DAEMON: dynamic auto-encoders for contextualised anomaly detection applied to security monitoring

Alexandre Dey, Eric Totel, Benjamin Costé

DAEMON: Dynamic Auto-Encoders for contextualised anomaly detection applied to security MONitoring

Alexandre Dey[1,3], Eric Totel[2], and Benjamin Costé[3]

[1] IRISA
alexandre.dey@irisa.fr
[2] SAMOVAR, Telecom Sud-Paris, Institut Polytechnique de Paris
eric.totel@telecom-sudparis.eu
[3] Airbus Cyber Security
benjamin.b.coste@airbus.com

Abstract. The slow adoption rate of machine learning-based methods for novel attack detection by Security Operation Centers (SOC) analysts can be partly explained by their lack of data science expertise and the insufficient explainability of the results provided by these approaches. In this paper, we present an anomaly-based detection method that fuses events coming from heterogeneous sources into sets describing the same phenomenons and relies on a deep auto-encoder model to highlight anomalies and their context. To implicate security analysts and benefit from their expertise, we focus on limiting the need of data science knowledge during the configuration phase. Results on a lab environment, monitored using off-the-shelf tools, show good detection performances on several attack scenarios (F1 score ≈ 0.9), and eases the investigation of anomalies by quickly finding similar anomalies through clustering.

Keywords: Anomaly Detection, Heterogeneous log analysis, Human-automation cooperation, Intrusion Detection, Machine Learning

1 Introduction

Security monitoring of large information systems is often entrusted to Security Operation Centres (SOC). Within a SOC, the activity of the monitored system is recorded in the form of security events. These events can come from various sources, including endpoint monitoring (e.g., process auditing, anti-viruses, file integrity monitoring, etc.), network layer monitoring (e.g., network flow, network intrusion detection system, etc.), as well as application logs (e.g., web server logs, authentication logs, etc.). Security Information and Event Management (SIEM) systems are often used to collect and analyse these events.

SOC analysts perform intrusion detection of monitored systems thanks to automated real-time detection systems that recognize known attacks often relying on sets of correlation rules. Moreover, the threat hunting analysts uncover unknown attack methodologies by exploring events in the search of potentially

suspicious activities. The number of false alarms raised by the real-time detection systems should be kept to a strict minimum in order to react as quickly as possible to known attacks and give threat hunters time to perform more thorough analysis. However, due to the huge amount of data that needs to be analyzed and pieced together to investigate anomalous (and potentially malicious) patterns, the threat hunting process greatly benefits from automation tools.

In recent years, multiple anomaly detection methods have been applied to security monitoring. One approach consists in prioritizing events so that the most anomalous ones are presented first to the analysts. However, multiple limitations still slows the adoption of these methods that are mainly based on machine learning. First, most machine learning based methods require extracting numerical metrics from the security events [6, 19]. This step can be tedious, especially for security analysts that often lack the required data science knowledge. While novel deep learning advances can partially alleviate this limitation by being more flexible regarding the input data [7], the results they provide may still be hard to interpret. This lack of explainability can be lowered by presenting contextualized events to the analysts [16]. Recently, multiple approaches have been applied to endpoint monitoring with interesting results such as information flow tracking [10, 2] and event causality [29]. However, in the context of a SOC, it is unlikely to have access to the level of detail necessary to find causal links between events, and we need to rely on approximations instead.

In this paper, we propose a method to detect traces of anomalous and potentially malicious activity by analysing security events coming from multiple sources (e.g., process auditing, network probes, web proxies, etc.). Our method does not require specific information from logs, and can therefore be configured for various monitoring strategies (i.e., ability to configure it for various sources of events, and various levels of visibility). The first step of this method is to regroup events into sets that describe the same action (e.g., a network connection seen as an opened socket by the endpoint monitoring tool, and as a network flow by network probes). To analyse these sets, we use an auto-encoder model, a specific type of deep neural network that is particularly suited for anomaly detection. We design the model so that it can analyse any attribute contained in security events, and link information from all the events within each set to detect anomalies based on the context (e.g., reading `/etc/password` at startup is normal for a web server, but reading the same file when answering a request might not be). The model can learn incrementally to adapt to the evolution of the normal behaviour of a system. To account for changes in the monitoring strategy, the structure of the model can be updated without requiring a complete retraining (which is computationally intensive). Due to the large volumes of events that needs to be analysed, whenever possible, we choose algorithms that can benefit from parallel and distributed computing. We give a particular attention to integrate easily into SOC analysts habits and capabilities. Specifically, the configuration step do not require advanced knowledge in data science, and instead focuses on available expertise from SOC analysts. The first contribution of our work is an event fusion method that can benefit from parallel

and distributed computing. The second is a neural network auto-encoder with dynamic structure that can be trained continuously, can be adapted based on the monitoring strategy of the system and can highlight anomalous behaviours based on the sets of events that describe them.

In section 2, we present security monitoring related work. The method used to regroup events into sets is explained in section 3, and the neural network that is used to analyse these sets is described in section 4. We assess our method and present results in section 5.

