EAI Endorsed Transactions
on Security and Safety

Research Article **ICST.ORG**

Identifying forensically uninteresting files in a large corpus

N. C. Rowe[1]*

[1] U.S. Naval Postgraduate School, GE-328, 1411 Cunningham Road, Monterey, CA 93943 USA

Abstract

For digital forensics, eliminating the uninteresting is often more critical than finding the interesting. We discuss methods exploiting the metadata of a large corpus. Tests were done with an international corpus of 262.7 million files obtained from 4018 drives. For malware investigations, we show that using a Bayesian ranking formula on metadata can increase malware recall by 5.1 while increasing precision by 1.7 times over inspecting executables alone. For more general investigations, we show that requiring two of nine criteria for uninteresting files, with exceptions for some special interesting files, can exclude 77.4% of our corpus. For a test set that was manually inspected, interesting files identified as uninteresting were 0.18% and uninteresting files identified as interesting were 29.31%. The generality of the methods was confirmed by separately testing two halves of our corpus. This work provides both new uninteresting hash values and programs for finding more.

Keywords: digital forensics, metadata, files, corpus, data reduction, hashes, triage, whitelists, classification, malware, camouflage.

Received on 01 November 2015, accepted on 15 December 2015, published on 08 December 2016

doi: 10.4108/eai.8-12-2016.151725

1. Introduction

As digital forensics has grown, larger and larger corpora of drive data are available. To speed subsequent processing, it is essential that the drive triage process first eliminate from consideration those files that are clearly unrelated to an investigation [13]. This can be done either by directly eliminating files from drive images or by removing their indexing. We define as “uninteresting” those files whose contents do not provide forensically useful information about usage of a drive in the form of either user-created or user-discriminating information. Mostly these are operating-system and applications-software files plus common Internet downloads. (Metadata on uninteresting files may still be interesting as in indicating time usage patterns.) This definition applies to most criminal investigations and data mining tasks. It can be further refined for malware investigations where the "user" of interest is the malware.

We can confirm that files are uninteresting by opening them and inspecting them. Additional files may also be uninteresting depending on the type of investigation, such as medical records in an investigation of accounting fraud. Uninteresting files usually comprise most of a drive, so eliminating them significantly reduces the size of the investigation. Unfortunately, uninteresting files occur in many places on a drive, so finding the uninteresting is not always straightforward.

Most decisions about interestingness can be made from file-directory metadata without examining file contents. That is important because directory metadata requires roughly 0.1% of the storage of file contents. Directory metadata can provide the name of a file, its path, its times, and its size, and this can give us a good idea of the nature of a file [1]. We also include in this the hash value computed on the contents of the file, which enables recognition of file copies. Forensic tools like SleuthKit routinely extract directory metadata and hash values from drive images.

We can generally eliminate files whose hash values match those in published "whitelisting" sets [8]. However,

*Neil C. Rowe. Email: ncrowe@nps.edu.

published hash values miss many kinds of files. This paper will discuss methods for improving this performance by additional filtering based on analysis of a large corpus of drives, in particular by correlating files across it. This provides both a new set of hash values and new methods for finding them.

2. Previous work

The standard forensic approach today is to eliminate from consideration those files whose hash values match those in the Reference Data Set of the National Software Reference Library (NSRL-RDS) from the U.S. National Institute of Standards and Technology (NIST). The quality of the data provided in the NSRL is high [10]. However, tests found that it did not provide much coverage [16]. Less than one file of four in our international corpus appeared in the NSRL, and there were surprising gaps in the coverage of well-known software. In part this is due to NIST's usual approach of purchasing software, installing it, and finding hash values for the files left on a drive. This will not find files created only during software execution, most Internet downloads, and user-specific configuration files. Furthermore, the fraction of files recognized by NSRL on a typical drive is decreasing as storage capacity increases. To fill the gap, commercial vendors like bit9.com and hashsets.com sell additional hash values beyond NSRL.

The work [4] investigates the problem of recognizing uninteresting files and suggests that pieces of files need to be hashed separately, a technique that considerably increases the workload. The work [19] details efficient methods for indexing and matching hash values found on files. Many of the issues are similar to the important problems of file deduplication [12] and file-existence checking [20] for which file hashes are useful. Analogous work has examined elimination of uninteresting network packets from analysis [6].

The work [19] investigated methods for improving a hash set of uninteresting files by using locality and time of origin to rule out portions of the hash values in the NSRL, and their experiments showed they could reduce the size of the hash set by 51.8% without significantly impacting performance. They also identified as uninteresting those files occurring on multiple drives, similarly to [16]. Their experiments were based on less than one million files, a weakness since files in cyberspace are highly varied. A more serious weakness is that they used human expertise to provide guidance in indicating uninteresting files, and then trained a model. This seems risky because it may miss forensic evidence that is atypical or unanticipated. Legal requirements also often dictate that forensic evidence be complete, in which case elimination of forensic evidence must be done by better-justified methods than heuristic ones.

3. Experimental setup

The experiments reported here, except for some in section 5.6, were done with a corpus assembled in January 2015. It consisted of 4018 drives with 262.7 million files having 35.80 million distinct hash values. It included the January 2015 version of the Real Drive Corpus [5] (3397 drives and 104 million files purchased as used equipment) supplemented with files from classroom and general laboratory computers at our school (157 drives and 126 million files) and miscellaneous sources including our laboratory (464 drives and 33 million files). The school computers were centrally managed and had much software in common, thus providing data representative of large organizations. The miscellaneous sources included files extracted from compressed archives in the Real Drive Corpus including ZIP, GZIP, RAR, and CAB formats.

We extracted directory metadata with SleuthKit and the Fiwalk tool for the non-school drives and with our own extraction programs calling upon the operating system for the school drives. All these drives had normal users, and we saw little concealment or camouflage on them. Thus hash values on their contents should not show any manipulation, an issue important in some forensic applications [7]. We still checked, however (see Table 7).

We also obtained the April 2015 version of the NSRL-RDS from www.nsrl.nist.gov. Our malware work used SHA-1hash values and our general-file work used MD5 hash values. Both are widely used and are catalogued for the NSRL.

The programs reported here were implemented in Python 3 with only default packages.

4. Finding uninteresting files in malware investigations

For malware investigations, uninteresting files are those not containing malware nor affected by malware. A sufficient condition for most files is if their hash values are unmodified from their initial values on installing the file. But this can entail looking up a large number of hash values, and there are many nonmalicious reasons to change a file’s contents. Thus it is valuable to have more specific criteria for when a file is worth checking. Although there has been much work on malware detection [3, 9, 11], it is almost entirely focused on analysis of file and packet contents, and methods that examine the smaller amount of metadata and hashes could be a useful first step.

4.1. Testing malware clues

The following methods were used to identify malware in our corpus [17]:

- Files in our corpus whose SHA-1 hash values were tagged as “threats” in the database of the Bit9 Forensic Service (www.bit9.com).

