Statistical Detection of Malicious PE-Executables for fast offline analysis

Ronny Merkel, Tobias Hoppe, Christian Kraetzer, Jana Dittmann

Otto-von-Guericke University of Magdeburg
ITI Research Group Multimedia and Security
Universitaetsplatz 2,
39106 Magdeburg, Germany

{ronny.merkel, tobias.hoppe, christian.kraetzer, jana.dittmann}@iti.cs.uni-magdeburg.de

Abstract. While conventional malware detection approaches increasingly fail, modern heuristic strategies often perform dynamically, which is not possible in many applications due to related effort and the quantity of files.
Based on existing work from [1] and [2] we analyse an approach towards statistical malware detection of PE executables. One benefit is its simplicity (evaluating 23 static features with moderate resource constrains), so it might support the application on large file amounts, e.g. for network-operators or a posteriori analyses in archival systems. After identifying promising features and their typical values, a custom hypothesis-based classification model and a statistical classification approach using the WEKA machine learning tool [3] are generated and evaluated. The results of large-scale classifications are compared showing that the custom, hypothesis based approach performs better on the chosen setup than the general purpose statistical algorithms. Concluding, malicious samples often have special characteristics so existing malware-scanners can effectively be supported.

Keywords: static malware detection, adaptive classification, high scalability, comparison of hypothesis based and statistical classification

1 Introduction and Motivation

Along with the capability of modern IT, also the spectrum of malicious software (malware) has grown more and more complex during the last decades. Current trends are the infiltration of secured connections at the client side (e.g. in online banking) or the extension to the so-called "Web 2.0". Measures for detection (e.g. conventional signature-based approaches) and prevention are increasingly bypassed or disabled. An upcoming challenge is malware detection in vast number of files. Due to performance issues, new algorithms are required to provide fast offline analyses. As a preselection measure, these can support existing detection algorithms, which are potentially more precise but significantly more time-consuming. Conceivable application scenarios are malware scans by network operators (performing live in real-time), a posteriori analyses in archival systems (to scan for malware which was not detectable at archival

time) or computer-forensic investigations that are increasingly performed after incidents and typically need to identify suspicious candidates from a vast number of files. Due to the static context of all these scenarios (the candidate files are not executed), utilising upcoming heuristic, behaviour-based approaches is difficult (since these usually require inspecting the dynamic execution of the code in an isolated environment and are more time-consuming). Therefore the optimal strategy for this scenario would be a heuristic approach, which limits itself to the evaluation of *static* features.

The aim of this work is the detection of malicious code based on statistical properties of executable files. On the example of the Windows PE format, a new prototype for statistical analyses is created, that extracts features of such files which can be obtained statically. This is extended by an adaptive, heuristical classification scheme which is hypothesis-based and can be adjusted for different application scenarios. We also test existing, statistical pattern recognition based classification algorithms to compare their accuracy. Also the combination with commercial products is determined. This way, an additional tool for malware classification is examined, which might also be useful to support and extend existing approaches.

In section 2, we introduce the relevant subset (Windows PE compliant files) considered in our investigations. Section 3 describes our new approach and introduces the selected features being evaluated. A hypothesis-based and a pattern recognition based classification scheme are explained. In section 4, the test setup is introduced and the results of the classifications in practical large-scale test are presented and compared. Section 5 concludes this paper and presents a short outlook.