2 Related work

2.1 Anomaly detection and security log analysis

Security event analysis can be seen as a special case of log analysis. He et al. [11] recently provided a comprehensive review of anomaly detection in logs. Essentially, the anomalous activity identification process can be decomposed in three major steps, namely, parsing the logs (i.e., going from unstructured logs to structured events), extracting interesting features and finally using an anomaly detection algorithm. Most machine learning algorithms require these features to be numerical [30, 15]. Authors have applied these methods to security log analysis [27, 17, 6] but the quality of the results are in these cases more dependant on the quality of the feature extraction and transformation (for non numerical features) processes, which can be hard to define for a security analyst with no background in data science. Recently, Dey et al [5] proposed methods that ease these processes by using deep learning method to be more flexible regarding the input data. However this method lacks the context that is valuable for investigating the alerts. Bertero et al. [1] and Du et al. [7] drew inspiration from natural language processing to take into account the context of the events. Debnath et al. [4] proposed a generic framework to automate the parsing phase of log analysis. However, both approaches fail to analyze the attributes of the events which make them less effective for security logs. For example, they would not detect a connection to an automatically generated command and control domain because they would see a normal DNS request followed by an HTTP(S) connection and would not see that the domain name looks weird. Pei et al. [21] proposed to represent security events as a graph and identifies communities as potential attacks. The features they use to weigh the edges between the events are learned in a supervised manner from examples of attacks, and is therefore not applicable in our unsupervised context. Liu et al. [14] chose to model users interactions by using a complex set of rules. While the approach is relevant for the author's use-case (insider threat scenario), it is difficult to adapt for a SOC which monitors multiple systems and where human resource is too scarce to fine-tune such a method specifically for each of these systems.

We want to model all the relevant information contained in security events as in [5], while taking the context of the log into consideration as in [1, 7, 4].

2.2 Alert correlation

Alert correlation methods are already part of SOC tools. Valeur [26] described extensively the correlation process, which can be decomposed into four major steps, the **normalization**, the **enrichment** and the **aggregation** of events, which consists in fusing similar alerts and correlating multiple steps of the attack to detect known patterns. The last step is a **global analysis** of these events.

In our context, we analyze both events and alerts (i.e., every behaviour, not only suspicious ones). This allows the detection of novel attacks, but also implies that the volume of information to process is greatly more consequent than what is processed by an alert correlation system. In this paper we do not focus on the normalization and enrichment of events as these are already part of SOC processes, and extensive public resources are already available for these (e.g., the Elastic Common Schema [8] and MITRE ATT&CK® data sources [18]). Instead we focus on the fusion part of the aggregation process to propose a solution that can process the high volume of events in parallel. Solutions for automatically identifying which events should be aggregated has already been studied in previous work [24, 28]. In our case, we adapt a model that is already common for alert fusion, by aggregating events that are close in time and using simple logic rules to separate unrelated events within these time windows.

3 Fusion of events in sets of meta-events

3.1 Events for security monitoring

The notion of data source From one IT system to another, the sensors that are deployed are likely to be different. In fact, each system has its own set of application (and the associated logs), its specific antivirus, NIDS, firewalls, etc. This can prove to be challenging when adapting a monitoring tool to a new IT system. However from the functionality point of view, there is often a significant overlap between two different systems. For example, two different IDSes still serve the same purpose, major OSes rely on a similar definition of a process, etc. The normalization process is supported by distribution of events into data sources. Data sources are categories of events often associated with a non-exhaustive list of valuable pieces of information called attributes that can be extracted from these events. Famous framework MITRE ATT&CK® proposes a few examples of data sources [18]. In our case, each data source is associated with a list of normalized attributes that are relevant for anomaly detection.

Definition of an event In an information system, activity is recorded by sensors inside logs. Each action is recorded differently by the various sensors (e.g., IDS will focus on information relevant for security, while application logs aim at facilitating diagnosis) in the form of events, with the aim of providing an audit trail that can be used by human analysts to better understand the activity of the system. An event can be decomposed into attributes (e.g., an antivirus log will contain information about the machine, the suspected file, the rule that triggered the event, etc.), among which only a subset may have value for security

monitoring. In essence, events can be seen as sets of key-value pairs that can be understood by humans. Normalization of events attributes is performed thanks to data sources attributes defined above.

Definition of a meta-event Among the events collected when monitoring an information system, finding duplicated events is highly probable. This can be attributed mostly to four facts. **(1)** The same action being performed several times in a short period of time (e.g., a process spawning a pool of child processes). Due to the asynchronous nature of modern computing, it is unlikely to have the exact time-stamp of each events. **(2)** The sensor that registers the events or the chosen attributes of these events might not provide enough information to distinguish two similar but different actions (e.g., multiple threads connecting to the same service in parallel will generate socket events that can be hard to distinguish if the source port of the connection is not considered). **(3)** Redundant sensors or multiple implementations of these sensors (e.g., two different IDSes) for high availability. **(4)** For performance reason, the log shipping method of a lot of modern monitoring solutions often follow the "at least once" delivery policy by default rather than the "exactly once" policy.

For this reason, we chose to consider a meta-event as a group of events pertaining to the same data source during a predefined period of time and with equal attributes[4]. As explained in Introduction, the proposed system aims to leverage knowledge of SOC analysts. The definition of a meta-event therefore depends on three choices, namely, the data sources, the size of the time window inside which events can be grouped together, and the list of attributes that needs to be equal for events to be grouped. As long as these choices remains in the hands of analysts, the performance of our system depends on assignments made. For the sake of this work, we define time window, data sources and event attributes based on expert knowledge provided by SOC analysts.

