OleDetection—Forensics and Anti-Forensics of Covert Channels in OLE2-Formatted Documents

Robert F. Erbacher, Jason Daniels, and Steena Montiero

***Abstract*–New and improved data hiding techniques pose a problem for forensic analysts investigating a computer crime incident. Computer criminals are able to hide information using covert channels available in commonly used document formats, thereby hindering a forensic investigator from acquiring possibly important and convicting evidence. In this paper, we focus on detecting, for use in forensic analysis, the use of covert channels of communication in the unused or dead space regions in the Object Linking and Embedding 2 (OLE2) specification used primarily by Microsoft's Office Suite. The OleDetection algorithm [19] presented in this paper is focused on detecting the use of these covert channels using a three-step process comprising the detection of dead regions in a document, the extraction of binary data and the generation of appropriate statistics using kurtosis and byte-frequency distribution, and the comparison of the calculated statistics with threshold values, which determines whether or not the document contains hidden data. This algorithm extends the work done by the StegOle algorithm [3]. Our experimental results show that the OleDetection algorithm can correctly identify 99.97 percent of documents with previous covert channel techniques with a false positive rate of only 0.65 percent. In addition, we present an anti-forensic technique wherein OLE2 documents can be modified to hide data with greater detection avoidance characteristics [19]; thus reducing the accuracy of the current OleDetection implementation.**

***Index Terms*–Forensics, Anti-Forensics, Covert Channels, OLE2.**

I. INTRODUCTION

Digital forensics is the science of collecting, discovering, and preserving digital data for use in court. In other words, computer forensics can be considered a series of analytic and investigative methods applied to evidence that is magnetically stored or encoded that must be identified, collected, examined, and preserved [4]. Acquiring digital data from a target system so that it can be used in an investigation is not an easy task. This is in part due to the ability for data to be hidden, such as with steganography or covert channels. Considerable research has focused on detecting *stenographic* algorithms and finding hidden data files in large disk spaces [5][7][8][15].

Sophisticated users can aggressively manipulate documents to hide information and then use these documents to covertly communicate sensitive information and avoid detection. Our OleDetection algorithm is designed to detect information hidden in the unused spaces of documents that use the OLE2 specification. Therefore, although no visual clues exist, that could indicate the presence of hidden data; there still exist signs that can be used to statistically detect modified documents. Cantrell et al. [2] and Byers [1] explored the ability to hide information in Microsoft Office documents and identified methods to hide information in the document metadata. Although this method works, this type of data hiding is easily detected. Research in this area points out that large sections of documents are uniformly 0x00 or 0xFF; however, when these sections are tampered with, the document becomes corrupt and cannot be reopened using the native MS Office application. Contrell et al. [2] specifically point out that nearly all types of documents are vulnerable to the insertion of data past the end of the EOF marker, in which case the documents can still be re-opened using the native MS Office application. This type of data appendation is being examined by Erbacher et al. [5].

Tsung-Yuan, et al., [11] introduced a steganographic method of hiding data in Microsoft documents by using the tracking mechanism available in Microsoft Word. By using a synonym dictionary with the track changes feature, the document can be changed to appear as though it had simply been through several editorial revisions when in reality; the tracked changes are hidden data.

Existing detection tools concentrate on finding hidden information in file systems [3] images [4], video files [16], executables [5] databases [19], and networks [17]. Currently, no known techniques exist for detecting covert channel use in common office documents.

Thus far, there has been little research carried out with regard to creating covert channels in MS Office documents, namely by Cantrell et al. [2] and Byers [1]. However, the work in [3][4] resulted in corrupt files that could not be opened by native applications. The OleDetection algorithm is able to encrypt and embed, and then extract and decrypt any message in an OLE2-formatted document. This algorithm does not change the size of the modified document or its appearance. Furthermore, the integrity of the document is not compromised, thereby allowing the file to be opened by its native application. This research is in essence an extension of that presented in Castiglione et al. [3].

R.F. Erbacher is with the Computer Science Department at Utah State University, Logan, UT 84322, USA (phone: 435-0797-3291; fax: 435-797-3265; e-mail: Robert.Erbacher@usu.edu).

J. Daniels was an M.S. in Computer Science at Utah State University, Logan, UT 84322, USA (e-mail: jason814@gmail.com).

S. Montiero is a PhD student in the Computer Science Department at Utah State University, Logan, UT 84322, USA (e-mail: steena.m@aggiemail.usu.edu).

