2011

A study on trading scams in massively multiplayer online role-playing games and risk mitigation techniques

Timothy Lee Meyer
Iowa State University

A study on trading scams in massively multiplayer online role-playing games and risk mitigation techniques

by

Timothy Lee Meyer

A thesis submitted to the graduate faculty

in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Co-majors: Information Assurance

Computer Engineering

Iowa State University

Ames, Iowa

2011

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ACKNOWLEDGEMENTS

I would like to take this opportunity to thank those who helped me with my work throughout my research. First, I would like to thank my major professor Dr. Yong Guan for all of his work and advise over the last semester in progressing towards the completion of my thesis. His knowledge and time spent working with me over the last semester has been greatly appreciated.

I would also like to acknowledge Dr. Joseph Zambreno and Dr. Phillip Jones for being committee members on my thesis work. Whether it was classwork, senior design project, research or competitions both of you have always been more than willing to spend your time to help me out in any situation, and for that I thank you.

ABSTRACT

Scamming in massive multiplayer online role-playing games (MMORPG) has become a significant issue in gameplay today. With the multitude of MMORPG games to choose from and the millions of users playing these games, there are significant opportunites for scammers to take advantage of unsuspecting users. The research first creates a taxonomy of online trading scams for aiding developers to prevent scamming in games. This taxonomy focuses on the targets, objectives, and strategies scammers use in MMORPGs. Secondly, it introduces a scam mitigation method to reduce opportunities for scams within the MMORPG trading environment. The mitigation method implements three services: Barter, Market, and Contract. Using the services can provide a trustworthy venue for trading, prevent common scams, and provide better overall gameplay for current and future users.

CHAPTER 1. OVERVIEW

1.1 Introduction

In the last decade, massive multiplayer online role-playing game's (MMORPGs) popularity has risen drastically. In 2010, World of Warcraft (WoW) boasted 12 million subscribers worldwide.[10] MMORPGs, such as WoW, allow players to assume the role of a character, while striving to advance the players skills, wealth, and belongings. This occurs in a persistent world where the game continually evolves whether or not the player is online. With a large number of players within one centralized environment, it has opened up a new channel of social interaction in online gaming. Current figures for online gaming show that MMORPGs have become a multi-billion dollar industry, with projections for 14 billion dollars in revenue by 2014.[1]

With the rise of popularity in MMORPGs, scamming and other malicious actions have also become part of the gaming environment. Game developers have had to mitigate threats of key loggers, viruses, and other malicious programs being spread around to take advantage of the virtual and real economies created by the MMORPGs.[4] Problems have arisen with games like Ultima Online and Second Life, where real world currency can be exchanged for virtual currency. Other games have spawned black markets where virtual currency or virtual items can be acquired or sold for real world currency. With an estimated future 14 billion dollar business, there is high potential to illegally acquire monetary gain through virtual-real trade.

In this thesis, scams in MMORPGs are reviewed in order to create a taxonomy of trading scams. This taxonomy is used to provide insight on how trading scams operate, and the similarities of differences of scams. Then, a method to mitigate scamming techniques is introduced to be placed into the trading mechanisms of MMORPGs. Chapter 2 gives background information on MMORPG gameplay and problems associated with it. Chapter 3 gives information on current MMORPG scams, motivation, strategies, related work, and provides the trading scam taxonomy. Chapter 4 discusses current protocols and defenses against scamming attempts. Chapter 5 introduces the scam mitigation method and its properties. Chapter 6 discusses the text-based implementation of the mitigation method. Chapter 7 discusses the implementation of the scam mitigation method into a MMORPG open-source game. Chapters 8 discuss special considerations for MMORPGs and the scam mitigation method. Chapter 9 provides the summary and future work related for the scam mitigation method.

CHAPTER 2. MMORPG BACKGROUND

2.1 Gameplay

MMORPGs use client-server system architecture in order to maintain a virtual world that operates at all times, independent of user login. Client software is used to provide a way for players to connect to the central service, either by using an installed program on the computer or a web browser applet for easy connection to gameplay. In MMORPGs, users play the role of a character, and control the character's interaction with the environment. The major difference with MMORPGs from traditional role-playing games is the significant number of other players interacting with each other in a single persistent game environment.

2.2 MMORPG Business Models

Games typically operate under three business models: free-to-play, freemium, and pay-to-play. Free-to-play revenue structures provide gameplay for free, while obtaining a revenue stream through marketing and advertising. Freemium model offers participation in the MMORPG gaming environment, but places restrictions on the user. Freemium business model is quickly growing as the model of choice by social networking sites and mobile gaming media.[11] These restrictions limit gameplay, whether through access to game areas, or the ability to perform advanced game operations in order to persuade the user to purchase advanced features. Lastly, the pay-to-play model typically charges the user a monthly fee to access the game environment.

2.3 Motivation to Acquire Virtual Goods

Similar to the real world, players spending time with gaming accounts and virtual characters place a personal value on their account, in much the same way someone would place value on personally making upgrades to their home. Vili Lehdonvirta, a researcher on virtual goods, explains that "As in the physical world, objects take on meaning based on perceived status, scarceness, and emotional value."[1]

This conclusion points to the fact that as a user spends more time towards advancing a character or account within a game, the user perceives virtual items of having real world value. This also introduces the black market economy, wherein individuals do not want to spend the time to acquire the wealth, instead they purchase virtual assets. The demand comes from individuals searching for an easier way to acquire virtual wealth, and malicious users creating a real world profit by supplying the demand through illegal channels or black markets. [2]

2.4 Economic Problems

2.4.1 Definitions

Virtual currency is in-game currency used for buying, selling, or trading virtual items.

Virtual wealth is the combination of all in-game items and virtual currency.

Real world currency is currency used outside of the virtual game environment.

2.4.2 Problems

Today, security professionals are concerned about the trade of virtual currency for real-world currency. In China, popular virtual currency Q coins have become so widely used that government officials have become concerned about it challenging the physical currency's legitimacy.[1] Government officials must ensure that virtual currency will not cause economic stability issues, while maintaining the ability for consumers to use both forms. New problems have arisen from the government's ability to tax income made off of virtual items being sold. Officials are struggling to apply tax laws to these virtual currency to real world profit situations.

Many other concerns with virtual currency come from government problems with controlling and regulating services like online gambling within MMORPGs. When games have a direct connection of virtual currency to real world currency like in Second Life, money laundering becomes difficult to track when fake contact information can easily be used to create an account.

Cases have shown that virtual assets are not being sold for small amounts of money. Recent examples have shown virtual islands and space stations in Project Entropia and virtual cities in Second Life selling for $30,000, $100,000, and $50,000 respectively. [7] This kind of wealth has led to sweat shops or ”gold-farmers” where groups of individuals acquire virtual assets to sell in order to make real world profit.

CHAPTER 3. MMORPG SCAMS

3.1 Importance

It is important to focus on trade scamming to MMORPGs because of the current relevance in criminal activities. It is now commonplace for players to be exploited, leading to them having their accounts or items stolen and sold on the black market. It is also common for new players to buy virtual currency and items to advance in the game. This allows them to avoid spending the normal time required to acquire the items. Players use of black markets to acquire virtual wealth is showing that as game worlds evolve and grow, and as the complexity of user interactions and game environment interactions increase the overall scope of gameplay is harder to control. [3]

3.2 Motivation

In trade scamming, scammers are usually invested in acquiring three types of virtual wealth.

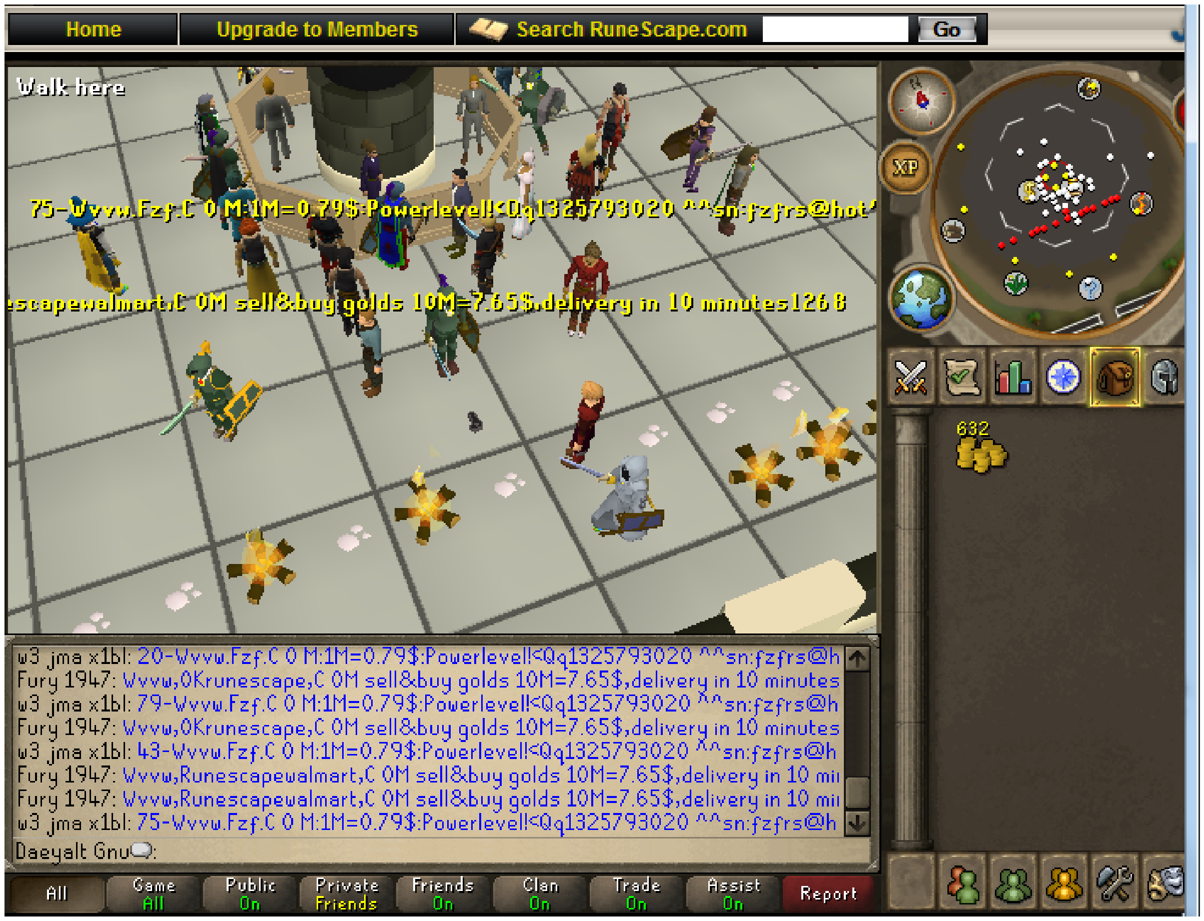
1. Virtual Wealth

2. User Accounts / Characters

3. Competitive Advantage

3.2.1 Virtual Wealth

Scamming virtual currency and items from other users has the best risk to reward ratio in MMORPGs. If a user is reported for scamming and found guilty of breaking the game's terms of service (TOS), game moderators will ban the user's account. Since items or currency are easy to transfer from player to player within games, as once these items are acquired they can be laundered through alternate accounts or immediately sold to other players before the scamming event can be reviewed. In order to provide a great gaming experience to all users, the available items within the persistent gaming environment must be kept in balance. However, this leads to higher level players having a higher likelihood of obtaining rare items, thus creating a black market for lower level players to buy items they cannot easily acquire. This creates a economic environment where scamming can produce virtual gains, which can lead to real world monetary gain.



```
75-Wvvw.Fzf.C 0 M:1M=0.79$:Powerlevell<Qq1325793020 ^^sn:fzfrs@hot
scapewalmart.C 0M sell&buy golds 10M=7.65$.delivery in 10 minutes1268

w3 jma x1bl: 20-Wvvw.Fzf.C 0 M:1M=0.79$:Powerlevell<Qq1325793020 ^^sn:fzfrs@h
Fury 1947: Wvvw,0Krunescape,C 0M sell&buy golds 10M=7.65$,delivery in 10 minutes
w3 jma x1bl: 79-Wvvw.Fzf.C 0 M:1M=0.79$:Powerlevell<Qq1325793020 ^^sn:fzfrs@h
Fury 1947: Wvvw,0Krunescape,C 0M sell&buy golds 10M=7.65$,delivery in 10 minutes
w3 jma x1bl: 43-Wvvw.Fzf.C 0 M:1M=0.79$:Powerlevell<Qq1325793020 ^^sn:fzfrs@h
Fury 1947: Wvvw,Runescapewalmart,C 0M sell&buy golds 10M=7.65$,delivery in 10 mi
Fury 1947: Wvvw,Runescapewalmart,C 0M sell&buy golds 10M=7.65$,delivery in 10 mi
w3 jma x1bl: 75-Wvvw.Fzf.C 0 M:1M=0.79$:Powerlevell<Qq1325793020 ^^sn:fzfrs@h
Daeyalt Gnu:
```

Figure 3.1 Bots promoting a blackmarket web site.

3.2.2 User Accounts / Characters

Furthering the idea of real world monetary gain introduces scamming to acquire player accounts and passwords for the ability to sell the account to a new player. Accounts sold to players typically are acquired through black market web sites by either bots (computer programs performing in-game actions in order to acquire wealth and advance player skill levels) or individuals. Player accounts typically have the highest average monetary value of all of the black market MMORPG items sold.

Players commonly want to buy accounts that have the skill levels, items, and wealth already obtained (to a specific level dependent on the account's price) for two main reasons. First, the new user does not want to spend the time to advance the character within the game, but just wants to take part in the higher-level gameplay that excites the user. Common accounts sold to this buyer are known as "pure". A pure account is an account that has all of the skills advanced to a certain level, without completing any tasks or quests the game has to offer. These pure accounts sell at a higher level because the player can participate in all the game has to offer immediately. Secondly, statistics have shown [5] that users commonly play MMORPGs with real-world friends, colleagues, or relatives. This leads to users wanting to buy an account/character that is at the same level as their friend instead of ruining the gameplay for the other while the player tries to increase to a similar level of gameplay.

