Behavior-Based Spyware Detection

State of the Field

Ryan Mord

Saint John's University

Student
Ryan Mord

Advisor
Dr. Michael Heroux

Year 2014

CONTENTS

I. Abstract

Spyware is quickly becoming a major security issue. Spyware programs are installed on a user's workstation to monitor actions and gather private information about a user's behavior. Current antispyware tools work in a way similar to antivirus tools where code signatures associated with known spyware programs are cross-referenced against newly-installed applications. Unfortunately, these techniques are very easy to evade. Behavior-based detection aims to address the holes left behind by other techniques. This paper examines current spyware detection techniques and outlines test results from detection method comparison.

II. Introduction

While surfing the web a person is exposed to many threats. These threats commonly take the form of worms, trojans, viruses, and the topic of this paper: spyware. Studies done by the Associated Press and ABC News stated that 77% of survey respondents said that their home computers were safe from these threats. However, when these machines were examined, it was found that 80% had forms of spyware installed on them. Malware is one of the fastest growing computer security threats. Kapersky Labs, a computer and Internet security software company reported an estimated 1.5 billion attacks worldwide in 2012. Spyware, a form of malware that aims at stealing personal information, makes up a portion of those 1.5 billion attacks. An estimated 90% of home computers in the United States are infected with spyware, and 40% of security downtime costs in small and medium-sized companies are attributed to this harmful form of computer software. The most prevalent detection methods have their flaws and although these methods are effective, the growing volume of malware instances poses a serious threat to their effectiveness. An alternative is behavior-based detection. This method fills in the cracks left behind by other popular methods. By monitoring computer system activity it has the ability to detect entire classes of malware without the need to tailor an application and its algorithms to specific spyware instances. To better understand these methods, spyware, as a whole must be explained.

III. Spyware

Spyware is computer software that helps in the collection of information about an organization or person without their knowledge. It also has the ability to take control over a computer system without the users knowledge or consent. Some common types of spyware include system monitors, trojans, adware, keyloggers, and tracking cookies. Although the term spyware suggests activity monitoring, the functions of most spyware programs reach far beyond that. Data collection – including personal surfinghabits, user logins, and bank or credit card information – browser hijacking, and pop-up generating are common activities for spyware programs. Programs can also interfere with user control of a computer by installing additional software. Some spyware can change computer settings. This can result in slow Internet connection speeds, unauthorized changes in browser settings, or changes to software settings. Unlike common computer viruses which are intended to multiply and spread to many systems, spyware generally does not attempt to transmit or copy itself to other computers. The goal of spyware is not to cause damage; the goal is to gather information about the user and, at times, control web browsing. However, spyware can lead to poor system performance due to poorly written code and resource consumption.

IV. Signature vs. Behavior-Based Detection

Signature-based detection is the most popular form of malware detection. The most well known anti-malware programs – i.e. Kapersky, AVG, Norton – all use these methods. The weaknesses left behind by signature-based detection are addressed by behavior-based detection. Signature-based detection works by searching for known patterns of malicious data within executable code documents. The program running the search relies upon a dictionary of known malicious data patterns, and compares code contained in the users file system to the dictionary for detection. Spyware programmers try to stay one step ahead off detection software by writing polymorphic code - code that uses an engine to mutate the software while keeping algorithms in tact - and simple obfuscation techniques to hide its true signature. The biggest downfall in signature-based detection is the reliance the detection program has on its reference dictionary. These dictionaries hold signatures of known spyware instances and are used during the scanning phase of detection. These dic-

tionaries are used by detection programs and require frequent updates when new malicious code signatures become available. However, it is possible for a computer to be infected with a form of malware for which no signature is yet known. The possibility of unknown forms of malware lead to the pressing need for better detection methods. This led to the inception of behavior-based detection. Year after year the goal for antivirus companies has been to collect the most signatures. These collections not only slow down the computer they are installed on, but they require a large amount of space on the systems hard drive. They also rely heavily on the user to update their antivirus program. This means that even on the day of purchase, most security suites are outdated and ill equipped to protect the user against the newest malware.