3.2 Linking meta-events

When monitoring an information system, the various sensors will record events differently (e.g., an endpoint audit mechanism may attribute network socket opening to processes, while network probes will focus on analyzing the network protocol). To extract as much information as possible from available logs, we regroup all the events related to the same action and jointly analyze them. This is known as the alert fusion step of the correlation process [26], and we extend it to take both audit events and IDS alerts as inputs (instead of only alerts).

To leverage the available knowledge of current SOC analysts, we chose an approach that is similar to what is actually done for alert fusion: first slicing the time into small windows, and then using logic in the form of rules to separate events describing different actions.

Group by time proximity While the asynchronous nature of modern computing may render the strict ordering of the events via their time-stamps highly improbable, the various sensors deployed inside the monitored system are likely

[4] Timestamp excepted

to react to the same action within a short period of time (at least if the whole system is synchronized using the same time server, or if the log shipping system is in charge of setting the time-stamps). Therefore, the first step we perform is to regroup events appearing inside the same time window. The size of the window is configurable as, depending on the tools available, the time between the first event and the last event matching the same action can vary from milliseconds to a few minutes (e.g., for NIDS that trigger alerts at the end of the connections). Each slice of time can be handled completely in parallel.

Correlation of meta-events Linking security events based on causal dependencies can ease the investigation process [29]. Indeed, each action inside an IS, such as attacks, is observed from various sensors. Each one records different pieces of information about the root action. We hence aim to uncover the root action through event correlation (cf. Section 2.2) of given groups of events. This correlation is based on two assumptions: **(1)** Regardless of the data source, two events coming from the same origin (e.g., same process, same host, same user, etc.) and within a short period of time are more likely to share a common cause. Indeed two correlated events can be either causally linked, caused by the same cause or a coincidence. The probability of a coincidence is lower when analyzing finer grained events (e.g., endpoint monitoring, syscall auditing, etc.). **(2)** Some event attributes are common to multiple data sources [13, 21], which enables us to regroup events from heterogeneous sensors. For instance, an endpoint auditing mechanism can log sockets opened by processes which can be linked with NIDS logs using the source IP address, and other endpoint logs using the process identifier. For some tools, some attributes are specifically designed to link events together (e.g., the zeek network probe [20] events have attributes that are meant to link each analyzers results to the original network flow).

Meta-events with attributes indicating the same origin are therefore pieced together. The list of such attributes is called a correlation rule. This regrouping operation is designed to be both associative and commutative (i.e., the processing order do not impact the result) and can therefore be considered as a reduction operator, which can be handled in parallel.

4 Anomaly score computation using dynamic auto-encoders

Grouping together events in sets of meta-events offer a better understanding of observed phenomenon. We go further by prioritizing sets of meta-events with neural network-based anomaly detection system by computing an anomaly score for each set and the meta-events that compose it. Fig. 1 provides a functional overview of the model, which is described in the following section.

4.1 Basics on neural networks and auto-encoders

A neural network is a composition of functions with trainable parameters. These functions are called the neurons of the network. A neural network is trained to

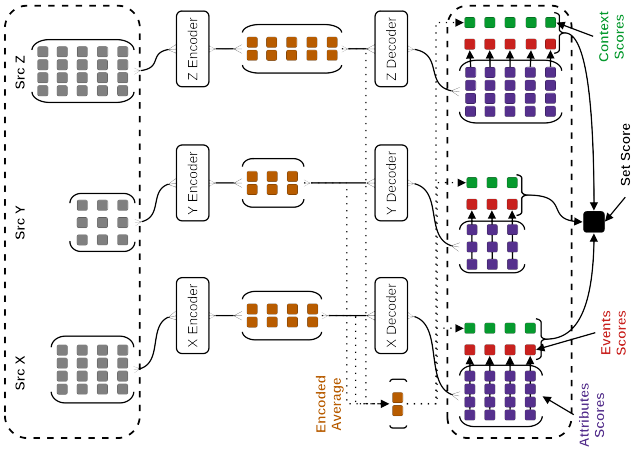

Fig. 1: Overview of the anomaly scoring for a set of meta-events

minimize an objective function, which often contains at least a measure of the divergence between the expected output and the one effectively predicted by the network. To solve this minimisation problem, the most used algorithms derives from the gradient back-propagation method [23], which requires the gradient of the objective function for each parameter of the network. Therefore, in this paper, we generalize the neural network definition to a composition of differentiable functions, where parameters are trained using any variant of the gradient back-propagation method.

Historically, neural networks have been used to map vectors from high-dimensional space (e.g., the pixels of an image, complex set of extracted features, etc.) into more easily comprehensible values (e.g., a class, a small vector, etc.). With the rise of deep learning, researchers moved from using only fully connected layers of formal neurons to incorporating layers specifically adapted to the structure of the data provided as input to the model (e.g., 2-D convolution for images, recurrent networks for text, etc.). For tasks that require the network to output in a high dimensional space (e.g., image or text generation), the structure of the network is often organised in two parts, an encoder which maps inputs into a latent representation, and a decoder which takes the latent representation as input and outputs value into the expected vector space. The auto-encoder is a special Encoder-Decoder model for which the output space is the same as the input space. Such a structure can be adapted to detect anomalies in security events [5].

