SpySaver: Using Incentives to Address Spyware

Stefan Saroiu
University of Toronto

Alec Wolman
Microsoft Research

ABSTRACT

Despite the many solutions proposed by industry and the research community to address spyware, this problem continues to grow. Many of today's anti-spyware approaches are inspired by techniques used against related security problems, such as worms, DoS attacks, computer viruses, and spam. Although these techniques have been retrofitted to address spyware, they remain ineffective because they rely on the compromised host to detect and remove spyware. Once a host is compromised, attackers often find simple ways to escape spyware detection and removal.

This paper presents SpySaver – a novel anti-spyware approach that reduces the incentive to deploy spyware. Our approach does not prevent spyware installations, nor does it recover from them. Instead, SpySaver decreases the value of the information spyware collects by creating counterfeit information. Our goal is to generate enough counterfeit information to devalue the information gathered by spyware to the point that we eliminate the incentive to collect it in the first place. In this paper, we present our approach and an initial design of a tool that produces realistic counterfeit information about the browsing patterns of Web users.

Categories and Subject Descriptors

D.4.6 [**Operating Systems**]: Security and Protection—*Invasive software*

General Terms

Economics, Security

Keywords

Spyware

1. INTRODUCTION

Today's anti-spyware tools have done little to stop the proliferation of spyware programs. Relying on an already infected host to detect and remove spyware programs has proven challenging. For example, attackers can easily adapt to anti-spyware tools that use signature-based techniques to discover and identify spyware programs. Recently, spyware programs themselves have started to use signatures to detect the presence of anti-spyware tools to disable and remove them from the infected machine. Recognizing the difficulties inherent in leveraging already compromised machines to defend against spyware, several research projects have started to propose systems that use special hardware (e.g., a graphics card) or virtual machines to protect sensitive data on commodity systems [3, 4]. While these systems make great strides towards protecting users from spyware, they have yet to be deployed at large-scale. Their lack of success is primarily due to the drastic changes they require in the usability or programmability of today's commodity systems.

Despite the many anti-spyware solutions, the spyware problem has continued to gain momentum. Recent findings suggest that millions of Internet hosts are infected with spyware programs [6, 19]. This large number of spyware infections is caused by two main factors. First, most Internet users remain unaware of the common mechanisms used by spyware to propagate, as well as the security and privacy implications of becoming infected by spyware. One study has found over 70% of those users infected with Claria or WhenU are unaware that spyware programs are running on their desktops [18]. Second, spyware writers have a direct financial incentive to find creative ways of infecting as many Internet hosts as possible. This is because the information collected by spyware programs has financial value. This information is important to Internet vendors and advertisers – they use it to build profiles of Internet behavior at large or to display targeted advertisements (e.g., browser pop-ups) to users. As spyware infects larger Internet user populations, the spyware programs become more effective: the gathered profiles become more accurate and more valuable, and the targeted advertising reaches larger user bases.

Spyware poses many risks to end users. Because these programs run surreptitiously and gather information without the explicit consent of users, spyware compromises these users' privacy. Spyware is also having a significant effect on reliability: a report by the Federal Trade Commission (FTC) mentions that spyware programs are responsible for as much as fifty-percent of all Windows crashes reported to Microsoft [10]. There are reports that spyware degrades the infected hosts' performance [20]. Spyware may also appropriate resources of the computer that it infects [21] or alter the functionality of applications [17]. Finally, an earlier study has found evidence that spyware programs pose security vulnerabilities [19].

In this paper, we use an idea inspired by economics to address the proliferation of spyware programs: creating counterfeit information. This technique does not prevent spyware installations, nor does it recover from them; instead, it focuses on decreasing the value of the information spyware collects. Information about the behavior of real users has value to the vendors that produce spyware, yet mixing in counterfeit information can significantly devalue the aggregate information collected by the spyware. This approach must satisfy two conditions in order to be successful. First, we must generate enough counterfeit information to devalue the aggregate information to the point that we eliminate the incentive to collect it in the first place. Second, we must make the counterfeit information sufficiently similar to the legitimate information that separating them is too expensive. Ours is not the first attempt to use an economic approach to address a security problem: [5, 14] investigate using economics to address the spam problem.

