GIAC
CERTIFICATIONS

Global Information Assurance Certification Paper

Copyright SANS Institute
Author Retains Full Rights

This paper is taken from the GIAC directory of certified professionals. Reposting is not permited without express written permission.

Interested in learning more?

Check out the list of upcoming events offering
"Hacker Tools, Techniques, and Incident Handling (Security 504)"
at http://www.giac.org/registration/gcih

Attributes of Malicious Files

GIAC (GCIH) Gold Certification

Author: Joel Yonts, jyonts@malicious-streams.com
Advisor: Antonios Atlasis

Accepted: June 30th 2012

Abstract

Malware has become a common component to most modern intrusions. Confirming a system is infected or finding the attacker-planted backdoor can be a daunting task. To compound the situation, attackers are taking steps to actively evade traditional detection mechanisms. The foundations laid in this paper begin to develop an alternate and supplementary approach for identifying malware through detecting anomalies in the low-level attributes of malicious files. Over 2.5 million malicious samples were analyzed and compared with a control set of non-malicious files to develop the indicators presented.

© 2012 SANS Institute, Author retains full rights.

© 2012 The SANS Institute Author retains full rights.

1. Introduction

One of the most challenging questions that an incident responder must answer is whether a particular file is malicious or benign. This question may take the form of "Is this email attachment safe to open", "Should we use the free utility that was downloaded from the Internet", or "Is this file found on our server the smoking gun for the intrusion"? Regardless of the form, however, a typical approach to the question might be to scan the file with anti-malware software or to attempt some form of behavioral analysis. While these are good first steps, we have seen that today's advanced threats often evade detection from anti-malware software and their behaviors have become quite stealthful (Verizon RISK Team, 2012).

The intent of this paper is to provide insight into building another layer of detection based on examining the building blocks and structures of potentially malicious files. More specifically, an approach to statistically examine attributes of a population of known bad files and compare those same attributes to a set of known good files to identify attributes that could be an indicator of maliciousness. Due to the vast nature of this undertaking and the expansive number of unique file types, this paper will focus on examining only the attributes of the PE32 Compact Executable file, also know as PE32 files (Microsoft Corporation, 2010). PE32 files were chosen because this file type is most widely used for malware and it serves as a good model for a process that could be utilized to explore the malicious attributes of other file types. Further, this research is based in part on the concepts and attributes presented in the Malware Analyst's Cookbook (Ligh, Adair, Harstein, & Richard, 2011).

Finally, the indicators presented in this paper are intended for use in the incident response phase of an investigation. This is important because these indicators may aid an investigator in creating a shortened list of potential malicious candidates but may not lend themselves to use as a real-time protection mechanism due to the potential for a heightened false positive rate.

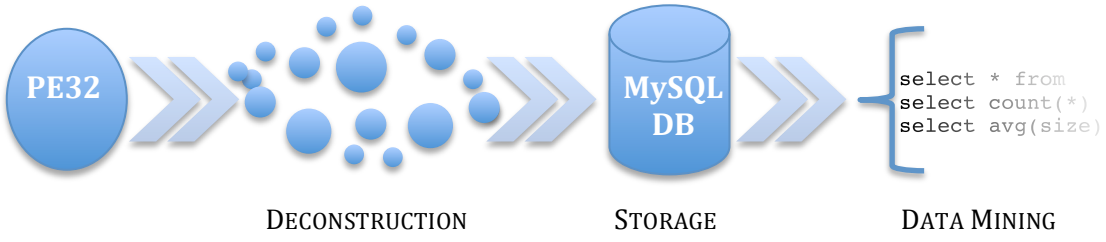
Joel Yonts, jyonts@malicious-streams.com

© 2012 The SANS Institute Author retains full rights.

2. Methodology

A collection of more than 2.5 million malicious PE32 files was used as the "known bad" set, henceforth referred to as the zoo set. This population of files incorporates a combination of worms, trojans, viruses, spyware, and network intrusion related files (password dumpers, covert channels, etc. collected while responding to confirmed network intrusion events). All files in this population are Win32 files with target platforms of MS Windows Server 2008, MS Windows XP, and MS Windows 7. The "known good" set, or control set, is a smaller population of 65,000 Win32 files extracted from a number of MS Windows XP, MS Windows 7, and MS Windows Server 2008 systems. The reason for the smaller size of the control set is largely because there is little variance in PE32 files from system to system for common application and operating system files. Secondarily, Copywrite issues prohibit the collection and sharing of executables from valid applications.

2.1. Technical Design



The technical approach for the analysis outlined above was broken into three phases. During the first phase all samples from each population ("known good" and "known bad") were deconstructed into a series of attributes. These attributes were primarily extracted from the file's PE32 headers (Microsoft Corporation, 2010), (Pietrek, 1994) with the addition of a few whole file attributes such as file entropy and file size. Appendix A contains a complete list of attributes collected during the deconstruction phase. The primary tool used to conduct this phase was the pefile (Carrera, 2011) utility coupled with a collection of custom python scripts.

Once the file attributes were successfully extracted their contents were stored in a series of MySQL database tables whose schema closely matches the PE32 structure names and attribute names from which they were extracted. Finally, the data was mined using specially crafted SQL statements that were executed across both sets. The results from these queries are the foundation for this research and provide the bulk of this paper's contents.

2.2. Process of Discovery

The process of determining malicious indicators involved data mining and statistical analysis of the control set population to determine normal values, ranges, ratios, and distributions of the various attributes within scope (see Appendix A for complete list). Once a baseline was established, these values were compared to the zoo set population. If a significant deviation was noted, an attempt was made to create logic and/or mathematical formulas to describe these variations with the intent of detecting as many samples in the zoo set as possible while minimizing the samples flagged within the control population. These formulas will be henceforth referred to as **Detection Rules**. The percentage of zoo set samples identified by a detection rule will be referred to as the **Detection Rate**. The percentage of samples incorrectly flagged within the control set by applying the detection rule will be referred to as the **False Positive Rate**. If a detection rule is able to achieve a significant detection rate with a minimal false positive rate, the detection rule will be referred to as a **Primary Indicator**. In some cases a detection rule can be constructed which partially describes the deviations between the sets but garners a false positive rate that prohibits the use of the detection rule as a primary indicator, these rules will be referred to as a **Secondary Indicator**. Secondary Indicators are most useful when they are used in conjunction with other data and attributes that help enhance the detection rate while minimizing the false positive rate.

3. Malicious Indicators

The investigation of malicious indicators and subsequent detection rules has been organized into sections that correspond to the PE32 structure for which they apply. Figure 3.0.1 provides a listing of these PE structures/section names that were considered in scope for this research project.

