DETECTING BOTS IN INTERNET CHAT

by

SRITI KUMAR

(Under the Direction of Kang Li)

ABSTRACT

Internet chat is a real-time communication tool that allows on-line users to communicate via text in virtual spaces, called chat rooms or channels. The abuse of Internet chat by bots also known as chat bots/chatterbots poses a serious threat to the users and quality of service. Chat bots target popular chat networks to distribute spam and malware. We first collect data from a large commercial chat network and then conduct a series of analysis. While analyzing the data, different patterns were detected which represented different bot behaviors. Based on the analysis on the dataset, we proposed a classification system with three main components (1) content-based classifiers (2) machine learning classifier (3) communicator. All three components of the system complement each other in detecting bots. Evaluation of the system has shown some measured success in detecting bots in both log-based dataset and in live chat rooms.

INDEX WORDS: Yahoo! Chat room, Chat Bots, ChatterBots, SPAM, YMSG

DETECTING BOTS IN INTERNET CHAT

by

SRITI KUMAR

B.E., Visveswariah Technological University, India, 2006

A Thesis Submitted to the Graduate Faculty of

The University of Georgia

in Partial Fulfillment of the Requirements for the Degree

MASTER OF SCIENCE

ATHENS, GEORGIA

2010

DETECTING BOTS IN INTERNET CHAT

by

SRITI KUMAR

Major Professor:	Kang Li
Committee:	Lakshmish Ramaxwamy Prashant Doshi

Electronic Version Approved:

Maureen Grasso
Dean of the Graduate School
The University of Georgia
December 2010

DEDICATION

I would like to dedicate my work to my mother to be patient with me, my father for never questioning me, my brother for his constant guidance and above all for their unconditional love.

Lastly, to all my friends and colleagues in the Department of Computer Science and at the University of Georgia. Especially, Nandini and Sourabh for just being there through all up and downs.

ACKNOWLEDGEMENTS

I wish to thank the members of my committee for their support and patience. I would like to thank Dr. Lakshmish Ramaswamy and Dr. Kang Li for directing and guiding me to toward a quantitative methodology. I would like to thank all my lab-mates for making the lab environment very friendly and always being open to discussions.

I would also like to thank the Department of Computer Science for providing me with resources. I have found all my coursework throughout the Masters program to be simulating and thoughtful, providing me with the tools to explore my ideas and implement it.

TABLE OF CONTENTS

Page

APPENDIX

LIST OF TABLES

Page

LIST OF FIGURES

Page

CHAPTER 1

Introduction

1.1 Problem

Chat bots abuse chat services to spread spam and malware. These bots pose a serious threat to Internet users and to the system that provide chat services. Chat Room is a form of synchronous conferencing that can be text-based or audio-visual based [1] over Internet. It is a way of communicating with people in the same chat room in real-time over Instant Messenger. According to a survey done on instant messaging in Carlton university, "Instant Messaging (IM) is a type of communication service over the Internet that enables individuals to exchange text messages and track availability of a list of users in real-time"[2]. This communication has been a break-through as it provides a platform for people located at different locations to come together and share ideas.

Bots are small programs that run automated tasks over the Internet. The program is a script or set of scripts that are created to execute functions in an automated fashion. Bots are being used to crawl websites for the search engine or to provide a virtual opponent in online games. Bots originated from IRC (Internet Relay Chat) but have an escalated use in DDoS (Distributed Denial-of Service) attacks or to host illegal data. Bots were used to moderate chat room services

but soon it became a platform for posting advertisements with hyperlinks. These automated systems are known as chat bots as they have the capability to endure a small conversation with one or more human users.

According to Gianvecchio et al. [42] the major efforts to combat the issue of chat bots in chat rooms is concentrated on two different approaches (1) keyword-based filtering (2) human interactive proofs. The keyword-based message filtering is being used by third party chat clients and by the chat service providers. It has a drawback as the spammers came to know the list of words that are being used to detect spam, bot makers update chat bots to evade published keyword lists. In 2007, Yahoo! Introduced CAPTCHA to the chat rooms to block bots, but it was not very effective. Bots are still entering chat rooms in huge quantity.

Each bot is unique in its way because of the underlying code. Bots functionality can either be simple or complicated. Irrespective of their code they are small programs structured for repetitive tasks. No effective bot detection system has been implemented inspite of their huge presence in chat rooms. Since there is no available chat room bot corpus, we first collected data from the chat room and classified the data as bot or non-bot. This classification by a human user gave us a base line of messages that are considered spam in a chat room environment. The classified set of spam messages was analyzed. Based on this analysis, a system was created to detect bots in live chat room.

Based on the observations we have made about spam bots in chat rooms, it is hypothesized that a bot can be detected if the pattern of messages posted by the bot is identified. Bots are small-automated programs; meaning they are programmed to do tasks over and over again, thus leaving implicit and explicit patterns. Chat bots are programmed to imitate human conversation, but they have limited number of text sentences in their database that leads these bots to post duplicate messages in the chat room. We consider this to be an important characteristic to detect bots in chat rooms.

The second hypothesis is that bots present in the chat rooms can be detected if we try to communicate with them. The bots in the chat room are not only posting messages in the chat room but at times also initiate private conversation with users. They communicate with the user for a short time period. Initially they try to gain trust by introducing themselves and then asking questions about the other user, so they can trick the user to click on the links. After the bots have posted the link they never communicate further. This property can also help us identify bots in chat rooms.

1.2 Motivation

The focus is on Internet Chat and how to save guard the service against malicious users. We are going to look into Yahoo! which is one of the most popular Instant Messenger and chat-room service provider. The existing filters in Yahoo! chat rooms and servers have not been very successful in detecting spam-spreading bots. The focus of this thesis is to develop a system that can effectively and efficiently detect bots in a live chat room environment. A third-party client

has been designed and developed to interact with Yahoo! servers and help in identifying bots in chat rooms.

1.3 Contributions

This thesis has three main contributions to the field of detecting bots in chat rooms, especially in Yahoo! chat room. They are:

1. Designing a Yahoo! client that collects all the public messages posted in the chat rooms.
2. Detection of patterns in text messages and usernames used by bots to enter chat rooms..
3. Detecting bots in real-time using Communicator. This program can initiate and respond to private conversations. These messages are analyzed using classifiers during the conversation.

1.4 Structure of the thesis

This thesis describes chat bots, analyzes the patterns produced by chat bots and then defines a system that tries to detect bots in Yahoo! chat room. The document is organized as follows: Chapter 2 describes IRC (Internet Relay Chat), which was one of the first chat systems to support chat rooms. It also describes the origin of bots and how it was used as a tool for spamming. Chapter 3 describes the popular Yahoo! chat system architecture. Chapter 4 provides a background of related work in detecting bots in chat rooms. Chapter 5 defines the properties of public-messages of Yahoo! chat room (dataset) and the data collection mechanism. In Chapter 6

we analyze the dataset based on the properties of the public messages. Chapter 7 introduces the system architecture designed and developed to detect chat bots. Chapter 8 presents experiments conducted to verify the effectiveness of the system. Chapter 9 concludes this thesis by summarizing the analysis and system effectiveness.

CHAPTER 2

Internet Relay Chat and Bots

IRC is a form of real-time text-based chat system, which allows multiple users to have group communication in a forum called *channels*. IRC was created in fall of 1988 and was created to be better than MUT (MultiUser Talk) and Outbox. Due to its advanced functionalities and high reliability, it soon became popular among users. IRC was a very important technology since it paved a path for chat systems with enhanced functionalities like support for multi-protocol, audio/video chat and file sharing. Many other protocols like AOL, YIM, AIM, ICQ or YMSG have been inspired by IRC and its underlying technology.

IRC [8] is one of the oldest chat systems, which is still in active use. IRC consists of servers and clients. The basic network structure of the servers is of a spanning tree, which is used as a backbone for the network. While sizable Internet users still actively use IRC protocol, a big chuck of Internet users are also using Web-based chat systems. According to the Dewes et al. [9], "There seem to be two main motivations: ease of use and ease of access." These web-based chat systems not only provide a protocol but are also superior in terms of visual appearance and technical realizations. There are many identical characteristics in Web-based systems and IRC-systems [9]. We are going to look in depth about IRC and its components. We are going to see

how different kinds of IRC-based bots came into existence. Lastly, we are going to discuss about how these bots are being used for different kinds of attacks on the Internet.

2.1 Internet Relay Chat (IRC)

During early 1980's World Wide Web was getting a boost and so was this new technology called IRC or formally referred as Internet Relay Chat. According to Toker[19][20], “Internet Relay Chat, provides a means by which one user can type a message in real time to one or more Internet users, and almost instantaneously, the message appears on the monitors of all the other users who are monitoring the transmission.They, in turn, can type messages that all the others may read”. IRC protocol has been designed in such a way that it works on TCP/IP network protocol. IRC is a tele-conferencing system, which runs machines in a distributed fashion. A setup consists of a client-server architecture in which client connects to a server performing the required message delivery/multiplexing and other functions.

IRC was created by Jarkko Oikarinen in Fall of 1988 and insipred by Bitnet Relay Chat. According to website owned by Andreas Gelhausen [13] , “IRC has steadily grown in popularity and currently has more than a million users at any one moment worldwide.” The IRC protocol was redefined in 1992 i.e. 5 years later in RFC 1459 [8], which lead to its popularity. Thus, currently there are many client programs that users can use to connect to an IRC network.

IRC channel not only provided users the ability to send message directlty to another user, but also allowed them to join a set of defined channels which are also known as chat rooms. When a

message is sent to a channel, all other users in the channel can see it. According to Mutton [21], "Each channel has a unique name and is usually inhabited by users with a common interest. Much like a real room, it is possible to infer social interactions between the users in a channel."

2.1.1 Servers

IRC have a client-server architecture. Servers are backbone of IRC, providing a point where all the clients can get connected and a place where all the information can be directed. RFC 2810 describes IRC servers which only allow network configuration that of a spanning tree where each server acts as a central node for the rest of the network.

2.1.2 Clients

A client is a end-user who is not another server. Each client gets connected to other clients through a server and has a unique username.

2.1.3 Channels

According to a memo by IRC, "A channel is a named group of one or more clients which will all receive messages addressed to that channel. The channel is created implicitly when the first client joins it, and the channel ceases to exist when the last client leaves. While the channel exists, any client can reference the channel using the name of the channel."

2.1.4 Communication

The are two types of communication which are supported by IRC:

1) One to One: client A sending message to client B using one server.

2) One to Many: client A sending message to a channel using one server.

