# AskStrider: *What Has Changed on My Machine Lately?*

Yi-Min Wang
Roussi Roussev
Chad Verbowski
Aaron Johnson
David Ladd

January 5, 2004

Technical Report
MSR-TR-2004-03

Microsoft Research
Microsoft Corporation
One Microsoft Way
Redmond, WA 98052

# AskStrider: *What Has Changed on My Machine Lately?*

**Yi-Min Wang, Roussi Roussev, Chad Verbowski, Aaron Johnson, and David Ladd**
*Microsoft Research, Redmond WA*

***Abstract***

*As computer programs grow more complex, extensible, and connected, it becomes increasingly difficult for users to understand what has changed on their machines and what impact those changes have. In this paper, we describe a tool, called AskStrider, that answers those questions by correlating volatile process information with persistent-state context information and change history. AskStrider automatically scans a system for active components, matches them against a change log to identify recently updated and hence more interesting state, and searches for context information to help users understand the changes. We use several real-world cases to demonstrate the effectiveness of using AskStrider to quickly identify the presence of unwanted software, to determine if a software patch is potentially breaking an application, and to detect lingering components left over from an unclean uninstallation.*

## 1. Introduction

As computer programs grow more complex, extensible, and connected, it becomes increasingly difficult for users to understand what is causing their machines to behave differently. Network connectivity allows programs to be automatically downloaded and installed. Browser extensibility allows components to be hooked into the browser in sophisticated ways that change users' browsing experience. Complexity of the programs and their interactions are often beyond users' comprehension and makes troubleshooting a very difficult task.

Today, problems due to unwanted software installations through "drive-by downloads" are becoming a top generator of support calls. Such software can be downloaded and installed without any consent from the user, if she visits a rogue Web site with the browser security setting set too low. Or, a user may unintentionally give consent to such downloads when she installs other free software packages, obtains electronic coupons, or clicks on the "OK" button without reading the fine print.

Unwanted software installations can cause a wide range of problems. "Ad-ware" that generates pop-up ads and browser plug-ins that add toolbars/menu items or replace the default search engines can be quite annoying. "Spyware" that monitors user's browsing patterns and/or occasionally hijacks user's browser sessions to redirect them to sponsor sites raises privacy concerns; such concerns become even more serious when "snoopware" is used to record keystrokes and screenshots. Finally, viruses, worms, or backdoors may be installed to cause significant security issues.

Software patching is another major source of changes to users' machines. Several recent serious outbreaks of viruses and worms that exploited security vulnerabilities of Windows software have demonstrated the importance of timely application of security patches. One of the reasons that users delay applying patches is that they are concerned about the patched programs potentially breaking their existing applications. This is particularly true in the corporate desktop environments and in the data center environments running mission-critical server applications. When an application stops working after a few patches are applied, it is in general very difficult to determine if any of the patches are at fault.

Incomplete removal of software packages is another common problem that causes user frustration. Due to the non-transactional nature of most uninstallation programs, a subset of components may be left over from a supposedly removed package and cause inconsistency or confusion. Some spyware may intentionally leave components running, even after uninstallation, so that they can continue to monitor user activities. In general, it is difficult for the users to determine if there are any left-over components from an unclean application removal still running on their machines.

We have built a tool, called AskStrider, to help users answer the question of *"what has changed on my machine lately"* and to help them understand the impact of those changes. AskStrider is completely automatic: it starts by scanning the system for active processes, libraries, drivers, etc., and their run-time dependencies. Then it queries the file change history (maintained by System Restore [SR] on Windows XP machines) and ranks the active components by how recent their corresponding files appeared on the machine so that "young" components are highlighted. Finally, it searches the local file system for context information for each recently updated file, such as which patch or software package a DLL came from, which device a driver file is used for, etc., so that the information becomes actionable by the users.

The outline of the paper is as follows. In Section 2, we describe the design and implementation of AskStrider. In Section 3, we present several real-world cases in which users can benefit from the use of AskStrider. In Section 4, we summarize the paper and discuss future work.

