Self-Healing Spyware: Detection, and Remediation

Ming-Wei Wu, *Student Member, IEEE*, Yi-Min Wang, *Member, IEEE*, Sy-Yen Kuo, *Fellow, IEEE*, and Yennun Huang, *Member, IEEE*

***Abstract*—Spyware has become a significant threat to most Internet users as it introduces serious privacy disclosure, and potential security breach to the systems. It has not only utilized critical areas of the computer system to survive reboots, but also grown resilient against current anti-spyware tools; they are capable of self-healing themselves against deletion. Because existing anti-spyware tools are stateless in the sense that they do not remember or monitor the spyware programs that were deleted, they fail to remove self-healing spyware from the system completely. This paper proposes a stateful approach that is based on characterizing spyware invasion as a trust information flow problem, and implements STARS (Stateful Threat-Aware Removal System), which is a tool that at run time monitors critical system behaviors, and ensures that removed spyware programs do not reinstall themselves, to enforce information flow policy in the system. If a reinstallation (self-healing) is detected, STARS infers the source of such activities, and discovers additional "suspicious" programs. Experimental results show that STARS is effective in removing self-healing spyware programs that resist removal by existing anti-spyware tools.**

***Index Terms*—Self-healing, spyware, stateful removal, system security, threat-aware.**

ACRONYM[1]

API	Application Programming Interface
ASEP	Auto-Start Extensibility Points
BHO	Browser Helper Objects
DLL	Dynamic Link Library
EULA	End-User License Agreement
IAT	Import Address Table
LSP	Layered Service Provider
PCM	Policy Configuration Module
PME	Process Monitoring Engine
PUP	Potentially Unwanted Programs
SRE	Stateful Removal Engine
SSM	System Snapshot Manager
STARS	Stateful Threat-Aware Removal System
TLM	Threat-Level Manager
VxD	Virtual Device Driver

Manuscript received January 15, 2007; revised May 1, 2007; accepted June 2, 2007. This work was supported by the National Science Council, Taiwan under Grant NSC 95-2221-E-002-068. Associate Editor: Y. Dai.

M.-W. Wu is with the Department of Electrical Engineering, National Taiwan University, Taipei, Taiwan.

Y.-M. Wang is with Microsoft Research, Redmond, WA 98052 USA.

S.-Y. Kuo is with the Department of Electrical Engineering, National Taiwan University, Taipei, Taiwan, and also with the National Taiwan University of Science and Technology, Taipei, Taiwan (e-mail: sykuo@cc.ee.ntu.edu.tw).

Y. Huang is with the Institute for Information Industry, Taipei, Taiwan.

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TR.2007.909755

[1]The singular and plural of an acronym are always spelled the same.

I. INTRODUCTION

SPYWARE is a new type of potentially unwanted programs (PUP) [1] the goal of which is to monitor users' online behaviors without user consent. Users infected with spyware commonly experience severely degraded reliability and performance such as increased boot time, sluggish feel, and frequent application crashes [2], [3]. Some other types of PUP include adware, malware, backdoor, and Trojan horse. Spyware, and other PUP are often distributed with freeware/shareware, especially peer-to-peer (P2P) file-sharing applications [4], [5], which might state what spyware programs are bundled inside the lengthy EULA (End-User License Agreement) [6]. Others are installed through drive-by downloads following browser vulnerability exploits with little or no user interaction [7]. Ordinary users often do not have sufficient knowledge about spyware [8], and removing spyware is beyond their ability [9].

Such increasing spyware threat prompted the development of a number of commercial anti-spyware products. For example, Lavasoft AdAware [10], Spybot Search & Destroy [11], and Microsoft Windows Defender [12] are popular tools that are able to remove a large number of spyware programs. In practice, it is essential to install multiple anti-spyware tools in order to minimize false negative spyware detection [13].

These anti-spyware tools are primarily based on the signature approach used by anti-virus software, where each program installation is investigated to determine its file, and registry signatures for use by scanner software to detect spyware instances. This approach has two major limitations.

First, the effectiveness relies on the completeness of the signature database for known spyware. Beyond the difficulty of manually locating and cataloging new spyware, this approach is further complicated because spyware are full-fledged applications that are generally much more powerful than the average virus, and can actively take measures to avoid detection & removal [14].

Second, popular spyware removal programs are commonly invoked on-demand, or periodically long after the spyware installation. This allows the spyware to collect private information, and makes it difficult to determine when the spyware got installed, and from where. A monitoring service that keeps track of suspicious system activities since installation time is essential to reduce exposure, and avoid re-infection.

Notably, some of these tools have provided real-time monitoring features [11], [12] that warn users when a program is attempting to make changes to critical areas of a Windows system registry. However, as programs which modify critical registry entries are not necessarily spyware, these real-time monitoring features introduce a high occurrence of false positives, and are likely to be disabled by users to avoid annoying warnings.

Our previous study revealed that quite a few spyware programs had become resilient against deletion, and were capable of restoring removed entries [14]. We refer to this class of spyware programs that recover themselves as self-healing spyware. The primary reason for the ineffectiveness of existing anti-spyware tools against self-healing spyware is because these tools are *stateless* in the sense that they do not remember what spyware were removed between scanning sessions, and cannot recognize if previously identified spyware programs have been completely deleted.

This paper presents a novel approach to spyware detection & remediation that overcomes the limitations of existing anti-spyware solutions. Our approach is based on characterizing spyware invasion as a trust *information flow problem*, which is independent of its binary representation, and therefore can be used to identify previously unknown spyware instances. We use a stateful approach to monitor sensitive program behaviors that may affect the confidentiality, integrity, and availability of the information flow, and hence endanger the trustworthiness, and reliability of the system. Through stateful monitoring, it also enables tracking the completeness of a spyware removal task. In addition, our technique is threat-aware, meaning that its complexity is *adaptive* to the severity of spyware infection. Specifically, it may choose to sacrifice system performance for system dependability when the threat level is elevated.

The remainder of this paper is organized as follows. Section I surveys related work on spyware detection, especially on self-healing ones. Section II describes how a self-healing spyware survives a removal. Section III introduces a stateful threat-aware removal system (STARS) for removing self-healing spyware. Section IV evaluates the effectiveness, and performance of STARS. Section V concludes the paper.

II. Self-Healing Spyware: Striving for Survivability

To complement the signature-based approach, we first characterize spyware invasion as a trust *information flow problem,* and introduce the concept of *Auto-Start Extensibility Points (ASEP)* as the key to spyware detection, and remediation. Our work is based on the observation that, to monitor users' behavior on an ongoing basis, and to maximize the time window for monitoring, an overwhelming majority of spyware programs infect systems in such a way that they are automatically started upon reboot, and at the launch of most commonly used applications. We use the term ASEP to refer to the subset of OS, and application extensibility points that can be *"hooked"* to allow programs to be auto-started without explicit user invocation. An ASEP may accept one or more *ASEP hooks*, each of which is associated with an auto-start program.

We distinguish two types of ASEP hooking: 1) as *a standalone application that is automatically run* by registering as an OS auto-start extension such as a Windows NT service, or a Unix daemon; or 2) as *an extension to an existing application* that is either automatically run (such as WinLogon.exe with its Notify extensions), or popular, commonly run by users (such as the Internet Explorer browser with its Toolbar extensions).

