Embedded Malware Detection using Markov n-grams

M. Zubair Shafiq[1], Syed Ali Khayam[2] and Muddassar Farooq[1]

[1]Next Generation Intelligent Networks Research Center (nexGINRC)
National University of Computer & Emerging Sciences (NUCES)
Islamabad, Pakistan
{zubair.shafiq,muddassar.farooq}@nexginrc.org
[2]School of Electrical Engineering & Computer Science (SEECS)
National University of Sciences & Technology (NUST)
Rawalpindi, Pakistan
khayam@niit.edu.pk

Abstract. Embedded malware is a recently discovered security threat that allows malcode to be hidden inside a benign file. It has been shown that embedded malware is not detected by commercial antivirus software even when the malware signature is present in the antivirus database. In this paper, we present a novel anomaly detection scheme to detect embedded malware. We first analyze byte sequences in benign files to show that benign files' data generally exhibit a 1-st order dependence structure. Consequently, conditional n-grams provide a more meaningful representation of a file's statistical properties than traditional n-grams. To capture and leverage this correlation structure for embedded malware detection, we model the conditional distributions as *Markov n-grams*. For embedded malware detection, we use an information-theoretic measure, called entropy rate, to quantify changes in Markov n-gram distributions observed in a file. We show that the entropy rate of Markov n-grams gets significantly perturbed at malcode embedding locations, and therefore can act as a robust feature for embedded malware detection. We evaluate the proposed Markov n-gram detector on a comprehensive malware dataset consisting of more than $37,000$ malware samples and $1,800$ benign samples of six well-known filetypes. We show that the Markov n-gram detector provides better detection and false positive rates than the only existing embedded malware detection scheme.

1 Introduction

Malware sophistication has evolved considerably during the last decade. In particular, due to emerging financial motivations for attackers, malware trends are now shifting towards stealthy attacks. The challenge faced by stealthy malcode is to reach vulnerable hosts undetected and then to stay undetected on the hosts. 'The longer a threat remains undiscovered in the wild, the more opportunity it has to compromise computers before measures can be taken to protect against it. Furthermore, its ability to steal information increases the longer it remains undetected on a compromised computer' [1]. Code obfuscation, (self-)encryption and polymorphism are commonly-used code transformations that are used by stealthy malware to avoid detection.

In their seminal work, Stolfo et al. discovered a new type of stealthy threat called *embedded malware* [2]. Under this threat, the attacker embeds the malicious code or file inside a benign file on the target host. It was shown that embedded malware cannot be detected by signature-based antivirus detectors even if a malware's exact signature is present in the detector's database [2], [3]. In fact, intelligently infected files can even be opened by their respective application software without providing any observable hint of the infection. Intelligent embedding can be further enhanced to allow automatic execution of embedded malcode when the benign file is opened [3]. Embedded malware is potentially a serious security threat and accurate anomaly detection techniques must be developed to mitigate it.

In this paper, we propose a novel statistical anomaly detection scheme for embedded malware detection. Using correlation analysis, we first show that benign files exhibit a clear 1-st order dependence structure which can be modeled using Markov chains. We therefore propose to characterize the statistical properties of a benign file using conditional n-gram distributions, referred to as *Markov n-grams*, instead of the traditional n-grams. For embedded malware detection, we compute running Markov n-grams over non-overlapping windows in a file. We then use an information-theoretic measure, called *entropy rate*, to quantify perturbations in the Markov n-grams due to embedded malware. The results of our experiments show that the entropy rate of Markov n-grams gets significantly perturbed at malware embedding locations. For automated detection, we observe that the aggregate entropy rate distribution of benign files approaches Gaussianity for large training samples[1]. Therefore, a statistical range of benign entropy rates can be defined using the parameters of the baseline Gaussian distribution. Entropy rate values outside this range can then be classified as malicious.

We compare the proposed Markov n-gram detector with the only known embedded malware detector [2] using two comprehensive and diverse infected datasets. The first dataset is created by randomly embedding malware into benign files. The second dataset is created by randomly embedding naively encrypted malware into benign files. Both datasets are generated from $1,800$ benign samples (including `DOC`, `EXE`, `JPG`, `MP3`, `PDF` and `ZIP` files) and $37,420$ malware samples (containing viruses, worms trojans, spyware, and exploit codes). We show that the Markov n-gram detector consistently outperforms the only existing embedded malware detector [2] in terms of both detection and false positive rates. In comparison to commercial-of-the-shelf (COTS) antivirus (AV) software, our detector provides a significantly higher detection rate at the cost of higher false positive rates. Therefore, we argue that, due to their complementary strengths, very high accuracy can be achieved when the Markov n-gram detector is deployed in conjunction with COTS AV software.

[1] This is a direct consequence of the central limit theorem.

Organization of the Paper. The rest of the paper is organized as follows. We present realistic attack scenarios for embedded malware in Section 2. In Section 3, we provide an overview of related research in the field of embedded malware detection. We then discuss in detail the infected datasets created for our research work in Section 4. Section 5 summarizes the results of our pilot experimental studies. In Section 6, we propose our Markov n-gram detector and in Section 7 we compare the detection accuracy of our proposed detector with other relevant techniques and state-of-the-art antivirus products using the infected datasets. In Section 8, we discuss the limitations of the proposed Markov n-gram detector. Finally, we conclude the paper with an outlook to our future work.