## 2. Design and Implementation

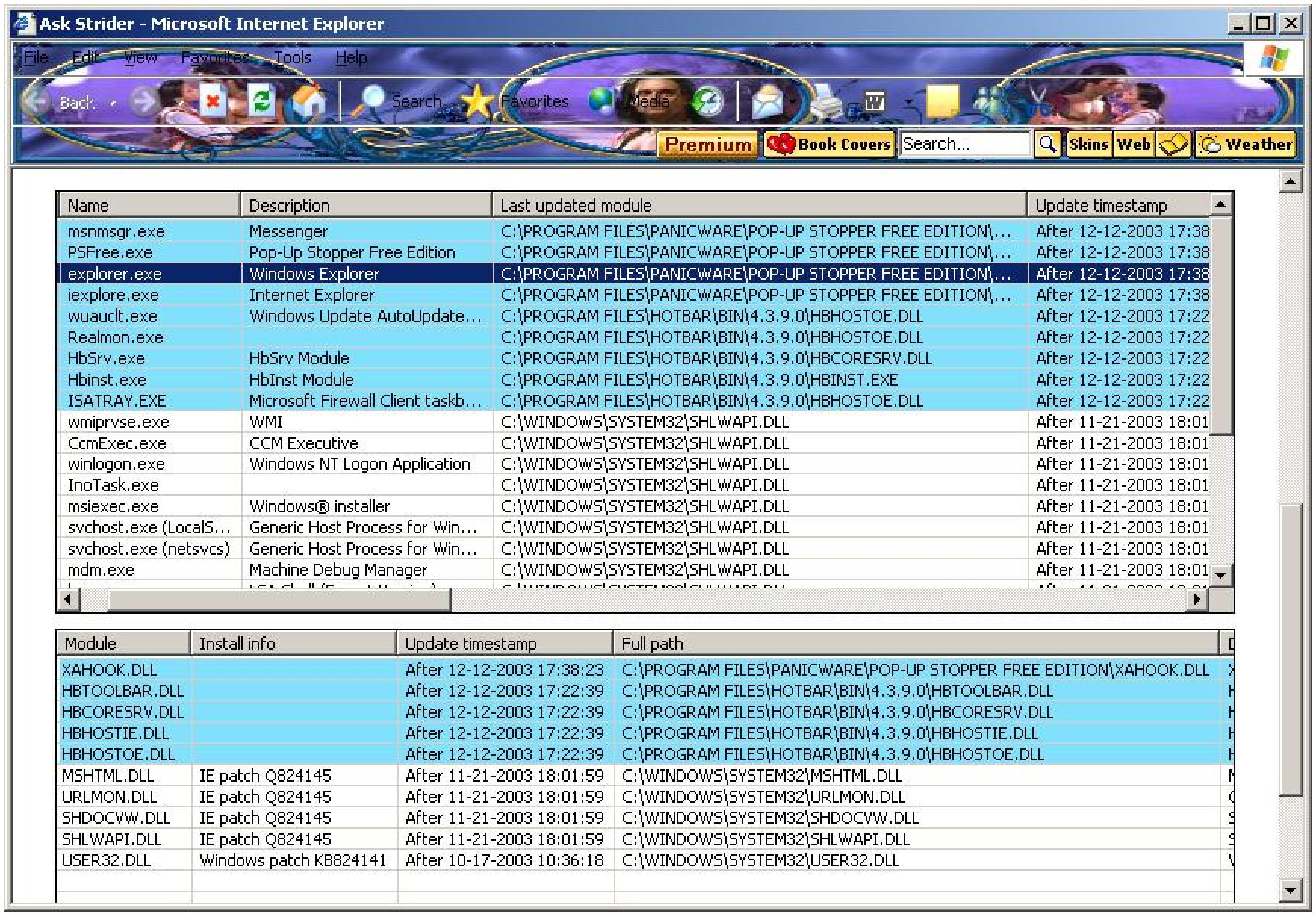
AskStrider is packaged in three different ways: (1) as a command line tool, it produces an XML output file that captures process/module dependencies in a hierarchical fashion; (2) as an ActiveX control for Web access, it displays the list of processes in the top pane and the list of modules loaded by the selected process in the bottom pane, as shown in Figure 1; (3) as an extension to Windows Task Manager, it adds the two panes below the regular display area of process statistics.

AskStrider consists of three main components: scanning for active components, ranking based on file age, and searching for context information.

### 2.1. Scanning for Active Components

In the first step of AskStrider, we query the operating system for all the running processes and their loaded modules, and all the device drivers currently loaded by the system. Then we look up the file associated with each such active component to obtain more detailed and/or user-friendly information such as version, description, company, etc.

