# A Large-Scale Study of File Prevalence

Michael Jonas
Arizona State University
Tempe, AZ 85281
mjonas@asu.edu

Cormac Herley
Microsoft Research
One Microsoft Way
Redmond, WA 98052
cormac@microsoft.com

Jack W. Stokes
Microsoft Research
One Microsoft Way
Redmond, WA 98052
jstokes@microsoft.com

Scott Sovine
Microsoft Corporation
One Microsoft Way
Redmond, WA 98052
ssovine@microsoft.com

Magdi Morsi
Microsoft Corporation
One Microsoft Way
Redmond, WA 98052

## ABSTRACT

We report on a large-scale telemetry project. The system gathered file reports from 20.7 million machines that ran the Windows Upgrade Advisor over a period of 13 months. The data gives a unique insight into file prevalence across a web-scale population. Among interesting findings is that over half of all files reported are seen on a single system and never seen again. There is considerable innovation, with new files being generated in the population at a rate of thousands per day. The client systems also exhibit extraordinary diversity with the least active and most active machines reporting very different file distributions. The data points to the extreme difficulty of building an exhaustive library of existing software. The longtail nature of the distribution shows that even 20 million systems is nowhere near enough to generate a comprehensive picture of the universe of Windows software. The largest publicly available library, the National Software Registry Library, contains a mere 0.26% of the files reported by our clients.

## Categories and Subject Descriptors

K.6.5 [**Computing Milieux**]: MANAGEMENT OF COMPUTING AND INFORMATION SYSTEMS—*Security and Protection*

## General Terms

Measurement, Security

## Keywords

whitelist, file prevalence, malware detection

## 1. INTRODUCTION

The Windows ecosystem may be the largest and most diverse among the major Operating Systems. Estimates from industry analysts are that about 340 million Windows PC's ship every year. Despite the scale of the installed base we have relatively few measurements on this vast population of machines.

In this paper we report on a large-scale telemetry project that gathered data from over 20 million reporting machines. The telemetry was received from the Windows Upgrade Advisor, a compatibility tool that advises users about whether their hardware is capable of upgrading to Windows 7. The tool gathers certain data on the machine and its installed programs. While no information that identifies the user or compromises privacy is gathered, the data gives us several insights into the what a web-scale population of users has installed. The telemetry is limited and reports consist primarily of SHA-1 hashes, and other metadata, of executable files that were installed through Add-Remove Programs or the Microsoft Installer. However, we do gain an insight into the spread of systems, share of OS and software versions, file prevalence and the longtail distribution of files in the ecosystem.

One surprising finding is the large number of single-instance files reported. One half of all files are reported by a single system and never seen again. The proportion of such single-instance files drops with scale, but does so at a surprisingly slow rate. When the population is 100,000 systems 99% of reported files have already been seen, by a millon systems this increases to 99.7% and 99.82% by 20 million. However the rate of increase is such that it appears unlikely that an exhaustive list can be built. This is definitely a long tail phenomenon.

One might imagine that the large population of single-instance files are generated by obscure software vendors. This appears not to be the case: the companies that produce the most popular applications at the head of the distribution (*e.g.*, Microsoft, Symantec, Sun and Adobe) are also very well represented in the tail.

There is a great deal of diversity among the reporting systems. The most active quintile of systems have on average $5.2\times$ as many programs installed as the least active quintile, and $6.1\times$ as many executable files.

We examine the intersection between our dataset and the National Software Registry Library (NSRL), an annually updated collection of 21 million file hashes maintained by NIST. Surprisingly we find small overlap. Only 0.26% of the files in our dataset were also in the NSRL, and most of these were found in the first five weeks of reporting. This suggests that making any such library exhaustive is almost impossible. The innovation rate is extremely high. The distribution of files over systems is a longtail phenomenon, it appears that even at 20 million systems we are nowhere close to having

"seen everything."

The remainder of this paper is as follows. Section 2 describes the methodology used to gather the data and important characteristics of the dataset. Section 3 explores system-level properties such as OS, language, number of programs and files installed, *etc*. Section 4 examines the distributions of programs and files, and the updating habits of users. Section 5 explores popularity or ubiquity of files, the distribution of rare and single-instance files in the population and the innovation rate of new files. Section 6 examines the overlap between our dataset and the National Software Registry Library and a large collection of malware signatures.

## 2. METHODOLOGY

In this section, we describe the methodology used to obtain and collect the data analyzed in this study. All of the raw telemetry data is generated by users running the Windows 7 Upgrade Advisor software. Prior to launching Windows 7, Microsoft deployed the "Windows 7 Upgrade Advisor" [14] website giving users with computers running either the Windows XP or Windows Vista operating system an opportunity to download an application, which we call *Upgrade Advisor*, to evaluate whether or not their computer can be safely upgraded to the Windows 7 operating system. The reason for a user to run this application is so that she does not start the Windows 7 upgrade process and hit a problem without a solution.

The reporting interval of the data analyzed in this study is 13.5 months ranging from June 2010 to July 2011. Files are represented by a File ID which is equivalent to the SHA-1 hash for files up to 30 MBytes. For files over 30 MBytes, the File ID is the SHA-1 of the first 30 MBytes of the file. It should be noted that we are not using cryptographic properties of the SHA-1 hash; the SHA-1 serves as a unique identifier only.

The Upgrade Advisor checks all of the hardware including the CPU (central processing unit), memory, and available disk drive space to determine if the computer meets the minimum requirements for running Windows 7. In addition, the Upgrade Advisor checks all of the peripheral hardware such as the printers, mice, keyboard, monitor, etc. to determine if they are supported by Windows 7. Finally, the Upgrade Advisor checks all installed applications to see if these programs would safely run on the new operating system.

There are many different ways to install a program on a computer. Extremely simple executable programs may be copied directly to the computer's hard drive. However, most programs are installed using the *Add/Remove Program* (ARP) functionality in the core Windows Operating System or a separate installer, such as the Microsoft Installer (MSI). The telemetry data includes several program installation sources detected by the Upgrade Advisor which are specified in Table 1. The majority (73.6%) of the programs were installed using ARP, and the MSI was used by 20.2%. While most of the programs were installed either by ARP or MSI, the Upgrade Advisor attempts to determine if other programs were installed on the machine using other methods such as copying programs or using installers other than MSI. The set of programs identified by this alternate detection method is labeled as *File* (6.0%) in Table 1. There is a separate mechanism to detect when OEMs (original equipment manufacturers) bulk install software on new computers. A very small percentage of applications (0.086%) were installed in this manner by the manufacturer. If the Upgrade Advisor detects that an application was installed using any of these methods,

| Source | Percent |
|---|---|
| Add/Remove Program | 73.752 |
| Msi | 20.175 |
| File | 5.987 |
| Oem | 0.086 |

**Table 1: Installation sources for the programs.**

| File Type | Percent |
|---|---|
| .exe | 98.08 |
| .sys | 1.59 |
| .cpl | 0.33 |

**Table 2: File types in the Windows telemetry data.**

it transmits a collection of telemetry data to a backend web service identifying the program including the version (e.g. Adobe Acrobat Reader v7.0, Firefox v3.6), and the files contained with this version of the program. Each program includes a collection of individual executable files that are associated with the program. It should be noted that the files are not transmitted to the web service. Instead an identifier is sent which specifies a unique mapping from a versioned program ID to the IDs of all files installed by the program. This program and file telemetry is analyzed in detail in Section 4. The telemetry data also contains information regarding the individual system (i.e. computer) used to install the program; the system data is described in Section 3.