- Files in our corpus whose computed hash values matched those of malicious software in the Open Malware corpus (oc.gtisc.gatech.edu:8080) of about 3 million files.
- Files in our corpus whose computed hash values matched those of malicious software in the VirusShare database (virusshare.com) of about 18 million files, after mapping its MD5 hash values to SHA-1.
- Files identified as threats by Symantec antivirus software (www.symantec.com/endpoint-protection) in a sample of files extracted from the corpus. The sample was downloaded to a home computer with the antivirus software installed, and every file that Symantec complained about was recorded. Only a sample could be tested because the corpus is too big to store online and extraction of files is time-consuming. The sample included about 300,000 random files plus 30,000 embedded files of type zip, gzip, cab, 7z, and bz2 because of their higher fraction of malware. Also included were 7,331 files from the Open Malware corpus whose hashcodes matched those of our corpus files, of which only 721 were flagged as malicious by Symantec.
- Files identified as threats by ClamAV open-source antivirus software (www.clamav.net) in the same sample of files tested by Symantec.

398,949 distinct hash values of malware were found in the 31 million distinct hash values in our 2015 corpus. Bit9 identified 238,704, Open Malware matched 4,786, VirusShare matched 145,449, Symantec identified 1,401, and ClamAV identified 877. Surprisingly, there was little overlap between the malware identified by the five methods.

For testing, we created a control set from a random sample of 303,322 distinct hash codes from files from our 2015 corpus minus those that appeared in any of the malware sets. While this did not exclude unrecognized malware, the low frequency of recognized malware suggests that the unrecognized malware was unlikely to have much statistical effect on the comparison results. A taxonomy of extensions, top-level directories, and immediate directories was used that we have been developing [16].

Table 1 shows the results of testing of a variety of possible metadata clues to malware. Only clues with some observed promise are shown. The quantity listed is the number of standard deviations for the occurrence of malware greater than the expected value, 0.0013 (the fraction of malware in the corpus) times the size of the sample. The count used was the number of distinct malware hash values associated with the clue, since we saw drives where the same malware hash value occurred in hundreds of files it had infected. The five malware identification methods clearly seem to be addressing different kinds of files, consistent with the results of [11] on a larger number of malware detection methods but fewer files. Taking as valid those clues occurring more than two standard deviations in the same direction on at least three of the five methods, the positive clues were files whose size had a natural logarithm of more than 15, files at the top level of the directory hierarchy, deleted files (not helpful because many were deleted by anti-malware software), files where the file extension category was incompatible with its type based on its header and other "magic numbers", files created at odd creation times for their directory, files with single-occurrence hash values, files with unusual characters in their paths, executables, files related to hardware, temporary files, and files not in major categories. Negative clues were files at level 10 or more in the file hierarchy, double extensions, files with no extension, video extensions, engineering-related extensions, game top-level directories, operating-system immediate directories, backup immediate directories, and data-related immediate directories.

One surprising result was that the number of drives on which malware occurred could be considerable (Figure 1). One malware occurred on 296 drives in our corpus, and many other kinds of malware that occurred on 10 or more drives. This result challenges the notion of using "reputation" as a factor in discovering possible new malware, since usually reputation is estimated as the number of places in which something occurs.

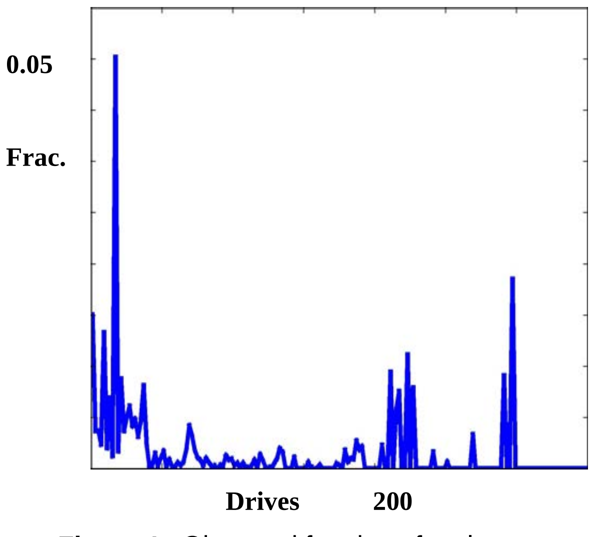

Figure 1. Observed fraction of malware versus number of drives on which a hash value appears. Highest peak is 0.08 at 16 drives.

Table 1. Strengths of various malware clues, measured as number of standard deviations plus or minus of the expected random frequency, counting by hash values.

Malware set	Bit9	Open Malware	VirusShare	Symantec	ClamAV
Total count in corpus identified as malicious	1,201	7,331	151,621	626	1,662
Total count in corpus identified as nonmalicious	303,332	303,332	303,332	303,332	303,332
File size 0 or 1	-0.3	-0.7	-0.9	-0.2	-0.3
Rounded log file size = 5	-5.5	-19.5	+112	-5.2	-7.1
Rounded log file size = 10	+9.3	+22.5	-9.0	-6.5	-3.1
Rounded log file size = 15	+2.5	+23.4	+16.8	+4.6	+16.9
Level = 1	+26.2	+9.7	-47.1	-2.4	+14.7
Level = 5	+2.5	+7.4	-85.2	-5.3	+5.5
Level = 10	-6.6	-15.8	-8.2	-1.2	-8.0
Level = 15	-2.3	-5.3	-24.5	-1.6	-2.7
Deleted file	+4.2	+3.1	+1159	-1.4	+10.6
Extension/ libmagic incompatible	+6.4	-4.1	-60.0	+9.1	+6.6
Odd creation time	+17.7	+9.1	-46.1	-2.0	+13.2
Rare hash value	-0.3	+2.1	-1.6	-0.2	-0.3
Rare extension	+2151	+583	-24.4	+2874	+3287
Double extension	-1.6	-6.2	-17.0	+12.8	+5.3
Long extension	-0.9	-1.5	+8.0	-0.7	-0.2
Encryption extension	-1.9	-4.2	-17.6	-0.6	-2.2
Odd characters in path	+6.4	+7.2	+29.0	-3.6	-0.6
Repeated pattern in path	-0.4	+0.8	+16.4	-0.3	+75.5
Misspelling in path	-1.2	-1.2	-11.9	-0.8	+0.1
Extension type: None	-10.5	-27.5	-17.9	-8.2	-12.8
Extension type: Photograph	-5.3	-14.8	+110	-3.1	-4.6
Extension type: Link	+4.8	-1.8	-15.9	-1.0	-1.1
Extension type: Video	-2.2	-5.0	-12.6	-1.6	-2.5
Extension type: Executable	+54.5	+162	-166	+18.2	+25.3
Extension type: Drive image	-1.1	-2.3	-8.1	-0.8	-1.3
Extension type: Query	-0.9	-2.3	+20.8	-0.7	-1.1
Extension type: Installation	+6.4	-5.0	-43.8	-2.2	-0.4
Extension type: Networking	-0.8	-0.9	-8.5	-0.6	-0.9
Extension type: Hardware	-0.4	+1.8	-14.0	-0.9	-1.5
Extension type: Engineering	-2.5	-6.2	-27.0	-1.8	-3.0
Extension type: Miscellaneous	-1.5	-1.1	+19.0	-1.1	+2.1
Top-level directory type: Hardware	-2.3	+3.0	+47.9	+11.3	+42.4
Top-level directory type: Temporaries	+2.6	+8.1	-121	+14.4	-2.3
Top-level directory type: Games	-2.5	+3.5	-46.1	-2.7	-3.2
Top-level directory type: Miscellaneous	+24.7	+20.2	-42.1	-1.8	+33.8
Immediate directory type: Operating system	+6.0	+9.5	-56.0	-8.8	-9.1
Immediate directory type: Backup	-5.7	-16.6	-80.0	-5.5	-5.7
Immediate directory type: Audio	-2.2	-3.8	+90.7	-2.6	+0.3
Immediate directory type: Data	+2.5	-6.4	+60.0	-2.0	-4.4
Immediate directory type: Security	+14.5	+20.8	+18.3	+20.1	+2.0
Immediate directory type: Games	+1.0	+10.8	+907	-0.2	-0.5
Immediate directory type: Miscellaneous	+15.8	+28.4	-22.5	-1.5	+50.1