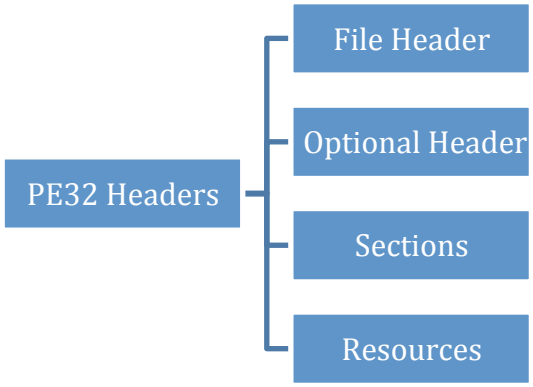


Figure 3.0.1 PE Header Structures Analyzed

3.1. PE FileHeader Indicators

The PE FileHeader or more precisely the COFF File Header describes the system environment and high-level file structure of the PE32 file. This header is used by the Operating System to ensure the target machine requirements are met and to provide guidance to the loader process concerning the size and file location of other structures within the PE32 file. The following subsections show the analysis of PE FileHeader attributes and corresponding detection rules.

TimeDateStamp

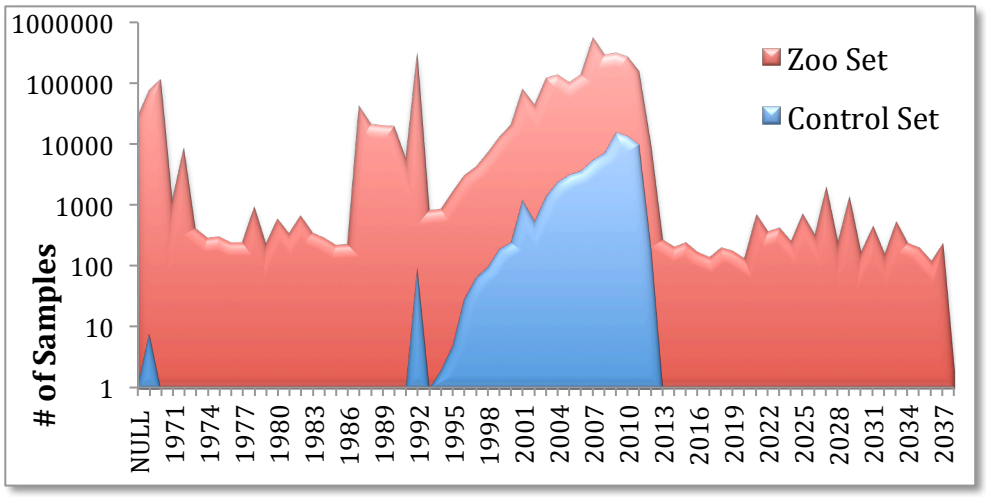


Figure 3.1.1 Distribution of Samples by Year of TimeDateStamp

The TimeDateStamp field is usually set to the build date and time of the PE32 file. While this field is set automatically at link or compiler time, it can easily be modified using specialized tools and therefore a potential indicator for malicious or abnormal PE32 files (Ligh, Adair, Harstein, & Richard, 2011). Examining this field within the control set we see a reasonable range of dates from 1992 through 2012 for 99.98% of the population. When we apply this same range to the zoo set we see that only 87.93% falls within this date range. Looking at the converse of this range we can construct a detection rule that identifies all samples whose TimeDateStamp is older than 1992 or samples with a future date (> 2012). Applying this detection rule to the zoo and control sets result in a respectable detection rate of 12.05% with only a 0.01% false positive rate earning this detection rule the right to be considered a primary indicator.

Year	Control Set	Zoo Set	Δ
<1992	0.01%	11.72%	11.71%
1992-2012	*99.98%*	*87.93%*	
>2012	0.00%	0.35%	0.35%

Table 3.1.1 Distribution of Samples by Year

One other interesting anomaly to note was observed when examining the hour of sample creation. For the control set a spike was observed around 21:00EST which translates to 18:00PST. One theory to explain this spike is to assume this represents an end-of-work-day build for west coast US companies. Considering that the Microsoft Corporation based in Washington State built many of the control files, this seems reasonable. The zoo set however showed an interesting periodicity with spikes every 8 hours starting at 2AM EST. While this seems significant, it was not possible to directly translate this into an algorithm for detection. One practical use of this approach could be to apply this method to a smaller more targeted subset of samples as an aid in discovering attacker TimeZone, Country of Origin, and periods of activity.