3.2.3 Competitive Advantage

Lastly, trade scams can be used to allow the scammer to gain competitive advantage over another player or competitor. This can be the most challenging area for game moderators to punish users for improper behavior. It is difficult to differentiate between skillful gameplay and scamming in different gameplay situations. Competitive advantage scamming refers to the ability to acquire advancement of a player's in-game skills by defrauding another player in the process. This becomes a crucial avenue for scammers in order to create high-level accounts to sell on the black market.

3.3 Cheating in Online Games

Previous work in cheating in online games has been published at a ACM workshop.[9] The topics in that paper can correlated to trading scams taking place in MMORPGs. Trading scams in MMORPGs can be categorized into these areas:

1. Exploiting Misplaced Trust
2. Collusion
3. Abusing Game Protocol
4. Artificial Intelligence
5. Altering Game Client
6. Account Takeover
7. Virtual Assets

Exploiting Misplaced Trust cheating involves a scammer earning the trust of a player over a period of time in order gain advantage. Alternatively, the trust could be immediately assumed due to physical appearance, contact information, status, etc. Collusion trading scams are defined by a scammer using multiple accounts or multiple individuals to induce an advantage through numbers to harm another player. It must be stated that not all collusion is prohibited in MMORPGs. Abusing Game Protocol or timing scams are defined by a scammer either avoiding penalties/losses or gaining wealth by not fully completing actions or trading sequences. This allows malicious players to use race conditions to give them virtual items without giving any items up. Artificial Intelligence scams are defined by a scammer using hardware/software programs in order to gain advantage on a player. Altering Game Client scams are defined by a scammer altering a game client in order to send or show altered information to the game server or other client. Account Takeover scams are defined by a scammer acquiring account passwords and/or recovery questions.

3.3.1 Cheating Related to Virtual Assets

The last category in the trading scam overview was Cheating Related to Virtual Assets. Here, [9] discusses the potential to offer a virtual item for real money but never deliver on the agreement. However, Yan and Randell only view this as a fairness violation in the game where the vulnerability is due to the player and not the underlying game system. This shows their assumption that game developers cannot develop a better gameplay system to protect player vulnerability. The scam mitigation method introduced in Chapter 6 will go on to show that properly designed protocols can handle player vulnerabilities and reduce trading scams.

3.4 Taxonomy Categories

The taxonomy focuses on three main categories to place each individual scam:

1. Target - Identifies the entity being exploited
2. Objective - Identifies the object being acquired
3. Strategy - Identifies the plan of action to achieve a successful scam

3.5 Classification by Target

When a scam is formulated, there are multiple entities that can be targeted in different game environments. A target is defined as selecting entities based on traits such as game demographics, gameplay or social behavior. The definition includes that the scammer typically will initiate communication between themselves and the target. A scammer gains an advantage by finding a weakness in the target and tailoring the scam's objective and strategy to reflect the knowledge of the target.

In-game targets : A scammer targets entities within the gameplay in order to carry out the scam.

Environment : The environment consists of all aspects of the virtual world where gameplay takes place. Typically scammers use the environment to broadcast information that is used in the scam to lure targets, or target parts of the gameplay environment itself.

Non-Player Character (NPC) : NPCs can be used to acquire wealth through automated NPC actions and responses.

Gameplay : Using additional software add-ons or artificial intelligence to gain advantage over other players.

Chat : Using in-game communication channels to acquire personal information from players.

Player / Customer : Consists of a scammer targeting a player or group of players to carry out the scam.

Individual : Targeting individuals in situations where the malicious user can gain their personal wealth.

Nave/Novice : Targeting unsuspecting or inexperienced players by using knowledge of gameplay as an unfair advantage.

Greed : Targeting users lured into a disadvantageous situation in order to profit from them.

Group : Targeting clans or individuals with ties to a group where the malicious user exploits one or more individuals with the scam.

Clan : Communicating with leaders of a group with intentions of defrauding the clan (association).

Pyramid : Using multiple targets, whether an individual or group, to recruit more targets for a wide-spread scam.

Outsider targets : A scammer targets entities outside of gameplay that have direct or indirect connection to the gameplay or game infrastructure.

Game Developer/Studio : A scammer targets the game studio / developer, or infrastructure.

Game Protocol : Altering game protocol methods in order to gain benefit from transaction with server protocol.

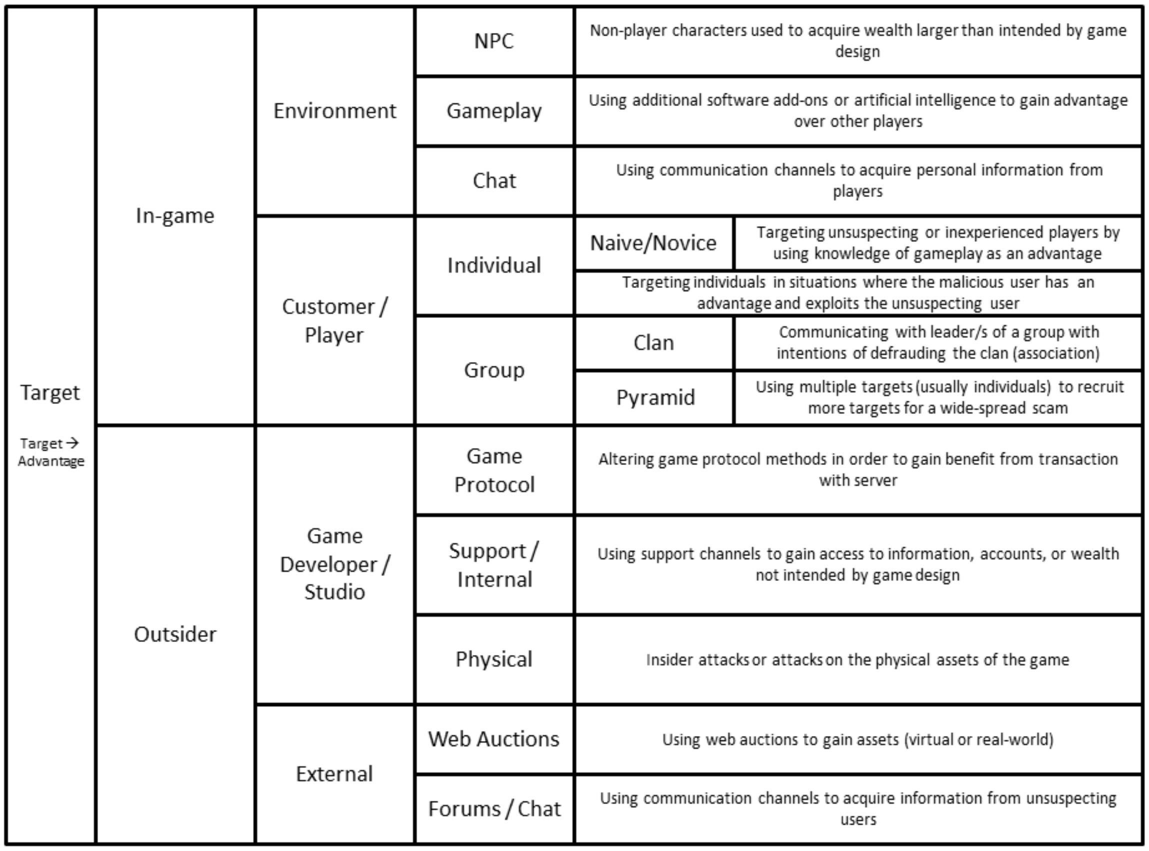
Support / Internal : Using support channels to gain access to information, accounts, or wealth not intended by game design. (Includes insider attacks).

Physical : Attacks on the physical assets of the game (Network, Code, Database, Distribution) allowing for an unintended advantage to the scammer.

External : The scammer uses venues commonly visited by MMORPG players that are not under the control of the game studio/developer.

Web Auctions : Using web auctions or other online trading sites to gain virtual or real assets.

Forums / Chat Rooms : Using communication channels to acquire information from unsuspecting users.



Target Target → Advantage	In-game	Environment	NPC		Non-player characters used to acquire wealth larger than intended by game design
			Gameplay		Using additional software add-ons or artificial intelligence to gain advantage over other players
			Chat		Using communication channels to acquire personal information from players
		Customer / Player	Individual	Naive/Novice	Targeting unsuspecting or inexperienced players by using knowledge of gameplay as an advantage
					Targeting individuals in situations where the malicious user has an advantage and exploits the unsuspecting user
			Group	Clan	Communicating with leader/s of a group with intentions of defrauding the clan (association)
				Pyramid	Using multiple targets (usually individuals) to recruit more targets for a wide-spread scam
	Outsider	Game Developer / Studio	Game Protocol		Altering game protocol methods in order to gain benefit from transaction with server
			Support / Internal		Using support channels to gain access to information, accounts, or wealth not intended by game design
			Physical		Insider attacks or attacks on the physical assets of the game
		External	Web Auctions		Using web auctions to gain assets (virtual or real-world)
			Forums / Chat		Using communication channels to acquire information from unsuspecting users

Figure 3.2 Graphical view of the Target Classification.

3.6 Classification by Objective

A scam can be formulated without specifying an objective, but successful scams identify what object or information it is attempting to acquire. An objective is defined as the desired result of the scam as set by the malicious user. The objective does not have to be defined as either success or failure, as it could include a success rate of performing the operation over time. The objective represents the end-goal, but can include tactical objectives that are intermediate steps to achieving the overall objective. The objective is an important part of the scammers risk assessment where the risk of punishment by game moderators can be outweighed by the gain.

Wealth : Acquiring valuable resources, virtual or real, or gaining control of the ownership, transportation, or storage of such assets.

- **Real-world Currency** : Create real-world profits from virtual assets not intended by developers.
- **Virtual Currency** : Advance player's overall wealth within the game environment.
- **Items** : Acquiring virtual assets through improper channels or methods.
  - **Quantity** : Obtaining quantities of items that could not be done in the specific timeframe without giving up similar value in return.
  - **Rare** : Obtaining rare, sometimes valuable, items without spending the required time to acquire them or giving up similar value in return.
  - **Cost** : Obscuring, inflating, or misrepresenting the value of an item in order to profit from its sale.

Account : Acquiring account information, username/password, or other information that allows for control of a user's game account.

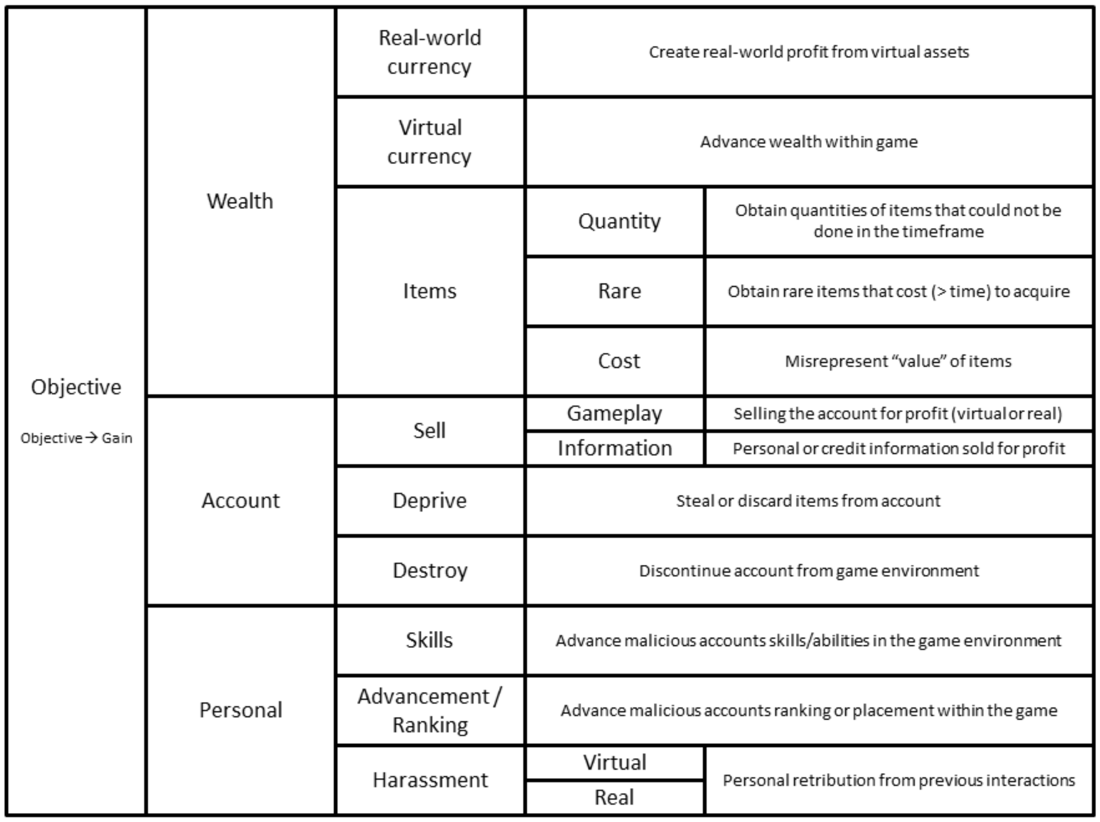
- **Sell** : Acquiring game accounts to sell of profit. (virtual or real-world)
  - **Gameplay** : Selling the account on a forum or web auction site for profit.
  - **Information** : Personal or credit card information sold for profit.
- **Deprive** : Stealing or discarding items from a game account.
- **Destroy** : Discontinuing gameplay or service to an account from the game environment.

Personal : Acquiring assets related to the player's in-game skill sets or ranking. Includes harming other individuals through types of harassment (virtual or real) or account degradation.

Skills : Advancing the malicious user's skills and/or harming the victim's skills within the game environment.

Advancement / Ranking : Advancing the malicious user's ranking and/or harming the victim's ranking within the game environment

Harassment : Personal retribution from previous interactions that result in virtual or real harassment or violence.



Objective Objective → Gain	Wealth	Real-world currency		Create real-world profit from virtual assets
		Virtual currency		Advance wealth within game
		Items	Quantity	Obtain quantities of items that could not be done in the timeframe
			Rare	Obtain rare items that cost (> time) to acquire
			Cost	Misrepresent "value" of items
	Account	Sell	Gameplay	Selling the account for profit (virtual or real)
			Information	Personal or credit information sold for profit
		Deprive		Steal or discard items from account
		Destroy		Discontinue account from game environment
	Personal	Skills		Advance malicious accounts skills/abilities in the game environment
		Advancement / Ranking		Advance malicious accounts ranking or placement within the game
		Harassment	Virtual	Personal retribution from previous interactions
			Real	

Figure 3.3 Graphical view of the Objective Classification.