2.2 IRC Bots and Chat Bots

A bot is a program that operates automatically as an agent for a user or another bot. An IRC bot is like a roBOT. It is a computer program that logs into the IRC and does things automatically, based upon its programming. They take advantage of certain IRC features to make the IRC experience more- or less- pleasant. Whereas chatbots are social robots in the form of conversation agents, their typical roles are online help or acting as a cyber agent representing an organization. According to Pan et al.[20], "However, there exists a new form of devious chatbots lurking on the Internet. It is effective interactive malware seeking to lure its prey not through vicious assault, but with seductive conversation. It talks to its prey through the same channel that is normally used for human-to-human communication."

Some of the most vicious bots are as follows:

2.2.1 PrettyPark Worm

This worm uses the host network connection to get connected to remote IRC server and allows the attacker to retrieve a variety of information about the system. [21]

2.2.2 mIRC based Bots

mIRC is a popular IRC client for windows.[15] These bots are executables and launch a set of scripts which allow IRC to become the command and control center of the botnet.

2.2.3 Backdoor bots

According to Canavan et al. [21], “Backdoor bot comes as a package that is downloaded as a self-extracting ZIP or RAR file or packed with an installer. When executed, it drops down between eight to twelve files to a subdirectory it creates on windows operating system”. Bot then connects to Internet using host network and notifies the hacker and waits for future commands.

2.2.4 Sdbot

Sdbot took one leap ahead by incorporating its own IRC client executables. Sdbot was a powerful tool for hackers. According to Canavan et al. [21], “These bots took windows malware tricks - setting it for load on startup via the registry’s Run key, using easily confusing, legitimate-looking process names and restricting itself to one binary for ease of execution.”

2.2.5 Agobot

Agobot is an IRC bot, which has features like packet sniffer, key logger and has the ability to harvest information. This family of bots uses a central C&C(Command and Control) IRC server.[21]

2.2.6 SpyBot

According to Canavan et al. [21], “Built on the Sdbot framework, Spybot parses commands similarly, tokenizing lines received and entering a long series of if/else statements checking for valid commands”.

2.2.7 ChatBot

Chat bot is a script that runs independently and connects to Internet Relay Chat as a client, thus appearing as a normal user to all other human users. It is programmed to imitate human users by posting messages in the chat room.

According to Pan et al.[20], “ These chatbots or Instant Messaging (IM) bots use natural languages dialogue system used in gaming technologies to deceive its targets.” The intentions of these bots can vary from gathering identification information from its targets or to lead them to a website containing malware. Another type of chat bot very prominent in Yahoo! chat rooms is Kelvir worm that uses predefined phases to do small talks with the potential victims before sending a link to malicious websites. [20]

There were steps taken to defend against the abuse of chat bots, though none of them are very effective. Yahoo! started using server-side CAPTCHA tests before allowing a user to enter a chat room in order to prevent chatbots to gain access of the chat room. Initially, this solution was a success and chat rooms were clean of spam and bots but it became ineffective when chat bots started bypassing CAPTCHA tests with human assistance. [22] The amount of chat bots in Yahoo! chat rooms are exponential as the chat bots are continuously joining chat rooms and even

becoming the majority number of members in a chat room. According to Gianvecchio et al. [22] , “Third-party chat clients filter out chat bots, mainly based on key words or key phrases that are known to be used by chat bots. The drawback with this approach is that it cannot capture those unknown or evasive chat bots that do not use the known key words or phrases.”

CHAPTER 3

Chat System Architecture

A Chat system is a tool that facilitates communication between two users based on text, audio or video. In this section we are going to introduce a high-level view of Yahoo! chat system and its underlying YMSG (Yahoo! Messenger) protocol. In the previous chapter we have talked about different kinds of chat systems and bots. Mainly, the concentration was on different kinds of bots present in IRC (Internet Relay Chat) and their origin. The goal of this thesis is to detect chat bots in Yahoo! chat room. To achieve this goal we first need to understand the chat system and its underling protocol, since this protocol allows two computers to communicate over the network. YMSG is the underlying network protocol used by Yahoo! Instant Messenger clients to support various services like IM(Instant Messaging), chat rooms, file sharing and audio/video chat etc.

3.1 YMSG

The Yahoo! chat protocol is called YMSG. Yahoo Messenger follows the basic communication architecture of most instant messengers. Following are the initial steps of the YMSG protocol :

Step 1: Contacting Yahoo! Servers to initiate connection.

Step 2: Authentication using username and password.

Step 3: Session creation.

Step 4: Successful authentication leads to server sending buddy list and details of various groups of friends etc. It also sends a couple of cookies.

Step 5: Server sends list of ONLINE friends with their status message and set your status as AVAILABLE.

In the following section we are going to briefly look into YMSG system architecture and what is the basic structure of an YMSG packet. This information will help us in designing an YMSG protocol based client. We are designing a client to help us capture and detect bots in Yahoo chat room.

3.1.1 YMSG System Architecture

Yahoo! is based on a client-server and symmetric architecture [29]. According to Jennings et al. [29] "In a symmetric architecture, each server performs identical functions, such that a client need not distinguish which server it contacts to engage in an activity with."

As shown in Figure 3.1, Yahoo! uses client-server architecture for all its services except file transferring and photo sharing. The symmetric approach describes that once a client is connected to a server then all its future communication or activities are routed through that particular server only. As explained by Jennings et al. [29] "YMSG connects to a random server in the cs##.msg.dcn.yahoo.com domain, where ## is a two-digit decimal number. All subsequent communication is routed through that server."

All Yahoo! communication use TCP over IP communication on port 5050 by default. HTTP route is used if the client is behind a firewall. HTTP request is used to upload and download messages from the server for the client. Yahoo! data resides in the data field of the TCP packet.

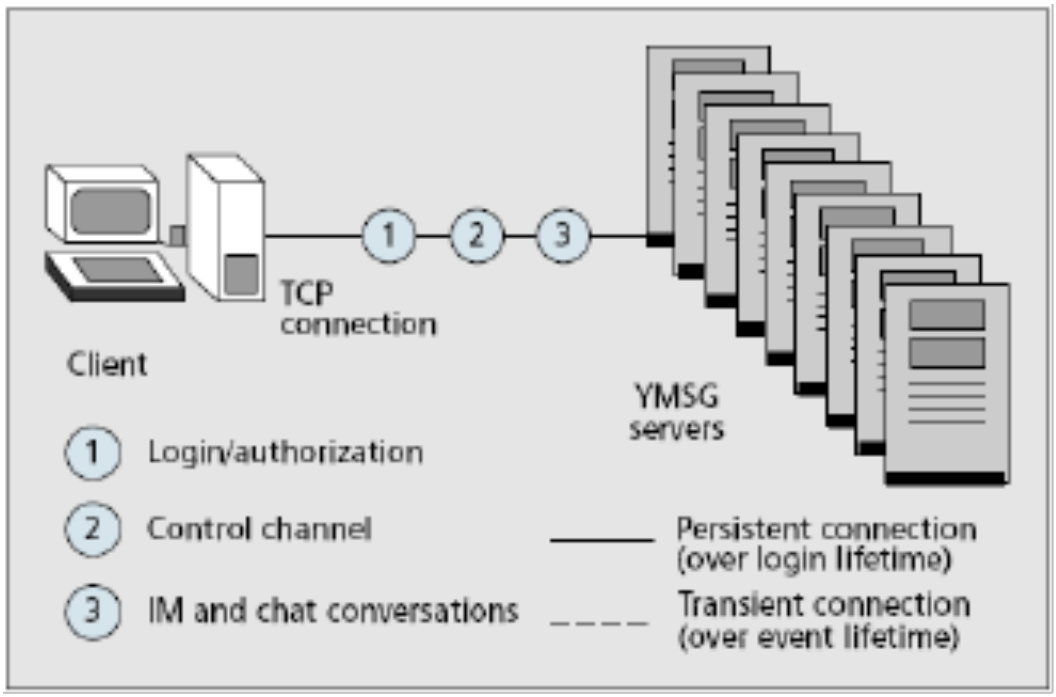

Figure 3.1: YMSG System Architecture

3.1.2 Yahoo! Packet Structure

The Yahoo Messenger protocol is a TCP/IP connection that transfers data in the form of custom packets between a client application and a Yahoo Messenger server. Each packet type has a defined purpose and usually contains data fields.

Yahoo! has its own application level header that is an extension of the TCP/IP. A Yahoo! header is 20 bytes long and is identified by the first 4 bytes being “YMSG.” The Yahoo! header also

includes the YMSG version, message length, service type, status, and session ID. [23] Yahoo! Coders Cookbook[30] shows a graphical representation of the Yahoo! header and data as shown in Figure 3.2

<table>
<tr><th>0 - 4</th><th>4 - 8</th><th>8 - 10</th><th>10 - 16</th></tr>
<tr><td colspan="2">Packet ID (YMSG)</td><td>Version</td><td>Length</td></tr>
<tr><td>Service Type</td><td>Status</td><td colspan="2">Session ID</td></tr>
<tr><td colspan="4">Data</td></tr>
</table>

Figure 3.2: Yahoo! Messenger Generic Header

<table>
<tr><td>Header</td><td>ID</td><td>C080</td><td colspan="3">Data</td><td>C080</td><td>ID</td><td>C080</td></tr>
<tr><td colspan="4">Data</td><td>C080 ID</td><td>C080</td><td colspan="3">Data</td></tr>
<tr><td colspan="9">Data</td></tr>
<tr><td colspan="2">Data</td><td>C080</td><td colspan="6"></td></tr>
</table>

Figure 3.3: Yahoo! Data Field Structure

Yahoo! Coders Cookbook [30] also represents the data portion of a Yahoo! packet as shown in Fig 3.3. Immediately following the session ID, the data format starts in the form of FIELD ID, FIELD SEPERATOR, FIELD DATA, FIELD SEPERATOR, …, and FIELD SEPERATOR. The field ID is represented as an ASCII integer that may consist of several characters. The Yahoo! field separator is the hexadecimal sequence of C080. [24]

3.2 Yahoo! Login

The first packet sent from the client to the Yahoo! server is an authentication request packet. As previously mentioned, any client who wants to use the services of Yahoo! first must log on to the instant messenger server. A user will need to provide a valid email address and password combination to complete registration.

A challenge-response mechanism generally verifies the password as shown in Figure 3.4. For example:

1. User opens the instant messenger and logs in using the username and password. A packet is sent to the Yahoo! server for authentication for which server sends an acknowledgement.
2. Server sends a packet with a key (the challenge).
3. The client combines the key, the password, and some additional information, and then calculates a hash. The most common method for calculating the hash is to use the MD5 or SHA1 algorithm.
4. The server receives the hash from the client. If the hash is correct, user is authenticated and list of ONLINE buddies with their status message is sent to the user.