2 Motivation of PE Executables as Chosen Code Representation

Today, the number of available representations of malicious code nearly equals the number of the various regular code forms in our modern IT environments. In general, code representations can be divided into natively executable code and interpreted code. Both can be categorised into different subclasses. Native code (also malicious) can appear in the form of self-sustaining programs or (not directly executable) code libraries. In the context of exploits, native malicious code can also be embedded into nearly arbitrary cover media. Also the wide spectrum of interpreted code ranging from simple textual script languages to complete programs in intermediate language notations being executed by interpreters or Just-In-Time-Compilers offers multifaceted ways to attackers to create malicious functionality. With respect to the context of web technologies, code representations can also be distinguished by means of their execution on the client or the server side. Especially in the context of the emerging "Web 2.0", the variety of code representations in the web technology domain is constantly growing. While the ideal solution for our practical implementation would have been a generic solution covering malicious code in all of these representations, we have to limit ourselves within this paper to a manageable subset of this wide spectrum. The most relevant reason is that some of these classes notably differ in their basic structure so the acquisition of common properties suitable for the classification would be very challenging. So we have to choose a compromise covering a relevant subset from the variety of code representations which can be evaluated comparably on

a preferably wide range of common features. Also the *practical relevance* of respective malicious code is an important requirement; for example a significant prevalence of respective malicious code is required in order to acquire sufficiently broad training and test sets.
Trading off these factors we first decided for the examination of native code representations having a very high prevalence and practical relevance today. Although interpreted representations have an increasing importance, these classes still have a much higher diversity which is the reason why we reserve them for future research. From the subclasses of native codes we decided for binary executables in Microsoft's Portable Executable (PE) format. One important reason is the high availability of respective reference code, since files in the PE format represent the biggest fraction of today's malicious code and also a sufficient number of non-malicious samples can be gathered without much effort as well, so the acquisition of appropriate training and test sets is realistic. Furthermore, the PE format is well documented [4] and relevant for programs and libraries on different kinds of systems (on PCs as well as on embedded systems like game consoles, PDAs, mobile phones or automotive devices).

3 Our Approach and the Selected Features

As stated in the introduction, the chosen approach is to perform a statistical evaluation of structural features of binary PE files. This heuristic approach on static features is expected to be a reasonable addition to former static approaches (since heuristic detection strategies today frequently rely on dynamic evaluations).

The approach is based on the assumption that many malicious code files contain structural indications which could lead to a successful identification in automated classifications. Some potential reasons we expect to cause such indications are:

- Structural changes performed by malicious code when bound to an existing (previously non-malicious) binary
- Binaries built by malicious code generators having characteristic structural differences compared to programs built in commercial development environments (e.g. non-conformance to the PE specifications in certain issues which are not affecting its ability to be executed)
- General malicious code to which characteristic packers, encrypters or other obfuscations have been applied
- Structural abnormalities caused by low-level programming techniques
- Characteristic functional features of malicious code, e.g. the inclusion of critical combinations of API functions
- Messages left by the malware authors (usually text strings)

Based on this assumption, our investigations and practical examinations follow a 4-step scheme:

Design of potential features: Based on existing work (e.g. in [5], [6] [7] and [8]), 44 potentially suited attributes A_i are identified, which can be extracted from PE files for malware classification purposes. To cover a wide spectrum of potential indications

for the presence of malicious code, different domains of PE files have to be covered. Therefore, characteristics from different domains[1] are selected:

- The outer file structure with several information from the main file headers (DOS header, PE headers), the inner file structure with the sections (information from their individual headers as well as the content within and between them)
- Some overall properties (like file entropy or contained string fragments) as well as functional aspects (e.g. scans for program flow redirection or statistically relevant combinations of imported functions).

In summary, 44 potential attributes are selected from these two domains. To remove unsuitable features, a 2-step feature reduction is performed on this basis (as results of this feature reduction step, the final features are introduced in section 4.2).

Modelling of measures about feature quality: As part of the theoretical concept, formulas are developed to describe the malware probability for each feature presentation as well as a quality measure for each feature. This is introduced in subsection 3.1.

Practical evaluation of feature qualities: Using a training set (see subsection 4.1 for details) we determine the relative occurrence of all feature presentations among non-malicious and malicious samples. The result is a statistic model describing the distributions of feature presentations, resulting quality measures as estimation for the discrimination of the features.

