A stateful protocol-based detection engine combining behavior use cases and system specifications

S. Seng, J. Garcia-Alfaro, I. Gazeau, L. Desmonts

A stateful protocol-based detection engine combining behavior use cases and system specifications

S. Seng[1,2] | **J. Garcia-Alfaro**[1] | **I. Gazeau**[2] | **L. Desmonts**[2]

[1]Télécom SudParis, France

[2]EDF R&D, France

Abstract

Faced with the increasing need for network monitoring, many detection methods have been proposed. In the last few years, AI-based methods, especially Machine Learning, have been the most popular. However, these methods are not yet fully operational and detection methods based on signatures or on specifications still keep all their legitimacy. In this letter, we propose a technique that combines a detection method based on protocol specification with a learning method train on a dataset specific to a use case. This combination leads to the definition of the notion of *protocol profile*.

Our solution is a continuation of a previous work which proposes an anomaly detection over-layer that are complementary to the pre-existing ones within a NIDS. The latter keeps its usual detection technique to which is added a stateful monitoring layer based on protocol specifications represented using Harel statecharts as well as our protocol profile layer. An algorithm has been proposed to automatically generate a protocol profile. It is based on event occurrence probabilities and an intermediate data format that we introduce: the Flow Graph Execution Log (FGEL). Other algorithms are also mentioned. A prototype has been realized and an experimentation with the POP3 protocol and simulated data sets has allowed to validate the concept.

KEYWORDS

Intrusion Detection System, Anomaly Detection, Protocol-specification-based detection, Protocol Modeling, Statechart

1 | INTRODUCTION

1.1 | Importance of Intrusion Detection System (IDS)

A Network Intrusion Detection System (NIDS) intercepts all the network packets that circulate within its perimeter. It must then determine for each packet whether it is normal, malicious or possibly an anomaly. There is no perfectly efficient solution for this packet classification action and this is the main challenge for NIDS: to obtain a high attack detection rate while limiting false-positives and false-negatives[1].

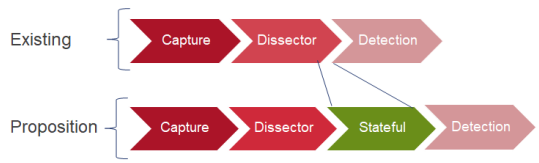

FIGURE 1 Addition of a stateful layer into actual NIDS

In a previous work[2], the authors represent the operation of NIDS using three successive layers: Capture, Dissection and Detection. They then propose the addition of an extra analysis dimension by adding a stateful monitoring layer of communication protocols between the Dissection and Detection layers (see Figure 1). This stateful monitoring layer is realized thanks to protocol models specified with Harel's statecharts[3].

The authors propose a tool implementing this layer, named STANPI (STAteful Network Protocol Intrustion detection tool). This tool corresponds to an extension plugin for the open source NIDS Zeek[4]. It instantiates a statechart for each session of a supported protocol. Each received network packet fires a transition in the statechart, which causes the active states to evolve and thus allows stateful monitoring. The authors insist that stateful monitoring is a complementary layer that does not replace the NIDS detection layer. Indeed, it offers anomaly detection capabilities on communication protocols as well as new contextual information made available to the detection layer (current states, exchange history, etc.). But it does not really have an attack detection logic. Moreover, it would be unable to detect attacks that respect the communication protocol.

1.2 | Proposal: protocol-profile-based detection

In this letter, we propose a new method that combines anomaly detection by protocol specification, as defined in[2], and behavioral anomaly detection by learning methods. More precisely, the objective is to have a model offering both anomaly detection capabilities on the protocol, but also on the usage of this protocol in a given use case, which we call a "protocol profile".

To reach this goal, we will start from the protocol model used in[2] (i.e. the representation of a protocol in Harel's Statechart) and we will propose a method to complete this model by a learning algorithm with a dataset of a given use case.

