Finding anomalous and suspicious files from directory metadata on a large corpus

Neil C. Rowe and Simson L. Garfinkel

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

Abstract: We describe a tool **Dirim** for automatically finding files on a drive that are anomalous or suspicious, and thus worthy of focus during digital-forensic investigation, based on solely their directory information. Anomalies are found both from comparing overall drive statistics and from comparing clusters of related files using a novel approach of "superclustering" of clusters. Suspicious file detection looks for a set of specific clues. We discuss results of experiments we conducted on a representative corpus on 1467 drive images where we did find interesting anomalies but not much deception (as expected given the corpus). Cluster comparison performed best at providing useful information for an investigator, but the other methods provided unique additional information albeit with a significant number of false alarms.

Keywords: forensics, directories, files, deception, extensions, clustering

This paper appeared in the 3rd International ICST Conference on Digital Forensics and Cyber Crime, Dublin, Ireland, October 2011.

1 Introduction

Preliminary assessment of drive data during a digital forensic examination can be daunting because of the amount of data. It is desirable to have ways to summarize drives quickly to determine if a drive and any particular files on it are worth investigating. One aspect of interestingness is the degree to which files on a drive are "suspicious" or appear to be out of character with similar files on other drives or appear to be concealing information. The contents of each file can be examined individually to find clues, but this takes time.

We are interested in determining suspiciousness automatically and solely from directory metadata (file names, extensions, paths, size, times, fragmentation, status flags, and hash codes) [3] that can suggest which file contents might be worth looking at. Automated analysis of suspiciousness could provide valuable consistency because different manual investigations find different files. Metadata analysis will not catch

more sophisticated methods to hide information like steganography or putting data into slack space [9]. But these will be rare in criminal investigations because routine concealment of information impedes its availability for use, and drives are often seized unexpectedly so criminals do not have time to hide things well.

Suspiciousness includes both anomalousness (the degree to which objects deviate from a norm) and deceptiveness (the degree to which a viewer is encouraged to perceive something other than the truth). Both are important, as there are many legitimate reasons that files can be anomalous on the one hand or apparently deceptive on the other. In the context of a digital forensic investigation, anomalousness can be measured by comparing statistics over a set of drives and within a drive. Deceptiveness can be found in specific clues to concealment or misdirection. An investigator seeking suspicious digital data should be alerted to both.

This paper reports on a tool **Dirim** ("directory imager") that will analyze the file metadata for a set of disks and report anomalies and suspiciousness that it finds, a process of 51 steps producing 204 analysis files. It takes advantage of data from a large corpus of representative disks to identify anomalousness with a wide perspective. **Dirim** is written in Python. It takes about a day to analyze 1467 drives on modern 64-bit machine cores, and ten minutes to analyze a new drive, most of which is clustering. It only needs to be run once on a corpus to provide baseline data, and the subsequent processing of a new drive can usually be done in a few minutes.

2 The testbed and its preliminary processing

The experiments reported here were conducted on a copy of the Real Drive Corpus (RDC) [8] as of November 2010. Our copy contained 1467 drive images containing 8.5 million files. The drives were purchased as used equipment. Some images were for computer disks and some were for portable devices, all are claimed to be non-U.S. in origin, and nearly all were purchased over a period of ten years. They represent a variety of users and usages including business users, home users, and servers of several kinds. We supplemented the corpus with 11 drive images created for testing purposes but not containing deception.

File metadata including filepath and name, file size, MAC times, NTFS flags (allocated, empty, compressed, and encrypted), fragmentation status, as well as cryptographic hash codes of the contents were extracted using our open-source **fiwalk** utility [7]. We found 8,673,012 files for which there were 2,693,135 unique paths. 32.1% of these were “unallocated” or marked for deletion. 60.7% of the unallocated files were marked as “orphan,” meaning that only the file name could be discerned without corresponding path information. Some of these were file fragments. Fragments on a drive that have entirely lost their metadata can only be found by "file carving" but this work does not consider them.

With so many deleted files, and knowing they are especially interesting for detecting deception, we try to correct and complete their file-path names for purposes of this analysis. FAT file systems often modify the first characters of directory and file names of deleted files to character 0xE5, as well as the last character of the file extension, and Fiwalk reported these as underscore symbols. We can exploit the

benefits of a large corpus to infer the missing characters from similar strings. For instance, _ONTENT.INI can only match CONTENT.INI in our corpus, and wzcsapi.dl_ can only match wzcsapi.dll. We also used a second heuristic of correcting generally unambiguous file extensions (e.g. ".ex_" to ".exe" and ".dl_" to ".dll"), and a third heuristic of inserting missing path information when there was a unique matching undeleted path with the same filename. Of the 322,676 opportunities to correct underscores in our corpus (11,165 with leading underscores of filepaths, 126,703 with ending underscores on file extensions, and 184,808 with leading underscores on subdirectory or file names), the first heuristic found 149,109 corrections and the second heuristic found 32,139 additional corrections, for a total of 56% of the opportunities. More than one correction was found for the same path in around 2% of the cases. Examination of a sample of the automated corrections found that they all were justified. More can be done to reconstruct deleted filepaths, however [12].

