ONLINE GAMING AND SPAM

by

BRANDON DOUGLAS TREADWAY

(Under the Direction of Kang Li)

ABSTRACT

With the rise of interactions between players around the world through online gaming, there are many security issues in the games causing problems for the players. The issue of in-game spamming is typically overlooked by the development studios but can have a substantial impact on the game for both the players and the developers. The purpose of this thesis will be to identify the different types of spam, new and old, found within online games, the legal issues they present, the problems they create for developers and players. It will then discuss a few tactics to counter the issue of spam.

INDEX WORDS: Spam, Security, Gaming, Massively Multiplayer Online Role Playing Game

ONLINE GAMING AND SPAM

by

BRANDON DOUGLAS TREADWAY

B.S., The University of Georgia, 2007

A Thesis Submitted to the Graduate Faculty of The University of Georgia in Partial Fulfillment

of the Requirements for the Degree

MASTERS OF SCIENCE

ATHENS, GEORGIA

2010

ONLINE GAMING AND SPAM

by

BRANDON DOUGLAS TREADWAY

DEDICATION

To anyone who has had his or her account hacked in any MMO.

ACKNOWLEDGEMENTS

Thank you to Dr. Li for his guidance through the research process. Thank you to Farhan Jiva for helping me with the survey website. Thank you to my older sister Jessica Sisson, and brother-in-law, Joel Sisson, for their assistance in proofreading this document. Thank you to my younger sister Larissa Treadway for her help in making my life easier while conducting my research. Thank you to my girlfriend Amy for pushing me to get things accomplished. Most of all, thank you to my mother and father, James and Diana Treadway, for their love and support as I worked to complete my Master's program.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

Purpose of the Study

With online games rapidly becoming the most popular applications on the internet [8], security issues in these games are quickly becoming more evident. Game Studios tend to spend the majority of their time enhancing the game mechanics and reducing the latency of the game, yet game security seems to fall by the wayside. Due to the lack of time spent on increasing security, spam is prevalent in every online game. Spam messages within games are typically an attempt to offer services to the players, most of which go against the game's terms of use. The purpose of this study is to present the game studios with the issues that cause frustration for the players and trouble for the developers. While there are several different online game genres, all of which contain some sort of spam, this thesis will focus on Massively Multiplayer Online Role Playing Games (MMORPG). A MMORPG is a game that creates a virtual online community in which thousands of people interact with each other and includes virtual economies, professions and organizations. Other non-MMORPG games usually connect a small number people together to play a game; an example of this type of game is online poker. Most, non-MMORPG games do not require a subscription fee in order to play and can be accessed through the internet for free, whereas most MMORPGs require a monthly charge in order for the game player to continue to play the game. In order to better demonstrate the security issues within a MMORPG, this paper will focus mainly on the game *World of Warcraft*, created by Blizzard Entertainment, and its security issues, as The Guinness Book of World Records lists this game the most popular Massively Multiplayer Online Role Playing Game [6]. *World of Warcraft* currently has over eleven million subscribers [3] and is certainly susceptible to spam issues. Yet, even though *World of Warcraft* is the focus of most of the examples in this study, other games will be discussed, along with spam's effects on their game-play and development.

Related Work

Several papers and books have been written dealing with the security issues in online games, yet the issue of spamming is consistently omitted from these studies, as many do not consider it to be a threat to the game's security. However, spam can be used to obtain user identifications and passwords, making it a valid form of cheating and can lead to security issues within a game player's account. This type of security threat can be defined as Compromising Passwords or Social Engineering [10]. Several other related works deal with security issues on various other types of websites, but there is not a lot of current direct work relating to spam and security issues within online gaming. The majority of the research used in this thesis is original research conducted by playing the online games and experiencing and analyzing the spam issues firsthand.

CHAPTER 2

SPAM

What is Spam?

Spam messages are disruptive commercial messages posted on computer networks or sent out through e-mails. The most traditional forms of spam utilize a list of e-mail addresses to send a message that advertises a good or service to each person on the list.. These e-mail addresses are usually obtained when consumers sign up on a website. Many of the spam companies that generate spam messages share their e-mail list with partner spam companies, which allows the partner companies to send their own spam messages to the people on the list. While most of these e-mail messages contain links to remove your e-mail address from the list, selecting the link does not always remove your address from the list. However, most e-mail providers or other e-mail applications do provide some level of spam prevention or protection. This security measure is accomplished when the e-mail provider moves the messages to a separate spam or a junk mail folder. While this is a temporary fix, the user must maintain these folders by clearing them when the mailboxes become full.

One of the biggest security issues of spam e-mail is the threat of virus files or websites contained in the message. Some spam messages contain attachments with viruses embedded within, while others contain links to outside websites containing spyware or viruses. Again, many of the e-mail providers and applications have a built-in check for these files; however, this check is not guaranteed to always work.

Another form of spam is found in “spam bots,” which are associated with some form of instant-messaging program. The term “bot” is derived from robot, which is essentially what these programs represent. These bots generate messages periodically and send them out to chat rooms in order to advertise a good or service. There are also bots that will provide a “human” response to questions from the chat room user in order to carry on a conversation and draw in the user. While the bots that constantly spam the same message are easy to detect and block, the humanlike bots are harder to combat due to the similarity to a normal conversation these bots create.

Game Spam versus Traditional Spam

In truth, game spam is not much different from traditional spam; the main difference is the type of mailing list system in the game. Spam messages in online games are sent to the players using either a mailing list or a chat bot. These messages are meant to entice players to spend actual money on virtual items that can be used within the game. However, unlike the traditional spam system, messages sent through the game do not give the recipient the ability to remove his name from the mailing list. Once the spam company has the username it is never removed from the list, unless the player changes his username. Yet, the process for changing a username can be costly, or even impossible, making it unlikely that the player will be able to change the username at all. This gives the spammer the upper hand, as they are able to have continuous access to the username on the mailing list.

Furthermore, game spammers can also reach the user’s e-mail address in order to advertise their products. This occurs when a game site requires one to sign up in order to view specific pages like game guides or hints to help the player succeed in the game. The e-mail address is then added to a traditional spam mailing list; however it is possible to be removed from this mailing list.

Chat bots are another very common practice used by spammers in online games. The idea behind this process is to position a character in a high population area and, through the process of macros, generate spam messages, which are sent to all the players through the general chat systems. Macros are a built in feature of the game that allows the user to create a list of commands which can be repeated easily, usually with the press of one button. In addition, these chat bots can also be adapted to send the same spam messages through the private chat systems to individual users in order to entice them to spend money on virtual tools that can be used in the game. Some online games have mechanics built into the game that can be exploited for the purposes of spam. These mechanics can include a virtual billboard that has a message placed on it or even an item that can be bought or made, and this item can send spam out for the purchaser.

Table 1: Types of Spam Found in Popular MMORPGs

	World of Warcraft	Warhammer Online	Star Wars Galaxies	Second Life	Aion
Chat and Macros	X	X	X	X	X
Bots	X	X	X	X	X
In-Game Mail	X	X	X	X	X
Game Mechanics			X	X	X
Game Exploits	X				

CHAPTER 3

TYPES OF SPAM

In-Game Mail

Every MMORPG has a setup for the exchange of messages and items between users through a mail delivery system. Like traditional e-mail spam, in-game mail spam begins by creating a list of character names and distributing spam messages through those lists.   Once a spammer has the character name on a list, the player will continue to receive messages until he changes his character name, whereas most traditional e-mail spam messages provide a link to remove the address from the list. Some MMORPGs support name changes through a paid service, but most games will not let the user change his character name after the user has established his account. Another problem with MMORPGs and in-game spam mail is a lack of any kind of spam filter, spam folder, or junk mail folders, like those in a traditional e-mail account to catch these spam messages.

The MMORPG itself also creates a major difference in the delivery of these spam messages from that of traditional e-mail spam. The games are a closed environment and require an account and access to the game to be able to send messages. This means that the spam companies must be registered with the studio and logged into the system in order to get their messages out. Since the spam companies must be registered with the game, they are susceptible to the consequences of being caught, the punishment for which is most likely banishment from the game. However, the spam companies have means they can use to keep from getting caught, and they can continue to send out their in-game spam messages. In contrast, traditional e-mail spammers do not have to register with the same type of service or game as the user in order to send out their spam messages..

Another difference between in-game mail spam and traditional mail spam is how the users are added to the list. For traditional spam, the spam companies obtain an e-mail address through some other medium or some process of data mining. However, within a game a spammer can obtain the user's name by simply logging in and seeing the player online and adding the character name to a list; the process of adding user names to a list is slow but effective. This method of gathering user names is found in games that do not allow the user to use custom add-ons or scripting with the user interface. However, some games, like *World of Warcraft* and *Warhammer Online*, allow one to change the way the user interface appears through the use of Lua scripting, which has the ability to access certain game events and data storage, allowing the spammer to create a username list to be used for spamming during the game. In *World of Warcraft,* for example, there is a command in the game called the "/who" command. This command will allow the user to pass search parameters such as a major city, name, or location in the world, in order to create a list of players that meet a particular criterion. The game event thrown is called, "WHO_LIST_UPDATE," and changes a variable in the client's game state. This list can then be accessed to get certain types of information, such as character or guild name. Lua scripting also allows the add-on to output data to a saved variables folder for persistent data storage. Figure 1 illustrates how an add-on can be used in order to access a list of names and store them.

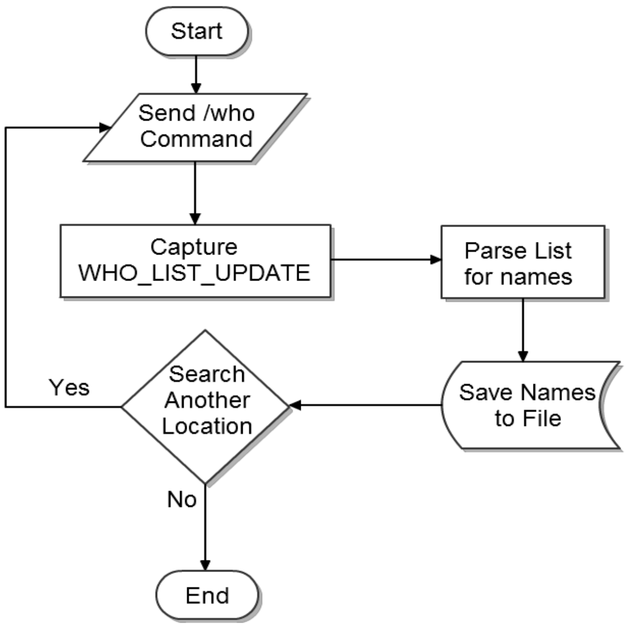


Figure 1: Add-on for Data Collection

This type of data collection can also be paired with a website provided by the game developers to gain access to even more user information. Some MMORPGs have a website setup in order to access information about characters, such as a gear or profession. Blizzard has provided a setup called "The Armory" for use in *World of Warcraft,* which can easily be accessed by using a Web Scraper. A web scraper is a program that will access a website, parse the code for data, and store the results for later use.

