Machine Learning to Automate Network Segregation for Enhanced Security in Industry 4.0

Firooz B. Saghezchi[1], Georgios Mantas[2], José Ribeiro[2], Alireza Esfahani[2], Hassan Alizadeh[1], Joaquim Bastos[2] and Jonathan Rodriguez[1]

[1] University of Aveiro, Campus Universitário de Santiago, 3810-193 Aveiro
{firooz, hassan.alizadeh, jonathan}@ua.pt
2 Instituto de Telecomunicações, Campus Universitário de Santiago, 3810-193 Aveiro
{gimantas, jcarlosvgr, alireza, jbastos}@av.it.pt

Abstract. The heavy reliance of Industry 4.0 on emerging communication technologies, notably Industrial Internet-of-Things (IIoT) and Machine-Type Communications (MTC), and the increasing exposure of these traditionally isolated infrastructures to the Internet, are tremendously increasing the attack surface. Network segregation is a viable solution to address this problem. It essentially splits the network into several logical groups (subnetworks) and enforces adequate security policy on each segment, e.g., restricting unnecessary intergroup communications or controlling the access. However, existing segregation techniques primarily depend on manual configurations, which renders them inefficient for cyber-physical production systems because they are highly complex and heterogeneous environments with massive number of communicating machines. In this paper, we incorporate machine learning to automate network segregation, by efficiently classifying network end-devices into several groups through examining the traffic patterns that they generate. For performance evaluation, we analysed the data collected from a large segment of Infineon's network in the context of the EU funded ECSEL-JU project "SemI40". In particular, we applied feature selection and trained several supervised learning algorithms. Test results, using 10-fold cross validation, revealed that the algorithms generalise very well and achieve an accuracy up to 99.4%.

Keywords: Industry 4.0, Cyber-Physical Production Systems, Security, Machine Learning, Network Segregation, IIoT, MTC, Traffic Classification.

1 Introduction

Recent advancements in information and communications technologies and the emergence of Industrial Internet of Things (IIoT) and Machine-to-Machine (M2M) communications bring about the fourth industrial revolution (Industry 4.0), where product, man and machine are fully interconnected across the whole supply chain from supplying raw material to providing the final product to the market. This allows more efficient, flexible and customized production as well as remote operation and control. However, connecting previously isolated production facilities to the Internet tremendously increases the attack surface, for most of the equipment is still legacy, primarily designed for reliable operation, with certain limited interfaces between the legacy equipment and the modern infrastructure [1]. Therefore, there is an urgent need to address cyber-physical security in these key infrastructures.

Network segregation is considered as an effective access control mechanism for information security management in ISO/IEC 17799:2005. It essentially divides the network into subnetworks, each called a network segment. Such splitting helps boost not only the network performance, by minimizing the local traffic, but also the network security through: i) limiting the broadcast domain to the local segment; ii) reducing the attack surface, in case of compromise in the machines hosted by a network segment; and iii) allowing the access privileges be independently controlled for each network segment. Furthermore, network segregation can also limit the effect of local failures on other network segments. Security Group Tagging (SGT) and Access Control List are common practices for implementation of network segregation at different layers. However, in an industrial network there are tremendously huge number of heterogeneous machines, mostly legacy, communicating with each other. There is limited or no documentation at all about their communication profiles. Therefore, it is impractical to manually define rules for identifying the communication patterns in order to group the end devices. As illustrated by Figure 1, a viable approach is to use Machine Learning (ML) to group network devices by learning their communication patterns as there exist considerable regularities in the way machines communicate or interact in an industrial network.

