Identifying Forensically Uninteresting Files Using a Large Corpus

Neil C. Rowe(✉)

U.S. Naval Postgraduate School, CS/Rp, GE-328, 1411 Cunningham Road, Monterey, CA 93943, USA
ncrowe@nps.edu

Abstract. For digital forensics, eliminating the uninteresting is often more critical than finding the interesting. We define "uninteresting" as containing no useful information about users of a drive, a definition which applies to most criminal investigations. Matching file hash values to those in published hash sets is the standard method, but these sets have limited coverage. This work compared nine automated methods of finding additional uninteresting files: (1) frequent hash values, (2) frequent paths, (3) frequent filename-directory pairs, (4) unusually busy times for a drive, (5) unusually busy weeks for a corpus, (6) unusually frequent file sizes, (7) membership in directories containing mostly-known files, (8) known uninteresting directories, and (9) uninteresting extensions. Tests were run on an international corpus of 83.8 million files, and after removing the 25.1 % of files with hash values in the National Software Reference Library, an additional 54.7 % were eliminated that matched two of our nine criteria, few of whose hash values were in two commercial hash sets. False negatives were estimated at 0.1 % and false positives at 19.0 %. We confirmed the generality of our methods by showing a good correlation between results obtained separately on two halves of our corpus. This work provides two kinds of results: 8.4 million hash values of uninteresting files in our own corpus, and programs for finding uninteresting files on new corpora.

Keywords: Digital forensics · Metadata · Files · Corpus · Data reduction · Hashes · Triage · Whitelists · Classification

1 Introduction

As digital forensics has grown, larger and larger corpora of drive data are available. To speed subsequent processing, it is essential in the triage process for a drive to first eliminate from consideration those files that are clearly unrelated to an investigation [8]. This can be done either by directly eliminating files to create a smaller corpus or by removing their indexing. We define as "uninteresting" those files whose contents do not provide forensically useful information about users of a drive. These are operating-system and applications-software files that do not contain user-created information, and also include common Internet-document downloads that do not provide user-discriminating information. (Metadata on uninteresting files may still be interesting, however, as in indicating time usage patterns.) This definition applies to most criminal investigations and data mining tasks but not to malware investigations. We can confirm that files are uninteresting

P. Gladyshev et al. (Eds.): ICDF2C 2013, LNICST 132, pp. 86–101, 2014.
DOI: 10.1007/978-3-319-14289-0_7

by opening them and inspecting them for user-created and user-discriminating data. Additional files may also be uninteresting depending on the type of investigation, such as presentation files 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, and some software directories do contain interesting user files, so finding the uninteresting is not always straightforward.

Most decisions about interestingness can be made from file-directory metadata without examining the 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]. One additional type of data is also very useful, a hash value computed on the contents of the file, which enables recognition of identical content under different file names. Forensic tools like SleuthKit routinely extract directory metadata from drive images.

We can eliminate files whose hash values match those in published sets [5]. This has the side benefit of detecting modified files since their hash values are different [9]. However, published hash values miss many kinds of software files [11], especially files created dynamically. This paper will discuss methods for improving on this performance, in particular by correlating files on drives and across drives on a corpus. This provides both a new set of hash values and new methods for finding them.

2 Previous Work

A standard approach is to eliminate files whose hash values match those in the National Software Reference Library (NSRL) from the U.S. National Institute of Standards and Technology (NIST). The quality of the data provided in the NSRL is high [6]. However, our tests [11] found that it did not provide much coverage. Around one file of four in our international corpus appeared in the NSRL, and there were surprising gaps in 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 uninteresting files like software 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.

Chawathe [2] investigates the problem of recognizing uninteresting files (which they call "whitelisting") and suggests that pieces of files need to be hashed separately, a technique that considerably increases the workload. Tomazic et al. [14] details efficient methods for indexing and matching hash values found on files. Many of the issues are similar to the important problem of file deduplication for which file hashes are useful [7].

Ruback et al. [13] is the closest work to ours. They 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 [11]. 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 corpus-specific heuristic ones.

