Gatekeeper: Monitoring Auto-Start Extensibility Points (ASEPs) for Spyware Management

Yi-Min Wang, Roussi Roussev, Chad Verbowski, Aaron Johnson – Microsoft Research
Ming-Wei Wu, Yennun Huang, and Sy-Yen Kuo – National Taiwan University

ABSTRACT

Spyware is a rapidly spreading problem for PC users causing significant impact on system stability and privacy concerns. It attaches to extensibility points in the system to ensure the spyware will be instantiated when the system starts. Users may willingly install free versions of software containing spyware as an alternative to paying for it. Traditional anti-virus techniques are less effective in this scenario because they lack the context to decide if the spyware should be removed.

In this paper, we introduce Auto-Start Extensibility Points (ASEPs) as the key concept for modeling the spyware problem. By monitoring and grouping ''hooking'' operations made to the ASEPs, our Gatekeeper solution complements the traditional signature-based approach and provides a comprehensive framework for spyware management. We present ASEP hooking statistics for 120 real-world spyware programs. We also describe several techniques for discovering new ASEPs to further enhance the effectiveness of our solution.

Introduction

Spyware is a generic term referring to a class of software programs that track and report computer users' behavior for marketing or illegal purposes. In addition, spyware may actively push advertisements to the user by popping up windows, and change the Web browser start page, search page, and bookmark settings. Spyware often silently communicates with servers over the Internet to report collected user information, and may also receive commands to install additional software on the user's machine. Users infected with spyware commonly experience severely degraded reliability and performance such as increased boot time, sluggish feel, and frequent application crashes. Reliability data shows that spyware programs account for fifty percent of the overall crash reports [FTC04]. Saroiu, et al. [SGL04] point out security problems caused by vulnerabilities in spyware programs. A recent study based on scanning more than one million machines show the alarming prevalence of spyware: an average of four to five spyware programs (excluding Web browser cookies) were running on each computer [E04A, E04B].

Current anti-spyware solutions [AA, SB] are primarily based on the signature approach used by anti-virus software: each spyware installation is investigated to determine its file and Registry signatures for use by scanner software to detect spyware instances. This approach has several problems.

First, many spyware programs may be considered ''legitimate'' in the following sense: their companies sponsor popular freeware to leverage their installation base; since users agree to an End User Licensing Agreement (EULA) when they install freeware, removing the bundled spyware may violate this agreement. In many cases, the freeware ensures the spyware is running on the user's system by refusing to run if its bundled spyware is removed.

Second, the effectiveness relies on 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 [C04], and can actively take measures to avoid detection and removal. Companies creating spyware generate revenue based on the prevalence of their applications and therefore have a financial incentive to create technologies that make it hard to detect and remove their software. They have the need and the resources to invest in developing sophisticated morphing behavior.

Third, some spyware installations may contain common library files that non-spyware applications use. If care is not taken to remove these files from the spyware signatures, scanners using these signatures may break non-spyware applications.

Finally, 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 was installed and where it came from. A monitoring service that catches spyware at installation time is essential for reducing exposure and avoiding re-infection.

To complement the signature-based approach, we introduce the concept of *Auto-Start Extensibility Points* (ASEPs) as the key to spyware management. Our work

is based on the observation that, in order 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 the launch of most commonly used applications. We use the term ASEPs to refer to the subset of OS and application extensibility points that can be "*hooked*" to enable auto-starting of programs 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 and commonly run by users (such as the Internet Explorer browser with its Toolbar extensions).

Figure 1 depicts a Windows-based systems with three layers of gates. The Outer Gates are the entrance points for program files from the Internet to get on user machines. The Middle Gates are the ASEPs that allow programs to hook a system to essentially become "part of the system" from a user's point of view. The Inner Gates control the instantiation of program files. Our solution, named *Gatekeeper*, identifies and monitors the Middle Gates and exposes all ASEP hooks to allow effective management of spyware.

Problem Formulation and Decomposition

Figure 2 illustrates the "life cycle" of the spyware management process and provides a problem decomposition that enables us to systematically reason about this complex problem. Note that our current solution does not address the issues of malicious software such as RootKits [P03]; we will briefly discuss malicious behavior in the Discussions section.

In Step (1), given a spyware-infected machine, since we do not have sufficient context information for already-installed spyware programs, we rely on the signature-based scanning and removal tool (such as Ad-

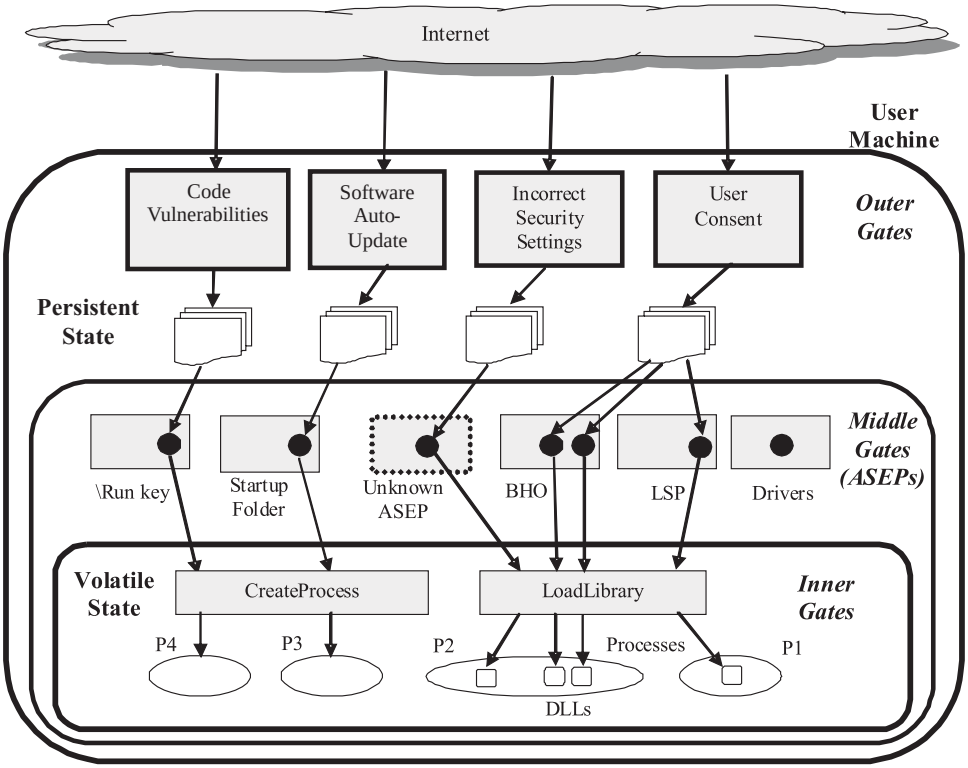


Figure 1: Outer, Middle, and Inner Gates: (1) Outer Gates are the entrance points for program files from the Internet to get on user machines. User Consent includes explicit consent to install, for example, a freeware program, and implicit consent to allow spyware programs bundled with the freeware to get installed as well. Incorrect Security Settings include the "Low" setting for Internet Zone security, incorrect entries in the Trusted Sites list, and incorrect entries in the Trusted Publishers list, which would allow drive-by downloads; (2) Middle Gates are the ASEPs that allow programs to survive reboots and maximize their chance of running all the time. BHO stands for Browser Helper Object. LSP stands for Layered Service Provider; and (3) Inner Gates control the instantiation of program files into active running program instances. They include CreateProcess, LoadLibrary, and other program execution mechanisms, and can be used to block any potentially harmful programs if they are not properly signed or on the known-good list.

Aware and SpyBot-S&D) to remove existing spyware. After Step (1), the Gatekeeper infrastructure is put in place to provide a spyware management framework.

In Step (2), we continuously monitor all ASEPs by recording, alerting, and blocking potentially undesirable ASEP hooking operations. It is essential that the signature database includes user-friendly descriptions of known-good [G03, NSRL, PP] and known-bad ASEP hooks to enable presentation of actionable information to the user.

If the user decides to install a freeware application after assessing the risks of bundled spyware programs as specified in the EULA, bundle tracing in Step (3) captures all components installed by the freeware and display them in Gatekeeper as a group with a user-friendly name enabling the user to manage and remove them as a unit.

In Step (4), we monitor the performance and reliability of the system since the bundle installation and associate any problems with the responsible component(s). These "credit reports" provide the user with a "price tag" for the freeware functionality, enabling the user to make value/cost judgments about the freeware.