4.2 Dynamic structure with coherent encoded representation

For each data source, an encoder and a decoder is initialized at the beginning of the training phase. One of the limitations highlighted in [5] was that the compressed representation of the events was not comparable from one source to another. Besides, each source of events needs its own model, which can be difficult

to manage once the number of sources grows too high. We propose to alleviate these limitations using a dynamic network structure combined with a penalty added to the objective function. It encourages the encoder to encode meta-events appearing frequently in the same sets into close vectors in the encoded space.

For each set of meta-events, we select the encoder-decoder pairs corresponding to each data source in the set and build an auto-encoder out of them. In addition to the reconstruction error (i.e., the difference between the original input and the output of the network), we compute the distance between the encoded representation of each meta-event of the set and the average value of these encoded representations for the whole set. We derive the **context score** from this distance, and minimizing it is added as an objective for the encoder, which encourages it to compute an encoded representation that is close for meta-events that frequently happen in the same context (e.g., an HTTP request is often accompanied by a DNS query).

4.3 Computing anomaly score

Attribute score The reproduction error of the auto-encoder (difference between the input and the output) will be higher for anomalous data. To guide the investigation of anomalies, it is interesting to determine which attributes are likely to be the cause of the anomaly. The problem is that the reproduction error from one attribute to another cannot be trivially compared. In [5] a Gaussian Mixture Model was used as an approximation of the distribution of the loss. The cumulative distribution function (CDF) of the Gaussian with the highest mean value was then used to compute an anomaly score between 0 and 1 for every attribute. However, in cases where the variance of the loss is close to 0, this method provides unsatisfactory results (i.e., high score for all events or low score even for anomalies). Besides, the GMM parameters cannot be approximated accurately within the network (i.e., using gradient descent), which forces a calibration phase after training the auto-encoder, and to maintain an additional model. In our case, we propose to use a logistic distribution instead of the Gaussian mixture, and train its parameters directly within the network (Eq. 2). In our case, we want to have a score that is close to 0 for most of the normal events, while still being close to 1 when the attribute is anomalous . One possibility would be to set μ (the point at which the logistic CDF is 0.5) to be the highest value on the (supposedly normal) data. To do so, we constraint the parameters of the logistic distribution (Eq. 4).

Meta-event score The score of a meta-event is the weighted mean of the score of all its attributes. The weight w_i of each attribute x_i is computed by a small neural network that takes as input the average value of the encoded representation for a set of meta-event. Minimizing the standard deviation of the weight vector is added as an objective to discourage the weighting neural network to amplify or attenuate excessively some attributes.

Set score The score of a set is the weighted mean of the scores of all the meta-events that constitutes it. The weight q_i for a meta-event is $q_i = \frac{s_i}{\sum_j s_j}$ with s_i the score of the i^{th} meta-event in the set.

$$f_{\lambda,\mu}(x) = \frac{1}{1 + e^{-\lambda(x-\mu)}} \quad (2)$$

$$E_\mu(X) = \alpha * g\left(\mu - \overline{X}\right) + g\left(\overline{X} - \mu\right) \quad (3)$$

$$g(x) = \begin{cases} x & \text{if } x \geq 0 \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

$$E(X) = E_\mu(X) + \overline{f_{\lambda,\mu}(X)} + g(-\lambda) \quad (4)$$

Fig. 2: (2) Cumulative distribution function for a logistic distribution. λ controls the steepness of the curve and μ the symmetry point $f(\mu) = 0.5$. (3) Constraint added to the μ parameter of the attribute scoring function. The lower α is, the closer to the maximal input value μ will be. $\overline{X}$ is the average value of the reproduction error for a batch of data. (4) Constraint for the whole scoring function. The last term is needed to avoid λ being negative (which would cause the score to be higher for normal values than anomalous ones)

4.4 Handling concept drift

Normal behavior of an IT system is bound to evolve with time, as new behaviours appear and old one cease to manifest. Indeed, new users are added, old ones are removed, software are updated, etc. In data science, this phenomenon is called concept drift and is handled by updating the model. In the context of security monitoring this phenomenon can manifest in different ways. First, correlations between variables evolves (e.g., a known user started using a known command) and/or the correlation rules used for regrouping events into sets of meta-events (data sources do not change, only the composition of the sets) need to be changed. In this case, only the model weights needs to be updated, and as neural networks learn incrementally by nature, it is only a matter of performing a few training steps on data containing the new behaviors. The second manifestation of concept drift is a modification of the input space which requires a modification of the pre-processing functions parameters (e.g., a new user, a new software deployed, etc.). Finally, in some cases the data sources schema must be updated (e.g., new type of sensor deployed, modification of the analyzed attributes, etc.). By combining the neural networks capacity to learn incrementally and the design in functional blocks (for each attribute of each datasource), we can handle concept drift without requiring a costly complete retraining of the model. Kirkpatrick et al.[12] proposed the Elastic Weight Consolidation (EWC) algorithm in order to attenuate catastrophic forgetting of neural networks, i.e., the network completely forgets older normal behaviors too quickly. EWC consists in adding penalties and constraints on the network's weight during training to avoid modifying weights that are essential to solve the previously learned tasks. Authors have also proposed the use of small sample of previous data either as a small knowledge base constituted during training and used during inference (episodic memory) [3, 25], or simply by selecting a sample of past experiences when training on new data (experience replay) [22]. We chose this second approach as it is simpler (computationally speaking) than episodic memory, and it also applicable to the adjusting of the transformation function parameters (EWC is only useful for network weights).

