DEMOGRAPHICS OF ADWARE AND SPYWARE

Except where reference is made to the work of others, the work described in this thesis is my own or was done in collaboration with my advisory committee. This thesis does not include proprietary or classified information.

_____

Kavita Sanyasi Arumugam

Certificate of Approval:

_____	_____
Dean Hendrix Associate Professor Computer Science and Software Engineering	David A Umphress, Chair Associate Professor Computer Science and Software Engineering
_____	_____
Cheryl Seals Assistant Professor Computer Science and Software Engineering	George T. Flowers Interim Dean Graduate School

DEMOGRAPHICS OF ADWARE AND SPYWARE

Kavita Arumugam

A Thesis

Submitted To

the Graduate Faculty of

Auburn University

in Partial Fulfillment of the

Requirements for the

Degree of

Master of Science

Auburn, Alabama
December 17, 2007

DEMOGRAPHICS OF ADWARE AND SPYWARE

Kavita Arumugam

_____
Signature of Author

_____
Date of Graduation

THESIS ABSTRACT

DEMOGRAPHICS OF ADWARE AND SPYWARE

Kavita Arumugam

Master of Science, December 17, 2007
(B.E., Sir M Visveswaraya Institute of Technology, 2004)

60 Typed Pages

Directed by David Umphress

The World Wide Web is the most popular use of the Internet. Information can be accessed from this network of web pages. Unknown to users, web pages can access their personal information and, sometimes, also provide information that the user has not asked for. Various kinds of software are used in the web pages. Web pages use Java, Perl scripts, XML, etc. There are additional software like adware and spyware being used. This software could be useful or threatening. Our interest lies in knowing which software being used presents a threat on the web.

A user who is worried about his online privacy may be more interested in knowing what types of technology is being used on the web. Adware and spyware are types of software that are present in web pages unknown to the user. If they are present when a web page is accessed, they make use of the user's Internet connection to track and send data and statistics via a server installed on the user's computer or the users' client. Most of the legitimate adware and spyware companies disclose in their

privacy statement the nature of data that is collected and transmitted; but there is no way a user can control the kind of data that is being sent.

A study was conducted to determine the percentage of web pages that contained certain types of adware and spyware. It was found that 16% of web pages contained web bugs while 1% of web pages contained ActiveX objects.

ACKNOWLEDGMENTS

I wish to express my deepest gratitude to Dr. David Umphress for his motivation and guidance throughout the research work. I wish to thank Dr. Dean Hendrix and Dr. Cheryl Seals for their time and unwavering support. My gratitude goes out to my parents, my sister, and my friends who have supported me and helped me in all the phases of my life at Auburn. I wish to thank Santosh for helping me during my stay in Auburn. A big thanks to all my teachers who have taught me to stay focused on my goals and the Almighty God for showing me the way.

Style manual used**:** ACS Computing Surveys.

Computer software used: Microsoft Word, Microsoft Excel, Microsoft Access, Java.

TABLE OF CONTENTS

LIST OF FIGURES

1 INTRODUCTION

1.1 Background

In today's world, one can shop online for clothes, tickets, etc., schedule payments of bills, and transfer money between different banks. Sensitive information such as credit card details; bank routing and account numbers; and social security number are used to complete these transactions. Most websites that carry out these transactions are secure. But sometimes, websites are monitored by certain programs and the user's online activity communicated to a third party. These programs are called adware and spyware. They are also known under the names of malware or trackware. They can be present in web pages as a hidden addition to a legitimate program that is being downloaded from a website, or can be directly installed on a computer.

Spyware and adware may be written so as not to reveal their existence on web pages. They take advantage of users' consent to install some piece of legitimate software. For example, Audio Galaxy is a company that makes Napster-style file sharing software. When users download Audio Galaxy, they also download VX2's spyware program. Audio Galaxy is a legitimate piece of software. VX2 is not. This spyware program keeps track of the websites a user visits and passes on user information to the company's servers [Benner 2002].

Adware is software that is usually free. When it is executed, it displays advertisements from the Internet. The advertisements can be viewed through pop-up windows or through search bars that appear at the bottom of the screen. The justification for adware is that it helps recover programming development costs through ad revenue. For example, Google's Blogspot service contains JavaScript that tries to convince users to install software. Pop up boxes appear to come from a website iWebTunes.com. This website gives bloggers music to add to their blogs or other websites. When a user views this blog, iWebTunes attempts to install extra programs on the users' machine. These programs pay iWebTunes a commission for every installation made [Edelman 2005].

Software components used for tracking and reporting user information are included in most adware. These components collects web browsing history, on-line purchasing behavior, and an inventory of the computer hardware and software it runs on. After collecting information from the user, companies target users with specific advertisements based on browsing history.

Spyware is software that is installed on the computer without the user's knowledge. It exists as an independent executable program, unlike adware. It is designed to track the surfing habits of a user, collect personal information such as credit card number, the games the user plays, the software being used, the keystrokes etc, all without the user's permission [Zhang 2005], [Anonymous 2004], [Daniels 2004], [Doyle 2003] and [Taylor 2002]. The software sends this information back to the origin (creator's servers) where it is collected [Urbach et al. 2004] and possibly used for identity theft operations including password harvesting and credit card number theft [Radcliff 2004]. Sometimes, the software hides the information on the user's hard drive for later retrieval [Doyle 2003].

Surveillance spyware is used by law enforcers and industrial spies, or by corporations to keep a check on their employees. This software monitors and records keystrokes or web activity, or occasionally captures an image of the monitor screen. That information is then either e-mailed to the spying party or hidden on the hard drive for later retrieval [Ferrer and Mead 2003].

Spyware is often concealed within another application. Web pages contain code that downloads and installs spyware, usually through exploits. Spyware is installed without the user's consent, as a drive-by-download, or as the result of clicking some option in a pop-up window [Thompson 2005]. Drive-by downloads are applications that install themselves on computers without the user's knowledge during visits to websites. Drive-by download is capable of remote monitoring and reporting to actual Trojans with remote administration capabilities [Schwartz et al. 2004].

When someone installs adware and spyware components, they are often assigned a unique identifier by the software. Thus, the program can track the user and target advertisements catered to the users' choice. For example, if a user is looking to buy a new house, the software provides housing advertisements and house loans as well as additional pop-up ads to match his needs. Some users like to view advertisements targeted to their needs; however, these users are not aware that the information collected by these marketers is then sold to third parties. The information can include the user's name, address, email address and any other information they may have gathered from the user's personal profile.

1.2 Problem Statement

Adware and spyware are not illegal; however, a user who is worried about his privacy may want to know what private information is being divulged without his consent. The issue that is of concern involves the use of the user's Internet connection to track and send data and statistics via a server installed on the user's computer or the users' client. Most of the legitimate adware and spyware companies disclose in their privacy statement the nature of data that is collected and transmitted. But there is no way a user can control the kind of data that is being sent.

Spyware wastes bandwidth, interferes with the programs running on the machine and uses disk space. Some spyware is also known to cause crashes and stability problems on users' computers [Digital Insight Security Bulletin 2005]. Other spyware offers a serious security risk by opening a backdoor on the system, offering the capability to secretly install additional software.

The web is very accessible. As long as the users do not face a problem in accessing the web, they are least worried about the kind of software being used on the web. Most users are unaware of the adware or spyware embedded in them. Users hence have no statistics of how many web pages have this kind of information. The goal of this research is to identify what percentage of web pages contain adware or spyware.

The main work is identifying the various kinds of adware or spyware embedded in web pages and how they can be detected in web pages. A web crawler with the ability to crawl through a representative number of web pages is used to search for adware or spyware embedded in them.

This work will benefit people who use the Internet. Most people are unaware of the spyware and adware that gets downloaded to their machine when they are using the web [Zhang 2005]. This work will give people an idea of what is present on the web.

2 RELATED WORK

2.1 Background Information

Millions of users have been impacted by the World Wide Web. The WWW is so popular that various kinds of software are integrated into it. It usually gives information to a user. Sometimes, it also takes information from the user. Users, most often, willingly give their information, but there is some software that does not ask the user for any information. It takes the information without the users' knowledge. This information usually is bought by a third party who uses it to his advantage.

Malware is short for malicious software. Malware requires special conditions if it is to execute and produce the intended results. The code most software vendors produce nowadays is not carefully designed or tested. This brings about software that has so many vulnerabilities. This gives perpetrators a chance to execute malware that exploits the vulnerabilities [Skoudis and Zeltser, 2003]. In some cases, malware gets installed on a computer without the user's knowledge. It can change browser settings as well as system settings, causing potentially harmful effects to occur on the computer.