Finally, our solution's effectiveness is directly related to completeness of the ASEP list. In Step (5), we discover new ASEPs of OS and popular frequently-run software by either analyzing indirection patterns in file and Registry traces or troubleshooting infected machines. In this paper, we will cover (2), (3), and (5) in the next three sections.

ASEP Management

ASEP Categorization

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

1) **ASEPs that start new processes**: for example, the HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Run Registry key and the %USERPROFILE%\Start Menu\Programs\Startup file folder are well-known ASEPs for auto-starting additional processes.
2) **ASEPs that hook system processes**: for example, HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon\Notify allows a DLL to be loaded into WinLogon.exe.
3) **ASEPs that load drivers**: for example, HKLM\System\CurrentControlSet\Control\Class\{4D36E96B-E325-11CE-BFC1-08002BE10318}\UpperFilters allows loading of a keylogger driver; HKLM\System\CurrentControlSet\Services allows loading of general drivers.
4) **ASEPs that hook multiple processes**: for example, Winsock allows a Layered Service Provider (LSP) DLL or a Name Space Provider (NSP) DLL to be loaded into every process that uses Winsock sockets; HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Windows\AppInit_Dlls allows a DLL to be loaded into every process that links with *User32.dll*.

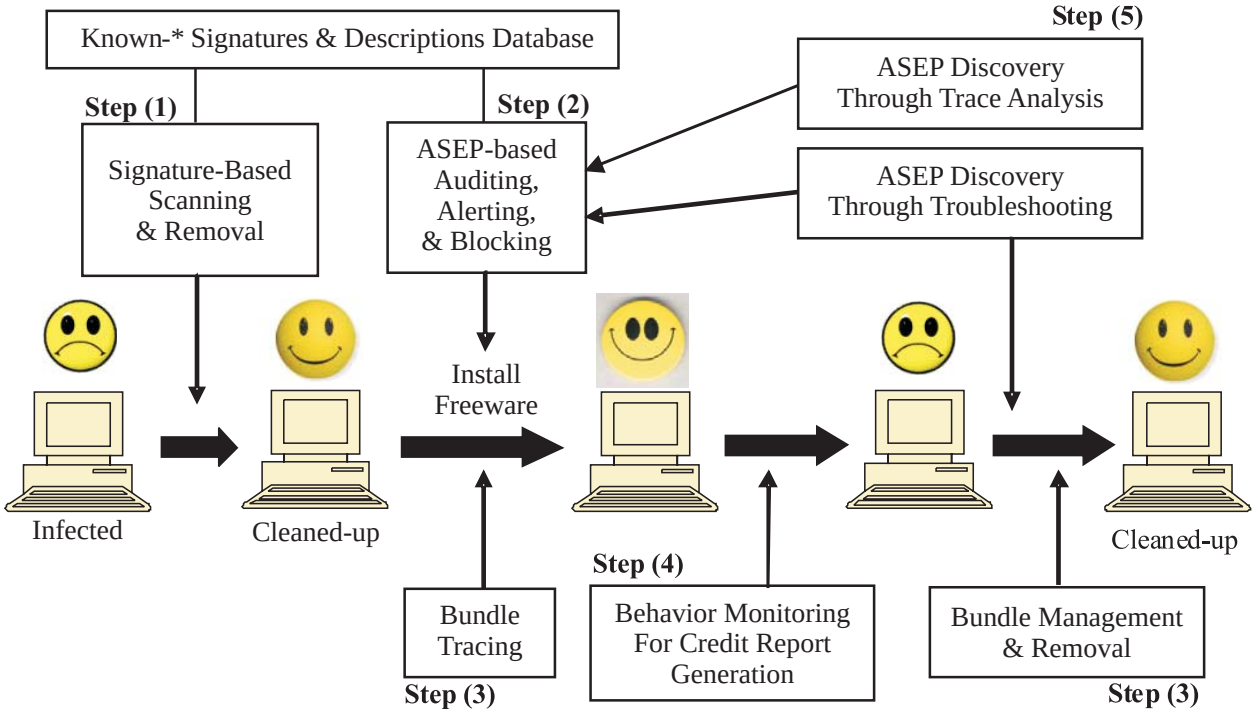


Figure 2: The Spyware Management "Life Cycle" and Problem Decomposition: see descriptions in the Problem Formulation and Decomposition section.

5) **Application-specific ASEPs**: 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 DLLs to be loaded by IE; HKLM\SOFTWARE\Microsoft\Internet Explorer\ Search\SearchAssistant and CustomizeSearch take URLs as input and control which search pages will be loaded.

ASEP Hooking Statistics

Figure 3 shows the number of spyware hooks to each of the 34 ASEPs hooked by at least one of the 120 spyware programs in our Spyware Zoo. Browser Helper Objects (BHOs), HKLM "Run" key, and IE "Toolbar" are the three most popular ASEPs. Figure 4 shows that most of the individual spyware programs hook only three or less ASEPs, 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 ASEPs, which would usually cause significant performance degradation. (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.)

ASEP Monitoring and Alerting

ASEP monitoring watches all known ASEPs for any of the following three types of changes: (1) adding a new ASEP hook; (2) modifying an existing ASEP hook; and (3) modifying the executable file pointed to by an existing ASEP hook.

Each of the above changes generates a new event log entry that contains the ASEP pathname, the ASEP hook name, the executable file pathname or URL, and the timestamp of the hooking operation. Optionally, a notification can be displayed to the user or forwarded to an enterprise management system for processing. Notifications for ASEP programs signed by trusted publishers can be optionally suppressed to reduce false positives.

Figure 5 shows a screenshot of a user notification alert. During the installation of a freeware screensaver, the user is notified of five new ASEP hooks. The "Screen Saver" hook alert is obviously expected. Searching the Signatures and Descriptions Database with the information from the other four alerts (by clicking on the alerts) reveals that they belong to "*eXact Search Bar*" and "*Bargain Buddy*." Based on the information provided for these two pieces of software and the benefit provided by the screensaver, the user can then make informed decision about whether to keep this bundle.

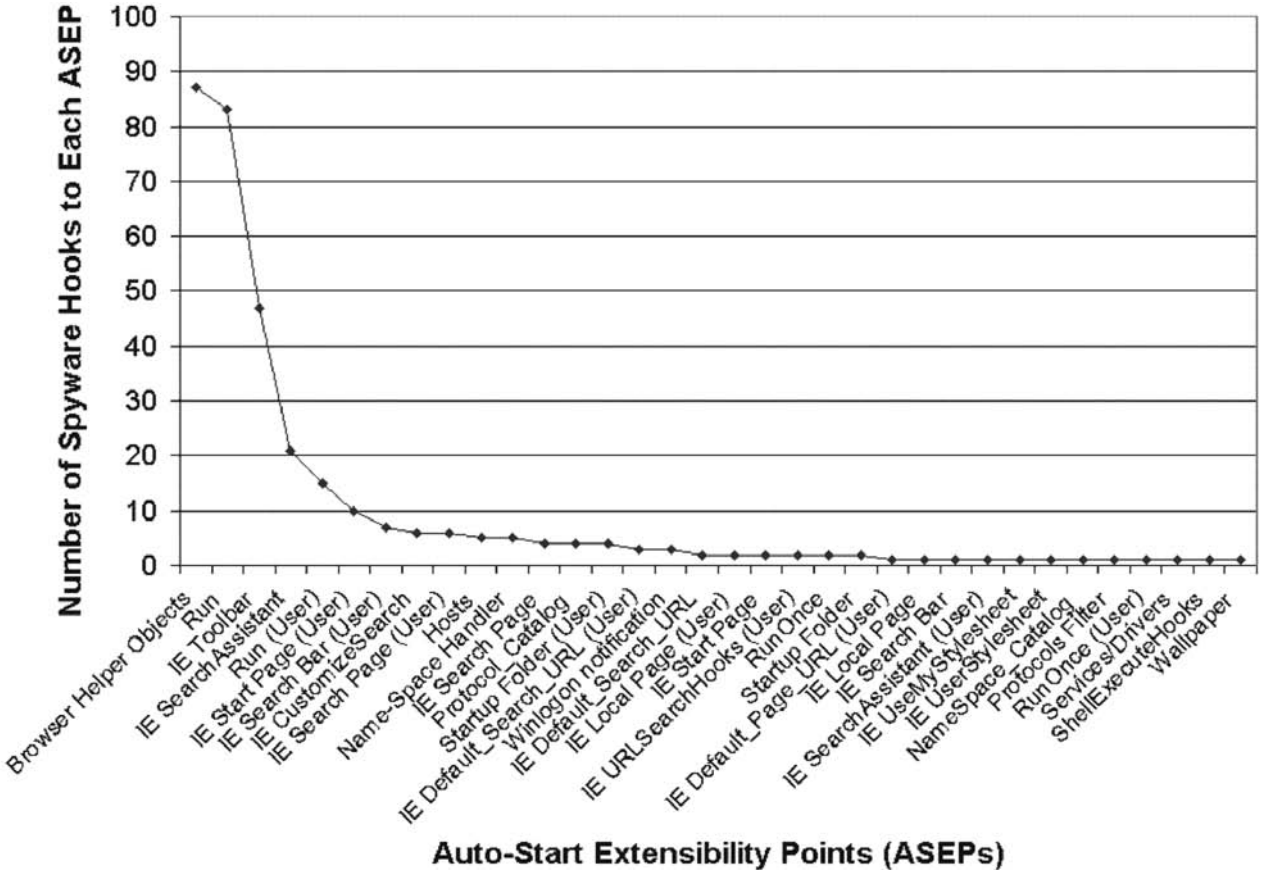


Figure 3: Distribution of spyware ASEP hooks: 120 spyware programs with 334 hooks to 34 ASEPs; ASEPs are sorted by popularity.