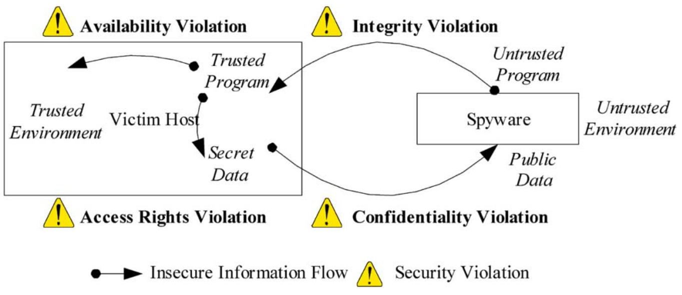


Fig. 1. A model of common spyware invasions.

In addition to ASEP hooking, to ensure its survivability in a system, advanced spyware are likely to conceal itself to avoid discovery, and use redundancy to resist a removal. Next, we give an overview of these advanced techniques employed by self-healing spyware.

A. A Trust Information Flow Problem

The primary objectives of information security systems are to protect confidentiality, integrity, and availability [15]. From our previous work [14], it is obvious that, for spyware, compromises in integrity are the main causes of compromises in confidentiality, and availability. The relationship is illustrated in Fig. 1 in which arrows represent insecure information flow that causes certain security violations. An untrusted program (e.g. drive-by downloads or any spyware program) downloaded from an untrusted environment (e.g. Internet) is installed onto the system resulting in the violation of system integrity. Integrity violation leads to escalation of access rights (e.g. spyware program disguised as a legitimate library to be loaded by the system program), which results in compromises in availability (e.g. spyware consumes CPU utilization, and networking resource), and confidentiality (e.g. cookie, and browsing history are being collected, and sent to unauthorized party). There is clearly a need for a mechanism that specifies insecure information flow (e.g. suspicious program accessing sensitive files), and enforces security policies (e.g. stateful removal of spyware program) within the client host.

We realize, of course, that benign components may also generate sensitive activities; but the depth, and frequency of these activities would differentiate them from spyware components. For example, a benign executable would never hide itself because it will prevent users from being able to click, and execute it. Another example is a benign executable would not recover itself when a user tries to uninstall, or remove it.

The key insight here is that each of the inspected activities on their own does not generally warrant suspicion, but an accumulated amount of critical activity are strong indicators of malicious behavior, as shown in Fig. 2. Therefore, we classify a binary executable as spyware if the component

1) attempts be executed automatically at system reboot by ASEP hooking, and
2) attempts to reinstall or recover removed ASEP entries.

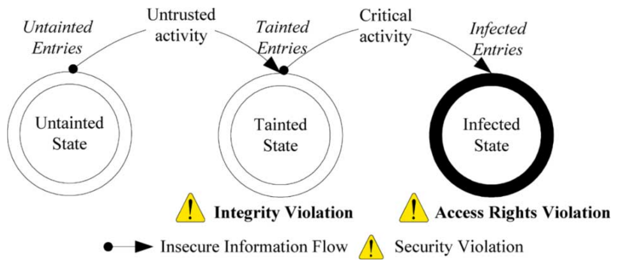


Fig. 2. A security information flow problem.

In the following section, we describe the background information of the monitored sensitive activities in our approach.

B. ASEP Hooking

On Windows platforms, most of the ASEP reside in the Registry. Only a few of them reside in the file system. We have found it useful to classify ASEP into the following categories:

1) **ASEP that start new processes:** for example, the HKLM Registry key, and the %USERPROFILE%Menu file folder are well-known ASEP for auto-starting additional processes.
2) **ASEP that hook system processes:** for example, HKLMNT allows a DLL (Dynamic Linking Library) to be loaded into WinLogon.exe.
3) **ASEP that load drivers:** for example, HKLM{4D36E96B-E325-11CE-BFC1-08002BE10318} allows loading of a keylogger driver; HKLM allows loading of general drivers.
4) **ASEP that hook multiple processes:** for example, HKLM_Catalog9 and NameSpace_Catalog5 allow a Layered Service Provider (LSP) DLL to be loaded into every process that loads networking components; HKLMNT_Dlls allows a DLL to be loaded into every process that links with User32.dll.
5) **Application-specific ASEP:** for example, *HKLM\SOFTWARE\Microsoft\Internet Explorer\Toolbar* allows a toolbar to be loaded into the IE browser; *HKCR\PROTOCOLS\Name-Space Handler,* and *HKCR\PROTOCOLS\Filter* allow other kinds of DLL to be loaded by IE; *HKLM\SOFTWARE\Microsoft\Internet Explorer\Search\SearchAssistant,* and *CustomizeSearch* take URL as input, and control which search pages will be loaded.

Fig. 3 shows the number of spyware hooks to each of the 34 ASEP hooked by at least one of the 120 spyware programs in our Spyware Zoo. Browser Helper Objects (BHO), HKLM "Run" key, and IE "Toolbar" are the three most popular ASEP. We also found that most of the individual spyware programs hook only three or less ASEP, but some hook as many as 13 or 17. When spyware, and freeware programs are bundled together in a single installation, it is not uncommon to see that a single bundle hooks 10 or more ASEP, which would usually cause significant performance degradation. We note that a freeware program may not have any ASEP hook if it is to be manually launched by the user as needed, but spyware programs always have ASEP hooks.

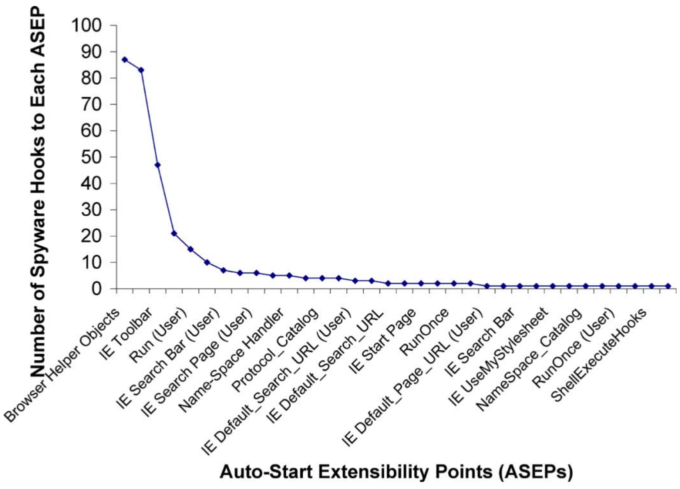


Fig. 3. Distribution of Spyware ASEP Hooks: 120 spyware programs with 334 hooks to 34 ASEP; ASEP are sorted by popularity.

C. Stealth Techniques

1) Spyware Might Be Installed Anywhere in the System: Spyware may not create its home directory in obvious places such as C:Files because it would be too easy for users to discover. Therefore, spyware programs, and data are usually scattered in privileged system directories (e.g. %windows directory%), or temporary directories (e.g. Temporal Internet Files) to bypass a straightforward inspection. Because modifications to information stored in Temporary Internet Files directory may incur system stability issues, many files and directories are managed internally by Internet Explorer. For example, Content.IE5 is not accessible to ordinary users.