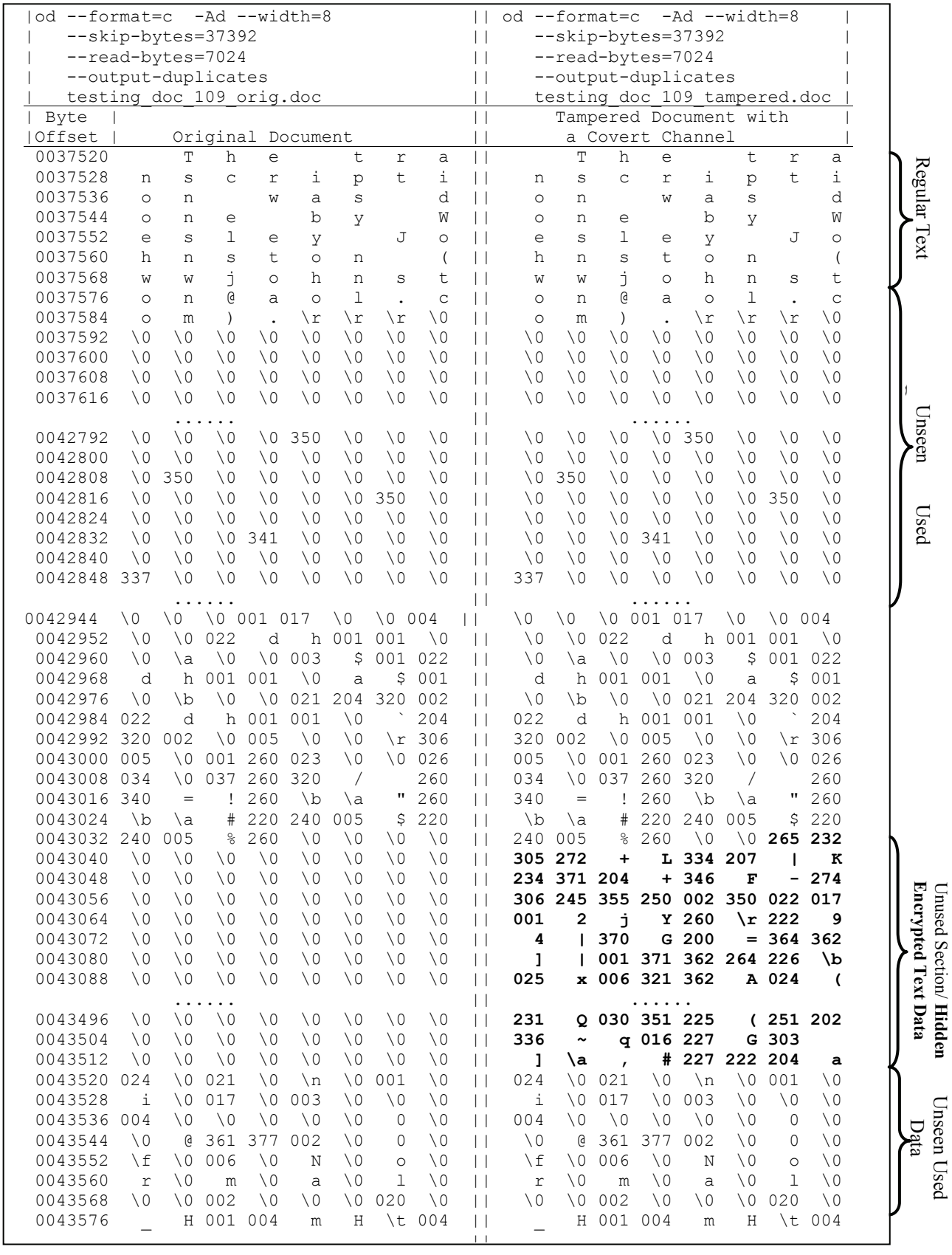

Figure 1: Character data dump from original Word document (left) and altered Word document (right).

The authors presented *StegOlè,* a method of hiding data in the unused sectors of a Microsoft compound document file, otherwise known as the OLE2 Compound Format. This research extends the work by Castiglione et al. by presenting a detection algorithm along with the base algorithm as well as more sophisticated anti-forensics that can be used to make detection more difficult.

II. OLE2 BACKGROUND

Microsoft Office Suite uses the Object Linking and Embedding 2 (OLE2) as its Compound Document format. The power of this format lies in its ability to have multiple document types within a single file, thereby allowing a single application to embed the contents of different applications. The OLE2 format makes use of the concept of sectors and streams. Thus, an OLE2 document has multiple streams representing the different objects that are embedded in that document. Each stream, and eventually the file as a whole, is partitioned into 512- byte sectors. In other words, regardless of the actual data size, the file size is always a factor of 512.

A master table identifies each of the streams and the starting sector. In order to manage the mapping of sectors to streams, a summary information stream maintains a block allocation table (BAT). This BAT is essentially an array of 4-byte integers that represents each sector in the document. Often, a block is unused and is labeled with a -1; other valid entries maintain a linked list of sectors. Each entry contains the next sector number in the stream. The terminating sector can be identified by the presence of a -2 entry in the BAT.

The creation of covert channels in the OLE2 format takes advantage of the properties of this format, namely, the fixed sector size and the unused sectors. Often, a particular stream does not terminate exactly at the 512-byte boundary. In such cases, the space between the end of the stream and the sector can safely be overwritten. An unused sector is considered as dead space in the BAT, and that particular sector can be overwritten entirely with no fear of overwriting valid data, figure 1.

III. A STATISTICAL APPROACH FOR DETECTING COVERT CHANNELS IN FILES

Research has shown that unknown files can be identified successfully using statistical analysis. This statistical analysis has been used to analyze data based on either the file headers or the data within a file. McDaniel, et al., [10] achieved a detection rate of 95.6 percent using byte frequency analysis and byte frequency correlation combined using file header/trailer techniques. Kolter et al. [9] provide a method of detecting unknown malicious code in executables using machine learning with boosted decision trees. By using 500 n-grams or data points from the byte code, they were able to gain a 98 percent detection rate and a false positive detection rate of 0.5 percent. Karresand, et al., [8] showed that a byte frequency distribution with a sliding window and its derivative could be effective in uniquely identifying different types of documents. Erbacher et al. [5] showed that data types contained within a file could be potentially identified using kurtosis, byte averages, standard deviations, and standard deviation averages. This current work builds on these previous works using statistical analysis to identify covert channels that have been inserted into OLE2 formatted documents.