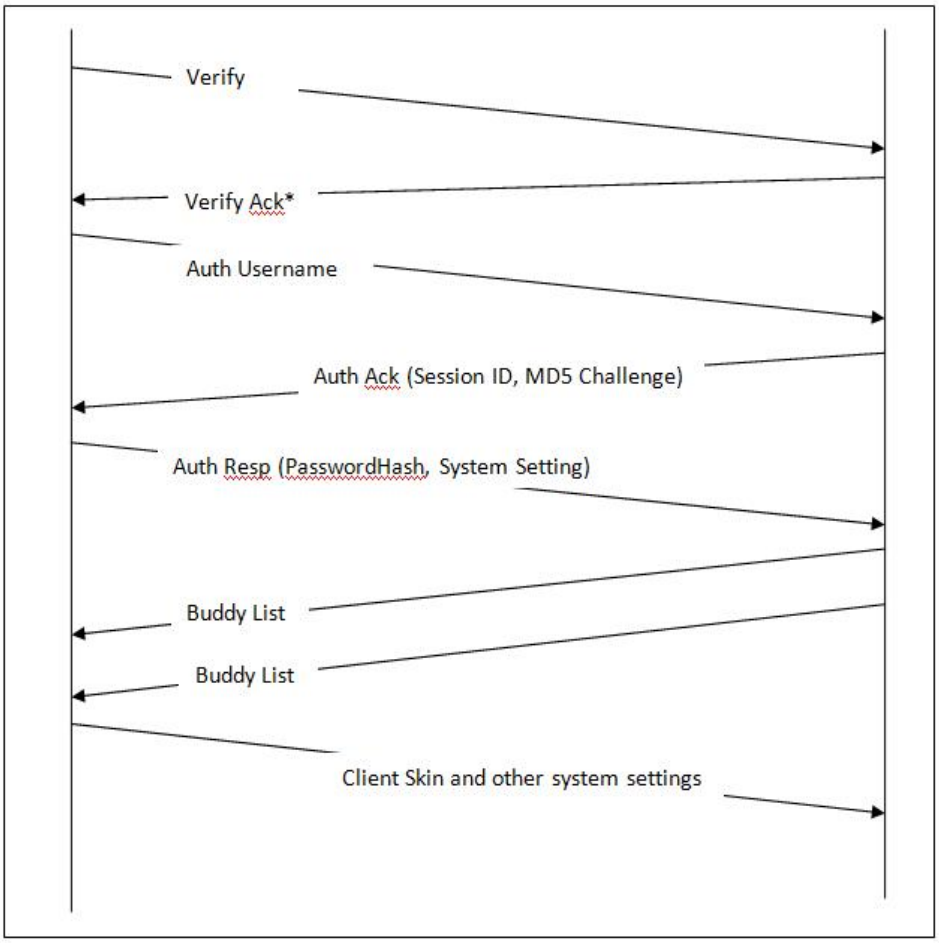

Figure 3.4: Sign-In Sequence

According to Hindocha et al.[31], "The challenge-response method is a fairly secure method for sending a password over an insecure network connection. However, the method is only as secure as the password itself, as the algorithm for calculating the hash remains static."

3.3 Yahoo! Chat Room Login

Beside IM services, Yahoo! also provides services for Chat Rooms. To join any chat room hosted on Yahoo! servers, users first need to login and authorize the Yahoo! client. The following steps are taken after login to join any Yahoo! chat room as shown in Fig 3.5.[24]

1. User requests to join a particular chat room. The user sends his username, room-name and room-number to Yahoo! server.
2. Yahoo! server first acknowledges the request and then sends a CAPTCHA for HIP(Human Interaction Proof).

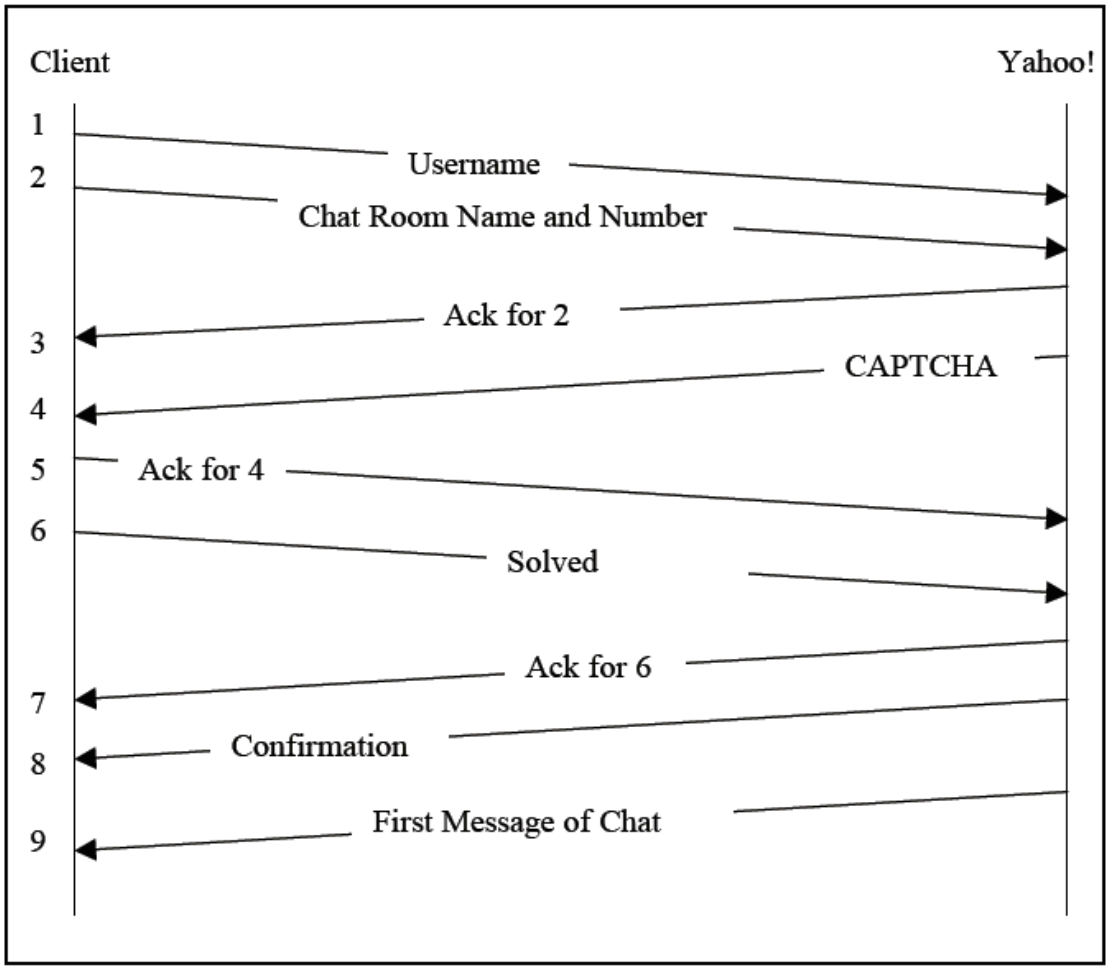

Figure 3.5: Yahoo! Chat Login Sequence

3. User solves the CAPTCHA and sends back to Yahoo! server. The server acknowledges this response.
4. After the server checks the CAPTCHA, it sends a message giving information about the room and also informs the user that Yahoo! is recording their IP address.
5. Server sends the list of users online in the chat room followed by the first message posted on the chat room after the user joined.

3.4 Third Party YMSG API

There are many options to create a Yahoo Messenger client. One can either use any third party clients or create a client by studying the YMSG protocol. There are also some APIs available which provide a platform to create a Yahoo! client. Some of these API's are discussed below:

1. jYMSG

   According to jYMSG at Sourceforge, "jYMSG is a Java API that allows you to interact programmatically with Yahoo's messaging and chat services." jYMSG has made a way to design clients in Java and communicate with the Yahoo! servers. We have created a Yahoo! client in java with the help of jYMSG API.

2. Libyahoo2

   libyahoo2 is a C library interface to the Yahoo! Messenger protocol. It supports almost all current features of the protocol.

3. Hamsam

According to Hamsam at Sourceforge, “Hamsam is a multi-protocol instant messaging API, that helps you to develop a wide range of real-time instant messaging applications.” Hamsam has been designed from the ground up to support a long list of features available in most instant messaging services, and an elegant way of handling protocols that do not support certain features. And more importantly, new protocols can be plugged in with minimal amount of code change.

CHAPTER 4

Related Work

In this section we briefly present some of the research literature related to bots, detecting and blocking spam bots in chat rooms.

Text-based communication over Internet is one of the most popular service used by Internet users and are known as chat systems [42]. There are various chat protocols [32] like IRC, MSN/WLM (Microsoft), YCHT/YMSG (Yahoo!), OSCAR (AOL), and Jabber/XMPP. Though IRC was the first mainstream chat protocol, its popularity declined due to its User Interface (UI) and command-line based operations. Due to its initial popularity and the vast number of users using these services, it became great platform for advertising. This in turn soon escalated to spamming users with unwanted information.

Detecting & controlling spam is a major research area in the field of computer science [33][34]. The Internet has introduced us to some major ground breaking internet-based services but also to spam. There are different kinds of spam but the most primitive is email spam. Regardless, the term is applied to similar abuses in different media like instant messaging spam (SPIM), image spam, link-based spam, social-networking spam etc.[Spam-Wikipedia]. One of the most popular techniques to spam is to use bots

4.1 Bots

There are many ways to spam different services and media. The attacks on the services can be for different reasons like DDoS (Distributed Denial of services), consume bandwidth or to steal information. Since bots are small scripts, they can be designed to coordinate and operate an automated attack on networked computers. [bots Wikipedia]. The systems that are infected with a bot then start communicating with a bot controller and other bots to form what is commonly referred to as a botnet. CAPTCHA's are one of the most popular ways to prevent bots to create dummy accounts that generate spam. All the popular services hosted on the Internet ask users to register themselves by giving personal information and correctly answering the captcha. But spammers have found a way around it by writing scripts to break captcha, using human solvers or finding bugs to bypass captcha [40][41]. Insufficient anti-automation [35] has lead researchers to find new ways to detect and control spam bots.

4.1.1 Handling Bots

According to the authors of "The Zombie Roundup"[39] there are three main approaches to handle botnet.

1. The first approach was to protect systems from getting infected by using various techniques like anti-virus, firewalls and automatic patching.
2. The second approach is to monitor communication between bots and controllers. According to Cook et al. [39], "Botnets today are often controlled using Internet Relay Chat (IRC) and one possible method of detecting IRC-based botnets is to monitor TCP port 6667, which is the standard port used for IRC traffic. One could

also look for non-human behavioral characteristics in traffic, or even build IRC server scanners to identify potential botnets" [38] [37]

3. Last but not the least, the third approach detects botnet by identifying secondary features of a bot infection such as propagation or attack behavior. Rather than directly attempting to find command and control traffic, this approach uses the correlation of data from different sources to locate bots and discover command and control connections.