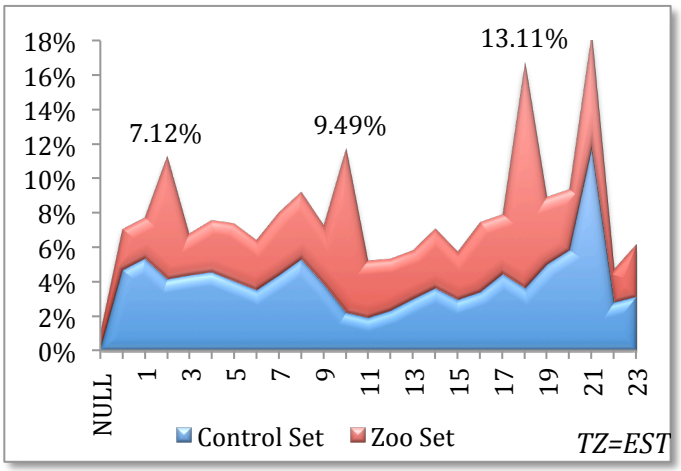


Figure 3.1.2 Hourly Distribution of Sample Population

NumberOfSections

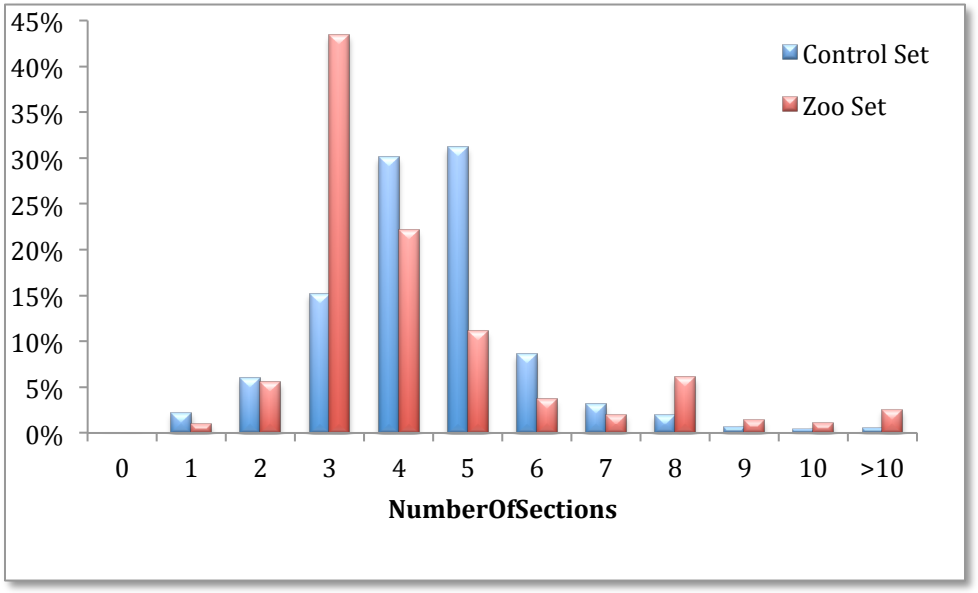


Figure 3.1.3 Distribution of Samples by NumberOfSections

In PE32 files, sections divide the file content between code, data, resources, and various types of variable and configuration data. While there are a vast number of section types and section names possible, in practice most non-malicious PE32 files use a small number of sections. As shown in Figure 3.1.3 and Table 3.1.2, most of the control set samples have between 1 and 8 sections. When comparing this range against the zoo set two interesting deviations are noted. First, Figure 3.1.3 shows a spike of zoo set samples where the NumberOfSections value is set to 3. In fact the data shows a 28.22% deviation between the control set and the zoo set for this value of NumberOfSections. While this deviation may be useful when considered in conjunction with other potential indicators, regretfully the false positive rate (15.21%) of considering this value alone as an indicator

Sections	Control Set	Zoo Set
1	02.22%	01.03%
2	06.03%	05.57%
3	15.21%	43.43%
4	30.08%	22.12%
5	31.19%	11.08%
6	08.64%	03.66%
7	03.19%	01.97%
8	01.93%	06.08%
9	00.65%	01.42%
10	00.35%	01.11%
Total:	99.49%	97.47%

Table 3.1.2 Distribution by Section Count

Detection Rule	Detection Rate	False Positive
N<1 or N>8	5.06%	1.52%
N<1 or N>9	3.64%	0.87%
N<1 or N>10	2.52%	0.51%
where N is the NumberOfSections		

Table 3.1.3 Detection Rules for NumberOfSections

would be too large to be used as a primary indicator. The second deviation noted in Figure 3.1.3 is a marked difference between the Zoo Set and Control Set for samples with a NumberOfSections value of 8 or higher. For these higher values, the Zoo set shows a small but potentially significant population while the control set population shows a steep decline. Table 3.1.3 attempts to capture the potential detection capability of using these ranges of values with the corresponding false positive rates. One final note, 0.02% of the zoo set samples had a NumberOfSections value of 0. Since there were zero occurrences of this value in the control set, this criterion was added (N<1) to the detection ranges in Table 3.1.3.

Symbol Attributes

The PointerToSymbolTable and NumberOfSymbols fields define the location and size of the COFF debugging information (Pietrek, 1994). A value of zero specifies no debugging information was included during compile time. In the majority of cases the control samples should not contain debug information, fields set to zero. The lack of debug information is driven by the deprecation of COFF debugging in favor of PDB files (Glaister, 2007) and by the common practice of stripping debugging symbols for production files. Table 3.1.4 shows the control set, indeed, has a very small population (0.29%) of samples having a PointerToSymbolTable value greater than zero. When examining the same field within the zoo set a seven-fold increase (2.06%) is observed in the occurrence of samples that have a non-zero value for this field. One could argue that this value alone would make a good detection rule (*PointerToSymbolTable>0*). However, to reduce the false positive rate of 0.29% further, Table 3.1.4 lists additional criteria that refine the detection rule and gains the listed reduced false positive rate.

Symbol Attributes
PointerToSymbolTable
NumberOfSymbols

Detection Rule	Detection Rate (Zoo Set)	False Positive Rate (Control Set)
PtrToSymTable > 0	2.06%	0.29%
PtrToSymTable>=Size	1.48%	0.03%
PtrToSymTable>=2xSize	1.46%	0.00%

As stated in the beginning of this section, NumberOfSymbols is a related field and shows a similar but weaker correlation between sets. The control set has a population of 0.41% of samples with a NumberOfSymbols value greater than zero where the zoo set has a population of 1.46% that meets this criterion. In addition, this detection rate has nearly a 100% overlap with the samples detected through the PointerToSymbolTable detection rules. Since these detection and false positive rates are weaker and there is no additive benefit of using NumberOfSymbols as a secondary indicator. This field may not be a useful indicator of maliciousness.

NumberOfSymbols > 0
Control Set: 0.41%
Zoo Set: 1.46%

Throughout the analysis work for this paper, each finding was scrutinized to determine if the finding was the result of a skewed data set. In most cases the findings appear consistent across a broad population and there was little population bias factoring into the results. While the PointerToSymbolTable detection rules seemed to hold true across the broader population, the actual detection rate was skewed by samples claiming a PE Header TimeDateStamp year value of 2004, see Figure 3.1.4. To correct for this potential sample bias, the PointerToSymbol detection rules were applied to a reduced sample set (Samples Year >2008) with the corresponding detection and false positive rates listed in Table 3.1.5.

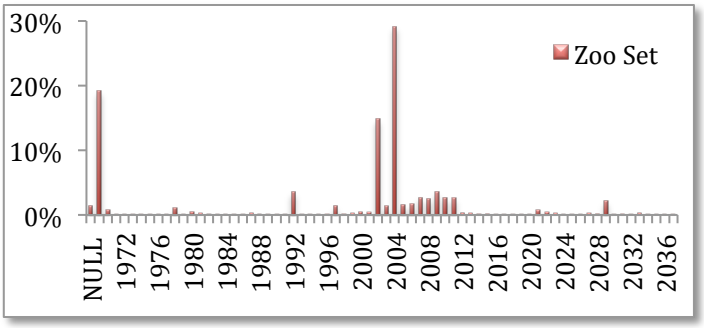


Figure 3.1.4 Distribution of PtrToSymbolTable>0 by Year

Detection Rule	Detection Rate (Yr>2008)	False Positive Rate (Yr >2008)
PtrToSymTable > 0	1.20%	0.17%
PtrToSymTable>=Size	0.83%	0.04%
PtrToSymTable>=2xSize	0.80%	0.01%

Table 3.1.5 PtrToSymbolTable Detection rules for Samples Yr. >2008

Characteristics

The characteristics field is a bit level flag field that is used to specify a combination of 16 different PE32 attributes (Microsoft Corporation, 2010). Figure 3.1.5 lists each of the flags as well as the corresponding population of the control and zoo set samples that have the particular flag set. As shown in the Figure 3.1.5, BYTES_REVERSED_HI and BYTES_REVERSED_LO both make ideal candidates as a primary indicator due to the significant detection rate with a low false positive rate. Not surprising, however, the populations detected by BYTES_REVERSED_LO and BYTES_REVERSED_HI indicators are nearly identical with a 99.88% overlap. RELOCS_STRIPPED also makes an ideal primary indicator due to the very high detection rate but it does come with a potentially significant false positive rate (>10%). Lastly, LOCAL_SYMS_STRIPPED and LINE_NUM_STRIPPED show a significant deviation between the two sets that may make them candidates as secondary indicators. Table 3.1.6 summarizes the detections rules outlined in this section.