Yet, instead of using character names as a way to gather information, it is simpler for the spammer to do a web crawl by using guild names. A guild is a group of players that form an organization within the game and work together under one name. By using the add-on program described above, spammers can alter the programming to parse out guild names instead of character names, and then use a scraper throughout the hosting website to obtain user information. There are currently over 237 servers, also known as realms, used for *World of Warcraft* and many of the guild names are duplicated across the realms. By making the adjustment to the add-on to include the guild names, it is easier, and quicker, to obtain large mailing lists to use for spam messaging.

The process of data collection through web scraping on the websites run by the game studios would also be fairly simple method for the spammer to use to gain the character names of the users. In order for the spammer to be successful, the scraper would need to start with a list of guild names used in the game and go through all of the guild names individually. For each guild name, the scraper would perform a search for all guilds containing that particular name. The guild list created by the scraper is then set to parse out the guild name, the faction[1], the URL to follow, and the realm name. Once the scraper obtains a guild page, it can then access the list of all players who are in that particular guild. The number of player names in a guild can vary, and a decent sized guild can easily be around fifty or more members. The scraper can then save the names to a file thus creating a mailing list for spam messages. After the character list for one guild is obtained, the scraper will then move to the next match for the guild name on the list. When all guild matches have been checked for a particular name, the scraper moves to the next guild name on the input list and repeats the process. Figure 2 shows a flowchart of a web scraping program for *World of Warcraft*.

[1] *World of Warcraft* contains two factions that are at war, the Horde and Alliance. The faction is important as *World of Warcraft* does not allow intercommunication between different factions.

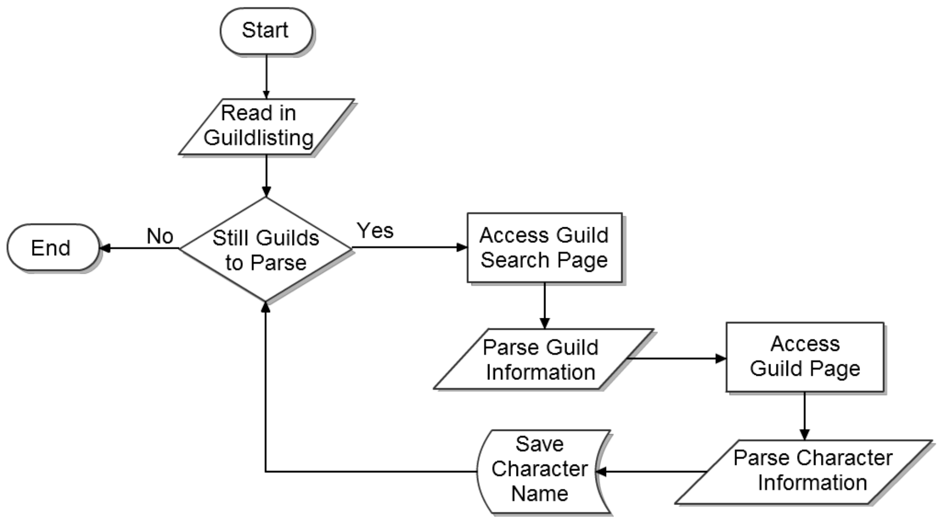


Figure 2: A Data Collection Web Scraper[2]

With an extensive guild-listing file, the scraper could easily obtain thousands of names in a matter of minutes. Table 2 shows the first ten results of this program using the names of the top two hundred ranked guilds.[3]

Table 2: Results from Web Scraping

Search Strings	Number of Guilds	Number of Players	Time (ms)
Premonition	123	1417	828
Adept	38	609	811
blood legion	223	1437	759
deus vox	60	235	268
cuties only	87	327	341
Exodus	216	4256	676
Vigil	53	743	221
Vodka	2	153	76
Might	56	634	223
Juggernaut	132	1299	520
Totals	990	11110	4723

[2] The source code for this program can be found in Appendix A
[3] Ranking based on Player verse Environment (PvE) according to www.wowprogress.com. The full table can be found in Appendix C

Using the table, the spammers will then want to get a list of guilds names that are more common as usually the more guild names you match, the more usernames you get for the spamming list. This is not always true, however, as the table above violates this hypothesis with the search strings ‘exodus’ and ‘blood legion.’ However, Figure 3 below displays the results of a scatter plot showing that, in general, that the more guilds you match the more names you get. This scatter plot is generated using the full table found in Appendix C and shows a linear trend upward.

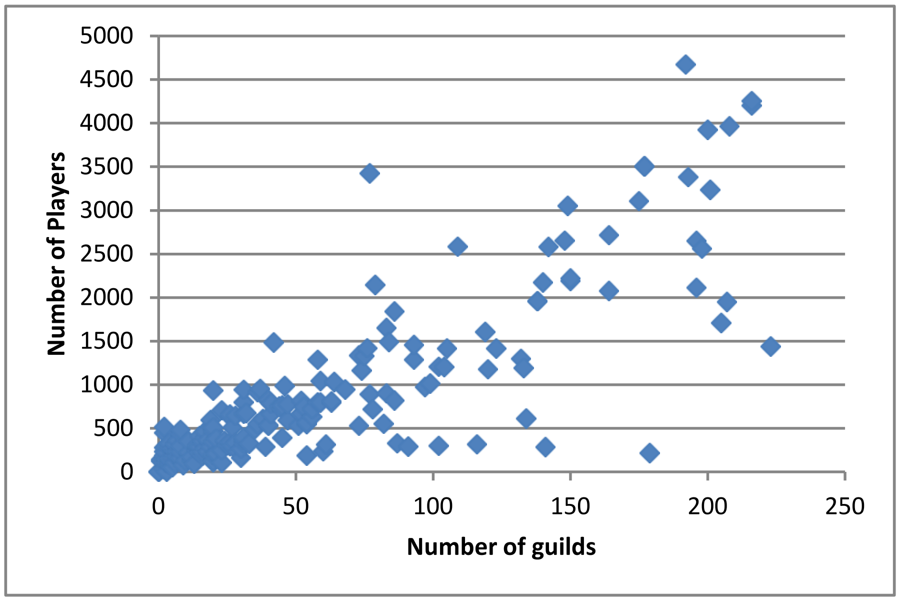


Figure 3: Number of Guilds Matched to Number of Players Scraped

Once the spammers have created a large mailing list, they can use an add-on[4], macro, or bot to distribute their spam messages. Since there is no way for the user to unsubscribe from the spammer’s list, a character will continually be spammed until the player moves servers or changes the character’s name. However, these temporary fixes will not always protect the user as the same web scraping tactics described above can be used to add the player to a new list.

[4] “Add-ons” are the common term for user customizations

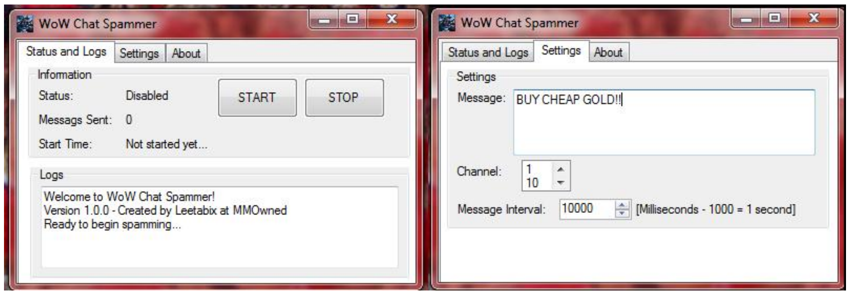
Chat and Macros

One of the purposes of an MMORPG is to provide a sense of community among the players. To do this, the MMORPG provides chat channels for the exchange of messages and socialization between players. However, spammers can use these chat channels to advertise their goods and services to everyone at once. In most cases, a spammer will create a character and then position their character in a large population area, such as a capital city in the realm, and spam the other players in the area. In order to make the task easier, the spammers setup a game mechanic called macros. Macros are used in games to provide a way for the player to easily chain together a series of commands that are then repeated to achieve maximum results. Some games also allow for macros to call themselves as a command, which allows for the continuous looping of the macros. Using these game mechanics, or macros, the spammer can spam his message to multiple channels with the press of one button. The player does have the ability to turn off certain chat channels, but by doing this the player is cut off from the other players in the game and is unable to communicate with their characters.

The chat system can also be exploited by spammers through the use of private messaging. MMORPGs have a system for personal messaging between two users; these systems are sometimes called a “tell” or “whisper.” Spammers can use the web scraper data collection method to send spam messages to users through the private messaging system. Some spammers will even try to disguise what they are doing by starting up a normal conversation about the game before sending their advertisements to the players. While this method of exploitation can be a slow process, it can be accelerated by automation through the use of bots.

Bots

Similar to macros, bots are used by the spammer to quickly distribute an advertisement; however, bots do not require a person to physically be at the computer. These bots contain a program that is run outside of the game and sends commands to message the spam to the players within the game ,and "are regarded as the most commonly-used and difficult-to-handle exploit."[5] These commands can be used in conjunction with the data collection methods listed above to spam individual users without the press of a button. Some more advanced bots are able to examine the memory space the game is occupying in order to access data from the game itself. Figure 4 below shows the setup for a chat spam bot that can be obtained for free from the Internet.



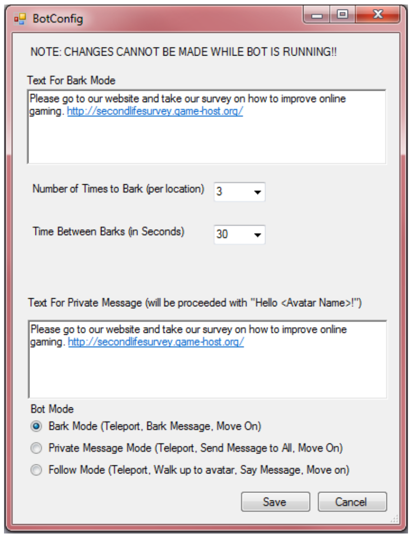
Information
Status: Disabled
Messags Sent: 0
Start Time: Not started yet...

Logs
```
Welcome to WoW Chat Spammer!
Version 1.0.0 - Created by Leetabix at MMOwned
Ready to begin spamming...
```

Settings
Message: BUY CHEAP GOLD!!
Channel: 1 10
Message Interval: 10000 [Milliseconds - 1000 = 1 second]

Figure 4: Setup of a Spam Bot [11]

Second Life is one of the few games that allows users to connect to the game system by using an open source version of the game, making it easier for spammers to create a bot that can access game data directly. Using the *Radegast* distribution of *Second Life* [9], a bot can easily be created that can directly access the game state variables. During my research, I created a bot using the *Radegast* distribution to perform tests on different styles of chat spam. The bot that I created has three different options for spamming messages: Barking, Private Message, and Follow. Figure 5 shows the configuration window for the bot.