In this paper, we employ supervised ML algorithms to identify communication patterns in an Industry 4.0 Cyber-Physical Production Systems (CPPS) by classifying the traffic flows crossing the network. The data that we analyse has recently been collected from a large segment of Infineon's network, which is around 5 GB network trace files, in PCAP format, containing only the packet headers plus the initial 20 bytes of each payload. The independence of our flow processing algorithms from the packets' payload is crucially important to ensure the preservation of user privacy as well as the protection of industry's intellectual property. We construct labelled datasets using a Deep Packet Inspection (DPI) tool and apply following supervised ML algorithms: One Rule (OneR), Decision Tree (DT), Naïve Bayes (NB), Bayesian Network (BN), Support Vector Machine (SVM), and k-Nearest Neighbour (k-NN). The results show that among them, DT and k-NN outperform the others, with an accuracy reaching up to 99.4%. To the best of our knowledge, this is the first attempt on applying ML for network segregation and traffic analysis in industrial networks.

The rest of this paper is organized as follows. Section 2 reviews the related work. Section 3 describes how we construct datasets from the raw data (which is in PCAP format) collected from Infineon's network. Section 4 defines the metrics that we use for performance evaluation. Section 5 presents the supervised ML algorithms that we use for traffic classification. Section 6 presents and discusses the results. Finally, Section 7 concludes the paper and draws some guidelines for the future work.

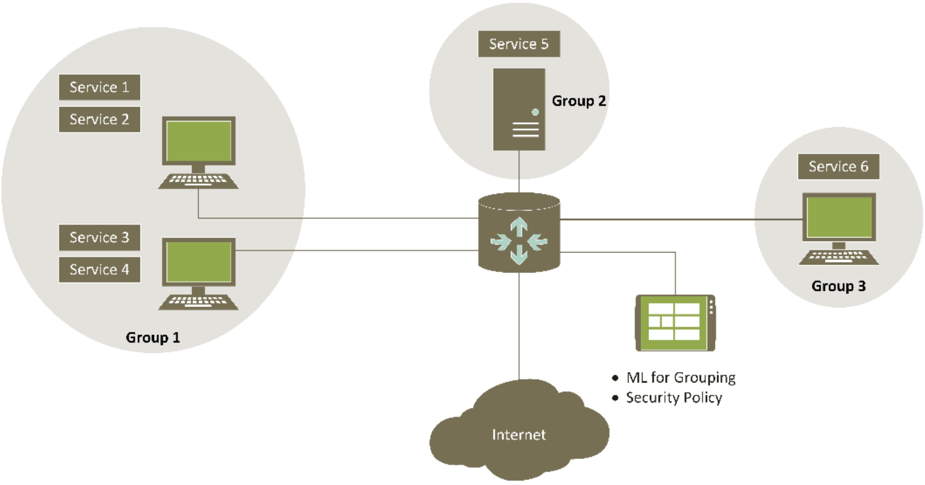

Figure 1. Machine Learning for automatic grouping of network endpoints to apply proper security policy on each network segment.

2 Related Work

There are three main approaches for network traffic classification: *port-based*, *payload-based* and *flow features-based* [2-13]. In the early days of the Internet, traffic classification relied on transport layer port numbers, typically registered with the Internet Assigned Numbers Authority (IANA) to designate well-known applications. Nonetheless, more recently, growing number of applications, notably those for Peer-to-Peer (P2P) file sharing, hide their identity, by using a random port number or the well-known port of other applications, which renders port-based approach inefficient [8].

On the other hand, traffic classification based on payload analysis is more reliable and is mostly incorporated by commercial solutions, e.g., *Bro*, *Prelude*, and *Snort*, where packet payloads are inspected for specific string patterns (also called *signatures*) of known applications. Although this approach is more accurate, it suffers from concerns for protecting intellectual property – which is especially sensitive in industrial networks – and violating user's privacy. Furthermore, it scales poorly for high bandwidths, is computationally expensive, and is inefficient for encrypted packets [8,9].

Finally, flow features-based approach adopts ML or statistical algorithms to build a model for each traffic type, by feeding a training set containing flow examples. The model is then able to predict class membership for new instances by examining the feature values for unknown flows. Learning algorithms that are used for traffic classification generally fall into two main categories: *supervised* and *unsupervised* [10]. The latter groups traffic flows into different clusters according to similarities observed in the feature values [12]. These clusters are not predefined and the algorithm itself determines their number and statistical nature. In contrast, supervised algorithms require the class membership of each training example, also called *label*, beforehand, and based on it, construct a general rule for determining the label of an unseen future flow [11, 13].