We are going to look at the work that is encompassed by the third approach. Monitoring data and identifying the secondary features is the key motivating factor behind our solution.

4.2 Chat Room Log and Text-Classification

Dewes et al. [42] conducted a study to measure IRC and web-chat traffic. This study revealed several measurable properties of chat traffic.

1. Chat sessions tend to last for a long time, and a significant number of IRC sessions.
2. According to Gianvecchio et al. [43] about Dewes[42] study, "Over an entire session, typically a user receives about 10 times as much data as he sends. However, very active users in Web-chat and automated scripts used in IRC may send more data than they receive".

There has been a lot of research done in Chat Room topic detection [44]. In Bengel et al.[44] the ChatTrack system uses a text-classification system that creates a concept-based profile. It is a summary of topics discussed by a particular user in a chat room. They have a chat archiving

system, which is used to retrieve details about the chat session. The research in chat room topic detection was a starting point for the analysis of messages posted in chat room. This analysis helped us in finding characteristics and patterns of a spam chatbot.

In Androutsopoulos et al.[45] spam filtering is done using text information in emails. In this system, emails bypasses through a series of content-based filters and then decision rules decide if an email is spam or a legitimate email. Since text filtering in chat rooms is the same as spam filtering in text-based emails many successful techniques have been applied to detect spam in chat room [43]. One of the most successful techniques used in text-based classification is Bayesian-based statistical approach. In our design we are also going to use Bayesian based filter to detect spam in Yahoo! chat room.

4.2.1 ChatBots and Detection

Bot detection has been accomplished mainly by analyzing logged chat room messages with relative success. One of the earliest works in detecting bots in Yahoo chat rooms was done in 2002 by Baird [46]. According to Baird et al. [46] , "Human Interaction Proof's (HIPs), defined broadly as a class of challenge/response protocols which allow a human to authenticate themselves as a member of a given group – e.g. human (vs. machine), herself (vs. anyone else), an adult (vs. a child), etc." All commercial uses of HIPs known to us exploit the gap in the ability between human and machine vision systems in reading images of machine-manipulated text. After this study in 2007, Yahoo introduced CAPTCHA test before letting users enter the chat rooms.

Spammers have used chat bots as a way to lure users into viewing spam and also to click on URLs. Gianvecchio et al. [43] have classified 14 different kinds of chat bots present in Yahoo Chat Room. There work has been focused on how to detect bots in a log-based classification. In Gianvecchio et al. [43] researchers have designed a system which has entropy based classifier that detects the time difference in messages posted by a human and a bot. It also has a machine learning system classifier trained on previously collected log. In another research, the system [47] Rishi: the bots are detected by their IRC nicknames, IRC servers, and uncommon server ports.

The above approaches have been effective in many ways and had positive results, which have supported its conclusions. All the above approaches have been passive i.e. the data is collected and then the filtering is done. By learning from these approaches, we are designing a process that can not only incorporate passive filtering but can also give results in real-time.

CHAPTER 5

Datasets

To design a system that can detect chat bots in a chat room, we need a reference data set. The dataset should contain captured data of the chat room. To the best of our knowledge, we were unable to find a bot corpus that was openly available for download. In order to design and evaluate our bot detection system, we developed a Yahoo! Messenger client with the help of a third party API namely jYMSG API [link: http://jymsg9.sourceforge.net/]. This client was designed in such a way that it included the entire regular Yahoo! client services. In addition this client included features that allowed the recording of all the activities in the Yahoo! Chat room joined by the client messenger.

jYMSG is a Java API that allows to programmatically interact with Yahoo's messaging and chat services. For example, with jYMSG we can have an instant message sent to our system administrator on a J2EE application server for critical error notification. jYMSG is an open source API that allows Java-based applications to use the Yahoo Instant Messenger protocol, version 10. We used this API to create a Yahoo! Messenger client that had the ability not only to record all the activities in the chat room but also to record all the activities directly related to the

client. Data from Yahoo! Chat room was collected for a time-period of six months, in other words from May till November 2009.

In this section we provide the details of the dataset we use in the experiments, including the source of the dataset, the methodology used to collect data and different kinds of datasets.

5.1 Data Collection Methodology

To begin with, we studied the network traffic generated by Yahoo! client messenger. The focus was on the traffic generated to- and from- chat room to produce a dataset useful for our purpose. Since Yahoo! has its own protocol it was simple to identify packets generated by Yahoo! client. Our goal was to collect sufficient data to analyze bots present in most of the different topic oriented chat rooms in Yahoo! US.

For the study purpose, we collected all the pubic messages posted in Yahoo! chat room and also all private messages sent to the data-collecting client. Specifically, we collected the following:

1. All the usernames present in the chat room
2. All the messages posted on the chat room with the time when they arrived on our client.
3. All the users who were leaving and joining the chat room.
4. All the private conversations held with the client.
5. All the conference invitations sent to the client.

We cannot collect the IP address of the users since Yahoo! has client-server architecture and all its data are being routed to the server and then to the client. Any message in a chat room is first sent to the central server where it is stripped from the client's information and then the server repackages it and broadcasts it to all the current users present in the chat room. Direct communication between two Yahoo! client users takes place only when data such as photos or a file, are transferred between the two Yahoo! clients. We also had the access to users' basic profile but most of the profiles were incomplete. Storing this information also violates the privacy laws of Yahoo!.

The data-collecting client did not communicate with any users present in the chat room. The reason for zero communication was to record all the activities seen by a Yahoo! server and to study bot classification approaches which can be implemented at the server end.

5.2 Data Source

A chat room is a dynamic environment and users are also having private conversations with one another, we have logged all the situations where our client was invited for a private conversation. The log contains the timestamp of the message, the username by which the message was posted, and the message itself.

5.3 Dataset

In September 2009, Yahoo! updated the protocol and stopped serving all old protocols having a lower version than 12, creating problems for the data collecting client. The data collection was

hindered for one week. But even this change of protocols had no effect on the number of bots present in the chat rooms.

Yahoo! US Servers have twenty broadly classified sections in the chat rooms. In each section there are subsections and each subsection can have up to thirty open chat rooms. Most of the chat rooms do not encounter significant amount of traffic for a sustained period of time except the Romance section. We collected the data from each of the broad classified section. This way we have the ability to analyze the data of each section and also to look at the popularity of different chat rooms. In total, 3015 hours of data was collected from 25 different chat rooms during the time period.

CHAPTER 6

Analysis

Gianvecchio et al.[43] have classified fourteen different kinds of chat bots in Yahoo! chat rooms. These are broadly classified as periodic, random, responder, and replay bots. Gianvecchio's detection process is mainly based on the inter-message delay and message size. All the activities captured by the data-collecting client in the chat rooms are analyzed in this section.

According to Gianvecchio et al.[43], text obfuscation is used to make bots more difficult to be characterized and also make them appear more human-like. There are different ways to achieve text obfuscation. In general, chat bots use three basic text obfuscation methods to escape filtering or detection. First, bots introduce random characters or spaces into their messages. Second, bots avoid using key words that are known to be filtered by the servers. By this method, a template with several synonyms for multiple words can lead to thousands of possible messages. Third, chat bots use short messages or will break up long messages into multiple messages to evade message filters that work on a message-by-message basis.

To uniquely identify characteristics of bots and non-bots, we need to have messages from bots. Log-based classification is a method in which a human user classifies a username as bot or non-

bot. Due to unavailability of open bot corpus we created bot dataset using log-based classification. The classification was performed on the chat logs collected from May and June 2009. We used this technique based on the criteria presented by Gianvecchio et al.[43]. The Gianvecchio et al.[43], also supports this technique by arguing that the best practice of current artificial intelligences [36] can rarely pass a non-restricted Turing test [49] making log-based classification the best approach to get accurate results.

According to our observation, the main activity of chat bots is to send spam link (URL) or send advertisements to chat users. There are four approaches that chat bots use to distribute spam URLs in chat rooms. The first approach is to post a message with a spam URL directly in the chat room. The second approach is to enter the spam URL in the chat bot's user profile and then convince the users to view the profile and click the URL. The third approach is by initiating private conversation to win the trust of the users and then to trick them into clicking the URL. The last approach is to invite users to a conference and as soon as the user joins in to post a bunch of spam URLs and leave the conference.

The focus of our analysis has been mainly on short-term statistics, as these statistics are most likely to be useful in chat bot classification both in log-based and live chat rooms. We have focused our attention on different metrics, namely text messages available in chat rooms, timestamps of messages, and the size of the data collected from each chat room. We have analyzed the log-based classified content to find patterns that represent chat bot. We were able to

find three major patterns that helped us to design content-based classifier in the chat bot detecting system.

6.1. Total Data Collection

In total 3015 hours of data was collected from different chat rooms. Figure 6.1 shows the total number of lines captured from a subset of rooms. The data represents the text message conversation took place in the chat rooms and the entire private conversations. These samples of data only contain messages from users of the chat rooms. We see that Romance section had approximately 200000 lines of text posted over a three-month period.

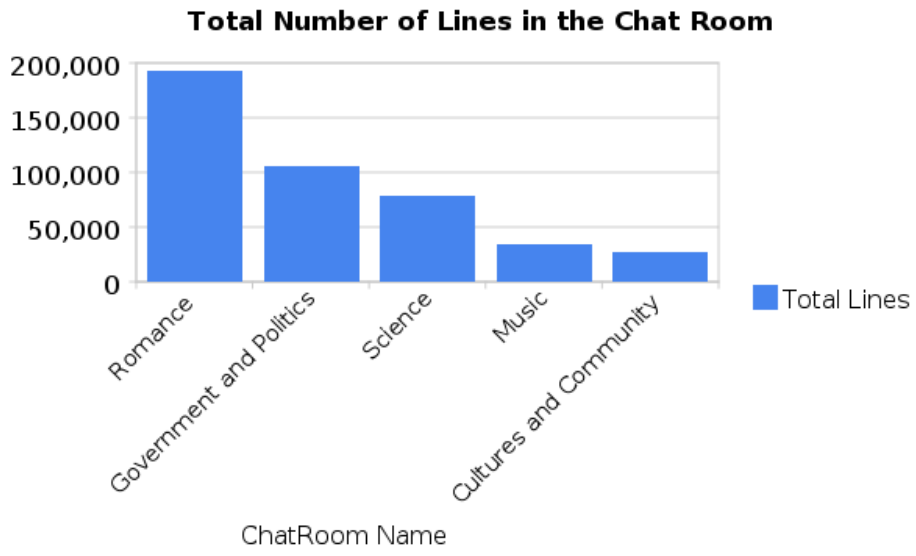

Figure 6.1: Total Number of Lines Posted in Selected ChatRooms

6.2 Analysis Methodology

In this method, we have to find a way to detect bots using the available information in a chat room. In this section the sample dataset will be the obtained using log-based classification.