On Windows machines, these queries are performed by invoking the Process Status Helper and Version Information API functions. For example, for process *"MsnMsgr.exe"*, AskStrider shows that its full path name is *"C:\Program Files\MSN Messenger\MsnMsgr.exe",* and one of its loaded modules is *"C:\Program Files\Microsoft Firewall Client\wspwsp.dll"* with a version "3.0" and a description *"Microsoft WinSock Proxy WS2.0 provider"*.



| Name | Description | Last updated module | Update timestamp |
|---|---|---|---|
| msnmsgr.exe | Messenger | C:\PROGRAM FILES\PANICWARE\POP-UP STOPPER FREE EDITION\... | After 12-12-2003 17:38 |
| PSFree.exe | Pop-Up Stopper Free Edition | C:\PROGRAM FILES\PANICWARE\POP-UP STOPPER FREE EDITION\... | After 12-12-2003 17:38 |
| explorer.exe | Windows Explorer | C:\PROGRAM FILES\PANICWARE\POP-UP STOPPER FREE EDITION\... | After 12-12-2003 17:38 |
| iexplore.exe | Internet Explorer | C:\PROGRAM FILES\PANICWARE\POP-UP STOPPER FREE EDITION\... | After 12-12-2003 17:38 |
| wuauclt.exe | Windows Update AutoUpdate... | C:\PROGRAM FILES\HOTBAR\BIN\4.3.9.0\HBHOSTOE.DLL | After 12-12-2003 17:22 |
| Realmon.exe |  | C:\PROGRAM FILES\HOTBAR\BIN\4.3.9.0\HBHOSTOE.DLL | After 12-12-2003 17:22 |
| HbSrv.exe | HbSrv Module | C:\PROGRAM FILES\HOTBAR\BIN\4.3.9.0\HBCORESRV.DLL | After 12-12-2003 17:22 |
| Hbinst.exe | HbInst Module | C:\PROGRAM FILES\HOTBAR\BIN\4.3.9.0\HBINST.EXE | After 12-12-2003 17:22 |
| ISATRAY.EXE | Microsoft Firewall Client taskb... | C:\PROGRAM FILES\HOTBAR\BIN\4.3.9.0\HBHOSTOE.DLL | After 12-12-2003 17:22 |
| wmiprvse.exe | WMI | C:\WINDOWS\SYSTEM32\SHLWAPI.DLL | After 11-21-2003 18:01 |
| CcmExec.exe | CCM Executive | C:\WINDOWS\SYSTEM32\SHLWAPI.DLL | After 11-21-2003 18:01 |
| winlogon.exe | Windows NT Logon Application | C:\WINDOWS\SYSTEM32\SHLWAPI.DLL | After 11-21-2003 18:01 |
| InoTask.exe |  | C:\WINDOWS\SYSTEM32\SHLWAPI.DLL | After 11-21-2003 18:01 |
| msiexec.exe | Windows® installer | C:\WINDOWS\SYSTEM32\SHLWAPI.DLL | After 11-21-2003 18:01 |
| svchost.exe (LocalS... | Generic Host Process for Win... | C:\WINDOWS\SYSTEM32\SHLWAPI.DLL | After 11-21-2003 18:01 |
| svchost.exe (netsvcs) | Generic Host Process for Win... | C:\WINDOWS\SYSTEM32\SHLWAPI.DLL | After 11-21-2003 18:01 |
| mdm.exe | Machine Debug Manager | C:\WINDOWS\SYSTEM32\SHLWAPI.DLL | After 11-21-2003 18:01 |

| Module | Install info | Update timestamp | Full path |
|---|---|---|---|
| XAHOOK.DLL |  | After 12-12-2003 17:38:23 | C:\PROGRAM FILES\PANICWARE\POP-UP STOPPER FREE EDITION\XAHOOK.DLL |
| HBTOOLBAR.DLL |  | After 12-12-2003 17:22:39 | C:\PROGRAM FILES\HOTBAR\BIN\4.3.9.0\HBTOOLBAR.DLL |
| HBCORESRV.DLL |  | After 12-12-2003 17:22:39 | C:\PROGRAM FILES\HOTBAR\BIN\4.3.9.0\HBCORESRV.DLL |
| HBHOSTIE.DLL |  | After 12-12-2003 17:22:39 | C:\PROGRAM FILES\HOTBAR\BIN\4.3.9.0\HBHOSTIE.DLL |
| HBHOSTOE.DLL |  | After 12-12-2003 17:22:39 | C:\PROGRAM FILES\HOTBAR\BIN\4.3.9.0\HBHOSTOE.DLL |
| MSHTML.DLL | IE patch Q824145 | After 11-21-2003 18:01:59 | C:\WINDOWS\SYSTEM32\MSHTML.DLL |
| URLMON.DLL | IE patch Q824145 | After 11-21-2003 18:01:59 | C:\WINDOWS\SYSTEM32\URLMON.DLL |
| SHDOCVW.DLL | IE patch Q824145 | After 11-21-2003 18:01:59 | C:\WINDOWS\SYSTEM32\SHDOCVW.DLL |
| SHLWAPI.DLL | IE patch Q824145 | After 11-21-2003 18:01:59 | C:\WINDOWS\SYSTEM32\SHLWAPI.DLL |
| USER32.DLL | Windows patch KB824141 | After 10-17-2003 10:36:18 | C:\WINDOWS\SYSTEM32\USER32.DLL |