3 Experimental Setup

The experiments reported here were done with the Real Drive Corpus [3], which at the time had 3471 drives purchased as used equipment in 28 non-U.S. countries, supplemented with additional drives from our previous research. There were 83,787,499 files on these drives with 21,021,187 distinct hash values. We extracted directory metadata with SleuthKit and the Fiwalk tool. As these drives were obtained from ordinary users, we saw very little concealment or camouflage on them. Thus the hash values we derived from them should accurately represent file contents, an issue important in some forensic applications [4].

We also obtained the December 2012 version of the National Software Reference Library Reference Data Set (NSRL-RDS, www.nsrl.nist.gov), the June 2012 version of the database of hashsets.com, and an April 2013 download of the database of the Bit9 Cyber Forensics Service (www.bit9.com). Because hash values in Bit9 are encoded, we were only able to test hashes in our corpus that were also in Bit9. Basic data is given in Table 1.

Table 1. Hash set sources used in our experiments.

	NSRL RDS (NSRL), December 2012	hashsets.com (HSDC), June 2012	Subset of RDC in Bit9 cyber forensics service (BIT9), April 2013	Real drive corpus (RDC), March 2013
Number of entries	95,909,483	17,774,612	321,847	83,787,499
Number of distinct hash values	29,311,204	6,464,209	321,847	21,021,187
Fraction distinct	0.306	0.364	1.0	0.251

This work used SHA-1 hash values. They are widely available and serve as the primary key for the NSRL. MD5 hash values are also widely available, but 128 bits as opposed to 160 does not provide a sufficiently low probability, in our view, of hash collisions.

4 Methods for Finding Uninteresting Files

As explained earlier, "uninteresting" files will be defined as those that do not contain user-created or user-discriminating data. Nine methods to identify them and then their hash values were investigated as summarized in Table 2. Thresholds used by these methods were set by the experiments reported in Sect. 6.

Table 2. Summary of our methods for finding uninteresting files.

Method	Scope	Primary data focus	Secondary data focus	Considers deleted files?
HA	Corpus-wide	Hash values		Yes
PA	Corpus-wide	Full paths		Yes
BD	Corpus-wide	File name and containing directory		No
TM	Single-drive	Creation times within the minute		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		No
EX	Single-drive	File extension		No

The methods were:

- HA, frequent hashes: Files on different drives with the same hash value on their contents. Hash values that occur on 2–3 drives in a corpus suggest possibly interesting sharing of information between investigative targets. But hash values occurring often are likely to be distributions from a central source and are unlikely to be forensically interesting. An example was C2A3FCD0224B14AD6-B6A562992C3802CC711E6A2 in our corpus but not in NSRL, 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. These represent information displayed with tabs in a Web browser. We set a threshold of occurrences on at least five drives for such "frequent hash values" based on our experiments. The threshold must be on number of drives, not occurrences, since copies of files on the same drive are common.
- PA, frequent paths: Files with the same full path (file name plus directories) on different drives. Frequently occurring paths are unlikely to be forensically

interesting since they are likely due to mass distribution. 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. We set a threshold of at least 20 occurrences, including deleted files, for hash values based on our experiments.

- BD, frequent bottom-level directory-filename pairs: Files whose pair of the file name and the immediate directory above it occurred especially frequently. This will catch versions of the same file in different directories under different versions of an operating system or software. 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 like the defaults "." and "..". We set a threshold of at least five undeleted occurrences based on our experiments.
- TM, clustered creation times: Files with the same creation time within a short period as that of many other files on the same drive. Such time 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 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 indicate installation. We set a threshold of at least 50 files created within the same minute based on our experiments.
- WK, busy weeks: Files created unusually frequently in particular weeks across drives, which suggest software updates. A period of a week is appropriate since it takes several days for most users to connect to a site and get an update. Figure 1 shows a typical distribution of times by week in our corpus, showing some sharp peaks. We count file creations per week and find "busy" weeks having at least five times the average amount of creations, of which there were 116 in our corpus. Then we find "busy" directories (full path minus the file name) in the busy weeks, those whose frequency of creation was at least 100 times greater than their average creation time per week. Again, thresholds were set by experiments; the 100 was necessary to discriminate against user copying of file directories. We then collect the hash values for those busy directories on those busy days as proposed uninteresting file content.
- SZ, frequent sizes: Files with the same size and extension. This enables recognizing fixed-size formats with different contents, as certain kinds of log records. However, to reduce false matches we 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 published hash sets. Based on experiments, we set a threshold of occurrences of at least five drives where the occurrence rate of the file extension in everything with that size was at least ten standard deviations above the average rate in the corpus.

