Oscar – File Type Identification of Binary Data in Disk Clusters and RAM Pages

Martin Karresand[1,2] and Nahid Shahmehri[1]

[1] Department of Computer and Information Science
Linköpings universitet,
Linköping, Sweden
[2] Department of Systems Development and IT-security
Swedish Defence Research Agency,
Linköping, Sweden
{g-makar,nahsh}@ida.liu.se

Abstract This paper proposes a method, called Oscar, for determining the probable file type of binary data fragments. The Oscar method is based on building models, called centroids, of the mean and standard deviation of the byte frequency distribution of different file types. A weighted quadratic distance metric is then used to measure the distance between the centroid and sample data fragments. If the distance falls below a threshold, the sample is categorized as probably belonging to the modelled file type. Oscar is tested using JPEG pictures and is shown to give a high categorization accuracy, i.e. high detection rate and low false positives rate. By using a practical example we demonstrate how to use the Oscar method to prove the existence of known pictures based on fragments of them found in RAM and the swap partition of a computer.

1 Introduction

There are many examples of situations within the information security field where the ability to identify the file type of a data fragment is needed. For example, in network based intrusion prevention the structure of the packet payloads can be used to filter out traffic not belonging to a certain service, and in that way defend against new and yet unknown attacks [1].

Another example would be a military network where only encrypted traffic is allowed. By looking at the structure of the payloads of the data it is possible to determine if there are any services sending unencrypted data over the network, without having to rely on packet headers, or checking well-known ports. It is even possible to do so without interpreting the payload, thereby avoiding the potential security compromise associated with an in-transit decryption.

A third example can be found in the computer forensics field where a forensic examiner might have to do a manual search through several hundreds of gigabytes of unrecognisable data. A possible scenario would be a forensic examiner conducting an examination of a computer with several large hard disks, all deliberately corrupted by the owner, who is suspected of trafficking in child pornographic pictures. The examiner uses the available tools and manages to retrieve a lot of JPEG picture headers and

Please use the following format when citing this chapter:
Author(s) [insert Last name, First-name initial(s)], 2006, in IFIP International Federation for Information Processing, Volume 201, Security and Privacy in Dynamic Environments, eds. Fischer-Hubner, S., Rannenberg, K., Yngstrom, L., Lindskog, S., (Boston: Springer), pp. [insert page numbers].

specific markers, but no complete and viewable pictures can be found. In theory the examiner could start brute force matching of every disc cluster against a database of known child pornographic pictures. In reality such a solution would not be feasible since the case would be closed due to time constraints and legal implications long before the matching operation had finished.

As described in the scenario the existing tools (for example PC Inspector [2], Sleuth Kit [3], and The Coroner's Toolkit (TCT) [4], or the commercial Encase [5], File Scavenger [6], and Search and Recover [7]) would not be of much help, because they all need some remnants of a file allocation table or file header to be able to recover lost files, and even if there are headers to be found the data might still be fragmented, making the tools unable to identify more than the header part of the files.

Unfortunately it is not only corrupted hard disks that cause problems for a forensic examiner, the situation has become even more complicated since the introduction of Knoppix and related live CD distributions [8]. These distributions have made it possible to run a full-fledged computer system from a CD, without ever touching the hard disk. All necessary files are loaded into RAM, with a working read/write virtual file system, and hence no traces are left on the hard disk. There are also several methods [9,10,11] for remote execution of code in RAM only on a victim host. Both RAM and swap partitions are fragmented and unstructured by nature, due to the constant changes induced by the CPU's process scheduling. Hence, if one of these techniques is used for illegitimate purposes, finding evidence in the form of complete binary data or executables would be hard considering the limited abilities of the existing tools.

Consequently there is a need for a method making it possible to identify the type of a data fragment by its internal structure only, and to the best of our knowledge there is no tool available that can do that. Our claim is supported by the National Laboratory of Forensic Science (SKL) [12] in Sweden. They have, together with the National Criminal Investigation Department (RKP) [13] in Sweden, expressed an interest in the ability to automatically bring order to a large amount of unsorted binary data. This paper proposes a method for identifying the type of a binary data fragment and subsequently mapping out the contents of various storage media.