3.7 Classification by Strategy

A scam's most visible characteristic is the strategy formulated using knowledge of the target to acquire the objective. A strategy is defined as a plan of action created and designed to achieve a specific outcome while being able to adapt when circumstances or environments change. The strategy attempts to exploit the weakness or vulnerability of the target through careful planning. Classifying strategies for a taxonomy becomes difficult, because for an effective strategy to survive and flourish in many MMORPGs, it must adapt to different settings. Therefore, the classification by strategy category focuses on the typical venues the strategy follows to achieve its outcome without citing specific events.

- **Social Engineering** : Manipulating individuals or groups into performing actions or divulging confidential information. [12]
  - **Personal Information** : Acquiring information about a victim's account that allows a malicious user to defraud them.
    - **Account Takeover** : Using social engineering in order to capture username/ password or security recovery questions and answers to control the game account.
  - **Confidence Trick** : Process of gaining a user's trust or inflating a user's confidence in order to defraud the victim at a later time. [13]
    - **Deceptive Representation** : Techniques to deceive user into believing they are receiving wealth/item/account that is worth more than its actual value.
    - **Pyramid Scheme** : Non-sustainable model promising participants payment for enrolling others into the scheme, believing that the end result will benefit all parties. [14]
    - **Luring** : Promises made to victim in order to place them in an environment where they normally would not be. This places the malicious user has advantage in order to defraud the victim.
  - **Collusion** : Agreement between two or more players, at least one being malicious, where the terms of service (TOS) of the game are broken by deceiving, misleading, or defrauding others to obtain wealth, assets, or an unfair advantage. This must be against the TOS to be considered a scam.
    - **Market Competition** : Using multiple parties to drive the market value of an item or product to a level that is beneficial to the malicious users.

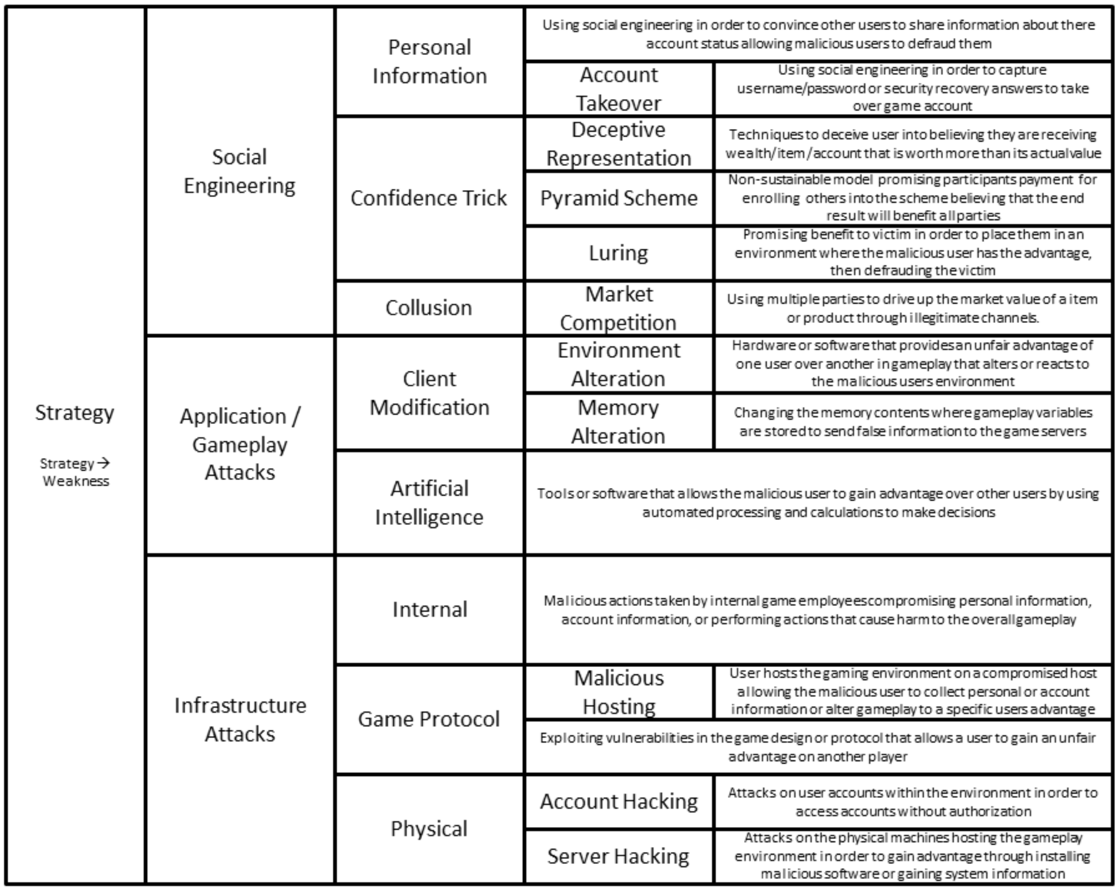
Application / Gameplay : Altering gameplay environment or introducing programs to gain an unfair advantage over other players.

- **Client Modification** : Alternations made to the MMORPG client allowing the malicious user to gain an advantage over other players.
  - **Environment Alteration** : Hardware or software that provides an unfair advantage of one user over another in gameplay that alters or reacts to the malicious users environment.
  - **Memory Alteration** : Changing the memory contents where gameplay variables are stored to send false information to the game servers.
- **Artificial Intelligence** : Tools or software that allows the malicious user to gain advantage over other users by using automated processing and calculations to make decisions.

Infrastructure : Using methods to alter, change, or destroy gameplay data or protocols in order to gain advantage over other players.

- **Internal** : Malicious actions taken by internal game employees compromising personal information, account information, or performing actions that cause harm to the overall gameplay.
- **Game Protocol** : Exploiting vulnerabilities in the game design, protocol, or infrastructure that allows a user to gain an unfair advantage on another player.
  - **Malicious Hosting** : User hosts the game environment on a compromised server allowing the malicious user to collect personal or account information or alter gameplay to a specific users advantage.
- **Physical** : Attacks on the physical machines, networks hosting the gameplay or the network services connecting users to the network or servers.
  - **Account Hacking** : Attacks on user accounts within the environment in order to access accounts without authorization.

Server Hacking : Attacks on the physical machines hosting the gameplay environment in order to gain advantage through installing malicious software or gaining system information.



Strategy Strategy→ Weakness	Social Engineering	Personal Information		Using social engineering in order to convince other users to share information about there account status allowing malicious users to defraud them
			Account Takeover	Using social engineering in order to capture username/password or security recovery answers to take over game account
		Confidence Trick	Deceptive Representation	Techniques to deceive user into believing they are receiving wealth/item/account that is worth more than its actual value
			Pyramid Scheme	Non-sustainable model promising participants payment for enrolling others into the scheme believing that the end result will benefit all parties
			Luring	Promising benefit to victim in order to place them in an environment where the malicious user has the advantage, then defrauding the victim
		Collusion	Market Competition	Using multiple parties to drive up the market value of a item or product through illegitimate channels.
	Application / Gameplay Attacks	Client Modification	Environment Alteration	Hardware or software that provides an unfair advantage of one user over another in gameplay that alters or reacts to the malicious users environment
			Memory Alteration	Changing the memory contents where gameplay variables are stored to send false information to the game servers
		Artificial Intelligence		Tools or software that allows the malicious user to gain advantage over other users by using automated processing and calculations to make decisions
	Infrastructure Attacks	Internal		Malicious actions taken by internal game employees compromising personal information, account information, or performing actions that cause harm to the overall gameplay
		Game Protocol	Malicious Hosting	User hosts the gaming environment on a compromised host allowing the malicious user to collect personal or account information or alter gameplay to a specific users advantage
				Exploiting vulnerabilities in the game design or protocol that allows a user to gain an unfair advantage on another player
		Physical	Account Hacking	Attacks on user accounts within the environment in order to access accounts without authorization
			Server Hacking	Attacks on the physical machines hosting the gameplay environment in order to gain advantage through installing malicious software or gaining system information

Figure 3.4 Graphical view of the Strategy Classification.

CHAPTER 4. CURRENT DEFENSE MECHANISMS

4.1 Research

Game developers have been responding to the risks of scamming since MMORPGs were first released, and have spent significant resources in order to stop scams from occurring to their players. It is in the game developer's best interest to stop scammers in order to maintain high player satisfaction and minimize player turnover. In creating a review of the current defenses being used by popular MMORPGs two things must be understood. First, most for-profit MMORPG developers do not release information about how scamming and fraud is handled internally. This creates a limitation on the knowledge of current defenses used. Secondly, most open-source MMORPGs spend minimal resources on protocols to prevent scamming techniques.

4.2 Defense Categories

The defensive strategy focuses on three main categorizes to handle scams:

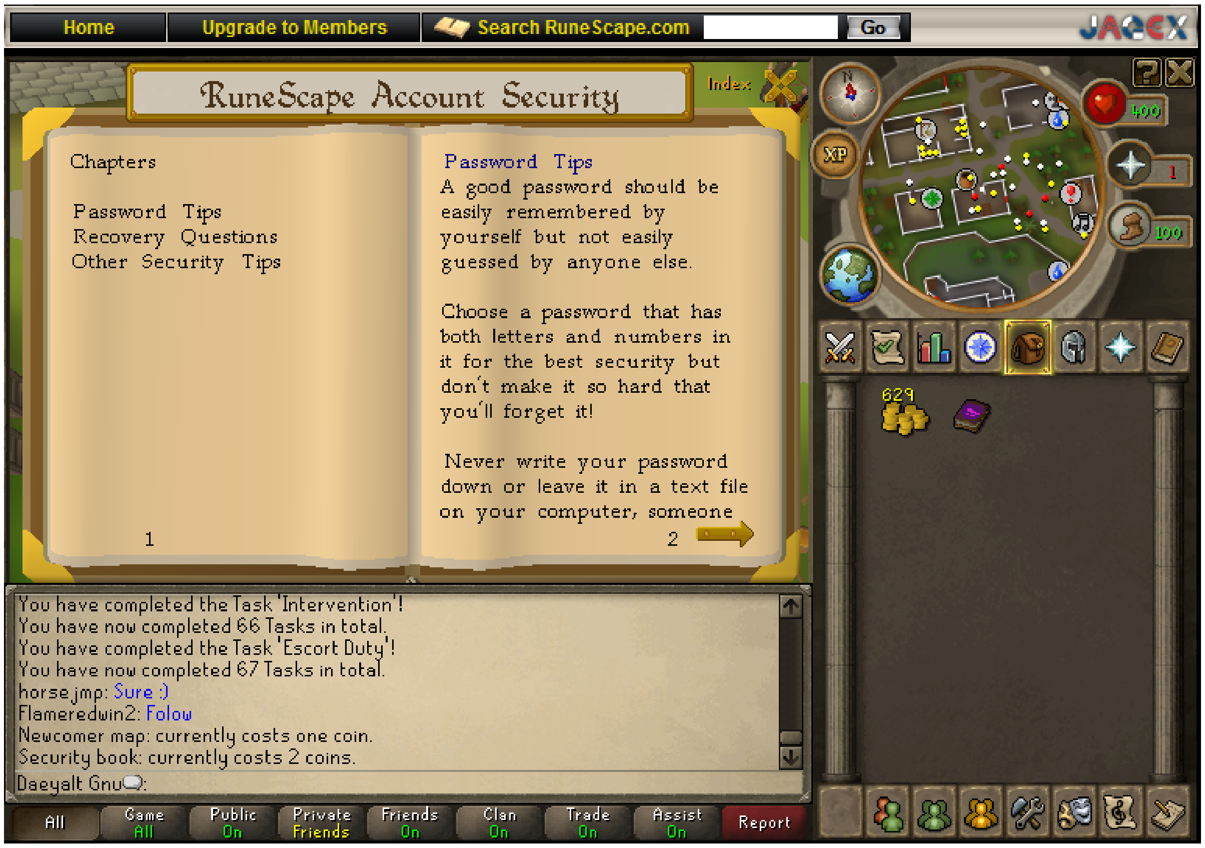
Prevention Focus on stopping scams before they affect users

Mitigation Placing services or protocols to limit the effects on users to reduces overall risk

Remedy Performing infrastructure, protocol changes or data recovery

4.2.1 Defense by Prevention

The primary prevention technique used by developers for scam prevention is through education. Many MMORPGs will introduce the user to the game environment with an in-game tutorial teaching the user how to perform basic operations wherein they have incorporated teaching lessons on common scams that affect new users. Runescape provides users an in-game book that has details on typical scams that a user would face, instructions on how to avoid them, and corresponding punishments for participating in actions that violate the terms of service.[16]



RuneScape Account Security

Chapters

Password Tips
Recovery Questions
Other Security Tips

1

Password Tips
A good password should be easily remembered by yourself but not easily guessed by anyone else.

Choose a password that has both letters and numbers in it for the best security but don't make it so hard that you'll forget it!

Never write your password down or leave it in a text file on your computer, someone

2

```
You have completed the Task 'Intervention'!
You have now completed 66 Tasks in total.
You have completed the Task 'Escort Duty'!
You have now completed 67 Tasks in total.
horsejmp: Sure :)
Flameredwin2: Folow
Newcomer map: currently costs one coin.
Security book: currently costs 2 coins.
Daeyalt Gnu:
```

Figure 4.1 Security Guide in Runescape

Other techniques used to prevent scams come from secure coding practices and monitoring systems for user protection. Blizzard, developer of World of Warcraft (WoW), has used several programs to ensure that users are protected from common attacks and that they are participating in the game on an even playing field with other users. First, the Blizzard Authenticator is a small device that enables a user to secure their account.[18] The authenticator creates a digital code during each login that the WoW authentication service verifies is created by the specific authenticator. After a user has registered the authenticator, it is required to produce a digital code every time the account logs onto the game servers.

Secondly, Blizzard uses an anti-cheating tool called Warden integrated into its games.[17] Warden uses API function calls to collect data about programs running alongside the game client, submits them to Blizzard's servers, and compares these to other programs running on other systems. Its use is intended to identify whether or not any programs are being used to create an unfair advantage for the user or if the game client has been altered. It can then be used to determine whether or not action should take place against a user violating the TOS. However, many complaints have arisen from users about privacy issues related to Warden's operation.

