Hidden Markov models for advanced persistent threats

Guillaume Brogi, Elena Di Bernardino

Hidden Markov models for advanced persistent threats

Guillaume Brogi*†, Elena Di Bernardino†
* Akheros and † Conservatoire National des Arts et Métiers, Laboratoire CEDRIC, EA4629, Paris, France
guillaume.brogi@akheros.com
elena.di_bernardino@cnam.fr

***Abstract*—Advanced Persistent Threats (APT), attack campaigns performed by competent and resourceful actors, are a serious security risk and tools suited to their detection are needed. These attack campaigns do leave traces in the system, and it is possible to reconstruct part of the attack campaign from these traces. In this article, we describe a stochastic model for the evolution of an APT. It is based on hidden Markov models (HMM) and is accompanied by a score. The aim of this model is to validate whether the evolution of the partially reconstructed attack campaigns are indeed consistent with the evolution of an APT. In addition, the introduced score is designed to take into account the inevitable presence of undetected attacks in the attack campaigns. It also allows comparing attack campaigns of varying length, which is necessary to be able to compare attack campaigns. We validate and illustrate both the model and the score using data obtained from experts.**

***Index Terms*—hidden Markov model, score, attack campaign, advanced persistent threat**

I. Introduction

In recent years, Advanced Persistent Threats (APTs) have emerged as a real threat to both private companies and government agencies [1]. While there is no exact definition of what is and isn't an APT, the idea is that the attack campaign is the work of professional hackers with well funded backing, usually state or large company. This means the attackers are competent, organised and resourceful, and they have precise goals that they need to achieve, such as extracting sensitive data or sabotaging machinery. These APTs can last for months or years, and are customised for their target. While APTs are a formidable threat, the attackers still leave traces, as [2] found traces of the breach in 86% of cases with no specific anti-APT solution. This fact is leveraged by ongoing research [3], [4] to defend against them.

In this article, we present a solution for adding context to individual attacks in order to assist with their evaluation. This builds on top of our previous work on reconstructing attack campaigns [5]. That initial tool is capable of linking related attacks together but can also sometimes link unrelated attacks. The model and accompanying score in this article are used to remove those unrelated attacks and keep only the rightly linked attack chains from the results of the first tool. Concretely, the contribution is a model and a score which can rank chain of attacks, already detected and linked by other tools, in order of most probably an APT to least probably an APT. The model is an hidden Markov model (HMM) and it uses a custom score designed specifically to handle missing observations and to compare chains of different length.

The rest of this article follows the usual outline. Section II presents related works, and Section III the security model. We introduce hidden Markov models and our score in Section IV and evaluate them in Section V. Finally, we conclude the article in Section VI.

II. Related Works

Most research studying APTs starts by decomposing them into several phases, as shown in [6]. In [3], Giura and Wang base their concept of an attack pyramid on this decomposition. The exact phases they use are "Reconnaissance", "Delivery", "Exploitation", "Operation", "Data Collection" and "Exfiltration". This is central to the attack pyramid concept, a construct to determine the path attackers can use to reach a given goal. They start by identifying the overall goals, targets of the "Exfiltration" phase, and from there move backwards to the earlier phases. For each goal, they list all the way the attacker could reach it. These become the goals of the previous phases. The main difference with attack trees is that attack pyramids can encompass several planes, such as the physical plane where someone can pick a lock or phishing on the user plane.

In [4], Sexton et al. once again decompose APTs into five phases: "Delivery", "Exploit", "Install", "C&C" and "Actions". They uses these phases to construct attack chains by associating each phase with a number of event types. They then combine events in different phases in order to detect APTs. The combination is done by computing a score for each type of event and for each phase; this score is then aggregated at the host and cluster level. If the score is high enough, an APT is detected.