The proposed method, called the Oscar method or simply Oscar, has been implemented in a few proof-of-concept pieces of software. Together, these utilities form a toolbox, the Oscar toolbox, consisting of a JPEG data extractor, a byte frequency distribution extractor, a data fragment categorizer, and a search tool for hexadecimal strings. To evaluate the concept we have used the JPEG standard and extracted fragments of a JPEG file from a disk image file. We have also tested the toolbox using the computer forensics scenario mentioned earlier.

2 Proposed solution

Hard disks store data in *sectors*, which are typically 512 bytes long. The operating system then combines several sectors into *clusters*. Depending on the size of the partition on the hard disk clusters can vary in size, but currently the usual cluster size is 4 kB, which often also is the size of *pages* in RAM. Using data blocks that are the same size as clusters and RAM pages, the Oscar method builds vector models, called *centroids* [14,

p. 845], of different file types based on statistics of their byte frequency distribution. The statistics are calculated using data in vector format, and by using the mean and standard deviation for each file type, unknown clusters or pages can be categorized. The similarity of the centroid and the sample vector to be categorized is then measured by calculating the distance between them in vector space. If the distance is less than a threshold, the sample is categorized as being of the same type as the centroid. To increase the detection rate and decrease the false positive rate, other type-specific features may also be used in the categorization process.

In Fig. 1 the histograms of the centroids of four different file types, .exe, JPEG, .ps, and .zip, are shown. As can be seen the histogram of the text-based .ps file is radically different from the binary-based JPEG, .exe, and .zip files. Please observe that the JPEG histogram is based only on the data part of the files used for the centroid. Even though the histograms of the binary files are fairly similar, they still differ, with the most uniform histogram belonging to the .zip type, then the JPEG, and last the .exe. All three binary file centroids also contain a higher number of 0x00 than other byte codes, probably due to padding. For the .exe the significantly higher rates of 0x00 and 0xff is worth noticing. The JPEG histogram shows a decrease in the higher byte codes, which might be a result of the RGB coding in combination with the lossy compression. Probably the cameras we used are tuned to what might be called a "normal" colour palette, and hence extreme values for one or more of the three parts of a RGB triplet is less likely in ordinary pictures. The lossy compression will also decrease the number of extreme values by reducing high frequencies in the pictures, i.e. rapid changes in colour and saturation. Additionally, the marker segment indicator 0xff is not to be used in the data part, apart from in a few special cases, and is therefore less likely to appear in the data.

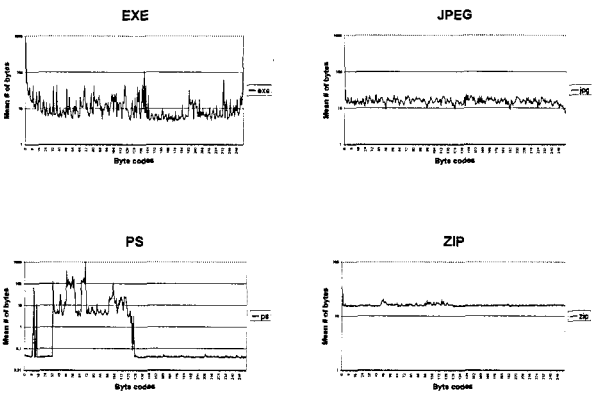

Figure 1. Histograms of the centroids of four different file types, .exe, JPEG, .ps, and .zip. Only the data part of the files was used when creating the JPEG centroid. A logarithmic scale is used for the Y-axis.

As mentioned earlier the JPEG standard has been used to illustrate the method. To further enhance the detection capability of the Oscar method we use the fact that byte

0xff is only allowed in combination with a few other bytes in the data part of a JPEG file. If there is any pair of bytes representing a disallowed combination, the block will not be categorized as JPEG.