Content analysis is a methodology to determine if a set of words or phrases exists within the text or set of texts. To conduct a content analysis on any text, the text is coded or broken down, into manageable categories on a variety of levels – word, word-sense, phrases, sentences or theme – and then examined it using one of the basic methods of content analysis: conceptual analysis or relation analysis.

In this section we first focused on the conceptual analysis of text, then on patterns in usernames and lastly on the duplication of messages. During this initial analysis process, we introduced *selective reduction* (i.e. breaking down the content of materials into meaningful and pertinent units of information such that certain characteristics of the message may be analyzed and detected). In the first section of the analysis we examine the text and looked for certain words that we coined as "trigger" words. We were only interested in quantifying these words and in verifying their existence in the dataset. In the second part of the analysis we looked into duplication of messages in the chat rooms because the primary goal of bots is to post spam messages or URLs. Lastly, we looked for a pattern in bot usernames to determine if there is a connection between bots and their usernames.

6.3 Analysis of Content

To get a perspective on the data and to know which chat room generates a certain amount of data, we segregated the data based on twelve different topics of chat rooms. Figure 6.2 gives the average data (in number of bytes) collected over a period of six months. The collection time period was defined for a period of three hours. Since the data-collecting client does not initiate or respond to any communication, Yahoo! Servers look at such a client as an idle client. Yahoo! Servers disconnect any idle client after three hours of inactivity. Thus, we have a collection time period of three hours.

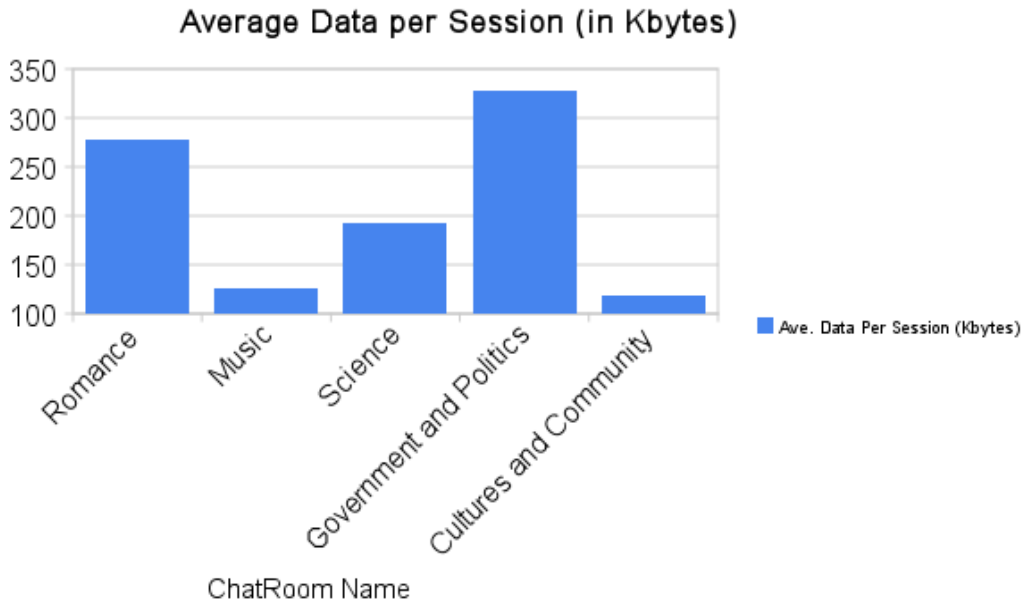

Figure 6.2: Average Data per Session (in Kbytes)

Figure 6.2 was mapped by dividing the total data collected and the total number of sessions in those chat rooms over six months. One session is a period of three hours and contains messages only posted in the chat room. All the private conversations were omitted while calculating the

average data per session. This graph gives us an insight on the volume of data being posted in different categories of chat rooms.

Our hypothesis for analysis is that bot messages are different from non-bot messages. The bot messages show far more duplication of physical content and post messages with words that attract users to click on the links. We have observed that bots do post multiple messages with text obfuscation and URLs.

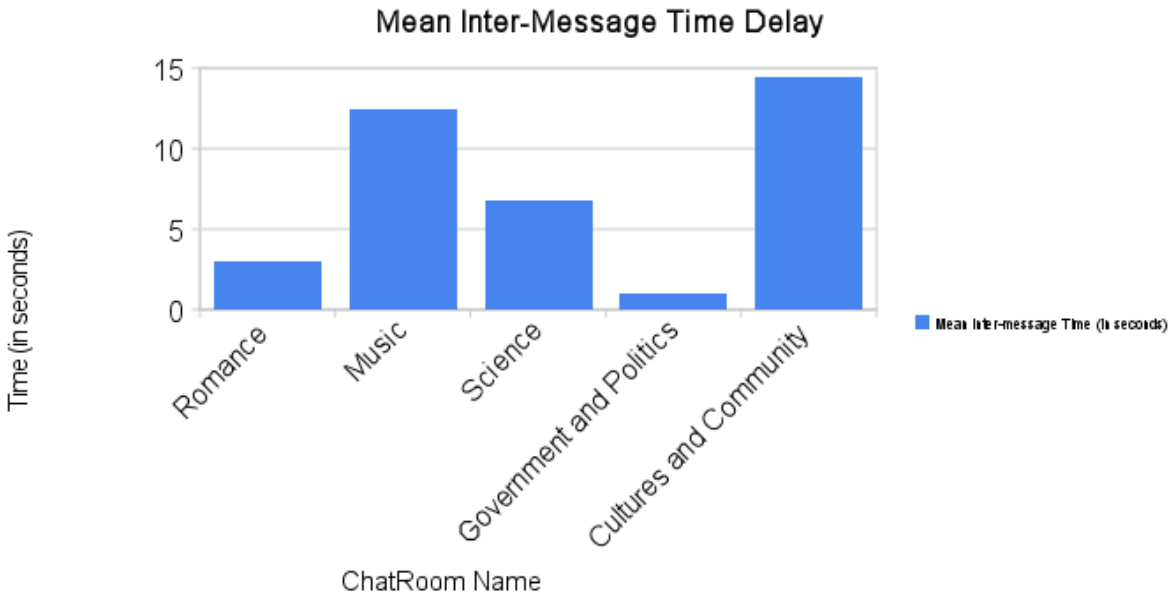

Figure 6.3: Mean Inter-Message Time Delay (in seconds)

Figure 6.3 shows the mean inter-message time delay between two consecutive messages in a chat room. Low inter-message time delay indicates high activity in the chat room, which provide chat bots with a large number of potential targets.

6.3.1 Trigger Words

When we analyzed the whole bot dataset (Section 5.2.1) we saw that many words were repeated and we called them trigger words. We recognized these trigger words and filtered them out. Then we tried to find these trigger words in the remaining unclassified dataset. Figure 6.4 shows the number of appearance of these trigger words in five different chat room sections.

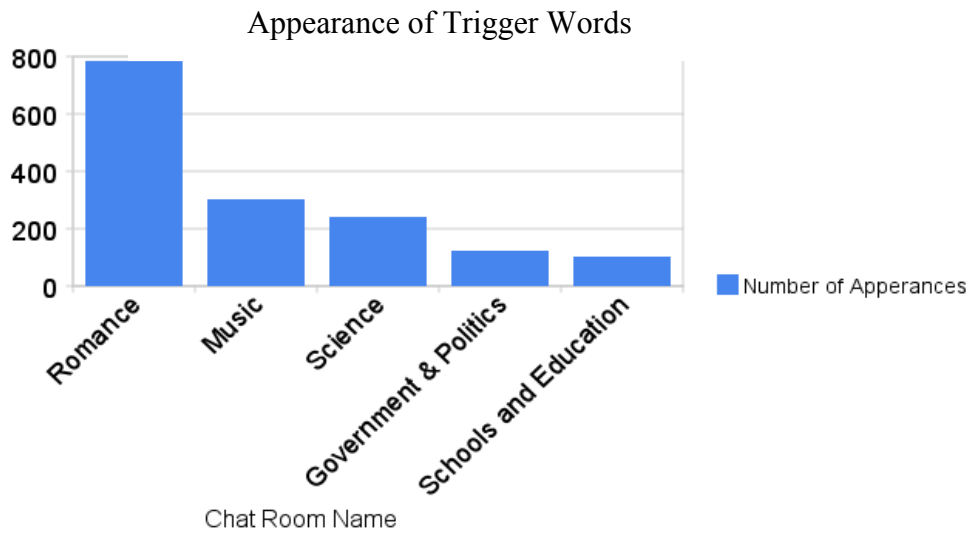

Figure 6.4 Appearances of Trigger Words

As shown in Figure 6.4 these trigger words have a major presence in bot messages. This analysis of content will help us to detect periodic bots and replay bots. See Appendix A for a full list of trigger words.

6.3.2 Duplicate Messages

While we saw trigger words, we also saw many duplicate messages posted by the same user either frequently or at intervals. These characteristics represent replay bots and random bots. While Yahoo! servers apply filters to detect duplicate messages, these bots try to overcome those filters by adding spaces, numbers, or random text in white color. These characteristics also represent the presence of bots.

The above analysis results are based on the hypothesis that bot messages duplicate physical content, URL and also that they post general messages with words to attract users.

6.4 Analysis of Usernames

Our hypothesis for this experiment is based on the structure of username that may provide a way to detect bots. We have observed that different bots post identical messages either in the same room or in different rooms. When we initiated a private conversation, they posted the same response irrespective of the conversation topic. This shows that there are many bots that have same controller. In other words, a controller controls a few hundred bots running in a few hundred-chat rooms at the same time. Since these are small programs that have one script running and controlling all the bots as well as creating new bots they tend to have similar patterns. Patterns are the basic structure which each bot will have as they are programmed by one controller. Though randomness may be involved, many of the aspects are hard coded. And one of the aspects that were hard coded and detected was the username pattern.

