
Eric Filiol
Operational Cryptology and Virology Laboratory
ESIEA Group / ESIEA Research
Laval, France
eric.filiol@esiea.fr

Sébastien Josse
Operational Cryptology and Virology Laboratory
ESIEA Group / ESIEA Research
Laval, France
sebastien.josse@esiea.fr

Abstract—Statistical methods have been used for a long time as a way to detect viral code. Such a detection method has been called spectral analysis, because it works with statistical distributions, such as bytes, instructions or system calls frequencies spectra. Most statistical classification algorithms can be described as graphical models, namely Bayesian networks. We will first present in this paper an approach of viral detection by means of spectral analysis based on Bayesian networks, through two basic examples of such learning models: naive Bayes and hidden Markov models.

Designing a statistical information retrieval model requires careful and thorough evaluation in order to demonstrate the superior performance of new techniques on representative program collections. Nowadays, it has developed into a highly empirical discipline. We will next present information theory based criteria to characterize the effectiveness of spectral analysis models and then discuss the limits of such models.

Index Terms—Bayesian Network, Naive Bayes, Hidden Markov Model, Spectral analysis

I. INTRODUCTION

Statistical methods have been used for a long time as a way to detect viral code. In this paper, we will focus on such a detection method that has been called spectral analysis, because it works with statistical distribution. Spectral analysis may apply to the byte content distribution, the statistical distribution of instructions, the API call sequences or even time or memory consumption spectra.

More recently, data mining techniques, well known for their applications in other research fields, such as genetic programming, speech recognition or text classification have been applied in several security research areas: cryptanalysis, spam filtering and viral detection. Most of these statistical classification algorithms can be described as graphical models, namely Bayesian networks. We will first present in this paper an approach of viral detection by means of spectral analysis based on Bayesian networks, through two basic examples of such learning models: naive Bayes and hidden Markov models.

While evaluating the security of an anti-virus product, a first mandatory stage is to analyze the design and specification documentation. Indeed, this documentation provides the required material for a theoretical analysis of the workings of the anti-virus software, particularly its security functions and interfaces. In order to perform the security evaluation tasks, we need to use both technical and theoretical tools. The purpose of this paper is to discuss the tools that an evaluator has at its disposal to conduct a theoretical analysis of a spectral analysis based virus detection engine’s efficiency.

We have proposed in our previous works [1] a statistical characterization of antivirus detection, providing a statistical variant of Cohen’s undecidability results of virus detection. In this framework, a detection technique is formalized as a set of statistical tests. We have then introduced the concept of statistical testing simulability, which may be defined as a way for an attacker to evade detection by using to his advantage the intrinsic flaws of a detection model or of its parameters. The general concept of testing simulability covers such techniques that intend to make viral code resistant to static content anomaly detectors. This general concept naturally leads to security criteria that may apply to spectral analysis based virus detection. We propose in this paper a characterization of the Bayesian network based models through precise criteria (soundness, completeness, robustness, complexity), for judging the appropriateness of a detection engine design.

1 Following the terminology proposed in [2], we define a detection scheme by the pair \((S_M, f_M)\) consisting of a detection pattern \(S_M\) and a detection function \(f_M\). We will see in the next section how the concept of detection scheme is expressed and more generally how we can formalize the problem of virus detection in a statistical framework.
and discuss the limits of these model (intrinsic limits, simulability, compositionality) and the required trade-off that they induce. This paper provides as a result a general methodology, a refined theoretical framework and precise criteria, based on measures that come from information theory, which can be used to assess the security of spectral analysis based detectors, from its specification and design documentation.

Related works

The testing simulability problem has already been studied by the research community, mainly with regards to the n-gram bytes distribution detection profiles, in the context of intrusion detection. Indeed, various attack mutation techniques have been experimented to evade IDS (intrusion detection systems). They have been called Mimicry [4] or Blending attacks [5], [6] and both try to evade IDS by modifying the attack characteristics so that it matches a normal profile, corresponding to a benign behavior. The former applies mainly to host-based IDS, whereas the latter applies to network-based IDS. The ASC engine [7] is probably the first one to explicitly use the knowledge of the detection model used in the IDS to evade detection, by showing how to perform alphanumeric encoding to pass through the subsequent filtering rules imposed by the detector. More recently, the CLET mutation engine [8] provides a way to take advantage of the knowledge of more sophisticated detection models used in the IDS to evade detection. Namely, CLET injects polymorphic shellcode into a vulnerable target process and introduces many innovations to defeat data mining methods, such as the neural approach or the n-gram content distribution based classifier.

Otherwise, the work described in [9] provides statistical measures that enables the qualification of mutation engine in terms of variation and propagation strengths. Those second order metrics may be sufficient to evaluate the strength of many statistical detection models, but may not apply to more sophisticated models, capable of capturing higher order information. Moreover, such metrics have been studied in the context of byte content spectral analysis.