**Figure 1. Screenshot of Web-based AskStrider. (The romance novel image at the top of the IE window was due to drive-by download of Hotbar. In this case, AskStrider highlights Hotbar and Pop-up Stopper as two recent additions to the machine.)**

## <u>Differentiating Multiple Instances with the Same Process Name</u>

Multiple process instances with the same name require additional information to tell them apart. For generic host processes such as *svchost.exe* and *rundll32.exe*, we query the Process Environment Block (PEB) to obtain additional *"CmdLine"* information, which includes the command-line switches and input that were used to start each process. For example, there are usually four *svchost.exe* processes on a machine. The

*CmdLine* information can distinguish them as *C:\WINDOWS\system32\svchost.exe -k rpcss, -k netsvcs, -k NetworkService*, and *–k LocalService*, respectively. Optionally, additional *"Services"* information can list all the Win32 services that are running inside each *svchost.exe*.

As another example, the three different actions *Control Panel→Add or Remove Programs, →AP Monitor,* and *→Firewall Client* would launch three processes with the same name *rundll32.exe*, which is a generic host process for running a DLL as an application. The *CmdLine* information can clearly associate each of the processes with its corresponding UI window. Alternatively, we can use the caption information of the window owned by each process to distinguish them.

Command prompt windows *cmd.exe* and Internet Explorer (IE) browser windows *iexplore.exe* are another two common examples of multiple instances with the same name. For *cmd.exe* processes, we use additional "Current Directory" information to distinguish them. For *iexplore.exe* processes, since multiple browser windows opened through *File→New→Window* belong to the same process, we use the concatenation of the captions of these windows to differentiate among multiple *iexplore.exe* processes.

## 2.2. Ranking Based on File Age

We use the following definitions throughout the paper: on a per-machine basis, the *age of a file* is the elapsed time since the file was last updated on the machine. The *age of a module instantiated from a file* is equal to the age of the file. The *age of a process* is the lowest age among the modules loaded by the process. The *age of the System process* is the lowest age among the driver modules that are currently loaded.

In the second step, we query the operating system for the age of each file associated with an active component. Then we compute the age of each process and use that to sort the process list, with the youngest process at the top. The module list is similarly sorted.

### Calculating File Ages

One obvious approach to calculating file ages is to use the creation or modification file timestamps. However, the *"Created"* and *"Modified"* file timestamps on Windows machines may not provide reliable information of when a file was last updated on the local machine. For example, a system DLL that comes with a patch usually have those two timestamps permanently tied to the software development process, independent of when the patch was applied to each individual machine.

To obtain reliable file age information on a given machine, we query the file change log of the System Restore service. System Restore uses a file system filter driver to monitor and log update operations made to files with selective extensions [SRM], which include all the "program" (i.e., non-data) file extensions and so suffice for our purpose. Whenever necessary, System Restore saves a copy of the to-be-modified file in the sub-folder under *C:\System Volume Information* that represents the latest restore point. Such pre-images are used at rollback time to restore the "program portion" of the file system.

Unfortunately, System Restore does not maintain the timestamp for each individual logged file update operation. Since the *Created* and *Modified* file timestamps of the pre-images may not correctly reflect the update timestamps, we use the following algorithm to calculate approximate file ages:

- For all the files with update operations recorded in a restore point sub-folder, their update timestamps are lower-bounded by the *Created* timestamp of the sub-folder and upper-bounded by the timestamp of the next restore point's sub-folder. Since System Restore typically creates a restore point every 24 hours if no manual or program-initiated restore points have been taken, this provides a minimum resolution of 24 hours for file update timestamps.
- To provide a finer granularity for update timestamps, our implementation scans the entries in the file change log of each sub-folder in their recorded sequence and examines the timestamps of their corresponding pre-image files, if available. Timestamps that are earlier than the sub-folder creation time are discarded because they cannot possibly reflect the actual update times. Timestamps that are not monotonically increasing are also discarded. The remaining ones are used to provide narrower ranges for update timestamps. Moreover, the *Created* timestamps of those folders that have a corresponding "folder creation operation" log entry can often provide additional timestamps.
- The age of any file that has been updated at least once since the earliest available restore checkpoint is calculated based on the lower-bound timestamp of the range for the latest update. The age of all other files is defined to be infinite.

