Detection of cyber-attacks in network control planes using Hidden Markov Model

Loïc Desgeorges, Jean-Philippe Georges, Thierry Divoux

Université de Lorraine, CNRS, CRAN, F-54000 Nancy, France
(e-mail: firstname.name@univ-lorraine.fr).

Abstract: Software Defined Networking (SDN) is a networking architecture within the control is centralized through a software-based controller. Like Cyber-Physical Systems manager, this centralization eases the support of advanced application. Being a single point of attack makes the controller a preferred target in case of attack. To enhance the control plane against cyber-attacks, an observer is introduced and is in charge of the detection of cyber-attacks on the nominal controller. In this objective, a detection of anomalies method in the activity of the control is proposed. This activity is defined as the events at the interface of communication between the controller and the network plant. In this paper, a non-deterministic control is considered which means that the decisions are stochastic. Hence, a probabilistic approach is proposed which aims to evaluate the deviation of the likelihood of the sequence of decisions taken by the controller. The formalism used to determine the likelihood is the Hidden Markov Model which permits to infer over the internal states of the controller through the observations. This method is discussed on a network case study.

Keywords: Detection, Security, Software-Defined Networking, Hidden Markov Models

1. INTRODUCTION

During the last decade, a huge activity in networking has on the Software-Defined Networking (SDN) architecture, as presented in Farhady et al. (2015). Fundamentally, SDN separates the control from the network devices. This control is then centralised through a software-based controller, which takes all the decisions related to the network. As a consequence, this architecture has several similarities with Cyber-Physical Systems (CPS) as presented in Molina and Jacob (2018) and in Fig. 1. However, from a security point of view, the controller is a preferential target as resumed in Kreutz et al. (2013). In case of an attack of the controller, the attacker has access to the whole network and can damage for example through a Denial of Service Attack by flooding the tables of the switches or by modifying the content of the commands sent by the controller as developed in Lee et al. (2017).

The topic of this paper is the security of the network control using a multi-controller architecture. Multi controller is already widespread in the literature for several reasons as presented in Li et al. (2017). One of these reasons concerns the safety aspect because, a failure of the controller inhibits the network service and a solution is to provide a redundancy of the main controller like in the work of Fonseca et al. (2012). Also, the use of a set of controllers presents some benefits in terms of security. First, there is the possibility to set up a decision-making security architecture as in Qi et al. (2016): to determine if rules coming from one controller are valid, a vote is launched between all the other controllers which limits the influence of a controller attack. Also, more recently, Blockchain has also been considered as an option to secure

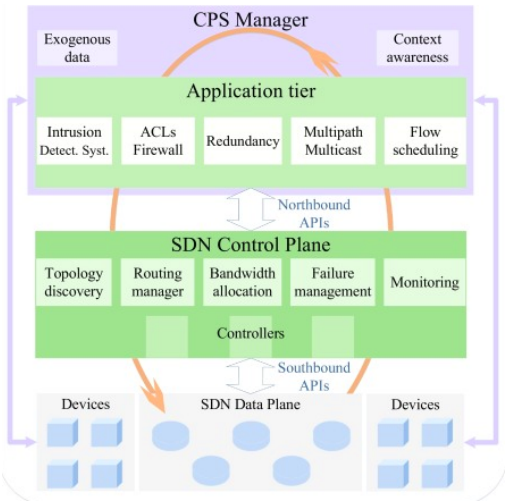


Fig. 1. SDN controller-based CPSs presented in Molina and Jacob (2018)

the control layer as in Yang et al. (2020). From another perspective, the use of the Moving Target Defense such as in Hyder and Ismail (2021) with the introduction of the notion of shadow controllers is also used to secure the control layer. Indeed, if a probing traffic is detected then a part of the shadow controllers are selected, randomly, in order to response to the traffic.