Once the backend web service receives the telemetry report, the data collected from the user's machine is analyzed, and the user is notified via the webpage whether or not the computer can be safely upgraded to Windows 7. If the Upgrade Advisor determines that the user cannot upgrade, it attempts to return the diagnosis to the Upgrade Advisor webpage displayed in the user's browser alerting them to the potential conflicts. If the conflict can be resolved, such as by adding more memory or buying a new printer for example, the user can revisit the Upgrade Advisor webpage at a later date to determine if the conflict has been successfully resolved.

The distribution of file types collected in the dataset is stored in Table 2. For the most part the Upgrade Advisor records only the top-level, user mode application files (i.e. .exe) (98.08%) but not any dynamically linked library (.dll), ActiveX Control (.ocx) or other executable types of executable files. The data includes a small amount (1.59%) of kernel mode drivers (i.e. .sys files) and a tiny fraction (0.33%) of Windows control panel files.

In the remainder of this paper, we analyze the telemetry data in detail. It should be appreciated that we do not have direct access to files; they reside on the user's remote computer. Also it should be noted that this telemetry data has been collected primarily to facilitate the rollout of Windows 7. Many things that would be interesting to report are not recorded. The schema cannot be changed and the purpose of this paper is to reach meaningful conclusions based on the existing data received from users contemplating upgrading to Windows 7.

**Privacy Concerns**: The Windows 7 Upgrade Advisor data collection is covered by the corresponding privacy policy [15]. All authors on this study were employed by Microsoft during the period when they had access to this data; four authors were full-time employees, and the fifth author was working as a summer intern. The

data collected by the Upgrade Advisor does not directly include any Personally Identifying Information (PII). While the data does contain information such as a machine ID and programs names, no data is collected which allow us to identify the owner of the computer. The machine ID is simply a randomly generated GUID. No IP address or path information of files are in the dataset.

# 3. SYSTEMS

A total of 20.7 million computers reported during the 13.5 month observation period. We study some basic statistics of the reporting computer population in this section.

## 3.1 Operating System and System Properties

The composition of operating systems on the reporting machines is given in Table 3, with approximately 45% of machines having XP Professional and 26% having Vista Home Premium. Aggregating by OS type we find that 47.91% of machines were running some version of XP, 37.43% were running some version of Vista and 14.53% were running some version of Windows 7. Recall that the triggering event for reporting was running the Windows 7 compatibility testing tool which evaluates whether or not the machine is capable of running Windows 7. Running the tool does not necessarily imply that the user installed Windows 7. We do not have a measurement of what fraction of users who ran the compatibility tool found that their machine was indeed Windows 7 capable, or of the fraction that proceeded to install. There was no significant change in the mix of OS's during the period. That is, the fraction of reporting machines that were running XP Professional for the first two months of the interval, 45.3%, was approximately equal to the fraction running it at the end of the interval, 44.1%. This makes sense: the two main reasons that this fraction might change would be if XP users upgraded to Vista and then subsequently upgraded to Windows 7, or if XP users were upgrading at a different rate to Vista users. The first possibility seems remote: we assume that few users would upgrade to Vista, since Windows 7 had already been available for 8 months by the beginning of our observation interval. This allows us to conclude with high confidence that there was no significant difference in the rates at which XP users and Vista users were upgrading.

One curiosity observed in Table 3 is that 13.94% of the systems are already running some version of the Windows 7 operating system. Why are these users visiting the Windows 7 Upgrade Advisor website if they are already running Windows 7? First, there is no restriction on running the Upgrade Advisor on Windows 7. Some users will run the program before adding a new feature to ensure that they have a machine that can handle it. For example, someone going from "Windows 7 Starter" to "Windows 7 Professional" will run the program to make sure that their hardware will work. Similarly, some users with Professional will run it before upgrading to Ultimate, and so on. It is very rare to see a top line OS running the tool. The other reason users run the Upgrade Advisor on Windows 7 is to diagnosis existing compatibility issues.

Table 4 shows the major language code of the machine. This code determines various system settings for the User Interface (UI). Language is the main setting affected by the code, though other localization factors, such as how date and time are presented also depend on it. The population is fairly diverse with just under 50% coming from machines with English as the UI culture. German, Chinese and Spanish are the next largest populations of reporting machines, accounting for between 6 and 8% each. Note that this indicates the fraction of systems running the Upgrade Advisor. This almost certainly indicates interest or intent to upgrade to Windows 7. However the percentages in Table 3 are probably only approximately indicative of the relative numbers of Windows machines in the various language regions. For example, it is quite likely that users from developed markets, like Germany, upgrade more quickly than those from developing markets like Brazil.

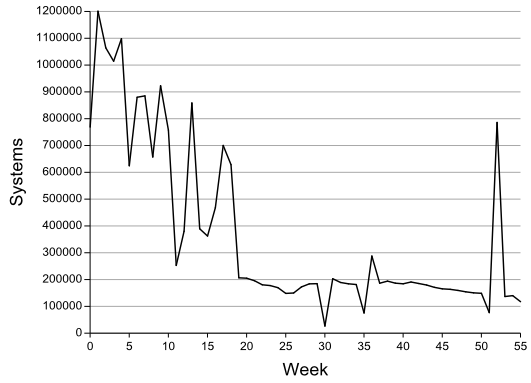
| OS version | Percent |
|---|---|
| XPProfessional | 44.98 |
| VistaHomePremium | 25.80 |
| Win7Ultimate | 5.64 |
| VistaHomeBasic | 4.65 |
| Win7HomePremium | 4.47 |
| VistaUltimate | 3.52 |
| XPMediaCenter | 2.77 |
| VistaBusiness | 2.67 |
| Win7Professional | 1.47 |
| Win7Starter | 1.41 |
| Win7HomeBasic | 0.95 |
| Other | 1.67 |

**Table 3: Major OS versions present in the dataset with percentages. On aggregate 47.91% of systems were running some variant of XP, 37.43% Vista and 14.53% Windows 7.**

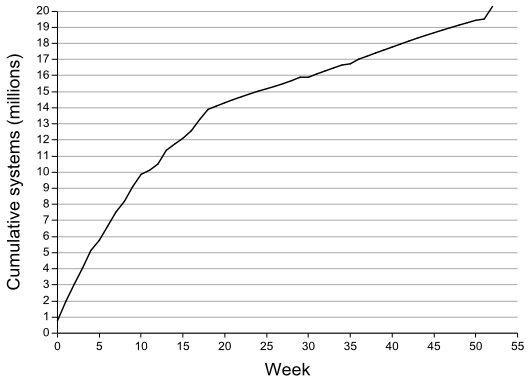
| Code | Language | Percent |
|---|---|---|
| en | English | 49.74 |
| de | German | 7.78 |
| zh-CHS | Chinese Standard | 6.48 |
| es | Spanish | 6.10 |
| fr | French | 4.85 |
| ru | Russian | 3.36 |
| ja | Japanese | 3.19 |
| pt-BR | Portugese (Brasil) | 3.06 |
| it | Italian | 2.55 |
| nl | Dutch | 2.18 |
| ko | Korean | 1.85 |
| zh-CHT | Chinese Traditional | 1.64 |
| tr | Turkish | 1.40 |
| sv | Swedish | 0.91 |
| pl | Polish | 0.91 |

**Table 4: Major UI culture percentages.**

Years ago, the CPU (central processing unit) manufacturers realized many users would need to access more the 4 gigabytes of memory and began to sell CPUs with 64-bit architectures. The most popular 64-bit architecture, X64, can either run a 32-bit or a 64-bit operating system since the 64-bit architecture is superset of the the standard 32-bit i386 architecture. The Upgrade Advisor telemetry two contains two fields related to 64-bit CPUs: X64 Capable and X64 Running. *X64 Capable* indicates if the CPU can run a 64-bit of the OS while *X64 Running* indicates whether or not the computer is actually running a 64-bit OS. For this dataset, 76.16% of the systems are capable of running a 64-bit operating system but only 12.01% of the systems are actually running a 64-bit OS. The reason only 12% of the machines are running 64-bit operating systems is probably because the computers running the Upgrade Advisor are mostly older computers manufactured when high-end computers did not have more than 4 gigabytes of RAM.




