Exchanging Security Events: Which And How Many Alerts Can We Aggregate?

Martin Husák*†, Milan Čermák*†, Martin Laštovička*†, and Jan Vykopal*
*Institute of Computer Science, †Faculty of Informatics
Masaryk University, Brno, Czech Republic
{husakm,cermak,lastovicka,vykopal}@ics.muni.cz

***Abstract*—The exchange of security alerts is a current trend in network security and incident response. Alerts from network intrusion detection systems are shared among organizations so that it is possible to see the "big picture" of current security situation. However, the quality and redundancy of the input data seem to be underrated. We present four use cases of aggregation of the alerts from network intrusion detection systems. Alerts from a sharing platform deployed in the Czech national research and education network were examined in a case study. Volumes of raw and aggregated data are presented and a rule of thumb is proposed: up to 85 % of alerts can be aggregated. Finally, we discuss the practical implications of alert aggregation for the network intrusion detection system, such as (in)completeness of the alerts and optimal time windows for aggregation.**

I. Introduction

Collaboration and data exchange is an emerging trend in network security and security incident response [1]. Network monitoring and intrusion detection systems more and more include an alert sharing component. Sharing security alerts among organizations allows them to see the "big picture" of the current situation on the global network and changes the approach taken to network security. Large-scale security incidents can be detected more easily and effectively mitigated. Also the analysis of alerts from distributed sensors provides valuable results. However, there are certain issues specific for this area that need to be addressed.

The first task of processing the shared security alerts is to cleanse and aggregate the data. Not only we have to eliminate alerts that are incorrectly formatted or that are missing important pieces of information. The alerts may be duplicated, reported repeatedly over time, and the same event may be reported by multiple sensors. The sensors, e.g., network intrusion detection systems, are typically not aware of alerts from other sensors, so it is up to the alert sharing platform to aggregate the data. Redundancy in the input data increase the cost of their analysis and may also influence its results, so the aggregation has to be done before any advanced analysis. Requirements on computing power and data storage should also be taken into consideration as the security alerts are considered to be "big data" at the time. Although many alert aggregation and correlation methods have been proposed, there are not enough case studies on the volumes of raw and aggregated data and reduction rates of individual alert aggregation methods.

To formalize the scope of our work, we pose three research questions which we shall answer:

(i) Which security alerts can be aggregated without the loss of information?
(ii) What is the volume of raw and aggregated data?
(iii) Is there an optimal time window for alert aggregation?

First, we have to understand why duplicates appear among shared security alerts. Second, we are interested in the volume of the data and its decrease after the aggregation. This problem has practical implications on the design of tools for alerts sharing and analysis. We are going to aggregate alerts in a database snapshot from the SABU platform[1]. The ratio of aggregable alerts in the snapshot will be the prime result of this case study. Finally, time plays a significant role in the aggregation process. We have to find which time window is optimal for the aggregation, so we can avoid aggregating unrelated alerts or leaving duplicates. The optimal time window should be set according to typical differences in the timestamps from duplicate security alerts.

II. Related Work

The alerts used in this work were mostly generated by network intrusion detection systems based on network flow monitoring, which is suitable for large-scale and high-speed networks. An overview of flow-based intrusion detection methods was presented by Sperotto et al. [2].

Chuvakin et al. [3] discussed which pieces of information can help with aggregating log data for the purpose of security analysis. The source of the alert (IDS, firewall) and destination IP and port are discussed as suitable, while one has to be cautious with aggregation by the source IP and port due to possible spoofing. The same suggestions are applicable for network data.

Alert aggregation and correlation belongs to the field of collaborative intrusion detection, which has recently been surveyed by Vasilomanolakis et al. [1]. Alert correlation in particular was surveyed by Elshoush and Osman [4].

Valdes and Skinner [5] introduced three levels of sensor correlation, out of which two corresponded to alert aggregation. The *synthetic attack threads level* is used for clustering alerts of the same attack within a single sensor. The *security incidents level* clusters the same attacks reported by different

[1] https://sabu.cesnet.cz/

sensors. Similarity functions are used for clustering in both cases. The reduction of alert volume in a live environment was estimated to be from one half to two thirds.