Using Statistical Algorithms for Detecting Covert Channels in OLE2-formatted Documents

With regard to OLE2 documents, covert channels are not actually encoded as part of the existing data; rather, they are hidden in the unused portions of the document. This method of hiding data coupled with the fact that OLE2-formatted documents are guaranteed to be split into 512-byte sections allows the use of a sliding window statistical analysis to locate any stegomedia. Based on previous research it was clear that a statistical analysis would be able to identify files that were modified to hide data. In this research, two statistical analysis techniques have been used to analyze each window, namely, byte frequency distribution (BFD) and kurtosis, both of which have been used successfully in the past for file type identification by Kerrasand [8] and Erbacher [5].

A. Kurtosis

Often used to identify unknown data, kurtosis is a statistical algorithm used to show the peakedness of a data set. Higher kurtosis values indicate large swings in the data, as opposed to more consistent data that typically have lower kurtosis values. It is calculated using the following formula:

$$K_j = N * \frac{\sum_{i=1}^{N} (X_i - \widetilde{X}_j)^4}{(\sum_{i=1}^{N} (X_i - \widetilde{X}_j)^2)^2}$$

Byte Frequency Distribution

Byte frequency distribution (BFD) is the second statistic used to detect the existence of convert channel data in a document. First presented by Karresand, et al., [8] BFD is a method of creating a unique numerical representation of a distribution of a set of bytes. This statistic is the distribution of the number of times each byte value is encountered in a set of data. The result is a distribution with 2^8 or 256 entries, the maximum value for a byte. Averaging multiple distributions of the same type of data creates a mean distribution and corresponding standard deviation distribution. The combination of the distribution of mean values and standard deviations define what is called a centroid, or a fingerprint that is theoretically unique for that particular type of data.

Given a centroid and any particular BFD, the relatedness between the two can be calculated by determining the distance between the two with a quadratic formula. This quadratic formula is given by

$$\sum_{i=0}^{n-1} (s_i - c_i)^2 / (\sigma_i + \alpha)$$

where, c_i is the mean of the Centroid; s_i , the window byte count; $sigma_i$, the corresponding, standard deviation; and alpha, a smoothing factor.

Both of these statistical algorithms were used jointly on each sliding window to create a unique numerical representation of the data. These representations are termed WindowPrints and are similar to fingerprints in that they have multiple points of reference for determining uniqueness. Comparing the distance between the pre-calculated WindowPrints for *StegOlè* data with those from other documents being analyzed determines if the said documents contain covert channels.

IV. STATISTICAL PROPERTIES OF OLE2-FORMATTED DOCUMENTS

Common office documents created by Microsoft Office, e.g., Word, Excel, and PowerPoint, are implementations of the OLE2 specification. The data used for the experiments and analysis in this paper has been primarily obtained from MS Word documents. Since the principles and techniques identified here are applicable to all documents that use the OLE2 specification, the results and conclusions from this research will be applicable to the general case. In addition, representative datasets for Excel and PowerPoint documents have been used in a smaller set of experiments for validation purposes.

Each document gathered for these experiments was collected in an adhoc fashion from random websites. The documents vary in size and content. Ultimately, 293 MS Word documents were acquired, ranging in size from 20.5 kilobytes to 4,858 kilobytes. The contents of these documents ranged from primarily text to forms with several tables, while others contained images of various sizes. To determine the impact caused by specific data, two additional Word documents were created and tested separately from the larger dataset, one containing only images and the other only embedded OLE objects. The PowerPoint dataset contains 99 documents ranging in content and size from 26.5 to 31,148 kilobytes. The Excel data set contains 109 files ranging in size from 10.0 to 8,108 kilobytes.

OleDetection experimental results

The experiments described here were run against the full dataset of 293 documents, of which ~50 percent were randomly modified using *StegOlè* to hide a message. The developed detection tool uses the kurtosis statistic in combination with the BFD to identify documents that have been modified to contain a covert channel. The following sections describe the experimental results obtained using this tool across the data set.

Data set selection

The OleDetection algorithm can identify dead space sectors in an OLE2 document and generate corresponding statistics. The generated statistics are then compared against a threshold for values obtained using the kurtosis and the BFD statistics. If the threshold is met, the document is considered to contain hidden data.

Several iterations of experiments were performed to determine the best threshold. Each threshold was tested against 11 different combinations of the 293 Word files, i.e. 11 different data sets. Each data set hid a different amount of data in its covert channel. For example, data set one had 148 randomly picked documents modified by *StegOlè* to hide two bytes of data. In addition, each subsequent data set was modified to hide increasingly more data, Table 1.