We assign files to semantically meaningful groups like pictures, spreadsheets, and word-processing documents. Three kinds of groups were defined based on file extensions (like "htm"), top-level directories in which the files occurred (like "WINDOWS"), and immediate directories in which the files occurred (like "pics"). For immediate directories that are ambiguous, we search their parent directories recursively to find one with an unambiguous group assignment. We defined the mapping of files to groups using expert-systems methodology, by researching extension names and software terms on the Web. Currently we assign all extensions and directories that occur at least 500 times in our corpus; we are gradually extending coverage to less-frequent ones currently classified as "Other". For grouping file extensions, we used Wikipedia's list of common extensions and the lists of www.file-extensions.org. For grouping directory names, we used common-sense knowledge for known items and we ran Google queries for unknown items. For instance, "pictures", "pics", "image", "art", "fotos", "gifs", and "sample pictures" all map to the "image" category of immediate directory, but "maps" does not because it could also be an abstract mapping. For the European-language words in the RDC (mostly Spanish and German) we used a dictionary (like for "fotos"). For Asian languages we looked for English subdirectory names to gives us clues to their meaning; for example, a directory with a Chinese name had a subdirectory of "microstar driver (f) 540", so it appears to be a hardware-control directory. Extensions and directories that were multipurpose (like extension ".enc" for encodings, encryptions, encyclopedias, electronic navigation charts, and the Encore music tool), overly general (like "files" and "other"), or represented user-specific names (like "joe"), were mapped to the category "Other". Currently 4214 items are mapped into 77 categories.

The proportion of file groups in our corpus gives a good picture of typical file usage (Table 1). We found this semantically-based grouping provided better understanding of the distinctiveness of a drive than the raw extension and directory names on the one hand, and traditional non-semantic metrics like average file and directory sizes on the other [1].

Most files of forensic interest on a drive are created and modified by users. So most of the analysis that follows excludes files of the operating system, applications software, hardware, most temporaries, and caches, leaving what we call the "user-file subset" (including temporary Internet files and deleted files since they can be quite

interesting). The techniques we describe are general, however, and will also be useful in forensic investigation of operating systems to find, say, evidence of malware in executables differing from those on most computers. Filtering was done by matching the extensions and directory names to the taxonomy above with designated exceptions

Table 1: File percentages in our corpus.

Extensions		MS OS	18.0%	Graphics	17.4%	None	12.7%
Other	11.3%	Camera images	7.6%	Web	6.3%	Execut-ables	4.3%
Config-urations	3.4%	Tempo-raries	3.2%	Non-MS docs.	2.7%	Logs	2.6%
Links	1.6%	MS Word	1.4%	Audio	1.4%	Help	1.4%
Encoded	1.1%	Program source	0.9%	XML	0.8%	Integers	0.6%
Copies	0.6%	Queries	0.5%	Spread-sheets	0.4%	Disk images	0.3%
Database	0.3%	Present-ations	0.2%	Video	0.2%	Geogra-phic	0.2%
Top dirs.		MS OS	33.6%	Deleted	17.6%	Software	17.5%
Docs. and Settings	13.6%	Non-MS OS	4.5%	Other	3.2%	Temps.	3.3%
Hardware	2.3%	Other docs.	2.3%	Root	0.9%	Games	0.5%
Immediate dirs.		OS	20.0%	Root	16.8%	Web	14.9%
Temps.	10.1%	Applics.	9.2%	Images	7.1%	Other	4.5%
Docs.	4.4%	Hardware	2.8%	Help	1.7%	Security	1.6%
Codes	1.4%	Programs	1.3%	Backup	1.0%	Sharing	0.9%
Audio	0.6%	Video	0.3%				

such as the "Applications Data" directory under Windows. For the RDC, the user-file subset was 1,583,951 files or 18.3% of the original number. The NIST National Software Reference Library Reference Data Set could aid this filtering, but (1) most of its entries are easy to classify by their extensions and directories, (2) it provides limited coverage of the many auxiliary files created by and used by software, and (3) we found that most non-user files that we miss in filtering are easily distinguished in clustering.

3 Finding anomalies

Anomaly detection requires norms for drives and the kinds of files stored on them. We would like to be able to say, for instance, that a drive has an unusually large number of JPEG images or an unusually large number of specialized applications.

Dirim uses two complementary methods for anomaly discovery: comparison of predefined semantic groupings, and comparison of derived clusters of files.

3.1 Comparing drive averages

A first way to find suspicious drives is to find those with feature counts or numeric properties (like file size) that are very high (above the 95th percentile) or very low (below the 5th percentile) for a large corpus of drives. On our corpus we found 1930 such high values and 211 low values over 106 counts and properties, excluding empty and unanalyzable drives.