Classification concept and prototype evaluation: A prototype is implemented evaluating the selected attributes to decide for each sample if it is expected to be non-malicious or malicious. Based on two test sets (see subsection 4.1 for details), the prototype is evaluated and optimised with respect to the detection rates. The custom, hypothesis-based classification scheme is introduced in subsection 3.3 while the statistical pattern recognition based classification algorithms, which is used for comparison, is introduced in subsection 3.4..

3.1 Modelling a measure for feature quality

Our formal model is based on the distribution of non-malicious and malicious programs in general, where P(s) describes the general frequency of malicious programs S and 1-P(s) the frequency of the non-malicious programs N. With respect to a given feature A_i and a certain feature presentation $a_{i,j}$ (a sample's value of feature j on attribute i), its probability of occurrence $P(a_{i,j})$ can be calculated from the measured occurrences $P_S(a_{i,j})$ and $P_N(a_{i,j})$ of the feature presentation among non-malicious and malicious code samples in training, respectively:

$$P(a_{i,j}) = P_S(a_{i,j}) * P(s) + P_N(a_{i,j}) * (1 - P(s)) \qquad (1)$$

Whenever a certain feature presentation is found within a given sample, its conditional malware probability can be determined:

$$P(s \mid a_{i,j}) = \frac{P_S(a_{i,j}) \cdot P(s)}{P(a_{i,j})} \qquad (2)$$

[1] Some of the available features can be heavily influenced by the use of runtime packers or crypters. However, as the evaluation (also see section 4) confirmed, most certain packers or crypters are preferably used for malicious (or vice-versa non-malicious) programs which is also respected by the approach.

Based on this, a discrimination measure for feature presentations $D(a_{i,j})$ is defined. It results from the absolute value of the difference between the conditional malware probability and the general probability of malware:

$$D(a_{i,j}) = | P(s \mid a_{i,j}) - P(s) | = | \frac{P_S(a_{i,j}) * P(s)}{P(a_{i,j})} - P(s) | \qquad (3)$$

Where $P(s|a_{i,j})$ is the probability of s, given $a_{i,j}$.

Finally, the quality measure of features (attributes) $0 \leq Q(A_i) \leq 100$ is described by the proportionately added discrimination measures of its feature presentations multiplied by a normalisation factor of 100/P(s). Within our work, we exemplarily assume P(s) having a value of 0.5 (i.e. there is no bias toward one of the two classes), so the normalisation factor is 200:

$$Q(A_i) = 200 * [P(a_{i,1}) * D(a_{i,1}) + P(a_{i,2}) * D(a_{i,2}) + \ldots + P(a_{i,n}) * D(a_{i,n})] \qquad (4)$$

A value of 0.5 has the advantage that the detection starts from a neutral position, so for each individual file there is no a-priory tendency in any direction. Also practical aspects might justify this choice, e.g. if previous measures like whitelists already reduced the amount of non-malicious test samples significantly. However, other values for P(s) are possible as well, only the normalisation factor in the formula given above would differ.

3.2 Choosing attributes and their feature presentations

For the demonstration of the feature quality assessment we use the number of PE sections as an exemplary attribute. The examination of their distribution among non-malicious and malicious software samples reveals, that most malicious code training samples have a number of 3 sections (47.2%) while this holds true for non-malicious training samples in only 10.9% of all cases. Figure 1 shows a graphical illustration of the distribution of the detected section numbers.

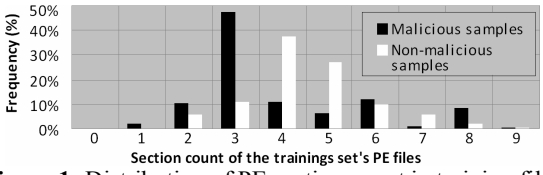

Figure 1: Distribution of PE section count in training files

After the inspection of these values in the malicious and non-malicious samples from the training set, for each attribute certain cosets of feature presentations are defined. This is done in a way that the assignment of the evaluation subjects to these classes into non-malicious and malicious software is as discriminative as possible. With reference to the exemplary feature of the section count in the PE file, the optimal quality for three cosets is reached when the first feature presentation covers files with 0-3 sections (general occurrence probability 0.38), the second one files with 4-7 sections (general occurrence probability 0.55) and the third one files with 8 and more sections (general occurrence probability 0.29). These three feature presentations have malware probabilities $P(s|a_{i,j})$ of 0.78, 0.27 and 0.79 percent, respectively. The calculated quality measure for this feature is $Q(A_i)=50$.