2 Attack Scenarios

In this section, we discuss potential real world attack scenarios that can be realized using embedded malware:

- As demonstrated in [3] and independently verified by us, 'intelligently' embedded malware inside benign (document, media or application) files does not affect their integrity as these infected files continue to open by their respective application software. In fact, our experimental studies have shown that even in the case of naive (i.e., completely random) malware embedding, 10% `DOC` files, 13% `EXE` files, 90% `JPG` files, 100% `MP3` files, 92% `PDF` and 95% `ZIP` continue to open with or without an error message. Moreover, most of the infected files are undetected by COTS AV software. Thus an attacker can embed malware inside common benign files –for instance, a `PDF` help file or a common executable file like `WINWORD.EXE`– and the infected file will go unnoticed through the COTS AV software deployed inside the network or on the host. Such infected files can be transported to different hosts using well-known peer-to-peer file sharing software or by making the file freely available for download. Later on, a user can be tricked into starting a trigger program (in the form of a plug-in or a macro) to launch the malicious code. Examples of similar attacks have recently been reported in [3]–[6].
- Disabling macros and plug-ins is not a viable option because there are many useful benign programs (e.g., MathType, Adobe PDF printer, flash player, etc.) that are launched as macros or plug-ins. Also, in [3] the authors show that the 'object oriented dynamic composability of modern document' formats such as `DOC`, `PPT` and `PDF` allows the user to include embedded objects such as video clips, wave sounds or bitmap images inside a document. The embedded objects can be invoked by simply clicking on the object. An attacker can create a fake embedded object which, in addition to some benign looking activity, executes the malcode [3].
- In our pilot studies, we have observed that `MP3` song files can serve as very potent carriers of embedded malware; 100% infected `MP3` files (with embedded malware) play from start to finish without any error or degradation in

quality[2]. Since most Internet song sharing portals use the MP3 file format, an attacker can use random embedding to infect a benign MP3 file by a malware and then can distribute the infected file via Internet song sharing portals or peer-to-peer file sharing software.

3 Related Work

A significant amount of research effort has recently been focused towards malware detection. To maintain focus, in this section, we describe only those approaches that target embedded malware.

- Stolfo et al. extended their previous work on identification of filetypes using n-gram analysis in [2]. In their earlier analysis, called fileprint analysis, they calculated 1-gram byte distribution of a file and compared it to various models of different file types for eventual identification of the filetype. In the context of malware detection, their work focused on embedded malware detection only in PDF and DOC files. They used 3 different models for representing the benign distributions namely single centroid, multi-centroids and exemplar files as centroids. Mahanalobis distance was calculated between the distributions obtained from these models and the n-gram distribution of a given file. To avoid repetition, details of these techniques will be provided in subsequent sections.
  The authors experimented with 1-gram (byte level) and 2-gram (word level) distributions. They tested their proposed scheme on a dataset comprising 31 benign application executables, 331 benign executables in the System32 folder and 571 viruses. The results of their experiments demonstrated that their scheme was able to detect a considerable proportion of the malicious files. However their approach was not capable of identifying the exact location of the embedded malware in a benign file. Therefore, it is impossible to devise an effective healing strategy for the infected files using their approach.
- In [3], the authors proposed two approaches for embedded malware detection in Microsoft Word documents. The first approach is based on static analysis and the second approach is based on run time dynamic analysis. In the static analysis approach, they used an open source application to decompose Word files into their constituent structures. They used a 5-gram model for benign and malicious documents because it provided reasonable memory and detection accuracy. Based on the 5-gram model for benign and malicious word documents, a "similarity" score was generated for both models for eventual classification. In dynamic analysis approach, they have employed sandbox-based tests to check OS crashes, unexpected changes to the underlying environment, and nonfatal application errors. However, it is acknowledged by the authors that the dynamic analysis approach is not practical to be used as an independent detection scheme.

[2] This is due to the frame-based structure of MP3 files. Each frame in the MP3 file format is preceded by a re-sync marker. Corrupt frames (without re-sync markers) are simply bypassed by the media players during playback.

Table 1. Statistics of Benign Files used in this Study

File type	Quantity	Average Size	Minimum Size	Maximum Size
		(kilo-bytes)	(kilo-bytes)	(kilo-bytes)
DOC	300	1, 015.2	44	7, 706
EXE	300	4, 095.0	48	15, 005
JPG	300	1, 097.8	3	1, 629
MP3	300	3, 384.4	654	6, 210
PDF	300	1, 513.1	25	20, 188
ZIP	300	1, 489.6	8	9, 860

4 Data

In this section, we first describe the benign and malware datasets used in this paper. We then introduce our tool NERGAL that embeds any given infection at any random location within a benign file. Using this tool, we produce a large embedded malware dataset[3].

4.1 Benign Dataset

The benign dataset for our experiments consists of six different filetypes: DOC, EXE, JPG, MP3, PDF and ZIP. These filetypes encompass a broad spectrum of commonly used files ranging from compressed to redundant and from executables to document files. Each set of benign files contained 300 typical samples of the corresponding filetype, which provide us a total of 1, 800 benign files. We ensured the generality of the benign dataset by randomizing the sample sources. More specifically, we queried well known search engines with random keywords to collect these files. In addition, we also collected typical samples on the local network of our virology lab.

Some pertinent statistics of the benign dataset used in this study are tabulated in Table 1. It can be observed from Table 1 that the benign files have diverse sizes varying from 3 KB to 20 MB, with an average file size of approximately 2 MB. We show later in the paper that this diversity in file sizes provides valuable insights into an important aspect of embedded malware detection, that is, whether or not a detector is able to detect the embedded malware in large files where the statistical contents of the malicious code are simply averaged out.

The executable files collected for this study include both compiled and compressed (installation) executables. The ZIP, JPG, and MP3 file formats are inherently compressed so the n-grams on the data portion of these files should provide distributions that are fairly uniform. Evaluation and detection of embedded malware in these uniform distributions is an important issue which was originally raised in [2].

[3] The complete dataset and the tool, NERGAL, are publicly available at http://www.nexginrc.org.

Table 2. Statistics of Malware used in this Study

Major Category	**Minor Category**	**Quantity**	**Average Size**	**Minimum Size**	**Maximum Size**
			(kilo-bytes)	(bytes)	(kilo-bytes)
Backdoor	Win32	3,444	285.6	56	9, 502
Constructor	DOS	178	104.2	62	7, 241
Constructor	Win32	172	398.5	371	5, 971
Email Flooder	-	148	343.5	1, 430	4, 262
Email Worm	Win32	935	73.5	148	762
Exploit	-	242	101.1	370	1, 912
Flooder	-	154	168.1	486	981
IRC Worm	-	485	34.4	56	1, 072
Nuker	-	140	188.1	4, 000	680
Trojan	BAT	649	20.2	12	708
Trojan	DOS	971	27.0	4	1, 818
Trojan	Win32	983	125.4	12	2, 998
Virus	Boot	1,514	32.5	108	1, 490
Virus	DOS	16,236	18.7	5	1, 860
Virus	MS Office	2,596	53.5	118	4, 980
Virus	Win32	991	44.3	175	1, 018
Worm	Win32	153	110.5	97	2, 733

4.2 Malware Dataset

Malware samples, especially recent ones, are not easily available on the Internet. Computer security corporations do have an extensive malware collection, but unfortunately they do not share their malware databases on the Internet. We could only locate 'VX Heavens Virus Collection' [11] database which is available for free download in the public domain. This is a comprehensive database that contains a total of 37, 420 malware samples. The sample consists of backdoors, constructors, flooders, nukers, sniffers, droppers, spyware, viruses, worms and trojans etc.