As we were analyzing the contents of the messages posted on the chat room we also realized a prominent pattern in the way each bot had structured its username. There are three major patterns that were recognized during the analysis. They are the following:

Pattern 1: Name+Numbers+Random alphabets

e.g. samantha786ngbye, megan756mbymo, alexis117hducp

Pattern 2: Name+ _ + LastName+ _ +Random numbers

e.g. kami_jarrad_409, kandace_wilner_605, stacy_altman_502

Pattern 3: Name+_+LastName+Random numbers

e.g. samy_duke700, randy_puk239, lily_bun201

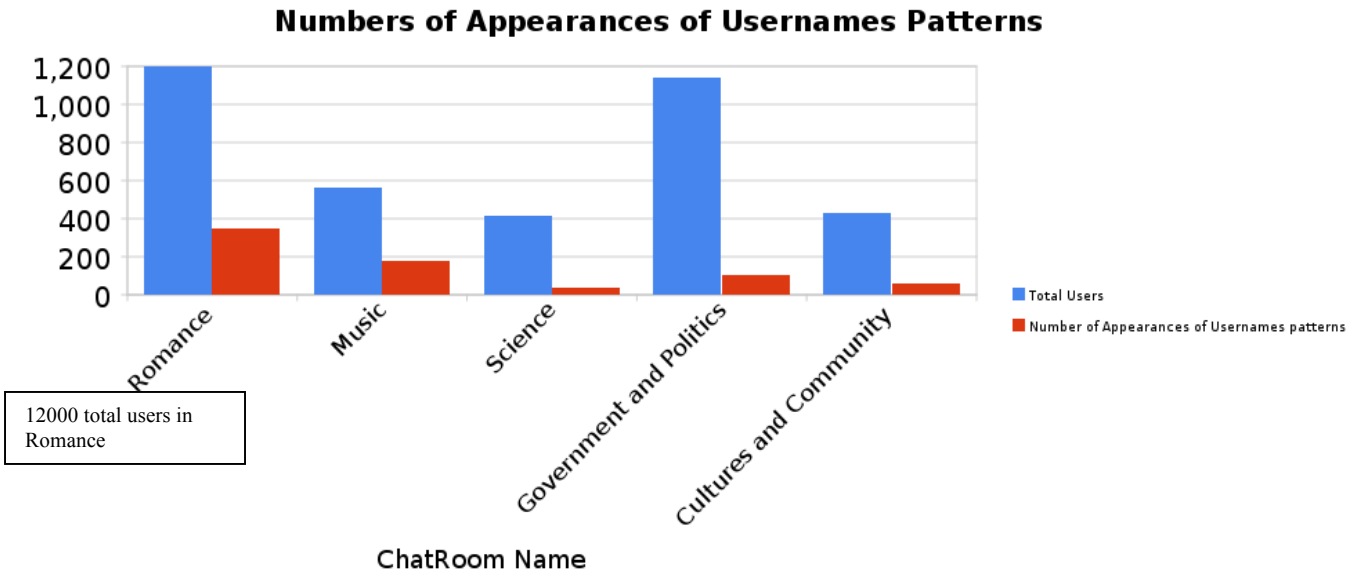

Figure 6.5 Numbers of Appearances of Username Patterns

Figure 6.5 shows the number of appearances of this structure of usernames in five different topics of chat rooms. Though the number of appearances is not as prominent as the trigger words, it shows that bots controlled by one controller are spamming multiple subsections of chat rooms in high quantity. This is also consistent with our hypothesis that different bots can have the same structure in their username. But this hypothesis can fail if the spammer (i.e. a human) removes the automation in username creation for multiple bots. In spite of this potential drawback, these characteristics are still very helpful in detecting bots when we do not have enough text information.

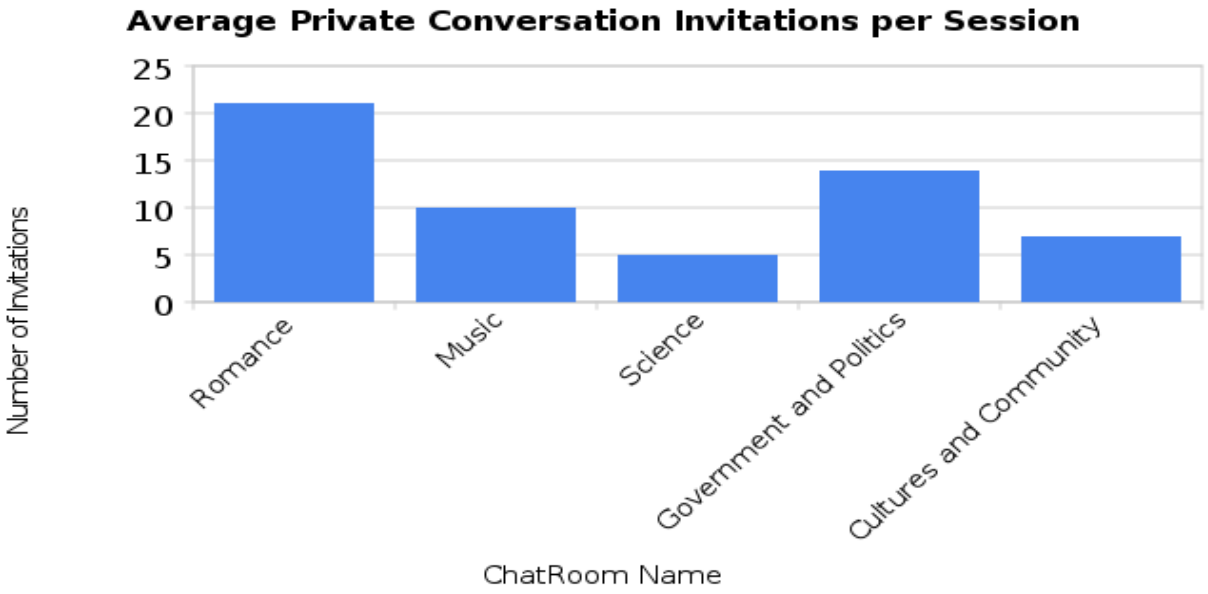

Figure 6.6 Average Private Conversation Invitations per Session

6.5 Private Conversations

During the data-collection process, the client used to get multiple invitations for private conversations at the time of login. The initial messages were very similar. Responding to these messages would trigger similar responses, which point to the user being a bot. Figure 6.6 represents the average private communication requests per session for the various chat rooms.

CHAPTER 7

Proposed Solution

This section describes the design of the chat bot detection system. The three main components of the detection system are content-based classifiers, machine learning classifier and communicator. The three components of the system are loosely coupled in a sequential manner. The content-based classifiers examine the content of the text messages to detect chat bots. The machine learning classifier uses the bot dataset to learn text patterns of bots and non-bots, and then use these patterns to classify bots. The communicator either initiates or responds to private conversation between the chat bot detection system and the suspected bot. The three components are explained in detail below.

7.1 Content-based Classifier

Based on the data analysis, three sub-classifiers were created under content-based classifiers. Each sub-classifier is customized to recognize one particular characteristic of the spam bot.

7.1.1 Word Classifier

In section 6 it had been explained that a bot's main aim is to trick users to click on a spam URL. Bot posts general messages to entice all users in that particular chat room. They use words like “pm me”, “webcam”, “sex”, “goto link”, “checkout my homepage” etc. As described in section

6, bot's public messages posted on chat rooms provided us a unique way to characterize bots. Bot messages, certain trigger words and username patterns in the text also helped us detect bots in a chat room.

7.1.2 Duplicate Classifier

In the previous chapter we analyzed different bot messages that were posted multiple times in a chat room. These messages can be from either random bots or periodic bots. Random bots posts messages in the chat room at random interval of times whereas periodic bots post messages in a fixed time interval. Their main aim is to post messages in the chat room.

We need to determine a threshold level on the number of duplicated messages that represents bot. In the sample data we collected all the duplicate messages in a three-hour time frame. Those messages were then grouped based on usernames and then ordered in ascending order. The median of those values was taken as the threshold value for detection of bots.

Thus to overcome false detection we classify only those usernames as bots that post four or more duplicate messages in a fixed time frame. This way we are assuming that non-bots or humans will not post duplicate message more than four times.

7.1.3 Username Classifier

Based on the previous analysis, we saw a new pattern emerging in the collected dataset. We saw patterns in the username of the clients suspected to be a bot. We think the reason for the emerged

pattern is due to a single source (or software-program) controlling multiple bots. During the analysis of the usernames, three different patterns were recognized for having highest presence in the dataset. These patterns are used as a classification for detecting bots.

7.2 Machine Learning Classifier

According to Elnahrawy et al.[14] comparative study, the problem of automatic monitoring of chat rooms can be solved efficiently by using the appropriate text categorization methods. The choice of the appropriate categorization method depends on two factors: the accuracy of the classification and the efficiency of the method. A comparison between the Naïve Bayes, the K-nearest neighbor, and the Support Vector Machine classifiers was performed. The results suggested that a simple Naïve Bayes algorithm might be an appropriate choice for this problem since its training and classification time are considerably short and it also performs satisfactorily with respect to the accuracy of classification.

Based on the Elnahrawy et al.'s[14] comparative study we assume that Naïve Bayes algorithm is a good choice for text classification problem. Identifying bots in text-based chat rooms is also a kind of text classification problem. If we need to identify bots based on the text messages posted on the chat room we need to collect the text messages. Since chat messages are text-based, the identification of chat bots perfectly fit into the domain of machine learning text classification.

Text classification is the task of assigning a Boolean value to each pair $(d_j, c_i) \in D \times C$, where D is a domain of messages and $C= \{c_1,\ldots\ldots, c_i\}$ is a set of predefined categories[11]. Value 1 for $f(d_j, c_i)$ indicates that the message d_j is in class c_i and value 0 indicates the opposite decision.

Abstractly, the probability model for a classifier is a conditional model $P(C| F_1, \ldots.,F_n)$ over a dependent class variable *C* with a small number of outcomes or *classes*, conditional on several feature variables F_1 through F_n. The problem is that if the number of features *n* is large or when a feature can take on a large number of values, then basing such a model on probability tables is infeasible. The formula can be reformulated to make it more tractable.

Using Bayes' theorem,

$$P(C| F1, \ldots.,Fn) = \frac{P(C)\, p(F_1,\ldots\ldots.,F_n|C)}{P(F_1,\ldots\ldots.,F_n)}$$

According to Gianvecchio[14], the probability that a chat message (M) is from bot can be calculated by computing the probability of a message(M). Thus, applying Bayes' theorem as

$$P(bot|M) = \frac{P(M|bot)P(bot)}{P(M)}$$

Given the abundance of implementations of Bayesian classification, we directly adopt one implementation, namely WEKA, as our machine learning classification component. The Waikato Environment for Knowledge Analysis (Weka) is a set of Java libraries that implement machine learning and data-mining algorithms. According to Witten et.al [4], “Applications written using the Weka class libraries can be run on any computer with a Web browsing capability; this allows users to apply machine learning techniques to their own data regardless of computer platform”. Elnahrawy et al.[14] concludes in her comparative study that based on the accuracy of the classification and the efficiency of the method simple Naive Bayes algorithm might be an appropriate choice for this problem since its training and classification times are considerably short. There are six Bayes based classifiers but based on the above comparative study we implement Naïve Bayes classifier.