BotConfig

NOTE: CHANGES CANNOT BE MADE WHILE BOT IS RUNNING!!

Text For Bark Mode

Please go to our website and take our survey on how to improve online gaming. http://secondlifesurvey.game-host.org/

Number of Times to Bark (per location): 3

Time Between Barks (in Seconds): 30

Text For Private Message (will be proceeded with "Hello <Avatar Name>!")

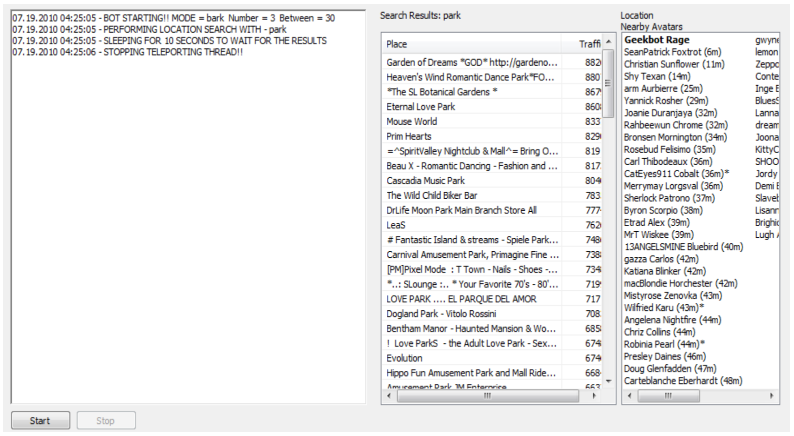
Please go to our website and take our survey on how to improve online gaming. http://secondlifesurvey.game-host.org/

Bot Mode

- [✓ Bark Mode (Teleport, Bark Message, Move On)]
- Private Message Mode (Teleport, Send Message to All, Move On)
- Follow Mode (Teleport, Walk up to avatar, Say Message, Move on)

Figure 5: Screenshot of the configuration window

In barking mode, the bot will teleport to a location and send a message out with a shout, viewable to everyone within 100 meters of the bot. The bot can be configured to set how many times it needs to 'bark' at each location and the duration between 'barks'. The second option for spammers to use is to send private messages through the bot. The bot teleports to a location and obtains a list of everyone in the simulation region. A personalized message is then sent to each of the avatars, after which the bot moves to a new location. Finally, the third option for a spammer is a follow bot. This mode teleports the bot to a location, similar to the barking and private messaging bot, however, once there, the follow bot will attempt to search for avatars within fifty meters of the bot's location. Once the bot finds an avatar, it will move to a location within five meters of the avatar and will send a personalized message through the whisper channel. The follow bot will then attempt to contact each avatar within a fifty mile range of the bot individually. However, if an avatar is beyond fifty meters from the bot, the bot will then teleport to the avatar's location and send the message, thus displaying its "following" function, as it can follow the avatars around the game in order to send them a private spam message through the whisper channel. Figure 6 below shows a view of the bot interface.



```
07.19.2010 04:25:05 - BOT STARTING!! MODE = bark  Number = 3  Between = 30
07.19.2010 04:25:05 - PERFORMING LOCATION SEARCH WITH - park
07.19.2010 04:25:05 - SLEEPING FOR 10 SECONDS TO WAIT FOR THE RESULTS
07.19.2010 04:25:06 - STOPPING TELEPORTING THREAD!!
```

Search Results: park

Place	Traffi
Garden of Dreams *GOD* http://gardeno...	882
Heaven's Wind Romantic Dance Park*FO...	880
*The SL Botanical Gardens *	867
Eternal Love Park	860
Mouse World	833
Prim Hearts	829
=^SpiritValley Nightclub & Mall^= Bring O...	819
Beau X - Romantic Dancing - Fashion and ...	817
Cascadia Music Park	804
The Wild Child Biker Bar	783
DrLife Moon Park Main Branch Store All	777
LeaS	762
# Fantastic Island & streams - Spiele Park...	748
Carnival Amusement Park, Primagine Fine ...	738
[PM]Pixel Mode : T Town - Nails - Shoes -...	734
*..: SLounge :.. * Your Favorite 70's - 80'...	719
LOVE PARK .... EL PARQUE DEL AMOR	717
Dogland Park - Vitolo Rossini	708
Bentham Manor - Haunted Mansion & Wo...	685
! Love ParkS - the Adult Love Park - Sex...	674
Evolution	674
Hippo Fun Amusement Park and Mall Ride...	668
Amusement Park JM Enterprise	663

Location
Nearby Avatars

Geekbot Rage	gwyne
SeanPatrick Foxtrot (6m)	lemon
Christian Sunflower (11m)	Zeppo
Shy Texan (14m)	Conte
arm Aurbierre (25m)	Inge E
Yannick Rosher (29m)	BluesS
Joanie Duranjaya (32m)	Lanna
Rahbeewun Chrome (32m)	dream
Bronsen Mornington (34m)	Joona
Rosebud Felisimo (35m)	KittyC
Carl Thibodeaux (36m)	SHOO
CatEyes911 Cobalt (36m)*	Jordy
Merrymay Lorgsval (36m)	Demi E
Sherlock Patrono (37m)	Slavel
Byron Scorpio (38m)	Lisann
Etrad Alex (39m)	Brighi
MrT Wiskee (39m)	Lugh
13ANGELSMINE Bluebird (40m)	
gazza Carlos (42m)	
Katiana Blinker (42m)	
macBlondie Horchester (42m)	
Mistyrose Zenovka (43m)	
Wilfried Karu (43m)*	
Angelena Nightfire (44m)	
Chriz Collins (44m)	
Robinia Pearl (44m)*	
Presley Daines (46m)	
Doug Glenfadden (47m)	
Carteblanche Eberhardt (48m)	

Figure 6: Screenshot of the Second Life bot interface

To further test the effects of this bot on gameplay, I created website with a survey for visitors to the game to take and describe which style of bot approached them during the game. The spam bot would approach the avatar and then direct them to the website that I created. A hit counter was also placed on the website to measure the number of visits by unique visitors. The experiment ran three bots in the same mode at the same time for three hours. Hits were measured in-between the modes as were the results. In the barking mode, the bot configuration was set to three 'barks' at a single location with an interval of thirty seconds between each one. The private messaging bot was set to appear at a location within a game to obtain a list of every character within that region of *Second Life*. Finally, the follow bot was set to teleport around different regions of the game and attempted to "whisper" message avatars within five meters of the bot. Table 3 shows the results from this experiment.

Table 3: Results from Second Life bot

Bot Mode	Number of Places Traveled	Number of Possible People	Number of Hits	Number of responses
Barking Mode	228	2515	23	11
Private Message Mode	788	7808	250	89
Follow Mode	138	1114	60	10

The results above show that the private messaging bots move quickly, as it does not take long to send the private message out to everyone in the region before moving on. This bot spamming method allows more people to be spammed in a shorter time frame. As a result, more people visited the website and responded to the survey. The follow bot is a newer type of bot designed specifically for this experiment. It is not common to see a bot walk up to your character and address it by name, due mainly to the game's built-in limitations, such as a difficulty in directly accessing game data. While it is the slowest moving of all three bot modes, it was effective in getting a larger number of hits on the website then the traditional barking mode. This response could be due to the personalized message delivered and the direct contact with the bot.

However, the downside to the follow bot is that people can send direct communications to the game studio about the spamming follow bot. As a result, during the three hours, the follow bot had two accounts banned by the game studio, whereas the barking mode bot only had one. The private message bot, while being the fastest moving bot, and most effective in reaching people, did have numerous suspensions, making it difficult to keep the bot running, and even more time consuming to start the bot back up. When an account is banned, a new account must be created in order to continue sending the spam. This is a slow process and requires that the spammer physically be at the computer.

My research demonstrates just how easy it is for the spammer to set up a spam bot and effectively send out their spam messages to the players in the game. As each bot mode has its pros and cons, it is up to the spam company to decide how to effectively deliver their messages. Based on the results of my experiment, the private message bot would most likely be the most effective method of sending out the spam.

Another purpose bots serve is to automate the process of farming or power leveling by the spam companies. Power leveling, also referred to as grinding, is a very time consuming process; yet bots can be used to increase the character levels while the provider is not present at the computer. Faming is a similar concept as it is the process of doing the same task repeatedly in order to accomplish something in the game. These leveling bots can also be used for the processes of farming or repeating the same task multiple times in order to acquire something of value. These are some of the services that spam companies will provide for payment.

Game Mechanics

Some spammers simply use the built in features of the game to get their message out to the players. *Star Wars Galaxies,* by Sony Online Entertainment, provides the ability to make vendors whose purpose is to allow players in the game to sell the items they make in their virtual professions. These vendors are activated by moving in close proximity to a character and are set up to display a message to the player when there are within the vendor's area. Spammers can utilize this game feature to their advantage by using it to advertise their websites.

Second Life, by Linden Research, Inc, is another game where the game mechanics allow spammers easy access for their messages. Within *Second Life*, there are numerous billboards and walls that contain advertisements for in-game and out-of-game services. *Second Life* also contains a large amount of dynamic loading, especially in relation to graphics; however, this creates latency issues when a player attempts to load a specific zone. All flat surfaces within the game can have customized content applied to them, including videos, pictures, and websites. Spammers can use this function to display advertisements for their own products. While Linden Research is working hard to monitor these signs, *Second Life* only has one server and has a vast virtual landscape making it difficult to catch all of the spam within the game.

Game Exploits

Game studios are beginning to take strides at reducing the amount of spam contained in their games, thus forcing spammers to get creative with their methods. Some spam companies have resorted to finding exploits or bugs within the games in order to get around the spam filters. Most recently, an exploit has been discovered in *World of Warcraft* that keeps the bodies of a dead character in the game until the character has resurrected; these bodies are still present even if the character is deleted. Figure 7 displays how the spammers are using this exploit to get their website names out to the players.