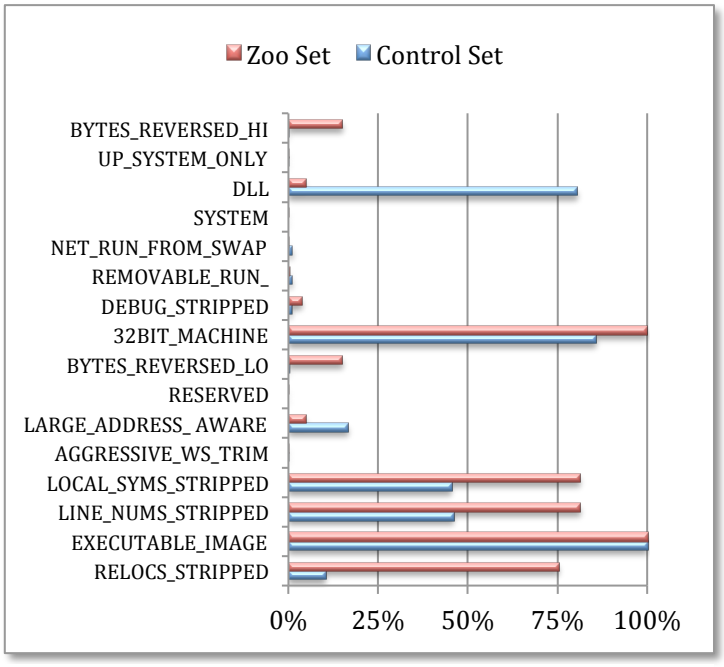


Figure 3.1.5 Distribution by Specified Flags

Detection Rule	Detection Rate	False Positive
BYTES_REVERSED_LO=1	14.99%	0.29%
BYTES_REVERSED_HI=1	14.98%	0.26%
RELOCS_STRIPPED=1	75.29%	10.29%
LOCAL_SYMS_STRIPPED=1	81.24%	45.40%
LINE_NUM_STRIPPED=1	81.33%	45.98%

Figure 3.1.6 Detection Rules for FileHeader Characteristics

Other Attributes

For the sake of completeness, the remaining PE_FileHeader attributes of Machine and SizeOfOptionalHeader were also analyzed but did not yield statistically significant detection rates.

3.2. PE OptionalHeader Indicators

The PE OptionalHeader defines the size and location of several in-memory structures, stores version information for multiple build and run-time attributes, and specifies a limited number of runtime configuration options. Contrary to the name, the OptionalHeader is only optional for object files. All P32 executables and dynamic libraries must have an OptionalHeader. The following subsections show the analysis of PE OptionalHeader attributes and corresponding detection rules.

OptionalHeader Version Attributes

Version Attributes
MajorLinkerVersion
MinorLinkerVersion
MajorOperatingSystemVersion
MinorOperatingSystemVersion
MajorImageVersion
MinorImageVersion
MajorSubsystemVersion
MinorSubsystemVersion

Four pairs of major:minor version number fields exist within the optional header, see Version Attributes table. These attributes are set at compile/link time and specify major and minor version numbers for the linker, operating system, image version, and the intended subsystem, respectively. Since these fields are related categorical data, a frequency table, Table 3.2.1, was constructed to show the finite list of major:minor version number combinations that occur within the control set for each of these attribute pairs with the exception of MajorSubsystemVersion and MinorSubsystemVersion. The subsystem version attributes showed no statistical deviation between the control and zoo sets and consequently were omitted.

In Table 3.2.1 the major:minor version combinations have been organized into a series of sets (H1, L1 … H3, L3) based on the population of the control set that have the specified values. Next, these well-defined sets were compared against the corresponding

	Linker Version (*Major:Minor*)	OS Version (*Major:Minor*)	Image Version (*Major:Minor*)
HIGH	**H1 =** {8:0, 9:0, 7:10, 6:0,7:0, 10:0, 2:56, 5:10, 5:12, 5:0, 2:25, 4:20, 3:10} **Frequency: 99.59%**	**H2 = {**4:0, 6:1, 5:0, 5:1, 5:2, 6:0} **Frequency: 99.74%**	**H3 = {**0:0, 6:1, 5:1, 5:2, 6:0, 8:0, 5:0, 1:0, 9:0, 4:0, 21315:20512, 10:0, 7:0, 2:0, 5:5, 4:1, 170:11, 1:100, 7200:700, 4:73, 6:2, 4:50, 0:1, 3572:2, 4:20} **Frequency: 99.96%**
LOW	**L1 = {**5:2, 3:0, 7:1, 6:20, 2:50, 2:55, 255:255, 10:10, 2:20, 6:24, 6:22, 2:42, 2:60, 83:82, 0:0, 2:43, 4:1, 5:1, 6:1, 6:10, 7:3, 9:12, 13:1} **Frequency: 0.41%**	**L2={**1:0, 6:2, 0:0, 7:0, 7:10, 3:51, 5:20} **Frequency: 0.26%**	**L3 = {**4:43, 6100:3, 1:1, 93:1026, 0:2, 4:35, 3:0, 1919:0, 4:62, 0:1423, 2:6, 3:1026, 2:7, 7104:0, 2:5, 688:0, 2:2, 8:4, 7:10, 2:31, 2:1, 500:0, 2009:1, 4:3, 3:8, 1159:0, 3:1, 8:31, 4:2, 4620:1, 1:3, 4:31, 2:3, 5:20, 1:5, 6:4, 7:2, 3:2, 1741:0, 1021:0, 2000:5, 11:212, 953:0, 3:51, 2:56, 0:8, 9000:15, 1:14, 6:7, 3:5, 3020:21, 7200:800, 2:14, 1:12, 7:1, 1:2, 3:35, 1:6, 833:0, 1000:14, 7000:0, 3:4041, 1:33, 4:8, 170:228, 1:526, 2:9, 9:7, 31:30, 9:1, 1:15, 6400:2, 4:14, 1:24, 0:40, 5:3, 3:32, 7200:600, 6:3} **Frequency: 0.34%**

Table 3.2.1 Distribution of Control Set OptionalHeader Version Attributes

attribute value pairs in the zoo set to identify statistical deviation that could be used for detection. Table 3.2.2 shows the results of this comparison.