2.1.1 Web Bug

A Web Bug is usually invisible to the user. It is also called a Web Beacon. It is a graphic image of size 1-by-1 pixel found on a web page or an email message [Doyle 2003]. Web bugs are represented by HTML IMG tags. The web bug might allow a third party to send pop up ads or just collect demographics. Usually, the web bug is loaded from a different web server than the remainder of the page. This is how a web bug can be differentiated from a normal 1-by- 1 pixel image [Smith 2003]. This spyware monitors the IP address of the machine that opened the page with the web bug. It also sends information about the type of the browser that opened it, the time it was viewed and the URL of the web page that had the bug.

If the web bug is embedded in an email, the image is requested when the user reads the email for the first time. It can also be requested every time the user opens the email again.

When a web page with a bug is downloaded, the server where the page resides stores the IP address of the computer requesting the page. This information can be retrieved from the server log files. When files are transferred using the Hypertext Transfer Protocol, web bugs send the server their URL, and the URL of the page containing them. The URL of the page containing the bug allows the server to determine which particular Web page the user has accessed. The URL of the bug can be appended with an arbitrary string in various ways while still identifying the same object. This extra information can be used to better identify the conditions under which the bug has been loaded. This information can be added while sending the page or by Java scripts after the download.

The following is an example of a web bug found on Quicken's home page www.quicken.com :

<img src="http://ad.doubleclick.net/ad/pixel.quicken/NEW" width=1 height=1 border=0>

This bug provides to DoubleClick, an Internet advertising company, information about the number of visitors [Smith 2003].

2.1.2 Adware Networks

When companies want online publicity, they approach software developers or web sites and pay them to allow their advertisements to be displayed when people use their software. This software is called adware networks [BLEEPING COMPUTER]. These ads are generally in the form of popups. The problem with these networks is that they place cookies on the computer each time the user opens an ad served by the particular network. This allows the advertising network to track the user's movements across the Internet by reading the information contained in the cookies every time a user connects to a site. Networks that employ this method include DoubleClick [www.doubleclick.com], Value Click [www.valueclick.com], Gain [www.gainpublishing.com], and Radiate [www.cexx.org/aureate.htm].

A very good example of adware networks is the 180solutions.com website. This is the world's largest adware networks. Software from 180solutions redirects many affiliate commissions to 180solutions. This transmits information about the web sites that the user

visits to its server. 180solutions.com shows pop up ads which cover all of the targeted web sites. Programs from 180solutions monitor users' activities and show targeted advertisements, but 180 programs also overwrite affiliate commissions to cause 180solutions to receive payments from merchants when users make online purchases. Benjamin Edelman, an assistant professor at Harvard Business School, conducted an extensive study and published results of his study on his web site. In his study, he showed that when he browsed for www.delta.com, the instructions caused 180solutions' software to show an ad for Hawaiian airlines that covered almost the entire delta.com web page that opened on his machine. Further in his research, he found that the advertisers who sponsored 180 solutions' web site paid as little as $0.015 per display of their ads [Edelman 2004].

2.1.3 Backdoor Santas

Users download programs from the Internet. On the surface, programs appear valid. However, they collect statistics on computer usage, browsing history, hardware a computer uses and transmit the information back to servers. These programs are called Backdoor Santas. A Backdoor Santa is a stand - alone program that gathers user information. It bypasses normal security controls to give the attacker access to useful and potentially valuable data; hence the name Backdoor Santa [Sipior et al. 2005].

A good example of a Backdoor Santa is a novelty cursor representing a seasonal icon or the likeness of Dilbert or a Peanuts character. When a program to make the customized cursor is downloaded, a Globally Unique Identifier (GUID) is issued. This

GUID helps the provider's servers to record without the users' permissions logs of cursor impressions/ cursor themes, the identity of referrers, Internet Protocol (IP) addresses, and system information. This data is sold by the providers to clients to inform them how many users have customized cursors obtained from certain websites [Tipton and Krause 2006], [Sipior et al. 2005] and [Smith 1999].

Programs of this type are distributed by Comet Cursor [www.cometcursor.com], Alexa [www.alexa.com], Hotbar [www.Hotbar.com], and Cuteftp [www.cuteftp.com].

2.1.4 Trojan Horse

Trojan horse is named after the Trojan horse tactic in Greek history, where something unknown and unexpected is delivered to the user in the form of a package, which the user normally accepts [Pastore 2002]. These types of software are often popular programs and are usually free downloads. Trojan horses involve installing programs that can be contacted by remote machines which take over control over the user's machine. Email attachments are a popular mechanism for delivering Trojan horses [Mikusch 2003].

A Trojan horse program masquerading as an advertising application was included with versions of programs BearShare [www.bearshare.com], LimeWire [www.limewire.com], Kazaa [www.kazaa.com] and Grokster [www.grokster.com]. The Trojan, called "W32.Dlder.Trojan", is found within an application called "ClickTillUWin" which promises users a chance to win prizes [Borland 2002]. The Trojan file Dlder is installed when users set up the file sharing applications. After

installation, this Trojan downloads an additional file called explorer.exe from a website 2001-007.com and installs it in the system folder. It then creates a startup key for the explorer.exe file. When the system is restarted the next time, the Trojan is connected to the 2001-007.com website. It keeps track of the users' web activity and reports this to a web server.

In some situations, if a user chooses not to install the program containing the Trojan horse, he/she will not be able to use main program. Examples of such programs are KaZaA Media Desktop [www.kazaa.com], Grokster [www.grokster.com], and Morpheus [morpheus.com].

2.1.5 Browser Hijackers

Browser hijackers change the default web page setting on the user's browsers without permission. The software changes the default homepage to another homepage no matter how many times a user changes it. It does this by making changes to the system registry [Mikusch 2003]. Sometimes Internet shortcuts will be added to the Favorites folder of the web browser without the user's permission. The purpose of this is to force the user to visit a web site of the hijacker's choice so that they can inflate their web site's traffic for higher advertising revenues [Healan 2005].

In 2005, AOL was labeled a browser hijacker. AOL placed its web site free.aol.com in Internet Explorer's trusted sites security zone, thus bypassing the most frequently used security settings. This occurred only after a user installed the AOL software. After that, AOL downloaded ActiveX components to the computer without the user's consent. These components led to pornographic web sites [Healan 2005].

One of the infamous hijackers known to date is the CoolWebSearch. It registers Winres.dll under the Windows directory and then changes the Start page to about-blank. It downloads and installs other searches such as 2020search, isearch, etc. which offer CoolWebSearch a fee in exchange for a visitors' use of their search program [Spywareguide 2007].

2.1.6 Dialers

Dialer is a colloquial term for Dialing Software. This software gets installed on a user's computer and has the ability to make phone calls from the computer if a modem is connected to it. These programs will connect to other computers, through the phone line, without the user's permission. Most of these numbers are not toll-free calls; consequently, the user gets charged for the amount of time the computer is connected to it [Shukla and Nah 2005], [Pastore 2002].

A good example is the Fairtale Dialer. A computer user downloads a software package. Among the programs is a dialer application that was not mentioned in any of the licenses or advertisements associated with the package. The dialer application is not an integral part of the software package. When the user opens the Web browser after installation of the software, the dialer opens in a hidden window, turns off the sound of the user's computer, and calls a phone number without the user's permission [Internet Security Services]. Once this is installed, it makes long distance calls or calls to 900 and 976 phone numbers without asking for the user's approval. These calls are usually adult

pay-per minute phone services. Thus, the user pays for the call even though he did not make it.

2.2 Related Work

Alexander Moshchuk, Tanya Bragin, Steven D. Gribble, and Henry M. Levy from the University of Washington conducted a crawler-based study of spyware on the web [Moshchuk et al. 2006]. In this experiment, the group studied and confirmed the existence of spyware. The objective of this study was to quantify the amount of spyware in executable web content and the number of web pages that contained embedded drive-by download attacks. These programs get downloaded by exploiting the web browser, or operating system bug.

To conduct the study, they used the Heritrix public domain web crawler. The data captured was analyzed using a virtual machine (VM) and Ad-Aware, an anti-spyware product from Lavasoft. Once installed, this product reduces the risks of pop-up ads, browser hijacks and theft of any information from that machine. It uses a technology that can detect known and unknown variants of malware. The web crawler crawled sites from eight different categories: adult entertainment, gaming, music, celebrity, screensaver, children zones, online news, and pirate sites. For each of these categories, web sites were selected using Google directory and key-word searches specific to that particular category.

