html

[Si06] Silkroad Tavern: Info on Bots, *October 2006*, http://www.silkroadtavern.com/forums/index.php?s=ee3d925859b58e5fffdc34cdabe1de02&showtopic=24101

[So05] Sophos: Suspected gang who stole from online game players arrested in Korea, *July 2005*, http://www.sophos.com/pressoffice/news/articles/2005/07/va_krarrests.html

[So07] Sony Online Entertainment: Station Exchange FAQ, *July 2007*, http://stationexchange.station.sony.com/faq.vm

[Wa07] Blizzard Entertainment: World of Warcraft game guide, *June 2007*, www.wow-europe.com/en/info/basics/guide.html

[We07] Webopedia: Phishing definition, *July 2007*, http://www.webopedia.com/TERM/p/phishing.html

[Wh04] D. Wheeler, IBM Secure programmer: Prevent race conditions, *October 2004*, http://www.ibm.com/developerworks/linux/library/l-sprace.html

[Wo07] World of Warcraft Community site: In-Game support, tips to prevent account compromise, *July 2007*, http://www.blizzard.com/support/wowgm/?id=agm01889p

[Ya02] Yan, J, Choi, HJ. : Security Issues in Online Games, *The Electronic Library, Vol. 20, No. 2*, 2002, pp. 125-133.

[Ya05] Yan, J, Randell, B.: Security in Computer Games: from Pong to Online Poker, *Technical Report Series CS-TR-889, School of Computing Science, Newcastle University*, UK, February 2005.

[YB00] Yan, J, Blackwell, A, Anderson, R, Grant, A.: The memorability and security of passwords – some empirical results. *Technical Report UCAM-CL-TR-500, Cambridge University*, 2000.

14 Bibliography

14.1 Electronic sources

Useful background information on MMORPG

http://en.wikipedia.org/wiki/MMORPG

World of Warcraft European homepage

http://www.wow-europe.com/en/info/basics/

Makers of Inner Space

http://www.lavishsoft.com/innerspace/index.php?page=1

Anti-cheating website with descriptions of cheating and anti-cheating measures

http://www.counter-hack.net/

PunkBuster FAQ

http://www.evenbalance.com/index.php?page=faq-cod.php

Home of Station Exchange

http://stationexchange.station.sony.com/

Background information on virtual sweatshops

http://www.techdirt.com/articles/20051209/021252_F.shtml

Dissertation writing tips

http://lorien.ncl.ac.uk/ming/Dept/Tips/writing/thesis/thesis-intro.htm