A detailed description of the malware used in our study is provided in Table 2. The average malware size in this dataset is 64.2 KB. Note that this size is significantly larger than the average size (2 MB) of the benign files. Moreover, the sizes of malware samples used in our study vary from 4 bytes to more than 14 MB. Clearly, small sized malware are harder to detect than larger ones.

4.3 Infected Dataset

We developed an inhouse software tool, called `NERGAL`, that could insert an infection into benign files at any given location in the benign file. `NERGAL` ensure that the infections are inserted after the header of the benign files to avoid file corruption. The tool also generates a detailed infection report, which provides details about the sample malware that was used to infect each benign sample and its offset in each sample.

We have created two infected datasets for this study. The first infected dataset is created by simple embedding malware inside benign files. The second infected dataset is created by encrypting the malware before embedding. We use the ROT-13 Caesar cipher for malware encryption. While more sophisticated encryption techniques are certainly possible, we use a simple substitution cipher because it does not alter the inherent statistical properties of the malcode. Therefore, while COTS AV software will not be able to detect this naively encrypted malcode, we can intuitively argue that the accuracy of anomaly detection techniques should remain unaffected under this simple encryption. (We show later that this is not the case.)

The complete virus dataset is used for every filetype mentioned in the benign dataset. Therefore, the embedded malware dataset for each filetype consists of $37,420$ files and the total number of files in both infected datasets are $449,040$. The average file size in both datasets is $2,267.5$ KB.

5 Pilot Experimental Studies

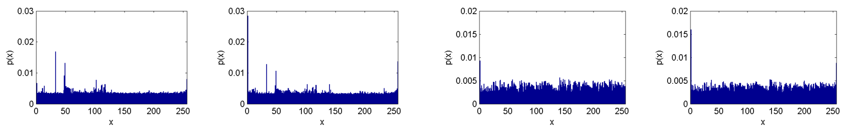
In this section, we repeat and extend the pilot experiments of [2] on our infected dataset. Moreover, we evaluate the accuracy of the Mahanalobis distance based detector which was proposed in [2].

In [2], the authors proposed to use n-gram analysis for embedded malware detection. An n-gram of a sequence is a normalized frequency histogram (or the distribution) of n bit symbols in the sequence. Stolfo et al. [2] used 1-Centroid, Multi-Centroids and Exemplar files as centroids for modeling benign and malware files. 1-gram and 2-gram distributions were used for this purpose. Mahanalobis distances of a given (unknown) file from the benign and the malware model were used for classification. We also wanted to compare our proposed scheme with the static detection approach proposed of [3]. However, it was not possible because their approach is specific to Microsoft Word and similar document formats.

Before evaluating the previous work, we highlight that two desirable accuracy objectives of an embedded malware detector are: 1) to detect infected files and 2) to identify the likely location of the embedded infection. We refer to these two objectives as *detection* and *location identification*, respectively. The technique of [3] reported a reasonable detection accuracy when the infection appeared at the start or the end of a file. However, their proposed scheme could not provide location identification.

5.1 Whole file n-grams for Embedded Malware Detection

One major assumption of the prior study was that the infection appears only at the start or the end of the benign file [2]. Therefore, n-gram analysis was applied only on the truncated files [2]. We argue that this assumption is unrealistic because it is not capable of detecting embedded malware in the middle of the file. In fact, in our experiments we observed that malware embedded at the start of benign files is detected more frequently by COTS AV software than malware embedded in the middle. Therefore, a pragmatic embedded malware detector should look at the statistical contents of an entire file rather than focussing



(a) 1-gram histograms of benign and infected PDF file

(b) 1-gram histograms of benign and infected JPG file

Fig. 1. Comparison of 1-gram histograms of benign and infected files.

on a specific location[4]. We hence revoke the assumption of file truncation and compute n-grams on whole files.

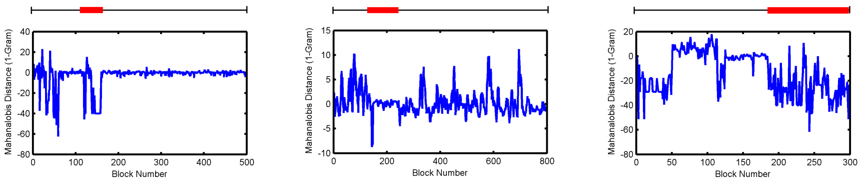
Figures 1(a) and 1(b) show the comparison of whole file 1-grams of sample benign and infected PDF and JPG files, respectively[5]. It can be clearly seen that no discernable change in the 1-grams is evident in Figures 1(a) and 1(b). It can be intuitively argued that whole file 1-grams of infected files do not change when the size of the benign file is significantly larger than the malware size because the statistical contents of the embedded malware are averaged out by large amounts of benign data. (Recall that the average sizes of the benign and malware files in Tables 1 and 2 were 2 MB and 64.2 KB, respectively.) This situation is quite common because malware are generally designed to have small sizes to make them fit inside buffer overflows/file pads/email attachments or to avoid network-based detection during an initial downloading stage.

5.2 Block-wise n-grams for Embedded Malware's Location Identification

The authors in [2] carried out n-gram analysis of a file in a block-wise manner in order to detect the exact location of the embedded malware. Experiments were repeated using block sizes of 500 bytes and 1000 bytes. The significance of the block size is that it sets an approximate bound on the minimum size of malware that can be possibly detected. We repeated these block-wise n-gram experiments on our datasets as well. Figure 2 shows some representative results of the Mahanalobis distance between the block-wise 1-gram distribution and the benign file model. We use a block size of 1000 bytes and plot the Mahanalobis distance between every block and the benign file model; qualitatively similar results were obtained for other block sizes. One can see in Figure 2 that the block-wise 1-gram Mahanalobis distance does not provide significant perturbations

[4] Here we acknowledge the complexity incurred by n-gram analysis of whole files. Nevertheless, we tradeoff complexity for accuracy throughout this paper. In other words, we expect that the proposed detector will be complemented by signature-based detector.