(a) The average number of machines reporting each week.




(b) The cumulative total number of machines.

**Figure 1: Arrival rate of new machines reporting, beginning in June 2010.**

| Day of Week | Percent |
|---|---|
| Monday | 14.12 |
| Tuesday | 13.60 |
| Wednesday | 12.64 |
| Thursday | 12.43 |
| Friday | 14.44 |
| Saturday | 17.01 |
| Sunday | 15.73 |

**Table 5: Systems running the Windows 7 Upgrade Advisor tool by day of week.**

Figure 1 shows the arrival rate of reporting machines. Figure 1 (a) shows the average number of machines per week, while (b) shows the cumulative population. It can be seen that machines arrive at a much greater rate at the beginning of the period. There is a propensity to run the Upgrade Advisor on weekends rather than weekdays as shown in Table 5. Over the observation period Saturdays were the busiest day, accounting for 17.01% of reports, while Wednesdays were the quietest with 12.64% of reports. Observe that 32.74% of reports arrive during weekends. This suggests that a majority of the reports are coming from consumer rather than enterprize systems.

## 3.2 Number of files and programs per system

We next investigate the number of programs and number of files per reporting system. These are shown in Figure 2 (a) and (b) respectively. The average number of programs installed is 110 and the median is 81. Recall that primarily programs that were installed using ARP or MSI are reported. The average number of files per system is 756 and the median is 566. In addition, only .exe, .sys and a few other filetypes were reported in the Upgrade Advisor dataset. Hence, .dlls, for example are not included. Thus the total file count can be expected to be higher than Figure 2 (b) indicates, by a factor of perhaps two or more. Observe that both the number of programs per system (Figure 2 (a)) and the number of files per systems (Figure 2 (b)) appear to have approximately log-normal distributions.

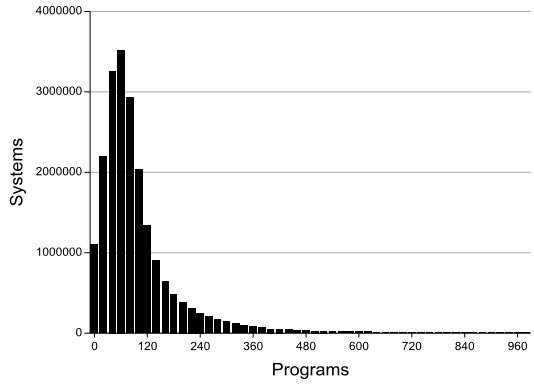
## 3.3 Space consumed on disk

The distribution of individual file sizes is discussed in Section 4. If we sum the sizes of all of the files reported by a particular machine we get an indication of the total size consumed by installed programs on the disk. This is shown in Figure 3. The average space consumed is 761 MBytes and the median is 499 MBytes. As with the programs and files per system (Figure 2) the distribution appears to have a roughly log-normal shape.

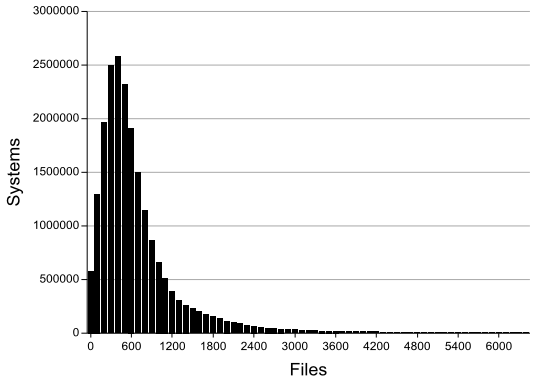
Again we caution that this includes only programs that were installed using ARP or MSI, and includes only the .exe files. It does not include any of the files associated with the operating system, any programs installed other than by ARP or MSI, or any user files such as documents, images, music files *etc*. Thus the actual space consumed on the disk by the installed programs could easily be a factor of two or more higher than this, and Figure 3 gives no guide whatever as to the total space occupied on the disk of the reporting system. It is nonetheless interesting to compare with the results of Douceur and Bolosky [9]. They found, in 1999, that 81% of the systems they studied had less than 2 GBytes of storage space. Thus the average system from 1999 would scarcely have space to store the programs that an average system has in 2010 (leaving no room for OS or user files).

## 3.4 Dividing into Quintiles

There is enormous diversity in the population of reporting machines. Reporting average, or median statistics, or even displaying histogram data can fail to give an accurate impression of diversity. This is particularly the case with heavytail phenomena where a small fraction of the population can have an outsized influence on the averages [17]. To this end we divide the population of reporting systems into quintiles by filecount. Quintile 1, Q1, contains the 20% of systems with the fewest files, Q2 contains the next 20% of systems and so on. Each of the quintiles contain (obviously) just over 4 million machines (*i.e.*, 20.7 million divided by five). These are effectively segments of Figure 2 (b). The quintile boundaries occur at 313, 475, 660, and 992 files. We will find, in Section 5, that there is enormous difference between the least and most active machines (*i.e.*, Q1 and Q5 respectively).




(a) Programs per system, the average is 110 and the median is 81.




(b) Files per system. The average is 756 and the median is 566.

**Figure 2: Histograms of the number of programs and number of files per system.**

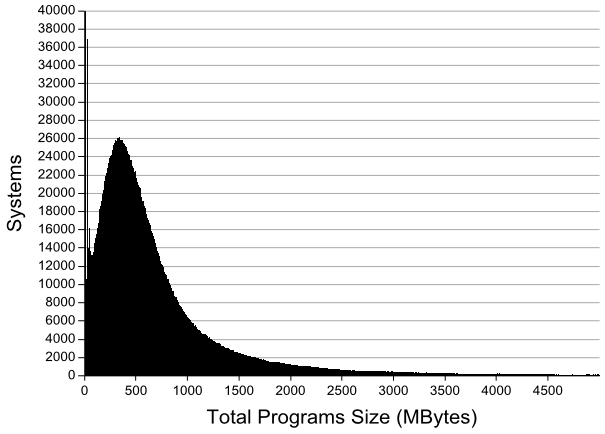



**Figure 3: Total size on disk of reported files. The median is 499 MBytes and the average is at 761 MBytes.**

# 4. PROGRAMS AND FILES

## 4.1 Arrival rate of files

A total of 36,883,613 files with distinct SHA-1 hashes were reported during the observation period. Figure 4 shows the arrival rate of distinct new files as a function of time over the observation period. Observe that the arrival rate drops very sharply during the first four weeks. This is due to the fact that as our dataset of file SHA-1's grows, more and more of the files that any given machine has to report have already been reported by another machine. For example, for the first machine to report every single file is previously-unseen. For the second machine to report, only files that the first machine didn't possess will count as previously-unseen. Thus, as the population increases, each new machine contributes only files that had never been seen on any of the previous reporting machines.