## Ranking Based on File Age

The display of processes and modules in AskStrider is ordered by their ages. In the top pane of process list, "younger" processes are displayed closer to the top. In the bottom pane of module list for the selected process, younger modules are displayed first. To further highlight recent changes to the system, processes and modules that are younger than one week are highlighted in color (see Figure 1); those with an infinite age are displayed in a separate pane that shows up only through a right-click menu selection.

The appearance of the AskStrider display naturally represents the "stability" of the system: if there has not been any update for a long time, both panes will contain only a small number of entries and none of them will be color-highlighted, which signifies that the system is stable. In contrast, if any of the widely shared OS components has been recently updated by a patch, the top pane will show a large number of highlighted young processes, which suggests that the system may be unstable. As time goes by, those processes will age and eventually disappear from the top pane.

### 2.3. Searching for Context Information

In some scenarios, the highlighted processes and modules alone provide sufficient information on what the user should do or investigate next. For example, they may reveal the name of the application that the user should try to uninstall to fix the problem. In other scenarios (some described in the next section), additional context information is required to make the displayed results actionable. In the third step, AskStrider gathers such information from the local file system.

**Patch Information Mapping**

If an application stops working after several patches are applied, it is important for the user to find out whether any of the patches may be responsible and, if so, which one. Today, given a recently patched DLL or driver file, it is a non-trivial task to find out which patch installed the file. Many patches do not provide a well-documented manifest file that specifies the list of target files and versions. Even for those critical security patches that do provide such a manifest in the *mssecure.xml* file that is downloaded as part of the Microsoft Baseline Security Analyzer [MBSA], some of the target files may not be updated on certain machines because they may have already acquired the right versions of those files from other patches.

Fortunately, most of the important patches back up the replaced files to allow selective removal of patches. Our approach is to query those backed up information on the local machine to determine which patch was the last one to update a particular file and when. For Windows OS patches, we search through the *C:\Windows\$NtUninstall…$* folders and the Windows Update patch history XML file on the local machine. For IE patches, we query related Registry entries and parse the patch INF files under the *C:\Windows\inf* directory. We are in the process of implementing the mapping information for Office, SQL, etc. patches.

**Change log-based Grouping**

Today, software installation programs are not obligated to specify the set of all programs that they install, to provide an uninstall option, or to cleanly remove all installed components upon removal. When an application is partially uninstalled, it can be very difficult for the users to determine which application a left-over module belongs to. For example, an application ABC may install most of its programs in *C:\Program Files\ABC*, but also install some DLLs under *C:\Program Files\Common Files* and some drivers under *C:\Windows\System32*. Suppose the software removal program removes *C:\Program Files\ABC*. The left-over DLLs and drivers then become orphans that give users little information on where they came from.

Fortunately, the proximity of file update operations in the System Restore file change log provides an opportunity for heuristic application grouping. Specifically, if a recently updated file does not belong to any known applied patch, AskStrider will query the file change log for other files that were updated around the same time and try to deduce the name of the application to which the file potentially belongs. Sometimes, installation programs that are System Restore-aware take a restore point before an installation begins and write an informative string (such as "Installed Windows Messenger 5.0") as the description for the restore point, which AskStrider can extract as useful context information.

**Device and Driver Information Coupling**

Our experience with extracting driver information shows that the description associated with a driver file sometimes does not provide useful information to the user. For example, the file *wlluc48.sys* on a laptop has a generic description of "NDIS 5.1 Miniport Driver" and the file *cmbp0wdm.sys* has a cryptic description of "PC/SC IFD handler for CardMan 4000".

To provide more friendly information to the users, we identify the device that a driver is used for, whenever possible, and augment the driver file information with the device description. For example, *wlluc48.sys* is used for "Toshiba Wireless LAN Mini PCI Card" and *cmbp0wdm.sys* is used for "Omnikey AG CardMan 4000 PCMCIA Smart Card Reader". Such information is clearly much more useful for users to understand the purpose of the drivers and to troubleshoot problems.

