INFERRING PREVIOUSLY UNINSTALLED APPLICATIONS FROM DIGITAL TRACES

Jim Jones†, Tahir Khan†, Kathy Laskey†, Alex Nelson‡, Mary Laamanen‡, Doug White‡

†George Mason University, Fairfax, Virginia, United States
‡National Institute of Standards and Technology, Gaithersburg, Maryland, United States

ABSTRACT

In this paper, we present an approach and experimental results to suggest the past presence of an application after the application has been uninstalled and the system has remained in use. Current techniques rely on the recovery of intact artifacts and traces, e.g., whole files, Windows Registry entries, or log file entries, while our approach requires no intact artifact recovery and leverages trace evidence in the form of residual partial files. In the case of recently uninstalled applications or an instrumented infrastructure, artifacts and traces may be intact and complete. In most cases, however, digital artifacts and traces are altered, destroyed, and disassociated over time due to normal system operation and deliberate obfuscation activity. As a result, analysts are often presented with partial and incomplete artifacts and traces from which defensible conclusions must be drawn. In this work, we match the sectors from a hard disk of interest to a previously constructed catalog of full files captured while various applications were installed, used, and uninstalled. The sectors composing the files in the catalog are not necessarily unique to each file or application, so we use an inverse frequency-weighting scheme to compute the inferential value of matched sectors. Similarly, we compute the fraction of full files associated with each application that is matched, where each file with a sector match is weighted by the fraction of total catalog sectors matched for that file. We compared results using both the sector-weighted and file-weighted values for known ground truth test images and final snapshot images from the M57 Patents Scenario data set. The file-weighted measure was slightly more accurate than the sector-weighted measure, although both identified all of the uninstalled applications in the test images and a high percentage of installed and uninstalled applications in the M57 data set, with minimal false positives for both sets. The key contribution of our work is the suggestion of uninstalled applications through weighted measurement of residual file fragments. Our experimental results indicate that past application activity can be reliably indicated even after an application has been uninstalled and the host system has been rebooted and used. The rapid and reliable indication of previously uninstalled applications is useful for cyber defense, law enforcement, and intelligence operations.

Keywords: digital forensics; digital artifact; digital trace; partial artifact; residual artifact; uninstalled application

1. INTRODUCTION

The practice of digital forensics is the art and science of inferring and proving past activity given some set of residual digital artifacts and traces. These artifacts and traces may be files, Windows Registry entries, log entries, memory contents, network traffic, etc., and past activity of interest may be legitimate and illegitimate user activity, system activity, application installation and usage, malware infection and operation, etc. While whole artifacts may be recoverable in some cases, many situations require inferring and proving past activity given residual partial artifacts and traces. We propose that past activity, specifically application installation and usage, can be reliably suggested from digital traces, even when the application in question has been uninstalled and usage of the system and media has continued. We assume that full artifacts created by an activity degrade monotonically and non-linearly over time. Specifically, files created as a consequence of application installation, usage, and uninstallation are subsequently deleted, and some sectors from these deleted files will be overwritten while other sectors may persist on the digital media. Given prior knowledge of the full file artifacts created by an application, we can then search media of interest for traces in the form of matching partial artifacts, i.e., sectors from the original full artifact, and reason over these matches to suggest past application presence. Our approach complements existing methods that rely on evidence from intact full-file artifacts, an uncleansed Windows Registry, intact log entries., or traces from other sources such as memory contents or network traffic.

In the sections that follow, we discuss prior work in this area, then we describe the two core elements of our approach: (i) building a catalog of sectors associated with specific applications, and (ii) reasoning over sectors that match entries in that catalog. Subsequent sections present our experimental results against a test set with known ground truth and the M57 Patents Scenario (Woods et al., 2011) disk images, for which we have some ground truth. We close the paper with a summary of our conclusions, limitations of this approach, and future research plans.

2. RELATED WORK

Related work to establish the presence of installed and uninstalled applications has generally relied on intact file artifacts (Koppen et al., 2013; Quick et al., 2013), log file analysis (Forte, 2004), and examination of the Windows registry when available (Laamanen et al., 2014; Nelson et al., 2014; Wong, 2007). Additional techniques and methods rely on traces such as email addresses, URLs, etc. extracted from raw data (Garfinkel, 2013), or data structures and other known-layout data from memory (Ligh et al., 2014). Intact file artifacts for uninstalled applications may be files remaining from an aborted or poorly written uninstall application, or may be user files which are created during application use and are deliberately not deleted as part of the application uninstall process, such as user preference files. Log files include varying levels of detail depending on the application creating the log, and establishing the integrity of the log file requires secure creation, transmission, and storage of the log file. Registry artifacts may include application specific keys as well as command line execution arguments, recently accessed files, and similar indicators of application installation and usage, whether the application in question has been uninstalled or not. In contrast, our work does not require recovery of any intact artifacts and is specifically designed to suggest applications that have been uninstalled.

Our work relies on recovery and analysis of file fragments in the form of disk sectors. Collange, Dandass, Daumas, and Defour (Collange et al., 2009), Garfinkel, Nelson, White, and Roussev (Garfinkel et al., 2010), and later Young, Foster, Garfinkel, and Fairbanks (Young et al., 2012) and Foster (Foster, 2012) examined sector content uniqueness as it relates to specific file identification. This initial work successfully identified files with distinct content, such as videos, from a limited number of sectors, but the later work also hinted at issues with sector content common across multiple files. These issues fully emerged in the work of Garfinkel and McCarrin (Garfinkel et al., 2015) in the form of "common data structures found in Microsoft Office documents and multimedia files." Garfinkel and McCarrin label such

file fragments "non-probative blocks" and developed heuristics to account for these blocks and reliably detect file presence from fragments. By comparison, we are inferring the past presence of applications based on blocks from multiple files. Further, our approach pre-selects potentially probative blocks then weights matching blocks based on their frequency in our catalog.

3. APPROACH AND METHODOLOGY

The theory underpinning our approach is that application installation and use creates files, and application uninstallation deletes these files. The sectors containing the contents of these deleted files are overwritten over time, but some sectors may remain intact until subsequent examination. These residual sectors, or traces, may be used to infer the likelihood that a particular application was previously installed on the examined system.