4.2. Building a better quick scan

These results can reduce the time to find malware on a system. Malware could hide anywhere, but our conditional probabilities enable us to rank its likelihood from context so we can try the most likely places first. This is useful in designing "quick scans" for malware in which we only check part of a drive.

To compute odds of each clue, the set of hash values in our corpus was split randomly. Files were found corresponding to the two half-sets of hash values, about 124 million files each. Conditional probabilities for the clues discussed above were calculated and converted into odds for one half of the corpus. Additional clues that were tested were the actual file extension, top-level directory, bottom-level directory, and file name. Clues relating to the times of the file were excluded, however, because prediction is the goal and there is no guarantee that current time patterns will reoccur. Clues were only included if they occurred at least R times and were significant at a level greater than 2.0 standard deviations above or below the expected value. Clues were then tested for each file in the other half of the corpus. Assessment was by a normalization of the Naïve Bayes odds formula:

$$o(\mathrm{M} \mid (\mathrm{C}_1 \wedge C_2 \wedge \ldots \wedge C_M)) =$$
$$[o(\mathrm{M} \mid \mathrm{C}_1) o(\mathrm{M} \mid \mathrm{C}_2) \ldots o(\mathrm{M} \mid \mathrm{C}_M) / (\mathrm{o(M)})^{N-1}]^{(1/\mathrm{N})}$$

Here o mans odds, M means "file was malicious", and C means clue. Odds were calculated with Laplace-smoothing constant K:

$$o(\mathrm{M} \mid \mathrm{C}) = [(n(\mathrm{M}\&\,\mathrm{C}) + (K * \mathrm{n(M)} / \mathrm{n(O)}) / \mathrm{n(M)}] /$$
$$[(\mathrm{n(O\&\,C)} + K) / \mathrm{n(O)}]$$

Here n means count and O means "file is nonmalicious". Normalization was necessary because files varied in the number of significant clues they presented.

Two constants R and K need to be optimized. R is the threshold for reliable counts on clues, and K represents the "background noise" of the clue. We did experiments on a different random sample of 30% of our corpus to vary R and K and measure the F-score (Table 2). There was not much variation in effect, but the best values appeared to be R=15 and K=30 and these were used in subsequent experiments.

Table 2. Effects of varying R (minimum count) and K (damping constant) on malware F-score.

	R=10	R=20	R=40	R=100
K=1	.1558	.1558	.1549	.1511
K=10	.1560	.1560	.1551	.1505
K=30	.1566	.1566	.1548	.1505
K=100	.1554	.1555	.1546	.1482

To test ability to rank malware, 100 evenly spaced threshold values on the combined odds were chosen and recall (fraction of malware over the threshold) and precision (fraction of files over the threshold that were malware) were calculated. Recall is important because a high value reduces the need and rate of doing full scans for malware, but precision is important too since a low value requires more files to be scanned unnecessarily. F-score is the classic way to trade them off. Malware was defined by our consensus list of malicious hashcodes, the union of the results of the five malware-identification methods.

We conducted this experiment three times on three random partitions of our corpus (with a total of 612,818 instances of malware and 128,776,919 instances of non-malware for training), using one half for training and one half for testing. The recall values were 0.343, 0.305 and 0.333; the precision values were 0.213, 0.211, and 0.211; and the resulting F-scores were 0.263, 0.249, and 0.259. So there was not much variation in the results, and this supports the generality of our corpus for training purposes. But if one is willing to accept a much lower precision of 0.010 with our methods, we can obtain a better recall in finding malware of 0.650. By comparison, selecting only the executable files gave 0.005 precision (for 22,940,397 executables total) and 0.190 recall (for 116,235 malicious executables) for an F-score of 0.0097. Hence our methods give 5.1 times better precision with 1.7 times better recall over inspecting executables alone. Similarly, selecting only the files in operating-system top-level directories gave 0.003 precision and 0.189 recall, and selecting only the files in applications top-level directories gave 0.00031 precision and 0.056 recall, so searching for malware in particular directories is an even poorer strategy. A possible objection is that malware in executables, the operating system, and applications directories is more serious than in other places, but this is questionable since malware loads from many kinds of files today.

Our clues are straightforward to compute, and can be done on a drive once upon setup, then recalculated every time a file changes. Note they will be significantly faster to obtain than signatures of files because most involve metadata, with only a few clues requiring computation of a hash value on a file, something often computed routinely in investigations.

5. Finding uninteresting files in standard investigations

In standard criminal investigations, files can be judged interesting or uninteresting by a much wider range of criteria. However, the criteria can be considerably stronger than with malware; for instance, a file that occurs on 100 different drives is unlikely to provide the evidential specificity to help in a criminal investigation. Again we use the definition that uninteresting files do not contain user-created nor user-discriminating data.

5.1. Proposed uninteresting-file identification methods

Nine methods to identify uninteresting files and then their hash values were investigated as summarized in Table 3. Parameters of these methods were set by the experiments reported in section 5.4. The methods were:

- HA, frequent hashes: Files on many different drives with the same hash value on their contents. Hash values that occur on only two drives in a corpus could suggest sharing of information between investigative targets. But hash values occurring more often are likely to be distributions from a central source and are unlikely to be forensically interesting. An example in our corpus was C161336552062A51C5130ECAB3F59BF3 which occurred on five drives as Documents and Settings/Administrator/ Local Settings/Temporary Internet Files/ Content.IE5/ ZBX73TSW/tabs[1].js, Documents and Settings/ Friend/Local Settings/Temporary Internet Files/Content.IE5/0P2NOXY3/ tabcontent[1].js, deleted file Documents and Settings/user/Local Settings/Temporary Internet Files/Content.IE5/ KLM7E1U9/tabcontent[1].js, deleted file tabcontent[1].js with lost directory information, and deleted file E5/322B0d01. It did not occur in NSRL. It represents tab information in a Web browser. The threshold must be on number of drives, not the number of files, since installation and backup copies of files on the same drive are common.