Our approach has two key advantages. The first advantage lies in its generality; we do not have to understand the behavior of individual spyware programs to be effective in combating them; in contrast, signature-based approaches need to continuously update their signature lists. Second, our approach need not be perfect to be successful. Ultimately, SpySaver's goal is to convince the spyware writers that there are easier ways to advertise on the Internet or to collect valuable information than through spyware programs. One weakness of our approach is that we do not address all types of malicious programs that could infect users' machines. In particular, malware programs such as keystroke loggers are not significantly harmed by SpySaver, as long as the information such programs collect can be validated independently.

We present an initial design of SpySaver, a program that uses a computer's idle time to create counterfeit information. The goal of SpySaver is to create the illusion of many fake users browsing the Web in addition to the real user, and to cause spyware programs to gather the forged information. Our tool runs in the background, similar to a screen saver, and it stops the fake browsing when the real user is active.

SpySaver uses a virtual machine architecture to isolate each user, both fake ones and real ones. In this way, the browsing experience for real users is largely unaffected by the existence of the fake users. SpySaver uses pre-recorded browsing sessions to drive the behavior of each fake browser; a browsing session is basically a transcript of all Web activity required to emulate a real user's Web surfing activity. For each fake user, SpySaver downloads profiles for browsing sessions from a centralized repository. Whenever a fake browser replays a browsing session, the HTTP requests are intercepted and answered by a local proxy cache to ensure that the fake browsing sessions do not generate a significant amount of additional network traffic. If the intercepted HTTP request cannot be matched with a pre-loaded response, the proxy cache forwards this request to the actual Web origin server. The proxy serves two roles: (1) to create the illusion that the fake Web users browse successfully even when using forged information and credentials; and (2) to reduce the amount of Web traffic generated by our tool. All of our tool's components run outside the virtual machine boundaries isolating users. In this way, the spyware cannot look for our tool's presence within their virtual machine boundaries.

2. UNDERSTANDING SPYWARE

In this section, we start by describing how desktop machines become infected with spyware, the kinds of information that spyware programs gather, and the ways this information is used. Our description of spyware is brief; for more details see [10]. SpySaver is only targeted to address the classes of spyware described below – it is not a general solution for all malicious programs.

Upon installation, spyware programs typically perform one of the following tasks: gathering information about the user's browsing habits; and delivering targeted advertisements to users (e.g., browser pop-ups.) The gathered information may include lists of previously visited URLs, the user's age, gender, zipcode, country, and e-mail address [1, 2]. Some spyware programs gather more specific information, such as the machine's MAC address or the names of the CDs being played [15]. There are reported cases where the collected information is transmitted in plaintext over the Internet, increasing the possibility of exposing it to even more foreign parties [2, 7]. The gathered information can be used to build repositories of information on how users browse the Web (e.g. 20% of users who visit site A will also visit site B). This information is valuable to Internet vendors, advertisers, or even malicious parties. For example, advertisers can mine this data to measure the effectiveness of their advertisements; similarly, Internet vendors can use it to extract valid e-mail addresses for targeted advertising [26].

Another class of spyware programs, often referred to as adware, displays pop-up advertisements on the user's screen. Some adware performs contextual advertising [11], creating pop-ups that are relevant to the user's browsing context. For example, a spyware program can detect when a user is browsing an e-commerce website and decide to pop-up a competitor's advertisement, or it can detect the existence of a keyword in the currently displayed web page and pop-up an advertisement related to that word. Contextual advertising is attractive to Internet vendors because it offers the possibility of increasing ad relevance. Spyware companies enter into agreements with these vendors to distribute and display their ads, and the spyware programs provide tracking mechanisms to collect the number of times that users click on the displayed ads. These tracking counters are essential to the business model of spyware vendors – they quantify the spyware program's effectiveness and the value of the services provided to Internet vendors. There is already evidence that inflated tracing counters are a major cause of concern for Internet advertisers [27]. It has been recently reported that several pop-up spyware programs inflate their click-counters in an attempt to overcharge their advertisers [9].