Detection Rule		Detection Rate	False Positive
x = MajorLinkerVersion:MinorLinkerVersion,	$x \notin H1$ $x \notin H1 \cup L1$	14.23% 9.13%	0.41% 0%
x = MajorOSVersion:MinorOSVersion,	$x \notin H2$ $x \notin H2 \cup L2$	6.32% 0.61%	0.26% 0%
x = MajorImageVersion:MinorImageVersion,	$x \notin H3$ $x \notin H3 \cup L3$	4.78% 3.78%	0.34% 0%

Table 3.2.2 Detection Rules for OptionalHeader Version Information

OptionalHeader Size Attributes

The OptionalHeader also has a number of size attributes that provide size details of the various code, data, and header areas within the file, see Size Attributes table. Since these values are largely intertwined with the overall size of the sample, it is logical to analyze these attributes as a ratio of attribute value compared to overall sample size. Table 3.2.3 shows the control set distribution of attribute/sample size ratios (rounded) with the exception of the SizeOfStackReserve, SizeOfStackCommit, SizeOfHeapReserve, and SizeOfHeapCommit attributes. These stack and heap related size attributes proved to be statistically insignificant for detection and therefor omitted for the sake of brevity.

Size Attributes

SizeOfCode
SizeOfInitializedData
SizeOfUninitializedData
SizeOfImage
SizeOfHeaders
SizeOfStackReserve
SizeOfStackCommit
SizeOfHeapReserve
SizeOfHeapCommit

	1	2	3	4	5	6	7	8	9+
SizeOfCode/Size	59.26%	0.05%	0%	0%	0%	0%	0%	0%	0%
SizeOfInitializedData/Size	29.92%	0.37%	0.22%	0.21%	0.03%	0.02%	0.02%	0%	0.10%
SizeOfUninitializedData/Size	0.18%	0.15%	0.04%	0.03%	0%	0%	0%	0%	0.01%
SizeOfImage/Size	81.05%	9.83%	3.69%	2.0%	1.53%	0.26%	0.4%	0.32%	0.92%
SizeOfHeaders/Size	0.04%	0%	0%	0%	0%	0%	0%	0%	0%
**Ratio of Attribute Value to Size of Sample (rounded)*									

Table 3.2.3 Control Set Distribution by Size Attribute to Sample Size Ratio

By using the observed ratio distribution in Table 3.2.3, detection rules were constructed (Table 3.2.4) that maximized detection of anomalous size attributes while minimizing the amount of false positives.

Detection Rule	Detection Rate	False Positive
SizeOfCode/Size >1	6.36%	0.06%
SizeOfInitializedData/Size >3	3.58%	0.38%
SizeOfInitializedData/Size >4	3.07%	0.17%
SizeOfUninitializedData/Size >1	13.63%	0.23%
SizeOfUninitializedData/Size >2	4.97%	0.08%
SizeOfImage/Size>8	5.80%	0.92%
SizeOfHeaders/Size>0	2.03%	0.04%
SizeOfHeaders/Size>1	0.14%	0.00%

Table 3.2.4 Detection Rules for OptionalHeader Size Attributes

OptionalHeader Location Attributes

Location Attributes
AddressOfEntryPoint
BaseOfCode
BaseOfData
ImageBase

Included in the OptionalHeader are three location attributes. These attributes specify the location or address of the sample's entropy point, base address of the code, and the base address of the data. In similar fashion as the size attribute above, these addresses have the best clarity when considered in conjunction with the overall size of the sample. Table 3.2.5 shows the distribution of attribute/sample size ratio (rounded) for the control set samples with the exception of ImageBase. The ImageBase attribute proved to be statistically insignificant for detection and consequently omitted for the sake of brevity.

Ratio*	1	2	3	4	5	6	7+
AddressOfEntryPoint/Size	39.23%	1.78%	0.2%	0.09%	0.02%	0.01%	0.02%
BaseOfCode/Size	5.62%	1.32%	0.06%	0.02%	0.01%	0%	0.01%
BaseOfData/Size	55.36%	2.57%	0.61%	0.74%	0.03%	0.01%	0.02%
**Ratio of Attribute Value to Size of Sample (rounded)*							

Table 3.2.5 Control Set Distribution by Attribute as a Ratio to Sample Size

By using the observed ratio distribution in Table 3.2.5, detection rules were constructed (Table 3.2.6) that maximized detection of anomalous size attributes while minimizing the amount of false positives.

Detection Rule	Detection Rate	False Positive
AddressOfEntryPoint /Size >2	12.73%	0.35%
AddressOfEntryPoint/Size >3	7.28%	0.15%
BaseOfCode /Size >2	4.90%	0.10%
BaseOfCode /Size >3	3.66%	0.04%
BaseOfData /Size >4	4.76%	0.05%
BaseOfData /Size >5	4.05%	0.02%

Table 3.2.6 Detection Rules for OptionalHeader Size Attributes

OptionalHeader Miscellaneous Attributes

Misc Attributes
Reserved
LoaderFlags
NumberOfRvaAndSizes

Of the three attributes covered in this section the first two (Reserved, LoaderFlags) are reserved fields and were largely not in-use by the control set samples, Table 3.2.7. The third attribute (NumberOfRvaAndSizes) describes the number of data directory entries within the OptionalHeader. NumberOfRvaSizes, like the Reserved1 and LoaderFlags attributes, has an almost binary like distribution of values where 100% of the control set population has the NumberOfRvaSizes attribute set to the value of 16. While these attributes are weaker from a detection prospective, nevertheless, they are interesting because of the their lack of value variance within the control set population, see Table 5.2.8.

Attribute/Value	True	False
Reserved1 = 0	100%	0%
LoaderFlags = 0	99.99%	0.01%
NumberOfRvaAndSizes = 16	100%	0%

Table 3.2.7 Control Set Distribution of Misc Attributes

Detection Rule	Detection Rate	False Positive
Reserved1 != 0	0.25%	0%
LoaderFlags != 0	0.80%	0.01%
NumberOfRvaAndSizes != 16	2.16%	0%

Table 5.2.8 Detection Rules for OptionalHeader Misc Attributes

OptionalHeader Other Attributes

For the sake of completeness, the remaining PE OptionalHeader attributes of Magic, SectionAlignment, FileAlignment, Magic, CheckSum, DLL Characteristics, and Subsystem were also analyzed but did not yield statistically significant detection rates.

3.3. PE Sections Indicators

As discussed earlier, PE32 files are divided into one or more sections. These sections house code, data, and configuration details of the PE32 file. Analysis of this PE32 area will follow a familiar approach of analyzing the section attributes of the control set to develop a baseline of normal values and value ranges. Next the base line is applied to the zoo set to determine patterns of deviation that could be used to construct detection rules. One deviation from the other PE32 areas analyzed so far, however, is the many-to-one ratio of sections and section attributes to a single PE32 file. Expanding attribute analysis to include the context of related sections (sections that share a common PE32 file ownership) may yield additional detection rules. Expanded analysis of related sections is considered out of scope for this research paper.