In the following subsections we present and discuss some of the main theoretical aspects of the Oscar method, together with some implementation specific features. Oscar is meant to be generalizable, but is at the moment mainly aimed at finding fragmented JPEG picture disk blocks. However, the metric used for categorization is also applicable to other types of data.

2.1 Metrics

There are many ways of measuring similarity between data samples. The approach chosen for Oscar is to measure the distance between a sample vector and a centroid [14, p. 845]. A weighted variant of a quadratic distance metric is used.

Creation of Centroid. The term centroid is defined as the average vector of a set of vectors belonging to a cluster [14, p. 845], i.e. the vectors being categorized as the same type. In our case the centroid consists of two vectors representing the mean and standard deviation of each byte value's frequency distribution.

To create the centroid for the JPEG type we used 255 pictures of normal family life. The pictures came from 5 scanned paper copy photographs and 250 digital photographs from three photographic sessions using two different digital cameras. The pictures' data parts were extracted and concatenated into one file. The resulting file was then used to create a matrix, which was fed to GNU Octave [15] for calculation of the mean and standard deviation. The result of the calculation constituted the centroid.

Length of data atoms. The data the Oscar method uses is in the form of 1-grams, i.e. bytes, the simplest form of an n-gram. An n-gram is an n-character long sequence where all characters belong to the same alphabet of size s, in our case the ASCII alphabet giving $s = 256$. The byte frequency distribution is derived by bin sorting the bytes into bins of size one. These bins then form a vector representing the sample to be compared to a centroid.

By using 1-grams we do not take the order of the bytes in the data into consideration. Considering the order of the bytes decreases the risk of false positives from disk clusters having the same byte frequency distribution as the data type being sought, but a different structure. Using larger n-grams increases the amount of ordering taken into consideration and would consequently be a nice feature to use. However, when using larger n-grams the number of bins, b where $b \leq s^n$, also increases. The size of b depends on whether the n-grams have a uniform probability distribution or not. The maximum value is required when the distribution is uniform, or nearly uniform and the fact that a JPEG file is compressed gives it an almost uniform n-gram probability distribution.

Since the alphabet used has size 256 and we use 4 kB large data fragments, the Oscar method is in practice limited to the use of 1-grams. Using larger n-grams requires $b > 4096$, which in this case gives a mean for the frequency of each n-gram less than

one. Every sequence of characters longer than one byte will therefore have too large an impact on the calculations of the distance, and consequently the method becomes unstable and hence not usable.

Measuring Distance. The distance metric is the key to a good categorization of the data. Therefore we chose to use a quadratic distance metric, which we extended by weighting the difference of each individual byte frequency with the same byte's standard deviation. In Equation (1) the standard deviation of byte i is represented as σ_i. A smoothing factor, α, is used to avoid division by zero when $\sigma_i = 0$

$$d(\boldsymbol{x}, \boldsymbol{y}) = \sum_{i=0}^{n-1} (x_i - y_i)^2 / (\sigma_i + \alpha) \quad . \tag{1}$$

The advantage of using a more computationally heavy quadratic-based metric over a simpler linear-based metric is the quadratic-based method's ability to strengthen the impact of a few large deviations over many small deviations. Assuming the vector sums, i.e. $\|\boldsymbol{x}\|_1$ and $\|\boldsymbol{y}\|_1$, are constant, Equation (1) gives lower distance values for two vectors separated by many small deviations, than for two vectors separated by a few large deviations. A linear-based method gives the same distance, regardless of the size of the individual deviations, as long as the vector sums remain the same. Since we are using 4 kB blocks of data $\|\boldsymbol{x}\|_1 = \|\boldsymbol{y}\|_1 = 4096$ and we are, in reality, forced to use a quadratic distance metric to achieve decent categorization accuracy.

Some file types generate fairly similar histograms and it is therefore necessary to know which byte codes are more static than others to discern the differences in spectrum between, for example, an .exe file and a JPEG picture. We therefore have to use the more complex method of looking at the mean and standard deviation of individual byte codes, instead of calculating two single values for the whole vector.