Regarding the related literature, it has to be noted that all approaches rely on communications between the controllers (the East-West interface in SDN). These interfaces are a threat in terms of security, as described in Kreutz et al. (2013). For instance, the decisions taken by one controller depend on the information given by

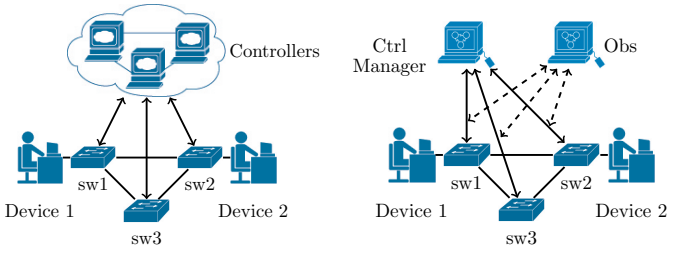
other controllers which might be malicious. That is why we proposed to develop a multi-controller architecture without communication between the controllers. The proposal consists of one observer which analyses the main activity of the controller and detects anomalies through this activity. In Desgeorges et al. (2021), the behaviour of the controller is assumed deterministic. Thus, when a decision is sent on the network, the observer expects the same as the one observed previously. As a consequence, the proposed algorithms cannot be used to detect anomalies of non-deterministic control (especially when it is based on machine learning or greedy algorithms). Therefore, the aim of the paper is to propose a method to detect anomalies for such non-deterministic control algorithm. In this objective, attention has been paid to probabilistic models as in Holgado et al. (2017), Shi et al. (2016) or Lefebvre et al. (2020).

The first part aims to introduce some preliminaries about the problem in section 2. Then, the detection problem and the proposals are developed in section 3. Then, the method is illustrated on a case study in section 4 and finally, a conclusion presenting the perspectives concludes the work.

2. PRELIMINARIES

2.1 Architecture SDN

The proposed multi-controller architecture is represented in Fig. 2. Without East-West interface, the detection method will not be based on the information from the other controller but on the capture of its activity (the messages exchange with the network infrastructure) and some *a priori* knowledge of the control logic.



(a) Classical mutli controller architecture

(b) The proposed multi controller architecture

Fig. 2. Differences between the classical multi controller architecture and our proposal

2.2 Formalisation of the language of the control

The exchanged packets between the controller and the set of switches, noted $\mathcal{N}$, are sent through the Southbound interface. This interface is normalized by the Openflow protocol ONF (June, 2012).

These messages are: the requests from the switches to the controller, the commands of the controller for the switches, the port status and the statistics from the switches. From the observer point of view then the set of events, Σ are: $\Sigma = \Sigma_{In} \cup \Sigma_{Out} \cup \Sigma_{Ps} \cup \Sigma_{Stats}$.

Firstly, Σ_{In}, corresponds to the "Packet_In" messages, named *pin*, which are the request from a switch to the controller. $\forall pin \in \Sigma_{In}$ there is $pin = (p, S, src, dst)$ with:

- $p \in \mathbb{N}$: the source port of the packet.
- $S \in \mathcal{N}$: the switch at the origin of the request.
- *src*: the IP source address of the packet.
- *dst*: the IP destination address of the packet.

The second type of events is related to the commands sent by the controller. These "Flow_Mod" events, noted $fmod$, are actions transmitted to the switch by the controller. They are stored by the switch in its flow table during a duration fixed by the controller. Also, as we will focus on routing application an action *act* ordered by the controller corresponds to the port of transmission of the flow. Thus: $\forall fmod \in \Sigma_{Out},\ fmod = (act, S, src, dst, idle, type_{fmod})$:

- $act \in \mathbb{N}$: the port on which the packet is transmitted.
- $S \in \mathcal{N}$: the switch destination of the packet.
- *src*: the IP source address of the packet.
- *dst*: the IP destination address of the packet.
- $idle \in \mathbb{R}^+$: the storage duration of the order by the switch.
- $type_{fmod} \in \{Add, Delete, Modify\}$: the type of the instruction.

Then, "Port_Status", noted *ps*, is a notification from the switches about the state of their ports. This means that there is an evolution of the network topology (at the data plane level). Then $\forall ps \in \Sigma_{Ps},\ ps = (reason, p, S)$:

- $reason \in \{Add, Delete, Modify\}$: the reason of the message: *Add* to notify the port was added, *Delete* if the port was removed and $Modify$ for a modification of the port state.
- $p \in \mathbb{N}$: the considered port.
- $S \in \mathcal{N}$: the switch source of the packet.

Finally, "MultiPart", noted *mp*, are statistics of the switches sent to the controller. These statistics are sent in response to requests of the controller through "MultiPartRequest" noted *stat*. According to ONF (June, 2012) there are several kinds of statistics given by the switch and in this work, we will only consider the statistics related to the flow. Then $\forall stat \in \Sigma_{Stats},\ stat = (byte, S, src, dst)$:

- $byte \in \mathbb{R}^+$: the number of bytes transmitted for the flow.
- $S \in \mathcal{N}$: the switch source of the packet.
- *src*: the source of the flow.
- *dst*: the destination of the flow.