3.3 Our Hypothesis testing based classification scheme

Respecting the class assignments (malicious / non-malicious) of the test samples and the contents of their extracted feature vectors, a hypothesis-based classification model is created. For this purpose, the distributions of the feature presentations are calculated to determine the malware probabilities $P(s|a_{i,j})$ introduced in subsection 3.1: for each attribute value the malware probability $P(s|a_{i,j})$ of its occurrence is determined (see subsection 3.2) from the training set, i.e. the statistical distributions obtained in training are used to test how characteristic each value is for malware.

For the classification of each file the 23 listed features are extracted. Since the real class assignments are known, these can be used to assess the classification's precision. To get stable (and sequence independent) classification results, the actual classification is performed as hypothesis testing, using the null hypothesis that the sample is non-malicious and the alternative hypothesis that the file is malicious. Out of several approaches discussed and compared in [1], the best performing approach uses a point system for weighting. If the malware probability $P(s|a_{i,j})$ for the determined feature presentation is above a threshold S_C of 60%, 1 point is granted for each additional exceedance by 1% (resulting in a maximal score of 40 points per attribute). To illustrate this by an example: During the analysis of the current sample file, the 3rd attribute ("number of sections", see subsection 3.2) is being evaluated. Given the file has two PE sections, its feature presentation is class 1 (0-3 sections). As explained in subsection 3.2, this feature presentation has a malware probability $P(s|a_{3,1})$ of 0.78. This malware probability exceeds the threshold S_C by 0.18, which means 18 points are added to the total score of the sample file.

This way, points are assigned and summed up for each of the 23 attributes of the current file, whereas a high point count indicates a high probability of malicious properties. The actual test is performed by the comparison of the total score with a threshold S_S. At this point, the assignment to the null- or the alternative hypothesis is done. The threshold value is freely adjustable and can be optimised for different application scenarios using the testing results (as shown in the results in subsection 4.3).

Continuing the example from above, after evaluating all 23 attributes the sample file might have been reached a total score of 78 points. If this exceeds the threshold S_S (e.g. 48), the file is finally classified as malicious.

3.4 The Statistical pattern recognition based classification scheme

To compare and assess the results achieved with the hypothesis based classification scheme, we investigate the use of alternate classification solutions. This evaluation is performed by feeding the acquired feature vectors from the training and test samples as respective input for more common or established classification algorithms.

First, we estimate expectable results by having a closer look at the feature vectors of all samples from the training and both test sets, obtained using our prototype. Every sample's feature vector contains 23 elements, each specifying the extracted feature presentation for the respective attribute. We determined the cardinal number of distinct feature vectors by removing duplicates. The results are presented and discussed in subsection 4.4.

For the actual classification tasks, we use the WEKA machine learning tool [3] and explicitly supply the training set and the test sets 1 and 2. From WEKA's broad range of classification algorithms, we select on the basis of initial tests: Simple Logistic [9] (regression models), Naïve Bayes [10], J48 [11] (decision tree), AdaBoostM1 [12], SMO (a multi-class SVM construct) and IB1 (1-nearest neighbour). All classifiers are evaluated with their default parameters, the results are presented in subsection 4.2.

4 Test Setup and results

This section presents the test setup (subsection 4.1) and the results of the feature reduction (subsection 4.2) and both introduced classification approaches (subsections 4.3 and 4.4). Finally, an interpretation of the results is given in subsection 4.5.