In summary, spyware gathers information because it has value. In some cases, this information is used to create repositories that reflect how users browse the Web. In other cases, this information is used to charge Internet advertisers and vendors for the services that spyware provides. In either case, mixing counterfeit browsing information with the browsing information from real users decreases the value of the information spyware collects.

3. THE PROBLEMS WITH CURRENT ANTI-SPYWARE APPROACHES

Although spyware is a relatively recent phenomenon, the industry, the government, and the research community have already started to investigate solutions to the spyware threat. Most solutions borrow from techniques used to address related security problems, such as Internet worms, denial-of-service attacks, computer viruses, and spam.

3.1 Anti-Spyware Tools

As the spyware threat has been gaining momentum, a cottage industry of anti-spyware tools have started to emerge. Most tools use a signature-based approach to detect and remove spyware from a local host: they scan the local file-system searching for known spyware signatures. The effectiveness of this approach is determined by how complete and up-to-date the lists of the signatures of spyware programs are. Generating signatures of spyware programs is difficult for three reasons. First, these signatures must be able to distinguish between legitimate programs and spyware. Second, widespread deployment of the signature scanning approach will lead the spyware developers to make their programs polymorphic, thus making them harder to detect. Third, maintaining complete and up-to-date signature lists requires constant maintenance, which increases the costs of the scanning tools.

Another research project [25] takes a different approach: they monitor a set of known locations ("auto-start extensibility points") within the system, and detect when new programs are added to these auto-start locations. This approach makes it more difficult for spyware to hide itself during the installation process.

3.2 Reverse Firewalls

Both industry [28] and the research community [22] have started to investigate reverse firewalls. Unlike regular firewalls, reverse firewalls prevent the local machine from initiating network connections to unknown or unauthorized locations. These solutions make it harder for spyware to "silently" relay the collected information back to their servers. The primary challenge for these approaches is to distinguish between legitimate and malicious programs, because some legitimate programs, such as peer-to-peer applications, do initiate connections to unknown network locations.

3.3 Protecting Sensitive User Data

Another project [3] has started to investigate ways to hide (e.g. encrypt) the information that spyware collects. The basic premise of this approach is to confine the trusted computing base (TCB) to a limited set of components that have access to users' data, and to restrict spyware from infecting this TCB. In [3], the authors use the graphics card as the only trusted component on clients for a limited set of applications, such as a video player application. The graphics card performs encryption and decryption services so that sensitive data is never stored in main memory on the client, and is never exposed to the client operating system.

The key difficulty of this approach is structuring applications so that they can function correctly without access to the plaintext version of the sensitive data. This difficulty prevents this approach from being applicable to complex applications, such as Web browsers. For example, Web browsers often use Javascript to validate input on Web forms, before the data entered is sent to the Web server. It is likely impossible for a browser to validate an encrypted versions of the user's input.

3.4 Legislative Approaches

A different approach to addressing the spyware threat is to adopt legislative measures against the creation and distribution of spyware programs [13, 8]. These solutions are similar in nature to the anti-spam legislation currently adopted. We believe that legislative solutions will have limited effectiveness for three reasons. First, spyware is hard to define in practice. A program that is legitimate to one user may be considered a spyware program by another user. For example, the Google toolbar records the list of visited Web sites out of list of websites contained in a Google query's result. This information cannot be collected by the Google search servers; it can only be gathered from the users' desktops. While Google's privacy policy stipulates that they never share non-aggregate user information with third parties, they reserve the right to share records of users' browsing habits [12]. While some users find the Google toolbar's behavior acceptable [23], some identify it as spyware [16]. Second, the techniques used by spyware to infect machines are continuously evolving and being refined. Over time, legislation that precludes certain kinds of activities is likely to be circumvented through innovation. Third, the Internet is an international network; it is hard to enforce legislation across a large number of countries with different judicial and law enforcement systems.