[5] Byte value 0 has the highest frequency because of zero padding that is used for block alignment.



(a) Block-wise 1-gram Mahanalobis distance for an infected `EXE` file

(b) Block-wise 1-gram Mahanalobis distance for an infected `PDF` file

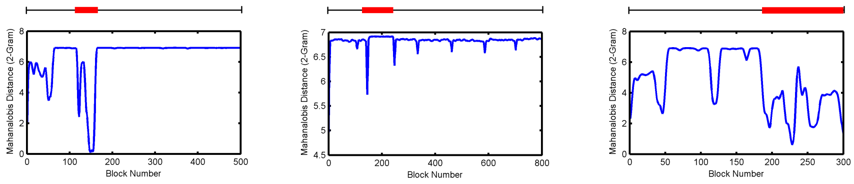
(c) Block-wise 1-gram Mahanalobis distance for an infected `DOC` file

Fig. 2. Block-wise 1-gram Mahanalobis distance is unable to show significant perturbations in the infected regions. The horizontal thick bars show the location of the embedded malware.

that could help in detecting the embedded malware. Figure 2(a) shows the best results with a considerable drop in the Mahanalobis distance. However, one can also observe similar or even larger drops in the benign file regions as well.

Another trend to be observed from Figures 2(a) and 2(b) is that the distance value stays more or less constant in the embedded malware blocks. Interestingly, however, even these trends could not be considered as a common feature across all our experiments as depicted in Figure 2(c). Thus, the Mahanalobis distance of 1-gram distribution of the infected files does not provide us with any concrete measure to robustly detect the embedded malware.

As a logical improvement of 1-gram analysis, we repeated our experiments to analyze the behavior of block-wise Mahanalobis distances of 2-gram distributions; block size is 1000 bytes. Figure 3 shows some representative results for the 2-gram block-wise Mahanalobis distance. These experiments reveal that, despite the increased computational complexity, the performance of the Mahanalobis distance based detector does not improve significantly. Figure 3(a) is an exception where the 2-gram clearly shows discernable decrease in the Mahanalobis distance. Here, we can intuitively argue that the block size of 1000 bytes does not provide enough data to compute an effective statistical distribution. Specifically, in case of 2-gram distribution we only have 1000 data values to fill $65,536$ bins. The ratio (data values to distribution bins) of about $1:65$ for 2-gram is in stark contrast to the ratio of about $4:1$ for 1-gram using the block size of 1000 bytes. A simple solution to this problem is to increase the block size. However, as stated previously, the block size roughly defines the lower bound on the minimum size of malware that can be detected. Therefore, there is an inherent tradeoff between the block size and the minimum malware size that can be detected: increasing the block size means higher false negative rates thereby degrading the accuracy of the detector. This reason stopped us from increasing the value of n and we did not extend our study beyond 2-grams.



(a) Block-wise 2-gram Mahanalobis distance for an infected `EXE` file

(b) Block wise 2-gram Mahanalobis distance for an infected `PDF` file

(c) Block wise 2-gram Mahanalobis distance for an infected `DOC` file

Fig. 3. Block-wise 2-gram Mahanalobis distance is also unable to show significant perturbations in the infected regions. The horizontal thick bars show the location of the embedded malware.

5.3 Discussion

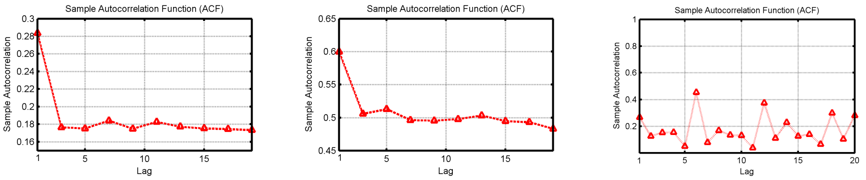
The pilot studies of this section indicate that the block-wise Mahanalobis distance of 1- and 2-gram distributions cannot accurately detect embedded malware. At this point, we conjecture that either n-gram analysis is not a good method for embedded malware detection or Mahanalobis distance is not a good enough quantification measure for differentiating between benign and malicious n-grams. Let us first analyze the latter conjecture which will inadvertently lead us to a substantiation of the former. To quantify changes in the n-gram distributions, we use the *entropy* measure which has been quite effective in quantifying changes in traffic feature distributions [9].

Entropy measures the degree of dispersal or concentration of a distribution [10]. In information-theoretic terms, entropy of a probability distribution defines the minimum average number of bits that a source requires to transmit symbols according to that distribution. Let X be a discrete random variable such that $X = \{x_i, i \in \Delta_n\}$, where Δ_n is the image of the random variable. Then entropy of X is defined as:

$$H(X) = -\sum_{i \in \Delta_n} p(x_i) \log_2 p(x_i). \tag{1}$$

For the present embedded malware detection problem, if the statistical contents of the malware are different from the benign file, then entropy of the blockwise distribution on the infected file should change at the embedding location. We, however, observed that entropy calculation on 2-grams provide qualitatively similar results to the Mahanalobis distance.

The failure of both Mahanalobis distance and entropy measures further strengthens our conjecture that a simple n-gram distribution does not provide sufficient information to detect embedded malware. Consequently, a detector based on simple n-grams meets neither the detection nor the location identification objectives that we set at the beginning of this section. To rectify this shortcoming



(a) Autocorrelation results for benign EXE file

(b) Autocorrelation results for benign DOC file

(c) Autocorrelation results for Code Red II worm

Fig. 4. Autocorrelation function of byte distributions of benign files shows 1-st order dependence. Autocorrelation function of the byte distribution for Code Red II worm shows that the structure of the 1-st order spatial dependence is disturbed.

in the n-gram distributions, in the following we provide a different method of computing n-grams in the following sections.