2.2 Implementation of Tools

We implemented a toolbox in C containing a number of tools to support testing the Oscar method. The Oscar toolbox consists of a JPEG data extractor, *extr_data*, a byte frequency distribution extractor, *extr_1-gram*, a categorizer, *type_mapper*, and a hexadecimal string search tool, *hexgrep*.

The *extr_data* tool steps through the markers in the header of a JPEG file until the data part starts and then extracts the data part to a file. The *extr_1-gram* tool is used for the creation of the centroid. It mainly sorts and counts the bytes in one 4 kB block at a time. The output of extr_1-gram is fed to GNU Octave [15], which is used to create the mean and standard deviation vectors constituting the centroid.

The *type_mapper* tool is given an image file of a disk, RAM or swap partition. The tool then calculates the distance to the centroid for each 4 kB block of the file. At the same time it counts and filters some JPEG-specific markers, which we use to lower the number of false positives from other file types also having more or less uniformly distributed byte frequencies, for example .zip files and encrypted files.

The marker filters of the type_mapper tool are set to only accept the occurrence of Restart (RST) markers, 0xffdx, the End-Of-Image (EOI) marker, 0xffd9, and the marker

0xff00 in the data. Disk clusters where the RST markers do not loop through $x = \mod 8$ in consecutive order are discarded. Likewise we filter out clusters where there are more than one EOI marker present. There is also a filter for 0xff00 set to require at least one such marker. The threshold is based on tests where we found the mean frequency of that marker to be 2365, with a standard deviation of 1686.1 for a 1 MB block of JPEG data. This corresponds to a mean of 9.2 and a standard deviation of 6.6 for a 4 kB block of data.

The *hexgrep* tool was created because we could not find an easy way to *grep* for hexadecimal data in a file. This would be useful when looking for JPEG Start-Of-Image (SOI) markers or other header related information. The tool could also be used for counting the number of blocks categorized as probable JPEG data in a type_mapper map.

3 Test Cases

Two evaluations of the Oscar method and toolbox were performed using two different setups. The first evaluation was done using a file containing several different types of data. The second evaluation was designed as a more practical experiment where two dump files, representing the RAM and swap partition of a computer, were searched for fragments of two JPEG files.

Although the detection rate of the Oscar method is important, the false positives rate is more important because of the intended use of the method. Therefore the test cases were focused on measuring the number of false positives generated in the different settings.

3.1 Evaluation Using Prepared File

The evaluation file for the first experiment was created by concatenating 53 files of 49 file types (see Table 1) into one. Three JPEG files were included to enable testing of the detection rate. Two of the pictures were taken by different digital cameras and one was produced by a scanner from an ordinary paper photograph. One of the digital camera JPEG files was previously unknown to the type_mapper tool. The scanner JPEG file contained RST markers (see Sect. 2.2) to test the filtering of markers.

Table 1. File types used for the evaluation.

.ACM	.ENU	.NLD	.SVE	.cpl	.html	.sys
.BMP	.INF	.NLS	.TLB	.dll	.jar	.tgz
.COM	.INI	.OCX	.UCE	.doc	.mp3	.tsp
.CPI	.ITA	.OLB	.ax	.dsk	.pdf	.txt
.DAT	.JPG	.RLL	.bin	.exe	.png	.vbs
.DEU	.MSC	.ROM	.bz2	.gif	.ps	.xls
.DRV	.MSI	.SQL	.chm	.gpg	.rpm	.zip

The files to be included in the evaluation were chosen based on their file size and picked subjectively from a laptop running Windows XP, SP 2. We wanted as large files as possible to decrease the impact of the file headers, because a header's structure of often differs significantly from the data part's structure. The total size of the evaluation file was 70.4 MB, giving an average size of the included files of 1,3 MB.

3.2 Evaluation of RAM and Swap Dump

The second experiment was based on the scenario described in Sect. 1, but changed from searching a hard disk to searching RAM and the swap partition of a computer. The evaluation was designed to measure the number of false positives generated when using the Oscar toolbox in a practical setting, and to see whether it was possible to apply the toolbox to such a scenario.