4. OUR APPROACH: USING INCENTIVES

Our approach is to create synthetic Web browsing sessions that imitate the behavior of real Web users, and to let spyware programs gather this forged information. By creating these forged Web sessions, our intent is to taint the information that spyware collects and to artificially inflate the tracking counters for contextual advertising spyware. We use a proxy cache to filter out most of the additional workload we create before it reaches the origin servers, so that we primarily affect spyware running on client machines – our intent is not to break the existing business model for Web advertisements. However, we are careful not to filter at the proxy the additional click-through traffic we generate for tracking-counter inflation.

Tainting information decreases the value of the aggregate information repositories collected by the spyware vendors that characterize how users browse the Web. Mining information from these tainted repositories may lead to drawing false conclusions about Web users' behavior. A list of e-mail addresses extracted from a tainted repository may have little value if a large fraction of them are counterfeit. Similarly, inflating tracking counters is also detrimental to the spyware vendors' business model because these inflated counters misrepresent the success of the spyware program at delivering pop-up ads to real users. With inaccurate tracking counters, it is hard to measure how many real users actually click on an advertisement. This makes it difficult for the spyware companies to measure their effectiveness and to charge for their services.

In order to imitate users browsing the Web, one simple technique we considered involved creating additional browsing actions with the user's browser while that user is away. The primary drawback of this technique is that it interferes with Web experience of the real user, because it alters the user's browser state. For example, Web cookies could contain information relevant to the counterfeit actions rather than the real actions. When pressing the "Back" button or accessing the "History" menu, the real user might be redirected to a Web page never encountered before.

SpySaver uses a different technique: it creates new synthetic Web users and generates Web browsing actions within the environments of these additional Web users. The browsers of the forged users are deliberately infected with spyware programs, causing the spyware to collect forged information. The forged users also generate clicks on pop-up advertisements and browse the Web in a manner similar to the real users. The forged users identities are also counterfeit: they use forged e-mail addresses and login credentials. One drawback of this technique in comparison with the previous one is that detecting an individual browsing stream being generated by SpySaver may be an easier problem to solve than detecting SpySaver requests blended in with a real user's browsing stream. However, we view minimal impact on the user experience as our primary design goal, and therefore we constrain our solution to only generating requests from entirely separate user environments.

To avoid interfering with the browsing experience of real users, we need an mechanism to isolate the forged web users from each other and from the real user. Isolation is needed primarily to contain the spyware programs that we deliberately introduce into the forged users environments, but also to prevent any other state changes from affecting real users (e.g. a fake user visits a web page which installs an ActiveX control). Once a spyware program infects one domain, it can then only monitor the user within that domain. The isolation mechanism also makes it harder for spyware programs to directly test for our tool's presence. All of our tool's components run outside the perimeters isolating each Web user (both real and forged.) Our current choice for isolation is to use virtual machines; we discuss more details of our design in Section 4.

To deliberately infect the environments of the forged users with spyware, SpySaver performs periodic but infrequent snapshots of real user environments. These snapshots are then copied into the environments of the forged users. This ensures that the spyware programs that are affecting real users are the same programs whose information SpySaver is devaluing.

SpySaver periodically downloads information from a central repository on how to generate workloads for synthetic Web users, and activates Web browsing sessions for the forged users when it detects that the machine is idle. SpySaver is reminiscent of systems like SETI@home: users running SpySaver are virtually donating their machines' idle time to address the proliferation of spyware.

Our overall approach has several attractive properties. First, it is easily deployable: users can individually run this tool on their hosts. Second, by using the hosts' idle time, it creates minimal inconvenience to users. Third, by using virtual machines to provide isolation, it does not interfere with the way in which Web users browse today. We believe that the main limitation of our approach is that its effectiveness depends upon widespread deployment. It is likely that the early adopters of SpySaver will not see a direct benefit. However, each single adopter forges tens of fake Web users, so even a modest deployment should generate a significant amount of counterfeit information.

