Blekinge Institute of Technology
Doctoral Dissertation Series No. 2010:02
ISBN 978-91-7295-177-8
ISSN 1653-2090

2010-03-23

Privacy-Invasive Software

Martin Boldt

School of Computing
Blekinge Institute of Technology
Sweden

School of Computing
Publisher: Blekinge Institute of Technology
Printed by Printfabriken, Karlskrona, Sweden 2010
ISBN 978-91-7295-177-8

To Lena

Contact Information

Martin Boldt
School of Computing
Blekinge Institute of Technology
PO Box 520
SE-372 25 Ronneby
SWEDEN

E-mail: martin.boldt@bth.se
Web: http://www.bth.se/tek/mbo/

Abstract

As computers are increasingly more integrated into our daily lives we become more dependent on software. This situation is exploited by villainous actors on the Internet that distribute malicious software in search for fast financial gains at the expense of deceived computer users. As a result, computer users need more accurate and aiding mechanisms to assist them when separating legitimate software from its unwanted counterparts. However, such separations are complicated due to a grey zone of software that exists between legitimate and purely malicious software. The software in this grey zone is often vaguely labelled spyware. This work introduces both user-aiding mechanisms and an attempt to clarify the grey zone by introducing the concept of privacy-invasive software (PIS) as a category of software that ignores the users' right to be left alone. Such software is distributed with a specific intent (often of commercial nature), which negatively affects the users to various degree. PIS is therefore classified with respect to the degree of informed consent and the amount of negative consequences for the users.

To mitigate the effects from PIS, two novel mechanisms for safeguarding user consent during software installation are introduced; a collaborative software reputation system; and automated End User License Agreement (EULA) classification. In the software reputation system, users collaborate by sharing experiences of previously used software programs, allowing new users to rely on the collective experience when installing software. The EULA classification generalizes patterns from a set of both legitimate and questionable software EULAs, so that computer users can automatically classify previously unknown EULAs as belonging to legitimate software or not. Both techniques increase user awareness about software program behavior, which allow users to make more informed decisions concerning software installations, which arguably reduces the threat from PIS.

We present experimental results showing a set of data mining algorithms ability to perform automated EULA classification. In addition, we also present a prototype implementation of a software reputation system, together with simulation results of the large-scale use of the system.

Acknowledgements

First of all, I would like to express my gratitude to my supervisor, Dr. Bengt Carlsson, for both his guidance throughout this work and for always finding the time. I would also like to thank my examiner, Professor Paul Davidsson, for helping me form this thesis.

Furthermore, I want to thank my friends and colleagues, especially Dr. Niklas Lavesson for his assistance and great knowledge in data mining; Dr. Andreas Jacobsson for many interesting discussions about spyware and other matters; Anton Borg for continuing some of the research ideas introduced in this thesis; and last but not least my assistant supervisor Dr. Stefan Axelsson and DISL members for valuable feedback and interesting discussions.

I am forever grateful to my parents Ingegärd and Jerker for their endless and unconditional support and for always being the best of parents. Special thanks also to my brother Christian and my sister Elisabeth for many great memories, and for many still to come. Most importantly, I want to thank Lena for keeping up with me during this work, including the sometimes odd working hours.

Preface

This thesis consists of the seven publications listed below. My contributions for each of these publications are as follows. For publication one and two I was responsible for the experiment design, setup, execution, and data analysis/filtering. I was main author of publication two, three, six and seven. I was responsible for the design of the presented system in publication four, as well as for writing the paper. In addition to jointly coming up with the idea of the content in publication five together with Niklas Lavesson; I also motivated the problem within a security context, and evaluated the state-of-the-art commercial tool in this work.

Publication 1: A. Jacobsson, M. Boldt and B. Carlsson, "Privacy-Invasive Software in File-Sharing Tools", in *proceedings of the 18th IFIP World Computer Congress (WCC2004)*, 2004, Toulouse France.

Publication 2: M. Boldt, A. Jacobsson, and B. Carlsson, "Exploring Spyware Effects", in *proceedings of the 9th Nordic Workshop on Secure IT Systems (NordSec04)*, Helsinki Finland, 2004.

Publication 3: M. Boldt and B. Carlsson, "Analysing Countermeasures Against Privacy-Invasive Software", in *proceedings of the IEEE International Conference on Software Engineering Advances (ICSEA'06)*, Papeete French Polynesia 2006.

Publication 4: M. Boldt, B. Carlsson, T. Larsson and N. Lindén, "Preventing Privacy-Invasive Software using Online Reputations", in *Lecture Notes in Computer Science (LNCS)*, Volume 4721, 2007.

Publication 5: N. Lavesson, M. Boldt, P. Davidsson, A. Jacobsson, "Learning to Detect Spyware using End User License Agreements", in print: *Springer International Journal of Knowledge and Information Systems (KAIS)*, 2009.

Publication 6: M. Boldt, A. Borg and B.Carlsson, "On the Simulation of a Software Reputation System", in *proceedings of the 5th International Conference on Availability, Reliability and Security (ARES'10)*, Krakow Poland, 2010.

Publication 7: M. Boldt and B. Carlsson, *"Stopping Privacy-Invasive Software Using Reputation and Data Mining"*, journal manuscript, 2010.

Throughout this thesis the use of first person plural signifies the fact that several people in addition to the author of this thesis have made contributions to this work. All papers have been scrutinized by both colleagues and members of our research group, and they have also been peer-reviewed at the corresponding conferences and journals.

Finally, the following publications are associated with, but not included in this thesis.

J. Wieslander, M. Boldt and B. Carlsson, "Investigating Spyware on the Internet", in the *proceedings of the 7th Nordic Workshop on Secure IT Systems (NordSec03)*, Gjövik Norway, 2003.

M. Boldt and B. Carlsson, "Privacy-Invasive Software and Preventive Mechanisms", in *proceedings of the International Conference on Systems and Networks Communications (ICSNC'06)*, Papeete French Polynesia, 2006.

M. Boldt, B. Carlsson, R. Martinsson, "Software Vulnerability Assessment – Version Extraction and Verification", in *proceeding, International Conference on Systems and Networks Communications (ICSEA 2007)*, Cap Esterel France, 2007.

M. Boldt, P. Davidsson, A. Jacobsson and N. Lavesson, "Automated Spyware Detection Using End User License Agreements", in *proceedings of the 2nd International Conference on Information Security and Assurance*, Busan Korea, 2008.

N. Lavesson, P. Davidsson, M. Boldt and A. Jacobsson, "Spyware Prevention by Classifying End User License Agreements", in *Studies in Computational Intelligence*, Volume 134, Springer, 2008.

J. Olsson and M. Boldt, "Computer Forensic Timeline Visualization Tool", in *Journal of Digital Investigation*, Elsevier, 2009.

Table of Contents

List of Figures

List of Tables

CHAPTER

1

Introduction

As computers are being increasingly more integrated into our daily lives, we entrust them with sensitive information, such as online banking transactions. If this data was to escape our control, negative effects to both our *privacy* and our economic situation could be impaired. Privacy is a central concept in this work, and it may be described as the ability for individuals to control how personal data about themselves are stored and disseminated by other parties [79]. Another important aspect of privacy is the individuals' right to keep their lives and personal affairs out of the public space. The amount of personal data that affect our privacy will continue to grow as larger parts of our lives are represented in a digital setting, including for instance e-correspondence and e-commerce transactions.

In parallel with this development, software known as *spyware* has emerged. The existence of such software is based on the fact that information has value. Spyware benefits from the increasing personal use of computers by stealing privacy-sensitive information, which then is sold to third parties. Conceptually, these programs exist in-between legitimate software and malicious software (such as computer viruses). However, there is no consensus on a precise definition for spyware since its exact borders have not yet been revealed. The lack of such a standard definition results in that spyware countermeasures do not offer users an accurate and efficient protection. Therefore, the users' computers are infested with spyware, which among many things, deteriorate the performance and stability of their computers, and ultimately present a threat to their privacy.

In this work, we contribute to the understanding of spyware by providing a classification of various types of *privacy-invasive software* (PIS). This classification does not only include spyware, but also both legitimate and malicious software. As there is no consensus regarding where to put the separating line between legitimate software and spyware, nor between spyware and malicious software, it is important to address both of these cases in the classification of PIS. After having classified PIS, we further explore how PIS programs affect the users' computer systems and privacy. To help mitigate the effects from PIS we propose the use of collaborative reputation systems for preventing the infections and distribution of PIS. We have developed a proof-of-concept system that allows users to share their opinions about software they use. In this system, the users are asked to continuously grade software that they frequently use. In return, the users are presented with the opinion of all previous users with regard to software that enters into their computer. In addition to this system we also propose the use of automated classification of End User License Agreements to notify users about PIS when installing software.

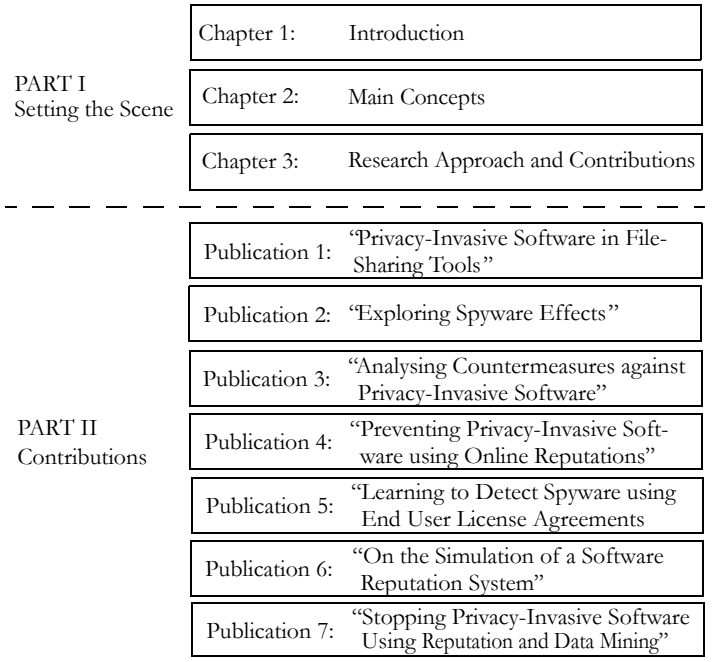

Figure 1.1 Thesis outline.

1.1 Thesis Outline

As presented in Figure 1.1, this thesis consists of two parts, where the purpose of part one is to set the scene for the thesis, using the first three chapters. In the next section we provide a background, and in Chapter 2 we present the related work and also provide an extended introduction to main concepts. In Chapter 3 the research approach, motivation, and questions are presented together with the thesis contributions.

The second part of the thesis includes six published papers and one manuscript. The first two papers focus on spyware and its consequences to both the infested computer and the user privacy. In the third publication we evaluate the accuracy of spyware countermeasures. In the fourth publication we introduce privacy-invasive software (PIS) and also describe the idea of using software reputation systems as countermeasures. Then, in the fifth publication we introduce data mining techniques to classify End User License Agreements (EULA). The sixth publication presents results from simulations of a software reputation system. Finally, the seventh publication is a journal manuscript that summarizes our findings related to PIS and the related countermeasures.

1.2 Background

In the mid-1990s the development of the Internet increased rapidly due to the interest from the general public. One important factor behind this accelerating increase was the 1993 release of the first graphical browser, called Mosaic [1]. This marked the birth of the graphically visible part of the Internet known as the World Wide Web (WWW). Commercial interests became well aware of the potential offered by the WWW in terms of electronic commerce, and soon companies selling goods over the Internet emerged with pioneers such as book dealer Amazon.com and CD retailer CDNOW.com, which were both founded in 1994 [54].

During the following years, personal computers and broadband connections to the Internet became more commonplace. Also, the increasing use of the Internet resulted in that e-commerce transactions involved considerable amounts of money [11]. As the competition over customers intensified, some e-commerce companies turned to questionable methods in their battle to entice customers

into completing transactions with them [10, 63]. This opened ways for illegitimate actors to gain revenues by stretching the limits, with for example, methods for collecting personal information or by propagating unsolicited commercial advertisements. Soon, the providing of such services became a business in itself; allowing less scrupulous marketers to buy such services that allowed them to get an advantage over their competitors, e.g. by using advertisements based on unsolicited commercial messages (also known as Spam) [39].

Originally, such questionable techniques were not as destructive to the computer system as the more traditional malicious techniques used in computer viruses or Trojan horses. Compared to these malicious techniques the new ones differed in two important ways. First, they were not necessarily illegal, and secondly, their main goal was gaining revenue instead of creating publicity for the creator by reaping digital havoc.

Behind this development were advertisers that understood the fact that Internet was a "merchant's utopia", offering a huge potential in global advertising coverage at a relatively low cost. By using the Internet as a global bulletin board, e-commerce companies could market their products through advertising agencies, which delivered online ads to the masses. In 2004, online advertisements represented $2 billions which in 2005 increased to $6 billion-a-year [45, 82]. The larger online advertising companies report annual revenues in excess of $50 million each [14]. In 2008, the online advertisement revenues had reached $23 billion; outperforming traditional TV commercials in some countries [37]. In the beginning of this development the advertising companies distributed their ads in a broadcast-like manner, i.e. they were not streamlined towards individual user interests. Some of these ads were served directly on Web sites as banner ads, but dedicated programs, called *adware*, soon emerged. Adware were used to display ads through pop-up windows without depending on any Internet access or Web pages.

In search for more effective advertising strategies, these companies began using the potential in ads that were targeted towards users' interests around the millennium shift. Once targeted online ads started to appear the development took an unfortunate turn. Now, some advertisers developed software that became known as *spyware*, collecting users' personal interests, e.g. through their browsing habits. Over the coming years spyware would evolve into a significant new threat to Internet-connected computers, bringing along

reduced system performance and less security. The information gathered by spyware was used for constructing user profiles, including personal interests, detailing what users could be persuaded to buy.

The introduction of online advertisements also opened a new way to fund software development by having the software display advertisements to its users. By doing so, the software developers could offer their software "free of charge" since they were paid by the advertising agency. Unfortunately, many users did not understand the difference between free of charge and a free gift. A free gift is given without any expectations of future compensation, while something provided free of charge may expect something in return. As an example a dental examination that is provided free of charge at a dentist school is not a free gift. The school expects their students to gain training value and as a consequence the customer suffers increased risks. As software were combined with spyware this became a problem for the computer users. When downloading software described as free of charge, the users had no reason to suspect that it would report on, for example, their Internet usage so that advertisements could be targeted towards their interests.

Some users probably would have accepted to communicate their browsing habits because of the feedback, e.g. offers relevant to their interests. However, the fundamental problem was that users were not properly informed about neither the occurrence nor the extent of such monitoring, and hence were not given a chance to decide on whether to participate or not. As advertisements became targeted, the borders between adware and spyware started to dissolve due to the combination of both behaviours; resulting in programs that both monitored users and delivered targeted ads. The fierce competition soon drove advertisers to further enhance the ways used for serving their ads, e.g. replacing user-requested content with sponsored messages before showing it on the screen.

As the quest for faster financial gains intensified, several competing advertisers turned to use even more illegitimate methods in an attempt to stay ahead of competition [9]; accelerating the situation and pushing the grey area of the Internet closer to the dark side [32]. During this development users experienced infections from unsolicited software that crashed their computers by accident, uninvitedly changed application settings, harvested personal information, and deteriorated their computer-experience through Spam and pop-up ads [49]. Over time these problems lead to the introduction

of countermeasures in the form of anti-spyware tools. These tools supported users in cleaning their computers from spyware, adware, and any other types of shady software located in that same grey area. As these tools were designed in the same way as anti-malware tools, such as anti-virus programs, they most often identified spyware that was already known, leaving unknown spyware undetected. To further aggravate the situation, illegitimate companies distributed fake anti-spyware tools in their search for a larger piece of the online advertising market. These fake tools claimed to remove spyware while instead installing their own share of adware and spyware on unwitting users' computers; sometimes even accompanied by the functionality to remove adware and spyware from competing vendors.

Another area that could be of interest for spyware vendors is the so called media centers that include the same functionality as conventional televisions, DVD-players, and stereo equipment, but combined with an Internet connected computer. These media centers are thought to reach vast consumer impact [38, 48]. In this setting, spyware could monitor and survey for instance which television channels are being used, when/why users switch channel, and what movies the users purchase and watch. This is information that is highly attractive for any advertising or media-oriented corporation to obtain. This presents us with a probable scenario where spyware is tailored towards these new platforms; the technology needed is to a large extent the same as is used in spyware today. Another example that most likely will attract spyware developer interest is the increasing amount of mobile devices that include GPS functionality. During the writing of the introduction to this thesis Google were awarded with a patent on using location information in advertisements [43], which is expected to have far-reaching effects on both Web-based and mobile advertising [73]. Gaining access to geographical position data allow advertisers to provide for example GPS-guided ads and coupons [67].

The enhanced mobile platforms that are in use today form an interesting breeding ground for augmented reality[1] applications [21]. In fact there are already free-of-charge products available for cellular phones and other mobile devices, one such example is Layar [44].

1. While *virtual reality* tries to replace the real world, *augmented reality* instead tries to combine the two; an example is applications that allow the user to point a phone's camera at various objects, which the phone automatically identifies and show additional information about.

Marketing in this setting allow advertising companies get access to users' personal geographical data so they can be served geographically dependant ads and coupons. When such geographic data is being harvested and correlated with already accumulated personal information, another privacy barrier is crossed. This raises the stake for the users, and stresses the need of the mechanisms that allow users to make informed decisions regarding software.

Today spyware programs are being added to the setting in what seems to be a never-ending stream, although the increase has levelled out over the last years. However, there still does not exist any consensus on a common spyware definition or classification, which negatively affects the accuracy of anti-spyware tools, further rendering in that spyware programs are being undetected on users' computers [31]. Developers of anti-spyware programs officially state that the fight against spyware is more complicated than the fight against viruses, Trojan horses, and worms [77]. We believe the first step for turning this development in favour for both users and anti-spyware vendors, is to create a standard classification of spyware. Once such a classification exists, anti-spyware vendors can make a more clear separation between legitimate and illegitimate software, which could result in more accurate countermeasures.

CHAPTER

2

Main Concepts

The concepts that are covered in this section form a basis for the remainder of the work and discussions in this thesis; and the purpose of this section is to declare our understanding and to motivate our use of the concepts.

2.1 Privacy

The first definition of privacy was presented by Warren and Brandeis in their work "The Right to Privacy" in 1890 [75]. They defined privacy as "the right to be let alone". Today, as we are part of complex societies, the privacy debate does not argue for the individual's right to physically isolate himself by living alone in the woods as a recluse, which could have been one motivation over a century ago. Instead the community presumes that we all must share some personal information for our society to function properly, e.g. in terms of health care services and law enforcement. Discussions in the privacy community therefore focus on how, and to what extent we should share our personal information in a privacy-respecting manner. Unfortunately, it is not possible to properly define privacy in a single sentence in this complex situation, or as Simson Garfinkel so concisely put it [26]:

> *"The problem with the word privacy is that it falls short of conveying the really big picture. Privacy isn't just about hiding things. It's about self-possession, autonomy, and integrity. As we move into the computer-*

ized world of the twenty-first century, privacy will be one of our most important civil rights."

However, for the clarity of the remaining part of this work we make an approach to present our interpretation and usage of privacy in this thesis. In the end, we share the general understanding of privacy with the work presented by Simone Fischer-Hübner [36] who divides the concept of privacy into the following three areas:

- *territorial privacy* focusing on the protection of the public area surrounding a person, such as the workplace or the public space
- *privacy of the person*, which protects the individual from undue interference (constituting for example, physical searches and drug tests)
- *informational privacy* concerning how personal information (information related to an identifiable person) is being gathered, stored, processed, and further disseminated.

Many aspects of these privacy concerns need attention but this thesis focuses on the computer science perspective. The problems analysed and discussed in this work are mostly related to the last two areas above, i.e. protecting the user from undue interference, and safeguarding personal information. Thus, our view of privacy does not only focus on the communication of personal information, but also include undue interference that affects the users' computer experience.

2.2 Malware

Malware is a concatenation of *malicious* and *software*. The concept of malware captures any software that is designed or distributed with malicious intent towards users. The distribution of malware has intensified over the last decade as a result of the widespread use of the Internet. An additional contributing factor is the mix between data and executable code in commonly used systems today. In these systems, executable code has found its way into otherwise traditionally pure data forms, e.g. Word documents and Web sites. The risk of malware infection exists for all situations where executable code is being incorporated. Throughout this thesis we use the following definition of malware [66, 71]:

> *"Malware is a set of instructions that run on your computer and make your system do something that an attacker wants it to do."*

Spyware is often regarded as a type of malware, since they (in accordance with the malware definition) executes actions that are defined by the developer. However, there are differences between spyware and malware, which we further explain when defining spyware in the next section. To further enlighten the reader, we include three definitions of malware types that are often being mixed-up in for example media coverage. We start with the *computer virus* which probably is the most publicly recognized malware type [66]:

> *"A virus is a self-replicating piece of code that attaches itself to other programs and usually requires human interaction to propagate."*

The second one is the *worm*, also publicly known through its global epidemics [71]. Although it is closely related to, and often mixed-up with, the computer virus, there exist some differences as shown in the definition [66]:

> *"A worm is a self-replicating piece of code that spreads via networks and usually doesn't require human interaction to propagate."*

The third malware type is the *Trojan horse,* which shares some similarities with spyware as it deceives users by promising one thing but also delivers something different according to their operator's desires [66]:

> *"A trojan horse is a program that appears to have some useful or benign purpose, but really masks some hidden malicious functionality."*

One common misconception is that viruses or worms must include a payload that carry out some malicious behaviour. However, this is not the case since these threats are categorized by their distribution mechanisms, and not by their actions. An interesting example are the so called "white" or "ethical" worms that replicate instantly fast between computers, and patch the hosts against security vulnerabilities, i.e. they are not set to spread destruction on the hosts they infect but instead help in protecting against future threats. One could wonder if it is possible to "fight fire with fire without getting burned" [66]. Most security experts would agree in that these "white" worms are not ethical but instead illegal, as they affect computer systems without the owners consent. Such an ethical worm could harm a system if it was to include a programming bug

that gave it another behaviour than intended, i.e. similar to what happened with the Morris worm [20]. Since various malware definitions do not say anything about the purpose of the attacker, they can not easily be related to spyware as these programs are classified according to their actions instead of their distribution mechanisms.

2.3 Adware

Adware is a concatenation of *advertising* and *software*, i.e. programs set to deliver ads by advertising agencies and showing them on the computer users' screen. Throughout this thesis we use the following definition of adware [2, 39].

> *"Any program that causes advertising content to be displayed."*

2.4 Spyware

In early 2000, Steve Gibson formulated an early description of spyware after realizing that software, which stole his personal information had been installed on his computer [27]. His definition reads as follows:

> *"Spyware is any software which employs a user's Internet connection in the background (the so-called 'backchannel') without their knowledge or explicit permission."*

This definition was valid in the beginning of the spyware evolution. However, as the spyware concept evolved over the years it attracted new kinds of behaviours. As these behaviours grew both in number and in diversity, the term spyware became hollowed out. This evolution resulted in that a great number of synonyms sprang up, e.g. thiefware, evilware, scumware, trackware, and badware. We believe that the lack of a single standard definition of spyware depends on the diversity in all these different views on what really should be included, or as Aaron Weiss put it [78]:

> *"What the old-school intruders have going for them is that they are relatively straightforward to define. Spyware, in its broadest sense, is harder to pin down. Yet many feel, as the late Supreme Court Justice Potter Stewart once said, 'I know it when I see it.'."*

Despite this vague comprehension of the essence in spyware, all descriptions include two central aspects. The degree of associated user consent, and the level of negative impact they impair on the user and their computer system. These are further discussed in Section 2.6 and Section 2.8 respectively. Because of the limited understanding in the spyware concept, recent attempts to define it have been forced into compromises. The Anti-Spyware Coalition (ASC) which is constituted by public interest groups, trade associations, and anti-spyware companies, have come to the conclusion that the term spyware should be used at two different abstraction levels [2]. At the low level they use the following, which is similar to Steve Gibson's original definition:

> *"In its narrow sense, Spyware is a term for tracking software deployed without adequate notice, consent, or control for the user."*

However, since this definition does not capture all the different types of spyware available they also provide a wider definition, which is more abstract in its appearance [2]:

> *"In its broader sense, spyware is used as a synonym for what the ASC calls 'Spyware (and Other Potentially Unwanted Technologies)'. Technologies deployed without appropriate user consent and/or implemented in ways that impair user control over:*
> *1) Material changes that affect their user experience, privacy, or system security;*
> *2) Use of their system resources, including what programs are installed on their computers; and/or*
> *3) Collection, use, and distribution of their personal or other sensitive information."*

Difficulties in defining spyware, forced the ASC to define what they call *Spyware (and Other Potentially Unwanted Technologies)* instead. In this term they include any software that does not have the users' appropriate consent for running on their computers. Another group that has tried to define spyware is StopBadware.org, which consists of actors such as Harvard Law School, Oxford University, Google, Lenovo, and Sun Microsystems [68]. Their result is that they do not use the term spyware at all, but instead introduce the term *badware*. Their definition span over seven pages, but the essence looks as follows [69]:

> *"An application is badware in one of two cases:*
> *1) If the application acts deceptively or irreversibly.*

> *2) If the application engages in potentially objectionable behaviour without: first, prominently disclosing to the user that it will engage in such behaviour, in clear and non-technical language, and then obtaining the user's affirmative consent to that aspect of the application."*

Both definitions from ASC and StopBadware.org show the difficulty with defining spyware. Throughout this thesis we regard the term spyware at two different abstraction levels. On the lower level it can be defined according to Steve Gibson's original definition. However, in its broader and in a more abstract sense the term spyware is hard to properly define, as concluded above. Throughout the rest of this chapter we presume this more abstract use of the term spyware, unless otherwise is stated. We also use the terms *illegitimate* and *questionable* software as synonyms to spyware.

One of the contributions of this thesis is our classification of various types of spyware under the term *privacy-invasive software* (PIS), which is introduced in Chapter 3. This classification was developed as a way to bring structure into the fuzzy spyware concept. However, as the PIS classification did not exist when we wrote the first two included publications we therefore use the term *ad-/spyware* in Chapter 4 and 5 instead of PIS.

2.5 Informed Consent

The degree of *informed consent* that is associated with software is an important and central part of spyware. Informed consent is a legal term which details that a person has understood and accepted both the facts and implications that is connected to an action. In this thesis we use the term when observing to what degree computer users comprehend that new software is installed and how it impact their computer-experience. We start by defining informed consent, before moving on to describe the relation between spyware and informed consent.

Throughout this thesis we use the same definition of *informed consent* as was originally defined by Friedman et al. [22]. This definition divides the term into the following two parts:

- *Informed*, i.e. that the user has been adequately briefed. The term informed is then further divided into *disclosure* and *comprehension*. Disclosure refers to that accurate information about both positive and negative feedback should be disclosed, without any

unnecessary technical details. Comprehension targets that the disclosed information is accurately interpreted.

- *Consent*, i.e. that both positive and negative implications are transparent and approved by the user. The term consent is then broken down into *voluntariness*, *competence*, *agreement*, and *minimal distraction*. Voluntariness refers to that the individual has the possibility to decline an action if wanted, i.e. no coercion is allowed. The term competence concerns that the individual possess both the mental, emotional, and physical capabilities that are needed to give an informed consent. Agreement means that an individual should be given a clear and ongoing opportunity to accept or reject further participation. Finally, minimal distraction declare that individuals should not be diverted from their primary task through an overwhelming amount of interruptions that seek to "inform the user" or to "seek consent", i.e. to utilize user interaction sparsely [24].

For a user to be able to give an informed consent, e.g. with respect to allowing software to enter the system, it is important that the implications of the software is fully transparent towards the user. Today, the main method used by software vendors to inform users of their software is not transparent as it was designed to primarily fulfill legal purposes. End-User License Agreements (EULA) are widely used today and they form a contract between the producer and the user of a certain software. Most often users are forced to affirm that they have read, understood and accepted the EULA content before being able to install a specific software. Questionable software vendors use the EULA to escape liability from their software actions, by including juridical escape routes inside the EULA content [70].

2.6 Spyware and Informed Consent

As touched upon earlier, installing software that are funded by included spyware components allow for the vendor to distribute their software free of charge. However, the inclusion of such components may also result in a mismatch between the software behaviour that users assume, and the actual behaviour they realize. Such divergences have formed a skeptical user-base that disapproves of *any* software that e.g. monitors user behaviour. As a consequence, such users also label legitimate software as spyware, even if their

behaviour is clearly stated in the corresponding EULA without the use of any deceptive techniques.

Many computer users today are not capable of reading through EULAs, as they are written in a formal and lengthy manner [31, 70]. User license agreements that include well over 6000 words (compared to, e.g. the US Constitution that includes 4616 words) is not unusual [30]. Prior research shows that users need skills that correspond to a degree in contract law to understand the full EULA content [8]. This is used by questionable software vendors as a legal lifeline when they are challenged to explain their practices in court, using it as an escape route from liability.

Since the majority of users either do not have the prerequisite knowledge, or the time, to base an opinion on EULA content prior to installing software, they just accept it without reading it, i.e. the consent is not based on an *informed* decision. In the absence of user informed consent, software that does not comply with the user's security preferences (e.g. in terms of behaviour or stability) is allowed to enter their system. Since users lack the aiding mechanisms inside the operating system to distinguish illegitimate software from legitimate, they get their computers infested with spyware.

Today, legitimate software vendors (that without any deceptive practices) state in the EULA that their software displays advertisement pop-ups still run the risk of being labelled as spyware by the users, since they rarely read through the associated EULAs [8]. Hence, the users can not deduce the pop-up ads on the computer screen with the approval of a software installation some time ago. So, once users think their computer-experience has been subverted by spyware, they become overly protective which further adds on this skepticism. We believe this to be very unfortunate since behavioural monitoring is both useful and an effective info-gathering measure to base tailored services towards users' individual needs [12, 56]. It is not the technology as such that is the main problem, but rather the uninformed manner in which it is introduced toward the users. Legitimate software vendors need standardized mechanisms inside the operating system to inform potential users in how their software impacts the user's computer system.

If the technology was provided in a true overt manner towards the users it could equally well provide most beneficial services. Because of the personalization of these services they would also increase

user benefits compared to non user-tailored services. Therefore, it is important for both software vendors and for users to safeguard users' right to make informed decisions on whether they want software to enter their system or not. In the end, we believe that an acceptable software behaviour is context-dependent, i.e. what one user regards as acceptable is regarded as unacceptable by others, and as a result only the user himself can reach such decisions [31]. This is further discussed in Section 3.3 as one of the contributions in this thesis. In the end we believe that user consent will become an increasingly more important aspect in computer security as computers are further introduced into people's daily lives.

2.7 Spyware Distribution

Distribution of spyware differs vastly from the spreading of malware types such as viruses and worms. As by definition viruses and worms are distributed using self-propagation mechanisms, which spyware does not include.

Instead, most spyware distribution is ironically being carried out by the users themselves. Of course the users are not being aware that they install spyware because of a number of deceptive measures used by spyware vendors. One commonly used strategy is to *bundle* (piggyback) spyware with other software, which users are enticed to download and install. When users find useful software being provided free of charge they download them without questioning or being aware of the bundled components enclosed. Although the associated EULA often contains information about the bundled spyware and its implications, users do not read them because of their length and formal language. So, spyware vendors basically use software that attracts users as bait for distributing their own programs as bundles, e.g. together with file-sharing tools, games, or screen-saver programs.

Another spyware distribution mechanism relies on the exploitation of security vulnerabilities in the users' computer system. Microsoft's Web browser, Internet Explorer, has often been used for such purposes because of its unfortunate history of security flaws and its dominating position. Utilizing such vulnerabilities inside software on the user's computer allows attackers to run any programs of their choice on the user's system. Such attacks on Web browsers often start when the user visits, or is fooled to visit, a Web site con-

trolled by the attacker. Next, the Web server sends a small program that exploits the security vulnerability in the user's Web browser. Once the attacker has gained this foothold, it is possible to deploy and start any software, for instance sponsored spyware programs. Because the users are kept totally out of this scenario without any choice for themselves, these installations go under the name *drive-by downloads*. For clarity, it should be added that spyware that rely on software vulnerabilities as a distribution mechanism are closely related to malware. It might even be the case that these programs should not be called spyware, but instead malware.

The third method used by spyware vendors is to distribute their software using tricks that deceive the user into manipulating security features that are designed to protect the user's computer from undesired installations. Modern Web browsers for example do not allow software to be directly installed from remote Web sites unless the user initiates the process by clicking on a link. With the use of deceptive tricks, spyware vendors manipulate users into unknowingly clicking on such links [47]. One example is that pop-up ads could mimic the appearance of a standard window dialog box which includes some attractive message, i.e. "Do you want to remove a new spyware threat that has been detected on your computer?". This dialog box could then include two links that are disguised as buttons, reading "Yes" and "No", and despite which button the user press the drive-by download is started.

2.8 Spyware Implications

As we have seen, many spyware programs are distributed by being bundled together with attractive programs. When users install such programs the bundled spyware follows, and with it, system implications. As mentioned previously, spyware exists in a grey area between legitimate software and traditional malware. One of the distinctions between the two software categories relate to their implications for systems. Spyware does not result in the same direct destruction as with traditional forms of malware. Instead users experience a gradual performance, security, and usability degradation of their computer system. These system effects could be structured as follows [3, 63, 65]:

- *Security implications:* As with any software installation, spyware introduces system vulnerabilities when deployed on computer systems. However, the fundamental difference between general

software installation and spyware, is the undisclosed fashion used by the latter. This covertness renders it virtually impossible for system owners to guarantee the software quality of their computer system. Poor software quality conveys an escalated risk of system vulnerabilities being exploited by remote malicious actors. If such a vulnerability was found and exploited inside one of the leading spyware programs, it could result in millions of computers being controlled by attackers because of the widespread use of these programs. In 2004, poorly written adware programs allowed remote actors to replace any files on users systems because of a poorly designed update function [57]. Fortunately, this vulnerability was first identified by an honest individual that made sure that the adware developer corrected the problem before making a public announcement about the vulnerability.

- *Privacy implications:* Spyware covertly monitors, communicates, and refines personal information, which makes it privacy-invasive. In addition, such programs also display ads and commercial offers in an aggressive, invasive, and many times undesirable manner. Such software behaviour negatively affects both the privacy and computer-experience of users [78, 82]. These privacy-invasions will probably result in greater implications for the users, as computers are being increasingly used in our daily lives, e.g. when shopping or online banking.
- *Computer resource consumption*: As spyware is installed on users' computer systems in an uninformed way, the memory, storage, and CPU resources are being utilized without the users' permission. Combined with the fact that users commonly have several instances of spyware on their systems makes the cumulative effect on computer capacity evident. Another threat to the local computation capacity comes from spyware that "borrow" the storage and computation resources from users' computers which it has infected. This combined storage and computational power were then combined into a distributed super computer, which could be rented by the highest bidder. Again, unwitting users (after some time) found their computers being covertly used in projects that were not compatible with their opinions and ethics [15].
- *Bandwidth consumption*: Along the same line of reasoning as above, the users' network capacity is being negatively affected by the continuous transmission of ads and personal information. Some users might even be even more upset, if these highly irritating and undesired behaviours use resources that instead

should be used for really important tasks. Bandwidth over consumption becomes even more significant when ads are being further enhanced using moving pictures and 3D graphics.

- *System usability reduction*: The existence of spyware on computer systems negatively impact a user's computer-experience [31]. Since spyware is installed in a covert manner users can not deduce the cause of strange system behaviours they experience. This makes it hard to identify what is inducing for instance the flow of pop-up ads, irreversible changes in application settings, installation of unrequested and unremovable software, or degradation of system performance and stability. In addition to this, underaged users could be exposed to offending material such as ads promoting adult material. These implications further result in that users are interrupted in their daily work, negatively influencing their general computer-experience.

As the aggregated amount of these implications became too overwhelming for the users to bear, a new group of software labelled *spyware countermeasures* emerged. These tools helped users to remove spyware from their systems.

2.9 Spyware Countermeasures

Today, *spyware countermeasures* are being implemented using the same techniques as traditional anti-malware tools use, e.g. anti-virus programs. However, an important difference between malware and spyware is that the former is well defined, while there is a lack of both knowledge and definition of the latter. Without a clear understanding of what kinds of programs that should be removed, countermeasure vendors both miss some spyware and wrongly remove legitimate software. The key problem is that malware include prohibited behaviour, such as virus and worm propagation mechanisms, while spyware does not. Anti-malware tools can therefore in an easier manner separate malware from legitimate software, by focusing on malware's illegal behaviours.

Spyware, on the other hand, often does not include prohibited behaviour, but instead, compared with malware, rather innocent behaviours, e.g. displaying messages on the screen, monitoring of the Web address field in browsers, or making non-critical configuration changes to programs, such as altering the default Web page. Unfortunately enough for anti-spyware vendors, spyware share

these behaviours with a vast number of legitimate software in general. Anti-spyware vendors therefore face a problem when trying to distinguish spyware from legitimate software based on software behaviour [76]. The anti-spyware vendors' removal strategies therefore need to be placed on a sliding scale, between two extremes. Either they prioritize the safeguarding of legitimate software, or they focus on removing every single spyware in existence. Unfortunately for the users, it is neither possible to remove every single spyware, because this would include many legitimate programs as well, nor to safeguard all legitimate software since this leaves most spyware untouched. Today, anti-spyware vendors have great difficulties in choosing where on this sliding scale they want to be, as none of these alternatives are very effective. Therefore the chosen strategy needs to be a compromise between these two extremes, rendering in both missed spyware programs and false labelling of legitimate software as spyware. In a prolongation, anti-spyware vendors need to chose to either miss spyware components, resulting in bad reputation, or to include legitimate software which leads to law suits.

This result in that spyware vendors somewhat arbitrarily decides what software to label as spyware and what not. Further, leading to a divergence between what software different countermeasure vendors target, i.e. some countermeasures remove one program while others leave it. These difficulties has further proved to result in legal disputes as software vendors feel unfairly treated by countermeasure vendors and therefore bring the case to court [31]. Such a situation is negative for both legitimate software vendors that find their products falsely labelled as spyware, and anti-spyware vendors that risk being sued when trying to protect their users' interests. This further result in that users' success rate in countering spyware depends on the *combination* of different countermeasure tools being used, since no single one offers full protection.

Current spyware countermeasures depend on their own classifications of what software that should be regarded as spyware. We believe that this model provides a too coarse mechanism to accurately distinguish between the various types of spyware and legitimate software that exist, since this is based on the individual users' own opinion. Most of the current spyware countermeasures are reactive and computer-oriented in their design, i.e. they focus on system changes to identify known spyware once they *already* have infected systems. Over the last years, some preventive countermeasures have also started to emerged which focus on hindering spyware *before* it has any chance to start executing on the computer.

However, such countermeasures still suffer from the issues connected to the per vendor governed spyware classifications. Each vendor has its own list of what software that should be regarded as spyware and these lists do not correlate.

We argue that there is a need for more user-oriented countermeasures, which should complement the existing computer-oriented anti-malware tools. Such complementing countermeasures should focus on informing users when they are forced to reach difficult trust decisions, e.g. whether to install a certain piece of software or not. However, the goal for such mechanisms should *not* be to make these trust decisions for users. In the end, it is up to the users themselves to consider advantages and disadvantages before reaching the decision.

2.10 Reputation Systems

Reputation systems are in essence an algorithm for calculating and serving reputation scores for a set of persons or objects [41]. The reputation scores are calculated dynamically based on incoming ratings from community users. It is the reputation servers that collect, aggregate and distribute the reputation scores for objects; in an attempt to facilitate trust by visualising reputation to the community users. It is the collective opinion within the community that determines an object's reputation score; giving the system both collaborative sanctioning and praising effects by praising high quality while sanctioning low.

As a result reputation systems help community users when making trust decisions even though most members do not know each other in real life. Reputation systems could therefore be seen as a way to take person-to-person gossip and personal experience (that we rely on in everyday-life) into an Internet setting. An example of a widely known reputation system is the one incorporated in eBay.com.

One problem with reputation systems is that some users are reluctant to spend the additional time needed for inserting their own experience into the system, e.g. by rating products [51]. Some users also feel unpleasant when assigning negative ratings and for that reason ignore the rating all together.

Another problem is that many online communities rely on pseudonyms instead of real names for identification purposes, which also means that any user can start another identity simply by setting up a

new pseudonym. This limits the effects of a reputation system since any user that is dissatisfied with the reputation can restart from scratch. Due to this reputation systems are vulnerable to various attacks such as the Sybil attack [34], where single users sign up for more than one pseudonyms each, which result in the power of several ratings per user within the system.

Reputation systems are related to both *recommender systems* [52] and *collaborative filtering* [29]. The main difference is that reputation systems calculate reputation scores based on explicit ratings from the community; while recommender systems instead rely on events such as book purchases when generating recommendations to the users. Collaborative filtering typically involves a two-step process where other users that have similar tastes are identified first, and then the ratings from these like-minded users are used for generating suggestions for the intended user.

2.11 Data Mining

The overarching goal within *data mining* is to transform data into information, which most often involves identifying patterns within large data sets that are impossible for humans to find manually. Data mining could therefore be defined as "the process of discovering patterns in data" [80]. As the size of databases is ever growing there is an increasing need for data mining techniques; for instance in text categorization [61], marketing/buying-habit analysis [4], fraud detection [33], and junk E-mail filtering [55].

In data mining, problems are often studied by analyzing samples, or observations of data. For example, if we are interested in developing a new method for distinguishing between Spam and legitimate E-mail, we may obtain a data set of E-mails from both categories. The objective could be to use a data mining algorithm for automatically learning how to classify new E-mails by generalizing from the studied data set. Obviously, we will never gain access to all existing E-mails and it would most certainly be computationally infeasible to sift through such a data set. Thus, we make the assumption that, by collecting a large enough representative sample – we may address the complete problem by studying these observations.

On an abstract level the data mining process can be explained using the following four steps; *data selection*, *pre-processing*, *mining*, and *result validation* [35, 80].

Data selection requires an understanding of the application domain and knowledge about the goal of the data mining procedure. Based on the goal, the researcher selects a subset of the data to focus further efforts on. However, most often it is not possible to chose the data; instead one has to stick with the data available.

During pre-processing noisy and irrelevant data is removed and it is decided what attributes that should be used; this is also referred to as *feature selection*. Pre-processing also involves organizing the data into structures that the chosen algorithms can handle, given the computational power at hand. This step also involves the division of data into a *training* and a *test* set; where the mining algorithms rely on the training set to find patterns, while the test set is used for verifying these patterns.

The third step involves the actual mining of patterns based on four types of techniques [46, 80]:

- *Classification*, classify the data entries into predefined discrete classes, e.g. legitimate E-mail or Spam.
- *Clustering*, arrange data entries together into groups just as classification, but with the distinction that no predefined groups exist, i.e. trying to group similar entries together.
- *Regression*, approximating a real-valued target function.
- *Association rule learning*, tries to find relationships among attributes within the data.

In the fourth step the patterns from the mining algorithms is verified to be valid not only for the training set, but also the test data. Patterns detected by mining algorithms do not necessarily have to be valid. It is for instance possible for the algorithms to detect patterns in the training data that does not exist when considering the wider data set, i.e. when including the test data. This problem is called overfitting and means that the mining algorithm has optimized too much on the training data, on the expense of generalizability [46].

This is the reason why another data set is used for validity and evaluation purposes, i.e. a data set that the algorithms have not been trained on. During the validation the identified patterns (from the training data) are applied on the test data. If the learnt patterns do not match both data sets the whole data mining process needs to be re-iterated using a different approach, but it could also be the case that patterns simply are not available in the data at all. On the other

hand, if the patterns match both data sets the analysis of the results can proceed.

CHAPTER

3

Research Approach

3.1 Motivation and Research Questions

Even though spyware and countermeasures are interesting to study from several perspectives, such as technology, law and human-computer interaction, we will keep a technology focus in this thesis. However we will also occasionally touch upon the other areas. When we began our research concerning spyware, the existing knowledge was rather sparse, even parsimonious in relation to the degree of negative impact these programs had on the users' computer experiences [49]. Today, the occurrence of illegitimate software has become a major security issue for both corporations and home users on the Internet. As we adopt to an increasingly more computerized life, it will be of great importance to manage the problems associated with questionable software so that the integrity and control of users' computers can be protected.

However, since no accurate definition or classification exists for such software, the reports and discussions of their effects are often vague and sometimes inconsistent. Although previous work shows that illegitimate software invades users' privacy, disrupt the users' computer experience, and deteriorates system performance and security, one could wonder what actually is being measured if there is no clear definition or encapsulation of the problem [3, 65].

Today, several countermeasures against questionable software exist, but many of them rely on techniques that work mostly for already

known threats. Furthermore these countermeasures often do not detect threats until they have already gained entrance to the system. Even though there exist preventive tools that lock down a system so that no software can enter unless the user allows it to, these are often difficult even for non-technical users to configure and operate. Such tools result in that users need to reach security related decisions based on the insufficient information presented to them through warning and notification messages. Messages that usually include a technical or juridical language which many users find hard to interpret and therefore benefit from. These problems have motivated us to investigate the classification and effects of spyware; and how preventive mechanisms against spyware could be designed. We therefore put forward the following five research questions (all assuming the more abstract use of the term spyware described in Chapter 2):

RQ1 How can a classification of spyware be formulated with respect to privacy-invasive behaviour?