It is important to note that just because we empirically establish that an application installation and use creates a specific set of files and corresponding sectors, this does not imply that the presence of these sectors or even intact files proves the current or past presence of the application in question. That is, if I know A causes B and I subsequently find B, I cannot logically conclude that A occurred. On the other hand, if A is established to be the only possible cause of B, then I can logically conclude that the presence of B does prove A. In the context of files and associated sectors, prior research (Garfinkel et al., 2010)(Garfinkel and McCarrin, 2015) showed that while a sector may not have only one possible producer, in practice it is likely to have only one, especially for high entropy sectors. In our work, a pre-processing step removes sectors appearing in our clean OS images, sectors with low entropy, and sectors appearing more than 100 times in our initial catalog, thereby removing sectors known to be produced by, or likely produced by, other processes. Further, we weight the influence of sectors based on the number of different catalog applications in which they appear. In practice, this is accomplished by our Inverse Document Frequency weight described below. Finally, we note that we are not proving the past presence of an application. Rather, we are suggesting an increased likelihood that a particular application was present at some past time, where proof to the standard required by the circumstances would have to be obtained from additional evidence.

Our approach, summarized in Figure 1, reasons over media sectors that match entries in a catalog associating sectors with specific application activities. The catalog was created for 16 Windows applications in a controlled environment using virtual machine snapshots. Catalog entries are post-processed to remove less useful sectors and to assess each sector's potential inferential value. We then match sectors from a digital storage device of interest, e.g., a hard drive, to the entries in the catalog and compute weighted measures that represent the likelihood that the associated application was previously installed on the media of interest.

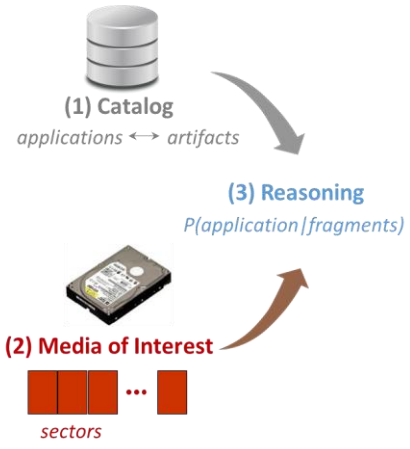

Figure 1: Approach Overview

3.1. Catalog Creation and Post-processing

We are leveraging the NIST Diskprinting effort (Laamanen et al., 2014) to collect application traces. Diskprinting uses virtual machine snapshots to record the state of a system before and after an action of interest. Each snapshot together with captured network traffic is called a *slice*. A series of slices, which reflect sequential activities regarding a single application, is called a *diskprint*. The contents of two adjacent snapshots may then be compared to extract differences (Figure 2). For our purposes, the file systems of adjacent snapshots are compared to identify new, modified, or deleted files. For the NIST diskprinting data, these activities are application Install, Open, Close, Uninstall, and system Reboot

(indicated as I, O, C, U, and R in Figure 3). Diskprints are made up of sequential and cumulative slices, hence the nomenclature B (Base), BI (Base + Install), ..., BIOCUR (Base + Install + Open + Close + Uninstall + Reboot) in Figure 3. Diskprints are created with shared baseline states, by rolling the virtual machine state back to a common point before applications were installed, in order to isolate effects of the operating system.

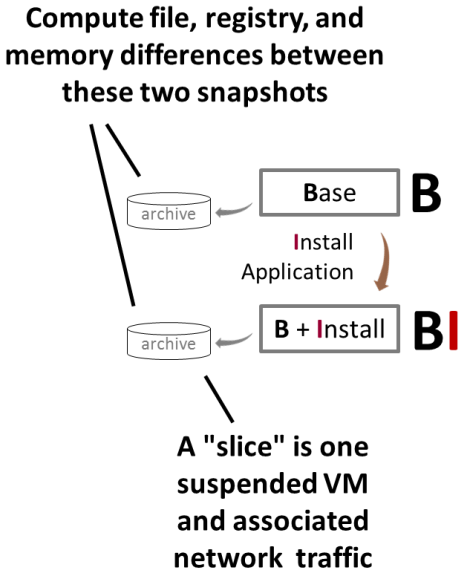

Figure 2: Slices and differencing

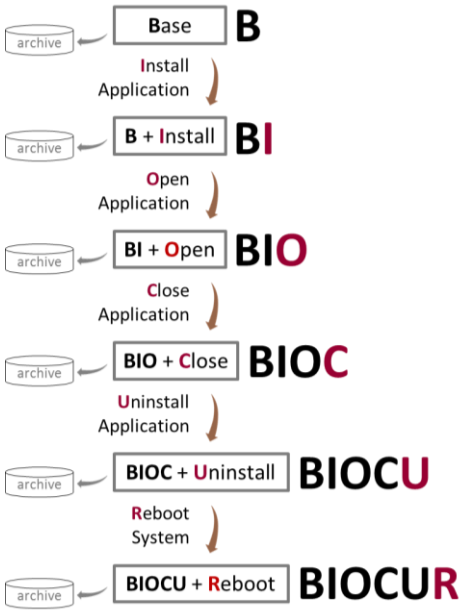

Figure 3: A diskprint is a series of related slices

We use 29 application diskprints of the NIST diskprint data (NIST, 2015), representing 16 applications across one or more different Windows platforms (Table 1) plus three clean Operating System diskprints: one WinXP and two Win7. The applications were selected in part to facilitate subsequent testing against the M57 Patents Scenario images.

Table 1: NIST diskprints

	WinXP	Win7x32	Win7x64
Adv Keylogger	✓		
Chrome	✓	✓	✓
Eraser		✓	
Firefox	✓	✓	✓
HxD hex editor		✓	
Invisible Secrets	✓		
MS Office	✓	✓	✓
Python	✓		
Safari	✓	✓	✓
Sdelete		✓	✓
Thunderbird	✓		
TrueCrypt	✓		
UPX		✓	✓
WinRar		✓	✓
WinZip		✓	✓
Wireshark		✓	✓

Each NIST Diskprint slice contains a snapshot of the system hard disk in the form of a VMDK file. For each pair of adjacent slices, we computed file differences and 512-byte sector-aligned MD5 hashes for each new or modified file (Garfinkel et al., 2012). For experimental purposes, we used MD5s because of their smaller bit count and acceptable impact of false positives from MD5 weaknesses (Dandass et al., 2008). However, an operational deployment of this research would need to employ a more secure cryptographic hash per NIST guidelines on hash selection (NIST, 2012). The diskprint sector hash data currently computes the final sector hash of each file based on file extant vs. padding the final sub-sector fragment with zeros and computing a 512-byte hash. We discard these sub-512 byte sector hashes since they will never match our media of interest hashes, which are always based on a full 512-byte sector hash. We have

discussed but not implemented padding sub-512 byte diskprint fragments with zeros prior to computing the MD5 hash. We process the diskprint sector hash data as described in the following paragraphs and ingest the data into a hashdb (NPS-DEEP, 2015) instance.