4.2.2 Defense by Mitigating Risk

MMORPGs have encountered real world situations turned into virtual problems. Scamming techniques have been successfully adapted to new virtual environments and MMORPGs developers have had to implement systems to handle these problems. The first and most basic of tactics taken by developers is the black-listing of accounts or IPs when abuse of the TOS is reported or logging has directly related an account to malicious actions. Many MMORPGs, including WoW, Eve Online, and Runescape, utilize employee moderators to handle problems occurring on game web sites, and even handle in-game situations when necessary. If customer support or a moderator finds the actions to be harming gameplay, they can place limitations on accounts or even ban accounts from gameplay.

In order to reduce bartering scams, services have been introduced to check item values (price checker) and the ability to prevent users from purchasing an item that they cannot use. Market services have also incorporated the item warnings and price checker services into the market GUI design.

Difficulties in handling scamming and gold-farmers occur when games are free to play or offer a free trial to play the game. This opens a venue for gold-farmers or bots to create new accounts, performs actions or scams, and move the wealth to storage before being discovered. This allows for a low risk, high reward system since if the accounts are reported or discovered, a new account can be created to replace the other account. To mitigate these problems games like Runescape have limited high value trades for new users. WoW has limited their trial accounts to not be able to trade or chat with other users until the full version is purchased. Runescape even prevents unbalanced trades to prevent gold-farmers from transferring all of their wealth to another player. Lastly, Eve Online has accepted and embraced scamming as part of the game environments culture, discussed in Chapter 9.

4.2.3 Defense by Remedy

MMORPGs have incorporated strategies on how to handle events where scams took place, were reported or discovered, and items or accounts were lost. First, when scams have taken advantage of improper game design, implementation or protocols, content patches have been released to fix the problems on both the server and client sides. Blizzard TOS states that if an incident is reported or discovered and the staff is able to identify the items or wealth lost, it will be returned to the user. However, the TOS states that they will only perform this action a limited number of times to prevent career victims.[15] Other developers have taken on a tactic to allow gold farmers to create accounts and operate, while monitoring their activity in order to locate the ring leaders or actual accounts being used to store or transfer virtual items.

CHAPTER 5. SCAM MITIGATION METHOD

5.1 Trade Protocol Advancement

This paper proposes a solution for the trading mechanisms found in most MMORPGs. While there is an assortment of services offered by various MMORPGs, the solution implements two services critical to most popular MMORPGs: barter and market trading. It also introduces a contract service not found in many MMORPG games. The barter service provides a method where users can exchange goods and/or currency to carry out trade between two entities. The market service provides a method where a user can place items for sale on an auction and other users can bid on the item until the sale ends. The contract service offers three subservices where a contract is created and posted, two entities agree on a service to be performed, and the resulting reward for the contract being completed.

5.2 Benefits

Currently, popular games like Runescape, WoW, and Guild Wars only offer trading mechanisms of barter and market trading. Eve Online is one of the only games found to have incorporated contracts into their trading mechanism. However, Eve Online has no form of prevention or punishment for breaching a contract with another player.

All three services can be implemented as a MMORPG trading service while maintaining virtual property protection for its users. Each service uses an escrow, where the game developer sets a trusted third-party within its backend infrastructure to receive and distribute virtual wealth to the appropriate users when a trade agreement is made. This system allows for not only proper transfer of items in bartering and market auctions, but also a contract verificiation system between users to ensure fairness.

The escrow service guarantees proper trading in all three services by doing the following:

1. Takes ownership of all items involved in the trade.
2. Reviews that both parties agree to requirements of the trade.
3. Validates that both parties meet the requirements of the trade.
4. Validates that both parties are following TOS and trading rules.
5. Distributes ownership of withheld items to correct individuals.

5.3 Infrastructure

In order to facilitate implementation that would be adaptable for use in future work, the infrastructure architecture used is defined as follows. The system is made of the following components:

- Server (Serv)
- Authentication Service (AuthServ)
- Escrow Service (EscServ)
- Database (DB)
- Game Clients (Client)

The game developer would be in control of all the components except the game client. The following sections focus on how the three services operate and the risk that can be mitigated by using the system.

The sets of components, users, items, and server inquiries in the barter server are defined as follows:

Components = (Serv, DB, EscServ)

Users = (ClientA, ClientB)

Items = (Item1, , ItemMax)

C is the set of communicating entities (Set C) where all of C has send/receive capability.

C = Components U Users

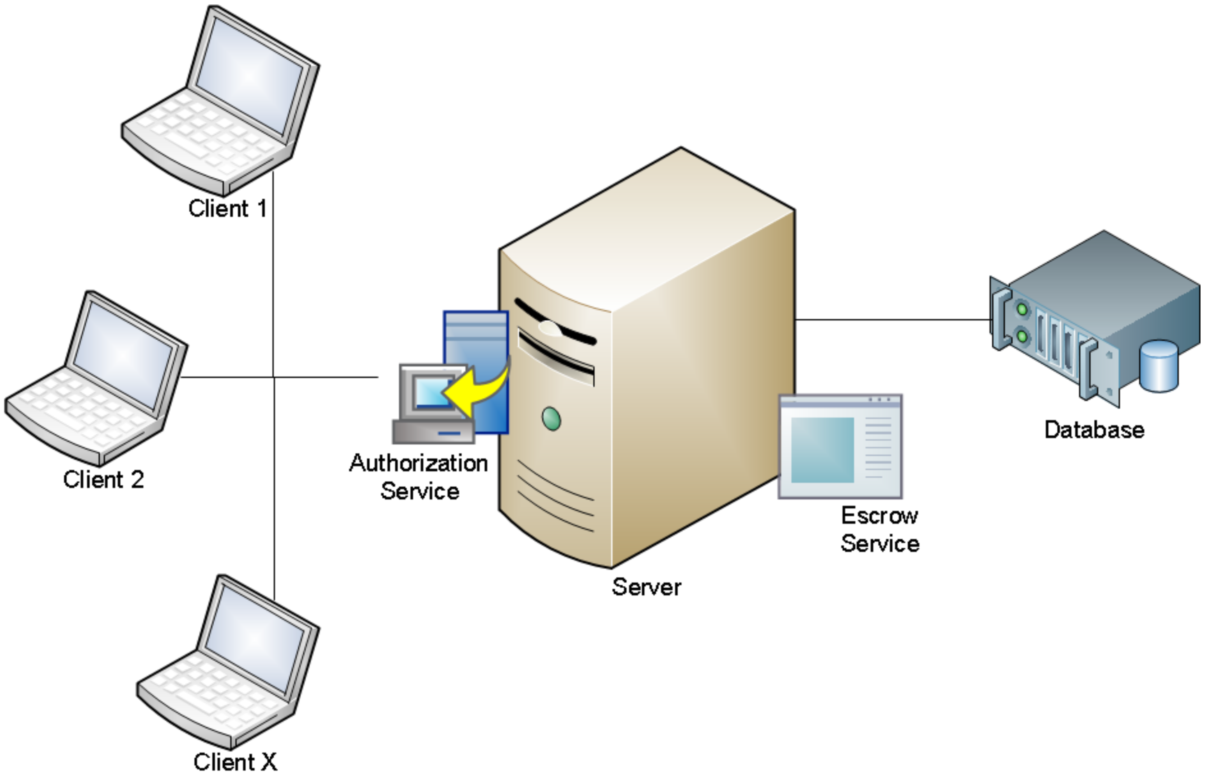


Figure 5.1 System Architecture Overview

The architecture of the system is shown in Figure 5.1. The system requires that the server and database be available at all times. Then, with the authentication and escrow service running on the server, the client is allowed to make a connection to the server. The authentication service must communicate with the connecting client, verify authentication credentials, and then establishes a connection to the database. The server locates the user's stored information, and all inventory and player attributes are loaded into the database for gameplay. The following sections make the assumption the system has started correctly, has run the AuthServ and EscServ, and placed individual user data into the DB.

5.4 Client Interaction

The different services have different number of game clients required to be involved in the process in order for the trading services to operate.

- Barter Service: Allows trade between 2 clients only, with unlimited number of trades
- Market Service: Uses 1 client to create an auction, with unlimited number of auctions where unlimited number of clients can bid on the auctions.
- Contract Service: Uses 1 client to create a contract, with unlimited number of contracts where one user can bid on a contract.

5.5 Barter Service

The following shows the operating sets used in describing the barter service. Set State denotes the states the barter protocol follows to complete its operation. Set Inquiry holds the server communications acceptable to and from the client. Set M holds the message contents available to be sent between entities.

State = (Waiting, CollectItems, AcceptTrade)

Inquiry = (”Accept Barter Request?”, ”Offer Items?”, ”Accept Offer?”)

M = ((Yes, No) U Items U Inquiry)

The formal specification of the barter service can be defined as a function using connection of clients A and B. The following function describes the operation of the barter service.

```
BarterProtocol(ClientA, ClientB)=
    If State=Waiting AND Inquiry = ”Accept Barter Request”
        If ClientA.M = ”Yes” AND ClientB.M=”Yes”
        Then {State = CollectItems}
        Else {Exit}
    If State=CollectItems AND Inquiry = ”Offer Items?”
```

```
        If ClientA.M = Items AND ClientB.M= Items
        Then {State = AcceptTrade}
        Else {Wait}
    If State=AcceptTrade AND Inquiry = "Accept Offer?"
        If ClientA.M = "Yes" AND ClientB.M = "Yes"
        Then {EscServ(Barter), Exit}
        Else If ClientA.M = "No" OR ClientB.M = "No"
        Then {Exit}
        Else {Wait}
```

The barter service handles direct trade interaction between two clients within the game environment. Since it is assumed any player can initiate the trade process, the first state in the barter process is to confirm the trade request is agreed upon by both clients. The second state in the process is to offer a medium where the client is able to place items, currency, or both up for trade. Once both clients have offered, the system moves to its final state. Both clients must approve of the barter, otherwise if either client declines the trade is immediately ended.

```
EscrowService(ClientA,ClientB, Barter)=
    If State=Begin
    If Remove ClientA.items = Successful
        Then {State = RemoveB}
        Else {Rollback,Exit}
    If State=RemoveB
        If Remove ClientB.items = Successful
        Then {State = Review}
        Else {Rollback,Exit}
    If State=Review
        If TradeReview() = Successful
        Then{Transfer Items}
        Else{Rollback, Exit}
```

If both clients accept the offer, the escrow service is initiated. The escrow service will then perform five actions to ensure proper transfer of virtual goods to both clients. First, the items from Client A's inventory are removed and placed into the escrow table. Second, the items from Client B are placed in the table following the same protocol. Third, the escrow service reviews the trade for improper trading properties (for instance unbalanced trades in Runescape). Lastly, after passing the review, the escrow changes ownership of the items, items are transferred to the proper accounts, and the trade is completed. If at any time one of the users disconnects, or there is a server side failure to acquire any of the items, the escrow will initiate a rollback. A rollback is a escrow service recovery method that returns all items to the original clients.

5.6 Market Service

The following shows the operating sets used in describing the market service. The market service is split into two functional entities: sell and buy. Sets StateS and StateB denotes the possible states market sell and market buy protocols can be in, respectiviely. Sets InquiryS and InquiryB hold the server question asked to the client for market sell and market buy, respectively. Set M holds the message contents available to be sent between entities.

StateS= (Waiting, SetPrice, Confirm)

StateB= (Waiting, BidOnItem, PlaceBid)

InquiryS = ("Sell Item?", "Bidding Info?", "Confirm Sell")

InquiryB = ("Buy Item?", "Offer?", "Confirm Buy")

M = ((Yes, No) U Items U Inquiry)

The formal specification of the market service can be defined as a function using connection of a single client. The market service handles clients placing items for sale to the entire game network in an auction format. The market is handled by splitting the service into two protocols: selling and buying.

When a client requests to sell an item on the market, the MarketSellProtocol is initiated. Then, necessary information for the market is taken in from the client: the item, minimum bid, starting price, sell now price, and auction deadline. Once the client confirms the information is correct, the escrow service is called to place the auction onto the market. Any information that is not a valid selection, whether an invalid item or invalid bidding information, the service retries to acquire the information, or exits if a request threshold has been met.

```
MarketSellProtocol(ClientA)=
    If StateS= Waiting AND Inquiry = "Sell Item?"
        If ClientA.M = Item
        Then {State=SetPrice}
        Else {Retry}
    If StateS=SetPrice AND Inquiry = "Bidding Info?"
        If ClientA.M = ValidInt(M)
        Then{State= Confirm}
        Else{Retry}
    If StateS=Confirm AND Inquiry = "Confirm Sell?"
        If ClientA.M = "Yes"
        Then {EscServ(MarketSell)}
        Exit
```

When a client requests to buy an item on the market, the MarketBuyProtocol is initiated. First, the escrow service is checked to see if there are any items currently available on the market, and displays this information to the client. Next, the client must select an item to bid on and verify that the selection is valid otherwise it will retry the client for an appropriate response. Once the item is selected, it will ask the client for a bid on the item, display the new bid to the client, and ask it to confirm that the information is correct. Once the information is confirmed, the escrow service is called to place a bid on the auction.

```
MarketBuyProtocol(ClientB)=
    PrintList(EscServ.Market)
    If StateB = Waiting AND Inquiry = "Buy Item?"
```

```
        If ClientA.M = ValidListNumber()
        Then{State = BidOnItem}
        Else{Retry}
    If StateB = BidOnItem AND Inquiry = "Offer?"
        If ClientA.M = ValidOffer()
        Then{State = PlaceBid}
        Else{Retry}
    If StateB = PlaceBid AND Inquiry = "Confirm Buy?"
        If ClientA.M = "Yes"
        Then{EscServ(MarketSell(Item, Price))}
        Exit
```

Once the seller or buyer has confirmed their submission to the market, the escrow service is initiated. For both services, buy and sell, the item must be removed from the client and placed into the escrow DB table. Then, the escrow will review the item to ensure that the offer is valid. These rule reviews would be up to each individual game to define. An example would be whether an item could be sold on the auction or if it was prohibited. The buying rule review would require the system to check and see if the bidding price was over the current bid. In the MarketSellProtocol, the item would be posted to the market for bidding. In the MarketBuyProtocol, the previous bidder's assets would need to be returned, then the market could be updated with the new client's bid.

```
EscrowService(ClientA, Market(Item,Price))=
    If State=Begin
        If Remove ClientA.item = Successful
        Then{State = Review}
        Else{Exit}
    If State=Review
        If MarketReview(Item,Price) = Succesfull
        Then
        If Market = MarketSell
        Then{Post to Market}
        If Market = MarketBuy
        Then{Update Market, Rollback Previous Bidder}
        Else{Exit}
    Else{Rollback,Exit}
```

5.7 Contract Service

The following shows the operating sets used in describing the contract service. The contract service is split into three functional entities: acquire, loan, service. Sets StateC and StateB denote the possible states contract create and contract bid/complete protocols can be in, respectively. Sets InquiryC and InquiryB hold the server question asked to the client for contract create and contract bid/complete, respectively. Set M holds the message contents available to be sent between entities.