7.3 Communicator

Bots being a software-program does not have the flow of the human conversation. Definitely one can program bots to reply in an intelligent way to imitate humans. One of the main motives of bots in the chat room is to lure users to click on URLs or push unwanted information like online advertisement. To achieve this purpose, after establishing a successful open communication with another user, bots quickly end the communication with a URL or an advertisement. Bots also have a property to keep replying to messages for a certain time period. Once they post a URL or advertisement in the private conversation, they will never reply to the messages.

Keeping the above observations we open a private conversation with a user who is being suspected to be a chat bot. We have upgraded the data-collecting client in such a way that it is able to initiate and reply to general conversation messages which we call the communicator. The communicator has its own list of sentences that it uses to form a response message. As soon as the private conversation is initiated the user is informed that they are having a conversation with a bot and they can stop the conversation by just closing the window. Usually humans disconnect or even if they keep the conversation after sometime they get bored and then they exit the conversation. If the user continues to reply and in the end they post a link and then never reply back, we can conclude that those users are bots. When we send special characters as a message, bots do not reply because they are not able to identify the characters. They are programmed to identify only characters from a-z, A-z and 0-9.

The communicator consists of 100 sentences. As soon as the message arrives the communicator looks for punctuation marks. Depending on the punctuation marks it replies with a general answer. It also looks for key words like “what”, “how”, “where” and “name”. It also recognizes short form of sentences like “asl” which is asking age, sex and language the user speak. The communicator’s initial conversation is mainly on geographical area, which is to establish trust based upon environment. In the communicator, most of the sentences are questions since we want the bot to reply. To keep the private conversation open the communicator replies with a answer followed with a question.

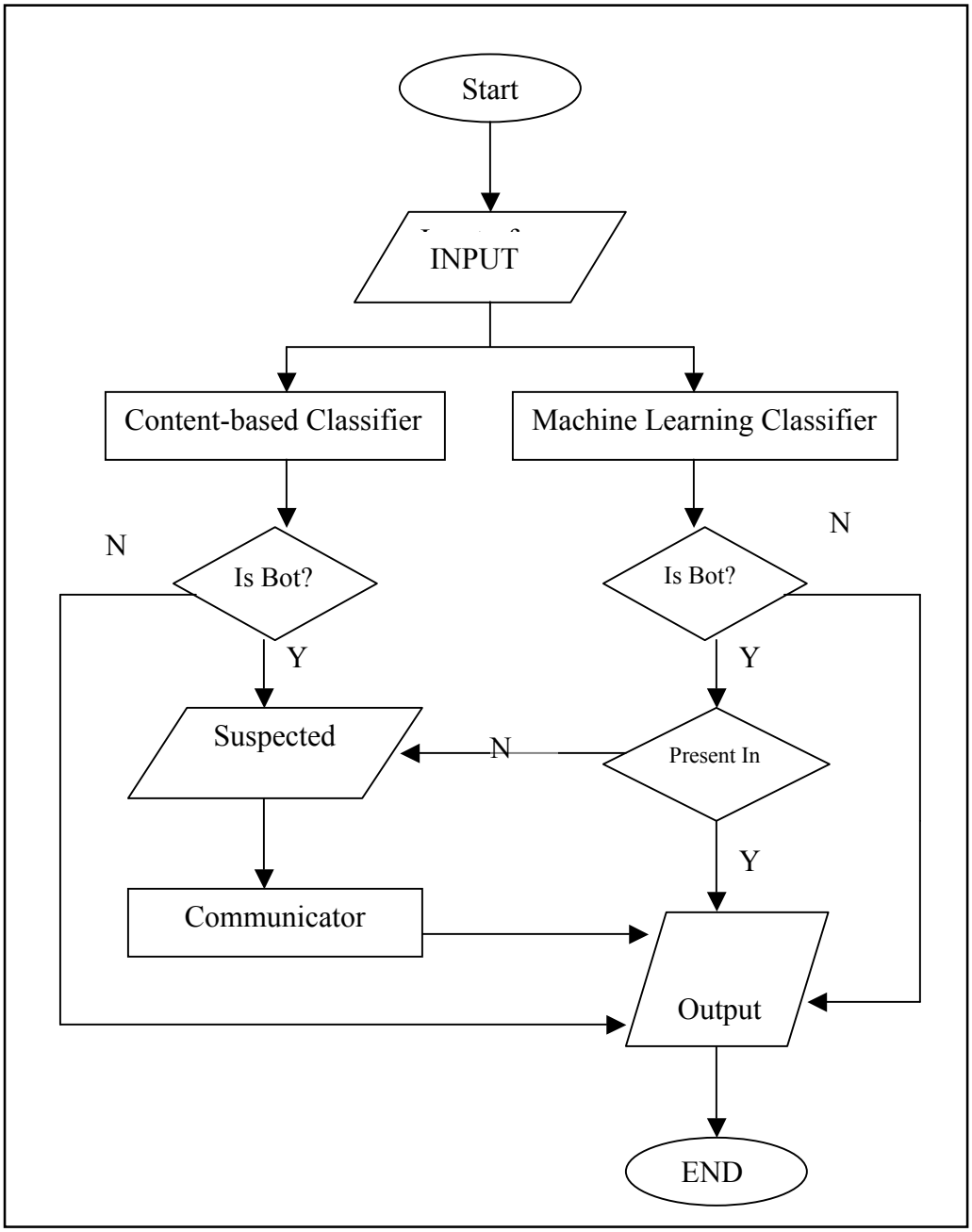

Figure 7.1 Flowchart of Proposed System

7.4 Proposed System

As shown in Figure 7.1, the system consists of three main components (1) context-based classifiers, (2) machine learning classifier and (3) communicator. When the client enters the chat room the system collects all the usernames and filters through the Username classifier. Now, the system is a member of the chat-room so it can receive all the messages being posted on the chat-room. All the messages posted on the chat-rooms pass through the keyword filter and duplicate filter. If any of the usernames are flagged as bots they are added to the suspect list. In parallel, all the messages are also given to machine learning classifier that predicts which username is a bot. If the predicted username is already in the list then it is declared a bot. If the predicted username is not in the list then it is added to the list. The communicator takes usernames from the list and opens a private conversation with each username.

Many times when the client joins a chat-room other users open a private conversation with the client. For example, when the client joins any room in romance it gets more than ten private conversation requests by users who we suspect are bots. These users do not post anything on chat-room but try to have private conversation with everyone who is joining the chat room. Any communication not initiated by the client is considered as a suspect of being a bot. So all the usernames who open a private conversation with the client are added to the suspect list. The communicator continues communicating with them to satisfy all the conditions that represent the user of being a bot.

CHAPTER 8

Experiments and Results

This section describes the experimental evaluation of process for the proposed classification system. The first classification test was based on chat logs collected from the Yahoo! Chat-rooms for the duration of six months (May till November 2009). The second classification test was to classify bots in real-time by feeding live data from the Yahoo! Chat rooms to the system. The first classification test results were analyzed against the chat logs from September and October 2009. The accuracy of the classification was measured in terms of false positive and false negatives. False positives are the users who are wrongly identified as bots and false negatives are the users who are wrongfully identified as humans.

8.1 Experimental Setup

In our experiments we used random bot messages from chat logs collected in May to September 2009 for training the machine learning classifier. In the second stage, the system was tested against the live messages posted in chat rooms. Thus, we were able to test the classifiers and communicator in both static and dynamic environment.

8.2 Experiment Results

The results obtained by conducting experiments on the above-described datasets are explained in the subsequent sub-sections.

8.2.1 Classification on Log Data

For the first part of the testing, we choose random three hours of data from all twenty-five rooms and then allowed the system to classify bots. Then the author went through the data and classified it as bots and non-bots. This step was taken to compare the effectiveness of the system to the classified data.

Figure 8.1 shows the comparison between human classification and system classification. As seen in the figure most of the time human and system classification were similar but at times they differ. Humans have more perception about the way a message is posted and they do not trust messages from the anonymous users. The system at times is able to detect more bots than human due to the username classifier and duplicate classifier. If a user does not post messages or post just one message we do not have enough information to classify but the system does know what kind of patterns to look for which helps in classifying bots.

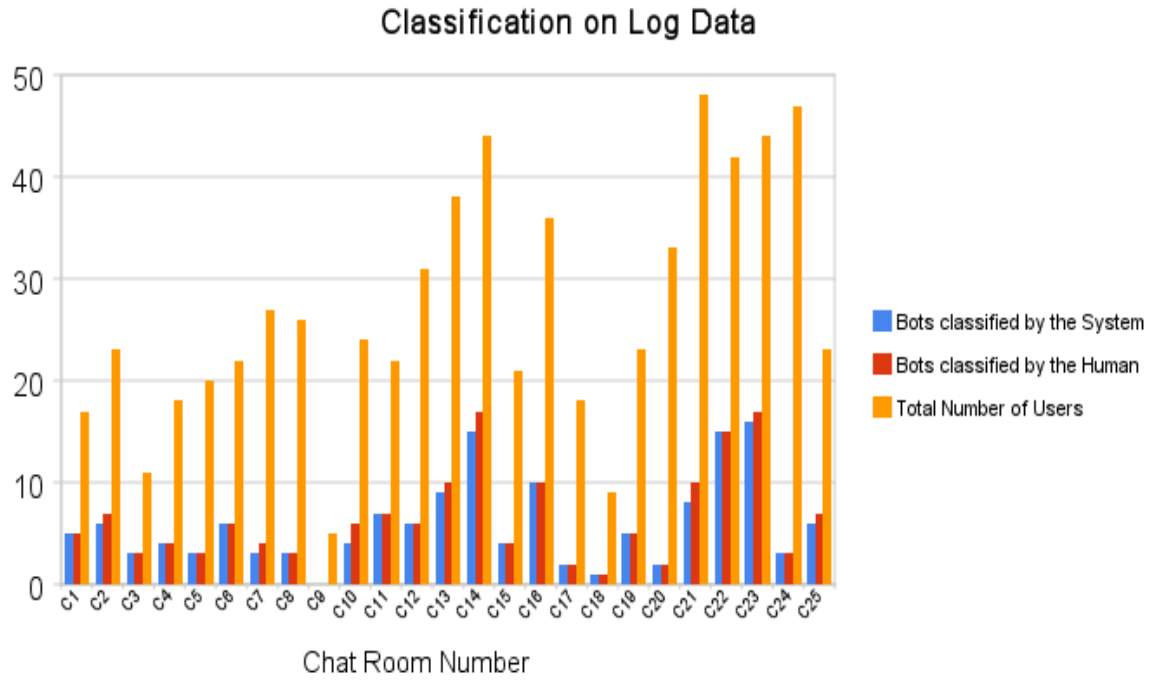

Figure 8.1 Classifications on Log Data

8.2.2 Classifications in LIVE Chat Room

For second part of testing we did not test our system on datasets from April 2009 to August 2009 because the machine learning classifier had been trained on that dataset. Our whole dataset is collected from eight different rooms. So the system was tested in all the eight rooms for a fixed-length time period.