RQ2 What negative effects are related to spyware, i.e. with regard to information theft, system performance, and security?

RQ3 How can a reputation system be constructed to counteract spyware using computer users' experience about software?

RQ4 What problems with regard to system accuracy are associated with the use of a reputation system for counteracting spyware?

RQ5 How can data mining algorithms be used for producing classifiers that can distinguish between EULAs of spyware-infested software and EULAs of legitimate software?

3.2 Research Methods

We have used a combination of different research methods including experiments, theoretical analysis, and simulation [18, 53]. Different research approaches were also used for each one of the five research questions. Both RQ1 and RQ3 were approached using theoretical analysis/modelling, aiming to find and understand already existing classifications and countermeasures. The outcome was then compiled and analysed in search of both strengths and weaknesses.

For RQ2 we relied on an experimental approach and an empirical method based on experiments to evaluate a set of software bundled with spyware and their consequences on the host system. These experiments were conducted in a systematic, repeatable, and logical way, and was based on data collection, data analysis and data verification.

Simulation was used in RQ4 when investigating the problems associated with a software reputation system. We implemented a simulator that was used to investigate different scenarios related to the use and misuse of a software reputation system.

RQ5 rely on an experimental approach together with statistical analysis and significance tests to evaluate a set of learning algorithms abilities to produce classifiers that separates EULAs associated with spyware-infested software from EULAs of legitimate software. Further information about the research methods used is presented in the included publications.

3.3 Thesis Contribution

The main contributions of this thesis are associated with the five research questions that are specified above. In addition, we also regard the extensive description of the spyware concept presented in Chapter 2 to be one of the contributions of this thesis. Another contribution is our conclusion that it is impossible to accurately define a global spyware categorization since many of the parts are subjective in respect to the users. This further leads to the introduction of user-oriented countermeasures where the user needs to define software as legitimate or not, based on new aiding mechanisms. Next we address the five research questions, which are discussed both in the research question part below and in the subsequent discussion, i.e. we want to address the questions both individually and together.

3.3.1 Research Question 1

Previous research has identified a problem with the lack of a standard spyware definition [30]. A joint conclusion is that it is important for both software vendors and users that a clear separation between acceptable and unacceptable software behaviour is established [8, 64]. As we conclude in Chapter 2 the concept of spyware is difficult

to capture in a short, but yet commonly agreeable definition. The reason for this is the subjective nature of many spyware programs included, which result in inconsistencies between different users' beliefs, i.e. what one user regards as legitimate software could be regarded as a spyware by others. As the spyware concept came to include increasingly more types of programs, the term became fuzzy and vague.

We therefore choose to introduce the term *privacy-invasive software* (PIS) to encapsulate all such software. We believe this term is more descriptive than other synonyms (e.g. trackware, scareware, evilware and badware) without having as negative connotation. Even if we use the word "invasive" to describe such software, we believe that an invasion of privacy can be both desired and beneficial for the user as long as it is fully transparent, e.g. when implementing specially user-tailored services or when including personalization features in software.

We used the work by Warkentins et al. as a starting point when developing a classification of PIS [74]. As a result we classify PIS as a combination of *user consent* and *direct negative consequences*. User consent is specified as either *low*, *medium* or *high*, while the degree of direct negative consequences span between *tolerable*, *moderate*, and *severe*. This classification allows us to first make a distinction between legitimate software and spyware, and secondly between spyware and malicious software. All software that has low user consent, or which impairs severe direct negative consequences should be regarded as malware. While, on the other hand, any software that has high user consent, and which results in tolerable direct negative consequences should be regarded as legitimate software.

In addition to the direct negative consequences, we also introduce *indirect negative consequences*. By doing so our classification distinguishes between any negative behaviour a program has been designed to carry out (direct negative consequences) and security threats introduced by just having that software executing on the system (indirect negative consequences). One example of an indirect negative consequence is the exploitation risk of software vulnerabilities in programs that execute on users' systems without their knowledge [57]. This classification is further described in Publication 4 and 7.

3.3.2 Research Question 2

To explore the effects that PIS have on computer systems we conducted a number of experiments with the goal to investigate PIS bundled with five, at that time, leading file-sharing tools [7]. The results showed that all these file-sharing tools included adware, spyware, and downloaders (programs that allow for new software and/or updates to be downloaded and installed without first asking the user). All file-sharing tools also included PIS involved in Internet communication. However, it was not practically possible to further investigate exactly what information was transmitted over the network since the traffic was encrypted. However, in one case our empirical results confirmed that one of these tools transmitted privacy-invasive data such as visited Web sites, zip code, country, lists of other software installed on the computer, and the exact version of the operating system. Our results also confirm that many of the PIS components introduce new security risks since they allow new software and/or updates to be automatically downloaded and installed, which can be hijacked by attackers.

We used two different versions of the same file-sharing tool, in this case KaZaa and KaZaa Lite K++, to investigate the resource utilization of PIS on a local computer. By subtracting the resource utilization of KaZaa Lite K++, which had all PIS components removed (only leaving the file-sharing functionality) from the original KaZaa version (which included bundled with PIS), we were able to get a measurement of the amount of resources that was consumed by PIS. The results show that both the utilization of system resources, and network bandwidth were significantly higher for KaZaa compared to the cleaned version. The increased utilization of bandwidth and number of contacted servers were due to transmission of pop-up ads, banners, and new software updates for the PIS components themselves. Although the CPU utilization was rather low at 0.48%, it is interesting that PIS introduces a 32 time increase compared to the clean version[1]. Also, the usage of RAM was significantly higher, with a 10 time increase, leaving the original version of KaZaa at a 65MB memory usage.

In contrast to PIS supported file-sharing tools, installing a cleaned software equivalence cause marginal impact to the system and net-

1. The experiments used identical computers which included a Pentium 4 2.8Ghz processor.

work resources. However, due to the occurrence of PIS components in file-sharing tools, users with several such applications installed simultaneously will suffer from a continuous system and network degradation due to the aggregated impact. This includes increased security and stability risks, unlike today's focus on information gathering. More information regarding how these experiments were designed, executed, and their results are presented in Publication 1 and 2.

3.3.3 Research Question 3

So far, developers of countermeasure tools have used the same techniques as in malware countermeasures (such as anti-virus programs) when fighting spyware. Although there are several similarities between spyware and malware there also exist a few profound differences. For instance, spyware rather includes functions that gather information, show messages on the screen, or monitors visited Web behaviour; instead of more malware-like behaviour such as rapid self-propagation. When fighting obvious malware it is instead possible to clearly separate software that is considered malware from legitimate. Furthermore this could be done without risking to include any legitimate software, since they are so different from malware that these two groups do not overlap.

However, when targeting spyware which is closer to legitimate software than malware, it is impossible not to (incorrectly) include innocent programs. This is a problem since vendors of anti-spyware tools rely on a central classification that differs not only among different anti-spyware vendors, but also for single anti-spyware vendors over time, as is shown in Publication 3. A problem here is that such static classifications do not respect users' personal opinion about software. Two users may disagree on whether a certain software should be classified as spyware or not; one might think it is a free useful tool that show valuable ads and offers, while the other finds it invasive and highly irritating. This results in that miss-classifications occur as a consequence of this static division, which may further result in law suits against the vendor, as explained in Publication 3. In other words, these techniques are not effective against PIS [70].

Instead of merely relying on the same techniques that anti-malware tools utilize, we believe that spyware countermeasures should focus on *user consent*, when distinguishing spyware from legitimate prod-

ucts that are beneficially tailored toward the users' needs. Any countermeasure not doing so is either forced to label legitimate software as spyware, or miss true spyware due to the user-centered opinion of spyware. We believe that this situation has originated from the lack of a proper understanding of the spyware concept which further has made spyware a fuzzy concept; resulting in that spyware absorbed new program behaviours over time, which further complicated the construction of a definition. User consent constitutes the essence of our definition of PIS, since we believe it reasonably should be up to the users themselves to distinguish legitimate software from illegitimate. This is impossible for any anti-spyware tool since they lack the personal and subjective preferences that each user has regarding software, i.e., some users accept targeted pop-up ads as something positive while others reject it with almost religious beliefs. Although this is hard today, we believe it could be possible in the future with the help from user-oriented countermeasures that aid users in this process.

Since spyware does not include as disastrous behaviour as malware, current anti-spyware tools face a more complicated task trying to pinpoint spyware. Therefore future countermeasures also need to focus on informing the users so they can distinguish legitimate and illegitimate software based on their own individual preferences. Providing users that are about to install a certain software with the knowledge from previous users of that software, could help them get a notion of either trust or mistrust. By also providing the user with additional information, such as an overall rating of the software vendor, would allow interested users to further investigate the software in question. We therefore propose the use of *collaborative reputation systems* for providing users with these services [50, 81]. Such systems could handle individual users' knowledge, and refine it into a commonly shared knowledge-base. Similar reputation systems are currently used by for instance Internet Movie Database (IMDb.com) for rating movies, and by eBay (eBay.com) where users rate the performance of other parties that they have undertaken transactions with. The overall intention with a reputation system is to use user ratings as a trust enabler in the system, which is further described in Section 3.4.

It should be noted that such a reputation system against PIS is tightly connected with the PIS classification. The introduction of this type of user-oriented countermeasures would transform the classification of PIS. As users are given a tool that allows them to make informed decisions regarding the behaviour and implications

of software, it is possible to apply a sharp boundary based on user consent between all software in the PIS classification. Using the added knowledge provided by the reputation system would render in that all PIS that previously have suffered from a medium user consent level, now instead would be transformed into either a high consent level (i.e. legitimate software) or a low consent level (i.e. malware). In other words, all software with medium user consent, i.e. spyware, is transformed into either legitimate software or malware in the classification. Since anti-malware tools address malicious and deceitful software, the information about the rest of the software could be trusted to be correct, i.e., any software using deceitful methods is regarded as malware and is treated as such.

This allows users to rely the information when reaching trust decisions regarding their computer system. Another aspect of this type of countermeasure is that no single organization, company or individual is responsible for the software ratings since these are calculated based on all votes submitted by the users. As a result this makes it harder for dissatisfied spyware vendors to sue the developer of the countermeasure for defamation.

3.3.4 Research Question 4

To evaluate the use of a software reputation system we implemented a simulator that allows us to simulate vast numbers of users. The simulator includes representations of both users and software. All simulated users (one million) are divided into three groups that represent novice, average and expert users. These groups determine the users' accuracy when rating the simulated software, i.e. experts are able to rate software most accurately of the three groups. During simulations the users rate software on a discrete scale (1-10) based on their past experience of each particular software. In our simulated scenarios we include 100,000 software programs and each one is randomly assigned a correct rating, which later is used for evaluating the accuracy of a certain simulated scenario. The correct rating is also used as a starting point when determining users' ratings, together with a randomized vote variance that is calculated based on group-specific distributions.

We have simulated scenarios that investigate how different user demographics (various proportions of the three groups) affect the overall accuracy of the reputation system, i.e. how accurately the reputation system can deliver software ratings to users within the community. Another scenario involves the simulation of malicious

users that try to deteriorate system accuracy or boost/downgrade certain targeted programs. We also simulate different properties of the reputation system trying to find suitable configurations. One such important property is the way users' trust factors (specifying the impact each user has within the reputation system) should be modified. To prevent some users from getting too much impact within the system we also simulate various max limits that restrict users' trust factors from getting too high.

Our results from the simulations indicate that using a factor of 1.25 for increasing and decreasing users' trust factors provides a reasonable overall accuracy (± 1 step from the correct value). We also conclude that a maximum trust factor limit of around 1000 results in a beneficial trade-off between system accuracy and user impact. If no limit at all is used the experts' trust factors reach astronomical proportion; resulting in that even small misratings from them being amplified within the system, which results in huge impact on the overall accuracy of the reputation system. The results also show that even with less than 5% experts it is possible to reach a quite accurate software reputation system, that provide users with software ratings that differ with less than 1 rating unit from the simulated correct rating. A more detailed description of both the simulator itself and the simulated scenarios and results are available in Publication 6.

3.3.5 Research Question 5

In contrast to clearly malicious software, spyware programs often include End User License Agreements (EULAs) that users have to accept during the software installation process. As a way to avoid legal repercussions, spyware distributors include information about the software programs behaviour and effects in the EULAs. However, this information is written in a way that makes it hard for users to understand, for instance by formal legal language and writing in a lengthy manner. Our idea is to investigate whether it is possible to detect spyware-hosting software by analysing the text within the EULAs. More precisely we evaluate whether it is possible to use data mining and machine learning techniques to identify intrinsic patterns within the EULA texts.

Based on an experiment we evaluate the capability of using data mining for separating between legitimate software and spyware based on the EULAs. First we manually gathered EULAs from 96

known spyware and 900 legitimate software programs. Then we compared the performance of 17 data mining algorithms with that of a baseline algorithm.

The results show that the majority of learning algorithms significantly outperform the baseline. The best performing algorithm in our experiment was Naive Bayes Multinomial with an Area Under the ROC Curve (AUC) measure of 0.939. The AUC present the probability that a randomly chosen instance is correctly classified by the classifier; where 0.5 is worst case (50% risk of misclassification) and 1.0 is optimal. Naive Bayes Multinomial further showed false alarms for 44 good EULAs, and missed 17 of the bad EULAs by misclassifying them as good. Even though 17 missed bad EULAs is too high for a practical tool, we have to keep in mind that the experiment includes a limited number of bad EULAs. Also, none of the algorithms used were optimized/fine-tuned to suite the problem at hand, i.e. all algorithms had default configurations.

We therefore conclude that automatic EULA classification can be used to assist users when making informed decisions during software installation, i.e. whether to install an application without having read the EULA. In publication 5 we describe the experiment and results in more detail, and also outline a design for a novel prevention tool and discuss the effects such a tool could have on spyware creators.

3.4 Discussion and Future Work

The research presented in this thesis has resulted in an idea of a preventive mechanism that uses a collaborative reputation system to increase user awareness about software behaviour. To evaluate the impact of such a system we have built a proof-of-concept reputation system that could be used as a test base for evaluating how to enable informed decision-making regarding software installation and execution.

In a way, the proposed reputation system would use the same software reputation that users today gain from for instance computer-magazines and Web sites. However, one important distinction is that these sources rely on the user to manually retrieve the information, i.e. the user needs to pull it, while our proposed countermeasure instead use an automatic push approach.

We argue that the reputation system should constitute an active part in the installation process of the operating system; allowing it to notify the user each time a previously unknown software is about to execute or install on the system. When such an event occurs, the execution or installation process should pause until the associated information has been gathered from the knowledge-base, and has been presented to the user. This would allow the reputation system to provide users with important information when installing or executing new or unknown software. One example could be that the software they are about to install is developed by a vendor that is known to rely on incorrect and deceiving information for sneaking their product into users' computers.

Although such a reputation system introduces many benefits, it is also associated with several security issues that need to be considered. Boosting the reputation of a specific software is definitely interesting from the perspective of a PIS vendor, e.g. for increasing its distribution and popularity. Another problem would be companies or users that form alliances with the goal of smearing specific software. To address these threats the reputation system could use the five techniques presented below.

1. A single user should only be allowed to cast a single vote on each specific software.
2. Secondly, users should only be allowed to vote on software that has been started on the local computer more times than a certain threshold value, or which has a total execution time that exceeds a pre defined value. This would assure that the user has used the program for some time and therefore has gained at least some modest opinion before rating it.
3. In addition to the rating of software, the system should use meta-ratings that allow users to anonymously rate other users' comments about a specific software. As a consequence any user that tries to boost the reputation of a specific software by inserting deceptive information would be down-rated by other users, which further affects his/her own reputation and influence within the reputation system negatively.
4. Even though all votes should be included there should be a distinction in the amount of influence they play. The exact factor should be calculated by the voting accuracy and other users' ratings on the user's contributions, i.e. new users would have a low influence, but if they provide the system with useful information they will become more trustworthy and will thereby gain

greater influence. This idea is similar to the PageRank technique that Google.com utilizes when ranking the importance of Web sites.

5. Signing up for using the system should include procedures that aggravate[2] automatically signing up a large amount of new users. In addition to this, there should also exist a restriction in the rate that the trustworthiness for a user is allowed to increase, i.e. it should be impossible to boost a user's influence to the highest level in a short amount of time. Therefore, a user must use his account on a frequent occasion over a relatively long period of time, e.g. 12 months, to be able to earn the highest vote impact. This measure forces any antagonistic actors to invest a considerable amount of time to increase the trustworthiness of their accounts, before being able to stage an effective attack. Do note that the user still needs to receive excellent ratings for his/her participation in the system by other users, to earn a higher influence.

We believe such a reputation system would mitigate PIS by refining the individual knowledge of all users in the system into software-based reputations that are shared collectively. Both users and legitimate software vendors would benefit from such a system. The legitimate software vendors could use the system to clarify and promote what their software have been designed for, and how it would impact the user's computer systems. Users, on the other hand, would automatically receive both suggestions of useful software that has been well received by previous users, and warnings against questionable software before allowing them to install.

Hopefully this combined benefit would make the users more willing to share information about their software installations with the reputation system. The proof-of-concept system relies on users to provide a valid e-mail address together with continuous information about their experiences concerning certain software. It is important that all such information is stored in an unlinkable format that hinders the reputation system to consolidate, for instance, all software a certain e-mail address has ranked. The privacy issues introduced with such a system needs to be properly addressed with regard to the users; so they are willing to trust it with their personal information. It is not only necessary to develop a well functioning system,

2. For instance, techniques similar to the character recognition schemes (Captcha) used at many online services, such as Hotmail.com.

but it is also of great importance that it is designed to handle the users' information in a privacy respective way. If not, the very nature of such a system could be privacy-invasive towards its own users, e.g. with respect to information leakage.

Offering users mechanisms that enhance informed decisions regarding the software installation would also increase the liability of the user. In a way, these mechanisms would transfer some of the responsibility concerned with the protection against PIS to the users themselves; confronting them with descriptions about behaviours and consequences for PIS, which they need to assimilate and use in a mature and reasonable way. Based on the reputation system it would be up to the users themselves to decide on whether or not to allow certain software to enter their system.

Computer users today face similar difficulties when evaluating software as consumers did a hundred years ago when evaluating food products. In the nineteenth century the distribution of snake-oil product flourished [72]. These products claimed to do one thing, for example to grow hair, while they instead made unwitting consumer addicted to habit-forming substances. In 1906 the Pure Food and Drug Act was passed by the United States Congress, which forced manufacturers to declare any ingredients in their products [42]. The result was that consumers could trust the information on the food container to be correct; allowing them to make informed decisions on whether they should consume a product or not. As long as the food does not include poisonous substances or use deceptive descriptions it is up to the consumer to make the final decision.

In addition to the reputation system, we also introduce the idea of using automatic EULA classification based on data mining techniques. By including the data mining classifier into a decision-support tool it would be possible for users to have software EULAs automatically detected and classified during installation; providing users with additional clues to allow the software to install or not. Such a tool could also work in parallel with the software reputation system; allowing both subsystems to cover for misjudgments from the other. It would in fact be fully possible to combine several dif-

ferent techniques into a combined tool that assist the users, as shown in Figure 3.1.

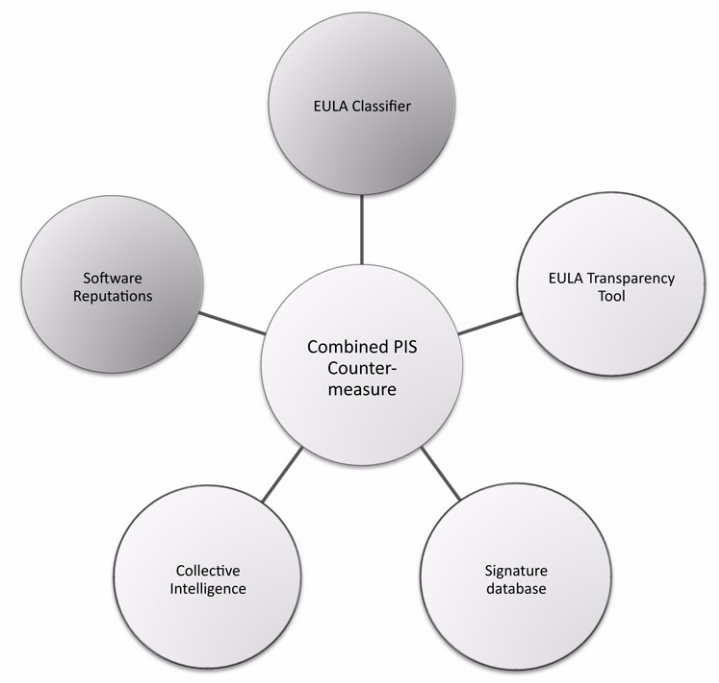

Figure 3.1 **Interaction of different techniques into a combined PIS countermeasure; where two out of five techniques are being addressed in this work.**

The *EULA Transparency Tool* could allow software vendors to extracts the content in the EULA and present it in a short and easily understood format, e.g. using short text descriptions in combination with pictograms. So, in addition to a EULA classification it would also be possible for the users to get one page summaries of EULA content that easily can be comprehended. Similar techniques are used for instance within the Creative Commons licenses [17].

Signature Database refers to a database with signatures (unique descriptions or "fingerprints") of known malware, which could be used for detection when other no other means are available. This particular subsystem would work like traditional anti-malware tools which need to be served with new signatures from anti-malware researchers.

Collective Intelligence refers to a novel idea of investigating software behaviour by analysing the users' search patterns from Internet search engines. Recently query data were used for predicting real-world influenza epidemics by comparing high ranked search queries with surveillance data [28]. In our setting, undefined programs show traces of good or bad behaviour depending on the collective intelligence of users formulating search query data. As the users' computers get infected with some PIS they might experience negative symptoms, which they try to learn more about using for example Google searches. By using data mining techniques it might be possible to detect PIS epidemics on the Internet based on such search queries.

A combined PIS countermeasure could allow a single user to get enough information for making an informed decision during software installations. The various parts within the system could also backup for each other; so a user get some feedback by the system even if one or two of the subsystems fail to detect a particular software.

For future work we will further investigate the use of collective intelligence when fighting PIS. We will also continue our research on EULA analysis in search for increased performance. Ongoing research (using the same data-set as in Publication 5) show great promise with an Area Under the Curve (AUC) measure of 0.998 in combination with 0 missed bad EULAs and a false alarm-rate of merely 0.017. As the next step we will increase the EULA data-set by co-operating with a leading anti-spyware corporation. We also intend to carry out a user study that investigates the impact such a decision-support system have on users' behaviour with regard to software installation.

In addition to this we will also further analyse the workings of a software reputation system; for instance by simulating more advanced malicious users, and a meta-reputation system where users could rate each others' feedback.

3.5 References

[1] M. Andreessen, "NCSA Mosaic Technical Summary", National Center for Supercomputing Applications, 1993.

[2] Anti-Spyware Coalition, "Anti-Spyware Coalition", http://www.antispywarecoalition.org, Last checked: 2010-02-11.

[3] K.P. Arnett and M.B. Schmidt, "Busting the Ghost in the Machine", in *Communications of the ACM*, Volume 48, Issue 8, 2005.

[4] M. J. Berry, and G. Linoff, *"Data Mining Techniques: For Marketing, Sales, and Customer Support"*, John Wiley & Sons Inc., New York NY, 1997.

[5] M. Boldt and B. Carlsson, "Analysing Countermeasures Against Privacy-Invasive Software", in *the proceedings of the IEEE International Conference on Software Engineering Advances (ICSEA'06)*, Papeete French Polynesia 2006.

[6] M. Boldt and B. Carlsson, "Privacy-Invasive Software and Preventive Mechanisms", in *the proceedings of the IEEE International Conference on Systems and Network Communications (ICSNC'06)*, Papeete French Polynesia, 2006.

[7] M. Boldt, A. Jacobsson, and B. Carlsson, "Exploring Spyware Effects", in *proceedings of the 8th Nordic Workshop on Secure IT Systems (NordSec04)*, Helsinki Finland, 2004.

[8] J. Bruce, "Defining Rules for Acceptable Adware", in the *Proceedings of the 15th Virus Bulletin Conference*, Dublin Ireland, 2005.

[9] B. Carlsson, "*Conflicts in Information Ecosystems – Modelling Selfish Agents and Antagonistic Groups"*, Doctoral Dissertation Thesis Series No. 2001:03, School of Engineering, Blekinge Institute of Technology, Sweden, 2001.

[10] Center for Democracy & Technology, "Following the Money", http://www.cdt.org, Last checked: 2010-02-11.

[11] C. Abhijit, J.P. Kuilboer, *"E-Business & E-Commerce Infrastructure: Technologies Supporting the E-Business Initiative"*, McGraw Hill, 2002.

[12] R.K. Chellappa and R.G. Sin, "Personalization versus Privacy: An Empirical Examination of the Online Consumer's Dilemma", in the *ACM Information Technology and Management*, Volume 6, Issue 2, 2005.

[13] E. Chien, "Techniques of Adware and Spyware", in the *Proceedings of the 15th Virus Bulletin Conference*, Dublin Ireland, 2005.

[14] C|NET Anti Spyware Workshop, "The Money Game: How Adware Works and How it is Changing", San Francisco CA, 2005.

[15] C|NET News.com, "Stealth P2P Network Hides Inside KaZaa", http://news.com.com/2100-1023-873181.html, Last checked: 2010-02-11.

[16] L.F. Cranor, "Giving Notice: Why Privacy Policies and Security Breach Notifications aren't Enough", in *IEEE Communications Magazine*, Volume 43, Issue 8, 2005.

[17] Creative Commons, "Licenses", http://creativecommons.org/about/licenses/ Last checked: 2010-02-11.

[18] J. W. Creswell, *"Research Design"*, Sage Publications, Thousand Oaks CA, 2003.

[19] P.M. Doney and J.P. Cannon, "An Examination of the Nature of Trust in Buyer-Seller Relationships", in the *Journal of Marketing*, Volume 61, 1997.

[20] M.W. Eichin and J. Rochlis, "With Microscope and Tweezers: An Analysis of the Internet Virus of November 1988", in the *Proceedings of the 1989 IEEE Computer Society Symposium on Security and Privacy*, Oakland Ohio, 1989.

[21] S. Feiner, B. Macintyre, and D. Seligmann, "Knowledge-based augmented reality", in the *Communications of the ACM*, Volume 36, Issue 7, 1993.

[22] B. Friedman, E. Felten, and L.I. Millett, "Informed Consent Online: A Conceptual Model and Design Principles", *CSE Technical Report*, University of Washington, 2000.

[23] B. Friedman, P.H. Kahn, and D.C. Howe, "Trust Online", in the *Communications of the ACM*, Volume 43, Issue 12, 2000.

[24] S. Furnell et al., "Considering the Usability of End-User Security Software", in the *Proceedings of the 21st International Information Security Conference (Sec2006)*, Karlstad Sweden, 2006.

[25] S. Ganesan, "Determinants of Long-Term Orientation in Buyer-Seller Relationships", in the *Journal of Marketing*, Volume 58, 1994.

[26] S. Garfinkel, *"Database Nation"*, O'Reilly & Associates, Sebastopol CA, 2001.

[27] Gibson Research Corporation, "OptOut – Internet Spyware Detection and Removal",
http://www.grc.com/optout.htm,
Last checked: 2010-02-11.

[28] J. Ginsberg et al., "Detecting influenza epidemics using search engine query data", in *Nature*, Volume 457, Issue 7232, 2009.

[29] D. Goldberg, D. Nichols, B.M. Oki and D. Terry, "Using collaborative filtering to weave an information tapestry", in *Communication of the ACM*, Volume 35, Issue 12, 1992.

[30] N. Good et al., "Stopping Spyware at the Gate: A User Study of Privacy, Notice and Spyware", in the *proceedings of the Symposium On Usable Privacy and Security (SOUPS 2005)*, Pittsburgh USA, 2005.

[31] N. Good et al., "User Choices and Regret: Understanding Users' Decision Process about Consentually Acquired Spyware", in *I/S: A Journal of Law and Policy for the Information Society*, Volume 2, Issue 2, 2006.

[32] S. Görling, "An Introduction to the Parasite Economy", in the *Proceedings of EICAR*, Luxembourg, 2004.

[33] C.S. Hilas, and P. A. Mastorocostas, "An Application of Supervised and Unsupervised Learning Approaches to Telecommunications Fraud Detection", in *Knowledge-Based Systems*, Volume 21, Issue 7, 2008.

[34] K. Hoffman, D. Zage and C. Nita-Rotaru, "A survey of attack and defense techniques for reputation systems", in *ACM Computing Surveys (CSUR)*, Volume 42, Issue 1, 2009.

[35] U. Fayyad, G. Piatetsky-Shapiro, and P. Smyth, *"From data mining to knowledge discovery: an overview"*, American Association for Artificial Intelligence, Menlo Park CA, 1996.

[36] S. Fischer-Hübner, *"IT-Security and Privacy: Design and Use of Privacy-Enhancing Security Mechanisms"*, Springer Verlag, Berlin Heidelberg, 2001.

[37] Internet Advertising Bureau (IAB),
http://www.iab.net/insights_research/1357
Last checked: 2010-02-11.

[38] International Consumer Electronics Association,
http://www.cesweb.org,
Last checked: 2010-02-11.

[39] A. Jacobsson, "*Exploring Privacy Risks in Information Networks"*, Licentiate Thesis Series No. 2004:11, School of Engineering, Blekinge Institute of Technology, Sweden, 2004.

[40] A. Jacobsson, M. Boldt and B. Carlsson, "Privacy-Invasive Software in File-Sharing Tools", in *proceedings of the 18th IFIP World Computer Congress (WCC2004)*, Toulouse France, 2004.

[41] A. Jøsang, et al., "A Survey of Trust and Reputation Systems for Online Service Provision", in *Decision Support Systems*, Volume 43, Issue 2, 2007.

[42] Landmark Document in American History, "Pure Food and Drug Act of 1906", http://coursesa.matrix.msu.edu/~hst203/documents/pure.html, Last checked: 2010-02-11.

[43] Los Angeles Times, "Google secures patent to use location data in ads", http://latimesblogs.latimes.com/technology/2010/03/google-location-patent.html, Last checked: 2010-03-02.

[44] Augumented Reality Browser, "Layer", http://www.layar.com, Last checked: 2010-02-11.

[45] P. McFedries, "The Spyware Nightmare", in *IEEE Spectrum*, Volume 42, Issue 8, 2005.

[46] T. M. Mitchell, *"Machine Learning (International Edition)"*, McGraw-Hill Book Co., Columbus USA, 1997.

[47] A. Moshchuk, T. Bragin, S. D. Gribble, and H. M. Levy, "A Crawler-based Study of Spyware on the Web", in the *Proceedings of the 13th Annual Network and Distributed System Security Symposium (NDSS 2006)*, San Diego CA, 2006.

[48] M.W. Newman et. al., "Recipes for Digital Living", in *IEEE Computer*, Vol. 39, Issue 2, 2006.

[49] Pew Internet & American Life Project, "The Threat of Unwanted Software Programs is Changing the Way People use the Internet", http://www.pewinternet.org, Last checked: 2010-02-11.

[50] P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom, and J. Riedl, "GroupLens: An Open Architechture for Collaborative Filtering of Netnews", in *Proceedings of ACM Conference on Computer Supported Cooperative Work*, Chapel Hill NC, 1994.

[51] P. Resnick, K. Kuwabara, R. Zeckhauser and E. Friedman, "Reputation systems", in *Communications of the ACM*, Volume 43, Issue 12, 2000.

[52] P. Resnick and H.R. Varian, "Recommender systems", in *Communications of the ACM*, Volume 40, Issue 3, 1997.

[53] C. Robson, *"Real World Research (2nd Edition)"*, Blackwell Publishing, Ltd., Oxford UK, 2002.

[54] R.S. Rosenberg, "*The Social Impact of Computers*", 3rd edition, Elsevier Academic Press, San Diego CA, 2004.

[55] M. Sahami et al., "A Bayesian Approach to Filtering Junk Email", in *Proceedings of the AAAI-98 Workshop on Learning for Text Categorization*, Madison USA, 1998.

[56] P.E. Sand, "The Privacy Value", in *I/S: A Journal of Law and Policy for the Information Society*, Volume 2, Issue 2, 2006.

[57] S. Saroiu, S.D. Gribble, and H.M. Levy, "Measurement and Analysis of Spyware in a University Environment", in *Proceedings of the 1st Symposium on Networked Systems Design and Implementation (NSDI)*, San Francisco USA, 2004.

[58] F.B. Schneider, "The Next Digital Divide", in the *IEEE Security & Privacy*, Vol. 2, Issue 1, 2004.

[59] B. Schneiderman, "Designing Trust into Online Experiences", in the *Communications of the ACM*, Volume 43, Issue 12, 2000.

[60] B. Schneier, "Inside Risks: Semantic Network Attacks", in *Communications of the ACM*, Volume 43, Issue 12, 2000.

[61] F. Sebastiani, "Machine Learning in Automated Test Categorization", in *ACM Computing Surveys*, Volume 34, Issue 1, 2002.

[62] K.B. Sheehan and M.G. Hoy, "Dimensions of Privacy Concern among Online Consumers", in the *Journal of Public Policy & Marketing*, Volume 19, Issue 1, 2000.

[63] S. Shukla and F. F. Nah, "Web Browsing and Spyware Intrusion", in *Communications of the ACM*, Volume 48, Issue 8, 2005.

[64] J.C. Sipior, "A United States Perspective on the Ethical and Legal Issues of Spyware", in *Proceedings of 7th International Conference on Electronic Commerce*, Xi'an China, 2005.

[65] J.C. Sipior, B.T. Ward, and G.R. Roselli, "A United States Perspective on the Ethical and Legal Issues of Spyware", in the *proceedings of*

the 7th International Conference on Electronic Commerce (ICEC 2005), Xi'an China, 2005.

[66] E. Skoudis, "*Malware – Fighting Malicious Code"*, Prentice Hall PTR, Upper Saddle River NJ, 2004.

[67] SpotOn GPS,
http://www.spotongps.com
Last checked: 2010-02-10.

[68] StopBadware.org, "StopBadware.org",
http://www.stopbadware.org,
Last checked: 2010-02-11.

[69] StopBadware.org, "Software Guidelines",
http://www.stopbadware.org/home/guidelines,
Last checked: 2010-02-11.

[70] Spyware: Research, Testing, Legislation, and Suits,
http://www.benedelman.org/spyware/,
Last checked: 2010-02-11.

[71] P. Szor, *"The Art of Computer Virus Research and Defence"*, Addison-Wesley, Upper Saddle River NJ, 2005.

[72] Technology Review, "The Pure Software Act of 2006",
http://www.simson.net/clips/2004/2004.TR.04.PureSoftware.pdf,
Last checked: 2010-02-11.

[73] United States Patent: 7668832,
http://patft.uspto.gov/netacgi/nph-Parser?Sect1=PTO2&Sect2=HITOFF&u=/netahtml/PTO/search-adv.htm&r=6&p=1&f=G&l=50&d=PTXT&S1=google.ASNM.&OS=an/google&RS=AN/google
Last checked: 2010-03-02.

[74] M. Warkentin, X. Luo, and G. F. Templeton, "A Framework for Spyware Assessment", in *Communications of the ACM*, Volume 48, Issue 8, 2005.

[75] S.D. Warren, L.D. Brandeis, "The Right to Privacy", in *Harvard Law Review*, Volume 4, Issue 5, 1890.

[76] Webroot Software, "State of Spyware – Q1 2006",
http://www.webroot.com/pdf/2005-q2-sos.pdf,
Last checked: 2010-02-11.

[77] Webroot Software, "Differences between Spyware and Viruses",
http://research.spysweeper.com/differences.html,
Last checked: 2010-02-11.

[78] A. Weiss, "Spyware Be Gone", in the *ACM netWorker*, Volume 9, Issue 1, 2005.

[79] A. Westin, "*Privacy and Freedom"*, Atheneum, New York NY, 1968.

[80] I.H. Witten, and E. Frank, *"Data Mining - Practical Machine Learning Tools and Techniques"*, Elsevier, San Francisco CA, 2005.

[81] G. Zacharia, A. Moukas, P. Maes, "Collaborative Reputation Mechanisms in Electronic Marketplaces", in *Proceedings for the 32nd Hawaii International Conference on System Sciences*, Wailea Maui Hawaii, 1999.

[82] X. Zhang, "What Do Consumers Really Know About Spyware?", in *Communications of the ACM*, Volume 48, Issue 8, 2005.

PUBLICATION

1

Privacy-Invasive Software in File-Sharing Tools

18th IFIP World Computer Congress (WCC2004), 2004

Andreas Jacobsson, Martin Boldt and Bengt Carlsson

Personal privacy is affected by the occurrence of adware and spyware in peer-to-peer tools. In an experiment, we investigated five file-sharing tools and found that they all contained ad-/spyware programs, and, that these hidden components communicated with several servers on the Internet. Although there was no exchange of files by way of the file-sharing tools, they generated a significant amount of network traffic. Amongst the retrieved ad-/spyware programs that communicated with the Internet, we discovered that privacy-invasive information such as, e.g., user data and Internet browsing history was transmitted. In conclusion, ad-/spyware activity in file-sharing tools creates serious problems not only to user privacy and security, but also to network and system performance. The increasing presence of hidden and bundled ad-/spyware programs in combination with the absence of proper anti-ad/spyware tools are therefore not beneficial for the development of a secure and stable use of the Internet.

4.1 Introduction

As the Internet becomes more and more indispensable to our society, the issue of personal information is recognised as decisively important when building a secure and efficient social system on the Internet [3, 19]. Also, in an increasingly networked world, where new technologies and infrastructures, from pervasive computing to mobile Internet, are being rapidly introduced into the daily lives of ordinary users, complexity is rising [15]. As a consequence, vulnerabilities in systems are more eminent and greater in number than ever before. At the same time, the business climate on the Internet is tightening; e-commerce companies are struggling against business intelligence techniques, social engineering and frauds. A powerful component in any business strategy is user/customer information. In general, the company with the most information about its customers and potential customers is usually the most successful one [13, 19]. With respect to personal customer information, consumers generally want their privacy to be protected, but businesses, on the other hand, need reliable personal information in order to reach consumers with offers [13]. Undoubtedly, these demands must be satisfied to establish sound e-commerce, and a secure and well-functioning use of the Internet. However, these conflicting goals leave the control of user information at great risk, and a consequence may be that the users feel uneasy about sharing any personal information with commercial web sites. Human activity on the Internet will only thrive if the privacy rights of individuals are balanced with the benefits associated with the flow of personal information [13].

The problem of assuring user privacy and security in a computerised setting is not new, it has been a discussion for more than 30 years now [9]. However, there are some new aspects, that need to be highlighted. In this paper, we intend to explore privacy aspects concerning software components that are bundled and installed with file-sharing tools. Since file-sharing tools are used exclusively when connected to the Internet, users constitute a good foundation for online marketing companies to display customised ads and offers for users. The displayed contents of these offers are sometimes based on the retrieval of users' personal information. Usually, this kind of software operation is considered to be an invasion of personal privacy [8]. One of the most simple and clear definitions of privacy was first proposed in 1890 by Warren and Brandeis in their article "The Right to Privacy" [23], where privacy was defined as

"*the right to be let alone*". In general, privacy is the right of individuals to control the collection and use of information about themselves [3]. In an Internet setting, the extraction of the definition by Warren and Brandeis has come to mean that users should be able to decide for themselves, when, how, and to what extent information about them is communicated to others [7]. Previous work has suggested that malicious software, or malware, set to collect and transmit user information and/or to display ads and commercial offers without the consent of users have been found bundled with file-sharing tools [11, 22]. There are two kinds of software programs that perform such actions: adware displays advertisements, and spyware goes further and tracks and reports on users' web browsing, keystrokes or anything else that the author of the software has some interest in knowing. In reality, this means that software can be adware and spyware at the same time. However, not all adware is spyware and most spyware is not easily detected by displaying ads [11].

Ad-/spyware has gained a lot of space and attention lately. According to the Emerging Internet Threats Survey 2003 [6], one in three companies have already detected spyware on their systems, while 60% consider spyware to be a growing and future threat. Also, 70% of the companies say that peer-to-peer (P2P) file-sharing is creating an open door into their organisation. When it comes to adware, the Emerging Internet Threats Survey, states that adware and the use of file-sharing tools in office hours are devious and offensive threats that frequently evade both firewalls and anti-virus defences [6]. In effect, ad-/spyware creates problems, not only to user privacy, but also to corporate IT-systems and networks.

In this paper, we investigate what kind of privacy-invasive software that come bundled with five popular file-sharing tools. We also look into the Internet traffic that is being generated by these hidden programs. A discussion concerning the occurrence of ad-/spyware and its effects on privacy and security is undertaken. In the end, we present conclusions and findings.

4.2 Privacy-Invasive Programs and their Implications

One of the major carriers of ad-/spyware programs are P2P file-sharing tools [16, 22]. P2P refers to a technology which enables two

or more peers to collaborate in a network of equals [12, 18]. This may be done by using information and communication systems that are not depending on central coordination. Usually, P2P applications include file sharing, grid computing, web services, groupware, and instant messaging [12, 18]. In reality, there is little doubt that P2P networks furnish in spreading ad-/spyware [16]. Besides legal difficulties in controlling the content of P2P networks, another contributing factor is that the user is forced to accept a license agreement in order to use the software, but the contract terms are often formulated in such a way that they are hard for the user to interpret and understand. The effect is that most users do not really know what they have agreed to, and thus really cannot argue their right to privacy.

The occurrence of ad-/spyware programs in file-sharing tools pose a real and growing threat to Internet usage in many aspects, and to other interested parties than only to end users. Some examples argued on this topic are [6, 16, 22]:

- **Consumption of computing capacity:** Ad-/spyware is often designed to be secretly loaded at system start-up, and to run partly hidden in the background. Due to that it is not unusual for users to have many different instances of ad-/spyware running covertly simultaneously, the cumulative effect on the system's processing capacity can be dramatic. Another threat is the occurrence of distributed computing clients, bundled with file-sharing tools, that can sell the users' hard drive space, CPU cycles, and bandwidth to third parties.
- **Consumption of bandwidth:** Just as the cumulative effect of ad-/spyware running in the background can have serious consequences on system performance, the continual data traffic with gathering of new pop-ups and banner ads, and delivery of user information can have an imperative and costly effect on corporate bandwidth.
- **Legal liabilities:** With the new directives[1] concerning the use of file-sharing tools in companies, it is the company rather than a single user who is legally liable for, for instance, the breach of copyright (e.g., if employees share music files with other peers)

1. Examples on legal directives are the "Directive on Privacy and Electronic Communications" [5] of the European Union, and the "Spyware Control and Privacy Protection Act" [2] of the Senate of California, U.S.

and the spreading of sensitive information (e.g., if spyware programs transmit corporate intelligence).

- **Security issues:** Ad-/spyware covertly transmits user information back to the advertisement server, implying that since this is done in a covert manner, there is no way to be certain of exactly what information is being transmitted. Even though adware, in its purest form, is a threat to privacy rather than security, some adware applications have begun to act like Trojan horses allowing installation of further software, which may include malware. Security experts use the term Trojan horse for software that carries programs, which mask some hidden malicious functionality, but many web users and privacy experts use it to describe any program that piggybacks another. It is claimed that most of the latter are P2P file-sharing software that emerged as ad-supported alternatives in the wake of Napster's decline. In effect, if a computer has been breached by a Trojan horse, it typically cannot be trusted. Also, there is a type of spyware that has nothing to do with adware, the purpose here is to spy on the user and transmit keystrokes, passwords, card numbers, e-mail addresses or anything else of value to the software owner/author. In reflect, most security experts would agree that the existence of ad-/spyware is incompatible with the concept of a secure system.
- **Privacy issues:** The fact that ad-/spyware operates with gathering and transmitting user information secretly in the background, and/or displays ads and commercial offers that the user did not by him-/herself chose to view, makes it highly privacy-invasive.

Most ad-/spyware applications are typically bundled as hidden components of freeware or shareware programs that can be downloaded from the Internet [22]. Usually, ad-/spyware programs run secretly in the background of the users' computers. The reason for this concealing of processes is commonly argued as that it would hardly be acceptable if, e.g., free file-sharing software kept stopping to ask the user if he or she was ready to fetch a new banner or a pop-up window. Therefore, the client/server routine of ad-/spyware is executed in the background. In practice, there would be nothing wrong with ad-/spyware running in the background provided that the users know that it is happening, what data is being transmitted, and that they have agreed to the process as part of the conditions for obtaining the freeware. However, most users are unaware of that they have software on their computers that tracks

and reports on their Internet usage. Even though this may be included in license agreements, users generally have difficulties to understand them [22].