- PA, frequent paths: Files with the same full path (file name plus directories) on many different drives. Frequently occurring paths are likely default locations for software. Such paths include different versions of the same file, such as configuration files for different users or successive versions of an updated executable. An example from our corpus was restore/WINDOWS/inf/fltmgr.PNF which occurred on six drives, and none of its hash values were in NSRL.
- BD, frequent bottom-level (“immediate”) directory-filename pairs: Files whose pair of the file name and the immediate directories above it occurred especially frequently. This will catch versions of software in different directories under different versions of an operating system. Examples from our corpus were WINDOWS/$NtUninstallWIC$ with 60 occurrences in our corpus and Config.Msi/4a621.rbf with 20 occurrences; neither of them had any hash values in NSRL. This will also catch many hidden files in common directories.
- TM, files with clustered creation times: Files created within a short period of time on the same drive. Such clusters suggest automated copying from an external source, particularly if the rate of creation exceeded human limits. An example from our corpus were seven files created on one drive within one second under the directory Program Files/Adobe/Adobe Flash CS3/adobe_epic/ personalization: pl_PL, pl_PL/., pl_PL/.., pt_BR, pt_BR/., pt_BR/.., and pt_PT. All were 56 bytes, and two hash values were not in NSRL. Creation times are more useful than access and modification times because they are often installation times.
- WK, files created in busy weeks: Files created unusually frequently in particular weeks across all drives, which suggest software updates. A period of a week is appropriate since it takes several days for most users to download an update. Figure 2 shows some example sharp peaks in a typical distribution of creation times by week in our corpus. We first find "busy" weeks, then "busy" directories (full path minus the file name) in the busy weeks, those whose frequency of creation was a threshold number of times greater than their average creation time per week. The hash values for files in those busy directories on those busy days are then the proposed as uninteresting.
- SZ, files with frequent sizes: Files with unusually common pairs of size and extension. This recognizes

Table 3. Methods for identifying uninteresting files.

Method	Scope	Primary data	Secondary data	Considers deleted files?
HA	Corpus-wide	Hash values	None	Yes
PA	Corpus-wide	Full paths	None	Yes
BD	Corpus-wide	File name and containing directory	None	No
TM	Single-drive	Creation times within the minute	None	No
WK	Corpus-wide	Creation times within the week	Paths minus file name	No
SZ	Corpus-wide	File sizes	Full paths	No
CD	Single-drive	Full paths in a directory	File extensions	No
TD	Single-drive	Front and inside of paths	None	Yes
EX	Single-drive	File extension	None	No

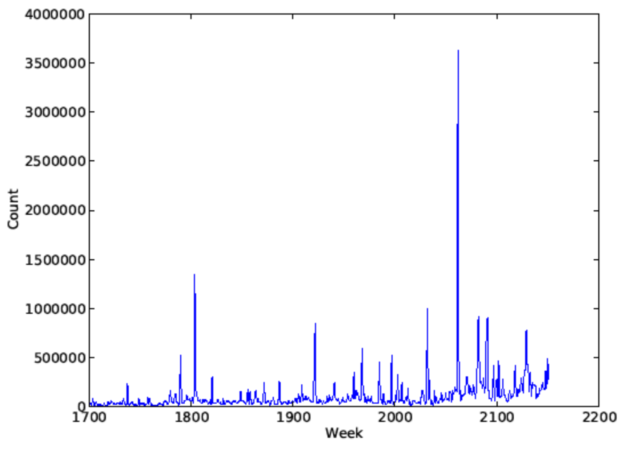

Figure 2. Portion of histogram of creation times per week in our corpus.

fixed-size formats with different contents like screensaver images. To reduce false matches it is important to apply the additional criterion that the extension must occur unusually often in all files of that size. Examples from our corpus were all 31 files of size 512 in directory System/Mail/00100000_S, and 8 files of size 2585 in directory Program Files/ Sun/JavaDB/docs/html/tools/ctoolsijcomref with extension html, none of which had hash values in NSRL. Files less than a minimum size are also very unlikely to contain forensically useful information. For example, there were 4,223,667 files of zero length in the corpus. We set a minimum of 6 bytes from experiments with samples of the corpus.

- CD, contextually uninteresting files. Directories in which more than a certain fraction of files were already identified as uninteresting by other methods suggest that the rest are also uninteresting by "contagion". An example was directory Program Files/Yahoo!/ Messenger/skins/Purple/images and the previously unclassified files bg_hover.png, _GLH0307.TMP, and bg_selected.png. This method can be used to bootstrap better performance on each run of a corpus.
- TD, files in known uninteresting top-level or mid-level directories. We manually compiled lists from study of the files remaining after filtering on the other criteria mentioned here, and obtained 5752 top-level and 750 mid-level directories. Example uninteresting top-level directories were APPS/Symantec AntiVirus, GTA Vice City, FOUND.029, Program Files (x86)/AutoCAD 2008, and sys/uninstall; example mid-level directories were /Help/ and /Java Update/. It is important not to exclude all applications and operating-system directories (WINNT, etc.) because some software keeps user information such as logs and configuration files there, and in some cases, user data.
- EX, files with known uninteresting extensions. Some extensions are exclusively associated with operating systems and software, such as exe, mui, and chm. We used a classification of file extensions that we are developing [16] that maps 14,396 extensions to 45 categories. The categories we labelled as nonuser and thus uninteresting were operating-system, graphics, database, help, executable, disk image, XML, geography, copies, dictionary, query, integer, index, configuration, installs and updates, security, known malicious, games, engineering, science, signals, and virtual machines. Some investigations may want to shorten this list, and exceptions can also be made with the methods described in the next section. Files with no extensions and extensions with more than one category were not considered ruled out by this criterion.

13.6% of the files in the corpus were identified as deleted. We excluded these as sources of new hash values with three exceptions because [16] found that deleted files can have corrupted metadata. Directories were often missing for deleted files, and we even saw inconsistencies in the sizes of reported by SleuthKit for deleted files. However, the same hash value or the same path appearing repeatedly is unlikely to be a coincidence even if they were all deleted, and long directory paths are likely correct, so we ignored deletion status in methods HA, PA, and TD.

Files for which drive acquisition could not find hash values can also be eliminated on some criteria; missing hash values can occur for deleted files with incomplete metadata. For those we applied the BD, TD, and EX criteria to individual file records, and eliminated files matching at least two criteria. HA will not work with missing hash values; PA will not work with the frequent missing fronts of deleted files; TM and WK will not work with unreliable timestamps; SZ will not work with unreliable file sizes; and CD will not work when whole directories are deleted. We also eliminated default directory files in this final filtering.