Both of these articles are interesting because they decompose APTs in phases. However, they both require knowledge beforehand. In [3], every path of attack must be thought of, while in [4], every type of attack must be associated with a specific phase. From these article, we keep the decomposition of APTs in phases. In this article, we use the following phases:

1) Reconnaissance (R)
2) Compromission (C)
3) Establish presence (EP)
4) Privilege escalation (PE)
5) Mission completion (MC)

In addition, the attacker can, at any time, go back to a previous phase. There are two main reasons for going back: either the current attack failed and the attacker must find another way to achieve their current goal, or the attack succeeded, but the current machine is only a stepping stone and the attacker must now find a machine closer to its ultimate goal to gain a foothold on. This last move is called a lateral movement and is expected in an APT.

In [7], Ourston et al. use a hidden Markov model (HMM) to model multi-stage network attacks. The phases of an attack are: "Probe", "Consolidate", "Exploit" and "Compromise". These are the hidden states of the HMM. Attacks are categorised and these categories are the observations of the HMM. This resembles closely what we are doing. However, on top of being restrained to network attacks, this article does not take into consideration two key elements of attack campaigns. First, they do not take into account the possibility that several attack

campaigns could be happening at the same time. Second, they do not take into account the possibility that some attacks are not detected. We address the first issue with our previous contribution [8] and build this contribution on top of it, adding a score specifically designed to handle missing observations, i.e. undetected attacks.

III. Security Model

In this article, we present a method to evaluate potential attack campaigns and decide whether they really are attack campaign. This work is a continuation of our previous article [8] where we presented the first half of TerminAPTor, a method to link attacks together. The method presented here is the second half of TerminAPTor and is meant to analyse those potential attack campaigns. Hence, we use the same threat model.

We propose to protect enterprise networks from APTs. Such a network is composed of machines being used as a mix of servers and workstations. They are monitored. In our case, we posit that each machine is equipped with a local monitoring agent which includes the first half of TerminAPTor. This agent detects attacks and links related ones together. The output is thus a list of potential attack campaigns, where each attack in the campaign is categorised. The second half of TerminAPTor, the subject of this article, analyses each potential campaign based on the category of the attacks it contains and ranks them in order of more likely an APT to least likely an APT.

The attacker is performing an APT. This means they have a precise goal and they are following the phases outlined in Section II: reconnaissance, compromission, establish presence, privilege escalation and mission completion; where, at any time, the attacker can start a lateral movement on the internal network usually by going back to the reconnaissance or compromission phase, but on a different machine. For each phase, the attacker is capable of either exploiting existing tools or deploying their own customised one; overall, they are capable of infiltrating any system they want to. In addition, the attacker tries to avoid detection and will thus avoid unneeded actions. This means that each attack in an attack campaign is necessary and is used to setup the next attack.

IV. Hidden Markov models for APTs

A hidden Markov model (HMM) is a two-level stochastic process. The first level is a Markov chain representing the state of the system. However, the states cannot be measured directly. Instead, states generate observations which can be measured and are then used to infer which state generated the observation. Thus, a HMM is defined by:

1) The N states of the system, denoted as
$$S = \{S_1, S_2, \dots, S_N\}. \tag{1}$$
In our case, these states would be the five phases of an attack outlined in Section II. The L states of a given chain of length L of a HMM are written as $s = s_1, s_2, \dots, s_L$.
2) The M observations of the system, denoted as
$$O = \{O_1, O_2, \dots, O_M\}. \tag{2}$$
In our case, these observations would be the alerts raised by an IDS. As expected, they do not tell us directly which phase the attacker is in, but they do give us information which we can leverage to infer the most probable state sequence. The L observations of a given chain of length L of a HMM are written as $o = o_1, o_2, \dots, o_L$.
3) The state transition probability matrix $A = [a_{ij}]$, where
$$\forall t,\ \mathbb{P}(s_{t+1} = S_j | s_t = S_i) = a_{ij} \text{ for } i, j \in [\![1, N]\!]. \tag{3}$$
A is a $N \times N$ matrix, and $\forall\ i,\ \sum_j a_{ij} = 1$.
4) The observation probability matrix $B = [b_{ij}]$, where
$$\begin{aligned} \forall t,\ \mathbb{P}(o_t = O_j | s_t = S_i) = b_{ij} \\ \text{for } i \in [\![1, N]\!] \text{ and } j \in [\![1, M]\!]. \end{aligned} \tag{4}$$
B is a $N \times M$ matrix, and $\forall\ i,\ \sum_j b_{ij} = 1$.
5) The initial probability vector $\pi = [\pi_i]$, where
$$\mathbb{P}(s_1 = S_i) = \pi_i \quad \text{for } i \in [\![1, N]\!]. \tag{5}$$
π is a vector of length N and $\sum_i \pi_i = 1$.

