Embedded Malware Detection using Markovian Statistical Model of Benign Files ⋆

.

M. Zubair Shafiq, Syeda Momina Tabish and Muddassar Farooq

Next Generation Intelligent Networks Research Center (nexGIN RC)
National University of Computer & Emerging Sciences (NUCES)
Islamabad, Pakistan.

Syed Ali Khayam

School of Electrical Engineering & Computer Science (SEECS)
National University of Sciences & Technology (NUST)
Rawalpindi, Pakistan.

Abstract

Embedded malware is a recently discovered security threat that allows malcode to be hidden inside a benign file. This hidden malcode can be remotely executed using a simple trigger program. It has been shown that embedded malware is not detected by commercial antivirus software even when the malware signature is present in their antivirus database. Therefore, embedded malware can be, more suitably, termed as a 'disaster in disguise'. 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 thousand 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.

Key words: Embedded Malware, Markov Chain, n-Gram
PACS:

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 the 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

⋆ This paper is an extended version of our earlier work, [7], available at `http://www.springerlink.com/content/978-3-540-70541-3/`

Email addresses:
`zubair.shafiq,momina.tabish,muddassar.farooq@nexginrc.org` (M. Zubair Shafiq, Syeda Momina Tabish and Muddassar Farooq), `khayam@niit.edu.pk` (Syed Ali Khayam).

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 `JPG`, `MP3`, `ZIP`, `EXE`, `PDF` and `DOC` 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. In this study, we also explore this proposition.

Organization of the Paper. The rest of the paper is organized as follows. We discuss in detail the infected datasets created for this study in Section 2. We present realistic attack scenarios, using different filetypes, for embedded malware in Section 3. In Section 4, we propose our Markov n-gram detector and in Section 5 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 6, we discuss the limitations of the proposed Markov n-gram detector. We provide an overview of related research in the field of embedded malware detection and n-gram based techniques in Section 7. Finally, we conclude the paper with an outlook to our future work.

2 Dataset

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

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

Table 1
Statistics of Benign Files used in this Study

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

any random location within a benign file. Using this tool, we produce a large embedded malware dataset [2].

2.1 Benign Dataset

The benign dataset for our experiments consists of six different filetypes: JPG, MP3, ZIP, EXE, PDF and DOC. 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 JPG, MP3, and ZIP 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].

[2] Some characteristic samples of the 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** (kilo-bytes)	**Minimum Size** (bytes)	**Maximum Size** (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

2.2 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' [14] 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.

2.3 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. The algo-

Table 3
`NERGAL` Algorithm

input: benign file, malicious payload
output: infected file with embedded malware

Read benign file in a byte array C
Read malicious payload in a byte array M
Create a infected-data byte array S such that $|S| = |C| + |M|$, where $|X|$ is the size of array X
Copy C's header to S: $S[1 \dots h] \leftarrow C[1 \dots h]$, where h is the size of $|C|$'s header
Generate a random number r: $h + 1 \leq r \leq |C|$
$S[h + 1 \dots r] \leftarrow C[h + 1 \dots r]$
$S[r + 1 \dots r + |M|] \leftarrow M[1 \dots |M|]$
$S[r + |M| + 1 \dots |C| + |M|] \leftarrow C[r + 1 \dots |C|]$

Write S to the infected file

rithm of `NERGAL` is shown in Table 3. `NERGAL` ensures that the infections are inserted after the header of the benign files to avoid filetype 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 simply 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 statistical 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.

3 Attack Scenarios

In this section, we will discuss potential real-world attack scenarios that can be realized using embedded malware. The potential threat of the embedded malware depends on its stealthiness. There are two possible ways in which its stealthiness can be compromised: 1) detection by anti-virus software, and 2) error message on opening or degraded quality. As explained by the authors in [3], and independently verified by us, embedded malware cannot be detected

Table 4
Results for opening

Cannot Open without Error Message	Cannot Open with Error Message	Open with Error Message	Open with Degraded Quality	Open without Error Message
JPG				
10.0%	0.0%	0.0%	88.3%	1.6%
MP3				
0.0%	0.0%	0.0%	0.0%	100.0%
ZIP				
1.1%	3.7%	47.9%	47.4%	0.0%
EXE				
7.0%	80.0%	1.0%	N/A	12.0%
PDF				
2.4%	4.9%	74.1%	6.5%	12.2%
DOC				
0.0%	89.5%	9.6%	0.0%	0.9%

by COTS AV software. Similarly, '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 without any error message or degradation in quality. To this end, we manually opened the $6,000$ infected files and observed how many of them open correctly after naive (i.e., completely random) malware embedding. The infected files followed various trends upon opening; some files opened with or without error messages, other did not open at all, and still some other opened with degraded quality. The opening of the infected files revealed interesting insights for different filetypes whose stats are summarized in Table 4[3]. We will now discuss these insights for every filetype separately.