Several works investigate the more semantically informational material of assembler instructions, that is less studied but may be used by real-time antivirus programs. HMM-based detection methods have already been the object of experimentations on the viral families generated by using virus generator kits G2, MPCGEN, NGVCK and VCL32 [10], [11]. In [10], the authors use a similarity score, called LLPO (Log Likelihood Per Opcode), corresponding to the log-likelihood of an observation Y, given the model, divided by the size |Y| of the observation. They were mainly interested in the discriminating power of such a model, by comparing the LLPO score with another similarity index, proposed by Mishra in [12].

More recently, [13] explores whether there are any exploitable weaknesses in this HMM-based detection approach or not. The author improves a metamorphic engine by inserting instructions sequences extracted from benign files to increase the similarity between the obtained metamorphic virus and normal programs in order to evade the HMM-based detection approaches proposed in [10], [11]. In short, the author proposes a metamorphic virus generating tool specifically designed to evade HMM-based detection. The principle is to make each distinct viral copy similar to a randomly selected normal file.

As a matter of fact, such experiments exploit weaknesses in the detector, but do not provide explanation. We investigate in this paper the theoretical criteria and models that may be used to explain such empirical results and, as we expect, provide a method and some theoretical tools to strengthen statistical model-based detectors.

Organization of the paper

The rest of this paper is organized as follows: section II will introduce Bayesian network based detection models and we will see how the concept of detection scheme is expressed and more generally how we can formalize the problem of virus detection in this theoretical framework; section III proposes a characterization of these models through precise criteria; section IV discusses their limits and the required trade-offs that they induce; section V concludes this paper with an overview of the main remaining technical and theoretical open problems and future works.

II. STATISTICAL DETECTION MODEL

Any evaluation of the detection engine based on objective criteria requires a modeling effort. Let us see how the concept of detection scheme is expressed and more generally how we can formalize the problem of virus detection in this theoretical framework.

The statistical modeling of the detection problem provides additional insight into and applies to the analysis of viral behavior, on the basis of statistical information. With such a model, we can capture the aspects of the program interactions with its environment. In this theoretical framework, a virus detection scheme can be given [1], [14] by:

- a probability law distribution characterizing the information or a model \( \lambda_M \) formed on training data (i.e. the model parameters are estimated on the training data);
- a decision rule generated on the basis of the training data, making it possible to evaluate the likelihood of an observation, given the model \( \lambda_M \).
With this formalism (the one of combinatorial or probabilistic models), the detection problem reduces to the problem of the likelihood of an observation, given a model $\lambda_M$. The detection function $f_M$ is a classification algorithm or a test, characterized by a threshold, making it possible to recognize the fact of being governed by a law or a model $\lambda_M$, the latter defining the detection pattern $S_M$.

As an example, we make our choice naturally for the Bayesian networks, due to their frequent use in filtering and intrusion detection systems.

Bayesian networks correspond to a particular type of graphical model, of which we recall here the principle.

The graphical models are very practical to describe the conditional independence and its consequences. This abstraction makes it possible to represent a big number of statistical ideas. More precisely, a graphical model is defined as a graph $\mathcal{G} = (V, E)$ where $V$ is a set of nodes and the set of edges $E$ is a subset of $V \times V$.

A given graphical model is associated with a collection of random variables and with a probability distributions family over this collection. The set of nodes $V$ corresponds exactly to the random variables, while the edges represent the properties of conditional independence of the random variables which are true for all the elements of the probability distributions family that are associated to.

In this section and the next, we are taking an interest in a peculiar type of oriented graphical model (namely where the edges are oriented) so called Bayesian network. We obtain a graphical representation of conditional independence relations by applying the following property, so called oriented local Markov property: a variable is conditionally independent of its non-descendants, knowing its immediate parents. As an example, the following Bayesian network:

$$
X_0 \rightarrow X_1 \rightarrow \cdots \rightarrow X_{t-1} \rightarrow X_t \rightarrow \cdots \rightarrow X_{T-2} \rightarrow X_{T-1}
$$

represents a Markov chain of order 1: the random variable $X_t$ is conditionally independent of the variables $(X_s)_{s \leq t-1}$, knowing $X_{t-1}$.

We will present in the section II-A the naive Bayes model, which corresponds to the following Bayesian network:

In this model, conditional on the random variable $X_t$, the random variable $Y_t$ is independent of $\{X_{-t}, Y_{-t}\}$.

In a graphical model, random variables can be either hidden, or observed. In the first case their values are unknown. They are supposed to be really random variables. In the second case, their value is known. We generally note $X$ the hidden random variables and $Y$ the observations.

These two models are the simplest examples of Bayesian networks. They are furthermore widely used in the filtering and intrusion detection systems.

In the spectral analysis context, each model $\lambda$ is designed to store information to summarize or compress the characteristic of a mutation engine. It is expected that during the training phase of a model, the parasitical information, eg resulting from the application of obfuscation transformations are not taken into account in the characterization of a viral family or that they are taken into account in a manner that does not interfere with detection.