5.2. Identifying explicitly interesting files

Since reducing false positives with uninteresting files (interesting files that were incorrectly identified as uninteresting) can be very important in a forensic investigation, despite our low observed rates, we investigated six methods to explicitly identify potentially interesting files, some of which were studied in [18]. The first five look for clues of deliberate concealment [2], a definite possibility in many criminal investigations. Several software packages also claim to find anomalous files (e.g. Redwood, try.lab41.org), and related work has addressed the finding of traces of intrusions in storage [14].

(i) Files with hash values that occurred only once in our corpus where their same path occurs many times with a predominant different hash value. These could be deliberate camouflaged.

(ii) Hashes that occurred just once in our corpus under a different file name than the predominant one for that hash. These could be deliberate renaming.

(iii) Files created in atypical weeks for the predominant week of their directory. These could be deliberately hidden files.

(iv) Files whose extension is inconsistent with the type assigned by "magic-number" header-tail analysis of the file contents such as by the Linux "file" utility. We used our extension taxonomy to classify extensions, and built a mapping from magic-number descriptions to the extension taxonomy classes (2,820 mappings were necessary) for comparison. These could be attempts at camouflage.

(v) Files whose hashes had inconsistent sizes. These were always associated with faulty information about deleted files for our corpus, but could also indicate deliberate attempts to conceal in other corpori.

(vi) Files with directories or extensions (when not in software directories) that are flagged explicitly as interesting. Directory examples are directories for encryption, disk wiping, and hacking tools. Extension examples are JPG and HTM when not in software directories.

5.3. Coverage and redundancy of the hash sets

We inspected the hash values in our corpus that matched the NSRL RDS and concluded they were highly reliable since, in random samples we saw no interesting files incorrectly included. This makes sense because the collection technique of the NSRL (buying the software, installing it, and inspecting its files) is a highly reliable at identifying forensically uninteresting files. So we eliminated files whose hash values matched NSRL entries as a first step. This reduced the number of distinct hashes from 35.80 million to 33.42 million.

At this point any further sets of uninteresting hash values can be applied, including the hashes of eliminated files from previous runs of our software. But in the experiments here, the methods described in sections 5.1 and 5.2 were applied to the full remaining data. To analyse the coverage and redundancy of the hash sets identified as uninteresting, we computed the sizes of their intersections (Table 4). The code labels are defined in section 5.1. It can be seen that there is some overlap but not a large amount, which argues for the use of all the methods together.

Table 4. Intersection sizes of uninteresting hash sets in our corpus, in millions.

	HA	PA	BD	TM	WK	SZ	CD	TD	EX
HA, frequent hashes	2.37	1.25	0.90	1.12	1.04	5.6	0.61	2.26	0.66
PA, frequent paths	1.25	5.86	3.17	1.50	1.14	1.73	2.15	3.77	0.72
BD, frequent immediate directories	0.90	3.17	4.69	1.03	0.87	1.74	1.77	4.15	0.75
TM, creation time clusters	1.12	1.50	1.03	7.29	0.63	1.65	2.00	4.56	2.46
WK, busy weeks	1.04	1.14	0.87	0.63	1.73	0.73	0.77	1.66	0.31
SZ, frequent sizes	5.6	1.73	1.74	1.65	0.73	7.08	1.34	4.04	0.71
CD, directory context	0.61	2.15	1.77	2.00	0.77	1.34	7.18	4.48	1.18
TD, uninteresting directories	2.26	3.77	4.15	4.56	1.66	4.04	4.48	15.31	2.83
EX, uninteresting extensions	0.66	0.72	0.75	2.46	0.31	0.71	1.18	2.83	3.43

5.4. Accuracy of our methods for finding uninteresting and interesting hash values.

To investigate the accuracy of our methods, we constructed a test set of 19,784 files by randomly sampling our corpus after elimination of all files in the NSRL. We laboriously inspected the metadata of the test set, and identified 18,223 of these as definitely uninteresting for all but special kinds of forensic investigation. We manually inspected the contents of as many as we could of the remaining 1,562 (some drive images were faulty), using associated software where possible such as image viewers for pictures, document viewers for documents, and a hexadecimal editor for the remainder. This process identified 270 more uninteresting files, for a final total of 1,292 interesting or possibly interesting files in the test set (6.53%) and 18,492 definitely uninteresting files. Some files were encoded and unclear in function, so the 1,292 could well have included more uninteresting files.

Our software described herein eliminated all but 5,658 of the test set as uninteresting using what we determined to be optimal parameter settings. After all filtering and adding back in of interesting hash values, 23 of the 14,126 eliminated files were actually or possibly interesting (false positives) according to our manual inspection, for a precision of 0.9982 for eliminated files. 5,396 of the 6,751 uneliminated files were actually uninteresting (false negatives), for a recall of 0.7069 for eliminated files. Precision here is considerably more important than recall since mistaken elimination of files removes them from an investigation whereas mistaken failure to eliminate files just increases the workload of the investigator a little, so the high observed precision is very encouraging. The 23 false positives included a frequently-seen government-retirement guide (whose presence could indicate a desire to retire), a cache file for an accounting system (but it could be a default), a geographical-information cache (also possibly a default), a document that was part of a large library downloaded at one time, and 11 unidentifiable deleted files that were missing directory information. The 5,396 false negatives included software installed in atpyical places rather than “Program Files” and “Applications”, as well as documents and Web pages associated with software.

Table 5 shows the results of tests on the precision and recall for each of our methods for identifying uninteresting files (Table 5), varying the key parameters to see their effects. The parameter values shown in the table were those having the best performance in an initial series of survey experiments. Measurements were made with the list of hash values delivered by the method. Here “mindrives” is the minimum number of drives on which the data occurs, “segcount” is the number of segments on the right end of the path that define the immediate directory, “weekmult” is how many times busier a week must be than the average week, "pathmult" is the minimum number of times of occurrence more than the typical path, "mindev" is the minimum number of standard deviations more than the mean number of occurrences that are required, "mindircount" is the minimum-size directory examined, and "fracmin" is the minimum fraction of the files in the directory that are already known to be uninteresting. Note that the "mindrives" parameter should be proportional to the number of drives in a corpus. Our corpus had 4018 drives, so mindrives should be multiplied by the number of drives in a new corpus divided by 4018. The other parameters do not need such adjustment.

It can be seen that false positives and false negatives trade off consistently as parameters vary, so the best set of parameters reflects the relative weight of precision to recall. It is impossible to eliminate false negatives completely for most corpori because there are several legitimate reasons for them:

- For HA, PA, and BD, some files that appear frequently may still be interesting, as for instance documents explaining how to make a bomb on drives obtained from criminals.
- For WK and SZ, interesting files may be loaded by coincidence during busy weeks or coincidentally have the same size as standard formats.
- For TM and CD, interesting files may occur in the midst of mostly uninteresting files by software design.
- For TD and EX, users may try to deceive investigators by camouflaging files. However, deception can be detected by other methods.