Adware is a category of software that displays commercial messages supported by advertising revenues [20]. The idea is that if a software developer can get revenue from advertisers, the owner can afford to make the software available for free. The developer is paid, and the user gets free, quality software. Usually, the developer provides two versions of the software, one for which the user has to pay a fee in order to receive, and one version that is freeware supported by advertising. In effect, the user can choose between the free software with the slight inconvenience of either pop-up ads or banners, or to pay for software free of advertising. So, users pay to use the software either with their money or with their time. This was the case until marketers noted three separate trends that pushed the development of adware into a different direction. Standard banner ads on the Internet were not delivering as well as expected (1% click-through was considered good) [22]. Targeted Internet advertising performed much better [21]. While office hours were dead-time for traditional advertising (radio, TV, etc.), many analyses showed a surprisingly high degree of personal Internet usage during office hours [21].

The conclusion was that targeted Internet advertising was a whole new opportunity for the marketing of products and services. All that was required was a method for monitoring users' behaviour. Once the adware was monitoring users' Internet usage and sending user details back to the advertiser, banners more suited to the users' preferences and personality were sent to the users in return. The addition of monitoring functionality turned adware into ad-/spyware, and the means to target advertising to interested parties accelerated. In reality, the data collected by ad-/spyware is often sent back to the marketing company, resulting in display of specific advertisements, pop-up ads, and installing toolbars showed when users visit specific web sites.

Spyware is usually designed with the same commercial intent as adware [20]. However, while most adware displays advertisements and commercial offers, spyware is designed with the intent to collect and transmit information about users. The general method is to distribute the users' Internet browsing history [22]. The idea behind this is that if you know what sites someone visits, you begin to get an idea of what that person wants, and may be persuaded to buy

[21]. Given the fact that more than 350 million users have downloaded KaZaa and supposedly also installed it on their computers [4], this enables for customised and personalised marketing campaigns to millions and millions of end users. Moreover, information-gathering processes have been implicated in the rising occurrence of unsolicited commercial e-mail messages (so called spam) on the Internet [6].

Besides the monitoring of Internet usage, there is an even greater danger, namely when spyware is set to collect additional and more sensitive personal information such as passwords, account details, private documents, e-mail addresses, credit card numbers, etc.

4.3 Experiment Design

4.3.1 Problem Domain

Programs designed with the purpose of locating and defeating ad-/spyware components are available throughout the Internet. Even so, these programs are not very refined. For instance, there is usually no linking between the identified ad-/spyware processes inside the computers and the corresponding servers outside, on the Internet. Also, there is no anti-ad-/spyware program that analyses what data content is being transmitted to other third parties on the Internet. So, even when using existing software, it is difficult do keep track of what is going on inside the computer, and what nodes outside it that obtain user-oriented information. As a consequence, Internet browsing records and/or credit card numbers could easily be distributed without the user's consent or knowledge.

In this light, the overall research problem for this paper was to explore the nature and occurrence of privacy-invasive software included in file-sharing tools used over P2P networks. On an experiment level, the research problem was divided into the following subquestions:

- What ad-/spyware programs can be found in file-sharing tools?
- What is the content and format of network data generated as a result of ad-/spyware programs involved in Internet communication?
- What is the extent of network traffic generated by such programs?

Even though there may be numerous components bundled with the installation of file-sharing tools, it is primarily the programs engaged in Internet communication that are of interest to us. There are two reasons for this. First, without this delimitation, the experiment data would be too comprehensive to grasp. Second, for ad-/spyware programs to leak personal information, they must be involved in communication over the Internet. This is of course particularly interesting from a privacy perspective.

Throughout this paper, we use the word ad-/spyware as a synonym for both adware and spyware. In general, both adware and spyware are namely considered to be privacy-invasive software. Also, since they typically are closely intervened with each other, and more or less perform similar actions it is problematic to separate adware from spyware [22].

4.3.2 Instrumentation and Execution

The experiment sample consists of the five most downloaded file-sharing tools [4]. The tools are, in order, the standard, freeware versions of KaZaa, iMesh, Morpheus, LimeWire and BearShare. Also, to be sure that the experiment results were derived from the installed file-sharing tools, we set up a reference computer, which was identical to the other work stations, i.e., the same configuration, but with no file-sharing tool installed. The experiment was executed in January 2004 as one consecutive session that lasted three days. This time range was chosen, because we wanted to avoid getting excessive data quantities, but at the same time be able to capture reliable results.

The experiment was carried out in a lab environment on PC work stations equally connected to the Internet through a NAT gateway. We used OpenBSD's packet filter to deny any inbound network requests, which allowed us to protect the work stations from external threats. The packet filter also helped in reducing the network traffic and in doing so, resulting in less data to analyse. By not downloading or sharing any content in the file-sharing tools we further reduced the amount of network data generated. All incoming and outgoing network traffic of the local computer's network interface were dumped into a file using Winpcap.

Hardware were equivalent for all work stations, which also contained byte identical installations of both the operating system

Microsoft Windows 2000 and program applications[2]. In order to reflect work stations in use, they were all set to browse the Internet according to a predefined schedule containing the 100 most visited web sites in the world [1]. This was done through an automatic surf program. Also, ten identical searches (e.g., "lord of the ring", "star wars", and "britney") were carried out in each of the file-sharing tools, but no files were downloaded. In the end of the experiment, several anti-ad-/spyware programs[3] were used to locate any known ad-/spyware programs previously installed.

Binding network communication to programs is a key feature in the experiment. For allowing continuous monitoring and logging of processes and their use of sockets, we developed a program in C++, which was based on Openport. We chose not to use any Win32 firewalls claiming to support outbound filtering on application level for two reasons. First, they fail in allowing real outbound filtering per application, and there are a number of programs capable of penetrating these fake protections [14, 17]. Second, we have no detailed knowledge in the internal workings of such firewalls and therefore cannot foresee what to expect from them. Finally, it should be emphasised that there exist ways for a malicious program to send network data undetected by the monitoring application, due to the architecture of Windows.

4.3.3 Data Analysis

After having performed the experiment, we compiled the data results and set to identify all programs that were bundled with each file-sharing tool. This data was provided by our own process-to-network mapping program in cooperation with the selected anti-ad-/spyware programs. We then isolated the operating system related programs found on the reference work station, since they were established as harmless. Next, we reduced all benign programs handling file-exchange tasks. Remaining were a set of programs that were not related to either the operating system or file-exchange tasks. Further, by using the results from the anti-ad-/spyware tools, we divided the set of programs into two subsets, namely known ad-/spyware programs and unknown programs. The nature of these unknown programs was analysed based on their corresponding net-

2. These configuration properties were enabled through a self-developed disc cloning system based on standard FreeBSD components.
3. For a detailed list of the programs used, see Appendix of this thesis.

work traffic. Also, in some cases we needed additional information and thus turned to Internet resources. Based on this analysis, the remaining ad-/spyware programs were located. In the final step, we divided the retrieved set of ad-/spyware programs into two subsets, namely those involved in Internet communication and those that were not. This analysis was founded on the data from our process-to-network mapping program. In effect, the results from the program analysis lead to a classification of programs as either ad-/spyware programs, system programs or unknown programs.

All data analysis was done in a Unix environment. The data was analysed and filtered using standard Unix programs such as sed, awk, sort, uniq and grep. Much of the analysis was automated using shell scripts and where this could not be done small programs in C were created. To analyse and filter network data, the program Ethereal was used.

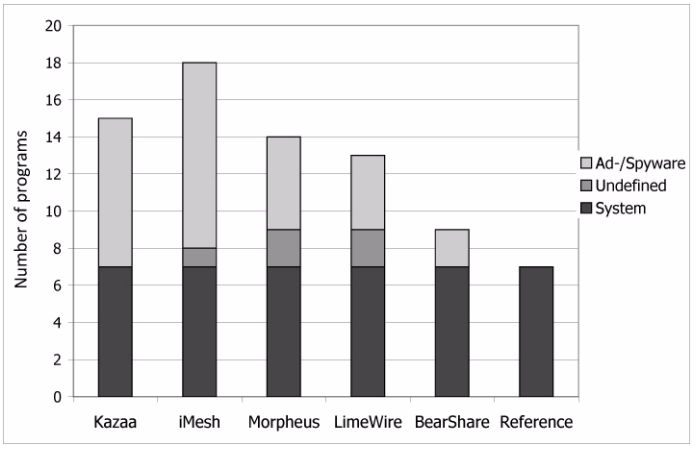

Figure 4.1 Amount of programs in the experiment sample.

In addition, we wanted to see if the corresponding servers were known ad-/spyware servers. Therefore, an effort to map the server names that were involved in Internet communication with a blacklist specifying known ad-/spyware servers [10] was also undertaken.

4.4 Experiment Results and Analysis

4.4.1 Ad-/Spyware Programs in File-Sharing Tools

According to the results, several programs were located for each file-sharing tool (see Figure 4.1). Of these programs, we identified 10 ad-/spyware programs for iMesh, and eight for KaZaa. Interestingly, these two file-sharing tools were among the two most popular ones [4]. The rates for the other file-sharing tools were five for Morpheus, four for LimeWire and two for BearShare. Also, iMesh, Morpheus and LimeWire contained programs that we were unable to define. However, these programs were all involved in Internet communication.

Table 4.1 Identified ad-/spyware programs.

Name	Host	Adware	Spyware	Download	Internet
BroadcastPC	M	x	x	x	X
KeenValue	K	x	x	X	X
Morpehus	M	X	x	X	X
BargainBuddy	I, K	x	x	x	
TopMoxie	L, M	x	x	x	
Cydoor	I, K	x	x		X
Gator	I, K	X	x		X
SaveNow	B	X	X		X
BonziBuddy	L	x	x		
Web3000	I	x	x		
ShopAtHomeSelect	I		X	X	X
WebHancer	K		x	x	
BrilliantDigital	K	x		X	X
MoneyMaker	L, M	X		X	X
Claria	I, K	x			X
iMesh	I	x			X
WeatherCast	B	x			X
CasinoOnNet	L	x			
MyBar	I, K, M	x			
New.Net	I			X	X
FavoriteMan	I			x	

We discovered that all of the file-sharing tools contained ad-/spyware programs that communicated with the Internet. KaZaa and iMesh included a relatively high amount of such programs. Even so, the anti-ad-/spyware tools defined several other ad-/spyware programs also installed on the computers. Although this was the case, these programs did not communicate with servers on the Internet during the experiment session.

In Table 4.1, a detailed list of the retrieved ad-/spyware components can be found. As can be seen, the ad-/spyware components were divided into "Adware" respectively "Spyware" based on their actions. Also, we included a category entitled "Download" because some of the ad-/spyware programs included functionality that allowed further software and/or updates to be downloaded and installed on the computers. In addition, programs involved in Internet communication are specified in the category called "Internet". In the column entitled "Host", the five file-sharing tools utilised as carriers of ad-/spyware are listed[4]. In the cases where the empirical results could confirm the recognised view shared by anti-ad-/spyware tools and Internet resources, the x-markers in the table are declared with bolded capital letters.

One reason to why we could not confirm that every ad-/spyware program was involved in Internet communication was that so called Browser Helper Objects (BHO) were installed in Internet Explorer. Malicious BHOs infiltrate the web browser with the intent to access all data generated by Internet Explorer in order to spy on the user and transmit user behaviour to third parties [20]. Such BHOs typically gain the same privileges as its host (i.e., Internet Explorer), which endorse them to penetrate personal firewalls. This means that any possible ad-/spyware traffic distributed via BHOs is highly problematic to detect since it may very well be ordinary browser traffic. In Table 4.1, we also included two programs, New.Net and FavoriteMan, even though they were not classified as neither adware nor spyware. However, they allowed for installation of further software, which may be malicious.

4.4.2 The Extent of Network Traffic

The results showed that a significant amount of network traffic was generated, although there was no exchange of files between the file-sharing tools and other peers on the Internet (see Figure 4.2). In that light, the amount of network traffic generated in this experiment can be seen as a minimum rate to be expected when running file-sharing tools. Notably, installing Morpheus and LimeWire resulted in a relatively high traffic quote, both when it came to incoming as well as outgoing traffic. On the contrary, iMesh, who

4. In the category entitled "Host", K is for KaZaa, I for iMesh, M for Morpheus, L for LimeWire and B is for BearShare.

also had the largest quantity of bundled programs, represented the least amount of network traffic.

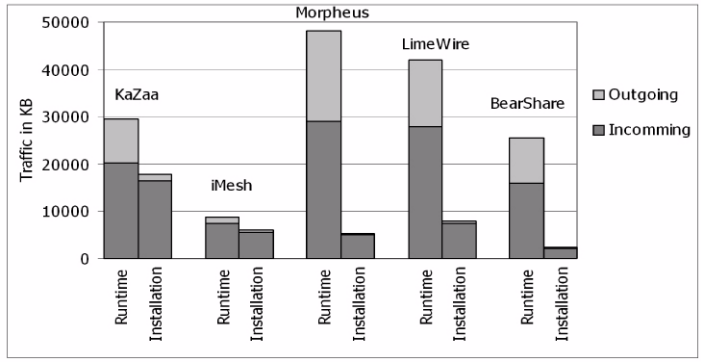

Figure 4.2 Network data traffic.

In Figure 4.2, we included compilations of network traffic for both the installation process and the runtime part per file-sharing tool. In the cases of Morpheus, LimeWire and BearShare, a considerable amount of network activity was generated after the installation. For KaZaa, a significant quantity of network traffic was caused during the installation. In comparison, iMesh produced a notably limited size of network traffic, both during and after installation.

Furthermore, the results suggested a diversity in Internet communication. This is shown in that programs in the file-sharing tools communicated with several different servers on the Internet. Although Morpheus did not contain a particularly great number of bundled programs, it generated notably much network traffic. In reflection, Morpheus communicated with the largest amount of Internet servers, whereas the rates for the other file-sharing tools were in a relatively low accordance with each other. In addition, the results substantiated that most of the invoked servers had domain names. Overall, each of the file-sharing tools contained programs that communicated with known ad-/spyware servers from the specified blacklist [10].

4.4.3 The Contents of Network Traffic

The outgoing network data was overall problematic to analyse and understand. In most cases the data was not readable, meaning that it was either encrypted or in a format not graspable. This is also an explanation to why we could confirm only two spyware programs

(see Table 4.1). Although most traffic data was not in clear text, we were able to extract and interpret some of the contents. We discovered that sensitive data such as information about the user (e.g., user name), geographical details (e.g., zip code, region and country) and Internet browsing history records were sent from identified ad-/spyware components to several servers on the Internet. Also, there were other types of information that were transmitted, for example, machine ID, details about program versions, operating system, etc.

According to the results, one spyware program (ShopAtHomeSelect) was found in the iMesh file-sharing tool. In the experiment, that program transmitted traffic measurement reports and Internet browsing history records to invoked servers on the Internet. Also, in BearShare, one spyware program (SaveNow) transmitted data such as Internet history scores and user-specific information.

The experiment results also reveal one of the methods for ad-/spyware programs to transmit user and/or work station data. In the BearShare tool, the information that was fed into the file-sharing software by the user was re-distributed within the tool to one or numerous ad-/spyware programs (SaveNow and WeatherCast) that transmitted the information to servers called upon. This method makes it difficult to map various program components to the actual file-sharing activity. Also, it undermines the ability to control what software objects are useful and legitimate in relation to the redundant or privacy-invasive programs that clog down the computers, systems and networks.

The analysis of the contents of the incoming network traffic was more problematic to conduct than in the case of outgoing traffic. Foremost, because the data quantity was both comprehensive and widespread. Since our focus was on privacy-invasive software, the outgoing traffic content was the most interesting so the efforts were mainly put into that. This, in combination, with vast quantities of incoming network data made it difficult to confirm adware recognised by the anti-ad-/spyware tools and Internet resources. Also, the same discussion concerning the occurrence of BHOs would apply for the unconfirmed adware. However, in the retrieved incoming data, a few interesting results were found.

The retrieved adware programs performed activities such as displaying commercial ads, causing browser banners and pop-ups. In particular, Morpheus and LimeWire proved to contain adware pro-

grams that generated much incoming data traffic. In LimeWire, results showed that lists of Internet sites and new programs were retrieved from the Internet by the adware MoneyMaker. In Morpheus, the P2P program itself downloaded and displayed ads and banners.

4.5 Discussion

With the occurrence of ad-/spyware technology in file-sharing tools, the monitoring of Internet usage has become a common feature. Today, most ad-/spyware programs gather and transmit data such as Internet browsing history records to third parties. That type of information can be correlated to a user and thus employed for marketing purposes.

The experiment has shown that all of the investigated file-sharing tools contained ad-/spyware programs. The ad-/spyware programs that operated inside the computers had an open connection to several Internet servers during the entire experimental session. We know that content-sensitive information was sent, but we may only guess the full extent of information harvesting, because most packets were not sent in clear text. Even though we saw no example of highly sensitive personal information, such as passwords and keystrokes, were transmitted by the ad/spyware programs in the experiment, we cannot be sure that these activities were not happening. Spyware may collect and transmit genuinely sensitive information about users such as, e.g., account details, private documents, e-mail addresses, and credit card numbers. The information is secretly sent back to numerous servers owned by companies that make a profit on these activities. Although it is problematic to elaborate on the business ethics of these companies, the occurrence of ad-/spyware programs are reasons enough to question this behaviour. In addition, ad-/spyware programs are responsible for all kinds of unwanted actions. Besides invasion of privacy, they can make the system unstable, degrade system performance, create scores of copies of itself to make removal difficult, and act as security holes in the system.

The actions performed by ad-/spyware programs are approaching the operations of a virus. Since users install them on voluntary basis, the distribution part is taken care of by the file-sharing tools. This makes ad-/spyware programs function like a slowly moving

virus without the distribution mechanisms usually otherwise included. The general method for a virus is to infect as many nodes as possible on the network in the shortest amount of time, so it can cause as much damage as conceivable before it gets caught by the anti-virus companies. Ad-/spyware, on the other hand, may operate in the background in such a relatively low speed that it is difficult to detect. Therefore, the consequences may be just as dire as with a regular virus. In addition, the purpose of ad-/spyware may not be to destroy or delete data on the work stations, but to gather and transmit veritably sensitive user information. An additional complicating factor is that anti-virus software companies do not usually define ad-/spyware as virus, since it is not designed to cause destruction. Overall, the nature of ad-/spyware substantiates the notion that malicious actions launched on computers and networks get more and more available, diversified and intelligent, rendering in that security is extensively problematic to uphold.

Ad-/spyware enables for the spreading of e-mail addresses that may result in the receiving of spam. Due to the construction of ad-/spyware, it may collect information that concerns other parties than only the work station user. For example, information such as telephone numbers and e-mail addresses to business contacts and friends stored on the desktop can be gathered and distributed by ad-/spyware. In the context that ad-/spyware usually is designed with the purpose of conveying commercial information to as many users as possible, not only the local user may be exposed to negative consequences of ad-/spyware. In other words, the business contacts and friends may be the subjects of ad-/spyware effects such as, e.g., receiving unsolicited commercial e-mail messages. This means that even though my computer may be secure, a breached computer owned by a network neighbour can cause me harm. So, the security of a neighbour very much becomes my own concern.

Besides security issues, ad-/spyware creates intrusion to privacy. An inconvenience commonly argued is that ad-/spyware programs display commercial messages based on the retrieval of personal information fetched without the explicit consent of the users. Even though the offers of these advertising campaigns may be in the interest of some users, there is a fine line between what users in general regard as useful information and what is an intrusion to personal privacy. One thought is that the more personalised the offers get, the more likely users are to regard them as privacy invaders. If so, what happens when users are presented with advertisements in such an extent that they hardly are able to distinguish the

possibly serious offers from all the offers. If users ignore marketing messages, there is evidently a great risk for the success of consumer-based e-commerce.

A second privacy concern is the spreading of content that the ad-/spyware distributor did not intend for. One example of this would be a malicious actor that gained control of ad-/spyware servers, and broadcasted offensive unsolicited messages (e.g., adult material, political messages and/or smearing campaigns, etc.) to a great number of users. Although users may consider regular commercial ads to be harmless, most people react negatively upon frequently receiving repulsive pictures and texts. This suffices for that the ad-/spyware providers need to take their own security with great seriousness. If they lose control of their servers, the damage may be devastating. This could be even more devastating if the ad-/spyware program updates on the company servers were replaced with malicious software. In effect, real and destructive malware (e.g., viruses, Trojans, and worms) could be spread to vast groups of ad-/spyware hosts.

4.6 Conclusions

The experiment has shown that all of the investigated file-sharing tools contained ad-/spyware programs. The ad-/spyware programs operating inside the computers had an open connection where the information was secretly sent back to numerous servers owned by companies that make a profit on these activities. Measurements suggested that the carriers of ad-/spyware, file-sharing tools, generated a significant amount of network traffic, even when not exchanging files. The presence of ad-/spyware programs and the network traffic that they generate contribute in over-consumption of system and network capacity.

Ad-/spyware is acting like a slowly moving virus, installed on a voluntary basis, with hidden properties problematic to detect and remove. The payload of ad-/spyware may not be to destroy or delete data on the work stations, but to gather and transmit veritably sensitive user information. The distribution part is taken care of by the file-sharing tools with an additional complicating factor; antivirus software companies do not usually define ad-/spyware as virus, since it is not designed to cause destruction.

The nature of ad-/spyware may lead to that not only host users are affected. Ad-/spyware may gather and distribute the details of business contacts and friends resulting in negative consequences to other parties than the infected desktop owner. This means that even though my computer may be secure, a breached computer owned by a network neighbour can cause me harm. So, the security of a neighbour very much becomes my own concern.

Furthermore, the occurrence of ad-/spyware can render in that privacy-invasive messages may be distributed and displayed to large amounts of users. Exposure to messages not chosen by the user, or collection and transmission of user information are two key privacy concerns. In this way, users' right to control what, how and when information about themselves is communicated to other parties is almost non-existing. In conclusion, the nature of ad-/spyware programs ignore users' right to be let alone. The increasing presence of hidden and bundled ad-/spyware programs in combination with the absence of proper anti-ad/spyware tools are therefore not beneficial for the development of a secure and stable use of the Internet.

4.7 References

[1] Alexa Web Search, http://www.alexa.com, 2010-03-11.

[2] California Senate Assembly Bill 1386, United States of America, 2003., http://info.sen.ca.gov/pub/01-02/bill/sen/sb_1351-1400/sb_1386_bill_20020926_chaptered.html, 2010-03-11.

[3] M. Caloyannides, "Privacy vs. Information Technology", in *IEEE Security & Privacy*, Volume 1, Issue 1, pp. 100-103, 2003.

[4] C|Net Download.com., http://www.download.com/, 2010-03-11.

[5] "Directive on Privacy and Electronic Communications", Directive 2002/58/EC of the European Parliament and of the Council of 12 July 2002 concerning the processing of personal data and the protection of privacy in the electronic communications sector, 2002.

[6] "Emerging Internet Threats Survey 2003", commissioned by Websense International, Ltd., February, 2003., http://www.websense.com/company/news/research/Emerging Threats2003EMEA-de.pdf, 2004-04-27.

[7] S. Fischer-Hübner, "Privacy in the Global Information Society", in *IT-Security and Privacy – Design and Use of Privacy-Enhancing Security*

Mechanisms, Lecture Notes in Computer Science LNCS 1958, Springer-Verlag, Berlin Germany, 2000.

[8] S. Garfinkel, "*Database Nation: The Death of Privacy in the 21st Century*", O'Reilly & Associates, Inc., Sebastopol CA, 2001.

[9] E. Grenier, "Computers and Privacy: A Proposal for Self-Regulation", in *Proceedings of the First ACM Symposium on Problems in the Optimization of Data Communications Systems*, ACM Press, New York NY, 1969.

[10] Gorilla Design Studio: The Hosts Files, http://www.accs-net.com/hosts/, 2010-03-11.

[11] M. McCardle, "How Spyware Fits into Defence in Depth", SANS Reading Room, SANS Institute, 2003., http://www.sans.org/rr/papers/index.php?id=905, 2010-03-11.

[12] A. Oram, "*Peer-To-Peer: Harnessing the benefits of a Disruptive Technology*", O'Reilly & Associates, Inc., Sebastopol CA, 2001.

[13] T. Otsuka, and A. Onozawa, "Personal Information Market: Toward a Secure and Efficient Trade of Privacy", in *Proceedings of the First International Conference on Human Society and the Internet*, Lecture Notes in Computer Science LNCS 2105, Springer-Verlag, Berlin Germany, 2001.

[14] Outbound, http://www.hackbusters.net/ob.html, 2004-04-27.

[15] L. Palen, and P. Dourish, "Unpacking Privacy for a Networked World", in *Proceedings of the ACM Conference on Human Factors in Computing Systems*, ACM Press, New York NY, 2003.

[16] B. Robertsson, "Five Major Categories of Spyware", in Consumer WebWatch, October 21, USA, 2002., http://www.consumerwebwatch.org/dynamic/privacy-investigations-categories-spy.cfm, 2010-03-11.

[17] Robin Keir's FireHole, http://keir.net/firehole.html, 2010-03-11.

[18] D. Schoder, and K. Fischbach, "Peer-to-Peer (P2P) Computing", in *Proceedings of the 36th IEEE Hawaii International Conference on System Sciences*, IEEE Computer Society Press, Los Alamitos CA, 2003.

[19] C. Shapiro, and H. Varian, "*Information Rules: A Strategic Guide to the Networked Economy*", Harvard Business School Press, Boston MA, 1999.

[20] E. Skoudis, "*Malware – Fighting Malicious Code*", Prentice Hall PTR, Upper Saddle River NJ, 2004.

[21] J. Sterne, and A. Priore, "*E-Mail Marketing – Using E-Mail to Reach Your Target Audience and Build Customer Relationships*", John Wiley & Sons Inc., New York NY, 2000.

[22] K. Townsend, "Spyware, Adware, and Peer-to-Peer Networks: The Hidden Threat to Corporate Security" (technical white paper), Pest-Patrol, 2003, http://www.pestpatrol.com, 2010-03-11.

[23] S.D. Warren, and L.D. Brandeis, "The Right to Privacy", in *Harvard Law Review*, No. 5, pp. 193-220, 1890-91.

PUBLICATION

2

Exploring Spyware Effects

9th Nordic Workshop on Secure IT Systems (NordSec04), 2004

Martin Boldt, Andreas Jacobsson and Bengt Carlsson

In this paper, we discuss various types of spyware programs, their behaviour, how they typically infect computers, and the propagation of new varieties of spyware programs. In two experiments, we investigate the occurrence and impact of spyware programs found in popular P2P applications. Based on the findings from the empirical investigations, we try to lift the perspective to a more general view on spyware deriving from the theory of (virtual) network effects. In a model, we categorize in what ways spyware might decrease the utility of belonging to a large virtual network. Here, the baseline is that spyware programs intrude systems and networks, but since they profit from user data they also intrude user privacy. In the model, the intrusions are classified as moderate, severe or disastrous. We found that spyware has the potential to overthrow the positive aspects of belonging to a large network, and network owners should therefore be very careful about permitting such programs in applications and on networks.

5.1 Introduction

During recent years, the world has seen the introduction of peer-to-peer (P2P) systems. P2P technology provides several beneficial solutions like, e.g., file-sharing, grid computing, web services, groupware and instant messaging (IM) [7]. P2P refers to a technology which enables two peers or more to collaborate in a network of equals [7 , 10]. This may be done by using information and communication systems that are not depending on central coordination. P2P technology was first widely deployed and popularized by file-sharing applications such as KaZaa and IM tools like ICQ.

Even though there are several benefits with belonging to a large virtual network such as a P2P file-sharing network, the rising occurrence of malicious software (malware) may seriously impact the positive utility of using P2P applications. Usually, only the positive effects that increase utility are emphasized when discussing participation in large networks [5]. One example is the theory of virtual network[1] effects. Network effects are usually described as when the value of a product to one user depends on how many other users there are [11]. Often, utility of the system is proportional to the aggregate amount of resources that the participants are willing to put together. On information technologies, users generally benefit from utilising a popular format, system or application [11]. Typically, technologies subject to strong network effects tend to exhibit long lead times until a critical mass of users is obtained [5]. Then, explosive growth is followed. From the perspective of a network owner, a large network may help to create a strategic advantage useful for competition and growth purposes [1]. From the perspective of a network user, the larger the network is, the more valuable it will be to participants and users [1].

There are two kinds of feedback from network effects: positive and negative [11]. Positive feedback can be explained in that when a person joins a network, the network gets bigger and better, to everyone's benefit. However, large networks may also be exposed to negative feedback, which bring about significant risks and severe consequences for all of the network nodes. Therefore, negative feedback may decrease the utility of belonging to that network. To

1. A virtual network describes a network of users bound together by a certain standard or technology, and where the exchange of information is the foundation for any information transaction. One example is the Internet.

large networks, such as P2P file-sharing networks, there could be numerous examples of applications (e.g., malware), which contribute in creating negative effects that impact network utility. However, in this paper, we focus on one of these applications, namely spyware.

There are many different kinds of spyware, and hundreds of such programs exist throughout the Internet today [9]. Spyware programming is a relatively new computing phenomenon. Although there is no precise definition, the term "spyware" is typically used to refer to a category of software that, from a user's perspective, covertly gathers information about a computer's use and relays that information back to a third party. In this paper, we use the term spyware in conformity with this common usage. However, in 5.2, we look into and discuss some of the current views on the concept of spyware.

Even though most people are aware of spyware, it seems that the research community has spent limited effort on understanding the nature and extent of the spyware problem. However, so far there have been some initial research attempts (see for example [4 , 9 , 17]) of which this paper is an additional effort. On the other hand, most network practitioners and experts agree that spyware is a real problem with increasingly negative effects. One example of this view is derived from the Emerging Internet Threats Survey 2003 [3], which states that one in three companies have detected spyware on their systems, while 60% consider spyware to be a growing and future threat. Also, 70% of the companies consider that file-sharing over P2P networks is creating an open door into their organisation. Another example is an investigation made by Earthlink (one of the major American ISPs) [13]. Earthlink set to measure the occurrence of spyware on more than 2 million computers connected to their network. A total number of 12.1 million different spyware types were detected. Out of these, Trojan horses and system monitors approached 700 000 instances, and the remaining 11.4 million instances were classified as adware. Also, experts suggest that spyware infect up to 90% of all Internet-connected computers [13].

In summary, spyware is a problem that should be taken seriously, because it may have the potential to threaten the utility of belonging to a large virtual network. In this paper, we focus on exploring the effects of spyware programs that are bundled with several P2P applications. The aim is to investigate the implications on system capacity, network bandwidth, security and privacy. Besides introduc-

ing results from empirical investigations, we also discuss the network effects of spyware.

The paper is organised as follows. First, we give an introduction to spyware, in which we discuss the various kinds of spyware programs, their behaviour, how they typically infect computers, and the proliferation of new varieties of spyware. Next, we investigate the occurrence and impact of spyware programs found in popular P2P applications. In 5.4, we discuss the findings from the experiments and also try to lift the perspective to a more general view on spyware deriving from the theory of virtual network effects. In the end, conclusions are presented.

5.2 On spyware

5.2.1 The Background of Spyware

As stated by [9], spyware exists because information has value. The idea with spyware is simply to fetch information. If a software developer can get revenue from advertisers, the owner can afford to make the software available for free. The developer is paid, and the user gets free, quality software. Usually, the developer provides two versions of the software, one for which the user has to pay a fee in order to receive, and one version that is freeware supported by advertising. In these cases, free software typically includes programs set to display advertisements and offers to the users (that is; adware). Therefore, the user can choose between the free software with the slight inconvenience of either pop-up ads or banners, or to pay for software free of advertising. So, users pay to use the software either with their money or with their time.

This method of including rather benign adware when developing and distributing free software was common until marketers noted three separate trends that pushed the development of adware into a different direction. The background was that:

- standard banner ads on the Internet were not delivering as well as expected (1% click-trough was considered good) [15],
- targeted Internet advertising typically performed much better [14], and

- while office hours were dead-time for traditional advertising (radio, TV, etc.), many analyses showed a surprisingly high degree of personal Internet usage during office hours [14].

The conclusion was that targeted Internet advertising was a whole new opportunity for the marketing of products and services. All that was required was a method for monitoring users' behaviour. So, once the adware was monitoring users' Internet usage and sending user details back to the advertiser, banners more suited to the users' preferences and personality was sent to the users in return. The addition of monitoring functionality turned adware into spyware, and the means to target advertising to interested parties accelerated [15]. In reality, the data collected by spyware is often sent back to the marketing company, resulting in display of specific advertisements, pop-up ads, and installing toolbars showed when users visit specific web sites. In this sense, spyware programs became technologies used to fetch valuable customer information.

5.2.2 The Operations of Spyware

The usual method for a spyware is to run secretly in the background of the users' computers [6]. The reason for this concealing of processes is commonly argued as that it would hardly be acceptable if, e.g., free file-sharing software kept stopping to ask the user if he or she was ready to fetch a new banner or a pop-up window [15]. Therefore, the client/server routine of spyware is normally executed in the background. In practice, there would be nothing wrong with spyware running in the background provided that the users know that it is happening, what data is being transmitted, and that they have agreed to the process as part of the conditions for obtaining the freeware. However, most users are unaware of that they have software on their computers that tracks and reports on their Internet usage. Typically, a spyware program covertly gathers user information and spreads it without the user's knowledge of it. Once installed, the spyware monitors, e.g., user activity on the Internet and transmits that information in the background to third parties, such as advertising companies. In reality, spyware run constantly, even when their carrier program, e.g., a file-sharing tool, has been terminated.

A more or less legal grey area is exploited by the spyware actors, since they in most program licenses specify that information may be gathered for corporate purposes. However, the usual model is to

collect more information than have been asked for [15]. Besides this, most license agreements are formulated in such a way that they are extensively hard for users to understand.

5.2.3 The Types of Spyware

There are many different kinds of spyware. For instance, one of the leading anti-spyware tools, PestPatrol, has a record of over 1400 instances of spyware published on their web site [8]. In order to make the spyware domain more graspable, we present the following classes of spyware. This classification is in conformity with a recently published study on measurement and analysis of spyware [9], although when presented here, the order of spyware types ranges from minimum to maximum user impact:

- Cookies and web bugs: Cookies are small pieces of state stored on individual clients' on behalf of web servers. Cookies can only be retrieved by the web site that initially stored them. However, because many sites use the same advertisement provider, these providers can potentially track the behaviour of users across many Internet sites. Web bugs are usually described as invisible images embedded on Internet pages used for locating a connection between an end user and a specific web site. They are related to cookies in that advertisement networks often make contracts with web sites to place such bugs on their pages. Cookies and web bugs are purely passive forms of spyware, they contain no code of their own. Instead they rely on existing web browser functions.
- Adware: Adware is a more benign form of spybot (see below). Adware is a category of software that displays advertisements tuned to the user's current activity. Although most "genuine" adware programs only display commercial content, some hybrids are involved in reporting the aggregate or anonymized user behaviour to a third party, as described in 5.2.1.
- Tracks: A "track" is a generic name for information recorded by an operating system or application about actions that the user has performed. Examples of tracks include lists of recently visited web sites, web searches, web form input, lists of recently opened files, and programs maintained by operating systems. Although a track is typically not harmful on its own, tracks can be mined by malicious programs, and in the wrong context it can tell a great deal about a user.

- Browser hijackers: Hijackers attempt to change a user's Internet browser settings to modify their start page, search functionality, or other browser settings. Hijackers, which predominantly affect Windows operating systems, may use one of several mechanisms to achieve their goal: install a browser extension (called a "browser helper object"), modify Windows registry entries, or directly manipulate and/or replace browser preference files. Browser hijackers are also known to replace content on web sites with such promoted by the spyware authors [12].
- Spybots: Spybots are the prototypes of spyware. A spybot monitors a user's behaviour, collects logs of activity and transmits them to third parties. Examples of collected information include fields typed in web forms, lists of e-mail addresses to be harvested as spam targets, and lists of visited URLs. A spybot may be installed as a browser helper object, it may exist as a DLL on the host computer, or it may run as a separate program launched whenever the host operating system boots.
- System monitors: System monitors record various actions on computer systems. This ability makes them powerful administration tools for compiling system diagnostics. However, if misused system monitors become serious threats to user privacy. Keyloggers are a group of system monitors commonly involved in spyware activities. Keyloggers were originally designed to record all keystrokes of users in order to find passwords, credit card numbers, and other sensitive information.
- Malware: Malware is a set of instructions that run on a computer and make the system do something that an attacker wants it to do [12]. Malware refers to a variety of malicious software that includes viruses, worms, and Trojan horses. Spyware is one form of malware, but as will be discussed later on, spyware may also include instructions for downloading and installing, e.g., a virus.

Spyware succeeds because some of today's desktop operating systems make spyware simple to build and install [9]. Many instances of spyware have the ability to self-update, or automatically download new versions of themselves to the local host. Self-updating allows spyware authors to introduce new functions over time, but it may also be used to evade anti-spyware tools by avoiding specific signatures contained within the tools' signature databases using polymorphic techniques.

5.2.4 On the Implications of Spyware

Spyware may occupy resources of the computer that it infects or alter the functions of existing applications on the affected computer to the benefit of a third party. In that sense, spyware poses several risks. One commonly argued is that spyware compromises a user's privacy by transmitting information about that user's behaviour [4]. Even so, a spyware can also detract from the usability and stability of the computing environment of the user [9]. In addition, a spyware has the ability to introduce new security vulnerabilities to the infected host by downloading software updates [6]. Due to that spyware is widespread, such vulnerabilities put numerous amounts of computers at risk.

To summarize, the occurrence of spyware programs raise a real and growing threat to Internet usage in many aspects, and to other interested parties than only to end users. Four categories frequently argued on this topic are [3 , 6 , 15]:

- Consumption of system capacity: Spyware is often designed to be secretly loaded at system startup, and to partly run hidden in the background. Due to that it is not unusual for users to have many different instances of spyware running covertly simultaneously, the cumulative effect on the system's processing capacity can be dramatic.
- Consumption of bandwidth: The continual data traffic with gathering of new pop-ups and banner ads, and delivery of user data can have an imperative and costly effect on both private and corporate bandwidth.
- Security issues: Spyware covertly transmits user information back to the advertisement server, implying that since this is done in a covert manner, there is no way to be certain of exactly what data is being transmitted. Even though spyware, in its purest form, is a threat to privacy rather than security, some spyware programs have begun to act like Trojan horses. Most security experts would agree that the existence of spyware is incompatible with the concept of a secure system.
- Privacy issues: The fact that spyware operates with gathering and transmitting user information secretly in the background, and/or displays ads and commercial offers that the user did not by him-/herself chose to view, makes it highly privacy-invasive. Also, spyware enables for the spreading of e-mail addresses that

may result in the receiving of unsolicited commercial e-mail (so called spam).

5.3 Experiments

We have developed a method for identifying and analysing spyware components and their behaviour on their host systems. This method has been used in several experiments (see, e.g., [4 , 17]). In this section, we present the method applied in two experiments. Thereafter, a compilation of the experiment results is given.

5.3.1 Method

The method is tightly coupled with our security laboratory. Mainly because our experiment method is based on state preservation of computer systems, which can be provided due to the computer architecture of the security laboratory[2]. By storing the initial baseline state of a system it is later possible to conclude what changes occurred with regards to this baseline. In practice, this means that we store the state of a base system before installing any application carrying spyware components. Afterwards, it is possible to conclude any changes between the two. By also capturing all network data sent and binding that traffic to the corresponding program, we can correlate network data to specific programs. It is also possible to include measurements of, e.g., CPU and network utilization during the experiments.

By using this method, all systems that are measured consist of identical hardware and network setups. Therefore, operating systems and their applications are bitwise identical for all subjects in the experiment sample. This suffices for the generation of reliable results. In order to be sure that the results are derived from a certain spyware, we included a “clean” reference computer in the experiment.

Since file-sharing tools are notoriously known for bundling spyware, we used such applications in both of the experiments. In this context, it should be pointed out that no file-sharing activity took

2. Throughout the experiments, we used 2.8Ghz Pentium 4 computers with 512MB primary memory.

place in terms of sharing or downloading any content on the P2P networks. Our examination was limited to software versions released between January and May 2004, and as such, our observations and results might not hold for other versions. Also, we used an Internet surfing program that automatically simulated a user visiting 100 preconfigured Internet sites. This was an attempt to trigger any spyware to either leak this information to third parties or to hijack the web sessions. In order to identify and locate the spyware programs, several anti-spyware tools were used.

5.3.1.1 Experiment 1

In the first experiment, we investigated the occurrence and operations of five popular file-sharing tools[3]. More specifically, we examined spyware programs that were bundled with the file-sharing tools, the content and format of network data caused by spyware involved in Internet communication, and the extent of network traffic generated by such programs. Even though there may be numerous components bundled with the installation of file-sharing tools, it was primarily the programs engaged in Internet communication that were of interest to us. There are two reasons for this. First, without this delimitation, the experiment data would be too comprehensive to grasp. Second, for spyware programs to leak user data, they must be involved in communication over the Internet.

5.3.1.2 Experiment 2

In the second experiment, we set to explore the effects in terms of resource usage that spyware bring about on a local system. A major problem introduced when setting up such an investigation involve how to choose the experiment sample. What we wanted was a program instance that was free of spyware and another instance (of the same program) that included spyware. Unfortunately it is almost impossible to remove only the spyware components and still have a working version of the original program since such components are very tightly coupled with the original program. We came to an acceptable solution by selecting KaZaa and KaZaa Lite K++ as the two subjects in the experiment sample. KaZaa Lite K++ is an instance of KaZaa where all spyware components have been removed by an independent group that reverse-engineered the original KaZaa program, carefully excluding or disabling all bundled

3. The file-sharing tools were the standard (free) versions of BearShare, iMesh, KaZaa, LimeWire, and Morpheus.

components not solely used for file-sharing purposes. By using these two KaZaa versions, it was possible to subtract the resource utilization of KaZaa Lite K++ from the utilization of the original KaZaa and thereby receive a measurement of resources used by the spyware programs.

5.3.2 Results and Analysis

5.3.2.3 Experiment 1

A detailed list of the identified spyware programs is presented in Table 5.1 After having analysed the captured data, we concluded that all file-sharing tools contained spyware.

The two main carriers of spyware were iMesh and KaZaa (they included ten respectively eight programs each). The rates for the remaining file-sharing tools were five for Morpheus, four for LimeWire, and two for BearShare. In addition to these findings, we also discovered that all file-sharing tools contained spyware that were involved in Internet communication.

Table 5.1 Identified spyware programs.

Name	Host	Adware	Spybot	Download	Internet
BroadcastPC	M	x	x	x	X
KeenValue	K	x	x	X	X
Morpehus	M	X	x	X	X
BargainBuddy	I, K	x	x	x	
TopMoxie	L, M	x	x	x	
Cydoor	I, K	x	x		X
Gator	I, K	X	x		X
SaveNow	B	X	X		X
BonziBuddy	L	x	x		
Web3000	I	x	x		
ShopAtHomeSelect	I		X	X	X
WebHancer	K		x	x	
BrilliantDigital	K	x		X	X
MoneyMaker	L, M	X		X	X
Claria	I, K	x			X
iMesh	I	x			X
WeatherCast	B	x			X
CasinoOnNet	L	x			
MyBar	I, K, M	x			
New.Net	I			X	X
FavoriteMan	I			x	

As can be seen in Table 5.1, the retrieved spyware components were divided into “Adware” and “Spybot” based on their operations. We

also included a category called "Download" because some of the components allowed for further software and/or updates to be downloaded and installed. In this category, examples such as hijackers and malware potentially could be included by the spyware distributors. In addition, all programs involved in any form of Internet communication were specified in a category called "Internet". Finally, the category entitled "Host" specifies which file-sharing tool that carried what spyware[4]. In the cases where our empirical results could confirm the view shared by anti-spyware tools, the markers in the Table 5.1 are declared with bolded capital letters.

When analysing the outgoing network communication from the spyware components, we discovered that most of this traffic was not sent in clear text. This means that the transactions between the spyware components and their corresponding servers were either obfuscated or encrypted. This is also an explanation to why we were able to only identify two genuine spybot components. Since most traffic was sent in non-clear text, we could not really measure the extent to which such traffic was broadcasted. However, we did manage to identify some network traffic sent to spyware servers on the Internet that included, e.g., web sites visited, zip codes, country, and information about programs and operating system versions on the local host. In example, one of the spybot programs (ShopAtHomeSelect) that was found bundled with the iMesh file-sharing tool transmitted Internet browsing history records to several invoked servers on the Internet. The Internet records that were transmitted could be correlated to the web sites included in our preconfigured web surfing program.

5.3.2.4 Experiment 2

A compilation of the results from the resource utilization measurement can be seen in Table 5.2. The measurements indicate that if KaZaa was installed, the rates for consumption of both system capacity (categories 1-4) and network bandwidth (categories 5-7) were significantly higher. This can be explained in that the spyware programs included in KaZaa affected both consumption of system capacity and network bandwidth. The high amount of network traffic was due to that the spyware components invoked numerous spyware servers on the Internet for the gathering of ads, pop-ups and banners. The accumulated local storage of collected commercial

4. B is for BearShare, I for iMesh, K is for KaZaa, L for LimeWire, and M for Morpheus.

messages can have noticeable consequences on hard drive size, which also was the case for KaZaa.