Below are results for a computer from the Palestinian Authority and a computer of ours whose file system was built for teaching forensics. The second is anomalous because it has too many files (i.e., it is in the 95th percentile on the logarithm of the file count), files unusually small, files too often created during the day, files with no extension, and files in the root directory. Drive averages also permit characterizing the time patterns of usage on a drive using our previous work [14].

Analysis of drive 1450: PS01-097.xml with 27762 file records
Operating system: Windows NT
92.6% nondeleted and 5.29% temporary
Time behavior: day-evening-business-user(3)
Above 95th percentile for frac_other_Microsoft_Office_extension: 0.0035
Above 95th percentile for frac_email_extension: 0.0038
Above 95th percentile for frac_documents_botdir: 0.0916

Analysis of drive 1453: seed1.xml with 101407 file records
Operating system: Windows NT
67.9% nondeleted and 3.13% temporary
Time behavior: business-user(0)
Above 95th percentile for log_count: 11.526
Below 5th percentile for log_file_size: 6.4341
Above 95th percentile for frac_day_creation: 0.9959
Above 95th percentile for frac_no_extension: 0.3779
Above 95th percentile for frac_root_directory: 0.3166

3.2 Exploiting associations between files

To find subtler anomalies, like a few encrypted data files with no counterpart on other drives, we can cluster the drives and compare their clusters. Clustering of files based on their properties can find similarities even if directories have been renamed or moved. Clustering of files for the narrower domain of scientific computing significantly aided understanding of the file system [6].

To cluster, a degree of association between pairs of files must be quantified. With only directory information, the possible factors are:

- Temporal association: Files whose creation or modification times are within a threshold, suggesting causal relationships.
- Spatial association: Files in the same directory of a file system, suggesting functional (as with software) or thematic (as with media) relationships.
- Format association: Files with the same extension (e.g. "htm") on their name, indicating access by the same software.
- File-name association: Files with the same name excluding the extension, like "manual.exe" and "manual.txt" as well as deleted copies of a file.
- File-immediate-directory association: Files in the same subdirectory.
- Top-level association: Files under the same top-level directory of a file system (like "Program Files" in Windows for applications software).
- Co-occurrence tendency: File pairs with a high probability that they will occur together on a drive, judging from data on a corpus.
- Content hash association: Files having identical cryptographic hashes on their contents, indicating almost-certain identity of content. This would likely be calculated in an investigation though it is not directory information.

An investigator may find it useful to cluster several ways with different weights of factors. For instance, file associations based primarily on time and space will be helpful in finding social connections between users of drives; associations based on co-occurrence will be helpful in connecting software to their data files; and associations based on hash codes suggest patterns of copying.

3.3 Associations based on cryptographic hashes and file co-occurrence

Our full corpus had 2,067,421 intra-drive hash-match pairs according to the SHA hash algorithm, so hash matches are not uncommon. Agglomerative clustering (the simplest for non-metric associations) resulted in a big cluster for the operating system and applications software, and just a few small clusters for the rest of the files. Restricting analysis to the user-file subset of the corpus reduced the number of associations found to 307,876 pairs that did form multiple interesting clusters. We did not check inter-drive hash matches because these can better be found by other means.

There were surprisingly few matches between filepaths in our corpus. Even replacing digits by "#" and code letters by "?", 2.2 million of the 8.5 million filepaths in our corpus occurred only once, and only a few hundred files occurred on 90 drives or more of the 1467. Since most disk drives were of the Windows operating system, this shows how often Windows renames and reorganizes paths with each release. It is thus necessary to generalize paths to find more associations. Best results were obtained with the "shortened filepath", the file name plus the immediate directory (e.g. "bin/emcs.exe"). Using the file name alone gave many false matches (e.g. on “readme.txt”), and using more than the immediate directory reduced matches considerably. Figure 1 shows the histogram for matches using the file name plus immediate directory, for those files that occurred on 10 or more drives. Note this is not a Poisson distribution.

Co-occurrence of shortened filepaths can be measured as the conditional probability on a disk of one filepath given the other, or the average of the conditional probabilities in both directions, or the ratio by which the occurrence exceeds the value

predicted by a log-linear model, or the F-score. In testing these on our corpus, best performance was obtained with the F-score, equivalent here to $2n_{i\&j} / (n_i + n_j)$ where n_i is the number of disks on which file i occurs and $n_{i\&j}$ is the number of

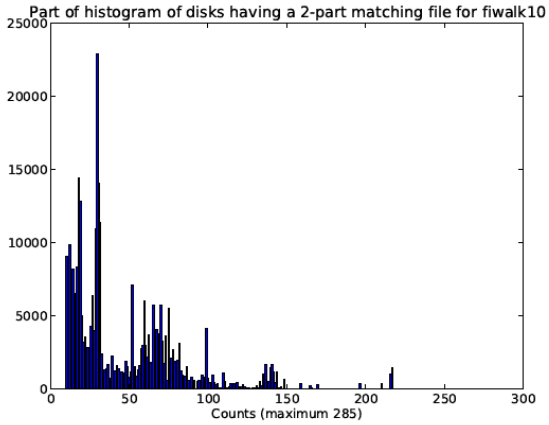

Figure 1: Number of file matches (vertical axis) occurring on a particular number of disks (horizontal axis), using just file name and immediate directory to match.

disks in which both i and j occur. The F-score is 1 for perfect bidirectional association, but unlike conditional probabilities, is not as close to zero for unidirectional associations that indicate, among other useful phenomena, software options.