Bundle Management

The term 'bundle' represents a set of applications and extensions added to a user's system as part of a single installation process. If a component of a bundle installs additional applications or components at a later time, these are also added to the installer's bundle. A bundle is intended to match an end user's ideal management unit for installing, disabling, and removing software on their system.

Bundle Tracing

Although multiple ASEP alerts appearing during a single installation typically indicate that the ASEPs belong to the same bundle, this time-based grouping is not robust against concurrent installations. For example, Figure 6 illustrates two concurrent installations of the *DivX* bundle (with two ASEP hooks) and the *Desktop Destroyer (DD)* bundle (with five ASEP hooks). Time-based grouping would incorrectly group all seven ASEP hooks in a single bundle.

Gatekeeper uses a bundle tracing technique built on top of the always-on Strider Registry and file tracing [WVD+03, DRD+04]. ASEP hooks created by processes belonging to the same process tree are assigned to the same bundle. If any *Add/Remove Programs (ARP)* entries are created by any process in the tree, the concatenation of their ARP Display Names is used to label the bundle. Referring to Figure 6, the upper process tree defines the *DivX* bundle with two ARP names, and the lower tree defines the *DD* bundle with three ARP names.

Any spyware that does not provide an ARP entry for removal will show up as a bundle with no name. For example, the *ClientMan* software creates one ASEP hook silently at installation time with no accompanying ARP entry. Since installations without ARP entries are uncommon, this installation will be flagged as potentially unwanted.

We have observed that some spyware may initially install partially, and delay the full installation until a later time to make it more difficult for the users to identify which Web site is actually responsible for installing the software. For example, after the partial installation with one ASEP hook, *ClientMan* would non-deterministically select a later time after several reboots to finish its installation with seven additional ASEP hooks.

Gatekeeper bundle tracing captures such devious behavior as follows. First, it performs URL tracing to link each Web-based bundle installation with its source URL. Although IE browser history already records the URL and timestamp for every Web site visited, it is a global history for all instances of IE and is garbage collected after a few weeks. We have implemented a Browser Helper Object to record the process ID of the IE instance that navigated to each URL so that the URL trace can be correlated with the ASEP hooking trace. Second, to handle latent installations, bundle tracing keeps track of all the files created by each bundle. If any of the files are later instantiated to create more ASEP hooks, these additional hooks are added to the original bundle.

Extensibility Point Add/Remove Programs (EP-ARP)

Figure 7 shows Gatekeeper displaying bundle information through a new "*Manage Auto-Start Programs*" button in the Control Panel ARP interface (called it *EP-ARP*). It scans all ASEPs and displays all current hooks by bundles. Users can also choose to

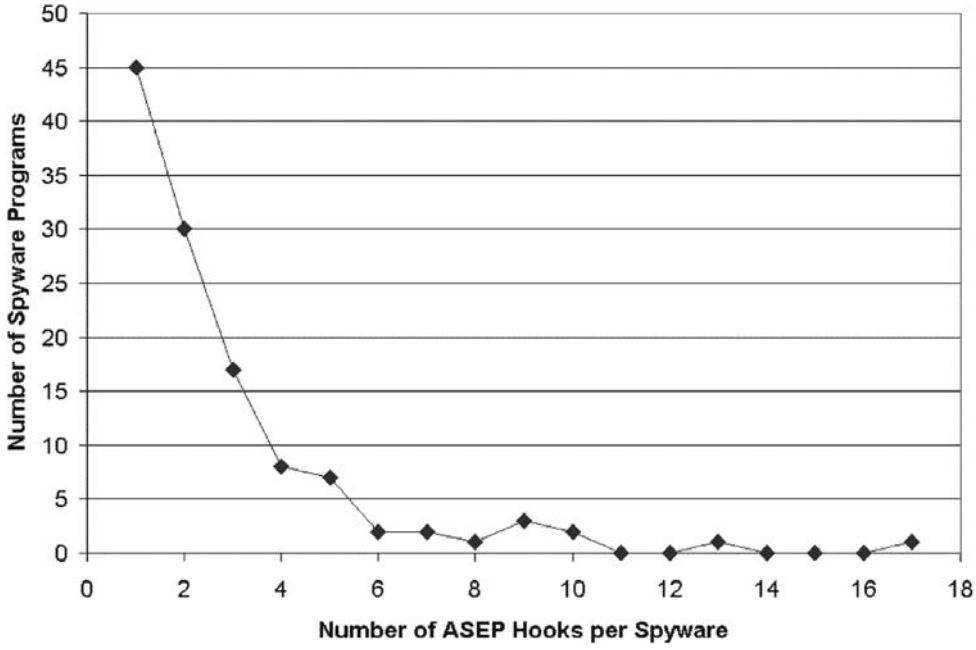


Figure 4: Number of ASEP hooks used by each spyware.

sort the ASEP hooks by the timestamps obtained from the event log in order to highlight newly installed ASEP programs. This is particularly useful when a user invokes EP-ARP immediately after she observes a problem to identify the potential problematic program.

The EP-ARP display also provides three options for bundle removal/disabling. For example, the bundle name clearly shows that "*eXact Search Bar*" and "*Bargain Buddy*" have been installed as part of the *DD bundle*. If the user wants to remove *DD*, she can click the "*Disable Bundle*" button and reboot the machine. This removes all five ASEP hooks, stopping the three bundled programs from automatically starting, despite their files remaining on the machine.

Alternatively, the user can look for the three ARP names in the regular ARP page and invoke their respective removal programs there. Since it is not uncommon for spyware to provide unreliable ARP removal programs, the user can double-check EP-ARP to make sure that none of the ASEP hooks gets left over after ARP removals. Gatekeeper also integrates with System Restore [SR01], as shown at the bottom of Figure 7. If both removal options fail, the user can click on the "*Restore*" button to roll back machine configuration to a System Restore checkpoint taken before the bundles were installed.

ASEP Discovery

In addition to well-known ASEPs and documented ASEPs, we discover new ASEPs through another two channels. The first channel involves troubleshooting machines with actual infections that cannot be cleaned up by Gatekeeper because of spyware using unknown ASEPs. We provide two tools for this purpose: the Strider Troubleshooter [WVD+03] and the automatic *AskStrider* scanner [WRV+04]. The second channel involves analyzing Registry and file traces collected from any machine to discover new ASEPs that can potentially be hooked by future spyware. Once new ASEPs are discovered, they are added to the Gatekeeper list to increase its coverage. The same ASEP discovery procedure can also be used by