Table 5.2 **Resource utilisation measurements.**

	KaZaa Lite K++	KaZaa	Alteration
1. CPU usage (in %)	0.015	0.48	0.47
2. RAM usage (in %)	1.4	14	12.6
3. Addition of new files	50	780	730
4. Change in hard disk size (in MB)	8.6	46	37.4
5. Amount of network traffic (in MB)	0.6	29	28.4
6. No. of programs involved in Internet communication	1	11	10
7. No. of corresponding servers	60	349	289
8. No. of spyware programs installed	0	8	8

In Table 5.2, the measurements for the reference subject is subtracted from the file-sharing tools. The column entitled "Alteration" is represented by the difference between KaZaa and KaZaa Lite K++, that is; the spyware resource usage. Interestingly, three computer resources were significantly affected by the installation of spyware. In the first category of Table 5.2, the occurrence of spyware had a measurable effect on CPU usage, KaZaa used 32 times more CPU capacity than KaZaa Lite K++. In category two, a significant difference was measured where the installation of KaZaa resulted in a ten times, or 65MB, increase of RAM usage. Finally, spyware programs had an imperative effect on the amount of network traffic generated by the file-sharing tools. More specifically, there was a 48 times augmentation of network traffic due to the spyware programs bundled with KaZaa. So, in contrast to KaZaa, installing a clean file-sharing tool (i.e., KaZaa Lite K++) caused marginal impact to system consumption and network bandwidth. However, due to the occurrence of spyware in file-sharing tools (see Table 5.1), users with several such applications installed will, as a result of aggregate spyware activity, suffer from a continuous system and network degrading.

5.4 Discussion

Based on the findings in 5.3, we can conclude that spyware programs exist, that they engage themselves in Internet communica-

tion, that they transmit user data, and that their existence have a negative impact on system and network capacity. Since we also can conclude that spyware programs are bundled with highly popular file-sharing tools[5], we can make out that spyware in accumulation may have a negative impact on networks and systems. In fact, the occurrence of spyware might decrease the overall utility of belonging to a large network such as a P2P file-sharing network. Thus, it might be relevant to elaborate on the theory of negative network effects to see whether spyware programs can threaten a large network.

In a model (Table 5.3), we specify in what ways spyware might decrease the utility of belonging to a large virtual network. The baseline is that spyware programs intrude systems and networks, but since they profit from user data they also intrude user privacy. In the model, the intrusions are classified as moderate, severe and disastrous.

Table 5.3 Spyware Effects.

	User	Computer	Network
Moderate	Commercially salable data	Consumption of capacity	Consumption of bandwidth
Severe	Personal data	Inferior code dissemination	Malware distribution
Disastrous	Critical data	Takeover	Breakdown

On user effects, some P2P providers include spyware in order to maximise profitability. Spyware may collect user data (such as e-mail addresses for spam distribution, surf records for personalised advertisement exposure, etc.) for commercial purposes. At present, spyware programs as such are rather benign, but cause problems to user privacy. In general, privacy is the right of individuals to control the collection and use of information about themselves [16]. This means that users should be able to decide for themselves, when, how, and to what extent information about them is communicated to others. Even though the user data exemplified in this category may not be that sensitive, spyware programs ignore user rights, and must therefore be considered privacy-invasive.

A more troublesome concern is the distribution of personal data, such as personal details (name, gender, hobby, etc.), e-mail conver-

5. As an example, there are more than 350 million downloaded instances of KaZaa [2].

sation, and chat records. This may be the result of spyware techniques intended not only for commercial purposes, but also motivated by malicious intentions. Although, such spyware programs may not be that wide-spread today, a technological platform for these kinds of operations is available. This mean that although the probability of being infected by such a spyware is very low, the consequences may be devastating.

A third view would be if the spyware program updates on the servers were replaced with, e.g., keyloggers. In effect, harmful software could be distributed to vast groups of P2P tool users with the purpose of transmitting personally critical information such as financial data, private encryption keys, digital certificates or passwords. In reflection, financial threats from spyware programs may signify disastrous outcomes to vast groups of users.

In the experiments, we established a correlation between the presence of spyware programs and the consumption of computer capacity. Typically, spyware components utilised significant amounts of system resources, rendering in that computer resources were exploited in a larger extent than would otherwise be necessary. In accumulation, spyware operations degrade system capacity.

Also, it is problematic to comment on the quality of the code in the spyware programs, since the software requirements that have been used during the development process are left out in obscurity. The result can be that possibly inferior code is executed locally, which may have a negative influence on the entire system (i.e., not only to security). For example, as an effect of executing insufficient code, a system may lack performance or crash with, e.g., loss of important data as a result. In addition to this, software vulnerabilities may be exploited by malicious persons when breaking into a system, or when infecting it with destructive software (e.g., viruses).

As an utmost consequence, spyware programs deprive control over the system from the system owner. In effect, the installation of spyware programs may render in further installations of malware such as viruses and/or Trojans. Local services that are based on defect code and executed without the knowledge of the system owner are vulnerable to exploits, which may allow malicious actors to gain access over the computer. This is a disastrous situation because a takeover of system control affects both the local system and the surrounding network. A conquered system can be used as a platform for further distribution of malware.

At the network level, spyware operations in accumulation may contribute in network congestion. On one hand, the effects are unnecessary costs for network maintenance and expansion. On the other hand, network performance may be degraded. In either case, it is the network users that in the long run bear the costs.

The operations performed by spyware programs are approaching the operations of a virus with both a distribution and a payload part. Since users install, e.g., file-sharing tools that contain spyware programs on a voluntary basis, the distribution part is taken care of by the users themselves. This makes spyware programs function like a slowly moving virus without the typical distribution mechanisms usually otherwise included. The general method for a virus is to infect as many nodes as possible on the network in the shortest amount of time, so it can cause as much damage as conceivable before it gets caught by the anti-virus companies. Spyware, on the other hand, may operate in such a relatively low speed that it is difficult to detect. Therefore, the consequences may be just as dire as with a regular virus. The payload of a spyware is usually not to destroy or delete data, but to gather and transmit user information, which could be veritably sensitive. An additional complicating factor is that anti-virus companies do not generally define spyware as virus, since it does not typically include the ability to autonomously replicate itself. Overall, the nature of spyware substantiates the notion that malicious actions launched on computers and networks get more and more available, diversified and "intelligent", rendering in that security is extensively problematic to uphold.

In theory, even a large network such as a P2P network may suffer an ultimate breakdown if it is continuously flooded with data. Should spyware programs continue to increase in number and to be more and more technologically refined, a network breakdown might be a final step. Although, in reality, this is not a plausible outcome. Nonetheless, if security and privacy risks are increasing as a result of being part of a P2P network, the positive value of using an application and thus belonging to that network will likely decrease. If users should experience that a threshold value (where the negative effects overthrow the positive aspects of using the application) is overstepped, then they will restrain from utilising that network. However, the experiment results indicate that even though spyware programs operate over P2P file-sharing networks, their effects are thus far rather modest. At least when it comes to system and network consumption. On the other hand, spyware programs that invade user privacy must be looked upon seriously. Spyware tech-

nologies mainly involved in gathering user data have a true value potential for marketers and advertisers. If these privacy-invasive activities should continue to evolve, there might be a great risk that spyware will be engaged in more malicious activities than simply fetching anonymized user/work station data. If so, that can lead to negative network effects and thereby cause a network to become less useful.

Hidden spyware components permit distribution of privacy-invasive information and security breaches within the network. Due to the construction of spyware, it may collect information that concerns other parties than only the work station user, e.g., telephone numbers and e-mail addresses to business contacts and friends stored on the desktop. In the context that spyware usually is designed with the purpose of conveying commercial information to as many users as possible, not only the local user may be exposed to negative feedback of spyware. As well, the business contacts and friends may be the subjects of network contamination, e.g., receiving vast amounts of spam or other unsolicited content.

With the continuous escalation of spyware programs and the refinement of spyware technologies, network availability may be degraded to such an extent that ordinary transactions are overthrown by obscure malware traffic. A disastrous situation may occur where a network is seriously overloaded by malware distributed by computerised systems that are controlled by malicious actors. In conclusion, spyware activity may persuade users to abandon networks.

5.5 Conclusions

Based on the discussions of spyware and on the findings from the two experiments, we can conclude that spyware have a negative effect on computer security and user privacy. We have also found that a subsequent development of spyware technologies in combination with a continuous increase in spyware distribution will affect system and network capacity. A disastrous situation may occur if a network is seriously overloaded by different types of spyware distributed by computerised systems that are controlled by malicious actors. Then, the risk is a network breakdown. However, a more plausible outcome may be that users will abandon the network before that happens. In effect, spyware has the potential to overthrow the positive aspects of belonging to a large network, and net-

work owners should therefore be very careful about permitting such programs in applications and on networks.

5.6 References

[1] S.-Y. Choi, D.O. Stahl, and A.B. Winston, "*The Economics of Electronic Commerce",* Macmillan Technical Publishing, Indianapolis IN, 1997.

[2] C|Net Download.com., http://www.download.com/, 2010-03-11.

[3] "Emerging Internet Threats Survey 2003", commissioned by Websense International Ltd., February, 2003. http://www.websense.com/company/news/research/Emerging_Threats_2003_EMEA-de.pdf, 2010-03-11.

[4] A. Jacobsson, M. Boldt, and B. Carlsson, "Privacy-Invasive Software in File-Sharing Tools", in *Proceedings of the 18th IFIP World Computer Congress,* Toulouse France, 2004.

[5] M.L. Katz, and C. Shapiro, "Systems Competition and Network Effects", in *Journal of Economic Perspectives 8:93-115*, 1994.

[6] M. McCardle, "How Spyware Fits into Defence in Depth", SANS Reading Room, SANS Institute, 2003. http://www.sans.org/rr/papers/index.php?id=905, 2010-03-11.

[7] A. Oram, "*Peer-To-Peer: Harnessing the Benefits of a Disruptive Technology"*, United States of America: O'Reilly & Associates Inc., 2001.

[8] PestPatrol, http://research.pestpatrol.com/Lists/NewPests(Pest Counts).asp, 2010-03-11.

[9] S. Sariou, S.D. Gribble, and H.M. Levy, "Measurement and Analysis of Spyware in a University Environment", in *Proceedings of the ACM/USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, San Francisco CA, 2004.

[10] D. Schoder, and K. Fischbach, "Peer-to-Peer (P2P) Computing", in *Proceedings of the 36th IEEE Hawaii International Conference on System Sciences (HICSS'03)*, IEEE Computer Society Press, Los Alamitos CA, 2003.

[11] C. Shapiro, and H. Varian, *"Information Rules"*, HBS Press, Boston MA, 1999.

[12] E. Skoudis,*"Malware – Fighting Malicious Code"*, Prentice Hall PTR, Upper Saddle River NJ, 2004.

[13] "Spyaudit", commissioned by Earthlink Inc., http://www. earthlink.net/spyaudit/press/, 2010-03-11.

[14] J. Sterne, and A. Priore, *"E-Mail Marketing – Using E-Mail to Reach Your Target Audience and Build Customer Relationships"*, John Wiley & Sons Inc., New York NY, 2000.

[15] K. Townsend, "Spyware, Adware, and Peer-to-Peer Networks: The Hidden Threat to Corporate Security" (technical white paper), Pest-Patrol, 2003., http://www.pestpatrol.com/Whitepapers/PDFs/SpywareAdware P2P.pdf, 2010-03-11.

[16] A. Westin, "*Privacy and Freedom"*, Atheneum, New York NY, 1968.

[17] J. Wieslander, M. Boldt, and B. Carlsson, "Investigating Spyware on the Internet*"*, in *Proceedings of the Seventh Nordic Workshop on Secure IT Systems*, Gjövik Norway, 2003.

PUBLICATION

3

Analysing Countermeasures Against Privacy-Invasive Software

International Conference on Software Engineering Advances, 2006

Martin Boldt and Bengt Carlsson

User privacy is widely affected by the occurrence of privacy-invasive software (PIS) on the Internet. Various forms of countermeasures try to mitigate the negative effects caused by PIS. We use a computer forensic tool to evaluate an anti-spyware tool, with respect to found PIS over a four years period. Within the anti-spyware tool PIS was slowly identified, caused classification problems, and formerly classified PIS were sometimes excluded. Background information on both PIS and countermeasure techniques are also presented, followed by discussions on legal disputes between developers of PIS and vendors of countermeasures.

6.1 Introduction

Technology has revolutionized the way we collect, store and process information. With the help of information technology it is possible to accumulate huge data quantities for instant or later use. The fact that information (such as user interests) creates value to advertisers has given rise to a parasitic market, focusing on information

theft [21]. Software vendors take advantage of these achievements based on questionable commercial incentives when creating and distributing questionable software. Throughout this paper we group such software together under the term *privacy-invasive software* (PIS). *Adware* and *spyware* are the two most dominating types of PIS that are not adequately addressed by anti-virus programs [4]. Adware displays advertisements and commercial offers on users' systems while spyware covertly collect and then transmit privacy-invasive information to third parties [2]. However, the term spyware is also used at a higher abstraction level to include any software that users dislike [3, 13]. Unfortunately there does not exist any proper definition of this notion of the term. All this software are to various degree encapsulated inside the term PIS [6]. Our use of the concept of *privacy* lies within Warren and Brandies original definition, "the right to be let alone." [42]. Since this paper target user privacy in the context of software programs, we focus on the following three parts:

- software that covertly sneaks into systems, or
- deceives users about their business, or
- exists without any control from users.

Users' privacy are trespassed by PIS that covertly collect privacy-invasive information, present unsolicited contents, or secretly exchange requested contents with sponsored information. Such software covertly sneaks into systems and hide deep inside the core, out of reach from user control. By also excluding normal program removal routines, usually provided by the operating system, such software assure future prosperity. Locating and removing PIS are therefore associated with great cost, which is further increased since widely deployed protection mechanisms, such as anti-virus tools, do not adequately address these threats [4]. Earlier work has analysed the behaviour and impact that PIS have on users' computers, with regard to performance, privacy and security [4, 7, 25].

Privacy-invasive software could integrate themselves into systems either by utilizing available software vulnerabilities, or by deceiving the user into installing them, i.e. to target and deceive users to install, what they think is a useful piece of software [22, 33]. So, even in a context where software vulnerabilities are being exterminated and where accurate and sophisticated protection mechanisms exist, systems would still be susceptible to PIS. Techniques that allow for users to make informed decisions in advance on whether to install a certain software or not could mitigate this problem. One

such approach, based on certification of "privacy friendly software", has been developed by TRUSTe [41]. However, until such certifications are being commonly used we will have to adopt to the fact that only visiting the wrong site on the Internet could be equivalent with PIS infection [4, 31]. Once a single PIS component has gained access to a system this piece of software could be used as a gateway for additional PIS to be installed [25]. Which leaves the user with trespassed systems containing unsolicited harmful software, that result in a reduction of performance, stability, privacy and ultimately the security [7].

In this paper however, we use *computer forensic* tools and methods to evaluate the accuracy of PIS countermeasures. This paper also touch upon the evolution of PIS countermeasures and the legal disputes between developers of PIS and related countermeasures.

6.2 Countermeasures

In an attempt to stop, or at least mitigate, the PIS hazard a whole new group of software, called *anti-spyware*, or spyware removal tools, has emerged [23]. In an ongoing struggle between anti-virus companies and virus distributors, refined detection mechanisms have to fight more and more sophisticated viruses searching for competitive advantages over each other. Anti-spyware vendors face three major problems to solve.

1. The need to identify new and previously unknown types of PIS. This should be done in an environment of highly dynamic and evolving variety of PIS.
2. After successfully identifying a PIS component, any proper anti-spyware tool should remove the component and thereby bring the system closer to a previously uninfected state.
3. The anti-spyware tools' ability to safeguard user data and system components during the removal phase, i.e. to keep and protect legitimate files.

The most common technique used when countering PIS is the *signature based* identification which relies on a database holding signatures of known PIS. A signature captures unique properties of PIS, and could be thought of as the associated fingerprint. By comparing items in a system with the signatures in the database it is possible to identify already known PIS. However, as soon as a new PIS emerge,

anti-spyware vendors need to find it, produce a signature associated with the new threat, and finally distribute the new signature to the users. This method is widely used, despite the delay in protection; since it is possible to create software that automate the detection process.

An emerging trend is that PIS developers sue anti-spyware vendors for defamation and ruined business strategy, by classifying and treating their product as PIS. Some of these cases have escaped captivity to public attention and several ended up in court [39]. The most recent case involves the online marketing company "180Solutions" that sued firewall company "Zone Labs" for classifying their advertising client as spyware [48]. We believe that vendors of countermeasure tools need to be more accurate in their classification of PIS in the future, and that their decisions need to be based on solid evidence that hold for use in court. We also believe that those anti-virus vendors not addressing PIS are at less risk, since developers of malicious software, such as viruses or worms, will not sue the company because their actions are without a doubt illicit.

To separate vendors of legal marketing tools from developers of PIS a general agreement on what should be considered to be fair business practices, need to be established between developers of PIS and countermeasures [8, 32]. At least until such an agreement is reached, any cautious anti-spyware vendor should keep trustworthy evidence to back up their PIS classification decisions. Using a method designed to deliver such solid evidence will be of paramount importance for every company that classifies and treats software as privacy-invasive.

6.3 Computer Forensics

Individuals and companies rely on computers in their daily work and for doing personal duties such as online banking errands. Criminals take advantage of this fact by using computers when committing crimes. To investigate such crimes, law enforcement agencies rely on *computer forensics* [11]. Main steps in computer forensic investigations involve, identification and collection of evidence, data harvesting, data reduction, reorganizing and search of data, analysis of data and finally reporting. These steps constitute a formalized process that help investigators reach conclusions that are repeatable,

based on evidence, and as free as possible from errors. Anti-spyware vendors could benefit from this if PIS developers sue the vendor for ruining their business strategy when removing their tool [39]. If clear and stringent evidence together with proper handling of the evidence, could be presented to a court, it would assist the anti-spyware vendor in reaching a favourable outcome in the case.

One important principle in forensic science is *Locard's exchange principle* [27] which determines that anyone or anything, entering a crime scene takes something of the scene with them, and leaves something of them behind as they leave. This principle could also be applied in most computer settings; involving for instance PIS infections since these types of software leave tracks in both file-system and network communication. In Section 5 we discuss this in more detail.

To aid computer forensic investigators in the investigation process there exist both public domain and commercial tools. These tools allow investigators to analyse copies of whole systems, i.e. the investigator can see everything stored in a file-system. In our investigation we used a commercial tool called *Forensic Tool Kit* (FTK) which is developed by AccessData [1]. FTK has been thoroughly tested not to alter the evidence that is being investigated.

6.4 Investigation

In previous investigations of PIS we used a manual investigation method that is based on system state preservation [7, 25, 46]. By preserving the state of a system, together with complementing information (such as network traffic), it is later possible to retrieve a specific system state for analysis. During both the planning and execution of our experiment we had two main goals concerning the laboratory environment:

1. Preserve identical hardware and software configurations during all investigation steps.
2. Use default software configurations and all available security updates.

To preserve bit-wise identical system states we rely on the standard BSD Unix component *dd*. This allow us to serialize a whole system into a bit-wise identical *clone file*. Such a clone file is a snapshot of a system at a specific time. From such a clone file it is later possible to

restore a system and its state for analysis. Initially a snapshot of a "clean" system is created, this is regarded as the *baseline*. Such a baseline only includes the operating system and the tools used for experiment measurements. Next, an action of some kind is executed which result in infection of PIS. Such actions could be for instance, surfing to certain Web sites or installing a program bundled with PIS. Immediately after this action is performed another snapshot is taken. Depending on the experiment, additional snapshots could be created at certain intervals. Using snapshots allow investigators to track system-changes between the points in time when the snapshots were taken. For instance, to identify any system-changes that were introduced during the installation of software A, we need to conceal all system parts in the post-installation snapshot that are identical with the baseline. In some sense we remove the baseline from the post-install snapshot. Now, only system changes that occurred during installation of software A remains.

Our method detects any system deviation that has occurred between two points in time. Simultaneous data collection and analysis is avoided since the method has a clear separation between collection and analysis of data. The method force investigators to collect data only once, and later take the time needed to analyse this data. The level of detail in the data captured is very high which results in extensive data quantities that need to be handled. We address this problem by automating much of the structuring and refinement steps through custom-made software. However, this method cannot be fully automated since steps involving for instance data recognition and reduction rely on the skills of the investigator. Since the method cannot be fully automated it is considerably more resource demanding than automated signature based anti-spyware tools. But we believe that computer forensic tools could reduce this problem to an acceptable level.

In an experiment we used this method to analyse the accuracy of an anti-spyware tool in identifying PIS, bundled with three *peer-to-peer* (P2P) file-sharing tools over a four year period [20, 29].

In earlier experiments we have investigated 10 different anti-spyware tools but in this work we choose to instead evaluate a single tool over four years development instead. We choose to investigate an anti-spyware tool that is developed by Lavasoft which is called Ad-Aware. This specific tool was selected since it is the most downloaded anti-spyware tool from Download.com (October 2005) and

since we could locate versions of this tool for the years 2002-2005. The experiment used 13 identical physical computers holding four versions of the three most downloaded P2P file-sharing tools; iMesh, LimeWire, and Kazaa, together with one reference machine without any file-sharing tool installed. The versions of the three file-sharing tools were all from 2002 until 2005, and claimed to be free from any forms of spyware. Since all of the investigated file sharing tools were developed for the Windows platform our experiment were executed in a Windows 2000 environment. Windows XP could not be used since it was incompatible with earlier versions of LimeWire. Even though file-sharing is not restricted to the Microsoft Windows platform most problems concerning PIS are [35].

In the beginning of the experiment each of the 13 computers were identical and the system state was stored with a baseline snapshot. However, system deviations began as soon as the various file-sharing tools were installed. Directly after the installation process was completed a new system snapshot was created for each system. After this the systems were left to execute continuously for 72 hours. During this time all computers were left uninterrupted, except for an automated Web surfing program that was set to simulate a user visiting a number of company Web sites, such as Amazon and Apple. This was done in an attempt to trigger any dormant PIS lurking in the system. In the end of the 72 hour execution new snapshots were taken for each system. As a final step we installed and executed six versions (from 2000 until 2005) of Ad-Aware on each of the 13 computers. The result of these Ad-Aware executions was stored for later analysis.

To analyse the data gathered from the experiment we mainly used FTK, which offers efficient techniques that are highly useful for an investigator. Such techniques are for instance pre-indexation of data, and a known-file-filter. Pre-indexation means that indexes all data once, when the evidence is loaded. Later, during data harvesting, this result in instant search results from all data in the investigated system. The known file filter is a technique based on cryptographic hash values that allows FTK to recognize and label files as, e.g. non tampered system files which could be concealed to the investigator. FTK also includes ways to inspect and label files based on various properties, e.g. encryption, text, binary, or image. This allows for an investigator to highlight all encrypted files through one button.

Any PIS components identified by Ad-Aware were checked against the actual system which allowed us to identify numerous false-positives, reported by Ad-Aware. On some occasions different versions of Ad-Aware reported a single PIS by several names. We choose to report all such PIS with the latest used name. Further, we investigated all added or modified programs and components, except for the file-sharing executables. To identify if any of the files missed by Ad-Aware should be considered PIS we used static analysis based on file properties such as filename, hash value, identification tags, and strings located inside the binary program [19]. This information was then checked against two resources for classification [16, 36].

6.5 Results

In Table 6.1 the total number of PIS, cookies, registry keys and other components is measured as the difference between the clean system and a system "infected" by different versions of Kazaa, iMesh and LimeWire. Different versions including their release date of Ad-Aware, an anti-spyware programme, are used for the examination[1]. The shadowed part shows the added components found by a present or future version of Ad-Aware, i.e. the actual protection against certain version of the P2P programs.

Table 6.1 **Total number of added components for three P2P-programs (iMesh, LimeWire and KaZaa) measured by six different versions (3.5 to SE1.06) of Ad-Aware between 2002 and 2005.**

Ad-Aware	3.5 aug-00	5.5 jun-01	5.7 mar-02	6.0 mar-03	1.05 sep-04	1.06 nov-05
2002	8	59	183	278	912	638
2003	6	24	15	18	222	232
2004	11	34	38	34	218	221
2005	0	2	5	4	142	128

In general, present versions of Ad-Aware find more components than older. 2002, 2003 and 2004 versions of the P2P-programs show a many times increase of added components. Ad-Aware 2005 reported fewer components on average than the 2004 version of the program.

Figure 6.1 presents the amount of PIS programs, registry keys, and other traces that are being injected into a system when a P2P-program is installed and been running for 72 hours. Registry keys are

1. We used one instance of each version of Ad-Aware. In Table 6.1 the release month of this instance is given.

not complete programs but are used by PIS during execution. The most dominating group of traces consist of registry keys followed by not specified traces, e.g. suspicious files and folders. The actual number of executable PIS is much lower compared to other traces. Kazaa 2002-2005 shows a large number of added components each year. iMesh shows a peak 2002 with progressive decreasing the years before and after, whereas LimeWire had very few added components outside the years 2002 and 2003.

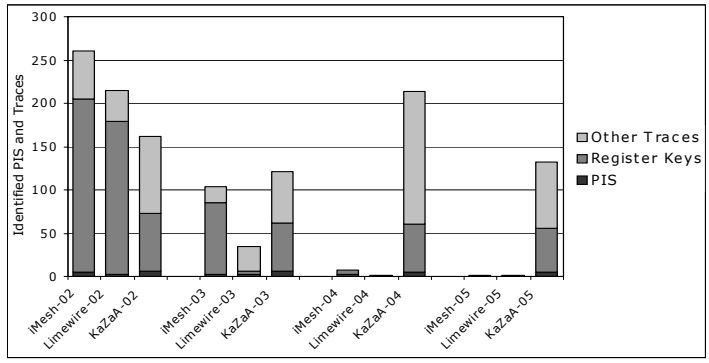

Figure 6.1 **Number of bundled PIS programs, registry keys, and suspicious files/ folders for iMesh, LimeWire and Kazaa reported by Ad-Aware over a four year period.**

In Table 6.2 all exclusive PIS programs found in Kazaa, iMesh and LimeWire are counted for different versions of Ad-Aware. Ad-Aware misleadingly reported some traces such as registry keys as fully functioning PIS. These false positives are presented as the numbers inside brackets in Table 6.2. The second column from the right presents the number of PIS found by either the manual forensic method or at least one version of Ad-Aware. PIS components detected by the manual method but missed by Ad-Aware are presented in the last column.

Table 6.2 **Number of PIS in three different P2P-programs (iMesh, LimeWire and Kazaa) measured by six different versions of Ad-Aware and our manual forensic method (FTK). Numbers in brackets indicate traces of PIS that misleadingly was reported by Ad-Aware as fully functioning PIS.**

Ad-Aware	aug-00	jun-01	mar-02	mar-03	sep-04	nov-05	FTK AdAw	FTK New
2002	1	3	3(2)	8(1)	8(2)	7(2)	11	4
2003	2	3	2(1)	2(2)	5(3)	5(3)	7	3
2004	2	3	2(1)	2(1)	4(1)	4(1)	5	3
2005	0	0	(1)	(1)	2(1)	3(1)	3	3

Most PIS programs were found in the 2002 version of the P2P programs with a total of 15 different programs. 11 of these programs were reported by Ad-Aware, but different versions reported a variable number. Ad-Aware prior to 2002 reported less PIS and later versions reported more, however not all of them. Besides not reporting all PIS, Ad-Aware contrarily also reported, in all three different, PIS which instead were only traces thereof, and therefore wrongly classified as functioning PIS. Our manual method found four additional PIS never reported by any version of Ad-Aware.

For the forthcoming years a similar interpretation of Table 6.2 shows that the number of PIS declines, especially for iMesh and LimeWire, but the number of unreported PIS programs are still about the same as for 2002.

Table 6.3 Total number of undiscovered PIS programs in three different P2P-programs (iMesh, LimeWire and Kazaa) measured by six different versions (3.5 to SE1.06) of Ad-Aware.

Ad-Aware	3.5 aug-00	5.5 jun-01	5.7 mar-02	6.0 mar-03	1.05 sep-04	1.06 nov-05
2002	14	12	11	7	4	4
2003	8	7	7	7	3	3
2004	6	5	5	5	3	3
2005	6	6	6	4	3	3

In Table 6.3 the earlier results of PIS found by Ad-Aware and the manual forensic method are presented as the failure numbers of Ad-Aware. This is the best possible result using all known versions of Ad-Aware, some PIS may in later versions be reclassified as harmless files. More recent versions of Ad-Aware (grey shadowed in Table 6.3) found a larger number of PIS than older versions. Sometimes, as for the P2P-tools from 2002, a delay exists in finding new PIS, i.e. later versions of Ad-Aware reported more PIS programs and traces. This delay lasted for the forthcoming two years.

Table 6.4 shows the 25 different PIS present in Kazaa, LimeWire and iMesh. In all 19 behaved as adware, 14 as spyware, 13 as hijackers that alter Web content, and two were able to independently download new programs.

Ad-Aware was able to find 18 out of 25 programs, or about 70% covering of PIS, but did not exclusively detect a certain PIS behaviour. Approximately 80% of all adware, 70% of all hijackers, 60% of all spyware, and 50% of the downloaders were detected by Ad-Aware.

Table 6.4 **Classification (adware, spyware, hijacker or downloader) of found PIS programs. In the host column K refer to Kazaa, L to LimeWire and I to iMesh. An X in the Ad-Aware column indicates that at least one of the investigated Ad-Aware versions found the PIS program.**

Name	Host	Adware	Spyware	Hijack	Download	Ad-Aware
AltnetBDE	K	X	X			X
BestOffers	K	X	X			X
BonziBuddy	L	X	X	X		X
Bullguard	K	X				X
Claria	I,K	X	X	X		X
CommonName	I	X		X		X
Cydoor	I,K,L	X		X		X
DownloadWare	K				X	X
eZula	I,L	X		X		X
FavoriteMan	I		X		X	
HotBar	K	X	X	X		
Instafinder	K			X		X
MarketScore	I		X			
MediaLoads	K	X	X	X		
MyWay Speedbar	I,K			X		
Need2Find	K	X	X	X		
NewDotNet	I,K	X	X	X		X
Nodopops	K	X	X			
PerfectNav	K	X		X		X
PromulGate	K	X				X
RX Toolbar	K	X	X			X
ShopAtHome	I		X			X
Stop-Sign AV	I	X				X
TopMoxie	L	X		X		X
WhenU	I,K	X	X			X

6.6 Discussion

Unlike viruses, PIS programs exist in a grey area between being legal (business facilitators) and being illegal, i.e. behave and/or being regarded as malicious software. Normally, a virus is rapidly identified, does not cause any classification problem, and once included in the anti-virus database it remains there. Ad-Aware, the investigated anti-spyware tool, was unsuccessful with respect to all three anti-virus qualities above, i.e. PIS was slowly identified, caused classification problems, and was sometimes excluded.

The first quality, speed of identification, compromises PIS that is not reported by Ad-Aware for certain version of the file sharing program. This could be due to that some PIS is not yet classified as PIS, i.e. they are detected but is not included into the signature database, or that PIS successfully conceal themselves from anti-spyware

tools. As was shown in Table 6.3 the failure numbers of Ad-Aware decreased over the time showing a gradual incorporation of new PIS into its database. It took one to two years for Ad-Aware to incorporate missing PIS in the database and there were still undetected programs. Compared to anti-virus programs this is too long time and with a remaining unacceptable failure number.

The second quality, classification consistency, suffers from the presence of false negatives and positives. Reclassification, unreported and undetected files may all be false negatives, i.e. PIS found during the forensic analysis but not reported or ignored by the anti-spyware tool. Ad-Aware found about 70% of all PIS and did not show any trend to exclusively favouring the detection of certain behaviour. Also, a lack of a deepened context analysis may influence the amount of false positives, i.e. warnings, generated by the anti-spyware tool that does not pose any risk at all. Ad-Aware did not distinguish between traces of PIS and executable programs.

The third quality, stability, was violated because executable program files, formerly by Ad-Aware classified as PIS, was later excluded. Three such programs, behaving as adware, spyware or hijackers were found. There was no obvious reason for reclassifying these programs because of more harmless actions. Instead there are different business considerations for anti-spyware tools compared to anti-virus tools, such as legal aspects of excluding third-part material.

In all, the 2005 version of Ad-Aware found 15 PIS out of 25 for the 2002-2005 versions of the three P2P tools. Also, later versions of P2P tools contained fewer PIS than older versions. So, the decrease in the number of PIS is probably not the result of more efficient countermeasures, but refined business strategies. Either a company tries to exclude its marketing program from the anti-spyware databases or choose another kind of marketing. Both strategies are found in the 2005 versions where iMesh and LimeWire excluded all PIS and Kazaa contained a bigger rate of undetected files.

We believe the failure from anti-spyware tools to deal with the three qualities above rely on both obsolete identification techniques, but also on the lack of a general agreement on what should be considered as privacy-invasive behaviour of software[2]. Without such an agreement it is a more arbitrary task to distinguish PIS from legitimate software than separating malicious software, such as virus and worms, from legitimate software. Since the inner workings of PIS

does not necessarily include any malicious behaviours, they rather include normal behaviour such as showing content on the screen (advertisements), it is not possible for countermeasures to only target PIS behaviour when distinguishing PIS from legitimate software. Instead they need to incorporate user consent when distinguishing between legitimate and illegitimate software. Without proper techniques that safeguard true informed user consent during installation, it is extremely hard (if not impossible) for any spyware countermeasure to accurately decide on what software to target since there exists no common guidelines to follow.

This fact combined with that the software that anti-spyware vendors classify as PIS are developed by companies that are ready to take legal actions if needed, pose a great risk for these vendors. This is a scenario most anti-virus vendors do not have to worry about when classifying and treating for instance a worm as malicious software. Both vendors of anti-spyware tools and marketing companies need to commonly establish where to draw the border between PIS and legitimate business facilitators. If such an agreement could be reached, both legitimate marketing companies and vendors of anti-spyware tools will benefit. Legitimate marketing vendors no longer need to be affected by decreased revenues since their advertising clients were wrongly classified as PIS, and anti-spyware companies face a lower risk of being sued by indignant marketing vendors. Additionally, every deceitful software developer creating PIS would be treated, rightfully, by the anti-spyware vendors.

If a general separation between privacy-invasive and legitimate software could be established it would be possible to certify software as "privacy friendly". Complementing such a certification with a short description on e.g. software behaviour, transmitted data, and removal routines it would be possible for users to make informed decisions on whether or not to install certain software. Such a service would provide users with an important tool that allow them to increase the amount of control they have over their systems and their digital security and privacy, on both home computers and mobile devices. TRUSTe has started one such promising approach called "Trusted Download Program" in which they will provide a guideline on how to distinguish legitimate software [41]. Based on this guideline they will publish a white-list of approved applications.

2. Despite we only used one anti spyware-tool, a lot of different versions during several years were investigated.

Any software vendor submit their software for certification must also enter into a contract with TRUSTe in which their software functionality is specified. Using these guidelines TRUSTe continuously evaluate the correctness and ongoing compliance of all certified software.

6.7 Conclusions

Identifying and removing PIS and keeping/protecting legitimate files are major problems to solve for anti-spyware vendors. The identification task is further complicated by the necessity to consider legal aspects which is a major distinction between anti-spyware and anti-virus tools. This paper evaluated the accuracy of a leading anti-spyware tool called Ad-Aware which uses signatures to counteract PIS. The effectiveness of comparable versions of Ad-Aware was correlated against a manual method using a forensic tool comparing a "clean" system with the system infected by added components from the file sharing tools.

The investigated anti-spyware program failed to report all PIS programs, marked earlier discovered PIS as ordinary programs, or wrongly classified traces of PIS as functioning PIS. There was also a palpably reduction of PIS programs included in later versions of two out of three file sharing programs. The manual forensic method managed to find all added executable files and to sort out traces of PIS.

Unlike viruses, PIS programs exist in a grey area between being business facilitators and being regarded as malicious software. Compared to the more established anti-virus approach, the investigated anti-spyware tool suffered from three quality attributes; rapid identification, classification consistency and conformity violations. This imply that the signature-based approaches, which is de facto standard in anti-virus tools, is not ideal when targeting PIS. We believe the failure from anti-spyware tools to deal with the three qualities above not only rely on obsolete identification techniques, but also on the lack of a general agreement on what should be considered as privacy-invasive behaviour of software.

It is of most importance to develop routines that allow users to make informed decisions during the software installation process, on whether to install a certain software or not. Until such routines

and mechanisms are being widely deployed, computer users risk being victims of systematic privacy invasions in their digital environment from questionable actors.

6.8 References

[1] AccessData Corporation, http://www.accessdata.com, 2010-03-11.

[2] W. Ames, *"Understanding Spyware: Risk and Response"*, in *IEEE Computer Society – IT Professional*, Vol. 6, Issue 5, 2004.

[3] Anti-Spyware Coalitions, http://www.antispywarecoalition.org, 2010-03-11.

[4] K. P. Arnett and M. B. Schmidt, "Busting the Ghost in the Machine", in *Communications of the ACM*, Vol. 48, Issue 8, 2005.

[5] Blue Coat Systems – Spyware Interceptor, http://www.bluecoat.com/products/interceptor/, 2010-03-11.

[6] M. Boldt, and B. Carlsson, "Privacy-Invasive Software and Preventive Mechanisms", in *Proceedings of the International Conference on Systems and Network Communications*, Papeete Tahiti, 2006.

[7] M. Boldt, B. Carlsson, and A. Jacobsson, "Exploring Spyware Effects*"*, in *Proceedings of the Eighth Nordic Workshop on Secure IT Systems*, Helsinki Finland, 2004.

[8] J. Bruce, *"Defining Rules for Acceptable Adware", in the Fifteenth Virus Bulletin International Conference (VB2005)*, Dublin Ireland, 2005.

[9] B. Carrier, "*File System Forensic Analysis"*, Addison-Wesley Professional, Upper Saddle River, NJ, 2005.

[10] H. Carvey, "*Windows Forensics and Incident Recovery"*, Addison-Wesley, Upper Saddle River, NJ, 2005.

[11] E. Casey, "*Digital Evidence and Computer Crime: Forensic Science and the Internet"*, Academic Press, London UK, 2004.

[12] D. M. Chess and S. R. White, "An Undetectable Computer Virus", in *Virus Bulletin Conference*, Orlando FL, 2000.

[13] E. Chien, "Techniques of Adware and Spyware", in *Fifteenth Virus Bulletin International Conference (VB2005)*, Ireland, 2005.

[14] F. Cohen, "Computational Aspects of Computer Viruses", in *Computers & Security*, Vol. 8, Issue 4, San-Fransisco CA,1989.

[15] F. Cohen, "Computer Viruses – Theory and Experiments", in *IFIP-Sec 84*, Toronto Canada, 1984.

[16] Computer Associates Spyware Information Center, http://www3.ca.com/securityadvisor/pest/, 2010-03-11.

[17] Download.com, http://www.download.com, 2010-03-11.

[18] "Emerging Internet Threats Survey 2003", commissioned by Websense International Ltd., February, 2003. http://netpartners.com/company/news/research/Emerging_Threats_2003_EMEA.pdf, 2010-03-11.

[19] D. Farmer, and W. Venema, *"Forensic Discovery"*, Addison-Wesley, Upper Saddle River NJ, 2004.

[20] A. Froemmel, "Dangers And Containment Of P2P Utilities On A Corporate Network", SANS Reading Room, SANS Institute, 2003. http://www.giac.org/certified_professionals/practicals/gsec/2828.php, 2010-03-11.

[21] S. Görling, "An Introduction to the Parasite Economy", in EICAR 2004, Luxemburg, 2004.

[22] G. Hoglund, and G. McGraw, *"Exploiting Software – How To Break Code"*, Addison-Wesley, Boston MA, 2004.

[23] L. Hunter, "*Stopping Spyware"*, Addison-Wesley, Upper Saddle River, NJ, 2005.

[24] A. Jacobsson, "Exploring Privacy Risks in Information Networks", in *Blekinge Institute of Technology Licentiate Thesis Series No. 2004:11*, Sweden, 2004.

[25] A. Jacobsson, M. Boldt, and B. Carlsson, "Privacy-Invasive Software in File-Sharing Tools", in *Proceedings of the 18th IFIP World Computer Congress,* Toulouse France, 2004.

[26] Lavasoft, http://www.lavasoft.com, 2010-03-11.

[27] E. Locard, *"L'enquête criminelle et les méthodes scientifiques"*, Flammarion, Paris France, 1920.

[28] R. Martin, "Spy vs. spy", in *Fortune Small Business*, Vol. 14, No. 4, 2004.

[29] A. Oram, "*Peer-To-Peer: Harnessing the Benefits of a Disruptive Technology"*, United States of America: O'Reilly & Associates Inc., 2001.

[30] S. Sariou, S.D. Gribble, and H.M. Levy, "Measurement and Analysis of Spyware in a University Environment", in *Proceedings of the ACM/USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, San Francisco CA, 2004.

[31] S. Shukla and F. Fui-Hoon Nah, "Web Browsing and Spyware Intrusion", in *Communications of the ACM*, Vol. 48, Issue 8, 2005.

[32] J. Sipior, B.T. Ward, and G.R. Roselli, *"A United States Perspective on the Ethical and Legal Issues of Spyware"*, in *The Seventh International Conference on Electronic Commerce (ICEC2005), Xian China, 2005*.

[33] E. Skoudis, *"Malware – Fighting Malicious Code"*, Prentice Hall PTR, Upper Saddle River NJ, 2004.

[34] Spyaudit, commissioned by Earthlink Inc., http://www.earthlink.net/spyaudit/press/, 2010-03-11.

[35] Spyware is Windows-only, http://www.securityfocus.com/news/9696/, 2010-03-11.

[36] SpywareGuide.com, http://www.spywareguide.com, 2010-03-11.

[37] Stay Safe Online, *"AOL/NCSA Online Safety Study – December 2005"*, http://www.staysafeonline.org/pdf/safety_study_2005.pdf, 2010-03-11.

[38] J. Sterne and A. Priore, *"E-Mail Marketing – Using E-Mail to Reach Your Target Audience and Build Customer Relationships"*, John Wiley & Sons Inc., New York NY, 2000.

[39] Threats Against Spyware Detectors, Removers, and Critics, http://www.benedelman.org/spyware/threats/, 2010-03-11.

[40] K. Townsend, "Spyware, Adware, and Peer-to-Peer Networks: The Hidden Threat to Corporate Security" (technical white paper), Pest-Patrol, 2003., http://www.moorecomputing.net/SpywareAdwareP2P.pdf, 2010-03-11.

[41] TRUSTe – Make Privacy Your Choice, http://www.truste.com, 2010-03-11.

[42] S.D. Warren and L.D. Drandeis, *"The Right to Privacy*", in *Harward Law Revien*, Vol. 4, Issue 5, 1890.

[43] Webroot Software, *"State of Spyware – Q3 2005*", http://www.webroot.com/resources/, 2010-03-11.

[44] Webroot Software – Phileas, http://www.webroot.com/resources/phileas/, 2010-03-11.

[45] A. Westin, *"Privacy and Freedom"*, Atheneum, New York NY, 1968.

[46] J. Wieslander, M. Boldt, and B. Carlsson, "Investigating Spyware on the Internet*"*, in *Proceedings of the Seventh Nordic Workshop on Secure IT Systems*, Gjövik Norway, 2003.

[47] X. Zhang, *"What Do Consumers Really Know About Spyware?"*, in *Communications of the ACM*, Vol. 48, Issue 8, 2005.

[48] Zone Labs Sued Over Spyware Classification, http://www.securityfocus.com/brief/68, 2010-03-11.

PUBLICATION

4

Preventing Privacy-Invasive Software using Online Reputations

Lecture Notes in Computer Science (LNCS), Volume 4721, 2007

Martin Boldt, Bengt Carlsson, Tobias Larsson and Niklas Lindén

Privacy-invasive software, loosely labelled spyware, is an increasingly common problem for today's computer users, one to which there is no absolute cure. Most privacy-invasive software is positioned in a legal greyzone, as the user accepts the malicious behaviour when agreeing to the End User License Agreement. This paper proposes the use of a specialized reputation system to gather and share information regarding software behaviour between community users. A client application helps guide the user at the point of executing software on the local computer, displaying other users' feedback about the expected behaviour of the software. We discuss important aspects to consider when constructing such a system, and propose possible solutions. Based on the observations made, we implemented a client/server based proof-of-concept tool, which allowed us to demonstrate how such a system would work. We also compare this solution to other, more conventional, protection methods such as anti-virus and anti-spyware software.

7.1 Introduction

Our society is continuously moving in an increasingly more computerized direction where software has a central role [15, 28]. Because of this development computer users are in need of more aiding mechanisms to help them distinguishing legitimate software from its questionable counterparts. Without such mechanisms we will experience a gradual increase in the negative consequences resulted by such software, affecting more and more of our daily lives by involving for instance mobile devices and media centres. Sources indicate that well over 80% of all home PCs and more than 30% of all corporate PCs connected to the Internet are infected by questionable software, often labelled *spyware* [32, 37]. Affected computer owners are not aware of the fact that their computer is infected with spyware since they rely entirely on anti-virus software and firewalls to protect them. However, anti-virus software does not focus on spyware, but rather on more malicious software types, such as viruses, worms and Trojan horses [2].