Table 6 shows the effects of varying the required number of clues to identify a file as uninteresting before we removed the explicitly interesting files from the hash sets. Requiring two clues appears to be best since recall decreases significantly for larger numbers with little effect on the near-perfect precision. Higher false positives could be acceptable in preliminary investigation of a corpus, but might be unacceptable in criminal investigation because of the possibility of excluding potential key evidence in a case.

Table 6 also shows the effect of choosing all low values for the parameters, all intermediate ("medium") values for the parameters, and all high values for the parameters. This tested the synergistic effect of having multiple clues. We further tested the effects of requiring 2 out of 8 of the clues, excluding each clue in turn, but it made little difference to the precision while decreasing the recall significantly. The last row gives statistics for the final result, after adding the uninteresting hashless files and default-directory files such as "..", and then subtracting the interesting files as discussed in the next section.

Table 5. Effects of parameter variation on performance of different methods of identifying uninteresting files.

Method	Parameter values	True positives	False positives	False negatives	Precision	Recall
HA, frequent hashes	mindrives = 2	11601	121	6770	.9897	.6315
HA	5	10816	25	7555	.9977	.5888
HA	12	9891	14	8480	.9986	.5384
PA, frequent paths	mindrives = 10	11922	67	6449	.9944	.6490
PA	20	11430	26	7941	.9975	.5677
PA	40	9970	21	8401	.9979	.5427
BD, frequent botdirs	mindrives = 25, segcount = 1	10885	76	7486	.9931	.5925
BD	25, 2	10338	23	8033	.9978	.5627
BD	25, 3	9827	17	8544	.9983	.5349
BD	50, 1	9109	66	9262	.9928	.4958
BD	50, 2	7824	22	10547	.9972	.4259
BD	50, 3	7320	17	11051	.9977	.3985
BD	100, 1	7336	51	11035	.9931	.3993
BD	100, 2	6021	17	12350	.9972	.3277
BD	100, 3	5574	13	12797	.9977	.3034
TM, time context	minimum count within minute = 25	7613	65	10758	.9915	.4144
TM	50	7408	55	10963	.9926	.4032
TM	100	7160	48	11211	.9933	.3897
WK, frequent weeks	weekmult = 2, pathmult = 50	7175	16	11196	.9978	.3906
WK	2, 100	7053	14	11318	.9980	.3839
WK	2, 200	6864	13	11507	.9981	.3736
WK	5, 50	6680	14	11691	.9979	.3636
WK	5, 100	6500	13	11871	.9980	.3538
WK	5, 200	6376	12	11995	.9981	.3470
WK	12, 50	5820	4	12551	.9993	.3168
WK	12, 100	5800	4	12571	.9993	.3157
WK	12, 200	5767	4	12604	.9993	.3139
SZ, frequent sizes	mincount = 5, mindev = 5	4818	138	13453	.9727	.2677
SZ	5, 10	4642	108	13729	.9773	.2527
SZ	5, 20	4101	87	14270	.9792	.2232
SZ	10, 5	4901	135	13470	.9732	.2669
SZ	10, 10	4633	106	13738	.9776	.2522
SZ	10, 20	4093	85	14278	.9797	.2228
SZ	20, 5	4862	132	13509	.9736	.2647
SZ	20, 10	4604	105	13767	.9777	.2506
SZ	20, 20	4079	84	14292	.9798	.2506
CD, directory context	mindircount=16, fracmin=0.5	7781	193	10590	.9758	.4236
CD	16, 0.8	7464	139	10907	.9817	.4063
CD	16, 0.95	7345	122	11026	.9837	.3998
CD,	40, 0.5	7229	185	11142	.9751	.3935
CD	40, 0.8	8137	534	9999	.9813	.3791
CD	40, 0.95	6858	114	11513	.9837	.3733
CD	100, 0.5	6525	165	11846	.9753	.3352
CD	100, 0.8	6263	114	12108	.9821	.3409
CD	100, 0.95	6149	97	12222	.9845	.3347
TD, unint. topdirs	none	14274	60	4133	.9958	.7755
EX, unint. extensions	none	3762	9	14609	.9976	.2048

Table 6. Additional effects of parameter variation on the accuracy of identifying uninteresting files.

Method	**True positives**	**False positives**	**False negatives**	**Precision**	**Recall**
Any 1 of the 9 clues, medium parameters	15825	356	2582	.9798	.8597
Any 2 of the 9 clues	14144	68	4263	.9952	.7684
Any 3 of the 9 clues	12251	18	6156	.9985	.6656
Any 4 of the 9 clues	10639	15	7768	.9986	.5780
Any 5 of the 9 clues	8553	8	9854	.9991	.4447
Any 6 of the 9 clues	6023	4	12384	.9993	.3272
Any 7 of the 9 clues	3706	1	14701	.9997	.2013
Any 8 of the 9 clues	1366	1	17041	.9993	.0742
All 9 of the 9 clues	113	0	18294	1.0000	.0061
All low values of the parameters, 2 out of 9 clues	14851	887	3285	.9436	.8188
All medium values of the parameters, 2 of the 9 clues	14144	68	4263	.9952	.7684
All high values of the parameters, 2 of 9 clues	11527	33	6609	.9972	.6356
Any 2 of 8 clues excluding HA	14144	68	4263	.9952	.7684
Any 2 of 8 clues excluding PA	13718	63	4689	.9954	.7453
Any 2 of 8 clues excluding BD	13874	64	4533	.9954	.7537
Any 2 of 8 clues excluding TM	14045	63	4362	.9955	.7630
Any 2 of 8 clues excluding WK	13653	55	4754	.9960	.7417
Any 2 of 8 clues excluding SZ	14112	62	4295	.9956	.7667
Any 2 of 8 clues excluding CD	13707	35	4700	.9975	.7447
Any 2 of 8 clues excluding TD	13657	52	4750	.9962	.7420
Any 2 of 8 clues excluding EX	12713	52	5694	.9959	.6907
Any 2 of 9 clues, medium parameters, minus potentially interesting files, plus uninteresting hashless and default-directory files	13011	23	5396	.9982	.7069

5.5. Accuracy of our methods of finding interesting hash values

Table 7 shows results on the accuracy of the six methods for detecting interesting hash values. Their performance was poor with the exception of the last method. For these experiments, true positives were defined as files identified both as interesting in our test set and by our methods, false positives were files identified as interesting only by our methods, and false negatives were files identified as interesting only in our test set. Here "mincount" is the minimum size of the directory considered, and "minfrac" is the minimum fraction of the most common hash or path. Results are likely poor due to the minimal concealment in our data. We used only the hash values from the last method in the final results reported here. However, files obtained during criminal and intelligence investigations are more likely to show concealment, and such files could be critical in an investigation. It is thus recommended that all these interesting-file methods should be run whenever the uninteresting-file methods are too.

Table 7. Effects on parameter variation on performance of different methods for identifying interesting files.