4.1 Test setup

A collection of around 7,200 malicious samples (provided by a project partner) and another collection of 15,000 non-malicious files (collected from different sources) were divided to yield two distinct sets for training and test. The training set consists of 5,387 malicious and 10,576 non-malicious samples at a total size of 2.18 GB while the test set contains 1,690 malicious and 4,334 non-malicious samples at a total size of 3.75 GB. To assess the generalisation of the results, another test set is added (referred to as test set 2) consisting of 10,302 malicious samples from a web collection.

4.2 Results of the Feature Reduction

During the Design of potential features (section 3), 44 considered features have been identified. This set of potential attributes has been reduced in two steps:
After this first, theoretic step, first practical evaluations have shown 6 of the considered features as inappropriate.

In the second step, the remaining 38 attributes were assessed as potentially characteristic features. The second, final reduction was performed after the *Practical evaluation of feature qualities (see section 3)*. As introduced in subsection 3.1 and exemplified in subsection 3.2, the feature qualities $Q(A_i)$ have been determined for these 38 attributes. This was performed in a training phase, where PE files from the training set have undergone a feature extraction. As result of the feature reduction, all attributes with a determined feature Quality $Q(A_i) \geq 14$ are considered as useful for anomaly detection. Table 1 lists short descriptions of these 23 most characteristic features with their quality values obtained during the evaluation on the training set (see [1] for a more extensive documentation). For most attributes 2 or 3 feature presentations are defined. In some cases 4 and (in 1 case) 10 of such cosets have shown to be the most promising configuration. These 23 features have been chosen for the classification tests with the heuristic prototype and the two introduced classification schemes.

Table 1: Attributes chosen for anomaly detection with their feature qualities $Q(A_i)$

Attribute	Q	Attribute	Q
1 DOSHeader/StandardValues	19	13 OptionalHeader/Checksum	53
2 DOSHeader/lfanew	35	14 OptionalHeader/NumberOfDataDirectories	14
3 FileHeader/NumberOfSections	50	15 OptionalHeader/DataDirectory/ImportTable	36
4 FileHeader/PointerToSymbolTable	19	16 SectionHeader/Name	70
5 FileHeader/SizeOfOptionalHeader	14	17 SectionHeader/PointerToRawData	14
6 OptionalHeader/SizeOfCode	68	18 SectionHeader/PointerToRelocations, lineNos.	14
7 OptionalHeader/SizeOfInitialisedData	46	19 SectionHeader/Characteristics	67
8 OptionalHeader/SizeOfUnitialisedData	21	20 Sections/CodeRedirection	15
9 OptionalHeader/AddressOfEntryPoint	**72**	21 ImportTable	55
10 OptionalHeader/BaseOfCode&-Data	61	22 FileStringSearch	60
11 OptionalHeader/BaseOfCode	53	23 FileEntropy	67
12 OptionalHeader/MajorImageVersion	61		

4.3 Results from the hypothesis testing based classification

Table 2 lists the detection and error rates at different thresholds. For a balanced compromise between the detection rates on both test sets, an optimal threshold of S_S=43 is identified, where the best average detection rate among both test sets (85% and 81%, respectively) is achieved. The false negatives rate (FNR) at this threshold is 1% for test set 1 and 18% for test set 2 with a false positives rate (FPR) of 14% for test set 1 (remarks: Here, FPR and FNR cover all possible errors, therefore detection rate + FPR + FNR =100%, even when test set 2s FPR must be equal to 0 in all cases since it only consists of malicious samples). For the files in both test sets, Figure 2 illustrates the distribution of the total point scores of all samples. For each possible point score (max score of 920 points = 23 attributes * 40 max. points). Malicious samples of both test sets mainly occur in the right part while most non-malicious samples are present in the left part. A problematic range can be identified at total scores between 27 and 63 points, where a significant overlap of non-malicious and malicious samples exists.