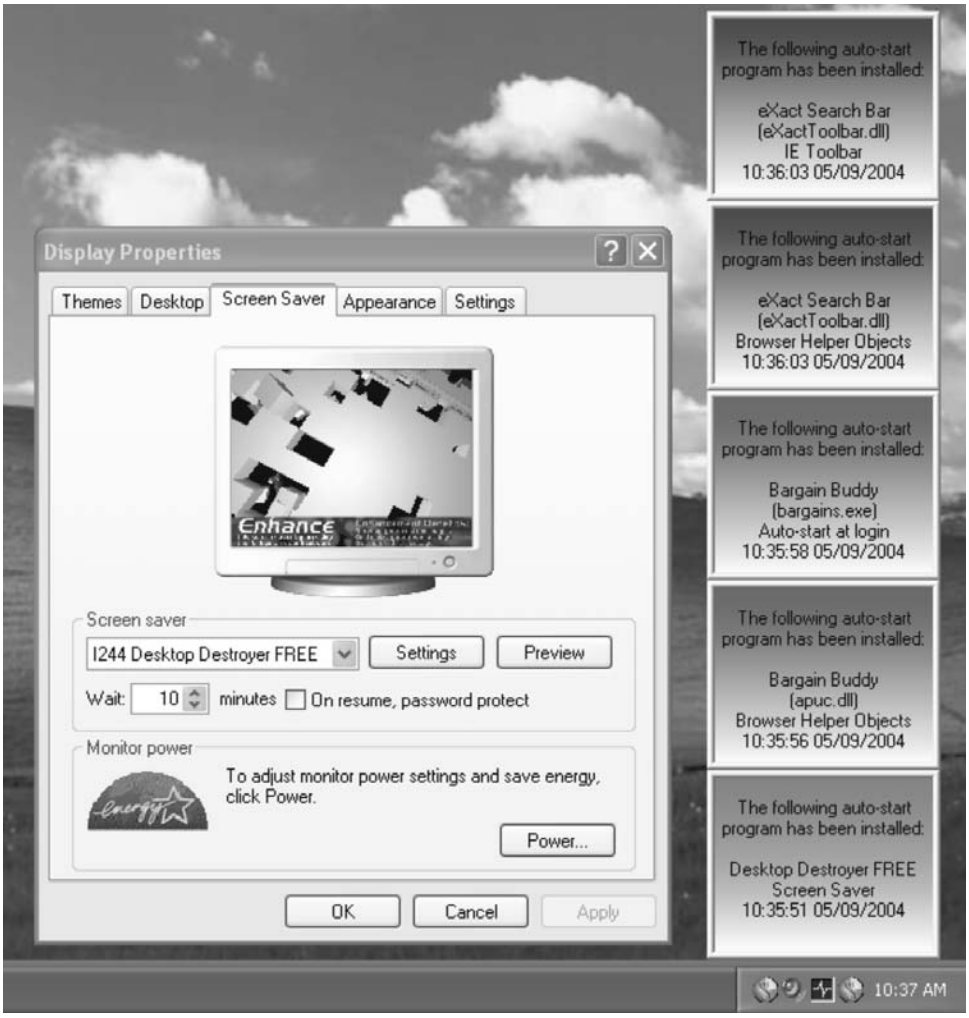


Figure 5: ASEP Hooking Alerts: One freeware screensaver (the bottom alert) bundling two other programs, each hooking two ASEPs (the other four alerts).

system administrators to discover ASEPs in third-party or in-house applications that do not come with a list of specified ASEPs.

ASEP Discovery Through AskStrider

The *AskStrider* scanner is an enhanced Windows Task Manager. In addition to displaying the list of running processes, *AskStrider* displays the list of modules loaded by each process and the list of drivers loaded by the system. More importantly, *AskStrider* gathers context information from the local machine to help users analyze this large amount of information to identify the most interesting pieces. Such context information includes the System Restore file change log, meta-data for patch installations, and driver-device associations [WRV+04].

Figure 8 shows two sample screenshots of *AskStrider*. The upper pane displays the list of processes sorted by the approximate last-update timestamps of their files, according to System Restore. Files that were updated within the past week are highlighted. The lower pane displays the list of modules loaded by the selected process in the upper pane, with the same time-sorting and highlighting. Additionally, if a file came from a patch, the patch ID is displayed as an indication that the file is much less likely to have come from a spyware installation.

Also illustrated in Figure 8 is an example of how *AskStrider* was used to discover a new ASEP. Figure 8 (a) shows that, after the installation of *SpeedBit,* a new process *DAP.exe* was started and the browser process *iexplore.exe* was loading four newly updated DLL files from the same installation. After we disabled all new ASEP hooks from Gatekeeper EP-ARP and rebooted the machine, *iexplore.exe* was still loading two new DLLs as shown in Figure 8 (b). Searching the Registry using the filename *DAPIE.dll* revealed that SpeedBit

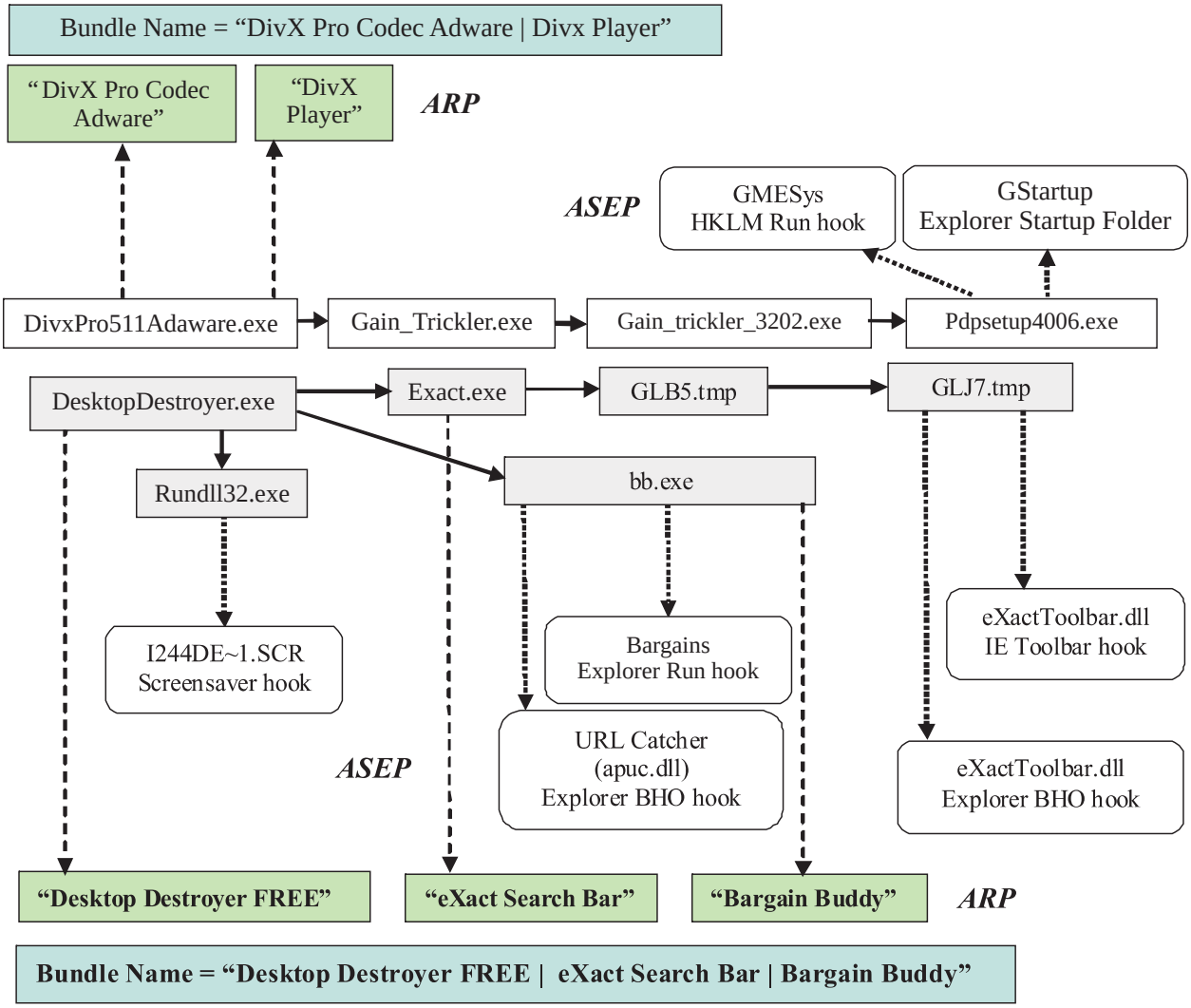


Figure 6: DivX and Desktop Destroyer Bundle Tracing: solid arrows represent creations of child processes; dashed arrows represent creations of ARP entries; dotted arrows represent creations of ASEP hooks. Each process tree defines the scope of the bundle, named by concatenation of ARP friendly names.

was hooking an additional ASEP under HKCR\PROTOCOLS\Name-Space Handler, which has since been added to the ASEP list monitored by Gatekeeper.

ASEP Discovery Through Strider Troubleshooter

The strength of *AskStrider* is that the scanning is completely automatic and typically takes less than a minute to run. The weakness is that it only captures running processes and loaded modules at the time of its scan. If a spyware program gets instantiated through an unknown ASEP and exits before *AskStrider* is invoked, *AskStrider* may not be able to capture any information revealing the unknown ASEP.