Dirim analyzes co-occurrences by keeping a count of the number of drives in the corpus that have each possible pair of shortened filepaths where both have more than a threshold count (currently 10). To avoid excessive storage requirements, it partitions the set of filepaths into manageable subsets (30 sufficed for our corpus), and works on one subset at a time, comparing a member of the subset to the other filepaths of the entire corpus. File co-occurrences are more significant if they are also hash matches on content or temporal associations within a threshold. So **Dirim** keeps two running counts for each shortened-filepath pair: its count of occurrences over drives, and its count of temporally or hash-associated occurrences over drives. We chose ten minutes as the threshold for creation and modification times because most software updates are accomplished in that period. (While we observed previously that some operating systems round times to the nearest hour, they round consistently.) We did not consider access times because many unrelated files are accessed on startup of an operating system or software.

5,343 pairs in the user-file subset of the corpus exceeded a co-occurrence F-score of 0.8 and a temporal window fraction of 0.8. While this is not many, those that were found were interesting forensically as user-created associations. The number of pairs found for the full corpus with the same parameters was 16,975,684, so there are considerably more associations among software-related files.

3.4 Clustering of drive files

The main way in which we exploited the associations between files listed in section 3.2 was to do clustering. Clustering using the K-Means algorithm is desirable because it is fast, robust, and usually accurate [15]. Parameters were optimized by testing on a "gold standard" of a 1000-item random sample of files that was clustered by our program and then manually adjusted to make the most intuitively reasonable clusters. Our metric of clustering performance was the fraction of pairs in the gold standard that were in the same cluster in the test set plus the fraction of pairs in the same cluster in the test set that were in the same cluster in the gold standard.

Dirim uses its own implementation with automatic splitting and merging to minimize the effects of the choice of the initial cluster centers. 15 cycles of clustering and merging sufficed to achieve convergence in nearly every case. To determine the starting number of clusters K for a desired target number of clusters M, we fit the relationship between number of data points and final number of clusters over a representative sample of drives. For N data points of this data, the relationship we found was linear with equation 2+(N/10) for N<10M, and then logarithmic with equation M(1+0.05*ln(N/M)).

Numerous experiments were conducted to optimize parameters. 24 properties of the files were identified as being the most useful. The numeric ones were the file size, the modification time minus the creation time, the access time minus the creation time, the access time minus the modification time, the depth of the file in the directory hierarchy, the length of the file name, the fraction of alphabetic characters in it, the number of non-ASCII characters in it, the number of drives in the corpus having this file, and the number of files in its immediate directory.

Extension and directory names are important independent information for clustering. Since they are not metric properties, and K-Means requires metric data, we assign a seven-element feature vector to each extension and directory group listed in section 2 and use those in metric clustering. The elements represent the major features of the extension or directory. They are: (1) the degree to which the group is associated with the operating system, (2) the degree to which it is associated with applications software, (3) the degree to which it is associated with documents, (4) the degree to which it is associated with media, (5) the degree to which it is associated with specialized usage of a computer system, (6) the degree to which files are updated frequently, and (7) the degree to which the file is "user-owned" in being created or modified by user initiative. Degrees of association were assigned by our judgment and most were clear. For instance, spreadsheet extensions were assigned (0.0, 0.2, 0.8, 0.0, 0.0, 0.8, 1.0) and disk-image extensions (1.0, 0.0, 0.0, 0.0,0.0, 0.2, 0.0).

All property values were normalized and mapped onto ranges of 0 to 1 by functions of the form $f(x) = \Phi((x - \mu_x)/\sigma_x)$ for ordinary properties, or $f(x) = \Phi((\log(x) - \mu_{\log(x)})/\sigma_{\log(x)})$ for widely varying properties like file and directory size, where Φ is the integral of the normal distribution with mean of 0 and standard deviation of 1, μ_x is the mean of the property over the entire corpus, and σ_x is the standard deviation. This transformation maps the value to its fractional rank order in the sorting set of all values assuming it has a normal distribution, and most of

the properties were close to normal; it provides a quick estimate of rank order without sorting. Hence μ_x maps with this transformation to 0.5 and $\mu_x + \sigma_x$ to 0.84. We have found that this transformation works well for input to K-Means since, unlike neural networks, most versions of K-Means do not adjust the weighting over time because it is difficult to know how. By spacing values evenly 0 to 1, we get a maximum discrimination of differences between them on the same range. If the ranges were not equal, an unusually large value of one property would override distinctions among the other properties, a problem we had frequently before adopting this transformation on input values.