Gelöscht: 3

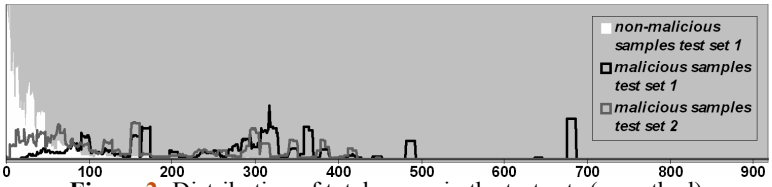

Figure 2: Distribution of total scores in the test sets (smoothed)

Gelöscht: 3

The adjustable threshold allows configuring the prototype with respect to its detection performance. Table 2 shows the detection rates for 7 exemplary threshold values together with the respective FNR and FPR (where available) values. Depending on the focus of the classification, the precision can be freely tuned. For example, by using a low threshold it is possible to achieve a high detection rate of malicious samples accepting a higher FPR. Or the prototype can be configured to achieve a low number of false alarms (using higher thresholds). This flexibility allows a free customisation of the prototype for a given application scenario. For an end user protection system, a low FPR might be intended while in a high security environment or a pre-selection process for further scanning a low FNR should be more important.

Table 2: Detection and error rates for test sets 1 and 2

Exemplary threshold	Test set 1 Detection rate	FNR	FPR	Test set 2 Detection rate	FNR
$S_S = 10$	47.5%	**0.02%**	52.4%	**99.7%**	**0.3%**
$S_S = 17$	58.4%	0.03%	41.6%	96.6%	3.4%
$S_S = 27$	66.8%	0.1%	33.1%	92.3%	7.7%
$S_S = 43$	85.0%	0.7%	14.3%	81.6%	18.4%
$S_S = 63$	88.4%	1.6%	9.99%	73.7%	26.3%
$S_S = 84$	**92.5%**	2.7%	4.8%	63.1%	36.9%
$S_S = 156$	91.2%	7.9%	**0.9%**	40.4%	59.6%

The described malware classification approach can be utilised to support existing mechanisms. We analysed this ability with reference to the two antivirus products *Avira AntiVir Personal* and *Kaspersky Anti-Virus*. First we scan our test sets with these products to compare their results with the result of our heuristic, hypothesis-based approach. We furthermore analyse the capability of our approach to extend such existing scanning tools. Table 3 shows the results of both commercial scanners using all signatures available at the time of the test (January 2009).

Table 3: Classification accuracies of two commercial scanners

		Avira AntiVir	Kaspersky
Test set 1	malicious samples	83,4%	**84,0%**
	non-malicious samples	**100,0%**	99,7%
Test set 2	malicious samples	97,5%	**99,9%**

As common for signature based methods, the tested antivirus products have a FPR close to zero. While this is very satisfying, the FNR for test set 1 is relatively high, i.e. a lot of malicious samples are not detected. One interesting aspect of these results is that the hypothesis-based classification approach performs quite well on these files because obfuscation techniques have been applied to many of the malware samples from test set 1 (taken from the wild), leading to characteristic changes. On the other hand it has difficulties on test set 2 containing relatively clean malware samples from a web collection. A combination of both approaches might therefore be reasonable, which we analysed by feeding malware not detected by the commercial scanners to our prototype (using the hypothesis-based classification strategy).

Table 4: Results on malicious files not detected using signatures

	Test set	Undetected malware (signature based)	Detection rates (prototype) on these files
Avira	1	281	**93,6%**
	2	254	78,7%
Kaspersky	1	**270**	92,6%
	2	**8**	**100,0%**

The results in Table 4 show, that the subsequent addition of our prototype can achieve a clear increase of the detection rates up to 15.5% on the cost of an FPR of 15% (balanced threshold of S_S=43). In scenarios where only an FPR of 1% is acceptable (S_S=156), still an increase of the detection rate by 9.4% would be possible. In scenarios like a high-performance pre-selection of potential malware for more exact but also

more expensive scans (which rather require a low FNR), a hybrid implementation of the evaluated signature based scanners and our heuristic prototype could be applied as well. E.g., at the thresholds of S_S=43 and S_S=10, at least 99% (99.6%) of all malicious samples would be classified correctly at the cost of an FPR of 15% (52%). In principle, the order of application could also be inverted due to the commutativity of both, independent approaches. While the overall FPR remains the same (in the worst case it can be expected to be the sum of both individual FPRs when both systems detect a distinct set of false positives), it might be better to apply the faster scanner first.

