Electronic Communications of the EASST
Volume 080 (2021)

Conference on Networked Systems 2021 (NetSys 2021)

Property Inference-based Federated Learning Groups for Collaborative Network Anomaly Detection

Jens Wettlaufer

5 pages

Guest Editors: Andreas Blenk, Mathias Fischer, Stefan Fischer, Horst Hellbrueck, Oliver Hohlfeld, Andreas Kassler, Koojana Kuladinithi, Winfried Lamersdorf, Olaf Landsiedel, Andreas Timm-Giel, Alexey Vinel
ECEASST Home Page: http://www.easst.org/eceasst/ ISSN 1863-2122

Property Inference-based Federated Learning Groups for Collaborative Network Anomaly Detection

Jens Wettlaufer

Universität Hamburg
jens.wettlaufer@uni-hamburg.de

Abstract: While the use of anomaly detection in network security has a long research history, it is rarely used in practice. Besides privacy concerns when applied in cross-network settings, and a more difficult attack interpretation, the major drawback consists of the high number of false alarms. One reason is the heterogeneity of sources the model is trained on. In this paper, we propose a network anomaly detection extension that counteracts the heterogeneity of participants by dividing them into learning groups during central or federated training. The learning groups finally contain similar behaving clients, e.g., light bulbs, or PCs of the same department. Similar behavior is extracted by hierarchically clustering the predictions of all individual client models similar to a passive property inference attack. Our preliminary results based on infiltration attacks of the IDS2017 dataset show that the method increases the accuracy and F1 score up to 4.4% and 2.5%, respectively.

Keywords: anomaly detection, unsupervised learning, property inference, network security, hierarchical clustering, collaborative learning

1 Introduction

Anomaly detection intends to identify deviations or non-conforming patterns from expected behavior. These patterns are called anomalies or outliers. It is broadly researched in tumor identification on MRI images, credit card fraud, identity theft, spam detection, and intrusion detection [CBK09]. The common problem of a high false positive rate does not concern for example credit card fraud that much, because the users can clarify the situation easily. However, in network intrusion detection of large networks, even a false positive rate of 1% in a million network flows a day is impractical to manually assess [SP10]. Additionally, the interpretation of an anomaly often requires expert knowledge when the anomaly detection task is not clearly formulated [SP10]. Furthermore, heterogeneous networks with different types of devices, end users, and thus behavior patterns make it hard to learn normal behavior, and interpret anomalies afterwards. Essentially, while anomaly models intend to capture a normal profile, such that subtle deviations are detected, the heterogeneity of entities counteracts this. In addition, real-world challenges arise for the usage of centralized anomaly detection. First, it is challenging for large networks to collect all data for training and prediction. Second, startups or small businesses may have too few data samples to train the model effectively. This could be circumvented by collaborating with others, which introduces privacy concerns. In consequence, network anomaly detection is rarely used in practice, even though it is capable of detecting unknown attacks in contrast to signature or policy-based approaches. To overcome these problems, we suggest to extend existing

anomaly detection procedures by dividing heterogeneous clients during the training into groups with similar behaviors, e.g., service usage or user behavior, in an unsupervised fashion. This enables given anomaly models to learn normal behavior more precisely in order to detect even subtle deviations, such as a slight increase of *ssh* traffic in the IT department. Further, anomalies can be interpreted and solved easier due to a decreased number of participating clients for a particular anomaly. Therefore, our main contributions include:

Identification of similar behavior. We provide an unsupervised method based on the principle of property inference to identify similar behaving clients, further called *learning groups*. The method is not restricted to, but evaluated on network anomaly detection in this work.

Central and collaborative learning. We show that this method can be applied to central and collaborative learning by providing a federated learning extension that does not interfere with its actual protocol nor its privacy and performance guarantees.

Preliminary results. Our preliminary results achieve an increased accuracy by 2.3-4.4% and F1 score by 1.2-2.5% on average between different models.

The rest of the paper is structured as follows. Section 2 identifies differences to existing solutions. Section 3 provides the main research questions, explains the approach, and discusses preliminary results. Finally, Section 4 summarizes the paper and states next steps.

2 Related Work

Network anomaly detection models for different attacks are widely researched. For example, Fernandes et al. [FRC+19] survey numerous papers emphasizing that the kind of data, i.e., tcpdump, SNMP, and IP, determines the detection capabilities. Our approach is not determined by a specific method or data type. Some works intend to profile IP hosts based on previously analyzed traffic or host parameters [YR08, MDBC10, JGL15]. Our method does not necessarily target a specific network anomaly problem, but rather improves existing models. Additionally, we intend to integrate the division of learning groups into the actual training without the need for a prior assessment of hosts, which might be infeasible in cross-network scenarios in regard to privacy. Aiello et al. [AKM+05] analyze host-level communities of interest, i.e., entities that share a common goal or environment, based on host connections to other hosts in regard to popularity and frequency. While this approach is not characteristic dependent, we argue that host-level goals are not necessarily interchangeable with similar behavior of hosts, e.g., regarding time shifts of actions. Nguyen et al. [NMM+19] present a federated anomaly detection in a two-fold approach. Calculated fingerprints are clustered to identify IoT device types, which are then used as static properties to apply their anomaly model within a group of devices of one device type only. In contrast, our approach aims at integrating the grouping step into the actual anomaly model training. Additionally, our work goes beyond device type similarities and allows for the creation of non-obvious learning groups.