Cuppens and Miege [6] proposed alert correlation in a cooperative intrusion detection framework. In their work, alerts from different intrusion detection systems are stored in a relational database. Then, a set of expert rules is used to group the alerts into clusters of the same occurrences of an attack.

Debar and Wespi [7] focused on the aggregation and correlation of intrusion detection alerts. Raw alerts are first preprocessed into a unified model with three attributes - problem, source, and target. Alerts are clustered based on these three attributes. Then, a relationship correlation takes place in which duplicates and consequent alerts are identified. Duplicate alerts, in this case, are alerts from different sensors. Consequent alerts are alerts linked in a specific order that occur within a given time interval.

Valeur et al. [8] present alert fusion as a part of their comprehensive approach to correlating intrusion detection alerts. The goal of alert fusion is to combine alerts which represent the detection of the same event by different intrusion detection systems. They show a reduction rate of up to 28 %. However, they have only one specific use case and use only 2 seconds as the time window for alert fusion.

Lin et al. [9] used adaptive learning to reduce the number of false positives and duplicates. Their method reduced the number of alerts in two datasets to 25 % and 8 %, mostly due to temporal correlation or association with a single attack. However, the datasets from 1999 and 2007 do not reflect the current situation and trends in cybersecurity.

III. Use Cases of Alert Aggregation

In this section, we describe the use cases for the aggregation of security alerts and the circumstances under which an event is reported multiple times. The aggregation of security alerts is not only the deduplication of records in the database. It requires an understanding of the security domain that extends the traditional data deduplication tasks.

A. Duplicates

Deleting obvious duplicates is the easiest to understand and their aggregation is easier to implement. However, the duplicates may not often be seen as their appearance means either an error or misconfiguration in the system. In the context of this paper, we do not assume system errors and focus on misconfiguration issues.

An example of alert duplication is a scenario in which the same alert is reported repeatedly. For example, we have a network with an intrusion detection system and reporting system. The reporting system receives alerts from the IDS and shares it via a sharing platform. However, the IDS also has a sharing capability and shares the alerts as well. Although none of the systems did anything wrong, the alert is duplicated. The reason behind the duplication in this example is a misconfiguration of the system. The network administrator should be aware that both systems share the data and allow only one of them to do it. An alternative example includes an IDS that reports alerts to two different alert sharing platforms. The two platforms then exchange alerts and thus, the duplicates appear.

If a duplicate appears in the system, it should not be exactly the same as the original alert. Following the example, the reporting system may add new items to the alert, such as timestamps from receiving the alert in the reporting system or an identifier of the reporting system as a mark that the system processed the alert. However, the key alert items, i.e., source, target, timestamps, and event type, are the same.

B. Consequent Alerts

A consequent alert is an alert of an event that was already reported, but the action, which caused the incident, still continues and the alert is reported repeatedly. Consequent alerts are very interesting from the security perspective as they provide information that the event reported in a previous alert is still present.

For example, an intrusion detection system detects network scanning in a 5-minute time window. An event is detected and an alert is raised and shared within a sharing platform immediately. However, network scanning may last much longer. The scanning is not mitigated and continues. The event is detected again in the consecutive time window and a new alert is raised and shared.

Apart from previous case, the two alerts look the same, but there is a significant difference in timestamps. Thus, from the security perspective, there is additional information about the timing of the event in the aggregable alerts. The expected method of aggregation is not just dismissing the later alerts, but also providing the information about the continuation of the event. Consider a sensor that raises alerts of an event detected in a time window. If two alerts list consecutive time windows, then the second alert could be dismissed and the time window in the first alert should be updated with a new value that spans the time windows in both alerts.

C. Overlapping Sensors

Another use case is the aggregation of alerts from overlapping sensors. The sensors in this case have to overlap in their detection scope, e.g., monitored network segment.

