Hidden Markov models for advanced persistent threats

Guillaume Brogi*†, Elena Di Bernardino†
* Akheros and † Conservatoire National des Arts et Métiers, Département IMATH, 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. Since APTs are the work of competent attackers, we can assume that not every step of the APT will leave traces. This fact must be taken into account when computing the fit with the model, which is why we introduce a new score. This score is based on the log-likelihood of the observations in the model, but also accounts for potentially missing observations. In addition, it also allows comparing attack campaigns of varying length. We validate and illustrate both the model and the score using synthetic and real-life data.**

***Index Terms*—hidden Markov models, score, attack campaign, advanced persistent threats**

I. Introduction

A. Advanced Persistent Threats

In recent years, Advanced Persistent Threats (APT) have emerged as a real threat to both private companies and government agencies [1]–[3]. While there is no exact definition of what is an APT, the idea is that it is the work of professional hackers, often state backed. This means they are competent, organised and resourceful, and have precise goals that they need to achieve, such as exfiltrating sensitive data or sabotaging machinery. Their attack campaigns can last for months, or even years, and are customised for their target. They will use existing tools if they are sufficient, but they do not hesitate to develop their own or to exploit 0-days (heretofore undisclosed vulnerabilities) when necessary. For example, one of their paramount concern is avoiding detection and they will rather spend time developing a new and undetectable tool than risk using an old and easily detectable one [4]–[6].

As we can see, APTs are a formidable threat, and defenses capable of detecting them are necessary. There is ongoing research [7], [8], but the existing tools on the market are not good enough [9]. An interesting fact we leverage is that APT actors do leave traces during their operations. For example, [10] found that in 86% of the cases they surveyed, there were traces of the breach in the log files. This is an even bigger failure on the defender's part if those traces were seen as attacks but dismissed as irrelevant. In this article, we present a possible solution for adding context to individual events and attacks in order to help with their detection and evaluation. In particular, we focus on checking whether individual attacks are part of larger attack campaigns, even if part of the attack campaigns have not been detected. This builds on top of our previous work on reconstructing attack campaigns [11] and adds the evaluation of potential attack campaigns against a model of APTs. Hence, the aim is to use some suitable model to check which reconstructed attack campaign is indeed an APT. More concretely, in the present paper, we aim to rank the different campaigns in order of most probably an APT to least probably an APT.

B. A stochastic model for APTs

In order to model APTs, we start by noting that despite each campaign being targeted and customised, every campaign follows the same generic steps, as described in [12]. First, there is a reconnaissance step, where the attacker gathers as much information about the target as possible, be it about the technology stack used or the people employed. Then, the attacker leverages this information to perform an initial compromise and establish a foothold inside the defensive perimeter. From then on, the attacker's goals are twofold: maintain their presence and find their targets. This leads to a number of steps similar to the ones described above but moving laterally inside the defensive perimeter instead. Finally, once the target is reached, the attacker will complete their mission, usually either exfiltrating data or sabotaging the target. In some cases, this is not the end of the mission and the attacker remains active and continues looking for more sensitive data. This generic scheme can be summarised with the following steps:

1) Reconnaissance
2) Compromission
3) Establish presence
4) Privilege escalation
   → Loop back to step 1 if necessary

5) Mission completion
   → Loop back to step 1 if necessary

In addition, the attacker can fail to execute any step which either brings them back a step or force them to start from step 1 again; in other cases, some steps can be skipped.

While quite generic, this model of the evolution of APTs is useful enough that it is used in most papers working on the detection of APTs [13]–[16]. In our case, we propose to model this attack strategy by using Markov chain. Indeed the attacker advances step by step, and the aim of a given step depends on what has already been achieved. On the defender's side, the current step of the attacker is not known directly, but it can be deduced from the actions observed. This behaviour can typically be modelled by using hidden Markov chains, that is, a chain which cannot be directly observed but whose states can be deduced from other observations. In this case, the observations we can use are the various alerts given by the defense mechanisms monitoring the system.