V. Infection

In order for a spyware program to infect a computer system it must first gain access to the system. Spyware programs typically infect computer systems through installation. Downloading media from unofficial sources and browsing without discretion are often to blame. It is common for spyware programs to be deceptively piggybacked along side popular software and downloaded to the users machine. Spyware is frequently disguised as browser toolbars or plugins. These objects fall under the category of Browser Helper Objects (BHO). BHOs sit along side the web browser for easy access to browsing information and traffic. From this position they can interact with the operating system (OS) by easily implementing various interfaces offered by the OS and the browser to execute whatever tasks the spyware is intended to perform.

VI. Detection Methods

A. Signature-Based Detection

Signature-Based Detection is perhaps the most popular detection method. The most well known anti-malware companies i.e. Kapersky, AVG, Norton – all use these methods. Developed in the 1990s, signature-based detection relies on a database of code signatures for known spyware instances. A code signature can be anything from a known malicious function to a specific file property. Files on the machine are then cross–referenced against the database of code signatures to check for matches.The popularity of this method is due to the success that comes along with a strong and reliable signature database. One issue with this method is the inconvenience that comes from continuously having to update the signature database. If a user neglects to update the database that complements their software, they are put at a greater risk of malicious consequences as time goes on. Another issue is that of self-modifying code. Code that uses polymorphic and metamorphic algorithms – methods that use an engine to mutate the software while keeping algorithms in tact – to alter its code signature to avoid detection falls under this category. Spyware authors typically use this as a type of encryption, disguising the true meaning and functionality of the source code so that the code signature appears non-malicious. The last issue, and perhaps the greatest issue associated with this method, is the insurmountable threat of unseen spyware instances; instances for which no signature exists.

McAfee, one of the most popular antivirus and anti-spyware software vendors in the world, reported nearly one hundred thousand new malware samples every day in 2013; roughly 69 new samples every minute – as seen in Figure 1.

This issue played less of a part in the early years of signature-based detection, but now it is one of the biggest threats with the rapid increase of new malware instances.

B. Specification-Based Detection

Specification-based Detection uses a list of action specifications to determine actions that are allowed and those that are not allowed. By checking a program against this list, one is

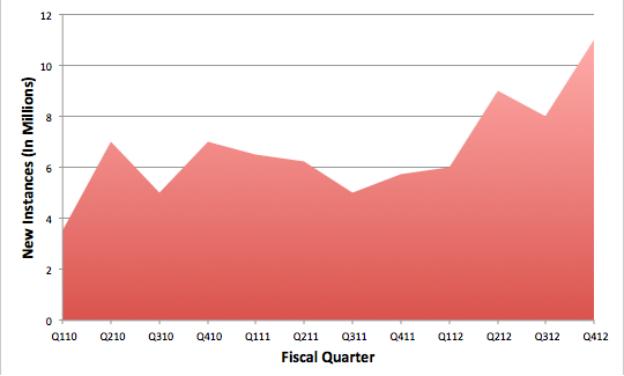

Fig. 1. General COM framework. COM objects are linked together to form an application.

able to label a specific application as malicious or not based on how it is supposed to behave. This detection method has two possibilities to perform this check. The first is a static analysis of the program in question. Here, the analysis gathers information about the program and investigates the source code or its binary representation for code sequences that could potentially be malicious. This information is then tested against the action specification list provided by the detection application to decide whether it is malicious or not. This method also suffers from the threat of encountering self-modifying code, similar to signature-based techniques. The second option is dynamic analysis. Here the detection application runs the questionable application in a controlled environment. This measure it taken to prevent any potential damage in the event that the program in question is indeed malicious. During this process the actions of the application are compared to the policy set by the detection application. If the actions performed are in violation of this policy, the application is marked as malicious. Since it bases detection on the actions of the application, this method tends to be more appealing than static analysis since it is not susceptible to self-modifying code.