The Strider Troubleshooter [WVD+03] can capture such behavior in an ''auto-start trace'' that records every single file and Registry read/write during the auto-start process. This tool asks the user of an infected machine to select a System Restore checkpoint (of files and Registry) that was taken before the infection. By comparing that checkpointed state with the current infected state, the tool calculates a diff set that contains all changes made by the spyware installation. Then it intersects the diff set with the auto-start trace to produce a report that contains all ASEP hooks made by the spyware installation and accessed during auto-start.

For example, in the case of *Praize Desktop*, HKCU\Control Panel\Desktop\Wallpaper was a previously unknown ASEP that allows running an HTML file as a desktop picture. It did not show up in *AskStrider*, but it showed up in the Strider Troubleshooter report as a newly discovered ASEP.

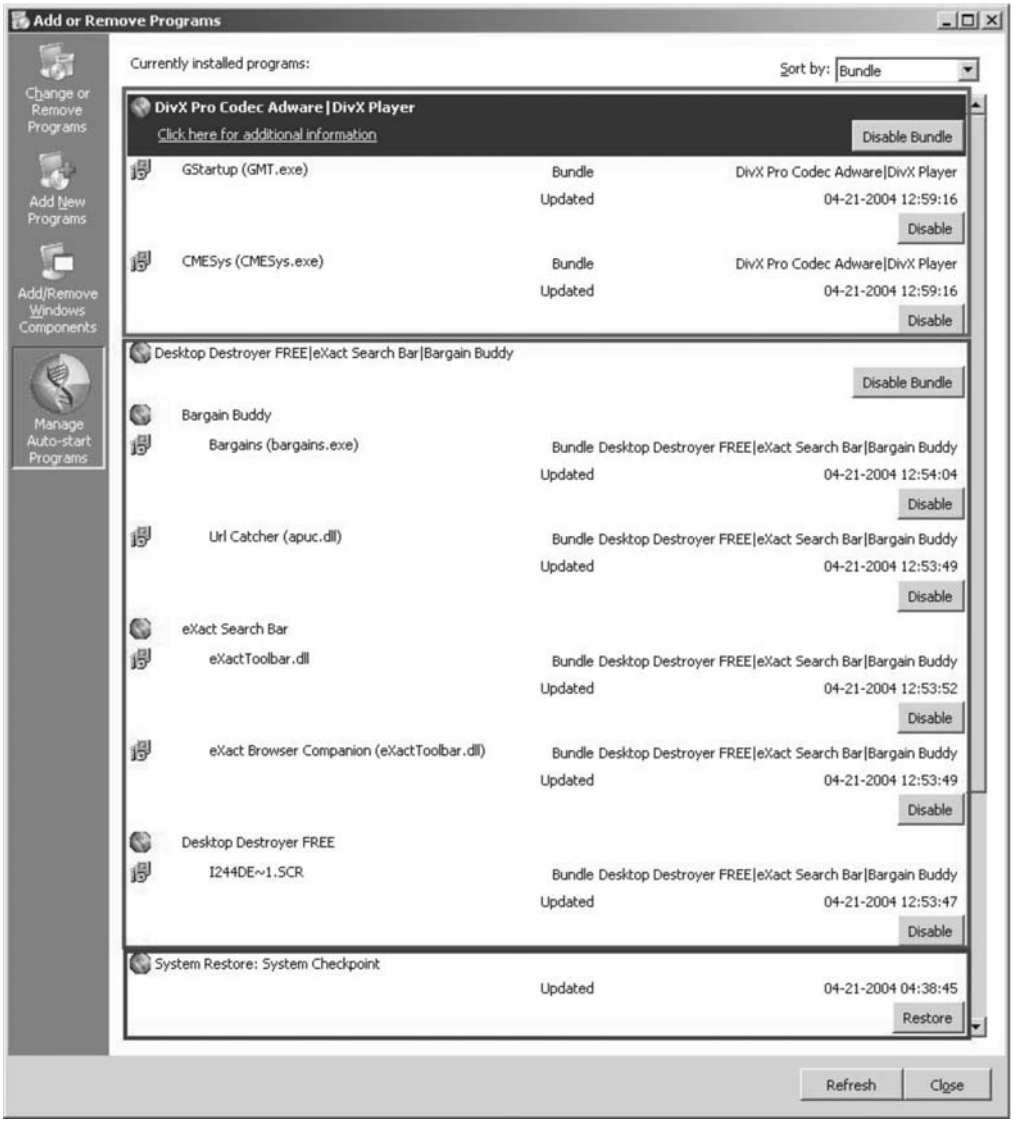
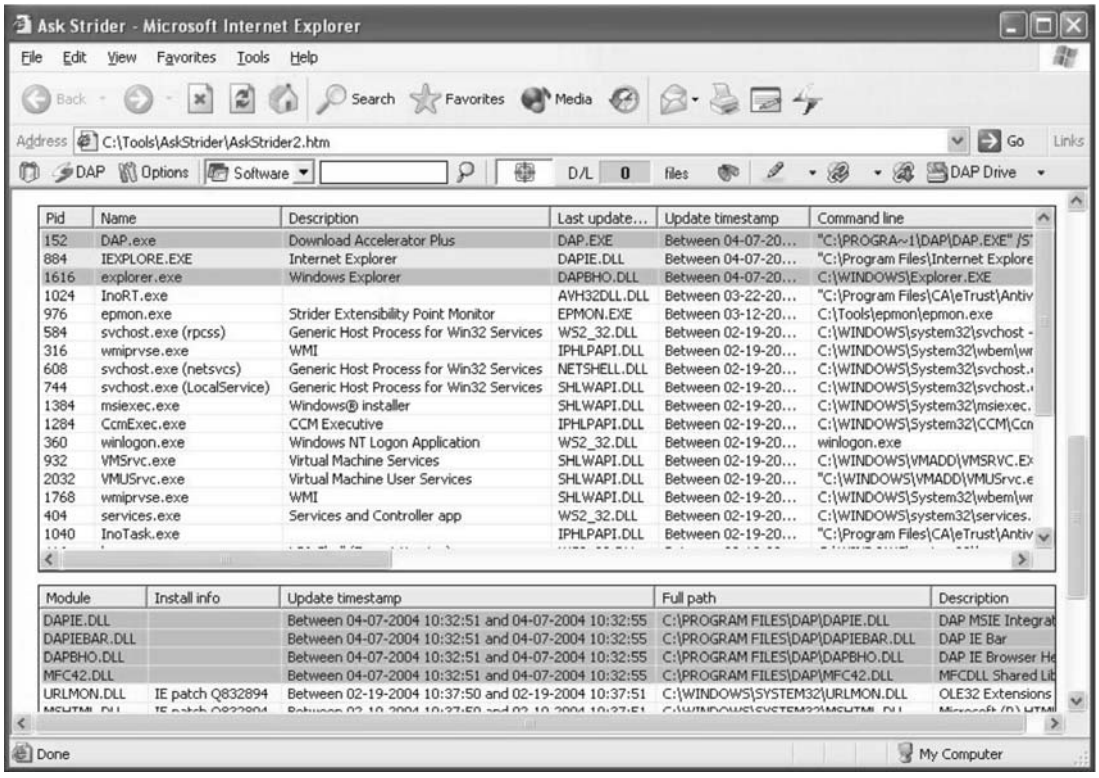
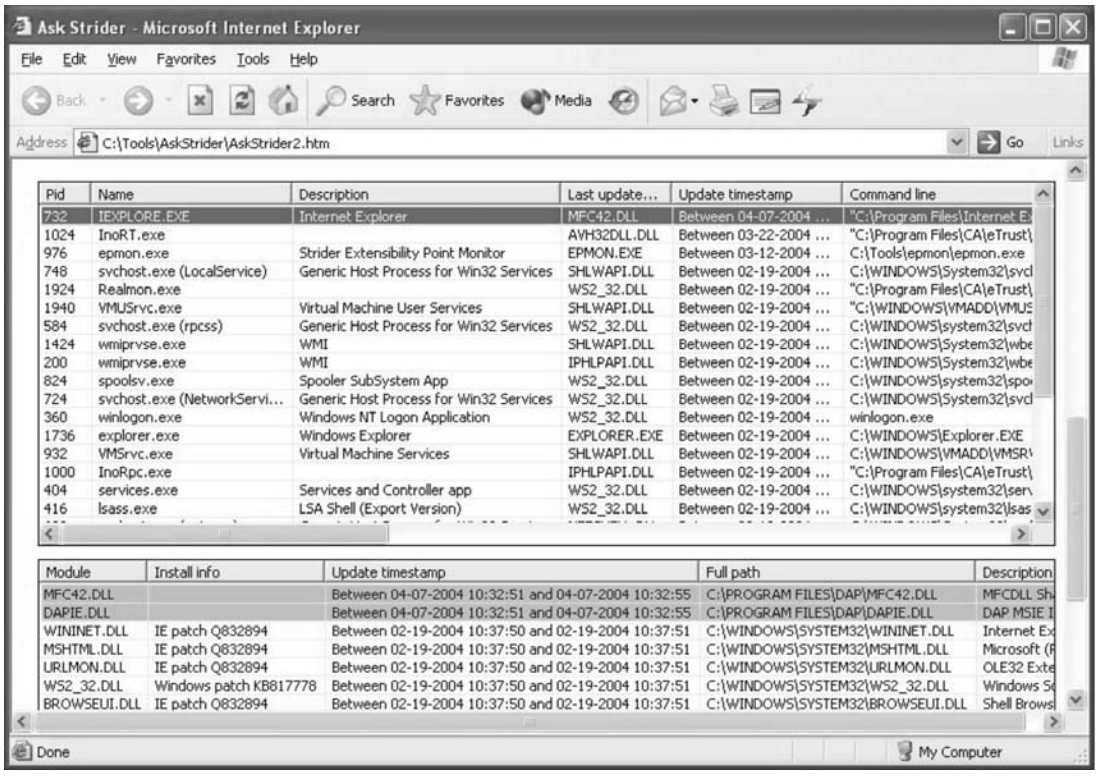