Although some spyware programs might be malicious, many are considered to be legitimate software distributed by highly profitable companies that are gathering information about its users, showing targeted ads, sending user behaviour patterns, visited websites and similar, storing them for an unknown period of time as user profiles in central databases. Spyware are often in a legal greyzone since they normally inform the users of their actions, but often in such a format that it is unrealistic to believe that normal computer users will read and understand the provided information. The *End User License Agreement* (EULA) that the user has to agree on before using or installing the software are often written in a legal format, sometimes spanning well over 5000 words, and most users choose to proceed without actually studying it, giving his or her consent to whatever might be stated in the EULA, i.e. anything the software developer wants[7, 14, 31].

There are numerous ongoing projects and attempts to produce effective countermeasures for removing spyware [22, 25, 33]. However, this requires a classification of some software as "harmful to the user" which is legally problematic. The main reason for this is because the information regarding system behaviour is stated in the license agreement that the user already has accepted, which could lead to law suits [14, 34]. Such legal disputes have already proved to be costly for anti-spyware software companies [29]. As a result of

this, they may be forced to remove certain software from their list of targeted spyware to avoid future legal actions, and hence deliver an incomplete product to their customers, being unable to correctly classify some software as privacy-invasive.

As the problem of spyware is widely spread, and no complete protection is available, there is a need for ways to better inform the user about the software he or she uses, while still not classifying it as "harmful to the user" and hence risking law suits. There are numerous well-known and popular websites based on the concept of letting users grade different services, applications, shops, and similar e.g., Flixster, IMDb.com, and Pricerunner [12, 18, 26]. The main concept is to help guide other consumers to, for example, find the best store to shop at and to avoid pitfalls and unethical sellers [38]. We have combined this concept with a client software that helps guide the user whenever a program is about to execute on his computer, by showing other users rating and comments of the particular software. Larsson and Lindén implemented this idea into a proof-of-concept tool during their masters thesis work[1] [21]. In this system the users are asked to rate their most frequently used software, by grading it between 1 and 10. In return they are given access to aggregated ratings for all software in the reputation system. By using the knowledge from previous users it is possible for new users to reach more informed decisions when installing a specific software, i.e. allowing them to stop questionable software *before* it enters their computer. The proof-of-concept tool has found a group of continuous users, which has rendered in well over 2000 rated software programs in the reputation database.

7.1.1 Background and Related Work

The usage of the term spyware has become increasingly popular, both by users, media and software vendors [1, 30]. It has been defined as software that "track users' activities online and offline, provide targeted advertising, and/or engage in other types of activities that users describe as invasive or undesirable [8, 13]. This means that it has come to include all kinds of malicious software, ranging from software that displays advertisements based on user behaviour (adware) to Trojan key loggers, as well as actual spying software (spyware) [31]. A better term to use instead of spyware, would be

1. More information about the tools is available at http://www.softwareputation.com

privacy-invasive software (PIS). In an attempt to clarify the usage of this term, Boldt and Carlsson based their classification of privacy-invasive software on user's informed consent and negative user consequence, as shown in Table 7.1 [4, 5].

Table 7.1 Classification of privacy-invasive software with respect to user's informed consent (high, medium and low) and negative user.

	Tolerable Negative Consequences	Moderate Negative Consequences	Severe Negative Consequences
High Consent	1) Legitimate software	2) Adverse software	3) Double agents
Medium Consent	4) Semi-transparent software	5) Unsolicited software	6) Semi-parasites
Low Consent	7) Covert software	8) Trojans	9) Parasites

User consent is specified as either *low*, *medium* or *high*, while the degree of negative consequences span between *tolerable*, *moderate*, and *severe*. This classification allows us to first make a distinction between legitimate software and spyware, and secondly between spyware and malicious software (malware). All software that has low user consent, *or* which impairs severe negative consequences should be regarded as malicious software. While, on the other hand, any software that has high user consent, *and* which results in tolerable negative consequences should be regarded as legitimate software. By this follows that spyware constitutes the remaining group of software, i.e. those that have medium user consent or which impair moderate negative consequences.

We base our work on Simone Fischer-Hübner's definition of *privacy*, in which she divides the concept into the following three areas [11]:

- *territorial privacy* focusing on the protection of the public area surrounding a person, such as the workplace or the public space
- *privacy of the person* which protect the individual from undue interference that constitute for instance physical searches and drug tests
- *informational privacy* protecting if and how personal information (information related to an identifiable person) is being gathered, stored, processed, and further disseminated.

Since our work has its origin in a computer setting we interpret the above three areas into a computer context. We argue that this is

motivated since computers are being increasingly more weaved together with our daily lives, affecting individuals' privacy. Our classification of privacy-invasive software is related to the last two areas listed above, i.e. protecting the user from undue interference, and safeguarding users' personal information, while using computers. Therefore our view of privacy does not only focus on the communication of personal information, but it also includes undue interference that negatively affects the users' computer experience.

In an attempt to mitigate the negative effects from PIS we propose the use of a reputation system where computer users collaborate with the goal to distinguish legitimate software from PIS. As described by Resnick et al. a reputation system "collects, distributes, and aggregates feedback about participants' past behaviour" [27]. This can either be part of a larger system, to give the users incentives to behave well in the future knowing that other users will be able to review past transactions e.g. on an auction site, or as a system itself used for rating e.g. resellers of home appliances, Hollywood blockbusters, or basically any kind of product or service. This helps new users to establish a trust relationship towards a particular reseller or company based on other users' past opinions about the other party, without any personal contact with the reseller or company in question. This is increasingly important considering the present development rate for e-commerce and online services where customers seldom, if ever, meet the business representatives they are dealing with.

7.2 Important Considerations

There are two main issues that need to be addressed when considering the design and implementation of the proposed system. How to protect users' privacy and at the same time address incorrect information in the system. We will address these two considerations individually; explaining the problem at hand, as well as proposing one or more possible solutions that may help prevent the problem, or at least reduce the impact of it [9].

7.2.1 Addressing Incorrect Information

There are a number of aspects to take into consideration when building a system that is to gather, store and present information from multiple, unknown users. Although the system has been set up

for a clear purpose, individual users, or groups of users, may find it more interesting to – for instance – intentionally enter misleading information to discredit a software vendor they dislike, use multiple computers in a distributed attack against the system to fill the database with bogus votes, enter irrelevant or indecent comments, and so on. When it comes to inventing new ways of disturbing peace, the stream of ideas seems to be never-ending.

Even though it may be done without malice, even in good faith, ignorant users voting and leaving feedback on programs they know nothing or little about may be a rather big problem for a software reputation system, especially at a budding phase. If the number of users is low, compared to the number of software to be rated, there is a big risk that many software will be without any, or with just a few, votes. Even worse, if these few votes and comments have been given by users with little actual knowledge about the software they are rating, they may – for example – give the installer of a program bundled with many different PIS a high rating, commenting that it is a great free and highly recommended program. In a normal environment, this would not be a problem, as a number of more experienced users would already have added negative feedback, warning other users of the potential dangers with installing this software package. However, in the cases where there are few users and votes available at any point of time, this may be a big problem.

We have identified three different approaches to mitigate the problem with unintentionally incorrect information. The first one involves allowing the users to rate not only the software but also the feedback of other users in terms of helpfulness, trustworthiness and correctness, creating a reliability profile for each user. This profile could be thought of as a trust factor that is used to weight the ratings of different users, making the votes and comments of well-known, reliable users more visible and influential than those of new users. It does not directly handle the problem of inexperienced users giving incorrect information and ratings, if they are the only ones commenting and voting, but as soon as more experienced users give contradicting votes, their opinions will carry a higher weight, tipping the balance in a – hopefully – more correct direction.

The second approach is to use bootstrapping of the program database at an early stage, preferably before the system is put to use, copying the information from an existing, more or less reliable, software rating database of programs and their individual ratings

into the database of the reputation system. That way, it would be possible to ensure that no common program has few or zero votes, and in the event of novice users giving the software unfair positive or negative ratings and comments, the number of existing votes would make their votes one out of many, rather than the one and only.

The third approach would be to have one or more administrators keeping track of all ratings and comments going into the system, verifying the validity and quality of the comments prior to allowing other users to view them, as well as working on keeping the program database updated, giving expert advice on certain programs, such as well-known white listed applications, etc. However, once the number of users has reached a certain level, this would require a lot of manual work, which could become expensive for maintaining a free program, as well as seriously decrease the frequency of vote updates.

In addition to the problem with users that unintentionally provide the reputation system with incorrect information is the more complex threat by individuals, or groups of people, that decide to purposely abuse the systems. In the preventive anti-PIS reputation system, one such attack would be to intentionally try to enter a massive amount of incorrect data into the database. Either to slow the system down, or even crash it, or to target specific applications, trying to subject them to positive or negative discrimination. The main question when it comes to vote flooding is how to allow normal users to be able to vote smoothly and yet be able to address abusive users that attack the system.

An important aspect to take into consideration is that the server must ensure that each user only votes for a software program exactly once. A common solution to this kind of problem would be to let the user register a user account at the server before being able to activate the client software. For each user account, only one vote and comment can be registered for a specific software. Using some non-automatable process, such as image verification, and requiring a valid e-mail address during the registration of a new user account would help prevent the system for users trying to automatically create a number of new accounts to avoid the limit imposed on the number of votes each user can give to each software [10].

7.2.2 Protecting Users' Privacy

As the system is built for protecting peoples' privacy, we need to make sure the system itself does not intrude on it more than absolutely necessary. If the system would store sensitive information about its users, such as IP addresses, e-mail address, and linking these to all software the user has ever cast a vote on, the system owner would control this sensitive information. Any leakage of such information e.g., through an attack on the reputation system database, could have serious consequences for all users. An attacker getting access to this information would find a list of hosts and software running on each host, where some of them could be vulnerable to remote exploits. However, not storing any data about which users have cast votes on a particular software could lead to vote flooding and similar, as the system would have no way of ensuring that a user only votes once.

As we need to make sure no users can vote more than once on each particular software, we cannot get rid of the concept of users and user accounts. However, one approach would be to ensure that all kinds of sensitive information that can be of use for an attacker, such as IP address, e-mail address, name, address, city, or similar, are excluded from the user information stored in the database of the reputation system server. The only thing necessary to store is some kind of unique identifier for each user, such as a user name.

As mentioned in the previous section, we need to prevent users from signing up several times in an automatic way, and one way of doing this would be to use their e-mail address as an identification item. However, this requires us to store the e-mail address in the database which might not be something that people would like to store in a database that keeps track of which software they are running and their opinions on it. A solution to this would be to only keep a hash value of the e-mail address, as this can be used to discover that two e-mail addresses are equal, while it is impossible to recreate the e-mail address from the hash value. However, it would still be possible to guess the correct e-mail address if relying on a brute force approach. This problem could be further solved by concatenating the e-mail address with a secret string before calculating the hash, rendering brute force attack to be computationally impossible as long as the secret string is kept secret. Protection of users' anonymity could be established by utilizing distributed anonymity services, such as Tor, for all communication between the client and

the server [36]. This would further increase user's privacy by their IP address from the reputation system owner.

7.3 System Design

As we have illustrated in the previous section, there are numerous aspects to take into consideration when designing a reputation system such as this. Information has to be gathered from the reputation system users in a way that address different ways of abuse, without interfering with normal usage and / or the protection of the users' privacy. When considering votes and comments, the system has to be able to handle possible abuse, as well as to properly balance the weight of different users' ratings and allow users to grade each others, thus improving the credibility of the more expert users and degrading users not taking voting and commenting in the system seriously.

The system will be comprised of three major parts, a client with a graphical user interface (GUI) running on each users' workstation, a server running on one or more machines handling all requests and commits from the clients, as well as a database storing all data. The system will also offer a web based interface, which gives the users more possibilities in searching the information stored in the database. This will be used as an extension to the GUI client, where users e.g. can read more information about some particular software program or vendor along with all the comments that have been submitted.

7.3.1 Client Design

The most important functionality of the client is the ability to allow its users to decide exactly what software is allowed to run on the computer, i.e. blocking all software which the user have not explicitly given his/her permission to. This filtering capability is implemented using a hooking device that captures the execution calls from the Windows API, in order to allow the user to choose whether or not he or she really wants to proceed with the execution of that particular software. Whenever software is trying to execute, the hooking device informs the client about the pending execution, which in turn asks the user for confirmation before actually running the software requesting to execute. The API hooking is used to capture the execution call that goes to the Windows kernel when the

operating system tries to allocate memory for the program. We used Anton Bassov's Soviet Protector code when implementing the API hooking functionality, with slight modifications added [16]. It consists of a system driver that replaces the API call to NtCreateProcess() with its own version, and a software component that communicates with the driver through a shared section of the memory[2].

The client uses different lists to keep track of which software have been marked as safe (the white list) and which have been marked as unsafe (the black list). These two lists are then used for automatically allowing or denying software to run, without asking for the user's permission every time, and thereby reducing the need for user interaction. When the driver discovers an execution and informs the client about it, the client traverses the white list and blacklist for an occurrence of the pending software based on a checksum calculated from the EXE-file content, using an algorithm such as for instance SHA-1. If the software is found in either of the two lists, the appropriate response is automatically sent to the driver without the need for user interaction, otherwise the client queries the server and fetches the information about the executing software to show the user and take action based on the user's decision.

The proof-of-concept tool also allows the user to submit ratings and comments, as described in the previous sections, as well as to view compiled information from other users and run statistics about the software about to execute. The user is only asked to rate software which he has executed more than a predefined number of times, currently 50 times. This ensures that the user has been using the software for some time and therefore has developed some sort of opinion about it. To minimize the user interruption there is also a threshold on the number of software the user is asked to rate each week, currently two ratings per week. So, when the user has executed a specific software 50 times she will be asked to rate it the next time it is started, unless two software already has been rated that week.

2. It might at first seem more reasonable to focus on API calls such as NtCreateProcess() [16].

7.3.2 Server Design

In addition to the processing of software ratings the server also handles the database containing registered user information, ratings and comments for different software that users have previously voted on. The clients communicates with the server through a web-server that handles the requests sent by the client software, as well as displaying web pages for showing more detailed information about the software and comments in the database. XML is used as the communication protocol between the client and the server.

The only data stored in the database about the user is a username, hashed password and a hashed e-mail address, as well as timestamps of when the user signed up, and was last logged in. The e-mail address is only there to make it more difficult for a person to create several different accounts, as it is possible to sign up only once per e-mail address. Each e-mail address used to sign up must be valid, since it is used for the confirmation and activation of the newly created account.

From this data it is not possible for us, or anyone else getting in hold of the database, to identify a specific user, as long as the username (over the contents of which we have little control) does not reveal too much detailed information. And as our implementation does not store any IP addresses associated with the users, it is also impossible to determine which hosts are running which software, and from there try to launch an attack against a specific host. What can be traced however, is every user's submitted rating, comment and answers for each software he or she has ever rated, as well as each user's submitted remark (positive for a good, clear and useful comment or negative for a coloured, non-sense or meaningless comment) for every comment he or she has ever rated. But as mentioned previously, it is impossible to directly or indirectly associate this data with a particular host, but only to a username, hashed password, hashed e-mail address and two timestamps, which does not put the user at any actual risk from using this software.

Software ratings are calculated at fixed points in time (currently once in every 24-hour period). During this work users' trust factors are taken into consideration when calculating the final score for a particular software. In addition to these software ratings the proof-of-concept tool also calculates specific software vendor ratings.

This is done by simply calculating the average score of all software belonging to the particular vendor.

As a protection mechanism, the reputation system has implemented a growth limitation on users' trust factors, by setting the maximum growth per week to 5 units. Hence, you can reach a maximum trust factor of 5 the first week you are a member, 10 the second week, and so on. Thereby preventing any user from gaining a high trust factor and a high influence without proving themselves worthy of it over a relatively long period of time. The second limitation of the trust factor is a minimum level of 1 (which is also the rating for new users), and a maximum of 100.

7.3.3 Database Design

Each software represented in the database will hold a set of information that is linked directly to the executable file. The most important information is the unique software ID number which is generated by utilizing a hash algorithm over the file content. Since this ID is a calculated out of the file data (its program instructions) it is also directly connected to the software behaviour. This means that it is impossible to change the software behaviour without also changing the software ID. In other words, it is impossible to alter the programs behaviour and still keep the ratings associated with the software in the database, which is an important property for a software reputation system. Since the software ID is generated through by a hash algorithm (e.g. SHA-1) the risk of two different files having identical fingerprints is virtually non-existent. In addition to user ratings and comments the following information is stored for each software in the database:

- ID of software executable e.g., a generated SHA-256 digest.
- File name of the software executable.
- File size of the software executable.
- Company name of the software company that produced the software executable.
- Software version number.

Information about both the company name and file version is dependant on the software developer to put these values into the program file, which unfortunately is not always true. The rest of the

data is meta-data that always can be retrieved once the complete file is in ones possession.

Since hash functions are used, the software ID will be different even between files with small modifications, in effect, two different versions of the same program will end up having different fingerprints. This also means they will be considered as separate software executables by the reputation system server, and as such their votes and ratings will be separated from each other. Although a drawback with this approach is that there will be many different database entries for slightly different versions of the same program, this may in fact be beneficial to the user. For example, one version of an application may be well known to cause degraded performance, display banners, and so on, while in the next version, the developers have fixed the performance issues and decided to use other means to finance their work, and thus the contents of the reputation system will correctly present this to the user.

However, questionable software vendors that want to try to circumvent the reputation system could try to make each instance of their software applications differ slightly between each other so that each one has its own distinct hash value. The countermeasure against such behaviour would be to instead map all ratings to the software vendor instead of mapping it to a specific software version from that vendor. To fight that countermeasure some vendors might try to remove their company name from the binary files. If this should happen it could be used as a signal for PIS since legitimate software vendors label their products with their company information [6].

Furthermore, it would be possible for the system to provide users with valuable information about the vendor of a specific software by calculating the mean value over all software ratings the company in question has received. Giving the user an indication of how well the software developed by this company has previously fared in the reputation system. That way, the user may choose to base his decision on ratings and comments given not only on the current software executable, but also on the derived total rating of the software developing company.

7.4 Discussion

In this section we will discuss what impact the introduction of a software reputation system would have on privacy-invasive software. We will also bring up some issues with the proof-of-concept implementation together with improvement suggestions. In the end we make a comparison between existing countermeasures against PIS and the software reputation system.

7.4.1 System Impact

Offering users mechanisms that enhance informed decisions regarding software installation increase the liability of the user. In a way, these mechanisms transfer some of the responsibility concerned with the protection against PIS to the users themselves. We believe this is a necessary consequence for new protection mechanisms that respect users' own personal needs and preferences. As users are being confronted with descriptions about behaviours and consequences for PIS, they are also assumed to assimilate and use this information in a mature and reasonable way. Based on the reputation system, it would be up to the users themselves to decide on whether or not to allow certain software to enter their system.

Table 7.2 Difference between legitimate software and malware with respect to user's informed consent and negative user consequences.

	Tolerable Negative Consequences	Moderate Negative Consequences	Severe Negative Consequences
High Consent	Legitimate software	Adverse software	Doubleagents
Low Consent	Covert software	Trojans	Parasites

Computer users today face similar difficulties when evaluating software as consumers did a hundred years ago when evaluating food products. In the nineteenth century food industry, distribution of snake-oil product flourished [35]. These products claimed to do one thing, for example to grow hair, while they instead made unwitting consumer addicted to habit-forming substances like cocaine and alcohol. In 1906 the Pure Food and Drug Act was passed by the United States Congress, allowing any manufacturer not complying with the rules to be punished according to the law [20]. As a consequence the manufacturers followed these rules, allowing consumers to trust the information on the food container to be cor-

rect. Further allowing them to make informed decisions on whether they should consume a product or not, based on individual preferences such as nutritiousness, degree of fat or sugar, price, or allergies. As long as the food does not include poisonous substances or use deceptive descriptions it is up to the consumer to make the final decision. Although the distribution of physical snake-oil products was mitigated in 1906, its digital counterpart continues to thrive under the buoyant concept of spyware. An important distinction between food products and software is that the former one relies on physical factories and companies with employed personnel, which software does not. It is possible for anyone with the programming skills to produce software which then is spread globally over the Internet. Since users do not always have the option to relate the software to a physical manufacturer we believe it is important for them to instead be able to use other users' previous knowledge about the product in question, offered to them by a software reputation system.

It should be noted that a reputation system against PIS tightly affect the PIS classification in Table 7.1. The introduction of this type of user-oriented countermeasure would transform the classification of PIS as shown in Table 7.2. As computer users are given a tool to make informed decisions regarding the behaviour and implications of software, it is possible to apply a sharp boundary based on user consent between all software in the PIS classification. Using the added knowledge provided by the reputation system would render in that all PIS that previously have suffered from a medium user consent level, now instead would be transformed into either a high consent level (i.e. legitimate software) or a low consent level (i.e. malware). In other words, all software with medium user consent, i.e. spyware, is transformed into either legitimate software or malware in the classification. Since anti-malware tools handle all malicious and deceitful software, the information about the rest of the software could be trusted to be correct, i.e. any software using deceitful methods is regarded as malware and are treated as such. This allows users to rely on the information when reaching trust decisions regarding software installation on their system. Another aspect of this type of countermeasure is that no single organization, company or individual is responsible for the software ratings, since these are calculated based on all votes submitted by the users. Making it hard for dissatisfied spyware vendors to sue the reputation system owners for defamation.

7.4.2 Improvement Suggestions

One issue that we soon discovered during tests of the proof-of-concept tool was the question of system stability. As we give the users the ability to deny the execution of important system components, we also handed them the ability to crash the entire system in a single mouse click. This further enhances the need for a white list system to ensure proper operating system functionality in order to avoid inadvertently bringing the operating system down when running the software client. However, given that the user has the free choice to block any program, there is no way to guarantee that the operating system will not be crashed, especially at an initial phase where the user is learning how to use the software client.

The proposed solution to this problem would be an enhanced white listing system that could examine the file about to execute, to determine if it has been digitally signed by a trusted vendor e.g., Microsoft or Adobe. In case the certificate is present and valid, the file is automatically allowed to proceed with the execution. It would also be possible to implement a signature handling interface in the reputation system client that allows the user to white list and blacklist different companies through their digital signatures, which – in turn – could considerably lower the need for user interaction. In that regard the proposed functionality would be somewhat similar to the capabilities of Microsoft's Restricted Software Policies [23].

The introduction of an enhanced white listing system with signature verification capabilities would provide an important building block for a software policy manager. By using the information available in the reputation system it would be possible for corporations or individual users to set up policies for what software is allowed to execute on their computers. Such policies could for instance take into account whether the software has been signed by a trusted vendor, the software and vendor rating, or any specific behaviour reported for the software e.g., if it show pop-up advertisements or include an incomplete removal routine. This would allow system owners to define policies for what software is allowed to install and run on their computers e.g., by specifying that any software from trusted vendors should be allowed, while other software only is allowed if it has a rating over 7.5/10 and does not show any advertisements. A solution like this implies that the reputation system also includes a preference module that holds the users' software preferences that should be enforced.

Another improvement suggestion involves allowing for instance organisations or groups of technically skilled individuals to publish their software ratings and other feedback within the reputation system. This information is then available for any other users of the reputation system. Allowing computer users to subscribe to information from organisations or groups that they find trustworthy, i.e. not having to worry about unskilled users that might negatively influence the information. The subscribed information could of course also be used in parallel with the other software feedback which is based on all reputation system members' votes.

7.4.3 Comparison with Existing Countermeasures

One major difference between traditional anti-spyware software and the reputation system based solution we propose is that in the latter we are able to gather more complete and useful information regarding the behaviour of software. Instead of a black and white world where an executable is branded as either a virus or not, we are able to touch the previously mentioned greyzone in between. We gather and present information about software that is important and useful to the users, and hard to find. For instance, although an application may not be classified as a virus or spyware, users may think twice about running it if they are informed that it displays pop-up ads, registers itself as a start-up program and does not provide a functioning uninstall option. This kind of discouraging information will not be provided by the vendor of the application and can only be received from users who have experienced it first-hand and are willing to share their experiences to help others.

Currently available countermeasures against PIS, such as anti-spyware and anti-virus applications, have the benefit of specialized, up to date and reliable information databases that are updated on a regular basis. The drawback is a vendor database that must be updated locally on the client, as well as traversed whenever a file is analysed. Furthermore, the organization behind the countermeasure must investigate every software before being able to offer a protection against it. The relevance and reliability of the information provided by the anti-spyware and anti-virus software may be more reliable than that of users of a reputation system. However, the reputation system is able to cover more details that may be useful to the user, such as if the software displays ads, alter system settings, and so on, and with a sufficiently large user base, the sheer amount of data gathered helps compensate for the afore mentioned reliabil-

ity issue. Also, by using a more flexible classification, where the user is provided the information about the software and is allowed to make an informed decision about allowing it to run or not, one is able to avoid the high contrast environment of anti-virus software and similar, where an executable is either strictly malicious or it is totally safe.

Different protection systems (e.g., anti-virus or anti-spyware tools) are built on different approaches, and the technology as well as pricing varies. In truth, it would be foolish to believe that either one approach would be a perfect solution to the problem at hand, and the view of the problem itself may differ. However, when looking at the development of the computer world, the Internet, and the ongoing arms race in virus and spyware development, it is obvious that more than just one kind of protection is needed, and that there is no silver bullet. At the same time, we firmly believe that a specialized reputation system such as the one we propose would be a useful way to be able to penetrate the gray zone of half-legitimate software and to better inform users of what to expect from the software they are about to execute. It can be seen as trying to share and transfer knowledge between users, improving their level of expertise, instead of creating an expert system that handles all the decisions for the users, being ultimately responsible for the failure when the protection fails.

7.4.4 Conclusions and Future Work

This paper explores how to construct a specialized reputation system to be used for blocking privacy-invasive software. The fundamental idea is that computer users could be strong together if they collaborate to mitigate the effects from privacy-invasive software. The co-operation is based on that each users rate the software that they use most frequently. These aggregated ratings from the users are then transformed into software reputations that are available for all participants in the system upon installation of new software. Various methods to address incorrect information in the system, be it intentional or unintentional, are proposed without deteriorating users' privacy.

To further explore the possibilities, we designed and implemented a client and server-based proof-of-concept tool, which currently include well over 2000 rated software programs. Each time a user is about to execute a program, the client pauses the execution, down-

loads information and rating about the particular software from the server, and asks the user whether he or she would like to allow or deny the software to run. We further propose how this system could be enhanced by adding functionality that allows users to produce software policies that are automatically enforced by the client program. Such policies could take into account whether the software in question has received a rating above a certain value, if it is digitally signed by a trusted vendor, or if it is free from a set of predefined unwanted behaviours.

As future work we will investigate how and to what extent this proof-of-concept tool affects computer users' decisions when installing software. In addition to this we will also examine the possibility of using runtime software analysis to automatically collect information about whether software has some unwanted behaviour, for instance if it shows advertisements or includes an incomplete uninstallation function [24]. The results from such investigations could then be inserted into the reputation system as hard evidence on the behaviour for that specific software. Furthermore, it would be interesting to investigate the use of alternative, and more reliable, security tokens than e-mail addresses when creating new accounts. Maybe also relying on the IP address and computational penalties through variable hash guessing [3]. Finally, it would be interesting to investigate how pseudonyms could be used as a way to protect user privacy and anonymity, e.g. through the use of idemix [17].

7.5 References

[1] Ames, W.: Understanding spyware: risk and response. IEEE IT Professional 6(5) (2004)

[2] Arnett, K.P.: Busting the Ghost in the Machine. Communications of the ACM 48(8) (2005)

[3] Aura, T.: DOS-Resistant Authentication with Client Puzzles. LNCS, vol. 2133. Springer, Heidelberg (2000)

[4] Boldt, M.: Privacy-Invasive Software – Exploring Effects and Countermeasures, Licentiate Thesis Series No. 2007:01, School of Engineering, Blekinge Institute of Technology, Sweden (2007)

[5] Boldt, M., Carlsson, B.: Privacy-Invasive Software and Preventive Mechanisms. In: The proceedings of the IEEE International Conference on Systems and Networks Communications (ICSNC06), Papeete Tahiti, IEEE Computer Society Press, Los Alamitos (2006)

[6] Boldt, M., Carlsson, B., Martinsson, R.: Software Vulnerability Assessment – Version Extraction and Verification. In: The proceedings of the Second International Conference on Software Engineering Advances (ICSEA'07), Cap Esterel France (2007)

[7] Bruce, J.: Defining Rules for Acceptable Adware. In: The Proceedings of the 15th Virus Bulletin Conference. Dublin Ireland (2005)

[8] Christodorescu, M., Jha, S.: Testing Malware Detectors. In: The proceedings of the ACM International Symposium on Software Testing and Analysis (2004)

[9] Dellarocas, C.: Immunizing Online Reputation Reporting Systems Against Unfair Ratings and Discriminatory Behaviour. In: The proceedings of the 2nd ACM Conference on Electronic Commerce (2000)

[10] Douceur, J.: The Sybil Attack. In: The proceedings for the 1st International Workshop on Peer-to-Peer Systems (2002)

[11] Fischer-Hübner, S.: IT-Security and Privacy: Design and Use of Privacy-Enhancing Security Mechanisms. Springer, Heidelberg (2001)

[12] Flixster, http://www.flixster.com, 2010-03-11

[13] Good, N., et al.: Stopping Spyware at the Gate: A User Study of Privacy, Notice and Spyware. In: The proceedings of the Symposium on Usable Privacy and Security, Pittsburgh, USA (2005)

[14] Good, N., et al.: User Choices and Regret: Understanding Users' Decision Process about Consentually Acquired Spyware. I/S: A Journal of Law and Policy for the Information Society 2(2) (2006)

[15] Greenfield, A.: Everyware – The Dawning Age of Ubiquitous Computing. New Riders, Berkeley CA (2006)

[16] Hooking the native API and controlling process creation on a system-wide basis, http://www.codeproject.com/system/soviet_protector.asp, 2010-03-11

[17] Idemix: pseudonymity for e-transactions, http://www.zurich.ibm.com/security/idemix/, 2010-03-11

[18] Internet Movie Database, http://www.imdb.com, 2010-03-11

[19] Kirda, E., Kruegel, C., Banks, G., Vigna, G., Kemmerer, R.A.: Behavior-Based Spyware Detection. In: The proceedings of the 15th USENIX Security Symposium (2006)

[20] Landmark Document in American History, Pure Food and Drug Act of 1906, http://coursesa.matrix.msu.edu/~hst203/documents/pure.html, 2010-03-11

[21] Larsson, T., Lindén, N.: Blocking Privacy-Invasive Software Using a Specialized Reputation System, Masters Thesis No. 2006:14, School of Engineering, Blekinge Institute of Technology, Sweden (2006)

[22] LavaSoft Ad-Aware, http://www.lavasoftusa.com/software/adaware, 2010-03-11

[23] Microsoft Technet, Using Software Restriction Policies to Protect Against Unauthorized Software (May 13, 2007)

[24] Moshchuk, T., Bragin, S.D., Gribble, H.M.: A Crawler-based Study of Spyware on the Web. In: The proceedings of the Network and Distributed System Security Symposium Conference Proceedings, Virginia USA (2006)

[25] Norton Internet Security, http://www.symantec.se/region/se/product/nis index.html, 2010-03-11

[26] Pricerunner, http://www.pricerunner.com, 2010-03-11

[27] Resnick, P., Kuwabara, K., Zeckhauser, R., Friedman, E.: Reputation Systems. Communications of the ACM 42(12) (2000)

[28] Rosenberg, R.S.: The Social Impact of Computers, 3rd Ed., San Diego CA. Elsevier Academic Press, Amsterdam (2004)

[29] See you later anti-Gators, CNET News.com, http://news.com.com/2100-1032 3-5095051.html, 2010-03-11

[30] Schultz, K.: Sticking It to Spyware. InfoWorld 27(38) (2005)

[31] Sipior, J.C.: A United States Perspective on the Ethical and Legal Issues of Spyware. In: Proceedings of 7th International Conference on Electronic Commerce, Xi'an China (2005)

[32] Spyaudit, http://www.earthlink.net/spyaudit/press/, 2010-03-11

[33] Spybot -Search & Destroy, http://www.safer networking.org, 2010-03-11

[34] "Spyware": Research, Testing, Legislation, and Suits, http://www.benedelman.org/spyware/, 2010-03-11

[35] Technology Review, The Pure Software Act of 2006, http://www.simson.net/clips/2004/2004.TR.04.PureSoftware.pdf, 2010-03-11

[36] Tor: anonymity online, http://tor.eff.org, 2010-03-11

[37] Webroot Software, —.: Internet Spyware and statistics about infection rate, http://www.webroot.com/resources/stateofspyware/excerpt.html, 2010-03-11

[38] Zacharia, G., Moukas, A., Maes, P.: Collaborative Reputation Mechanisms in Electronic Marketplaces. In: the proceedings of the 32nd Hawaii International Conference on System Sciences (1999)

PUBLICATION

5

Learning to Detect Spyware using End User License Agreements

International Journal on Knowledge and Information Systems (KAIS), 2010

Niklas Lavesson, Martin Boldt, Paul Davidsson and Andreas Jacobsson

The amount of software that hosts spyware has increased dramatically. To avoid legal repercussions the vendors need to inform users about inclusion of spyware via End User License Agreements (EULAs) during the installation of an application. However, this information is intentionally written in a way that is hard for users to comprehend. We investigate how to automatically discriminate between legitimate software and spyware associated software by mining EULAs. For this purpose, we compile a data set consisting of 996 EULAs out of which 9.6% are associated to spyware. We compare the performance of 17 learning algorithms with that of a baseline algorithm on two data sets based on a bag-of-words and a meta data model. The majority of learning algorithms significantly outperform the baseline regardless of which data representation is used. However, a non-parametric test indicates that bag-of-words is more suitable than the meta model. Our conclusion is that automatic EULA classification can be applied to assist users in making informed decisions about whether to install an application without having read the EULA. We therefore outline the design of a spyware prevention tool and suggest how to select suitable learning

algorithms for the tool by using a multi-criteria evaluation approach.

8.1 Introduction

The amount of spyware has increased dramatically due to the high value for marketing companies of the information that is collected. Spyware is designed to collect user information for marketing campaigns without the informed consent of the user. This type of software is commonly spread by bundling it with popular applications available for free download. A spyware application is typically difficult to remove once it has been installed on a computer system and it can seriously degrade system performance and compromise the privacy of the user [2, 5, 30]. This paper presents a novel approach based on data mining, aimed at stopping spyware at the door.

From now on we are going to use the terms *bad* and *good* to signify applications that host spyware and legitimate applications, respectively. Distributors of bad software usually try to disguise it as good in an attempt to reach as many users as possible. However, to avoid legal repercussions these distributors are required to mention in the End User License Agreement (EULA) that spyware will indeed be installed. Yet, this information is given in a way most users find difficult to understand. Even EULAs for legitimate software can be hard to comprehend due to their length and their extensive use of legal terminology [18].

Consequently, we recognize the need for an efficient method for helping users to distinguish between good and bad software during the installation process. If spyware is detected through such a method, it could assist users in keeping their computers clean from bad software by warning them about the application they are about to install.

8.1.1 Background

Definitions of spyware exist on two different abstraction levels, were the low level definition focuses on the concept of information theft while the high level definition also includes other negative consequences associated with spyware. At the low level, spyware can be defined by using Steve Gibson's original definition from early 2000[1]:

"*Spyware is any software which employs a user's Internet connection in the background (the so called backchannel) without their knowledge or explicit permission.*"

This definition was valid in the beginning of the spyware evolution, but as the spyware concept evolved it attracted new kinds of behaviors. From originally just gathering private information spyware programs now also began to modify network traffic, such as competitors' advertisements, and degrading user experience by altering system configurations, such as browser start pages. As these behaviours grew both in number and in diversity, the term spyware became hollowed out, which in turn resulted in that a great number of synonyms were employed, e.g., thiefware, trackware, evilware, scumware, and badware.

Due to this development, the Anti-Spyware Coalition, which consists of leading parties from both academia and industry, defined spyware using a wider approach that incorporated all negative consequences associated with such software[2]. Our view of spyware also coincides with this definition:

"*Technologies deployed without appropriate user consent and/or implemented in ways that impair user control over:*
– Material changes that affect their user experience, privacy, or system security;
– Use of their system resources, including what programs are installed on their computers; and/or
– Collection, use, and distribution of their personal or other sensitive information."

To further clarify the concept of spyware, Boldt [3] defined a categorization of various types of spyware programs that relied on a three-by-three matrix where the two axis represent the level of user consent and the amount of negative consequences associated with each software type. Given these two dimensions Boldt provides a more structured view of spyware under the term privacy-invasive software. The increase of spyware has resulted in vast numbers of users experiencing loss of control over personal information and decreased computer performance [34]. Anti-virus techniques are used for removing malicious software (such as: computer viruses and worms). Malicious software is undoubtedly illegal in most

1. Gibson Research Corporation, www.grc.com/optout.htm
2. Anti-Spyware Coalition, http://www.antispywarecoalition.org

countries but this is not necessarily true for spyware since it resides in a gray zone between what is considered to be legal and illegal.

Thus, the techniques used by spyware can be interpreted as either legal or illegal depending on who is asked; what one individual regards as spyware could be considered a legitimate business application by another.

McFedries [24] explores the purposes behind spyware as well as its commercial impact. The main conclusion is that spyware is a very good means to gain revenue for online marketing companies, since it provides them with personal user information that can be exploited in order to deliver targeted advertisements.

Zhang [36] makes visible the urgent need for user education to raise awareness about the spyware threats and the methods available for addressing these threats. This is important, Zhang argues, since most computer users cannot keep up with the rapid development of new spyware that affect their computer systems, personal data, and ultimately their privacy.

Moshchuk et al. [26] introduce methods for measuring spyware activity on the Internet. In 2005, they performed an automatic analysis of 18 million Internet addresses (URLs) for executable files. Out of the 21,200 applications an astonishing 13.4% were identified as spyware.

Fox presents a report on how user behaviour is affected by the occurrence of spyware in home and work computers[3] based on performing a telephone interview survey, which featured a nationally representative sample of 2,001 adults living in continental United States telephone households. The report states that "*only about one in ten internet users say the current practice of clicking through a user agreement or disclaimer is adequate consent to install adware on a person's computer*". Townsend elaborates on how spyware infected applications may violate corporate security policies and procedures[4], explaining that spyware could circumvent existing security mechanisms used by the

3. Fox, S.: Spyware – the threat of unwanted software programs is changing the way people use the Internet, http://www.pewinternet.org/pdfs/PIP_Spyware_Report_July_05.pdf (2005)
4. Townsend, K: Spyware, Adware, and Peer-to-Peer Networks – The Hidden Threat to Corporate Security, Technical White Paper, Pest Patrol (2003)

corporations, thus enabling the extraction of sensitive and/or classified information. Moreover, Good et al. [18] point out the fact that users agree to accept spyware as part of a software bundle as a cost associated with gaining the functionality they desire and demonstrate that interface design can be a significant factor in eliciting informed consent to software installation.

8.1.2 Related Work

In a pilot study [23], we investigated whether it was possible to take advantage of the fact that the installation of bad software is mentioned in the EULA. We addressed this problem by applying supervised learning algorithms to classify EULAs of both good and bad applications in order to detect if the associated software hosts spyware. The results indicate that the approach is feasible, however the amount of data was scarce (the data set featured 100 EULAs in total). EULA classification, as a problem, is quite analogous to that of spam classification, i.e., to distinguish between unsolicited commercial e-mails (spam) and legitimate e-mails. Much work has been done in the area of spam classification, e.g., using different learning algorithms, such as: rule learners, support vector machines, instance-based learners, decision trees, and stacking [1, 7, 9, 13, 28]. More recently, Koprinska et al. [21] investigate the performance of the random forest algorithm for the same type of problem, claiming that it outperforms some of the earlier mentioned algorithms on several problems. Kang et al. [19] also study the spam classification problem and applies an unsupervised feature selection algorithm and clustering to classify unlabelled documents. The results from the analysis can be interpreted as follows: the absence of a certain term is a characteristic shared across the e-mails of a given category; whereas the presence of certain keywords shows a larger variability across e-mails of a given category.

8.1.3 Scope and Aim

The aim of this paper is to further investigate EULA classification as a means for categorizing software as good (legitimate) or bad (associated with spyware). For this purpose, we have collected a set of 996 EULAs (out of which 9.6% are associated with software that includes spyware). We will investigate different ways to represent these documents in order to be able to use supervised learning as an approach to solve the EULA classification problem.

8.1.4 Outline

The remainder of this paper is organized as follows: we begin by presenting the EULA classification problem in Section 2. Next, we describe the data collection process and the representation of data in Section 3. We then describe the experiments in Section 4. This is followed by a presentation of the results in Section 5. In Section 6, we analyse the results and outline the design of a spyware prevention tool based on EULA classification. Finally, we draw conclusions and give some pointers to future work in the last section.

8.2 EULA Classification

We want to investigate whether or not it is possible to classify software applications as legitimate (good) or associated with spyware (bad) based on their EULA. Intuitively, we can look upon this problem as a text classification (TC) task, for which there are two main approaches; knowledge engineering and machine learning.

Knowledge engineering systems have been known to often outperform machine learning systems on the TC task, although according to Feldman and Sanger [16] the gap in performance steadily shrinks. The main drawback of the former approach is the amount of skilled labour and expert knowledge required to generate and maintain the knowledge-encoding rules.

In contrast, the latter approach requires only a set of labelled (classified) training instances, which are less costly to produce. As a consequence, most of the recent work on categorization is concentrated on the machine learning approach and our study is no exception. We will now present the TC task more formally with regard to EULA classification.

Suppose that we have a collection, I, of EULAs, each labelled as either *good* or *bad*, depending on whether or not it is associated with legitimate software or spyware. The set of all possible classes can thus be defined as $C = \{\text{good, bad}\}$. We would like to approximate the unknown target function, $F : I \times C = \{1,0\}$. The value of $f(i,c)$ is equal to one if the EULA, i, belongs to the class c or zero otherwise.

It is now possible to define a classifier as an approximation function, $M : I \times C = \{1, 0\}$. The objective of the learning task is to gen-

erate a classifier that produces results as close to that of F as possible.

8.2.1 Supervised Learning

The machine learning approach to automatically build a classifier by learning the properties of the classes from a set of pre-classified training instances is known as supervised learning.

Most supervised learning algorithms cannot process text documents in their original form [16], hence they will not be able to process the EULAs. Consequently, we need to preprocess the EULAs by converting them to a manageable representation.

8.2.2 Representation

The choice of representation depends on what one regards as meaningful units of text and the meaningful natural language rules for combining these units, i.e., the problem of lexical and compositional semantics, respectively. The problem of compositional semantics has often been disregarded in TC [29], however, exceptions exist, see for example Denoyer et al. [12]. One approach that addresses the problem of lexical semantics is to represent each EULA as a feature vector. We now describe two quite different feature vector representations of EULAs.

8.2.2.1 The Bag-of-Words Model

The bag-of-words model is a common approach to represent documents as feature vectors [16]. In fact, it has been found in several experiments that more sophisticated representations do not yield any significant effectiveness [29], although there are some recent approaches that have shown promising results. For example, Wang et al. [32] automatically constructed a thesaurus of concepts from Wikipedia and introduced a unified framework to expand the bag-of-words representation with semantic relations. More research is needed in order to establish whether this type of expansion really increases performance significantly over the traditional model.

In the bag-of-words model, each word in a document is used as a feature. Thus, a complete document is represented by a vector with one feature for each word that occurs in the document. A collection

of documents is then represented by a set of feature vectors and the dimension of the feature space is equal to the number of different words in all of the documents. Some studies have used phrases, rather than individual words, as terms. However, their experimental results have not been as encouraging as those of studies that use words as terms [29]. More recently, Metzler and Croft [25] review several attempts to go beyond the bag-of-words representation but few show consistent improvements in retrieval effectiveness.

There are basically two methods for associating weights to features. The simplest is the binary method, which either assigns a value of one if the word is present in the document or a value of zero otherwise. The binary method can be used if the chosen learning algorithm can only handle nominal attributes. However, what appears to be a more common method is to take the frequency of the word into account. We adopt this method and calculate frequencies using the standard Term Frequency Inverse Document Frequency (TF IDF) scheme [16]. The TF IDF function embodies the intuitions that: the more often a term occurs in a document, the more it is representative of its content, and the more documents a term occurs in, the less discriminating it is [29]. There exist several versions of TF IDF that differ from each other in terms of logarithms, normalization or other factors.

We use Weka's implemented version of TF IDF. Thus, given a word, *w*, in a EULA, *d*, the TF IDF weight is calculated as follows:

$$weight(w,d) = TermFreq(w,d) \cdot \log \frac{N}{DocFreq(w)} \qquad (1)$$

where N is the total number of EULAs, *DocFreq* is the number of EULAs containing the word and *TermFreq* represents the frequency of the word in the particular EULA.