Weighting the properties evenly gives a broad understanding of the data, but other weightings enable exploration of different aspects of the data. **Dirim** provides seven weights as input to provide investigator control: on extension groups, on directory groups, on file size, on time measures, on depth, on filename properties, and on file counts.

We explored several ways of incorporating the results of co-occurrence analysis as per section 3.3. When we merged clusters having any pair of files with a significant co-occurrence, this tended to create far too many merges since many co-occurrences were due to software updates. So now, after regular clustering is done, **Dirim** modifies the cluster assignment of one element of the pair to be in the same cluster as the other, choosing to change the cluster assignment of the item in the smaller cluster. This works well at preserving cluster structure while enforcing the co-occurrence relationships.

Although K-Means is a stochastic algorithm, we saw little variation in its results on the disks on repeated runs (though there was more variation in the superclustering to be described). Apparently the capability to split and merge clusters compensates for a poor choice of starting cluster centers.

3.5 Superclustering of drive clusters

Once a corpus has been clustered, we can compare clusters on a new drive with those on the corpus to find anomalous clusters on the new drive. A good way to do this is to cluster the clusters for each drive into "superclusters". Then clusters on the new drive outside the superclusters will be anomalous. Comparing clusters requires considerably less effort than comparing individual files and can be done in minutes. Superclustering can use the same properties as clustering, now as the means of clusters found previously, with the useful addition of the size of each cluster (since two drives with similarly-sized clusters are more alike than otherwise). The target number of superclusters should be larger than the average target number of clusters over the drives since superclusters cover more diversity than a single drive. We found that superclustering had more difficulty converging than clustering and required a more careful choice of parameters such as starting number of clusters and adjustment rates of the merging and splitting thresholds. This suggests that the clusters found for each disk did not always have clear counterparts.

The drives in our corpus included mobile devices and storage devices as well as computers. It makes sense to supercluster only within similar device types. To do this, **Dirim** identifies the operating system (if any) on each drive using a set of rules based

on the directories and files found on the drive. On the corpus it identified 68 drives as Windows 95 or 98, 19 Windows 2000 or 2003, 139 Windows NT, 108 Windows XP, 1 Windows 7, 44 MS-DOS, 1 IBM OS/2, 6 Macintosh or Linux, 216 mobile devices with cameras, 206 storage devices, 27 other kinds of mobile devices, 1 printer, 75 unidentifiable, and 556 empty of the 1467 total. Then we do three major superclusterings: Windows computers, mobile devices, and storage devices.

3.6 Results

We clustered 335 Windows drives with 50 as the target number of clusters, and then superclustered the clusters to obtain 63 superclusters with 100 as the a target number, using an even weighting of the factors. Superclusters were analyzed for extreme values. For example, supercluster 695 was cited for having many files that change often, for having a high fraction of documents, and being unusually big.

Figure 2 plots the superclusters on the two largest principal components of the properties used for clustering them, and Figure 3 plots the second and third components. Size of the circle represents the number of clusters in the supercluster. 63 superclusters were found with a target goal of 100, of which the largest (in the lower left) contained 2863 clusters. Generally speaking the clusters in the lower left of Figure 2 were high on the property of being associated with the operating system and software, while those on the upper right were high on the scale of being user-owned and involving specialized usage. The third component defies easy categorization.

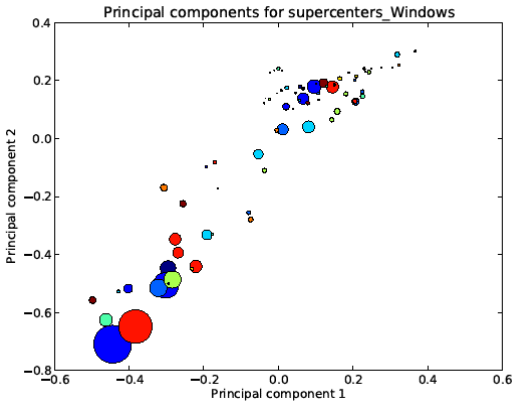

Figure 2: First versus second principal components of the superclusters of user files on Windows drives.

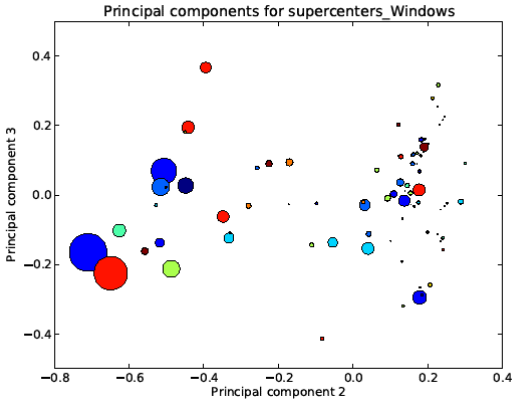

Figure 3: Second versus third principal components for the superclusters.