The test was performed by storing and viewing two JPEG pictures in a computer running Knoppix 3.9. Both pictures were taken from the set used to construct the centroid: one was from a digital camera and one was a scanned picture. The two dump files were created and searched for remnants of the two JPEG files. It is worth mentioning is that the swap file Knoppix used was also used by a Gentoo Linux installation on that computer, therefore the file could have contained data fragments from previous sessions.

The tools were applied in the following way:

1. the type_mapper tool was applied to the RAM and swap dump files containing fragments of the pictures,
2. the disk clusters marked as "probably JPEG data" were extracted, and
3. the hexgrep tool was used to find any matches between the extracted fragments and the two JPEG pictures used.

The work-flow described above could be used by a forensic examiner looking for pictures of child pornography. We used pictures portraying a pile of wood and a wedding, not child pornography, but since the algorithms of the Oscar method work on a binary level, the subject of the pictures does not affect the result of the evaluation.

4 Result

In this section the results of the two evaluations are presented. We also discuss whether it is possible to regard a match between a 4 kB data fragment and a complete picture as evidence of the complete picture once having existed on the storage media where the fragment was found.

4.1 Evaluation Using Prepared File

The evaluation file contained 17608 full 4 kB blocks, of which 476 should be categorized as JPEG. The algorithm categorized 466 of them correctly and produced 1 false positive (see Table 2).

Table 2. Result of the evaluation using the prepared file.

	# of 4 kB blocks	% of category
True pos.	466	97.90
True neg.	17131	99.99
False pos.	1	0.01
False neg.	10	2.10

All false negatives were generated by the digital camera picture included in the centroid. The reason for the false negatives was that some parts of the picture lacked 0xff00 codes. We repeated the experiment with the 0xff00 filter disabled and got a 100% detection rate, but 20 false positives.

The false positive was generated by the *win32.sys* binary, which happened to have an almost perfectly symmetrical fragment at one place, giving a distance measure of 764.4. The mean distance of the JPEG data in our tests was approximately 1500. In this case using longer n-grams might have helped, but the current level of 1 false positive out of 17132 possible might be acceptable.

4.2 Evaluation of RAM and Swap Dump

There were 119 disk blocks categorized as possibly containing JPEG data in the RAM dump. When comparing them to the two JPEG files used, all matched. There were no false positives; the reason for this is not completely clear to us because we did not have full control of the contents of the RAM and swap, but we know that there were no other pictures present.

In the swap dump there were 126 fragments categorized as possibly JPEG data. All of them matched the pictures used, and consequently, there were no false positives in this dump either.

The fact that there were no false positives in this test, but a large number of true positives, is encouraging. The results also show that technically the method described in Sect. 3.2 for finding traces of child pornography is working.

There is, however, a legal aspect of the problem: evidence used in a trial must be trustworthy. Thus, if a certain number of fragments belonging to the same known child pornographic picture is found on a hard disk, the disk has probably contained that specific picture. What must be determined, then, is the number of matching fragments necessary to say with certainty that the picture once existed on the disk. While a 100% match cannot be achieved unless the entire image can be retrieved from the disk, it is possible to estimate the possibility that a partial match is equivalent to a complete match.

To find an approximate value for the probability of a partial match we have to start by figuring out the number of pixels stored in a disk cluster sized data fragment, i.e. the compression rate of JPEG. An estimation of the compression rate, c, can be made using the equation $c = \frac{3*x*y}{s}$, where 3 comes from the number of bytes used to encode RGB colour, x and y gives the size of the image in pixels, and s is the size of the data part of the JPEG file in bytes. The compression rate depends on the diversity of the pixel

values, but for normal pictures, such as those in Fig. 2, we found the compression rate to be in the range of 8 to 15.

Figure 2. The four pictures that shared the same 12 to 22 first bytes in the data part of the files.