C. Behavior-Based Detection

Recently, research has been conducted on the topic of behavior-based spyware detection which overcomes the shortcomings of other technologies that use signature-based techniques. This technique enables the detection of spyware based on its behavior. The benefit that this method provides is the ability to detect entire classes of malware without the need of specifically tailoring an application and its algorithms to specific spyware instances. This is the difference between specification-based detection and behavior-based detection. Specification-based systems and their algorithms must be focused and calibrated to specific forms of spyware AND activities, while behavior-based detection is more general and does not focus on an specific application. The advantage of behavior-based systems is that they monitor operating system interactions and function calls and link any suspicious activity the application calling it. This is made possible by the tightly knit integration between web browser plugins which will be detailed later in this paper and the operating system. Behavior-based detection is appealing because it eliminates the need for large signature databases. It is also resilient to polymorphic and metamorphic code because it judges maliciousness on the actions of an application. The benefits and possibilities of this detection method have inspired this paper, thus the technical aspects of behavior-based detection will be examined and explained. The Windows Operating System and Internet Explorer will be used as examples throughout this paper. This is due to their common association with spyware and their attractiveness to spyware authors.

VII. The Component Object Model

To fully understand the actions taken by spyware programs it is important to have a basic understanding of Microsofts Component Object Model, or COM. An application is typically a monolithic block of code that almost never changes throughout its lifetime from compilation to replacement. This tends to be less than ideal when maintaining an application after it has been put into production. Small adjustments that are made to specific parts of the

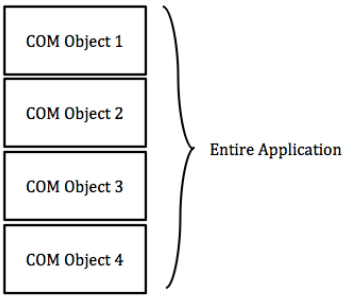

Fig. 2. General COM framework. COM objects are linked together to form an application.

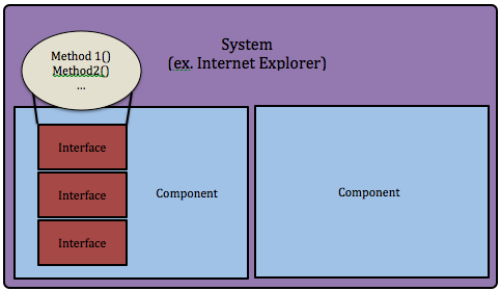

Fig. 3. Components are made up of multiple interfaces, and are linked together to form a larger application

application require the entire application to be updated and relaunched. Attempts to overcome this shortcoming have led to component based solutions that intended to break the large blocks of code into smaller components. This is what Microsoft had in mind when they developed the Component Object Model. These COM objects are essentially the building blocks for many Microsoft applications. A single component can be seen as a miniature application that has the ability to interact with the other COM objects. Multiple COM object interfaces are linked together to form a larger, more complex application.

The core benefit of this technology is that objects can be linked dynamically. Dynamic linking allows the app to evolve sensibly without having to redistribute the entire application if only some of the components change. Without the ability of dynamic linking there is virtually no benefit over standard monolithicblocks of code. Since the idea behind the component model is to have components interact with each other, there must be a standard to do so. This accomplished by having the components implement stable interfaces provided by Microsofts core libraries, or interfaces that are created by the user. It is important to note that a component providing the functionality to an interface is called a server, whereas a component using an interface is called a client (BHO). While an interface comprises a set of functions, a component is made up of a set of interfaces, leaving a system to be a set of components. Through multiple inheritance, a component is allowed to implement any number of interfaces.