5 Approach Assessment

5.1 The evaluation dataset

Assessing the performance of an anomaly-based security log analysis method requires the collection of enough logs to model the normal behaviour of the experimental monitored system. For confidentiality reasons, it is not possible to use production data, so the data needs to be collected from a lab environment and user interaction with this environment must be simulated. As accounting for every possible cases of a real IT system is impossible, this simulation is bound to be biased. However, considering that each user action is done by human operators as it would be done on a production environment, the collected logs will be similar to the ones collected on a production system.

In order to increase signification of collected events (e.g., admin tasks should not be performed from user workstations), the monitored system's architecture is intended to be reasonably secure (well separated infrastructure, administrator and user zones, controlled outgoing and incoming traffic, updated software and antivirus, etc.). Six simulated users perform various office tasks (documents writing, mail, web browsing, etc.) across Linux and Windows workstations while an administrator maintains the infrastructure of the system and sometimes connects to the users endpoints to install software or update configurations.

The first two days of the dataset serves as a baseline of "normality" (i.e., without any attack traces) that is required for anomaly detection system training. The presence of attack traces inside the training data would prevent detection of said attack, but attacks employing different techniques would still be detected. The first attack scenario (day 3) emulates an attacker with custom tooling (to avoid detection by antivirus), but is noisy (multiple IDS alerts, a few antivirus alerts, more trial and errors, etc.). The day after (day 4), a new software is deployed on a Centos machine. The second scenario (day 5) reemploys the same tooling as the first scenario, but is more subtle (generates less IDS alerts, no antivirus alerts, actions are more precise, etc.). During the attack, the administrator diagnoses a problem on windows machines (and therefore generate unusual activity). After this (day 6), a new sensor (Sysmon) is enabled on the user endpoints, and is expected to generate a large amount of false positives. The last scenario (day 7) is much more discreet than the previous ones and relies on new tools. The attacker limits its activity to a minimum to avoid generating too much events. This scenario aims at emulating a more advanced threat actor.

The dataset contains 4.2 million events, over a period of 7 working day. These events come from typical monitoring tools commonly found inside IT systems (i.e., Sysmon, Auditd, Windows Audit Logs, Windows Defender, Zeek IDS, Suricata IDS and Squid HTTP Proxy). We configure these tools to record a higher volume of information than what is usually deployed on IT systems (e.g., all available IDS rules, process monitoring with windows audit, socket monitoring with Sysmon and auditd, etc.). This enables us to test various levels of verbosity for assessing our method without regenerating a datasets.

5.2 Defining the assessment strategy

While commonly used quantitative metrics (e.g., False/True Positives Rates, F1 score, etc.) can provide valuable information regarding the performance of a detection method, they only give a partial view of the usefulness of the detection system. For instance, missing 50 out of 100 failed connection attempt to a service in a brute force attack would not prevent analysts from finding out the source and the target of the attack, while increasing the False Negative Rate (FNR). On the other hand, missing a single event that characterizes an attack would not have a major impact on the FNR, while greatly reducing the quality of the detection.

We alleviate this limitation with an alternative definition of a False Negative that is more inline with analysts practices. In fact, we know every actions that are performed by the attacker, but annotating individually the thousands of events that can be attributed to these actions would be time consuming and error prone. Instead, we start from a high threshold above which we consider a set of meta-events as an anomaly, and we lower it until we can find anomalies characterising every step of the attack (i.e., what is the machine impacted, the technique employed, etc.). Any set below this threshold that is a consequence of malicious activity will not be considered as a False Negative, as it will not add useful information to characterise the attack. Empirically, we find that a threshold of 0.4 is a good choice for the three scenario we have. However, this threshold is not optimal as multiple legitimate actions can have a score higher than it. Therefore, we gradually increase this threshold and we compute the precision (proportion of True Positive in all the anomalies), recall (proportion of the attack accurately detected) and F1 score (harmonic mean of precision and recall) to quantify the performance. For all these metrics, a value close to 0 implies a useless anomaly detection while a score of 1 is synonym of a good one. We consider the optimal threshold to be the one that maximizes the F1 score.

We mitigate the impact of a lucky (or unlucky) initialisation of the model training 10 models with different initialisation of the parameters on the first two days of normal data. This data is randomly splitted for each model, with 90% used for training the model and 10% to control that the model is not over-fitting the training data (often called validation set). We test the performance of each of these models on the first scenario. After that, we keep only one model and use it as the base for incremental learning and evaluating the performance on the other two scenarios.