Concerning the HMM model, some of the information stored in an HMM relates to the hidden Markov chain, which we hope will contain, after the training phase, a synthetic information on the virus mutation engine so we can recognize all programs resulting therefrom. One of the interests of this type of model, as compared with the more simple naive Bayes model, is that it makes it intrinsically possible, and computationally in an efficient way, to recover the states sequence $X$ and information on the model structure, from the given model $\lambda$ and the observed sequence $Y$. Therefore, it seems to be possible to characterize a model on the basis of structural and qualitative information. We expect that from the hidden part of a model, we can compare two models to each other, based on specific criteria.

The detection procedure which is based on the use of Bayesian network based models can be specified in a very similar way than on pattern matching-based approach: given a set $\lambda_1, \ldots, \lambda_n$ of Bayesian network based models, and a code $Y$, we calculate for each model $\lambda_i$ the likelihood of the observed sequence $Y$, given the model. In the domain of intrusion detection, such an approach is said to be based on knowledge. If the likelihood exceeds a certain threshold $T$, the program $Y$ is regarded as belonging to the viral family $\lambda_i$. 
A. Naive Bayes test specification

We present in this section a classical method to produce a decision rule on the basis of training data. This method, so called naive Bayes (NB) classification due to the very strong (naive) conditional independence hypothesis on which it is based, is implemented in much software dedicated to filtering and intrusion detection. It is very simple to implement and the computing complexity of the algorithm is optimal with regards to the other classification methods [15]. We present its application in the context of spectral analysis. A Bayesian test can be built on the basis of this classification method.

The naive Bayes algorithm assumes that there exists a generation model for executables: they are produced by a blending model, the components of which are the categories of executables:

\[ c_j \in \mathcal{C} = \{c_1, \ldots, c_{|\mathcal{C}|}\}. \]

These latter are hidden variables, insofar as they are not observed. We denote \( X \) the random variable which represents the category of executables. The generation of an executable by this model is done as follows:

- the choice of an executable class \( X = c_j \);
- the generation of the executable \( Y_i \) on the basis of a set of words (typically the compiler's instruction set \( \mathcal{W} \)), with parameters which vary according to the class of executable that has been chosen.

The algorithm is said to be « naive » because it relies on a strong hypothesis: the occurrence of an instruction \( w_i \) in the program is supposed to be independent of the presence of the other instructions. In addition, the positions of the instructions in the program and their arrangement relative to each other are not taken into account.

We can identify two classical variants of the naive Bayes algorithm, which correspond to two different generative models:

- the multivariate Bernoulli model ;
- the multinomial model.

Both variants are described and compared in [16], where they are applied to the classification of documents. We present their use in the context of spectral analysis.

An executable generated by the multivariate Bernoulli model is characterized by the presence or the absence of the words \( w_i \) of the instruction set \( \mathcal{W} \): it can be characterized by a binary vector \( b = (b_1, \ldots, b_{|\mathcal{W}|}) \), with \( b_i = 1 \) if the instruction \( w_i \) is present in the executable and \( b_i = 0 \) otherwise.

An executable generated by the multinomial model is characterized by the instructions of the program and also includes their number of occurrences.

In both cases, we use the training data to form a decision rule for, from the spectral characteristics extracted from a program, deciding its class membership. Given a program \( d \), we determine the most probable/likely class by computing:

\[
g(w_1, \ldots, w_{|\mathcal{W}|}) = \arg\max_c p(X = c) \prod_{i} p(W = w_i \mid X = c). \]

We can build a Bayesian test from the classification algorithm corresponding to the multinomial generative model, in the case where \( |\mathcal{C}| = 2 \). Let \( c \) and \( \neg c \) be the two corresponding classes of programs. Typically, in the context of spectral analysis, the class \( X = c \) corresponds either to programs that are produced from a virus generator kit (or a polymorphic shellcode generator) or to a viral code family using the same mutation engine. The decision rule is then as follows for the file being analyzed: if the log-likelihood ratio:

\[
\ln \left( \frac{p(X = c|Y = d)}{p(X = \neg c|Y = d)} \right) > 0, \]

then we reject the null hypothesis \( H_0 \): the file is probably infected by a virus of the family \( c \). Otherwise, we cannot reject the null hypothesis: the file is either benign, or infected by a virus of another viral family.

This testing methodology is very simple, but in practice leads to a too high false alarm rate. To remedy this, a first step aims at calibrating the detector on the test data. Consider a given viral population \( \mathcal{V} \), and a set of benign programs \( \mathcal{B} \). We form several subsets from \( (\mathcal{V}, \mathcal{B}) \):

- subsets \((\mathcal{V}_1, \mathcal{B}_1)\), with \( \mathcal{V}_1 \subset \mathcal{V} \) and \( \mathcal{B}_1 \subset \mathcal{B} \) are the training data ;
- subsets \((\mathcal{V}_2, \mathcal{B}_2)\), with \( \mathcal{V}_2 \subset \mathcal{V} \) and \( \mathcal{B}_2 \subset \mathcal{B} \) are the test data.