Given the alphabet Σ, the set of all possible words of length $n \geq 1$ is noted Σ^n and defined as:

$$\Sigma^n = \{\sigma_1, \sigma_2 \dots \sigma_n | \forall i \in [1, n] \sigma_i \in \Sigma\} \tag{1}$$

For $n = 0$, $\Sigma^0 = \epsilon$ with ϵ the empty string. Hence, all the possible words are noted Σ^* and defined as:

$$\Sigma^* = \bigcup_{n \geq 0} \Sigma^n \tag{2}$$

All these packets are used by the controller in order to set up the data plane in a graph $\mathcal{G} \in \{\mathcal{N}, \mathcal{N}^2\}$ composed of the set of nodes ($\mathcal{N}$) and the set of links between the nodes ($\mathcal{N}^2$). Basically, a data plane $\mathcal{P}$ is a set of routes. There is one route by traffic demand, the total number of demand is noted $|demand|$. A route $\mathcal{R}$ is formalised as a set of

decisions $\mathcal{R} = \{D_1 \dots D_n\}$. With $\forall i \in [1, n] D_i \in \Sigma_{Out}$: the set of decisions taken by the controller which corresponds to the transmission port for a switch of the path to reach the next hop (i.e. the $fmod$ events). Then:

$$\mathcal{P} = \mathcal{R}^{|demand|} \in (\Sigma_{Out}^n)^{|demand|} \subset \Sigma^* \tag{3}$$

This data plane is set up by the controller and the first step for the observer is to verify that this data plane satisfies a necessary condition in order to be considered as consistent (Desgeorges et al. (2021)) such that each routes which compose the plan has to fulfil three properties: no loop, no dead node and the destination is reached.

To ease the readability, in what follows when we will refer to the decisions of the controller we will consider the consistent data plane $\mathcal{P}$ instead of the concatenation of the $fmod$ events which leads to the data plane.

3. DESIGN OF THE OBSERVER

The satisfaction for each route of the three properties defined above is necessary to evaluate the decisions but not sufficient. Indeed, an attacker may set up consistent, according to the three rules introduced, data plane which degrade the performance of the network. This section aims to present the detection method of the observer.

3.1 Detection Problem

A part of Σ^* corresponds to the language of the controller, noted L_{Ctrl}. Basically, it corresponds to the decisions taken in case of a nominal control (i.e. unfaulty) and L_{Att} is introduced as the decisions taken in case of attack. In this work, we do not consider the possibility to train the model under attack/faulty situations contrary to other works as Sampath et al. (1996) or Li et al. (2020). Then the words in case of attack are all the other words:

$$\Sigma^* = L_{Ctrl} \cup L_{Att} \tag{4}$$

Also, it is assumed that a cyber-attack which leads to a nominal behaviour of the control is not a problem regarding the control activity and so:

$$L_{Ctrl} \cap L_{Att} = \emptyset \tag{5}$$

The aim of our detection problem is, given a trace of the activity of the command $\mathcal{T} \in \Sigma^*$, to determine whether $\mathcal{T} \in L_{Ctrl}$ or $\mathcal{T} \in L_{Att}$. As an example the trace might be a request from a switch for a flow 1, pin_{flow1}, and the reaction of the controller which is the data plane, $\mathcal{P}$: $\mathcal{T} = \{pin_{flow1}, \mathcal{P}\}$. Another example, in case of a proactive controller where the routes are installed whatever the intents, this trace might be a sequence of data plane set up by the controller: $\mathcal{T} = \{\mathcal{P}_1, \dots \mathcal{P}_n\}$.

Moreover, a non-deterministic control algorithm is considered which means that the decisions of the control are stochastic. Even if techniques such as the residual approach proposed in Roth et al. (2011) are not directly applicable, the principle proposed in this paper shares the same base: evaluate the deviation between the observed behavior to a model of the activity of the command as defined in Isermann and Balle (1997).

During a learning phase the activity of the control, L_{Learn}, is observed. It corresponds to the nominal behavior of the control: $L_{Learn} \subset L_{Ctrl}$. Then, the question is: given a trace of the activity of the control $\mathcal{T}$, what is the deviation between $\mathcal{T}$ and L_{Learn}, i.e. are the decisions sent over the network likely compared to the activity learned?