3 Property Inference-based Federated Learning Groups

This work hypothesizes that if groups of similar behaving clients, e.g., smart light bulbs, web servers, or PCs of the HR department, separately train together, the anomaly detection results in

a more accurate prediction. Thus, we aim at answering the following research questions:

RQ1 *How can we automatically infer client behavior properties during training?*
The goal is to identify similar behaving clients independent of the anomaly detection task, model, and training procedure.

RQ2 *Which (non-obvious) properties exist that improve the anomaly detection after training in property-based learning groups?*
[NMM+19] achieve promising results with device type specific learning groups. However, we also aim at recognizing non-obvious groups of devices that improve the prediction.

RQ3 *How can we apply learning groups in a privacy-preserving and collaborative scenario?*
To further apply this method in large networks with higher performance and privacy requirements, we need to show that the method also works for collaborative learning.

The following paragraphs first sketch an answer to *RQ3* and focus further on *RQ1*, while addressing *RQ2* partially in the discussion of our preliminary results.

Approach. Federated learning (FL) is a distributed round-base learning technique between clients with private data and a coordinating server. We modify the FL procedure as follows. First, the server distributes a tuple (M_{global}, s_n), i.e., a randomly initialized global model and n random samples of network traffic, to the participating clients. Then, each client c_i trains the received M_{global} with its own data, makes a prediction $p_n^{c_i}$ of the random samples s_n, and extracts its model updates $U_{global}^{c_i}$. It then sends the tuple $(U_{global}^{c_i}, p_n^{c_i})$ back to the server. The server now applies the learning group metric to all received predictions $p_n^{c_i}$ and determines the learning groups. Finally, the clients continue FL training together within their assigned learning groups.

Learning group metric. Client behavior is expressed through client data. Client data is used to train ML models, which then become a representation of the data. Thus, client behavior can be inferred based on the trained model. Therefore, a metric to identify similarities between models is required. Related work in the field of property and membership inference attacks on ML has shown that it is feasible to infer properties of the training data that are solely represented in the model predictions [SSSS17]. Consequently, we measure the similarity by comparing the prediction outputs of each trained client model for a given input. To identify client behaviors that are closest to each other, we choose hierarchical clustering, in particular Agglomerative Clustering. While we used three default distance metrics for our first results, further distance metrics [PAH+19] will be analyzed subsequently.

Dataset and implementation. The preliminary results are based on the IDS2017 dataset [SHG18] containing simulated corporate communication of ten clients and two servers. We choose a subset of the attacks containing infiltration attacks. Then, we apply the recommended task related flow-based features [SHG18] to our simple anomaly detection models, i.e., *Isolation Forest* and *Elliptic Envelope*. We provide four baselines including all clients training together and separate, as well as training in obvious learning groups, i.e., clients/servers and operating systems. The values in Table 1 are the average of five rounds. Further, we test three variants of establishing learning groups by varying the percentile of the training data before clustering, while keeping three as a fixed number of clusters. Additionally, we apply three different clustering metrics (CM) to the client-wise predictions of the samples s_n with $n = 1000$: *average* (average distance),

complete (maximum distance), and *ward* (minimizes the variance of the clusters being merged).
Preliminary results and discussion. Table 1 compares all performed tests, where *min* and *max* correspond to the result of a single learning group, while *mean* averages all of them. Cells highlighted in red provide the information that they consist of one learning group that scores worse than the *all together* prediction. The goal, however, is to achieve a set of learning groups that all score equal or greater than the *all together* learning group. Thus, none of the baselines address the most effective client properties for learning group separation.

Our experiments have several limitations, including the lack of comparison in regard to the fixed size of clusters, the use of simple ML models with discrete instead of probabilistic predictions, and a restricted set of features.