An HMM will then be denoted as $\lambda = (A, B, \pi)$.

Given a sequence of observations and a HMM, we can compute the most probable path which generated that sequence of observation as well as its probability in the model using the Viterbi algorithm [9]. This is done by walking forward in the trellis of possible transitions and computing the maximum probability of reaching each state at each time step, taking the observations into account. Once the end of the chain is reached, the trellis is walked backward starting from the highest probability and finding which state in the previous step lead to it and so on until the beginning of the chain is reached. Let us take the well-known example HMM in [10]. The idea is to find if the temperature of years in the distant past was hot or cold based on the observed tree ring size. Tree rings do not directly tell us whether a year was hot or cold, but colder years will have a tendency to make trees have smaller rings and warmer years will make them larger. Hence, the HMM described has two states "hot year" ("H") and "cold year" ("C") and three observations "small tree rings" ("S"), "medium tree rings" ("M") and "large tree rings" ("L"). The matrices are:

$$A = \begin{bmatrix} 0.7 & 0.3 \\ 0.4 & 0.6 \end{bmatrix},\ B = \begin{bmatrix} 0.1 & 0.4 & 0.5 \\ 0.7 & 0.2 & 0.1 \end{bmatrix},\ \pi = \begin{bmatrix} 0.6 & 0.4 \end{bmatrix}. \tag{6}$$

Let us walk through the Viterbi algorithm for the following chain: "S-M-S-L", as shown in Fig. 1. From (5), we know directly that $\mathbb{P}(s_1 = \text{"H"}) = \pi_{\text{"H"}} = 0.6$ and that $\mathbb{P}(s_1 = \text{"C"}) = 0.4$. We can then compute the probability that the chain is at either state and generates the first observation "S":

$$\mathbb{P}(o_1 = \text{"S"} | s_1 = \text{"H"}) \cdot \mathbb{P}(s_1 = \text{"H"}) = 0.06. \tag{7}$$
$$\mathbb{P}(o_1 = \text{"S"} | s_1 = \text{"C"}) \cdot \mathbb{P}(s_1 = \text{"C"}) = 0.28. \tag{8}$$

Then, for each state, we use the result of (7) and (8), multiply by the transition probability and take the maximum of the two. For example, if the second state is "C":

$$\begin{aligned} \max(&0.06 \cdot \mathbb{P}(s_2 = \text{"C"} | s_1 = \text{"H"}), \\ &0.28 \cdot \mathbb{P}(s_2 = \text{"C"} | s_1 = \text{"C"})) = 0.168. \end{aligned} \tag{9}$$

We then multiply this result by the observation probability $0.168 \cdot \mathbb{P}(o_2 = \text{"M"} | s_2 = \text{"C"}) = 0.168 \cdot 0.2 = 0.0336$. We do this for every state at each step and obtain the numbers in Fig. 1. We then retrace the trellis by selecting the state with the highest probability ("H" in this case) and finding which state at the previous step led to this probability ("C" in this case). We then do the same for that state and so on until we reach the first step. In this case we find that the chain of observation "S-M-S-L" was most probably generated by the state sequence "C-C-C-H" and that $\mathbb{P}(o = \text{"S-M-S-L"}, s = \text{"C-C-C-H"} | \lambda) = 0.002822$.

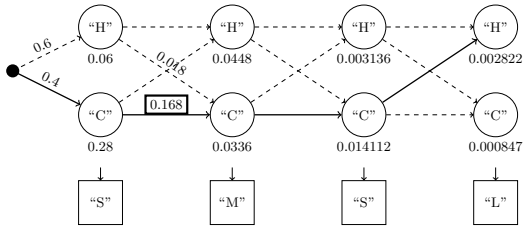


Fig. 1. Walking the trellis in the Viterbi algorithm for the "temperature of the year and tree ring size" example.