Every COM object must inherit from the IUnknown interface. This interface provides a function that allows the COM object to query to any interface or set of functionalities that the author needs. These features contribute to the attractiveness of the Windows operating system for spyware authors. Microsofts COM framework provides spyware authors the tools that they need to access browser activity and user information. COM is used by many other important Microsoft technologies like ActiveX and Windows Runtime, and because COM is so fundamental to the Windows operating system, alteration to the COM framework to make it less susceptible to spyware is unrealistic. It is a common exploitation method for spyware authors and the open backdoor it provides makes a strong case for behavior-based detection methods.

VIII. Spyware & Browser Helper Objects

To further understand spyware it is necessary to know where it is typically installed how it interacts with the operating system to exploit user information. Sensitive user information is often accessed through web-interfaces and browser plugins. Browser plugins and graphical plugins in the form of toolbars fall under the category of Browser Helper Objects (BHOs). These objects are commonly linked to a web browser – in our example Internet Explorer – and are often used to gather user data. The

reason internet Explorer is so attractive for spyware authors is its tightly knit integration with the Windows operating system. This makes it easier for spyware to use system resources and gather and send information more easily. It is also attractive for its wide use among businesses. As of March 2014, Internet Explorer held a 48% marketshare among businesses and corporate america. Browser helper objects in Internet Explorer fall under the category of COM objects as mentioned above. The benefit that BHOs gain from following this framework is the ability to access any other COM interface that is implemented by the browser. The list of possible COM objects is far too vast for the scope of this paper, but the basic functionalities needed to gather and send user data are all available with a simple interface implementation.

To register a BHO as a COM object, the BHO must simply implement the IObjectWithSite interface. It is the functionality provided by this interface that gives any object that implements it the label of Browser Helper Object. With this implementation, a spyware instance will be recognized as a BHO by the browser and will be loaded at browser launch. In Internet Explorer and the Windows operating system, all browser helper objects that are registered as COM objects are accessed in the registry – a hierarchical database that stores information about the user and the system – via the key *\HKLM\SOFTWARE\Windows\CurrentVersion \Explorer\Browser Helper Objects*. For every BHO that gets loaded the browser passes a pointer to one of its own IUnknown interface that provides access to all other interfaces the browser implements. This is another exploitation available to spyware authors. There are a few interfaces that are commonly accessed by malicious browser helper objects. The main interface of interest to spyware objects as BHOs is the IWebBrowser2 interface. This interface provides a great deal of functionality to any BHO that implements it. IWebBrowser2 gives the ability to browse to a certain webpage, access the current URL, navigate backwards and forwards, as well as close the browser all together. BHO objects can be seen as plugins to Internet Explorer with the additional ability to do things like automatically displaying a certain webpages HTML code, viewing Portable Document Format (PDF) files embedded in the browser, or configuring a toolbar for a preferred web search engine. Since BHOs and Toolbars have complete access to the browsers interfaces and the applications memory space they can completely control the browsers behavior and be notified of user events. Because of this and the powerful possibilities it offers, BHOs and toolbars are frequently used as main components for spyware. However, the threat posed by browser helper objects that utilize the COM framework also brings about the possibility for behavior-based techniques.

IX. Putting it all Together

Spyware often takes the form of web browser plugins known as Browser Helper Objects. These objects have the ability to implement a wide array of interfaces made available by the browser itself through Microsofts Component Object Model. These interfaces provide the functionalities needed to gather user information, log keystrokes and search histories, and even control browser completely. While signature-based detection would examine these BHOs directly by looking for malicious code signatures, behavior-based techniques skip this process and focus on the actions taken by the BHOs upon execution. By listening very generally to the system, behavior-based detection applications fill in the holes left behind by signature-based detection. The BHOs are allowed to implement whatever interfaces they would like as COM objects, but as soon as they execute a malicious action they are flagged and caught by the detection application. It is for these reasons why behavior-based detection is so appealing as a new technology.