To measure this gap, as the trace $\mathcal{T}$ has a length n it is necessary to consider the learned words with length n also. The learned language of length n is denoted as $L_{Learn}^n = \{\sigma \in \Sigma^n | \sigma \in L_{Learn}\} = \Sigma^n \cap L_{Learn}$ with Σ^n the set of words of length n.

Finally, a trace of the activity of the observed control $\mathcal{T}$ is part of L_{Ctrl} if and only if there is a word learned of a similar length n, W_n, for which $W_n \equiv \mathcal{T}$. This operator $\equiv$ is defined here after and detailed in the next section.

$$\mathcal{T} \in L_{Ctrl} \Leftrightarrow \exists W_n \in L_{Learn}^n | \mathcal{T} \equiv W_n \tag{6}$$

At the observation of a trace $\mathcal{T}$, this trace is part of the language of the controller (and so considered as consistent) iff there is a sequence W_n observed during the learning which is equivalent to this trace.

3.2 Evaluation of the likelihood

Since the decisions of the control cannot be predicted exactly (this is a stochastic process); the likelihood of the observed sequence is analysed. And so, we will assume that two sequences of decisions $\mathcal{T}$ and W_n are equivalent if and only if the ratio between the two likelihoods, $\mathscr{L}(\mathcal{T})$ and $\mathscr{L}(W_n)$, is less than a limit TD.

$$\mathcal{T} \equiv W_n \Leftrightarrow \frac{\mathscr{L}(\mathcal{T})}{\mathscr{L}(W_n)} < TD \tag{7}$$

There are several possibilities to determine the likelihood of a word W, $\mathscr{L}(W)$, which depends on the considered control. However, there is a constant: the decisions taken by the controller depend on the intern variables of the controller. Here, in the SDN proposed architecture there is no East-West interface with the controller in order to do not consider malicious information in case of attack. As a consequence, the observer do not have access to the evolution of the intern variables of the command algorithm. It just has access to the observation of the activity of the command, i.e. the events resulting of the evolution of the intern variables, which corresponds to the words defined in Σ^*. The process is represented in Fig. 3.

We assumed that the evolution of the intern variables of the controller follows a Markov Process. In fact, the decisions of the control depend on the previous state of the network which is the result of the previous decision. Thus, the Hidden Markov Model (HMM) formalism introduced in Baum and Petrie (1966) is used. This formalism has already been used in the context of the security of the SDN controller in Wang et al. (2018) for example or more generally in cyber security as in Keroglou and Hadjicostis (2016) or Holgado et al. (2017). The objective is to re-estimate the interns variables of the controller based on the

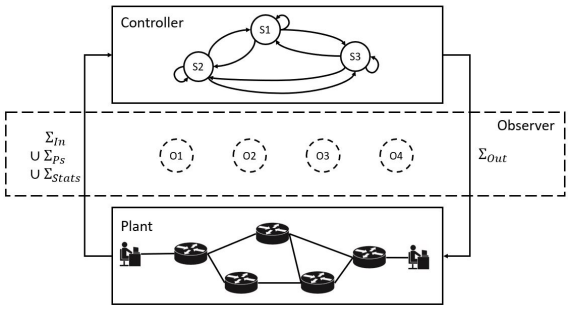


Fig. 3. Inference process

observation. To subsequently, determine the likelihood of an observation by inference over the internal states using the probabilistic theory such as the Bayes' theorem.

3.3 Definition of HMM

A Hidden Markov Models (HMM) is characterized according to Rabiner (1989) by the following:

- N: the number of states
- $S = \{s_1, \dots s_N\}$: the set of states.
- M: the number of observations
- $O = \{o_1, \dots o_M\} = \Sigma$: the set of observations.
- $A \in M_{N,N}$: the state transition probability distribution,
  $A = (a_{i,j} = p(q_t = s_j | q_{t-1} = s_i))$: represents the probability of moving from state s_i to state s_j
  $\forall i : \Sigma_{j=1}^{N} a_{i,j} = 1$
- $B \in M_{N,M}$: observation probability matrix,
  $B = (b_{i,j} = p(o = o_i | s = s_j))$: each expressing the probability of an observation o_i being generated from a state s_j
  $\forall i : \Sigma_{j=1}^{N} b_{i,j} = 1$
- $\pi \in M_{1,N}$: an initial probability distribution over states

Therefore, a HMM is defined by 3-tuple $HMM = (\pi, A, B)$, which depends on the parameters defined above..