In our case, we aim to use HMMs to model APTs. This means that the states will be the different phases of an attack campaign (see Section II) and the observations are the alerts given by the IDSs. Determining the exact model will require computing the probability matrices π, A and B.

A. The base score

As mentioned, we want to rank the different campaigns in order of most probably an APT to least probably an APT. In particular, we want to be able to compare chains of different lengths. However, most scores are used to do model selection. This means they are used to find the best model for a given chain, and are not meant to compare the fit of several chains. In effect, scores such as the Akaike information criterion (AIC) [11] or the Bayesian information criterion (BIC) [12], try to balance the accuracy of the model (measured through the log-likelihood $\mathcal{L}$) with its complexity (measured through the number of parameters of the model), the aim being to find a model that is accurate enough but not too complex. The BIC, furthermore, aims to limit the amount of data required for model creation. Thus, we propose, in the following, a new score adapted to our APT evaluation problem. In particular, while it is based on the log-likelihood, this new score must not be sensitive to the length L of the chain.

The log-likelihood $\mathcal{L}$ of a given chain is defined as $\mathcal{L} = \ln(\mathbb{P}(o, s|\lambda))$, where s is obtained using the Viterbi algorithm, meaning it is the most probable chain of states having generated these observations (see [9] for details). Then:

$$\begin{aligned} \mathcal{L} &= \ln(\mathbb{P}(o, s|\lambda)) \\ \mathcal{L} &= \ln(\mathbb{P}(o_L|s_L) \cdot \mathbb{P}(s_L|s_{L-1}) \cdots \mathbb{P}(o_1|s_1) \cdot \mathbb{P}(s_1)) \\ \mathcal{L} &= \sum_{l=1}^{L} \ln(\mathbb{P}(o_l|s_l)) + \sum_{l=2}^{L} \ln(\mathbb{P}(s_l|s_{l-1})) + \ln(\mathbb{P}(s_1)). \end{aligned} \tag{10}$$

The log-likelihood is, thus, a sum of negative terms, and the number of terms increases with the length L of the chain. We can easily find an upper and a lower bound on the log-likelihood (see Equation (11)). These bounds are dependent on L but the ratio $\frac{\mathcal{L}}{L}$ is bounded by values independent of L (see Equation (12)). In keeping with the spirit of AIC and BIC we define our score as:

$$\mathcal{S} = -\frac{\mathcal{L}}{L}. \tag{13}$$

B. Adapting the score for missing observations

Now that we have a base score, we also have to take into account the fact that we may miss observations. As we have seen, APTs are executed by skilled actors, this means some steps of the attack will probably escape detection, and the score must account for this possibility. We do this by modifying the Viterbi algorithm.

Due to the Chapman-Kolmogorov equations for homogeneous HMMs, the transition probability of going from state S_i to state S_j in two steps is

$$\mathbb{P}(s_{t+2} = S_j | s_t = S_i) = \sum_k a_{ik} \cdot a_{kj} = A^2(i, j). \tag{14}$$

One can then generalise (14) and show that the transition matrix of taking k steps when going from one observation to the next is A^k. Similarly, the initial probability vector when reaching the first observation in k steps is $\pi \cdot A^{k-1}$. Thus, we denote the HMM where we take k steps for each observation as $\lambda^{(k)} = (A^{(k)}, B, \pi^{(k)})$ where $A^{(k)} = A^k = [a_{ij}^{(k)}]$ and $\pi^{(k)} = \pi \cdot A^{k-1}$. $A^{(k)}$ denotes the transition matrix for the case where there are k steps between one observation and the next, and A^k is the standard matrix power notation.

The idea is to modify the Viterbi algorithm to include the additional transitions described by $A^{(k)}$. We denote as p the probability that the monitoring agent does not detect an attack. We can then weigh each $A^{(k)}$ and $\pi^{(k)}$ by the probability that the monitoring agent misses $k-1$ attacks in a row, i.e. p^{k-1} for independent events. Ideally, we would rewrite the Viterbi algorithm taking into account $k \in [\![1, +\infty[\![$. However, for computational reasons, we only use $k \in [\![1, K]\!]$, where K is chosen arbitrarily. Note that the bigger k gets, the smaller p^{k-1} so the approximation we make by taking $k \leq K$ is limited. It does require that we normalise the probabilities so that they do sum to 1.