Section Size Attributes

Size Attributes
Raw Size
Virtual Size
Number Of Relocations
Number Of Line-Numbers

The size of each section is described by two attributes, Raw Size and Virtual Size. These attributes refer to the size of the section while stored on disk and the size of the section when stored in memory, respectively. These attributes, examined independently, revealed negligible deviation between the control and zoo sets with the exception of the condition when Raw Size = 0. This condition occurs in 13.13% of the zoo set sections but only 0.62% of the control set sections. Examining these attributes as a ratio revealed the zoo set also had a significantly higher percentage of sections whose Virtual Size was either smaller than Raw Size or much larger than Raw Size.

Raw Size = 0
Zoo Set: 13.13%
Control Set: 0.62%

Detection Rule	Detection Rate	False Positive Rate
Virtual Size < Raw Size	86.19%	41.79%
Virtual Size / Raw Size > 10	3.22%	0.71%

Table 3.3.1 Distribution of Virtual Size vs. Raw Size

The Number Of Relocations and Number of Line-Numbers attributes are also size related attributes but refer to PE32 functionality that is either less used or deprecated. Table 3.3.2 validates that these fields are unused in nearly 100% of the control set. Comparatively the zoo set contains a number of samples with non-zero Number of Relocations and Number of Line-Numbers fields.

Detection Rule	Detection Rate	False Positive Rate
Number of Relocations != 0	0.44%	0%
Number of Line-Numbers != 0	0.9%	0.01%

Table 3.3.2 Detection rules for Number of Relocation and Line-Numbers

Section Address Attributes

Address Attributes
Virtual Address
Misc Physical Address
Pointer To Raw Data
Pointer To Relocations
Pointer To Line Numbers

The PE Sections area contains five address related attributes that describe the location of the section's contents, relocations, and line number data. Of these attributes the three Pointer attributes proved to have conditions that could be used to identify potentially malicious sections, see Table 3.3.3.

Detection Rule	Detection Rate	False Positive Rate
Pointer To Raw Data = 0	86.19%	41.79%
Pointer To Relocation != 0	0.84%	0.01%
Pointer To Line Number != 0	1.58%	0.02%

Table 3.3.3 Distribution of Section Pointer Attributes

Section Characteristics Attribute

The Characteristics attribute of PE Sections is a bit level flag attribute that stores up to 40 unique flag values. Table 3.3.4 shows the flags contained within this attribute as well as the frequency of occurrence in the control and zoo sets.

FLAG	Control Set	Zoo Set	FLAG	Control Set	Zoo Set
Reserved1	100%	100%	IMAGE_SCN_ALIGN_4BYTES	0.99%	0.23%
Reserved2	0.00%	0.06%	IMAGE_SCN_ALIGN_8BYTES	0.49%	0.14%
Reserved3	0.00%	0.06%	IMAGE_SCN_ALIGN_16BYTES	0.20%	0.04%
Reserved4	0.00%	0.05%	IMAGE_SCN_ALIGN_32BYTES	0.27%	0.06%
IMAGE_SCN_TYPE_NO_PAD	0.00%	0.08%	IMAGE_SCN_ALIGN_64BYTES	0.00%	0.00%
Reserved5	0.00%	0.07%	IMAGE_SCN_ALIGN_128BYTES	0.00%	0.01%
IMAGE_SCN_CNT_CODE	24.00%	24.25%	IMAGE_SCN_ALIGN_256BYTES	0.00%	0.01%
IMAGE_SCN_CNT_INITIALIZED_DATA	75.49%	70.12%	IMAGE_SCN_ALIGN_512BYTES	0.00%	0.00%
IMAGE_SCN_CNT_UNINITIALIZED_DATA	0.46%	9.65%	IMAGE_SCN_ALIGN_1024BYTES	0.00%	0.00%
IMAGE_SCN_LNK_OTHER	0.00%	0.05%	IMAGE_SCN_ALIGN_2048BYTES	0.00%	0.01%
IMAGE_SCN_LNK_INFO	0.02%	0.06%	IMAGE_SCN_ALIGN_4096BYTES	0.00%	0.00%
Reserved6	0.00%	0.04%	IMAGE_SCN_ALIGN_8192BYTES	0.00%	0.00%
IMAGE_SCN_LNK_REMOVE	0.00%	0.05%	IMAGE_SCN_LNK_NRELOC_OVFL	0.00%	0.02%
IMAGE_SCN_LNK_COMDAT	0.00%	0.05%	IMAGE_SCN_MEM_DISCARDABLE	21.74%	2.45%
IMAGE_SCN_GPREL	0.00%	0.02%	IMAGE_SCN_MEM_NOT_CACHED	0.00%	0.06%
IMAGE_SCN_MEM_PURGEABLE	0.00%	0.02%	IMAGE_SCN_MEM_NOT_PAGED	4.29%	0.62%
IMAGE_SCN_MEM_16BIT	0.00%	0.02%	IMAGE_SCN_MEM_SHARED	0.23%	4.95%
IMAGE_SCN_MEM_LOCKED	0.00%	0.10%	IMAGE_SCN_MEM_EXECUTE	25.36%	36.98%
IMAGE_SCN_MEM_PRELOAD	0.00%	0.02%	IMAGE_SCN_MEM_READ	99.98%	99.66%
IMAGE_SCN_ALIGN_1BYTES	1.20%	0.28%	IMAGE_SCN_MEM_WRITE	22.71%	63.60%
IMAGE_SCN_ALIGN_2BYTES	1.26%	0.30%			

Red value indicates potential indicators

Table 3.3.4 Section Characteristics Distribution

Section Entropy

The final PE Section attribute to consider is section entropy. This attribute is actually not contained within the PE32 structure but rather calculated via external tool. In the case of this research the entropy algorithm utilized is the one contained within the pefile utility. With this algorithm a small floating-point number is generated based on the data supplied to the algorithm with a higher number representing a heighted level of data entropy. Figure 3.3.1 shows the results of applying this entropy function to the control and zoo sets.