Method	Parameter values	True positives	False positives	False negatives	Precision	Recall
Hashes occurring only once for a given path	mincount=10, minfrac=0.5	4	565	1203	.0070	.0033
same	10, 0.8	4	475	1203	.0094	.0033
same	10, 0.95	1	439	1206	.0023	.0008
same	30, 0.5	3	503	1204	.0059	.0025
same	30, 0.8	3	451	1204	.0066	.0025
same	30, 0.95	1	438	1206	.0023	.0008
same	100, 0.5	2	422	1205	.0047	.0017
same	100, 0.8	2	388	1205	.0051	.0017
same	100, 0.95	0	384	1207	.0000	.0000
Path names occurring only once for a given hash	mincount=10, minfrac=0.5	5	293	1202	.0168	.0041
same	10, 0.8	3	247	1204	.0120	.0025
same	10, 0.95	2	171	1205	.0116	.0017
same	30, 0.5	4	226	1203	.0174	.0033
same	30, 0.8	2	201	1205	.0099	.0017
same	30, 0.95	1	165	1206	.0060	.0008
same	100, 0.5	1	144	1206	.0069	.0008
same	100, 0.8	1	134	1206	.0074	.0008
same	100, 0.95	1	120	1206	.0083	.0008
Files created in atypical weeks for their directories	mincount=10, minfrac=0.5	0	216	1207	.0000	.0000
same	10, 0.8	0	212	1207	.0000	.0000
same	10, 0.95	0	27	1207	.0000	.0000
same	30, 0.5	0	23	1207	.0000	.0000
same	30, 0.8	0	20	1207	.0000	.0000
same	30, 0.95	0	0	1205	.0000	.0000
same	100, 0.5	0	211	1207	.0000	.0000
same	100, 0.8	0	203	1207	.0000	.0000
same	100, 0.95	0	20	1207	.0000	.0000
Inconsistency between extension and magic-number analysis	None	5	2602	1202	.0019	.0041
Inconsistency in file size for the same hash value	None	0	67	1207	.0000	.0000
Explicitly identified interesting extension or directory	None	552	1478	655	.2719	.4573

5.6. Additional analysis

Overall statistics on file and hash eliminations are given in Table 8. The full run our corpus files with the optimal parameters took 2748 minutes (just short of two days) as a single process on a single Redhat Enterprise Linux 7 system. This is a one-time cost that does not affect the speed of online usage of the hash values. The test machine had 515 gigabytes of main memory shared with other researchers, and we found we had to limit main memory usage to 200 gigabytes maximum by splitting files as necessary. Input was DFXML-format metadata (as defined at www.nsrl.nist.org). Much of the processing could be partitioned to separate processors to decrease completion time if desired.

Table 8. Overall statistics on file and hash elimination and their data reduction percentages.

Set	Number of files (millions)	Number of distinct hashes (millions)
Full corpus	262.7	35.80
Full corpus minus NSRL	200.1 (76.2%)	33.42 (93.4%)
Same minus those files identified by 2 of 9 methods as uninteresting	56.2 (21.4%)	19.27 (53.8%)
Same minus uninteresting hashless or default-hash files	46.0 (17.5%)	19.26 (53.8%)
Same plus potentially interesting files	59.5 (22.6%)	20.32 (56.8%)

We compared the types of files before and after filtering out uninteresting files using our aforementioned taxonomy on file extensions. Operating-system files went from 6.1% to 1.0%, installation and update files went from 1.5% to 0.4%, executables went from 10.9% to 3.3%, program source (including scripts) from 9.3% to 7.0%, XML from 3.0% to 2.2%, integer extensions from 0.8% to 0.4%, indices from 4.4% to 0.1%, and games from 1.5% to 0.1%. Camera images increased from 3.6% to 10.3%, Web files and links from 9.6% to 12.2%, documents from 2.9% to 6.8%, audio from 1.4% to 2.8%, video from 0.3% to 1.1%, email and messaging from 0.1% to 0.4%, temporaries from 0.9% to 1.6%, logs from 0.3% to 0.7%, database-related files from 0.4% to 0.7%, geography-related files from 0.1% to 0.3%, and extensionless files from 9.7% to 16.8%. All other file-type percentages did not change significantly, including notably graphics, configuration files, copies, security-related files, and files with multiuse extensions. Note that a hash value appearing with multiple paths was eliminated if there were two clues for any of the paths, not necessarily for all of the paths.

A criticism made of some hash-value collections is that their values will rarely occur again. So an important test for our proposed new "uninteresting" hash values is to compare those acquired from different drives. For this we split the corpus into two pieces C1 and C2 based on the drives, roughly drives processed before 2012 and those processed in 2012 and 2013. We did not include any drives from our school in this experiment because they are centrally managed and share much software. We extracted uninteresting hash values using our methods for both separately, and then compared them. Table 9 shows the results.

Table 9. Statistical comparison of hashes derived from partition of our corpus into two halves, where HA=hashes, PA=paths, TM=time, SZ=size, BD=bottom-level directory, CD=directory context, TD=top-level directory, EX=extension.

Method of obtaining new hash values	Fraction of values found for C1 also found for C2	Fraction of values found for C2 also found for C1	Fraction of all hash values for C1 identified by method using C1	Fraction of all hash values for C2 identified by method using C2	Fraction of all hash values for C2 identified by method using C1	Fraction of all hash values for C1 identified by method using C2
HA	.717	.597	.438	.355	.344	.389
PA	.661	.458	.399	.291	.299	.312
TM	.529	.365	.668	.493	.430	.477
WK	.457	.234	.445	.303	.358	.323
SZ	.339	.246	.196	.187	.157	.145
BD	.583	.484	.474	.384	.350	.406
CD	.640	.486	.558	.447	.411	.467
TD	.553	.436	.627	.485	.419	.474
EX	.603	.439	.497	.373	.338	.397

This table provides two kinds of indicators of the generality of a hash-obtaining method. One is the average of the second and third columns, which indicates the overlap between the hash sets. On this SZ is the weakest method and WK is second weakest, which makes sense because these methods seek clustered downloads and some clusters are forensically interesting. Another indicator is the ratios of column 7 to column 4 and column 6 to column 5, which indicate the degree to which the hash values found generalize from a training set to a test set. The average for the above data was 0.789 for the average of column 7 to column 4, and 0.938 for the average of column 6 to column 5, which indicates a good degree of generality. No methods were unusually poor on the second indicator.

We also compared the uninteresting hashes found for the 2013 version of the corpus with those of two commercial hashsets, the April 2013 download of the Bit9 Cyber Forensics Service (www.bit9.com) and the June 2012 download of the hash list of Hashsets.com. We computed statistics on the file types represented in the two sets and confirmed broad coverage of a variety of file types in the sets, not just coverage of executables. Nonetheless, hashsets.com matched hashes on 5,906 of the remaining hashes, an unimpressive 0.06%. Bit9 recognized 607,693 hash values not in NSRL out of the 10,342,915 that it recognized, of which 93,436 (0.95% of the remaining hashes) were not found by our methods. So Bit9 is not much more help in eliminating uninteresting files beyond our own methods which eliminated millions of files, something important to know since it is expensive to purchase.