File differencing as implemented on the diskprint data has the potential to capture spurious traces, i.e., file differences that are not related to the activity in question but are the product of unrelated system activity. We describe this property of a file as *attribution*, where *positive attribution* means a file is a result of the activity in question, *negative attribution* means the file is not the result of the activity in question, and *marginal attribution* means the file is due to the activity in question but in a non-probative way (such as the $BitMap or pagefile.sys files on a Windows system).

Positive attribution is determined by keyword searching of the filename and path associated with each sector hash. This information is stored in files using differentially-annotated Digital Forensics XML, or DFXML (Garfinkel, 2012; Nelson et al., 2014)), a language that associates file system metadata with file content summaries, including file paths, full-file hashes, and sector-level hashes. The DFXML language facilitates interaction between tools, such as those used in our processing steps.

For each application, sector hashes whose source file paths contain matching keywords from Table 2 are retained. Keywords were derived by examining string frequencies in the collective file path names for each application and selecting the most common, subject to human review.

Table 2: Keyword Whitelist

Application	**keyword(s)**
Adv Keylogger	keylogger
Chrome	chrome,google
Eraser	eraser
Firefox	firefox,mozilla
HxD hex editor	hxd
Invisible Secrets	"invisible secrets"
MS Office	office,"microsoft shared"
Python	python
Safari	safari
Sdelete	sdelete
Thunderbird	thunderbird
TrueCrypt	truecrypt
UPX	upx
WinRar	winrar
WinZip	winzip
Wireshark	wireshark

For example, whitelisting the Firefox19 on 64-bit Windows 7 (Win7x64) diskprint reduced trace files from 1,054 to 289, and reduced associated sector hashes from 16,096,960 to 157,530. This whitelisting approach is something of a blunt instrument, yet we obtain good results in our subsequent experiments. In the section on future work, we propose alternative catalog construction techniques to increase the quality of collected file fragments (sectors).

Sector hashes, including those from files with positive attribution, are not necessarily unique. We describe this property of a sector as its *frequency*, where *distinct* means the sector only occurs once in the post-processed diskprint data, *application common* means the sector occurs in one or more application diskprints but not elsewhere, and *global* means the sector occurs outside of the application diskprints (i.e., in the baseline OS states).

We limit sector hash value frequency in the hashdb instance to 100. While somewhat arbitrary, this limit allows for some multi-application or multi-print hashes to remain while removing hashes not likely to have discriminatory value and keeping the hashdb to a manageable size. If desired, we can later select hash values below the f=100 threshold, or we can reprocess the original diskprint sector hash data if results indicate that sector hashes with frequency greater than 100 have inferential value.

Table 3: Total hashes and files per application diskprint

Diskprint	Total Hashes	Total Files
AdvKeylogger-WinXP	4,716	23
Chrome28-W7x32	686,986	669
Chrome28-W7x64	670,051	499
Chrome28-WinXP	1,035,098	624
eraser-W7x32	69,984	24
Firefox19-W7x32	103,341	132
Firefox19-W7x64	106,270	146
Firefox19-WinXP	96,377	115
HxD171-W7x32	4,774	12
InvSecrets21-WinXP	6,689	19
OfficePro2003-W7x32	1,090,216	3,800
OfficePro2003-W7x64	1,077,126	3,804
OfficePro2003-WinXP	656,354	2,801
Python264-WinXP	86,287	2,355
Safari157-W7x32	316,224	907
Safari157-W7x64	569,645	1,504
Safari157-WinXP	343,824	918
sdelete-W7x32	642	5
sdelete-W7x64	642	4
Thunderbird2-WinXP	68,102	172
TrueCrypt63-WinXP	24,520	16
UPX-W7x32	1,796	19
UPX-W7x64	1,813	19
Winrar5beta-W7x32	9,196	41
Winrar5beta-W7x64	18,328	81
Winzip17pro-W7x32	240,229	149
Winzip17pro-W7x64	262,854	153
Wireshark-W7x32	171,515	617
Wireshark-W7x64	209,666	611
TOTALS	7,933,265	20,239

As a practical matter, hashdb supports a maximum frequency parameter when the hashdb instance is created. However, this only prevents the addition of more hash values which have already reached the maximum frequency - it does not remove the hash value from the hashdb instance. To prevent undesired

effects on our subsequent calculations, we set a maximum frequency of 101 prior to ingest, then we remove all hashes with frequency of 101 after ingest is complete. Without this extra step, the catalog would contain all sector hashes in the original data and all sector hashes with frequency greater than 100 would be retained with frequency equal to 101, regardless of the actual frequency of these sector hashes. With this extra step, the catalog contains sector hashes with accurate frequency counts, and only sector hashes with actual frequencies of 100 or less.

Certain low entropy sector contents, such as all zeros or all ones, occur with high frequency in many sources and have no discriminatory value. For example, the following sector hashes with the noted content occur thousands or millions of times in the original NIST diskprint sector hash data.

```
'bf619eac0cdf3f68d496ea9344137e8b' # repeated 00
'393a0fa0f348fb03871ab93726057ddc' # repeated 01
'de03fe65a6765caa8c91343acc62cffc' # repeated FF
'c5d77850e62433f25d5496bfad94c1b2' # repeated 00; 06 @offset 510
```

These sector hashes would be removed by our maximum frequency processing step above. However, we filter these sector hashes out at an earlier processing stage simply to speed up subsequent processing.

The NIST diskprint data includes diskprints of non-application activities on three operating system variants: two WinXP and one Win7x64. Any hash value appearing in these base OS diskprints does not have discriminatory value for a subsequently installed application, so we remove these hash values from the hashdb instance. As a practical matter, this was accomplished by using hashdb's *add_repository* command to build a hashdb instance of all OS hash values, then using hashdb's *subtract_hash* command to remove those hash values from the original hashdb instance.