5. SPYSAVER: USING COMPUTER IDLE TIME TO FORGE WEB USERS

In this section, we present an initial design of SpySaver, a tool that exploits idle periods on desktop computers to generate Web browsing sessions by forged Web users. The goal of SpySaver is to create the illusion of multiple users browsing the Web in such a way that the spyware programs cannot distinguish between real users and forged users, and to let spyware programs gather information about all the users (both real and forged). SpySaver runs in the background, similar to a screensaver, and it halts all the fake browsing activities whenever a real user is active. SpySaver consists of a controller, a centralized repository of Web browsing information, and a proxy cache. These additional components are placed outside the perimeters isolating each Web user. In this way, the spyware programs cannot directly look for our tool's presence within their isolation boundaries. In this section, we start by defining the goals of our tool, and then we describe each component of SpySaver in turn.

5.1 Our Goals

To be effective, a tool that creates counterfeit Web browsing information must meet three design criteria:

1. It must be hard to disambiguate the real users from the counterfeit users. We believe that no solution can *guarantee* that the real information cannot be separated from the counterfeit information: one can develop ever more sophisticated algorithms that attempt to "scrub" the gathered information. Instead, our goal is simply to raise the bar: we want to make the *cost* of scrubbing the gathered information *more expensive* than the *value* of the information itself.

2. The tool's presence must be transparent to the real user. We believe that the success of a solution rests on its ability to not interfere with the user's experience when browsing the Web. In particular, we reject any design alternative that leads to changing the way users browse the Internet today.

3. As a consequence of the previous criterion, our solution's performance requirements must be minimal. To be successful, our tool must use lightweight techniques that conserve the host's resources, including CPU, RAM, disk storage, and network bandwidth.

5.2 Our Non-Goals

It is also important to emphasize what a tool creating counterfeit information *will not* do. Our tool does not attempt to protect information about real users from the spyware programs. For example, when a user does an online purchase, the spyware programs may still gather personal or sensitive data, such as email addresses and credit cart numbers. The goal of our tool is not to prevent spyware from collecting personal data, but rather to forge a significant fraction of the information it gathers.

5.3 The Controller

SpySaver uses virtual machines to provide isolation between user environments. Virtual machines provide the illusion of running multiple independent local hosts, each with its own hardware resources. This property aligns well with our second design requirement: making SpySaver transparent to the user. A real user just sees a normal operating system installation with a standard web browser. Any activity by forged users in other VMs will not affect the real user's browsing experience: all the browser state (including the browser cache, the history list, the user's cookies, plugins and ActiveX controls, etc...) remains unchanged.

We implement the SpySaver controller within the virtual machine monitor (VMM). The controller detects when the computer is idle and decides when to start and stop the different VMs to simulate the presence of additional Web users. Each user, including the real one, browses within a single VM. Each spyware program can only gather information specific to its own VM. This will make it harder for a spyware program to determine whether the user being monitored is fake or real. We deliberately attempt to infect the fake VMs with spyware programs, because otherwise it is unlikely that fake VMs would contain spyware. Our initial approach to infect VMs with spyware is to have the VMs download and run a large number of free software programs initially. Previous work has shown that many popular, free software programs contain multiple spyware instances [19]. In addition, we plan to periodically snapshot real user environments and copy them into the forged user environments, so that over time the spyware in the fake VMs will match the spyware that affects real users.

5.4 The Centralized Profile Repository

```
Wed Mar 26 08:00:00 2008 GMT: User: type URL: www.google.com
Wed Mar 26 08:00:02 2008 GMT: User:Type in search box: "Mortgage"
Wed Mar 26 08:00:03 2008 GMT: User: Click Submit button
Wed Mar 26 08:00:04 2008 GMT: Browser: Issue HTTP GET request
Wed Mar 26 08:00:05 2008 GMT: Browser: Received HTTP 200 response
```

Figure 1: ***A sample browsing session. A browsing session contains the information required for SpySaver to emulate a user accessing a specific set of Web pages.***

To generate a workload for the Web browsers running inside the fake VMs, the SpySaver controller replays pre-recorded Web browsing sessions. A browsing session contains enough information to emulate a user browsing a specific set of Web pages. Examples of browsing sessions are: issuing a query to a search engine and traversing the results of that query, browsing a news site, or reading and sending Web-based e-mail. A browsing session includes the user actions (e.g., clicking on HTML links, filling out forms, clicking on buttons, etc.) together with the browser's HTTP requests and responses. Figure 1 illustrates a browsing session describing a user issuing a query to a search engine.