Their study was divided into two parts. In the first part, almost 20 million URLs were crawled in search of executable content. They reached the conclusions that adware

constituted the largest portion of spyware. It also showed that gaming sites and sites that allowed wallpaper downloads had more spyware in them than the other categories. The second part of their study involved three crawls of 45,000 URLs in the eight web categories. In this part of the study, they examined drive-by download attacks over a time slot of 5 months.

This study has a few limitations. The results were based on samplings of web pages on Google selected domains and URLs in the eight categories. Using an anti-spyware tool like Ad-Aware allows the detection of only what it considers a threat. Overall, the study was able to quantify the nature of the spyware threat and thus confirmed the existence of spyware over the web.

In another work carried out at the University of Washington, the authors used passive network monitoring to measure the extent to which four specific adware programs had affected computers in the university [Saroiu et al. 2004]. This work used the honeypot technique. A honeypot is a closely monitored network that can provide early warning about new attacks and exploitation trends, and allow detailed examination of adversaries during and after the attack. Since physical honeypots are time consuming and expensive to set up, Niels Provos came up with a framework for virtual honeypot called Honeyd. This simulates computer systems at the network level [Provos 2004].

The Strider HoneyMoney project developed by the Microsoft Research team is also inspired by honeypot techniques. The system the researchers built was called HoneyMonkey Exploit Detection System. This consists of a pipeline of monkey programs running possibly vulnerable browsers on virtual machines [Wang et al. 2006]. A monkey program is a program that drives a browser in a way that mimics a human user

operation. These virtual machines have different patch levels and they patrol the web to seek and classify web sites that exploit browser vulnerabilities. The HoneyMonkey project report focuses more on the construction and design of the tool.

Quite a few commercial anti-spyware companies have used web crawlers to find new spyware on the web. Webroot's Phileas system uses a cluster of computers to scan web content for known threats and patterns that suggest of new browser threats [Webroot Software Inc, 2007]. Sunbelt Software has also built a web crawler SPECTRE that identifies new spyware outbreaks [Sunbelt Software 2007].

3 APPLICATION DEVELOPMENT

3.1 System Architecture

When users surf the internet, they have little knowledge of how much malware is there. A study of the amount of malware over the internet will enlighten users with what technology they are encountering in their daily life. To study the web, a stable, powerful, and accurate web crawler is needed. Once the crawler gets the results, the results need to be stored for later retrieval. JDBM is a package that is used for this purpose. It stores data using a hash table and a B+ tree data structure. In order to study the data collected, the results need to be imported into a database like MS Access. A statistical analysis of the data will help people make a decision of what malware is being found excessively in web pages.

3.1.1 Web crawler

In order to detect adware and spyware on the web, the program written needs a web crawler that can search the web thoroughly. A web crawler is a program that browses the World Wide Web in a systematic fashion. A crawler resides on a single machine. Web crawlers start by parsing a specified web page, noting any hypertext links on that page that point to other web pages. They then parse those pages for new links, and keep doing it in a recursive manner. Figure 3.1 shows a generic web crawler architecture.

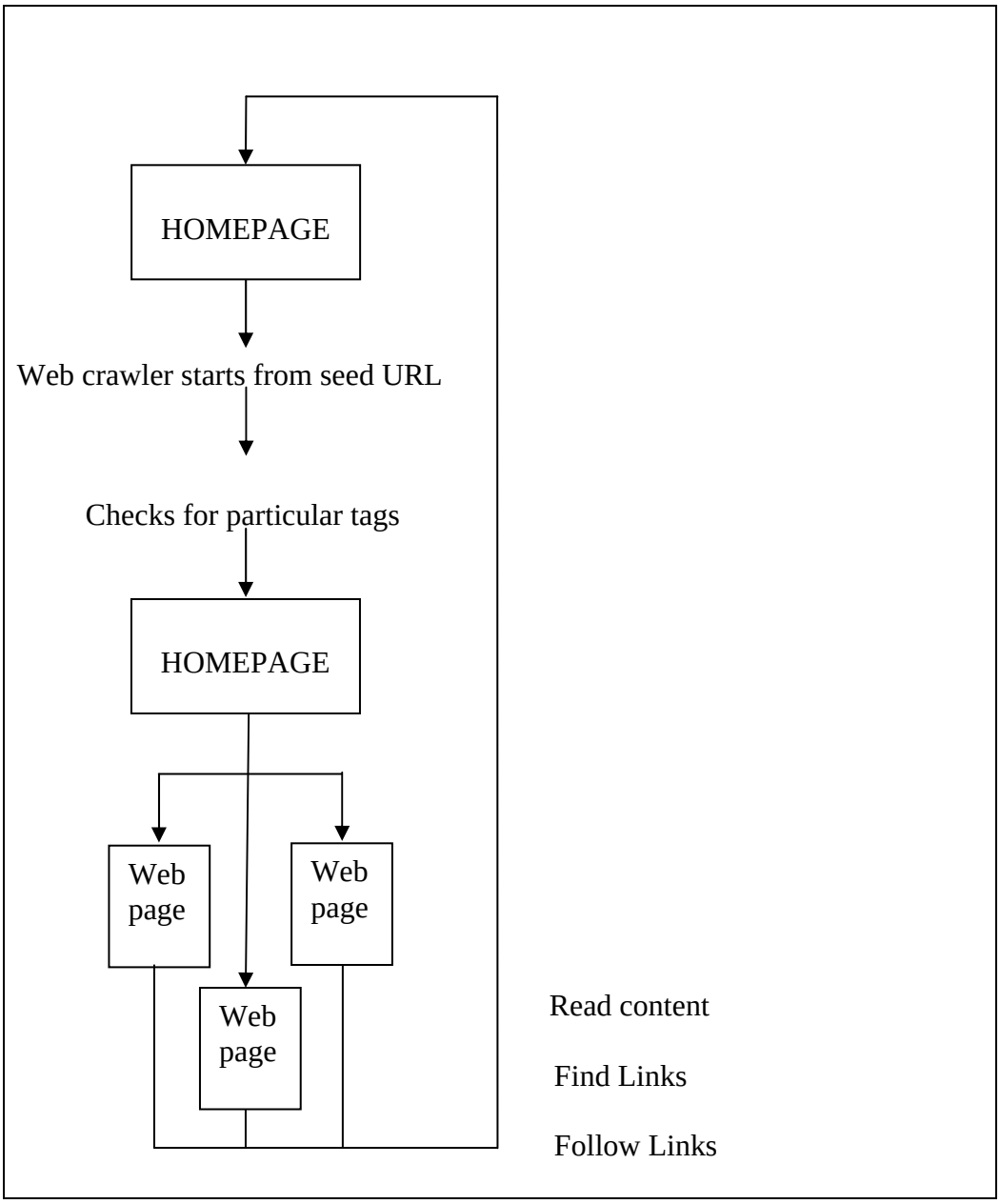

Figure 3. 1: Web crawler architecture

Search engines such as Google, etc. use web crawlers [Hawking 2006]. These engines operate multiple data centers that are distributed throughout the world. This ensures fault tolerance. Fault tolerance is important because should an agent crash, the other agents will decide who should fetch a certain host. Within a data center, PCs are

clustered according to the services they provide. Clusters are dedicated to specialized functions, such as crawling, query processing, and result caching. Currently, the amount of web data that search engines crawl and index is in the order of 400 terabytes [Hawking 2006].

Certain terms used frequently while discussing web crawlers need to be defined. A URL or Uniform Resource Locator is a web page address. The term crawling refers to traversing the web by recursively following links from a seed. A seed is a URL provided at the start of the crawl.

The simplest web crawling algorithm uses a queue of URLs yet to be visited and a fast mechanism for determining if it has already seen a URL. The crawler initializes the queue with one or more seed URLs. Crawling proceeds by making a HTTP request to fetch the page at the first URL in the queue. When the crawler fetches the page, it scans the contents for links to other URLs and adds each previously unseen URL to the queue. The crawler saves the page content for indexing**.** Crawling continues until the queue is empty.