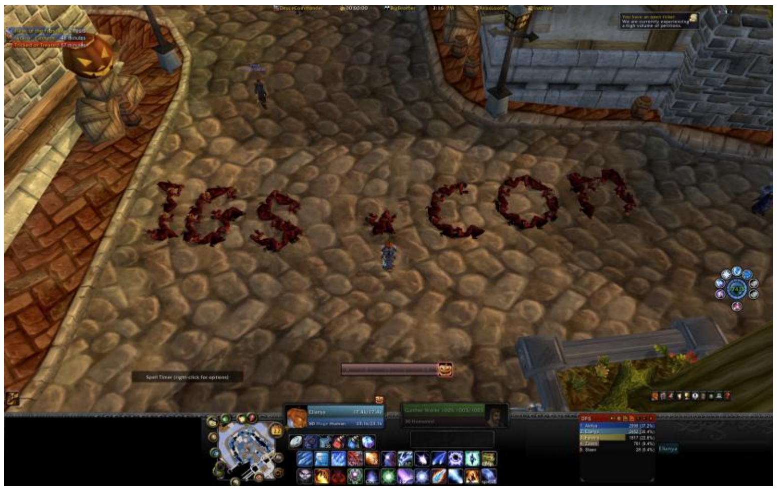


Figure 7: Using Corpses to Spell a Website[5]

By exploiting the “dead body bug,” spammers can continuously delete and create new characters in order to use the bodies to spell out spam messages. The figure above is a screenshot of one such spam message located within a capital city. This figure illustrates that if one method of spamming is blocked, the spam companies will find a new way around the security measure. While this is a slow process, it is very effective when done in a high population zone. Table 4 shows the approximate time needed to accomplish this task.

[5] Screenshot taken from the MMORPG *World of Warcraft*. The chat logs have been blocked out for privacy reasons.

Table 4: Creating Corpse Messages

Average Travel Time[6] (seconds)	Average Bodies per Letter[7]	Bodies per Period	Average time to Create website in figure 7
363.25	6	2	13803.5 (~ 4 Hours)

While it takes some time to position the characters and repeat the process, a bot can be written to automate the process. This puts a lot of pressure on the game studios to develop even newer counter-measures to protect their players from spamming. Blizzard has recently attempted to fix this bug, thus preventing this method to be used anymore. The fix causes the corpse to disappear after a character logs out of the game, and, if the character logs back in, the corpse is only visible to that player. While the corpse messages are still possible, it would require more than a hundred computers working together to accomplish the task of creating the spam message. This is one way that the game studios are trying to combat the game exploitations that the spam companies are using to get their spam messages into the game.

Though many of these types of spam can be found outside of the game client, either through e-mail or instant messaging clients, there are a few differences which place game spam into its own category. Table 5 illustrates both the similarities and the differences between traditional spam and game spam. It illustrates that, while there are some similarities between the two different forms of spam, online game communities provide a more closed-off environment. This closed environment forces the spam companies to become more creative in their tactics in order to get their spam out to the game players.

[6] Time to get from where the character is created to the nearest major city. Times located in Appendix D
[7] Letter breakdown can be found in Appendix D

Table 5: Differences and Similarities between Game SPAM and Traditional SPAM

Type of SPAM	Online Game Spam	Traditional Spam (IM, E-Mail)
Mail	• Commercial messages sent out through in-game mail • Closed environment that requires a subscription to the game	• Commercial messages sent out through E-mail • Only requires an e-mail provider
Chat	• Chat channel abuse • Closed environment within the game	• Chat room abuse • Semi-closed environments
Bots	• Used to repeatedly send out messages without user interaction • Ability to locate a person, move to them, and directly address them	• Used to send out messages without user interaction • Unable to move around and directly address the user
Game Mechanics	• Uses mechanics of the game to get out messages	• Banner type advertisements on a website or within an instant messenger client
Game Exploits	• Use of bugs or unexpected results in the game to send out messages (i.e. dead bodies)	• Not Present

CHAPTER 4

ISSUES CREATED FROM SPAM

Legal Issues

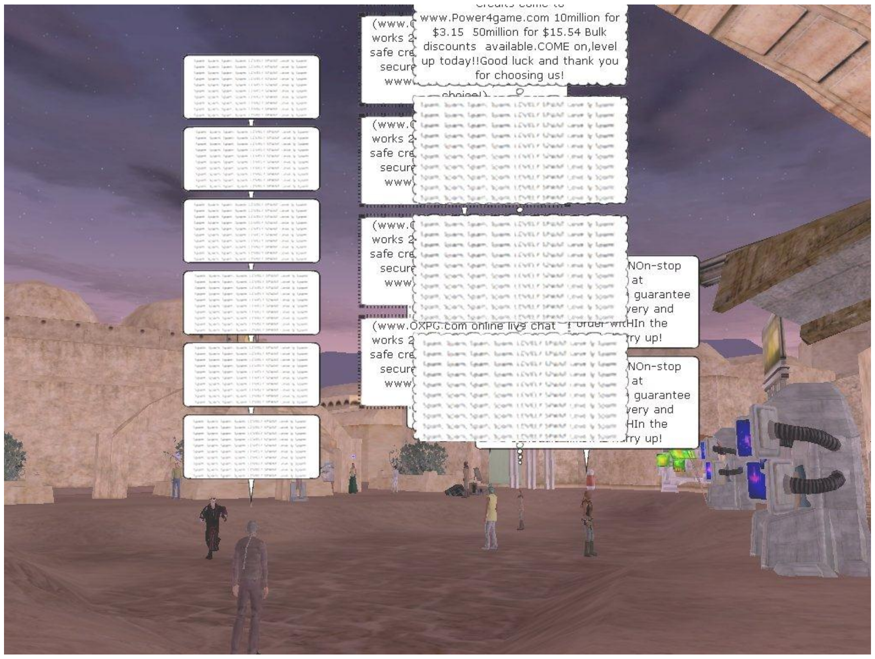
The majority of spam received during the game is related the selling of game resources and accessories, the purpose of which is primarily to make a profit for someone not connected with the creators of the game. However, for most online games, this is a direct violation of the terms of agreement to which every player in the game must agree. Section 2, part B of the *World of Warcraft* Terms of Use states that gamers are not allowed to:

> B. exploit the Game or any of its parts, including without limitation the Service, for any commercial purpose, including without limitation (a) use at a cyber cafe, computer gaming center or any other location-based site without the express written consent of Blizzard; (b) for gathering in-game currency, items or resources for sale outside the Game; or (c) performing in-game services in exchange for payment outside the Game, e.g., power-leveling; [4]

Other MMORPGs have a similar statement within their terms of use, rules of conduct, or other form of legal agreement made between the player and the game studio. The End User License Agreements also mention that any accounts found to violate or cheat the game will be banned from the game. This gives the studio the right to close and remove the spammer's accounts if they are found in violation of any part of the agreement. However, most of the spam companies are located in other parts of the world, making the pursuit of a legal action against these companies very difficult.

Player Annoyances

The purpose of a role-playing game is to provide people with a form of entertainment in which a person can become someone that they are not. Spammers, however, can change the way the game appears to the players. Chat spam can cause players to miss out on an activity they want to participate in during the game by filling up a player's chat windows and hiding messages sent between players and used to invite people to groups or other game events. In *Star Wars Galaxies*, chat messages appear above the heads of the characters. Figure 8 illustrates that multiple people spamming outside of a major area can cause an annoyance to the players by taking up the entire screen.



www.Power4game.com 10million for $3.15 50million for $15.54 Bulk discounts available.COME on,level up today!!Good luck and thank you for choosing us!

Figure 8: Chat Spam in Star Wars Galaxies[8]

Some games allow the player to turn off the chat bubbles above the characters heads, but the spamming can still block the chat frame window located in the user interface.

Virus infestations are another risk that players face, which can lead to an annoyance. Most of the spam messages that players receive contain website addresses for the players to visit in order to purchase the goods and services. Some of these websites contain viruses and spyware that can infect a person's computer. The primary type of virus found on these websites is a key logger virus that records the keyboard input and saves it to an encrypted file; this file is then returned to the person who distributed the virus. This type of virus is used to steal bank

[8] Screenshot obtained from http://swgblog.net/?m=200708

information or someone's identity. In the gaming industry, these viruses are created to target the acquisition of player's account information.

Within the game realm, account theft is the same concept as identity theft[9]. While this is a prevalent issue in games, the game studios put the account security in the hands of the player. Section 10 of the *World of Warcraft* terms of use states: "You are responsible for maintaining the confidentiality of the Login Information, and you will be responsible for all uses of the Login Information, whether or not authorized by you," [4]. Blizzard Entertainment does have a service that can be used to recover a stolen account; however, this process is very time consuming and can take several weeks. This recovery process is very frustrating to a player who is paying for a service he cannot access because the studio is researching the lost items.

Blizzard Entertainment has made an attempt to combat this type of identity theft. This protection is accomplished by the Blizzard Authenticator, which generates a six-digit pin number to be entered along with the password. While the actual encryption algorithm is kept secret, it is known that the supported types of algorithms are: DES, 3DES, and AES. In addition, every authenticator has a built-in clock and unique serial number used in the DIGIPASS algorithm. While this significantly reduces the number of accounts that are hacked, it still requires the player to purchase an additional piece of hardware in order to play the game. Recently Blizzard has acknowledged that a new virus has been released, using a different kind of attack method called a "Man in the Middle." This attack grabs the information typed by the player and sends it to the distributor of the virus, who then sends the information on to Blizzard. Figure 9 below illustrates how this attack is performed.

[9] During the process of my research, this account theft method actually happened to my own *World of Warcraft* account.

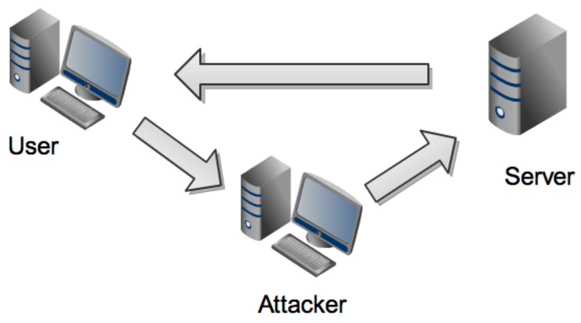


Figure 9: "Man in the Middle" Diagram

Since the information is correct, Blizzard logs the character in, and the player does not even know he has been hacked. The hacker, however, has only approximately sixty seconds in order to login to Battle.net[10], change the password and put a new authenticator on the account. While this still reduces the amount of accounts that are hacked, it is still possible for the player's account to be hacked without his knowledge.

The reason that account theft is important to spammers is that it provides a large source of in-game currency faster than farming would provide. When an account is hacked, all the character's items are sold, the gold (or currency) is split between the characters, and the characters are then transferred to the servers on which the spam companies need the currency most; in essence, the spammers gain control of the player's characters. The currency is then routed through many others characters as a form of embezzlement. This makes it difficult for game studios to discover where everything goes, making it difficult to block the spammer.

In addition to websites containing viruses, some of the websites that come through the spam are designed to look like the websites owned by the game studio. These websites are also sent to a player's personal e-mail address in a form that looks exactly like an official e-mail from the game studio. These websites appear to be entirely legitimate with the exception of where the

[10] Battle.net is the central location for all of a player's Blizzard Games. All the account management to change passwords and add and remove authenticators is done through this website.

data is sent. Yet, through simple investigation of the headers contained in the e-mail, the source code reveals that these messages are not legitimate. Still, the average player is not able to view and understand the source code in the headers, or on the websites, and the player falls prey to the spam companies.