8.2.2.2 The Meta EULA Model

FaceTime Security Labs[5] has created the EULA Analyzer[6], which is a web-based tool, for assisting people in better understanding the EULAs of software downloadable on the Internet. This tool

5. http://www.facetime.com
6. http://www.spywareguide.com/analyze

requires a user to copy the EULA from the pop-up window that appears when a software application is to be installed on a computer. The EULA is analyzed and the results are given as scores according to 10 different metrics. The user is then able to consider the metric scores to make a decision about whether or not to proceed with the software installation. However, EULA Analyzer does not provide a classification of the analyzed EULAs, instead this task is left to the user. Since EULA Analyzer is a proprietary service, its design and inner workings are not publicly available.

Since the objective of EULA Analyzer is to assist people in better understanding EULAs by supplying meta models of the EULAs, we hypothesize that this meta EULA model can be used as an alternative means for representing the EULAs as feature vectors.

EULA Analyzer calculates the number of flagged, or suspicious, sentences. In addition, it calculates some basic statistics, such as: the number of characters, words, sentences and the average number of words per sentence. EULA Analyzer also performs more advanced calculations of readability metrics. For example, Flesch score and Gunning-Fog index use word and sentence lengths to indicate how difficult it is to comprehend a certain text, where low scores represent difficult texts [17]. The Flesch grade is calculated based on the same measures but instead gives the number of years of education that is needed to understand the text [17]. Automated readability and Coleman-Liau are two other readability tests that indicate the required level of education needed to comprehend a text [10, 31]. Both of these tests differ from the earlier mentioned metrics by relying on a factor of characters per word instead of syllabus per word.

Our approach is to use the aforementioned EULA Analyzer metrics as attributes for a meta EULA data set. Thus, each EULA is described as a vector of 10 values and it has the same class label as the corresponding bag-of-words instance.

8.3 Data Sets

We now describe the data collection and the representation of the data. The collection, and labelling, of EULAs can be carried out in several ways. Our primary objective was to collect a large set of

EULAs and at the same time allow for an automatic or semi-automatic labelling of EULA instances.

8.3.1 Data Collection

One of the contributions of this study is the collection of a large data set of labelled EULAs. The raw data set and different representations are available on the web[7]. We strived to collect approximately 1,000 EULAs (compared to 100 as featured in the aforementioned pilot study). The prerequisite was that each application should be easily downloaded from the Internet and present the user with a EULA that can be copied and pasted as ASCII text. We collected the good instances from Download.com[8] and the bad instances from links obtained from SpywareGuide.com[9]. The software programs in the SpywareGuide.com database are divided into nine categories; adware, data-miners, dialers, loyaltyware, miscellaneous security, password crackers, trojans, viruses, and worms. None of the virus or worm programs include EULAs, hence the programs from these categories were omitted. We chose to label all programs from the remaining seven categories as spyware since we target the general concept of spyware. Additionally, we wanted to avoid multiple spyware classes (since the total number of available spyware programs was small) in order to maximize the amount of training data for each category. Thus, our collection consists of EULAs from two categories; legitimate (good) and spyware (bad). Based on the number of applications downloadable from both of these locations we hypothesized that a ratio of 1:10 would serve as a sufficient model of the real world, i.e., that spyware accounts for approximately 10% of all freely available applications. This is not too far from the figure reported by Moshchuk et al. [26] (13.4%). Thus, we aimed to collect 900 good and 100 bad EULAs). The collection was systematically performed:

–for the good EULAs, the first 900 applications were downloaded,
–for the bad EULAs, every working web link was followed.

In both cases, we omitted applications that did not have a EULA. Additionally, we also omitted applications for which the EULA could not be extracted, e.g., the text could not be copied as ASCII

7. http://www.bth.se/com/nla.nsf/sidor/resources
8. http://www.download.com
9. http://www.spywareguide.com

text. Even though this only applies to 0.5% of the downloaded software we note that this (dys)function could possibly be used by the spyware vendors to prevent EULA analysis. For each software application, we collected the associated EULA together with basic information such as: software title, version, and vendor. The result is a collection featuring 900 good and 96 bad instances of real-world EULAs.

8.3.2 Data Representation

The raw data set, which consists of 996 text documents, must be converted to a manageable format. We perform the conversion into two different feature vector representations, as described in Section 2.2. We now describe how the two data sets are generated from the raw data set and highlight some characteristics for each of the generated sets of data.

8.3.2.3 Bag-of-Words Data Set

The total number of words, contained in all of the EULAs, is close to 10,000 but experimental evidence suggests that this number can be reduced to around 10% without reducing classifier performance [16]. In order to reduce the number of features we convert all characters to lower case and consider only alphabetic tokens. Furthermore, we remove stop words and words that occur only once (hapax legomena), and store a maximum of 1,000 words per class. Finally, we apply Weka's iterated Lovins stemmer to be able to store only the stems of the remaining words. The result is a data set with 1,269 numeric features and a nominal target feature that indicates if the EULA is bad or good.

Table 8.1 EULA Analyzer results for the complete data set.

Metric	Min	Mean (Std.Dev)	Max
Flagged Sentences	0	2.14 (5.89)	84
Number of characters	72	7308 (6880)	70566
Number of words	15	1262 (1160)	11649
Number of sentences	1	42.53 (34.9)	376
Average words per sentence	3.75	29.00 (8.83)	79.35
Flesch score	-36.87	24.56 (12.0)	95.87
Flesch grade	1	17.05 (3.54)	37
Automated readability index	3	19.97 (4.68)	47
Coleman-Liau index	6	19.71 (3.20)	29
Gunning-Fog index	12	43.95 (8.77)	93

8.3.2.4 Meta EULA Data Set

The processing of EULAs through EULA Analyzer is carried out by custom-made Unix shell scripts and Apple Automator programs. This setup allows us to automatically fetch the text from each of the 996 EULAs, paste it into the text field on the EULA Analyzer website, send the text for analysis, and finally retrieve the scores of the 10 metrics that represent the results of the analysis. Using the results according to these metrics, for each EULA, we generate a meta EULA data set and convert it to the Weka ARFF format. Table 8.1 shows the minimum, maximum, and mean value for each EULA Analyzer metric. As can be seen in the, the Flesch score can be negative. Unfortunately, some algorithms cannot handle negative numeric attributes. For example, Naive Bayes Multinomial, which outperformed all other algorithms in the pilot study of EULA classification, is one such algorithm. Consequently, we opt to remove the Flesch score attribute from the meta EULA data set to resolve this issue. For our data set of 996 instances, the *number of flagged sentences* attribute values range from 0 to 84 but most EULAs are assigned a value close to zero. In analyzing the distribution of good and bad EULAs across the range of the flagged sentences attribute, we observe that this attribute can be used to achieve a linear separation between a cluster of 25 bad instances and the rest of the EULAs. Approximately 30 additional bad instances can be isolated together with a small number of good instances. In total, this amounts to a separation of approximately 55 bad instances from the rest of the instances. In this case, there are still 41 bad instances classified as good. In analyzing the distributions for the remaining attributes, we do not observe any trivial ways to separate good and bad instances.

8.4 Experiments

In this section, we discuss the choice of algorithms and algorithm configurations. In addition, we discuss the metrics we use to evaluate classifier performance and present our experimental approach.

8.4.1 Algorithm Selection and Configuration

We want to determine if it is feasible to apply supervised learning algorithms to solve the EULA classification problem. In order to investigate the usefulness of different learning techniques we

include a diverse population of 17 algorithms from different paradigms, for example: perceptron and kernel functions, instance-based learners, Bayesian learners, decision tree inducers, meta-learners, rule inducers, and so forth. We use original Weka 3.5.7 algorithm implementations and, for most algorithms, we use the default configuration.

Table 8.2 Learning algorithm configurations.

Weka algorithm	Configuration
NaiveBayes	Kernel estimator: false, Supervised discretization: false
NaiveBayesMultinomial	N/A
RBFNetwork[a]	Ridge: 1.0E-8
SMO[b]	Kernel: Polynomial, Complexity: 1.0
VotedPerceptron	Exponent: 1.0, Max kernel alterations: 10,000
IBk[c]	Number of neighbors: 10, Distance weighting: false
KStar	Missing values treatment: average column entropy curves
AdaBoostM1	Classifier: DecisionStump
Bagging	Classifier: REPTree (Pruning: true)
Stacking	Meta: SMO, Committee: SMO, VotedPerceptron, NaiveBayes-Multinomial
HyperPipes	N/A
JRip[d]	Pruning: true, Number of optimizations: 2
PART	Binary splits: false, Pruning: true (confidence factor: 0.25)
Ridor	N/A
DecisionStump	N/A
J48[e]	Pruning: Subtree raising, Pruning confidence factor: 0.25
RandomForest	Number of trees: 10

[a] Radial Basis Function Network
[b] Support Vector Machines
[c] K-nearest Neighbor
[d] Ripper
[e] C4.5

The main emphasis in this study is not to find the optimal configuration for each algorithm, that is, the configuration for which the algorithm generates the best performing classifiers. Such an objective would most certainly require extensive parameter tuning. Instead, this study focuses on investigating if it is possible, in general, to distinguish between good and bad EULAs. Consequently, we do not perform any systematic parameter tuning at all.

We alter the configuration for the following algorithms: the number of neighbours (k) for IBk is changed in order to distinguish the algorithm from the 1-nearest neighbour (IB1) algorithm. We use $k = 10$ based on standard practice but point out that it is arbitrary. The k parameter can be optimized for a particular problem by systematically performing repeated training and testing runs with different k values. We use SMO as a meta classifier for the Stacking algorithm and include the following algorithms in the ensemble: Naive Bayes Multinomial, SMO, and VotedPerceptron. These three algorithms are selected based on their high performance, as reported in the pilot study. The specification of the key algorithm

configurations is provided in Table 8.2. The Weka default configurations [35] are provided for the sake of reproducibility, hence they will not be further described.

8.0.1 Evaluation of Classifier Performance

8.0.1.1 Classification Accuracy

We need to select relevant evaluation metrics in order to measure classifier performance. Traditionally, the accuracy metric (the number of correct classifications divided by the total number of classifications) has been widely used. However, several issues have been raised against the use of this particular metric as the only means to measure performance [27]. However, when used in conjunction with other metrics, we believe that accuracy is still a useful metric to consider.

8.0.1.2 The Areas Under the ROC Curve

In addition to accuracy, we therefore need to select metrics that are suitable for our particular application. For the purpose of EULA classification, we argue that the cost of misclassification is different for the two classes (good and bad). For example, classifying a bad EULA as good is sometimes far worse than the opposite and this is particularly true if the classification should be the basis for a decision support system that should aid the user in making the decision to install an application or to abort the installation.

If a good EULA is classified as bad, the user might think twice before installing the associated application and instead will try to find alternative applications. If, on the other hand, a bad EULA is classified as good, the user might install an application that contains spyware, believing that the application is legitimate. This is actually worse than not using any EULA classification at all, since the user is under the (false) impression of being informed about the legitimacy of the application. Consequently, EULA classification represents a problem in which different types of misclassification errors have asymmetric costs. This is important to keep in mind if one intends to develop a EULA-based anti-spyware application. Zhao [37] reviews this problem in detail and compares two cost-sensitive learning approaches, instance weighting and post hoc threshold adjusting. The results indicate that both approaches are suitable, but for different types of learners. Additionally, the author concludes

that symbolic learners are affected by cost-ratio changes to a larger extent than methods that produce continuous probability estimates.

In order to address the misclassification cost issue in this study, we consider four important metrics; true/false positives and true/false negatives, as documented in Table 8.3. In particular, we will use the Area under the ROC curve (AUC), which is based on the true positives rate (TPR) and the false positives rate (FPR) since it does not depend on equal class distribution or misclassification costs [35]. The calculation of, and motivation for, AUC is described in detail by Fawcett [15].

Table 8.3 Evaluation metrics.

Metric	Abbreviation	Meaning
True positives	TP	Good EULAs classified as good
False positives	FP	Bad EULAs classified as good
True negatives	TN	Bad EULAs classified as bad
False negatives	FN	Good EULAs classified as bad
True positives rate	TPR	TRP = TP / (TP+FN)
False positives rate	FPR	FPR = FP / (FP+TN)

8.0.0.1 Multi-criteria Evaluation

In addition to the ACC and AUC metrics we will evaluate each algorithm using the Candidate Evaluation Function (CEF) [22]. The purpose of CEF is to capture application-specific trade-offs by combining multiple relevant metrics. We shall now demonstrate how multi-criteria metrics can be used as an approach to trade-off some of the important aspects of EULA classification. CEF normalizes the metrics in order to get a uniform output domain and it is also possible to specify explicit weights for each metric to ensure that application-specific trade-offs can be properly represented. CEF itself does not dictate which metrics to use, it merely dictates how metrics are combined. Finally, CEF makes it possible to specify the acceptable range for each metric, pertaining to a particular application.

We define m_j as a metric with index j from an index set, J, over the selected set of metrics. Each metric is associated with a weight, w_j, and an acceptable range, $r = \left[b_j^l, b_j^u\right]$. The lower bound, b_j^l, denotes the least desired acceptable score. Similarly, the upper bound, b_j^u, denotes the desired score. Note that, in the original CEF definition a metric was normalized according to the best and worst score of that particular metric obtained from the studied set of classifiers. The current normalization uses the lower and upper

bound to generate a smooth distribution from 0 (least desired) to 1. CEF is now defined as specified in Equation 2.

$$CEF(c, D) = \begin{cases} 0 : \exists j(\overline{m}_j(c,D) < 0) \\ \sum j \in J^{w_j \overline{m}_j(c,D)} \quad otherwise \end{cases}$$

$$where \quad \sum j \in J^{w_j} = 1 \quad and \qquad (2)$$

$$\overline{m}_j(c,D) = \begin{cases} 1 : \frac{m_j - b_j^l}{b_j^u - b_j^l} > 1 \\ \frac{m_j - b_j^l}{b_j^u - b_j^l} \quad otherwise \end{cases}$$

To address the fact that a low FPR is more important than a high TPR for our particular application, we define an example CEF metric that combines m_1 =TPR with r_1 = [0.9,1.0] and m_2 =FPR with r_2 = [0.4, 0.0]. Furthermore, we let w_1 = 0.2 and w_2 = 0.8 to make m_2 four times more important than m_1. The weights and ranges are provided as examples to illustrate how CEF can be used to customize the evaluation for our application. However, these properties should be selected by domain experts after careful consideration, preferably using some reliable systematic approach.

8.0.0.2 Baseline Classifier

We have reviewed two possible data representations that can be used when applying supervised learning to EULA classification. Besides the comparison between these two representations, we need to quantify the utility of the EULA classification approach compared to the behaviour of an average user. In addition, we need to find out if it is even possible to discriminate between bad and good EULAs using the machine learning approach.

Consequently, we argue that it is necessary to compare the performance of the classifiers (generated from both data sets) to that of a baseline classifier. We choose to include Weka's ZeroR classifier for this purpose. ZeroR classifies instances based on the majority of the class distribution. For our particular case, this means that ZeroR will classify all instances as good. Our model of an average user assumes that the user does not read the EULA but instead continues with the installation. Thus, the behaviour of this model is equiv-

alent to that of ZeroR. We use AUC for all statistical comparisons. There are several rationales for this decision:

- the class distribution is skewed, which suggests that AUC is more appropriate than accuracy [27],
- the weights and ranges of the CEF metric are provided as examples and thus there is little sense in using the CEF results for statistical tests of significance, and
- no learned classifier should have an AUC less than 0.5 [14] and ZeroR always achieves an AUC score of 0.5, which makes ZeroR a simple and meaningful baseline if AUC is used.

ZeroR, which has no predictive power, can thus be used as a baseline to determine if a particular classifier has predictive power, i.e., if it can discriminate between good and bad EULAs. If, for a certain data set, at least one classifier is shown to have predictive power we can also assume more generally that it is possible to discriminate between bad and good EULAs using that data set.

8.1 Experimental Procedure

Since we have a limited amount of data for training and testing (996 instances), we choose to estimate performance using the average of ten stratified 10-fold cross-validation tests. We use Weka for all experiments and train the 17 learning algorithms on our two data sets. Moreover, we let X represent the set of performance results obtained from the bag-of-words set. Similarly, Y represents the set of performance results obtained from the meta EULA set. We formulate the following null hypotheses:

– $h_0^x : x = 0.5$

– $h_0^y : y = 0.5$

h_0^x is tested for each $x \in X$ and h_0^y is tested for each $y \in Y$ Both of the main hypotheses are tested using the corrected resampled t-test as recommended by Witten and Frank [35] for comparisons of two algorithms on the data set and using ten 10-fold cross-validation tests. Thus, we use 99 degrees of freedom and two-sided probability values. We are aware of the issues related to multiple hypothesis testing, cf. Demzar [11]. Thus, since we test 17 hypotheses for each data set, the family-wise error can be elevated. The

Bonferroni correction can be applied to maintain the family-wise error but, as Demzar notes, it is overly radical. We will therefore present results using both the regular probability value ($p < 0.05$) and the Bonferroni corrected value ($p < 0.001$), which is calculated by dividing the regular p-value by the number of tested hypothesis for each data set.

The third null hypothesis, h_0^z, is that there is no difference in performance between the bag-of-words and the meta EULA generated classifiers. This null hypothesis is tested using the two-sided Wilcoxon signed rank test. In this test, we compare the AUC result for each classifier on the bag-of-words data set to that of the corresponding classifier on the meta EULA data set. We reject h_0^z if the results for one data set are significantly better than those obtained from the other. The rationale for using a non-parametric test instead of, e.g., the paired t-test is that the latter assumes normality, which might be violated in our test [11]. In testing this hypothesis we are aware that the results can only apply for the selected algorithms and configurations.

8.2 Results

We first present the performance results for each data representation approach separately and then review the tested hypotheses. This is followed by an analysis and discussion about both approaches.

Table 8.4 Results on the bag-of-words data set.

Algorithm	Type	ACC	TPR	FPR	AUC	t
NaiveBayes	Bayes	0.874 (0.035)	0.881 (0.037)	0.193 (0.136)	0.873 (0.065)	4.204**
NaiveBayes-Multinomial	Bayes	0.939 (0.023)	0.951 (0.023)	0.173 (0.122)	0.926 (0.059)	5.040**
RBFNetwork	Function	0.928 (0.025)	0.950 (0.024)	0.276 (0.132)	0.798 (0.100)	2.708*
SMO	Function	0.950 (0.019)	0.980 (0.014)	0.335 (0.169)	0.822 (0.084)	3.192**
VotedPerceptron	Function	0.949 (0.021)	0.979 (0.016)	0.334 (0.146)	0.827 (0.076)	3.408**
IBk	Lazy	0.923 (0.014)	0.997 (0.006)	0.768 (0.126)	0.847 (0.078)	3.570**
KStar	Lazy	0.911 (0.009)	1.000 (0.001)	0.927 (0.080)	0.576 (0.043)	1,053
AdaBoostM1	Meta	0.942 (0.019)	0.975 (0.016)	0.369 (0.145)	0.875 (0.076)	3.909**
Bagging	Meta	0.942 (0.022)	0.980 (0.016)	0.416 (0.151)	0.887 (0.069)	4.233**
Stacking	Meta	0.948 (0.021)	0.978 (0.015)	0.334 (0.168)	0.822 (0.084)	3.192**
HyperPipes	Misc	0.947 (0.019)	0.989 (0.011)	0.446 (0.167)	0.827 (0.094)	3.065*
JRip	Rules	0.932 (0.027)	0.964 (0.024)	0.368 (0.163)	0.799 (0.083)	2.982*
PART	Rules	0.928 (0.023)	0.963 (0.021)	0.401 (0.154)	0.800 (0.100)	2.726*
Ridor	Rules	0.931 (0.021)	0.981 (0.018)	0.537 (0.190)	0.722 (0.093)	2.092*
DecisionStump	Trees	0.941 (0.021)	0.971 (0.017)	0.343 (0.136)	0.814 (0.069)	3.435**
J48	Trees	0.928 (0.021)	0.968 (0.015)	0.443 (0.169)	0.729 (0.127)	1,846
RandomForest	Trees	0.941 (0.017)	0.992 (0.009)	0.542 (0.149)	0.881 (0.068)	4.198**
Average		0.933 (0.021)	0.971 (0.017)	0.424 (0.147)	0.813 (0.080)	
ZeroR		0.904 (0.004)	1.000 (0.000)	1.000 (0.000)	0.500 (0.000)	

*p<0.05, two-tailed
**p<0.05, two-tailed, Bonferroni corrected

8.0.1 Bag-of-words Results

The evaluation results for the bag-of-words data set are presented in Table 8.4. The features evaluation scores for the 17 learning algorithms, the average learning algorithm score, and score of the baseline classifier. Each algorithm evaluation result is shown using four metrics: classification accuracy (ACC), true positives rate (TPR), false positives rate (FPR), and the Area Under the ROC Curve (AUC).

SMO, Voted Perceptron, and Stacking yield the highest ACC. However, the difference between the best and worst performing supervised algorithm is rather small (0.076). The accuracy of Naive Bayes Multinomial is mediocre and the regular Naive Bayes algorithm actually performs worse than the baseline.

With regard to AUC, the performance of the top ACC algorithms is mediocre while Naive Bayes Multinomial is the best performing algorithm. The difference in AUC between the best and worst performing algorithm is quite large (0.350). This difference is much due to the low performance of the KStar algorithm.

The TPR (the rate of correctly classified bad EULAs) is quite high for all algorithms, except the regular Naive Bayes algorithm. Unfor-

tunately the FPR (the rate of bad EULAs classified as good) is also relatively high, even for the best performing algorithms. Among the worst performing algorithms (according to FPR) are the two instance-based learners: KStar and IBk.

Table 8.5 Results on the meta EULA data set.

Algorithm	Type	ACC	TPR	FPR	AUC	t
NaiveBayes	Bayes	0.892 (0.024)	0.926 (0.025)	0.423 (0.131)	0.851 (0.059)	4.152**
NaiveBayesMultinomial	Bayes	0.812 (0.035)	0.828 (0.039)	0.343 (0.131)	0.797 (0.070)	3.226**
RBFNetwork	Function	0.919 (0.014)	0.987 (0.013)	0.711 (0.128)	0.846 (0.059)	4.093**
SMO	Function	0.921 (0.010)	1.000 (0.000)	0.821 (0.101)	0.590 (0.051)	1.145
VotedPerceptron	Function	0.904 (0.004)	1.000 (0.000)	1.000 (0.000)	0.500 (0.000)	0.000
IBk	Lazy	0.929 (0.016)	0.995 (0.007)	0.686 (0.134)	0.825 (0.071)	3.505**
KStar	Lazy	0.900 (0.024)	0.960 (0.021)	0.662 (0.137)	0.732 (0.083)	2.314*
AdaBoostM1	Meta	0.934 (0.017)	0.978 (0.018)	0.479 (0.136)	0.809 (0.073)	3.286**
Bagging	Meta	0.939 (0.017)	0.984 (0.013)	0.484 (0.139)	0.869 (0.062)	4.258**
Stacking	Meta	0.921 (0.010)	1.000 (0.000)	0.821 (0.101)	0.590 (0.051)	1.145
HyperPipes	Misc	0.923 (0.012)	0.997 (0.006)	0.769 (0.110)	0.684 (0.053)	2.297*
JRip	Rules	0.938 (0.016)	0.984 (0.014)	0.490 (0.140)	0.747 (0.068)	2.722*
PART	Rules	0.937 (0.017)	0.984 (0.013)	0.498 (0.141)	0.820 (0.076)	3.335**
Ridor	Rules	0.932 (0.016)	0.984 (0.013)	0.554 (0.163)	0.715 (0.079)	2.198*
DecisionStump	Trees	0.934 (0.017)	0.978 (0.018)	0.479 (0.136)	0.749 (0.065)	2.806*
J48	Trees	0.936 (0.016)	0.982 (0.014)	0.495 (0.138)	0.742 (0.076)	2.522*
RandomForest	Trees	0.929 (0.019)	0.977 (0.017)	0.524 (0.128)	0.807 (0.074)	3.243**
Average		0.918 (0.017)	0.973 (0.014)	0.602 (0.123)	0.745 (0.063)	
ZeroR		0.904 (0.004)	1.000 (0.000)	1.000 (0.000)	0.500 (0.000)	

*p<0.05, two-tailed
**p<0.05, two-tailed, Bonferroni corrected

8.0.1 Meta EULA Results

The evaluation results for the meta EULA data set are presented in Table 8.5 using the same setup as the Table 8.4.

The best performing algorithms are tree and rule learners, as well as meta learners that use tree learners in their ensemble. For ACC, the best performing algorithms are: Bagging, JRip, PART, and J48. The difference between the best and worst performing algorithm, according to ACC, is large (0.128). The accuracy of Voted Perceptron is identical to that of ZeroR. Meanwhile, Naive Bayes, Naive Bayes Multinomial, and KStar all perform worse than ZeroR.

In terms of AUC, the top ACC algorithm (Bagging) is the best performing algorithm. The difference in AUC between the best and worst performing algorithm is very large (0.369). The worst performing algorithms, according to AUC, are: Stacking, SMO, and Voted Perceptron. However, these three algorithms all share perfect TPR scores. The low AUC score is due to their high FPR scores. Similarly to the bag-of-words data set, the lowest FPR scores are achieved by the Bayesian algorithms.

8.0.2 Tested Hypotheses

The results from the two first main hypotheses are indicated in Table 8.5, respectively. At $p < 0.05$, 15 out 17 algorithms perform significantly better than the baseline on the bag-of-words set. For the meta EULA set, 14 out 17 algorithms perform significantly better than the baseline for the same confidence level. More generally, we can therefore conclude that both data sets can be used for training classifiers that can discriminate between good and bad EULAs.

The two-sided p-value of obtaining our results if h_0^z holds is 0.01675, which means that we can reject this null hypothesis. We therefore conclude that the bag-of-words model is statistically significantly better than the meta EULA model, at least for the included algorithms.

8.0.3 CEF Results

We now review the results from the CEF evaluation. Table 8.6 shows the CEF score for each algorithm and data set. The scores for the TPR metric (m_1) and the FPR metric (m_2) have been calculated using the TPR and FPR scores from Table 8.4 and Table 8.5, respectively. As can be observed, several algorithms fail completely, since they do achieve an acceptable score for at least one of the metrics. Eight algorithms manage to achieve a valid CEF score on the bag-of-words data set. However, most decision tree and rule learners fail since they cannot achieve an acceptable FPR. The Bayesian learners are exceptionally strong with regard to FPR but the Naive Bayes algorithm fails to achieve an acceptable TPR. For the meta EULA data set, all algorithms fail. The only algorithm that achieves a good enough FPR is Naive Bayes Multinomial, however it fails because of the unacceptable TPR score.

Table 8.6 CEF evaluation results.

Algorithm	Bag-of-words			Meta EULA		
	m_1	m_2	CEF	m_1	m_2	CEF
NaiveBayes	0.000	0.518	0.000	0.260	0.000	0.000
NaiveBayesMultinomial	0.510	0.568	0.556	0.000	0.143	0.000
RBFNetwork	0.500	0.310	0.348	0.870	0.000	0.000
SMO	0.800	0.163	0.290	1.000	0.000	0.000
VotedPerceptron	0.790	0.165	0.290	1.000	0.000	0.000
IBk	0.970	0.000	0.000	0.950	0.000	0.000
KStar	1.000	0.000	0.000	0.600	0.000	0.000
AdaBoostM1	0.750	0.078	0.212	0.780	0.000	0.000
Bagging	0.800	0.000	0.000	0.840	0.000	0.000
Stacking	0.780	0.165	0.288	1.000	0.000	0.000
HyperPipes	0.890	0.000	0.000	0.970	0.000	0.000
JRip	0.640	0.080	0.192	0.840	0.000	0.000
PART	0.630	0.000	0.000	0.840	0.000	0.000
Ridor	0.810	0.000	0.000	0.840	0.000	0.000
DecisionStump	0.710	0.143	0.256	0.780	0.000	0.000
J48	0.680	0.000	0.000	0.820	0.000	0.000
RandomForest	0.920	0.000	0.000	0.770	0.000	0.000

8.1 Discussion

The experiment raises several interesting issues which will now be discussed. We first review some technical aspects related to the featured algorithms and their performance on EULA classification.

The best performing algorithms for the bag-of-words data set, according to ACC, are predominantly kernel function based. SMO, especially, is known to perform well according to ACC on large text classification data sets [20]. However, in terms of AUC, both Voted Perceptron and SMO are outperformed by Naive Bayes Multinomial and Bagging.

Caruana and Niculescu-Mizil [8] note that support vector machines are not designed to predict probabilities and explain that their predictions can be calibrated with, e.g.: Platt Scaling or Isotonic Regression. We perform no such calibration and thus this could be related to the poor SMO performance according to AUC.

Several studies have reported on the poor performance of Naive Bayes Multinomial compared to that of, e.g., SMO on text classifi-

cation tasks. However, Kibriya et al. [20] argued that the performance, especially in terms of AUC, of Naive Bayes Multinomial can be dramatically improved by applying TF IDF during preprocessing. Our results on the bag-of-words data set indicate that this claim is valid. Consequently, the obtained results seem to be aligned with related studies of these algorithms for similar domains.

The case is different for the meta EULA data set, which has a small number of attributes compared to the bag-of-words set. Few algorithms seem to perform well on this data set with a no exception in Bagging.

Bagging seems to be quite suitable for our problem. It is the superior algorithm for both ACC and AUC on the meta EULA data set. In addition, it is the second best algorithm, according to AUC, on the bag-of-words data set and it is positioned in the upper echelon of algorithms for the ACC metric on this data set as well. However, compared to the Bayesian learners it performs poorly with respect to FPR, which is an important metric for this application.

The default configuration of Bagging uses REP trees as base classifiers. In analyzing the REP trees generated from the complete meta EULA set we observe that they seem to be very diverse in structure but most are rather complex in terms of tree size (most of them use more than 30 nodes). The aggregated Bagging classifier can thus be described as a quite complex model. This seems to be aligned with the essential property of Bagging as Breiman [6] concludes: if perturbing the learning set can cause significant changes in the constructed classifier, then Bagging can improve performance. Consequently, the REP trees learned from the different bootstrap generated sub sets of data are diverse and this may contribute to the overall good performance.

As stated earlier, most algorithms perform poorly on the meta EULA set and this is especially true for Voted Perceptron and SMO. The former performs identically to that of ZeroR and SMO is only slightly better. Moreover, all algorithms perform poorly according to FPR on the meta EULA set, which suggests that it is difficult to find structural patterns in data compiled from the EULA Analyzer metrics.

It is not always trivial to determine what triggers a certain classification, mainly due to the opaque nature of most of the well-perform-

ing classifiers. However, the rule based classifiers share a substantial number of words used for classification, as can be seen in Table 8.7.

Table 8.7 Rule-based classifiers generated using the complete bag-of-words data set.

Algorithm	Classifier
JRip	(**search** >= 1.67) AND (**upgrad** >= 1.01) THEN bad (**web** >= 2.10) AND (**uninst** >= 1.42) THEN bad (**opportun** >= 1.89) AND (**adver** >= 1.34) THEN bad ELSE good
PART	(**visit** <= 1.48) AND (**search** <= 0) AND (**opportun** <= 0) AND (**ad** <= 1.19) AND (**graph** <= 0) AND (**definit** <= 0) AND (**view** <= 0) THEN good (**contextu** > 0) THEN bad (**redirect** <= 0) AND (**panel** > 0) AND (**fraud** <= 0) THEN bad (**redirect** <= 0) AND (**cook** <= 0) AND (**vi** <= 1.50) AND (**timel** <= 0) AND (**patch** <= 0) AND (**pc** <= 0) AND (**tot** <= 0) THEN good (**redirect** <= 0) AND (**zip** <= 0) AND (**accru** <= 0) AND (**explicit** <= 1.61) AND (**adver** <= 3.10) AND (**detect** <= 0) AND (**asser** <= 0) AND (**injunct** <= 0) AND (**encour** <= 0) AND (**born** <= 0) THEN good (**charg** <= 0.99) AND (**rever** <= 0.21) THEN bad (**tort** <= 0) AND (**pop** <= 0) THEN good ELSE bad
Ridor	(**search** > 0.84) AND (**upgrad** > 0.50) AND (**vulgar** <= 1.24) THEN bad (**inform** > 0.93) AND (**download** > 1.88) THEN bad (**pol** > 0.49) AND (**visit** > 1.91) AND (**whichever** <= 0.93) THEN bad ELSE good

Table 8.8 shows the corresponding rules generated from the meta EULA set. The classifier representations for the latter data set are more simple than those obtained from the bag-of-words data set, mainly for two reasons: (i) the bag-of-words data set is more complex in terms of the number of attributes and (ii) the information gain of the meta EULA attributes seem to be low in general except for flagged sentences. The consequence of the second point is that most tree and rule learners build very simple classifiers, based solely on the flagged sentence attribute.

We analyze the rules and trees generated from the bag-of-words data set in terms of which word stems are associated with a classification of EULAs as bad. Not surprisingly, some frequently used stems are: *search*, *upgrad*, *web*, *uninst*, *opportun*, and *adver*. These words stems can easily be associated with typical concepts that would be mentioned by spyware vendors. In summary, the bag-of-words model seems to be more suitable as a basis for EULA classification. However, even for this data set, many supervised learners achieve

FPR scores too high to be acceptable in a real-world decision support system. We hypothesize that a larger collection of bad instances along with careful parameter tuning of the algorithms might contribute to decrease this rate. Additionally, several approaches have been presented to increase the performance on imbalanced data sets. For example, Wang and Japkowicz [33] use ensembles of SVMs to increase the classification performance for both the majority and minority class. Since EULA-based spyware detection is typically based on imbalanced data, it may be important to investigate such approaches further in future work.

Table 8.8 Rule-based classifiers generated using the complete meta EULA data set.

Algorithm	Classifier
JRip	(**flagged_sentences** >= 10) THEN bad ELSE good
PART	(**flagged_sentences** <= 9) AND (**flagged_sentences** <= 2) AND (**automated_readability_index** <= 20) THEN good (**flagged_sentences** <= 9) THEN good ELSE bad
Ridor	(**flagged_sentences** > 9.5) THEN bad ELSE good

8.0.1 A Novel Tool for Spyware Prevention

Assuming that the EULA classification approach can make distinctions between good and bad software, we would like to use the approach to help users make informed decisions about the software they download. We therefore suggest that the EULA classification method outlined in this paper can be incorporated in a spyware prevention tool.

This tool could presumably be designed as a middleware that operates between the operating system and the application installer software. Furthermore, we suggest that it could be executed as a background process that identifies and analyzes EULAs as they appear on the screen during an installation. To accomplish this automatic identification, the tool can plant a hook into the operating system function for displaying dialogues.

Based on the classification of the EULA, the tool can provide the user with a recommendation about whether to install the application or not. This allows the tool to assist users in making informed decisions about the installation of software without forcing them to

read (and understand) the lengthy and intricate EULAs. For example, if the tool classifies the EULA as bad, the user can take appropriate actions against it, e.g., by disagreeing with the EULA and exiting the installation process.

It should be noted that any tool based on our method should not be used in isolation, but rather as a complement to other approaches, e.g., anti-spyware software.

In addition, tree and rule learners generate classifiers that can be used for visualizing the decision process. Despite the fact that this category of learners does not seem to be the most appropriate at solving the classification task, their visualization could increase the number of correct classifications since they may allow the user to make a decision based on more information than what is provided by the opaque learners.

We outline basic requirements for the imagined prevention tool as follows: first, we need to make sure that the tool is accurate in its classifications since this is the main functionality. The tool should essentially be able to detect all, or a very high quantity of, bad software but it is also desirable that it manages to classify good software correctly.

Furthermore, we need the tool to respond rather quickly when an application presents a EULA.

However, the actual training phase could be performed on a central server and the generated classifier(s) could then be downloaded by the tool periodically. Thus, there are no specific requirements related to classifier training time. Finally, it is desirable that the tool can visualize what element(s) in the EULA triggered a certain classification.

8.0.2 Potential Problems

It could be argued that, if the prevention tool is made available, the spyware authors would tailor their EULA around it. We believe that this argument does not hold since, in order to avoid legal repercussions, the spyware authors are in most countries required to mention in the EULA that spyware will be installed. We exploit this fact and use it against the spyware distributors.

Another argument against the tool is that there are already quite sophisticated tools for prevention and removal of spyware (e.g.: Ad-Aware[10]). However, the idea is not to create a replacement for such products. Essentially, the spyware prevention tool should work as a complement that could be used to detect spyware that has not yet been classified by anti-spyware software.

8.0.3 Comparison to Ad-aware

In an earlier experiment Boldt and Carlsson [4] analyzed the accuracy of Ad-Aware over a four year period. By manually analyzing spyware infested systems using a leading commercial computer forensic tool Boldt and Carlsson tracked down every change in the infected system. By comparing these changes with the detected spyware components found by Ad-Aware it was possible to quantify the accuracy development of the tool between 2002 and 2005.

The results show that Ad-Aware failed to report 30 per cent of the spyware programs. New versions would mark earlier discovered spyware programs as legitimate programs, or wrongly classified traces of spyware as fully functioning spyware programs. However, the manual forensic method managed to find all added executable files and also to differentiate traces of spyware from executable files. The conclusion is that the problem of identifying and removing spyware programs while at the same time keeping/protecting legitimate files, is difficult to solve for anti-spyware vendors. Not only do these vendors have to cope with technical problems, but they also need to consider legal aspects, which is a major distinction between anti-spyware and anti-virus tools.

When it comes to comparing the accuracy of our novel technique with existing anti-spyware tools some complicating factors emerge. We have used already known spyware programs included in SpywareGuide.com. To enable a fair comparison between our proposed tool and already available anti-spyware tools we need to obtain data from the shared repositories of the leading anti-virus companies. It is presently unclear whether the data available in these repositories would be made available to researchers for the purpose of conducting these experimental comparisons.

10. http://www.lavasoft.com

8.0.4 Conclusions and Future Work

We have investigated the relationship between the contents of End User License Agreements (EULAs) and the legitimacy of the associated software applications. For this purpose, we collected a data set that features 996 EULA instances of legitimate (good) and spyware associated (bad) software. This is a text classification task and we argue that supervised learning is a suitable approach to the problem. Since most supervised learning algorithms cannot handle unstructured text input, we had to convert the data set to a manageable format. We therefore opted to use the bag-of-words model, in which text documents are converted to feature vectors. We compared this model to a meta EULA model that describes each EULA using several text analysis metrics.

We applied 17 supervised learning algorithms from several algorithm categories, such as: kernel functions, instance-based learners, tree and rule inducers, and Bayesian learners. The main objective was to investigate the possibility to classify software as good or bad by training classifiers on the associated EULAs.

For both data models, the experimental results show that a majority of the featured algorithms significantly outperformed a baseline classifier based on majority class decision. However, the results indicate that the bag-of-words model was more suitable than the meta EULA model, at least for the studied algorithms.

The results support our hypothesis that EULAs can indeed be used as a basis for classifying the corresponding software as good or bad. Based on this, we conclude that it would be quite possible to use the EULA classification method in a spyware prevention tool that classifies the EULA when it is shown to the user during an application installation. The result from such an analysis gives the user a recommendation about the legitimacy of the application before the installation continues. There are several directions for future work. For example, we intend to:

– gather a data set that includes both the EULAs and binaries of spyware and legitimate software in order to perform a fair comparison between our spyware detection approach and the detection approaches provided by the leading commercial anti-virus and anti-spyware tools,
– extend the EULA classification problem from two classes into

a more fine-grained multi-class approach, thus enabling not only the distinction between spyware and legitimate software but also the detection of specific types of spyware,
– select metrics, weights, and ranges for CEF in a more informed manner, e.g., by interviewing experts and potential users,
– develop a spyware prevention tool, based on EULA classification, that can help users to make informed decisions about the software they install,
– investigate metric-based algorithms and other approaches to optimize the CEF metric, including to minimize the false positives rate,
– select a subset of the best performing learning algorithms in order to perform parameter tuning for the purpose of improving classification performance even further,
– merge the meta EULA and the bag-of-words data sets to find out if the classification performance can be improved by having access to both the term frequencies and the meta information,
– compare the contents of the flagged sentences with the words discovered for EULAs classified as bad in the bag-of-words.

Acknowledgements. We thank the anonymous reviewers for their useful comments and suggestions. This work was funded by Blekinge Institute of Technology.

8.1 References

[1] Androutsopoulos I, Paliouras G, Karkaletsis V, Sakkis G, Spyropoulos CD, Stamatopoulos P (2000) Learning to filter spam E-mail: A comparison of a naive bayesian and a memory-based approach. In: 4th European Conference on Principles and Practice of Knowledge Discovery in Databases: Workshop on Machine Learning and Textual Information Access, Springer, Berlin / Heidelberg, Germany, pp 1–13

[2] Arnett KP, Schmidt MB (2005) Busting the ghost in the machine. Communications of the ACM, 48(8)

[3] Boldt M (2007) Privacy-invasive Software – Exploring Effects and Countermeasures, Licentiate Thesis Series, No 2007:01, Blekinge Institute of Technology, Sweden.

[4] Boldt M, Carlsson B (2006) Analysing Countermeasures Against Privacy-Invasive Software. In: 1st IEEE International Conference on Systems and Networks Communications

[5] Boldt M, Carlsson B, Jacobsson A (2004) Exploring spyware effects. In: Eight Nordic Workshop on Secure IT Systems, Helsinki University of Technology, Espoo, Finland, no. TML-A10 in Publications in Telecommunication and Software Multimedia, pp 23–30

[6] Breiman L (1996) Bagging predictors. Machine Learning 24(2):123–140

[7] Carreras X, Márquez L (2001) Boosting trees for anti-spam email filtering. In: Mitkov R, Angelova G, Bontcheva K, Nicolov N, Nikolov N (eds) European Conference on Recent Advances in Natural Language Processing, Tzigov Chark, Bulgaria, pp 58–64

[8] Caruana R, Niculescu-Mizil A (2006) An empirical comparison of supervised learning algorithms. In: 23rd International Conference on Machine Learning, ACM Press, New York City, NY, USA, pp 161–168

[9] Cohen W (1996) Advances in Inductive Logic Programming, IOS Press, Amsterdam, the Netherlands, chap Learning Rules that Classify E-Mail

[10] Coleman M, Liau TL (1975) A computer readability formula designed for machine scoring. Journal of Applied Psychology 60:283–284

[11] Demzar J (2006) Statistical comparisons of classifiers over multiple data sets. Machine Learning Research 7:1–30

[12] Denoyer L, Zaragoza H, Gallinari P (2001) HMM-based passage models for document classification and ranking. In: 23rd European Colloquium on Information Retrieval Research

[13] Drucker H, Wu D, Vapnik V (1999) Support vector machines for spam categorization. IEEE Transactions on Neural Networks 10(5):1048–1054

[14] Fawcett T (2001) Using rule sets to maximize ROC performance. In: IEEE International Conference on Data Mining, IEEE Press, New York City, NY, USA, pp 131–138

[15] Fawcett T (2003) ROC graphs – notes and practical considerations for data mining researchers. Tech. Rep. HPL-2003-4, Intelligent Enterprise Technologies Laboratories, Palo Alto, CA, USA

[16] Feldman R, Sanger J (2007) The Text Mining Handbook. Cambridge University Press, Cambridge, MA, USA

[17] Flesch R (1948) A new readability yardstick. Journal of Applied Psychology 32:221–233

[18] Good N, Grossklags J, Thaw D, Perzanowski A, Mulligan DK, Konstan J (2006) User choices and regret: Understanding users' decision process about consensually acquired spyware. I/S Law and Policy for the Information Society 2(2):283–344

[19] Kang N, Domeniconi C, Barbara D (2005) Categorization and keyword identification of unlabeled documents. In: Fifth IEEE International Conference on Data Mining, IEEE Press, New York City, NY, USA, pp 677–680

[20] Kibriya AM, Frank E, Pfahringer B, Holmes G (2004) Multinomial naive bayes for text categorization revisited. In: Seventh Australian Joint Conference on Artificial Intelligence, Springer, Berlin / Heidelberg, Germany, pp 488–499

[21] Koprinska I, Poon J, Clark J, Chan J (2007) Learning to classify E-mail. Information Sciences 177:2167–2187

[22] Lavesson N, Davidsson P (2008) Generic methods for multi-criteria evaluation. In: Eighth SIAM International Conference on Data Mining, SIAM Press, Philadelphia, PA, USA, pp 541–546

[23] Lavesson N, Davidsson P, Boldt M, Jacobsson A (2008) New Challenges in Applied Intelligence Technologies, Studies in Computational Intelligence, vol 134, Springer, Berlin / Heidelberg, Germany, chap Spyware Prevention by Classifying End User License Agreements

[24] McFedries P (2005) The spyware nightmare. IEEE Spectrum 42(8):72–72

[25] Metzler D, Croft WB (2005) A markov random field model for term dependencies. In: 28th ACM SIGIR Conference on Research and Development in Information Retrieval, pp 472–479

[26] Moshchuk A, Bragin T, Gribble SD, Levy HM (2006) A crawler-based study of spyware on the web. In: 13th Annual Symposium on Network and Distributed Systems Security, Internet Society, Reston, VA, USA

[27] Provost F, Fawcett T, Kohavi R (1998) The case against accuracy estimation for comparing induction algorithms. In: 15th International Conference on Machine Learning, Morgan Kaufmann Publishers, San Francisco, CA, USA, pp 445–453

[28] Sakkis G, Androutsopoulos I, Paliouras G, Karkaletsis V, Spyropoulos CD, Stamatopoulos P (2001) Stacking classifiers for anti-spam filtering of E-mail. In: Sixth Conference on Empirical Methods in

Natural Language Processing, Carnegie Mellon University, Pittsburgh, PA, USA

[29] Sebastiani F (2002) Machine learning in automated text categorization. ACM Computing Surveys 34(1):1–47

[30] Shukla S, Nah F (2005) Web browsing and spyware intrusion. Communications of the ACM 48(8)

[31] Smith EA, Kincaid P (1970) Derivation and validation of the automated readability index for use with technical materials. Human Factors 12:457–464

[32] Wang P, Hu J, Zeng H-J, Chen Z (2009) Using Wikipedia knowledge to improve text classification. Knowledge and Information Systems 19: 265–281

[33] Wang BX, Japkowicz N (2009) Boosting support vector machines for imbalanced data Sets. Knowledge and Information Systems, Online First

[34] Weiss A (2005) Spyware be gone. ACM Networker 9(1):18–25

[35] Witten IH, Frank E (2005) Data Mining: Practical Machine Learning Tools and Techniques. Morgan Kaufmann Publishers, San Francisco, CA, USA

[36] Zhang X (2005) What do consumers really know about spyware? Communications of the ACM 48(8):44–48

[37] Zhao H (2008) Instance weighting versus threshold adjusting for cost-sensitive classification. Knowledge and Information Systems 15: 321–334

PUBLICATION

6

On the Simulation of a Software Reputation System

5th International Workshop on Frontiers in Availability, Reliability and Security, 2010

Martin Boldt, Anton Borg and Bengt Carlsson

Today, there are difficulties finding all malicious programs due to juridical restrictions and deficits concerning the anti-malicious programs. Also, a "grey-zone" of questionable programs exists, hard for different protection programs to handle and almost impossible for a single user to judge. A software reputation system consisting of expert, average and novice users are proposed as a complement to let anti-malware programs or dedicated human experts decide about questionable programs. A simulation of the factors involved is accomplished by varying the user groups involved, modifying each user's individual trust factor, specifying an upper trust factor limit and accounting for previous rating influence. As a proposed result, a balanced, well-informed rating of judged programs appears, i.e. a balance between quickly reaching a well-informed decision and not giving a single voter too much power.