For each fake Web user, the SpySaver controller periodically downloads browsing profiles from a centralized repository. These profiles are used to generate browsing sessions, which are replayed according to the timestamps of their actions. The role of timestamps is to ensure that the timing of actions generated by the SpySaver controller appear realistic, and therefore are difficult to distinguish from actions generated by real users.

We are currently investigating two approaches for creating browsing profiles: 1) volunteer Web users running instrumented Web browsers; and 2) network traces of real Web users' traffic. The advantage of the first approach is the ease of collecting information about a user's actions, whereas the latter approach provides the benefit of easier deployment for a large set of users. From network traces, we extract the URLs visited by each user along with each accesses timestamp. For each URL, we determine whether the user requested it by manually entering it, or by clicking on a link or a button. We decide that a URL was entered by clicking a link or button when that link was present in the content of a previous HTTP reply; otherwise we decide that the URL was manually entered. We use this distinction to delineate browsing sessions: each manually entered URL defines the start of a new browsing session.

5.5 The Proxy Cache

When the controller plays a browsing session, the corresponding HTTP requests are intercepted and served from the SpySaver proxy cache. The role of the proxy cache is to serve Web content only to those virtual machines driven by the SpySaver controller, and not to the VM serving the real user. Intercepting the requests generated by the controller is important because this allows the browsing sessions to contain realistic user behavior such as purchasing an item from a Web storefront, or accesses to a Web-based email site. Another benefit of the proxy cache is that it reduces the amount of extra Web traffic generated by our tool, alleviating any potential pressures from ISPs against a wide SpySaver deployment.

Before the SpySaver controller starts to replay a particular browsing session, the proxy cache is pre-loaded with all the Web content needed to replay that session. Whenever a browser driven by SpySaver issues an HTTP request, the proxy intercepts it and answers it with the corresponding HTTP response pre-loaded from the browsing session. If the intercepted HTTP request cannot be matched with a pre-loaded response, the SpySaver proxy cache forwards this request to the actual Web origin server.

There are two reasons why fake VM's may issue HTTP requests other than those that exactly match the browsing session. First, differences in browser configuration may cause minor variations in the HTTP headers when the browser is replaying a browsing session. In this case, the browser's HTTP requests should be sufficiently similar to be matched. Second, many spyware programs send the information they gather to the spyware servers using HTTP. These requests should not match any of the requests in the browsing session, and therefore SpySaver will not interfere with spyware communicating the forged information.

The proxy cache runs in a trusted environment and it communicates with the controller to determine which VMs are legitimate and which are fake. While proxy caches traditionally run on dedicated hardware, we envision the SpySaver proxy cache running in an additional trusted virtual machine. Figure 2 depicts our current architecture. In this figure, we illustrate the SpySaver proxy cache separate from the browsing virtual machines.

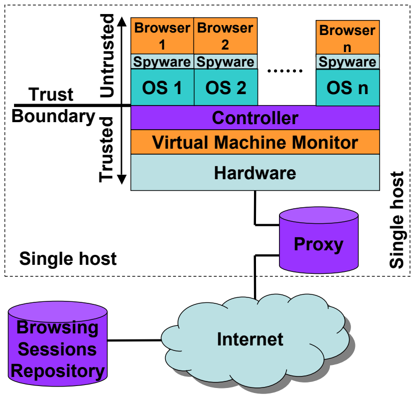

Figure 2: ***The SpySaver Architecture: SpySaver creates the illusion of multiple users browsing the Web. SpySaver runs in the background, similar to a screensaver, and it stops the fake browsing activity whenever the real user is active. A controller drives multiple browsers according to browsing sessions downloaded from a centralized repository. When a browsing session is replayed, the browser's HTTP requests corresponding to the browsing session's actions are intercepted and served from a proxy cache. Spyware programs are isolated within virtual machines without knowing whether the machine is fake or real.***

6. RESEARCH CHALLENGES

In this section, we outline some of the remaining challenges and our current thoughts on how to address them.