The rest of this article is organised as follows. In Section II, we present hidden Markov models and how to apply them for APT modelling. In particular, we introduce a score we designed to rank the potential APTs. In Section III, we use the proposed model in a detailed case study. Conclusions are postponed to Section IV.

II. 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 steps of an attack outlined in Section I-B. 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 intrusion detection system. As expected, they do not tell us directly which step the attacker is on, 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
$$\forall t,\ \mathbb{P}(o_t = O_j | s_t = S_i) = b_{ij} \quad \text{for } i \in [\![1, N]\!] \text{ and } j \in [\![1, M]\!]. \tag{4}$$
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 probability that this sequence of observations was generated by the HMM:
$$\begin{aligned}\mathbb{P}(o|\lambda) &= \sum_S \mathbb{P}(o|s, \lambda) \cdot P(s|\lambda) \\ &= \sum_S \mathbb{P}(o_L|s_L) \cdot \mathbb{P}(s_L|s_{L-1}) \cdots \mathbb{P}(o_1|s_1) \cdot \mathbb{P}(s_1).\end{aligned} \tag{6}$$
This can be efficiently computed using Baum's forward-backward algorithm [17]. In addition, we can also compute the most probable path which generated that sequence of observation as well as its probability in the model using the Viterbi algorithm [18]. The Viterbi algorithm 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 [19]. 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
$$A = \begin{bmatrix} 0.7 & 0.3 \\ 0.4 & 0.6 \end{bmatrix},\ B = \begin{bmatrix} 0.1 & 0.4 & 0.5 \\ 0.7 & 0.2 & 0.1 \end{bmatrix},\ \pi = \begin{bmatrix} 0.6 & 0.4 \end{bmatrix}. \tag{7}$$
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":
$$\mathbb{P}(o_1 = \text{"S"} | s_1 = \text{"H"}) \cdot \mathbb{P}(s_1 = \text{"H"}) = 0.06. \tag{8}$$
$$\mathbb{P}(o_1 = \text{"S"} | s_1 = \text{"C"}) \cdot \mathbb{P}(s_1 = \text{"C"}) = 0.28. \tag{9}$$
Then, for each state, we use the result of (8) and (9), multiply by the transition probability and take the

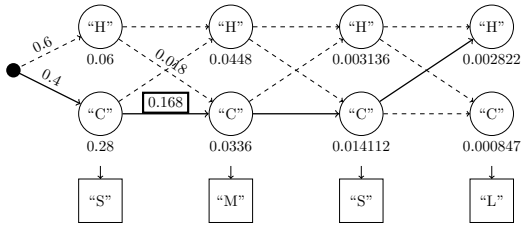

Fig. 1. Walking the trellis in the Viterbi algorithm for the "temperature of the year and tree ring size" example.

maximum of the two. For example, if the second state is "C":

$$\begin{aligned}\max(&0.06 \cdot \mathbb{P}(s_2 = \text{"C"}|s_1 = \text{"H"}),\\ &0.28 \cdot \mathbb{P}(s_2 = \text{"C"}|s_1 = \text{"C"})) = 0.168.\end{aligned} \tag{10}$$

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 our case this leads to the finding that the chain of observation "S-M-S-L" was most probably generated by the state sequence "C-C-C-H".

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

A. The 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 and matching different models. 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) [20] or the Bayesian information criterion (BIC) [21], 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 detection problem. In particular, 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. Then:

$$\begin{aligned}\mathcal{L} &= \ln(\mathbb{P}(o, s|\lambda))\\ \mathcal{L} &= \ln(\mathbb{P}(o_L|s_L) \cdot \mathbb{P}(s_L|s_{L-1}) \cdots \mathbb{P}(o_1|s_1) \cdot \mathbb{P}(s_1))\\ \mathcal{L} &= \sum_{l=1}^{L} \ln(\mathbb{P}(o_l|s_l)) + \sum_{l=2}^{L} \ln(\mathbb{P}(s_l|s_{l-1})) + \ln(\mathbb{P}(s_1)).\end{aligned} \tag{11}$$