Of the 13.28 million hash values of the NSRL RDS in our corpus, only 1.45 million (10.9%) were also found by our methods without help from NSRL. So there is still a justification for building the NSRL even though it requires more manual labour per hash than our methods.

5.7. Proposed file-elimination protocol

We suggest then the following protocol for eliminating uninteresting files from a corpus in a standard investigation:

(i) Run methods PA, BD, TM, WK, and SZ to generate hash sets of candidate uninteresting files on the full corpus.
(ii) Eliminate all files in the corpus whose hash values are in NSRL, our list at digitalcorpora.org, and any other confirmed "uninteresting" lists available.
(iii) Run the methods HA, CD, TD, and EX on the remaining files to generate additional hash values of uninteresting files. These methods do not benefit from seeing the entire corpus.
(iv) Find hash values that occur in at least two uninteresting hash sets, and remove files from the corpus with those hash values.
(v) Eliminate default-directory files and those without hash values that match on two of the three criteria BD, TD, and EX.
(vi) To the remaining files, add files matching the "interesting-directory" and "interesting-extension" criteria.
(vii) Save the final list of eliminated hash codes for bootstrapping with future drives when doing step (ii).

6. Conclusions

Although uninterestingness of a file is a subjective concept, most forensic investigators have a precise definition for each investigation that is usually based whether a file contains user-created or user-discriminating information. It appears that relatively simple methods can be used to automate this intuition, and can eliminate considerable numbers of uninteresting files beyond using the NSRL hash library alone. On our corpus, NSRL eliminated 23.8% of the hashes while our methods eliminated an additional 53.6%, while keeping false positives (incorrectly eliminated files) to 0.18%. Our methods do need a large corpus of file examples; however, more and more file data is becoming available to researchers. It also appears that commercial hash sets are of limited additional value to most forensic investigations if the methods proposed here are used.

Our methods can eliminate files unique to a drive, but they also will provide hashes that should be useful for other corpora. Investigators can choose which methods to use based on their investigative targets, can set thresholds based on their tolerance for error, and can choose to eliminate further files based on time and locale as in [19].

We have published a list of our uninteresting hashes for free download on digitalcorpora.org. Further methods for identifying additional uninteresting files are definitely possible given the low 6.53% "potentially interesting" rate in our test set. Future directions are to extend the ideas to hashes on portions of files [15] and to many-to-one mappings recognizing similar but not identical files such as pictures.

Acknowledgements
Riqui Schwamm, Jeromy Santos, Nicholas Moore, Jacob Beel, and Christopher Herridge assisted with the experiments, and Simson Garfinkel provided the Real Drive Corpus. The views expressed are those of the author and do not represent those of the U.S. Government. This paper is an extension and revision of a paper in the 5th International Conference on Digital Forensics and Computer Crime, Moscow, Russia, September 2013.

References

[1] Agrawal, N., Bolosky, W., Douceur, J., and Lorch, J. (2007) A five-year study of file-system metadata. *ACM Transactions on Storage*, **3** (3): 9:1-9:32.
[2] Bell, J., and Whaley, B., *Cheating*. New York: Transaction Publishing, 1991.
[3] Chang, J., Venkatasubramanian, K., West, A., and Lee, I. (2013) Analyzing and defending against Web-based malware. *ACM Computing Surveys*, **45**(4), August, 49:1-35.
[4] Chawathe, S. (2012) Fast fingerprinting for file-system forensics. In *Proc. IEEE Conference on Technologies for Homeland Security*, Waltham, MA, November, 585-590.
[5] Garfinkel, S., Farrell, P., Roussev, V., and Dinolt, G. (2009) Bringing science to digital forensics with standardized forensic corpora. *Digital Investigation*, **6,** S2-S11.
[6] Gugelmann, D., Schatzmann, D., and Lenders, V. (2013) Horizon Extender: Long-term preservation of data leakage evidence in Web traffic. Proc. ASIA CCS, Hangzhou, China, 499-504.
[7] Ke, H.-J., Wang, S.-J., Liu, J., and Goyal, D. (2011) Hash-algorithms output for digital evidence in computer forensics. In *Proc. Intl. Conf. on Broadband and Wireless Computing, Communication and Applications,* Barcelona, Spain, October, 399-404.
[8] Kornblum, J. (2008) Auditing hash sets: lessons learned from Jurassic Park. *Journal of Digital Forensic Practice*, **2** (3), 108-112.
[9] Malin, C., Casey, A., and Aquilina, J. (2008) *Malware Forensics: Field Guide for Windows Systems* (Syngress, Waltham, MA, US).
[10] Mead, S. (2006) Unique file identification in the National Software Reference Library. *Digital Investigation*, **3** (3), 138-150.
[11] Mohaisen, A., and Alrawi, O. (2014, July) An evaluation of antivirus scans and labels. Proc. 11th Intl. Conf. on Detection of Intrusions and Malware and vulnerability Assessment, Egham UK, pp. 112-131.
[12] Panse, F., Van Keulen, M., and Ritter, N. (2013) Indeterministic handling of uncertain decision in deduplication. *ACM Journal of Data and Information Quality*, **4** (2), 9.
[13] Pearson, S. (2010) *Digital Triage Forensics: Processing the Digital Crime Scene* (Syngress, Waltham, MA, US).
[14] Pennington, A., Linwood, J., Bucy, J., Strunk, J., and Ganger, G. (2010) Storage-based intrusion detection. *ACM Transactions on Information and System Security*, **13** (4) 30.

[15] ROUSSEV, V. (2012) Managing terabyte-scale investigations with similarity digests. In *Proc. of Advances in Digital Forensics VIII, IFIP Advances in Information and Communication Technology Volume 383*, Pretoria SA, 19-34.

[16] ROWE, N. (2012) Testing the National Software Reference Library. *Digital Investigation*, **9S**, S131-S138.

[17] ROWE, N. (2015) Finding contextual clues to malware using a large corpus. ISCC-SFCS Third International Workshop on Security and Forensics in Communications Systems, Larnaca, Cyprus, July 2015.

[18] ROWE, N., AND GARFINKEL, S. (2012) Finding suspicious activity on computer systems. In *Proc. 11th European Conf. on Information Warfare and Security*, Laval, France.

[19] RUBACK, M., HOELZ, B., AND RALHA, C. (2012) A new approach to creating forensic hashsets. In *Proc. of Advances in Digital Forensics VIII, IFIP Advances in Information and Communication Technology Volume 383*, Pretoria SA, 83-97.

[20] TOMAZIC, S., PAVLOVIC, V., MILOVANOVIC, J., SODNIK, J., KOS, A., STANCIN, S., AND MILUTINOVIC, V. (2011) Fast file existence checking in archiving systems. *ACM Transactions on Storage*, **7** (1), 2.