3.1 JPG files

Most of the JPG files are previewed after being infected as shown in Figures 1(b), 1(d) and 1(d); only 10% files are not opened. Most of the infected files have not incurred any distortion as shown Figure 1(b) and (d). In some cases, such as Figure 1(f), some distortion is visible in the infected files. Here we highlight that this image distortion is introduced because we are randomly embedding malware inside the JPG files without any regard to the inherent structure of the benign file's content. However, sophisticated watermarking techniques can be used to minimize the distortion due to malware embedding. Furthermore, we highlight that an automated detector that does not understand the file's semantics and the original file will not be able to understand

[3] The category of 'degraded quality' is not applicable for EXE files.

(a) Original Image

(b) Image after Malware Embedding

(c) Original Image

(d) Image after Malware Embedding

(e) Original Image

(f) Image after Malware Embedding

Fig. 1. JPEG files before and after malware embedding.

or quantify this distortion in the image quality.

3.2 MP3 files

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 messages or perceptual distortion in audio quality*. This came as a surprise to us but we found out that it 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. Since most Internet song sharing portals use the `MP3` file format, an attacker can use simple 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.

3.3 ZIP files

The `ZIP` files are stored in an object-wise manner. Every file in the overall `ZIP` file has a separate header. This protects the complete `ZIP` file from getting corrupted. 95% of the `ZIP` files open with an error message because of failed CRC checksum. However, they extract successfully without any further errors.

3.4 EXE files

The Microsoft Portable Executable (PE) file has several well-known vulnerabilities [18]. A random code can be embedded in free spaces between different sections or in the padding portions of the file without violating the integrity of executable file. The author in [18] has also described a number of intelligent embedding techniques which can automatically execute the embedded code once the infected file is executed. Our results show that with random embedding, 12% of the files can open without any error message. Since the embedded files are undetected by COTS AV software, an attacker can embed malware inside common benign files – for instance, `WINWORD.EXE` – and the infected file will go unnoticed through the COTS AV software deployed inside the network or on the host. Such infected file 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]. It should be noted that 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.

3.5 PDF files

The `PDF` files have an object-oriented encoding format. The random infection only corrupts a particular object and other objects remain intact. Adobe reader notices the corruption due to the infection but the files still open. Most

of the PDF files open with or without error message. The error messages typically appear when we scroll down to the document page where the infection is introduced. PDF is a modern document format with 'object oriented dynamic composability' which 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].

3.6 DOC files

The DOC file format also supports embedded objects. However, 90% of the files displayed an error message on double clicking and could not be opened. The authors in [3] have shown that intelligent embedding in DOC files can add embedded objects to a DOC file that can be executed from the DOC file editor. Similar to PDF, DOC is also a modern document format which allows the user to include embedded objects which can be invoked by deceiving a naive user.

4 Modeling and Quantification of n-gram Information

In this section, at first, we present pilot experimental studies carried out to evaluate previously proposed schemes for embedded malware detection. Then we analyze byte-level correlation structure present in files which leads us to a statistical model (Markov n-gram) of benign byte sequences. We quantify perturbations in this statistical model using a well-known information-theoretic measure (entropy rate). Entropy rate thresholds are selected for different file-types using the Gaussianity properties of *the sum of sampled entropy rate distributions* of benign files.

It should be noted 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.

4.1 Mahanalobis n-gram Detector

In pilot studies, we build on the Mahanalobis n-gram detector, the only scheme designed specifically to detect embedded malware, proposed by the authors 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.