Spyware is becoming a substantial threat to modern computer usage both in terms of resource consumption and user privacy violations. As of 2013 an estimated 80% of computers were protected from viruses and malware with some sort of antivirus software. This number is up from 60% in 2003. This increase in awareness among computer users combined with the need for more comprehensive spyware prevention will lead to substantial growth in both behavior-based detection, as well as legislation prohibiting spyware. These are the two areas in which we expect to see the most change over the next 5 years.

X. Prototype

Further research comparing signature-based techniques and behavior-based techniques has led to a comparison of both detection methods to determine and compare success rates. Testing consisted of installing and running 20 known malicious spyware instances in the same environment as two spyware detection applications. The 20 samples consisted of 10 malicious toolbar instances, and 10 malicious application instances. The names of the instances can be seen below:

1) 4loot Toolbar
2) Allin1Convert Toolbar
3) ConnectSo Toolbar
4) Freshy Toolbar
5) Motitags Toolbar
6) SearchFlybar2 Toolbar
7) Swagbucks Toolbar
8) Trojan.ISTbar Toolbar
9) Zwinky Toolbar
10) Winload Toolbar
11) WebfettiInitialSetup1.0.1.1
12) BrowserUpDateForFree
13) Crypt2
14) softlicious.infoDLNEI1.7.2
15) Adware.SpyClean.N
16) Spyware.1893
17) Spyware.2535
18) Adware.Spywarestop.B
19) Adware.SpyClean.K
20) Trojan.316DAA73

The applications used were Threatfire Zero-Day Malware Protection to represent behavior-based techniques, and Spybot Search & Destroy to represent signature-based detection.

The testing environment was a Toshiba Satellite laptop running Windows XP and Internet Explorer was used as the web browser. Before any testing was done, the machine was formatted and a fresh copy of Windows XP to eliminate the possibility of interference from previously installed programs or malware. All spyware instances were downloaded to a USB flash drive on a separate machine and installed in the testing environment directly from the USB drive. This step was taken to ensure that only the desired spyware instance was installed, eliminating the possibility of piggy-backed software.

	Threatfire						
	Installation	Interaction	Spybot Detection	hompage alteration	Adware	Registry	Other
Toobar1			X	X	X		
Toolbar2				X	X		
Toolbar3		X				X	
Toolbar4		X	X	X	X		
Toolbar5		X		X			X
Toolbar6		X	X		X	X	
Toolbar7			X	X			X
Toolbar8		X	X		X	X	
Toolbar9		X		X	X		
Toolbar10		X	X	X		X	

Fig. 4. Table showing malicious activity for each toolbar spyware instance. Detection by both programs is also noted

	Threatfire						
	Installation	Interaction	Spybot Detection	hompage alteration	Adware	Registry	Other
Application1		X	X		X	X	X
Application2		X	X	X	X		
Application3			X		X		
Application4		X	X	X			
Application5		X	X	X			
Application6		X	X		X		
Application7		X		X		X	
Application8		X	X	X	X		
Application9		X				X	X
Application10		X	X	X		X	

Fig. 5. Table showing malicious activity for each application spyware instance. Detection by both programs is also noted

First, each spyware instance was installed and run while Threatfire was running and monitoring the system. After installation, normal browsing patterns were conducted for five minutes to try and provoke the instance to perform malicious activities. Our normal browsing patterns consisted of static Google searches and video streaming. Any detections made by the Threatfire application were noted and recorded.

Browsing activity was the halted and Threatfire monitoring was then stopped. The Spybot application was then run to try and detect the spyware instance that was just installed. After the scan was complete, malicious properties were noted and recorded for comparison. The next spyware instance was then loaded and installed onto the machine, and the process was started from the beginning. This was done until all 20 instances had been tested.

The results returned from the test were as follows:

Detections were generally triggered for a few issues like homepage alteration and adware, and occasional registry changes. Unfortunately, none of the instances showed behavior that was extremely malicious, such as browser hijacking. This is not surprising since these activities are less common and we had a sample size of only 20. We would hope that as the sample size grows, we would come across a few instances of this nature. You can see that Threatfire never caught programs as they were being installed, but only upon interaction. This was expected since the spyware instance does not run its code upon installation. Since Threatfire only monitors actions, it would not be able to detect an instance on installation.