We denote the state S_i reached in k steps as $S_i^{(k)}$ and replace Equations (3), (4) and (5) with, respectively:

$$\text{for } i, j \in [\![1, N]\!], \quad l \in [\![1, M]\!], \quad k, m \in [\![1, K]\!],$$

$$\forall t,\ \mathbb{P}(s_{t+1} = S_j^{(k)} | s_t = S_i^{(m)}) = a_{ij}^{(k)} \cdot \frac{p^{k-1}}{\sum_{n=1}^{K} p^{n-1}}, \tag{15}$$

$$\forall t,\ \mathbb{P}(o_t = O_l | s_t = S_i^{(k)}) = b_{il}, \tag{16}$$

$$\mathbb{P}(s_1 = S_i^{(k)}) = \pi_i^{(k)} \cdot \frac{p^{k-1}}{\sum_{n=1}^{K} p^{n-1}}. \tag{17}$$

We note those weighed matrices as $A_p^{(k)}$ and $\pi_p^{(k)}$. Note, in particular, that the value of (15) does not depend on the value of m which means it does not depend on how many steps it takes to arrive on S_i but only on how many steps are taken going from S_i to S_j. We can then use Equations (15)-(17) in the new trellis to compute the Viterbi path and its associated log-likelihood. We note the log-likelihood computed with the Viterbi algorithm allowing for up to K steps from one observation to the next as $\mathcal{L}^{(K)}$, and the score computed from this log-likelihood is $\mathcal{S}^{(K)} = -\frac{\mathcal{L}^{(K)}}{L}$.

If we adapt the example taken from [10] with $K = 2$ and $p = 0.8$, we have the following matrices:

$$\begin{aligned} A_p^{(1)} &= A \cdot \frac{1}{1+p} = \begin{bmatrix} 0.389 & 0.167 \\ 0.222 & 0.333 \end{bmatrix}, \\ A_p^{(2)} &= A^2 \cdot \frac{p}{1+p} = \begin{bmatrix} 0.271 & 0.173 \\ 0.231 & 0.213 \end{bmatrix}, \\ B &= \begin{bmatrix} 0.1 & 0.4 & 0.5 \\ 0.7 & 0.2 & 0.1 \end{bmatrix}, \\ \pi_p^{(1)} &= \pi \cdot \frac{1}{1+p} = \begin{bmatrix} 0.333 & 0.222 \end{bmatrix}, \\ \pi_p^{(2)} &= \pi \cdot A \cdot \frac{p}{1+p} = \begin{bmatrix} 0.258 & 0.187 \end{bmatrix}. \end{aligned} \tag{18}$$

$$L \cdot \ln(\min_{ij}(b_{ij})) + (L-1) \cdot \ln(\min_{ij}(a_{ij})) + \ln(\min_{i}(\pi_i)) \leq \mathcal{L} \leq L \cdot \ln(\max_{ij}(b_{ij})) + (L-1) \cdot \ln(\max_{ij}(a_{ij})) + \ln(\max_{i}(\pi_i))$$
$$2 \cdot L \cdot \ln(\min_{ij}(b_{ij}, a_{ij}, \pi_i)) \leq \mathcal{L} \leq 2 \cdot L \cdot \ln(\max_{ij}(b_{ij}, a_{ij}, \pi_i)), \tag{11}$$

$$2 \cdot \ln(\min_{ij}(b_{ij}, a_{ij}, \pi_i)) \leq \frac{\mathcal{L}}{L} \leq 2 \cdot \ln(\max_{ij}(b_{ij}, a_{ij}, \pi_i)). \tag{12}$$