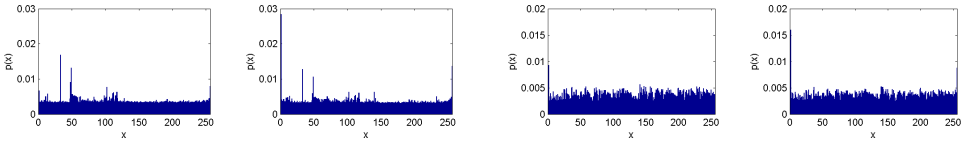
One major assumption of the prior study was that the infection appears only at the start or the end of the benign file [2]. Therefore, n-gram analysis was applied only on the truncated files [2]. We argue that this assumption is unrealistic because it is not capable of detecting embedded malware in the 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 on a specific location [4]. Mahanalobis n-gram detector provides reasonable detection accuracy when the infection appears at the start or the end of a file. However, it cannot provide location identification. We hence revoke the assumption of file truncation and compute n-grams on whole files.

Figures 2(a) and 2(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 2(a) and 2(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.

The authors in [2] also 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 3 shows some representative results of the Mahanalobis distance between the block-wise 1-

[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) 1-gram histograms of benign and infected `PDF` file

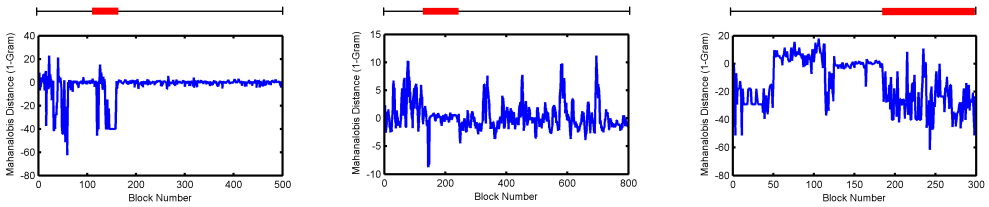
(b) 1-gram histograms of benign and infected `JPG` file

Fig. 2. Comparison of 1-gram histograms of benign and infected files.

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 3 that the block-wise 1-gram Mahanalobis distance does not provide significant perturbations that could help in detecting 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 was 1000 bytes. These experiments revealed that, despite the increased computational complexity, the performance of the Mahanalobis distance based detector did not improve significantly. 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.

We also tried the *entropy* measure instead of Mahanalobis distance, but it also failed to accurately detect embedded malware [12], [13]. From the pilot studies, we can conclude that the Mahanalobis distance or entropy of simple 1- and 2-gram distributions cannot accurately detect embedded malware. This shows that a simple n-gram distribution does not provide sufficient information to detect embedded malware. To rectify this shortcoming in the n-gram distributions, in the following we provide a different method of computing n-grams in the following sections.



(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. 3. 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.

4.2 Correlation Structure in File Data

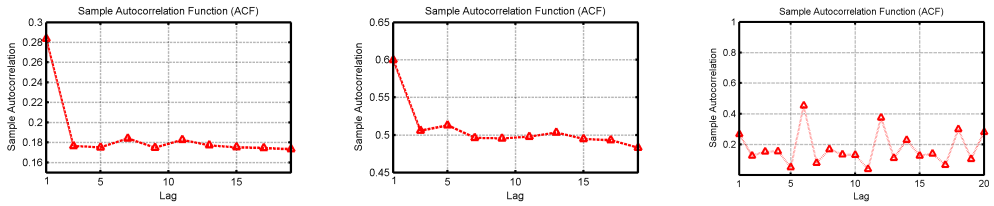
We first note that the 2-gram distribution is in fact the joint distribution of two 1-gram symbols. This joint distribution may contain some redundant information which is not pertinent to the present embedded malware detection problem. For accurate detection, it is important that this redundancy is 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.

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:

$$\rho[k] = \frac{E\{X_0 X_k\} - E\{X_0\}E\{X_k\}}{\sigma_{X_0}\sigma_{X_k}}, \tag{1}$$

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



(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.