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 (12)). These bounds are dependent on L but the ratio $\frac{\mathcal{L}}{L}$ is bounded by values independent of L (see Equation (13)). In keeping with the spirit of AIC and BIC we define our score as:

$$\mathcal{S} = -\frac{\mathcal{L}}{L}. \tag{14}$$

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

$$\mathbb{P}(s_{t+2} = S_j|s_t = S_i) = \sum_k a_{ik} \cdot a_{kj} = A^2(i, j). \tag{15}$$

One can then generalise (15) 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 the state S_i reached in k steps as $S_i^{(k)}$. Thus we have:

$$\mathbb{P}(s_{t+1} = S_j^{(k)}|s_t = S_i^{(l)}) = \mathbb{P}(s_{t+1} = S_j^{(k)}|s_t = S_i) = a_{ij}^{(k)}. \tag{16}$$

Note, in particular, that this does not depend on the value of l 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. If we simply rewrite the Viterbi trellis by adding the states reachable in up to K steps, then we duplicate each state S_i K times in the trellis. Hence, we have to divide each transition probability by K

$$L \cdot \ln(\min_{ij}(b_{ij})) + (L-1) \cdot \ln(\min_{ij}(a_{ij})) + \ln(\min_{i}(\pi_i)) \leq \mathcal{L} \leq L \cdot \ln(\max_{ij}(b_{ij})) + (L-1) \cdot \ln(\max_{ij}(a_{ij})) + \ln(\max_{i}(\pi_i))$$
$$2 \cdot L \cdot \ln(\min_{ij}(b_{ij}, a_{ij}, \pi_i)) \leq \mathcal{L} \leq 2 \cdot L \cdot \ln(\max_{ij}(b_{ij}, a_{ij}, \pi_i)), \tag{12}$$

$$2 \cdot \ln(\min_{ij}(b_{ij}, a_{ij}, \pi_i)) \leq \frac{\mathcal{L}}{L} \leq 2 \cdot \ln(\max_{ij}(b_{ij}, a_{ij}, \pi_i)). \tag{13}$$

to compensate the effect of this duplication. This means replacing Equations (3), (4) and (5) with, respectively:

$$\text{for } i, j \in [\![1, N]\!], \quad l \in [\![1, M]\!], \quad k \in [\![1, K]\!],$$
$$\forall t, \qquad \mathbb{P}(s_{t+1} = S_j^{(k)} | s_t = S_i) = \frac{a_{ij}^{(k)}}{K}, \tag{17}$$
$$\forall t, \qquad \mathbb{P}(o_t = O_l | s_t = S_i^{(k)}) = \frac{b_{il}}{K}, \tag{18}$$
$$\mathbb{P}(s_1 = S_i^{(k)}) = \frac{\pi_i^{(k)}}{K}. \tag{19}$$

We can then use Equations (17)-(19) 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 [19] with $K = 2$, we have the following matrices:

$$A^{(1)} = A = \begin{bmatrix} 0.7 & 0.3 \\ 0.4 & 0.6 \end{bmatrix}, \quad A^{(2)} = A^2 = \begin{bmatrix} 0.61 & 0.39 \\ 0.52 & 0.48 \end{bmatrix},$$
$$B = \begin{bmatrix} 0.1 & 0.4 & 0.5 \\ 0.7 & 0.2 & 0.1 \end{bmatrix},$$
$$\pi^{(1)} = \pi = \begin{bmatrix} 0.6 & 0.4 \end{bmatrix}, \quad \pi^{(2)} = \pi \cdot A = \begin{bmatrix} 0.58 & 0.42 \end{bmatrix}. \tag{20}$$