The combined whitelist and frequency limit processing resulted in an overall file count of 99,227 and sector hash count of 44,677,825 (file and sector hash counts before whitelist and frequency limits were not computed). Removing the base OS diskprint sector hash values reduced the overall file count to 20,239 and sector hash count to 7,933,265.

To facilitate later calculations of application likelihood, we count and store the total sector hashes and total files per application in the final catalog. These totals (Table 3) are extracted from the final noise-reduced hashdb instance using hashdb's *hash_table* command (v.1.0.0 and prior) with subsequent grep expressions (for hash totals) and hashdb's *sources* command with subsequent grep expressions (for file totals). These totals are for all slices in each diskprint, where each diskprint contains 5-6 slices. 5-slice diskprints result from applications where the "Open" and "Close" steps were combined as a single "Run" step.

3.2. Image Processing

Media of interest is assumed to be a raw image of a hard disk or similar. We use the md5deep tool (https://github.com/jessek/hashdeep) to compute sector-aligned 512-byte MD5 hashes for the entire disk or disk image, storing the results in a DFXML file. We then use hashdb's *scan_expanded* command (v1.0.0) to identify hash values in the DFXML file that match hash values from the hashdb instance. The hashdb *scan_expanded* command output includes the file source and repository information from the hashdb instance. We require these details, as we are using the repository name to hold the diskprint (application) identifier, and the source file information allows us to compute which files in the catalog, and how much of each file, is matched. Matches are written to an interim text file.

The matches text file is processed to compute the various measures of diskprint matching, i.e., application presence. Output includes the number and fraction of distinct hashes matched for each diskprint (application), the number and fraction of total files matched for each diskprint where a file match is declared if one or more hash values from that file are matched. Output also includes weighted versions of these two measures, which are discussed below.

After we eliminated weak or non-probative sector hash values in our noise reduction process, we then applied weights to matching sector hashes based on their occurrence across applications, i.e., their frequency in the catalog. A sector hash that occurs in N different diskprinted applications is weighted with a factor 1/N (this is the hyperbolic formulation of Inverse Document Frequency described by Zobel and Moffat (Zobel et al., 1998)). A sector hash value that occurs in only one diskprinted application is weighted 1/1=1.0; a sector hash value that occurs in 2 different diskprinted applications is weighted 1/2=0.5; and so on. This calculation is shown below, and the results are shown in the sample output of Table 4 under the heading w_sector% (weighted sector %). In the formula, each matching sector for a given diskprint is weighted by its inverse frequency in the catalog; these weighted matching sector counts are then summed and divided by the total number of sectors in the catalog for that diskprint to give a weighted sector % for that diskprint.

$$\text{weighted sector \%}_{DP} = \left(\sum_{S=1}^{\text{num\_sec\_matches}} 1 / \text{freq}_S \right) / \text{sectors\_total}_{DP}$$

Instead of declaring a file *present* if one or more hash values from that file are found (as the data in Table 4 under the heading "files_found" does), we compute the percent of each file that is matched and weight the summation accordingly. For example, if we match M sectors for a file out of N total sectors in the catalog for that file, then that file hit is worth M/N. We sum all of the weighted file hits for each diskprint and divide by the total number of files in the catalog for that diskprint to give a weighted file % for that diskprint. This calculation is shown below, and the results are shown in the sample output of Figure 4 under the heading w_file% (weighted file %).

$$\text{weighted file \%}_{DP} = \left(\sum_{F=1}^{\text{num\_file\_matches}} \frac{\text{matched\_sectors}_F}{\text{total\_sectors}_F} \right) / \text{files\_total}_{DP}$$

Sample output for one of the test images is shown in Table 4 below. This Win7x64 test image had Chrome installed, opened, closed, and uninstalled, then the system was rebooted and the snapshot taken. The three Chrome diskprints (for Chrome on WinXP, Win7x32, and Win7x64) are the three highest valued hits based on both weighted sector % and weighted file % (the sort key in the table). Other data are included in this verbose output, to include the total sector hashes and total files for each diskprint, as well as hits and % of total for each.

Table 4: Sample analysis output for source image "Chrome Win7x64"

diskprintName	sectors found	sectors total	sector%	w_sector%	files found	files total	file%	w_file%
Chrome28-W7x64	66795	670051	9.97%	3.63%	153	499	30.66%	21.46%
Chrome28-WinXP	40831	1035098	3.94%	1.16%	208	624	33.33%	21.10%
Chrome28-W7x32	66795	686986	9.72%	3.54%	152	669	22.72%	16.26%
Winzip17pro-W7x32	2186	240229	0.91%	0.46%	41	149	27.52%	3.63%
Winzip17pro-W7x64	2162	262854	0.82%	0.41%	42	153	27.45%	3.53%
Firefox19-W7x32	4183	103341	4.05%	0.59%	18	132	13.64%	2.44%
Firefox19-WinXP	4183	96377	4.34%	0.63%	17	115	14.78%	2.40%
Firefox19-W7x64	4184	106270	3.94%	0.57%	18	146	12.33%	2.37%
Thunderbird2-WinXP	17	68102	0.02%	0.01%	6	172	3.49%	1.09%
Winrar5beta-W7x64	9	18328	0.05%	0.01%	7	81	8.64%	0.38%
Winrar5beta-W7x32	9	9196	0.10%	0.02%	7	41	17.07%	0.38%
Safari157-WinXP	573	343824	0.17%	0.02%	31	918	3.38%	0.32%
Safari157-W7x32	573	316224	0.18%	0.02%	31	907	3.42%	0.30%
Safari157-W7x64	575	569645	0.10%	0.01%	35	1504	2.33%	0.24%
sdelete-W7x64	1	642	0.16%	0.04%	2	4	50.00%	0.17%
Wireshark-W7x32	51	171515	0.03%	0.01%	10	617	1.62%	0.16%
sdelete-W7x32	1	642	0.16%	0.04%	2	5	40.00%	0.14%
OfficePro2003-WinXP	1014	656354	0.15%	0.02%	33	2801	1.18%	0.13%
OfficePro2003-W7x32	1014	1090216	0.09%	0.01%	33	3800	0.87%	0.11%
OfficePro2003-W7x64	1014	1077126	0.09%	0.01%	33	3804	0.87%	0.08%
Wireshark-W7x64	11	209666	0.01%	0.00%	5	611	0.82%	0.02%
eraser-W7x32	21	69984	0.03%	0.02%	2	24	8.33%	0.01%
TrueCrypt63-WinXP	1	24520	0.00%	0.00%	1	16	6.25%	0.01%
Python264-WinXP	23	86287	0.03%	0.01%	6	2355	0.25%	0.00%
AdvKeylogger-WinXP	0	4716	0.00%	0.00%	0	23	0.00%	0.00%
InvSecrets21-WinXP	0	6689	0.00%	0.00%	0	19	0.00%	0.00%
UPX-W7x32	0	1796	0.00%	0.00%	0	19	0.00%	0.00%
HxD171-W7x32	0	4774	0.00%	0.00%	0	12	0.00%	0.00%
UPX-W7x64	0	1813	0.00%	0.00%	0	19	0.00%	0.00%