2) Spyware Might Intercept System Calls: In some cases, spyware programs can stay stealthy to common system inspection by using rootkit technology [16], such as intercepting, and modifying kernel function ZwQueryDirectoryFile(), which is responsible for querying file, and directory information on Windows 2K/XP; or IFSMgr_InstallFileSystemApiHook(), which is used to install a file system hook inside Windows 98/ME. For examples, the former may be writing a new routine in the DriverEntry() function to replace the old ZwQueryDirectoryFile(), and the latter may be implementing a filter function inside the VxD (Virtual Device Driver) [16]. Similarly, when ZwEnumerateValueKey(), and ZwQueryValueKey() functions in Windows 2K/XP, or _RegEnumValue() in Windows 98/ME are intercepted, registry keys, including sub-keys, and their data, could also be manipulated.

However, no matter how stealthy spyware files are stored in file systems, they still need to be executed to do their work. So, a running spyware must exist in system memory as a process with thread(s), or loaded as modules by other processes. To avoid being seen on the process list, spyware programs need to intercept system calls to stay hidden. On Windows 2K/XP, spyware can intercept ZwQuerySystemInformation() to hide its existence [16]. On Windows 98/ME, a spyware process can call RegisterServiceProcess() of kernel32.dll to hide itself in system services as Windows task manager never lists processes of system services.

3) Identification of Spyware: Randomized Filename: Like the names of people, the file name of a spyware program is an important attribute that has been used to recognize the spyware. However, it is not uncommon to see spyware programs using either partially, or fully randomized filenames for different users on different machines. For example, both Look2Me, and VirtuMonde spyware programs were observed to generate randomized filenames.

4) Spyware Might Be Disguised in Legitimate DLL: While DLL files can be automatically loaded by using Windows system interfaces directly, they can also be loaded by other DLL files, and processes. For example, utility Rundll32.exe can be called to load a DLL file. By default, Rundll32.exe launches the Dllmain() entry of the DLL file, and sends a DLL_PROCESS_ATTACH message to itself. However, a running Rundll32.exe shown in the process list can become very suspicious to experienced users.

Another approach is to replace existing DLL files, especially system DLL files, with spyware-infected DLL files. Using this approach, users may not notice the difference between a good DLL file, and a bad DLL file. Instead of intercepting all functions of a targeted DLL, the replacing DLL forwards the original function calls to the replaced, renamed DLL, and only intercepts selected functions.

There are also spyware programs which use a relatively complicated method called DLL injection. It utilizes remote thread technique to force other processes to load/unload a DLL [18].

D. Redundancy for Robustness

After spyware programs are installed and executed silently, they could still be discovered, and removed. If so, all the intrusion efforts are wasted. Therefore, advanced spyware programs adopt redundant or paired execution & monitoring, and healing of ASEP to ensure survivability.

1) Robust Execution: Paired Processes: 180search Assistant, eXact, BargainBuddy, and IBIS Toolbar programs are examples that form rescue teams of running processes that watch each other; when one process is killed, another one recovers it. We will illustrate the IBIS Toolbar example in the next section. For 16-bit Windows applications, one can check the hPrevInstance pointer of WinMain() for the handler of previous instances; while for 32-bit Windows applications, one can create Mutex objects for inter-processes communication to determine whether the corresponding program is up, and running.

2) Robust Configuration: Setting Recovery: CoolWebSearch is a well-known spyware that uses self-healing Winlogon/Notify ASEP hooks. Windows OS supports the Winlogon Notification Package, which is a DLL that exports functions to handle Winlogon.exe events. These event messages includes lock, unlock, logoff, logon, startup, shutdown, startscreensaver, stopscreensaver, and startshell [19]. Some spyware programs use the Winlogon Notification Package as an alternative to Windows Services because Winlogon/Notify requires much less code to start a program. One simply creates a DLL with specific functions to run as specified in appropriate sub-keys of HKEY_LOCAL_MACHINE. For example, MyDLL.dll may export two functions, i.e. StartAtLogon(), and CloseAtLogoff(), then set appropriate registry entries such as HKLM/../Notify/Dllname/MyDLL.dll, HKLM/../Notify/Logon/StartAtLogon, and HKLM/../Notify/Logon/CloseAtLogoff. MyDLL code can be placed in either function, and is activated when users logon or logoff. A malicious Notify DLL can reside inside WinLogon.exe, and heal its own ASEP entries. Because WinLogon.exe is an essential service that is owned by the system account, it cannot be easily terminated by ordinary users.

III. THE STATEFUL THREAT-AWARE REMOVAL SYSTEM (STARS)

We propose a Stateful Threat-Aware Removal System (STARS) to remove self-healing spyware. By intercepting critical system API, STARS monitors the activities performed by any running process in a system. Some attribute-manipulation activities, such as installing hidden executables, or placing files in a hidden directory, are considered suspicious by STARS; and thus are recorded, and alerted. While existing anti-spyware tools are stateless (memoryless) between spyware removal tasks, STARS is designed to be stateful by monitoring the effectiveness of a spyware removal task over a period of time, setting policies, and detecting any violation of the policies as a sign of having self-healing spyware in the system. To identify any illegal reincarnation of removed entries, STARS performs Soft Rollback, which is a procedure that periodically re-executes the previous removal processes. If there is no reincarnation of suspicious processes, the re-execution would fail. But if a re-execution is successful, a self-healing spyware program would be reincarnated. In this case, STARS learns to remember the source of such reincarnation by adding new policy rules to prevent this reincarnation from happening again in the future. In addition, the source of every reincarnation is considered an instance of the self-healing spyware, and is added into the so called *rescue team* set maintained by STARS. As it is necessary to avoid re-instantiation to defeat self-healing, STARS employs a Soft Termination mechanism to suspend every instances of the rescue team one-by-one so that they are not able to watch the status of each other, and then terminate the whole set of suspended processes all together.

The system architecture of STARS is illustrated in Fig. 4, and the labeled numbers are “step numbers” in the STARS procedure presented in Fig. 5. The Process Monitoring Engine (PME) is responsible for monitoring running processes, and critical API of system DLL. Table I lists the API that STARS intercepts in order to gather information about the activities performed by a running process, especially those that access critical areas of the system registry (ASEP), launch processes, access critical files, and perform network communication. For example, running program calls RegCreateKeyEx (RegCreateKeyExW for Unicode, and RegCreateKeyExA for ANSI opens specified registry key, and creates it if it doesn’t exist. Note that key names are not case sensitive.

PME utilizes Windows API, such as Process32First(), and Process32Next(), to enumerate all on-going processes; and intercepts critical API that are called by these processes. The Registry Monitoring Module, File/Directory Monitoring Module, and Network Monitoring Module of STARS hook critical Windows API calls by employing the Windows IAT (Import Address

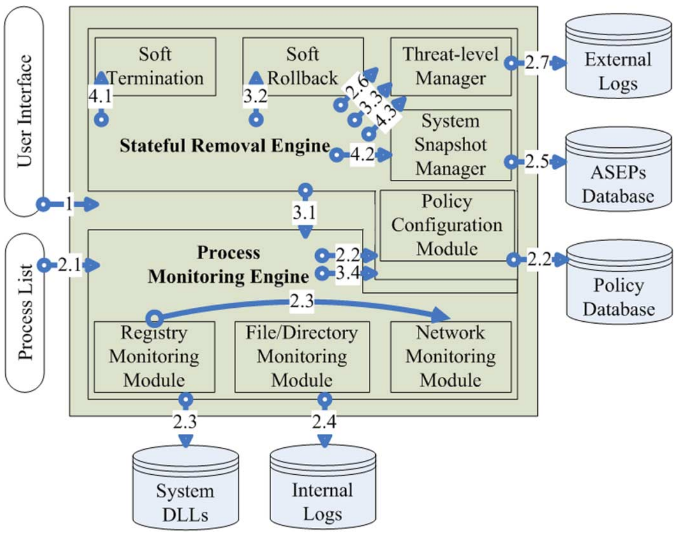


Fig. 4. System architecture of STARS.