An example for this case is network scanning that is detected by two network probes. The first probe monitors a backbone network and the second probe monitors a campus network that is connected to the backbone network. The alert types and addresses of the scanning host are the same, but there are two main differences. First, the identifiers of the sensors are different. Second, the timestamps are different as the network traffic flows from one sensor to another, although this difference might be in an order of milliseconds.

Alerts from overlapping sensors should be treated with caution. The alerts can be aggregated without loss the of information, but it is not clear which data entries to dismiss and which to keep. Naturally, the first reported alert can be kept and the others can be dismissed. Following this example, a problem arises when the two probes have different

monitoring scopes and the target is not specified. Then, the location of the monitoring probe indicates a possible target. Thus, the alert generated by the probe in a campus network indicates more information than alert generated by the probe in the backbone network. Similarly, the differences in thresholds used by detection methods may influence the alerts, although this actually has more influence if an alert is raised by a sensor with higher threshold.

D. Non-overlapping Sensors

An even more complex use case of security alert aggregation is the aggregation of alerts from non-overlapping sensors. In this case, the sensors do not overlap in the scope of their detection, e.g., each of them is located in a different network.

Consider an attacker who is scanning a large network. Network sensors in subnetworks detect network scanning from the same IP address at approximately the same time and share the information. Thus, the sharing platform receives multiple reports of the same event, but each report declares a different target due to the limited monitoring scope of the sensors.

This use case is the most complex one due to its high context sensitivity and potential loss of information. If two alerts are found to be aggregable, then it is important to keep the information from both. Following the scanning example, the targets are typically scanned IP ranges, which should be merged during the aggregation. The same applies for timestamps and time windows contained in the alerts. Similarly to the alert from overlapping sensors, additional information about the (non-)overlapping of sensors is required. Finally, the type of security event also plays a role. The scanning example is straightforward, the attacker scanned a larger network segment than the monitoring scope of the individual probes. However, if an attacker at approximately the same time exploits two honeypots, each in a different network, then the two alerts are definitely worth correlating, but it is not a case for aggregation.

IV. Results and Discussion

In this section, we present and discuss the results of the SABU alert database analysis. First, we describe the dataset used for the analysis. Then, the results of the four aggregation methods are presented and followed by the overall results of all the methods combined.

A. Dataset

The dataset was obtained from the SABU alert sharing platform in June 2016. It contains 8,043,378 real-life alerts from one week in IDEA[2] format. The number of alerts per day ranged between 1,029,193 and 1,327,705. In total, 25 sensors from 7 organizations and 1 third-party alert sharing system reported alerts of 17 event types. The sensors, mostly flow-based intrusion detection systems and honeypots, were deployed in the backbone network and campus networks. A third-party sharing system contributed with 126,510 alerts.

We selected 7-tuple of data elements for the aggregation – source and destination IPs and ports, event type, sensor ID, and timestamp. Only 136 alerts did not contain an IP address in the source or target nodes and only 167 alerts had IPv6 address in the source or target nodes. Thus, we decided to omit alerts without IPv4 address due to their insignificant numbers. The mandatory timestamp in IDEA contains a time of detection. Only 412,029 alerts added the real starting time of an event. Thus, only the mandatory timestamps were used.

[2]https://idea.cesnet.cz/

B. Duplicates

Finding duplicates was the first and easiest task of alert aggregation. We aggregated the alerts in the dataset according to its source, destination, event type, sensor ID, and timestamp. All of these five traits had to be equal for at least two alerts to be considered duplicates.

Simple aggregation shows that 108,298 alerts had to be aggregated, out of which 8,861 alerts turned out to be unique and 99,437 alerts were duplicates. Thus, 1.24 % of alerts in the dataset could be aggregated using this method.

C. Consequent Alerts

The next step in aggregating alerts in the dataset was the aggregation of consequent alerts. We aggregated the alerts according to their source IP, destination IP, destination port, event type, and sensor ID. Source port was not included in the list due to its volatility in time. A list of timestamps was assigned to each unique tuple. If there were more than one timestamp assigned to a unique tuple, the alerts could be aggregated. The earliest timestamp is kept, while the duration of the event or detection time windows should be updated, if applicable.