Months	Total users	System Classified as Bots	System Classified as Humans	False Positives	False Negatives	Correct Classification of bots (%)
Oct 2009	487	50	437	5	7	86.5384615384616
Nov 2009	376	25	351	3	8	73.3333333333333

Table 1: Bots Classified in Live Chat Room

Table 1 gives a overall view of the system classification done by the proposed system for the eight Yahoo chat rooms. The table shows the total number of users present in the chat rooms for both the months and also the percentage of correct classification.

The system had significant false positives due to URLs. Sometimes users post links in the chat room and they also use keywords like “pm me” and “webcam”. These users genuinely want people to contact them and look at their webcam. Since most bots are using this strategy to lure normal users to click on the URL, the system at times classifies normal users as bots. The reason we are getting false negatives is due to silent bots. Many times bots post messages on chat room and then go silent for a while before they start posting again. Due to this time interval we have to look for longer time duration but our clients can only be present in the chat room for a maximum of three hours. These conditions prevent us to look into a longer history to detect bots.

CHAPTER 9

Conclusion

The analysis of the datasets shows us that the problem of bots present in Yahoo! Chat room is a major concern. Steps have been taken to prevent bots from entering chat rooms without much success. It is necessary to be able to detect bots in a dynamic environment. CAPTCHA was introduced as a HIP filter, but now spammers are cracking CAPTCHA by using humans thus reducing the effectiveness of CAPTCHA and failing the whole purpose. The system that we have designed, created and tested is to classify bots in a dynamic environment. The system tries to classify bots in real-time with the information which is currently available and also on the information which it gets from communicating bots. This system is unique in a way that it combines two ways of classification. One by looking into already classified bots nature and second one by communicating with bots thus exploiting the way humans and bots behave. The results of this system has shown that it is effective but due to its dependence on logs it needs to be updated to be continually effective in classifying bots in Yahoo! chat room.

REFERENCES

[1] http://www.multicians.org/thvv/mail-history.html

[2] “Secure Public Instant Messaging: A Survey (2004)” by Mohammad Mannan , P. C. Van Oorschot

[3] http://www.multicians.org/shell.html

[4] http://www.qlinklives.org/

[5] http://www.shvoong.com/internet-and-technologies/133409-instant-messenger-chat-room/

[6] Wikipedia – Instant Messaging

[7] R. Puri, "Bots & Botnet: An Overview," SANS Institute 2003.

[8] J. Oikarinen and D. Reed,” Internet Relay Chat Protocol RFC 1495”,1993

[9] Dewes, C., Wichmann, A., and Feldmann, A. 2003. An analysis of Internet chat systems. In *Proceedings of the 3rd ACM SIGCOMM Conference on internet Measurement* (Miami Beach, FL, USA, October 27 - 29, 2003). IMC '03. ACM, New York, NY, 51-64. DOI= http://doi.acm.org/10.1145/948205.948214

[10] Simpson, C. (2000). Internet Relay Chat. In: Educational Media and Technology Yearbook, 2000.

[11] http://www.messengeroo.com/yahoo-messenger/chat-client/best-yahoo-messenger-third-party-chat-client-and-anti-boot-2009/

[12] Jarkko Oikarinen. RFC 1459 - Internet Relay Chat Protocol, http://www.faqs.org/rfcs/rfc1459.html

[13] Andreas Gelhausen. Summary of IRC Networks, http://irc.netsplit.de/networks/ (Accessed 14 November 2009).

[14] Symantec, Jan 2008, A Evolution of Malicious IRC Bots, White Paper.

[14] Eiman Elnahrawy, “Log-Based Chat Room Monitoring Using Text Categorization: A Comparative Study”, In Proceedings of the IASTED International Conference on Information and Knowledge Sharing (IKS 2002), St.Thomas, US Virgin Islands, November 2002.

[15] Bejtlich, R. 2005 Extrusion Detection: Security Monitoring for Internal Intrusions. Addison-Wesley Professional.

[16] Provos, N. and Holz, T. 2007 Virtual Honeypots: from Botnet Tracking to Intrusion Detection. First. Addison-Wesley Professional.

[17] Toyer, K. (1997). "Learn Internet relay chat." Plano, TX: Wordware Publishing, Inc.

[18] Toyer, K. (1997). "Learn advanced Internet relay chat." Plano, TX: Wordware Publishing, Inc.

[19] P .Mutton, “Inferring and visualizing social networks on internet relay chat,” Proc Information Visualization, IEEE Computer Society, 2004. London, UK.

[20] Pan Juin Yang Jonathan, Chun Che Fung and Kok Wai Wong, “Devious Chatbots - Interactive Malware with a Plot”. Page 110-118, SpringerLink 2009

[21] J. Canavan, "The evolution of malicious IRC bots," in *Proceedings of Virus Bulletin Conference 2005*, pp. 104-114, October 2005.

[22] S.Gianvecchio,M.Xie,Z.Wu,andH.Wang.Measurement and classification of humans and bots in internet chat. In Proceedings of the 17th USENIX Security Symposium (Se- curity’08), San Jose, CA, July 2008.

[23] Mary Czerwinski , Edward Cutrell , Eric Horvitz, “Instant Messaging and Interruption: Influence of Task Type on Performance” (2000)

[24] Mary Czerwinski, Edward Cutrell and Eric Horvitz, Microsoft research , Instant Messaging: Effects of Relevance and Timing

[25] Mohammad Mannan , P. C. Van Oorschot, “Secure Public Instant Messaging: A Survey”

[26] A study of instant messaging

[27] TCP -Wikipedia

[28] Malicious Threats and Vulnerabilities in Instant Messaging

[29] Jennings III, R. B., Nahum, E. M., Olshefski, D. P., Saha, D., Shae, Z.-Y., and Waters, C. A study of internet instant messaging and chat protocols. *IEEE Network Vol. 20*, No. 4 (2006), 16–21.

[30] Yahoo! CookBook

[31] N. Hindocha and E. Chien. Malicious threats and vulnerabilities in instant messaging. In H. Martin, editor, *Proceedings 13th Virus Bulletin Conference*, pages 114–124, Toronto, CA, Sept. 2003. Virus Bulletin, Virus Bulletin

[32] Jennings III, R. B., Nahum, E. M., Olshefski, D. P., Saha, D., Shae, Z.-Y., and Waters, C. A study of internet instant messaging and chat protocols. IEEE Network Vol. 20, No. 4 (2006), 16–21

[33] Collaboration, Electronic messaging, Anti-Abuse and Spam Conference, July 12-13,2010

[33] MIT Spam Conference , March 25-26,2010

[34] http://projects.webappsec.org/Insufficient+Anti-automation

[35] John Kristoff. Botnets. 32nd Meeting of the North American Network Operators Group, October 2004.

[36] Stephane Racine. Analysis of Internet Relay Chat Usage by DDoS Zombies. Master's thesis, Swiss Federal Institute of Technology Zurich, April 2004.

[37] The Honeynet Project. Know your enemy: Tracking botnets. http://www.honeynet.org/papers/bots/, March 2005.

[38] Evan Cooke, Farnam Jahanian, Danny McPherson, "The Zombie Roundup: Understanding, Detecting, and Disrupting Botnets", SRUTI '05 Paper

[39] Jeff Yan, Ahmad Salah El Ahmad, "Breaking Visual CAPTCHAs with Naive Pattern Recognition Algorithms," Computer Security Applications Conference, Annual, pp. 279-291, Twenty-Third Annual Computer Security Applications Conference (ACSAC 2007), 2007

[40] http://www.simplepixel.com/2008/04/16/spambots-crack-hotmail-captcha-creating-thousands-of-email-accounts/

[41] Dewes, C., Wichmann, A., and Feldmann, A. An analysis of Internet chat systems. In Proceedings of the 2003 ACM/SIGCOMM Internet Measurement Conference (IMC'03) (Miami, FL., USA, October 2003).

[42] S.Gianvecchio, M.Xie, Z.Wu, and H.Wang. Measurement and classification of huamans and bots in internet chat. In Proceedings of the 17th USENIX Security Symposium (Se- curity'08), San Jose, CA, July 2008.

[43] Bengel J., Gauch S., Mittur E., Vijayaraghavan R.(2004). ChatTrack: Chat Room Topic Detection Using Classification, submitted to The Second Symposium on Intelligence and Security

Informatics, Tucson, Arizona, June 2004

[44] Androutsopoulos, I., Koutsias, J., Paliouras, G., Karkaletsis, V., Sakkis, G., Spyropoulos, C., & Stamatopoulos, P., (2000) Learning to filter spam e-mail: A comparison of a naive bayesian and a memory-based approach. in Workshop on Machine Learning and Textual Information Access, at 4th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD).

[45] H. S. Baird and K. Popat, "Human interactive proofs and document image analysis," in Proc., IAPR 2002 Workshop on Document Analysis Systems, (Princeton, NJ), August 2002.

[46] J. Goebel and T. Holz. Rishi: Identify bot contaminated hosts by irc nickname evaluation. In USENIX Workshop on Hot Topics in Understanding Botnets (HotBots'07), 2007.

[47] "An approach of the Naive Bayes classifier for the document classification".

[48] Weka: Practical Machine Learning Tools and Techniques with Java Implementations Ian H. Witten, Eibe Frank, Len Trigg, Mark Hall, Geoffrey Holmes, and Sally Jo Cunningham, Department of Computer Science, University of Waikato, New Zealand.

[49] Turing, A.M. (1950). Computing machinery and intelligence. Mind, 59, 433-460.

APPENDIX A

Trigger Words

Below is a list of words, which are characterized as trigger word/phrases by analyzing the log-classified bot dataset from May and June 2009. In addition to that there are words/phrases that has been added afterwards by looking into the data.

pm me	webcam	meet real girls
naaked	myCam	MUST delete
profile	Copy this	turned 21
21 years	Goto link	got pics
my pics	show me	Click below
Click here link	No age restrictions	profile homepage
real beautiful girls	Real women	Copy url
Me real	Chat me	Swingerforme
Realwomen	asl plz	hotwebmcam
Goto homepage	Real-dates	Dategirl
Girllyaction	As.Sexy.As	Real-dates
Can be friends?	Sexyfriend	Hookups now
Formen	Earn$	$$$

I am waiting	Call me	Lovely cam
H0t girlz	For free	Feeling flirty
.net	Got pics?	female for friendship
feeling flirty	interested me	Truthful girlz
Bored here	cutevpmsvkyir	
Neew caam	nakeed	
pimpkindaplonle	find ur job	