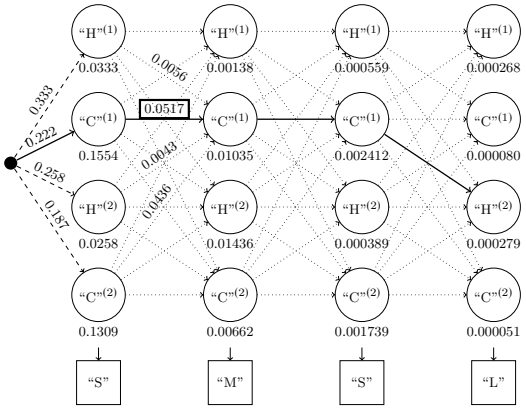


Fig. 2. Trellis of the Viterbi algorithm with $p = 0.8$ and $K = 2$.

Walking the tree is done as in the previous example but using the new equations. In this case, as shown in Fig. 2, we find that the chain "S-M-S-L" was most probably generated by the sequence "$C^{(1)}$-$C^{(1)}$-$C^{(1)}$-$H^{(2)}$" which actually means that the observation sequence is "S-M-S-X-L" and the state sequence is "C-C-C-Y-H", where "X" means that the observation was not observed and "Y" that the state is unknown because the associated observation was not observed.

As we can see, the result with our customised Viterbi algorithm differs from the original result. While in the case of tree rings, it does not make sense that observations are missed, especially with a $p = 0.8$, in the case of APT models, due to the skills of attackers, there will be missed observations, and it is necessary to take them into account.

We can now use our customised Viterbi algorithm to compute the log-likelihood used in the score defined in (13). This gives us a score which is not dependent on the length of the chain and which takes into account missed observations.

V. An APT case study

In this case study, we want to apply the model and the score defined above to the problem of APT detection. More specifically, we receive as input a number of sequence of observations and we must determine which of these sequences are more probably APTs. This means that the score must order the sequences from most probably an APT to least probably an APT. Additionally, the model should show the hidden states, including the number of unobserved states, if any.

A. The proposed model

The first step is to define the list of states S and observations O. Our model uses the five states in Section II: $N = 5$ and $S = \{$"reconnaissance (R)", "compromission (C)", "establish presence (EP)", "privilege escalation (PE)", "mission completion (MC)"$\}$. For the observations we use the output of a tool which defines seven categories of observations: $M = 7$ and $O = \{$"scan", "arbitrary execution", "credential theft", "application exploit", "backdoor", "remote access tool", "data exfiltration"$\}$. Once we have the states and observations, we require statistics about the transitions and observations in order to compute the matrices. There are two sources of information that we leverage. First, there are publicly available APT reports. They are useful in knowing which tools are used in which phase, but they are not precise enough in the evolution of the APT. Hence, we use them to create the observation matrix B. For the initial probability vector π and the transition matrix A, we use expert knowledge. We setup a website where experts are shown scenarios created at random. Each scenario can be rated as "strongly an APT", "weakly an APT" or "not an APT". The scenarios are shown as chains of states with an observation presented for each state as an indication. For example, the scenario "R", "C", "PE", "EP" and "MC" was rated as "strongly an APT" while the scenario "MC", "C", "MC", "EP" and "R" was rated as "Not an APT". Thanks to these two sources of information, we obtain the following HMM model $\lambda = (A, B, \pi)$:

$$A = \begin{bmatrix} 0.045 & 0.091 & 0.318 & 0.5 & 0.045 \\ 0.071 & 0.071 & 0.643 & 0.143 & 0.071 \\ 0.045 & 0.182 & 0.045 & 0.682 & 0.045 \\ 0.16 & 0.04 & 0.04 & 0.04 & 0.72 \\ 0.2 & 0.4 & 0.267 & 0.067 & 0.067 \end{bmatrix},$$
$$B = \begin{bmatrix} 0.3 & 0.02 & 0.02 & 0.02 & 0.6 & 0.02 & 0.02 \\ 0.5 & 0.14 & 0.01 & 0.1 & 0.13 & 0.07 & 0.4 \\ 0.01 & 0.3 & 0.01 & 0.3 & 0.05 & 0.03 & 0.3 \\ 0.05 & 0.4 & 0.05 & 0.05 & 0.05 & 0.1 & 0.3 \\ 0.4 & 0.03 & 0.01 & 0.09 & 0.01 & 0.4 & 0.06 \end{bmatrix}, \tag{19}$$
$$\pi = \begin{bmatrix} 0.7 & 0.21 & 0.03 & 0.03 & 0.03 \end{bmatrix}.$$