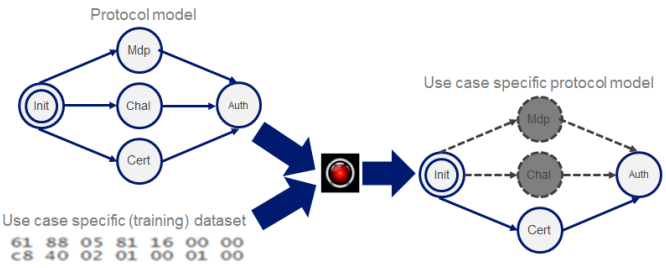

FIGURE 2 Extraction of sub-model from a use case dataset

As an example, Figure 2 shows our system which takes two inputs: 1) a model of a protocol accepting three authentication methods (password, challenge-response, certificate). 2) network traces of this protocol coming from a use case where only one authentication method is effectively used (certificate). The output of this system is an extract of the input model 1) from which the not applicable states to the use case have been invalidated.

A prototype has been realized, it follows the work conducted in[2]. The use case used is based on the POP3 protocol, with simulated datasets.

The rest of this letter is structured as follows. Section 2 elaborates further on our problem domain. Section 3 provide our contribution, while section 4 reports experimental and results work. Section 5 provides some additional discussions. Section 6 concludes the work.

2 | PROBLEM DOMAIN

Our protocol profile combines protocol specification-based methods with machine learning-based anomaly detection methods. Each of these two methods has been the subject of several studies.

The literature on specification-based detection techniques is abundant. However, these techniques do not have a convergent positioning and use among the different types of detection engines. Some authors[5], see it as a synonym for knowledge-based, while others see it as another type of behavior-based anomaly detection[6] or as a category in its own right[7]. Where opinions converge is in its definition: a model based on specifications defined by an expert to describe a system. This specification work is potentially complex, time-consuming and prone to human error. This divergence in positioning seems to stem from the system to be specified. If the specifications aim to model attacks, then these are more likely to be considered knowledge-based. If, on the other hand, the specifications aim to model a complete system, such as an IS, then they will be considered behavior-based.

In our study, it is not the final system that is modeled, but the protocol it uses: protocol-specification-based.

In contrast, anomaly-based detection techniques are the focus of the majority of recent proposed techniques. This is due in part to the recent explosion of AI methods, and in particular those based on machine learning, notably Deep Learning. Many of these studies report excellent detection rates, with very few false positives and false negatives. However, these excellent results need to be put into perspective[1,8].

Except for the joint use of specification and Machine Learning methods, the notion of combining or synthesizing these two topics seems to have been little studied.

3 | OUR PROPOSAL: GENERATE A PROTOCOL PROFILE FROM A SPECIFICATION MODEL AND A USE CASE DATASET

3.1 | Methodology

To generate a protocol profile, we partly reuse the work in[2], in particular the generic protocol model and the STANPI (STAteful Network Protocol Intrustion detection tool) anomaly detection engine based on a stateful monitoring. We then use the results of STANPI to generate an intermediary network exchange data format. This format will then be used to extract metrics that will lead to the generation of protocol profiles.

We will consider two phases of operation in our methodology: the first phase called TRAINING, in which the training algorithm is applied, leading to the generation of the protocol profile. The second phase, called RUN, corresponds to the nominal operation of STANPI. It then acts as a NIDS and detects anomalies with respect to the protocol and the protocol profile.

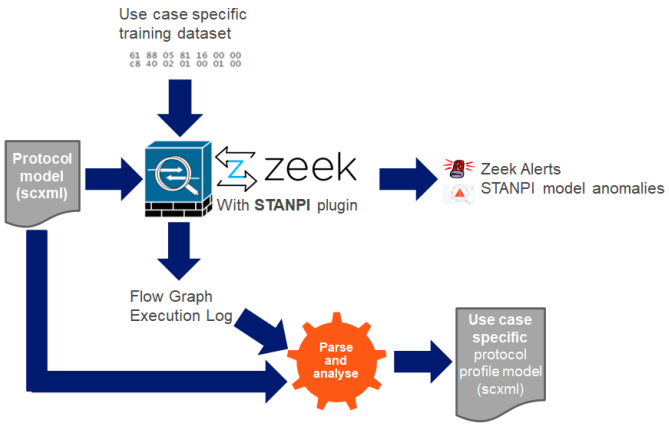

FIGURE 3 Training phase: protocol profile building chain

3.2 | Protocol Specification Model

The protocol model used is considered as input data. It contains the specifications of a protocol as they might be described by an expert. This model is generic and does not take into account the use made of the protocol; in other words, it is agnostic of any use case.