StateC= (Waiting, RewardItem, SetContractInfo, Confirm)

StateB= (Waiting, Contract)

InquiryC = ("Service Type?", "Reward Items?", "Contract Info?", "Confirm Contract?")

InquiryB = ("Service Type?", "Select Contract Number?")

M = ((Yes, No) U Items U Inquiry)

The formal specification of the contract service can be defined as a function using connection of a single client. The contract service handles individual clients listing contracts for the entire game network to review, bid, complete. The contract service is handled by splitting the service into three protocols: create, bid, and complete.

When a client requests to create a contract, the ContractCreateProtocol is initiated. First, the type of contract service is selected from: acquire, loan, service.

Acquire Contract requires the bidder to acquire a specific quantity of an item for the contract to be completed.

Loan Contract requires bidder to loan an item to the client for a set amount of time.

Service Contract requires bidder to perform actions on provided items and produce an end result.

Then, the reward for the service is set by the client for contract completion. Next, the contract information, like quantity of items or loan time, is set. Lastly, the client confirms the contract information is correct, and the escrow service is called to post the contract.

```
ContractCreateProtocol(ClientA)=
    If StateC = Waiting AND Inquiry = "Service Type?"
        If ClientA.M = Service
        Then{State = RewardItem}
    If StateC = RewardItem AND Inquiry = "Reward Items?"
        If ClientA.M = Item
        Then{State= SetContractInfo}
        Else{Retry}
    If StateC = SetContractInfo AND Inquiry = "Contract Info?"
        If ClientA.M = ValidEntryInfo()
        Then{State= Confirm}                Else{Retry}
    If StateC = Confirm AND Inquiry = "Confirm Contract?"
        If ClientA.M = "Yes"
        Then{EscServ(ContractCreate(Service, Reward, ContractInfo))}
        Else{Exit}
```

When a client requests to bid on a contract, the ContractBidProtocol is initiated. First, the type of contract service is selected from: acquire, loan, service. Then, the available listings in that service are displayed to the user. Once the user selects the contract to bid on, the escrow service is called to validate the bid.

```
ContractBidProtocol(ClientA)=
    If StateB = Waiting AND Inquiry = "Service Type?"
        If ClientA.M = Service
        Then{PrintList(EscServ.Contract(Service)), State = BidContract}
        Else{Retry}
    If StateB = BidContract AND Inquiry = "Select Contract Number?"
        If ClientA.M = ValidContractNumber()
        Then{EscServ(ContractBid(Service, Number)}
        Else{Exit}
```

The ContractCompleteProtocol is only used by acquire and service parts of the contract service. The loan service does not use the completion method, as the contract automatically completes at the end of the loan time period. When used, the client must choose what type of service contract to complete. Then, the client's outstanding contracts for the specific contract types are displayed. Once the user selects the contract to complete, the escrow service is called to validate and complete the contract.

```
ContractCompleteProtocol(ClientA)=
    If StateB = Waiting AND Inquiry = "Service Type?"
        If ClientA.M = Service
        Then{PrintList(EscServ.Contract(Service)), State = Contract}
        Else{Retry}
    If StateB = Contract AND Inquiry = "Select Contract Number?"
        If ClientA.M = ValidContractNumber()
        Then{EscServ(ContractComplete(Service, Number)}
        Else{Exit}
```

The escrow service can be called by all three types of contract services: acquire, loan, service. Every time the service is called it must attempt to remove an item each time (reward item, loan item, or end result). When the contract create service is called, the escrow service must review if the contract input is valid and properly formatted for the type of service, the reward has been successfully removed from the user, and then it can post the contract to the listing. When the contract bid service is called, the escrow must review the bid to ensure it meets the contract requirements. Such examples are when a bidder has the loan item needed for the contract, or the bidder has skills necessary to perform the service. Then, the escrow must remove the contract from the public listing, and start the countdown for completing the contract. The contract complete service, when called, must review the client's items to see the if they match the contract requirements, and notify the user if the contract has not been successfully completed. If the contract requirements have been met the reward is given to the client, and any other contract items are given to the proper owners.

```
EscrowService(ClientA, Contract(Item,Info))=
    If State=Begin
        If Remove ClientA.item = Successful
        Then{State = Review}
        Else{Exit}
    If State=Review
        If ContractReview(Item,Price) = Successful
        Then
            If Contract = ContractCreate
            Then{Post to Contract Listing}
            If Contract = ContractBid
            Then{Remove from Contract Listing, start contract completion counter}
        If Contract = ContractComplete
        Then
            If ClientA contract requirements met
            Then{Return items and rewards to appropriate sources}
            Else{Inform client contract reqs not met}
        Else{Rollback,Exit}
```

5.8 Market and Contract Service Operations

One important note for both the market and contract services is for the handling of items when no bid is placed. Since the escrow holds ownership of the items when they are up for bid, the escrow must return all items to the original owner after the designated period for bidding has passed. Therefore, the server must be able to trigger an event when any auction or contract meets its deadline without a bid. This trigger must then call the escrow service to return any outstanding items to the owner, and remove the auction or contract from the public listing. The server can decide to either directly deposit these items into the archived logs, or place them in the database and return the items the next time the user logs onto the gaming service.

5.9 Advantages

5.9.1 Independent

With new MMORPGs entering the market at a rapid pace, not all games may need all three trading services. For example FarmVille, a very popular game on Facebook, may not have enough gameplay options to need the market or contract trading services, but may only need the barter service for quick trades between clients or ”friends”. The advantage of the mitigation method is that all services are completely separate from each other, and can be implemented according to need.

5.9.2 Logging Abilities

The unique use of the escrow service across all trading mechanism in this paper's implementations provides a great position for logging and research. Understanding player gameplay is important to game developers, and logging all transactions taking place within the escrow could give insight into future trends taking place in the game, or areas to expand on to retain long-time players. The logging capability would also provide options to analyze any suspicious players, and be able to create a network of associates through trade events.

CHAPTER 6. TEXT-BASED MITIGATION IMPLEMENTATION

6.1 Setup / Infrastructure

The first use of the scam mitigation method was to build a text-based implementation. The code was written in Java, using the Eclipse SDK 3.6.2 for compiling and debugging. MySQL is used for the database, using MySQL Workbench 5.2.33 CE for table editing and debugging. All source code was created from scratch. The Windows command prompt is used to handle client and server startup and interaction.

6.2 Client-Server Communication

The infrastructure uses a client-server model to provide a server to handle multiple client connections. The server is in charge of the creating and maintaining connections with the clients, the authentication service, escrow service and database communication. The server must initiate the authentication service to ensure each client is a valid user. Then, the client information must be pulled and placed into their inventory on the database, in order for game-play to resume. The communication operates using the Java readline command, which waits for input before progressing in the code. This means that the server and client must respond to each piece of input it receives to progress forward in the protocol.

6.3 Trade Services

6.3.1 Initialization

To start the trading platform, the MMOServer must be started. The MMOServer gives three choices to start the barter, market, and contract services. In order to maintain code simplicity each server instance only runs one service, so three servers must be running to provide all services at the same. Each of the services can run independently of each other on different port selections. For each new instance of the service, the server creates a new thread, and accepts a socket connection from the client. Each server thread stores the client socket information, and creates a PrintWriter and BufferedReader to handle input/output operations for each client. Then, the appropriate service is initiated and the client can interact with the trading protocol. The protocol uses a FileInputStream and FileOutputStream to handle writing player information to and from its log archive. In this case, a simple text file is used to store all player information.

6.3.2 Service Initialization

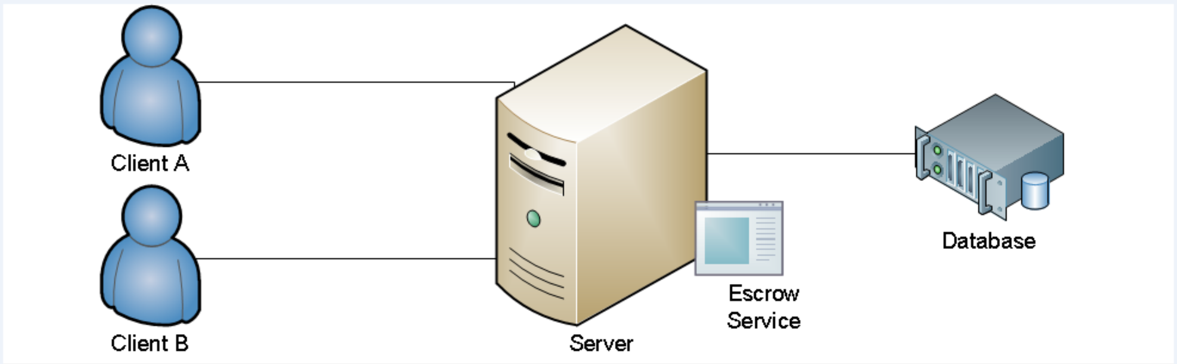
Each protocol starts with authentication to check the database to see if a valid username and password has been given. If the login is incorrect for a set number of times, the socket connection is terminated. If correct, the stored log reports are opened and searched for the individual user, and a table is created in the database holding the client's inventory data for gameplay.

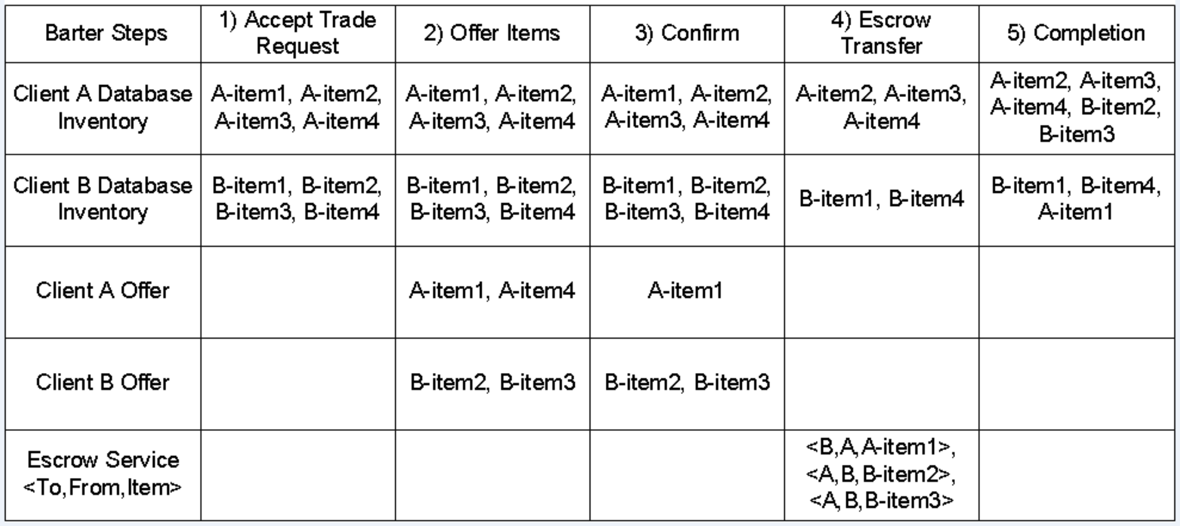
Each protocol runs two checks to ensure the escrow service is operating properly. First, the service reviews the protocol to see if any auctions or contracts have reached the posting deadline without any bids. In order to provide a situation that could be tested, the implementation uses a deadline countdown where each authentication into the system reduces the time remaining by 1. In this way simulated auctions and contract could occur without waiting for a specific time.

If any of the public listings have reached their deadline without a bid, the escrow service places any items it is holding for the client in a return table and deletes the listing. If any of the public listings have reached their deadline and have a bid, the escrow service places the item in the return table with the highest bidder as the owner, while the bid is placed in the return table to the auction owner. Second, the return tables are checked for items belonging the current client connected, and returns the item to the client's inventory and deletes the item from the escrow service.

6.3.3 Barter Service

The protocol holds several variables to handle the input/output operations of two clients. The protocol is initiated by setting the PrintWriter and BufferedReader for the clients. A state machine structure is created using a Java enum type, creating a constant source capable of storing values associated with it. It then establishes the database connection, and creates two inner classes for all operations associated with the clients called BarterStatus. The BarterStatus holds information on the client's state machine status, boolean operation flags, username, and inventory listing.





Barter Steps	1) Accept Trade Request	2) Offer Items	3) Confirm	4) Escrow Transfer	5) Completion
Client A Database Inventory	A-item1, A-item2, A-item3, A-item4	A-item1, A-item2, A-item3, A-item4	A-item1, A-item2, A-item3, A-item4	A-item2, A-item3, A-item4	A-item2, A-item3, A-item4, B-item2, B-item3
Client B Database Inventory	B-item1, B-item2, B-item3, B-item4	B-item1, B-item2, B-item3, B-item4	B-item1, B-item2, B-item3, B-item4	B-item1, B-item4	B-item1, B-item4, A-item1
Client A Offer		A-item1, A-item4	A-item1		
Client B Offer		B-item2, B-item3	B-item2, B-item3		
Escrow Service <To,From,Item>				<B,A,A-item1>, <A,B,B-item2>, <A, B, B-item3>	

Figure 6.1 Barter Flow Chart

The barter service performs five steps to keep the service as simple as possible. The first step of the service ensures that both parties want to participate in the trade request. Then, the service queries the database to pull the inventory of each client. The clients are only presented with a listing of items that they have permission to trade, excluding any items that are untradeable, loaned items, or contract items. Once both clients choose the items to trade, they must review and accept the trade offer.