B. Experiment 1: Model adequacy checking

The aim of this first experiment is to check that the model differentiates APT chains from non-APT chains. To do so, we compare the scores of various scenarios rated as APT and as non-APT by our expert raters. We can then plot the scores and check that non-APT chains score higher than APT chain. In this first experiment, our score does not take into account possible missing observations. This means that we use the score $\mathcal{S}^{(1)}$. During the poll we explicitly told the expert raters to consider the steps shown as the whole APT, with no missing step, so this matches how the model was created.

The results can be seen in Fig. 3. The "Strong APT" chains appear in blue, the "Weak APT" chains in green and the "Not APT" ones in red. They show that both the proposed model and the associated score are able to separate APT from non-APT, with all APT chain below 1.5 and all non-APT above that mark.

We also created one chain of each kind with a length of 50 and 100 to check that the score is still consistent when chains

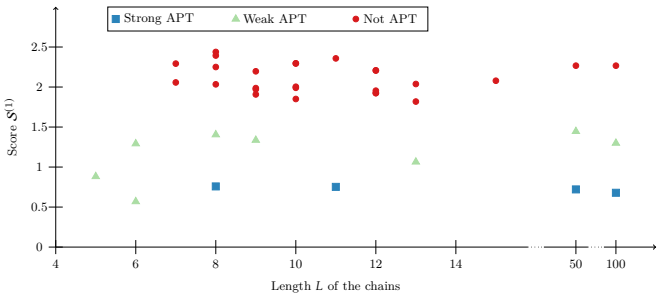


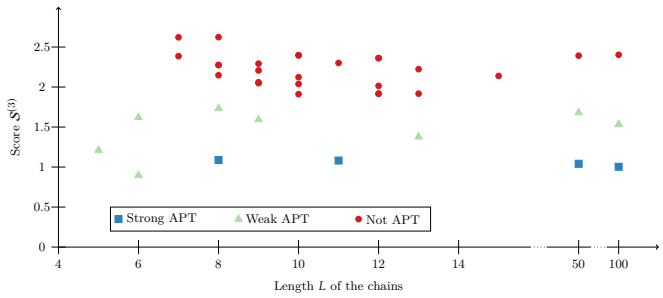
Fig. 3. The score separates APTs from non-APTs

are that long. Indeed, we see that the score does not depend on the length of the chain L, which was another crucial goal of our score.

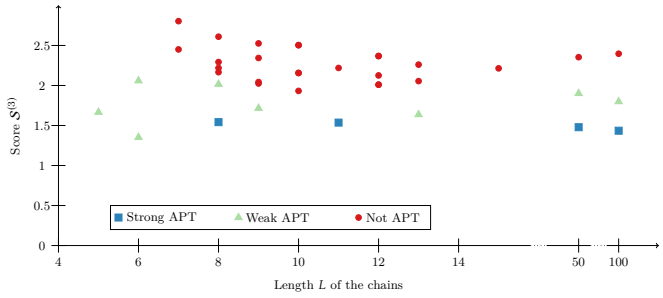
C. Experiment 2: Analysing the impact of missing observations

The aim of this second experiment is to check the impact of taking into account possible missing steps on the score and, more importantly, on the rank of each chain evaluated.