5.3 Results

For each of the 10 randomly initialized models, we record the best F1 score on the first attack scenario. The average value of this score is 0.88 with the lowest value at 0.85 and highest at 0.91. The figure 3 has been generated using the best model (F1: 0.91, recall: 0.93 and precision: 0.88). The dashed black line shows the number of anomalies per hour detected during a single day without attack. On the opposite, red line shows number of anomalies per hour produced when an

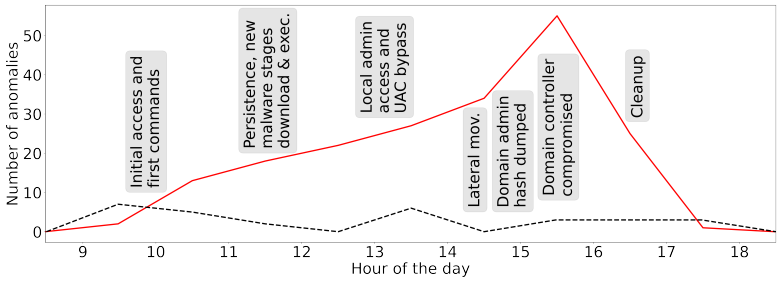

Fig. 3: Aggregated number of meta-event sets with a score above 0.5 per hour for scenario 1.

attack occurred. As expected, we can see that the number of anomalies increases as the attack progresses. The maximum number of anomalies is reached when most of the machines are compromised, and especially the domain controller. The cleanup phase is accompanied by a decreasing number of anomalies as machines progressively stop exhibiting attack behaviours. This scenario shows that our system can be used as intrusion detection system as it detects attack-related events and because it does not report more than fifty anomalies per hour from an initial number of events of approximately 20000 per hour.

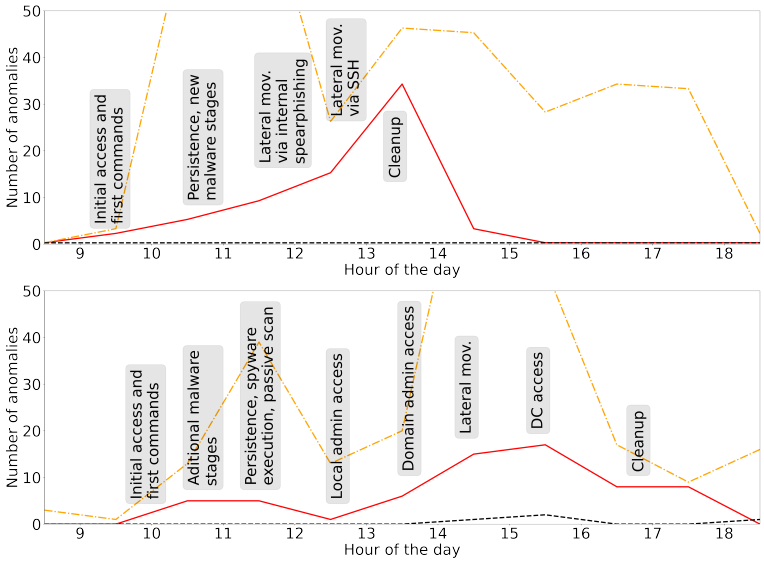

Fig. 4: Aggregated number of meta-event sets with a score above 0.5 per hour, for scenario 2 (top) and 3 (bottom)

For the second scenario, we use the false positives from the first scenario and the fourth day of the collected dataset as the input to incrementally update the

anomaly detection model. We reach a F1 score of 0.97 with a recall of 0.99 and a precision of 0.95. Similarly, for the third scenario, we reach an F1 score of 0.92 with a precision of 0.86 and a recall of 0.98. Without retraining (i.e., using the same model used for scenario 1), the best F1 score drops below 0.3 for both scenarios. To highlight the effectiveness of the retraining, in Figure 4, we add the plot (orange dot-dashed line) of the same normal day than the black dashed plot, but this time analysed by the outdated model.

Similarly to the first scenario, there is an increasing number of anomalies reported during the attack process of the scenario 2. We see lower numbers of anomalies, which is expected considering the attack a bit less noisy. As the third scenario is designed to be as discreet as possible, we see far less anomalies. Nevertheless, meta-events raised by our system are sufficient for an analyst in order to recognize the attack pattern. The orange line shows the necessity of the incremental learning process for our anomaly based detection system.

The latent representation provided by the model enables the use of standard clustering algorithms to cluster meta-event groups. Most of the time, new behaviours reappear regularly. Thus, the manifestation of these new behaviours in the log should form clusters of similar activity. In our case, we use the DBSCAN algorithm [9] to perform this clustering and we find that the discovered cluster permits to reduce the annotation time. Fig. 5 shows two anomalous meta-event groups that are found to be similar. The first one corresponds to the events generated when the attacker enumerated local users and groups and the second to the enumeration of domain users and groups. Preliminary results are encouraging, but a more in depth evaluation of the clustering performance is still required.

win_process_audit | win_process_sysmon

timestamp	score	agent.name	process.parent.command_line	process.parent.executable	process.executable	process.command_line
2021-06-23 08:22:00.956000000	0.505922	WIN-2	cmd.exe	C:\WINDOWS\SYSWOW64\CMD.EXE	C:\WINDOWS\SYSWOW64\WHOAMI.EXE	whoami /all
2021-06-23 08:22:09.728999936	0.506463	WIN-2	cmd.exe	C:\WINDOWS\SYSWOW64\CMD.EXE	C:\WINDOWS\SYSWOW64\NET.EXE	net localgroup
2021-06-23 08:22:09.763000064	0.520170	WIN-2	net localgroup	C:\WINDOWS\SYSWOW64\NET.EXE	C:\WINDOWS\SYSWOW64\NET1.EXE	C:\Windows\system32\net1 localgroup