Table) redirection mechanism [20], [21]. Functions to be intercepted in a program are imported to the program using the IAT redirection mechanism. In the current implementation of Windows, IAT holds file-relative offsets to the names of imported functions referenced by a program, and calls to the imported functions are routed using the IAT with an indirect JMP instruction. We overwrite a specific IAT entry with the address of our desired routine; PME can gain control before the original function gets a chance to be executed. PME provides real-time monitoring, as well as internal logging of system activities including 1) registry access, 2) file/directory access, and 3) network access performed by each process.

Another core engine, called Stateful Removal Engine (SRE), enforces spyware removal actions according to the existing threat-level, and a pre-defined time constraint. Similar to the System Restore feature of Windows, System Snapshot Manager (SSM) checkpoints ASEP from time to time. It provides the reference information of a set of healthy ASEP so that the spyware-infected ASEP can be detected. Threat-level Manager (TLM) is able to adjust the spyware threat-level by parsing external logs created by third-party anti-spyware tools. Whenever a spyware threat level is raised, SRE performs Soft Rollback to re-execute the spyware removal procedure from time to time to make sure that the removed entries remain removed. However, if Soft Rollback discovered some removed ASEP entries are being reinserted by a suspicious process, it hands over the process for Soft Termination, and enables a policy rule to auto-reject subsequent activities issued by the suspicious process. In other words, STARS can monitor the source of policy violations, and insert rules to stop future self-healing of spyware programs. Policy rules are maintained by the Policy Configuration Module (PCM), which stores pre-defined, customized rules in the Policy Database so that PME is able to respond to any spyware event with appropriate reactions. For example, one may configure STARS to reject one set of spyware processes, but accept another set of spyware processes to log their activities.

A. *STARS Detection and Recovery Procedure*

The STARS procedure is described in Fig. 5. Generally speaking, the low threat level indicates no spyware has been found recently; and STARS simply performs process monitoring, and API interception, which consumes very minimal system resources. In addition to customized policy database, STARS also leverages existing anti-spyware tools by monitoring their external logs to detect the existence, and removal of known spyware programs, which raises the threat level from low to medium. When STARS is in the medium threat level, not only all system activities are logged but also Soft Rollback is applied from time to time, which may incur more performance overhead. However, the longer it stays in the medium threat state, the better chance that STARS can detect a self-healing spyware with its rescue team, and insert new policy rules to block future reincarnation of the spyware. Therefore, the duration for STARS to stay in the medium threat level after a spyware is detected is a tradeoff between security, and performance. Our experience shows that most self-healing spyware processes today reincarnate themselves immediately after a spyware removal procedure is complete. If a self-healing spyware in a system is detected, STARS would elevate its state to the high threat level. Then it launches Soft Termination to remove all self-healing spyware processes from the system immediately, and restores ASEP entries from the previous ASEP checkpoint. Apparently this is a simple rollback, and may accidentally remove good ASEP installed by legitimate applications since the time of spyware installation. We have addressed this issue with bundled tracing (determining which ASEP hooks belong to the same package), and ASEP recovery that attempt to restore infected ASEP entries without over-killing good ones [14].

IV. Experimental Results

To evaluate the effectiveness and performance of STARS, we have selected several real-world spyware programs for a demonstration at each threat-level.

A. *Threat-Level Is Low*

While STARS enumerates all running processes, it intercepts critical API of system DLL rather than all system calls, and consumes minimal system resources (with CPU time less than 1 percent, and memory usage of 2,200KB) at the low threat level.

Although STARS does not have a large database of signatures to identify known spyware as most commercial anti-spyware tools do, it has powerful API interception to trace critical activities taken by running programs. When surfing on the web, Internet users may be prompted to download a file, but the malicious website actually pushes two files. The trick seems incredibly simple, but it works. Fig. 6 is the log of STARS, and it shows that Internet Explorer (iexplore.exe) downloaded both a wanted file (e.g. Norton_Internet_Security_v2005.zip), and an unwanted file (e.g. activate_crack.exe). When a user carelessly clicked the wrong file, i.e. activate_crack.exe (at timestamp 03:35:57), a BHO plug-in was installed without the user's consent. In this case, STARS later recognizes that activate_crack.exe is malicious because it suspiciously inserts

Step 1. **Initialization**: STARS is launched, Stateful Removal Engine and Process Monitoring Engine are activated.

Step 2. **When spyware threat-level is low:**

2.1 **Process enumeration**: Enumerate all running processes and applies default rules accordingly.

2.2 **Policy configuration**: With Policy Configuration Module, one can add new rules or customize existing rules to instruct STARS to either accept or reject a given request. All policy rules are maintained in the Policy Database.

2.3 **API monitoring**: Registry Monitoring Module, File/Directory Monitoring Module, and Network Monitoring Module hooked critical APIs of system DLLs to control their calls according to the defined policy rules.

2.4 **Activity logging**: The subsequent activities taken by each process can be logged in Internal Logs.

2.5 **Snapshot maintenance**: System Snapshot Manager checkpoints ASEP periodically for future rollback and recovery.

2.6 **Process monitoring**: User may occasionally run some third-party anti-spyware tools (e.g. Ad-Aware SE Personal 1.06) and Threat-level Manager is informed of its presence (e.g. Ad-Aware.exe).

2.7 **Log monitoring**: Parse the External Logs of the anti-spyware tools (e.g. C:\Documents and Settings\[User]\Application Data\Lavasoft\Ad-Aware\Logs). If evidence of spyware threat is found, the threat-level is raised from low to medium and STARS jumps to step 3.1.

Step 3. **When spyware threat-level is medium**

3.1 **Removed ASEP entry monitoring**: As threat-level is changed to medium, SRE instructs PME to log all system activities and identify any action that violates existing policy rules – i.e. a recreation of the removed ASEP entries which indicates the existence of self-healing spyware. If a violation is detected, go to step 3.4. If not, continue to step 3.2.

3.2 **Soft Rollback**: In addition to ASEP monitoring, STARS redoes spyware removals periodically. If there is no reincarnation of suspicious processes, the redo would fail. But if a redo is successful, self-healing spyware process is reincarnated and STARS will learn to remember the source of such reincarnation by adding new policy rules to auto-reject them next time. In this case, the threat-level is raised to high and the STARS jumps to step 3.4.

3.3 **Returning to low threat level**: If no suspicious activities are detected within a certain interval (e.g. 10 minutes), Threat-level Manager reduces the threat-level to low and STARS go to step 2. Otherwise, go to step 3.1

3.4 **Identifying suspicious processes**: With the occurrence of policy violation, STARS identifies the self-healing spyware (and members of the rescue team) and learns to add new policy rules to auto-reject them next time. Threat-level is raised to high and STARS go to step 4.1.