We graph the evolution of this reporting rate in Figure 5. This figure displays the average number of previously-unseen files that each reporting machine contributes as a function of time. Effectively Figure 5 is the arrival rate of files (Figure 4) divided by the arrival rate of the population (Figure 1 (a)). As can clearly be seen this drops very sharply at first, but plateaus quite rapidly. During the first week the average machine reports 5.3 previously unseen files, but by week five this has dropped to 1.6. However, it remains effectively unchanged for the remaining 50 weeks of the observation period. Thus, the rate at which systems have new files to report appears to hit a limit when the population reaches six million (*i.e.*, the population at week five from Figure 1 (b)). There is little reason based on Figure 5 to believe that at any population size we would achieve the point where we have "seen everything."

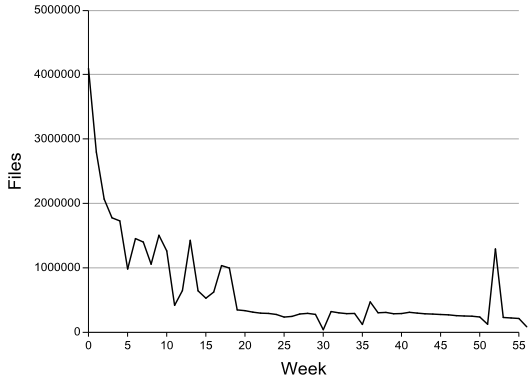



**Figure 4: Arrival rate of new files by week since June 2010.**

## 4.2 Distribution of file sizes

Figure 6 shows the histogram of individual file sizes. The average, 3.437 MBytes, is greater than the median, 0.402 MBytes, by a factor of 8.5. The large gap between the mean and median indicates that the distribution of file sizes is heavytailed. That is, even though the number of files having large sizes is small, the tail of Figure 6

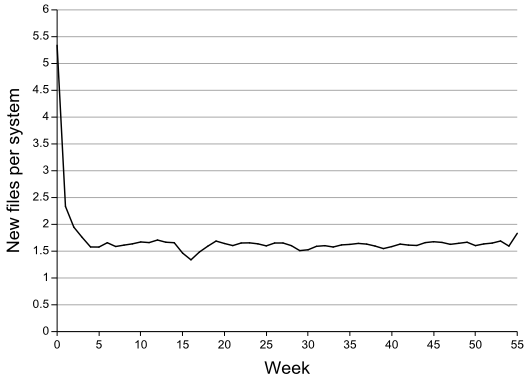



**Figure 5: Average number of previously-unseen files that a reporting machine contributes to the dataset as a function of time. Observe that this number drops very sharply at first, but then plateaus. During the first week machines contributed on average 5.3 such files each, by week 55 this had dropped to about 1.6.**

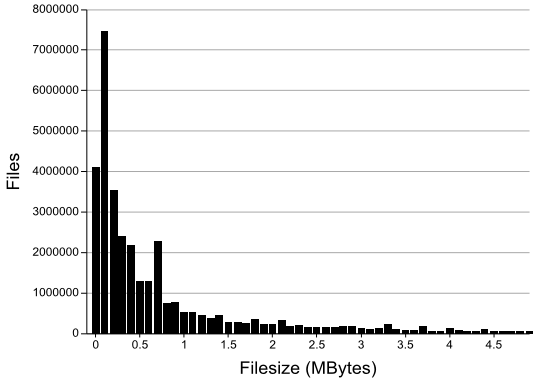



**Figure 6: Histogram of file sizes in MBytes. The average is 3.437 MBytes, while the median is 0.402 MBytes.**

has considerable weight. Fully half of the area under the histogram lies to the right of 3.437 MBytes.

A total of 546,973 or 1.45% of files have reported size greater than 30 MBytes. These are larger than the 30 MBytes limit imposed by the SHA-1 algorithm described in Section 2. This suggests that the 30 MBytes cutoff is a reasonable compromise, in that it saves considerable compute time on the client (calculating SHA-1 of a file is an expensive operation) while introducing minimal risk of hash collisions. In order for a hash collision to occur two files would have to be bit for bit the same for their first 30 MBytes.

## 4.3 Common publishers

Table 6 shows the most common "Company Name" fields for files in the dataset. These are the companies that have produced the largest number of files, not (necessarily) those that have produced the most common files. We remind that this is correlated with, but not the same as, the most common program publishers. We caution

| Company Name | All files | SI files |
|---|---|---|
| Microsoft Corporation | 7.94 | 9.18 |
| Symantec Corporation | 5.57 | 7.12 |
| Sun Microsystems, Inc. | 1.64 | 2.20 |
| Adobe Systems, Incorporated | 1.4 | 1.93 |
| Adobe Systems Incorporated | 0.98 | 1.36 |
| Nero AG | 0.76 | 0.70 |
| Parallels Software International, Inc. | 0.50 | 0.56 |
| Hewlett-Packard | 0.46 | 0.35 |
| Macromedia, Inc. | 0.45 | 0.53 |
| Realtek Semiconductor Corp. | 0.44 | 0.53 |
| Adobe Systems, Inc. | 0.42 | 0.40 |
| Logitech Inc. | 0.31 | 0.28 |
| Electronic Arts, Inc. | 0.31 | 0.38 |
| Autodesk, Inc. | 0.31 | 0.35 |
| Google Inc. | 0.28 | 0.33 |
| RealNetworks, Inc. | 0.27 | 0.26 |
| Macrovision Corporation | 0.27 | 0.35 |
| BitRock SL | 0.24 | 0.28 |
| Nokia | 0.22 | 0.18 |
| Intel Corporation | 0.21 | 0.24 |
| InstallShield Software Corporation | 0.21 | 0.28 |
| Apple Inc. | 0.21 | 0.25 |
| Alexander Roshal | 0.21 | 0.26 |
| Hewlett-Packard Co. | 0.21 | 0.20 |
| Yahoo! Inc. | 0.18 | 0.24 |
| web technology Corp. http://www.webtech.co.jp/exepress/ | 0.18 | 0.14 |
| Atomix Productions | 0.16 | 0.07 |
| WildTangent | 0.16 | 0.15 |
| Igor Pavlov | 0.16 | 0.15 |
| NVIDIA Corporation | 0.16 | 0.15 |

**Table 6: Most common company names across all files and single-instance files.**

that the relative number of files is not indicative of market share of any particular product. Most of the large software vendors are represented. The names "Alexander Roshal" and "Igor Pavlov" are those reported by the WinRAR and 7-Zip applications respectively.

We also tabulate the companies that produce the greatest number of single-instance files (see Section 5.2).

## 4.4 Distribution of systems per file

Figure 7 shows the histogram of the number of systems per file. To dispel the risk of confusion it is worth explaining how this differs from histogram of files per system (shown in Figure 2 (b)). While these are similarly named they are quite different, and neither is derivable from the other. The files per system histogram (Figure 2 (b)) is found by counting the number of files on each system and then forming into bins (of width 100 in Figure 2 (b)). The systems per file histogram in Figure 7 is formed by counting the number of systems that each particular file appears on. As can be seen from Figure 7 there are 18 million files (*i.e.*, half of the total reported) that appear on only one system each. There are about 5 million files that appear on two systems each, 2 million that appear on three systems each and so on.