At that compression rate, 4096 bytes of JPEG data approximately correspond to an area of between 10900 (104^2) and 20500 (143^2) pixels. Large areas of the same colour and saturation are compressed more by the JPEG algorithm than highly detailed areas where colour and saturation change rapidly. In other words, the entropy of a group of bytes is almost the same regardless of what part of the picture they represent. Therefore the larger part of the bytes of a fragment is made up of details of the picture. If two fragments are equal, it is likely that their details are equal too, and if so we can in reality draw the conclusion that the fragments came from the same source.

One interesting thing to notice is that of the pictures shown in Fig. 2, skogPICT0049.jpg and skogPICT0063.jpg have the first 22 bytes of the data part in common, and f23.jpg and skogPICT0026.jpg share their first 12 bytes. The file skogPICT0026.jpg also shares its first 14 bytes with skogPICT0049.jpg and skogPICT0063.jpg.

5 Related Work

Wang and Stolfo have written a paper [1] presenting PAYL, an intrusion detection system built on a method related to the Oscar method. Another paper [16] by Li, Wang, Stolfo and Herzog apply the fundamentals of PAYL to create so called *fileprints*, which are used for file type identification. Similarly to our method PAYL and the fileprints use a centroid modelling the mean and standard deviation of individual bytes. To speed up the file type identification Li, Wang, Stolfo and Herzog experiment with truncating the data and only using a certain number of bytes from the beginning of each data block.

To detect anomalies both PAYL and the fileprints make use of what is called a *simplified Mahalanobi distance* to measure the similarity between a sample and a centroid. The simplified Mahalanobi distance is described by Equation (2), where two vectors, called x and y, represent the sample and the centroid respectively:

$$d(\boldsymbol{x}, \boldsymbol{y}) = \sum_{i=0}^{n-1} \left(|x_i - y_i| / (\sigma_i + \alpha) \right) \quad . \tag{2}$$

The standard deviation of byte i is represented as σ_i, the value of α is used as a smoothing factor to avoid division by zero when $\sigma_i = 0$. As can be seen this is similar to a linear-based distance metric between the sample and a model, weighted by the standard deviation of the byte frequency distribution.

In Sect. 2.1 we described the problem of a linear-based method giving the same result regardless of whether there is one large byte frequency deviation, or several small deviations. Since Equation (2) is linear-based its results depend heavily on the standard deviations, σ_i, of the bytes frequencies. If the standard deviation of the σ_i:s is low, the results of the equation will be almost equal, regardless of the distribution of the deviations of the sample vectors and the centroid. Our metric, where each byte frequency difference is squared, can in a better way handle byte frequency distributions where the standard deviation of the σ_i:s is low, because it favours many small differences over a few larger ones.

When Li, Wang, Stolfo and Herzog use truncated data for creating fileprints it makes them dependent on header data to be able to categorize files, which in turn requires the files not to be fragmented and preferably the file system to be intact. The Oscar method uses only the structure of a data fragment to identify its file type and therefore can work regardless of the state of the file system or degree of fragmentation.

Another method related to the Oscar method was presented by McDaniel and Heydari [17] in 2003. They use their method for determining the type of contents of different files, but use complete files or the header and trailer parts of files to create the fingerprints used as models. In order to incorporate byte order, to a certain extent, into the fingerprints the authors use what they call *byte frequency cross-correlation*. Most of the paper describes ways to improve the statistical features of the algorithms, enabling them to better handle discrepancies in the byte frequency of single files during creation of the fingerprints.

The main differences between the Oscar method and the method described in [17] is that the latter depends on the header and footer parts of files to create fingerprints.

Two of the more popular open source tools for forensic file carving are *lazarus* and the *Sorter* tool. The first one is part of the Coroner's Toolkit (TCT) [4] and the second tool is included in the Sleuth Kit [3]. Both of them depend on the Unix *file* [18] utility, which uses header information to categorize different file types. Consequently these two tools depend on header information for file type identification, which the Oscar method does not.