Here, the HMM is from the observer point of view and so the set of observations $W_{Obs} \in O^n$ corresponds to a sequence of events of the activity of the control: $W_{Obs} \in \Sigma^*$.

HMM introduced three problems presented in Rabiner (1989) and resumed as follows:

(1) Given a HMM $\lambda = (\pi, A, B)$ and an observation sequence W_{Obs}, determine the likelihood $P(W_{Obs}|\lambda)$.
(2) Given an observation sequence W_{Obs} and an HMM $\lambda = (\pi, A, B)$, determine the best hidden state sequence $Q^* = argmax_Q p(Q|W_{Obs}, \lambda)$
(3) Given an observation sequence W_{Obs}, the set of states S, determine the HMM parameters π, A, B and $\lambda^* = argmax_\lambda p(W_{Obs}|\lambda)$

To solve these issues there are well-known algorithms proposed in the literature, as resumed in Ramage (2007):

(1) Forward-Backward algorithm: determines the likelihood of a sequence of observation.
(2) Viterbi algorithm: determines the best hidden state sequence given an observation sequence.
(3) Baum-Welch algorithm: determines the HMM parameters given an observation sequence.

Hence, to determine the likelihood of a sequence of observation the forward backward will be used during the running phase while the Baum-Welch algorithm is used for the learning phase to initialise the model as presented in Ramage (2007). Hence, the learning and the running phases are built on different sequence of data planes and independent.

4. CASE STUDY

Then, in this section the proposition is applied on a case study. First, the scenario is introduced and then the method is applied.

4.1 Scenario

The considered topology is the one of the network GEANT (with 23 nodes) which is the European data network for the research and educational community. The topology with 23 nodes and 37 links is considered and is represented in Fig. 4. The GÉANT topology is built and deployed in Mininet [1] .

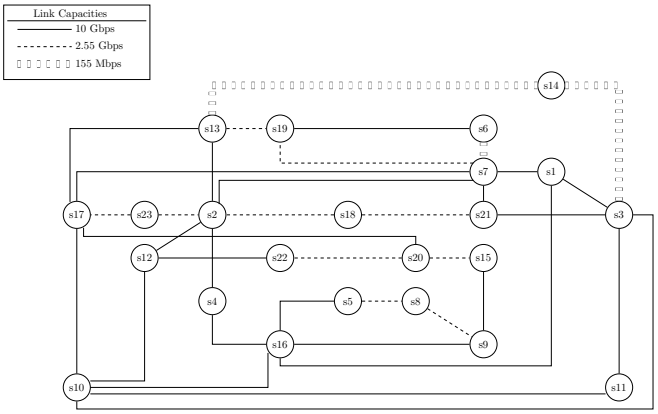


Fig. 4. Topology of the network GEANT with 23 nodes.

The considered traffic is the dataset TOTEM [2] which provides intra-domain traffic matrices for the GÉANT topology. The controller considered is the one proposed in Casas-Velasco et al. (2020) which is a multi-objective proactive routing through reinforcement learning technique. The metrics used are delay, packet loss and the available link bandwidth. This control is pro active, which means that periodically a new data plane is set up.

The aim of the considered attack is to do a degradation of the service by setting up malicious data plane as developed in Lee et al. (2017). To take control of the controller, the Kali Linux tools have been used.

4.2 Analysis of the results using HMM

The HMM proposition is evaluated depending on the depth of the considered sequence (i.e. the number of elements in a sequence).

[1] `http://mininet.org/`
[2] `https://totem.info.ucl.ac.be/dataset.html`

For what follows, we assume that no failure of a link at the level of the switches and no evolution of the traffic demand such that the context does not evolve $\Sigma_{Out} = \Sigma$.

First, let us describe the learning phase. A sequence of 2500 planes (which corresponds to one day of run) is used to determine L_{Learn}. The Baum-Welch algorithm to determine the parameters of the HMM. The number of internal states is set to $N = 3$. An id is associated to each data plane then an observation corresponds to a data plane. Here, 14 different data planes are observed: $\Sigma_{Out} = \{1, 2 \dots 14\}$.