4. RESULTS

We generated eight test images, five containing the installation, use, and uninstallation of a single catalog application and three containing the installation, use, and uninstallation of multiple catalog applications. We also processed the four final day disk images from the M57 Patents Scenario case. We also processed WinXP, Win7x32, and Win7x64 images with no applications of interest installed and found no more than 1% matching sectors per application.

Table 5: Single application test case results

Source Image: Chrome28-W7x64			**Source Image: UPX-W7x64**		
diskprintName	w_sector%	w_file%	diskprintName	w_sector%	w_file%
Chrome28-W7x64	3.63%	21.46%	UPX-W7x32	2.97%	52.16%
Chrome28-WinXP	1.16%	21.10%	UPX-W7x64	2.94%	52.16%
Chrome28-W7x32	3.54%	16.26%	Winzip17pro-W7x32	0.44%	3.52%
Winzip17pro-W7x32	0.46%	3.63%	Winzip17pro-W7x64	0.41%	3.45%
Winzip17pro-W7x64	0.41%	3.53%	Firefox19-W7x64	0.01%	1.69%
...	...	...	...	...	...
Source Image: Winrar5beta-W7x64			**Source Image: Firefox19-W7x64**		
diskprintName	w_sector%	w_file%	diskprintName	w_sector%	w_file%
Winrar5beta-W7x32	8.39%	56.18%	Firefox19-WinXP	6.88%	57.32%
Winrar5beta-W7x64	4.21%	32.80%	Firefox19-W7x32	6.42%	51.52%
Winzip17pro-W7x32	0.44%	3.53%	Firefox19-W7x64	6.26%	47.25%
Winzip17pro-W7x64	0.41%	3.46%	Winzip17pro-W7x32	0.44%	3.57%
sdelete-W7x32	0.04%	0.14%	Winzip17pro-W7x64	0.41%	3.48%
...	...	...	...	...	...
Source Image: sdelete-W7x64					
diskprintName	w_sector%	w_file%			
sdelete-W7x64	7.75%	33.95%			
sdelete-W7x32	7.75%	27.16%			
Winzip17pro-W7x32	0.44%	3.52%			
Winzip17pro-W7x64	0.41%	3.45%			
Firefox19-W7x64	0.01%	1.67%			
...	...	...			

4.1. Single-application test cases

For each single application test case, we started with a fresh install of the appropriate OS (WinXP, Win7x32, or Win7x64) and mimicked the diskprint activity as described in the diskprint data, e.g., install, open, close, uninstall, and reboot. These test cases did not use NIST's source media for the OS or application, and did not strictly follow the details of activity performed by NIST personnel when creating the diskprint images. Results from the post-reboot snapshot of the seven single application test cases are summarized in Table 5, where only the top 5 weighted file % results are shown. In each test case, the installed/uninstalled application was correctly identified and the weighted sector % and weighted file % measures indicate a sharp drop off for catalog applications that were not present on that test image.

4.2. Multiple-application test cases

Table 6: Multiple application test case results

Source Image: Firefox, Chrome, & Safari		**Source Image: WinRAR & WinZip**	
diskprintName	w_file%	diskprintName	w_file%
Safari157-W7x32	94.32%	Winzip17pro-W7x64	35.60%
Safari157-WinXP	92.80%	Winzip17pro-W7x32	34.88%
Safari157-W7x64	57.12%	Winrar5beta-W7x32	9.97%
Firefox19-WinXP	46.57%	Winrar5beta-W7x64	9.29%
Firefox19-W7x32	42.83%	Firefox19-WinXP	2.66%
Firefox19-W7x64	37.73%	Firefox19-W7x64	2.60%
Chrome28-WinXP	22.10%	Firefox19-W7x32	2.23%
Chrome28-W7x64	12.88%	Thunderbird2-WinXP	1.49%
Chrome28-W7x32	9.84%	sdelete-W7x64	0.17%
Winzip17pro-W7x32	3.62%	Wireshark-W7x32	0.16%
...	...	...	...
Source Image: Chrome & Firefox			
diskprintName	w_file%		
Firefox19-WinXP	57.21%		
Firefox19-W7x32	52.04%		
Firefox19-W7x64	47.33%		
Chrome28-W7x64	20.45%		
Chrome28-WinXP	20.37%		
Chrome28-W7x32	15.58%		
Winzip17pro-W7x32	3.64%		
Winzip17pro-W7x64	3.53%		
Thunderbird2-WinXP	1.68%		
Winrar5beta-W7x64	0.42%		
...	...		

Three test cases were constructed in a manner similar to the single application test cases, but multiple applications were installed, used, and uninstalled, and multiple reboots occurred. Two of these test cases incorporated two applications and one incorporated three applications. Results from the post-reboot snapshot of these three multiple application test cases are summarized in Table 6. For these cases, the top 10 results based on weighted file % are shown. In all three cases, all installed/uninstalled applications are correctly identified, after which the weighted file % drops off sharply.

4.3. M57 Patents Scenario images

The M57 Patents Scenario is a publicly available data set. The scenario was created for educational and research purposes by faculty and students at the Naval Postgraduate School. The creators of the data set mimicked criminal activity in a lab environment over the course of a month, capturing disk and device images and network traffic during the exercise. Scenario documentation includes a description of the systems and networks involved, characters, and a storyline. For our purposes, the final day snapshots are sufficiently realistic system images, by merit of having been physical machines operated for a real-world month. Also, we have some ground truth about installed and uninstalled applications based on the scenario documentation (availability restricted to faculty at accredited institutions), work by Roussev & Quates (Roussev et al., 2012) that analyzed the same images, and our own direct analysis of the scenario

images. Results from processing the final day (2009-12-11) images for the four scenario users (Charlie, Jo, Pat, and Terry) are summarized in Table 7, where the host OS is indicated after the system name.