9.1 Introduction

Today several hundred thousands of software programs exist, making it almost impossible for a single user to by herself decide what is good and what is bad. Of course tools to prevent and remove viruses and spyware have existed for a long time, but not all malicious programs are found due to juridical restrictions, i.e. the legal status of these applications are questioned, placing them in an grey-zone between good and bad software. This results in a large amount of applications that anti-malware developers are being cautious about removing, due to the potential for legal retribution. So, a "grey-zone" of questionable programs exists, hard for different protection program to handle and almost impossible for a single user to judge. Also, the availability of preventive software has been limited, already installed malicious software are found and removed but then the damage might already be done.

The inability of traditional anti-malware applications to handle, due to restrictions put upon them, the programs that exist in the previously mentioned grey-zone, leaves users unprotected. A complement, to using anti-malware software for deciding about unwanted programs, is to use a reputation system, i.e. ranking of new and previous unfamiliar software as a method for investigating the "true" appearance of a program. Using professional experts for doing this is both expensive and unrealistic due to the huge amount of non-investigated programs. Instead we propose a pool of ordinary users with different skills making necessary decisions about the quality of different software. However, there is still a need for more traditional anti-malware tools for targeting the clear-cut malware types that by no means could be placed inside the "grey-zone" between good and bad software, such as viruses and worms.

The purpose of this work is to investigate how many and what impact expert users need to have on a reputation system making it reliable, i.e. if it is possible to get a stable system by having few experts compensating for a vast majority of users with limited ability to rate an application. We simulate a reputation system with input from different skilled users and investigate a way of mitigating bad user ratings by using trust factors rewarding good users' good actions and punishing bad actions. The goal of the simulation is to find critical parameters for correctly classifying large number of different programs with a realistic base of different skilled users.

The remaining part of this paper is organized as follows. First we discuss the related work in Section II and introduce the software reputation system in Section III. We continue in section IV by introducing the simulator software and in Section V we present the scenarios. In Section VI we present our results, which then are discussed in Section VII. We conclude by stating our conclusions and suggestions for future work in Section VIII and IX respectively.

9.2 Related Work

Recommender systems are used to provide the user with an idea of other users' thoughts about products, i.e. whether they are good or bad. These kinds of systems are mostly used in commercial websites suggesting additional products, which the user might consider buying, exemplified in Amazon [2]. Recommender systems are not limited to commercial services, but also exist in other recommendation services such as Internet Movie Database (IMDb) [6]. IMDb uses recommender systems to suggest movies to users based on the opinions of other users that have shown similar tastes. Adomavicius and Tuzhilin provide, in their survey on the subject, a deep introduction of recommender systems, as well as some of the limitations [1].

eBay [5] makes use of a reputation system that allows users to rate buyers and sellers within the system, as a way to establish reputation among users. This reputation system makes it easier for users to distinguish dishonest users from trustworthy users. Experiments conducted by Resnick et al. also show that users with a higher reputation have a higher likelihood to sell items [10]. So, while recommender systems deals with the items involved, reputation systems instead deals with the involved users. In this paper we refer to our system as a reputation system due to the importance of the trust factors associated with the users.

Since reputation systems rely on the input of the users to calculate the ratings, it has to be able to establish trust between users and towards the system [7][11]. This is especially important when one considers the fact that the users of a software reputation system will have varying degrees of computer knowledge, and their ability to rate an application will thus be of different quality. There also exists the possibility of a user acting as several agents and actively reporting an erroneous rating in order to give a competitor a bad reputa-

tion or increase rating of a chosen object, i.e. a Sybil attack [4]. Even though this can be a potential problem to our proposed system, it is not within the scope of this paper to further analyze such scenarios. Furthermore there exist proposed solutions to this problem, for instance SybilGuard by Yu et al. [15].

The problem of erroneous ratings will, in a system such as IMDb, correct itself over time, but in a system such as the one proposed by Boldt et al. [3], where the intent is to advice on malicious software to users who might not be able to tell the difference, this presents a greater problem. Whitby et al. has put forth an idea of how to solve this problem [14] by filtering the unfair ratings, and their simulations show that the idea has merit. Traupman and Wilensky [13] try to mitigate the effects of false feedback in peer-to-peer applications by using algorithms to determine a user's true reputation. However, these ideas might not be ideal under all circumstances, as they add another layer of complexity to the system, as well as another step of work to be done.

Jøsang et al. [7] summarize, among other things, different ways of computing the rating of an object and one of the conclusions is that reputation systems originating from academia have a tendency to be complex compared to industrial implementations. We have opted for a simpler system, where the rating is weighted by trustworthiness of the user.

Among simulations done on the area of reputations systems, Jøsang et al. has conducted a simulation on an e-market, concluding that reputation systems are necessary in order for the e-market to become healthy [8]. They also come to the conclusion that reputation systems should be configured to forget old ratings in order for new ratings to have impact, i.e. the system should be able to change opinion concerning an object.

9.3 Software Reputation System

As presented in the previous section, ranking of new and previous unfamiliar software is a common method for investigating the "true" appearance of a program before installing it. Using professional experts for doing this is both expensive and unrealistic due to the huge amount of software programs that are developed every year. Instead the opinions are gathered from a pool of ordinary vol-

untary users that agree to benefit from the common knowledge by providing ratings for the software they are most familiar with. In this way each participant is asked to rate software on a discrete scale (1 to 10) after they have used that software during a certain time-frame, i.e. the user have had time to form an opinion about that particular software program. The ratings given by the system users should be all-embracing, i.e. including different parts such as (but not limited to) the software's features, behaviour, usability, stability, performance and privacy aspects.

9.3.1 System Design

We propose a client-server based system where each user has a small client software installed through which it is possible for the user to both send and retrieve information from the central server that handles all software reputation calculations. The client identifies software by calculating hash digests on the actual program file data, e.g. SHA-256. This means that a software reputation is associated with each new version of a program, but the reputation of several subversions can be propagated up to one major version that is then presented to the user. It is also possible to calculate for instance the average rating of a certain software vendor based on the individual reputations of all programs that a particular vendor has developed.

In an attempt to get as accurate ratings as possible from the user, the client software asks the user to rate the software he/she uses most frequently, i.e. the user is familiar with the software and has an opinion about it to base the rating on. Each rating includes one mandatory field that represent an overall rating on some grading scale, in this case [1,10] inclusive. It is also possible for the user to provide additional information, but this is optional. We believe it is of great significance not to ask the users to provide too much information since many users would find this most annoying, and therefore provide random or no feedback at all. However, we believe computer users would accept to rate a few software per month if they in return get access to all previous users' ratings for software programs that the user is considering installing.

To address the ever-existing problem with participants that provide false information to a collaborative system we incorporate user-individual *trust factors* (TF). This means that each user is assigned a TF that states the amount of influence each user have in the system.

New users are always given the lowest possible TF, but as they use the system and prove to be trustworthy this value increases. Each time a user uses the client program to submit a rating it is forwarded to the reputation server for further processing. On the server-side the rating is compared to the average of all previous ratings on that particular software and if it is close then the user's TF is increased, otherwise it is decreased. That way the TF of users that provide accurate ratings increases which give them more influence in the system, while it decreases for the rest of the users. Although the effect of this implementation is that the input from some users is amplified to dominate a large portion of the overall system, we believe it is important to include *all* users' votes when calculating the resulting software ratings in the system. This way, even non-expert users such as novice and new users can rest assured that their voice is listened to.

It is of significant importance to make sure the users' privacy is sufficiently protected in a software reputation system, since it handles sensitive information about what software each user have installed on their computer and their associated ratings. A situation where it would be possible to combine IP addresses with the information about what software these computers include could for instance reveal which computers that are vulnerable to certain remotely exploitable vulnerabilities. In addition to this it is also important to protect users' privacy since one of the main goals of a software reputation system itself is to assist users in protecting against potentially malicious programs that invade privacy. It is therefore important to make sure that the system does not intrude on users' privacy more than absolutely necessary. However, we still need to store some minimal amount of information about the user to address the problem with vote flooding, i.e. we need to distinguish between unique users' votes for each software to guarantee that duplicate votes do not occur. A thorough description of the techniques and design choices used for this software reputation system is available in [3].

9.4 Software Reputation System Simulator

In this section we start by describing the design and workings of the simulator and then move on to explain how we modelled the users in our experiments.

9.4.1 Simulator Design

The simulator itself was implemented in Java and all configuration of the simulator is carried out through configuration files that allow the operator to fine-tune every aspect of the scenario that should be simulated. The simulator is deterministic, meaning that it is possible to rerun a scenario several times and always get the same results, or more interestingly to change a certain variable in the scenario setup and be sure that the changes in the end-result are due to the alteration of that particular variable.

Individual objects represent users and software that are simulated, i.e. one Java object per simulated user and software. These objects are stored in two different linked lists that keep track of all user and software objects. A simulation basically consists of iterating through the linked list of all user objects in sequence, allowing each user to rate a randomly selected software object, until the correct number of votes has been simulated. An important addition to this process is that the linked list of all user objects is shuffled before each iteration proceeds. At certain intervals, for instance every 10% progress, the simulator outputs various degrees of statistics depending on the particular configuration.

Each software object includes variables that store information about its "correct" rating, the number of votes it has received and the sum of all weighted votes, which makes it possible to calculate the software's weighted average rating as explained in Equation 2 in the next subsection. The "correct" rating mentioned above is used for two purposes in our simulator. First, it is used for evaluating the accuracy of the simulated reputation system. Even though such a correct rating might not exist in the real world due to users' subjective beliefs, we use them as a way to evaluate the accuracy of the simulated reputation system. Secondly it is also used when constructing the user's vote as described in the next subsection.

The evaluation of a simulation consists of summarizing the absolute distance between software's correct rating and weighted average rating, and finally dividing it with the number of software included in the simulation. The resulting value is the *evaluation score* (ES), i.e. the average distance from all software programs' correct ratings. This score represent how accurate the simulated software reputation system is when providing software ratings to its users. One should always strive to reach an as low ES as possible, since an ES of 0.0

represent that the software reputation system on average provides its users with ratings that are 0.0 votes from its correct value. In other words bang on target. In the next subsection we present how the users in the simulations are modelled.

9.4.2 User Models

We have divided the simulated users into three groups based on their technical knowledge and accuracy in rating software. Each user simulated belongs to exactly one of these groups, which determines the users voting variance. Figure 9.1 shows each groups' voting variance (or error rate), which lies within the interval [+5, -5] inclusive.

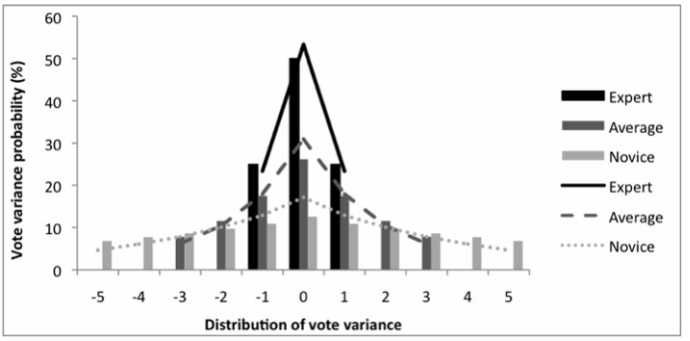

Figure 9.1 **The voting variance for each of the three simulated user groups, which lies between +5 and −5 grades away from the software's correct rating. The modelled voting variances are shown as bars, and the actual ones as lines.**

The expert users in Figure 9.1 rate software correctly 50% of the times, and in the remaining part rate the software either one step below or above its correct rating, i.e. the expert users always manage to rate a software within a 1 step wide window around its correct rating. The second group is the average users that tries to rate software correctly, but with lesser accuracy than the experts, i.e. they rate up to 3 steps above or below the correct rating due to lack of skills. Still an average user is better than a novice user that has an error margin of 5 steps above or below the correct rating. Figure 9.1 also shows, as lines, the actual outcome of the distribution during our simulation. The discrepancy of these values are due to problems of giving the worst rated grades for certain types of programs that already are close to one of the rating scale borders. In such cases a new vote variance is randomized based on the user's voting

distribution in Figure 9.1, hence the greater probability for a voting variance close to 0.

In addition to the users' own ability in rating software it is also possible for the simulator to simulate that they are influenced by previous user's ratings. This would for instance occur when a user is unsure what rating to assign a certain software and therefore use that software's current average rating as guidance when making up his/her mind. In our simulations we refer to this as the *previous rating influence* (PRI), which is represented as a number on the continuous scale [0-1] inclusive. We argue that the PRI effect increases as users become less confident about how to rate software. Experts are not very influenced by the already existing rating and thus have a relatively low PRI, in this case 6.25%. The group of average users is more likely to be influenced, thus earning them a PRI of 12.5%. The novice group on the other hand will most likely be very influenced by the already existing rating, and be more inclined to give a rating that is similar to the existing. To simulate this we give the novice group a PRI of 25%. As shown in Equation 1. below users' votes are generated by adding the user's vote variance to the software's correct rating, which results in that users from the different groups rate software differently.

$$(1 - PRI) + (CorrectRating + VoteVariance) + (PRI \times AverageRating) \quad (1)$$

When no PRI is used a user's vote is calculated by simply adding the software's correct rating with the user's randomized vote variance. However, when PRI is used the vote is instead pushed towards the software's average rating to various degrees, based on the amount of PRI that is simulated. An important aspect in the simulations is how to adjust the users' trust factors. The simulator allows its operator to tune four different variables that directly control how the TF is being calculated. First of all the operator has to decide if the TF should increase or decrease in an exponential or linear fashion. Secondly, decide what the change-rate should be, e.g. if a linear value of 2.0 is used then TF would increase or decrease by 2.0 based on whether the user manages to pinpoint the software's correct rating or not. If, on the other hand, an exponential value of 1.25 is used the user's TF will either increase or decrease with a factor 1.25 based on the current value. The third variable that is available to the operator is the potential to include a maximum level, or ceiling, which the TF cannot exceed. Finally it is also possible to decrease the TF faster than it increases by enabling the *decrease factor* (DF), i.e.

a DF of 1.5 will result in that a user's TF decreases with a factor 1.5 more than it increases. The DF could be used as a sanction method against misbehaving or cheating users. However, it has not been further investigated in the experiments presented in this paper.

9.4.3 Simulation Steps

During the initialization of the simulator each program is randomly assigned its "correct" rating which is used for evaluation purposes. Next, the users are assigned to the simulated groups according the proportions defined in the configuration files. Then the simulation starts and executes according to the following steps:

1. Shuffle the list of users
2. Sequentially select each user from the list
3. Randomly select a software
4. Randomize new vote variance for user
5. Create vote (Equation 1.)
6. Increase vote counter by 1
7. Update software's weighted average (Equation 2.)
8. Update user's trust factor
9. Repeat for each user in list
10. Repeat until specified number of votes are reached

The software's weighted average score is calculated based on both the user's vote and trust factor, as explained in Equation 2. This renders in that users' votes are being weighted differently based on their individual trust factors, i.e. amplifying the votes from trustworthy users.

$$AverageRating = \frac{\sum_{i=1}^{n} (vote_i \times TrustFactor_i)}{\sum_{i=1}^{n} TrustFactor_i} \quad (2)$$

After each vote the user's TF is updated based on how far ¤away from the software's current weighted average the vote is. If the vote is exactly the same as the weighted average the TF is increased, while it is kept as is if the vote is 1 step above or below. However, if the distance is further than 1 step the TF is decreased.

9.5 Simulated Scenarios

As seen in the background section, ranking of new and previously unfamiliar software is a common method for investigating the "true" appearance of a program before installing it. Using professional experts for doing this is both expensive and unrealistic due to the huge amount of non-investigated programs. Instead a pool of ordinary users with different skills is proposed where each participant repeatedly votes between 1 and 10 before new programs are installed. The voting is based on the user's skill, but also on the previous rating of the program. A skilled user may improve his own reputation by repeatedly giving votes close to the "true" value, i.e. similar to the professional expert. By doing so it is possible to increase the value of the vote either by a linear or an exponential increase.

In a recommendation system different actors may appear. We used the previous described groups of experts, average and novice users in the simulation of our reputation system. All users have one equal valued vote to start with and are supposed to repeatedly rate new software. All simulations in this work include a fixed population of different skilled users with 9.4% experts, 27.1% average users and 63.5% novice users. We decided to use these estimates based on the PEW Internet & American Life Project's statistics of user demography of information technology users [9].

Within this population all groups give a vote based on actual skills, and in some cases also based on the influences from previous voters, i.e. the weighted average for that particular software. Some of the simulations also measures what effects scaling up or down the proportion of different users have on the system, e.g. how system accuracy is affected when scaling down the number of expert users by half.

The above-mentioned groups were simulated for one million users voting for 100000 different programs, i.e. it is unlikely that one user will vote more than once for a single program. We chose to include 100000 programs based on the number of software application included in Web-based reputation systems, e.g. Softonic [12]. One million users are argued to be a realistic number due to the fact that such a system is globally accessible and therefore benefiting from network effects. The scenarios that we simulate in this paper include 48 and 96 million votes, which represent a two or four years

use of a system with one million users, and an average voting frequency of two votes/month.

9.6 Results

First we investigate how big an impact we may give a single voter without looking at the result other voters have given for the population of voters described above. Next, we look at the impact of previous rating influence where the different groups of voters are more or less influenced by the judgment already done. Then the proportions of experts, average and novice users are varied. Finally we investigate how system accuracy is when the correct rating for 25% of all simulated software programs are known before the simulation is started, i.e. that they are bootstrapped with the correct rating.

9.6.1 Trust Factors and Limits

Different trust factors varying in range from 1.05 to 2.0 were investigated with either linear or exponential increase. Each group of users starts with a TF set to 1.0 with an increase of the chosen trust value for each vote that is placed within one step above or below the software's current average rating. If the vote is up to one step away from the correct value the TF is left unchanged. Otherwise, i.e. two steps or more, the TF is decreased by the same trust factor. A user can never have a TF lower than 1.0. Each participant voted 48 times each and in all 1 million users voted for 100000 programs in this simulation. Figure 9.2 shows the outcome for the chosen number of different users and various maximum TF limits.

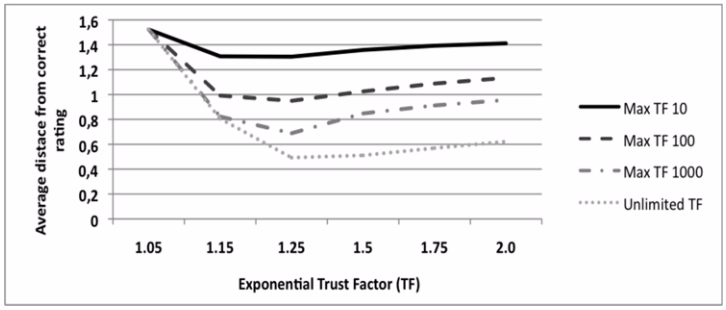

Figure 9.2 **9,4% expert, 27,1% average and 63,5% novice users voting with a varying trust factor during 48 votes each.**

The linear outcome for different TFs shows a very limited improvement with increasing TFs and thus is not further investigated. The lowest average distance was reached for an exponential TF or 1.25 where both lower and higher TF showed worse performance. For this reason we decided to use a 1.25 exponential TF during the rest of the simulations.

Next we investigated the need for a maximum limit of the user's TF. Figure 9.3 shows what happens when the following limits are specified for the TF; 1, 10, 100 and 1000 and unlimited.

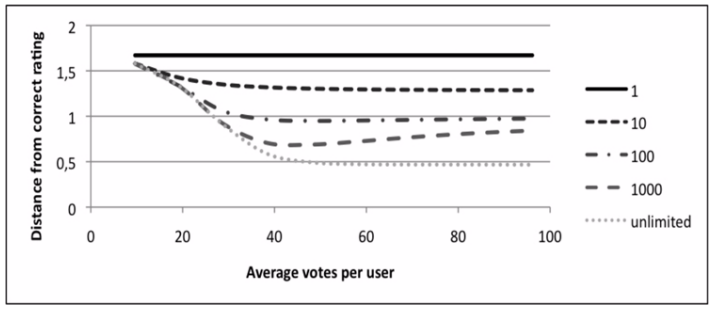

Figure 9.3 **9,4% expert, 27,1% average and 63,5% novice users voting with a 1,25 exponential trust factor during 96 votes each.**

An unlimited TF settles around 0.5 from the correct value but may give a single voter an un-proportional big impact on the voting system. Limiting the TF to 1000 is a reasonable compromise where the first 25 votes will behave as unlimited and settles around 0.7 with a slightly increase during prolonged voting time. This has to do with a relative TF increase of less skilled average users due to the expert reaching the 1000 TF limit as shown in the next figure. Figure 9.4 shows the outcome for each category on a logarithmic scale. The novice user hardly reaches above 1 where the average user has a small but constant increase. However, the TF of the expert users reaches 10, 100 and 1000 respectively when these limits are specified, and reaches 1 million after 48 votes when no limit at all is used. So, in all cases the expert voter has a dominant position, but not a 100% voting accuracy. As even expert users commit errors when voting they should not be given unrealistically high TF. In our setting an expert is predicting the correct value 50% of the time, i.e. the impact from an expert with huge TF giving a wrong value distort the result from the correct decision. The "knee" on the curve

associated with the experts in Figure 9.4 is due to the lack of measurements for TF limits between 10000 and unlimited.

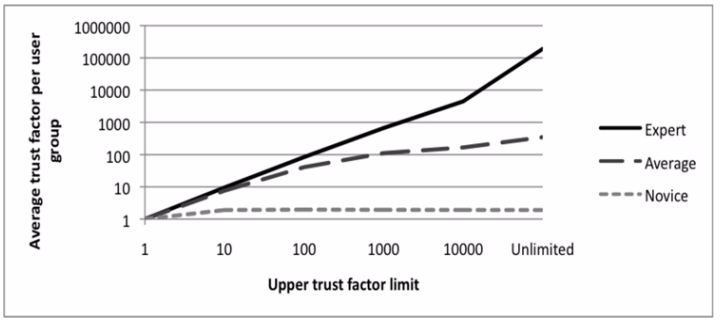

Figure 9.4 **Number of votes for each user group with different trust factor limits during 48 votes per user.**

9.6.2 Previous Rating Influence

Some commercial recommendation systems make previously given votes available to the users when they decide on their vote, i.e. the previous rating influence (PRI). This can give both positive and negative consequences that should be considered. The different user groups have different levels of knowledge, and thus are guided by the already existing rating to different degrees. To measure the effects that PRI have on a software reputation system we simulated a scenario where each user-group were influenced by 6.25%, 12.5% and 25% for expert, average and novice users respectively.

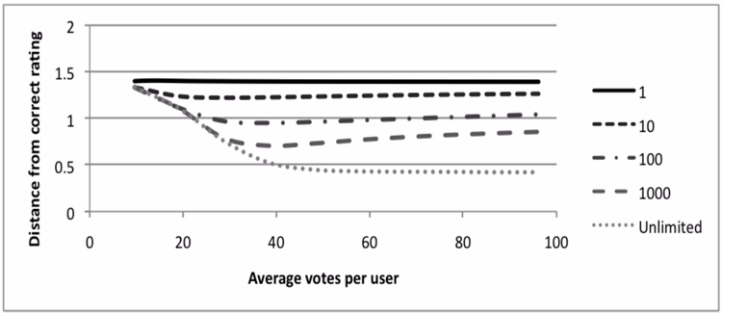

Figure 9.5 **Simulation with a previous rating influence (PRI) of 6.25%, 12.5%, 25% for experts, average and novice users. With an exponential trust factor (TF) of 1.25 and 96 votes per user, which corresponds to four years of system usage if each user votes on average twice a month.**

When for instance a novice user rate a software his vote is to 75% decided based on his level of knowledge and to 25% based on that program's current average rating. Figure 9.5 shows the results of this simulation.

As seen in Figure 9.5, there is a slight improvement in the beginning of the simulation when PRI is used. However, performance then stabilizes around the same levels as when no PRI is used. There is for instance no noticeable improvement in the case with a TF limit of 1000 compared to the results in Figure 9.3. When activating PRI in the simulation both novice and average users will improve their marking ability, which in turn result in that their TF increases. In Figure 9.6 the trust factor of both the novice and average users are plotted on a logarithmic scale, which show that both has increased several times compared to the results in Figure 9.4.

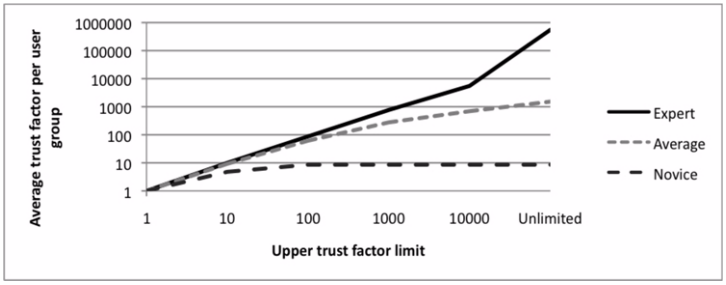

Figure 9.6 **Development of the trust factor (TF) for each user group when previous rating influence (PRI) is enabled. These results are based on an exponential trust factor of 1.25 without any TF limit.**

The results in Figure 9.6 show that the TF of all user groups has increased when PRI is enabled. This can be attributed to the fact that for instance the novice group has a higher tendency to follow the already set ratings, i.e. the results from the average and expert users. Due to this, they will rate an application more accurately, which will increase their trust in the system. As already stated, this results in a higher TF for the novice group. Based on the size of the novice group they will have a higher impact on the system. Even though the novice group has a higher tendency to follow the average ratings, they will still introduce votes that are distanced from the correct rating of software, which deteriorates the overall system accuracy. We have also simulated scenarios with higher PRI values for all user groups, but without any significant improvement of the system accuracy. These results therefore show that the system accu-

racy is not significantly improved when PRI is enabled and that the reputation system is more stable without it.

9.6.3 Demography Variations and Bootstrapping

Figure 9.7 shows how the user demography of the simulated users affects the system accuracy. We varied the size of the different user groups around the survey results presented by PEW Internet & American Life Project. We can see that the size of the expert and average groups clearly make a difference in the beginning of the simulation, i.e. higher number of expert and average users increases system accuracy. However, as the simulation progresses and the TF of each simulated user is fine-tuned the system accuracy increases for all user constellations. An increased number of experts perform better than increasing the number of average users in relation to novice users.

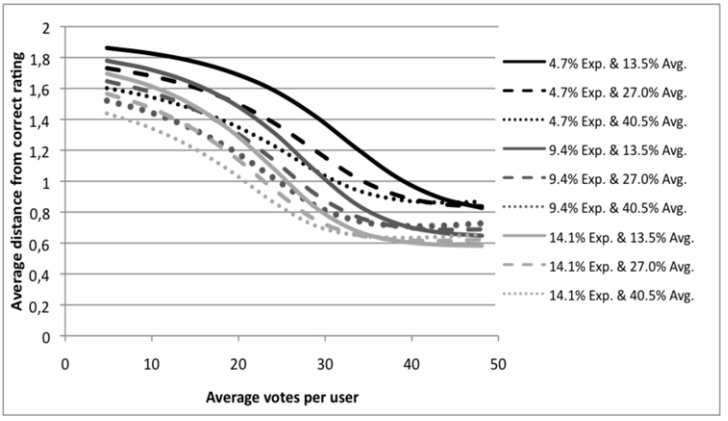

Figure 9.7 **System accuracy for several different user group constellations with varying number of expert and average users. In this experiment the previous rating influence was turned off and an exponential factor of 1.25 was used together with a trust factor limit of 1000.**

To improve the system accuracy further we also simulated a scenario where already available software reputations were collected from third parties and used to initialize the software reputation database before it was put into use. Figure 9.8 shows how the accuracy of the system is affected when 25% of the software inside the reputation database is bootstrapped with trusted data with a total value of 5000, i.e. equal to five votes from full-fledged experts. When compared with Figure 9.7 it is clear that the system accuracy is positively affected in the beginning of the simulation, but as the

simulation continues the difference between whether bootstrapping is used or not decreases.

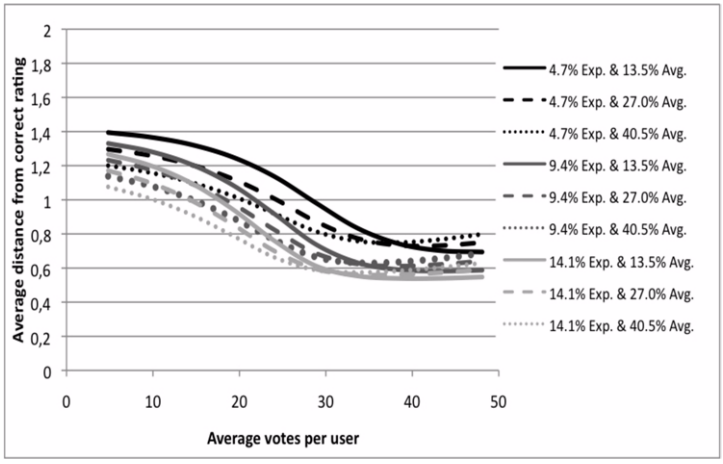

Figure 9.8 **System accuracy per user group constellation with 25% of the reputation data bootstrapped from trusted sources.**

We also ran a set of experiments without any expert users at all just to investigate whether or not the software reputation system would still function properly in such a scenario. When omitting all expert users, leaving 27% average and 73% novice users, the system shows an accuracy score of approximately 1.8. in the beginning of the simulation which then improves towards 1.2. Finally we also simulated the use of a software reputation system during an extended period of time, in this case 1200 votes per user, without seeing any tendency of degraded system accuracy. Even though the scenario is questionable, since none of the software programs are updated, it doesn't show any evidence of decreased performance, i.e. the reputation system seams to be stable. In fact the accuracy is continuously improving through out the simulation during this extended simulation, and the overall system accuracy stops at 0.95. During the whole simulation the TF of the novice users are quite low, with an average of 2.8. At the same time the TF of the average users stabilize around 890 while the experts quickly reach the TF limit of 1000.s the difference between whether bootstrapping is used or not decreases.

9.7 Discussion

The idea behind the simulated reputation system is to reward users that provide accurate ratings and punish faulty or not properly thought through decisions, which will improve the software reputations within the system. The experiments assume that there exists a "correct" rating of each software for initial settings and evaluation purposes. However, whether such a correct rating exists in a real world setting is not necessarily true. It is therefore hard for a single person to define exactly what the correct rating for the particular software should be, but based on several users' ratings it is possible to come to a common compromise, which then is used as a baseline when deciding whether or not to increase or decrease users' TFs.

In this work we have used simulation as the means to identify how the users' TF should be adjusted within a software reputation system to reach an accurate, stable and sustainable system to mimic the view of a professional expert. In this investigation we primarily look at the behaviour of the voter. From a single program's point of view different skilled voters may end up voting close to the "correct value", i.e. on average votes below and above this value are compensated. So, the average value for a single program is better than the user's performed absolute distance value. By introducing a trust factor and/or previous rating influence the overall performance for various groups of users was increased, i.e. the absolute distance value coming closer to the average value. This was true even in groups dominated by unskilled novice and average users making the outcome of the voting procedure more robust.

Unlike professional experts the simulated population consists of different skilled users voting twice a month during two or four years. In the beginning of the simulations each voter has given a very limited amount of votes, i.e. only a small fraction of all available programs are being rated by a single voter. The simulation results show an exponential increase of the TF to be better than a linear increasing, and that a factor of 1.25 was most promising for an accurate system. Through the simulations we also found that an upper TF limitation of 1000 is preferable. If this limitation is being omitted the TF of the expert users quickly increases to levels that make the whole system unstable because of these users' small, but still existing, rating fluctuations. Therefore the upper TF limit creates a balance between fast reaching and well-informed ratings, without giving a single user too much influence. In a real situation

other factors, such as malicious behaviour, makes this argument even more important within the system. If a single user can get extremely high TF values it is possible for malicious actors to use this to manipulate the rating of the particular software for their own personal gain.

Intuitively it might seem interesting to allow users to see all previous ratings (PRI) when they make up their mind about how to rate a certain software, since this at first could be thought to improve the decisions of the less skilled users. However, if the system makes use of TFs this is not the case since novice and average users will improve their TFs, i.e. their normal voting variance will exceedingly negative influence the program evaluation. Through the simulations we also investigated how the proportion of experts, average and novice users affects the system accuracy and stability. By increasing and decreasing the number of experts and average users we were able to draw the conclusions that the system accuracy improves as expert users sign up to the system. For all investigated populations the system accuracy improves from initial not-to-well judgments in the beginning of the simulation to closer to the correct ratings, for instance when 85% novice users and less than 5% experts are being simulated.

In our simulations we have also showed that it is possible to improve the initial system accuracy before the system is made publicly available by bootstrapping the database with trusted reputation data. Such bootstrapping data could for instance be gathered from web-based services available on the Internet. Furthermore we also show that the system is stable and accurate when users submit ratings with a higher frequency. In this case when each user submits on average 1200 votes over a four years period, i.e. about one vote per day.

When summarizing all simulated scenarios we come to the conclusion that it is possible for computer users to rely on a stable software reputation system that assist them when identifying well-reputed software, as well as when avoiding questionable programs.

9.8 Conclusions

A software reputation system consisting of expert, average and novice users was simulated as an alternative to let anti-malware pro-

grams or dedicated human experts decide about questionable programs. Within the simulated population, the different skilled users voted twice a month during at most four years. Each voter starts with a single vote and by varying the increase of the trust factor (TF), the upper TF limit and previous rating influence (PRI), with a resulting balanced, well-informed rating of judged programs as the proposed outcome.

An exponentially increased TF was better than a linear, and a system that allowed voters to see previous users' votes performed better, i.e. with PRI. More precisely an exponential increase of 1.25 for the TF with a TF limit of 1000 within an experimental setting of less than 5% experts in a population exceeding 80% novices still performed well. Such a setup allowed a balance between quickly reaching a well-informed decision and not giving a single voter too much power.

In our opinion the reputation systems will become a more commonplace advisory tool in the future with the possibility to provide an advice to the user that most likely will be helpful, i.e. being able to handle erroneous, good and bad ratings, and without losing the integrity of the system.

9.9 Future Work

Our simulated environment lacks some real world parameters that will be further investigated as the work progresses. First, we will simulate various attack schemes that are used by malicious actors to gain control over the reputation system. Secondly, we will investigate how system accuracy is affected when users are handled more dynamically, e.g. when new users join up during the simulation and that established users' leaves. Finally, this should also include scenarios where new and unknown programs are being added on the fly or when old programs are being overridden.

9.10 Acknowledgements

We would like to thank Martin Hylerstedt at Blekinge Institute of Technology in Sweden for his assistance in proof-reading this work.

9.11 References

[1] G. Adomavicius and E. Tuzhilin, Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions. Knowledge and Data Engineering, IEEE Transactions on Knowledge and Data Engineering (2005) vol. 17 (6) pp. 734-749.

[2] Amazon, http://www.amazon.co.uk/, 2010-03-11.

[3] M. Boldt, B. Carlsson, T. Larsson and N. Lindén, Preventing Privacy-Invasive Software Using Collaborative Reputation Systems, Vol. 4721 of Lecture Notes in Computer Science, 2007.

[4] J. Douceur, *The sybil attack*, In the Proceedings of the 1st International Workshop on Peer-to-Peer Systems (IPTPS), 2002.

[5] eBay, http://www.ebay.com, 2010-03-11.

[6] Internet Movie Database, http://www.imdb.com, 2010-03-11.

[7] A. Jøsang, R. Ismail and C. Boyd, A Survey of Trust and Reputation Systems for Online Service Provision, *Decision Support Systems*, Vol. 43(2), pp. 618-644, 2007.

[8] A. Jøsang, S. Hird, and E. Faccer, *Simulating the Effect of Reputation Systems on E-Markets*, Volume 2692 of Lecture Notes in Computer Science, 2003.

[9] PEW Internet & American Life Project: A Typology of Information andCommunicationTechnologyUsers, http://www.pewinternet.org/Reports/2007/A-Typology-of-Information-and-Communication-Technology-Users.aspx, 2010-03-11.

[10] P. Resnick et al., The value of reputation on eBay: A controlled experiment, *Springer Journal on Experimental Economics*, Vol. 9(2), pp. 79-101, 2006.

[11] S. Ruohomaa, L. Kutvonen and E. Koutrouli, *Reputation management survey*, In the Proceedings of the 2nd Second International Conference on Availability, Reliability and Security (ARES), 2007.

[12] Softonic, http://softonic.com, 2010-03-11.

[13] J. Traupman and R. Wilensky, Robust Reputations for Peer-to-Peer Marketplaces, Vol. 3986 of Lecture Notes in Computer Science, 2006.

[14] A. Whitby, A. Jøsang and J. Indulska, *Filtering out Unfair Ratings in Bayesian Reputation Systems*, In the Proceedings of the 3rd Interna-

tional Conference on Autonomous Agents and Multiagent Systems AAMAS, 2004.

[15] H. Yu et al., Sybilguard: Defending against Sybil Attacks via Social Networks. ACM Proceedings of the 12th SIGCOMM, 2006.

PUBLICATION

7

Stopping Privacy-Invasive Software Using Reputation and Data Mining

Journal manuscript, submitted 2010

Martin Boldt and Bengt Carlsson

Privacy-invasive software (PIS) is described as a category of software that ignores users' right to be left alone and that is distributed with a specific intent, often of a commercial nature that negatively affect its users, i.e. most malware, spyware and legitimate commercial driven applications. From a single user's perspective all software are acceptable or deniable, i.e. white or black listed, if s/he has an informed consent, i.e. all greyzone programs can be eliminated. PIS is classified with respect to user's informed consent and amount of user's negative consequences with the purpose of adding mechanisms that safeguard users' consent during software installation. Two techniques based on a software reputation system and automated End User License Agreements (EULAs) classification is suggested for counteracting PIS. Both the reputation system and the automated EULA classification increase user awareness about the software behaviour, which allow users to make more informed decisions concerning software installation, i.e. reducing but not eliminating the threat from grey-zone programs. Results from the automated EULA classification is presented together with a prototype implementation and a usability study of a software reputation

system. The main conclusion is that both automatic EULA classification and a software reputation system can be combined into a decision-support system for computer users.

10.1 Introduction

A powerful component in any business strategy is to gather information about present or future customers. The company with the most information about its customers and potential customers is usually the most successful one [25]. This situation has given raise to a parasitic market, where questionable actors focus on short time benefits when stealing personal information for faster financial gain by distributing spyware [7, 10, 20]. However, this group of software differs from traditionally malicious software (malware) in that it is not necessarily illegal, and that they can be linked to a certain company. To further complicate matters, the borders on both sides of this grey-zone are vague and context dependent, and there is no clear definition that accurately captures this problem. Spyware definitions either focus solely on stealing information, and therefore miss additional functionality [20], or are too broad so that they include unrelated software as well [3]. Due to this some have simply abandoned the term spyware all together and replaced it with another term, such as badware [43].

Without a clear definition of a threat it is hard to accurately address it by implementing countermeasures. As a result users need to rely on several countermeasure tools in parallel since none of them stand on their own. In an attempt to mitigate this situation we introduce the novel concept of *privacy-invasive software* (PIS), which could be described as a category of software that ignores users' right to be left alone and that is distributed with a specific intent, often of a commercial nature that negatively affect its users [7]. This categorization includes both legitimate and malicious software, as well as the grey-zone in between. The goal with this classification is to transform software in the intermediate grey-zone into either legitimate or malicious software for each individual user. We intend to do this by providing users with condensed information about a software program's behaviour using both automated EULA analysis and a software reputation system. Based on this information, the users can decide on whether or not they want to allow installation of certain software on their computer.

Today, the most commonly used method for users to give their consent to software installations is by accepting the terms stated in the End User License Agreement (EULA), which is shown during software installation. However, users are not capable of comprehending the content of these EULAs, since they are disclosed in a very legal, formal, and lengthy manner [29]. Most users simply accept the license agreements without reading through them first, and with the risk of unknowingly allow illegitimate software to enter [27]. The underlying problem is that software vendors don't have any standardized and usable method for disclosing information to their customers about their product's implications on the user's computer system. So, what we want for any computer-user is to know what they install and, with the help of aiding mechanisms, learn to distinguish between what they believe is acceptable and unacceptable software, *prior* to any actual software is installed on their system.

The outline of this paper is as follows. First we start by shortly describing traditional countermeasures in Section 10.2, and defining privacy-invasive software in Section 10.3. Then we present the idea of using automatic EULA classification in Section 10.4, and software reputation systems to address the problem with PIS in Section 10.5. Next, in Section 10.6 we present the results from a number of simulated scenarios of a software reputation system. In Section 10.7 we discuss our results and in Section 10.8 we make our conclusions and state our future work.

10.2 Traditional Countermeasures

Traditionally the problem with harmful software came from a rather small group of software with the primary goal to destroy computer systems, often in the shape of computer viruses [40]. As a counter reaction companies that created anti-virus tools to protect computer-users from such threats emerged. Fortunate enough for these anti-virus vendors their targets were clearly illegal and therefore separated from legitimate software in terms of legal aspects. So removing malware was a question of technical acrobatics using the resources and techniques available at the time [45]. However, this situation changed during the late 1990s as a new group of software, often referred to as spyware, emerged. One important reason for this was that Internet began to reach the big masses, which resulted in a new market for unscrupulous companies that began stealing personal information for distributing targeted online advertise-

ments. The consequence was that a grey-zone of software appeared in between legitimate software and malware.

A complicating factor compared to the black and white situation that existed before was that this grey-zone overlapped with the group of legitimate software. In fact, what one user regards as a useful product others could see as a spyware program. Due to this subjective nature a static division between spyware and legitimate software was not possible [7, 30], which resulted in problems for the anti-virus companies. Their most commonly used technique for combating malware was blacklisting through signatures, i.e. statically dividing between legitimate and malicious software. This requires a copy of the malware to first be captured in the wild so a unique signature can be created, and then being distributed to all customers [45]. An obvious drawback with this approach was the fact that the anti-virus tools were irreparably one step behind the creators of malware, since they first had to find a copy of a malware in the wild before any signature could be created. Another drawback was related to the vast amount of malware that spread on the Internet, which made the signature databases grew rapidly resulting in impractical scan-times on the customers' side.