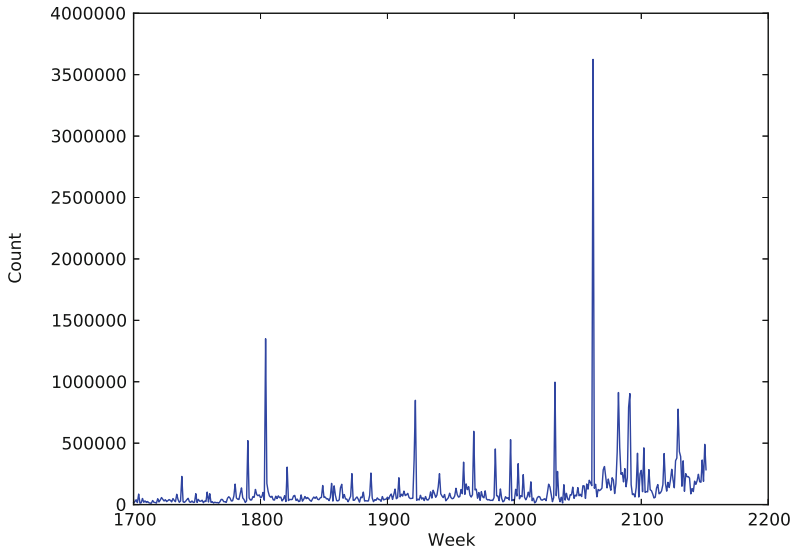

Fig. 1. Histogram of a sample range of creation times in our corpus.

- Small files (included in SZ): Files less than a minimum size are 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 5 bytes from experiments.
- 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 from our corpus is directory Program Files/Yahoo!/Messenger/skins/Purple/images and previously unclassified files ".", "..", bg_hover.png, _GLH0307.TMP, and bg_selected.png. We set a threshold of 50 % known hash values in directories based on experiments. This method can be used to bootstrap better performance on each run on 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 5749 top-level and 337 mid-level directories. Example top-level directories were APPS/Symantec AntiVirus, Documents and Settings/Admin/Templates, FOUND.029, Program Files (x86)/AutoCAD 2008, WINDOWS, and system/install; example mid-level directories were /Help/ and /Default User/. It is important not to exclude all applications directories (Program Files, Applications, etc.) because some software keeps user files there.
- DR, directory records: Default directory records such as "WINNT/Cursors/." as a path. These were excluded only in the final phase of processing (Sect. 7) since they can help establish directory and time trends.
- EX, files with known uninteresting extensions: Some extensions are exclusively associated with operating systems and software, such as exe, mui, log, dat, bin, and config. We used a classification of 5802 file extensions that we are developing [11] that maps extensions to 45 categories. The categories we labeled 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. Files with no extensions and extensions with more than one category were excluded.

36.7 % of the files in our corpus were identified by SleuthKit as deleted (unallocated). We excluded these as sources of new hash values with two exceptions because [11] found that once files are marked for deletion, their metadata can become corrupted. Directories were often missing for deleted files, and we even saw inconsistencies in the sizes of reported by SleuthKit for files with the same hash value, which should be virtually impossible. However, the same hash value or the same path appearing repeatedly is unlikely to be a coincidence even if they were all deleted, so we ignored deletion status in collecting frequent hash values and frequent paths.

These methods can be deceived into marking interesting files as uninteresting when a user engages in deliberate camouflage. For instance, a user could put copies of an interesting file on multiple drives to qualify for the HA, PA, BD, WK, or SZ sets; copy sensitive files to innocuous directories to qualify for TD, CD, or BD sets; and change the file extension to qualify for EX. But such actions can be detected as anomalous usage and found by statistical tests as described in [12].