Initial exploration of threshold values

Based on data from a trial experimental run, the first round of tests used thresholds set to a single standard deviation from the average. The kurtosis value was set to 1.81 and the BFD distance was set to 490. The results showed on average a 69.2 percent detection rate, well short of the desired mark. seta change in the thresholds to the max value encountered, a kurtosis value of 1.87, and BFD distance of 547 improved performance with an average discovery of 73.5 percent; however, this still was not sufficiently high. The following four sections describe my experiments in achieving the desired threshold.

Experiment 1

Experiment 1 required that a WindowPrint have a kurtosis value of less than 1.99 and a BFD distance of less than 900 in at least 50 percent of the windows. The results were promising yielding a majority of true-positive detection rates around 97 percent and no false positives.

Experiment 2

Experiment 2 varied from Experiment 1 only in the fact that if a single window had kurtosis values of less than 1.99 and BFD distances of less than 900 it would be considered to contain hidden data. This effectively qualified most of the documents that had been missed in the first experiment. The slight change caused the true positive detection rate jumped 99.7 percent with a false positive rate of 0.62 percent.

Experiment 3

Experiment 3 was an attempt to get a 100 percent true positive detection while still keeping the false positive rate low. The thresholds were increased to 2.2 for the kurtosis and less than 1400 for the BFD distance. The increased threshold provided the best result with an average detection rate of 99.97 percent, while the false positive rate only increased to 0.65 percent, figure 2.

Experiment 4

Experiment 4 used the same threshold values as Experiment 3—a kurtosis threshold of less than 2.2 and a BFD distance of less than 1400. In this experiment, a file was considered to be modified if a single WindowPrint met either of the two thresholds. This resulted in a 100 percent true positive identification but also an unacceptably high false positive rate of 65.18 percent.

Table 1: Details for each test data set.

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**
# Bytes Hidden	2	16	32	64	128	256	512	1024	2048	4096	Assorted
# Tampered Files	148	155	148	144	131	151	133	143	137	140	147

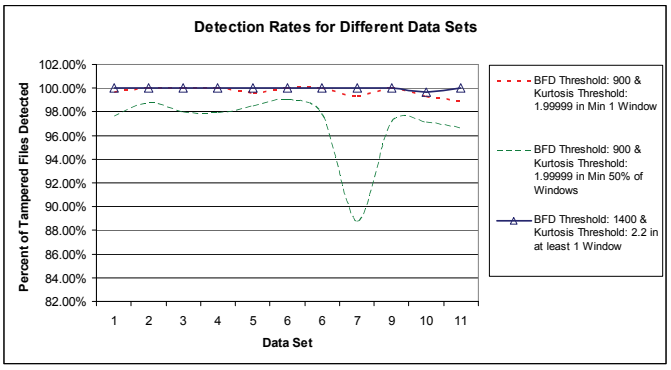

Figure 2: Graph of detection rates for the different datasets.

Table 2: Summary of experimental thresholds.

	Kurtosis Threshold	Predicate	BFD Threshold	Required Window Count
Experiment 1	< 1.99	AND	< 900	> 50%
Experiment 2	< 1.99	AND	< 900	At least 1 Window
Experiment 3	< 2.2	AND	< 1400	At least 1 Window
Experiment 4	< 2.2	OR	< 1400	At least 1 Window

This experiment shows how the combination of statistics contributes to the success of this detection algorithm. Taken individually, they do not provide sufficient data to accurately identify the difference between a modified and an unmodified file. A summary of the thresholds used for each of the experiments is presented in Table 2. The summary of the experimental results is presented in Table 3.

Table 3: Average accuracy of each experiment.

	Exp. 1	Exp. 2	Exp. 3	Exp. 4
Avg. True Positive	97.00%	99.71%	99.97%	100.00%
Avg. False Negative	0.00%	0.62%	0.65%	22.99%
Avg. Algorithm Value	93.82	98.81	99.29	77.01

Experimental Validation using Excel and PowerPoint Documents

The encoding and detection scheme works for Excel and PowerPoint documents as well. Files were randomly selected from each of the corresponding sets of files and modified to hide data of assorted sizes. OleDetection was executed on the set of Excel and PowerPoint documents using the thresholds used in Experiment 3. For the Excel dataset, of the 109 documents, 30 were modified, and OleDetection correctly identified 100 percent of the modified documents while yielding five false positives. In the PowerPoint dataset, of the 99 documents, 42 were modified with hidden data. Of those, 38 or 90.5% were correctly identified with a false positive rate of 9.5%.

The experiment using this dataset highlights two points: first, OleDetection works for all OLE2 document types; second, the threshold can be tweaked for each document type for better results.

Experiment Summary

The value of any detection algorithm lies in its ability to accurately identify whether files do or do not contain steganographic data. A point of comparison is the detection algorithm developed by Kolter, et al. [9], which was able to achieve detection rates of unknown malicious executables at around 98 percent with a false positive rate of 0.5 percent. Using this as the benchmark, Experiment 3 yields excellent results with a 99.97 percent positive detection rate and a 0.65 percent false positive rate. As stated above, even though the results for the Excel and PowerPoint datasets were a bit lower, a few modifications to the thresholds should result in a higher detection rate.