STANPI allows to follow in real time the evolution of the automata (statechart model) for each session of a protocol. It is thus possible to know the active states and the transitions fired. By recording this information in a log, we obtain what we call a Flow Graph Execution Log (FGEL). The figure 3 illustrates this methodology.

By analyzing the FGEL, we can extract metrics specific to the use of the protocol in a dataset, typically a training dataset. From these metrics, we can build a protocol profile. Several metrics and methods are possible, two of which are presented in the following sub-sections, and others are discussed in the following section 5.

3.2.1 | Naive method: unvalidate transitions

The most obvious and simplest method is to invalidate, in the automaton, transitions that are never crossed and, by transitivity, any states that are never reached. The construction of the protocol profile then consists in creating a new automaton from the FGEL.

Such a profile seems to meet our objective to build a protocole profile. Furthermore, it can be reused as is, as an input model in STANPI, replacing the generic protocol model (agnostic of any use case).

From an implementation point of view, we prefer not to create a new model as mentioned above, but to create a complementary model to the generic model, or to modify the generic model to integrate profile-related metrics. This allows us to maintain two distinct levels of anomaly: the generic model and the profile. Thus, when an anomaly occur, it is possible to distinguish whether the anomaly is a non-conformity with the generic model (and therefore also with the profile) or whether it conforms to the generic model but not to the profile.

This method of invalidating transitions is rigid and fails to identify certain anomalies related to the frequency with which transitions are used. For example, if our training dataset contained a single password authentication error, then the profile will never raise an anomaly when multiple password authentication errors occur (typically a bruteforce attack). Another example is the number of transmission errors in a protocol session. If the training data set contains a very low number of transmission

errors, then our profile will not raise any anomaly when a very high number of transmission errors occur. These two examples could be a sign of a potential attack or network problem that it might be worthwhile identifying. This rigidity can also lead to false negatives. We therefore propose another method based on probabilities.

3.2.2 | Probabilist method: Use occurency probability

Another method to build a profile is to define a probability of occurrence for each transition in the automaton. During the NIDS RUN phase, it is then possible to compare this probability of occurrence with the utilization rate of a transition, and thus eliminate potential anomalies in the event of divergence.

In practice, such a probability of occurrence is a static metric that does not take into account the possible diversity of exchanges in a training dataset. As it stands, it is an average value with no estimation error. This lack of tolerance is likely to result in a large number of false positives. To overcome this limitation, we introduce a margin of one standard deviation from this mean value.

Note that the probabilistic method is a superclass of the transition invalidation method. In fact, a probability of occurrence of 0% and a standard deviation of 0% are enough to invalidate a transition, all other values corresponding to a valid transition.

4 | EXPERIMENTATION AND RESULTS

4.1 | Use case based on POP3 protocol

In order to verify the relevance and the efficiency of our solution, we carried out an experiment. As a prototype, modifications were made to STANPI in order to take into account the notion of protocol profile. The goal of these experimentations was to confront our work to some datasets and see if it had the expected beheaviour in cases it should and should not raise an anomaly event.

Those trails were conducted by using the POP3 mail protocol as reference, as it is a protocol that is easy to read, which made the result's interpretation all the more efficient compared to a more complexe protocol. Figure 4 represents the generic model of the POP3 protocol in statechart in the W3C SCXML format[9].

We have not identified a freely accessible dataset that uses both the POP3 protocol and a use case sufficiently detailed for our context. We therefore simulated our own datasets. Our use case is represented by a normal user, wishing to retrieve his e-mails using the Mozilla Thunderbird e-mail client. The POP3 server used is Dovecot. Using a client such as Mozilla Thunderbird and a predefined server imposes sessions with their own POP3 message sequences.