5 Coverage and Redundancy of the Hash Sets

We assessed the coverage, redundancy, and accuracy of the methods. To analyze redundancy we computed the sizes of the intersections of the hash sets. Table 3 provides a summary of intersection counts for the hash sets for the 78,240,703 nonempty hash values of the 83,787,499 files in our corpus. Here and in Table 4 the row and column codes are:

- NS (only in Table 3): hashes in the NSRL RDS
- HS (only in Table 4): hashes in hashsets.com
- B9 (only in Table 4): hashes in our corpus that were also in the Bit9 database
- HA (only in Table 3): frequent hashes in our corpus
- PA: hashes of frequent paths
- BD: hashes of frequent immediate-directory plus filename pairs
- TM: hashes of files with time-clustered creation times
- WK: hashes of files created in unusually busy weeks
- SZ: hashes of files with unusually frequent sizes, plus files under 6 bytes
- CD: hashes of files in directories with mostly known-uninteresting files
- TD: hashes of files with top-level or mid-level uninteresting directories
- EX: hashes of files with non-user extensions.

We inspected a sample of hash values returned for each method and concluded that the NS hashes (NSRL RDS) and HA hashes (occurring on at least five drives in our corpus) were highly reliable since, in samples of size 200, we saw no interesting files

incorrectly included. This makes sense for the NSRL hashes because the collection technique of NSRL (buying the software and inspecting its files) is a highly reliable at identifying uninteresting files. Similarly, the diversity of our corpus means that any file that occurs on at least five drives is highly likely to be a distributed sharable resource and thus uninteresting. So we recalculated our counts excluding NS and HA as shown in Table 4. Excluding NSRL hashes reduced the number of distinct hashes from the corpus from 21,021,187 to 19,735,599 (a reduction of 6.1 %) and the number of files from 78,240,703 to 62,774,546 (a reduction of 19.8 %). Excluding hash values occurring on at least five different drives in the corpus reduced the number of distinct hashes further to 19,343,552 (a further reduction of 1.6 %) and the number of files to 43,909,093 (a significant further reduction of 30.1 %). The remaining hash sets had significant amounts of overlap, supporting their validity for file exclusion.

Table 3. Number of files in the RDC corpus in March 2013 (in millions) that have hashes in each of two hash sets.

	NS	HA	PA	BD	TM	WK	SZ	CD	TD	EX
NS	21.0	18.9	18.4	17.8	20.2	15.4	3.8	20.7	20.0	17.2
HA	18.9	33.1	30.0	29.0	31.5	25.1	5.6	31.9	31.4	26.8
PA	18.4	30.0	35.4	33.1	33.5	26.5	6.6	33.8	34.2	26.8
BD	17.8	29.0	33.1	35.8	33.5	25.3	7.0	33.2	34.1	26.4
TM	20.2	31.5	33.5	33.5	47.3	27.5	9.2	38.3	41.0	32.4
WK	15.4	25.1	26.5	25.3	27.5	27.9	5.1	27.0	27.2	21.8
SZ	3.8	5.6	6.6	7.0	9.2	5.1	10.5	7.7	8.9	6.2
CD	20.7	31.9	33.8	33.2	38.3	27.0	7.7	40.9	38.1	30.2
TD	20.0	31.4	34.2	34.1	41.0	27.2	8.9	38.1	45.6	32.2
EX	17.2	26.8	26.8	26.4	32.4	21.8	6.2	30.2	32.2	35.7

Table 4. Number of files in the RDC corpus in March 2013 (in millions) that have hashes in each of two hash sets, after excluding NS (the National Software Reference Library) and HA (hashes occurring on at least five drives).

	HS	B9	PA	BD	TM	WK	SZ	CD	TD	EX
HS	0.9	0.0	0.3	0.0	0.7	0.2	0.1	0.9	0.9	0.1
B9	0.0	0.5	0.0	0.0	0.3	0.0	0.0	0.2	0.3	0.0
PA	0.3	0.0	5.0	4.3	4.0	1.5	1.6	4.7	5.0	1.5
BD	0.0	0.0	4.3	6.2	4.8	1.3	2.2	5.8	6.1	1.7
TM	0.7	0.3	4.0	4.8	14.0	2.3	3.5	12.1	12.1	5.1
WK	0.2	0.0	1.5	1.3	2.3	2.5	0.8	2.3	2.4	0.5
SZ	0.1	0.0	1.6	2.2	3.5	0.8	4.5	3.9	4.0	1.2
CD	0.9	0.2	4.7	5.8	12.1	2.3	3.9	15.5	14.7	6.2
TD	0.9	0.3	5.0	6.1	12.1	2.4	4.0	14.7	16.6	6.2
EX	0.1	0.0	1.5	1.7	5.1	0.5	1.2	6.2	6.2	7.1