That being said, Spybot did not detect anything upon installation either. This outcome is also expected for a couple of reasons. First, the signature-based detection application requires a scan to be initiated by the user. With this not being the case during installation, it would be impossible for the application to detect an instance during this process. Secondly, similar to the signature-based application, the code of the spyware instance is not fully on the system until the installation completes. That means that until the application is fully installed, there is no way for the detection application to discover the threat.

Biggest thing to notice is the success that the Behavior-based technique experienced. It only missed one sample, where the other technique missed two. But both detection rates are relatively good with Behavior Based having a slight edge. Overall success rates for both methods were as follows:

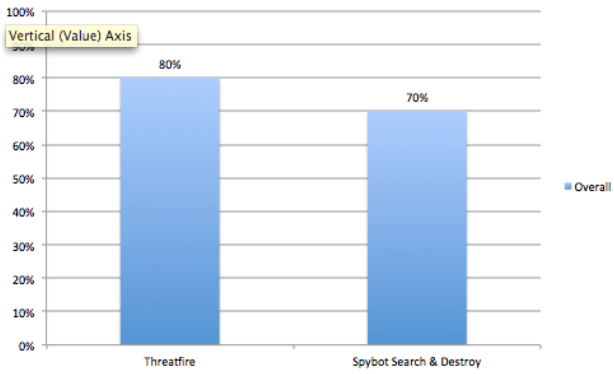

Fig. 6. The first table shows the overall detecton rate for each application.

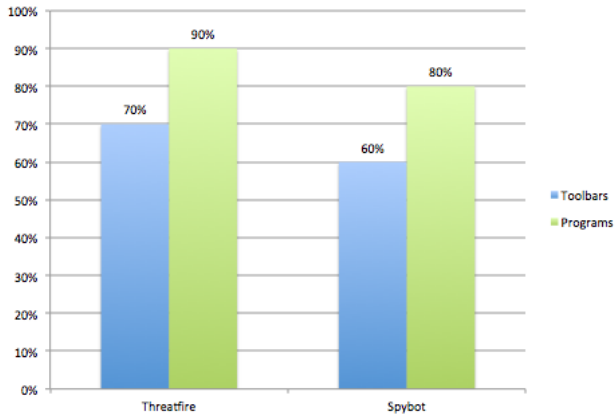

Fig. 7. This table breaks down each detection percentage for each application showing the percentage for each instance type.

XI. Future Trends

A. 0 – 3 Years

Over the next three years we expect to see behavior-based detection techniques trickle in to the mass market of antivirus tools. Some of the big names in antivirus software have already implemented their own behavior based monitoring. Companies like Avast!, AVG, and Failsafe already offer behavior monitoring right along side their successful and widely used signature-based tactics. This trend is likely to continue. Both detection methods address different aspects of malware detection. Signature-based detections inability to detect new forms of malware is specifically addressed by behavior-based techniques. For this reason, the thought of combining the two methods into one package is plausible.

The next three years could also bring the introduction of new legislation against spyware. Most legal action is taken against spyware distributors by the Federal Trade Commission. The fact that adware and spyware often cross back and forth between legal and illegal activities makes this a complex issue. Several spyware related bills were introduced

for debate in the 108th and 109th Congressional sessions from the year 2004 to the year 2006. These bills only made it to to various House and Senate committees before being defeated. One of the most promising anti spyware bills to make its way through Washington was the SPYBLOCK (Software Principles Yielding Better Levels of Consumer Knowledge) bill which outlawed the most common spyware and adware practices in detail. Its sections covered zombies, modem hijacking, denial of service attacks, endless popup advertisement loops, stealing personal information, disabling security, modifying browser settings, deceptive installation notices, keyloggers, file damage, and software bundling. Unfortunately this bill was defeated. A recent trend, however, seems to be that of state-by-state legislation. Many states such as California have implemented their own moderate anti-spyware laws to limit the abuse that is taking place with user information.