Developer Issues

Spam within video games can cause problems for the development studios. Exploits and bugs used to cheat and spam within the game cause the game studios to spend more time on fixing older parts of the games instead of developing new content for the game. This can slow down the development cycle of the game and upset the players, as they are not given anything new to play; this can lead to the loss of players as they become bored with the game.

Spam can have a negative monetary impact on the game studios. If the clients are upset about the issues in the game, developers will see an increase in customer complaints to the customer service representatives. As a result, the studio must hire more representatives and extend the hours of the services in order to handle the larger quantity of complaints. In extreme cases, some players will get so upset because of these issues that they will stop playing the game entirely, thus cancelling their subscription fee. This loss of revenue is the worst possible situation that can happen to the gaming studio and can result in a loss of income or even the eventual death of the game.

CHAPTER 5

COUNTER-MEASURES

Game Masters

Game masters are a part of every online game; these are not professional players of the game but are actual employees of the studio. They serve as customer service representatives within the game to resolve issues that may arise. Essentially, they make up the police force of the game studio. The game masters also monitor certain types of spam they encounter in the game. Primarily, they monitor the game for the use of bots and macros used to generate spam. The game masters' characters cannot be seen within the game, thus the players never know where he is located. Game masters monitor major population areas and ban players who send spam messages through the chat channels. They can also remove the spam caused by exploitation of programming flaws and bugs. However, the virtual worlds in the more popular MMORPGs are very large, and game masters cannot be in all places at once.

While game masters can take care of spam issues they visually observe, spam issues can also be reported through the in-game ticketing system. This ticketing system is a way to petition the game studio for help. This is frequently the only way to directly contact game masters, as their identities remain private. The downside to this ticketing system, is that most studios do not have large amounts of game masters, so the response time to a ticket can be very long. Currently, game masters are not the best counter-measure to combat the spam found in chat messages.

The Warden

Blizzard has also implemented a controversial form of monitoring its game, known as the warden. The warden's sole purpose is to police the game and watch for cheating and the use of third party software to gain an advantage in the game. However, the warden borders on the lines of spyware, as it not only monitors the memory space of *World of Warcraft*, it also monitors other processes running on the player's computer [7]. Section six of the *World of Warcraft* end user license agreement contains a clause giving consent for Blizzard to monitor the random access memory on the computer during game play, in order to watch for third party cheats [2]. Therefore, in order to play the game, the player must give Blizzard permission to access other processes on the player's computer, thus making it similar to spyware.

Every now and then the warden will activate on a player's computer and will scan all the open processes in memory for the title of open windows, URLs that may be opened in browsers, and even the code for other programs that are running [7]. This information is then sent back to Blizzard to investigate if the player is using a cheat. Primarily, the warden is a counter-measure used to detect bots and other programs running outside of the game that could be altering the game's memory or sending commands to the game for an automation process.

Chat Spam Filters

The best way to combat spam found in the chat system is to apply a filter. This method is one that should be implemented by the developers; however, most game studios do not have such a spam filter in place. *Aion*, by NCSoft, is a game that was flooded with gold spam when it was released in the United States. Yet, this MMORPG was one of the first to implement an effective chat spam filter in the game. By using this spam filter, the developers have practically eliminated spam messages found in the public channels.

Another way to filter spam is by using the ignore list built-in to most games. This is a list of character names from which one does not see any messages, private or public. However, most games have a limitation on how many names you can have on this list. Spammers are constantly creating new character names when old ones are blocked, thus it is a never-ending cycle of one spammer name being ignored while a new spammer name is created. Therefore, this method of spam filtering is not very effective in blocking spam messages in the game.

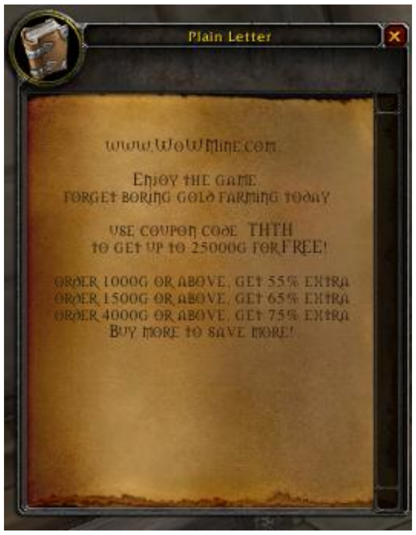
World of Warcraft has added a feature to the chat system that allows a player to report another character as spam. However, this only puts the person on a twenty-four hour ignore, while Blizzard's game masters investigate the character. If Blizzard does not complete the investigation of the reported character within the twenty-four hours, the name is removed from the ignore list, and the player can continue to receive spam messages from that name. This is a temporary fix to a larger problem and puts the majority of the work on the player to actually report the spammer.

Games with customizable features, like *World of Warcraft*, have one other way to combat spam through the use of a third party add-on. These add-ons monitor the incoming messages the chat window receives and filters out messages based on keywords commonly used by spammers. The add-on also takes other factors into account, such as the level of the character sending the message and blacklisting, in order to block incoming spam messages. Still, these add-ons are not perfect and can actually block messages that are legitimate and ones the player would want to see.

Proposed E-mail Filter

With every MMORPG, e-mail is the best way to send spam messages, as no MMORPG has a filter on these messages. *World of Warcraft* has added a way to report spam e-mail messages similar to the way a player reports chat spam. Just like the chat report system, this method puts the reported character name on a twenty-four hour ignore while the game masters investigate.

When looking into different types of filtering systems, two options present themselves, Bayesian filtering and blacklisting. Spam blockers for traditional e-mail services often use Bayesian filtering. This type of filtering involves calculating the possibility of a message being spam by checking each word in the message and the probability of each word being contained in a spam message. This method can be difficult to implement to protect against game spam sent through e-mail, especially in a game where spammers use words commonly included in legitimate messages. Also, extensive research must be done in order to calculate all the probabilities used in the equation. The calculation process can also be time consuming depending on the depth of the word dictionary and the time it takes to check each words probability. Figure 10 shows an example of a spam mail message received in *World of Warcaft*, and the difficulty the Bayesian filter might have when trying to calculate and parse out the words. For Bayesian to work effectively, every word must be parsed and weighted before the calculations can be made.



www.WoWMine.com

Enjoy the game
Forget boring gold farming today

Use coupon code THTH
to get up to 25000G for FREE!

Order 1000G or above, get 55% extra
Order 1500G or above, get 65% extra
Order 4000G or above, get 75% extra
Buy more to save more!

Figure 10: Spam E-mail in World of Warcraft

A simpler filtering system can be implemented without putting too much strain on the game studio, and this system is called blacklisting. All of the spam e-mail messages sent through the game contain a website for the player to visit. While the spam companies are devising new ways to present the website, it is still fairly simple to extract a website from a string variable. By using blacklisting, the website can be checked against a list of known websites to determine whether or not the messages should be blocked. In the event the website is not on the blacklist, the filter can then get the source for the index page on the website and extract the Title and Meta data tags. These tags contain information about the material contained in the website. By looking for certain key words and phrases, the system can either add the website to the blacklist or allow the website to continue sending messages. In the event that the content of the webpage cannot be determined, the system can submit the web address to the game masters for approval or blocking. While this could delay the delivery of a legitimate message, it would greatly reduce the amount of spam messages that are delivered to the player. Figure 11 depicts a program that uses blacklisting and HTML tag checking to block certain websites.

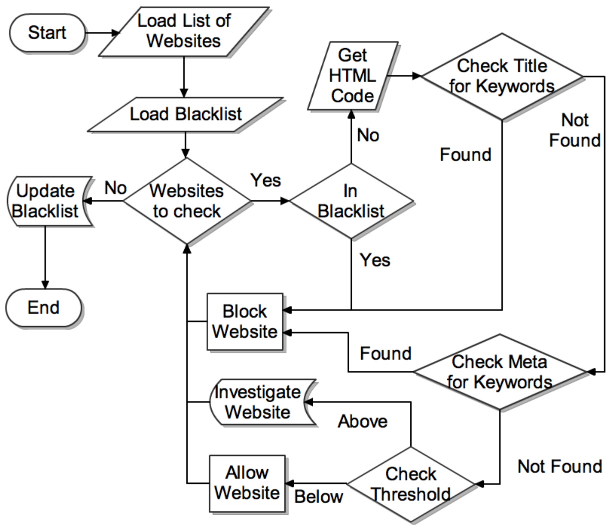


Figure 11: Blacklist Program[11]

While testing this program, sixteen different websites were analyzed using the HTML keyword checking. Ten known gold selling and power leveling websites were used, as well as three

[11] The source code for this program can be found in Appendix B

personal websites and three websites containing information about gold. Table 6 shows the results from running these websites through the program shown above in Figure 10.

Table 6: Results from Blacklisting

	Known Websites for Selling gold	Personal Websites	Similar Websites Involving Gold
Total Sites Tested	10	3	3
Allowed	0	3	2
Blocked	10	0	0
Investigate	0	0	1

Table 5 illustrates that blacklisting is an effective form of blocking spam messages from getting through to the player. However, there are some drawbacks to the blacklisting filter; these drawbacks are the delay in the blocking of a message due to a large blacklist and the occasional false positive. Yet, if the game studio has a powerful enough system, the delay issue can be overcome, as the blacklist can be scanned at a faster rate. This method can also be effectively applied to chat messages sent within the game, as long as the delay between the blacklist scan and the blocking of the message is minimal.

Both the Bayesian and blacklist methods would help to reduce the amount of spam messages received by the player. Bayesian filtering can be effective but very slow as the system has to determine the probability of each word and then do a complex calculation. While this method would be effective for in-game mail messages, it would not be efficient for instant messages. Blacklisting is the better option for the game studios to implement, as it provides a similar and more efficient way to successfully block spam messages both in-game mail and instant.

CHAPTER 6

CONCLUSION

Spam has become an ever-present problem in online gaming. As long as there is a demand for online game programs, there will great supply of spammers ready to take advantage of the players. This thesis has identified the immediate issues with spam and the problems they can cause for both the player and the game studio. Blizzard Entertainment has taken steps to stop spamming services by monitoring transactions conducted during the game in an effort to find non-legitimate sources of items and users dealing in online gold. Blizzard has stated the following: “Players who buy gold are supporting spamming, botting, and keylogging – activities that diminish the gameplay experience for everyone,” [1]. Yet, the current counter-measures are only temporary fixes to a larger problem. In-game mail spam is also a security issue that most of the game studios have yet to take any action to prevent. Two different forms of filtering have been presented and should be investigated in order to observe the effects they have on the game itself. One such method of filtering, blacklisting, has been shown as a viable option to help reduce the amount of spam sent through the in-game mail system.