To compare the coverage of the published hash sets and our corpus, we classified the file extensions of their associated files using our aforementioned taxonomy (Table 5). We did this for the four primary sources we used; the last column will be explained in Sect. 7. We used a sample of 30 million records of the Bit9 Cyber Forensics Service, not just those matching our corpus. These counts are on files and not hashes, so the same hash was weighted by how often it occurred. "None" means files with no extension and "miscellaneous" includes ten lesser categories as well as extensions occurring less than 200 times in our corpus. The statistics confirm that all these hash sets had broad coverage of a variety of file types, not just executables. That suggests that their coverage gaps are due to difficulties in covering all software rather than in covering all file types.

Table 5. Distribution of files by extension type for five hash sets.

Type of extension	NSRL RDS	Hashsets.com	Bit9 sample	Real data corpus	Final RDC filtering
None	10.56 %	13.78 %	9.62 %	21.85 %	10.21 %
Oper. system	3.74 %	4.55 %	1.53 %	6.89 %	0.00 %
Graphics	16.23 %	13.64 %	13.86 %	13.03 %	13.14 %
Camera images	3.14 %	0.80 %	2.36 %	6.13 %	22.11 %
Temporaries	0.08 %	0.02 %	0.06 %	2.20 %	4.25 %
Web	8.25 %	8.83 %	17.56 %	4.45 %	6.82 %
Misc. documents	1.71 %	1.74 %	1.46 %	2.00 %	4.69 %
MS Word	0.17 %	0.03 %	0.16 %	0.71 %	2.98 %
Presentations	0.26 %	0.02 %	0.07 %	0.13 %	0.51 %
Database	0.29 %	0.18 %	0.21 %	0.73 %	1.04 %
Other MS Office	0.09 %	0.11 %	0.05 %	0.21 %	0.15 %
Spreadsheets	0.43 %	0.38 %	0.14 %	0.46 %	1.60 %
Email	0.11 %	0.03 %	0.09 %	0.12 %	0.33 %
Links	0.01 %	0.04 %	0.05 %	1.08 %	2.00 %
Compressed	1.33 %	7.05 %	2.22 %	0.65 %	1.23 %
Help	0.94 %	0.28 %	0.51 %	1.01 %	0.00 %
Audio	1.47 %	0.38 %	0.71 %	3.21 %	4.42 %
Video	0.20 %	0.04 %	0.16 %	0.35 %	0.79 %
Program source	7.16 %	11.44 %	8.98 %	2.20 %	4.11 %
Executables	18.70 %	14.51 %	18.59 %	12.90 %	0.00 %
Disk images	0.78 %	1.87 %	1.40 %	1.15 %	0.52 %
XML	0.94 %	2.17 %	1.24 %	1.00 %	0.61 %
Logs	0.04 %	0.05 %	0.06 %	0.76 %	2.29 %
Geographic	0.25 %	0.05 %	0.09 %	0.18 %	0.20 %

(*Continued*)

Table 5. (*Continued*)

Type of extension	NSRL RDS	Hashsets.com	Bit9 sample	Real data corpus	Final RDC filtering
Copies	0.09 %	0.04 %	0.28 %	0.40 %	0.33 %
Integers	1.03 %	1.80 %	0.83 %	2.17 %	4.59 %
Configuration	5.32 %	5.10 %	3.66 %	5.14 %	2.35 %
Update	0.06 %	0.01 %	0.07 %	0.16 %	0.00 %
Security	0.22 %	0.20 %	0.13 %	0.33 %	0.26 %
Malicious	0.01 %	0.01 %	0.00 %	0.02 %	0.00 %
Games	2.44 %	1.70 %	1.64 %	3.24 %	0.00 %
Sci. and eng.	0.85 %	0.03 %	0.51 %	0.21 %	0.35 %
Virtual machine	0.12 %	0.04 %	0.09 %	0.08 %	0.08 %
Multipurpose	2.79 %	3.76 %	3.48 %	2.46 %	5.64 %
Miscellaneous	9.89 %	5.32 %	7.55 %	2.41 %	2.21 %