6 Modeling and Quantification of n-gram Information

We first note that the 2-gram distribution is in fact the joint distribution of two 1-gram symbols. This joint distribution may contain some redundant information which is not pertinent to the present embedded malware detection problem. For accurate detection, it is important that this redundancy is removed. To this end, we analyzed a number of statistical properties of the benign files' n-grams. One relevant property that provided us interesting insights into statistical properties of file data was the analysis of byte level autocorrelation of benign files.

6.1 Correlation in File Data

Autocorrelation describes the correlation between the random variables in a stochastic process at different points in time. For a given lag k, the autocorrelation function of a stochastic process, X_i (where i is the time index) is defined as:

$$\rho[k] = \frac{E\{X_0 X_k\} - E\{X_0\}E\{X_k\}}{\sigma_{X_0}\sigma_{X_k}}, \tag{2}$$

where $E\{.\}$ represents the expectation operation and σ_{X_i} is the standard deviation of the random variable at time lag i. The value of the autocorrelation function lies in the range $[-1, 1]$, where $\rho[k] = 1$ means perfect correlation at lag k (which is obviously true for $k = 0$) and $\rho[k] = 0$ means no correlation at all at lag k.

To observe the level of spatial dependence in the byte sequences of benign files, we computed their sample autocorrelation functions. Figures 4(a) and 4(b) show the autocorrelation function plotted versus the lag for EXE and DOC files, respectively. These autocorrelation results clearly show that the byte sequences in benign files have 1-st order dependence because the autocorrelation value takes a fairly significant dip at $k = 2$ and remains constant for higher values of lag. In other words, once a byte S_i appears, it is more likely that it will be followed by S_i at the next byte location. Clearly, if we are in a zero padded region of a benign file, a zero valued symbol is highly likely to be followed by another zero valued symbol.

This 1-st order spatial dependence of benign files has direct implications on the present embedded malware detection problem mainly because this structure is not observed in malware files, see Figure 4(c). In fact, instead of the 1-st order dependence, we can instead observe high correlation at $k = 6, 12$, and 18. This lack of 1-st order spatial dependence of a malware can be easily observed by examining the signature of the Code Red II Worm given below [8]:

```
   GET /default.ida?XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX%u9090%u6858%ucbd3
%u7801%u9090%u6858%ucbd3%u7801%u9090%u6858%ucbd3%u7801
%u9090%u9090%u8190%u00c3%u0003%u8b00%u531b%u53ff
%u0078%u0000%u00=a HTTP/1.0
```

We can see in the signature of Code Red II that it consists of sub-blocks of 6 bytes, as a result, the high correlation values are observed at $k = 6$ or its integral multiples.

Discussion. In addition to the CodeRed example shown in Figure 4(c), we also conducted correlation experiments on other malware and benign filetypes. These correlation results were consistent with the already presented results. We hence deduce that the 1-st order dependence structure due to zero pads in benign files is not present in malcode. This result is also intuitive because a main objective of effective malcode development is to limit the size of the malcode. (Small sized malware can fit into buffer overflows and can avoid arousing suspicion during transmission over the network.) This objective is clearly defeated if an attacker allows a large zero pads inside the malcode file.

We note that the difference in 1-st order correlation structure of benign and malicious files is actually a distinguishing feature that can be used to detect embedded malware. Therefore, in the following section we model and quantify this distinguishing feature.

6.2 A Statistical Model of Benign Byte Sequences

We now focus on developing a model for the correlation structure observed in benign files. Since the correlation shows 1-st order dependence, the underlying random process (i.e., the byte sequence of benign files in the present context) can be modeled using an order-1, discrete time Markov chain [10]. Here we note that a Markov chain characterizes a process in terms of conditional distribution of its states. For a byte level distribution, a Markov representation simply implies $2^8 = 256$ conditional probability distributions, each corresponding to a different byte value. These conditional distributions reduce the size of the underlying sample space which in the present problem corresponds to removing redundant information from the joint distribution.

The Markov Chain used to model the conditional byte distribution is an order-1 (256 state) Markov chain. The transition probabilities are computed by counting the number of times byte i is followed by byte j. These probabilities can also be expressed as a transition probability matrix. If the probability of moving from state i to j is $p_{i,j}$, then the transition matrix for the present problem is given by:

$$\mathbf{P} = \begin{bmatrix} p_{0,0} & p_{0,1} & \cdots & p_{0,255} \\ p_{1,0} & p_{1,1} & \cdots & p_{1,255} \\ \vdots & \vdots & \ddots & \ddots \\ p_{255,0} & p_{255,1} & \cdots & p_{255,255} \end{bmatrix}$$

Each row of this transition probability matrix provides the conditional distribution for a distinct byte value. Thus the total number of variables that characterize this random process ($65,536$ floating point values) is the same as the 2-gram distribution. However, these Markov chains provide an alternative, non redundant and conditional representation of the jointly distributed 2-gram values. Henceforth, we refer to this representation as *Markov n-grams*.

We now need an accurate measure that can quantify changes in the Markov transition probabilities. This measure is presented in the following section.

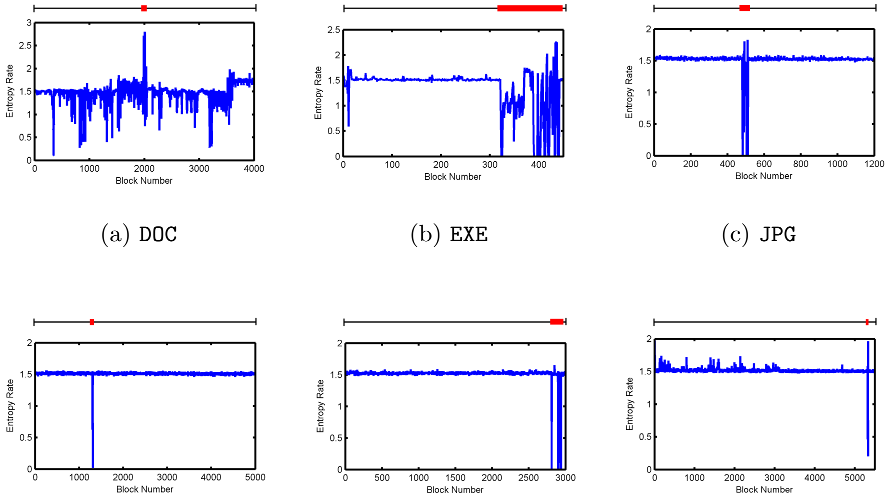
6.3 Quantification of Perturbations in Markov n-grams