We refer to files that appear only on a single system as single-instance (SI) files. The fact that there are 18 million such files does

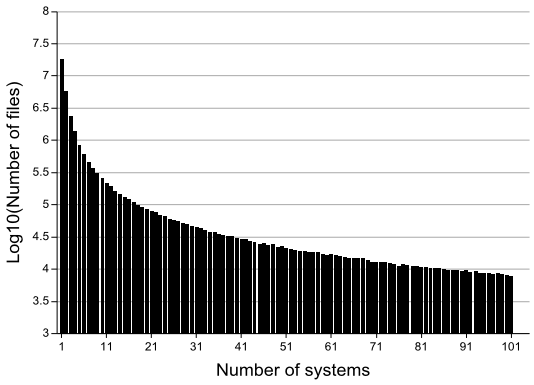



**Figure 7: Histogram of number of systems per file. Observe that one half of all files (*i.e.*, $10^{7.27} \approx 18.5$ million) appear on only one system, that is a second copy never appears after the first report.**

| Version | Release date | Percent |
|---|---|---|
| 4.0 | April 1999 | 1.78 |
| 5.0 | May 2001 | 9.96 |
| 5.1 | July 2003 | 1.79 |
| 6.0 | Jan. 2005 | 6.41 |
| 7.0 | Nov. 2006 | 16.08 |
| 8.0 | June 2007 | 6.17 |
| 8.1 | NA | 13.36 |
| 8.2 | NA | 2.07 |
| 9.0 | July 2008 | 7.46 |
| 9.1 | May 2009 | 7.54 |
| 9.2 | Oct. 2009 | 6.17 |
| 9.3 | Jan. 2010 | 11.12 |
| 9.4 | May 2010 | 4.78 |
| 10.0 | Nov. 2010 | 3.95 |

**Table 7: Major versions of AcroRd32.exe with percentages. Observe that a 89.9% of systems have versions that were out-of-date by the beginning of the observation interval (June 2010). Further, 72.62%, 57.6%, and 42.2% have versions that were more than one, two or three years out-of-date respectively.**

not, of course, imply that most of the 20 million reporting machines contain a SI file. In fact we will see in Section 5 that this is not the case. Most of the 18 million SI files are concentrated among a relatively small number of systems. In fact one reporting system alone accounted for 1,528 SI files.

## 4.5 Updating Software

The reluctance of users to update software is well known. In Table 7 we tabulate the percentages of systems reporting different major subversions of `AcroRd32.exe`, an executable associated with Adobe Acrobat Reader. We also tabulate the release date for the subversions. Observe that the majority of systems report versions that were significantly out-of-date by the time the observation period began (June 2010). Almost all systems, 89.9%, had versions earlier than 9.4, the latest available at the beginning of the period. Versions more than one year out-of-date accounted for 72.6% of systems, and 57.6% and 42.2% of systems were running versions that were more than two and three years out-of-date respectively. We observe a similar pattern across other popular applications such as Firefox, iTunes, *etc*. The conclusion seems clear that users lack motivation to upgrade software, even for applications which are frequently targeted by malware authors.

# 5. UBIQUITY

Some programs and files are very common, reaching a majority of Windows machines, while others are installed on only a handful, or even one. The population statistics of our dataset give valuable insights into the penetration of files in the population.

## 5.1 Penetration of rare files

We saw in Section 4.4 that fully half of all the files reported were single-instance, meaning that they were seen on only a single system. We now explore the distribution of files by rareness. Our motivation for doing so is guided in part by our interest in the feasibility of using prevalence as a feature in determining reputation, which we deal with in Section 5.5 below.

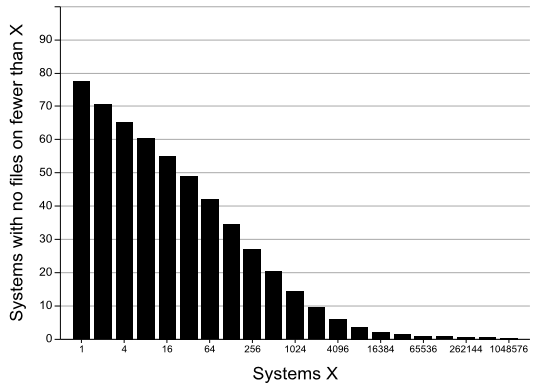
Figure 8 (a) shows the percent of systems that have no files that appear on fewer than $X$ other systems (for various values of $X$). That is, 77.5% of systems do not have a single-instance file: every file that they report is also reported by at least one other system. The plot shows how this evolves for the population as we increase $X$ through six orders of magnitude. The percentage drops quite rapidly, so that 48.9% of all machines have no files that weren't also present on at least 32 other systems, and only 27% of systems have no files that aren't present on at least 256 other systems.

Figure 8 (b) graphs this percentage for Q1, the least-active quintile. This shows that rare files are far less common on the least active machines. Fully 88.7% of Q1 machines had no single-instance file, 74.7% had no files that weren't present on at least 32 other systems, and 53.8% had no files that weren't also reported by at least 256 other systems. At the extreme 2.3% of these systems do not have a single file that wasn't also present on at least $2^{10} \approx 1$ million other systems.

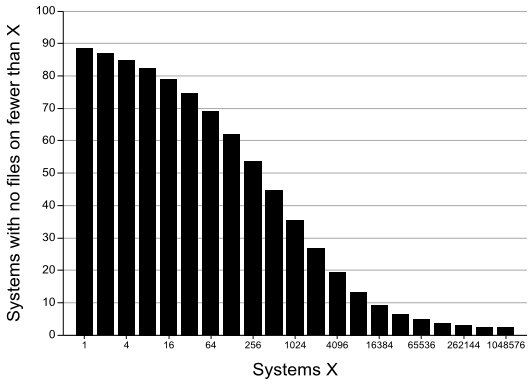
## 5.2 Single-instance files

We saw, in Figure 7, that half of the files reported were single-instance. It is not hard to imagine how some files end up on millions of systems. We can easily picture the .exe's associated with popular applications such as Firefox, Acrobat and Office achieving great popularity. But what of the other extreme? Where do the 18 million single-instance files in Figure 7 come from? A logical possibility is that they they are the files associated with extremely obscure applications. This appears not to be the case. Table 6 shows the most common Company name associated with single-instance files. That is, which companies accounted for which percentages of the single-instance files. As can be seen, the producers of the single-instance files are the same producers who produce the most popular applications and programs: Microsoft, Symantec, Sun and Adobe head the list. Thus, while one might imagine that the head of the distribution (*i.e.*, popular files) occupied by Microsoft, Adobe, *etc*, and the tail (*i.e.*, single-instance files) occupied by obscure software vendors, this appears not to be the case. The tail of the distribution has just as much representation from top vendors as the head.

This suggests that many popular applications are packaged and delivered to users in a fashion that makes individual executable files unique. It appears this practice is quite common. This would ex-




(a) The evolution for the overall population.




(b) The evolution for the least active quintile of machines, Q1 (as defined in Section 3.4).

**Figure 8: Percent of systems having no files that are present on fewer than X other machines, as a function of X.**

plain why the arrival rate of previously-unseen files does not drop beyond 1.6 even as the population grows from 5 to 20 million systems.

## 5.3 Estimating the unseen files

At any population size, of course, we have seen only the reports from a fraction of the total population of Windows machines. There remain files that have not been seen, simply due to the sample size. For example, if a file is present on a fraction $p$ of systems then the probability that we see a copy in a sample population of size $Q$ is $1-(1-p)^Q$ (assuming unbiased sampling). As $Q$ increases, so do our chances of seeing the file.