Now, the controller is observed during one hour (which corresponds to a sequence of 125 planes) to constitute the nominal sequence. To simulate the attacker, for one hour, the data plane setting up is drawn randomly among Σ using a uniform distribution. According to the first problem of HMM, the likelihood of each observed sequence is determined using the Forward algorithm in order to determine whether it is consistent or not according to the equation 7. This equation can be reformulated more simply using the worst likelihood L_{WC} observe in L_{Learn} as follows:

$$\mathcal{T} \in L_{Ctrl} \Leftrightarrow \mathscr{L}(\mathcal{T}) < TD \times L_{WC} \tag{8}$$

Assuming : $TD = -1$ (as the log-likelihood is considered, it corresponds to a ratio of 10^{-1}). Then, the limit $LTD = TD \times L_{WC}$ depends on the depth considered and the values are reported in the table. 5.

Depth	LTD
1	-3
2	-5
3	-6.5
4	-8.5

Fig. 5. The used thresholds

For one nominal sequence and one attack sequence the log-likelihood is represented depending on the depth of the considered observation on Fig. 7. The alarms raised are represented by a point. If the depth = 1, the likelihood of the observed plan (without considering the last ones) is determined. Thus, each plane is considered consistent as far as it has already been observed them. Indeed, even if it is rare, it is possible. This means that this is not an issue to observe it again once, therefore it is important to consider deeper sequences. This is the reason why there is no alarm raised on Fig. 7a.

Considering greater depth makes the analysis more accurate. Even if the observed plan is consistent and one of the most frequently seen, the sequence does not correspond to the expected distribution. And so, the likelihood decreases significantly when a sequence of at least 2 observations is considered. Hence, deeper sequence permits to distinguish the abnormal sequence from the nominal ones as it can be observed by comparison on Fig. 7b, Fig. 7c and Fig. 7d: the number of false negative decreases when the depth increases and so the accuracy increases. Nevertheless, since $depth = 2$ the cyber-attack is detected.

To verify this phenomenon, let us study the function $f(\Sigma, depth)$, which evaluates the distribution between the nominal sequences and the inconsistent ones according to the *depth* of the sequence analysed, defined in equation. 9.

$$f(\Sigma, depth) = \frac{|L_{Ctrl}^{depth}|}{|L_{Att}^{depth}|} = \frac{|L_{Ctrl}^{depth}|}{|\Sigma^{depth}| - |L_{Ctrl}^{depth}|} \tag{9}$$

Here: $|\Sigma^{depth}| = |\Sigma|^{depth}$ and $|L_{Ctrl}^{depth}|$ is determined experimentally. Regarding our experiment the evolution of the function f is given in Fig. 6 for $depth > 1$.

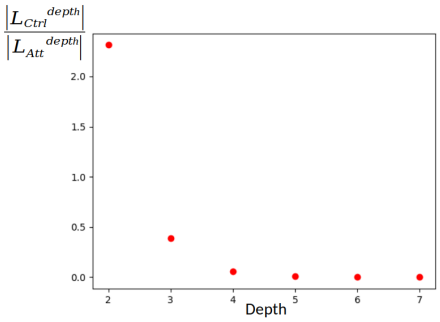


Fig. 6. Evolution of the function f depending on the *depth*.

For $depth = 1$, $\Sigma^1 = L_{Ctrl}^1$ which means that all the observed plan is considered consistent as soon as they have already been observed as already mentioned.