For flow feature, different traffic attributes are extracted, such as flow duration, max/min/average/standard deviation of packet size, number of sent/received packets, packet inter-arrival time in the forward or backward direction, TCP flags, the size of the first ten packets, and so forth [3, 8, 9]. Finally, due to the limitations of port-based and payload-based approaches, current research is primarily focused on ML approach.

3 Dataset Generation

We use *libprotoident 2.0.12*[1] DPI to construct a labelled dataset. The output file is in a Comma Separated Values (CSV) format (see Table 2), where, for each row, the first element indicates the label, i.e., the Application protocol and the next elements indicate flow attributes, representing training features in the order listed by Table 1.

Table 1. List of training features, representing different columns of the dataset.

Column	Feature
1	*Application layer protocol (label)*
2	Transport protocol (e.g., 6 stands for TCP and 17 stands for UDP)
3	Total number of packets sent in the forward direction
4	Total number of bytes sent in the forward direction
5	Total number of packets sent in the backward direction
6	Total number of bytes sent in the backward direction
7	Minimum payload size sent in the forward direction
8	Mean payload size sent in the forward direction
9	Maximum payload size sent in the forward direction
10	Standard deviation of payload size sent in the forward direction
11	Minimum payload size sent in the backward direction
12	Mean payload size sent in the backward direction
13	Maximum payload size sent in the backward direction
14	Standard deviation of payload size sent in the backward direction
15	Minimum packet inter-arrival time in the forward direction
16	Mean packet inter-arrival time for packets sent in the forward direction
17	Maximum packet inter-arrival time in the forward direction
18	Standard deviation of packet inter-arrival time in the forward direction
19	Minimum packet inter-arrival time in the backward direction
20	Mean packet inter-arrival time in the backward direction
21	Maximum packet inter-arrival time in the backward direction
22	Standard deviation of packet inter-arrival time in the backward direction
23	Flow duration (in microseconds)

Table 2. Few training examples from the constructed dataset.

Label	Training Examples
HTTP	,6,3,19,5,85,0,6,19,9,0,17,85,34,81,386,982,421,0,0,1,0,1163,1499110420.978695
DHCP	,17,23,2254,0,0,98,98,98,0,0,0,0,0,0,0,1,0,0,0,0,0,11,1499110681.654235
DNS	,17,23,4002,0,0,174,174,174,0,0,0,0,0,0,0,1,0,0,0,0,0,11,1499110482.156615
RTP	,17,0,0,2,344,0,0,0,0,172,172,172,0,0,0,0,0,0,1,2,1,2,1501080810.070283

[1] https://github.com/wanduow/libprotoident

4 Evaluation Metrics

For numerical evaluation, we perform k-fold cross validation, with k=10. That is, we divide the whole data into k subsets and repeat the test k times. In each trial, we use one of the k subsets as the test set and the rest k-1 subsets as the training set. We then calculate the average performance over all k trials. This in fact provides a good indication of algorithm's generalisation capability when classifying an unseen data point [9]. Finally, we use the following standard evaluation metrics [10]:

- *Accuracy*: the percentage of correctly classified instances over the total number of instances;
- *Precision*: the number of class members classified correctly over the total number of instances classified as class members;
- *Recall* (or *true positive rate*): the number of class members classified correctly over the total number of class members.
- *F-Measure*: a combination of precision and recall defined specifically as their harmonic mean. The traditional F-measure, also called balanced F-score or F_1 measure, is calculated as follows:

  $$F = 2\frac{Precision \times Recall}{Precision + Recall}, \tag{1}$$

  which is a special case of the general F_β measure ($\beta \geq 0$).

  $$F_\beta = (1+\beta^2)\frac{Precision \times Recall}{\beta^2 Precision + Recall} \tag{2}$$

  Two other commonly used F measures are the F_2 measure, which weights recall more than precision, and the $F_{0.5}$ measure, which puts more emphasis on precision than recall.