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 [11]:

```
GET /default.ida?XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX%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.

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

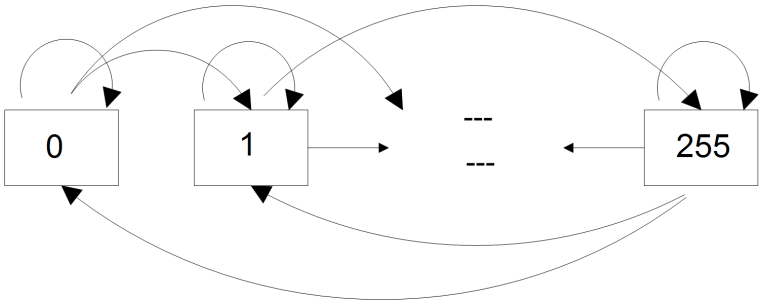


Fig. 5. 1-st order, 256 state, Markov Chain.

if an attacker allows 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.

4.3 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 [13]. 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 as shown in Figure 5. 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.

4.4 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 [13]. Entropy rate for a sequence of discrete finite random variables X_1, X_2,..., X_n is defined as:

$$R = \lim_{N \to \infty} \frac{H(X_1, X_2, ..., X_n)}{N}, \tag{2}$$

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 R can be defined in terms of entropy. 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:

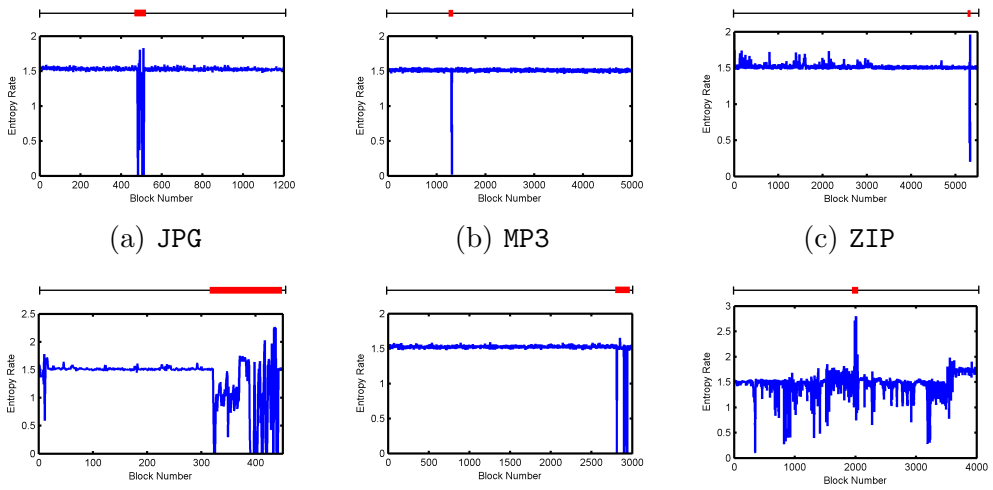
$$H(X) = -\sum_{i \in \Delta_n} p(x_i) \log_2 p(x_i). \tag{3}$$

Further,

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

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 [13]. 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.



(a) JPG (b) MP3 (c) ZIP

(d) EXE (e) PDF (f) DOC

Fig. 6. Entropy Rate of infected files. The horizontal thick bars show the location of the embedded malware.

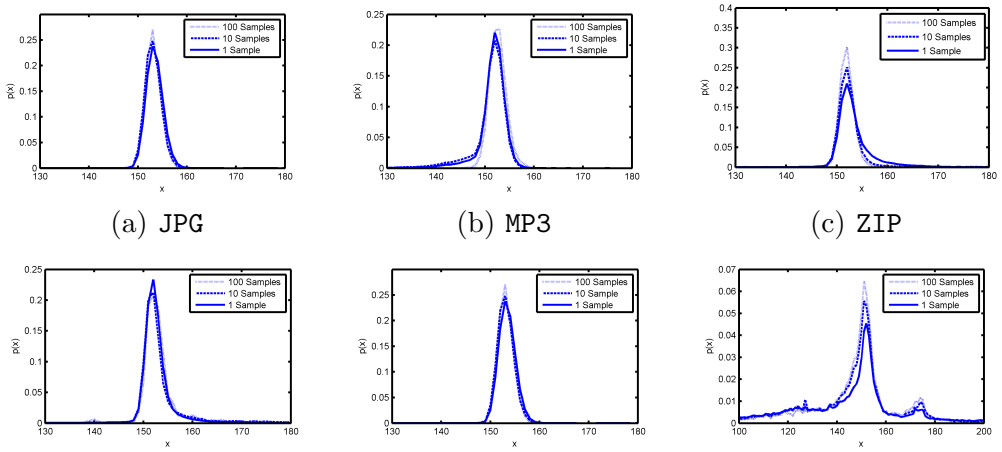
Since we rely on the premise that the statistical properties of the embedded malware will be different from the statistical properties of the benign file in which it is embedded, *expected entropy* of the consequent Markov chain (derived from the infected file) should be perturbed at the embedding locations.

Figure 6 shows the entropy rate of infected files of every filetype used in our study. It is clear from Figure 6 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 6 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 an infection would be detected. The following section provides a flexible yet accurate method of defining this threshold.

4.5 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



(a) JPG (b) MP3 (c) ZIP

(d) EXE (e) PDF (f) DOC

Fig. 7. Sampled entropy rate distributions for different filetypes.

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 7 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 [19]. 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 7. DOC filetype shows worst convergence to Gaussianity, i.e., significant amount of area is present at the fringes (see Figure 7(f)). This logically leads us to set a relatively higher value of the threshold (k) for DOC files.

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.

5 Classification Results

As mentioned in Section 2, we perform classification on two infected datasets, each consisting of $224,520$ infected files created by infecting benign files of six common types: `JPG`, `MP3`, `ZIP`, `EXE`, `PDF`, `DOC`. For the training of our proposed scheme, we use 5% of the benign dataset.

Table 5 provides the detection rates of 3 fully updated [6] commercial antivirus products: McAfee Antivirus [15], AVG Antivirus [16] and Kaspersky Antivirus [17], Mahanalobis n-gram detector by the authors in [2] and our Markov n-gram detector [7]. We emphasize that embedded malware's signatures are present in the COTS antivirus' databases i.e., the malware are detected as standalone files or exploit codes by the COTS antivirus software. The results tabulated in Table 5 reaffirm that: *commercial antivirus products are not effective in detecting embedded malware*. Moreover, and as expected, the detection rate for all COTS AV software degrade to 0% for the encrypted dataset. Thus, after simple substitution-type encryption, COTS antivirus completely fail to detect the embedded malware in the infected files, irrespective of the type of file being infected. In the un-encrypted infected