We need a mathematical measure to quantify changes or perturbations in the Markov n-gram's transition probability matrix. To this end, we use an information theoretic measure, called *entropy rate*, which quantifies the time density of the average information in a stochastic process [10]. Entropy rate for a sequence of discrete finite random variables X_1, X_2,..., X_n is defined as:

$$R = \lim_{N\to\infty} \frac{H(X_1, X_2, ..., X_n)}{N}, \tag{3}$$

where $H(X_1, X_2, \ldots, X_n)$ is the joint entropy of random variables $X_1, X_2, \ldots, X_n$. R does not exist in general. However, for the present n-gram Markov chain with 256 states, the entropy rate can be computed using (1) as:

$$R = \sum_{i=0}^{255} \pi_i H(X_i), \tag{4}$$



(a) DOC

(b) EXE

(c) JPG

(d) MP3

(e) PDF

(f) ZIP

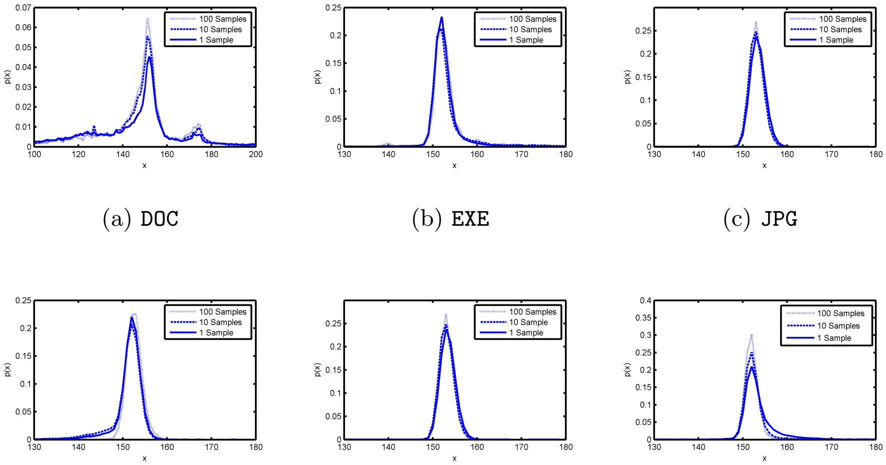
Fig. 5. Entropy Rate of infected files. The horizontal thick bars show the location of the embedded malware.

where π_i represents the equilibrium probability of being in state i and $H(X_i)$ is the entropy of the conditional distribution of state i (i.e., the entropy of row i of the transition probability matrix).

Asymptotic properties of the entropy rate measure are applicable only in case of stationary Markov chains [10]. We acknowledge that in general stationarity will not hold for the present problem. However, the entropy rate expression does provide us with the *expected entropy* of a discrete time Markov chain. Since we rely on the premise that the statistical properties of the embedded malware will be different from the statistical properties of the benign file in which it is embedded, *expected entropy* of the consequent Markov chain (derived from the infected file) should be perturbed at the embedding locations.

Figure 5 shows the entropy rate of infected files of every filetype used in our study. It is clear from Figure 5 that the perturbation is more profound as compared to those obtained using 1-gram Mahanalobis distance or 2-gram Mahanalobis distance. This clearly verifies our earlier hypothesis that the conditional distribution discards the redundant information contained in the joint n gram distribution, thus providing us a compact representation of the file data.

The results of Figure 5 show that the entropy rate of Markov n-grams can quantify and highlight perturbations at the locations of the embedded malware. Thus this measure satisfies the detection and location identification objectives that we have set for an effective embedded malware detector. However, for automated detection, we must threshold entropy rate values above and below which



(a) `DOC` (b) `EXE` (c) `JPG`

(d) `MP3` (e) `PDF` (f) `ZIP`

Fig. 6. Sampled entropy rate distributions for different filetypes.

an infection would be detected. The following section provides a flexible yet accurate method of defining this threshold.

6.4 Classification using Entropy Rate Thresholding

For classification purposes, we need to set an appropriate threshold value on the block-wise entropy rate values. For this purpose, we develop a generic model of block-wise entropy rate values in the benign files. During our pilot studies, we observed that the block-wise values of entropy rates varied in the range of $[0, 3]$. Therefore, we generated an entropy rate histogram using 300 equal sized bins. We normalized this histogram to obtain the *sampled entropy rate distribution*. Figure 6 shows that the sum of sampled entropy rate distributions approaches Gaussianity as the number of samples (i.e., the benign files used for training) approaches infinity. This is a consequence of the central limit theorem which asserts that, for independent finite variance entropy rate distributions, an aggregated distribution should be normally distributed.

Normal distribution is completely specified by its first and second central moments: mean (μ) and variance (σ^2). Since 99.99% of the times a normal distribution does not deviate from its mean by more than 5 standard deviations, we can set the upper and lower detection thresholds as: $\eta_{low} = \mu - 5\sigma$ and $\eta_{high} = \mu + 5\sigma$, respectively. Moreover, we integrate the sampled entropy rate distribution of a test file outside these points to obtain area at the fringes of the distribution and set a classification threshold k on this area. If the area outside the $[\eta_{low}, \eta_{high}]$ range is greater than k then the test file is classified as malicious.

Conversely, if the area inside the $[\eta_{low}, \eta_{high}]$ range is less than or equal to $(1-k)$ only then an alarm is raised. The value of this threshold was tuned for the best performance in the ROC space using a randomly sampled training dataset which was 5% of the total testing dataset [16]. Suitable values of this threshold were different for different filetypes. The highest value of k was observed for the DOC filetype. An intuitive feel for the high value of k for DOC files can be developed with the help of Figure 6. DOC filetype shows worst convergence to Gaussianity, i.e., significant amount of area is present at the fringes (see Figure 6(a)). This logically leads us to set a relatively higher value of the threshold (k) for DOC files.