## 3. Case Studies

### 3.1. Drive-by Downloads, Spyware, and Snoopware

Detection of unwanted software is typically done through a signature-based approach; examples include the anti-virus software and the Ad-aware spyware detection software [AA]. However, the proliferation of different types of unwanted software and their variants make it difficult for the signature-based approach to always have an up-to-date, complete coverage.

AskStrider provides a complementary approach by identifying all unwanted software in a uniform way. It can help speed up the discovery and reporting of unwanted software and hence the update of signatures. Since AskStrider typically takes only 5 to 45 seconds to scan a system, it can be invoked every morning when a user comes to work or every night when a user finishes her browsing activities; if anything suspicious shows up, more thorough signature-based scanning can follow. (To contrast, a full scan using Ad-aware on a laptop with a 40GB disk takes about seven minutes.)

**HotBar Drive-by Download:**

The default Internet-zone security setting of the IE browser is set to *Medium* to protect user machines against unauthorized download of ActiveX controls. But many users change it to *Low* to avoid getting security warning dialog boxes and broken displays at certain Web sites. By doing so, however, the users allow "drive-by downloads"; that is, ActiveX controls can be downloaded and installed on the machines without explicit consent from the users. Later on, if the users suspect that such software may be responsible for serious machine performance degradation or unreliability, it is in general very difficult for them to even know what software has been installed.

For example, drive-by downloading happens when a browser with *Low* security setting is directed to http://www.romancebookcovers.com/hotbar/hotbar02.html. Code will be automatically downloaded, even if the user clicks the "No" button on the dialog box. After a while, it will make the top of the IE and Control Panel windows look like the cover of a romance novel (see Figure 1). It will also monitor browsing activities and serve pop-up ads.

By running AskStrider, a user can clearly see one or two new processes appear near the top, which are instantiated from newly installed EXE files under *C:\Program Files\Hotbar*. Other processes such as IE, Explorer, Outlook, Instant Messenger, and Word, etc. also appear near the top because five DLLs from that same directory have been injected into these processes in various combinations.

In this case, the company has kindly provided an uninstallation option at *Control Panel→Add or Remove Programs*. Once the user identifies that the installed software was from Hotbar, she can remove the software and then re-run AskStrider to make sure that it is indeed no longer running.

**Internet Washer Unauthorized Download:**

Unauthorized download can also happen when a user clicks on a pop-up window that has a serious blue-screen look with a "System Update" title (see http://a.rn11.com/yh/pu/yhgeouspu2.htm). The download of a program called Internet Washer will begin even if the user clicks the "Cancel" button, which is in fact a fake button on an image. AskStrider shows that a new process *C:\Program Files\Internet Washer Pro\iw.exe* is started and two new DLLs *C:\Program Files\Httper\Httper.dll* and *C:\Program Files\Zipclix\Zipclix.dll* are loaded into every IE process.

By going through *Add or Remove Programs* and looking for these names, the user can try to uninstall these three applications. In several of our experiments, though, the uninstallation simply removed the entries from *Add or Remove Programs* and left the software running. Without the information from AskStrider, the user would have had no idea of what to look for and would have been uncomfortable removing an application disguised under a name like "Httper" that makes it look like an OS component. (A further search on the Web revealed that "Httper" and "Zipclix" can potentially hijack browser sessions, serve pop-up ads, and be directed by their controlling server to download and execute arbitrary code as a self-updating feature.)

**Gator PrecisionTime:**

Precision Time [PT] is software that synchronizes a computer's clock with the U.S. Atomic Clock. Users can obtain the software for free but, in return, they allow the providing company to deliver pop-up ads based on their online surfing behavior.

Invoking AskStrider several minutes after the installation of Precision Time revealed that four new processes were started and two DLLs were injected into every IE process. The software provides an uninstallation option at *Add or Remove Programs*, but it does not always work. In one experiment, AskStrider showed that three of the processes were left running after an unclean uninstallation (followed by a reboot) without any error message. A spyware removal program was eventually used to remove those left-over components.

**iOpus STARR:**

STARR [ST] belongs to the category of commercial snoopware that can be covertly installed on a local or remote machine to record keystrokes and screenshots. Invoking AskStrider after the installation of STARR reveals that a new process named *wsys.exe* was started and a new DLL named *wsys.dll* was injected into most of the processes. It would have been very difficult to identify these two components were it not for the file age-based ranking of AskStrider because of the way the two files were named and because they were both placed into the *C:\Windows\System32* directory.