dataset, some of the embedded malware are detected by COTS antiviruses. An important observation from the experiments is that *the malware detected by COTS antiviruses are mostly embedded at the start of the infected files*. It can also be observed from the Table 5 that code embedding inside `EXE` and `PDF` files is detected more often by COTS than other filetypes. While the detection algorithms of these COTS antivirus software are proprietary, we can deduce some properties of these algorithms based on our experimental results. For instance, it is obvious that `EXE` files are scrutinized/scanned more thoroughly than other filetypes. This must be because `EXE` files are stand alone, execution-ready files which can be

[6] by January 2008.

[7] 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.

Table 5
Detection (TP) rate and False Positive (FP) rate of Antivirus Software and Anomaly Detectors

	McAfee Antivirus [15]	**AVG Antivirus** [16]	**Kaspersky Antivirus** [17]	**Mahanalobis n-gram Detector**	**Markov n-gram Detector**	**Percentage Improvement**
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 `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%**
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 `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 `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%**

triggered without requiring any additional software. Therefore, most contemporary malware are bundled in executables files. It is for the same reason that

email filters and network firewalls generally remove executable attachments. Similarly, we infer that high detection rates for `PDF` files are due to the small sizes of these files. We argue that antivirus scanners generally scan a small portion of a file's payload. Infection is detected only if a known malware signature is present in that portion of the file. It can be said to evade detection by COTS antivirus malicious payload should be embedded towards the end of the file. 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.

The reason for less than 100% detection rate of Markov n-gram detector can be traced back to our earlier comment: 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 ([14]) 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 5 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. This 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. As our earlier 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.

6 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 signature-based detector. 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 signature-based detector can then scan the new (extracted) file for the presence of a known malcode signature. Clearly, this hybrid detection strategy can 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. Our attempts to use a hybrid of COTS AV software and Markov n-gram detector were not successful. Our investigation shows that the extracted portion of the file still remains an embedded malware for COTS AV software because perfect alignment of the extracted portion cannot be achieved due to naivety of our extraction algorithm. This problem can be solved in two ways: 1) development of intelligent extraction algorithms to

embark *exact* location of embedded malware so it can be detected by COTS AV software, and 2) development of an in-house signature based detector enabled with deep inspection. Second proposition is not in the scope of current work, however, we plan to pursue the first proposition as our future work.

- One form of the embedded malware discussed in this paper is dormant (see Section 3 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 [20] 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 [20].

7 Related Work

A significant amount of research effort has recently been focused towards malware detection. To maintain focus and logical flow of thought, in this section, we describe only the approaches that target embedded malware detection or n-gram based malware detection.