win_process_audit | win_process_sysmon

timestamp	score	agent.name	process.parent.command_line	process.parent.executable	process.executable	process.command_line
2021-06-23 11:39:09.503000064	0.506828	WIN-2	cmd.exe	C:\WINDOWS\SYSWOW64\CMD.EXE	C:\WINDOWS\SYSWOW64\NET.EXE	net user /domain
2021-06-23 11:39:09.548999936	0.520004	WIN-2	net user /domain	C:\WINDOWS\SYSWOW64\NET.EXE	C:\WINDOWS\SYSWOW64\NET1.EXE	C:\Windows\system32\net1 user /domain
2021-06-23 11:39:16.097999872	0.507229	WIN-2	cmd.exe	C:\WINDOWS\SYSWOW64\CMD.EXE	C:\WINDOWS\SYSWOW64\NET.EXE	net group /domain
2021-06-23 11:39:16.116000000	0.524452	WIN-2	net group /domain	C:\WINDOWS\SYSWOW64\NET.EXE	C:\WINDOWS\SYSWOW64\NET1.EXE	C:\Windows\system32\net1 group /domain

Fig. 5: These two anomalous meta-event groups belong to the same cluster in the latent space.

Even though these results are promising, some limitations can be highlighted. In fact, it still relies on parameters that needs to be defined by an expert. More

precisely, an inadequate choice of attributes for the datasources, i.e., missing important attributes or adding useless ones, can lead to unsatisfactory results. Although the redundancy of the datasources limits the effect of missing features, and the weighting system of the anomaly scorer lowers the impact of useless ones, we think that the scoring system still requires some adjustments to improve the tolerance to human error. For instance, with the current scoring system, a datasource with more than 10 attributes will rarely cause alerts because most of the attributes would seem normal and dilute the score of anomalous attributes. In addition to that, the chosen time-window can have an impact on the quality of the fusion step. Indeed, during testing, we realised that a time-window below 120 seconds lead to a significant amount of missed fusion opportunity between datasources that have different ways of setting the timestamp (e.g., a network probe often emits an event at the end of a connection, while an endpoint monitoring would do so at its beginning). Symmetrically, we found that a time-window above 300 seconds would lead spurious correlations between meta-events.

6 Conclusion

Modern SOCs handle alerts based on attack patterns or signatures in order to perform intrusion detection. However, to detect unknown threats, gaining a deeper visibility of the system is required. This is done by analysing both events and alerts (and not only alerts), in search for anomalies. Consequently, a significant growth in the volume of information that needs to be analysed and the number of false positives occurs. While machine learning based approaches can help analysts detect novel attacks, their adoptions into SOC have been slow. Part of the explanation for this is the lack of explainability and the need for data science expertise (that is scarce in SOC teams).

We therefore proposed a new anomaly detection approach that takes advantage from available security analysts expertise to highlight and contextualise anomalies in security logs. This approach is based on the fusion of fine-grained events into sets of meta-events, and on an autoencoder neural network with dynamic structure to compute an anomaly score for each of these sets. Our system is designed to adapt to new behaviours (concept drift) by learning incrementally while not forgetting old behaviours too quickly. For validation purpose, we performed several scenario-based evaluations on a lab environment monitored using tools often deployed on production systems. The results show that our system reports a low number of anomalies per hour and that this number is correlated with the progress of the attack scenarios. For all of our evaluation scenario, the system reports enough information to completely reconstruct the attack.

Future work includes publishing the dataset used for experimentation alongside additional results, as well as correcting some identified limitations of the method. Specifically, while the involvement of security experts in the configuration of the system allows for results that are more understandable to them, it also exposes the system to human error which could degrade its performance.

Our focus will be on diminishing the impact of these errors and reinforcing the transparency of the system for human operators.