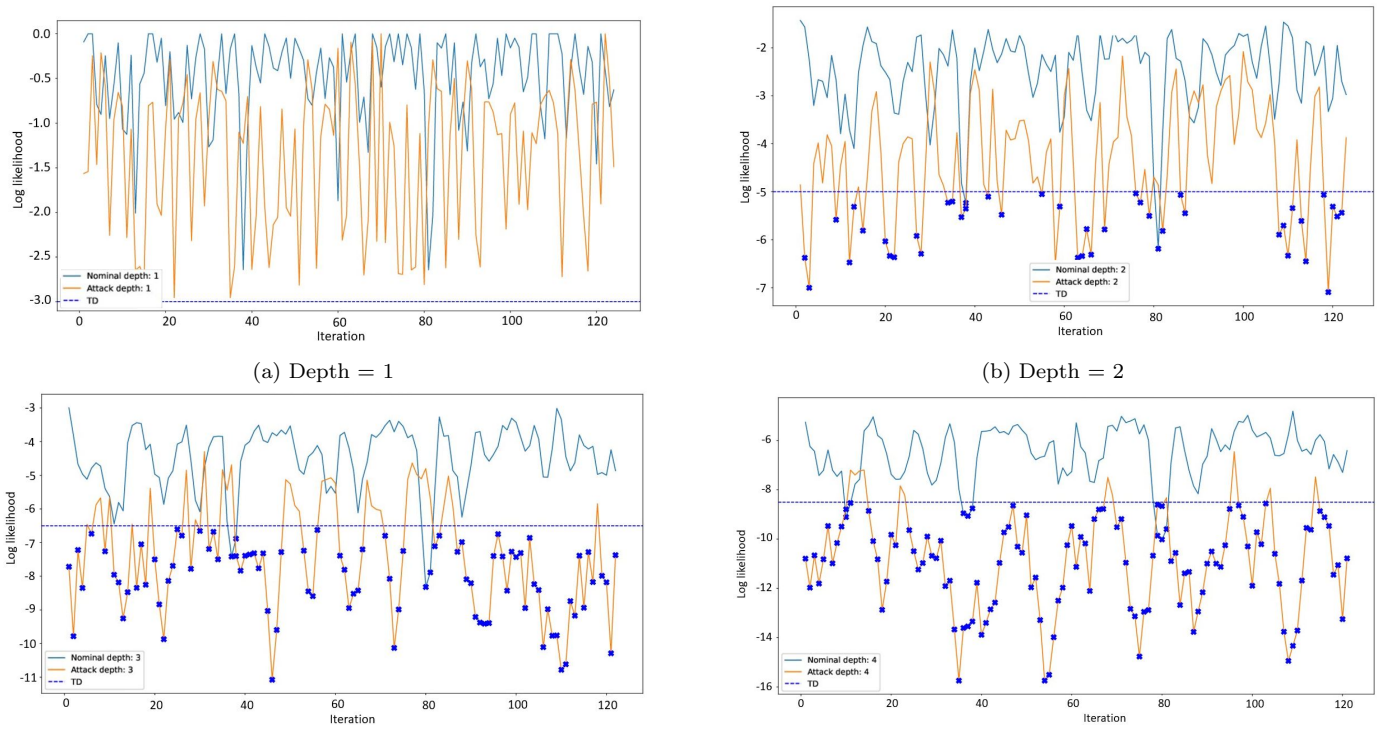
Then, increasing the length of the sequence permits to reduce the proportion of the consistent sequences in the set of observable sequences as it can be observed in the decrease of the points of Fig. 6. Hence, the greater the depth of the sequence considered by the observer, the more complicated it will be for an attacker to set up a sequence of data plane that could damage the network without being considered as inconsistent. This explains the increasing of the gap of the offsets on Fig. 7. Else, even if the controller is under attack but the activity of the command is still correct then there is no problem from the observer point of view.

It has to be mentioned that the complexity of the algorithm used to determine the likelihood (named forward algorithm) is $O(N^T)$ with N the number of states and T the depth of the sequence. As a consequence, the choice of the depth has to be a compromise between the time computation (which impacts the reactivity of detection) and the accuracy needed.

To conclude, the abnormal sequences, which corresponds to a cyber-attack, are detected for $depth > 2$ and the number of false negative decreases when the *depth* increases. This is due to the impact of the *depth* on the distribution between the nominal and abnormal sequence as represented by the evolution of the function f defined in the equation 9. Hence, $depth = 4$ is the more accurate solution.

5. CONCLUSION

As a conclusion, this work presents a technique to detect cyber-attacks in a control of network architecture. It relies on an observer checking the activity of the controller without communicating directly to the main controller to avoid the threats of the East-West interface. We proposed



(a) Depth = 1

(b) Depth = 2

(c) Depth = 3

(d) Depth = 4

Fig. 7. Log-likelihood of a nominal sequence compared to an attacking one.

to analyse the deviation between the observed sequence and the nominal sequences. The approach is based on the likelihood of the sequence and the Hidden Markov Model formalism has been used to determine such likelihood. Even if no formal conclusion can be provided, a case study shows the ability of the methods to detect the cyber-attacks of the control events and also the limits of the proposed method.

In future works, we aim at extending the detection algorithm, firstly, by comparing the results with other proposals also detecting cyber-attacks and the performance of HMM to other formalism such as Probabilistic Finite Automaton and Recurrent Neural Network. Also, new techniques to detect anomalies in the context of an encrypted communication between the controller and the switches is part of the perspectives.

ACKNOWLEDGEMENTS

This work was supported partly by the French PIA project "Lorraine Université d'Excellence", reference ANR-15-IDEX-04-LUE.

REFERENCES

Baum, L.E. and Petrie, T. (1966). Statistical inference for probabilistic functions of finite state markov chains. *The annals of mathematical statistics*, 37(6), 1554–1563.