Step 4. **When spyware threat-level is high**

4.1 **Soft Termination**: STARS employs Soft Termination to terminate the detected self-healing processes and kill their rescue team.

4.2 **ASEP rollback**: STARS removes ASEP entries that do not exist in the previous ASEP checkpoint.

4.3 **Returning to medium threat level**: STARS reduces the threat-level to medium and go to Step 3.1.

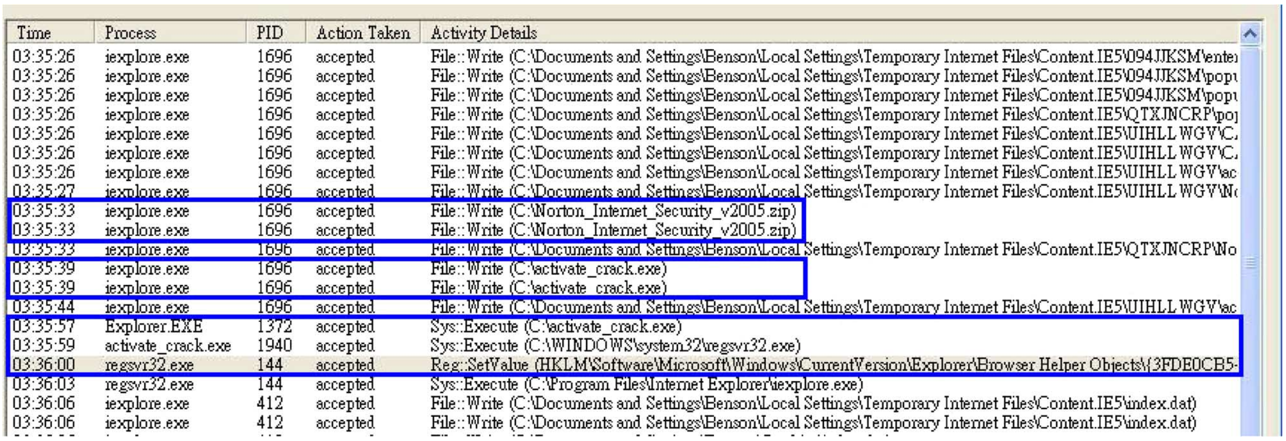
Fig. 5. The STARS procedure.

TABLE I
CRITICAL SYSTEM API CALLS INTERCEPTED BY STARS

Registry Monitoring	**Process Monitoring**
RegCreateKeyExA()	CreateProcessA()
RegCreateKeyExW()	CreateProcessW()
RegOpenKeyExA()	**File Monitoring**
RegOpenKeyExW()	CreateFileW()
RegSetValueExA()	DeleteFileW()
RegSetValueExW()	MoveFileWithProgressW()
RegCloseKey()	**Network Monitoring**
	Connect()
	Listen()

a BHO entry (at timestamp 03:36:00), and invokes hidden Internet Explorer (at timestamp 03:36:03) to access sensitive files (at timestamp 03:36:06) such as user cookies, and browsing history (i.e. index.dat file of Content.IE5 directory is pre-defined by STARS belonging to the set of sensitive files). Because Internet Explorer iexplore.exe is indirectly invoked by a suspicious program (activate_crack.exe) not belonging to the pre-defined white-list (list of trusted program) maintained by our system, and utilizes IE to access sensitive files, STARS immediately rejected these malicious accesses, and learned that activate_crack.exe is malicious.

With STARS, users are able to identify suspicious behavior during the installation stage, especially those trying to install themselves in hidden directories. As mentioned earlier, self-healing spyware programs often manipulate system attributes to protect their existence. DashBar is an adware-supported search toolbar from the GAIN Network that would change the browser settings to make DashBar the default search engine. STARS found that, in addition to using the standard Program Files directory, Dashbar also launches another suspicious executable



```
Time      Process            PID   Action Taken  Activity Details
03:35:26  iexplore.exe       1696  accepted      File::Write (C:\Documents and Settings\Benson\Local Settings\Temporary Internet Files\Content.IE5\094JJKSM\ente
03:35:26  iexplore.exe       1696  accepted      File::Write (C:\Documents and Settings\Benson\Local Settings\Temporary Internet Files\Content.IE5\094JJKSM\popu
03:35:26  iexplore.exe       1696  accepted      File::Write (C:\Documents and Settings\Benson\Local Settings\Temporary Internet Files\Content.IE5\094JJKSM\popu
03:35:26  iexplore.exe       1696  accepted      File::Write (C:\Documents and Settings\Benson\Local Settings\Temporary Internet Files\Content.IE5\QTXJNCRP\po
03:35:26  iexplore.exe       1696  accepted      File::Write (C:\Documents and Settings\Benson\Local Settings\Temporary Internet Files\Content.IE5\UIHLLWGV\C.
03:35:26  iexplore.exe       1696  accepted      File::Write (C:\Documents and Settings\Benson\Local Settings\Temporary Internet Files\Content.IE5\UIHLLWGV\C.
03:35:26  iexplore.exe       1696  accepted      File::Write (C:\Documents and Settings\Benson\Local Settings\Temporary Internet Files\Content.IE5\UIHLLWGV\ac
03:35:27  iexplore.exe       1696  accepted      File::Write (C:\Documents and Settings\Benson\Local Settings\Temporary Internet Files\Content.IE5\UIHLLWGV\N
03:35:33  iexplore.exe       1696  accepted      File::Write (C:\Norton_Internet_Security_v2005.zip)
03:35:33  iexplore.exe       1696  accepted      File::Write (C:\Norton_Internet_Security_v2005.zip)
03:35:33  iexplore.exe       1696  accepted      File::Write (C:\Documents and Settings\Benson\Local Settings\Temporary Internet Files\Content.IE5\QTXJNCRP\No
03:35:39  iexplore.exe       1696  accepted      File::Write (C:\activate_crack.exe)
03:35:39  iexplore.exe       1696  accepted      File::Write (C:\activate_crack.exe)
03:35:44  iexplore.exe       1696  accepted      File::Write (C:\Documents and Settings\Benson\Local Settings\Temporary Internet Files\Content.IE5\UIHLLWGV\ac
03:35:57  Explorer.EXE       1372  accepted      Sys::Execute (C:\activate_crack.exe)
03:35:59  activate_crack.exe 1940  accepted      Sys::Execute (C:\WINDOWS\system32\regsvr32.exe)
03:36:00  regsvr32.exe       144   accepted      Reg::SetValue (HKLM\Software\Microsoft\Windows\CurrentVersion\Explorer\Browser Helper Objects\{3FDE0CB5-
03:36:03  regsvr32.exe       144   accepted      Sys::Execute (C:\Program Files\Internet Explorer\iexplore.exe)
03:36:06  iexplore.exe       412   accepted      File::Write (C:\Documents and Settings\Benson\Local Settings\Temporary Internet Files\Content.IE5\index.dat)
03:36:06  iexplore.exe       412   accepted      File::Write (C:\Documents and Settings\Benson\Local Settings\Temporary Internet Files\Content.IE5\index.dat)
```

Fig. 6. Example of drive-by download, BHO installation and unauthorized access to sensitive files.

called trickler4104.exe, which uses a hidden directory C:and Settings[user] Settings_tmp as its home directory to maintain its profile settings. We believe that installing executables in hidden directories without user consent is considered malicious, and STARS has a default policy to prohibit such activities.