The only way that studios can protect their players from spamming is to invest the money and time necessary to create anti-cheating and spam-blocking programs. By creating a more secure environment, the players will be more likely to continue their accounts. Therefore, if the game studios spend more time on the prevention of spam and cheating, the studios will be able to create a better and more secure game environment for their players, thus giving the players a better experience overall.

REFERENCES

[1] Blizzard, “Gold Selling: Effects and Consequences”, available at http://www.wow-europe.com/en/info/faq/antigoldselling.html.

[2] Blizzard, “*World of Warcraft* End User License Agreement”, available at http://www.worldofwarcraft.com/legal/eula.html.

[3] Blizzard, “*World of Warcraft* subscriber Base Reaches 11.5 Million Worldwide”, available at http://eu.blizzard.com/en/press/081223.html.

[4] Blizzard, “*World of Warcraft* Terms of Use”, available at http://www.worldofwarcraft.com/legal/termsofuse.shtml.

[5] Steven Gianvecchio and Zhenyu Wu, et al. “Battle of Botcraft: fighting bots in online games with human observational proofs”, in *Proceedings of the 16th ACM conference on Computer and communications security*, November, 2009

[6] Guinness, “Guinness World Records Gamer’s Edition”, available at http://gamers.guinnessworldrecords.com/records/pc_gaming.aspx.

[7] Greg Hoglund and Gary McGraw, Exploiting Online Games: Cheating Massively Distributed Systems, Boston: Pearson Education Inc, 2008.

[8] S McCreary and K Claffy, “Trends in Wide Area IP Traffic Patterns: A View from Ames Internet Exchange”, in *Proceedings of the ITC Specialist Seminar on IP Traffic Modeling, Measurement, and Management*, Montery, CA, USA, Sept. 2000.

[9] “*Radegast Openmetaverse Client*”, available at http://radegastclient.org/wp/

[10] Jeff Yan and Brian Randell, “A Systematic Classification of Cheating in Online Games”, in *Proceedings of 4th ACM SIGCOM workshop on Network and system support for games*, October, 2005.

[11] “*WoW Chat Spammer*”, available at http://www.wowbootybay.com/2009/02/21/wow-chat-spammer/

APPENDIX A

WEB SCRAPER SOURCE CODE

```
/**
 * GuildScraper.java
 *
 * Part of the software tool that will scrape data
 * from the wow armory in order to obtain player names
 */

package wowarmoryscraper;

// imports
import java.io.BufferedReader;
import java.io.File;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.net.MalformedURLException;
import java.util.ArrayList;
import javax.xml.parsers.SAXParser;
import javax.xml.parsers.SAXParserFactory;
import org.xml.sax.Attributes;
import org.xml.sax.SAXException;
import org.xml.sax.helpers.DefaultHandler;

/**
 * This files scrapes the guild names from the search page
 *
 * @author Brandon Treadway
 */
public class GuildScraper extends DefaultHandler
{
    ArrayList<Guild> guilds;        // list of guilds parsed

    ArrayList<String> guildList;    // list of guilds to search

    ArrayList<Results> resultsList; // list of results

    private Guild tempGuild;        // placeholder object

    public int count;               // counts the number of names scraped

    /**
     * Default constructor. Intializes lists and sets counter variable to 0
     */
    public GuildScraper()
    {
        guilds = new ArrayList();
        guildList = new ArrayList();
        resultsList = new ArrayList();
        count = 0;
    }
```

```
/**
 * Parses the guild listing file and then generates the search. Search each
 * guild namein the file
 */
public void getGuilds()
{
   // get the guilds from the file listing
   try
   {
      // open file for reading
      File guildListing = new File("GuildListing.dat");
      FileReader fr = new FileReader(guildListing);
      BufferedReader reader = new BufferedReader(fr);

      // line read from file
      String line = null;

      // read all the names from the file
      while((line=reader.readLine()) != null)
      {
         guildList.add(line);    // add to the guild list
      }

      // close the files
      reader.close();
   }
   catch(Exception e)
   {
      e.printStackTrace();
   }

   // loop through the guild list and parse each one.
   for(int i = 0 ; i < guildList.size() ; i++)
   {
      GuildScraper gsTemp = new GuildScraper();   //create a new scraper

      double startTime = System.currentTimeMillis();  // start time
      gsTemp.parseDocument(guildList.get(i));         // parse the search page
      double endTime = System.currentTimeMillis();    // end time
      gsTemp.printData();                             // print results

      Results rs = new Results();     // create an empty results object
      rs.search = guildList.get(i);   // set the search string
      rs.numberOfPlayers = gsTemp.count;          // get player count
      rs.numberOfGuilds = gsTemp.guilds.size();   // get guild count
      rs.timeToParse = endTime - startTime;       // get time to parse

      resultsList.add(rs);     // add to results list
   }

   printHTML();    // print the HTML result file
}
```

```
/**
 * This is the method that actually uses a SAX XML parser to get the
 * character tag from the document and parse the attributes.
 */
private void parseDocument(String guildName)
{
   //get a factory
   SAXParserFactory spf = SAXParserFactory.newInstance();
        try
   {
      // get a new instance of parser
      SAXParser sp = spf.newSAXParser();

      // parse the search page from the website
      sp.parse("http://www.wowarmory.com/search.xml?searchType=all&" +
            "searchQuery=" + guildName + "&selectedTab=guilds", this);
   }
   catch (Exception e)
   {
      //e.printStackTrace();

      // keep attempting to get the page if the server rejects the request
      parseDocument(guildName);
   }
}

/**
 * Prints the data results from the parsing
 */
private void printData()
{
   // total number of guilds found that match the seach
   System.out.println("No of guilds: " + guilds.size());

   // loop through each guild in the list
   for(int i = 0 ; i < guilds.size() ; i ++)
   {
      // delay to avoid connection refusal
      try
      {
         Thread.sleep(1000);
      }
      catch (InterruptedException ex)
      {
         ex.printStackTrace();
      }

      // get a new player scraper object for the guild object
      PlayerScraper ps = new PlayerScraper(guilds.get(i));

      // print the guild data
      System.out.println("GUILD NAME: " + guilds.get(i).getName());
      System.out.println("REALM NAME: " + guilds.get(i).getRealm());
      System.out.println("FACTION: " + guilds.get(i).getFaction());

      // parse the player names from the guild page
```

```
            ps.getPlayers();

            // spacers
            System.out.println("****************************************");
            System.out.println();

            // increment count
            count = count + ps.getSize();
        }

        // display the total number of players scraped
        System.out.println("TOTAL PLAYERS SCRAPED: " + count);
    }

    /**
     * Prints the result list to an html file for easier viewing
     */
    public void printHTML()
    {
        // try to write to the html file
        try
        {
            // open the file for writing
            FileWriter fw = new FileWriter("results.html");

            // write the basic html title stuff
            fw.write("<html><title>Scraping Results</title>\n");
            fw.write("<body>\n");

            // start the tables
            fw.write("<table border=3>\n");

            // table headers
            fw.write("<tr>\n<td>SEARCH STRING</td>\n<td>NUMBER OF GUILDS</td>" +
                    "\n<td>NUMBER OF PLAYERS</td>\n<td>TIME (MS)</td></tr>\n");

            int totalGuilds = 0;    // total number of guilds in the results
            int totalPlayers = 0;   // total number of players in the results
            double totalTime = 0.0; // total time to complete all parses

            // Loop through the result list and print the table rows
            for(int i = 0 ; i < resultsList.size() ; i ++)
            {
                // write the table row to the file
                fw.write("<tr><td>" + resultsList.get(i).search + "</td>" +
                        "<td>" + resultsList.get(i).numberOfGuilds + "</td>" +
                        "<td>" + resultsList.get(i).numberOfPlayers + "</td>" +
                        "<td>" + resultsList.get(i).timeToParse +
                        "</td></tr>\n");

                // increment the counting variables
                totalGuilds = totalGuilds + resultsList.get(i).numberOfGuilds;
                totalPlayers = totalPlayers + resultsList.get(i).numberOfPlayers;
                totalTime = totalTime + resultsList.get(i).timeToParse;
            }
            // close the table
            fw.write("</table><br /><br />\n");

            // print the total stats
            fw.write("<h2>TOTAL SEARCHES: " + resultsList.size() + "<br />\n");
            fw.write("TOTAL NUMBER OF GUILDS: " + totalGuilds + "<br />\n");
            fw.write("TOTAL PLAYERS SCRAPED: " + totalPlayers + "<br />\n");
            fw.write("TOTAL TIME: " + totalTime + " milliseconds<br />\n");

            // close all other tags
            fw.write("</h2></body></html>");

            // close the file to save it
            fw.close();
        }
        catch(Exception e)
        {
            e.printStackTrace();
        }
    }

    /**
     * OVERRIDE METHOD
     * gets the start of an element with the provided name
     * @param uri
     * @param localName
     * @param qName
     * @param attributes
     * @throws SAXException
     */
    public void startElement(String uri, String localName, String qName,
            Attributes attributes)
        throws SAXException
    {
        // find the guild tags in the xml document
        if(qName.equalsIgnoreCase("guild"))
        {
            // create a new instance of guild
            tempGuild = new Guild();

            // extract the attributes from the tag
            tempGuild.setFaction(attributes.getValue("faction"));
            tempGuild.setName(attributes.getValue("name"));
            tempGuild.setRealm(attributes.getValue("realm"));
            tempGuild.setURL(attributes.getValue("url"));

            // add the guild to the list
            guilds.add(tempGuild);
        }
    }
```

```
    /**
     * Main Method for running the application
     * @param args
     * @throws MalformedURLException
     * @throws IOException
     */
    public static void main(String[] args)
        throws MalformedURLException, IOException
    {
        // pretend to be firefox browser so we get the actualy XML and not
        // the HTML
        System.setProperty("http.agent",
                "Mozilla/5.0 (Macintosh; U; Intel Mac OS X 10.6; en-US; " +
                "rv:1.9.2.3) Gecko/20100401 Firefox/3.6.3");

        // create a new guild scraper object
               GuildScraper gs = new GuildScraper();

        // get the data
               gs.getGuilds();
    }
}
```

```
/**
 * PlayerScraper.java
 *
 * Part of the software tool that will scrape data
 * from the wow armory in order to obtain player names
 */

package wowarmoryscraper;

// imports
import java.io.File;
import java.io.FileWriter;
import java.util.ArrayList;
import javax.xml.parsers.SAXParser;
import javax.xml.parsers.SAXParserFactory;
import org.xml.sax.Attributes;
import org.xml.sax.SAXException;
import org.xml.sax.helpers.DefaultHandler;

/**
 * This files scrapes the player names from the guild page
 *
 * @author Brandon Treadway
 */
public class PlayerScraper extends DefaultHandler
{
    ArrayList<Player> players;  // list of players

    private Player tempPlayer;  // place holder class

    private Guild theGuild;     // the guild object we are scraping

    /**
     * The default constructor. Takes a guild object and creates a player
     * scraper
     *
     * @param g The guild object we are scraping
     */
    public PlayerScraper(Guild g)
    {
        players = new ArrayList();
        theGuild = g;
    }

    /**
     * Method to actually parse the players from the xml sheet
     */
    public void getPlayers()
    {
        parseDocument();    // parse the document and get characters
        printData();        // print results
    }

    /**
     * Gets the number of characters in the list
     *
 * @return the size of the list
 */
public int getSize()
{
   return players.size();
}

/**
 * This is the method that actually uses a SAX XML parser to get the
 * character tag from the document and parse the attributes.
 */
private void parseDocument()
{
   //get a factory
   SAXParserFactory spf = SAXParserFactory.newInstance();
        try
   {
      // get a new instance of parser
      SAXParser sp = spf.newSAXParser();

      // parse the xml document with the character listing on it
      sp.parse("http://www.wowarmory.com/guild-info.xml?"
            + theGuild.getURL(), this);
   }
   catch (Exception e)
   {
      //e.printStackTrace();

      // keep trying if the server rejects it.
      parseDocument();
   }
}

/**
 * Write the names to the files based on their faction and realm
 */
private void printData()
{
   // print the number of characters in the list
   System.out.println("No of players: " + players.size());

   try
   {
      //  create the horde directory if it doesn't exist
      File f = new File("Horde");

      if(!f.exists())
         f.mkdir();

      // create the alliance directory if it doesn't exist
      f = new File("Alliance");

      if(!f.exists())
         f.mkdir();

      // Open the file for appending
```

```
            FileWriter fw = new FileWriter(theGuild.getFaction() + "/"
                    + theGuild.getRealm() + ".dat", true);

            // write the names to the file
            for(int i = 0 ; i < players.size() ; i++)
            {
                fw.write(players.get(i).getName() + "\n");
            }

            // close the file
            fw.close();
        }
        catch(Exception e)
        {
            e.printStackTrace();
        }
    }

    /**
     * OVERRIDE METHOD
     * gets the start of an element with the provided name
     * @param uri
     * @param localName
     * @param qName
     * @param attributes
     * @throws SAXException
     */
    public void startElement(String uri, String localName, String qName,
            Attributes attributes)
        throws SAXException
    {
        // get the character tag
        if(qName.equalsIgnoreCase("character"))
        {
            // create a new instance of guild
            tempPlayer = new Player();

            // extract the attributes from the tag
            tempPlayer.setFaction(theGuild.getFaction());
            tempPlayer.setName(attributes.getValue("name"));
            tempPlayer.setRealm(theGuild.getRealm());

            // add the object to the list
            players.add(tempPlayer);
        }
    }
}
```

```
/**
 * Player.java
 *
 * Part of the software tool that will scrape data
 * from the wow armory in order to obtain player names
 */

package wowarmoryscraper;

/**
 * This file represents a player object
 *
 * @author Brandon
 */
public class Player
{
    String name;        // name of the character
    String faction;     // name of the faction
    String realm;       // name of the realm

    /**
     * Gets the faction of the player object
     *
     * @return faction name
     */
    public String getFaction()
    {
        return faction;
    }

    /**
     * Sets the faction of the player object
     *
     * @param faction
     */
    public void setFaction(String faction)
    {
        this.faction = faction;
    }

    /**
     * Gets the name of the player object
     *
     * @return player name
     */
    public String getName()
    {
        return name;
    }
    /**
     * Sets the name of the player object
     *
     * @param name
     */
    public void setName(String name)
    {
        this.name = name;
    }

    /**
     * Gets the realm name of the player object
     *
     * @return realm name
     */
    public String getRealm()
    {
        return realm;
    }

    /**
     * Sets the realm name of the player object
     *
     * @param realm
     */
    public void setRealm(String realm)
    {
        this.realm = realm;
    }
}
```

```
/**
 * Guild.java
 *
 * Part of the software tool that will scrape data
 * from the wow armory in order to obtain player names
 */

package wowarmoryscraper;

/**
 * This file represents a guild object
 *
 * @author Brandon
 */
public class Guild
{
    String Name;        // name of the guild
    String Realm;       // name of the realm the guild is on
    String Faction;     // name of the faction for the guild
    String URL;         // url to the guild page

    /**
     * Gets the URL of the guild
     *
     * @return guild url
     */
    public String getURL()
    {
        return URL;
    }

    /**
     * Sets the url for the guild object
     *
     * @param URL
     */
    public void setURL(String URL)
    {
        this.URL = URL;
    }

    /**
     * Gets the faction for the guild object
     *
     * @return guild faction
     */
    public String getFaction()
    {
        return Faction;
    }
    /**
     * Sets the faction for the guild object
     *
     * @param Faction
     */
    public void setFaction(String Faction)
    {
        this.Faction = Faction;
    }

    /**
     * Gets the name of the guild object
     *
     * @return guild name
     */
    public String getName()
    {
        return Name;
    }

    /**
     * Sets the name of the guild object
     *
     * @param Name
     */
    public void setName(String Name)
    {
        this.Name = Name;
    }

    /**
     * Gets the name of the realm the guild is located on
     *
     * @return realm name
     */
    public String getRealm()
    {
        return Realm;
    }

    /**
     * Sets the realm name for the guild object
     *
     * @param Realm
     */
    public void setRealm(String Realm)
    {
        this.Realm = Realm;
    }
}
```

```
/**
 * Results.java
 *
 * Part of the software tool that will scrape data
 * from the wow armory in order to obtain player names
 */

package wowarmoryscraper;

/**
 *
 * @author Brandon
 */
public class Results
{
    String search;
    int numberOfGuilds;
    int numberOfPlayers;
    double timeToParse;
}
```