We display the same chains as in Fig. 3 but using $\mathcal{S}^{(3)}$ instead of $\mathcal{S}^{(1)}$ and with different values of p in Fig. 4. As we can see, the higher p is, the more some "Not APT" are on the same level as "APT" chains. This reflects that the higher p is, the higher the probability of missing one or even two consecutive observations is too, which means that our score will consider missing observations more. This is exactly what we wanted: by inserting well chosen and positioned unobserved states in the chain, their likelihood of being APTs is much higher. This means that if we have an APT where we did not observe some of the attacks, we can still reconstruct the probable unobserved attacks and evaluate that chain as an APT. For example, if we have a chain for which Viterbi gives us $\mathcal{S}^{(1)} = 1.62$ and the states "R"-"EP"-"MC". From Fig. 3, this chain is on the "Not APT" side, and there are phases missing in the APT, between "R" and "EP" for example. Switching to $\mathcal{S}^{(3)}$, if $p = 0.3$, $\mathcal{S}^{(3)} = 1.87$ and the states do not change; from Fig. 4a, the chain is still on the "Not APT" side, and there are still missing phases. However, if $p = 0.7$, $\mathcal{S}^{(3)} = 2.04$, the states become "C"-"Y"-"PE"-"MC". From Fig. 4b, this is now on the "APT" side. This is explained by the fact that if we replace the missing state ("Y") by "EP" phase, we now have a very reasonable chain of attacks for an APT.



(a) $p = 0.3$



(b) $p = 0.7$

Fig. 4. $\mathcal{S}^{(3)}$ of the original scenarios for different values of p.

VI. Conclusion

As we have shown, the threat of an APT, where the attacker executes a long-lived and multi-step attack campaign, can be adequately modeled by an HMMs. The contribution of this article is the fact we use such a model to highlight APTs. It builds on top of our previous contribution [8], which disentangle multiple concurrent APTs, and introduces a score specifically designed to take into account missing observations. This is crucial since the defense cannot guarantee they will detect every attack in an attack campaign.

The score and the model have potential. However, a study with real life data is still required to validate the approach. In addition, it is necessary to evaluate the influence of p and K on the ranking of potential APTs. Setting K is a trade-off between accuracy and computing power. The value of p is more subjective. However elaborate the IDS, attackers will find ways to circumvent it, so while a higher value of p could reflect badly on the quality of an IDS, it also shows the down to earth position of the defense team.

References

[1] C. Tankard, "Advanced persistent threats and how to monitor and deter them," *Network security*, vol. 2011, no. 8, pp. 16–19, 2011.

[2] Verizon, "2010 data breach investigations report," 2010. [Online]. Available: http://www.verizonenterprise.com/resources/reports/rp_2010-data-breach-report_en_xg.pdf

[3] P. Giura and W. Wang, "A context-based detection framework for advanced persistent threats," in *Cyber Security (CyberSecurity), 2012 International Conference on*. IEEE, 2012, pp. 69–74.

[4] J. Sexton, C. Storlie, and J. Neil, "Attack chain detection," *Statistical Analysis and Data Mining: The ASA Data Science Journal*, vol. 8, no. 5-6, pp. 353–363, 2015.

[5] G. Brogi and V. Viet Triem Tong, "TerminAPTor: Highlighting advanced persistent threats through information flow tracking," in *New Technologies, Mobility and Security (NTMS), 2016 8th IFIP International Conference on*. IEEE, 2016, pp. 1–5.

[6] Mandiant Intelligence Center, "APT1: Exposing one of china's cyber espionage units," 2013. [Online]. Available: http://intelreport.mandiant.com/Mandiant_APT1_Report.pdf

[7] D. Ourston, S. Matzner, W. Stump, and B. Hopkins, "Applications of hidden Markov models to detecting multi-stage network attacks," in *System Sciences, 2003. Proceedings of the 36th Annual Hawaii International Conference on*. IEEE, 2003, pp. 10–pp.

[8] G. Brogi, "APT models," 2016. [Online]. Available: https://aptmodels-guiniol.rhcloud.com/

[9] G. D. Forney, "The Viterbi algorithm," *Proceedings of the IEEE*, vol. 61, no. 3, pp. 268–278, 1973.

[10] M. Stamp, "A revealing introduction to hidden Markov models," *Department of Computer Science San Jose State University*, 2004.

[11] H. Akaike, "A new look at the statistical model identification," *IEEE transactions on automatic control*, vol. 19, no. 6, pp. 716–723, 1974.

[12] G. Schwarz, "Estimating the dimension of a model," *The annals of statistics*, vol. 6, no. 2, pp. 461–464, 1978.