B. 3 – 5 Years

By the end of this five year period we expect to see full behavior-based integration with the vast majority of well known anti-virus and anti-malware applications. With the advancement of behavior-based technologies taking hold with some of the major players – such as AVG and Avast! – it is reasonable to expect the lesser known software companies to follow suit. In terms of legislation it can be expected that most states with have significant laws prohibiting certain spyware activities. It is likely that activities like key logging and browser hijacking will be forbidden and punishable by fines and other means of punishment.

Current spyware authors are unlikely to quit the spyware game until the costs and risks become greater than the potential benefits involved. Enabling effective enforcement at home and abroad of new and current laws will increase the time, money and manpower needed to circumvent them for a successful spyware campaign. The other side of the equation is reducing the profitability of spyware. User education is a key element, but the more people are taught to practice safe surfing techniques, the the less appealing producing spyware becomes. However, regardless of the amount of regulation that is put on spyware and its authors, there will always be a threat. Legislation cannot defeat spyware all together. Solid anti-spyware software can address what would be left behind. Combining signature and behavior-based techniques is a very viable option for combatting these threats.

XII. Coursework

The computer science curriculum and the education provided by Saint Johns was of great help when writing this paper. There were many courses that helped the ease of understanding of various topic areas. Some knowledge was hands-on and technical, but the vast majority was referencing what was learned in the classroom to better understand how things work conceptually. The three courses that were of the most help were Software Development, Computer Organization, and Ethical Issues in Computing.

A. Software Development

The fact that this class offered insight into the topic should not be a very big surprise. The research conducted was centered around malicious software, so having the knowledge of software development and the process involved was of great benefit. There were some instances where I was able to see the source files for some malware samples. This was extremely satisfying because of the thorough understanding of the Java programming language provided by the Software Development course. This made it possible to walk through the given malware sample step-by-step to see how it conducted itself, and fully understand what was taking place at any given line of code.

The theoretical implications were also strong. Reading about many malware and spyware instances and their actions always pro-

voked the question of how the actions in question might be carried out. Referencing software core libraries and the system integration offered by them improved understanding greatly.

B. Computer Organization

There was one specific research area where the coursework offered by the Computer Organization class was useful, and this area was stack traces. Referring to behavior-based detection techniques, a question that arose was how a behavior-based application links a behavior to the application calling it. This question stood for a long time until the term stack traces was mentioned. This reference brought the topics of this course into question and led to a better understanding of how an application could trace an action to its executer.

C. Ethical Issues in Computing

This course was referenced the most throughout the research process. The main issue addressed by this course was whether or not various topics in computer science are ethical. Spyware is software that intentionally takes information from the user, most often without their knowledge or consent. Because of this understanding, there were two primary sides to the ethics of spyware: installation and execution.

Installation was found to be a gray area of sorts. Since spyware can be directly downloaded, piggybacked with other software, and forcefully installed, there were many factors to consider. We figured that in the case of direct download when the user downloads a spyware instance alone without being deceived it was ethical since nothing was being forced upon the user and the infection was due to user negligence. Piggybacking was found to be unethical on all accounts since it is deceptively hidden along with the software that was intended to be downloaded. Here, however, the blame can be put on the software distributor that has chosen to carryout the piggybacking. This distributor could be the malware developer themselves, or some third-party. Lastly, forceful or drive-by installation installation that is carried out unwillfully upon visiting a malicious URL is unethical on all accounts.

Execution is rather black and white. We found that execution carried out without the users consent, that performs actions that are unknown to the user, is unethical. Alternatively, actions that the user is made aware of are ethical.