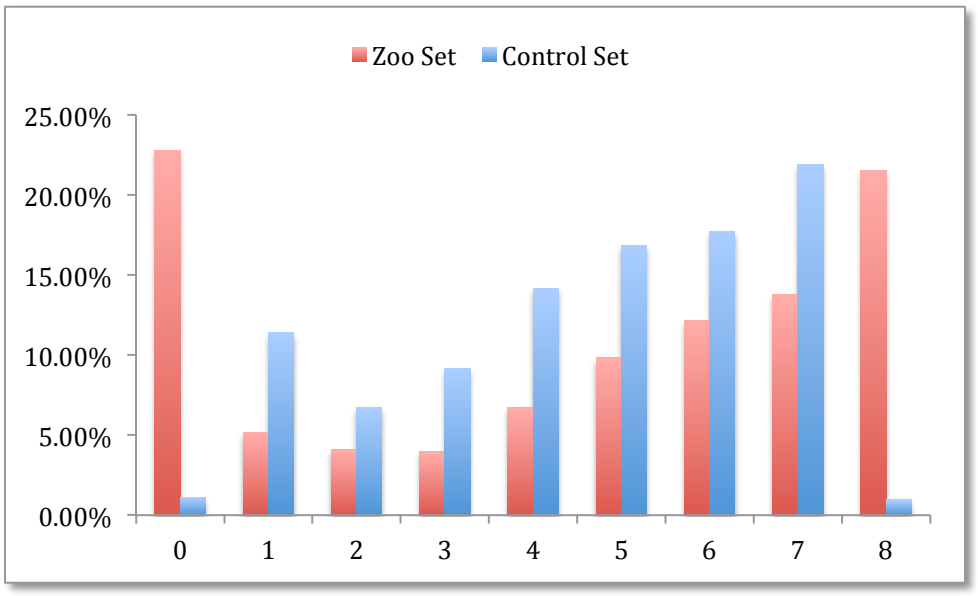


Figure 3.3.1 Entropy(*rounded*) Distribution of Section Data

Figure 3.3.1 shows significant variation between the sets for boundary values of entropy. More specifically the zoo set had a much greater tendency to have sections with very low or very high entropy. This is most likely a result of the malware packing process where the malicious payload is stored, compressed and encrypted, in one section and at run time is decrypted and written into a separate "stub" section for execution. Using these boundary conditions as an indicator yielded the results listed in Table 3.3.5. Expanding the entropy investigation to include entropy analysis of the whole file also yielded significant variation and potential as a detection mechanism, Figure 3.3.2.

Detection Rule	Detection Rate	False Positive Rate
Entropy < 1	22.78%	1.13%
Entropy > 7	21.52%	0.96%

Table 3.3.5 Section Entropy Distribution

File Entropy > 6.9
Zoo Set: 56.18%
Control Set: 3.12%

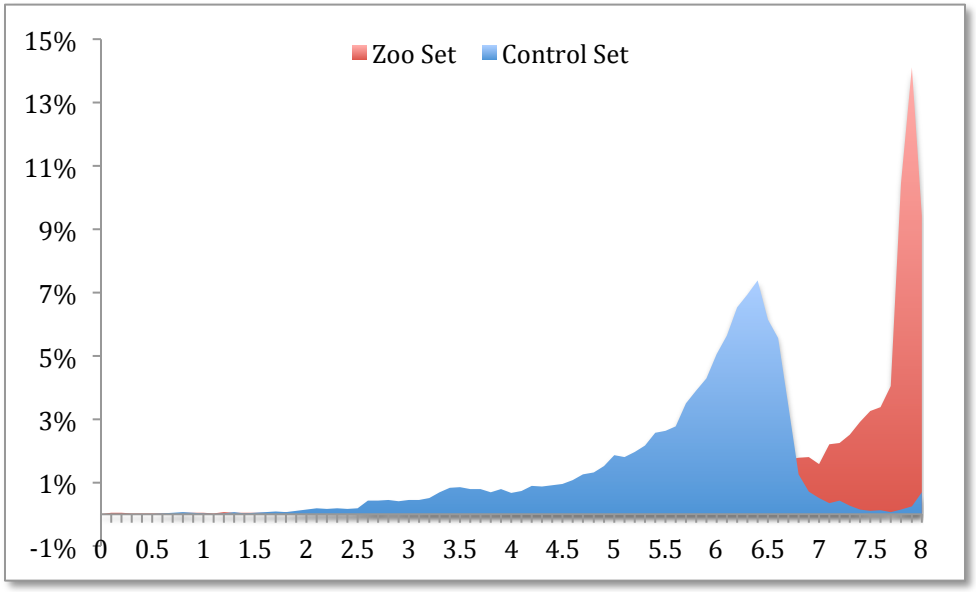


Figure 3.3.2 Entropy Distribution of File Entropy

3.4. PE Resource Indicators

The resource directory (.rsrc section) of a PE32 file is used to store supporting images, fonts, icons, strings, and a variety of similar elements. In the case of malware, PE32 resources are often used to store configuration data and code that assists with delivering the malicious functionality.

Resource Language Attributes

Language Attributes
Language
Sub-Language
Code Page

Analysis of the language-based attributes (see Language Attributes table) revealed that Language and Sub-Language have merit as a potential indicator, although only in the boundary cases listed in Table 3.4.1. Surprisingly, the analysis of the zoo set did not reveal statistical bias toward language configurations for countries known to be larger malware producers. This lack of bias is largely attributed to most professional software solutions (control set PE32 files) having

Detection Rule	Detection Rate	False Positive
Language=0	36.68%	7.85%
Language>127	0.07%	0%
Sub-Language=0	36.66%	0.85%
Sub-Language=2	19.54%	4.30%

Table 3.4.1 RSRC Language Based Detection Rules

multi-lingual support combined with many build tools including multi-lingual support transparent to the user. Values for the Code Page attribute were consistent across both sample sets therefore not useful as a malicious indicator.

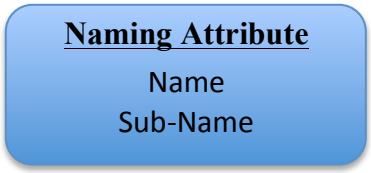
Resource Size

As described in the introduction of this section, the purpose of valid PE32 resources is for providing supporting content and resources for the execution of the PE32 file. With regards to size of these resources, it would be a reasonable hypotheses to assume the size of each resource should only be a fraction of the size of the overall PE32 file. Table 3.4.2 shows the results of analyzing the ratio of resource size to sample size across the control and zoo sets. As seen in the table, the zoo set has a significantly higher population of resources whose size is 25% or greater compared to the size of the overall sample. Many contributors lead to this deviation but one major component may be the practice of malware authors using resources as a storage place for additional execution code. In some cases an entirely new PE32 file is stored within a resource only to be dropped onto a system post-infection. To verify this statement about prevalence of execution code within malware resources, a type identification tool called pescanner (Ligh M. , 2012) was used to identify the type of each resource in the control and zoo sets. The results of this investigation revealed 0.70% of the zoo set resources contained PE32 or PE32 like executable code. Comparatively only 0.21% of the control set had resources that were identified as executable code.

Detection Rule	Detection Rate	False Positive Rate
Resource Size > Sample Size	0.25%	0%
Resource Size / Sample Size > 0.50	0.66%	0.08%
Resource Size / Sample Size > 0.25	1.05%	0.25%
Resource Size / Sample Size > 0.10	1.87%	0.92%
Resource Size / Sample Size > 0.05	2.99%	1.92%

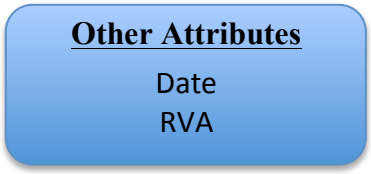
Table 3.4.2 RSRC Language Based Detection Rules

Resource Naming Attributes



The name and subname attributes are largely user defined strings. While many resource creation tools will use predefined strings such as RT_STRING, RT_ICON, and RT_DIALOG these name fields vary greatly across the sample populations with nearly 14,000 unique resource names obseved in the zoo. Often these unique names provide strong clues that a PE32 file may be malicious but the variability makes it impractical to attempt a structured detection rule.