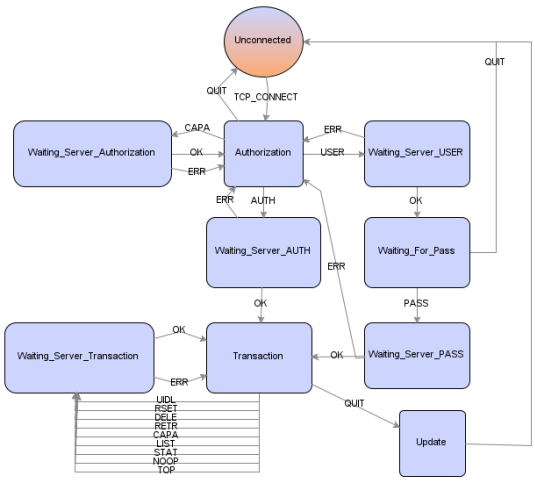

FIGURE 4 Automata model of POP3 protocol

Four datasets were then created:

- A training dataset containing normal POP3 exchanges, as representative and exhaustive as possible of our use case.
- A control dataset, representing the same use case and generated under the same conditions as the training dataset. However, the scenarios differ in terms of session duration and number of messages exchanged. This dataset is not expected to raise any issues.
- A control data set, in which a brute force authentication attack is performed. This data set is intended to detect anomalies.
- An alternative control dataset, in which an authentication other than that corresponding to the use case is used. Typically, AUTH authentication (CRAM-MD5) is used instead of USER/PASS. This dataset is intended to detect anomalies.

Training and control datasets must be healthy (i.e. free of attacks) and as representative and exhaustive as possible of the use case to be modeled.

4.2 | Modeling and STANPI Modifications

Changes have been made to the protocol models and STANPI to take account of protocol profiles. The SCXML format used for the model natively allows additional information to be added to each object, including transitions. For STANPI, the modifications consist in ensuring that no invalid transitions are crossed, or in removing any anomalies.

For the probabilistic method, in the model, we simply replace the binary flag with a probability of occurrence and a tolerance of one standard deviation.

From a design point of view, STANPI's profile management, in particular decision making, has been abstracted by isolating it in a dedicated library named Oracle. This makes it easy to evolve profile-related capabilities or methodologies, while limiting STANPI modifications.

4.3 | Results

The results show that our prototype works well. The brute force dataset also revealed anomalies with respect to the protocole profile caused by exceeding the occurency probability. The alternative authentication dataset did raise anomalies caused by an invalidation of the alternative authentication in the protocole profile.

About performance, in terms of hardware resources (memory, processor) or in terms of execution time, implementing the POP3 protocol profile within STANPI has a negligible overhead. It was not possible to measure a significant difference with or without a profile. The execution time of the training phase is similar to that of a RUN phase, which in turn depends on the NIDS Zeek. In any case, it remains much faster than real-time training if it is performed from a network dump (pcap). Finally, FGEL and profile generation takes just a few seconds on an inexpensive laptop.

This experimentation shows that the notion of protocol profile integrated into a NIDS allows to detect anomalies related to the non respect of a protocol use case. It also shows that the method of profile creation is efficient and that its implementation in STANPI has a negligible additional cost.

5 | DISCUSSION

5.1 | Evaluation Performances

The first point to note is that our experiment is a proof of concept, and so performance evaluation in terms of detection rates, false-negatives or false-positives, is not fully qualified. To measure these performances, we would need a quality training and control dataset, labeled and specific to a use case, which we do not have. Finally, even if our solution is capable of detecting attacks, we would point out that it can only detect anomalies in protocol usage, and that the role of attack detection or classification is the responsibility of the detection layer, to which our protocol profile layer provides complementary information.

5.2 | Profile Limitations and Other Methodologies

The two profile generation methods proposed, while relatively well adapted, do not take advantage of the latest advances in Machine Learning. In addition, our methodologies do not take into account two characteristics that may be specific to a use case:

- the temporal aspect of network exchanges, such as throughput and latencies between requests and responses, or periodic requests such as keep-alive requests issued at fixed intervals.
- sequences between message types: some use cases use sequences of messages, in a precise order within an existing set of message types.