One issue is our claim that SpySaver eliminates the incentive for spyware companies to collect information. If the counterfeit traffic is similar to the legitimate traffic, then perhaps the aggregate information still provides monetary value to the spyware companies, even if they cannot say for certain whether the information about a specific user is real. There are two reasons why we believe this will not be a problem. First, we can deliberately skew the high-level properties of the aggregate workload that we generate, as long as we do it in an unpredictable manner. Second, much of the value in collecting information about a user's surfing habits lies in the specifics. Examples include correlating activity across sites (i.e. to be able to say that 60% of the users who access site A also access site B) and characterizing users' activities at a specific site (e.g. a sporting goods vendor would like to know which sporting news sites its customers spent the most time browsing in order to better spend its advertising dollars). With widespread deployment of SpySaver the spyware companies would not be able to answer such questions.

In Section 3, we described how SpySaver addresses adware by inflating the tracking counters that the spyware companies use to charge for their services. To counterattack, the spyware companies could attempt to measure the inflation created by SpySaver and discount their per-ad-price accordingly. In practice, discounting will be difficult because (1) it is hard to measure the inflation factor when you cannot distinguish between real and fake users; and (2) the controller will vary the click-through rate in an unpredictable manner, thus increasing the difficulty of setting a discount price.

Our proxy filters out most HTTP accesses to URLs generated by the SpySaver controller, yet it allows all other HTTP accesses to bypass the proxy. Therefore, we must prevent spyware from exploiting this filtration mechanism to detect whether it is running in a fake VM or a real VM.

Although the purpose of our proxy is to vastly reduce the amount of additional network traffic generated by fake users, there is one additional source of network traffic generated by SpySaver that the proxy does not help with. SpySaver must periodically download new browsing profiles used to generate synthetic Web users. To address this concern, we are investigating techniques to represent the profiles in a compact manner.

The controller needs to carefully manage the policy that decides which of the fake VMs it should activate when the machine becomes idle. We need to protect against timing analysis attacks that deduce which VM is real based on the times when it is active. The controller will record the active times distribution for the real user, and deliberately generate similar distributions for its fake users.

To maintain credibility, fake users must perform an array of Web actions, including online purchases and other transactions requiring authentication. Because these Web transactions are encrypted in an end-to-end fashion over SSL, our proxy cache cannot intercept them. Similarly, because our fake users are using forged credentials, the proxy cache cannot forward encrypted fake requests to an origin server. One solution to this problem is to use previously proposed proxy certificates [24]. Another solution is to move the SSL functionality outside all VMs, both the fake ones and the real one. In this way, all traffic leaving the VMs remains unencrypted. The proxy cache will use SSL to encrypt all legitimate traffic that is confidential.

Spyware developers may attempt to fight back against SpySaver by installing it and attempting to generate signatures of the traffic that SpySaver generates. There are a variety of techniques that the central repository can adopt to make online identification of the profiles challenging, such as varying the profiles that are given out on a per-destination basis. Remember that the goal of SpySaver is simply to raise the bar to convince the Spyware companies to find less invasive ways of collecting information about browsing habits.

The final research challenge is how to quantify our tool's effectiveness. There are two open questions that our research must address. First, how much counterfeit information is needed to eliminate spyware? In other words, how widespread must the deployment of SpySaver become before the paying clients of the spyware companies start to question the value of what they're paying for? Second, will the privacy benefits of our approach outweigh the costs of creating the counterfeit information?

7. CONCLUSIONS

This paper uses an idea inspired from economics to address the proliferation of spyware: devaluing the information spyware collects by creating counterfeit information. While information about real users is valuable to the spyware companies, counterfeit information is not. This approach's key advantage lies in its generality; to be effective, we do not have to understand the characteristics of individual spyware programs.

8. REFERENCES

[1] CEXX.ORG. Adware, Spyware, and other unwanted "malware" – and how to remove them, February 2005. `http://www.cexx.org/adware.htm`.

[2] L. D. Cheveallier. Spyware and Network Security. SANS Institute White Paper, August 2001.