Discussion. Based on the results of this section, we conclude that entropy rate of Markov n-grams can achieve both accuracy objectives (i.e., detection and location identification) that we expect from an embedded malware detector. To the best of the authors' knowledge, the location identification objective cannot be achieved by any existing embedded malware detector. Moreover, and again in contrast to existing techniques [2], [3], malware data is *not* required to train our proposed detector. This makes the Markov n-gram detector a true anomaly detector that detects maliciousness by flagging deviations from a robust model of normal behavior. In addition to these desirable properties, the following section shows that the proposed detector provides better accuracy than existing schemes.

7 Classification Results

As mentioned in Section 4, we perform classification on two infected datasets, each consisting of $224,520$ infected files created by infecting benign files of six common types: DOC, EXE, JPG, MP3, PDF, ZIP. For the training of our proposed scheme, we use 5% of the benign dataset.

Table 3 provides the detection rates of 3 fully updated[6] commercial antivirus products: McAfee Antivirus [12], AVG Antivirus [14] and Kaspersky Antivirus [13], and our Markov n-gram detector. The results tabulated in Table 3 reaffirm that: *commercial antivirus products are not effective in detecting embedded malware*. Moreover, and as expectd, the detection rate for all COTS AV software degrade to 0% for the encrypted dataset. The false positive rates for these AV products are also 0% because they mostly use signature-based scanning techniques. Mahanalobis n-gram detector performs much better than COTS AV software. However, its performance also degrades for the encrypted dataset. In comparison, our proposed Markov n-gram detector achieves the best average detection rate with a reasonable average false positive rate. Furthermore, the accuracy of the Markov n-gram detector persists for the encrypted dataset. This is because entropy of a random variable is not dependent on the values or image of the underlying random variable. Therefore, a shift in the histogram does not change the entropy rate values.

[6] by January 2008.

Table 3. Detection (TP) rate and False Positive (FP) rate of Antivirus and Anomaly Detectors

	McAfee	**AVG**	**Kaspersky**	**Mahanalobis**	**Markov**	**Percentage**
	Antivirus	**Antivirus**	**Antivirus**	***n*-gram**	***n*-gram**	**Improvement**
	[12]	[14]	[13]	**Detector**	**Detector**	
unencrypted DOC						
TP rate	0.1%	0.0%	0.0%	65.6%	66.3%	**0.7%**
FP rate	0.0%	0.0%	0.0%	48.8%	29.2%	**19.6%**
encrypted DOC						
TP rate	0.0%	0.0%	0.0%	57.6%	67.7%	**10.1%**
FP rate	0.0%	0.0%	0.0%	46.2%	31.4%	**14.8%**
unencrypted EXE						
TP rate	2.7%	1.3%	0.1%	54.1%	84.9%	**30.8%**
FP rate	0.0%	0.0%	0.0%	47.3%	16.7%	**10.6%**
encrypted EXE						
TP rate	0.0%	0.0%	0.0%	56.1%	84.5%	**28.4%**
FP rate	0.0%	0.0%	0.0%	54.3%	17.2%	**37.1%**
unencrypted JPG						
TP rate	0.0%	0.0%	0.0%	76.3%	95.4%	**19.1%**
FP rate	0.0%	0.0%	0.0%	35.7%	2.7%	**33.0%**
encrypted JPG						
TP rate	0.0%	0.0%	0.0%	68.9%	94.6%	**25.7%**
FP rate	0.0%	0.0%	0.0%	46.7%	3.5%	**43.2%**
unencrypted MP3						
TP rate	0.0%	0.0%	0.0%	63.8%	95.0%	**31.2%**
FP rate	0.0%	0.0%	0.0%	32.3%	0.2%	**32.1%**
encrypted MP3						
TP rate	0.0%	0.0%	0.0%	58.6%	96.1%	**37.5%**
FP rate	0.0%	0.0%	0.0%	48.3%	0.2%	**48.1%**
unencrypted PDF						
TP rate	5.2%	2.5%	3.6%	75.4 %	84.5%	**9.1%**
FP rate	0.0%	0.0%	0.0%	46.8%	31.8%	**15.0%**
encrypted PDF						
TP rate	0.0%	0.0%	0.0%	63.2%	84.8%	**21.6%**
FP rate	0.0%	0.0%	0.0%	45.5%	31.9%	**13.6%**
unencrypted ZIP						
TP rate	0.0%	0.0%	0.0%	60.0%	90.4%	**30.4%**
FP rate	0.0%	0.0%	0.0%	29.9%	8.3%	**21.6%**
encrypted ZIP						
TP rate	0.0%	0.0%	0.0%	55.5%	90.6%	**35.1%**
FP rate	0.0%	0.0%	0.0%	28.0%	8.9%	**19.1%**

The reason for less than 100% detection rate of Markov n-gram detector can be traced back to our earlier comment in Section 5.2: the lower bound on size of detectable embedded malware is roughly set by the block's size. Now recall that we have used a block size of 1000 bytes, while the size of 23.6% files in the VX Heavens malware dataset ([11]) is less than 1000 bytes. Nevertheless, even under this block's size limitation, the smallest malware that the proposed Markov n-gram detector is able to detect is `Worm.Win32.Netsp`, which is only 343 bytes. We also note that 8.2% of files in the VX Heavens malware dataset are smaller than 343 bytes. These file comprise malware that the proposed detector is unable to detect.

We argue that it is not entirely fair to compare the accuracy of the Markov n-gram detector with the scheme proposed in [2] because their scheme focuses on *detection* while ignoring *location identification*, whereas the detector proposed in this paper caters for both of these objectives. Despite this fact, the accuracy of the Markov n-gram detector is significantly higher than the Mahanalobis detector. The bold right column in Table 3 gives the percentage improvement in the TP rate and the FP rate for the Markov n-gram detector as compared to the Mahanalobis n-gram detector. It can be observed that the TP rate of the Markov n-gram detector is on the average 20.2% and 26.4% greater than the TP rate of the Mahanalobis detector for non-encrypted and encrypted datasets, respectively. Similarly, the FP rate of the Markov n-gram detector is on the average 21.9% and 29.3% smaller than the FP rate of Mahanalobis detector for non-encrypted and encrypted datasets, respectively. The clearly indicates the superior detection accuracy and robustness of our proposed detector as compared to the detector proposed by the authors in [2].