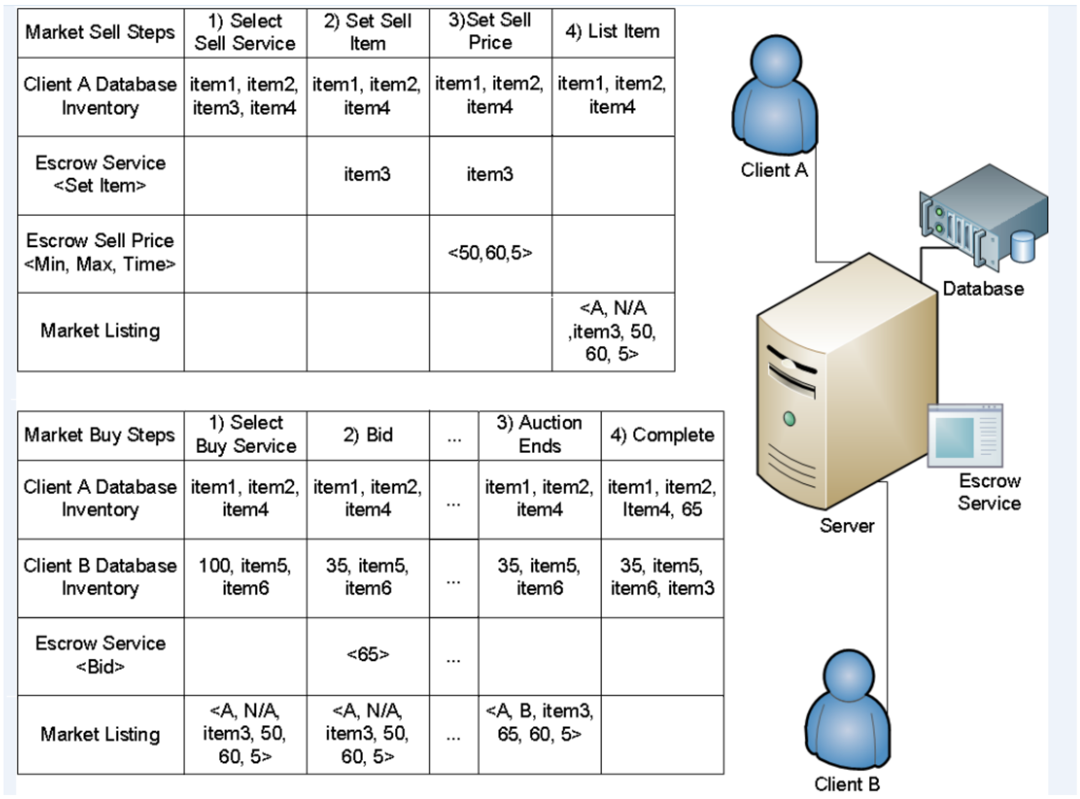
With the offer accepted, the escrow service first places the items in the escrow database, and deletes the items from the user's inventory. Once all items have been removed from both clients, the items placed in the opposite client inventory, and deleted from the escrow. Therefore the trade is complete and the barter system ends. If at any time the escrow service fails to successfully remove and place an item going to or from the database, the service can retry the action, or it can perform a rollback and cancel the trade. Lastly, when the clients close their connection their database inventory is written to an archive for storage.

6.3.4 Market Service

The market protocol stores variables to handles input/output interaction with the client. The protocol stores the PrintWriter and Buffered Reader for communication to the client and stores database connection for SQL queries. Similar to the BarterStatus, the inner class MarketStatus holds client state machine status, operating flags, username, and inventory listing. To start the protocol service, the client must select if they want to sell an item on the market, or bid on an existing auction. When the clients close their connection their database inventory is written to an archive for storage.

6.3.4.1 Sell

The sell service starts by allowing the client to create a new auction or to check on the status of their current auctions. To check on current auctions, the protocol simply queries the database public listing for any items where the client is the owner. To create an auction, the protocol queries the database to provide the client with a list of its inventory that it is able to sell on the market. The client must choose an item, starting bid, minimum sell price, sell now price, and time limit. The escrow service must then validate the bidding information, remove the item from the client inventory in the database, and post the auction to the public listing. If at any time the escrow service fails to accomplish the public listing, a rollback occurs and the client is informed of the failure.



Market Sell Steps	1) Select Sell Service	2) Set Sell Item	3)Set Sell Price	4) List Item
Client A Database Inventory	item1, item2, item3, item4	item1, item2, item4	item1, item2, item4	item1, item2, item4
Escrow Service <Set Item>		item3	item3	
Escrow Sell Price <Min, Max, Time>			<50,60,5>	
Market Listing				<A, N/A ,item3, 50, 60, 5>

Market Buy Steps	1) Select Buy Service	2) Bid	...	3) Auction Ends	4) Complete
Client A Database Inventory	item1, item2, item4	item1, item2, item4	...	item1, item2, item4	item1, item2, Item4, 65
Client B Database Inventory	100, item5, item6	35, item5, item6	...	35, item5, item6	35, item5, item6, item3
Escrow Service <Bid>		<65>	...		
Market Listing	<A, N/A, item3, 50, 60, 5>	<A, N/A, item3, 50, 60, 5>	...	<A, B, item3, 65, 60, 5>	

Figure 6.2 Market Flow Chart

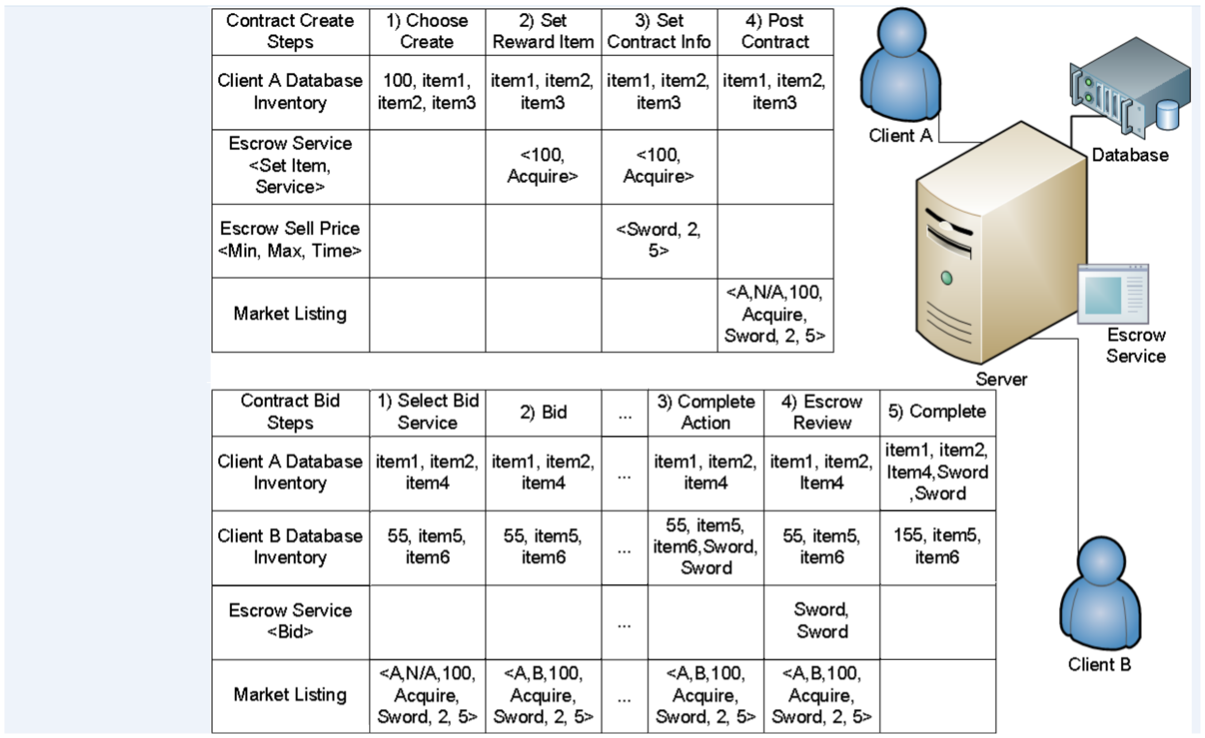
6.3.4.2 Buy

The buy service allows the client to bid on an existing auction or to check the status of their bids on current auctions. To check on their current auctions, the protocol simply queries the database public listing for any auctions where the client is the current bidder, or previous bidder on an item, and the status of that auction. To bid on an auction, the public listing of auctions is presented to the client. The client can then choose an auction to bid on, and present the item or currency used to bid.

The escrow service checks to see if the offered price is greater than the start price and current bid price. Then, the item or currency is removed from the client, and the auction is updated to show the new bid price and bidder. If there was a previous bidder, the previous bid is removed and placed into the market return for the next time the user logs into the system. This prevents users from bidding on an item without having enough money to cover the final bid, but ensures the bid is immediately returned to the user if his bid is no longer valid. The user is placed into the previous bidder list to ensure that the user can find the auction to place a higher bid.

6.3.5 Contract Service

The contract protocol initialization differs only slightly from the market service. Similarly, the protocol only handles one client's input/output information, state machine status, and name. However, it holds several empty instances of items needed to be read by the contract in order for the escrow service to determine whether or not a valid contract has been created. To start the protocol service, the client must select if they want to create, bid, or complete a contract. When the client closes their connection to the service, their database inventory is written to an archive for storage.



Contract Create Steps	1) Choose Create	2) Set Reward Item	3) Set Contract Info	4) Post Contract
Client A Database Inventory	100, item1, item2, item3	item1, item2, item3	item1, item2, item3	item1, item2, item3
Escrow Service <Set Item, Service>		<100, Acquire>	<100, Acquire>	
Escrow Sell Price <Min, Max, Time>			<Sword, 2, 5>	
Market Listing				<A,N/A,100, Acquire, Sword, 2, 5>

Contract Bid Steps	1) Select Bid Service	2) Bid	...	3) Complete Action	4) Escrow Review	5) Complete
Client A Database Inventory	item1, item2, item4	item1, item2, item4	...	item1, item2, item4	item1, item2, Item4	item1, item2, Item4,Sword ,Sword
Client B Database Inventory	55, item5, item6	55, item5, item6	...	55, item5, item6,Sword, Sword	55, item5, item6	155, item5, item6
Escrow Service <Bid>			...		Sword, Sword	
Market Listing	<A,N/A,100, Acquire, Sword, 2, 5>	<A,B,100, Acquire, Sword, 2, 5>	...	<A,B,100, Acquire, Sword, 2, 5>	<A,B,100, Acquire, Sword, 2, 5>	

Figure 6.3 Contract Flow Chart

6.3.5.1 Create

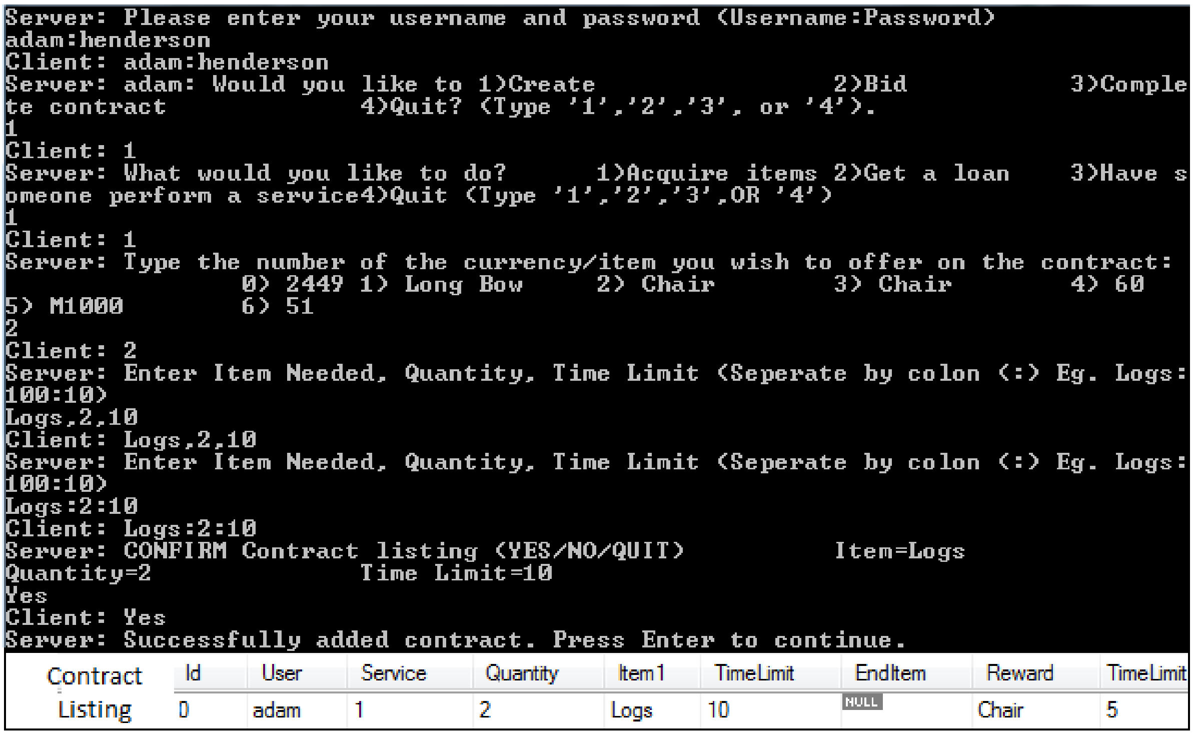
To create a contract, the client's inventory is shown for the user to select an item to provide as a reward for successful completion of the contract. Then, the contract information must be entered for the specific contract service selected:

Acquire : Item Needed, Quantity, and Time Limit

Loan : Item Needed and Time Limit

Service : Items to perform the service (up to 2 items), End Item Needed, and Time Limit