The standard way of estimating the unseen portion of the distribution is Good-Turing estimation [10]. This estimates that the probability that a reported file is new at population $Q$ is:

$$\Phi(Q) = \frac{N_1(Q)}{\sum_{k=1}^{\infty} k \cdot N_k(Q)} \quad (1)$$

where $N_k(Q)$ is the number of files that are present on $k$ systems when we have a population of size $Q$. That is, the $N_k(Q)$ are as displayed in the histogram Figure 7, so that $N_1(Q) \approx 18$ million, *etc*(*i.e.*, the $x$-axis of Figure 7 is $k$ and the $y$-axis is $\log_{10} N_k(Q)$). That is, the probability of unseen files is estimated to be the same as the fraction of single-instance types. As the population grows this clearly shrinks: the cumulative collection of files grows as time goes by, so the probability of any newly encountered file being unknown should drop. Figure 9 shows the evolution of $\Phi(Q)$ as our population $Q$ grows. It falls very sharply a first, but then slows. At $\log_{10} Q = 4$ (*i.e.*, ten thousand systems) $\Phi(Q) \approx 0.03$, indicating that about 97% of files will have been seen before. At $\log_{10} Q = 5$ (*i.e.*, one hundred thousand systems) $\Phi(Q) \approx 0.01$, indicating that 99% of reporting files have already been seen. This increases with scale, but at a slowing rate. At $\log_{10} Q = 6$ (*i.e.*, population of one million) $\Phi(Q) \approx 0.003$ indicating that about 99.7% of arriving files have already been seen. $\Phi(Q)$ continues to fall slowly reaching $0.0018$ at $Q = 20$ million. At this population approximately 99.82% of arriving files have already been seen. Thus, scale helps in the exhaustiveness of the list, but at an extremely slow pace.

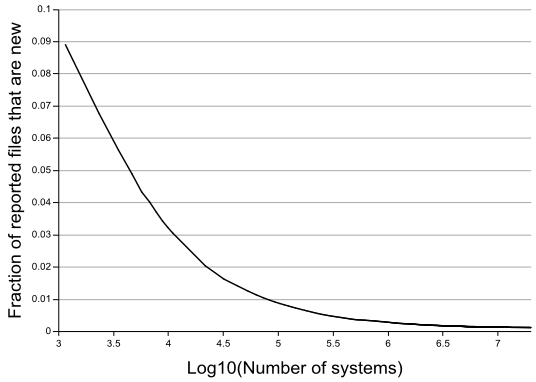



**Figure 9: Fraction of reported files that are new as a function of population size. This continues to drop slowly with scale, reaching $0.0018$ at 20 million systems. That is, 99.82% of reported files have already been seen when the population reaches 20 million.**

Since machines report an average of 756 files (from Section 4) we thus expect about $0.0018 \times 756 \approx 1.36$ of the files reported per system to be previously unseen. This cross-checks well with the observed rate of about 1.6 found in Section 4 (*i.e.*, the plateau rate of new files per machine shown in Figure 5).

## 5.4 Innovation Rate

It is natural to wonder how many files there are, and how many new are created every day. The two measures that we have (previously unseen files as a function of time in Figure 5 and fraction of reported files are new as a function of population in Figure 9) suggest a stubbornly slow decay of the innovation rate as population increases. At a population of 20 million machines not only have we not "seen everything" but the rate at which new things arrive is not decreasing. This strongly suggests that even if we had a reporting population of 100 million or a billion we will continue to learn of

new files.

An elegant means of estimating the number of new files that we might expect to see is given by Good and Toulmin [11]. They suggest that a doubling of the sampled population (*i.e.*, if we had $2Q$ instead of $Q$ systems) will result in

$$\sum_{k=1}^{\infty}(-1)^k N_k = N_1 - N_2 + N_3 - N_4 + N_5 - \cdots,$$

previously unseen files being observed (where as before $N_k$ is the number of files that have been seen $k$ times at population $Q$). Evaluating this for our dataset, (*i.e.*, summing the $N_k$ in Figure 7) suggests that 14.12 million new files should be expected if we doubled the sampled population from 20.7 to 41.4 systems. Again, this accords well with the persistent rate of arrival of new files shown in Figure 5. While the scale of the sampled population is large by no means is it large enough to have "seen everything."

As a motivating example of the innovation rate of new files, we consider the program Google Chrome. We examined in detail the installation pattern of closely related subversions. We did this by installing on machines that we controlled, and thus we could examine all files and the directory structure (*i.e.*, these were not drawn from the Upgrade Advisor dataset). We notice that comparing installations of two subversions, 5.0.375.126 and 5.0.375.127, subdirectories are created for each new subversion of the Chrome browser which include all files but two (chrome.exe and wow_helper.exe). Interestingly, all but one of the 64 files in each of these two subdirectories have new SHA-1 hashes, even though pairs of files share identical file names and file sizes. Clearly, these two installations which differed by only 10 days (Aug. 10, 2011 and Aug. 20, 2011), generate files that are new as far as SHA-1 is concerned, even though they are almost certainly identical in function.

## 5.5 Ubiquity based reputation

Current anti-virus (AV) techniques predominantly rely on signatures of known-bad files. These signatures are generated by analysts who examine suspect files, make a determination as to whether they are good or bad, and then create a signature if they are regarded as bad. These signatures are not always simple hashes, when possible they produce generic signatures that allow identification of a whole family of malware. While analysts can be assisted using automatically generated reports from static and dynamic analysis of the files the process is manually intensive. This process scales very poorly. The more ingenious malware authors will strive to make their code polymorphic so that a signature generated for one instance will not detect others. This places a potentially unmanageable burden on AV companies. Spectacular false positives occasionally occur, such as when the McAfee VirusScan AV program wrongly labeled `svchost.exe` (an important Windows file that acts as a host for DLL files) as malicious, or when Microsoft Security Essentials classified a version of Google's Chrome browser as bad [16].

Whitelists are sometimes suggested as an alternative. The idea is simple in principle: instead of allowing files to run unless they are determined to be bad (*i.e.*, an AV signature declares that they are known bad) forbid all files from running unless they are known to be good. This is the approach taken, for example, by the iPhone: there is a single point through which applications can be installed (the AppStore), applications are scrutinized before they can appear (the iPhone App Certification Process) and everything that is not approved is forbidden. This has the advantage that if an approved App is later determined to be bad (*e.g.*, when a screen-lock app was, in Apple's view, inappropriately gathering password data [7]) the company can remove it from the AppStore). This approach has the disadvantage, however, that it is a *curated* whitelist: the list must be manually maintained. This may be feasible where the number of applications runs to thousands or tens of thousands, but would appear to be completely unworkable when, as we have seen, there are scores of millions of files.

An alternative approach is a *non-curated whitelist* where inclusion does not require manual inspection. Symantec incorporates a reputation based strategy in their products [4]. Similarly, Microsoft uses some form of reputation in Internet Explorer 9's AppRep system [8, 12]. Specific algorithmic details have not been disclosed for either system. We now investigate the challenges faced in trying to develop a non-curated whitelist based on the data received from the Upgrade Advisor. For example, popularity in a large population might be an indication of reputation. For example, any file that reaches X% of the population (as the most popular files we found in Section 4.3 do) is almost certainly good.

Based on the data in Figure 8 it does not appear that prevalence alone can be used to block files. We argue in Section 6.2 below that at least 99.5% of files in our dataset are not malware. Assuming this to be the case the enormous number of single-instance files make a popularity threshold hard to enforce. While 77.5% of machines don't have a single-instance file this means that 22.5% of systems do. We have no insight into severity the consequences of denying these files the right to run might be.

It does not appear feasible that rareness alone can be used to blacklist an executable in the general population. There are simply too many files which appear seldom. While 69% of the Q1 machines have no files whatever that aren't seen on fewer than 64 other machines, this doesn't allow us to conclude that the remaining 31% are at risk.