The ten largest superclusters found were:

- two big superclusters of small operating-system data files not excluded by our imperfect initial filtering (sizes 3653 and 2863)
- a diverse supercluster of large downloaded files (size 1660)
- a supercluster of graphics used by the operating system (size 763)
- a supercluster of large operating-system data files (size 751)
- a supercluster of downloaded Web files (size 604)
- a supercluster of downloaded media files (size 465)
- a supercluster of foreign-language operating-system files (size 411)
- a supercluster of specialized files (size 380)
- a supercluster of cache files (size 362)

The superclustering of the mobile-device files found two well-separated groups of superclusters distinguished mostly on the principal component by depth in the file hierarchy and use of specialized extensions (clearly a distinction of system versus user files). The second component mainly distinguished access time minus creation time. The superclustering of the storage-device files was not especially interesting. The superclustering of the full set of files on the Windows drives (including the operating system and applications software) showed, unsurprisingly, large clusters for the operating system with the user-owned files off to the side.

To test the robustness of the superclusters found, we ran clustering and superclustering on random samples of half the Windows drives. We compared superclusters on the reduced set with those of the full set by counting, for each supercluster in the reduced set, the number that were mapped to the most popular supercluster in the full set for elements of the reduced-set supercluster. We obtained an average of 63.0% on this metric. This is encouraging in confirming consistency considering that many superclusters were in wide areas where clusters could be formed in many ways, as on the right side of Figure 3.

Small superclusters are of the most interest to an investigator. Five single-element superclusters and five two-element superclusters were found for our corpus for the

Windows drives. Five of these contained links files; one contained browser settings; one contained browser history; one contained fonts; one contained temporary data files; and one contained JPEG images. All of these could be of interest, but the JPEG cluster was especially interesting because the fact it was not merged into another supercluster indicates that its images had anomalous properties. The other small clusters could indicate anomalous Internet browsing.

Larger superclusters with anomalous centroids like the ones in the upper left and lower right of Figure 3 are not as interesting to the investigator as the small superclusters, but can be examined to find odd frequent usage that could confuse an investigation. Absence of one drive's clusters from a large supercluster might be helpful in suggesting there is something odd about a drive, but the drives in our corpus were so diverse that this metric could not usefully be applied to them. Deleted files missing most of their metadata were inconsistently clustered, so filling in their directory information by analogy when possible (see section 2) was quite valuable.

In comparison to 1140 flagged extremes in the raw drive comparisons of section 3.1, these clustering results are considerably more precise, and will save much time of an investigator. Note that superclustering also permits us to measure similarity of two drives as the fraction of their clusters that are in the same supercluster.

4 Finding deception

What we have described will find anomalies in files on a drive. Files of interest can also evidence deception with a set of clues. Lacking an investigatory focus as in [5], a forensic investigator can conduct a broad survey for deception, analogously to investigations of accounting fraud [2] and clandestine databases [10]. We were hampered in experiments by not having much deception in our corpus. Nonetheless, we found some interesting things.

Deception encompasses many phenomena. A useful start is Bell and Whaley's six-part taxonomy [4]: masking (pure concealment), repackaging (embedding characteristics of a suspicious pattern in a larger nonsupicious pattern), dazzling (providing distractions), mimicking (copying characteristics of something nonsuspicious), inventing (providing something new as a distraction), and decoying (providing a substitute for the original suspicious pattern). Masking and dazzling are hard to do in digital media where everything is bits and there are good automated tools for content searching (and masking by encryption is itself suspicious). Thus the most useful deception methods for directory information are repackaging and mimicking. Another taxonomy based on case grammar [13] enumerates 32 deceptions in space, time, participant, causality, quality, essence, and preconditions. Most opportunities for deception in a drive directory occur for qualities of a file, which are mainly conveyed by its path (file name and location in the directory scheme), as well as its times, flags, and fragmentation. By far the most useful deception for a criminal is concealment. So **Dirim** tests clues for apparent concealment.

4.1 Deceptive file extensions

Clue to suspicious files occur in their file extensions. Many references [11] cite double file extensions as suspicious since the last extension may conceal another. **Dirim** found 6,983 suspicious double extensions on the corpus after excluding 749,481 doubles judged as legitimate. Links, copies, and compression extensions like "lnk", "bak", and "zip" have legitimate double extensions to represent what is being pointed to or compressed. Files holding Web or email addresses often preserve the periods in the address, e.g. MSN.com.url and editor@www.washingtonpost.txt, and these look like double extensions. File names may contain periods for abbreviations, like "Oct. 2008 -- Inventory.xls", that can be confused with extensions. Some software may use multiple extensions for indexing, like "4.6.2.9". Some double extensions exemplify poor naming like "refs.txt.backup" which should be "backup.refs.txt". Unrecognized extensions longer than 3 characters are also suspicious as a way to conceal purposes. There were 17,365 of these in the corpus, and 6,123 of these were user files (probably reflecting user creativity in naming).