V. OLEDETECTION IMPLEMENTATION DETAILS

The OleDetection application is a console-based program that can identify OLE2-formatted documents that have been tampered with to contain hidden data. The implementation can examine a directory full of documents or, if needed, a single file. The approach used in OleDetection has a three-step process for identifying OLE2 documents with hidden data. Step 1 subjects the OLE2 document to OleSteganography and determines dead space regions. Step 2 extracts the data binary data from those regions and generates the statistics. Finally, Step 3 compares the calculated statistics against the predetermined threshold; if that threshold is met, the document is considered to contain hidden data.

OleSteganography – Finding the Dead Space Regions

OleSteganography is a library that has been developed as part of this research with the goal of locating the dead space regions in OLE2-formatted documents. It is built on top of POIFS, an open-source Java implementation of the OLE2 file system specification [14]. It should be noted that OLE2 is a compound file format; or rather, a file containing other documents. The ability of a document to contain other data makes it similar to a file system. The contents and the different streams of data contained in the document are determined by reading the BAT starting from an offset 0x4C from the start of the document. The dead space regions in the document are identified and their offsets are noted.

OleDetection—Extracting Data and Calculating Statistics

Once the dead space regions of the document are identified, the data needs to be extracted and the statistics—kurtosis and BFD—have to be calculated. It is in this phase that the WindowPrint, which encapsulates the data and the statistics for a dead space region, is generated.

Threshold Comparison

The next step is the process of determining how closely the set of WindowPrints came to meeting the threshold. The statistics of each WindowPrint are compared against the individual statistic thresholds. If the set of statistics are determined to meet the threshold, a running counter is increased. The result is a count of WindowPrints representing a covert channel in the document. As the final step of determining whether the document as a whole has been modified, this running total is compared against the required number of WindowPrints specified for a match to occur.

Using OleDetection

Figure 3 shows the running of the application for detecting the presence of a covert channel in a single file, and Figure 4 shows the scanning of an entire directory for tampered files.

```
C:\Tools\jdk\bin\java oledetection.OleDetection -f testDoc.doc
HIDDEN DATA DETECTED:        testDoc.doc
```

Figure 3: Using OleDetection to test an individual file for a covert channel.

```
C:\Tools\jdk\bin\java  oledetection.OleDetection -d ".\Document Repository\\"
Detecting tampered files in: .\Document Repository\2 Bytes
untampered file     :        .\Document Repository\2 Bytes\testing_doc_0.doc
HIDDEN DATA DETECTED:        .\Document Repository\2 Bytes\testing_doc_1.doc
untampered file     :        .\Document Repository\2 Bytes\testing_doc_10.doc
HIDDEN DATA DETECTED:        .\Document Repository\2 Bytes\testing_doc_100.doc
untampered file     :        .\Document Repository\2 Bytes\testing_doc_101.doc
HIDDEN DATA DETECTED:        .\Document Repository\2 Bytes\testing_doc_102.doc
HIDDEN DATA DETECTED:        .\Document Repository\2 Bytes\testing_doc_103.doc
HIDDEN DATA DETECTED:        .\Document Repository\2 Bytes\testing_doc_104.doc
untampered file     :        .\Document Repository\2 Bytes\testing_doc_105.doc
untampered file     :        .\Document Repository\2 Bytes\testing_doc_106.doc
untampered file     :        .\Document Repository\2 Bytes\testing_doc_107.doc
HIDDEN DATA DETECTED:        .\Document Repository\2 Bytes\testing_doc_108.doc
HIDDEN DATA DETECTED:        .\Document Repository\2 Bytes\testing_doc_109.doc
HIDDEN DATA DETECTED:        .\Document Repository\2 Bytes\testing_doc_11.doc
HIDDEN DATA DETECTED:        .\Document Repository\2 Bytes\testing_doc_110.doc
HIDDEN DATA DETECTED:        .\Document Repository\2 Bytes\testing_doc_111.doc
.............
.............
untampered file :     .\Document Repository\2 Bytes\testing_doc_93.doc
untampered file :     .\Document Repository\2 Bytes\testing_doc_94.doc
HIDDEN DATA DETECTED:        .\Document Repository\2 Bytes\testing_doc_95.doc
HIDDEN DATA DETECTED:        .\Document Repository\2 Bytes\testing_doc_96.doc
untampered file :     .\Document Repository\2 Bytes\testing_doc_97.doc
untampered file :     .\Document Repository\2 Bytes\testing_doc_98.doc
untampered file :     .\Document Repository\2 Bytes\testing_doc_99.doc


The list of tampered files are:
testing_doc_1.doc
testing_doc_100.doc
testing_doc_102.doc
testing_doc_103.doc
testing_doc_104.doc
testing_doc_108.doc
testing_doc_109.doc
testing_doc_11.doc
testing_doc_110.doc
testing_doc_111.doc
.....
testing_doc_95.doc
testing_doc_96.doc

. . . . . .
***** Summary *****
Files Examined   :  293
Error Count is   :  0
Detected Tampered file count :     148
.....................................................................
The undetected files   :
Number of Undetected Files :       0
.....................................................................
The incorrectly identified as being tampered with files:
Number of false positives  :       0
.....................................................................
Number of Known Tampered Files :   148
Number of Corre
ctly Dectected Files:       148
Number of Undetected Tampered Files:       0
Number of false positives  :       0
Correct Detection Accuracy :       1.0
```

Figure 4: Using OleDetection to test a directory of files for covert channels.