B. Threat-Level Is Medium

STARS is aware of third-party anti-spyware tools, and automatically parses their external logs when changes are detected. If evidence of spyware threat is found, the threat-level is raised from low to medium. At the medium threat-level, STARS is responsible for enforcing the spyware removal policy, and monitoring any policy violation. With the extra overhead for monitoring processes and policies, STARS at the medium threat-level has an average CPU utilization of 2% to 3%, and memory consumption of 3,000 KB.

In the Zango Search spyware case, when exercising Soft Rollback to re-execute a previous spyware removal, STARS learns that a removed entry in HKLM/../Run key is immediately recovered by MediaGateway.exe (Zango Search), and thus inserts a new policy rule to reject such modification in the future. In another example, STARS recognizes that istsvc.exe (IST Bar) continuously sets the registry HKLM/../Run key (once per second). Based on our experience, excessive modifications (one access per 1$\sim$5 seconds) of an identical registry entry often implies the presence of self-healing spyware. STARS considers such an aggressive registry access as a malicious behavior. Both Zango Search, and IST Bar are examples of self-healing spyware that uses executable programs to recover their ASEP entries, which violates STARS policies, and thus the threat-level is elevated to high (self-healing spyware presents).

C. Threat-Level Is High

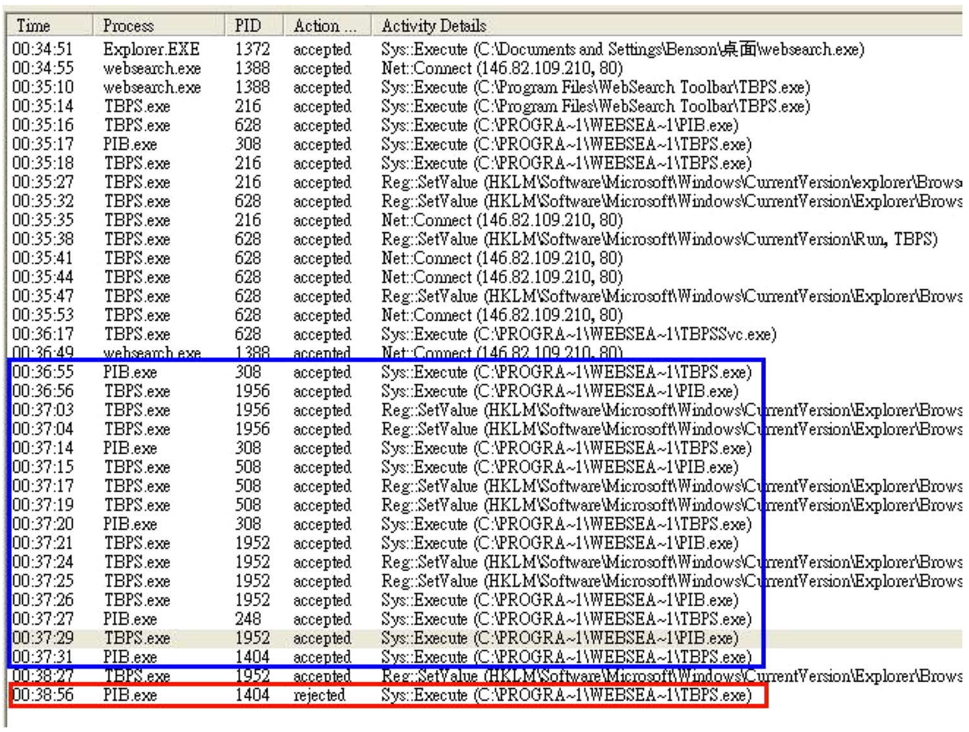
Because STARS can detect the sources of policy violations, it can identify the root process of a spyware program, and launch Soft Termination to terminate the root, and all its child processes together. In this section, we demonstrate the effectiveness of STARS with a spyware called the WebSearch Toolbar. Using STARS, we find that, in addition to the WebSearch Toolbar, the package installs several other executables that are self-healing such as TBPS.exe, TBPSSvc.exe, and PIB.exe. They not only set ASEP to ensure an auto-execution at each system reboot, but also help each other for their survivability. In our experiments, when a user discovers the existence of spyware using taskmgr.exe, and tries to terminate these processes one by one with Task Manager, the surviving processes would quickly bring up the killed ones. As a result, existing anti-spyware tools cannot terminate the WebSearch spyware successfully.

Because members of the rescue team, namely TBPS.exe, TBPSSvc.exe, and PIB.exe, repeatedly violate the predefined policies, STARS first identifies the source of policy violations, and recognizes members of the spyware rescue team. Then STARS inserts new policy rules to reject future reincarnation of these processes, and runs Soft Termination to kill these processes all together so that no rescue team can recover the removed processes, and their ASEP entries.

Fig. 7 shows a snapshot of STARS in action. While a user had been killing spyware programs manually with Task Manager (taskmgr.exe) repeatedly because they kept recovering after being terminated (from timestamp 00:36:55 to 00:38:27), STARS recognized that PIB.exe, and TBPS.exe had been launching each other excessively, so they must be members of the rescue team. Hence, STARS learned to reject, and kill paired processes at timestamp 00:38:56. As a result, STARS effectively removed malicious running processes, and stopped the reinsertion of spyware entries into ASEP.

Our experiments on using STARS against today's spyware suggest that a stateful approach towards detecting & removing self-healing spyware is feasible. First, the violation of confidentiality is a sign of spyware intrusion. Areas that stored sensitive profile, and valuable information should be monitored consistently, and any arbitrary access to these areas should be carefully examined against defined security policy. In our first example (when threat level is low), even the spyware can be stealthy, and get installed without user consent; it got caught by STARS when trying to access sensitive data on the system. Apparently, false positives occur when any legitimate process not on the white-list is trying to access certain sensitive data; in these situations STARS would prompt the user with the call history of that process, and help make the decision to permit or deny it.

Second, the violation of integrity is another sign of spyware intrusion, and modifications made to critical areas of the system