Figure 7: Extensibility Point-Add/Remove Programs (EP-ARP): the ''DivX Pro Codec Adware | DivX Player'' bundle includes two ASEP hooks GMT.exe and CMESys.exe that came from Gator. The ''Desktop Destroyer FREE | eXact Search Bar | Bargain Buddy'' bundle includes five ASEP hooks. Clicking on the ''Restore'' button at the bottom can roll back the system and remove the two bundles.



(a) After Installing SpeedBit: a new process DAP.exe and four new DLLs are highlighted.



(b) After Disabling All New ASEP Hooks from Gatekeeper and Rebooting the Machine: two new DLLs are still loaded through a previously unknown ASEP.

Figure 8: *AskStrider* for ASEP discovery.

ASEP Discovery Through Strider Trace Analysis

By definition, ASEP programs must (1) appear in the ''auto-start trace'' that covers the execution window from the start of the booting process to the point when the machine ''finishes all initializations and is ready to interact with the user''; and (2) get instantiated through an extensibility point lookup, instead of having their instantiation hard-wired into other programs that are auto-started.

New ASEPs can therefore be discovered by analyzing the auto-start trace from any machine to identify the following indirection pattern: an executable filename is returned as part of a file or Registry query operation, followed by an instantiation of that executable file.

In an experiment, we collected auto-start traces from five Windows XP machines for analysis. By looking for the indirection pattern, we were able to validate some of the known ASEPs in our list and discover 17 new ASEPs (including five ASEPs for a third-party, auto-start anti-virus program). There are three distinctive classes of patterns:

1) ASEPs that accommodate multiple hooks: for example, HKLM\SOFTWARE\Microsoft\InetStp\Extensions allows for multiple administrative extensions for the IIS server; HKLM\SOFTWARE\Microsoft\Cryptography\Defaults\Provider allows for multiple providers; HKLM\SOFTWARE\Microsoft\Windows NT\Current Version\Winlogon\Userinit allows for multiple initialization programs specified in a comma-separated string.
2) ASEPs with a single hook: for example, the ASEP HKCR\Network\SharingHandler appears to allow only one handler.
3) ASEPs that require multiple indirections for lookup: for example, every hook to the ASEP HKLM\SOFTWARE\Microsoft\Windows\Current Version\ShellServiceObjectDelayLoad contains a Class ID that is used in an additional Registry lookup to retrieve the executable filename from HKCR\CLSID\<Class ID>\InProcServer32.

We have observed a couple of interesting cases where our analysis may produce ''false-positive'' ASEPs in the sense that it is arguable whether they should be included in our list for monitoring. First, some DLL files do not export any functions and are only used as resource files to provide data; so they may not be considered ASEPs. But they are still potential ASEPs if a *DllMain* routine can be added to cause code execution.

Another case is organization-specific ASEPs. For example, all the machines in the same organization may run an auto-start program deployed by its IT department that exposes its own ASEPs. Obviously, such ASEPs should not be added to the global list; but the system administrators in the organization may want to add them to their local list if they are concerned about these ASEPs being hooked.

Discussions

Although Gatekeeper is proven to be effective against today's spyware, there are many different ways in which spyware can evolve to evade detection. In this section, we discuss such limitations and potential future work to address them.

Limitations of ASEPs

In general, the following problem is intractable:

given the static persistent-state image stored on a hard drive, determine what code will be executed when a machine is booted into the OS image stored on that drive.

Ideally, we need to trace all executions by actually booting into that OS and recoding all processes, modules, drivers, and code segments that are loaded or injected.

We introduced the concept of ASEPs in the context of spyware management as an approximation to the non-existing solution to the above problem. This approximation has at least five limitations:

1) **Definition of "popular and commonly run programs"**: beyond OS programs, we have included only the Web browser for ASEP consideration. Many commercial or freeware applications may have a sufficiently large install base and running frequency that make them attractive spyware targets.
2) **Cascading ASEP programs**: any ASEP programs can provide their own custom ASEPs for other programs to hook. So, in theory, there can be chains of an infinite number of ASEPs that allow cascading auto-starts of ASEP programs. It is also possible for an ASEP program to serve as a custom task scheduler that allows spyware programs to be launched after any definition of the ''auto-start phase.''
3) **Non-ASEP auto-start programs**: The ASEP-based approach does not capture programs that auto-start through non-extensibility mechanisms. Although we have not seen many spyware programs infecting system files directly today, that approach may be popular among Trojans [I04] and may become popular among spyware programs once Gatekeeper exposes all ASEP hooks. We need to rely on additional signature or file-hashing mechanisms to protect system files. If a malicious spyware program uses code injection and thread hijacking to evade detection, ASEP monitoring should still be useful in capturing the first instantiated spyware program. In theory, it may also be possible for a program to hide inside an input file and get instantiated when the file is read by an auto-start program by exploiting code vulnerabilities.

4) **ASEP hijacking**: Gatekeeper assumes that the underlying operating system has not been compromised, so the list of ASEPs on a spyware-infected machine is the same as that on a clean machine. It is possible that a malicious spyware program can ''hijack'' the ASEPs by replacing system files; essentially, the machine can be considered to be running a different operating system in such cases. For example, a Web posting [CAR04] describes a way to modify the binary file of explorer.exe to create arbitrary ASEPs. In our current work, we consider such malicious programs targets of anti-virus programs, not Gatekeeper. In our future work, we plan to rely on digital signatures and file hashes to verify that the underlying operating system is not compromised.
5) **ASEP hook hiding**: Another way for malicious software to defeat ASEP-based scanning is to intercept all file and Registry query operations and remove the software's own ASEP hooks from the query results before they are returned to Gatekeeper. Many RootKits are known to provide such capability [P03]. There have been recent reports that an open-source RootKit is being used to hide spyware programs from anti-spyware tools [HD]. We plan to augment Gatekeeper with an external scanning mechanism, which is required to combat such malicious programs that essentially take over the entire machine once they get started [WVR+04].

Finally, the operating system can be configured to auto-start programs based on generated system events resulting from the insertion of removable media like CDs, hot-pluggable hardware like USB key rings, etc. However, as long as they require explicit user actions to connect the media to the machine and are not automatically started upon reboot, they are not considered ASEPs in this paper.

Bundle Management Challenges

Our bundle tracing technique assumes that the spyware installation programs that we monitor do not try to intentionally confuse or maliciously attack Gatekeeper. One can imagine that a deceptive spyware could hijack ARP and ASEP hook entries of other good software so that they are incorrectly included in the bad bundle. A malicious spyware running with administrator privileges could even disable the bundle tracer or modify the recorded bundle information.

There are two additional challenges that do not involve malicious behavior. First, since our bundle tracer does not track inter-process communications, any bundle installation that involves communications between multiple process trees may appear as multiple, separate bundles. Second, if two toolbars from two different bundles are loaded into the browser at the same time and one of them expands the bundle by hooking an additional ASEP, process tree-based tracking will not provide sufficiently fine-grain information to determine which bundle the new ASEP hook should belong to.