5 Classification Algorithms

In the following, we briefly elaborate ML algorithms that we use for traffic classification in an Industrial Network, which have widely been employed for Internet traffic classification [8, 9] as well. Note that all of these algorithms are supervised, i.e., requiring labels for training.

One Rule (OneR): is a simple yet effective classification algorithm based on only one rule. During the training phase, it creates one rule for each feature and picks the one that leads to the minimum classification error as the general classification rule.

Decision Tree (DT): creates a model based on a tree structure where each node represents a test on a feature and the resulting braches represent possible outcomes of this test, and each leaf node represents a class label. Determining the label of a test instance is the matter of tracking the path of nodes and branches to a terminating leaf.

Naïve Bayes (NB): is based on Bayes rule for inferring the posterior probability using prior class probabilities and the conditional probabilities (likelihood). It is literally called Naïve because it makes a naïve assumption that all features are independent from each other. However, despite this unrealistic assumption, the algorithm works well in most of the cases even if this assumption is violated.

Bayesian Network (BN): is a directed acyclic graph whose nodes represent features and edges represent their probabilistic relations. Each node contains a table for the conditional probabilities of its representing feature given the outcomes of the parent node. Every node is assumed to be dependent only on its immediate parent node. BN may outperform the NB algorithm if the conditional independence assumption between the features in the NB algorithm is violated.

Support Vector Machine (SVM): constructs the optimal separating hyperplane, which maximises its distance to the closest example, from any class. It leads to a maximum-margin separation between the classes. In two-dimensional space, the hyperplane is reduced to a line dividing the plane in two parts where the examples of each class lay in either side.

k-Nearest Neighbour (k-NN): computes the Euclidian distance between a new test example and the k nearest examples from previously classified examples, in the n-dimensional feature space, and assigns the test tuple the majority label of these k nearest neighbours. We use k=1, which means that we assign a new test example to the class of its nearest neighbour example amongst the previously classified examples. Unlike other training examples which normally include a computationally expensive training phase and simple calculation for the test phase, the k-NN algorithm essentially requires no training phase and the test phase is computationally expensive.

6 Performance Evaluation

For performance evaluation, we conduct three experiments. The first one studies the performance of different learning algorithms using all 22 original raw features. The second experiment, examines the performance of the same algorithms when we apply the Principal Component Analysis (PCA) technique with 95% variance coverage, which reduces the number of features down to 13. Finally, the last experiment investigates the impact of the variance retained by the PCA algorithm on the accuracy of a learning algorithm.

6.1 Experiment 1: training and testing with all original features

Our training dataset contains 448,724 examples from network traffic generated by 39 applications, including HTTP, DHCP, DNS, NTP, Skype, SNMP, etc. Each example is composed of 23 traffic attributes listed above in Table 1. Note that, in this table, the first attribute, in fact, indicates the output label, which is the application protocol. Hence, the training set actually contains 22 training features. For the purpose of simulations, we conduct several experiments. For the first experiment, we train the six abovementioned classification algorithms (OneR, DT, NB, BN, SVM, k-NN with k=1) and test them using 10-fold cross validation method. In this experiment, we do not apply any feature selection algorithm and perform the training phase using all 22 original features. Table 3 summarizes the results. Here, Accuracy means the ratio of test examples that are correctly classified. We observe that DT and k-NN algorithms outperform the others, achieving an accuracy of up to 0.994, which means that among 448,724 test examples, these algorithms successfully classify 99.4% of them and commit mistakes in only 6% of them.

Table 3. Results for Experiment 1, training with all 22 original features.

Algorithm	Accuracy	Precision	Recall	F-Measure
OneR	0.904	0.914	0.904	0.896
DT	**0.994**	**0.993**	**0.994**	**0.993**
NB	0.391	0.535	0.391	0.404
BN	0.969	0.976	0.969	0.972
SVM	0.890	0.881	0.890	0.878
k-NN	**0.994**	**0.994**	**0.994**	**0.994**

Table 4. Results for experiment 2, employing PCA with 95% variance retain and training with 13 selected features.