We examined a random sample of 100 of the suspicious extensions that the software found, and confirmed that 78% appeared to use some form of concealment, whether for legitimate commercial purposes or not -- for instance, accwiz.exe.txt. We also checked the Windows drives of the corpus to see if these double extensions tended to occur more often in the small superclusters, but no malicious extensions or double extensions appeared in the 3962 files of our superclusters of size 10 clusters or less, and only 6 extensions that were long and unknown, numbers well below the expected random count of 67.6. So we conclude that identification of double file extensions is not a shortcut to clustering in finding interesting files.

Rare extensions are also suspicious since they also can conceal function. So **Dirim** counts on how many disks each extension occurs (8,809 in our corpus), m_j for extension j of M extensions. Then for each disk i, average rarity of its extensions can be estimated as $r_i = \sum_{j=0}^{M-1} (o_{ij} / m_j) / \sum_{j=0}^{M-1} o_{ij}$ where o_{ij} means extension j occurred at least once on disk i. We found this metric mainly useful for rating the interestingness of a drive, as it was not helpful with discriminating clusters. All the drives above the 95th percentile on this metric for our corpus were definitely unusual, appearing to have many game files.

Sources like www.fileextensions.org list extensions generally associated with malware. Even if malware is not a primary target in forensic investigation, it suggests careless usage that may explain other things. We found 360 known malware extensions in our corpus, all in software directories. Some appear to be legitimate uses that predate the occurrence of the associated malware, as Corel/Suite#/Shared/IDAPI/EUROPE.BLL. So known malware extensions are probably too infrequent to be much help.

4.2 Suspicious paths

Files can also be deceptive with obfuscation in their paths. We found it useful to look for filepaths with significant numbers of code words and numbers. This was aided by implementing two generalized-character classes, substituting "#" for digits and "?" for letters adjacent to them or punctuation, and treating digits surrounded by letters, or leading or trailing a name with letters, as letters. This let us treat as identical many cache files like Norton Antivirus quarantine files 525D1233.tmp and 7EDC38F2.tmp. Directory names that start with a punctuation mark can also be suspicious because it is an easy way to obfuscate. We found 18,115 instances of apparent obfuscation in the full set of files and 8,462 in the user-file subset, so non-software files were disproportionately suspicious (although they included Windows "Documents and Settings" directory which has many odd subdirectory names).

We also sought directory and file names that were misspellings of common names, as another way to obfuscate, and found 7,088 occurrences of 3,952 distinct items in the corpus. False alarms were reduced by only counting misspellings differing by one alphabetic letter that were at least 10 times less common than their properly spelled counterpart in names at least 5 characters long.

We checked a random sample of 100 suspicious paths found by the software and confirmed that 34% were legitimately questionable compared to 1% of a random sample of all files; cache files were the major cause of false alarms. Example suspicious paths were #??????#???##########????#afaf/msgina.dll and DUBA######_SKY_##_###.zip which may be hashes but should have identified a program with which they are associated. Suspicious paths and misspellings did correlate with small superclusters in the Windows drives on our corpus. For 3962 files in superclusters of size 10 clusters or less, 93 of these had suspicious paths when the expected count was 47.0, and 93 of these had misspellings when the expected count was 67.2. This suggests a modest benefit can be obtained by looking for these clues first when investigating a drive.

An overt clue to concealment is encryption, so user-owned encrypted files should be especially interesting to an investigator. Files can be encrypted in several ways, and different methods are needed to detect each way. Analysis of the RDC found no instances of NTFS encryption, suggesting that this technology is rarely used. We are currently implementing a file-level encryption detector for the **fiwalk** framework.

4.3 Rating drive suspiciousness

To rate the overall suspiciousness of a drive, we can combine clues. We used a weighted average of the fraction of bad extensions on a drive, the fraction of bad paths, plus the rarity metric for extensions. Bad paths were weighted half because there were more legitimate reasons for them. Figure 4 shows the distribution of this metric on the 462 drives with at least 100 files. For 10 random drives rated over 0.2 by this metric, 6 featured large numbers of directory names that were code words, 2 had specialized file types, 1 had many Internet downloads, and 1 had considerable file sharing. For 10 random drives rated below 0.05, 1 had large numbers of directory names that were code words, 3 had had large numbers of specialized file types, and

the rest were not noteworthy. So the ratings appear useful in identifying disks with interesting features. Results did not differ much with changes to the weights on the three factors.

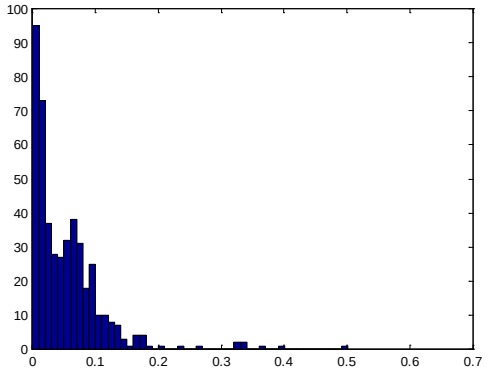

Figure 4: Histogram of overall drive suspiciousness based on its paths.