Limitations of *AskStrider*

AskStrider extracts the last-update timestamps of files from the System Restore file change log and is therefore subject to its limitations. First, System Restore only monitors files with certain filename extensions [SRM]; spyware programs with extensions outside the monitored set will not be captured by *AskStrider's* highlighting of recent changes because their updates will not be captured in the file change log.

Second, System Restore excludes certain temporary folders from monitoring. As a result, file updates inside those folders will not be captured in the change log. Third, a malicious spyware program may delete or corrupt the file change log or hide its processes and modules from the *AskStrider* scan.

Finally, a main feature of *AskStrider* is that it highlights recent changes to aid troubleshooting. We have found that such filtering mechanisms are essential for reducing the complexity in many systems management problems. An obvious limitation of this approach is that it will not work well if a user invokes *AskStrider* long after a spyware installation.

Other Web Browsers and Non-Windows Platforms

In this section, we study the ASEPs and spyware-related issues in other Web browsers and operating systems.

Other Web Browsers

As shown in the Outer Gates in Figure 1, code vulnerabilities can be used as an infection vector for spyware through ''drive-by downloads.'' The problem is not unique to the IE browser; other browsers also have known vulnerabilities, such as Mozilla [KVM] and Netscape [HN00]. Exploits of vulnerabilities in the Mozilla and Firefox Web browsers have been widely publicized in news articles [M04, MO04, BZ]. Also, Secunia Advisories [SEC] often describe vulnerabilities that affect the Opera and Mozilla Web browsers.

Other browsers also expose extension mechanisms similar to Windows ActiveX as another infection vector for spyware through plug-in installations with explicit or implicit user consent. Those affecting Mozilla have used a *.xpi* file which is essentially a *.zip* file containing a JavaScript installer and the files/directories to install [CNPM]. An example of this is the Flingstone XPI extension at http://www2.flingstone.com/cab/sbc_netscape.xpi which contains *install.js* and *sbc_netscape.exe.* When installed through Mozilla Firefox, Flingstone adds several ASEP hooks to the BHO and HKLM Run key. This infects the Windows OS in addition to IE although it does not appear to infect Firefox itself [DSM04]. The bundle includes the well-known software bridge.dll [FB04]. We also found a Flingstone-clone *ist_netscape.xpi* containing

install.js and *istinstall_netscape.exe* and exhibiting essentially the same behavior.

Just like IE, other Web browsers expose ASEPs that can potentially be hooked by spyware. For example, Mozilla Firefox has a file system-based ASEP at C:\Program Files\Mozilla Firefox\plugins; all plug-in DLL files placed in that directory are automatically loaded by Mozilla. It also scans a Registry-based ASEP at HKLM\SOFTWARE\MozillaPlugins to locate plug-ins that register with the browser through PLIDs [PLUG].

The homepage and search page related ASEPs of non-IE browsers are generally stored in application specific preference files rather than the Registry. For example, there are two user preference files in the profile directory of Netscape/Mozilla: *prefs.js* that contains automatically generated default preferences, and *user.js* which contains options that override settings in *prefs.js*. Spyware can hijack the home page and the default search page of these browsers by altering the value of user_pref("browser.startup.homepage," "<home page>") and user_pref("browser.search.defaultengine," "<search page>") in *prefs.js* [NCPI]. For example, the Lop.com software has been known to hijack Netscape/Mozilla home page [LOP]. In general there appears to be less spyware targeting non-IE browsers, presumably because their smaller install base is less attractive to spyware developers.

In some cases, the search and download functionalities of the browser software itself may raise similar privacy concerns. It was reported [NNB02] that, while data on searches conducted from IE's search pane was sent directly to the designated search site and was not intercepted by Microsoft, searches performed by using Netscape Navigator's Search button were intercepted by Netscape and tagged with information that can potentially identify individual machines. The term "*File Download Spyware*" [FDS00] refers to those file downloaders that by default track user's entire file download history tagged with a unique ID, the machine's IP address, or even the user's personal email address.

Non-Windows Platforms

ASEPs on UNIX operating systems such as Linux, AIX, and Solaris can be roughly classified into four categories:

1) **The *inittab* and *rc* files**: The file /etc/inittab instructs the *init* process what to do when the system is up and initializing. It typically asks *init* to allow user logons (*gettys*) and start all the processes in the directories specified by the /etc/rc.d/rc file and other *rc* files such as /etc/rc.d/rc.local, which is a common place for the root user to customize the system, including loading additional daemons.
2) **The *crontab* tool**: The *cron* daemon is started from either the *rc* or the *rc.local* file, and provides task scheduling service to run other processes at a specific time or periodically. Every minute, cron searches /var/spool/cron for entries that match users in the /etc/passwd file and also searches /etc/crontab for system entries (note that any modification to this file requires root privileges.) It then executes any commands that are scheduled to run.
3) **Configuration profiles for user environment** (such as *.bash* for bash shell, *.xinitrc* or *.Xdefaults* for X environment, and other profiles in /etc/) are potential ASEPs. Users are typically unaware of what is loaded when they log on, or start an X window session. A simple script file that contains the command

```
script -fq /tmp/.syslog
```

can be used to hook an ASEP to record the terminal activities of the whole system or a specific user account, depending on the ASEP location. The recording is usually stored in a hidden file (i.e., a filename that begins with a ".") under the global-writable /tmp directory
4) **Loadable Kernel Modules (LKMs)** are units of object code that can be dynamically loaded into the kernel to provide new functionalities. By default most LKM object files are placed in the directory /lib/modules. However, some customized LKM files can reside anywhere on the system [LKMP]. The programs *insmod* and *rmmod* are responsible for inserting and removing LKMs, respectively.

Our preliminary investigation shows that spyware is not a substantial threat to the current Unix/Linux world. Perhaps this is because Unix/Linux has a much smaller install base than Windows in the consumer desktop market, which makes it less attractive to spyware writers. Another reason might be that most Unix/Linux users do not run as administrators; many, if not most, of the spyware programs require administrator privileges to install and run. Finally, Unix/Linux users who do run as administrators are advanced users who are unlikely to fall into the trap of installing spyware.

Related Work

Earlier versions of commercial anti-spyware programs focused on the signature-based, on-demand scanning approach. The latest Ad-Aware Ad-Watch real-time monitor [AP] and Spybot-S&D TeaTimer [ST] provide real-time monitoring similar to Gatekeeper ASEP monitoring. But they do not seem to include centralized auditing and bundle tracing, and the context information that they provide to the users is limited, making them less effective as a management solution as compared to Gatekeeper. On the other hand, they put more emphasis on blocking and protection. The Autoruns tool [AR04] and the Windows XP SP2 IE Add-on Manager both cover only a subset of ASEPs known to be hooked by spyware.

An alternative approach to combating spyware programs is to cut off their communications with remote servers so that collected personal information will not be sent out. One way to achieve this is to use the *Hosts* file to map all blacklisted host names to the local loopback address [BUP]. This approach essentially applies known-bad signatures to the host names and similarly lacks the context information for proper spyware management. Moreover, it addresses only the privacy issue, not the reliability and performance issues.

Saroiu, et al. [SGL04] presented a measurement study of four widespread spyware programs in a university environment by analyzing a week-long trace of network activity. Their results showed that the spyware problem is of large scope. They also described a specific vulnerability in actual spyware programs to demonstrate that the potential for spyware to introduce substantial security problems is real.

Summary

In this paper, we have modeled the spyware management problem as an ASEP tracking and bundling problem. We have described the Gatekeeper solution that provides visibility into important system changes and answers the following critical questions for every potential spyware program:

1) **"Where did it come from?"** Our URL source tracing identifies the Web site from which the program was downloaded; bundle tracing identifies the freeware that bundled the spyware.
2) **"When was it installed, where was it installed, and what was installed?"** Our ASEP monitoring detects and records the installation events, and context lookup determines which file is installed where.
3) **"How does it get instantiated?"** The ASEP to which the program is hooking determines how it will get auto-started.
4) **"How do I disable/remove it?"** Our extended Add/Remove Programs user interface exposes each ASEP hook and allows simple disabling; alternatively, the ARP entries that are bundled with the ASEP hooks can be used for removal.

