Detecting APT through graph anomaly detection

Maxime Lanvin, Pierre-François Gimenez, Yufei Han, Frédéric Majorczyk, Ludovic Mé, Éric Totel

Detecting APT through graph anomaly detection

Maxime Lanvin[†], Pierre-François Gimenez[†], Yufei Han[*], Frédéric Majorczyk[‡], Ludovic Mé[*], Éric Totel[§]
[*]Inria, Univ. Rennes, IRISA, {firstname.lastname}@inria.com
[†]CentraleSupélec, Univ. Rennes, IRISA, {firstname.lastname}@centralesupelec.fr
[‡]DGA-MI, Univ. Rennes, IRISA, frederic.majorczyk@intradef.gouv.fr
[§]Samovar, Télécom SudParis, Institut Polytechnique de Paris, eric.totel@telecom-sudparis.eu

***Abstract*—Despite fruitful achievements made by unsupervised machine learning-based anomaly detection for network intrusion detection systems, they are still prone to the issue of high false alarm rates, and it is still difficult to reach very high recalls. In 2020, Leichtnam et al. proposed Sec2graph, an unsupervised approach applied to security objects graphs that exhibited interesting results on single-step attacks. The graph representation and the embedding allowed for better detection since it creates qualitative features. In this paper, we present new experiments to assess the performances of this approach for detecting APT attacks. We achieve better detection performances than the original work's baseline detection methods on the DAPT2020 dataset. This work is realised in the context of the Ph.D. thesis of Maxime Lanvin, which started in October 2021.**

***Index Terms*—anomaly detection, intrusion detection, graph analysis, machine learning**

I. Introduction

Over the recent years, the number of cyberattacks has soared. To supplement preventive security mechanisms, deploying Network Intrusion Detection Systems (NIDSes) to analyse network traffic and look for evidence of attacks in this traffic is common. Snort and Suricata are two examples of NIDSes, using signatures to detect known attacks among the traffic. Unfortunately, signature-based detectors cannot detect new attacks, like zero-day attacks. Anomaly detectors have been developed to overcome this difficulty. They approximate the system's normal behaviour with a model and detect deviations from it, i.e., anomalies. The excellent results obtained by Machine Learning (ML) for a decade in applications like natural language or image processing led to its use in anomaly detection. The unsupervised learning approach is especially interesting because it does not require labelled data.

However, current IDSes fail at detecting sophisticated attacks because they focus on single-step attack detection, leaving to the correlation phase [10] the task of identifying multiple steps of the same attack. This is an issue because nowadays attacks are becoming more complex, and they are performed in multiple steps. Notably, Advanced Persistent Threats (APT) aims at a stealthy and long-term presence targeting an IT infrastructure to gain access to highly sensitive data or harm the system's health. A typical APT attack includes the following stages, namely *reconnaissance* (the attacker discovers the information systems with, e.g., scanning techniques), *foothold establishment* (the attacker sets up a remote access), *lateral movement* (the attacker moves to other machines) and finally, the goal of the attacker is performed, for example, *data exfiltration*. These stages and others have been formalised in the MITRE ATT&CK matrix [9].

APT attacks are particularly challenging to detect. To this aim, this paper expands a previous work [5], namely Sec2graph, in which a "security objects graph" is built from the network traffic. The nodes of the graph represent network information (e.g., an IP address, a file transfer, or an HTTP connection), and they are linked when they appear in the same network event (e.g., an IP packet or a DNS request). Anomaly detection is then performed on top of this graph, using an AutoEncoder (AE). This approach exhibited good detection results in terms of recall and false-positive rate on both CICIDS datasets [8], [1].