Then, the escrow service must confirm that the correct contract information and valid reward has been chosen. If the contract has been successfully created, the items and reward are removed from the client's inventory. Then, the contract is placed into the public listing, and the client is informed of a successful contract creation.



```
Server: Please enter your username and password (Username:Password)
adam:henderson
Client: adam:henderson
Server: adam: Would you like to 1)Create                2)Bid           3)Comple
te contract             4)Quit? (Type '1','2','3', or '4').
1
Client: 1
Server: What would you like to do?      1)Acquire items 2)Get a loan    3)Have s
omeone perform a service4)Quit (Type '1','2','3',OR '4')
1
Client: 1
Server: Type the number of the currency/item you wish to offer on the contract:
                0) 2449 1) Long Bow     2) Chair        3) Chair        4) 60
5) M1000        6) 51
2
Client: 2
Server: Enter Item Needed, Quantity, Time Limit (Seperate by colon (:) Eg. Logs:
100:10)
Logs,2,10
Client: Logs,2,10
Server: Enter Item Needed, Quantity, Time Limit (Seperate by colon (:) Eg. Logs:
100:10)
Logs:2:10
Client: Logs:2:10
Server: CONFIRM Contract listing (YES/NO/QUIT)          Item=Logs
Quantity=2              Time Limit=10
Yes
Client: Yes
Server: Successfully added contract. Press Enter to continue.
```

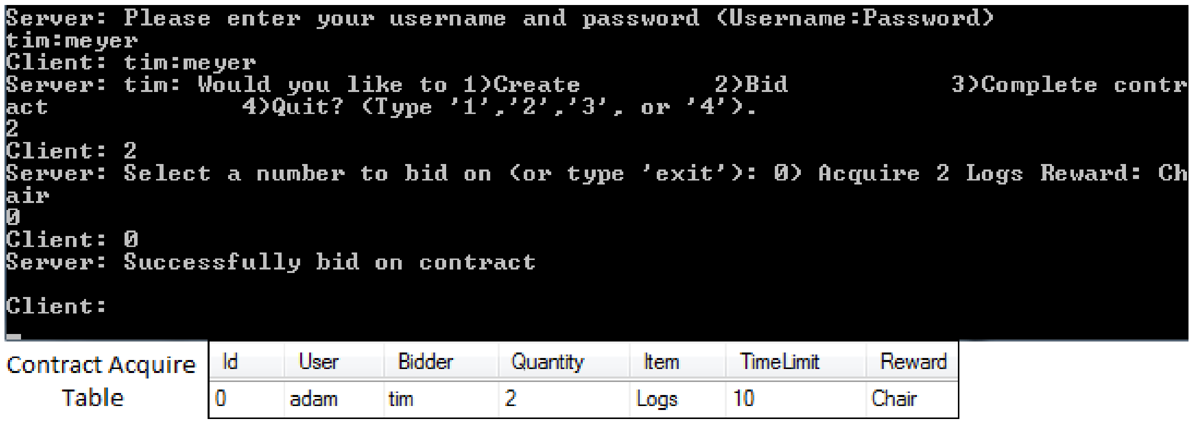
Contract Listing	Id	User	Service	Quantity	Item1	TimeLimit	EndItem	Reward	TimeLimit
	0	adam	1	2	Logs	10	NULL	Chair	5

Figure 6.4 Creation of an Acquire Contract.

6.3.5.2 Bid

To bid on a contract, the public listing of contracts is presented to the client. The client can then choose a contract to bid on, and see if the client meets the contract requirements. The acquire contract service has no requirements to accept, the loan contract service requires that the client have the item needed in its inventory to accept the contract, and the service contract requires that the client have the necessary skills to create the end item.

If the escrow service shows that the client has met the requirements to bid on the item, escrow service then deletes the contract from the public listing and places it into the escrow's contract sub-tables (contractacquire, contractloan, or contractservice). For the loan service, the item must be removed from the bidder's inventory and stored into the escrow immediately. Now, every time a client logs on, any items provided to it from a contract are placed into their inventory, however a lock is placed on this item, preventing certain operations from occurring.



```
Server: Please enter your username and password (Username:Password)
tim:meyer
Client: tim:meyer
Server: tim: Would you like to 1)Create         2)Bid           3)Complete contr
act             4)Quit? (Type '1','2','3', or '4').
2
Client: 2
Server: Select a number to bid on (or type 'exit'): 0) Acquire 2 Logs Reward: Ch
air
0
Client: 0
Server: Successfully bid on contract

Client:
```

Contract Acquire Table

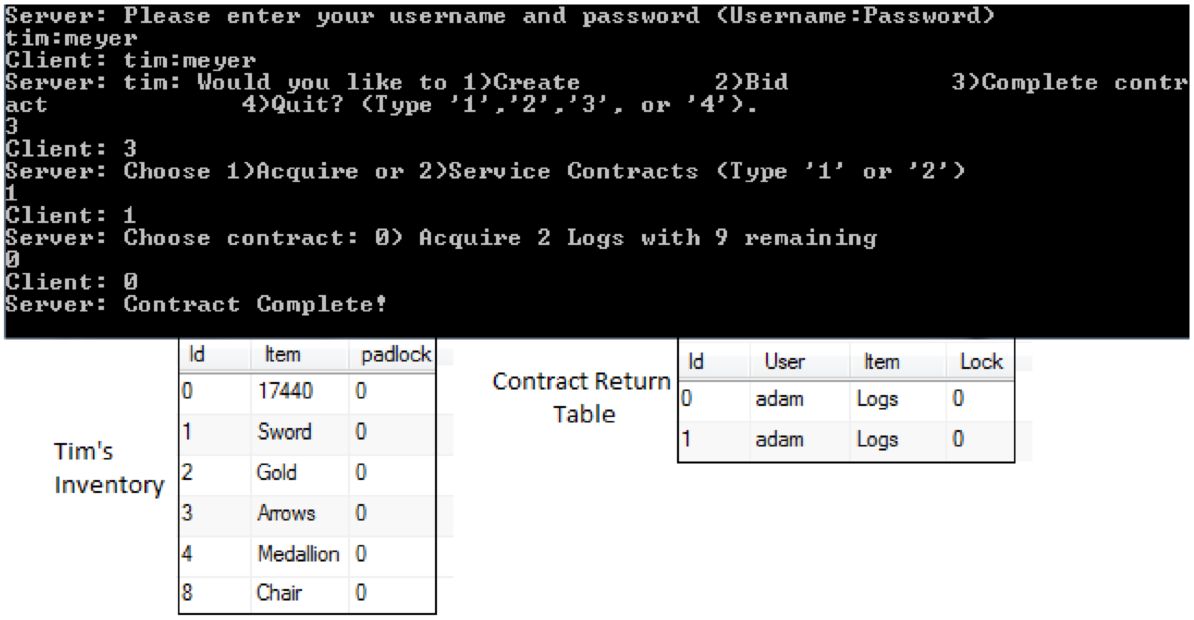
Id	User	Bidder	Quantity	Item	TimeLimit	Reward
0	adam	tim	2	Logs	10	Chair

Figure 6.5 Bidding on an Acquire Contract.

The locking ability allows the escrow to control items even when the escrow service is not storing it. For the contract loan service, every time a user logs in with a current contract loaning an item to them, the escrow service places the loaned item into their database. Then, when a client views their inventory the item behaved as any other item owned. However, each loaned items properties indicate a lock that prevents the temporary owner from trading the item or dropping the item into the game environment. This maintains that the escrow controls the ownership of the item at all items, and that the user cannot circumvent the system to breach the contract. When the client logs off and normal procedures write the client's inventory to the archive, the log does not write the item, to prevent duplication of the item.

6.3.5.3 Complete

The complete service is required to notify the escrow service when a contract bidder wishes to complete the contract and acquire the reward. An important note is that the loan service under the contract system does not use the complete service, once an item has been contracted to be loaned for a period of time, the escrow will provide the loaned item to the client until the time limit of the loan has ended, in which the escrow automatically completes the trade, sending the reward and loaned item to the contract bidder, and deleting the contract from the contractloan table.



```
Server: Please enter your username and password (Username:Password)
tim:meyer
Client: tim:meyer
Server: tim: Would you like to 1)Create         2)Bid           3)Complete contr
act             4)Quit? (Type '1','2','3', or '4').
3
Client: 3
Server: Choose 1)Acquire or 2)Service Contracts (Type '1' or '2')
1
Client: 1
Server: Choose contract: 0) Acquire 2 Logs with 9 remaining
0
Client: 0
Server: Contract Complete!
```

Tim's Inventory

Id	Item	padlock
0	17440	0
1	Sword	0
2	Gold	0
3	Arrows	0
4	Medallion	0
8	Chair	0

Contract Return Table

Id	User	Item	Lock
0	adam	Logs	0
1	adam	Logs	0

Figure 6.6 Completion of an Acquire Contract.

For the other two services, the client must review outstanding contracts, and select which one it would like to attempt to complete. For the acquire service, the escrow service reviews whether or not the client inventory holds the quantity of items that the contract requires. If it is shown that the client doesn't hold the necessary quantity of the specific item, it denies the completion and cancels the service. If the client has the specific quantity of items, the escrow service removes the items from the inventory and places the items into the contractreturn table to be given to the user the next time the contract owner logs in. If completed the service gives the reward to the client, and deletes the contract from the contractacquire table.

For the service system, the escrow service reviews the client's inventory and queries the server to see what end items can be created (according to game design). This process is made to mimic typical MMORPG gameplay where single or multiple items can be used to create new or altered items. For example, if the contract is asking for a certain end item, the escrow reviews the table to see what items and skills are needed to create the end item, if the user's inventory has these items, and the contract progresses. The contract owner's items, while locked to preventing users from trading them, are removed from the bidder's inventory. The end item is then placed into the contractreturn table to give the user the completed contract service item, and the reward is placed into the current client's inventory. Lastly, the service contract is deleted from the table.

6.4 Testing

Provided below are the testing criteria for all three mitigation methods: barter, market, contract as well as subservice testing. The barter testing handles all incorrect input from both users and review ability to recover or cancel the service.

6.4.1 Authentication and Service Selection

For each service, authentication of clients was tested to ensure proper login and logging services. Also, each service selection and subservice selection was tested with the following tests.

Testing Area	Individual Test	Test Result
Authentication	Correct Input (2 Users)	Pass
	Correct Input (1 User)	Pass
	Incorrect Login Max (1,2 Users)	Pass
	Invlaid Symbol Usage	Pass
	Same User Login	Pass
Service Select	Buy/Sell/Quit	Pass
	Improper Answers	Pass
	Invalid Symbols	Pass

Table 6.1 Testing of Authentication and Setup

6.4.2 Barter Testing

The barter protocol was tested with the following situations to ensure correct operation, and stated successful testing if both server response and database tables correctly matched expected behavior.

Testing Area	Individual Test	Test Result
Barter Request	Yes/No	Pass
	Improper Answers	Pass
	Invalid Symbols	Pass
Item Acquire	Proper Numbers	Pass
	No Items	Pass
	IndexOutOfBounds	Pass
	Not an Integer	Pass
	Single/Multiple Items	Pass
Barter Accept	Yes/No	Pass
	Improper Answers	Pass
	Invalid Symbols	Pass

Table 6.2 Barter Service Testing

6.4.3 Market Testing

The market protocol tested both the buy and sell methods associated with the service. The table represents the testing done on the subservices within the market protocol.

Testing Area	Individual Test	Test Result
Sell Check Auction	Auctions Availible	Pass
	No Auctions	Pass
	Ending Auction	Pass
Sell Item Acquire	Proper Input	Pass
	No Items	Pass
	IndexOutOfBounds	Pass
	Not an Integer	Pass
	Single/Multiple Items	Pass
Sell Auction Update	Correct Input	Pass
	Invalid Symbols	Pass
	Empty	Pass
	Not an Integer	Pass
	IndexOutOfBounds	Pass
Check Auctions	Auctions Availible	Pass
	No Auctions	Pass
	Ending Auctions	Pass
Item Acquire	Proper Input	Pass
	No Items	Pass
	Not an Integer	Pass
Item Bid	Proper Input	Pass
	Exit Entry	Pass
	Not an Integer	Pass
Item Sold	Proper Bid/Return	Pass
	No Bid	Pass
	Bid Below Minimum	Pass

Table 6.3 Market Service Testing

6.4.4 Contract Testing

The contract protocol tested the setup and includes testing on all three services: Create, Bid, and Complete a contract. Within each service, the three types of contracts: Acquire, Loan, Service were tested.

Testing Area	Individual Test	Test Result
Create Item Acquire	Proper Input	Pass
	No Items	Pass
	Not An Integer	Pass
Auction Update	Proper Input	Pass
	Invalid Symbols	Pass
	Empty	Pass
	Not an Integer	Pass
	IndexOutOfBounds	Pass
Contract Bid	Proper Input	Pass
	No Items	Pass
	IndexOutOfBounds	Pass
	Contract Bid Fail	Pass
	Not an Integer	Pass
Complete Acquire	User Meets Reqs	Pass
	User Fails Reqs	Pass
	Delivered/Returned	Pass
	Reward Given	Pass
	Escrow Listing Deleted	Pass
Complete Loan	Auto Complete Loan	Pass
	Correct Deadline End	Pass
	Reward Given/Returned	Pass
	Item Returned	Pass
	Escrow Listing Deleted	Pass
Complete Service	Check Contract Items	Pass
	Pass/Fail End Item	Pass
	Reward Given/Returned	Pass
	Delivered/Returned	Pass
	Escrow Listing Deleted	Pass

Table 6.4 Contract Service Testing

6.5 Results

The text-based implementation successfully implemented the scam mitigation method's architecture and protocol. The method provides client-server implementation, authorization service, escrow service, and three trading mechanisms: Barter, Market, and Contract.

6.5.1 Overall System Results

The scam mitigation method provides protection to all clients whether they are buying or selling. It provides three services where a client can trust the escrow service to carry out proper trade, without having to trust the individual user. The escrow system can place locks on items to ensure only limited functionality can occur on items when loaned or contracted for service. The rollback service included in the escrow service provides a venue to return all items to the original owners if improper client actions or network failures occur. The service does all of this functionality without human interaction through automated checks on all trading.

6.5.2 Barter Service Results

The barter service passed testing requirements and provides trading between two clients. The barter service within the scam mitigation method provides several advantages. The barter service works with the escrow service to ensure that the client cannot trade any items it does not have permission to give away. Escrow service can review all items when both users have accepted, and perform checks to ensure the trade follows game rules and TOS. Escrow service maintains ownership of trade items, until all items have been acquired by the service and the users have accepted all trade information.