Clam AntiVirus was also run on a sample of our Windows drive images. Clam found 6874 files on 48 disks whose contents matched virus signatures. Overall correlation with the suspiciousness metric was mild, though some of the most-infested drives did have high suspiciousness ratings. We concluded that antivirus software is not helpful in assessing drive suspiciousness.

4.4 Clusters of deletions

Other clues to concealment are clusters of activity at suspicious times like just before the drive was captured from an insurgent during a military operation. To find deletion clusters, **Dirim** counts the deleted files (marked by the "unallocated" flag) by day for each drive. Days having unusually high numbers of deletions can be subject to further inspection; 10 times the expected rate was a good threshold. We saw many clusters of deletions in the corpus at the end of a drive's usage, representing when it was being prepared for being sold. [14] discusses more of what can be detected in analysis of file times.

5 Conclusions

Thorough analysis of a drive with hidden data of forensic importance is challenging. The **Dirim** tool described here should give investigators a good start using the more-accessible metadata. It permits investigators a way to size up a drive and its major uses in the context of a corpus of similar drives. We tested it with analysis of the directory information for 1467 drives, a larger number of general-

purpose images than has been previously studied. The most useful technique for finding anomalies was a clustering of the files of the drive and comparison of those clusters to those of other known drives in a corpus. The performance of the techniques for finding clues to deception was hard to assess because there was no major suspicious activity on these drives; nonetheless, some interesting concealment phenomena were found. Future work will test our software on disks with deliberately constructed deception.

This approach will not find all anti-forensics tricks (e.g. http://www.metasploit.com/research/projects/antiforensics) but it can find many clues even when filepaths and directory information have been manipulated. Our software and results on our corpus are freely available for further research.

Acknowledgements: The views expressed are those of the authors and do not necessarily reflect those of the U.S. Government. Michael Hom helped with the virus analysis, and Hector Guerrero and Jose Ruiz helped build specialized word lists.

6 References

1. Agrawal, N., Bolosky, W., Douceur, J., Lorch, J.: A Five-Year Study of File-System Metadata. ACM Transactions on Storage, 3, 3, p. 9 (October 2007)
2. Alsagoff, C.: Microsoft Excel as a Tool for Digital Forensic Accounting. In: Intl. Conf. on Information Retrieval and Management, Shah Alam, Malaysia, p. 97 (March 2010)
3. Buchholz, F., Spafford, E. On the Role of File System Metadata in Digital Forensics. Digital Investigation, 1, 298-309 (2004)
4. Bell, J., Whaley, B.: Cheating and Deception. Transaction Publishing, New York (1991)
5. Carrier, B., Spafford, E.: Automated Digital Evidence Target Definition Using Outlier Analysis and Existing Evidence. In: Proc. Fifth Digital Forensic Research Workshop (2005)
6. Doraimani, S., Iamnitchi,, A.: File Grouping for Scientific Data Management: Lessons from Experimenting with Real Traces. In: Proc. HPDC'08, Boston, MA (2008)
7. Garfinkel, S.: Automating Disk Forensic Processing with SleuthKit, XML and Python. In: Proc. Systematic Approaches to Digital Forensics Engineering, Oakland, CA (2009)
8. Garfinkel, S., Farrell, P., Roussev, V., and Dinolt, G.: Bringing Science to Digital Forensics with Standardized Forensic Corpora. Digital Investigation, 6, pp. S2-S11 (2009)
9. Huebner, E., Bem, D., Wee, C.: Data Hiding in the NTFS File System. Digital Investigation, 3, 211-226 (2006)
10. Lee, G., Lee, S., Tsomko, E., Lee, S.: Discovering Methodology and Scenario to Detect Covert Database System. In: Proc. Future Generation Communication and Networking, Jeju, China, p. 130 (December 2007)
11. Munson, S.: Defense in Depth and the Home User: Securing the Home PC. www.sans.org/reading_room/hsoffice/defense-in-depth-home-user-securing-home-pc_894.
12. Naiqi, L., Zhongshan, W., Yujie, H., QuiKe: Computer Forensics Research and Implementation Based on NTFS File System. In: Proc. Intl. Colloquium on Computing, Communication, Control, and Management, Guangzhou, China, pp. 519-523 (August 2008)
13. Rowe, N.: A Taxonomy of Deception in Cyberspace. In: Proc. Intl. Conf. on Information Warfare and Security, Princess Anne, MD, pp. 173-181 (March 2006)
14. Rowe, N., Garfinkel, S., Global Analysis of Disk File Times. In: Fifth International Workshop on Systematic Approaches to Digital Forensic Engineering, Oakland CA (May 2010)
15. Xu, R., Wunsch, D.: Clustering. Wiley-IEEE, New York (2008).