However, the attacks included in these datasets are mainly single-step attacks. These datasets do not contain APT traces or are limited to the reconnaissance and foothold establishment phases, without any lateral movement and data exfiltration. Besides, the reference [7] has, on the one hand, proposed a new dataset called DAPT2020 containing full APT traces, and, on the other hand, presented results obtained with various unsupervised ML anomaly detection approaches (Stacked AE, Stacked AE with LSTM, and OneClass SVM) on classical datasets (UNB2015 [6], CICIDS 2017 [8], 2018 [1]) and their new dataset. The authors conclude that the detection on DAPT2020 is overall poor and that APT detection is particularly challenging. This demonstrates that new approaches are needed to detect such attacks.

In this context, our objective is to study how the security objects graph-based approach applied to CICIDS data could be extended to DAPT2020.

The rest of this paper is organised as follows. Section II presents a quick overview of the Sec2graph approach. Exploiting this approach, section III compares detection results on DAPT2020 with those presented in [7]. Finally, section IV concludes this paper and proposes future work.

II. Principle of anomaly detection based on security objects graph

The result we present in this paper were obtained using the analysis of the security objects graph introduced by [5]. The advantage of employing a graph-based representation is two-fold. First, it enables a much more interpretable view of the network traffic data. When an anomaly is detected, one

This work has been partly realised thanks to a doctoral grant from the cyber excellence pole (PEC : DGA, Brittany Region).

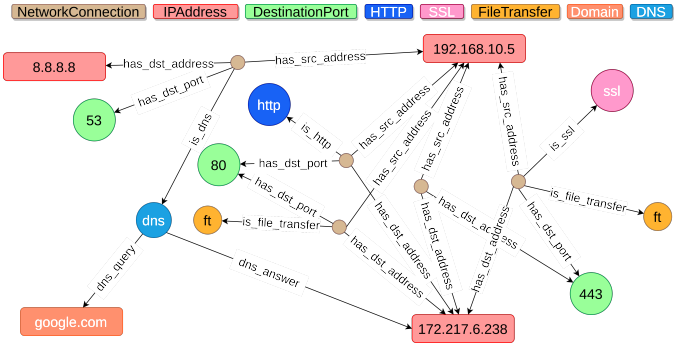

Fig. 1. Example of a security objects graph

can visualise in the graph what happened and quickly obtain an overview of the situation that would be harder to explain using the network logs directly. Second, it brings a compact representation of the verbose information delivered by network analysers and provides a notion of neighbourhood in the graph that doesn't exist in the log files. For example, with such security objects graph, it is easy to identify the IP addresses related to a domain name as they will be indirectly linked through a domain name. In contrast, in the network logs, these pieces of information are not close to each other.

This approach is composed of three steps: (1) the construction of the security objects graph from a set of network logs, (2) the encoding of this graph in a form usable by an AE [3], and (3) the use of this AE (learning then detection). During the training phase, the AE learns to rebuild the learning data. During the detection phase, the AE produces reconstructed vectors that are compared to input ones to get the reconstruction error. This error quantifies the anomaly. The following subsections detail these three phases.

A. Building a graph of security objects

Although the detection concerns network connections, all the information extracted from the IP layer to the application layer (HTTP request, mail, etc.) are considered. The raw data used to build the graph is a collection of network logs containing this kind of information. Due to the diversity of the information, the graph is heterogeneous.

Figure 1 shows an example of a security objects graph constructed from network logs. In this example, a client requested the IP address of the domain `google.com` via the DNS protocol, then contacted the `google.com` server with an HTTP request to obtain an SSL certificate before initiating an HTTPS connection with the same server. The graph presented in the figure is constructed from logs of these various protocols. The graph's nodes correspond to the network information in this scenario (client, DNS server, HTTPS server, network ports, network connections, SSL certificate, etc.). These nodes are directly linked when the information they represent appears in the same network event. The interested reader will find more technical details on this construction in [5].