Anti-virus companies therefore began creating alternative techniques for solving the problem. They for instance used agent-based approaches [33, 37], and techniques from the area of artificial intelligence, e.g. artificial neural networks [2, 30]. Another technique was heuristic analysis that kept a suspicious program inside a sandbox, such as a virtual machine, while monitoring its execution as a way to discover any deviant behaviour [27, 48]. Even though this could be done for computer viruses, e.g. by detecting the self-replication routines, it was much harder to distinguish spyware or adware programs from their legitimate counterparts. In the end, adware and spyware programs simply show information on the screen or transmit rather small quantities of data over the network interface, i.e. behaviour that is shared by most legitimate programs as well.

As the amount of malware continued to increase on the Internet it was proposed that white-listing could be a potential solution, since the set of malicious software began to reach similar size as the set of legitimate software. The basic principle is to pinpointing all legitimate software so that everything else could be treated as malicious [31]. Obviously, the problem here was exactly how to pinpoint all legitimate software, and at the same time have safeguards that prevent any malware or spyware program from sneaking in. One

approach is for a trustworthy organization to scrutinize software programs according to a list of predefined requirements, and then add all software that fulfilled these requirements to a list of approved software, e.g. TRUSTe's Trusted Download Program [47]. However, the problem is that such scrutinizing is highly time-consuming and therefore costly, and as a result software vendors have to pay for having their software analyzed and approved, which is something that spyware vendors for obvious reasons are reluctant to do. Another problem is the huge amount of new software that is being released, which is more than a single group of experts can manage on their own. It simply does not scale very well.

The bottom line here is that the above-mentioned techniques try to statically divide software as either legitimate or not. However, for the programs in the grey-zone this is impossible due to its subjective nature, i.e. any anti-virus company that removes or labels these programs as spyware (or any of its synonyms) will still have unsatisfied customers (since some disagree) and they also risk legal retributions from the creators [41]. Especially since many of these spyware companies make significant revenues, which they have no plan on abandoning. This problem, in combination with the increasing amount of malware on the Internet, renders techniques that focus on static division unrealistic.

We therefore present automated EULA classification and a collaborative software reputation system as a potential way forward. These techniques do not try to make any trust-decisions for the users, but instead create mechanisms that support the users when making such decisions [23]. In the end it is up to the users to make the final decision.

10.3 Privacy-Invasive Software

Our classification of privacy-invasive software (PIS) is a further development of the work by Warkentin et al. [48]. The classification consists of a matrix based on the combination of *negative consequences* and the level of *informed consent* from the user. Our use of the term informed consent is based on the work by Friedman et al. [16], and in our classification the consent level is represented as a continua spanning between full, medium and none. The negative consequences are also represented as a continua spanning between tolerable, moderate and severe. This results in a three-by-three matrix as

shown in Table 10.1, which represent the amount of negative consequences on one axis and the amount of user consent on the other. The shades of grey in Table 10.1 signify the difference in software behaviour where darker tones represent increased levels of unwanted behaviour.

Table 10.1 **Classification of privacy-invasive software with respect to user's informed consent (full, medium and none) and amount of user's negative consequences (tolerable, moderate and severe).**

	Tolerable	Moderate	Severe
Full	Legitimate software	Adverse software	Double agents
Medium	Semi-transparent software	Unsolicited software	Trojan horses
None	Covert software	Semi-parasites	Parasites

Our classification of PIS in Table 10.1 presents three different groups of software; the first has high user consent to provide its service (top row); the second group include software that has some kind of user consent but it does not correspond to the software's full behaviour (middle row); the last group include software that does not have any consent from the user at all (bottom row). By inspecting these three groups as software with PIS behaviour, we can narrow down what types of software that should be regarded as spyware, and which ones that should not.

The top row includes software that has received full permission from the user, and where the behaviour is fully transparent towards the user, i.e. they should be regarded as legitimate. Adverse software does not include any covert behaviour, and the negative effects of the double agents are fully transparent to the user, but the positive effects of the software outweighs the negative effects so the users chose to install it anyway [23].

The middle row includes software that has gained some sort of consent from the user, but it was not based on an informed decision and should therefore be regarded as spyware. Requested semi-transparent software includes components that the user is unaware of, which then could introduce vulnerabilities to the user's system. Unsolicited software is requested software that covertly installs further software components, i.e. users are unaware of the existence and behaviour of these bundled components. Trojan horses include

covert functionality that causes major negative consequences for the user, e.g. installation of a backdoor.

The bottom row includes any software that installs and executes without any user consent at all, and should therefore be regarded as malware. Covert software is software that secretly installs themselves on systems without causing any direct negative consequences, but they might still include for instance vulnerabilities. Semi-parasites are pushed on users when, for instance visiting Web pages, e.g. by deceiving users into thinking these components are needed to access the Web page. Parasites are a group of software with severe negative consequences that gains entrance to the user's system without any consent at all from the user.

In our view spyware is located in the middle row in Table 10.1, i.e. the grey-zone between legitimate software and malware. However, this does not automatically define spyware as a lesser problem than malware programs. Some of the most common problems associated with spyware include [1, 10, 18, 23, 34]:

- Displaying of unsolicited advertising content at varying frequency and substance
- Personally identifiable information being covertly transmitted to third parties
- Poor or non existing removal mechanisms
- Resources invested in recovering systems from such unsolicited programs
- Settings and properties of other software being modified
- Third party software installation without user consent
- Unauthorized resource-utilization causing deteriorate system stability

The main difference between spyware and malware is that spyware present users with some kind of choice during the entrance into their system [17]. This means that any software that installs itself without asking for the user's permission should no longer be treated as spyware, but instead as malware. The difference between legitimate software and spyware is based on the degree of user consent associated with certain software, i.e. informed consent means legitimate software and otherwise spyware. Hence, the problem really boils down to the fact that computer users only have inaccurate mechanisms to evaluate software's degree of appropriateness to enter their system.

In the following sections we present a number of mechanisms that could aid users in their evaluation of software before allowing the actual installation to proceed.

10.4 Automated EULA Classification

Spyware distributors usually disguise their software as legitimate to reach as many users as possible. However, to avoid legal repercussions they often mention in the End User License Agreement (EULA) that spyware will indeed be installed. However, this information is given in a way most users find hard to understand. Even legitimate EULAs can be hard to fully comprehend due to their length and the juridical terminology used. There exist a number of tools that try to support the users in separating EULAs as either legitimate or not. The state-of-the-art tools concerning EULA analysis rely on blacklists that are used for highlighting interesting sentences in EULAs. Each user then has to read these sentences and make up their mind about whether or not the program is legitimate or not, i.e. the actual classification is left to the individual user.

We therefore propose the use of a fully automated EULA analyzer should make use of text classification techniques and supervised algorithms used in machine learning and data-mining. By comparing a set of general accepted legal EULAs with another set of questionable EULAs, such text classifiers could find general patterns outside the scope of black-listed words. The resulting text classifier could then automatically classify previously unseen EULAs as either "good" or "bad", without having to depend on the user's own classification skill as is the case with today's state-of-the-art tools. In an experiment we evaluated 17 learning algorithms' ability to classify the 996 EULAs in our data set as either good or bad [29]. The EULA set that we collected included 900 legitimate and 96 spyware EULAs.

Within the data mining area the most widely used metric has traditionally been *accuracy* (ACC), which is defined as the number of correct classifications divided by the total number of classifications. However, ACC gives erroneous results when used on unbalanced data sets, i.e. classifying all 996 EULAs in our data set as legitimate would render in an accuracy of just above 90%. As a result we therefore need more suitable metrics. Two such examples are the *true positive rate* (TPR) which also is called the hit rate, and the *false*

positive rate (FPR) which is the miss rate. Both TPR and FPR are combined within the *ROC curve*, which plot the TPR on the Y-axis and the FPR on the X-axis, i.e. a graph that specifies the amount of correct hits versus false alarms [19]. In many experiments the integral of the ROC curve, also called the *Area under the ROC Curve* (AUC), is used as the main metric since it is independent of both class distribution and misclassification costs [50]. When evaluating the algorithms, we relied on 10 times 10-fold cross validation for dividing the data set into 10 pairs of training and testing data. The algorithms were trained using the 10 training data sets respectively, while each time being tested using the data in the remaining fold [50]. This whole process was then repeated 10 times with different selection of instances between training and testing data in each iteration.

Finally it is also important to point out that the cost of misclassification is different for the two classes (good and bad), since classifying a bad EULA as good is worse than classifying a good EULA as bad. In a decision support system the first case would mean that the user installs software with questionable behaviour, while in the second case the user would think twice before installing the legitimate software, and maybe even try to find an alternative to that particular software.

10.4.1 Results

Table 10.2 includes the results from the experiment together with the different metrics. These results reveal that Naive Bayes Multinomial was the best performing algorithm with an ACC of 0.939 and an AUC of 0.926. The FPR translated into absolute numbers mean that the Naive Bayes Multinomial algorithm classified 17 out of 96 bad EULAs as good, while the TPR mean that 856 out of the 900 good EULAs were really classified as good. These results could be compared with the worst behaving algorithm for this particular task, i.e. KStar with a TPR of 1.00 (no good EULAs classified as bad), and a FPR of 0.926 which means that 89 of the 96 bad EULAs were classified as good. Even though the results in Table 10.2 are limited due to the high levels of FPR most of the evaluated algorithms still perform significantly better than the state-of-the-art tool available on the Internet. In fact 10 out of the 17 investigated algorithms in Table 10.2 showed significantly better results than the state-of-the-art tool based on a two-tailed t-test with a significance interval of 0.05.

Table 10.2 Results from experiment evaluating 17 [32] learning algorithms ability to classify EULAs as either belonging to legitimate or spyware-infested programs.

Algorithm	ACC(Std.dev)	TPR(Std.dev)	FPR(Std.dev)	AUC(Std.dev)	t-test
NaiveBayes	0.874(0.035)	0.881(0.037)	0.193(0.136)	0.873(0.065)	4.204**
NaiveBayesMultinomia	0.939(0.023)	0.951(0.023)	0.173(0.122)	0.926(0.059)	5.040**
RBFNetwork	0.928(0.025)	0.950(0.024)	0.276(0.132)	0.798(0.100)	2.708*
SMO	0.950(0.019)	0.980(0.014)	0.335(0.169)	0.822(0.084)	3.192**
VotedPerceptron	0.949(0.021)	0.979(0.016)	0.334(0.146)	0.827(0.076)	3.408**
IBk	0.923(0.014)	0.997(0.006)	0.768(0.126)	0.847(0.078)	3.570**
KStar	0.911(0.009)	1.000(0.001)	0.927(0.080)	0.576(0.043)	1.053
AdaBoostM1	0.942(0.019)	0.975(0.016)	0.369(0.145)	0.875(0.076)	3.909**
Bagging	0.942(0.022)	0.980(0.016)	0.416(0.151)	0.887(0.069)	4.233**
Stacking	0.948(0.021)	0.978(0.015)	0.334(0.168)	0.822(0.084)	3.192**
HyperPipes	0.947(0.019)	0.989(0.011)	0.446(0.167)	0.827(0.094)	3.065*
JRip	0.932(0.027)	0.964(0.024)	0.368(0.163)	0.799(0.083)	2.982*
PART	0.928(0.023)	0.963(0.021)	0.401(0.154)	0.800(0.100)	2.726*
Ridor	0.931(0.021)	0.981(0.018)	0.537(0.190)	0.722(0.093)	2.092*
DecisionStump	0.941(0.021)	0.971(0.017)	0.343(0.136)	0.814(0.069)	3.435**
J48	0.928(0.021)	0.968(0.015)	0.443(0.169)	0.729(0.127)	1.846
RandomForest	0.941(0.017)	0.992(0.009)	0.542(0.149)	0.881(0.068)	4.198**
Average	0.971(0.017)	0.424(0.147)	0.813(0.080)		
ZeroR	0.904(0.004)	1.000(0.000)	1.000(0.000)	0.500(0.000)	

*p<0.05, two-tailed
**p<0.05, two-tailed, Bonferroni corrected

10.5 Software Reputation System

Reputation systems include an algorithm that allows members of a community, such as eBay.com, to estimate other members' behaviour before committing in any interaction [29]. A collaborative reputation system presents an interesting method to address PIS by collecting the experience from previous users' knowledge regarding software. While techniques such as third party certification or software deeds aim at increasing user awareness, reputation systems instead collect and refine user experiences. Such experiences are then used in a collaborate manner to inform users about the general opinion that exist for a specific software prior to the installation. The fundamental idea is that users make more accurate trust decision when incorporating such information.

Users' experiences are collected in the form of ratings and comments for the software that they frequently use, but the system could also allow the users to include additional information in their software reviews, e.g. whether they decide to install the software or not which then can be compiled into statistics that is presented to future users. All collected information from the users are then processed and transformed by the reputation system into software repu-

tations. These reputations are shown to subsequent users that wish to install software; providing them with a collective view of the software, e.g. what previous users' has thought about the software and whether or not they decided to install it.

There are a number of aspects that we took into consideration when designing our prototype that was to gather, store and present information from multiple, unknown users. Although the system was set up for a clear purpose; individual users, or groups of users, may find it more interesting to – for instance – intentionally enter misleading information to discredit a software vendor they dislike; use multiple computers in a distributed attack against the system to fill the database with bogus votes; enter irrelevant or indecent comments; and so on. The problem with poor input to the system is not only posed by users with a malicious intent, but also come from users voting and leaving feedback on programs they know nothing or little about. Even though this is a rather big problem for a software reputation system we mitigate the problem by incorporating individual trust factors that express each user's trustworthiness within the reputation system.

Furthermore we incorporate a meta-reputation system that allow users to rate the feedback of other users in terms of helpfulness, trustworthiness and correctness, which then is used for creating a reliability profile for each user. This profile could be thought of as a *trust factor* (TF) that is used to weight the ratings of different users, making the votes and comments of well-known, reliable users more influential than those of new users, i.e. amplifying the input from experienced users. Another important parameter within the reputation system is the *TF-limit*, which defines the maximum TF that can be reached. High TF-limits allow one expert user to compensate for a larger number of normal users, but on the other hand this gives single users considerable power within the system leading to instability and a threat from malicious actors. This is further investigated in the next section about simulation of a software reputation system.

A very important consideration for a software reputation system is the concept of the system users' privacy [45]. Since such a reputation system utilize sensitive information, e.g. what software users install, makes it crucial that privacy concerns are seriously addressed. Therefore the protection of peoples' privacy has been highly prioritized during both system design and implementation so that the system itself does not intrude on users' privacy more than

absolutely necessary. Unfortunately there must be some degree of privacy-invasion since the system needs to make sure that no user votes more than once for a particular software. So we cannot get rid of the concept of users and user accounts. However, the reputation system can be designed to ensure that all kinds of interesting information for an attacker (e.g. IP addresses, e-mail addresses, names, addresses, etc.) are excluded from the user information stored in the database on the reputation system servers. The only thing necessary to store is some kind of unique identifier such as an e-mail address.

Due to privacy concerns we don't want to store the users' e-mail addresses in the central database since it creates a link back to the individual user. We therefore only use the e-mail address for verification purposes when a new account has been created, and from thereon the system instead keeps a hash value while the e-mail address itself is removed. The hash digests are then used for prohibiting the same e-mail address being used for more than a single user account. Concerned users' could also safeguard their anonymity by utilizing distributed anonymity services, such as Tor, for all communication between the client and the server [36]. This would further increase user's privacy by hiding their IP address from the reputation system owner.

10.5.1 Prototype Implementation

Our prototype reputation system was implemented using a client server architecture where the client is responsible for gathering user input and for detecting new software on the users' computer, while the server handles reputation calculations and distributing them to the clients. The client software integrates tightly with the Windows XP operating system by hooking the API call that is invoked when the Windows kernel tries to allocate memory for a starting program. This way the client software starts every time the user executes a software program, even if it is previously unknown by the client. When this happens the client generates that particular pro-

grams identifier (e.g. a SHA-512 hash digest) and then requests the reputation for that particular identifier.

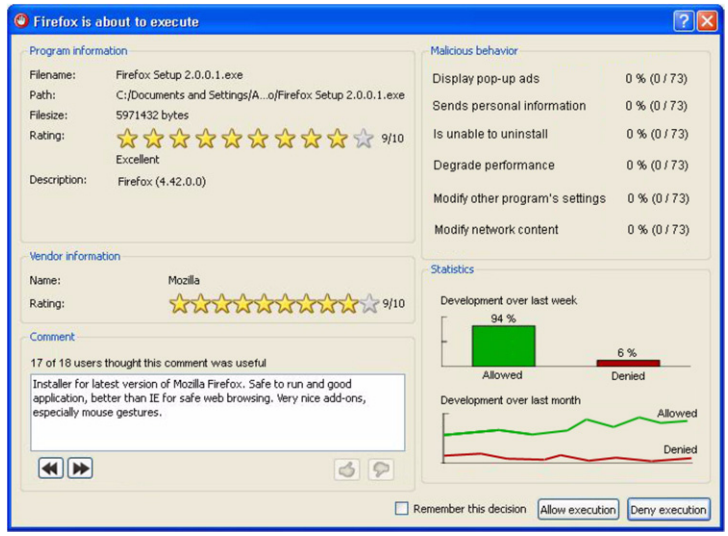

Figure 10.1 **The Graphical User Interface from our prototype that presents the software reputations to users.**

When the server receives a reputation request it looks up that particular software in the database and returns a reputation summary. The client then displays the reputation information using the GUI as shown in Figure 10.1. The reputation includes a rating for that particular software program together with a collective rating for the software vendor, which is based on all software programs being developed by that particular vendor. The GUI also includes text comments as well as statistics of suggested malicious behaviour that have been shared by previous users, e.g. if the program display pop-up ads or degrades performance. There are also two diagrams showing how many of the previous users that either allowed or denied the program to execute, one diagram for the previous week and another with statistics over the previous month. There is also a possibility to add software to either a black- or a white-list to avoid interrupting the user with further questions, i.e. it will be automatically blocked or accepted in the future.

Users' input is gathered using a similar GUI that allows them to rate the software on the discrete scale (1-10 inclusive). This rating is mandatory, but there are also optional fields that allow the users to write comments and answer questions about predefined types of malicious software behaviour. Hence, the quickest rating of pro-

gram requires the user to just decide on a rating and then submit, which shouldn't take more than a couple of seconds of the users time. For users that want to submit extended information about a particular program there are such optional possibilities as well.

10.5.2 Usability Study

In an experiment we allowed 51 users to install five different software programs both with and without the assistance of our software reputation system. The goal of the experiment was to measure to what degree a reputation system impact computer users' decisions during software installation. Each user therefore first installed all five software with the reputation system disabled, i.e. they did not receive any feedback on the software being installed. For each software that was installed, the subjects were asked a number of survey questions. When all five software programs were installed the software reputation system was enabled and the users were asked to install the five software programs again, but this time they should take each software's reputation into account, as presented in Figure 10.1.

The experiment was designed in a "one factor and two treatments" fashion [35], where the factor was users' decisions concerning software installation, and the treatments were how and to what extent these decisions were affected by either having access or not having access to the reputation of these programs. When the reputation system was disabled the users had to decide whether or not to install the software based on their gut feeling and possible clues that might be available in the programs' installation routines, but when the reputation system was enabled they could instead base their decision on this information.

Two of the five software programs were legitimate, while the other three were bundled with spyware. Also, one of the spyware-bundled and one of the legitimate programs were well known, while the remaining three were not. It was therefore expected that some of the subjects should have a clear opinion about these two programs based on first hand experience, but not about the remaining three.

The experiment was conducted in a closed environment in the Security Lab at Blekinge Institute of Technology and all subjects worked individually during the experiment. We chose subjects with firm understanding in both computer security and spyware by using

a convenience sampling strategy. Out of the 51 subjects, 43 were selected from the Bachelor program in Security Engineering. The remaining 8 subjects were selected from various Masters programs within Computer Science and Software Engineering. Since we deliberately selected subjects with firm computer security understanding we argue that average computer users would show even worse results in detecting spyware programs without the assistance from any aiding mechanisms, i.e. average computer users would benefit from a software reputation system to greater extent than the subjects in our experiment.

As briefly mentioned the experiment included a Web-based survey as a way of gathering data from the subjects. We chose to use a Web-based tool since it provides simple graphical user interface that is easily understandable by the subjects. For each software the users installed they had to answer survey questions about whether or not they:

- were familiar with the software
- thought the software were legitimate
- suspect that the software conveys negative consequences
- would allow the software to be installed on their own computer.

By measuring the difference in the subjects answers when the reputation system was disabled compared to when it was enabled, we could identify to what extent the reputation system affect the users' decision-making during software installation. In the next section we present the results from the experiment.

In the first stage of the experiment we asked the subjects regarding their knowledge about spyware and whether or not they had had any first hand negative experience from such software. It turned out that 63% of the subjects had good or very good knowledge about spyware programs, while 27% were neutral and 10% thought they weren't that educated about spyware. Also, 52% of the subjects had had first-hand experience from the negative effects associated with spyware, while 24% were neutral and 24% had no such experience. Finally, 88% of the subjects were either concerned or very concerned about the consequences from spyware, while the remaining 12% were neutral in this regard.

After trying the prototype of the software reputation system 92% of the subjects thought the information the system provided was

either helpful or very helpful during software installation, while 4% were neutral and the remaining 4% did not find the information useful.

Table 10.3 show the percentage of subjects that allowed and denied software installation, both with and without the software reputation system (SRS) enabled. Without any software reputation system on average 28% of the subjects allowed spyware programs to install, which was lowered to just over 5% when the subjects were provided with feedback from the reputation system. At the same time 44% on average allowed all legitimate programs to install when not presented with the software' reputations, which increased to 79% when assisted by the software reputation.

Table 10.3 Percent of subjects that allow or deny software installation both with and without the software reputation system (SRS) enabled.

	Without SRS (%)		With SRS (%)		Difference
	Yes	No	Yes	No	
Spy.: ABC Scrabble	29	71	6	94	23
Leg.: TMR	6	94	69	31	63
Spy.: RadLight	53	47	10	90	43
Spy.: Italian Soccer	2	98	0	100	2
Leg.: WinZip	82	18	88	12	6

The subjects' influence by the software reputation system was more extensive for the first three programs in Table 10.3, while it decreases for Italian Soccer and WinZip. This can be described by the fact that WinZip is a widely known utility program with around 200 Million downloads from Download.com, which explains why 92% of the subjects were already familiar with the program. Therefore 82% of the subjects decided to install WinZip without any assistance from the reputation system. Italian Soccer was only known by 2% of the subjects. However, the installation process of the program includes some quite obvious clues that this program is questionable that explains why only 2% agreed to install the program.

10.6 Simulation of Software Reputation System

We have implemented a simulator in Java to investigate scenarios concerned with the usage of a software reputation system [6]. The simulator is deterministic which means that it is possible to rerun scenarios several times and always reach the same results, or more interestingly to change a certain variable in the scenario setup and be sure that the changes in the end-result are due to the alteration

of that particular variable. The scenarios simulated are mainly concerned with the effects that user demography has on system accuracy, i.e. whether a few experts can compensate for a larger group of normal users with limited expertise. The simulator can simulate groups of malicious users that are engaged in attacks on the system accuracy, e.g. through Sybil attacks [16]. Simulating such scenarios allows us to identify what system configurations that provides a well behaving system for different user demographics, and what level of malicious users the system can withstand.

Each software being simulated is associated with a "correct" rating that is used for evaluating the accuracy of a simulated scenario. The evaluation consists of summarizing the absolute distance between each software's correct rating and its weighted average rating, and finally dividing the sum with the total number of software being simulated. We refer to the resulting value as the *evaluation score* (ES) that represent the average distance between all programs correct rating and actual rating. An ES of 0.0 therefore mean that the software reputation system on average provides its users with ratings that are right on target.

In the simulations we divide all users into three groups based on their accuracy in rating software, and to some extent their technical knowledge. Each user simulated belongs to exactly one of these groups, which determines the users voting variance (or error rate), i.e. the probability of rating software at some distance away from its correct rating. Each group's voting variance lies within the interval [+5, -5] inclusive. Expert users rate software correctly 50% of the occasions, and in the remaining part rate the software either one step below or above its correct rating, i.e. the expert users always manage to rate a software within a 3 step wide window around its correct rating. The second group is the average users that tries to rate software correctly, but with lesser accuracy than the experts, i.e. they rate up to 3 steps above or below the correct rating due to lack of skills to an increasing extent. Still an average user is better than a novice user that has an error margin of 5 steps above or below the correct rating.

Even worse than the novice users are the malicious users that have a list of targeted software to attack, and their goal is to either boost or zero-out the reputation of targeted software. Malicious users begin by first building up their trust within the system, i.e. increasing their trust factor (TF), by legitimately rating other software within the system before starting to attack the targets. After the

malicious users have increased their TF they begin mixing malicious votes on the targeted software with legitimate votes for any other software.

10.6.1 Results

To find suitable configuration settings for the SRS we simulated different configurations while measuring the accuracy within the system. One important configuration setting that we simulated was the factor that modifies the users' TF either up or down depending on how accurate the vote is, e.g. 1.25 which means a user's TF would either increase or decrease with a factor of 1.25 based on that user's vote accuracy. Another configuration setting that we simulated was the TF-limit that specifies the maximum TF allowed within the SRS. Generally we want to have an as high TF-limit as possible to amplify the impact from a few experts so they compensate for a larger group of less skilled users. However, at the same time we don't want a small group of users to have a too high TF; since this would give them too much control over the total outcome within the SRS, and it can be exploited by malicious actors. Figure 10.2 shed some light on this trade-off by plotting different TF-limits in combination with different modification factors for the TF.

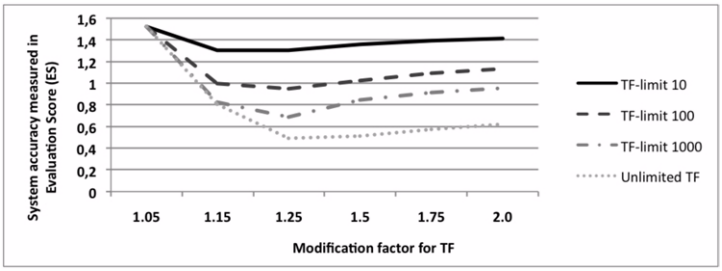

Figure 10.2 **Simulation results showing system accuracy at various modification factors of the TF and different TF-limits.**

Figure 10.2 show that the optimal setting of the TF-limit (within the simulated scenario) is when an unlimited TF is used, i.e. when no limit exists. Unfortunately such a setting would not work in practice since the TF of the expert users quickly reach astronomical proportions, giving a small number of users huge impact within the system, which can be misused. Based on the results in Figure 10.2 we therefore recommend that a TF-limit of 1000 is used in combination with a TF modification factor of 1.25. In another scenario we simulated how different proportions within the three user groups

would affect the system accuracy, e.g. what impact halving the number of experts would have on the system accuracy. We therefore simulated the baseline of 9.4% experts, 27.0% average, and 63.6% expert users, in combination with doubling and halving the number of expert and average users. The result of these simulations can be seen in Figure 10.3 below.

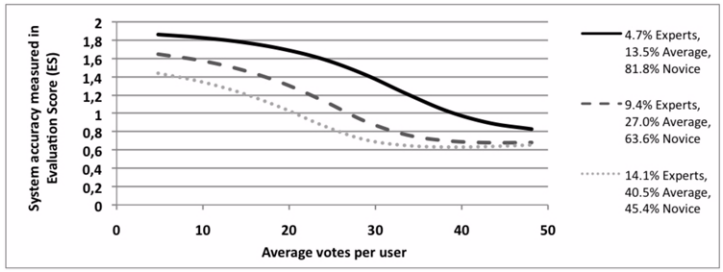

Figure 10.3 **Simulation results showing how user demography affects the system accuracy.**

Figure 10.3 show that all three scenarios end up with approximately the same accuracy when each user has voted 48 times. When a larger proportion of experts is included the accuracy increases more rapidly. Even if the proportion of experts within the system is below 5% it is still possible to reach acceptable system accuracy within the SRS.

We also simulated groups of malicious actors with the goal of either increasing or decreasing the reputation of certain targeted software within the SRS. These simulations included 0.1%, 1.0%, 5.0% and 10% malicious users out of the total population of one million users. The goal of the malicious users was to stage a collaborative Sybil attack against the same 1000 targeted software by decreasing their reputation. The malicious users were modelled so they wouldn't start voting maliciously until they had increased their TF by first voting legitimately on other software within the SRS (this threshold was 50% of the TF-limit of 1000), i.e. they would act as normal users with only brief exceptions where malicious votes are casted on the targeted software.

The results in Figure 10.4 show that a TF-limit of 1000 quickly deviates the reliable software rating with increasing number of malicious users present. While larger TF-limits would show a delayed deviation of the rating, i.e. after one to two years when each user is simulated to vote twice a month. This means that we either need to limit the growth rate of the users TF or restrict the number of votes

available for each user. However, we do not need to discriminate the voting behaviour of malicious users since reputation systems with large TF-limits will withstand severe Sybil attacks from a fairly high proportion of malicious users (in this scenario between 0.1-1.0%).

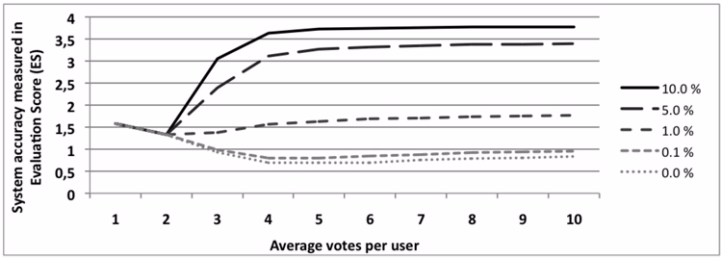

Figure 10.4 **A directed attack against 1000 pre-selected software where a population of 0.1, 1.0, 5.0 and 10% malicious users first establish a reputation in the system and then start a Sybil attack.**

10.7 Discussion

In major operating systems today, e.g. Microsoft Windows, software installations is carried out in an ad-hoc manner, which expects the user to know what she is doing. A common scenario is when the user retrieves a software package from an unknown Web site, and then executes it on the system. The disadvantage with this installation process is the lack of control over what software that enters the system. Also, the instrumentation that allows users to evaluate the software prior to the actual installation is inaccurate or non-existing. Without such instrumentation it is troublesome, if not impossible, for the users to give an informed consent for the software to enter their system. However, if countermeasures based on a software reputation system and automatic EULA classification was used they would increase users' awareness about software. This would transform the PIS classification so that any software that presents complete and correct information to the user during the installation is represented as one of the three legitimate software types on the top row in Table 10.4. Exactly which one depends on their behaviour and consequences for the system, and its user. All types of software that currently is targeted by traditional anti-spyware mechanisms are either removed by the introduction of the preventive mechanisms, or are fully covered by anti-virus tools. As a result both anti-spyware and anti-virus tools will join up in combined tools that address the software represented on the bottom line in Table 10.4.

Table 10.4 Transformation of PIS matrix as a consequence of users having access to automatic EULA classifications and a software reputation system.

	Tolerable	Moderate	Severe
Full	Legitimate software	Adverse software	Double agents
None	Covert software	Semi-parasites	Parasites

By using the information about software given by the countermeasures it is possible for users to decide on which software they regard as acceptable and which they don't. In other words, there exist a possibility of individual adjustment based on personal preferences, i.e. each user can decide which of the software types on the full informed consent row that they allow to be installed. The pros and cons for a single software are entirely up to the user to decide about. However, any software not playing by the rules, in terms of properly announcing their intent prior to the installation should rightfully be targeted and handled by anti-virus tools. This imply that anti-virus tools should not only target software that use exploitable system vulnerabilities to gain entrance to systems, but also software that deceive users about their business by using inferior user disclosure. From this follows that anti-virus mechanisms should handle any software not subordinate to the rules of complete prior disclosure and consent from the users. Any dishonest software slipping through the anti-virus tools' detection would impact a few initial systems. Then affected users would downgrade the responsible software via the reputation system so that subsequent users are more restrictive. Over time the reputation will catch up on these questionable software vendors, forming a future deterrent effect.

As an analogy it is interesting to discuss how the food industry looked like in the end of the 19th century, when customers had great problems evaluating food products due to the flourishing distribution of "snake-oil" product [46]. Such products claimed to do one thing, for example to grow hair, while they instead made unwitting consumer addicted to habit-forming substances like cocaine and alcohol. In 1906 the Pure Food and Drug Act was introduced in the United States, which made it possible to punish any food manufacturer according to the law if not complying with the rules. As a consequence the manufacturers followed these rules, allowing

consumers to trust the information on the food container to be correct.

This further allowed customers to make informed decisions on whether they should consume a product or not, based on their individual preferences and by taking for instance nutrition, allergies, degree of fat and sugar into account. As long as the food does not include dangerous substances or use deceptive descriptions it is up to the consumer to make the final decision.

By creating a distributed control system based on software reputations it is possible to provide information about software to the users. Based on this information users can decide exactly what software they regard as legitimate according to their own preferences, and allow only that software to enter their computer. However, it would also be possible for the users to create their own *installation policies* that automatically evaluate every software being installed. Such policies could for instance take into account:

- the software's overall rating;
- if any negative effects that are unacceptable for the user have been reported;
- if the software is cryptographically signed by a legitimate software vendor;
- the outcome from the EULA classification.

Using these constructs makes it possible to create a policy that trusts any software from certain vendors using digital signatures [44]. However, since far from all software vendors sign their software it would also be interesting for the users to include the overall rating in the policy. By specifying a certain threshold, users could signal that they only trust software that has an overall rating above that particular threshold. This could also be extended by taking into account whether a majority of the users has reported any unacceptable behaviour for the software, e.g. that it lacks a functioning uninstall routine. It would furthermore be possible to include the outcome from the automatic EULA classification in the installation policy. That way, only software with a legitimate EULA is allowed to be installed on the computer. In any borderline case it would of course also be possible to ask the user for his opinion, but such user interaction should be kept to a minimum since many users find it

annoying. In Figure 10.5 below we present a pseudo policy as an example of this technique.

```
if Software.Behaviour == Unacceptable; then:
   Deny installation;
else:
   if Software.Signature == TrustedVendor; then:
      Allow installation;
   else:
      if (Software.Rating >= 7.5) AND (Software.EULA == Legitimate); then:
         Allow installation;
      else:
         Ask permission;
      end if
   end if
end if
```

Figure 10.5 Example of installation policy in pseudo code.

The example policy in Figure 10.5 denies any program from being installed if it has been marked with one or more behaviours that the user has specified as unacceptable in the local policy, e.g. showing pop-ups. For any other software it is allowed to install if it is signed by one of the vendors trusted by the user, or if the software has both a rating that is 7.5 or higher and includes a EULA that has been found to be legitimate. If the software has a lower rating or does not includes a legitimate EULA, the user is asked to manually allow or deny the software to be installed. At this point the user could also be given a summary of the software's reputation and results from the EULA classification. However, in many cases the installation policy would verify the software without any interaction from the user at all.

If this kind of decision support system for software installation based on software reputations and automatic EULA classification was used the users would experience the following three benefits:

- a customizable lowest level with regard to software behaviour that is accepted for software on their computer;
- a basis on which software behaviour and consequence can be evaluated prior to any software installation and system modification;
- possibilities for users to define individual software preferences, which allow for transparent decision making that decrease the need for user interaction.

Today most users are incapable of protecting themselves from grey-zone programs and therefore have to either trust inaccurate anti-spyware tools, or try to carry out time-consuming online investiga-

tions to determine if a software is legitimate or not. Compared to this situation we believe many computer users would be greatly benefited by the techniques we propose.

10.8 Conclusions and Future Work

Users need to know what software they install, and learn how to distinguish between acceptable and intolerable software, a priori to any software installation. Everyone should be presented with accurate and condensed information about the software's function before the installation process begins. We argue that additional mechanisms that safeguard user's informed consent are required.

The concept of privacy-invasive software, PIS, together with two countermeasure techniques is introduced. We classify PIS using a matrix, where tolerable, moderate and severe negative consequences where matched against full, medium and none informed consent. A decision support system for software installation based on software reputations and automatic EULA classification is proposed. Our simulation indicates that software reputation systems would be helpful even in the presence of malicious users, conducting a Sybil attack, if they are kept low (below 1% of the population) or their number of votes is restricted. Automated EULA classification use data mining and text classification techniques that are capable of classifying previously unseen EULAs as either good or bad.

In the future, software behaviour may be investigated by analyzing users search patterns on e.g. Google. Recently query data were used for detecting influenza epidemics by comparing high ranked search queries with surveillance data [21]. In our setting undefined programs show traces of good or bad behaviour depending on the "collective intelligence" of users formulating search query data. For this to take place, a future investigation including a participating search engine is necessary. So it could be possible for computer users to combine an automatic EULA analysis, a semi-automatic reputation system, and a "query analyzer" when making informed decisions regarding PIS.

10.9 References

[1] W. Ames, "Understanding Spyware: Risk and Response", in the *IEEE Computer Society*, Volume 6, Issue 5, 2004.

[2] W. Arnold and G. Tesauro, "Automatically generated Win32 heuristic virus detection", in the *Proceedings of the 10th International Virus Bulletin Conference*, Orlando USA, 2000.

[3] Anti-Spyware Coalition, http://www.antispywarecoalition.org, Last checked: 2010-03-11.

[4] AOL/NCSA Online Safety Study, http://www.staysafeonline.org, 2010-03-11.

[5] K. P. Arnett and M. B. Schmidt, "Busting the Ghost in the Machine", in *Communications of the ACM*, Volume 48, Issue 8, 2005.

[6] M. Boldt, A. Borg, and B. Carlsson, "On the Simulation of a Software Reputation System", in the *Proceedings of the International Conference on Availability, Reliability and Security (ARES)*, Krakow Poland, 2010.

[7] M. Boldt and B. Carlsson, "Analysing Privacy-Invasive Software using Computer Forensic Methods", *ICSEA, Papeetee*, 2006.

[8] M. Boldt, B. Carlsson, T. Larsson, N. Lindén, "Preventing Privacy-Invasive Software using Online Reputations", in *Lecture Notes in Computer Science*, Volume 4721, Springer Verlag, Berlin Germany, 2007.

[9] J. Bruce, "Defining Rules for Acceptable Adware", in the *Proceedings of the Fifteenth Virus Bulletin Conference*, Dublin Ireland, 2005.

[10] S. Byers, L.F. Cranor, and D. Kormann, "Automated Analysis of P3P-Enabled Web Sites", in the *Proceedings of the Fifth International Conference on Electronic Commerce (ICEC2003)*, Pittsburgh USA, 2003.

[11] Center for Democracy &Technology, "Following the Money", http://www.cdt.org, 2010-03-11.

[12] E. Chien, "Techniques of Adware and Spyware", in the *Proceedings of the Fifteenth Virus Bulletin Conference*, Dublin Ireland, 2005.

[13] Clearware.org, http://www.clearware.org, 2010-03-11.

[14] L. F. Cranor, "Giving notice: why privacy policies and security breach notifications aren't enough", in *IEEE Communications Magazine*, Vol. 43, Issue 8, 2005.

[15] L. F. Cranor, "P3P: Making Privacy Policies More Useful", in the *IEEE Security & Privacy*, Volume 1, Issue 6, 2003.

[16] L. F. Cranor, *"Security and Usability"*, O'Reilly, Sebastopol, 2005.

[17] J. Douceur, "The sybil attack", in the *Proceedings of the 1st International Workshop on Peer-to-Peer Systems (IPTPS)*, 2002.

[18] Earthlink Spy Audit, http://www.earthlink.net/spyaudit/press/, 2010-03-11.

[19] Fawcett T (2003) ROC graphs – notes and practical considerations for data mining researchers. Tech. Rep. HPL-2003-4, Intelligent Enterprise Technologies Laboratories, Palo Alto, CA, USA.

[20] FreeBSD Ports, http://www.freebsd.org/ports/, 2010-03-11.

[21] B. Friedman, et. al., "Informed Consent Online: A Conceptual Model and Design Principles", *CSE Technical Report*, University of Washington, 2000.

[22] Gibson Research Corporation, "OptOut – Internet Spyware Detection and Removal", 2010-03-11.

[23] J. Ginsberg et al., "Detecting influenza epidemics using search engine query data", in *Nature*, Volume 457, Issue 7232, 2009.

[24] N. Good, et. al., "Stopping Spyware at the Gate: A User Study of Privacy, Notice and Spyware", in the *Proceedings of the Symposium on Usable Privacy and Security,* Pittsburgh USA, 2005.

[25] N. Good et al., "User Choices and Regret: Understanding Users' Decision Process about Consentually Acquired Spyware", in *S/A Journal of Law and Policy for the Information Society*, Volume 2, Issue 2, 2006.

[26] S. Görling, "An Introduction to the Parasite Economy", in EICAR 2004, Luxemburg, 2004.

[27] G. Jacob, H. Debar, E. Filiol, "Behavioral Detection of Malware: from a Survey Towards an Established Taxonomy", in *Journal in Computer Virology*, Volume 4, Issue 3, 2007.

[28] A. Jacobsson, M. Boldt, and B. Carlsson, "Privacy-Invasive Software in File-Sharing Tools*"*, in *Proceedings of the 18th IFIP World Computer Congress,* Toulouse France, 2004.

[29] A. Jøsang, et al., "A Survey of Trust and Reputation System for Online Service Provision", in *Decision Support Systems*, Volume 43, Issue 2, 2007.

[30] J. O. Kephart, "Biologically inspired defenses against computer viruses", in the *Proceedings of the 14th International Joint Conference on Artificial Intelligence (IJCAI'95)*, Montreal Canada, 1995.

[31] Lavasoft, http://www.lavasoftusa.com, 2010-03-11.

[32] N. Lavesson, M. Boldt, P. Davidsson and A. Jacobsson, "Learning to Detect Spyware Using End User License Agreements", in *Knowledge and Information Systems*, Springer, to appear.

[33] J.S. Lee, J. Hsiang and P.H. Tsang, "A Generic Virus Detection Agent on the Internet", in *Proceedings of the Thirtieth Hawaii International Conference on System Sciences*, Wailca HI, 1997.

[34] S. Mansfield-Devine, "The Promise of Whitelisting", in *Network Security*, Volume 2009, Issue 7, 2009.

[35] D. M. Martin Jr, et. al., "The privacy practices of Web browser extensions", in *Communications of the ACM*, Volume 44, Issue 2, 2001.

[36] Moshchuk, et. al., "A Crawler-based Study of Spyware on the Web", in the *Proceedings of the 13th Annual Network and Distributed System Security Symposium (NDSS 2006)*, San Diego CA, 2006.

[37] T. Okamoto and Y. Ishida, "A Distributed Approach to Computer Virus Detection and Neutralization by Autonomous Heterogeneous Agents", in the *Proceedings of the 4th International Symposium on Autonomous Decentralized Systems (ISADS'99)*, Tokyo Japan, 1999.

[38] C. Robson, *"Real World Research, 2nd Edition"*, Blackwell Publishing, Malden MA, 2008.

[39] Schneier, "Inside risks: semantic network attacks", in *Communications of the ACM*, Volume 43, Issue 12, 2000.

[40] Shapiro and H. R. Varian, *"Information Rules – A Strategic Guide to the Network Economy"*, Harvard Business School Press, Boston Massachusetts, 1999.

[41] S. Shukla and F. F. Nah, "Web Browsing and Spyware Intrusion"*,* in *Communications of the ACM*, Volume 48, Issue 8, 2005.

[42] J. C. Sipior, "A United States Perspective on the Ethical and Legal Issues of Spyware", in *Proceedings of Seventh International Conference on Electronic Commerce*, Xi'an China, 2005.

[43] E. Skoudis,*"Malware – Fighting Malicious Code"*, Prentice Hall PTR, Upper Saddle River NJ, 2004.

[44] "Spyware": Research, Testing, Legislation, and Suits, http://www.benedelman.org/spyware/, 2010-03-11.

[45] S. Steinbrecher, "Design Options for Privacy-Respecting Reputation Systems within Centralised Internet Communities", in *Proceedings of the 21st IFIP SEC 2006*, Karlstad Sweden, 2006.

[46] StopBadware.org, http://www.stopbadware.org, 2010-03-11.

[47] W. Sun et al., "Expanding Malware Defense by Securing Software Installations", in *Lecture Notes In Computer Science*, Volume 5137, Springer Verlag, Berlin Germany, 2008.

[48] P. Szor, *"The Art of Computer Virus Research and Defence"*, Pearson Education, Upper Saddle River NJ, 2005.

[49] Technology Review – The Pure Software Act of 2006, http://www.simson.net/clips/2004/2004.TR.04.PureSoftware.pdf, 2010-03-11.

[50] TRUSTe – The Trusted Download Program (Beta), http://www.truste.org/trusteddownload.php, 2010-03-11.

[51] M. Warkentin, et al., "A Framework For Spyware Assessment", in *Communications of the ACM*, Volume 48, Issue 8, 2005.

[52] Webroot Software, *"State of Spyware – Q3 2005*", http://www.webroot.com/resources/, 2010-03-11.

[53] I. H. Witten, and E. Frank, *"Data Mining – Practical Machine Learning Tools and Techniques"*, Elsevier, San Francisco CA, 2005.