Other Attributes



The remaining PE Resource attributes of Resource Date and RVA were also analyzed but no statistical deviation was observed between the control and zoo sets for these attributes.

4. Conclusion

The exploration of the foundational attributes of malware is a vast topic. The analysis presented in this paper barely scratches the surface of the possibility and the utility of detecting malware based on low level attributes. The beauty of many of the detection rules and anomalies noted in this paper is that they are a by-product of the malicious functionality and often outside the control/knowledge of the malware author (i.e. low level compiler artifacts). These factors are leading reasons that detection based on file attributes provides an excellent supplement to the normal Anti-Virus detection regiment and have a potential for a much longer life span than traditional detection signatures.

To maximize the value of the findings of this research, the explicit detection rules defined in this paper could be expanded to a malicious scoring system. In a true scoring system a function could be created that creates a weighted score based on the ratio of the detection rate / false positive rate balanced with the magnitude of the detection and false positive rates. Such a system would better represent subtle statistical deviations between

the control and zoo sets and provide a better mechanism for correlating the various indicators defined within this paper and with external contributors. The described scoring system is beyond the scope of this paper. Until such a system is developed, an alternate and simpler detection capability could be built that utilizes a scripting language to check for the anomalous values and data ranges identified as primary indicators. The table below summarizes the primary indicators discovered during this investigation.

	Detection Rule	Detection Rate	False Positive
FILE HEADER	*Year* < 1992 or *Year* > 2012	12.05%	0.35%
	NumberOfSections < 1 or *NumberOfSections* >9	3.64%	0.87%
	PtrToSymTable > 0	1.20%	0.17%
	Characteristics (BYTE_RESERVED_LO=1)	14.99%	0.29%
	Characteristics (BYTE_RESERVED_HI=1)	14.98%	0.26%
	Characteristics (RELOCS_STRIPPED=1)	14.99%	0.29%
OPTIONAL HEADER	*MajorLinkerVersion:MinorLinkerVersion* ∉ H1 (Table 3.2.1)	14.23%	0.41%
	MajorOSVersion:MinorOSVersion ∉ H2 (Table 3.2.1)	6.32%	0.26%
	MajorImageVersion:MinorImageVersion ∉ H3 (Table 3.2.1)	4.78%	0.34%
	SizeOfCode /Sample Size >1	6.36%	0.06%
	SizeOfInitializedData / Sample Size >3	3.58%	0.38%
	SizeOfUninitializedData / Sample Size >1	13.63%	0.23%
	SizeOfImage / Size > 8	5.80%	0.92%
	SizeOfHeaders / Sample Size >0	2.03%	0.04%
	AddressOfEntryPoint / Samples Size >2	12.73%	0.35%
	BaseOfCode / Samples Size >2	4.90%	0.10%
	BaseOfData / Samples Size >4	4.76%	0.05%
	NumberOfRvaAndSizes != 16	2.16%	0%
SECTIONS	*Raw Size* = 0	13.13%	0.62%
	Virtual Size / Raw Size > 10	3.22%	0.71%
	PtrToLineNumber != 0	1.58%	0.02%
	Characteristics (IMAGE_SCN_CNT_UNINITIALIZED_DATA=1)	9.65%	0.46%
	Characteristics (IMAGE_SCN_MEM_SHARED=1)	4.95%	0.23%
	Section Entropy < 1	22.78%	1.13%
	Section Entropy > 7	21.52%	0.96%
	File Entropy > 6.9	56.18%	3.12%
RSRC	*Sub-Language* = 0	36.66%	0.85%
	Resource Size / Sample Size > 0.25	1.05%	0.25%

Future research in this area would expand analysis to other interesting structures within the PE32 file format. Examples of additional PE32 structures would include imports, exports, TLS entries, and version information. In addition to expanding the in-scope structures, more sophisticated data mining and machine learning tools may be useful in drawing less obvious connections between anomalous attribute values and malicious files.

5. References

Carrera, E. (n.d.). *pefile*. Retrieved 2012, from Google Code: http://code.google.com/p/pefile/w/list

Glaister, A. (2007, August). *Debugging with Symbols*. Retrieved June 2012, from MSDN: http://msdn.microsoft.com/en-us/library/windows/desktop/ee416588(v=vs.85).aspx

Ligh, M. H., Adair, S., Harstein, B., & Richard, M. (2011). *Malware Analyst's Cookbook.* Indianapolis, IN: Wiley Publishing, Inc.

Microsoft Corporation. (2010, October 05). *Microsoft PE and COFF Specification.* Retrieved January 2012, from MSDN: http://msdn.microsoft.com/en-us/windows/hardware/gg463119.aspx

Quist, D. (n.d.). *Malware Collection*. Retrieved from Offensive Computing: http://www.offensivecomputing.net

Pietrek, M. (1994, March). *Peering Inside the PE: A Tour of the Win32 Portable Executable File Format*. Retrieved May 2012, from MSDN: http://msdn.microsoft.com/en-us/library/ms809762.aspx

Appendix A: In-Scope PE Attributes

PE32 structures and attributes covered in this paper.

A.1.1 FileHeader	A.1.2 Sections	A.1.3 Resources
NumberOfSections	Name	Date
TimeDateStamp	Virtual Address	Name
PointerToSymbolTable	Virtual Size	Subname
NumberOfSymbols	Raw Size	Language
Machine	PtrRawData	Sub-Language
Characteristics	Charact	Code Page
SizeOfOptionalHeader	Misc	RVA
	Misc_Phy_Addr	Size
	PtrRelocs	Type
	PtrLineNums	
	NumRelocs	
	NumLineNums	
	Entropy	

A.1.4 OptionalHeader	
AddressOfEntryPoint	MinorOperatingSystemVersion
BaseOfCode	MinorSubsystemVersion
BaseOfData	NumberOfRvaAndSizes
CheckSum	SectionAlignment
DllCharacteristics	SizeOfCode
FileAlignment	SizeOfHeaders
ImageBase	SizeOfHeapCommit
LoaderFlags	SizeOfHeapReserve
Magic	SizeOfImage
MajorImageVersion	SizeOfInitializedData
MajorLinkerVersion	SizeOfStackCommit
MajorOperatingSystemVersion	SizeOfStackReserve
MajorSubsystemVersion	SizeOfUninitializedData
MinorImageVersion	Subsystem
MinorLinkerVersion	