A Graph-based Representation of Relations in Network Security Alert Sharing Platforms

Martin Husák, Milan Čermák
Institute of Computer Science and Faculty of Informatics
Masaryk University, Brno, Czech Republic
{husakm,cermak}@ics.muni.cz

***Abstract*—In this paper, we present a framework for graph-based representation of relation between sensors and alert types in a security alert sharing platform. Nodes in a graph represent either sensors or alert types, while edges represent various relations between them, such as common type of reported alerts or duplicated alerts. The graph is automatically updated, stored in a graph database, and visualized. The resulting graph will be used by network administrators and security analysts as a visual guide and situational awareness tool in a complex environment of security alert sharing.**

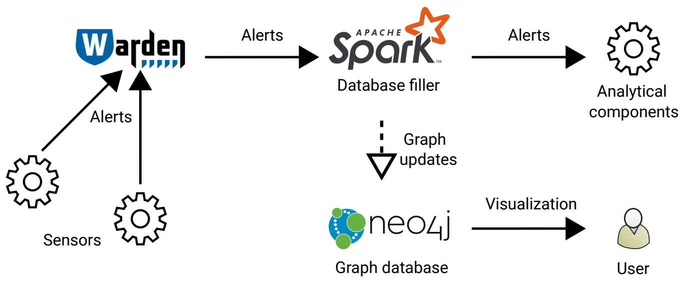

Fig. 1. System architecture.

I. Introduction

Collaborative intrusion detection and information exchange are emerging trends in network security [1]. The alert sharing platforms allow exchange of security alerts between the intrusion detection systems. Various methods of alert correlation were proposed to analyze complex network attacks. However, it is relatively hard to understand the correlations in a complex collaborative environment without knowing the network background. The alert sharing platforms mostly focus on the core issues, i.e., security analyzes, and may underestimate issues of platform management [1], [2].

One of the basic problems in collaborative network security environment is getting a quick overview of the collaborative platform. Having a list of contributing sensors and a taxonomy of alerts is a necessity. However, these information are not sufficient and do not tell anything about the actual relations in the platform, especially in highly distributed and heterogeneous alert sharing platforms. The users, typically network administrators and security specialists, need to know what sensors are providing what piece of information or if there are more sensors reporting redundant alerts [2], [3]. Further, the users would like to know which sensors report bogus alerts and which alert types are worth correlating.

We surveyed the users of an alerts sharing platform and obtained the following list of typical questions on contributing sensors and shared alerts:

1) What types of alerts does a certain sensor provide? How many alert types is the sensor able to report?
2) Which sensors report a certain alert type? In which networks is such event observed?
3) Which sensors report duplicate or similar events? Is there an overlap in their detection scopes?
4) Which sensors report bogus alerts? How much?

To answer those questions, we designed a graph-based representation of relations between alert types and sensors in alert sharing platforms, which reflects the aforementioned questions. A framework was implemented to automatically obtain and represent the relations in SABU[1], an alert sharing platform developed by CESNET and Masaryk University and deployed in Czech academic network.

II. System Architecture

The framework was implemented as a component of SABU, alert sharing and analysis platform. SABU includes sensors (mostly network intrusion detection systems), analytical tools, and other components. The security alerts are shared via a hub named Warden[2]. IDEA[3] is used as a common data exchange format and alert taxonomy.

The relation representation framework consists of graph database and a database filler, as depicted in Figure 1. The database filler was implemented as one of the SABU analytical tools using Apache Spark[4]. It receives the alerts from Warden and constructs the graph. Well-known open-source graph database Neo4j[5] is used for storing the resulting graph. Neo4j has a web frontend, which can be used for querying and visualization, but that requires at least a basic knowledge of the query language. However, a custom frontend with predefined requests will be included in another SABU component.

The database filler is an extensible framework for collecting and updating information on nodes and edges in the graph.

[1]https://sabu.cesnet.cz/
[2]https://warden.cesnet.cz/
[3]https://idea.cesnet.cz/
[4]http://spark.apache.org/
[5]https://neo4j.com/

Nodes and edges have a type and a set of values. For example, a node of type *sensor* has a set of values like name of the sensor and number of reported alerts by the sensor. Similarly, an edge type stands for the relation it represents. For example, an edge of type *detects* connects *sensor* and *alert_type*. For each node and edge, there is a separate script that collects required information. All the edges are unidirectional due to restrictions of Neo4j. However, the query language allows to ignore edge direction. The framework continuously collects the data from the scripts and updates the database accordingly. The framework can be extended by additional scripts, the only limitation is that edges have to connect existing nodes.