APPENDIX B

BLACKLIST PROGRAM SOURCE CODE

```
/**
 * BlackListFilter.java
 *
 * This program simulates the filtering system using blacklisting.  It first
 * checks the blacklist to see if the website is already contained in it. If it
 * is not, then the program checks the html title and meta tags for search
 * strings.
 */

package blacklistfilter;

// imports
import java.io.BufferedReader;
import java.io.File;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.InputStreamReader;
import java.net.HttpURLConnection;
import java.net.URL;
import java.util.ArrayList;
import java.util.Locale;
import java.util.regex.Matcher;
import java.util.regex.Pattern;

/**
 * Main class of the prgram and represnets a filter object
 *
 * @author Brandon Treadway
 */
public class BlackListFilter
{
    ArrayList<String> blacklist;    // the blacklist

    ArrayList<String> websites;     // list of websites to check

    ArrayList<String> mainSearchStrings;        // main search stings list

    ArrayList<String> secondarySearchStrings;   // other search stings

    ArrayList<String> investigate;  // the further investigation list

    final int THRESHOLD = 2;        // threshold value for investigation

    /**
     * Default Constructor - Initializes lists
     */
    public BlackListFilter()
    {
        // Initialize list
        blacklist = new ArrayList();
        websites = new ArrayList();
        mainSearchStrings = new ArrayList();
        secondarySearchStrings = new ArrayList();
        investigate = new ArrayList();

        // populate the main searh strings
        mainSearchStrings.add("wow gold");
        mainSearchStrings.add("cheap wow gold");
        mainSearchStrings.add("warcraft gold");
        mainSearchStrings.add("power level");

        // populate the secondary search stings
        secondarySearchStrings.add("gold");
        secondarySearchStrings.add("wow");
        secondarySearchStrings.add("cheap");
        secondarySearchStrings.add("leveling");
    }

    /**
     * Load the blacklist from a file into a local list
     */
    public void LoadBlackList()
    {
        try
        {
            // open the file for reading
            File blackL = new File("blacklist.dat");
            FileReader fr = new FileReader(blackL);
            BufferedReader br = new BufferedReader(fr);

            // input string
            String line = null;

            // read the file
            while((line = br.readLine()) != null)
            {
                line = line.trim();     // remove whitespace and new line chars
                blacklist.add(line);    // add to the black list
            }

            // close the file
            br.close();
            fr.close();
        }
        catch(Exception e)
        {
            e.printStackTrace();
        }
    }
```

```
/**
 * Write the blacklist to the file
 */
public void writeBlackList()
{
    try
    {
        // open file for friting
        FileWriter fw = new FileWriter("blacklist.dat");

        // write the entries in the blacklist
        for(int i = 0 ; i < blacklist.size() ; i++)
        {
          fw.write(blacklist.get(i) + "\n");
        }

        // close the file
        fw.close();
    }
    catch(Exception e)
    {
        e.printStackTrace();
    }
}

/**
 * write the investigation list
 */
public void writeInvestigateList()
{
    try
    {
        // open the file for appending
        FileWriter fw = new FileWriter("NeedInvestigation.dat", true);

        // write the entries in the investigation file
        for(int i = 0 ; i < investigate.size() ; i++)
        {
          fw.write(investigate.get(i) + "\n");
        }

        // close the file
        fw.close();
    }
    catch(Exception e)
    {
        e.printStackTrace();
    }
}
```

```
/**
 * Chec the blacklist for matches
 * @param s - the url to check
 * @return - true if the url is in the blacklist, false if not
 */
public boolean checkBlackList(String s)
{
    // iterate through the blacklist and check for matches
    for(int i = 0 ; i < blacklist.size() ; i++)
    {
        if(blacklist.get(i).equalsIgnoreCase(s))
            return true;
        else if(blacklist.get(i).toLowerCase().contains(s.toLowerCase()))
            return true;
        else if(s.toLowerCase().contains(blacklist.get(i).toLowerCase()))
            return true;
    }

    // not contained in the blacklist
    return false;
}

/**
 * Check the html code from the website for key words and phrases
 * @param s - the url to check
 * @return - true if a match is found, false if not
 */
public boolean checkHTMLCode(String s)
{
    // append the http:// if it is not present
    String header = "http://";
    String newURL = s.toLowerCase();
    if(!s.contains(header.subSequence(0, header.length())))
        newURL = header + s;

    // whether we need mor investigation
    boolean doInvestigate = false;

    // check html code
    try
    {
        // create the url
         URL url = new URL(newURL);

         // get the connection
         HttpURLConnection connection = (HttpURLConnection)
                url.openConnection();

         // get the input stream from the connetion
         BufferedReader reader = new BufferedReader(new InputStreamReader
                (connection.getInputStream()));

         // input strings
         String html = "";
         String temp = "";
```

```
// read the input stream
while((temp = reader.readLine()) != null)
{
   html += " " + temp;
}

// close the input stream
reader.close();

// clarify the whitespace
html = html.replaceAll("\\s+", " ");

// create the regex pattern for looking for the title
Pattern p = Pattern.compile("<title>(.*?)</title>");

// create the matcher
Matcher m = p.matcher(html);

// look for matches
while(m.find() == true)
{
   // get the matching string
   String check = m.group(1);

   // convert to lowercase
   check = check.toLowerCase(Locale.US);

   // check for the main search strings and return a match if found
   for(int i = 0 ; i < mainSearchStrings.size() ; i++)
   {
      if(check.contains(mainSearchStrings.get(i)))
         return true;
   }

   // initialize the score value
   int score = 0;

   // check the alternative search values
   for(int j = 0 ; j < secondarySearchStrings.size() ; j++)
   {
      if(check.contains(secondarySearchStrings.get(j)))
         score++;   // increment the score value
   }

   // check the threshold
   if(score >= THRESHOLD)
   {
      doInvestigate = true;  // need to investigate
      investigate.add(s);    // add to investigate list
   }
}

// create the regex pattern for finding meta data tags
Pattern p2 = Pattern.compile("<meta (.*?)>");

// create a matcher for the pattern
```

```
        Matcher m2 = p2.matcher(html);

        // look for matches
        while(m2.find() == true)
        {
            // get the matching string
            String check = m2.group(1);

            // convert to lowercase
            check = check.toLowerCase(Locale.US);

            // check for the main search strings and return a match if found
            for(int i = 0 ; i < mainSearchStrings.size() ; i++)
            {
                if(check.contains(mainSearchStrings.get(i)))
                    return true;
            }

            // initialize the score value
            int score = 0;

            // check the alternative search values
            for(int j = 0 ; j < secondarySearchStrings.size() ; j++)
            {
                if(check.contains(secondarySearchStrings.get(j)))
                    score++;   // increment the score value
            }

            // check the threshold
            if(score >= THRESHOLD)
            {
                doInvestigate = true;  // need to investigate
                investigate.add(s);    // add to investigate list
            }
        }
    }
    catch(Exception e)
    {
        e.printStackTrace();
    }

    // if we are investigating, print the results, or print allowed
    if(doInvestigate)
        System.out.println("RESULT: " + s + " - SENT FOR INVESTIGATION");
    else
        System.out.println("RESULT: " + s + " - ALLOWED");

    // return false for not blocking
    return false;
}
```

```
/**
 * Load the website to check from a file
 */
public void loadWebsites()
{
    try
    {
        // open the file for reading
         FileReader fr = new FileReader("websites.dat");
         BufferedReader br = new BufferedReader(fr);

         // input string
         String line = null;

         // read the file
         while((line = br.readLine()) != null)
         {
             websites.add(line);
         }

         // close the file
         br.close();
         fr.close();
    }
    catch(Exception e)
    {
        e.printStackTrace();
    }
}

/**
 * Main Method to run, checks the websites in the list for blacklisting
 */
public void checkSites()
{
    // iterate through the list
    for(int i = 0 ; i < websites.size() ; i++)
    {
        // if it is not in the blacklist
        if(!checkBlackList(websites.get(i)))
        {
            // blocked because of html code
            if(checkHTMLCode(websites.get(i)))
            {
                blacklist.add(websites.get(i));     // add to blacklist

                // print results
                System.out.println("RESULT: " + websites.get(i) +
                        " - BLOCKED and added to blacklist");
            }
        }
        // already in the blacklist
        else
        {
            // print results
            System.out.println("RESULT: " + websites.get(i) +
                    " - BLOCKED in the blacklist");
            }
        }
    }

    /**
     * Main Method - creates a BlackListFilter object and executes the methods
     * @param args the command line arguments
     */
    public static void main(String[] args)
    {
        // pretend to be firefox browser so we don't appear to be a robot
        System.setProperty("http.agent",
                "Mozilla/5.0 (Macintosh; U; Intel Mac OS X 10.6; en-US; " +
                "rv:1.9.2.3) Gecko/20100401 Firefox/3.6.3");

        // create a new black list filter object
        BlackListFilter blf = new BlackListFilter();


        blf.LoadBlackList();    // load the black list
        blf.loadWebsites();     // load the website list
        blf.checkSites();       // check the websites
        blf.writeBlackList();   // write the black list
        blf.writeInvestigateList();     // write the investigation list
    }
}
```