Table 7: M57 Patents Scenario results

Charlie (XP)		Jo (XP)		Pat (XP)		Terry (Vista)	
diskprintName	w_file%	diskprintName	w_file%	diskprintName	w_file%	diskprintName	w_file%
Python264-WinXP	98.98%	Python264-WinXP	98.83%	Python264-WinXP	98.91%	Python264-WinXP	85.52%
InvSecrets21-WinXP	63.16%	TrueCrypt63-WinXP	50.00%	Thunderbird2-WinXP	24.94%	Thunderbird2-WinXP	27.81%
Thunderbird2-WinXP	61.00%	Thunderbird2-WinXP	24.73%	AdvKeylogger-WinXP	21.97%	Winzip17pro-W7x64	10.37%
Safari157-W7x32	10.25%	Safari157-W7x32	11.35%	HxD171-W7x32	8.39%	Winzip17pro-W7x32	10.05%
Safari157-WinXP	10.16%	Safari157-WinXP	11.26%	Firefox19-WinXP	3.17%	HxD171-W7x32	8.37%
Safari157-W7x64	6.69%	Safari157-W7x64	7.37%	Firefox19-W7x64	2.93%	Safari157-W7x32	5.46%
Firefox19-WinXP	3.26%	Firefox19-WinXP	3.24%	Firefox19-W7x32	2.78%	Safari157-WinXP	5.35%
Firefox19-W7x32	2.77%	Firefox19-W7x32	2.74%	Winzip17pro-W7x64	2.03%	Chrome28-WinXP	4.83%
Firefox19-W7x64	2.50%	Firefox19-W7x64	2.62%	Chrome28-WinXP	1.64%	Chrome28-W7x64	4.81%
Chrome28-WinXP	2.11%	Chrome28-WinXP	2.15%	Chrome28-W7x64	1.63%	Firefox19-WinXP	3.59%
Winzip17pro-W7x64	2.08%	Chrome28-W7x64	2.03%	Winzip17pro-W7x32	1.50%	Chrome28-W7x32	3.59%
Chrome28-W7x64	2.02%	Chrome28-W7x32	1.52%	Chrome28-W7x32	1.22%	Firefox19-W7x64	3.56%
Chrome28-W7x32	1.52%	sdelete-W7x64	1.35%	TrueCrypt63-WinXP	1.22%	Firefox19-W7x32	3.55%
Winzip17pro-W7x32	1.51%	Winzip17pro-W7x64	1.26%	Winrar5beta-W7x64	0.85%	Safari157-W7x64	3.47%
sdelete-W7x64	1.35%	sdelete-W7x32	1.08%	Winrar5beta-W7x32	0.84%	Winrar5beta-W7x64	2.21%
sdelete-W7x32	1.08%	Winrar5beta-W7x64	0.95%	Safari157-WinXP	0.62%	Winrar5beta-W7x32	2.19%
TrueCrypt63-WinXP	0.73%	Winrar5beta-W7x32	0.94%	Safari157-W7x32	0.54%	TrueCrypt63-WinXP	0.97%
Winrar5beta-W7x32	0.64%	Winzip17pro-W7x32	0.72%	OfficePro2003-WinXP	0.47%	OfficePro2003-W7x32	0.39%
Winrar5beta-W7x64	0.64%	OfficePro2003-WinXP	0.43%	OfficePro2003-W7x32	0.45%	OfficePro2003-WinXP	0.35%
OfficePro2003-WinXP	0.37%	OfficePro2003-W7x32	0.41%	OfficePro2003-W7x64	0.42%	OfficePro2003-W7x64	0.35%
OfficePro2003-W7x32	0.32%	OfficePro2003-W7x64	0.37%	Safari157-W7x64	0.39%	Wireshark-W7x32	0.09%
OfficePro2003-W7x64	0.31%	Wireshark-W7x32	0.07%	Wireshark-W7x32	0.10%	eraser-W7x32	0.05%
Wireshark-W7x32	0.06%	HxD171-W7x32	0.04%	Wireshark-W7x64	0.02%	Wireshark-W7x64	0.05%
eraser-W7x32	0.01%	eraser-W7x32	0.02%	eraser-W7x32	0.02%	AdvKeylogger-WinXP	0.03%
AdvKeylogger-WinXP	0.01%	Wireshark-W7x64	0.02%	InvSecrets21-WinXP	0.00%	InvSecrets21-WinXP	0.00%
Wireshark-W7x64	0.00%	AdvKeylogger-WinXP	0.01%	UPX-W7x32	0.00%	UPX-W7x32	0.00%
UPX-W7x32	0.00%	InvSecrets21-WinXP	0.00%	sdelete-W7x32	0.00%	sdelete-W7x32	0.00%
HxD171-W7x32	0.00%	UPX-W7x32	0.00%	UPX-W7x64	0.00%	UPX-W7x64	0.00%
UPX-W7x64	0.00%	UPX-W7x64	0.00%	sdelete-W7x64	0.00%	sdelete-W7x64	0.00%

Legend	True positive	True negative	False positive	False negative	Different OS

In the M57 results of Table 7, green cells indicate true positives, which are confirmed installed or uninstalled programs based on the scenario documentation, other published analysis, and direct forensic examination of the scenario images. White cells are true negatives, similarly verified. Red cells indicate false positives, which we define as weighted file % scores above the lowest true positive. Blue cells are false negatives, which we define as a known installed applications with a weighted file % lower than at least one true negative. Yellow cells indicate other OS versions of detected applications. For the true positives, Python and Firefox installations are confirmed for all four systems. For the Charlie system, Thunderbird is also confirmed by the scenario documentation, and Invisible Secrets is suggested by the scenario documentation ("...emails proprietary information steganographically hidden in JPEG image...") and confirmed by Roussev and Quates as well as a direct examination of the image. The presence of TrueCrypt on the Jo system, Advanced Keylogger on the Pat system, and Chrome on the Terry system are all confirmed in the scenario documentation. Advanced Keylogger is also confirmed in the scenario documentation to have been uninstalled prior to the Pat 2009-12-11 image.