III. Monitored Relations

In the demonstration, we are going to show the relations between sensors and alert types in the SABU alert sharing platform. Sensors and alert types are represented as nodes. For both node types, we store the following information:

- node type (*sensor* or *alert_type*),
- sensor name or alert type,
- number of reported alerts in total and in the last hour,
- number of duplicate and continuing alerts [3].

The edges represent the relations between arbitrary nodes and an edge can has a type representing the watched relation. Each edge has the following properties:

- edge type (one of the relations, see below),
- number of observations in total and in the last hour,
- average, minimal, and maximal time differences between the correlated alerts.

Currently, we monitor the following relations:

- *detects* – A sensor reports alert of a certain type.
- *same_source_sensor* – The two sensors reported the same source of an event.
- *same_target_sensor* – The two sensors reported the same target of an event.
- *same_source_alert* – The two alerts of different categories are sharing the same source.
- *same_target_alert* – The two alerts of different categories are sharing the same target.
- *duplicate* – The two sensors reported duplicate alerts, i.e., alerts with the same type, source, and target.

Figure 2 shows a sample of the graph obtained by monitoring the alerts in the SABU platform. By traversing the graph, we can easily answer the questions stated in the introduction. For example, listing the neighbors of a *sensor* node over the edges of a specified type answers the question 1, similarly for question 2 and *alert_types*. The *same_** relations suggest that the alerts from the two sensors or of the two types are worth correlating. This information, along with the time differences between correlated alerts, is highly interesting for further analysis. The *duplicate* relation indicates an overlap in detection scope of the two sensors and appearance of duplicated alerts (question 3). Thus, it is closely tied and actively used by aggregation component of the SABU platform [3]. Counting the number of duplicates and continuing alerts (question 4) is

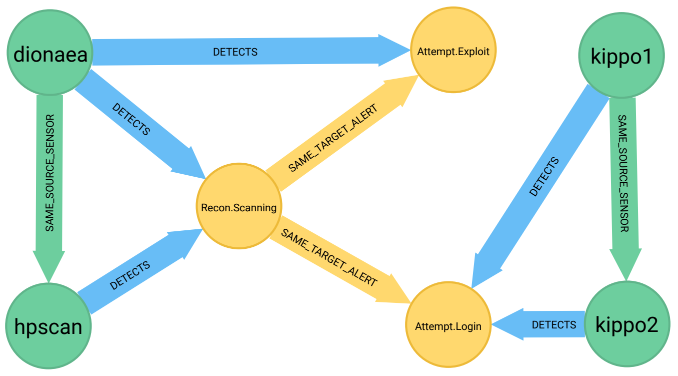

Fig. 2. Graph sample depicting 4 sensors, 3 alert types, and 3 relation types.

a very similar functionality of the two components and will be further integrated.

IV. Conclusion and Future Work

We presented a framework for graph-based representation of relations between sensors and alerts in an alert sharing platform. The framework helps the users in understanding the data from distributed intrusion detection systems and getting an overview of the alert sharing platform. It does not require any prior knowledge of the platform, but continuously captures actual alerts to get the insight into it. Users may quickly identify which sensors reports which alerts, alerts from which sensors are similar or duplicated, where the overlaps in detection scopes are, and other information. The framework is extensible, so that new relations can be easily added and continuously monitored.

In our future work, we are going to further develop the framework and integrate it into the SABU alert sharing platform. The data from the deployment will be used in data analyzes and further development of SABU. For example, checking for overlaps in detection scopes is going to be used by the alert aggregation component [3]. The graph-based representation of alert relations will also be examined for the purpose of security alert analysis.

Acknowledgments

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.

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] O. Serrano, L. Dandurand, and S. Brown, "On the design of a cyber security data sharing system," in *Proceedings of the 2014 ACM Workshop on Information Sharing & Collaborative Security*, 2014, pp. 61–69.

[3] M. Husák, M. Čermák, M. Laštovička, and J. Vykopal, "Exchanging Security Events: Which And How Many Alerts Can We Aggregate?" in *2017 IFIP/IEEE International Symposium on Integrated Network Management (IM)*, 2017.