Preventing Privacy-Invasive Software using Collaborative Reputation Systems

M. Boldt, B. Carlsson, T. Larsson, and N. Lindén

Blekinge Institute of Technology, Box 520, SE-372 25, Sweden
{mbo,bca}@bth.se, {tola01,nili02}@student.bth.se

Abstract. Privacy-invasive software, loosely labeled spyware, is an increasingly common problem for today's computer users, one to which there is no absolute cure. Most of the privacy-invasive software are positioned in a legal gray zone, 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.

1 Introduction

Our society is continuously moving in an increasingly more computerized direction where software has a central role [13][24]. 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 labeled *spyware* [28][33]. 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 is 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 gray zone, since they normally inform the users of their actions, but often in such a format that it is unrealistic to believe that nor-

mal computer users will read and understand the provided information. The *End User License Agreements* (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, which can be anything the software developer wants [5][12][27].

There are numerous ongoing projects and attempts to produce effective countermeasures for removing spyware [19][21][29]. 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 which the user already has accepted, which in term could lead to law suits [12][30]. Such legal disputes has already proved to be costly for anti-spyware software companies [25]. 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 [10][15][22]. 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 [34]. 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] [18]. 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.

1.1 Background and Related Work

The usage of the term spyware has become increasingly popular, both by users, media and software vendors [1][26]. 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" [6][11]. 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) [27]. A better term to use instead of spyware, would be pri-

1. The tool is available for free from: http://www.softwareputation.com

vacy-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 1: [3][4].

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 a low user consent, *or* which impairs severe direct 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.

	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

Table 1: Classification of privacy-invasive software with respect to user's informed consent (high, medium and low) and negative user consequences (tolerable, moderate and severe).

We base our work on Simone Fischer-Hübner's definition of *privacy*, in which she divides the concept into the following three areas [9]:

- *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 which affect 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, both 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 affect 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" [23]. 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 trust 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.

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 fully prevent the problem, or at least reduce the impact [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 program, highly recommended. 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. How-

ever, 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 whitelisted 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 email 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 [8].

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 this 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 all 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. Protection of users' anonymity could be established by utilizing distributed anonymity services, such as Tor, for all communication between the client and the server [32].

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 that of 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. The overall design is presented in Figure 1.

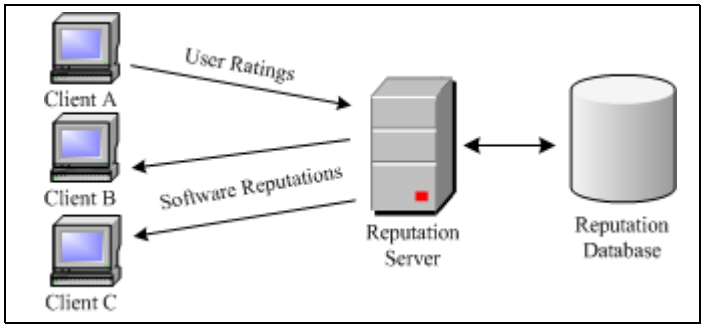

Figure 1 The architectural design for the proof-of-concept reputation system.

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 a 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 [14]. It consists of a system driver that replaces the API call to `NtCreateSection()` with its own version, and a software component that communicates with the driver through a shared section of the memory.

The client uses different lists to keep track of which software have been marked as safe (the whitelist) and which have been marked as unsafe (the blacklist). These two lists are then used for automatically allowing or denying a 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 whitelist and blacklist for an occurrence of the pending software based on a MD5 checksum calculated from the EXE-file content. 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.

3.2 Server Design

In addition to the processing of software ratings the server also handles the database containing registered user information, ratings and comment 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, and each address used to sign up must be a valid e-mail address, 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 mechanisms, 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, an 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.

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 product 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 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 fingerprint is generated through an hash algorithm (in this case an MD5 digest) 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:

- Fingerprint of software executable, in this case a generated MD5 sum.
- 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 metadata that always can be retrieved once the complete file is in ones possession.

Since hash functions are used, the fingerprints 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.

Furthermore, it is 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.

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.

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. 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.

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 [31]. 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 to the rules to be punished according to the law [17]. As a consequence the manufacturers followed these rules, allowing consumers to trust the information on the food container to be correct. 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 were mitigated in 1906, its digital counterpart continue 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 1:. The introduction of this type of user-oriented countermeasure would transform the classification of PIS as shown in Table 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 soft-

ware 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 allow users to rely on 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. Making it hard for dissatisfied spyware vendors to sue the reputation system owners for defamation.

	Tolerable Negative Consequences	**Moderate Negative Consequences**	**Severe Negative Consequences**
High Consent	Legitimate software	Adverse software	Double agent
Low Consent	Covert software	Semi-parasites	Parasites

Table 2: Difference between legitimate software and malware with respect to user's informed consent and negative user consequences.