With these capabilities, the Gatekeeper tool in its current form is useful for technical users and system administrators to gain back control of their machines and to effectively manage spyware. But there remains one critical piece to the puzzle to make the tool useful and make the presentation actionable by average users:

5) **"What does it do?"** This will require detailed experiments and analysis of the program and matching the program's behavior against a list of objective criteria. It will then allow the user to make an informed decision about whether to remove the program, based on the trade-off between the benefit and the potential privacy/security/reliability/performance concerns of all the bundled programs.

Author Information

Yi-Min Wang manages the Systems Management Research Group and leads the Strider project at Microsoft Research, Redmond. 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 systems and security management, fault tolerance, home networking, and distributed systems.

Roussi Roussev is currently a Ph.D. student at Florida Institute of Technology. He was an intern at Microsoft Research in 2003 and 2004. His research interests include security, systems management and software verification.

Chad Verbowski has been a Development Lead at Microsoft for the past six years working on multiple Windows OS components and Systems Management software. Previously Chad worked for several years building and deploying management systems in real world environments. He is currently with Microsoft Research. Chad can be reached at chadv@microsoft.com .

Aaron Johnson has contracted with the Systems Management Research Group at Microsoft Research. He received a Bachelors degree in Computer Information Systems in 1990 from DeVry Institute of Technology in Phoenix. He has nine years of experience troubleshooting Windows hardware and software problems.

Ming-Wei Wu is a software engineer at Institute for Information Industry and the chair of TWING (TaiWan Internet Next Generation) at Taiwan Network Information Center. He received his MS degree from National Chiao Tung University in 2003 and has been a Ph.D. candidate in Electrical Engineering at National Taiwan University since 2004. His research interests include network security, P2P networking and fault tolerance.

Yennun Huang received his MS and Ph.D. degrees from University of Maryland. He worked for AT&T/Lucent Bell Labs for 12 years and was the Department Head of the AT&T Dependable Computing Research Department before he joined a startup as VP of Engineering in 2001. His research interests include dependable distributed computing, mobile infrastructure and applications, and middleware.

Sy-Yen Kuo has been with National Taiwan University since 1991 and was Head of Department of Electrical Engineering from 2001 to 2004. He received his Ph.D. in Computer Science from the University of Illinois at Urbana-Champaign in 1987, was a faculty member in the Department of Electrical and Computer Engineering at the University of Arizona from 1988 to 1991, and was the Chairman of the Department of Computer Science and Information Engineering, National Dong Hwa University, Taiwan from 1995 to 1998. His current research interests include software reliability engineering, mobile computing, and dependable systems and networks.

References

[AA] *Ad-Aware*, http://www.lavasoft.de/ms/index.htm .

[AP] *Ad-Aware Plus*, http://www.lavasoftusa.com/software/adawareplus/ .

[AR04] *Autoruns*, http://www.sysinternals.com/ntw2k/freeware/autoruns.shtml .

[BUP] *Blocking Unwanted Parasites with a Hosts File*, http://www.mvps.org/winhelp2002/hosts.htm .

[BZ] *Bugzilla Bug 249004*, http://bugzilla.mozilla.org/show_bug.cgi?id=249004 .

[C04] *Spyware Cures May Cause More Harm Than Good*, http://news.com.com/2100-1032-5153485.html , Feb, 2004.

[CAR04] "Create New Autorun by Patching Explore.exe," *The Online Rootkit Magazine*, http://www.rootkit.com/newsread.php?newsid=118 , 2004.

[CNPM] *Creating New Packages for Mozilla*, http://www.mozilla.org/docs/xul/xulnotes/xulnote_packages.html .

[DRD+04] Dunagan, John, Roussi Roussev, Brad Daniels, Aaron Johson, Chad Verbowski, and Yi-Min Wang, "Towards a Self-Managing Software Patching Process Using Black-Box Persistent-State Manifests," in *Proc. Int. Conf. Autonomic Computing*, May, 2004.

[DSM04] "Discussion of sbc_netscape.xpi from MozillaZine," http://forums.mozillazine.org/viewtopic.php?t=66531 .

[E04A] "EarthLink Finds Rampant Spyware, Trojans," *InfoWorld*, http://www.infoworld.com/article/04/04/15/HNearthspyware_1.html , April 15, 2004.

[E04B] "FTC to Look Closer at Spyware," *Washington Post*, http://www.washingtonpost.com/wp-dyn/articles/A22514-2004Apr18.html, April 19, 2004.

[FB04] *Flingstone Bridge*, http://www.kephyr.com/spywarescanner/library/flingstonebridge/index.phtml .

[FDS00] *The Anatomy of File Download Spyware*, http://grc.com/downloaders.htm .

[FTC04] "Monitoring Software on Your PC: Spyware, Adware, and Other Software," *Workshop Transcript*, Federal Trade Commission, http://www.ftc.gov/bcp/workshops/spyware/transcript.pdf .

[G03] Garfinkel, Simson L., *A Web Service for File Fingerprints: The Goods, the Bads, and the Unknowns*, http://www.simson.net/clips/2003.15_972.FinalPaper.pdf .

[HD] "Nasty New Parasite," *SpywareInfo Newsletter*, June 18, http://www.spywareinfo.com/newsletter/archives/0604/8.php .

[HN00] "Hacker finds hole in Netscape," *Wired*, http://www.wired.com/news/technology/0,1282,38087,00.html .

[I04] *Institution 2004 Remote Admin Tool*, http://www.evileyesoftware.com/ees/content.php?content.43 .

[LKMP] *The Linux Kernel Module Programming Guide*, http://tldp.org/LDP/lkmpg/ .

[KVM] *Known Vulnerabilities in Mozilla*, http://www.mozilla.org/projects/security/known-vulnerabilities.html .

[LOP] *Lop.com ("Live Online Portal") Parasite*, http://www.doxdesk.com/parasite/lop.html .

[M04] "Mozilla Flaw Lets Links Run Arbitrary Programs," *eWeek*, http://www.eweek.com/article2/0,1759,1621451,00.asp , July 8, 2004.

[MO04] "Mozilla to squash security bugs," *CNET News.com*, http://news.com.com/Mozilla+to+squash+security+bugs/2100-1002_3-5286138.html , July 27, 2004.

[NCPI] *Netscape Communicator Preferences Index*, http://developer.netscape.com/docs/manuals/communicator/preferences/ .

[NNB02] *Netscape Navigator Browser Snoops on Web Searches*, http://www.computeruser.com/news/02/03/08/news5.html .

[NSRL] *National Software Reference Library (NSRL) Project Web Site*, http://www.nsrl.nist.gov/ .

[P03] Poulsen, Kevin, "Windows Root Kits a Stealthy Threat," *SecurityFocus*, http://www.securityfocus.com/news/2879, Mar 5, 2003.

[PLUG] *PluginDoc for Mozilla*, http://plugindoc.mozdev.org/notes.html#scan-acroread .

[PP] *Pest Patrol*, http://research.pestpatrol.com .

[SB] *Spybot-S&D*, http://www.safer-networking.org/microsoft.en.html .

[SEC] *Secunia Advisories*, http://secunia.com/advisories/ .

[SGL04] Saroiu, S., S. D. Gribble, and H. M. Levy, "Measurement and Analysis of Spyware Infections in a University Environment," *Proc. of the First USENIX/ACM Symp. on Networked Systems Design and Implementation (NSDI)*, 2004.

[SR01] *Windows XP System Restore*, http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnwxp/html/windowsxpsystemrestore.asp .

[SRM] *System Restore Monitored File Extensions*, http://msdn.microsoft.com/library/default.asp?url=/library/en-us/sr/sr/monitored_file_extensions.asp .

[ST] *Spybot-S&D TeaTimer*, http://www.safer-networking.org/en/faq/33.html .

[WRV+04] Wang, Yi-Min, Roussi Roussev, Chad Verbowski, Aaron Johnson, and David Ladd, "AskStrider: What Has Changed on My Machine Lately?," *Microsoft Research Technical Report MSR-TR-2004-03*, Jan, 2004.

[WVD+03] Wang, Yi-Min, Chad Verbowski, John Dunagan, Yu Chen, Helen J. Wang, Chun Yuan, and Zheng Zhang, "STRIDER: A Black-box, State-based Approach to Change and Configuration Management and Support," *Proc. Usenix Large Installation Systems Administration (LISA) Conference*, pp. 159-171, October 2003.

[WVR+04] Wang, Yi-Min, Binh Vo, Roussi Roussev, Chad Verbowski, and Aaron Johnson, "Strider GhostBuster: Why It's A Bad Idea For Stealth Software To Hide Files," *Microsoft Research Technical Report MSR-TR-2004-71*, July 2004.