6 Accuracy of the New Methods of Finding Uninteresting Hash Values

These methods can err in identifying interesting hash values as uninteresting for several reasons:

- PA, frequent paths: Some configuration files can give clues about a user although they appear frequently in the corpus.
- BD, frequent bottom-level directories: A directory-filename pair may occur frequently if its words are common, such as pics/portrait.jpg, yet still be interesting.
- TM, clustered creation times: During an automated software download, the user may be working on their own files so that creation times are interspersed.
- WK, busy weeks: Users may also be working during busy weeks for software updates.
- SZ, frequent sizes: A user file may accidentally be the same size as a standard size used by an application. Also, the standard size may reflect a format imposed on a user, as with camera formats.
- CD, contextually uninteresting files: Users may put files in a directory that is mostly created by software.
- TD, files with known uninteresting top-level or mid-level directories: Users may copy executables to their own directories for backup.
- EX, files with known uninteresting extensions: Users may assign their own extensions to files.

To investigate how often these conditions occurred, we took random samples of 200 files produced by each method, 1600 in all. We then inspected the full paths, and did Web research as necessary, to determine which files were user-related or otherwise potentially interesting in an investigation (Table 6). A few files were unclear in function so we counted them at half weight.

Table 6. Testing the new hash sets for uninteresting files.

New hash set	Number of hash values not in any other set	Fraction of files in the set that were actually interesting
PA, frequent paths	4,050	.010
BD, frequent bottom-level directories	57,739	.005
TM, clustered creation times	1,739,083	.020
WK, busy weeks	1,316,465	.000
SZ, frequent sizes	159,229	.070
CD, contextually uninteresting	23,527	.025
TD, known uninteresting top-level or mid-level directories	434,840	.005
EX, known uninteresting extensions	457,558	.015

Example files judged as interesting were system/apps/NaturalRecorder/Message14678.amr (user data), WINDOWS/Recent/p-0.dwg (72).lnk (user link), BILDER/Freunde Mohaa/Engelsfeuer/PIC_0512.JPG (user picture), and Documents and Settings/hariom/Local Settings/Temporary Internet Files/Content.IE5/STIVWXYZ/actors[1].jpg (user download). Examples judged as uninteresting were WINDOWS/system32/localsec.dll (operating system), WINDOWS/Fonts/CONSOLAZ.TTF (font file), System Volume Information/_restore{8907E5C6-24EC-4C3A-BC96-8740D90875EC}/RP22/A0320774.exe (system backup), Program Files/Condition Zero/valve/gfx/env/blackbk.tga (game graphics), Shortcut to GlobeSpan Dial-Up PPP Connection.lnk (frequent link), program files/Timbuktu Pro/Help/art/icon_exchange.gif (help graphics), and Temp/crt4349.tmp (display temporary).

The second column of Table 6 indicates the uniqueness of the method and the third column indicates the accuracy. The calculations for the third column were used to set thresholds for the methods, aiming at better than a 2 % error rate; unfortunately, the SZ (size) rate cannot be easily adjusted. Such error rates could be acceptable in preliminary investigation of a corpus, but might be unacceptable in legal proceedings because of the possibility of incorrectly excluding key evidence in a case. That suggests that we use only hashes that appear in results of at least K methods for some integer K (Table 7). K = 2 will give an estimated maximum error rate of 0.00175 when combining SZ and CD since the methods are relatively independent (and confirmed by experiments to be described in Sect. 7). So eliminating files with hash values appearing in at least two of our remaining eight methods, we obtained 11,181,072 hash values and 34,190,203 remaining files, 40.8 % of the original set of files, without any recourse to commercial hash sets.

Ruback et al. [13] suggested another criterion for interesting files, whether their "libmagic" or header analysis is inconsistent with their file extension, because then they may be camouflaged. SleuthKit/Fiwalk can provide libmagic strings for the files it analyzes. We manually defined a mapping on the 2304 distinct libmagic strings

Table 7. Number of files and hashes remaining after filtering out files having a given number of matches to our eight new hash sets.