**SoftActivity Activity Logger:**

Activity Logger [AL] is another commercial snoopware that records to a log file the visited URLs, typed text in email, chat, and other applications, programs user runs, etc. and silently emails the log file when computer goes online. AskStrider similarly shows that a new process was started and a new DLL was injected into many other processes; both have a full path name under *C:\Program Files\Activity Logger*.

In the remainder of this paper, we describe five real-world cases that would have greatly benefited from the use of AskStrider. To preserve privacy for the actual users, we do not use their real names.

## 3.2. Pop-up Stopper

AccuRadio (http://www.accuradio.com/) has been Scott's favorite online radio Web site. When he selects a channel by clicking on an image, the browser opens another window to play the music. One day when he came home after work, he discovered that AccuRadio no longer worked on his home machine: the second window simply flashed for a second and disappeared. However, clicking on an image provided by a sponsor on the same page correctly brought up another window.

Scott invoked AskStrider and discovered that a new process called *PSFree.exe* appeared at the top and several other processes including IE had loaded a new DLL called *XAHook.dll*. The fact that they came from *C:\Program Files\Panicware\Pop-up Stopper Free Edition* suggested to Scott that his kid might have installed a program for killing pop-up ads, which was indiscriminately killing the pop-up radio-channel windows. By locating the pop-up stopper settings menu and disabling the blocking action, Scott was able to get AccuRadio to play music again.

## 3.3. MSN Explorer Sign-in Failure

Barry uses MSN Explorer to establish dial-up connections from his vacation home every weekend. For some reason, the sign-in process started failing this week. Barry suspected that the failure had something to do with the security patch that he had installed from Windows Update earlier in the week. So he invoked System Restore to roll back the machine to last weekend's state, at which all dial-in attempts through MSN Explorer were known to be successful. The rollback did fix the problem and that “confirmed” Barry’s suspicion. Unfortunately, the security patch was removed by the restore operation and Barry was unwilling to install the “bad” patch again. His machine would now remain vulnerable to exploits.

By running AskStrider, Barry would have realized that the MSN Explorer process did not load any files recently updated by that security patch at all, so the sign-in failures were most likely due to problems with configuration settings. We have used the Strider Troubleshooter [WVD+03] to discover that the real root cause was a bad proxy setting: the corporate proxy setting was used by MSN Explore in a non-corporate environment; removing that setting would have resolved the issue and allowed the machine to be protected by the security patch.

This case demonstrates that, by combining process/module dependency tracking, file age-based process and module ranking, and file-to-patch mapping, AskStrider can be an efficient and effective tool for pointing the right direction in the diagnosis of potential patch-related problems by providing either positive or negative evidence.

### 3.4. AP Monitor Does Not Display Available Wireless Access Points

Nick has a Toshiba 9000 series laptop and he sometimes uses *Control Panel→AP Monitor* to find out the available wireless Access Points (APs) in his proximity. One day the AP Monitor stopped working: the display was simply empty and there was no error message, even though all his colleagues’ laptops could still see the APs.

Nick invoked AskStrider to see if AP Monitor’s *rundll32.exe* process was using any recently updated files. The answer was negative. However, a device driver named *wlluc48.sys* had recently been upgraded to version 7.43.0.9 and so appeared at the top. Since the device/driver coupling information showed that *wlluc48.sys* was a driver for the wireless LAN card, it occurred to Nick that perhaps the new version of driver was incompatible with the APs. He then launched the Device Manager, entered the wireless network adapter property page, and chose the “Roll Back Driver” option. The rollback solved the problem, and AskStrider showed that the version of *wlluc48.sys* is now 7.16.0.189.

### 3.5. Cannot Open Microsoft Word Documents

Chris has been able to double click on a Word document inside Windows Explorer to launch Word to open the document. One day, that functionality suddenly stopped working: double-clicking on *TestPlan.doc* would pop up an error dialog box, saying that “The file TestPlan.doc is not available”. Performing a traditional symptom-based search by typing: *“Microsoft Word” and “The file” and “is not available”* in a company internal support database would generate a list of 60 articles, none of which provides useful information to solve the problem.

By running AskStrider, Chris noticed that *winword.exe* appeared near the top of the list because it had loaded the following recently updated DLL:

*C:\Program Files\Norton AntiVirus\DJSMAR00.DLL*.