Walking the tree is done as in the previous example but using the new equations. From (19), we have $\mathbb{P}(s_1 = \text{“H”}^{(1)}) = \frac{\pi^{(1)}_{\text{“H”}}}{K} = \frac{0.6}{2} = 0.3$, $\mathbb{P}(s_1 = \text{“C”}^{(1)}) = 0.2$, $\mathbb{P}(s_1 = \text{“H”}^{(2)}) = 0.29$ and $\mathbb{P}(s_1 = \text{“C”}^{(2)}) = 0.21$. We can then compute the probability for each state that the chain generates the observation “S”:

$$\mathbb{P}(o_1 = \text{“S”} | s_1 = \text{“H”}) \cdot \mathbb{P}(s_1 = \text{“H”}^{(1)}) = 0.015. \tag{21}$$
$$\mathbb{P}(o_1 = \text{“S”} | s_1 = \text{“C”}) \cdot \mathbb{P}(s_1 = \text{“C”}^{(1)}) = 0.07. \tag{22}$$
$$\mathbb{P}(o_1 = \text{“S”} | s_1 = \text{“H”}) \cdot \mathbb{P}(s_1 = \text{“H”}^{(2)}) = 0.0145. \tag{23}$$
$$\mathbb{P}(o_1 = \text{“S”} | s_1 = \text{“C”}) \cdot \mathbb{P}(s_1 = \text{“C”}^{(2)}) = 0.0735. \tag{24}$$

Then, for each state, we use the results of (21)-(24), multiply by the transition probability from (17) and take the maximum:

$$\begin{aligned} \max(&0.015 \cdot \mathbb{P}(s_2 = \text{“H”}^{(2)} | s_1 = \text{“H”}), \\ &0.07 \cdot \mathbb{P}(s_2 = \text{“H”}^{(2)} | s_1 = \text{“C”}) \\ &0.0145 \cdot \mathbb{P}(s_2 = \text{“H”}^{(2)} | s_1 = \text{“H”}) \\ &0.0735 \cdot \mathbb{P}(s_2 = \text{“H”}^{(2)} | s_1 = \text{“C”})) \\ =&0.03822. \end{aligned} \tag{25}$$

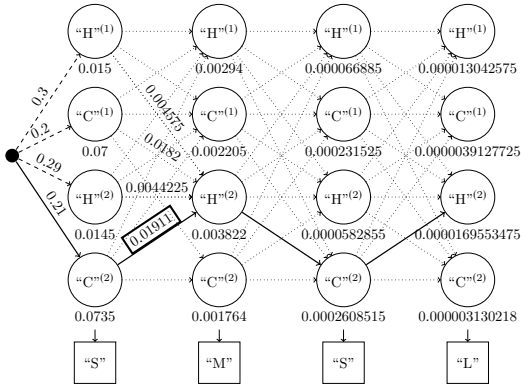

Fig. 2. Trellis of the Viterbi algorithm with up to K missing steps

We then multiply this result with the observation probability from (18) and obtain $0.01911 \cdot \mathbb{P}(o_2 = \text{“M”} | s_2 = \text{“H”}^{(2)}) = 0.01911 \cdot \frac{0.4}{2} = 0.003822$. We do this for each state at each step and once we reach the end of the trellis, we start at the state with the highest probability and walk back to find which succession of states lead to this probability. In this case, we find that the chain “S-M-S-L” was most probably generated by the sequence “$C^{(2)}$-$H^{(2)}$-$C^{(2)}$-$H^{(2)}$” which actually means that the observation sequence is “X-S-X-M-X-S-X-L” and the state sequence is “Y-C-Y-H-Y-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. It is interesting to note that if we compare this second result to the initial one of “C-C-C-H”, the only known state that changed is the second one, going from “C” to “H”. The associated observation is “M” which is the most ambiguous one; small rings are almost always due to a cold year and large rings to hot years, but medium rings could be either. Seeing only the state associated with this ambiguous observation change when the information is slightly modified, here simply making allowances for incomplete information, is remarkable.

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, 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 (14). This gives us a score which is not dependent on the length of the chain and which takes into account missed

observations.