We examined the scenario images directly using Autopsy 4.0.0 in an effort to understand the apparent false positives and the lone false negative (eraser on the Terry image). A summary of our preliminary

findings is below in Table 8. A more extensive analysis is underway to establish if these are in fact false positives, or if some of them represent as yet undocumented true positives. The results of this analysis will be reported in future work.

Table 8: False positive and false negative preliminary analysis

(System(s)) **Application**	Preliminary Analysis
(Charlie/Jo/Terry) **Safari**	Apple's QuickTime and Apple's software update applications are present on the Charlie and Jo systems and may explain the Safari results due to catalog artifacts in common (Safari would include the Apple software update application and possibly QuickTime as well). The Terry system also indicated Safari, although at a lower level than the Charlie and Jo systems, but the Terry system does not show indications of a QuickTime installation.
(Jo/Pat/Terry) **Thunderbird**	Thunderbird is known to have been installed on the Charlie system on 11-12-2009, but is not documented or apparent on the other three systems. It is possible that Thunderbird was installed on all four systems on 11-12-2009 but immediately uninstalled on the three non-Charlie systems.
(Pat/Terry) **HxD**	HxD may have been installed and uninstalled between snapshots, hence no entries were found in locations like Program Files. The Cygnus hex editor was confirmed on the Charlie system, so the scenario operators are know to have installed a hex editor, although a different one than HxD detected on the Pat and Terry systems.
(Terry) **Winzip**	Possibly due to compression libraries bundled in Windows Vista and also used by Winzip, but not bundled in Windows XP.
(Terry) **Eraser**	Likely due to a difference in application versions between the catalog and the M57 image. Most of the eraser sectors in the catalog come from the eraser.exe file, hence a minor change in the compiled code would prevent sector matches. The eraser application has a small number of files, hence is more susceptible to such a variation than other applications with large numbers of files and hence unchanged sectors across versions.

Of particular interest in the M57 results is the successful detection of Advanced Keylogger on the Pat system after uninstallation and continued use. Such detections are the main goal of our work and is distinct from other work such as Roussev and Quates that relied on mid-scenario snapshots to detect Advanced Keylogger. In contrast, our approach detected Advanced Keylogger using only the final scenario snapshot, after Advanced Keylogger had been uninstalled and the system used for five additional days. Figure 4 shows the presence and persistence of Advanced Keylogger sector artifacts over the life of the scenario. The data consists of 17 snapshots over 26 calendar days, where days without snapshots are indicated by an asterisk along the X-axis of Figure 4. The vertical axis in the graph, sector %, is the matched sectors as a fraction of the total sectors associated with Advanced Keylogger in the catalog. Advanced Keylogger was installed between the 12/2 and 12/3 snapshots, and uninstalled between the

12/4 and 12/7 snapshots. Subsequent system usage further destroyed probative sectors, yet our weighted file % measure still detected Advanced Keylogger in the 12/11 snapshot (21.97% based on the remaining 24 sectors from 8 different files). We speculate that 100% of the catalog sectors were not matched in the 12/3 and 12/4 snapshots due to slight differences in artifacts created during installation and use of Advanced Keylogger the different systems of the catalog and the M57 scenario.

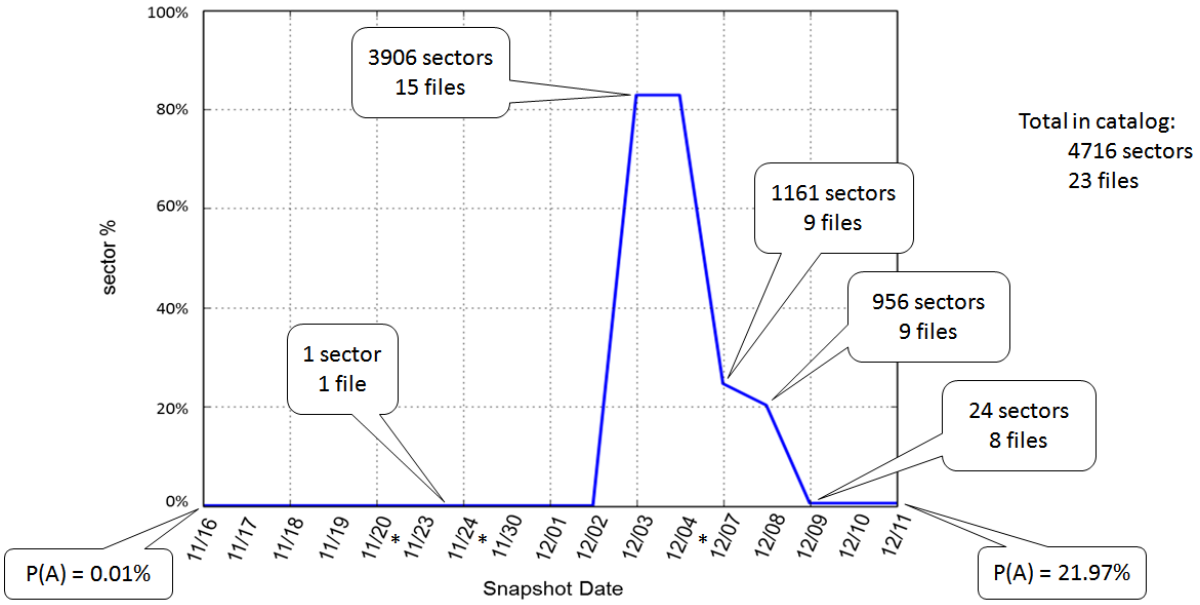

Figure 4: Sector artifact persistence for Advanced Keylogger on Pat's M57 system

One unresolved issue is to determine the threshold at which an application should be considered present or previously present. While the M57 results might indicate a weighted file % threshold of about 3%, the contents of deleted files are modified (destroyed) over time, so a single threshold for uninstalled applications is unlikely to exist. However, we are conducting related work to model the persistence of deleted files over time under different artifact and system usage conditions. This related works aims to provide a basis for asserting the implications of a particular weighted file % for a specific application after a known amount of time and activity. Additionally, our catalog of 16 applications tested on the four images of the M57 data set is not large enough to conclude statistical significance. However, as a practical matter, the current use cases for our approach are (a) for an analyst to work down the list in decreasing weighted file % score until applications are no longer confirmed or no longer of interest, or (b) to have a specific set of applications of interest and only seek to confirm those in decreasing order of weighted file %. Regarding the first use case, our M57 results indicate that present or previously present applications almost always score higher than non-present or never installed applications. The second use case also addresses part of the scalability question, in that our approach need not catalog a great number of applications in order to be of use, but rather only catalog applications of interest to the analyst.