We place ourselves in the case where our model is trained only on viral training data. We use the subset \( \mathcal{V}_1 \) to estimate the probabilities \( p(X = c) \) and \( p(W_i = w_i \mid X = c) \). We then use our model to compute the probability that a program of the set \( \mathcal{V}_2 \cup \mathcal{B}_2 \) belongs to the same programs family that the training data. Having computed these likelihoods, we can empirically determine a threshold \( S \) for which classification operates without error.

We can now form two additional subsets \( \mathcal{V}_3 \subset \mathcal{V} \) and \( \mathcal{B}_3 \subset \mathcal{B} \) that will allow us to calculate the type I and II errors which characterize our probabilistic detector.

The benefit of this approach is that each model can be associated with a virus family, for example resulting from the use of a virus generator kit or a polymorphic shellcode generator. Unlike an approach by pattern matching, a single properly calibrated model can be applied to all variants of a family.
B. Hidden Markov Model test specification

Definition 2.1: (Hidden Markov Model [17]). A Hidden Markov model is defined by \( \lambda = (A, B, \alpha(0)) \) where \( A = (a_{ij})_{0 \leq i \leq N-1, 0 \leq j \leq N-1} \) is the transition matrix of a Markov chain (of order 1) with \( a_{ij} = P(X_{n+1} = x_j | X_n = x_i) \); \( B = (b_{jk})_{0 \leq j \leq N-1, 0 \leq k \leq M-1} \) is a matrix \( N \times M \) with \( b_{jk} = b_j(k) = P(Y_n = k | X_n = x_j) \) and \( \alpha(0) \) is the initial distribution of \( X_0 \).

Consider the hidden chain of length \( T; X = (X_0, \ldots, X_{T-1}) \) and the corresponding observations \( Y = (Y_0, \ldots, Y_{T-1}) \). We have:

\[
P(X = (x_0, \ldots, x_{T-1})) = a_{x_0}(0)b_{x_0}(0)a_{x_0x_1,b_{x_1}}(1) \cdots a_{x_{T-2}x_{T-1},b_{x_{T-1}}}(T-1)
\]

We can identify three fundamental problems that must be resolved so that the hidden Markov Model could have concrete applications. This characterization is due to J. Ferguson, who introduced it during lectures at Bell laboratories. These three problems are the following [18]:

- a) Given such a model \( \lambda = (A, B, \alpha(0)) \), and a sequence of observations \( Y \), we can try to determine the likelihood of the observed sequence, given the model.
- b) Given such a model \( \lambda = (A, B, \alpha(0)) \), and a sequence of observations \( Y \), we can try to determine the hidden part of the model. It consists therefore in finding a sequence of states \( X \) which is optimal for the underlying model.
- c) Given a sequence of observations \( Y \) and the dimensions \( N \) and \( M \) (giving the number of states of the Markov chain \( X \) and the size of the observed sequence), we can try to find the model \( \lambda = (A, B, \alpha(0)) \) which maximizes the probability of \( Y \). It consists therefore in providing training data to the model, in order to estimate the parameters of the model.

The process of iterative re-adjustment of the model \( \lambda = (A, B, \alpha(0)) \) is the following:

1) Initialize the model \( \lambda = (A, B, \alpha(0)) \) randomly.
2) Use the Forward and Backward algorithms to re-estimate the model \( \lambda = (A, B, \alpha(0)) \).
3) If \( P(Y | \lambda) \) increases sufficiently (namely increases of \( \delta \geq \Delta \), where \( \Delta \) is a predetermined threshold) or if the number of iterations exceeds a certain threshold, repeat step 2.

Thus we see that it is possible to form a Markov Model \( \lambda = (A, B, \alpha(0)) \) on training data (made up with the instructions spectra from a family of viral programs) in order to estimate the parameters of the model, by using the re-estimation algorithm of Baum-Welsh.

Then, given such a model, and a sequence of instructions extracted from a file \( F \), we can calculate the likelihood of the observed sequence, given the model, by using the Forward algorithm.

In [18], an HMM is compared with a set of urns each containing a certain number of colored balls. Sampling of such a model is then made by choosing at each step a new urn, this choice depending only on the urn previously chosen, then by choosing in replacement a ball in this urn. The sequence of urn choices is not public (this sequence is hidden). However, the balls choices are known (they are observed).

In the context of viral detection, we observe a sequence of bytes or assembly instructions, and we are interesting not only in how an HMM produces these two sequences, but also in the distribution of the sequences produced by an HMM and in the way to compare them. During the training phase of the model, the idea is to use a set of viral programs generated from a same mutation engine: typically, a set of variants produced by a polymorphic virus or a set of viral programs produced from a virus generator kit (or a polymorphic shellcode generator).

III. CRITERIA

Now that we have recalled some of the currently mostly used Bayesian network based virus detection models, let us give a characterization of these models through a few common criteria.

In this section, we present a review of theoretical criteria for judging the appropriateness of the design of a detection engine. Unfortunately it appears difficult to identify design criteria powerful enough to compare algorithms that are fundamentally different in their approaches, as is the case for example of spectral analysis and syntactic or semantics analysis. However, among a set of methods stemming from the same approach, we think it is possible to compare between them two detection algorithms, on the basis of the criteria of soundness and completeness in particular.