	Learning Groups	**Isolation Forest** *Accuracy*			**Isolation Forest** *F1*			**Elliptic Envelope** *Accuracy*			**Elliptic Envelope** *F1*		
		min	***mean***	***max***	***min***	***mean***	***max***	***min***	***mean***	***max***	***min***	***mean***	***max***
Baselines	all together		0.863			0.927			0.914			0.955	
	all separate	0.835	0.886	0.946	0.901	0.939	0.972	0.891	0.926	**0.973**	0.942	0.961	**0.986**
	machine	0.883	0.899	0.930	0.938	0.947	0.964	0.903	0.911	0.925	0.949	0.953	0.961
	machine + os	0.850	0.893	0.940	0.919	0.943	0.969	0.907	0.923	0.955	0.951	0.960	0.977
10% Training 3 Clusters	CM: average	0.870	0.899	0.944	0.931	0.947	0.971	0.915	0.921	0.934	0.956	0.959	0.966
	CM: complete	0.844	0.883	0.944	0.915	0.938	0.971	0.915	0.921	0.934	0.956	0.959	0.966
	CM: ward	**0.902**	0.908	0.926	**0.949**	**0.952**	0.961	**0.929**	0.935	0.949	**0.963**	0.966	0.974
20% Training 3 Clusters	CM: average	0.863	0.885	0.926	0.926	0.939	0.962	0.926	0.936	0.949	0.961	**0.967**	0.974
	CM: complete	0.893	**0.907**	0.946	0.943	0.951	0.972	0.926	0.936	0.949	0.961	**0.967**	0.974
	CM: ward	0.882	0.901	0.940	0.937	0.948	0.969	0.926	0.936	0.949	0.961	**0.967**	0.974
30% Training 3 Clusters	CM: average	0.867	0.897	0.940	0.929	0.945	0.969	**0.928**	**0.937**	0.949	0.963	**0.967**	0.974
	CM: complete	0.870	0.901	**0.950**	0.930	0.947	**0.975**	0.924	0.933	0.949	0.961	0.965	0.974
	CM: ward	0.850	0.892	0.950	0.919	0.942	0.974	0.924	0.933	0.949	0.961	0.965	0.974

Table 1: Evaluation of predicting normal behavior on the infiltration attacks dataset. Bold values represent the maximum in a column. Highlighted values are less than the *all together* baseline.

4 Conclusion and Future Work

In this paper, we present an unsupervised division of participants into learning groups during the semi-supervised training of an anomaly detection model. Clients within learning groups have trained the initial model in a similar way, meaning that they have similar data and thus similar behavior. We showed that it is also feasible to introduce this method in a collaborative learning setup without revealing private data. By applying it to network anomaly detection of infiltration attacks, it achieves an increased accuracy and F1 score by up to 4.4% and 2.5%, respectively, in contrast to the common training procedure.

Teh next steps include advancing from a fixed cluster size to a threshold, a detailed analysis of distance metrics for hierarchical clustering, and a deeper look into the data to answer *RQ2*. Additionally, further models, e.g., neural networks and datasets of different domains, e.g., IoT, industrial networks, or image recognition, will be evaluated to show the independence of this approach. Subsequent work will reduce the attack surface on the property inference step, e.g., by introducing secure multi-party computation, and generalize the approach on other ML tasks.

Bibliography

[AKM+05] W. Aiello, C. Kalmanek, P. McDaniel, S. Sen, O. Spatscheck, J. Van der Merwe. Analysis of Communities of Interest in Data Networks. In *Passive and Active Network Measurement*. 2005.

[CBK09] V. Chandola, A. Banerjee, V. Kumar. Anomaly detection: A survey. *ACM Computing Surveys*, 2009.

[FRC+19] G. Fernandes, J. J. P. C. Rodrigues, L. F. Carvalho, J. F. Al-Muhtadi, M. L. Proença. A comprehensive survey on network anomaly detection. *Telecommunication Systems*, 2019.

[JGL15] A. Jakalan, J. Gong, S. Liu. Profiling IP hosts based on traffic behavior. In *IEEE International Conference on Communication Software and Networks*. 2015.

[MDBC10] G. Münz, H. Dai, L. Braun, G. Carle. TCP Traffic Classification Using Markov Models. In *Traffic Monitoring and Analysis*. 2010.

[NMM+19] T. D. Nguyen, S. Marchal, M. Miettinen, H. Fereidooni, N. Asokan, A.-R. Sadeghi. DÏoT: A Federated Self-learning Anomaly Detection System for IoT. 2019.

[PAH+19] V. B. S. Prasath, H. A. A. Alfeilat, A. B. A. Hassanat, O. Lasassmeh, A. S. Tarawneh, M. B. Alhasanat, H. S. E. Salman. Distance and Similarity Measures Effect on the Performance of K-Nearest Neighbor Classifier – A Review. *Big Data*, 2019.

[SHG18] I. Sharafaldin, A. Habibi Lashkari, A. A. Ghorbani. Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization:. In *Proceedings of the 4th International Conference on Information Systems Security and Privacy*. 2018.

[SP10] R. Sommer, V. Paxson. Outside the Closed World: On Using Machine Learning for Network Intrusion Detection. In *IEEE Symposium on Security and Privacy*. 2010.

[SSSS17] R. Shokri, M. Stronati, C. Song, V. Shmatikov. Membership Inference Attacks against Machine Learning Models. 2017.

[YR08] T.-F. Yen, M. K. Reiter. Traffic Aggregation for Malware Detection. In *Detection of Intrusions and Malware, and Vulnerability Assessment*. 2008.