5. CONCLUSIONS AND FUTURE WORK

In this work, we leveraged an existing catalog of full-file artifacts from specific applications to detect and reason over matching sectors recovered from media of interest. We used these matching sectors to suggest past uninstalled applications on test images and the drive images of the M57 Patents Scenario. Our results suggest that:

- Partial file contents (traces) remain after files are deleted due to application uninstallation
- These traces can be used to suggest the past presence of uninstalled applications

Current approaches to determine prior application presence rely on intact artifact recovery, log analysis, Windows registry analysis, and trace evidence analysis. Our approach complements these methods, especially when intact artifacts and traces are not available and the Windows registry has been cleaned or is unavailable, e.g., on a non-Windows system.

Our approach requires that applications of interest be processed into the catalog prior to trace detection and computation. While processing new applications is relatively straightforward, it does require resources as well as knowledge of, and access to, applications of interest. Additionally, utilities that overwrite unallocated space would likely defeat our approach as we rely on fragments of deleted files residing in this unallocated space. Our approach is also vulnerable to deliberate deception, as the placement of specific file fragments in the unallocated space of a device or image, or even the creation and deletion of selected full-file artifacts, would cause spurious suggestion of an application that in fact had never been installed.

We are considering combining the weighted sector and weighted file measures, and also adding sector entropy and relative partial artifact location on the media to our measure of application presence calculation. Additionally, we are examining methods for more robust and precise noise reduction at the point of catalog creation, and we are considering sector differencing as an alternative to file differencing. Future work will extend our approach to malware applications and mobile platforms.

ACKNOWLEDGEMENTS

[removed for review]

AUTHOR BIOGRAPHIES

[removed for review]

REFERENCES

Collange, S., Dandass, Y. S., Daumas, M., & Defour, D. (2009). Using graphics processors for parallelizing hash-based data carving. In *System Sciences, 2009. HICSS'09. 42nd Hawaii International Conference on* (pp. 1-10). IEEE.

Dandass, Y. S., Necaise, N. J., & Thomas, S. R. (2008). An empirical analysis of disk sector hashes for data carving. *Journal of Digital Forensic Practice*, *2*(2), 95-104.

NPS-DEEP. (2015). Hashdb. Last accessed 10.4.15, https://github.com/NPS-DEEP/hashdb.

Forte, D. V. (2004). The "Art" of log correlation: Tools and Techniques for Correlating Events and Log Files. Computer Fraud & Security, 2004(8), 15-17.

Foster, K. (2012). *Using distinct sectors in media sampling and full media analysis to detect presence of documents from a corpus* (Doctoral dissertation, Monterey, California. Naval Postgraduate School).

Garfinkel, S. (2012). Digital forensics XML and the DFXML toolset. *Digital Investigation*, *8*(3), 161-174.

Garfinkel, S. L. (2013). Digital media triage with bulk data analysis and bulk_extractor. Computers & Security, 32, 56-72.

Garfinkel, S. L., & McCarrin, M. (2015). Hash-based carving: Searching media for complete files and file fragments with sector hashing and hashdb. *Digital Investigation*, *14*, S95-S105.

Garfinkel, S., Nelson, A., White, D., & Roussev, V. (2010). Using purpose-built functions and block hashes to enable small block and sub-file forensics. *digital investigation*, *7*, S13-S23.

Garfinkel, S., Nelson, A. J., & Young, J. (2012). A general strategy for differential forensic analysis. *Digital Investigation*, *9*, S50-S59.

Koppen, J., Gent, G., Bryan, K., DiPippo, L., Kramer, J., Moreland, M., & Fay-Wolfe, V. (2013). Identifying Remnants of Evidence in the Cloud. In *Digital Forensics and Cyber Crime* (pp. 42-57). Springer Berlin Heidelberg.

Laamanen, M., Nelson, A. (2014). NSRL Next Generation - Diskprinting. Forensics @ NIST, Gaithersburg, MD, December 3, 2014. Last accessed 10.4.15, http://www.nsrl.nist.gov/Documents/Diskprints.pdf.

Ligh, M. H., Case, A., Levy, J., & Walters, A. (2014). The art of memory forensics: detecting malware and threats in windows, linux, and Mac memory. John Wiley & Sons.

Nelson, A., Laamanen, M., Tebbutt, J., Long, D. (2014) Indexing the Windows® Registry for Software Detection. The American Academy of Forensic Sciences 66th Annual Scientific Meeting , February 20, 2014, Seattle, WA. Last accessed 10.4.15, http://www.nsrl.nist.gov/Documents/20140220%20Diskprint%20AAFS.pdf.

Nelson, A. J., Steggall, E. Q., & Long, D. D. (2014). Cooperative mode: Comparative storage metadata verification applied to the Xbox 360. *Digital Investigation*, *11*, S46-S56.

NIST. (2015). Diskprint Data Downloads. Last accessed 10.4.15, http://www.nsrl.nist.gov/dskprt/sequence.html.

NIST. (2012). Recommendation for Applications Using Approved Hash Algorithms, Special Publication 800-107 Revision 1. 2012. Last accessed 10.5.15. http://csrc.nist.gov/publications/nistpubs/800-107-rev1/sp800-107-rev1.pdf

Quick, D., & Choo, K. K. R. (2013). Digital droplets: Microsoft SkyDrive forensic data remnants. *Future Generation Computer Systems*, *29*(6), 1378-1394.

Roussev, V., & Quates, C. (2012). Content triage with similarity digests: the M57 case study. *Digital Investigation*, *9*, S60-S68.

Wong, L. W. (2007). Forensic analysis of the Windows registry. *Forensic Focus*, *1*.

Woods, K., Lee, C. A., Garfinkel, S., Dittrich, D., Russel, A., & Kearton, K. (2011). Creating realistic corpora for forensic and security education. ADFSL Conference on Digital Forensics, Security and Law.

Young, J., Foster, K., Garfinkel, S., & Fairbanks, K. (2012). Distinct sector hashes for target file detection. *Computer*, (12), 28-35.

Zobel, J., & Moffat, A. (1998). Exploring the similarity space. In *ACM SIGIR Forum* (Vol. 32, No. 1, pp. 18-34). ACM.