B. Encoding a graph of security objects

The second step of the approach is to encode a graph in the form of vectors usable by the AE. This step is classically named "graph embedding" [4] in the ML literature, and its goal is to compress the topological structure information and the node attributes into a vectorised feature representation. In this embedding, each edge of the graph is processed independently and leads to a vector. This vector encodes the triplet (source node, edge type, destination node), thus encompassing only the immediate neighbourhood (i.e., within a one-hop distance). The one-hot encoding of the edge type, the source and destination nodes attributes are concatenated to create the vector of a triplet. One-hot encoding is a classical technique to embed categorical variables. The idea is to create for each variable a vector whose size is the number of categories. Each feature of this vector contains a 0, except the feature associated with the variable category that receives a 1. Discrete and categorical attributes are processed differently. For discrete attributes (such as port numbers or IP addresses), the number of possible categories is limited by keeping the most frequent categories and merging the rare categories into a single one. For continuous attributes (such as time duration or the number of packets exchanged), a clustering for each attribute is performed with Gaussian Mixture Model (GMM). The resulting cluster membership labels are used as the categories in the one-hot representation.

C. Anomaly detection

For this one-class classification problem, we use an AE to learn how to reconstruct the normal network traffic accurately. Then, during the detection phase, the AE produces a reconstruction error of an input network traffic record. Lower (resp. higher) reconstruction error indicates that the corresponding input is less (resp. more) likely to represent an anomaly.

In the detection phase, the reconstruction error for each edge of the security objects graph is computed. The average reconstruction error for all the edges relative to a single network connection is then computed and compared to a predefined threshold. Any link with an average reconstruction error above the threshold is considered as an anomaly and then its related network connection.

The reference [5] shows that this unsupervised approach is, on the CICIDS 2017 data, as good as or better than other supervised approaches of the literature. However, some attack types are poorly detected (infiltration and botnet attacks), and the false-positive rate remains quite high.

III. Experiments on the DAPT2020 dataset

DAPT2020 was produced to propose a dataset to assess IDSes on APT attacks. It offers a wide variety of attacks. In contrast with previous datasets, it includes every stage of an APT, including the final ones such as lateral movement and data exfiltration. The DAPT2020 dataset includes five days of traffic. The Monday network traffic is fully benign, whereas the other days contain one APT step per day, from the reconnaissance step to the data exfiltration step.

The authors of DAPT2020 performed several experiments on this dataset, using three different models: Stacked AE (SAE), SAE with LSTM layers and OneClass SVM. The detection performances are evaluated with the Area Under the Curve (AUC) of both the ROC and the Precision-Recall (PR) curves. Although SAE outperformed the other models on both metrics for all attacks, its performances are very low in comparison with supervised approaches. In this section, we evaluate and compare the performances of Sec2graph.

The experimental protocol is as follow. The authors of [7] learn one model per APT step: for each step, the training data is composed of the benign traffic located in all the days that do not include attacks related to this step. However, due to some labelling imprecision with their dataset, the extraction of benign traffic in the days that include attacks may be incorrect. In addition, their experimental protocol does not follow the current practices in IDS evaluation. For these reasons, Sec2graph model was learnt from the Monday data that contains no malicious traffic. Since we use only a subset of the training data used by [7], we can legitimately compare our results to theirs.

Furthermore, we discovered that the "data exfiltration" step was not correctly labelled: there are several errors in the csv file describing the malicious network connections. There are some swapped source and destination IP addresses and inaccurate timestamps. These errors are partially related to a buggy version of the tool CICFlowMeter (identified by [2]) used to create the DAPT2020 dataset. For this reason, we only performed the detection on the first three steps of the APT attack.

Our results are presented in Table I. The AUC for the ROC and PR curves are computed for each step of the APT attack. The *Sec2graph* column corresponds to our approach and the *SAE* column comes from [7]. For all three steps, namely reconnaissance, foothold establishment, and lateral movement, our approach performs as good or better than the SAE with regard to the two metrics. The greatest AUC for the ROC curve is reached for the foothold establishment step, which is also the case for the SAE. This behaviour is due to the high proportion of malicious traffic on this phase, according to the authors of [7].