VI. MODIFYING OLE2 DOCUMENTS FROM AN ANTI-FORENSICS PERSPECTIVE

Now we will examine a method of encoding the covert channels in OLE2 documents such that the resultant document is statistically similar to the original parent document. This not only includes picking a single target statistical value and encoding the covert channel to match, but also the variable statistical values similar to those found in the document itself. By matching the statistical values in this way, distinguishing between documents that actually contain covert channels and those that do not becomes even more difficult

Exploring OLE2 Vulnerable Regions

Every OLE2 document has dead space regions that can be exploited to contain a covert channel. These regions have unique properties that differ from the rest of the OLE2 document, because the data has no value to the document itself. From the dataset of 293 unique documents described earlier, the dead space regions were extracted and their statistical properties were reviewed. As shown in Tables 4 and 5, the single most common data points are regions of only zeros, accounting for 39.56 percent of all the dead space regions. We focus on non-zero data regions due to only limited potentials for storing data in the zero data regions. Of those 220 regions, 98 generated unique kurtosis values (excluding outliers). Figure 5 shows the distribution of the 98 different kurtosis values, the majority of values being between 1 and 10 and between 20 and 30.

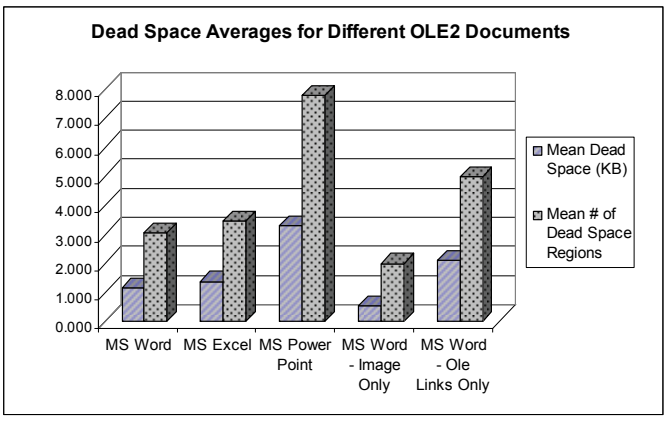

Figure 5: Dead space averages for different OLE2 documents. This table shows the number of individual dead space regions and the total size of those regions for different OLE2 document types.

Table 4: General kurtosis *properties* of dead space regions. Essentially, this table shows that a large number of the identified dead space regions contain only zeros that cannot be used to hide data. Even for the regions that do contain non-zero values only a subset provide unique kurtosis values and thus could be used to hide data.

	Total Population	# Generating Unique Kurtosis Values*
Dead Space Regions	365	98
Zero Count	220	0
Non-Zero Values	144	98

**Excludes Outliers*

Table 5: General kurtosis *values* of dead space regions. This table shows the range of kurtosis values found in the dead space regions. Deviations from these values are used for detection. Additionally, we will attempt to force inserted data to match the given ranges to avoid detection.

	Total Population	# Generating Unique Kurtosis Values*
Minimum	1.04	1.11
Maximum	508.94	126.01
Average	27.22	18.18
Std.	47.94	16.26
Median	21.69	20.63

**Excludes Outliers*

Experiments with Different Encodings of the Payload

The purpose of these next set of experiments was to determine the best combination of payload and filler values needed to blend modified documents with the existing documents. Several approaches were considered during the experimentation. The first was simply encoding the payload with a base 64 encoding, the second attempt was to insert data into only every third byte. In addition, several more experiments that modified the payload density until the payload was statistically similar to the surrounding data were carried out. In these experiments, the unit of measure that was used to indicate the payload density was bytes per 512 bytes region. For example, if the payload is 52, there are 52 bytes of actual data encoded in a 512 byte region, while the rest is filler data, either 0 or 255.

Each experiment explored how the kurtosis and BFD distance thresholds need to change in order to account for the new encoding and how detection results change. The various statistical measurements and thresholds used for each of the experiments are shown in Table 6. The false positive and false negative percentages at various values of the kurtosis and BFD statistics for each type of encoding are shown in Tables 7-13.

Experiment Conclusion

Diluting the payload density is a careful balance between detection and actually being able to provide enough bandwidth to still be usable. Limiting the payload to 32 bytes per 512 region is probably sufficient to avoid most detection attempts,

Table 6: Detection thresholds needed to detect each encoding.

Experiment Name	Average Kurtosis	Low Kurtosis	High Kurtosis	Average BFD Distance	Low BFD Distance	High BFD Distance	Threshold Kurtosis	BFD Threshold
StegOle (Reference)	1.80	1.75	1.87	406	296	547	2.2	1400
Base 64	4.30	2.99	7.80	2477	716	3601	8	4000
Every 3rd Byte	4.02	3.56	4.42	68776	2559	80559	5	80000
52 Byte Load w/255	13.31	10.65	15.99	305609	305354	306723	17	350000
52 Byte Load w/Zeroes	14.62	12.29	17.33	143378	132082	144583	18	145000
42 Byte Load w/Zeroes	18.88	17.48	21.41	151028	151024	151034	22	155000
32 Byte Load w/Zeroes	24.23	22.02	25.76	157621	157614	157627	26	160000
22 Byte Load w/Zeroes	34.66	30.51	37.92	164278	163672	164362	38	165000

taken a step further to alter regions between an "every third byte" encoding and a "32-byte payload" encoding would more closely blend in with the overall existing data patterns that already exist in the dead space regions.