A. Soundness and Completeness

The methods of programs analysis can be classified according to the set of properties that they can establish with some confidence. Let us introduce the following definitions [14]: a detection scheme \( (\lambda_V, f_V) \) is sound for a viral set \( V \) if it is sufficiently precise to avoid the risk that a benign program (or from another viral strain) is wrongly considered as belonging to this set. The detection function does only recognize malware belonging to this set. In other words, \( \alpha \), the false alarm probability, is near zero ; a detection scheme is complete for a viral set when it recognizes all the possible variants of viruses belonging to this set. In other words, \( \beta \), the non detection risk, is near zero. The residual risk in this case is that a benign program (or from another strain) also belong to this set.

B. Robustness

We can define the robustness as the difficulty of a detection scheme evasion. As it is often the case, we cannot always
prove formally the unconditional security of a detection scheme. The evaluator has then to try to evade the detection scheme, through an evaluation of the intrinsic limits of the model; or by exhibiting some theoretical weakness (lack of resistance) with regards to some obfuscation transformation for example.

In the same way that we have defined the security of a stealth system\(^1\), we can define the precision of a detection function based on a Bayesian network in a dual way, in the context of spectral analysis:

**Definition 3.2 (Precision of a model [19]):** Let \( p_Y = p(Y_0, \ldots, Y_{T-1}) \) and \( p_Y^\lambda = p(Y_0, \ldots, Y_{T-1}) \) be the real distribution and the distribution under a model \( \lambda \) of the observed variables \( Y \). The precision of a model \( \lambda \) is proportionally greater as the Kullback-Leibler distance \( d_{KL} \) between the distributions \( p_Y \) and \( p_Y^\lambda \) approaches zero:

\[
d_{KL}(p_Y||p_Y^\lambda) = \sum_y p_Y(y) \log \frac{p_Y(y)}{p_Y^\lambda(y)}.
\]

In the general case, we consider models \( \lambda \) with a KL distance different from zero (KL-distance error):

\[
d_{KL}(p_Y||p_Y^\lambda) = \varepsilon > 0
\]

The accuracy of a model \( \lambda \) reflects the relative entropy between the actual distribution of a viral family and the detection model built from the training data.

**C. Complexity**

At last, one of the crucial points in a detection model evaluation is its complexity in space and time. All the provided models haven’t got the same processor and memory consumption efficiency. Moreover, it is sometimes difficult to evaluate the complexity of an algorithmic implementation of a given model theoretically. The training and scoring stages may be quite asymmetric. Furthermore, the implementation of the information extraction function may have great incidence on the whole detector efficiency.

We have seen in section II that the naive Bayes’ computing complexity is optimal with respect to the other classification methods, and that the HMM detection stage’s computing complexity is linear with respect to its input size, but is quadratic with respect to the number of hidden states. Both approaches may be efficient provided that the information extraction process (for example, the disassembly stage, static or dynamic) is fast enough.

**IV. LIMITS AND COMPROMISE**

We have introduced in the previous sections Bayesian network based detection models and proposed a characterization of these models through precise criteria, for judging the suitability of such a detection engine criteria; the purpose of this section is to discuss the limits of these models and the required trade-offs that they induce.

**A. Intrinsic Limits**

There are several ways to qualify a detection model in terms of limits. We define an intrinsic limit of a detection model as anything relative to the model that can be exploited by an attacker to evade the corresponding detection function (assuming that the attacker knows the model). As we shall see, such a definition covers several aspects, ranging from the scope of the detection model, the choice of the model itself, to its adjustment and settings.

1) **Scope of the model:** As we have already mentioned, the scope of a detection model, i.e. the set of functional components which are covered by the model, is a first characterization of the limits of the model. Such a limitation is often due to the problem of the information extraction process modeling. When this function is not supposed to be provided by specialized oracle, by this way placed outside the scope of the model, it is simply ignored. In the latter case, this point is supposed to be clarified by the design or implementation choices of the detection engine. Currently, no model takes the dynamic information extraction process (occurring for example during the emulation of viral code) into account. The execution through a complete software interpreter machine is nevertheless a convenient way to get accurate information from a program. If you want to include this function into the scope of the model, you have to model the information extraction process through dynamic models. Currently, most of the models used in viral detection apply only in a static analysis context and reduce the scope to scoring function only.
2) Model choice, adjustment, setting: Next come the weaknesses induced in a detection model by the choice of the model itself, or by its adjustment and settings. With the question of the choice of a model comes the question of its theoretical limits. A theoretical analysis of a detection model must be sufficient to exhibit its limits. But when it comes to its adjustment and settings, we have to take caution to the method used to train the model or populate its knowledge database. Both analyses are useful to evaluate the detection engine.