6.5.3 Market Service Results

The market service successfully provided a venue to place a item onto a public auction, and provide other clients ability to bid on the items. The market service within the scam mitigation method uses several techniques for ensure proper bidding during auctions. The market service prevents clients from creating an auction by denying use of items which the client does not have ownership. Escrow service can review auction listings to deny auctions of untradeable items. For bidding, the escrow service takes ownership of the currency used for the bid before making an update to the current bid, while all previous currency from bids are returned to the proper user. This ensures no bid can be placed from users without proper funds. When an auction completes, the market service immediately transfers auction items and winnings to the appropriate online user, or places them into tables to be immediately returned to the users on their next log in.

6.5.4 Contract Service Results

The contract service within the scam mitigation method provides new techniques for inserting a contract service into MMORPGs. The contract successfully provides mechanisms for three services. First, the acquire contract allows for clients to contract a user to obtain a quantity of an item in a certain amount of time. Second, the loan contract allows a client to borrow a user's item for a period of time, while the escrow service maintains ownership throughout the service. Third, the service contract allows for a user to provide items to another user in order for a service to be performed that the user cannot perform. Therefore the contracts provide a safe venue for users to have these services performed, instead of trusting other individuals to agree to a verbal contract within the game. The escrow maintains ownership of all contract items until the contract has been completed or the contract deadline has been reached, where the items are given to the proper users. This escrow ownership prevents users from breaching a contract and stealing items. The contract service verifies contract completion through logical checkpoints, and does not need human verification.

CHAPTER 7. MMORPG MITIGATION IMPLEMENTATION

7.1 Background

In order to prove that the scam mitigation method implemented in the text-based service in the chapter before would work in MMORPGs used today, open-source MMORPGs were reviewed. Several games were analyzed and used to implement the scam mitigation method. The following sections explain failures and successes associated with each game.

7.2 Implementation Reviews

7.2.1 Alien Arena

Alien Arena was the first game that successfully downloaded the open-source code, compiled and played the game. However, immediate problems surfaced when connecting to the online servers and MMO gameplay. It also appeared that the game would not offer a trading mechanism that was needed to implement the scam mitigation method. This failure led to reviewing a list of open-source games focusing on two core needs. First, games would need to provide successful gameplay on first compile from the downloaded source. Secondly, the game would have to have a working trade mechanism that shared similarities to popular games such as WoW, RuneScape, and Eve Online.

Result : Failure - No Trading Mechanism

7.2.2 0 A.D

0 A.D was a real-time strategy game that was open-source and compiling was very successful and easy. However, after using the game for a period of time, it was noticed that the trading

talked about in some of the overview did not correlated to the implementation within the game. The only way to acquire other player's assets would be to conquer territories, and therefore did not meet the necessary requirements.

Result : Failure - No Trading Mechanism

7.2.3 WorldsForge

WorldsForge is a relatively well-known open-source MMORPG game with many developers still working on the project. The source code was easy to obtain, and the sites forum had many followers discussing updates being made on the game. The game did contain a trading platform, however most of the trading in the system was text-based trading with NPCs only, making testing difficult. After having difficulties compiling and playing the game, other options were reviewed.

Result : Failure - Poor Gameplay and Trading Mechanism options

7.2.4 Planeshift

Planeshift is another well-known open-source MMORPG game with developers currently working on it. The source code was easy to obtain, and compiling could be done after some minor changes to the host system. The game held a typical trading platform that would be seen in popular MMORPGs and met the needed criteria. However, after careful review of the code, fail-safes had been built into the trading mechanism having players drop (or lose) items if any error occurred during the trading process. This flaw in the game design would cost a lot of developing and debugging time to remove, on top of the implementation of the escrow service, so the game was set aside.

Result : Failure - Poor Coding Practices

7.3 Eclipse Origins

Eclipse Origins is a 2D MMORPG focused on helping individuals build and customize their own MMORPG world. Eclipse Origins has a game engine to help create personalized games, but also allows the source code to be edited. Eclipse Origins provide source code for both the server and client and was a suitable for implementing the escrow service. The issue with the game was that the source code was written in Visual Basic 6.0, an older piece of software that was difficult to acquire.

Normal trading operation of Eclipse Origins handles simple barter trading between multiple players. A player must select another player on the board, and send a trade request. Then, the second player must accept the trade request. The game then allows for users to select items from their inventory until both users are satisfied, and each user must accept the trade, and the items are transferred. Comparing the code base, the setup for the barter system was complete, and the only implementation needed to implement the scam mitigation method was the escrow service and some helper functions.

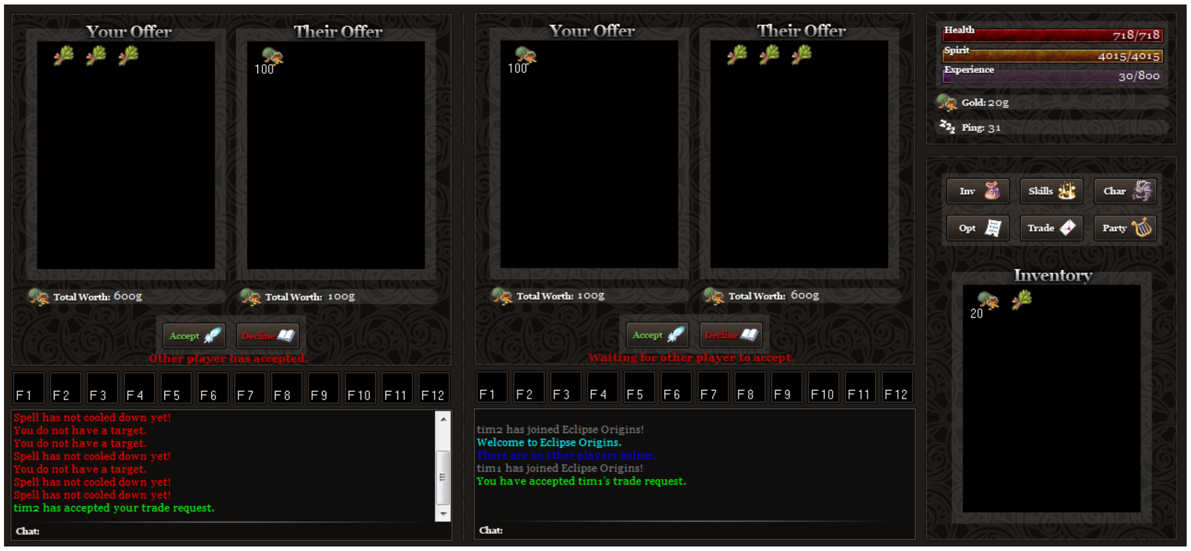
No changes on the client were needed and minimal changes were needed in the server source code. The first step was to create the escrow variables to store the items of player trading. This was placed in the modTypes file that handled all information stored by the server on the gameplay.

```
Public Escrow As EscrowInvRec
Public Type EscrowInvRec
    ' Trader 1
    Num(1 To MAX) As Long
    Value(1 To MAX) As Long
    ' Trader 2
    Num2(1 To MAX) As Long
    Value2(1 To MAX) As Long
    Approval As Long
```

End Type

The next step in the implementation process was to locate and remove the temporary variables transferring the items to the players, and replace them with a link to the escrow storage. The code first clears the escrow service of any previous trade interaction, then replaces the temporary item storage lists and replaces it with the escrow services private storage area. Then, a new escrow function is introduced into the system.

The EscrowTradeCheck function reviews the items placed into the escrow and determines whether or not the trade can move forward. For this implementation, the escrow checks two properties of the trade before approving its status. First, it reviews that both players have offered an item in the trade, preventing one-way trades. Secondly, it reviews the total value of each players trade, and if the trade benefits a player by 500% or more. If either fail review, the escrow denies the trade, and performs a rollback on all items to their original owner. If the trade passes these two requirements, the escrow status changes to approved, and the items can be transferred by the game process.



Other player has accepted.

```
Spell has not cooled down yet!
You do not have a target.
You do not have a target.
Spell has not cooled down yet!
You do not have a target.
Spell has not cooled down yet!
Spell has not cooled down yet!
tim2 has accepted your trade request.
```

Waiting for other player to accept.

```
tim2 has joined Eclipse Origins!
Welcome to Eclipse Origins.
There are no other players online.
tim1 has joined Eclipse Origins!
You have accepted tim1's trade request.
```

Figure 7.1 Eclipse Origins 2-player trade

7.3.1 Results

The implementation was successful and provided an escrow service for the bartering system within the game. The game client protocol was successfully changed without introducing any changes on the client code or gameplay. Two game rules were introduced in order to show the full functionality of the service. The escrow service implemented the game rules in the trade review function in order to show ability to follow TOS and game rules.

The escrow service implementation was able to easily replace the temporary variables used to transfer the items between accounts, and provide a system where the escrow service took ownership of the items, was able to review and approve the trade, and then transferred the items to the owners without altering gameplay.

CHAPTER 8. SPECIAL CONSIDERATIONS

8.1 MMO's Embracing Scamming

MMORPG game developers may not want to implement the escrow service discussed in the previous chapters. The most prevalent of examples comes from the game Eve Online. The developers of Eve Online purposely do not deal with dangers such as piracy, theft, and ransom in the game in order to create a unique gaming environment.[19] Therefore, the players within the game are forced to make decisions to trade or interact with other players on whether or not they can be trusted. Here, the scam mitigation method would likely not be implemented, as it would reduce the possibility to scam during trading. For example, in Eve Online the players can contract with for a player to deliver goods to a different area in the game as a courier. However, the game allows the player to see what cargo it is moving and is able to steal the cargo at will. The scam mitigation method would prevent this from happening and alter the gameplay entirely.

CHAPTER 9. SUMMARY AND FUTURE WORK

9.1 Summary

With the increasing popularity of MMORPGs it is important to look at how trading within the game's virtual economy handles and mitigates scamming attempts. The scam taxonomy provided in this paper presents a framework for identifying and categorizing scams within MMORPGs. Using this taxonomy, new scams can be categorized and handled quicker with knowledge of similar scams. The scam mitigation method can also be used as a first defense against scams taking place in MMORPGs with virtual economies using barter, market and contract services. Using the system can prevent scams, provide better overall gameplay for the users, and prevent user attrition.

9.2 Future Work

9.2.1 Efficiency

The scam mitigation method implementation makes server database queries when the client logs into the game environment. Even though this is needed to see if any items need to be placed into their inventory, whether to return or to give a contracted item, these could cost an increase in startup time, or in-game lag. Review would need to be done to see if the mitigation method would make an impact on usability and gameplay.

9.2.2 Multi-Server Implementation

Secondly, and most important, future work would be to review how the escrow system can be expanded across a multiple server implementation. With current MMORPGs handling millions of users in a single day [10], it's important to understand how the escrow service can perform when handling trades over multiple platforms.

9.2.3 Contract Requirements Checklist

In order for the implementation of the scam mitigation method contract service to work the backend servers must be able to verify steps have been taken to meet the end goal. For example, if a user is contracted to make a diamond ring, he may need a gold bar and a diamond. The server watch actions performed on these contracted items, and only allow proper actions when creating the diamond ring. While the text-based implemenation showed one way of accomplishing this, there are multiple ways implementation could be handled. Important future work would be to look at how the system handles individual calls, and the frame work that would be needed to "check-off" steps towards contract completion.

9.2.4 Prestige

Since the implementation uses the escrow service at all times, it could log proper use by individual players. Then, when players enter into a barter or contract with another player, the escrow could tell the prestige, or confidence, that the other player will successfully carry out the trade or contract.

BIBLIOGRAPHY

[1] Greengard, (2011). Social games, virtual goods. *Commun. ACM, April 2011*, *54*(4), 19–22. New York, NY, USA, ACM

[2] Federal Trade Commission, (2009). A Staff Report on the Federal Trade Commission's Fraud Forum.

[3] Jeffrey Bardzell, Markus Jakobsson, Shaowen Bardzell, Tyler Pace, Will Odom, Aaron Houssian, (2007). Virtual Worlds and Fraud: Approaching Cybersecurity in Massively Multiplayer Online Games. *Situated Play. Proceedings of DiGRA 2007 Conference* http://www.digra.org/dl/db/07311.42219.pdf.

[4] Dr. Igor Muttik, (2008). Securing Virtual Worlds Against Real Attacks. *White Paper. www.mcafee.com*

[5] Nick Yee. Motivations of Play in MMORPGs, Results from a Factor Analytic Approach. http://www.nickyee.com/daedalus/

[6] Daniel M. Berry, (2011). Liability issues in software engineering: technical perspective. *Commun. ACM, April 2011*, *54*(4), 98–98. New York, NY, USA, ACM

[7] Sean F. Kane and Benjamin T. Duranske, (2008). Virtual Worlds, Real World Issues.

[8] Chris Simmons, Charles Ellis, Sajjan Shiva, Dipankar Dasgupta, Qishi Wu. AVOIDIT: A Cyber Attack Taxonomy. *Office of Naval Research(ONR) grant N00014-09-1-0752*

[9] Yan, Jeff and Randell, Brian. A systematic classification of cheating in online games. Proceedings of 4th ACM SIGCOMM workshop on Network and system support for games. *NetGames 2005*, 1–9. New York, NY, USA, ACM

[10] World Of Warcraft Subscriber Base Reaches 12 Million Worldwide http://us.blizzard.com/en-us/company/press/pressreleases.html?101007

[11] Jason Kincaid, (2009). Startup School: Wired Editor Chris Anderson On Freemium Business Models http://techcrunch.com/2009/10/24/startup-school-wired-editor-chris-anderson-on-freemium-business-models/

[12] Joan Goodchild, (2010). Social Engineering: The Basics http://www.csoonline.com/article/514063/social-engineering-the-basics

[13] http://definitions.uslegal.com/c/confidence-tricks%20

[14] Debra A. Valentine, (1998). The U.S. Federal Trade Commission on "Pyramid Schemes". http://www.ftc.gov/speeches/other/dvimf16.shtm

[15] Blizzard, (2010). World Of Warcraft Term Of Use. http://us.blizzard.com/en-us/company/legal/wow_tou.html

[16] Runescape, (2006). Runescape Security Book. http://runescape.wikia.com/wiki/Security_book

[17] Mark Ward, (2005). Warcraft game maker in spying row. http://news.bbc.co.uk/2/hi/technology/4385050.stm

[18] Battle.net Authenticator. http://us.blizzard.com/store/details.xml?id=1100000822

[19] Eve Online Career Options http://play.eveonline.com/en/getting-started/career-options.aspx