Web crawlers start by parsing a specified web page, noting any hypertext links on that page that point to other web pages. They then parse those pages for new links, and so on, recursively. Web-crawler software doesn't actually move around to different computers on the Internet, as viruses or intelligent agents do. A crawler resides on a single machine. The crawler simply sends HTTP requests for documents to other machines on the Internet, just as a web browser does when the user clicks on links. All the crawler really does is to automate the process of following links.

3.1.2 JDBM

JDBM is a transactional persistence engine for the crawler. This is used to store objects and Binary Large objects (BLOB), and all updates are done in a transaction safe manner. It provides data structures such as B+ tree to support persistence of large objects [Groot 2000].

A B+ tree is a specialized tree designed to branch out in a number of directions such that the height of the tree is relatively small. A B+ tree of order M is an m- ary tree that has the data items stored in its leaves and whose root is either a leaf or has between 2 and M children. There are two types of nodes. The internal nodes contain key values and node pointers. The leaf nodes contain key and record pointer pairs. Each internal node is designed to fit into one I/O block of data. Hence, an internal node can keep a lot of keys. Each node except the root has between |m/2| to m children. Each node except the root has between |m/2|-1 and m-1 keys. Figure 3.2 sketches a B+ tree with keys stored in the node.

In a B+ tree, data records are only stored in the leaves. If a target key is less than a key in an internal node, then the pointer just to its left is followed. If a target key is greater or equal to the key in the internal node, then the pointer just to its right is followed. The leaves are also linked together so that all of the keys in the B+ tree can be traversed in ascending order, just by going through all of the nodes in this linked list along the bottom level of the tree [Carlson 2007].

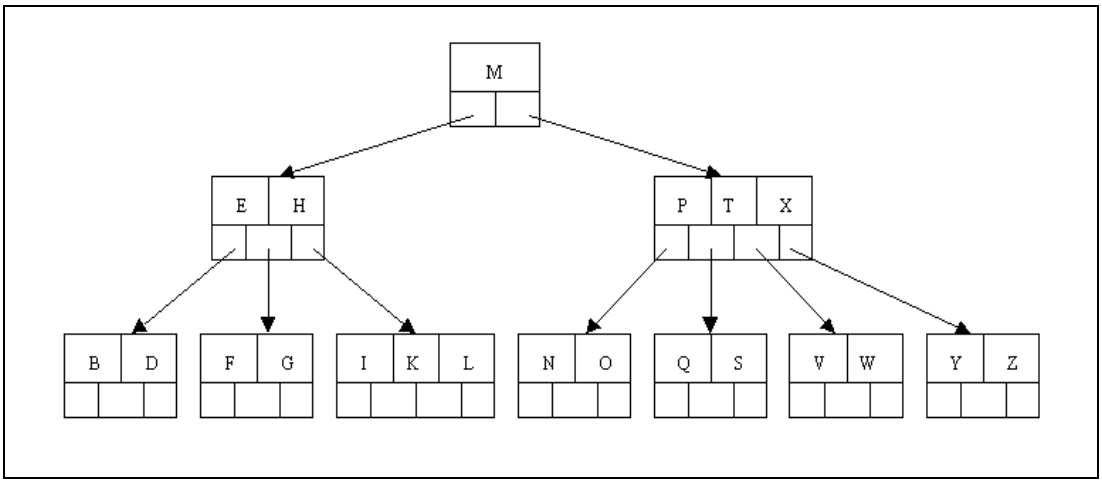

Figure 3. 2: A B+ tree

3.2 Architecture of the program

Figure 3.3 shows the system architecture for the IDemo application. The application is composed of seven core components which interact with each other in order to crawl the web and record data. Among all, the GUI component is primarily responsible for handling all the user interface related operations of the application. Apart from displaying the results and allowing the user to configure various run-time parameters like the number of threads, the total number of links, etc., the GUI is also responsible for receiving the seed URL for IDemo component from the user. On receiving the seed URL, the IDemo component - which controls the entire application - sets up other components like the queue component (theQueue) to store various URLs found during the crawl, the depth check component (theDepthCheck) to store the depth of the parsed URLs for each domain crawled and the database component (theDatabase) to store the parsed URLs as well as their statistics. IDemo then spawns a specified number of connection Threads (CThread) to crawl the web starting from the seed URL received.

The connectionThreads then crawl the web and record its demographics in the database. For each valid URL retrieved from the queue, the connectionThread checks to see if the depth check fails for that domain. If not, the connectionThread parses the URL for known technologies and records the results in the database. URLs discovered during such parses are inserted into the queue for a later retrieval. The connectionThread continues this fetch-check-parse-record cycle till it either has recorded the specified number of URLs in the database or till the queue component runs out of URLs. In case of the latter the user is prompted to enter the seed URL again. In the case of the former, it has to run till completion.

Upon completion, the user can generate a report of the IDemo run by invoking the reporter component. The reporter reads the database populated by the connectionThreads and outputs a text file containing technology statistics recorded for every URL in the database. This file can then be used by the user to analyze the demographics of the Internet.

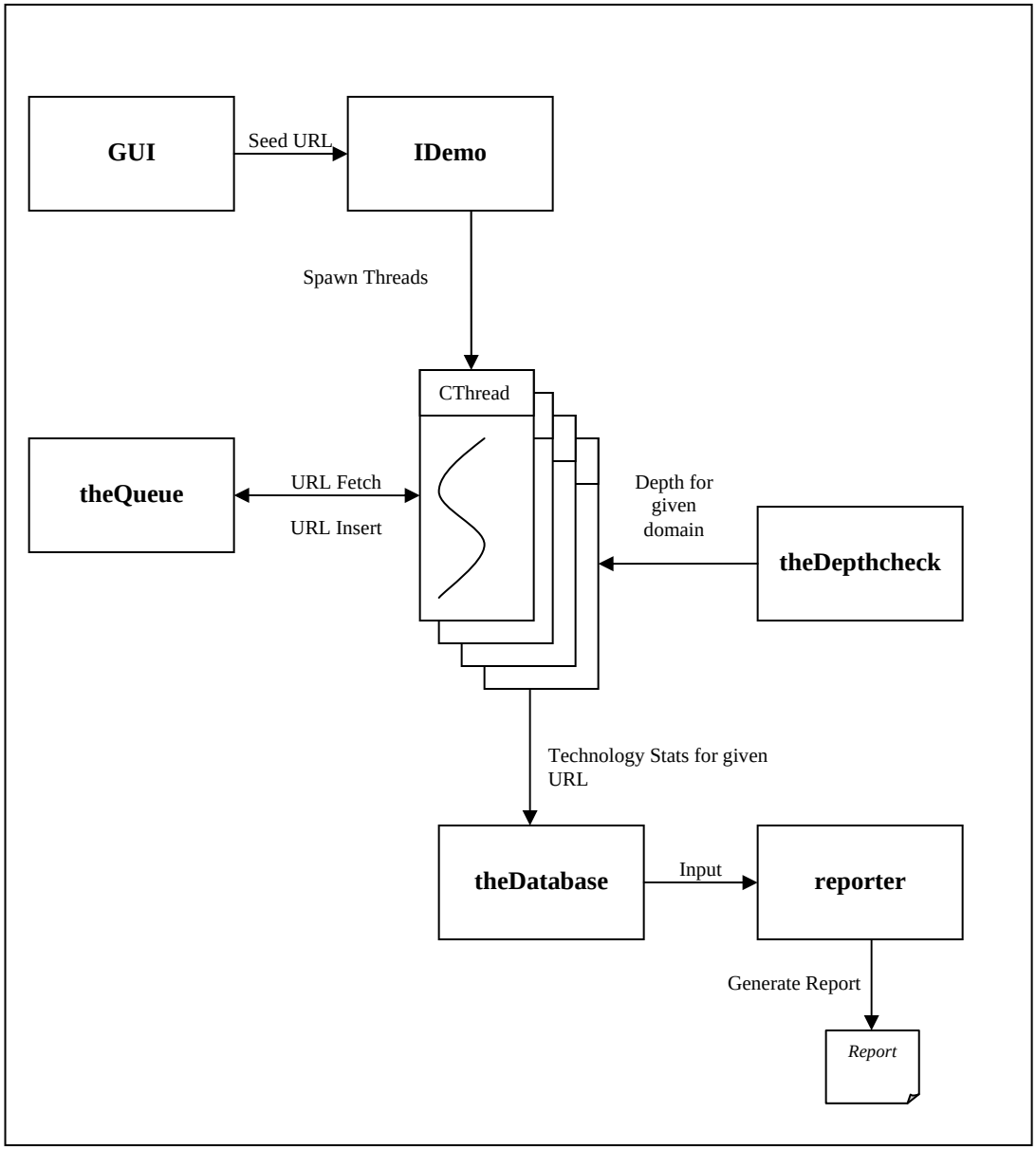

Figure 3. 3: System Architecture of Internet Demographics

3.3 Understanding the program

The program contains several classes. An understanding of the classes is essential to understand the various aspects of the program.