As an example, choosing a good dictionary is essential in any classification problem. In the case of spectral analysis, an adapted instruction set has to be built. Indeed, depending on the number of instructions that compose it, the results can vary quite significantly for a same classification method and different performances can be observed between different methods. The reduction in size of the dictionary also has an impact on the performances. The Information theory provides tools that are usually used to reduce the size of a dictionary by keeping only the words with sufficient discriminating power. Using the Information theory, through the concepts of entropy and mutual information, we can describe a classification model, in terms of accuracy. Entropy and conditional entropy of a random variable X are defined as follows:

**Definition 4.1 (Entropy [20]):** Entropy $H(X)$ is a measure of the uncertainty on the random variable X:

$$H(X) = - \sum_{x} P_X(x) \log P_X(x).$$

The conditional entropy $H(X|Y)$ represents the uncertainty on the variable X after observation of the random variable Y:

$$H(X|Y) = E_{P_Y} \left[ -E_{P_{X|Y}}[\log P_X|Y] \right].$$

The mutual information of two random variables X and Y is defined as follows:

**Definition 4.2 (Mutual Information [20]):** The mutual Information $I(X, Y)$ represents the reduction of uncertainty about the random variable X after observation of the random variable X:

$$I(X, Y) = E_{P_X,Y} \left[ \log \frac{P_{X,Y}}{P_X P_Y} \right]$$

$$= \sum_{x,y} P_{X,Y}(x, y) \log \frac{P_{X,Y}(x, y)}{P_X(x) P_Y(y)}$$

$$= H(X) - H(X|Y).$$

Mutual Information can be viewed as a transmission rate through a noisy channel:

$$X \xrightarrow{\text{Channel}} Y$$

Mutual information $I(X, Y)$ is also the KL distance [21] between the joint distribution, $P_{X,Y}$, and the independent product of the distributions, $p_X p_Y$. Therefore, another way to see the mutual Information is that it is the distance between the correlated and non-correlated distributions of X and Y:

$$I(X,Y) = d_{KL}(p_{X,Y}||p_X p_Y)$$

$$= \sum_{x,y} p_X(x) p_Y(y) \log \frac{p_X(x)p_Y(y)}{p_{X,Y}(x,y)}.$$  

Because the KL distance equals zero if and only if $p_{X,Y} = p_X p_Y$, it follows that the mutual information captures all the dependencies between random variables, not only for example a second order dependency, such as the one captured by the covariance.

Using mutual Information, we can select the instructions that carry a discriminatory power by choosing the most characteristic words of a program category. For each instruction, we look if it is correlated with the classes. If so, this means that it carries semantic/meaningful information and therefore that we have to keep it.

Let us consider a Bayesian model. Using the following calculation, we can check, for each instruction $W = w_i$, whether the entropy of the distribution of classes decreases well enough or not when we assume that the instruction $w_i$ is present:

$$I(X, W_i) = H(X) - H(X|W_i)$$

$$= \sum_{B_i \in \{0,1\}} \sum_{j=1}^{|C|} p(X = c_j, B_i) \log \frac{p(X = c_j, B_i)}{p(X = c_j)p(B_i)}.$$

Such a method can be used to refine a model and has obviously an impact on its results. Let us give another example of setting that may have a great impact on an HMM-based model: the number of hidden states.

3) Model intrinsic limits example: number of hidden states: We have seen that one of the important properties of a Hidden Markov Model is its accuracy. We will see in this section that one of the factors affecting its precision is a too small number of hidden states. This goal corresponds to solve the optimization problem (problem b) that we have already identified.

Using the hidden part of a model, we can compare two models to each other, based on specific criteria. Note that in the case of a signature search, it is also possible to extract qualitative information on the accuracy of a signature, through a black box analysis of the detection engine. This approach is described in [22] and also responds to that will to characterize a detection scheme on the basis of precise criteria.

We have already mentioned the importance of the number of hidden states to ensure the accuracy of an HMM model. To understand this criterion, it is useful to adopt an informational vision of an HMM model, through the concept of noisy channel: let $Y_t$ be the observation at time $t$ (an assembler instruction in the case of spectral analysis) and $Y_{<t}$ the observations collection at the other times except the time $t$. The dependence of $Y_t$ in relation to $Y_{<t}$ can be seen as passing through a noisy channel. The information
transmission rate in this channel is given by the mutual information \( I(Y_{-t}, Y_t) \) between these two sets of variables. By definition of the HMM model, any observation \( Y_t \) is independent of \( \{X_{-t}, Y_{-t}\} \), conditionally to the hidden state \( X_t \). Therefore, a hidden state \( X_t \) separates \( Y_t \) from its context \( Y_{-t} \):

\[
\begin{align*}
Y_{-t} & \xrightarrow{\text{Channel I}} X_t \xrightarrow{\text{Channel II}} Y_t
\end{align*}
\]

In a way, an HMM compresses information regarding \( Y_t \) stored in \( Y_{-t} \) into the discrete variable \( X_t \). For the HMM model to be accurate, the noisy channel that is represented above must have the same information transmission rate as the following noisy channel:

\[
\begin{align*}
Y_{-t} & \xrightarrow{\text{Channel}} Y_t
\end{align*}
\]