In total, we discovered that 5,055,871 alerts could be aggregated, out of which 676,615 alerts had a unique combination of source IP, target IP, target port, event type, sensor ID, and timestamp. The remaining 4,379,256 alerts shared the key values, but differed in timestamp. Thus, 54.45 % of alerts from the dataset were aggregated using this method.

Figure 1 shows differences in the timestamps of consequent alerts. The shape of the graph corresponds to our assumptions. There are two apparent horizontal lines representing the two clusters of similar timestamp differences, one at 5 minutes, the second at 1 hour. These are caused by the typical settings of an intrusion detection system working in fixed time windows, typically 5 minutes and 1 hour. The fluent transitions are caused by sensors detecting in real time.

D. Overlapping Sensors

The aggregation of alerts from overlapping sensors includes the aggregation of alerts according to their source IP, source port, target IP, target port, and event type. A list of sensor IDs was assigned to each unique tuple. If there were more sensor IDs assigned to a unique tuple, the alerts could be aggregated. The earliest timestamp is kept, while the detection time windows and event duration should be updated, if applicable.

We discovered that 67,956 alerts could be aggregated, out of which 33,344 alerts had a unique combination of source IP, source port, target IP, target port, and event type. The remaining 34,612 alerts were differed in sensor ID and timestamp.

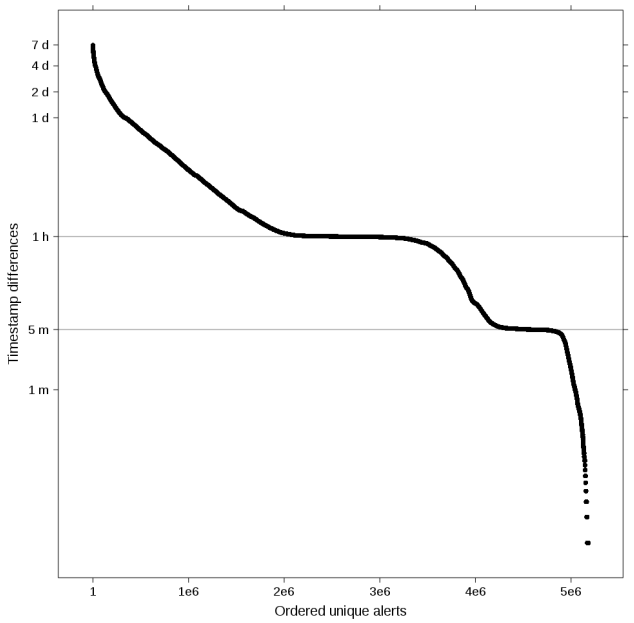

Fig. 1. Timestamp differences of aggregable alerts in decreasing order.

Using this method, 0.43 % of alerts from the dataset were aggregated.

E. Non-overlapping Sensors

The aggregation of alerts from non-overlapping sensors first aggregates the alerts according to their source IP, target port, and event type. The alerts were aggregated if more target IPs and sensor IDs were assigned to one couple. The earliest timestamp is kept, while the detection time windows, event duration, and range of targets should be updated, if applicable.

We discovered that 2,573,196 alerts could be aggregated, out of which 171,033 alerts had a unique combination of source IP, target port, and event type. The remaining 2,402,163 alerts were aggregable with the unique alerts. Using this method we aggregated 29.86 % of alerts from the dataset.

F. Aggregated Data

Using the four methods presented earlier, we aggregated 85.98 % of alerts from the dataset. Following the order of methods, we prevented the same alerts from being aggregated twice. Our results are comparable to the results of Lin et al. [9], who estimated the number of unique alerts to be 8-25 %. Although Lin et al. used an older and smaller dataset and considered only two types of aggregates (continuing events and the same events detected by multiple sensors), the results are similar. Our results exceed estimates by Valdes and Skinner [5], who assumed a 50-66 % reduction rate was possible. The reduction rate of 25 % discussed by Valeur et al. [8] for alerts reported by different sensors corresponds to our finding on overlapping and non-overlapping sensors. We believe that, based on our experience and with related work in mind, a rule of thumb may be set for alert aggregation: up to 85 % of alerts can be aggregated by all the methods combined.