Time	Process	PID	Action ...	Activity Details
00:34:51	Explorer.EXE	1372	accepted	Sys::Execute (C:\Documents and Settings\Benson\桌面\websearch.exe)
00:34:55	websearch.exe	1388	accepted	Net::Connect (146.82.109.210, 80)
00:35:10	websearch.exe	1388	accepted	Sys::Execute (C:\Program Files\WebSearch Toolbar\TBPS.exe)
00:35:14	TBPS.exe	216	accepted	Sys::Execute (C:\Program Files\WebSearch Toolbar\TBPS.exe)
00:35:16	TBPS.exe	628	accepted	Sys::Execute (C:\PROGRA~1\WEBSEA~1\PIB.exe)
00:35:17	PIB.exe	308	accepted	Sys::Execute (C:\PROGRA~1\WEBSEA~1\TBPS.exe)
00:35:18	TBPS.exe	216	accepted	Sys::Execute (C:\PROGRA~1\WEBSEA~1\TBPS.exe)
00:35:27	TBPS.exe	216	accepted	Reg::SetValue (HKLM\Software\Microsoft\Windows\CurrentVersion\explorer\Brows
00:35:32	TBPS.exe	628	accepted	Reg::SetValue (HKLM\Software\Microsoft\Windows\CurrentVersion\Explorer\Brows
00:35:35	TBPS.exe	216	accepted	Net::Connect (146.82.109.210, 80)
00:35:38	TBPS.exe	628	accepted	Reg::SetValue (HKLM\Software\Microsoft\Windows\CurrentVersion\Run, TBPS)
00:35:41	TBPS.exe	628	accepted	Net::Connect (146.82.109.210, 80)
00:35:44	TBPS.exe	628	accepted	Net::Connect (146.82.109.210, 80)
00:35:47	TBPS.exe	628	accepted	Reg::SetValue (HKLM\Software\Microsoft\Windows\CurrentVersion\Explorer\Brows
00:35:53	TBPS.exe	628	accepted	Net::Connect (146.82.109.210, 80)
00:36:17	TBPS.exe	628	accepted	Sys::Execute (C:\PROGRA~1\WEBSEA~1\TBPSSvc.exe)
00:36:49	websearch.exe	1388	accepted	Net::Connect (146.82.109.210, 80)
00:36:55	PIB.exe	308	accepted	Sys::Execute (C:\PROGRA~1\WEBSEA~1\TBPS.exe)
00:36:56	TBPS.exe	1956	accepted	Sys::Execute (C:\PROGRA~1\WEBSEA~1\PIB.exe)
00:37:03	TBPS.exe	1956	accepted	Reg::SetValue (HKLM\Software\Microsoft\Windows\CurrentVersion\Explorer\Brows
00:37:04	TBPS.exe	1956	accepted	Reg::SetValue (HKLM\Software\Microsoft\Windows\CurrentVersion\Explorer\Brows
00:37:14	PIB.exe	308	accepted	Sys::Execute (C:\PROGRA~1\WEBSEA~1\TBPS.exe)
00:37:15	TBPS.exe	508	accepted	Sys::Execute (C:\PROGRA~1\WEBSEA~1\PIB.exe)
00:37:17	TBPS.exe	508	accepted	Reg::SetValue (HKLM\Software\Microsoft\Windows\CurrentVersion\Explorer\Brows
00:37:19	TBPS.exe	508	accepted	Reg::SetValue (HKLM\Software\Microsoft\Windows\CurrentVersion\Explorer\Brows
00:37:20	PIB.exe	308	accepted	Sys::Execute (C:\PROGRA~1\WEBSEA~1\TBPS.exe)
00:37:21	TBPS.exe	1952	accepted	Sys::Execute (C:\PROGRA~1\WEBSEA~1\PIB.exe)
00:37:24	TBPS.exe	1952	accepted	Reg::SetValue (HKLM\Software\Microsoft\Windows\CurrentVersion\Explorer\Brows
00:37:25	TBPS.exe	1952	accepted	Reg::SetValue (HKLM\Software\Microsoft\Windows\CurrentVersion\Explorer\Brows
00:37:26	TBPS.exe	1952	accepted	Sys::Execute (C:\PROGRA~1\WEBSEA~1\PIB.exe)
00:37:27	PIB.exe	248	accepted	Sys::Execute (C:\PROGRA~1\WEBSEA~1\TBPS.exe)
00:37:29	TBPS.exe	1952	accepted	Sys::Execute (C:\PROGRA~1\WEBSEA~1\PIB.exe)
00:37:31	PIB.exe	1404	accepted	Sys::Execute (C:\PROGRA~1\WEBSEA~1\TBPS.exe)
00:38:27	TBPS.exe	1952	accepted	Reg::SetValue (HKLM\Software\Microsoft\Windows\CurrentVersion\Explorer\Brows
00:38:56	PIB.exe	1404	rejected	Sys::Execute (C:\PROGRA~1\WEBSEA~1\TBPS.exe)

Fig. 7. STARS learned to reject paired self-healing processes.

(e.g. ASEP) should also be monitored. In our second example (when the threat-level is medium), advanced spyware is so stubborn that it repeatedly resets the ASEP to survive registry deletion made by the anti-spyware tools. However, such aggressive access to the same set of ASEP implies the suspicious program is trying to compromise system integrity, and STARS would insert a new rule to block subsequent tries from that program.

In our last example (when the threat-level is high), STARS is no longer dealing with a single spyware program at a time, but a group of spyware programs that rescue each other. They not only break system integrity, but also greatly reduce the availability of the resources on the system because they consume unnecessary CPU utilization, memory usage, and file I/O. Note that we did not find any legitimate software that would form a rescue team to recover terminated peers, except the Windows Security Center (wscntfy.exe) introduced in Service Pack 2, and a few others. These programs are very common in every system, and are included in the white-list of STARS to reduce false positives.

As the present implementation of STARS heavily relied on the Windows IAT (Important Address Table) technique to gain control before the original function gets a chance to be executed, it is possible that there are spyware instances (e.g. those that integrate rootkit functionalities) that may try to disable or bypass STARS, and lead to the occurrence of false negatives. In our future work, we would implement STARS as a system driver to gain more control in the kernel.

V. Conclusion

Spyware threats are prevalent as they are often bundled in freeware, shareware, or add-on plug-ins. Without a careful examination, computer users can hardly detect if a software package contains spyware. Although existing anti-spyware tools are effective in identifying & removing known spyware using signatures, our study shows that some self-healing spyware are immune to these signature-based anti-spyware tools. This is due to the fact that existing anti-spyware tools are stateless (memoryless) in the spyware removal process. That is, once a removal is done, they do not monitor & identify the recurrence of spyware entries, and thus self-healing spyware programs are able to reincarnate themselves. We described a Stateful Threat-Aware Removal System (STARS) that can 1) keep track of critical activities performed by running processes; 2) follow up on checking the effectiveness of a spyware removal task over time; and 3) trade off system dependability, and system performance depending on the severity of the spyware threat in a system. Experimental results demonstrated that STARS is effective in detecting & removing self-healing spyware that existing commercial anti-spyware tools fail to do.

Although we had designed STARS to be adaptive to the severity of spyware threats, and can trade off system performance for system dependability, the performance overhead for removing self-healing spyware is small. However, there are rooms for improving the comprehensiveness of STARS because not all attributes manipulation discussed in Section I can be automatically detected by STARS. In its current implementation, STARS cannot detect hidden Registry entries, and DLL injection. The difficulties are that 1) Registry entries are relatively more complicated to maintain than files/directories, and 2) it requires remote thread monitoring to identify DLL injection. These features will introduce additional performance overhead to STARS.

The battles between spyware, and anti-spyware programs will never end; the former evolves against security measures to evade detection, while the latter extensively monitors, and filters known, and suspicious activities. We see similar confrontation in other security threats such as malware, backdoor, Trojan horse, and rootkit. We are currently working on applying STARS to these security problem domains.

REFERENCES

[1] P. McFedries, "Technically speaking: The spyware nightmare," *IEEE Spectrum*, vol. 42, no. 8, p. 72, 2005.

[2] M. B. Schmidt and K. P. Arnett, "Spyware: A little knowledge is a wonderful thing," *Communications of the ACM*, vol. 48, no. 8, pp. 67–70, 2005, New York.