Algorithms such as naive bayes are particularly effective for identifying sequences. Similarly, unsupervised learning algorithms such as Recurrent Neural Networks (RNN) could be effective for identifying sequences and temporal management of exchanges. It should be noted, however, that these AI algorithms offer little capacity for explanation[1].

5.3 | Improvement

The notion of oracle, especially if used in conjunction with prediction algorithms such as RNN, could solve a NIDS operating difficulty that STANPI does not take into account: packet loss. Indeed, packet loss could cause a protocol's state tracking within STANPI to become out of sync, leading to false-positive anomalies. For example, if a packet is lost during the authentication phase, the automaton may be blocked in the authentication phase, when in fact it has been completed. All subsequent packets are then likely to raise anomalies. Thanks to its predictive capabilities, the oracle could take charge of this packet loss, suggest the missing packets and resynchronize the automaton accordingly.

6 | CONCLUSION

In this letter we propose a new method for combining a network anomaly detection method based on protocol specifications with a learning method based on the behavior of a given use case. This combination of two methods led to the introduction of the notion of "protocol profile". A protocol profile then corresponds to the modeling of the network behavior for a specific system. Two algorithms have been proposed to generate this protocol profile: transition invalidation and probability of occurrence.

A model and dataset-specific exchange history format has been defined: the Flow Graph Execution Log (FGEL). It corresponds to an intermediate data format that enables an easy pivoting between several protocol profile generation algorithms.

A prototype has been built using the STANPI tool from previous work, which it complements with our new protocol profile layer, positioned between the stateful monitoring layer and the detection layer. Our NIDS, based on Zeek and STANPI, then operates using several layers: Capture, Dissection, Stateful Monitoring, Protocol Profile Monitoring and Detection. The last three layers offer anomaly or attack detection capabilities on different perimeters.

Limitations and avenues for improvement were also discussed. In particular, the possibility of generating protocol profiles from RNN-based AI algorithms.

REFERENCES

1. Seng S, Garcia-Alfaro J, Laarouchi Y. Why Anomaly-Based Intrusion Detection Systems Have Not Yet Conquered the Industrial Market?. In: Aïmeur E, Laurent M, Yaich R, Dupont B, Garcia-Alfaro J., eds. *Foundations and Practice of Security*Springer International Publishing 2022; Cham:341–354
2. Seng S, Garcia-Alfaro J, Laarouchi Y. Implementation of a Stateful Network Protocol Intrusion Detection Systems. In: 2022:398–405.
3. Harel D. Statecharts: a visual formalism for complex systems. *Science of Computer Programming.* 1987;8(3):231–274. doi: 10.1016/0167-6423(87)90035-9
4. Zeek . The Zeek Network Security Monitor. 2021.
5. Kaouk M, Flaus JM, Potet ML, Groz R. A Review of Intrusion Detection Systems for Industrial Control Systems. In: 2019:1699–1704. ISSN: 2576-3555
6. Mitchell R, Chen IR. A survey of intrusion detection techniques for cyber-physical systems. *ACM Computing Surveys.* 2014;46(4):55:1–55:29. doi: 10.1145/2542049
7. Liao HJ, Richard Lin CH, Lin YC, Tung KY. Intrusion detection system: A comprehensive review. *Journal of Network and Computer Applications.* 2013;36(1):16–24. doi: 10.1016/j.jnca.2012.09.004
8. Tavallaee M, Stakhanova N, Akbar Ghorbani A. Toward Credible Evaluation of Anomaly-Based Intrusion-Detection Methods. *Toward Credible Evaluation of Anomaly-Based Intrusion-Detection Methods.* 2010;40(5):516–524. Num Pages: 9 Place: New-York, NY Publisher: Institute of Electrical and Electronics Engineers.
9. Barnett J, Akolkar R, Auburn R. State Chart XML (SCXML): State Machine Notation for Control Abstraction. 2015.