3.3.1 Class IDemo

This is the root class. It is the main controller of the entire application. It is responsible for creating the queue, database and depth. When the queue is empty, IDemo creates the GUI object so that the user can enter the seed URL. When the “Start” button is clicked on the GUI, IDemo creates and starts the specified number of crawl threads. If the queue becomes empty at any point of time, it prompts the user to enter a seed URL again. After the program is ended, it closes all the data structures created.

3.3.2 Class crawlthread

This class is the worker bee of this application. It makes use of two variables: One to name the thread and the other to keep track of the current thread number. When started, the thread does the following:

- Retrieves a new URL from IDemo’s queue.
- Converts the URL to a proper URL by removing hex symbols from the URL and concatenating a “/” to the ones with no path.
- Makes sure that the user-specified depth limit is not exceeded.
- Creates a new connection to the URL obtained through the previous steps.

If the established connection is good, the following steps occur:

- Makes sure that the URL has MIME type text/html.
- Makes sure that the URL is not already in the database.
- Makes sure that the URL does not have a null domain.
- Makes sure that the user specified depth limit is not exceeded.

- Creates token server and passes token stream of the above URL to it.

For every token retrieved, it does the following:

- Tries to identify the token.
- Checks if the token is a link whose technology can be identified.
- Makes sure that the link is a valid URL.
- Makes sure that the depth or the max limits are not exceeded.
- Finally, makes sure that the link is already not in the database before inserting it into the queue.

If the established connection was good, it adds the URL entry to the depth tree and adds the URL to the database. Otherwise, it discards the entry. The statistics on GUI is updated. It also checks if the thread needs to be killed.

3.3.3 Class identifyToken

This class is used to parse all the tokens returned from the token server. It is also used to maintain statistics on the various technologies found in a given URL. The constructor identifyToken does the following:

- Checks to see if the no-follow flag is set on the token.
- Checks to see if the token returned indicates an applet.
- Checks to see if the token returned indicates a form.
- If the token begins with “base”
    - Extracts the base URL enclosed along with “href”.
    - Store the base URL if it starts with “http”.

- If the token begins with “script”
    - Checks to see if the script present is javascript or vbscript.
- If the token begins with “object”
    - Checks to see if the application type is “activex”.
- If the token begins with “img”.
    - Checks to see if the dimensions are 1x1 in order to find “web bugs”.
- Invokes method lookforlinks().

The method lookforlinks does the following:

- Extracts the link from tokens with tags like a, area, link, frame and iframe.
- Ignores links to extensions such as .jpg, .mpg, .gz, etc.
- Resolves relative links.
- Makes sure that the links are of type “http” only.
- Invokes countTechnologies() for the candidate link in order to update stats and examines what is returned.
- If countTechnologies() was able to determine the link technology the isLink flag is set to true.

For a given candidate link, the method countTechnologies checks to see if the link has

- ActiveX
- Web bugs

If the technology on the given link has been found, it returns NULL. Otherwise, it returns a crawl link.

3.3.4 Class connectionThread

This class is responsible for establishing a HTTP connection to the given URL. When started, this thread does the following:

- Opens a HTTP connection to the URL taking into account redirections.
- If there are already a set of cookies for the URL, it retrieves them and sets them in the HTTP connection request property.
- Establishes connection to the URL.
- Handles cookies.
- Opens a buffered reader to the URL and sets up a tokenizer to tokenize stream based on angular brackets.

3.3.5 Class urlTokenServer

This class returns HTML tokens from the input stream. The method returnToken does the following:

- Returns NULL only if the end of stream is encountered.
- Ignores comments if encountered.
- Ignores new line while returning a token.
- Returns tokens enclosed between angular brackets one at a time.

3.3.6 Class connectionTimer

This class sets up a timer for specified number of seconds. It sets a flag called ‘loop’ to indicate that the timer has started at the start. When the timer has expired, the flag is turned off.

3.3.7 Class theCount

This class maintains a variable called ‘countOfPages’ to keep track of the page count. The functionality of this class can in fact be accomplished with the help of a simpler static variable. This class contains only basic functionalities like set, get and increment of ‘countOfPages’ variable.

3.3.8 Class cookie

This class is used to store session related information for a given connection. The whole class consists of a bunch of set() and get() methods for values like name, information, domain, path, secure and expiry.

3.3.9 Class GUI

This class is responsible for handling all the UI related operations of the application. Figure 3.4 illustrates the GUI in operation:

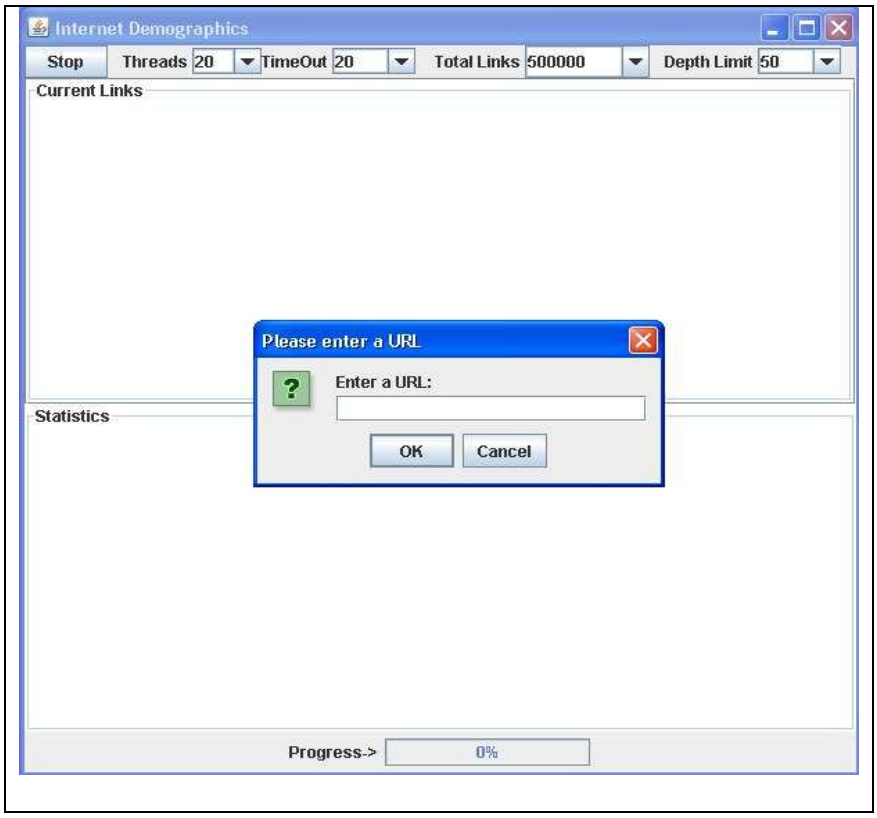

Figure 3. 4: Internet Demographics GUI

The UI allows the user start/stop the crawl. It lets the user configure the number of threads that can be run, the timeout value for each connection, the total links that are to be passed and the maximum depth of the tree. If the queue is empty the user is prompted to enter a seed URL. The user entered value is then inserted in the queue.

The progress of the whole operation is displayed in the progress bar at the bottom of the screen. The statistics of the number of links in the database, in the queue, depth checker and the total links examined are all displayed in the 'Statistics' frame.

The class also handles typical window operations like minimize, maximize and close. The window close operation also triggers IDemo's data structure cleanup.

3.3.10 Class theDatabase

This is the main data structure where all the parsed URLs and their associated technologies are stored. It makes use of java RecordManager and java BTree to implement a database. First, it creates record manager for the indicated file ("dbasefile"). Then it checks to see if a persistent binary tree for the above record manager already exists. If so, that binary tree is loaded off the disk. If a persistent binary tree does not exist, a new one is created and a reference for the tree is stored in the record manager using setNamedObject().

Insertion is done using dinsert() – which does the following:

- Creates a database entry for the given URL and technology stats array.
- Stores the above created entry with the URL in the binary tree making sure that duplicate entries are not allowed.
- The transaction for the record manager is then committed.