III. 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 I-B: $N = 5$ and $S = \{$"reconnaissance","compromission", "establish presence","privilege escalation", "mission completion"$\}$. For the observations we use the output of a tool which defines seven 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 APT reports. These are publicly available reports about APTs. They are useful in knowing which tools are used in which step, but they are not precise enough in the evolution of the APT. Hence, they can be used to create the observation probability matrix B, but not the transition probability matrix A nor the initial probability vector π. For those, we use expert knowledge instead. 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". Additionally, individual steps can be removed from the scenario in order to obtain a better match with an APT. Fig. 3 shows three chains, one in each category, as they were rated by an expert. As we can see, these are the states of the chains, as the observation probability matrix B is not computed from these ratings. The answers from each expert are compared with each other expert by computing an agreement score [22]. This allows removing mis-rated models or even experts who never agree with anyone else. We can then use the best rated scenarios for learning the transition matrix A as well as the initial probabilities π. Once we have the model λ, we can check how it and our score perform.

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. 4. 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 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: Improving the score in the case with unobserved steps

The aim of this second experiment is to check that taking into account possible missing steps does improve the score. To do this, we take all the chains rated as APT and remove states so that their log-likelihood $\mathcal{L}^{(1)}$ of being an APT diminishes. We also add the two "Strong APT" chains of length 50 and 100, which are chains 9 and 10 respectively. We then recompute the score by taking into account possible missing observations. This is shown in Fig. 5. We take the APT chain presented in Fig. 3 (chain (A) in both figures). We compute its $\mathcal{S}^{(1)}$ score. We then remove states so that the score increases and obtain chain (D). Finally, we compute the $\mathcal{S}^{(2)}$ and $\mathcal{S}^{(3)}$ scores of the chain (D) with removed states.

The results can be seen in Fig. 6. First we plot the original score $\mathcal{S}^{(1)}$ in blue. We then remove some steps and plot in red the score $\mathcal{S}^{(1)}$ of the resulting chain. Finally, we compute the scores $\mathcal{S}^{(2)}$ and $\mathcal{S}^{(3)}$ taking into account up to two and three steps from one observation to the next respectively, and plot them in teal and green respectively. The figure shows that by taking into account missing steps, our score is capable of improving the score of chains whose missing steps make them appear as not APTs. In this test, the improved scores $\mathcal{S}^{(2)}$ and $\mathcal{S}^{(3)}$ usually ends up close to the $\mathcal{S}^{(1)}$ score of the original chain which is encouraging. We can also note that in certain cases, such as chains 1,2 and 7, $\mathcal{S}^{(2)}$ and $\mathcal{S}^{(3)}$ are the same. This is due to the fact that having three steps from one observation to the next is not always better than having to. In those specific chains, none of the cases where the score consider two steps to the next observation are improved when considering three steps instead.

D. Experiment 3: Evaluating the impact of unobserved steps

The aim of this first experiment is to show the influence of taking into account possibly unobserved steps on both APT and non-APT chains. The key point is that when evaluating whether a chain is an APT, we do not know if

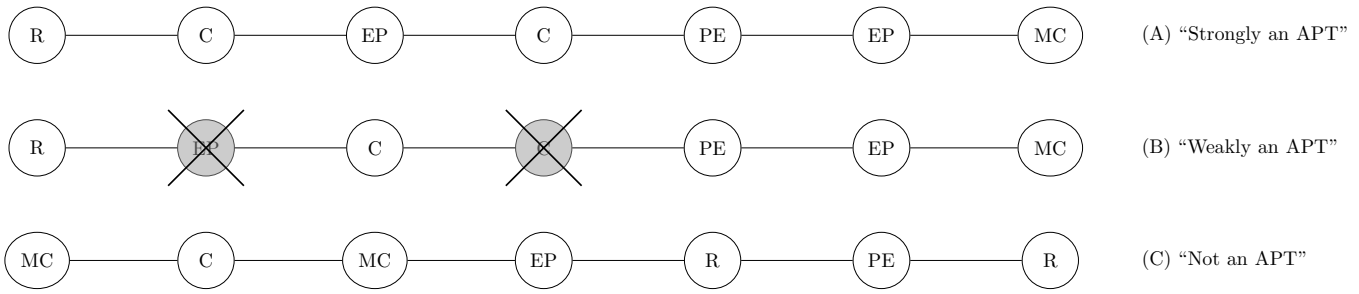

Fig. 3. Example of chains rated by experts. We note “reconnaissance” as “R”, “compromission” as “C”, “establish presence” as “EP”, “privilege escalation” as “PE”, and “mission completion” as “MC”.