[3] "Battling 'spyware': Debate intensifies on controlling deceptive programs," Microsoft, April 20, 2004 [Online]. Available: http://www.microsoft.com/presspass/features/2004/apr04/04-20Spyware.mspx

[4] G. Lawton, "Invasive software: Who's inside your computer," *IEEE Computer*, vol. 35, no. 7, pp. 15–18, 2002.

[5] S. Saroiu, S. D. Gribble, and H. M. Levy, "Measurement and analysis of spyware in a university environment," in *Proceedings of the First Symposium on Networked Systems Design and Implementation (NSDI '04)*, 2004.

[6] W. Aaron, "Spyware be gone," *NetWorker*, vol. 9, no. 1, p. 18, 2005, New York.

[7] Y.-M. Wang, D. Beck, X. Jiang, R. Roussev, C. Verbowski, S. Chen, and S. King, "Automated web patrol with strider honeyMonkeys: Finding web sites that exploit browser vulnerabilities," in *Proceedings of the Network and Distributed System Security (NDSS) Symposium*, February 2006 [Online]. Available: http://research.microsoft.com/HoneyMonkey/

[8] Q. Hu and T. Dinev, "Is spyware an internet nuisance or public menace," *Communications of the ACM*, vol. 48, no. 8, pp. 61–66, 2005, New York.

[9] W. Ames, "Understanding spyware: Risk and response," *IEEE IT Professional*, vol. 6, no. 5, pp. 25–29, 2004.

[10] Lavasoft Ad-Aware [Online]. Available: http://www.lavasoftusa.com/software/adaware/

[11] Spybot Search & Destroy [Online]. Available: http://www.safer-networking.org/

[12] Microsoft Windows Defender [Online]. Available: http://www.microsoft.com/athome/security/spyware/software/default.mspx

[13] W. Harrison and T. Bollinger, "User confidence—and the software developer," *IEEE Software*, vol. 21, no. 6, pp. 5–8, 2004.

[14] Y. M. Wang, R. Roussev, C. Verbowski, A. Johnson, M. W. Wu, Y. N. Huang, and S. Y. Kuo, "Gatekeeper: Monitoring auto-start extensibility points (ASEPs) for spyware management," in *Proceedings of the Usenix 18th Large Installation System Administrations (LISA'04)*, 2004.

[15] R. S. Sandhu, "Lattice-based access control models," *IEEE Computer*, vol. 26, no. 11, pp. 9–19, 1993.

[16] Y. Kaplan, "API spying techniques for Windows 9x, NT and 2000," [Online]. Available: http://www.internals.com/articles/apispy/apispy.htm

[17] Y. M. Wang, D. Beck, B. Vo, R. Roussev, and C. Verbowski, "Detecting stealth software with strider GhostBuster," in *Proceedings of the International. Conference on Dependable Systems and Networks (DSN-DCCS)*, 2005, pp. 368–377.

[18] DLL Injection Tutorial [Online]. Available: http://www.codeproject.com/dll/DLL_Injection_tutorial.asp

[19] T. Truong, "Taking advantage of the winlogon notification package," *The Code Project,* Jan. 2001 [Online]. Available: http://www.codeproject.com/system/winlogon_notification_package.asp

[20] J. Richter, *Advanced Windows*, 3rd Bk&Cdr ed. : Microsoft Press, February 1997, ISBN: 1572315482.

[21] O. Zaytsev, *Rootkits, Spyware/Adware, Keyloggers and Backdoors: Detection and Neutralization*. : A-List Publishing, September 1, 2006, ISBN: 1931769591.

Ming-Wei Wu is a Product Manager, and Security Consultant at Armorize Technologies, where he manages an R&D team responsible for automated static analysis on Web codes. Previously, he worked for the National Information & Communication Security Taskforce, Executive Yuan, Taiwan as an associate researcher to plan government security policy. He received his MS degree from National Chiao Tung University, Taiwan in 2003, and has been a PhD candidate in Electrical Engineering at National Taiwan University, Taiwan since 2004. His research interests include Web application security, intrusion tolerance, and P2P networking. He is a student member of the IEEE.

Yi-Min Wang is a Director and Principal Researcher at the Internet Services Research Center (ISRC), Microsoft Research-Redmond, where he leads an R&D organization responsible for systems, infrastructure, cyber-intelligence, and search quality. Yi-Min received his B.S. degree from National Taiwan University in 1986. He received his Ph.D. in Electrical and Computer Engineering from University of Illinois at Urbana-Champaign in 1993, worked at AT&T Bell Labs from 1993 to 1997, and joined Microsoft in 1998. His research interests include security, systems management, dependability, home networking, and distributed systems. He is a member of the IEEE.

Sy-Yen Kuo is a Chair Professor, and Dean of the College of Electrical and Computer Engineering, National Taiwan University of Science and Technology, Taipei, Taiwan. He is also a Distinguished Professor at the Department of Electrical Engineering at National Taiwan University where he is currently taking a leave of absence, and was the Chairman at the same department from 2001 to 2004. He received the BS (1979) in Electrical Engineering from National Taiwan University, the MS (1982) in Electrical & Computer Engineering from the University of California at Santa Barbara, and the PhD (1987) in Computer Science from the University of Illinois at Urbana-Champaign. He spent his sabbatical years as a Visiting Professor at the Computer Science and Engineering Department, the Chinese University of Hong Kong from 2004–2005, and as a visiting researcher at AT&T Labs-Research, New Jersey from 1999 to 2000, respectively. He was the Chairman of the Department of Computer Science and Information Engineering, National Dong Hwa University, Taiwan from 1995 to 1998, a faculty member in the Department of Electrical and Computer Engineering at the University of Arizona from 1988 to 1991, and an engineer at Fairchild Semiconductor and Silvar-Lisco, both in California, from 1982 to 1984. In 1989, he also worked as a summer faculty fellow at Jet Propulsion Laboratory of California Institute of Technology. His current research interests include dependable systems and networks, software reliability engineering, mobile computing, and reliable sensor networks.

Professor Kuo is an IEEE Fellow. He has published more than 270 papers in journals and conferences, and also holds several patents. He received the distinguished research award between 1997 and 2005 consecutively from the National Science Council in Taiwan, and is now a Research Fellow there. He was also a recipient of the Best Paper Award in the 1996 International Symposium on Software Reliability Engineering, the Best Paper Award in the simulation and test category at the 1986 IEEE/ACM Design Automation Conference (DAC), the National Science Foundation's Research Initiation Award in 1989, and the IEEE/ACM Design Automation Scholarship in 1990 and 1991.

Yennun Huang received his BS in EE from National Taiwan University in 1982; and MS, and PhD in CS from the University of Maryland. He worked for AT&T/Lucent Bell Labs for 12 years, and started the Dependable Computing Research Program at AT&T. He left AT&T in 2001 to become the VP of Engineering of a startup company in 2001. Yennun Huang was a Visiting Research Fellow in National Taiwan University in 2004. He returned to AT&T Labs in late 2004 and was the Executive Director for Dependable Distributed Computing and Communication Research Department. His work was to improve the dependability and quality of emerging services such as mobile, digital home, and IPTV for AT&T. He is currently an Executive Vice President of Institute for Information Industry, Taiwan. His current research interests are dependable computing, P2P computing, mobile computing, middleware platforms, IPTV, and digital home applications. He is a member of the IEEE.