The method dsize() returns the current size of the tree. The dsearch() method is used to search for an entry in the database. If found, the technology stats array associated with the URL is returned.

3.3.11 Class theDomain

The methods in this class are used to determine the domain name of the given URL. By default, all unknown domains are set to "???". By default, all numeric domains are set to "999". The function parseTLD() does the following:

- Ignores the initial string of "http://"

- Gets the name of the domain from the substring found before the first '/' and after the first '.'
- Checks to see if the domain is numeric or not.
- Returns string if valid domain found, NUMERIC_DOMAIN if domain was numeric and UNKNOWN_DOMAIN if the domain is unknown.

3.3.12 Class theDepthCheck

This class is used to control the number of URLs that can be retrieved from a domain. It defines the upper limit on the URLs that can be retrieved from a given domain. It makes use of java RecordManager and java BTree to implement a database. First, it creates record manager for the indicated file ("depthfile"). Then, it checks to see if a persistent binary tree for the above record manager already exists. If so, that binary tree is loaded off the disk. If a persistent binary tree does not exist, a new one is created and a reference for the tree is stored in the record manager using setNamedObject().

The depthInsert() method is used to insert an entry into the database. It checks to see if an entry for the URL already exists. If there is no new entry, then a new entry is created and the count is set to 1. Otherwise, the count is retrieved, incremented by one and stored again overwriting the previous entry. The transaction for the record manager is then committed.

The method depthSize () returns the current size of the tree. The depthSearch () method is used to search for an entry in the database. If found it returns the number of

instances of a particular domain that is found in the tree. The findDomain() method is used to extract the host out of the given URL.

3.3.13 Class theQueue

Technically, this is not a queue as the insertions and deletions are not done in any particular order. This class makes use of java RecordManager and java BTree to implement a database. First, it creates record manager for the indicated file (“queuefile”). Then it checks to see if a persistent binary tree for the above record manager already exists. If so, that binary tree is loaded off the disk. If a persistent binary tree does not exist, a new one is created and a reference for the tree is stored in the record manager using setNamedObject().

The qinsert() method is used to insert a URL into the queue. It inserts a URL into the database without duplicates. It also performs a mass commit on the insertions when 100 entries are inserted. This is done to improve performance speed.

The method qsize() returns the current size of the tree. The qsearch() method is used to search for an entry in the database. If found it returns true else it returns false.

The qretrieve() method is used to retrieve an entry from the queue. It retrieves an entry from the tree and converts it into a URL to be returned, removes the entry from the tree and returns the URL if successful in the above operations. Otherwise, it returns NULL.

3.3.14 Class reporter

This class is used to convert the statistics collected in “dbasefile” database into a human readable format. It makes use of java RecordManager and java BTree to implement a database. First, it creates record manager for the indicated file (“dbasefile”). Then, it checks to see if a persistent binary tree for the above record manager already exists. If so, that binary tree is loaded off the disk. If the persistent binary tree is not found, it terminates reporting an error. It prompts the user to enter the path and filename where the output is to be stored. For each node found in the tree, the URL and its domain along with the technology statistics is output to the file.

3.4 Designing the software

For the purpose of this thesis, web bugs and embedded objects are the two technologies being detected.

3.4.1 Web bug detection

Typically, web bugs are images of size 1 by 1 pixel that get loaded from a different server than the remainder of the web page. However, during the study of web bugs, certain images whose sizes were not a 1 by 1 pixel were found to be loaded from a different server than the domain server. These images were found to behave like a typical web bug. Also, some of the web bugs of both 1 by 1 pixel and an M by N pixel were found to contain query strings. URL query strings can be used to pass information to the server in order to perform functions like displaying different data, passing information,

entering different mode and changing display format among a lot of other things. When a query string is present in the SRC field of an image tag whose display size is either set to 1 by 1 or to an M by N pixel, it can be reasonably counted as a web bug.

So, there are three cases of web bug detection that can be used to find most of the web bugs.

Case 1: A 1 by 1 pixel image with no query string

Since a web bug is always an image, to detect an image in Java requires the use of the IMG tag. The web bug is of specific dimensions, that is, height =1 and width =1. Figure 3.5 shows the code snippet used to detect these web bugs. The code snippet relevant to this case is indicated by the case number. The lowercasetoken looks for a particular token, i.e., a string value in the HTML code of a web page. If the image fits the specified height and width of one pixel, a successful match is obtained. The variable techArray gets incremented and stores the image details, such as the web page address, the type of web page, etc.

Case 2: A 1 x 1 image with a query string

A 1 x 1 image that has a query string in the SRC presents a stronger case for a web bug. The query string indicates that some kind of information exchange is taking place. In Figure 3.5, the code snippet under case 2 is relevant to this type of web bugs.

Case 3: An M x N image with a query string

An M x N image that is getting loaded from a different server than the web page itself may not be considered a web bug. But if the URL has a query string, then it could possibly be a web bug. These web bugs are also accounted for. The code snippet for this case is shown in Figure 3.5 under case 3.

```
Pattern heightq = Pattern.compile("height\\s*=\\s*\"{1}1\"{1}\\s*");
Pattern widthq  = Pattern.compile("width\\s*=\\s*\"{1}1\"{1}\\s*");
Pattern height  = Pattern.compile("height\\s*=\\s*1{1}\\s*");
Pattern width   =  Pattern.compile("width\\s*=\\s*1{1}\\s*");
Matcher hmatchq = heightq.matcher(parameters);
Matcher wmatchq = widthq.matcher(parameters);
Matcher hmatch  = height.matcher(parameters);
Matcher wmatch  = width.matcher(parameters);
//CASE 1: This is for links that contain query string
if(hmatch.find(0) && wmatch.find(0)|| (hmatchq.find(0) && wmatchq.find(0)))
{   if(lowercasetoken.indexOf("?") == -1)
   {   techArray[11]++;
   System.out.println("Caught a web bug with 1x1!!!\n");
       return;
    else{
//CASE 2: This is for links that don't contain the query
    techArray[12]++;
    System.out.println("Caught a web bug with 1x1 and query !!!\n");
        return;
    }}
else{
//CASE 3: This is for links that contain the query
        if(lowercasetoken.indexOf("?") != -1)
         {  techArray[13]++;
          System.out.println("Caught a web bug with MxN and query !!!\n");
          return;
         }}}
```

Figure 3. 5: Code snippet to detect web bugs

The results from these 3 cases will help in analyzing what percentages of web pages contain each category of web bugs.

3.4.2 ActiveX objects detection

ActiveX objects are inserted in web pages using the <OBJECT> tag. The object tag is part of standard HTML and is used to insert an object into a web page. The object tag contains a classid attribute [Detert 1999]. The classid is a long number that is found in the registry. The classid may begin with clsid and is found to be of the following format:

classid="clsid:CAFEEFAC-xxxx-yyyy-zzzz-ABCDEFFEDCBA"

In this form, the “xxxx”, “yyyy” and “zzzz” are 4 digit numbers that identify the specific version of Java plug-in to be used. Java applet, python applet and ActiveX controls use the classid attribute of the OBJECT tag [Quinn 2007]. The classid may also have a codebase attribute. This attribute tells the browser where to download the program needed, if it is not available on the machine [Sun Microsystems 2007].

To categorize an ActiveX object, the classid attribute should be of the following format:

classid = “clsid:xxxx”

These objects begin with clsid and are recorded as ActiveX objects. The code snippet to detect this is in Figure 3.6 highlighted as case1. If it is a java applet, the classid attribute is of the following format:

classid = “java: xxx”

If the classid attribute begins like above, it appears to be a java applet and is recorded under the java applet category. The code to detect this category is shown in Figure 3.6 under case 2. If the object embedded is a python applet, the classid attribute is as follows:

classid = “xxx.py”

In this case, the application ends with a .py extension. This kind of applets is recorded under the Python applet category. The code to detect python applets is shown in case 4 in Figure 3.6.

```
if(lowercasetoken.startsWith("object"))
{  if (lowercasetoken.startsWith("clsid"))
    {
        /* CASE 1: Found Active X object */
        System.out.println("Found Active X object");
          techArray[3]++;
      } else if (lowercasetoken.startsWith("java:"))
     {
        /* CASE 2: Found Java Applet within <object> tag */
          System.out.println("Found Java Applet");
          techArray[2]++;
     }  else if (lowercasetoken.endsWith(".py"))
     {
      /* CASE 3: Found Python Applet within <object> tag */
           System.out.println("Found Python Applet");
           techArray[14]++;
      }  else
      {
  /* CASE 4: Found an Unknown Applet within <object> tag */
           techArray[15]++;
       }
       return;
         }
```

Figure 3. 6: Code to detect ActiveX objects

4 VALIDATION AND RESULTS

The program was tested to detect a few known web sites that contained web bugs and ActiveX objects. The program was run with the seed URL being the particular web page.