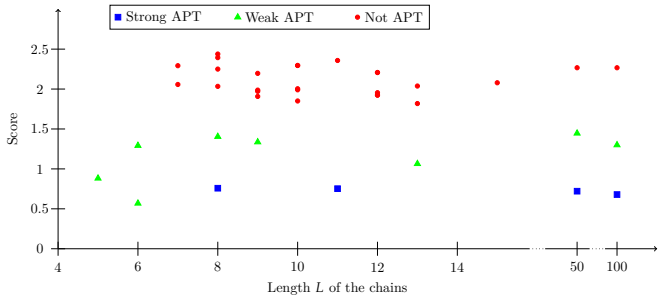

Fig. 4. Experiment 1: the score separates APTs from non-APTs

the chain is an APT, but we also do not know a priori if some steps are missing. Hence, we want to see the influence of the score when comparing APT and non-APT chains with and without missing steps. To do that, we start by removing at random up to 25% of the steps in each chain, so some chains remain untouched. We then compute the score allowing for up to three steps between each observation (i.e. $K = 3$). The results are shown in Fig. 7 and can be directly compared with Fig. 4; it uses the same scale and the same color code.

The first observation is that the scores are much lower in this experiment. This is to be expected since taking into account unobserved steps improves the score for every chain, and not just APT ones.

In addition, while the APT chains tend to score lower than the non-APT ones, the discrepancy is less clear than in the first experiment, with some APT chains having scores higher than some non-APT ones. This is to be expected since even chains which were originally APTs have possibly had random states removed, which means that we cannot assure that their log-likelihood of being an APT is still high. Since the score does not know if there are missing steps, it looks for the optimal unobserved steps to insert which means that the score of every chain, be it APT or non-APT and whether it has missing steps or not, is improved. If a chain's $\mathcal{S}^{(1)}$ points to it not being an APT because it is missing some crucial step, the chain's $\mathcal{S}^{(3)}$ can consider that crucial step as missing and improve on the original $\mathcal{S}^{(1)}$, which makes that chain much more likely to be an APT. This is exactly the effect we wanted when we introduced this score. Since APTs are conducted by skilled actors, they are more likely to invest more time and skill in crucial steps which are then less likely to be detected. It is paramount that our score can take that into account and improve the score of chains accordingly. Of course, there is a trade-off to make in the selection of the number K of unobserved steps possible between two observed steps because the score does not know which steps are missing and only considers the best possible solution. If K is too large, the score can always find a way to add missing steps to make the chain fit the model.

IV. Discussion and perspective

A. Are HMMs a good fit for modelling APTs?

The original idea for modelling APTs with HMMs stems from the fact that, while it is possible to detect unwanted activity, it is hard to determine what step of an attack campaign that activity is occurring at. And it is that second information, the step of the attack, which is the most important. As such, we needed a model where the actual state is unknown but partial information about the process is measured and is linked to the state of the process. This is something that HMMs excel at. However, in order to use HMMs, the modelled process must follow the Markov hypothesis, ie. the state of the process at a given time must only depend on the state of the process at the previous time. In this paper, we assumes that this hypothesis is followed. This is justified by the fact that an attack campaigns proceed step by step and that the direction each attack takes depends on the result of the previous step, with the attacker proceeding forward if the previous step succeeded or backtracking if it did not. While there is no complete assurance that this will always be the case, this assumption will be valid in most cases.

In addition, we can evaluate the performance of the model we created by polling our expert raters. As we have seen in Section III, the basic hidden Markov model does separate chains rated as non-APT from chains rated as APT. This is a very encouraging result. First, it seems to indicate that the assumptions we made above are correct and that HMMs can be used to model APTs. Second, it shows that the actual model we created is accurate. However, these results should be confirmed by a future large scale study of real-life APTs.