Classifying files as good or bad is extremely hard. Our data points to a very longtail distribution of files. While rareness alone does not seem a strong enough feature to make a binary good/bad classification it certainly gives a measure of the consequences of a false positive or false negative. That is, a count of the number of systems per file (such as in Figure 7) helps allocate analyst effort. The cost of a false positive (mistakenly classifying good software as malicious) and a false negative (classifying malicious software as good) both increase with the number of systems that a file is reported on.

# 6. OVERLAP WITH OTHER DATASETS

In this section, we explore the overlap of the Windows Upgrade Advisor telemetry data with two other datasets: the National Software Registry Library (NSRL) and a large set of telemetry received from a suite of commercial antimalware products during April 2011. The goal of these comparisons is to understand how the set of files identified by the Upgrade Advisor compares to similar large datasets collected for other purposes.

## 6.1 NSRL Collection

The NSRL [1] is a collection of file signatures maintained by the National Institute of Standards and Technology (NIST). The main purpose of the NSRL is to aid law enforcement officials in profiling matching files found on a suspect's computer. It contains the SHA-1, MD5 and CRC hashes of over 21 million files. The

file meta-data includes the file name, file size and associated operating system. The files are largely commercial software and the hashes are generated from disk and web crawls. NIST points out that the files cannot be considered either "good" or "bad." There is no prevalence information in the dataset. Thus, it is not possible to determine whether a file has been seen once or millions of times. Our copy of the NSRL dataset has 20.1 million distinct files. We find an overlap of 98,594 with the file hashes collected by the Windows Upgrade Advisor. This count is surprisingly low, accounting for only 0.26% of the files observed in our dataset. That is 99.74% of the files in our dataset were unknown to NSRL.

This raises two interesting questions. First, why is so little of what is in the NSRL database found by the Upgrade Advisor dataset? Second, why is so little of what is in the Upgrade Advisor dataset in NSRL?

Tackling the first of these questions: one reason for the low overlap is that, as Table 8 shows, many of the files in the NSRL database were observed running on older versions of Windows (e.g. Windows 95), operating systems other than Windows (e.g. Linux), or unknown operating systems. In fact, the largest category for relatively recent operating systems is Windows XP with a percentage of 1.06%. Similarly, the largest category for Windows Vista (not shown in the table) is 0.10%. Although, a generic Windows operating system designation ("WIN") was provided for over 25% of the files. It appears NSRL covers legacy executables dating back over 25 years (e.g. Win95 and earlier), while the vast majority of files in the Upgrade Advisor set are reported by machines running Windows XP or later (see Table 3). Thus NSRL has copies of programs and files that have long been superseded by newer versions. This partially explains why our dataset has so few of the NSRL files: a significant percentage of what is in NSRL is old and derives from operating systems other than in our reporting population.

As to the second question: why are 99.74% of the files that we find not in the NSRL? First, and most obviously, the NSRL contains hashes only of files possessed by NIST. These are obtained by web crawls and scans of disks. Our dataset merely contains SHA-1 hashes, and we do not have copies of any of the files. To keep a single copy of all 36.9 million files in our dataset would require 122 TBytes (using the average file size computed in Section 4). Second, our findings in Section 5.4 show that generating an exhaustive list is all but infeasible: previously unseen files continue to arrive at a rate that barely slows as the population increases. Figure 10 shows the arrival pattern of files in the Upgrade Advisor dataset that overlap with the NSRL collection. The vast majority arrive very early, with 74% arriving in the first five weeks. This indicates that the intersection consists primarily of files that are relatively common in the Upgrade Advisor set. For example, if a file is reported with probability $p$ per week; then after $m$ weeks it has a $1-(1-p)^m$ chance of being in the dataset. This follows a power law decay roughly similar to Figure 10; the fact that the decay is very sharp indicates that $p$, the probability of being reported each week is not small.

## 6.2 Anti-Malware Telemetry Data

In this section, we investigate the overlap of the Upgrade Advisor telemetry data with a second large file collection of file hashes obtained from the telemetry data transmitted by a set of anti-malware (AM) products manufactured by our company. These reports were primarily generated by three main sources: Windows Defender, Microsoft Security Essentials (MSE), and the Microsoft Malicious

| OS | Percent |
|---|---|
| "358" | 43.25 |
| "WIN" | 25.11 |
| "UNK" | 5.36 |
| "Gen" | 3.01 |
| "Linux" | 2.98 |
| "WIN2000" | 2.76 |
| "WIN95" | 2.54 |
| "WIN98" | 2.34 |
| "Solaris" | 1.48 |
| "WINNT" | 1.37 |
| "Mac" | 1.07 |
| "WINXP" | 1.06 |
| "XP SP2" | 0.68 |
| "WINNT4.0" | 0.52 |
| "Mac OS 9+" | 0.44 |

**Table 8: Distribution of operating systems in the NSRL.**

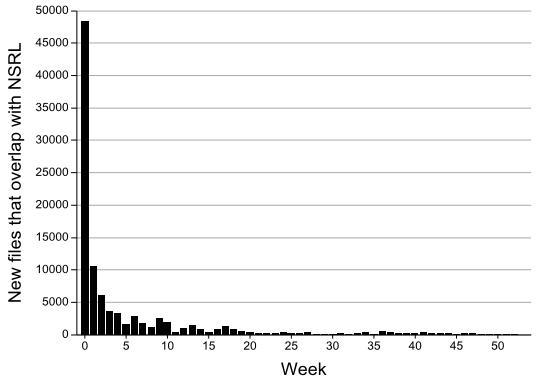



**Figure 10: Arrival rate of new files that overlap with the NSRL dataset. Observe that the majority arrive very early in the observation period, with 74% arriving in the first five weeks.**

Software Removal Tool (MSRT). Windows Defender is an anti-spyware product installed by default in the Windows Vista and Windows 7 operating systems. MSE is an anti-virus product which can be downloaded from the internet. The MSRT is a limited, anti-virus system also included in various versions of Windows which automatically cleans approximately 100, well known families of malware. Similar to the Upgrade Advisor data, we do not have access to the files encountered by the anti-malware products because the files only reside on the remote computers. In particular, we investigated a collection of telemetry reports from 156 million distinct files monitored by all of these anti-malware products over a one month period during April 2011.

Comparing the two file hash collections, we observed an overlap of 1.78% of all files in the AM telemetry with the Upgrade Advisor dataset. In addition, we found an overlap of 0.5% of files detected by the AM products as malicious (i.e. known malware). The small overlap can be explained as follows. Most of the files observed in the Upgrade Advisor dataset are the most common files found on all computers. The majority were installed via ARP or MSI. On the other hand, the telemetry reports from the AM products are searching for known and previously undetected malware. To

limit the number of reports to the AM backend service, reports are transmitted only concerning unsigned and therefore untrusted files. Since the malware's goal is too evade detection, we do not expect it to be commonly installed via ARP or MSI. This is confirmed by the small overlap between the two datasets.

# 7. RELATED WORK

Despite the explosive growth in the number of users of and machines connected to the Internet, large-scale measurements studies have been rare. In 1999, Douceur and Bolosky [9] carried out a methodical study of 4,801 machines. They examine file size, type, age as well as attributes related to directory structure. They were able to measure many attributes that are not gathered in our telemetry. However, our dataset is larger by three orders of magnitude, and we study prevalence, a factor not examined in detail by Douceur and Bolosky. This was the largest study for almost a decade. In 2007, Agrawal *et al.* [2] published a five year study of the file system changes in a population of 60,000 Windows PCs. They measured various temporal changes related to files and directory structure. Both of these studies were conducted on computers within a large software development company. As such, the diversity of systems was limited. Since our study covers any user who chose to run the Upgrade Advisor, it provides a better representation of the general population as a whole.