4.4 Results from the statistical pattern recognition

As described in subsection 4.2 we first remove all duplicates from the set of all feature vectors because these are redundant for the statistical pattern recognition. This paragraph presents the results from the analysis of the distinct feature vector sets. From the training set (15,963 samples) 1,241 distinct feature vectors have been retrieved. The feature vectors obtained from test set 1 (6,024 samples) and test set 2 (10,302 samples) could be reduced to 796 and 1,460 distinct feature vectors, respectively. Within the classification of the first test set, 546 distinct feature vectors appear that are already known from training, while the other 250 feature vectors have not been observed before. As a first pre-estimation, around 546/796 feature vectors (68.6%) should be classified correctly. Since we face a 2 class problem, about 125 of the 250 unknown feature vectors can be expected to be classified correctly (right guesses), so a total classification accuracy of ca. 84.3% (671/796) can be expected. For test set 2 - having far more unknown feature vectors (1,224) than ones known from training (236) - the estimated classification accuracy can equivalently be determined around 58.1% (848/1460).

Table 5 shows the results of the tests with WEKA's classifiers. The values contained in the second column (error on training data) describe the efficiency of the model generation using the respective algorithm. They can be interpreted as the maximum precision which can be expected as achievable detection rates within real tests. The last two columns contain their classification accuracies as measured in the tests on test set 1 and 2, respectively.

Table 5: Classification accuracies of the statistical classifiers

Classifier	Training set	Test set 1	Test set 2
SimpleLogistic	85.64%	83.37%	62.74%
NaiveBayes	81.13%	80.64%	64.11%
J48	**88.92%**	85.27%	**67.40%**
AdaBoostM1	83.88%	83.73%	49.79%
SMO	84.95%	82.42%	61.78%
IB1	88.62%	**87.53%**	50.55%

The results of the tests with the statistical pattern recognition strategies (Table 5) show that these do not achieve a significantly better precision in malicious code detection compared to the hypothesis testing based approach (Table 2). On test set 1, the hypothesis testig based approach achieves a detection accuracy of 85.0% (balanced mode; S_S=43). On this test set, all except 2 established algorithms achieve worse

results: One exception is the J48 algorithm which is performing slightly better (0.27%), the other one is IB1 achieving a gain of 2.53%. However, all tested established algorithms have clear drawbacks on test set 2: The achieved results are between 14.2% and 33.8% below the results of the hypothesis-based approach (81.6%).

Out of the classification algorithms tested from WEKA, the J48 achieved the best general performance on our malware detection task. An exemplarily performed attribute selection on this classifier using WEKAs attribute selection mechanisms (here CfsSubsetEval evaluation using BestFirst search with default parameters - for details see [3]) showed that nine out of the 23 attributes are responsible for more than 94% of the classification accuracy achieved (see Table 6). The attributes selected as being useful are feature no. 2, 3, 6, 9, 10, 12, 13, 22, 23 (see Table 1 for their short descriptions and their calculated quality values). These results closely match the results obtained by using our quality measure $Q(A_i)$.

Table 6: J48 classification accuracy using attribute selection

Classifier	Training set	Test set 1	Test set 2
J48 with 23 attributes	88.92%	85.27%	67.40%
J48 with 9 attributes	88.92%	84.68%	63.36%

4.5 Interpretation of the results

The tests done with the hypothesis based classification strategy (subsections 3.3/3.4) show that the classification accuracy is clearly lower on test set 2. This is also evident in the tests with the statistical classifiers (subsections 3.4/4.4) to an even greater extent. A reason might be that the malicious code from test set 1 mostly contains obfuscated in-the-wild samples provided by a project partner, while test set 2 contains less obfuscated, relatively clean malware samples from an internet collection. In general, while our heuristic approach on *static* features achieves high detection rates at above 90%, this accompanies high FPRs (see Table 2).