Number of matches to hash sets required	0	1	2	3	4	5	6	7	8
Logarithm of number of files matched	18.1	17.5	17.0	16.8	16.5	16.3	16.0	15.5	13.5
Logarithm of number of distinct hash values matched	16.8	16.1	15.9	15.5	15.0	14.6	14.0	12.2	9.5

generated for our corpus to our set of 45 file-extension groups, so for instance "GIF Image Data" was mapped to "graphics extension". We compared the extension groups for the files in our corpus with the assigned libmagic extension groups. Of the 17,814,041 file records having libmagic values (since it was only added to Fiwalk recently), only 27.8 % groups matched between extensions and libmagic. A significant number of files do not have extensions, libmagic strings are insufficiently detailed about configuration files, and some records indicated outright errors in either extensions or libmagic. We conclude that libmagic classifications are insufficiently reliable as a way to detect camouflage. However, this subject needs further investigation.

7 Final Hash and File Eliminations

Files for which SleuthKit did not find hash values, 16.2 % of our remaining files, can also be eliminated if they match some of our criteria. Missing hash values occur for deleted files with incomplete metadata. For these we applied the BD (bottom-directory), TD (top-directory), and EX (extension) criteria to individual file records, and eliminated files matching at least two criteria. 2.5 million files matched BD, 10.0 million matched TD, 1109 matched EX, and 5.9 million matched two of the three criteria. For deleted files having just a file name, we inferred a path when it was unambiguous in 0.28 million cases, correcting underscores for missing characters if necessary. For instance, file name REXXUTIL.DL_ was inferred to have path OS2/DLL/REXXUTIL.DLL since only one path in our corpus had that file name after correcting the underscore.

We also eliminated in the final processing 2.9 million records of default directory files (the DR criterion), 4.1 million files smaller than 6 bytes, and executables, and files with extensions strongly suggesting uninterestingness: executables, support for the operating system, installations, updates, help, hardware-related files, and games.

This reduced the number of potentially interesting files to 16,923,937, 20.2 % of the original set of files. (If we allow matching to only one of the three criteria and not just to files without hash values, we reduce the number of files to 11.1 % of the

original set, but increase the error rate as estimated in the last section.) 83.9 % of the remaining files were deleted (they are ignored by most methods) and many could also be eliminated in investigations. To assess correctness, we took a random sample of 1000 files excluded in all the phases and found that only one file was possibly incorrectly excluded, a flowchart that had been downloaded with a cluster of other files. Thus we estimate the false negative rate at 0.1 %. We also took a random sample of 1000 of the files identified as interesting and found that 19.0 % of them were uninteresting false positives, not too large a number.

The last column of Table 5 gives the distribution of extension types for the remaining "interesting" files. Filtering increased the fraction of camera images, temporaries, Word documents, presentations, spreadsheets, and integer extensions as expected, but did not exclude all non-user files due to missing metadata.

As mentioned earlier, we also tested two commercial hashsets, the full database of Hashsets.com and a sample of the database of Bit9. Hashsets.com matched hashes on 5,906 of the remaining hashes, an unimpressive 0.06 %. We confirmed matches to the Bit9 database for 79,067 of the remaining hashes; we had difficulty identifying which actually matched, but we estimate Bit9 had hash codes for about 154,000 or 1.56 % of the remainder, 0.73 % of the original number of hash values. (Bit9 stores hashes in a proprietary encoding, so to guess which hashes it matched we used the file name and size returned by Bit9 and matched them to our own data, then extrapolated the number.) So Bit9 is also not much help in reducing files beyond our own methods, something important to know since it is costly.

8 Generality of the Hashes Found Over Corpora

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, roughly drives processed before 2012 and those processed in 2012. We extracted uninteresting hash values using our methods for both separately, and then compared them. Table 8 shows the results.

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.