- 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 filetypes 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.
- Kolter et al. present a hybrid approach to detect malicious executables in the wild [8]. They use n-gram analysis to extract features from 1971 benign and 1651 malicious samples. The most relevant n-grams (4-grams in the paper) were selected for prediction by computing the information gain. A variety of inductive learning methods were used, such as naive Bayes, decision trees, support vector machines and boosting. Their results show that the boosted decision trees outperform rest of the classifiers.
- In [9], the authors have proposed a hybrid approach for detection of new malicious code. They use n-gram analysis to automatically generate virus signatures from a given sample of malware and benign programs. Authors have shown that the n-gram signatures are more robust as compared to the traditional signature based schemes.
- Henchiri et al. propose a data mining approach to generate robust signatures for computer viruses [10]. They claim to find robust signatures using n-gram analysis. They carried out their analysis on different families of viruses. A subset of most frequent n-grams were chosen from each virus family. Feature

elimination was carried out on the collected n-grams to get the final set of features that were the eventual signatures. Authors used various machine learning algorithms on the top of these features for final classification. They carried out their tests on a dataset of 1512 viruses and 1488 benign files and achieved up to 93% overall accuracy.

8 Conclusions

In this paper, we have 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 appropriately used in conjunction with a signature-based detector.

Acknowledgments.

We acknowledge the help of Ch. Junaid Anwar in dataset 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 US, 2007.

[3] W. J. Li, S. J. Stolfo, A. Stavrou, E.Androulaki and A. D. Keromytis, *A Study of Malcode-Bearing Documents*, Detection of Intrusions and Malware & Vulnerability Assessment (DIMVA), Volume 4579 of Lecture Notes in Computer Science, pp. 231-250, Springer Verlag, Zurich, Switzerland, 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. Zubair Shafiq, S. A. Khayam, M. Farooq, *Embedded Malware Detection using Markov n-grams*, Detection of Intrusions and Malware & Vulnerability Assessment (DIMVA), Volume 5137 of Lecture Notes in Computer Science, pp. 88-107, Springer Verlag, Paris, France, 2008.

[8] J. Z. Kolter, M. A. Maloof, *Learning to Detect and Classify Malicious Executables in the Wild*, Journal of Machine Learning Research, Volume 7, pp. 2721-2744, 2006.

[9] T. Abou-Assaleh, N. Cercone, V. Keselj, and Ray Sweidan, *Detection of New Malicious Code Using N-grams Signatures*, Advances in Soft Computing Journal, Springer-Verlag, 2003.

[10] O. Henchiri and N. Japkowicz, *A Feature Selection and Evaluation Scheme for Computer Virus Detection*, IEEE International Conference on Data Mining, 2006.

[11] 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`.

[12] A. Lakhina, M. Crovella, and C. Diot, *Mining anomalies using traffic feature distributions*, ACM SIGCOMM, September 2005.

[13] T. M. Cover and J. A. Thomas, *Elements of Information Theory*, Wiley-Interscience, June 1991.

[14] *VX Heavens Virus Collection*, VX Heavens, `http://hvx.netlux.org/vl.php`

[15] *McAfee Anti-virus and Internet security*, McAfee, CA.

[16] *AVG Anti-Virus and Internet Security*, GRISOFT Inc., FL, USA.

[17] *Kaspersky Antivirus*, Kaspersky Lab HQ, Moscow, Russia.

[18] Yinrong Huang, *Vulnerabilities in Portable Executable (PE) File Format For Win32 Architecture*, TR, Exurity Inc., Canada, 2006.

[19] T. Fawcett, *ROC Graphs: Notes and Practical Considerations for Researchers*, TR, HP Labs, CA, 2003-4, USA.

[20] D. Wagner and P. Soto, *Mimicry Attacks on Host-Based Intrusion Detection Systems*, ACM CCS, Nov. 2002.