4.1 Validation of web bugs

After analyzing web bugs, we came to the conclusion that 3 cases of web bugs were to be discussed. A web bug could be a 1 x 1 pixel image and may or may not have a query string. A web bug could also be an image of any size and may contain a query string. These 3 cases were tested with web sites known to contain them.

Case 1: A 1 x 1 pixel image that contains a query string

The U.S. Air Force web site (www.af.mil) is known to have a web bug that also had a query string embedded in it. The website was given as the seed URL to the program and the results obtained from this experimental study are shown in Figure 4.1.

URL	TLD	1x1 without query string	1x1 with query string	M x N with query string
http://www.af.mil	mil	0	1	0
http://www.af.mil/art/	mil	0	1	0
http://www.defenselink.mil/	mil	0	0	0
http://www.esgr.mil/	mil	0	0	0
http://www.mnf-iraq.com/	com	0	0	0
http://www.nationalmuseum.af.mil/	mil	0	1	0
http://www.navy.mil/	mil	0	0	0
http://www.posturestatement.hq.af.mi	mil	0	0	0

Figure 4. 1: Screenshot of the results from www.af.mil

Case 2: A 1 x 1 pixel image that does not contain a query string

The CNN News web site provided a very good test data set for web bugs with no query strings. The result of the program run is shown in Figure 4.2.

URL	TLD	1x1 without query string	1x1 with query string	M x N with query string
http://edition.cnn.com/	com	14	0	0
http://money.cnn.com/data/markets/dow/	com	4	1	1
http://money.cnn.com/data/markets/sandp/	com	4	1	1
http://topics.cnn.com/topics/michael_vick	com	10	2	0
http://www.cnn.com	com	16	1	0
http://www.cnn.com/2004/LAW/05/19/penalty.box/index.html	com	6	1	0
http://www.cnn.com/2007/HEALTH/08/15/abortion.pills.ap/index.h	com	6	1	0
http://www.cnn.com/2007/HEALTH/diet.fitness/07/27/senior.fitnes	com	10	1	0
http://www.cnn.com/2007/HEALTH/family/08/14/par.toy.tips/index	com	6	1	0
http://www.cnn.com/2007/LIVING/homestyle/08/13/o.dorm.tips/in	com	6	1	0

Figure 4. 2: Screenshot of the results from www.cnn.com

Case 3: An image pixel of any size that contains a query string

Indiatimes.com is a web site that provides news, shopping information, chat and mail features. This web site has web bugs that are not necessarily of size 1 x 1 pixels. A look at the page source shows that the web bug contains a query string. The result of the program run is shown in Figure 4.3.

URL	TLD	1x1 without query string	1x1 with query string	M x N with query string
http://auto.indiatimes.com/quickies/2284440	com	0	1	8
http://broadband.indiatimes.com/videoshow/2	com	0	1	21
http://chat.indiatimes.com	com	0	1	4
http://chat.indiatimes.com/	com	0	1	4
http://cricket.indiatimes.com/Aussie_crowd_	com	0	1	7
http://cricket.indiatimes.com/Chance_to_siz	com	0	1	7
http://cricket.indiatimes.com/Do_India_need_	com	0	1	7
http://cricket.indiatimes.com/News/NewsYou	com	0	1	7
http://cricket.indiatimes.com/statshow/22334	com	0	1	6
http://cricket.indiatimes.com/topbar.cms	com	0	0	0
http://in.indiatimes.com/default.cms	com	0	1	25
http://www.simplymarry.com	com	0	0	0
http://www.simplymarry.com/	com	0	0	0

Figure 4. 3: Screenshot of the results from www.indiatimes.com

4.2 Validation of ActiveX objects

In chapter 3, ActiveX objects were analyzed and it was shown that the object tag could have the classid attribute used in various ways. Depending on the inputs to the classid attribute, the cases have been formed. The program was tested on a few known web sites that satisfied the case requirements.

Case 1: ActiveX objects

The U.S. Navy website has some ActiveX objects embedded in them. The seed URL for this run was www.navy.mil. The results are shown in Figure 4.4.

URL	TLD	Active X	Python Applets	Unknown Applets
http://www.access-board.gov	gov	0	0	0
http://www.access-board.gov/	gov	0	0	0
http://www.access-board.gov/sec508/FAQ.htm	gov	0	0	0
http://www.access-board.gov/sec508/standards.htm	gov	0	0	0
http://www.adobe.com/products/acrobat/alternate.html	com	0	0	0
http://www.adobe.com/products/acrobat/readermain.html	com	0	0	0
http://www.navy.mil	mil	1	0	0
http://www.navy.mil/media/OtherMedia/YearInReview2006/	mil	1	0	0
http://www.navy.mil/midway/	mil	1	0	0
http://www.navy.mil/midway/CTM.html	mil	1	0	0
http://www.navy.mil/midway/Driscoll.html	mil	0	0	0

Figure 4. 4: Screenshot of the results from www.navy.mil

Case 2: Python applets

The Sourceforge web site offers a few examples for python applets that are found in the CLASSID attribute of the object tag. The seed URL given was http://grail.sourceforge.net/. The results from the run are shown in Figure 4.5.

URL	TLD	Active X	Python Applets	Unknown Applets
http://grail.sourceforge.net/	net	0	1	1
http://grail.sourceforge.net/demo/	net	0	0	0
http://grail.sourceforge.net/demo/audio/index.html	net	0	0	0
http://grail.sourceforge.net/demo/cache/index.html	net	0	0	0
http://grail.sourceforge.net/info/manual/test.html	net	0	1	0
http://grail.sourceforge.net/info/oldbugs.html	net	0	0	0
http://grail.sourceforge.net/info/papers/	net	0	0	0
http://grail.sourceforge.net/info/patches/	net	0	0	0
http://grail.sourceforge.net/info/security.html	net	0	0	0
http://grail.sourceforge.net/info/summary.html	net	0	0	0
http://grail.sourceforge.net/info/tables.html	net	0	0	0
http://grail.sourceforge.net/info/team.html	net	0	0	0

Figure 4. 5: Screenshot of the results from http://grail.sourceforge.net

4.3 Results

In order to quantify the amount of adware and spyware on the web, a sizeable measure of data was to be collected. For the purpose of this thesis, data collected from one million web pages was set as a target.

The seed URL for this experiment was www.cnn.com. From here, the program crawled web pages for four weeks to meet the target. The data base had a collection of one million web pages. The results were imported into MS Access for computational purposes.

From the study, it was found that 16.56% of web pages could have web bugs. ActiveX objects were found in approximately 1% of web pages. Various types of applets accounted for 0.06% of web pages.

Case 1: Web bugs

Images of M x N size that contained a query string in the SRC field were found in 6.5% of web pages. As explained earlier, there is a strong probability that these images are being used as web bugs. The images of size 1 x 1 pixel that contained a query string in the SRC field were found in 4.75% of web pages. This is a strong indication that a 1 x 1 pixel containing a query in the SRC field is a web bug. The images of size 1 x 1 pixel with no query string were found in 5.35% of web pages. This indicates the presence of web bugs.

Case 2: ActiveX objects

ActiveX objects that had the classid attribute beginning with clsid accounted for 0.97% of web pages. Python applets that make use of the classid attribute were found in 0.0002% of web pages. Applets which used classid attribute that contained unknown extensions were found in 0.01% of web pages. Since the presence of ActiveX objects is really low, we can say that these objects are not found too often in web pages.

5 CONCLUSIONS AND FUTURE WORK

In accordance with the goal set out in Section 1.2, a program that could detect the adware and spyware was set up. The program could detect two types of spyware: web bugs and ActiveX objects. Initial experiments were performed to ensure that the program would detect the listed spyware without errors. A sample of one million web pages was analyzed to quantify the percentage of web bugs and ActiveX objects present. This study of web bugs and ActiveX objects enables a web user to know what kind of spyware is present in web pages and how many web pages use them. It also indicates the popular choice of web technologies for spyware.