Casas-Velasco, D.M., Rendon, O.M.C., and da Fonseca, N.L. (2020). Intelligent routing based on reinforcement learning for software-defined networking. *IEEE Transactions on Network and Service Management*, 18(1).

Desgeorges, L., Georges, J.P., and Divoux, T. (2021). A technique to monitor threats in sdn data plane computation. In *2021 IEEE 22nd International Conference on High Performance Switching and Routing (HPSR)*.

Farhady, H., Lee, H., and Nakao, A. (2015). Software-defined networking: A survey. *Computer Networks*, 81, 79–95.

Fonseca, P., Bennesby, R., Mota, E., and Passito, A. (2012). A replication component for resilient openflow-based networking. In *2012 IEEE Network operations and management symposium*, 933–939. IEEE.

Holgado, P., Villagrá, V.A., and Vazquez, L. (2017). Real-time multistep attack prediction based on hidden markov models. *IEEE Transactions on Dependable and Secure Computing*, 17(1), 134–147.

Hyder, M.F. and Ismail, M.A. (2021). Securing control and data planes from reconnaissance attacks using distributed shadow controllers, reactive and proactive approaches. *IEEE Access*, 9, 21881–21894.

Isermann, R. and Balle, P. (1997). Trends in the application of model-based fault detection and diagnosis of technical processes. *Control engineering practice*, 5(5), 709–719.

Keroglou, C. and Hadjicostis, C.N. (2016). Probabilistic system opacity in discrete event systems. In *2016 13th International Workshop on Discrete Event Systems (WODES)*, 379–384. IEEE.

Kreutz, D., Ramos, F.M., and Verissimo, P. (2013). Towards secure and dependable software-defined networks. In *Proceedings of the second ACM SIGCOMM workshop on Hot topics in software defined networking*, 55–60.

Lee, S., Yoon, C., Lee, C., Shin, S., Yegneswaran, V., and Porras, P.A. (2017). Delta: A security assessment framework for software-defined networks. In *NDSS*.

Lefebvre, D., Seatzu, C., Hadjicostis, C., and Giua, A. (2020). Probabilistic verification of attack detection

using logical observer. *15th IFAC Workshop on Discrete Event Systems, WODES 2020*, 53(4), 95–100.

Li, D., Wang, S., Zhu, K., and Xia, S. (2017). A survey of network update in sdn. *Frontiers of Computer Science*, 11(1), 4–12.

Li, Y., Tong, Y., and Giua, A. (2020). Detection and prevention of cyber-attacks in networked control systems. *IFAC-PapersOnLine*, 53(4), 7–13.

Molina, E. and Jacob, E. (2018). Software-defined networking in cyber-physical systems: A survey. *Computers & electrical engineering*, 66, 407–419.

ONF (June, 2012). *OpenFlow Specification v1.3 https://opennetworking.org/wp-content/uploads/2014/10/openflow-spec-v1.3.0.pdf, last visited the 14/09/2021.*

Qi, C., Wu, J., Hu, H., Cheng, G., Liu, W., Ai, J., and Yang, C. (2016). An intensive security architecture with multi-controller for sdn. In *2016 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, 401–402. IEEE.

Rabiner, L.R. (1989). A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2), 257–286.

Ramage, D. (2007). Hidden markov models fundamentals. *CS229 Section Notes*, 1.

Roth, M., Lesage, J.J., and Litz, L. (2011). The concept of residuals for fault localization in discrete event systems. *Control Engineering Practice*, 19(9), 978–988.

Sampath, M., Sengupta, R., Lafortune, S., Sinnamohideen, K., and Teneketzis, D.C. (1996). Failure diagnosis using discrete-event models. *IEEE transactions on control systems technology*, 4(2), 105–124.

Shi, D., Elliott, R.J., and Chen, T. (2016). Event-based state estimation of discrete-state hidden markov models. *Automatica*, 65, 12–26.

Wang, W., Ke, X., and Wang, L. (2018). A hmm-r approach to detect l-ddos attack adaptively on sdn controller. *Future Internet*, 10(9), 83.

Yang, H., Liang, Y., Yuan, J., Yao, Q., Yu, A., and Zhang, J. (2020). Distributed blockchain-based trusted multidomain collaboration for mobile edge computing in 5g and beyond. *IEEE Transactions on Industrial Informatics*, 16(11), 7094–7104.