V. Conclusions

In this paper, we have presented a case study into the aggregation of alerts from network intrusion detection systems in a security alert sharing platform, named SABU. We presented four prime use cases for alert aggregation and analyzed a real-life dataset to discover volumes of raw and aggregated data and to set an optimal time window for alert aggregation. The data for the experiment were obtained from the SABU alert sharing and analysis platform deployed in the national research and education network of the Czech Republic.

In our future work, we are going to implement alert aggregation tools and deploy them in the SABU platform. The measured figures will be taken into consideration in the further development of the SABU platform and its components, as well as in further research. There are also several implications regarding the sensors and quality of the data, which should be resolved if the data from the sensor are going to be exchanged. Incomplete reports, such as network scanning reports without target port, are ambiguous for further correlation. An interesting challenge would be to optimize sensor settings to minimize the time windows between the aggregated alerts. Further, deeper examination of the structure of the network would provide potentially interesting information on the causes of sensor overlaps.

Acknowledgments

We would like to thank CESNET for their support with data acquisition and processing.

This research was supported by the Security Research Programme of the Czech Republic 2015 - 2020 (BV III / 1 VS) granted by the Ministry of the Interior of the Czech Republic under No. VI20162019029 The Sharing and analysis of security events in the Czech Republic.

Martin Laštovička is Brno Ph.D. Talent Scholarship Holder – Funded by the Brno City Municipality.

References

[1] E. Vasilomanolakis, S. Karuppayah, M. Mühlhäuser, and M. Fischer, "Taxonomy and Survey of Collaborative Intrusion Detection," *ACM Comput. Surv.*, vol. 47, no. 4, pp. 55:1–55:33, May 2015.

[2] A. Sperotto, G. Schaffrath, R. Sadre, C. Morariu, A. Pras, and B. Stiller, "An Overview of IP Flow-Based Intrusion Detection," *Communications Surveys Tutorials, IEEE*, vol. 12, no. 3, pp. 343–356, Third 2010.

[3] A. Chuvakin, K. Schmidt, and C. Phillips, *Logging and Log Management: The Authoritative Guide to Understanding the Concepts Surrounding Logging and Log Management*. Syngress Publishing, 2013.

[4] H. T. Elshoush and I. M. Osman, "Alert correlation in collaborative intelligent intrusion detection systems – A survey," *Applied Soft Computing*, vol. 11, no. 7, pp. 4349–4365, 2011.

[5] A. Valdes and K. Skinner, "Probabilistic alert correlation," in *International Workshop on Recent Advances in Intrusion Detection*, 2001, pp. 54–68.

[6] F. Cuppens and A. Miege, "Alert correlation in a cooperative intrusion detection framework," in *Security and Privacy, 2002. Proceedings. 2002 IEEE Symposium on*, 2002, pp. 202–215.

[7] H. Debar and A. Wespi, "Aggregation and correlation of intrusion-detection alerts," in *International Workshop on Recent Advances in Intrusion Detection*. Springer, 2001, pp. 85–103.

[8] F. Valeur, G. Vigna, C. Kruegel, and R. A. Kemmerer, "Comprehensive approach to intrusion detection alert correlation," *IEEE Transactions on Dependable and Secure Computing*, vol. 1, no. 3, pp. 146–169, July 2004.

[9] H.-S. Lin, H.-K. Pao, C.-H. Mao, H.-M. Lee, T. Chen, and Y.-J. Lee, "Adaptive alarm filtering by causal correlation consideration in intrusion detection," in *New Advances in Intelligent Decision Technologies*. Springer, 2009, pp. 437–447.