More recently, authors from Symantec have published two papers related to malware classification using file reputation. Nachenberg *et al.* [4] describe a high-level overview of the reputation system used in their products. While algorithmic details are lacking in this paper, it provides evidence that reputation is a good approach to malware detection. In [5], Chau *et al.*propose a system which simultaneously builds a joint reputation of files and the machines that report them. Microsoft provides some insight into the reputation system employed by Internet Explorer 9's application reputation system [8, 12]. Several security startups or companies offer application white-listing solutions. Bit9 offers several products [3] which assign files to one of three categories: whitelisted, blacklisted, and graylisted. Files which cannot are not known to be malicious or benign are assigned to the graylist.

Forensics experts employ whitelists to filter out files previously known to be benign. However, a very small change in a cryptographic file has (e.g. SHA-1) completely alters the hash value. Chawathe [6] proposed using locality sensitive hashes as a way of finding near-matches to implement a whitelist for the purpose of forensics. Other authors have proposed methods for controlling file execution on computers. Lu *et al.* [13] propose a log-based monitoring system which creates a backup mechanism and monitors the system changes due to file execution. When the system administrator determines that the program altered the system in some forbidden way (e.g. writes a file to disk), the proposed system allows the administrator to remove the file and back out the file's changes to the system.

Despite frequent claims that blacklisting is broken or non-scalable, there has been relatively little effort to study the feasibility of whitelisting. Efforts in this space have largely centered around a curated white-list. The iPhone appstore, for example, requires that all applications go through a vetting procedure before release. This vetting is to some degree manual. Mistakes occasionally happen, as when an app was capturing user passwords in an unauthorized way. However an ecosystem with a 4 year history, a central point of control and thousands of apps, clearly faces a very different challenge from a distributed ecosystem with two decades of history, an installed based of over a billion machines and executable applications that run into the tens, or hundreds of millions. Since there is no central point of control nobody really knows how many applications have been written for the Windows platform.

# 8. CONCLUSIONS

One of our primary motivations for conducting this study was to ascertain the feasibility of creating a non-curated whitelist for consumers which would automatically block the installation of any new programs not currently on the whitelist. After analyzing the rate of the incoming data, we believe this is an extremely daunting task. Products such as Internet Explorer and Symantec's security products avoid this issue by alerting the user if the incoming file does not have a sufficient "good" reputation, but this strategy places the burden on the end user. Making the correct decision in this case is challenging even for computer security experts. We believe that running a semi-curated whitelist, such as employed by Bit9, has more merit in some cases for the enterprise. By semi-curated we mean the following: a security vendor provides a large whitelist for known benign files and a large blacklist for known malware. Any files not on these lists are presented to the IT staff who make the final decision as to whether or not to add the program to the whitelist allowing it to run. In the case where an employee wants to install a new version of, *e.g.*, FireFox, the IT staff has the final say. For many companies where the vast majority of employees should not be installing software in any case, this strategy is in practice today. For other cases where employees are allowed to install software on their machines, this inserts a set of checks and balances to help ensure the safety of the company's computer infrastructure. To some extent, IT staffs have been practicing this today without a dedicated whitelist by verifying OS and application security patches off-line before widely deploying to all of the company's computers.

To deploy an automatically blocking whitelist for consumers would require cooperation between software developers and operating system manufacturers. Apple uses this policy today with their AppStore. All iPad and iPod Touch applications are downloaded and managed via the Apple AppStore. While this works for a new computing platform such as the iPad, it is difficult to imagine how to use this solution for general purpose home PCs given the current widespread available of millions of applications. However, future versions of the operating systems could allow users to "opt-in" to such a system to at least provide some help in the fight against installing malware. To be effective, the software developers would probably want to ensure that they release their software to the app store first before releasing it to end users through other distribution channels such as on their web site or in shrink wrapped boxes. Doing so would avoid the problem where an end user was the first to download the new software update which blocked because it had not generated a sufficient good reputation.

This app store model then places the burden of verifying that the software is not malicious on the operating system manufacturer. One would assume that the OS manufacturer would automatically trust software developed by large software companies. For example, Microsoft could automatically trust any software developed by Google, and vice versa. The question is what to do about the small and mid-tier developers in the middle. Clearly, more exhaustive testing would need to be applied to products from these small companies. Over time, manufacturers would build a reputation, good or bad, for these small entities which would affect their ability to publish through the app store. As a result, legitimate developers

would have an incentive to only publish trustworthy code.

# 9. REFERENCES


[1] National Software Reference Library, version 2.34 , october 2011. http://www.nsrl.nist.gov/.

[2] N. Agrawal, W. J. Bolosky, J. R. Douceur, and J. R. Lorch. A five-year study of file-system metadata. *Proceedings of the ACM Transactions on Storage (TOS)*, 3:9:1–9:32, 2007.

[3] Bit9. Bit9 parity suite. *http://www.bit9.com/products/index.php*.

[4] Z. R. C. Nachenberg, V. Seshadri. An analysis of real-world effectiveness of reputation-based security. In *Proceedings of Virus Bulletin Conference (VB2010)*, pages 178–183, 2010.

[5] D. Chau, C. Nachenberg, J. Wilhelm, A. Wright, and C. Faloutsos. Polonium: Tera-scale graph mining and inference for malware detection. In *Proceedings of SIAM International Conference on Data Mining (SDM) 2011*, 2011.

[6] S. Chawathe. Effective whitelisting for filesystem forensics. In *Proceedings of the IEEE International Conference on Intelligence and Security Informatics (ISI'09)*, pages 131–136, 2009.

[7] CNET. iphone lock-screen password app pulled. *http://news.cnet.com/8301-27076_3-20071405-248/iphone-lock-screen-password-app-pulled/*.

[8] R. Colvin. Smartscreen application reputation: Building reputation. *http://blogs.msdn.com/b/ie/archive/2011/03/22/smartscreen-174-application-reputation-building-reputation.aspx*.

[9] J. Douceur and W. Bolosky. A large-scale study of file-system contents. *ACM SIGMETRICS Performance Evaluation Review*, 27(1):59–70, 1999.

[10] W. Gale. Good-Turing Smoothing Without Tears. Statistics Research Reports from AT&T Laboratories 94.5, AT&T Bell Laboratories, 1994.

[11] I. Good and G. Toulmin. The number of new species, and the increase in population coverage, when a sample is increased. *Biometrika*, 43(1-2):45, 1956.

[12] J. Haber. SmartScreen Application Reputation in IE9. *http://blogs.msdn.com/b/ie/archive/2011/05/17/smartscreen-174-application-reputation-in-ie9.aspx*.

[13] H.-J. Lu and S.-Z. Leng. Log-based recovery scheme for executing untrusted programs. In *Proceedings of International Conference on the Machine Learning and Cybernetics*, pages 2136–2139, 2007.

[14] Microsoft. Windows 7 upgrade advisor. *http://windows.microsoft.com/en-us/windows/downloads/upgrade-advisor*.

[15] Microsoft. Windows 7 upgrade advisor privacy statement. *http://windows.microsoft.com/en-US/windows/downloads/upgrade-advisor-privacy*.

[16] R. Nariaie. Faulty Microsoft AV update nukes Chrome browser. *http://www.zdnet.com/blog/security/faulty-microsoft-av-update-nukes-chrome-browser/9515*.

[17] M. Newman. Power laws, Pareto distributions and Zipf's law. *Contemporary physics*, 46(5):323–351, 2005.