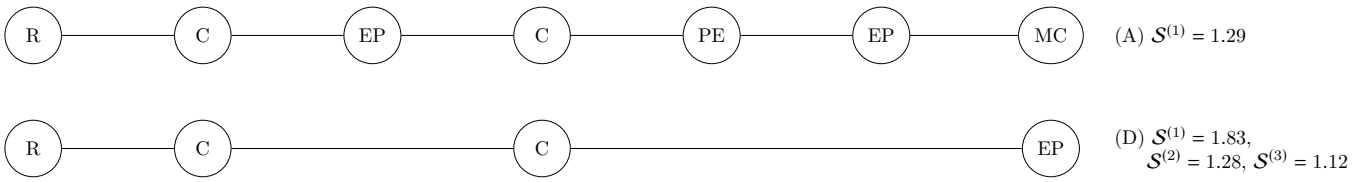

Fig. 5. An APT chain and the same chain with states removed. We note "reconnaissance" as "R", "compromission" as "C", "establish presence" as "EP", "privilege escalation" as "PE", and "mission completion" as "MC".

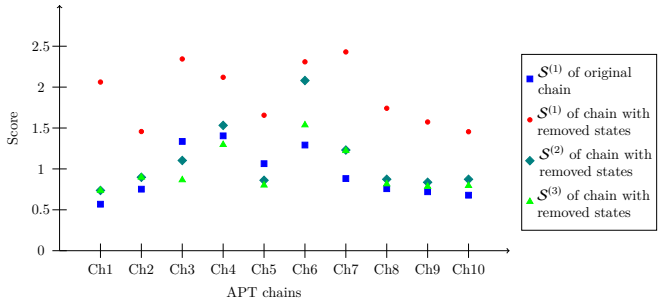

Fig. 6. Experiment 2: improving the score through missing states

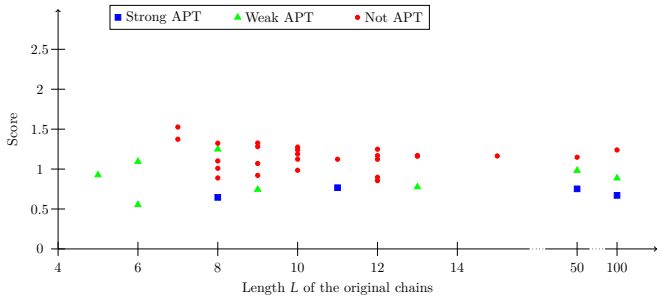

Fig. 7. Experiment 3: scores and recovered states

B. Can the score handle missing steps?

Our aim when we created the score presented in this paper was twofold. First, we wanted to be able to compare the score of chains of different lengths. We have proved in (13) that the score is bounded by constants of the model. As such, it does not increase with the length of the chain. This was also verified in Fig. 4 where we saw that even much longer chains still have scores consistent with shorter chains.

The second property that we wanted to integrate into our score is the ability to take into account potential missing steps. This is necessary because we cannot assume that every step of an APT has been detected. As we have shown in Fig. 6, our score clearly takes into account missing steps and places a chain with missing steps at roughly the same level as an original chain. However, as we can see in Fig. 7, our score improves the performance of every chain. This means there is a trade-off to make around the number K of consecutive missing steps possible. If K is as large as the number N of states in the model, then the score is able to find the optimal route between two measured steps every single time. In our case, this would make every chain into a potential APT, thus rendering the score useless. Since in our model, we have $N = 5$, we values of 2 or 3 for K, the latter being the value used in most experiments in this paper. The impact of the choice of the value K on the performance of our model and the determination of its optimal value are left for a further study. An idea that could be explored is the use of clustering algorithms, such as k-means, and the elbow method to find the optimal value. The idea would be to make sure that clusters corresponding to the expected classes, "APT" and "not APT" in our case, can still be identified.

References

[1] E. M. Hutchins, M. J. Cloppert, and R. M. Amin, "Intelligence-driven computer network defense informed by analysis of adversary campaigns and intrusion kill chains," *Leading Issues in Information Warfare & Security Research*, vol. 1, p. 80, 2011.