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, and the Windows operating system is not prone to graceful degradation, 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 inadvertedly 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.

A possible approach to this problem would be an enhanced whitelisting system that could examine the file about to execute, to determine if it has been digitally signed by a trusted vendor e.g., Microsoft. 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 whitelist and blacklist different companies through their digital signatures, which – in turn – could considerably lower the need for user interaction.

The introduction of an enhanced whitelisting 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 corpo-

rations 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 imply 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.

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 gray zone 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 large database that must be downloaded and 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 reliability 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 on-going 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.

5 Conclusions and Future Work

This paper explore 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, downloads 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 allow 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, whether 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 affect 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 [20]. The results from such investigations could then be inserted into the reputation system as hard evidence on the behaviour for that specific software.

6 References

[1] W. Ames, "Understanding spyware: risk and response", in *IEEE IT Professional*, Volume 6, Issue 5, 2004.

[2] K. P. Arnett, "Busting the Ghost in the Machine", in *Communications of the ACM*, Volume 48, Issue 8, 2005.

[3] M. Boldt, "Privacy-Invasive Software - Exploring Effects and Countermeasures", Licentitate Thesis Series No. 2007:01, School of Engineering, Blekinge Institute of Technology, Sweden, 2007.

[4] M. Boldt and B. Carlsson,"Privacy-Invasive Software and Preventive Mechanisms", in the *proceedings of the IEEE International Conference on Systems and Networks Communications (ICSNC06)*, Papeete Tahiti, 2006.

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

[6] M. Christodorescu and S. Jha, "Testing Malware Detectors", in the *proceedings of the ACM International Symposium on Software Testing and Analysis*, 2004.

[7] C. Dellarocas, "Immunizing Online Reputation Reporting Systems Against Unfair Ratings and Discriminatory Behaviour", in the *proceedings of the 2nd ACM Conference on Electronic Commerce*, 2000.

[8] J. Douceur, "The Sybil Attack", in the *proceedings for the 1st International Workshop on Peer-to-Peer Systems*, 2002.

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

[10] Flixster, http://www.flixster.com, 2006-09-13.

[11] 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.

[12] 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.

[13] A. Greenfield, "*Everyware - The Dawning Age of Ubiquitous Computing"*, New Riders, Berkeley CA, 2006.

[14] Hooking the native API and controlling process creation on a system-wide basis, http://www.codeproject.com/system/soviet protector.asp, 2006-11-23.

[15] Internet Movie Database, http://www.imdb.com, 2007-02-23.

[16] E. Kirda, C. Kruegel, G. Banks, G. Vigna, R. A. Kemmerer, "Behaviorbased Spyware Detection", in the *proceedings of the 15th USENIX Security Symposium*, 2006.

[17] Landmark Document in American History, "Pure Food and Drug Act of 1906", http://coursesa.matrix.msu.edu/~hst203/documents/pure.html, Last checked: 2006-10-16.

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

[19] LavaSoft Ad-Aware, http://www.lavasoftusa.com/software/adaware, 2006-09-19.

[20] A. Moshchuk, T. Bragin, S. D. Gribble, H. M. Levy, "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.

[21] Norton Internet Security, http://www.symantec.se/region/se/product/nis index.html, 2006-09-19.

[22] Pricerunner, http://www.pricerunner.com, 2006-09-13.

[23] P. Resnick, K. Kuwabara, R. Zeckhauser, E. Friedman, "Reputation Systems",in *Communications of the ACM*, Volume 42, Issue 12, 2000.

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

[25] See you later - anti-Gators, CNET News.com, http://news.com.com/2100-1032 3-5095051.html, 2006-09-19.

[26] K. Schultz, "Sticking It to Spyware", in *InfoWorld*, Volume 27, Issue 38, 2005.

[27] 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.

[28] Spyaudit, http://www.earthlink.net/spyaudit/press/, 2006-09-12.

[29] Spybot -Search & Destroy, http://www.safer-networking.org, 2006-09-19.

[30] "Spyware":Research, Testing, Legislation, and Suits, http://www.benedelman.org/spyware/, 2007-03-01.

[31] Technology Review, "The Pure Software Act of 2006", http://www.simson.net/clips/2004/2004.TR.04.PureSoftware.pdf, Last checked: 2006-10-16.

[32] Tor: anonymity online, http://tor.eff.org, 2007-02-24.

[33] Webroot Software — Internet Spyware and statistics about infection rates, http://www.webroot.com/resources/stateofspyware/excerpt.html, 2006-09-12.

[34] G. Zacharia, A. Moukas, P. Maes, "Collaborative Reputation Mechanisms in Electronic Marketplaces", in the *proceedings of the 32nd Hawaii International Conference on System Sciences*, 1999.