[3] D. L. Cook, R. Baratto, and A. D. Keromytis. Remotely Keyed CryptoGraphics - Secure Remote Display Access Using (Mostly) Untrusted Hardware. Technical Report CUCS-050-04, Columbia University, December 2004.

[4] R. S. Cox, J. G. Hansen, S. D. Gribble, and H. M. Levy. A Safety-Oriented Platform for Web Applications. In *Proceedings of the 2006 IEEE Symposium on Security and Privacy*, Oakland, CA, May 2006.

[5] C. Dwork and M. Naor. Pricing via Processing or Combatting Junk Mail. In *Proceedings of the 12th Annual International Cryptology Confernce*, August 1992.

[6] Earthlink. Earthlink Spy Audit, June 2004. `http://www.earthlink.net/spyaudit/press/`.

[7] B. Edelman. WhenU Violates Own Privacy Policy, July 2004. `http://www.benedelman.org/spyware/whenu-privacy/`.

[8] B. Edelman. State Spyware Legislation, February 2005. `http://www.benedelman.org/spyware/legislation/`.

[9] B. Edelman. The Spyware – Click-Fraud Connection – and Yahoo's Role Revisited, April 2006. `http://www.benedelman.org/news/040406-1.html`.

[10] Federal Trade Commission. Monitoring Software on Your PC: Spyware, Adware, and Other Software, April 2004. `http://www.ftc.gov/bcp/workshops/spyware/transcript.pdf`.

[11] Google. Contextual Advertising FAQ, February 2005. `https://adwords.google.com/select/ct_faq.html`.

[12] Google. Google Privacy Center: Privacy Policy, February 2005. `http://www.google.com/privacy.html`.

[13] House of Representatives. SPY Act, February 2005. `http://thomas.loc.gov/cgi-bin/query/z?c109:H.R.29:`.

[14] T. Loder, M. W. V. Alstyne, and R. Wash. An Economic Answer to Unsolicited Communication. In *Proceedings of the 5th ACM Conference on Electronic Commerce*, May 2004.

[15] M. McCardle. How Spyware fits into Defense in Depth. SANS Institute White Paper, January 2003.

[16] C. Metz. Is Google Invading Your Privacy?, February 2003. `http://www.pcmag.com/article2/0,4149,904096,00.asp`.

[17] S. Olsen. Software replaces banner ads on top sites. C|Net News.Com article, August 2001.

[18] PCPitstop. Eight-Seven Percent of WhenU Users are Unware They Are Using It, March 2004. `http://www.pcpitstop.com/spycheck/whenu.asp`.

[19] 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, CA, March 2004.

[20] D. Saurino. Adware and Spyware: A Growing Privacy and Security Problem. SANS Institute White Paper, August 2004.

[21] J. Schartz. "Acquitted man says virus put pornography on computer". New York Times article, August 2003.

[22] M. Shaw and S. D. Gribble. Reverse Firewalls in Denali. In *Work in Progress Presented at the 5th Symposium on Operating Systems Design and Implementation (OSDI)*, Boston, MA, December 2002.

[23] SpywareInfo. Is the Google Toolbar Spyware?, December 2002. `http://www.spywareinfo.com/newsletter/archives/december-2002/12102002.php`.

[24] S. Tuecke, V. Welch, D. Engert, L. Pearlman, and M. Thompson. RFC3820: Internet X.509 Public Key Infrastructure (PKI) Proxy Certificate Profile, June 2004.

[25] Y.-M. Wang, R. Roussev, C. Verbowski, A. Johnson, M.-W. Wu, Y. Huang, and S.-Y. Kuo. Gatekeeper: Monitoring Auto-Start Extensibility Points (ASEPs) for Spyware Management. In *Proceedings of the 18th Large Installation System Administration Conference (LISA)*, Atlanta, GA, November 2004.

[26] WhenU.com. How WhenU Works, February 2005. `http://www.whenu.com/how_whenu_works.html`.

[27] Wired. How Click Fraud Could Swallow the Internet, January 2006. `http://www.wired.com/wired/archive/14.01/fraud.html`.

[28] Zone Labs. Zone Labs: Zone Labs, Internet security products, online safety, software, protection., February 2005. `http://www.zonelabs.com`.