Assuming the (perfect) accuracy of the HMM model, we have:

\[
d_{KL}(p_Y||p_Y^{HMM}) = \sum_y p_Y(y) \log \frac{p_Y(y)}{p_Y^{HMM}(y)} = 0.
\]

Under this condition, we have the following result:

Theorem 4.1: (Necessary condition for the precision of an HMM [19]) The following conditions are necessary for the precision of an HMM:

- the transmission rate \( I_{HMM}(Y_{-t}, X_t) \) between \( Y_{-t} \) and \( X_t \) must be greater than \( I(Y_t, Y_{-t}) \);
- the transmission rate \( I_{HMM}(X_t, Y_t) \) between \( X_t \) and \( Y_t \) must be greater than \( I(Y_t, Y_{-t}) \);
- the variable \( X_t \) must have enough storage capacity to encode the information circulating through the two noisy channels. More precisely, the number of hidden states \( N = |X| \) must satisfy the following condition:

\[
N \geq 2^I(Y_t, Y_{-t}).
\]

The first two conditions are intuitively quite natural: if one of these conditions is not met, one of the two channels (channel I or II) will become a bottleneck. Each of the two channels must have sufficient capacity. The imprecision of an HMM may result from the use of an improper family of observations distributions, which corresponds to the use of a channel with insufficient capacity.

The latter condition is probably the most important because in all cases, a channel bottleneck may come from the fixed number of hidden states. In the context of spectral analysis, if the assembler instructions generated by a mutation engine have significant mutual information, the approximation by an HMM model may fail because of the too high number of hidden states required. Let us recall that the mutual information \( I(Y_1, Y_2) \) between two random variables \( Y_1 \) and \( Y_2 \) is also the \( KL \) distance between the joint distribution \( p_{Y_1,Y_2} \) and the independent product of distributions \( p_{Y_1}p_{Y_2} \). This necessary condition for the accuracy of the model, and therefore its security against a simulation attack, requires a weak level of dependence between the assembler instructions produced by the mutation engine that we try to model by HMM.

In other words, a mutation engine able to induce a high level of dependence between the assembler instructions might be able to evade any HMM model having too small a number of hidden states.

B. Model Simulability

We have introduced in our previous works [1] the concept of statistical testing simulability, which may be defined as a way for an attacker to evade detection by using to his advantage the intrinsic flaws of a detection model or of its parameters.

1) Simulability of a Bayesian network: We can define the simulability of a detection function based on Bayesian models as follow:

Definition 4.3: (Simulability of spectral analysis based on Bayesian models): To simulate spectral analysis based on a Bayesian models \((\lambda_i)_{1\leq i \leq n}\), the mutation engine must randomly modify the instructions \( Y \) so that the Kullback-Leibler distance \( d_{KL} \) between the distributions \( p_Y \) and \( p_Y^{\lambda_i} \) is greater than \( \varepsilon \), for each of the models \( \lambda_i \) used by the detection function. We must thus have:

\[
\min_{i=1,\ldots,n} d_{KL}(p_Y||p_Y^{\lambda_i}) \geq \varepsilon.
\]

In other words, the distribution of the viral program must remain at a respectful distance from the distributions recognized by the detector. The safest way to do this is to come close to the distribution of the benign programs of the system. Such an approach has two advantages:

- a detection function by spectral analysis taking into account such a malware family takes the risk of significantly increasing the risk of false positives. Indeed, the likelihood of an observed sequence \( Y \) given the corresponding model will be too high, and will exceed the value of the detection threshold \( T \), once the program \( Y \) is a benign program used by the malware as "reference". The detection scheme might become unsound.
- observe that such a viral program corresponds to a stealth system at least \( T \)-secure against the detection function, even if here an alert is triggered (false alarm).

- a detection function by spectral analysis that does not take into account the malware families of which statistical distribution comes too close to the one corresponding to the benign programs of the system takes the risk of increasing the false negative rate. The detection scheme might become incomplete.

We thus see another limitation of detection by spectral analysis: viruses hosted by benign programs are intrinsically difficult to detect. Metamorphic viruses using code
introduction techniques\textsuperscript{3} may be even more (because of an increase in the required number of hidden states, in the case of an HMM).

Let us now consider the characteristics of the observed sequence that can make it difficult to construct an accurate HMM model. The first characteristic concerns obviously the distribution of the observed sequence. If it is chosen improperly, the detection function will remain ineffective. This criterion is related to the extraction of the information used to train a model: those sources of information must be reliable. In the case of spectral analysis, if the observed sequences are machine instructions, disassembly must be correct. The same constraint applies during the detection stage using this model. This criterion, when applied to the detection of stealth viruses, is even more important since the probe dedicated to the recovery of the observed sequences should not interact with them. If so, the model must be adjusted accordingly.

C. Compromise

We have already mentioned that it appears difficult to identify design criteria powerful enough to compare algorithms based on fundamentally different approaches, as is the case for spectral analysis and syntactic or semantics analysis.