In comparison, the manually developed, hypothesis testing based classification strategy has a better overall performance compared with the statistical pattern recognition approach. One important reason for the better performance might be that our hypothesis-based approach better takes care of the distribution of certain feature presentations amongst non-malicious and malicious samples. It respects this knowledge provided by $P(s|a_{i,j})$ when assigning the penalty points. Since the statistical classification algorithms (subsections 3.4/4.4) only get input in form of distinct, raw feature vectors and the respective class labels, these can not directly utilise such additional information.

5 Summary and Outlook

In this paper, we present results of a work on new approaches for malware detection in static context. Based on the identification and evaluation of significant features within PE files, a prototypical implementation has been created for feature acquisition and sample classification.

The three major results of this paper are:

1. A new feature extractor for statistical analyses on PE files
2. A new application scenario adapted classification approach based on hypothesis testing, which allows by threshold adjusting for an optimisation of false positive and false negative rates
3. The evaluation of the feature extractor and the classifier on large test sets, leading to very good results, e.g. as a promising combination with commercial antivirus products.
4. A direct comparison with statistical pattern recognition based classifications shows that the hypothesis based scheme achieves a better overall accuracy.

One major advantage of the newly introduced classification approach is that it is scaling much better than any signature based approach or statistical pattern recognition approach ever could. This is due to the fact that in the test phase of the classification task (i.e. the deployment of the system) for each sample to be checked only one comparison against the threshold S_S has to be made, in contrast to a check against the complete signature data base.

In future work, especially the appropriateness for generalisation of the test sets acquired for the practical evaluation could be analysed more intensely to further substantiate the potential and restrictions of the approach. Also the extension to other code representations, especially in the context of Web 2.0 technologies is a promising topic for future research.

References

1. R. Merkel: Statistische Merkmale zur Anomaliedetektion in ausführbaren Dateien. Diploma thesis, Otto-von-Guericke-University of Magdeburg, 2009
2. T. Hoppe, R. Merkel, C. Krätzer, J.Dittmann: Statistische Schadcodedetektion in ausführbaren Dateien. In: D-A-CH Security 2009; Syssec, 2009
3. Witten, I.H., Frank, E.: Data Mining: Practical Machine Learning Tools and Techniques, 2nd Edition, Morgan Kaufmann, San Francisco, 2005
4. Microsoft Corporation: Microsoft Portable Executable and Common Object File Format Specification, Revision 8.1, 2008
5. F. Cohen, „Computer Viruses - Theory and Experiments", publication for PhD thesis, 1984, http://all.net/books/virus/index.html (last access: March 2010)
6. P. Szor, „The Art of Computer Virus Research and Defense", 2005, Symantec Press
7. E. Skoudis, L. Zeltser, „Malware: Fighting Malicious Code", 2nd ed., 2004, Prentice Hall
8. S. Treadwell, Mian Zhou: A heuristic approach for detection of obfuscated malware, Proceedings of the 2009 IEEE ISI, p.291-299, June 08-11, 2009, Richardson, Texas, USA
9. N. Landwehr, M. Hall, and E. Frank: Logistic Model Trees, Proc.ECML'03, 2003
10. C. Borgelt, H. Timm, and R. Kruse: Probabilistic networks and fuzzy clustering as generalizations of naïve bayes classifiers. In Computational Intelligence in Theory and Practice, B. Reusch and K.-H. Temme, Heidelberg, Germany, 2001
11. R. Quinlan: C4.5: Programs for Machine Learning. Morgan Kaufmann Publishers, San Mateo, USA, 1993
12. Y. Freund and R. E. Schapire: Experiments with a new boosting algorithm. Proc International Conference on Machine Learning, Morgan Kaufmann, San Francisco, 1996