Web bugs constitute 16.56% of web pages. ActiveX objects constitute approximately 1% of web pages and applets constitute 0.06% of web pages. Applets and ActiveX objects seem to be used less frequently as a spyware as compared to web bugs. It was found that 11.5% of web pages that contained web bugs belonged to the .com domain. 0.4% of web pages that contained ActiveX objects were found mostly on .com domains followed very closely by .org domains which contained 0.39% of web pages that had ActiveX objects. Though a web bug is typically defined as a 1 x 1 pixel image, we found that M x N images are used more than 1 x 1 images as web bugs. 6.4% of web pages contained M x N web bugs while only 4.7% of web pages contained 1 x 1 web bugs.

This program can be extended to detect other types of spyware. Adware networks, backdoor santas, Trojan horse, browser hijackers and dialers are becoming very popular in web pages. A quantification of these technologies is important, so one can understand its use and how widely it is used in web pages. The program used for this study did not browse through the robots.txt file. This is a file that every server maintains. This file has information that tells a web crawler what folders it should not browse. Browsing through the robots.txt before traversing the web site indicates a “well-behaving” crawler. This change must be implemented in the program. A research on the target set for this type of study is essential. This can give a future researcher a better vision of the data collected with respect to the target set.

BIBLIOGRAPHY

ANONYMOUS, Spyware: Spycatcher *New Media Age*, (January 8, 2004), 24.

BENNER, http://www.wired.com/science/discoveries/news/2002/01/49960, (2002).

BLEEPING COMPUTER, Bleeping Computer, http://www.bleepingcomputer.com/glossary/definition231.html, Last accessed in 2006.

BLUM Thom, KEISLAR Doug, WHEATON Jim, WOLD Erling, Writing a Web crawler in the Java Programming Language, Muscle Fish, LLC, January 1998.

BORLAND John, *File sharing program carry Trojan horse*, CNET, http://news.com.com, January 2002.

CARLSON Br. David, Software Design Using C++, Computing and Information Science Department, St. Vincent College, Last accessed: January 9, 2007.

CEES DE GROOT, JDBM, http://jdbm.sourceforge.net, 2000.

COMPUTER ASSOCIATES SECURITY ADVISOR, http://www3.ca.com/securityadvisor/glossary.aspx

DANIELS J, Scumware.biz Educates about Dangers of Adware/Scumware, *Computer Security Update*, (5)2, (February 2004).

DELIO Michelle, What they know could hurt you, *http://www.wired.com/politics/security/news/2002/01/49430* , (January 3, 2002)

DENIZ Ekram, http://www.ekremdeniz.com/article3.htm, (January 11, 2005).

DETERT Ryan, The Amazing ActiveX - Part 1, *Internet Related Technologies*, (August 9, 1999).

URL: http://www.irt.org/articles/js178/index.htm

DOYLE E, Not All Spyware is as Harmless as Cookies: Block it or Your Business Could Pay Dearly, *Computer Weekly, (*November 25, 2003), 32.

EDELMAN Ben, http://www.benedelman.org/news/022205-1.html , (February 22, 2005).

EDELMAN Ben, The Effects of 180solutions on Affiliate Commissions and Merchants, http://www.benedelman.org/spyware/180-affiliates/#targeted-ads, (2004).

FERRER Daniel Fidel, MEAD Mary, Uncovering the Spy Network, *Computers in Libraries*, (23)5, (2003), 16.

HAWKING David, Web Search Engines: Part 1, *IEEE Computer Society*, (39)6, (June 2006).

HEALAN Mike, www.spywareinfo.com, (January 12, 2005).

HOWES Eric, Spyware Warrior, http://www.spywarewarrior.com/, (2006).

INTERNET SECURITY SERVICES, http://xforce.iss.net/xforce/xfdb/14338, IBM Internet Security Systems.

LAVASOFT, Lavasoft (2007).

MIKUSCH R., Adware, Spyware – Oh My!, *Beyond Numbers*, (427), (October 16, 2003).

MORRIS John, Programming Languages, Data Structures and Algorithms, http://www.cs.auckland.ac.nz/software/AlgAnim/ds_ToC.html, 1998.

MOSHCHUK Alexander, BRAGIN Tanya, GRIBBLE Steven D and LEVY Henry M., A Crawler based study of Spyware on the Web, *Network and Distributed System Security Symposium (NDSS)*, (February 2006).

NASD www.nasd.com, (January 28, 2005).

PASTORE Michael, Inside Spyware: A Guide to Finding, Removing and Preventing Online Pests, *Intranet Journal*, (2002).

PROVOS Niels, A virtual honeypot framework, *In Proceedings of the 13th USENIX Security Symposium*, San Diego, CA, (August 2004).

QUINN Liam, Object- Embedded Object, *Web Design Group*, http://htmlhelp.com/reference/html40/, (Last accessed: August 14, 2007).

RADCLIFF Deborah, Spyware, *Network World*, (21)4, (2004), 51.

SANDERS Tom, http://www.vnunet.com/vnunet/news/2152335/anti-adware-group-threatens, (2006).

SAROIU Stefan, GRIBBLE Steven D, LEVY Henry M., Measurement and Analysis of Spyware in a University Environment, *Proceedings of the First Symposium on Networked Systems Design and Implementation, San Francisco, CA, USA*, (March 29–31, 2004).

SCHWARTZ A, DAVIDSON A and STEFFAN M, Ghosts in Our Machines: Background and Policy Proposals on the "Spyware" Problem, *Washington, D.C.: Center for Democracy and Technology*, (July 16, 2004).

URL: http://www.cdt.org/action/spyware/

SHUKLA Sudhindra, NAH Fiona Fui-Hoon, Web browsing and Spyware Intrusion, *Communications of the ACM*, 48(8), (August 2005), 85-90.

SIPIOR Janice C, WARD Burke T, ROSELLI Georgina R, A United States Perspective on the Ethical and Legal Issues of Software, *ACM International Conference Proceeding Series*, Vol. 113, (August 2005), 738-743.

SKOUDIS Ed, ZELTSER Lenny, Malware- Fighting Malicious Code, (2003).

SMITH G, Tracking brawl: Is Big Brother watching you online, or are you just paranoid?, *ABCNews.com*, (17 December, 1999).

SMITH Richard M, http://www.computerbytesman.com/ , (2003).

Spywareguide, Facetime Communications Inc., www.spywareguide.com , Last accessed June 7, 2007.

STAFFORD Thomas F, Introduction to the special issue on spyware, *Communications of the ACM*, 48 (8), (August 2005), 34-36.

SUN MICROSYSTEMS, The Java Tutorials, http://www.irt.org/articles/js178/index.htm, (Last accessed: August 14, 2007).

SUNBELT SOFTWARE Inc., Sunbelt Software, http://www.sunbelt-software.com/CounterSpy/docs/battling_spyware_3.pdf, (2004).

SUNBELT SOFTWARE Inc., Counterspy Enterprise, http://www.sunbelt-software.com/Business/Counterspy-Enterprise/ , (Last accessed: June 4, 2007).

TAYLOR C, What Spies Beneath, *Time*, (160)15, (October 7, 2002), 106.

THOMPSON Roger, Why spyware poses multiple threats to security? *Communications of the ACM*, 48 (8), (August 2005), 41-43.

TIPTON Harold F., KRAUSE Micki *Information Security Management Handbook*. Fifth edition, Volume 3, Auerbach Publications, 2006.

UMPHRESS D, Web Software Demographics, *Proceedings of the 41st Southeastern Conference*, (March 2003, Savannah, GA), 457-462.

UNKNOWN, Spyware Causes, Effects and Prevention, *Digital Insight Security Bulletin*, (March 16, 2005).

URBACH Ronald R, KIBEL Gary A. Adware/Spyware: An Update Regarding Pending Litigation and Legislation, *Intellectual Property & Technology Law Journal*, (16)7, (2004), 12-16.

WEBROOT SOFTWARE Inc Automated threat research, *http://research.spysweeper.com/automated_research.html*, (Last accessed: June 4, 2007).

ZHANG Xiaoni What do consumers really know about spyware?, *Communications of the ACM*, 48 (8), (August 2005), 44-48.

WANG Yi-Min, BECK Doug, JIANG Xuxian, ROUSSEV Roussi, VERBOWSKI Chad, CHEN Shuo, KING Sam Automated Web Patrol with Strider HoneyMonkeys: Finding web sites that exploit browser vulnerabilities, *Proceedings of the Network and Distributed System Security Symposium*, San Deigo, CA, (February 2006).