Table 7: Detection results for Base 64 Encoding.

Used Threshold			
Kurtosis Threshold	**BFD Threshold**	**False Positive Percent**	**False Negative Percent**
2.2	**1400**	0.71%	97.37%
8	**4000**	0.71%	0.00%
5	**80000**	19.86%	0.00%
17	**350000**	33.33%	0.00%
18	**145000**	32.62%	0.00%
22	**155000**	46.81%	0.00%
26	**160000**	64.54%	0.00%
38	**165000**	89.36%	0.00%

Table 8: Detection results for every third byte encoding.

Used Threshold			
Kurtosis Threshold	**BFD Threshold**	**False Positive Percent**	**False Negative Percent**
2.2	**1400**	0.63%	97.04%
8	**4000**	1.27%	74.81%
5	**80000**	17.09%	4.44%
17	**350000**	31.65%	0.00%
18	**145000**	31.65%	0.00%
22	**155000**	43.67%	0.00%
26	**160000**	61.39%	0.00%
38	**165000**	90.51%	0.00%

Table 9: Detection results for 52 encoding with 255 filler values encoding.

Used Threshold			
Kurtosis Threshold	**BFD Threshold**	**False Positive Percent**	**False Negative Percent**
2.2	**1400**	0.68%	86.90%
8	**4000**	0.68%	81.38%
5	**80000**	13.51%	45.52%
17	**350000**	27.03%	0.00%
18	**145000**	27.03%	0.00%
22	**155000**	41.89%	0.00%
26	**160000**	58.78%	0.00%
38	**165000**	86.49%	0.00%

Table 10: Detection results for 52 byte encoding.

Used Threshold			
Kurtosis Threshold	**BFD Threshold**	**False Positive Percent**	**False Negative Percent**
2.2	**1400**	0.00%	95.73%
8	**4000**	0.00%	82.32%
5	**80000**	18.60%	73.17%
17	**350000**	31.01%	0.00%
18	**145000**	30.23%	0.00%
22	**155000**	41.86%	0.00%
26	**160000**	54.26%	0.00%
38	**165000**	86.05%	0.00%

Table 11: Detection results for 42-byte encoding.

Used Threshold			
Kurtosis Threshold	**BFD Threshold**	**False Positive Percent**	**False Negative Percent**
2.2	**1400**	1.50%	92.50%
8	**4000**	1.50%	79.38%
5	**80000**	12.03%	77.50%
17	**350000**	25.56%	0.63%
18	**145000**	25.56%	0.00%
22	**155000**	39.85%	0.00%
26	**160000**	54.14%	0.00%
38	**165000**	88.72%	0.00%

Table 12: Detection results for 32-byte encoding.

Used Threshold			
Kurtosis Threshold	**BFD Threshold**	**False Positive Percent**	**False Negative Percent**
2.2	**1400**	0.63%	90.37%
8	**4000**	1.27%	74.81%
5	**80000**	15.19%	74.81%
17	**350000**	27.22%	4.44%
18	**145000**	27.22%	0.00%
22	**155000**	41.77%	0.00%
26	**160000**	54.43%	0.00%
38	**165000**	87.34%	0.00%

Table 13: Detection results for 22-byte encoding.

Used Threshold			
Kurtosis Threshold	**BFD Threshold**	**False Positive Percent**	**False Negative Percent**
2.2	**1400**	0.68%	91.72%
8	**4000**	0.68%	80.00%
5	**80000**	14.86%	80.00%
17	**350000**	25.00%	12.41%
18	**145000**	23.65%	1.38%
22	**155000**	39.86%	0.69%
26	**160000**	56.76%	0.00%
38	**165000**	91.89%	0.00%

VII. DISCUSSION

The techniques discussed in this paper can be applied under a variety of different scenarios. The ubiquity and shear number of OLE2-formatted documents provides enormous opportunity for the distribution of covert data in this way. For instance, an attacker or insider could attempt to extrude/exfiltrate sensitive data. Many avenues of transmittal of the data could be protected. For instance, e-mail attachments can have the files checked. Alternatively, it would be feasible to simply zero out portions of OLE2 documents where data could be hidden. Such techniques wouldn't work for other transmittal mechanisms, such as encrypted file transfers.

The ability to zero out portions of OLE2 documents also assumes the file passes through a point of control; which often isn't the case. Terrorists for instance disseminate information through steganographic data on web sites. In such cases it would not be possible to zero out the data but analysts must rely on detection capabilities.

VIII. FUTURE WORK

This work shows how covert channel data can be detected in OLE2-formatted documents. This work can be expanded by finding other file types with similar vulnerabilities and applying the principles demonstrated here to those files.

Additionally, this paper shows that the kurtosis and BFD algorithms have valuable properties that can be used to detect the presence of hidden data; however, more work can be done to find other statistical metrics that can be used individually or in combination to detect covert channel data or malware. The thresholds for these statistics can also be modified slightly to more accurately identify covert channels hidden in Excel and PowerPoint documents. Another direction for future work would be the addition of this detection method to a runtime

environment such as a mail server to detect and handle documents containing covert channels.