APPENDIX C

FULL TABLE FROM WEB SCRAPING

The table below was generated by passing an input list of the top two hundred ranked guilds according to www.wowprogress.com. These rankings are determined based on Player versus Environment progression.

SEARCH STRING	NUMBER OF GUILDS	NUMBER OF PLAYERS	TIME (MS)
premonition	123	1417	828
adept	38	609	811
blood legion	223	1437	759
deus vox	60	235	268
cuties only	87	327	341
exodus	216	4256	676
vigil	53	743	221
vodka	2	153	76
might	56	634	223
juggernaut	132	1299	520
tsunami	54	550	1633
gentlemens club	133	1191	694
forlorn legacy	97	976	481
eternal reign	134	612	575
casual	93	1288	1233
phoenix	192	4674	2013
midwinter	20	125	581
no chicks allowed	2	107	165
gong show	4	138	141
tasty beverage	179	218	589
raiding rainbows	2	189	153
fusion	193	3382	614
uprising	142	2581	2051
mediocrity	42	1484	149
pie chart	5	52	143
edge	78	717	1275
drow	29	234	147
tupan techno	25	306	377
incite	20	934	936
escaper	5	271	418

bad news bears	73	532	449
surprise mutiny	3	150	271
infallible	40	535	1618
obsidium	20	201	142
huge in japan	31	367	144
gigantor	7	181	840
fh	8	158	1176
vamo feder	3	225	2767
spike flail	141	283	1400
assent	16	189	424
not steamboat	5	163	176
seriously casual	44	749	261
elitist jerks	59	796	305
critical mass	105	1416	609
riot act	14	247	180
legions	46	987	1093
alpha	77	890	1518
acquired taste	33	326	234
incarnate	35	494	858
avast	16	323	670
out of line	12	169	175
nurfed	31	665	937
iron	18	228	928
call of dusk	6	187	172
void	138	1960	352
downtime	26	635	797
simple math	11	222	171
arathian knights	2	448	138
odyssey	109	2585	1176
rush	86	821	1509
just do work	4	108	164
demise	149	3050	1526
ropetown	17	393	337
fused	17	460	868
tan do li ga	2	279	166
just crusade	102	299	346
play	16	231	1079
a team	99	1015	754
rage	205	1711	693
elementium	20	510	961
dark pact	52	671	307

whatever were awesome	11	352	194
blades of wrath	45	390	283
intent	31	942	142
foundation	84	1490	279
tg	32	677	163
the dark ministry	8	482	160
ahh	21	212	1040
enigma	196	2651	1822
fallen	198	2561	1866
obscure reference	8	233	202
gwen stefani	6	320	324
integrity	79	2148	1664
warpath	104	1202	1332
predestined	31	798	575
temerity	40	532	147
something novel	20	110	277
angry	20	469	916
og	47	591	1022
ebayed raiders	54	187	316
titan	120	1175	528
aftermath	208	3963	672
hallowed	54	568	1329
crisis	93	1456	1344
eiysium	3	100	188
afterlife	164	2717	729
fever	31	328	1077
dawn of valor	10	131	172
trismegistus	39	288	765
encore	76	1420	1287
unholy trinity	40	828	308
intrepid	74	1163	1131
the flying hellfish	26	294	187
remedy	51	532	1036
keepers of the faith	37	951	279
burning sensations	4	323	170
redux	41	782	1608
quantum	63	810	325
impulse	140	2173	487
reawaken	23	707	376
stygian	26	341	142
space people	4	81	169

superiority complex	47	775	708
vanquisher	17	268	1010
fallout	150	2189	1969
legends	207	1948	1972
defenestrate	7	238	930
we know girls	61	314	409
scripted encounters	3	6	171
dark nemesis	36	916	350
ascension	216	4204	767
khazuals	14	304	194
hero	164	2075	813
irregulars	18	365	274
tyanny	0	0	174
voracity	19	527	621
reckoning	177	3506	1665
delirium	150	2218	477
halcyon	75	1331	3532
raiding robots	3	295	154
dark bane	13	95	159
flavour country	4	291	155
insomniax	19	597	560
the renamed	3	113	201
damage networks	116	317	432
iniquity	58	797	1036
handlebarz	9	79	640
hat	6	241	148
resurgence	77	3425	302
zephyr	30	267	880
civilian	56	720	825
temporary insanity	58	1287	359
prominence	52	817	802
fatum imperium	4	151	211
scandalous	28	331	811
eternal	196	2115	1533
bad	63	795	1900
clique	27	510	848
red sun	19	289	143
insomnia	200	3924	694
contempt	82	552	1343
tribunal	59	1045	933
roadrunners	30	162	247

makaveli	23	247	1058
anguish	119	1607	2778
reckoning	177	3506	1565
yarg	21	206	775
plaguechill	2	232	180
cadia	16	355	652
stalk and kill	36	548	183
overwhelming	6	109	140
roll initiative	8	294	173
singularity	83	902	995
heretic	64	1029	1120
insurrection	175	3105	1526
reanimated	28	644	951
in the mountains	45	720	320
simplicity	86	1843	286
excessive gaming	1	141	93
did it for whitney	1	128	162
exigence	24	356	435
licious only	2	36	89
tba	21	380	557
death jesters	68	946	347
vindicatum	15	290	344
ainur	8	439	644
the fabled	14	291	161
talisman	41	661	926
ladies of destiny	6	316	163
ascent	102	1205	1081
parnoia	0	0	170
whar lewts plz halp	7	357	136
meteor	91	293	1270
promethean	18	232	463
cryhavoc	20	348	143
ethos	45	761	705
renegade soldiers	15	412	179
casually addicted	31	393	172
risen	148	2653	1480
mors certa	12	130	162
intolerance	40	810	753
paradox	201	3235	1713
synergie	15	223	488
too soon	27	301	236

resurrected	83	1652	1139
void	138	1960	1297
bipolar	26	658	1656
dota ar br banlist on	2	516	145
inertia	73	1333	1387
taint invaders	23	109	183
Totals	11479	171787	137748

APPENDIX D

DATA FOR CORPSE SPELLING

Bodies per letter

Letter	Number of Bodies
A	7
B	9
C	5
D	6
E	5
F	5
G	6
H	6
I	4
J	5
K	5
L	4
M	8
N	8
O	7
P	7
Q	9
R	7
S	7
T	4
U	6
V	4
W	8
X	4
Y	4
Z	6
Average	6

Travel Times from Starting Zone to Nearest Major City (Draenei and Blood Elf are not included as they require purchase of the expansion pack and are not eligible under trial accounts)

Race	Starting Location	Major City	Travel Time (seconds)
Night Elf	Shadowglen	Dalaran	369
Human	Northshire Valley	Stormwind City	206
Gnome	Coldridge Valley	Ironforge	358
Dwarf	Coldridge Valley	Ironforge	358
Orc	Valley of Trials	Orgrimmar	400
Troll	Valley of Trials	Orgrimmar	400
Tauren	Camp Narache	Thunderbluff	482
Undead	Deathknell	Undercity	333
		Average	363.25