[2] F. Li, A. Lai, and D. Ddl, "Evidence of advanced persistent threat: A case study of malware for political espionage," in *Malicious and Unwanted Software (MALWARE), 2011 6th International Conference on*. IEEE, 2011, pp. 102–109.

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

[4] F. Labs, "Apt30 and the mechanics of a long-running cyber espionage operation," 2015. [Online]. Available: https://www2.fireeye.com/rs/fireye/images/rpt-apt30.pdf

[5] M. Labs, "Operation red october," 2013. [Online]. Available: https://kc.mcafee.com/resources/sites/MCAFEE/content/live/PRODUCT_DOCUMENTATION/24000/PD24250/en_US/McAfee_Labs_Threat_Advisory_Exploit_Operation_Red_Oct.pdf

[6] T. Micro, "Safe: a targeted threat," 2013. [Online]. Available: http://www.trendmicro.com/cloud-content/us/pdfs/security-intelligence/white-papers/wp-safe-a-targeted-threat.pdf

[7] B. Binde, R. McRee, and T. J. O'Connor, "Assessing outbound traffic to uncover advanced persistent threat," in *SANS Institute. Whitepaper*, 2011.

[8] K. D. Bowers, M. Van Dijk, R. Griffin, A. Juels, A. Oprea, R. L. Rivest, and N. Triandopoulos, "Defending against the unknown enemy: Applying flipit to system security," in *Decision and Game Theory for Security*. Springer, 2012, pp. 248–263.

[9] G. Ács Kurucz, Z. Balázs, B. Bencsáth, L. Buttyán, R. Kamarás, G. Molnár, and G. Vaspöri, "An independant test of apt attack detection appliances," 2014. [Online]. Available: https://blog.mrg-effitas.com/wp-content/uploads/2014/11/Crysys_MRG_APT_detection_test_2014.pdf

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

[11] 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.
[12] M. I. Center, "Apt1: Exposing one of china's cyber espionage units," 2013. [Online]. Available: http://intelreport.mandiant.com/Mandiant_APT1_Report.pdf
[13] S. Chandran, P. Hrudya, and P. Poornachandran, "An efficient classification model for detecting advanced persistent threat," in *Advances in Computing, Communications and Informatics (ICACCI), 2015 International Conference on*. IEEE, 2015, pp. 2001–2009.
[14] I. Friedberg, F. Skopik, G. Settanni, and R. Fiedler, "Combating advanced persistent threats: from network event correlation to incident detection," *Computers & Security*, vol. 48, pp. 35–57, 2015.
[15] A. Vance, "Flow based analysis of advanced persistent threats detecting targeted attacks in cloud computing," in *Infocommunications Science and Technology, 2014 First International Scientific-Practical Conference Problems of*. IEEE, 2014, pp. 173–176.
[16] Y. Wang, Y. Wang, J. Liu, and Z. Huang, "A network gene-based framework for detecting advanced persistent threats," in *P2P, Parallel, Grid, Cloud and Internet Computing (3PGCIC), 2014 Ninth International Conference on*. IEEE, 2014, pp. 97–102.
[17] P. A. Devijver, "Baum's forward-backward algorithm revisited," *Pattern Recognition Letters*, vol. 3, no. 6, pp. 369–373, 1985.
[18] G. D. Forney, "The viterbi algorithm," *Proceedings of the IEEE*, vol. 61, no. 3, pp. 268–278, 1973.
[19] M. Stamp, "A revealing introduction to hidden markov models," *Department of Computer Science San Jose State University*, 2004.
[20] H. Akaike, "A new look at the statistical model identification," *IEEE transactions on automatic control*, vol. 19, no. 6, pp. 716–723, 1974.
[21] G. Schwarz, "Estimating the dimension of a model," *The annals of statistics*, vol. 6, no. 2, pp. 461–464, 1978.
[22] M. Banerjee, M. Capozzoli, L. McSweeney, and D. Sinha, "Beyond kappa: A review of interrater agreement measures," *Canadian journal of statistics*, vol. 27, no. 1, pp. 3–23, 1999.