The new encoding scheme presented serves as a new way that computer criminals might potentially use to hide important data or evidence and avoid detection. This highlights the need for continued work towards general covert channel data detection techniques. The reliance in this research on statistics and not on any specific encodings within the target documents is a step in that direction. Further, the thresholds proposed could be modified to likely handle a wide array of new anti-forensic strategies.

IX. CONCLUSION

OleDetection is an application that analyzes and detects covert channel data embedded into the dead or slack space of OLE2-formatted documents, namely those using *StegOle*. This detection technique should also be applicable to any technique that similarly stores data in OLE2 documents. Using kurtosis and byte frequency distributions (BFD) statistics, this application identified the sampled Word documents with covert channels with a 99.97 percent average true positive accuracy and a false positive rate of only 0.65 percent. These solid results show that a statistical approach can be used to detect hidden data with excellent accuracy.

The thresholds for OleDetection were adjusted with respect to Word documents. Moreover, the detection results of covert channels in Excel and PowerPoint documents were not quite as satisfactory with an average detection rate of 95 percent detection rate between the two and a false positive percentage of ~12 percent. It is believed that these results will improve with larger data sets and slight modifications to the thresholds in order to better account for the type of data contained in these additional OLE2 documents. The new encoding mechanism presented in this paper lowers the detection of covert channel data in OLE2-formatted documents. From an anti-forensic perspective, this can be considered as a new step in the arms race between the hiding and detection of covert data.

X. ACKNOWLEDGEMENTS

We appreciate the comments by the reviewers in helping improve the quality and clarity of this paper.

XI. REFERENCES

[1] Byers, S. Information leakage caused by hidden data in published documents. *IEEE Security and Privacy 2*, 2 (March/April 2004), 23-27.

[2] Cantrell, G. and Dampier, D. Experiments in hiding data inside the file structure of common office documents: a steganography application. In *Proceedings of the 2004 International Symposium on Information and Communication Technologies*, 2004, 146-151.

[3] Castiglione, A., De Santis, A., and Soriente, C. Taking advantages of a disadvantage: Digital forensics and steganography using document metadata. *J. of Systems Software 80*, 5 (2007), 750-764.

[4] Computer Forensics World. Computer forensics basics: Frequently asked questions, April 2008, www.computerforensicsworld.com.

[5] Erbacher, R. and Mulholland, J. Identification and localization of data types within large-scale file systems. In *Proceedings of the 2nd International Workshop on Systematic Approaches to Digital Forensic Engineering*, 2007, 55-70.

[6] Fairbanks, K., Lee, C., Xia, Y., and Owen, H. TimeKeeper: A metadata archiving method for honeypot forensics. In *IEEE SMC Information Assurance and Security Workshop*, June 2007, 114-118.

[7] Johnson, M. and Farid, H. Exposing digital forgeries through chromatic aberration. In *Proceeding of the 8th Workshop on Multimedia and Security*, 2006, 48-55.

[8] Karresand, M. and Shahmehri, N. File type identification of data fragments by their binary structure. In *Proceedings of the IEEE Information Assurance Workshop*, 2006, 140-147.

[9] Kolter, J. and Maloof, M. Learning to detect and classify malicious executables in the wild. *J. of Machine Learning Research 7*, (December 2006), 2721-2744.

[10] McDaniel, M. and Heydari, M. Content-based file type detection algorithms. In *Proceedings of the IEEE 36th Hawaii International Conference on System Sciences,* 2003, 332-331.

[11] Liu, T. and Tsai, W. A New Steganographic method for data hiding in Microsoft Word documents by a change tracking technique. *IEEE Trans. on Information Forensics and Security 7*, 1(March 2007), 24-30.

[12] Monroe, K., Bair, A., and Smith, J. Digital forensics file carving advances. In *Digital Forensics Research Workshop File Carving Challenge*, October 2006, http://www.korelogic.com/Resources/Projects/dfrws_challenge_2006/DFRWS_2006_File_Carving_Challenge.pdf.

[13] Pavlou, K. and Snodgrass, R. Forensic analysis of database tampering. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, 109-120.

[14] POI - POIFS - Java implementation of the OLE 2 Compound Document format. 2002. http://poi.apache.org/poifs/index.html. 2008.

[15] Voloshynovskiy, S., Herrigel, A., Rytsar, Y., and Pun, T. Stegowall: Blind statistical detection of hidden data. In *Proceedings of the International Society for Optical Engineering*, 2002, 57-68.

[16] Wang, W. and Farid, H. Exposing digital forgeries in video by detecting double MPEG compression. In *Proceeding of the 8th Workshop on Multimedia and Security*, 2006, 37-47.

[17] Wang, X., Li, Z., Xu, J., Reiter, M., Kil, C., and Choi, J. Packet vaccine: Black-box exploit detection and signature generation. In *Proceedings of the 13th ACM Conference on Computer and Communications Security*, 2006, 37-46.

[18] Harris, R. Arriving at an anti-forensics consensus: Examining how to define and control the anti-forensics problem. In *Proceedings of the 6th Annual Digital Forensic Research Workshop*, 2006, 44-49.

[19] Jason Daniels, "Forensic and Anti-Forensic Techniques for OLE2-Formatted Documents", MS Thesis, Department of Computer Science, Utah State University, Logan, UT 84322, 2008.