We, however, do admit that the FP rates for `DOC` and `PDF` files are still significantly high albeit much smaller as compared to the Mahanalobis detector. Our investigation revealed that *embedded objects* are allowed both in `PDF` and `DOC` files. The entropy rate at the location of these objects, at times, also shows a significant perturbation. As a result, our detector is mislead to classify these objects as malware. We will shortly introduce our hybrid strategy that will solve this problem of high FP rate.

Another important conclusion of the research by the authors in [2] is: if the size of the benign file in which the infection is inserted is between 10 KB and 10 MB then on the average the false positive rate of their scheme surges to 50%. In comparison, our scheme has two desirable features: 1) capability to identify block/blocks of benign file in which the infection was inserted; 2) a significantly smaller false positive rate relative to the Mahanalobis n-gram detector. Here, we must note that sizes of more than 90% of the benign files in our dataset also lie in the 10 KB to 10 MB range (average size = 2 MB). This encouraging performance clearly substantiates the potential of the Markov n-gram detector for embedded malware detection.

8 Limitations of the Markov n-gram detector

In this section, we present the limitations of the Markov n-gram detector proposed in this paper.

- The first, and perhaps the biggest, shortcoming of the proposed Markov n-gram detector is its high false positive rate for certain types of files. We, however, believe that these false positives can be significantly reduced if we use the proposed detector as a preprocessor to the COTS AV detection software. During this preprocessing stage, the Markov n-gram detector can be utilized to detect the presence and the location of embedded malware inside a benign file, albeit with false positives. We can then extract a small portion of the file around the infected location into a separate standalone file. The COTS AV software can then scan the new (extracted) file for the presence of a known malcode signature. Clearly, this hybrid detection strategy will significantly lower the false positives, while still maintaining the high detection rate of our detector. This hybrid detection strategy is only realizable because our detector can identify the location of the embedded malware.

- One form of the embedded malware discussed in this paper is dormant (see Section 2 for more details), which does not pose any direct threat to the victim machine. The dormant form of embedded malware requires another program (such as a trojan) to extract and activate it. As a results, this problem is similar to detecting watermarks or steganographic content. It is unlikely that our scheme will detect a malware embedded using the advanced steganographic embedding schemes. However, the use of advanced steganographic schemes have two major drawbacks when considered in the context of embedded malware: 1) the steganographic embedding and extraction algorithms have high memory and computational overheads; 2) they are media specific, i.e. they are specific for images, audio or video content, so they cannot be generalized to all filetypes. The first drawback implies that the steganographic extraction algorithm should be present on the victim machine. Transfer of extraction program to the victim machine is an additional and undesirable overhead. Also, the computational overhead, at the victim machine, imposed due to the extraction algorithm allows for host based anomaly detection. The second drawback limits the scope of the threat posed by embedded malware.
- Since the underlying principle of our proposed detector is based on statistical analysis, a crafty attacker may launch a mimicry attack [17] by modifying the malcode to have a benign looking statistical distribution [3]. Polymorphic attack engines can be used to modify the statistical distribution of a code segment to avoid detection by our proposed detector. This unfortunate limitation is not specific to the Markov n-gram detector and is applicable to any anomaly detector, including the Mahanalobis n-gram detector and COTS AV software [17].

9 Conclusions

In this paper, we proposed a novel embedded malware detection scheme based on the principles of statistical anomaly detection. This scheme, to the best of our knowledge, is the first anomaly based malware detection approach that has the capability to locate the position of the infection in an infected file. Our proposed Markov n-gram detector has significantly better detection rate than exiting detectors. Moreover, due to its ability to identify the location of an embedded malware, the proposed detector can provide very low false positive rates when used in conjunction with existing COTS AV software.

Acknowledgments. We acknowledge the help of Syeda Momina Tabish and Ch. Junaid Anwar in result collection and data set generation.

This work is supported by the National ICT R&D Fund, Ministry of Information Technology, Government of Pakistan. The information, data, comments, and views detailed herein may not necessarily reflect the endorsements of views of the National ICT R&D Fund.

References

1. Symantec Internet Security Threat Report XI: Trends for July – December 2007, September 2007.
2. S. J. Stolfo, K. Wang and W. J. Li, Towards Stealthy Malware Detection, *Advances in Information Security*, Volume 27, pp. 231-249, Springer, 2007.
3. W. J. Li, S. J. Stolfo, A. Stavrou, E.Androulaki and A. D. Keromytis, *A Study of Malcode-Bearing Documents*, DIMVA, 2007.
4. John Leyden, *Trojan exploits unpatchedWord vulnerability* The Register, May 2006.
5. Joris Evers, *Zero-day attacks continue to hit Microsoft*, News.com, September 2006.
6. David Kierznowski, *Backdooring PDF Files*, September 2006.
7. M. Damashek, *Gauging Similarity with n-Grams: Language-Independent Categorization of Text*, 267, pp. 842-848, Science, 1995.
8. S. Friedl, *Steve Friedl's Unixwiz.net Tech Tips: Analysis of the new 'Code Red II' Variant*, Webpage, `http://www.unixwiz.net/techtips/CodeRedII.html`.
9. A. Lakhina, M. Crovella, and C. Diot, *Mining anomalies using traffic feature distributions*, ACM SIGCOMM, September 2005.
10. T. M. Cover and J. A. Thomas, *Elements of Information Theory*, Wiley-Interscience, June 1991.
11. *VX Heavens Virus Collection*, VX Heavens, `http://hvx.netlux.org/vl.php`
12. *McAfee Anti-virus and Internet security*, McAfee, CA.
13. *Kaspersky Antivirus*, Kaspersky Lab HQ, Moscow, Russia.
14. *AVG Anti-Virus and Internet Security*, GRISOFT Inc., FL, USA.
15. Yinrong Huang, *Vulnerabilities in Portable Executable (PE) File Format For Win32 Architecture*, TR, Exurity Inc., Canada, 2006.
16. T. Fawcett, *ROC Graphs: Notes and Practical Considerations for Researchers*, TR, HP Labs, CA, 2003-4, USA.
17. D. Wagner and P. Soto, *Mimicry Attacks on Host-Based Intrusion Detection Systems*, ACM CCS, Nov. 2002.