Even if we outperform the SAE, the AUC of the PR curve does not have a value high enough to ensure a low false-positive rate. We are confident these results could be improved. For instance, the embedding we use with our model is certainly suboptimal when detecting some attack that requires a general view of the information system. Indeed, we have rather a local view since each edge is encoded independently.

TABLE I
DETECTION PERFORMANCES OF SAE AND OUR METHOD ON DAPT2020

	AUC ROC		**AUC PR**	
APT attack steps	*SAE*	*Sec2graph*	*SAE*	*Sec2graph*
Reconnaissance	0.641	0.888	0.262	0.613
Foothold Establishment	0.846	0.924	0.498	0.480
Lateral Movement	0.634	0.802	0.014	0.603

IV. CONCLUSION AND FUTURE WORK

The application of Sec2Graph to DAPT2020 shows that the generalisation of this approach is possible with minor adjustments. However, these results are still far from being satisfactory. Moreover, the AUC is not sufficient to judge the efficiency of the detection because it doesn't ensure to get both a low FPR and a high recall.

Thus, we plan to work on several enhancements to detect APT steps better while ensuring a low FPR.

In the case of APT attacks, one should explicitly take time into account in the detection process as APT attacks consist of different steps logically and temporally linked in a cyber kill chain. Therefore, we are working on including this temporal aspect in the detection either through the data representation or the model architecture. The authors of [7] got poor results using LSTM AEs, but thanks to the graph representation, we hope to find more interesting temporal patterns.

Furthermore, we plan to get a more comprehensive view of the information system to allow a finer-grained detection by adding other data sources like system logs. Some APT steps leave more traces at the system level than in the network traffic and could be detected more easily by analysing system logs. Combining network and system logs in a single graph structure can be a way to detect and link the different steps of an APT attack.

Finally, we also plan to work on graph embedding to leverage the graph structure better and extract more topological information. ML techniques depend heavily on the quality of the input data, so extracting more relevant features from the graph and embedding them into vectors could improve the detection results.

REFERENCES

[1] CSE-CIC. A realistic cyber defense dataset (CSE-CIC-IDS2018), https://registry.opendata.aws/cse-cic-ids2018, 2018.

[2] ENGELEN, G., RIMMER, V., AND JOOSEN, W. Troubleshooting an intrusion detection dataset: the CICIDS2017 case study. In *SPW* (2021), pp. 7–12.

[3] GOODFELLOW, I., BENGIO, Y., AND COURVILLE, A. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[4] GOYAL, P., AND FERRARA, E. Graph embedding techniques, applications, and performance: A survey. *Knowledge-Based Systems 151* (2018), 78–94.

[5] LEICHTNAM, L., TOTEL, E., PRIGENT, N., AND MÉ, L. Sec2graph: Network attack detection based on novelty detection on graph structured data. In *DIMVA* (2020), pp. 238–258.

[6] MOUSTAFA, N., AND SLAY, J. UNSW-NB15: a comprehensive data set for network intrusion detection systems. In *MilCIS* (2015), pp. 1–6.

[7] MYNENI, S., CHOWDHARY, A., SABUR, A., SENGUPTA, S., AGRAWAL, G., HUANG, D., AND KANG, M. DAPT 2020 - constructing a benchmark dataset for advanced persistent threats. In *MLHat* (2020), pp. 138–163.

[8] SHARAFALDIN, I., LASHKARI, A. H., AND GHORBANI, A. A. Toward generating a new intrusion detection dataset and intrusion traffic characterization. In *ICISSP* (2018).

[9] THE MITRE CORPORATION. Mitre att&ck, https://attack.mitre.org/, 2015.

[10] VALEUR, F. *Real-Time Intrusion Detection Alert Correlation*. PhD thesis, University of California, Santa Barbara, USA, 2006.