Table 8. 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

9 Proposed File-Elimination Protocol

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

1. Run methods HA (hashes), PA (paths), WK (weeks), SZ (sizes), and BD (bottom-level directories) to generate hash sets of candidate uninteresting files on the full corpus to see trends.
2. Eliminate all files whose hash values are in NSRL, our list at digitalcorpora.org, and any other confirmed "uninteresting" lists available.
3. On the remaining files, run methods TM (times), CD (directory context), TD (top-level directories), and EX (extensions) to generate further hash sets of candidate uninteresting files.
4. Find hash values that occur in at least two candidate hash sets, and eliminate files from the corpus with those hash values.
5. Eliminate files without hash values in the remainder from the corpus that match on two of the three criteria BD, TD, and EX, are directories, are small files, and have strongly-uninteresting extensions.
6. Save the list of eliminated hash codes for bootstrapping with future drives in step 2.

For the final run on our own corpus with additional new data, eliminating NSRL hashes reduced the number of files by 25.1 %, eliminating hashes found by the nine methods of this paper reduced the number by an additional 37.1 %, and eliminating files by the remaining criteria reduced the number by an additional 21.6 %, resulting in

20.2 % of the original number of files. The disadvantage of this approach is that does require additional computation on a corpus before starting to investigate, not just matching to hash lists; the advantage is that if finds considerably more files to eliminate.

10 Conclusions

Although uninterestingness of a file is a subjective concept, most forensic investigators have a precise definition 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 such uninteresting files beyond just looking them up in the NSRL hash library. 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 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 [13]. We have already published some of our common-hash and common-path data for free download on digitalcorpora.org and will be publishing more soon based on this current work. Future work will extend this work to hashes on portions of files as in [10].

Acknowledgements. Riqui Schwamm assisted with the experiments, and Simson Garfinkel provided the corpus. The views expressed are those of the author and do not represent those of the U.S. Government.

References

1. Agrawal, N., Bolosky, W., Douceur, J., Lorch, J.: A five-year study of file-system metadata. ACM Trans. Storage **3**(3), 9:1–9:32 (2007)
2. Chawathe, S.: Fast fingerprinting for file-system forensics. In: Proceedings of the IEEE Conference on Technologies for Homeland Security, pp. 585–590 (2012)
3. Garfinkel, S., Farrell, P., Roussev, V., Dinolt, G.: Bringing science to digital forensics with standardized forensic corpora. Digit. Invest. **6**, S2–S11 (2009)
4. Ke, H.-J., Wang, S.-J., Liu, J., Goyal, D.: Hash-algorithms output for digital evidence in computer forensics. In: Proceedings of the International Conference on Broadband and Wireless Computing, Communication and Applications (2011)
5. Kornblum, J.: Auditing hash sets: lessons learned from jurassic park. J. Digit. Forensic Pract. **2**(3), 108–112 (2008)
6. Mead, S.: Unique file identification in the national software reference library. Digit. Invest. **3** (3), 138–150 (2006)
7. Panse, F., Van Keulen, M., Ritter, N.: Indeterministic handling of uncertain decision in deduplication. ACM J. Data Inf. Qual. **4**(2), 9 (2013)
8. Pearson, S.: Digital Triage Forensics: Processing the Digital Crime Scene. Syngress, New York (2010)

9. Pennington, A., Linwood, J., Bucy, J., Strunk, J., Ganger, G.: Storage-based intrusion detection. ACM Trans. Inf. Syst. Secur. **13**(4), 30 (2010)
10. Roussev, V.: Managing terabyte-scale investigations with similarity digests. In: Advances in Digital Forensics VIII, IFIP Advances in Information and Communication Technology vol. 383, pp. 19–34. Pretoria SA (2012)
11. Rowe, N.: Testing the national software reference library. Digit. Invest. **9S**, S131–S138 (2012). (Proc. Digital Forensics Research Workshop 2012, Washington, DC, August)
12. Rowe, N., Garfinkel, S.: Finding suspicious activity on computer systems. In: Proceedings of the 11th European Conference on Information Warfare and Security. Laval, France (2012)
13. Ruback, M., Hoelz, B., Ralha, C.: A new approach to creating forensic hashsets. In: Advances in Digital Forensics VIII, IFIP Advances in Information and Communication Technology vol. 383, pp. 83–97. Pretoria SA (2012)
14. Tomazic, S., Pavlovic, V., Milovanovic, J., Sodnik, J., Kos, A., Stancin, S., Milutinovic, V.: Fast file existence checking in archiving systems. ACM Trans. Storage **7**(1), 2 (2011)