6 Conclusion and Future Work

We propose a method, called Oscar, for categorizing binary data on hard disks, in memory dumps, and on swap partitions. The method is based on creating models of different data types and then comparing unknown samples to the models. The foundation of a model is the mean and standard deviation of each byte, given by the byte frequency distribution of a specific data type.

The proposed method has been shown to work well in some smaller evaluations using JPEG files. The Oscar toolbox has been used to identify fragments of known pictures in RAM and memory dumps. The same methodology for using the tools can be put to immediate use by the police in the hunt for child pornography.

Future work will include further development of the tools, providing better integration between them as well as extended functionality. The number of centroids of different file types will be increased and eventually included in the type_mapper tool. Another example of planned improvements are the possibility to extract fragments directly to file when generating a map of the input data.

The idea of incorporating ordering in the centroid is interesting and thus the use of 2-grams and larger n-grams must be investigated further, as well as other ways of creating the centroid and measure the similarity of samples and centroids. We will also look further into finding a good method to recreate files from fragments found using the Oscar method.

References

1. Wang, K., Stolfo, S.: Anomalous payload-based network intrusion detection. In E. Jonsson el al., ed.: Recent Advances in Intrusion Detection 2004. Volume 3224 of LNCS., Springer-Verlag (2004) 203–222
2. CONVAR Deutschland: Pc inspector. (`http://www.pcinspector.de/file_recovery/uk/welcome.htm`) accessed 2005-10-31.
3. Carrier, B.: The Sleuth Kit. (`http://www.sleuthkit.org/sleuthkit/index.php`) accessed 2005-10-25.
4. Farmer, D., Venema, W.: The Coroner's Toolkit (TCT). (`http://www.porcupine.org/forensics/tct.html`) accessed 2005-10-25.
5. Guidance Software: Encase forensic. (`http://www.guidancesoftware.com/products/ef_index.asp`) accessed 2005-10-31.
6. QueTek Consulting Corporation: File scavenger. (`http://www.quetek.com/prod02.htm`) accessed 2005-10-31.
7. iolo technologies: Search and recover. (`http://www.iolo.com/sr/3/`) accessed 2005-10-31.
8. Brand, N.: Frozentech's livecd list. (`http://www.frozentech.com/content/livecd.php`) accessed 2005-10-28.
9. grugq: Defeating forensic analysis on unix. Phrack **11**(59) (2002) `www.phrack.org/show.php?p=59&a=6`, last visited 2004-11-19.
10. grugq: Remote exec. Phrack **11**(62) (2004) `www.phrack.org/show.php?p=62&a=8`, last visited 2004-11-19.
11. Pluf, Ripe: Advanced antiforensics : SELF. Phrack **11**(63) (2005) `http://www.phrack.org/show.php?p=63&a=11`, accessed 2005-11-03.

12. Rhodin, S.: Forensic engineer, Swedish National Laboratory of Forensic Science (SKL), IT Group. (several telephone contacts during October and November 2005)
13. Ericson, P.: Detective Sergeant, National Criminal Investigation Department (RKP), IT Crime Squad, IT Forensic Group. (telephone interview 2005-10-31)
14. Damashek, M.: Gauging similarity with n-grams: Language-independent categorization of text. Science **267**(5199) (1995) 843–848
15. Eaton, J.: Octave. (`http://www.octave.org/`)
16. Li, W.J., Wang, K., Stolfo, S., Herzog, B.: Fileprints: Identifying file types by n-gram analysis. In: Proceedings from the sixth IEEE Sytems, Man and Cybernetics Information Assurance Workshop. (2005) 64–71
17. McDaniel, M., Heydari, M.: Content based file type detection algorithms. In: HICSS '03: Proceedings of the 36th Annual Hawaii International Conference on System Sciences (HICSS'03) - Track 9, Washington, DC, USA, IEEE Computer Society (2003) 332.1
18. Darwin, I.: file(1). (`http://www.die.net/doc/linux/man/man1/file.1.html`) accessed 2005-10-25.