Each model has its strong points and weaknesses. Some of them are compositional. This is apparent in the case of statistical detection models: let us assume that the detector performs several statistical tests, applied sequentially, each of them applying to the results of the previous one. If we assume that the testings are independent one from the other, with respective non-detection probabilities $\beta_i$ and false positive probabilities $\alpha_i$, $i = 1, \ldots, n$, then the residual non-detection and false positive probabilities, $\beta$ and $\alpha$, are given by \cite{1}:

$$\alpha = \prod_{i=1}^{n} \alpha_i \text{ and } \beta = \prod_{i=1}^{n} \beta_i.$$  

Under such conditions, soundness and completeness appear to be compositional. Indeed, the detection scheme resulting from the composition of several sound (resp. complete) detection schemes is a sound (resp. complete) detection scheme. However, if these hypotheses are not met, empirical experiments remain the method to choose and adapt a detection model.

One of the limitations of a Bayesian network based model is that it requires a sufficient amount of training data. When using a malware generator kit or a polymorphic shellcode generator, this constraint does not seem to be a problem insofar as we have the mutation engine. It is more troublesome, however, in the case of malware with too few variants or are difficult to capture. This is particularly the case for malware using entropy sources on their system or network environment to mutate. This approach appears therefore, as such, complementary to the traditional approach by signature.

V. Conclusion

Based on the concept of statistical testing simulability, it is possible to characterize a detection scheme by a measure of the difficulty to bypass it. We have illustrated this position by a study of detection algorithms which are based on the instructions spectrum analysis. We have considered in the first place, in our previous work \cite{1}, elementary detection models and proved the simulability of the corresponding statistical tests. We have next taken here an interest in the bypass possibilities of more sophisticated detection schemes, namely the Bayesian networks. We have studied in detail the application of two of these models to spectral analysis: the naive Bayes model and the hidden Markov model. We expect this formal framework to complete usefully the other frameworks which are already used with this goal (complexity theory and formal grammars, abstract interpretation theory).

Open problems

Scope: we have already noticed that a detection engine efficiency strongly relies on the accuracy of its information extraction process. This process can be static, that is to say, conducted without running the viral program, or dynamic. Here, if you want to include this function into the scope of the model, you trigger the occurrence of a big problem: the problem of the information extraction process modeling through static versus dynamic models. Currently, most of the models apply only in a static analysis context.

Model choice: in the spectral analysis context, each model $\lambda$ is designed to store information to summarize or compress the characteristic of a mutation engine. It is expected that during the training phase of a model, the parasitical information, eg resulting from the application of obfuscation transformations are not taken into account in the characterization of a viral family or that they are taken into account in a manner that does not interfere with detection.

Concerning the HMM model, some of the information stored in an HMM relates to the hidden Markov chain, which we hope will contain, after the training phase, a synthetic information on the virus mutation engine so we can recognize all programs resulting therefrom. One of the interests of this type of model, as compared with the more simple naive Bayes model, is that it makes it

\textsuperscript{3} The virus mixes with the instructions flow of its host, by using a disassembler and a compiler engine so it can rebuild the host binary. Such a technique is notably used by the mutation engine of the virus Zmist, called Mistfall \cite{23}.
intrinsically possible, and computationally in an efficient way, to recover the states sequence $X$ and information on the model structure, from the given model $\lambda$ and the observed sequence $Y$. Therefore, it seems to be possible to characterize a model on the basis of structural and qualitative information. However, no research has been conducted to understand what is embedded in the hidden states, after the training stage. Such work has been done in other research areas (speech recognition, genetics), but not in the malware spectral analysis context. It might be interesting to understand what is captured by the model.

Model setting: we have also observed in section IV that the imprecision of an HMM may result from a channel bottleneck coming from the fixed number of hidden states. In the context of spectral analysis, if the assembler instructions generated by a mutation engine have significant mutual information, the approximation by an HMM model may fail because of the too high number of hidden states required. Actually, a necessary condition for the accuracy of the model, and therefore its security against a simulation attack, requires a feeble level of dependence between the assembler instructions produced by the mutation engine that we try to model by HMM. In other words, a mutation engine able to induce a high level of dependence between the assembler instructions might be able to evade any HMM model having a too small number of hidden states.

Clearly, more experiments are required to give an answer to these specific questions. However, such a knowledge is essential to define the security of a model better, against simulability attacks.

Future works
Bayesian methods and HMM are already used in a large variety of applications, including text, voice and speech recognition, genetics, cryptanalysis, SPAM filtering, and viral detection. As regards with the latter application, additional works are with no doubt required to refine existing models or to find more adapted models in the range/extent of graphic models or Bayesian networks.

More work has to be done to specify the security of a detection model, increase the scope of the models to exploit the advantage of the current implementations (emulation engine notably).

Those classification methods are promising, and their utility in the detection by spectral analysis of viral families which are generated by a generation kit or a viral mutation engine, is undeniable. Indeed, the malware generator kits provide generally a sufficient amount of data to train the models. The simulability of statistical tests turns out to be a very useful concept to guide the rigidity analysis of these detection schemes.

REFERENCES