These guidelines bring software like Google Toolbar into question. This piece of software is offered by a very reputable company, and provides many great features to the users like bookmarking, sharing, and customization. However, the toolbar logs the browsing history made through the toolbar to help Google with directed advertisements. It is activities like these in which the ethicality is up to the individual since information is being taken, but it is to being used in a malicious manner.

XIII. Conclusion

Spyware is becoming a substantial threat to computer system resource consumption and user privacy. Modern prevention software predominantly uses signature-based detection methods that can be evaded by code encryption and are unable to detect previously unseen instances. Behavior-based detection works around this issue by listening for specific actions carried out by the spyware instance. A tightly knit integration with the Windows operating system makes internet explorer very susceptible to spyware. Utilizing COM objects and interfaces to gain access to the browser makes gathering information fairly simple for any spyware instance that marks itself as a browser helper object. Application camparisons were conducted in which 20 malicious spyware instances were installed an executed on a machine running both a signature-based detection application and a behavior-based detection application. Results showed that the behavior-based application was able to accurately detect

a greater percentage of the spyware instances than the signature-based detection. Moving forward we hope to see method integration among major players in the antivirus software industry. Combining signature-based detection with behavior-based detection offers promise in the ongoing fight against malicious software.

REFERENCES

[1] Dr. B. B. Meshram Ashwini Mujumdar, Gayatri Masiwal. Analysis of signature-based and behavior-based anti-malware approaches. *International Journal of Advanced Research in Computer Engineering and Technology*, 2(6):2037–2039, June 2013.

[2] Rossie Cortes. Yahoo introduces new spyware-detection feature. *Caribbean Business*, 32(25):42, 2004.

[3] Manuel Egele. *Behavior-based Spyware Detection Using Dynamic Taint Analysis*. VDM Verlag Dr. Mller Aktiengesellschaft & Co. KG, 2008.

[4] Andrew Garcia. Join the spyware fight. *eWeek*, 23(6):37 – 42, 2006.

[5] Steve Gibson. Spyware was inevitable. *Communications of the ACM*, 48(8):37 – 39, 2005.

[6] Engin Kirda Manuel Egele, Christopher Kruegel. Dynamic spyware analysis. 2006.

[7] Kirk P. Arnett Mark B. Schmidt. Spyware: A little knowledge is a wonderful thing. *Communications of the ACM*, 48(8):67–70, August 2005.

[8] Philip Okane, Sakir Sezer, Kieran McLaughlin, and Eul Gyu Im. Malware detection: program run length against detection rate. *IET Software*, 8(1):42 – 51, 2014.

[9] Anne M. Payton. A review of spyware campaigns and strategies to combat them. In *Proceedings of the 3rd Annual Conference on Information Security Curriculum Development*, InfoSecCD '06, pages 136–141, New York, NY, USA, 2006. ACM.

[10] Anne M Payton. A review of spyware campaigns and strategies to combat them. *InfoSecCD Conference*, pages 136–41, September 2006.

[11] Ashkan Sami, Babak Yadegari, Hossein Rahimi, Naser Peiravian, Sattar Hashemi, and Ali Hamze. Malware detection based on mining api calls. In *Proceedings of the 2010 ACM Symposium on Applied Computing*, SAC '10, pages 1020–1025, New York, NY, USA, 2010. ACM.

[12] Mark B. Schmidt and Kirk P. Arnett. Spyware: A little knowledge is a wonderful thing. *Communications of the ACM*, 48(8):67 – 70, 2005.

[13] Douglas Schweitzer. Detecting and removing spyware. *Certification Magazine*, 6(9):40, 2004.

[14] Sudhindra Shukla and Fiona Fui-Hoon Nah. Web browsing and spyware intrusion. *Communications of the ACM*, 48(8):85 – 90, 2005.

[15] Peter Szor. *The Art of Computer Virus Research and Defense*. Addison-Wesley, 1 edition, 2005.

[16] Zhang Xiaoni. What do consumers really know about spyware?. *Communications of the ACM*, 48(8):44 – 48, 2005.