References

1. Bertero, C., Roy, M., Sauvanaud, C., and Trédan, G. Experience report: Log mining using natural language processing and application to anomaly detection. In *2017 IEEE 28th International Symposium on Software Reliability Engineering (ISSRE)* (2017), IEEE, pp. 351–360.
2. Brogi, G., and Tong, V. V. T. Terminaptor: Highlighting advanced persistent threats through information flow tracking. In *2016 8th IFIP International Conference on New Technologies, Mobility and Security (NTMS)* (2016), IEEE, pp. 1–5.
3. de Masson d'Autume, C., Ruder, S., Kong, L., and Yogatama, D. Episodic memory in lifelong language learning. *CoRR abs/1906.01076* (2019).
4. Debnath, B., Solaimani, M., Gulzar, M. A. G., Arora, N., Lumezanu, C., Xu, J., Zong, B., Zhang, H., Jiang, G., and Khan, L. Loglens: A real-time log analysis system. In *2018 IEEE 38th International Conference on Distributed Computing Systems (ICDCS)* (2018), IEEE, pp. 1052–1062.
5. Dey, A., Totel, E., and Sylvain, N. Heterogeneous security events prioritization using auto-encoders. In *Risks and Security of Internet and Systems: 15th International Conference, CRiSIS 2020, Paris, France, November 4–6, 2020, Revised Selected Papers* (2020), Springer Nature, p. 164.
6. Ding, Z., and Fei, M. An anomaly detection approach based on isolation forest algorithm for streaming data using sliding window. *IFAC Proceedings Volumes 46* (2013), 12–17.
7. Du, M., Li, F., Zheng, G., and Srikumar, V. Deeplog: Anomaly detection and diagnosis from system logs through deep learning. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security* (2017), ACM, pp. 1285–1298.
8. Elastic. Elastic common schema. https://github.com/elastic/ecs. 25-03-2021.
9. Ester, M., Kriegel, H.-P., Sander, J., Xu, X., et al. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Kdd* (1996), vol. 96, pp. 226–231.
10. Hassan, W. U., Bates, A., and Marino, D. Tactical provenance analysis for endpoint detection and response systems. In *2020 IEEE Symposium on Security and Privacy (SP)* (2020), IEEE, pp. 1172–1189.
11. He, S., Zhu, J., He, P., and Lyu, M. R. Experience report: System log analysis for anomaly detection. In *2016 IEEE 27th International Symposium on Software Reliability Engineering (ISSRE)* (2016), IEEE, pp. 207–218.
12. Kirkpatrick, J., Pascanu, R., Rabinowitz, N., Veness, J., Desjardins, G., Rusu, A. A., Milan, K., Quan, J., Ramalho, T., Grabska-Barwinska, A., et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences 114* (2017), 3521–3526.
13. Leichtnam, L., Totel, E., Prigent, N., and Mé, L. Forensic analysis of network attacks: Restructuring security events as graphs and identifying strongly connected sub-graphs. In *2020 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)* (2020), IEEE, pp. 565–573.

14. Liu, F., Wen, Y., Zhang, D., Jiang, X., Xing, X., and Meng, D. Log2vec: A heterogeneous graph embedding based approach for detecting cyber threats within enterprise. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security* (2019), pp. 1777–1794.
15. Liu, F. T., Ting, K. M., and Zhou, Z.-H. Isolation forest. In *2008 Eighth IEEE International Conference on Data Mining* (2008), IEEE, pp. 413–422.
16. Milajerdi, S. M., Gjomemo, R., Eshete, B., Sekar, R., and Venkatakrishnan, V. Holmes: real-time apt detection through correlation of suspicious information flows. In *2019 IEEE Symposium on Security and Privacy (SP)* (2019), IEEE, pp. 1137–1152.
17. Mirsky, Y., Doitshman, T., Elovici, Y., and Shabtai, A. Kitsune: an ensemble of autoencoders for online network intrusion detection. *arXiv preprint arXiv:1802.09089* (2018).
18. MITRE. Att&ck data sources. https://github.com/mitre-attack/attack-datasources. Accessed: 16-03-2021.
19. Pascoal, C., De Oliveira, M. R., Valadas, R., Filzmoser, P., Salvador, P., and Pacheco, A. Robust feature selection and robust pca for internet traffic anomaly detection. In *2012 Proceedings Ieee Infocom* (2012), IEEE, pp. 1755–1763.
20. Paxson, V. Bro: a system for detecting network intruders in real-time. *Computer networks 31* (1999), 2435–2463.
21. Pei, K., Gu, Z., Saltaformaggio, B., Ma, S., Wang, F., Zhang, Z., Si, L., Zhang, X., and Xu, D. Hercule: Attack story reconstruction via community discovery on correlated log graph. In *Proceedings of the Annual Computer Security Applications Conference (ACSAC)* (2016), ACM.
22. Rolnick, D., Ahuja, A., Schwarz, J., Lillicrap, T. P., and Wayne, G. Experience replay for continual learning. *CoRR abs/1811.11682* (2018).
23. Rumelhart, D. E., Hinton, G. E., and Williams, R. J. Learning internal representations by error propagation. Tech. rep., California Univ San Diego La Jolla Inst for Cognitive Science, 1985.
24. Sadoddin, R., and Ghorbani, A. A. An incremental frequent structure mining framework for real-time alert correlation. *Computers & Security 28* (2009), 153–173.
25. Sprechmann, P., Jayakumar, S. M., Rae, J. W., Pritzel, A., Badia, A. P., Uria, B., Vinyals, O., Hassabis, D., Pascanu, R., and Blundell, C. Memory-based parameter adaptation, 2018.
26. Valeur, F. *Real-time intrusion detection alert correlation.* Citeseer, 2006.
27. Veeramachaneni, K., Arnaldo, I., Korrapati, V., Bassias, C., and Li, K. Ai2: training a big data machine to defend. In *2016 IEEE 2nd International Conference on Big Data Security on Cloud (BigDataSecurity), IEEE International Conference on High Performance and Smart Computing (HPSC), and IEEE International Conference on Intelligent Data and Security (IDS)* (2016), IEEE, pp. 49–54.
28. Viinikka, J., Debar, H., Mé, L., Lehikoinen, A., and Tarvainen, M. Processing intrusion detection alert aggregates with time series modeling. *Information Fusion 10* (2009), 312–324.
29. Xosanavongsa, C., Totel, E., and Bettan, O. Discovering correlations: A formal definition of causal dependency among heterogeneous events. In *2019 IEEE European Symposium on Security and Privacy (EuroS P)* (2019), pp. 340–355.
30. Zhou, C., and Paffenroth, R. C. Anomaly detection with robust deep autoencoders. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (2017), ACM, pp. 665–674.