By adding *"Norton AntiVirus"* to the search string, Chris received new query results from the support database that contained only six articles, and the second one explained a known incompatibility issue between old versions of Norton AntiVirus and the latest version of Word.

Since Chris did not really need Norton AntiVirus, he decided to uninstall it. The uninstallation appeared to be successful, but the problem remained. Chris ran AskStrider again and discovered that *winword.exe* still had the Norton AntiVirus DLL loaded. Chris then used the Msizap utility from http://msdn.microsoft.com/library/?url=/library/en-us/msi/setup/msizap_exe.asp to completely remove Norton AntiVirus, and that solved the problem.

This case demonstrates that the state information provided by AskStrider can be used to augment the search strings in symptom-based troubleshooting to improve the search results. Also, AskStrider can easily detect unclean uninstallations, which commonly mislead users and support engineers in their troubleshooting efforts.

### 3.6. TurboTax Installs Additional Drivers

By invoking AskStrider on her home machine, Wendy discovered that a new driver called *CdaC15BA.SYS* and a new process called *CDAC11BA.EXE* were running, and both were installed into the *C:\Windows\System32\drivers* directory. The driver file does not give any information and the EXE file only indicated that it was from Macrovision. But Wendy was pretty sure that she did not install any software from that company.

Fortunately, the context information showed that these two files were created around the same time when a lot of files were created under *C:\Program Files\TurboTax\Deluxe 2002* and a folder called *C:\C_DILLA\SafeCast Product Licenses* was updated. A Web search based on those information revealed that TurboTax 2002, which Wendy installed and uninstalled a few days ago, additionally installed the C-Dilla product license protection software without user knowledge, and that software was left running even after TurboTax was uninstalled.

## 4. Summary and Future Work

We have demonstrated the power of correlating volatile process information with persistent-state context information and change history to answer important questions about unwanted software installation, patch impact, and unclean software removal, which used to be very difficult to answer. We have implemented the idea in the AskStrider tool and showed that we can already go pretty far with the limited and scattered information stored on today's Windows machines.

We will be pursuing future work in three directions to make AskStrider a ubiquitous tool for systems management, support, and troubleshooting: first, we are investigating ways to gather more context information in order to provide a more complete coverage for file mapping.

We also recognize that the utility of AskStrider is enhanced by tuning its output to more accurately inform the end user. Given that, we will be investigating with the appropriate experts the cognitive issues surrounding the presentation of information in the appropriate form to various user types. For example, this may result in the creation of a simplified interface that will display only high-level information that can be understood by novice PC user to enhance the user's ability to identify sources of instability on his or her machine. We plan to explore the possibility of a pilot rollout in a commercial deployment.

Given that there is such a strong importance placed on the integrity of the information contained in the System Restore files, we must also investigate the methods by which we can secure this information; otherwise it becomes possible to spoof AskStrider, and would offer an inviting target to writers of spyware and malicious code. We will be investigating a number of solutions which may include encryption/digital rights management or a combination of hardware and software to provide a secure solution.

Finally, we will provide feedback to the Windows software development organizations by demonstrating the importance and benefits of recording precise timestamps of all file update operations and simplifying the retrieval of context grouping information so that AskStrider can provide more accurate information in a more useful way.

## References

[AA] LavaSoft Ad-aware spyware removal program, http://www.lavasoft.de/software/adaware/.
[AL] SoftActivity Activity Logger, http://www.softactivity.com.

[MBSA] Microsoft Baseline Security Analyzer, http://www.microsoft.com/technet/treeview/default.asp?url=/technet/security/tools/mbsahome.asp.

[PT] Precision Time, http://www.claria.com/products/index.html.

[ST] iOpus STARR Computer & Internet Monitoring Software, http://www.iopus.com/starr.htm.

[SR] 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.

[WVS03] Y. M. Wang, C. Verbowski, and D. R. Simon, "Persistent-state Checkpoint Comparison for Troubleshooting Configuration Failures," in *Proc. Int. Conf. on Dependable Systems and Networks (DSN)*, 2003.

[WVD+03] Y. M. Wang, C. Verbowski, J. Dunagan, Y. Chen, Y. Chun, H. J. Wang, and Z. Zhang, "STRIDER: A Black-box, State-based Approach to Change and Configuration Management and Support," in *Proc. Usenix Large Installation Systems Administration (LISA) Conference*, pp. 159-171, October 2003.