Algorithm	Accuracy	Precision	Recall	F-Measure
OneR	0.972	0.971	0.972	0.972
DT	**0.993**	**0.992**	**0.993**	**0.993**
NB	0.446	0.511	0.447	0.442
BN	0.974	0.989	0.974	0.980
SVM	0.795	0.771	0.795	0.755
k-NN	**0.994**	**0.994**	**0.994**	**0.994**

6.2 Experiment 2: applying PCA with 95% variance retain

Applying PCA, with 95% variance retain, considerably reduces the number of features from 22 down to 13. In order to assess the impact of this remarkable dimensionality reduction on the classification performance, we conducted the second experiment, where we first applied PCA algorithm and then again trained and tested the same ML algorithms employed in the first experiment. Table 4 summarizes the results. We observe that applying PCA algorithm with 95% variance retain, surprisingly boosts the overall performance of learning algorithms. This is due to the fact that some learning algorithms such as NB are quite sensitive to the violation of the underlying assumption about the independence of all training features one from another. Applying PCA, provided that the variance retain is high enough, can help extract a set of independent but informative features out of the original ones. In particular, DT and k-NN still outperformed other algorithms and applying PCA has no impact on their performance. Furthermore, the PCA algorithm considerably improves the performance of OneR and the Bayesian classifiers (NB and BN) and only slightly deteriorates the performance of

SVM algorithm. It is worthwhile to stress that the combination of OneR and PCA algorithms results in a quite promising performance, considering its pretty simple decision rule for classification.

6.3 Experiment 3: feature selection with PCA

In this experiment, we apply PCA algorithm choosing different values for the covered variance, ranging from 95% down to 10%. As summarized by Table 5, the number of remaining attributes after applying PCA algorithms is 13 for 95% variance retaining and only one attribute for 10% variance coverage. We also incorporated DT algorithm for classifying the output instances of the PCA algorithm. The results are presented by the last column of this table. An interesting finding is that reducing the variance retain down to 70% reduces the number of remaining attributes considerably, resulting in only 6 attributes, out of 22 original ones, yet the sacrifice in the accuracy of the learning algorithm is negligible, only 0.01%. Furthermore, choosing variance retain of 50% leads to elimination of two additional features while reducing the accuracy marginally, 0.16% comparing to the case with 70% variance retain. Finally, applying PCA with only 10% variance coverage results in only one remaining feature while witnessing a minor reduction in the classification accuracy, only 3.65% additional sacrifice relative to the case with 50% variance coverage. This highlights the importance of performing feature selection before implementing any ML algorithm for traffic classification. For example, the output of PCA algorithm with 10% variance retain is the following feature, where the parameters are the abbreviations of attributes listed by Table 1: maximum forward packet length (*maxfpktl*), maximum backward packet length (*maxbpktl*), standard deviation of backward packet length (*stdbpktl*), standard *deviation* of forward packet length (*stdfpktl*), average backward packet length (*meanbpktl*).

$$0437maxfpktl + 0.43\ maxbpktl + 0.425stdbpktl + 0.413stdfpktl \\ + 0.292meanbpkt \ldots$$

Table 5. Results for experiment 3, feature selection with PCA with different variance retains.

Variance Retain	Number of Remaining Attributes	Accuracy of DT Algorithm (%)
0.95	13	99.36
0.90	11	99.35
0.80	8	99.35
0.70	**6**	**99.35**
0.50	4	99.19
0.20	2	98.97
0.10	1	95.56

7 Conclusion

Security of CPPSs is a mounting concern in Industry 4.0, where IIoT and MTC technologies are massively employed to connect all stakeholder, including man, product, and machine, across the whole supply chain. In spite of this revolution, still, there is considerable legacy equipment in factories that cannot be replaced immediately, all at once, due to excessive capital expenditure and typically long lifespan of the machineries. To address this concern, network segregation seems essential to divide the network into different segments, based on the communications needs, to control the access to machines sitting in each segment, and to restrict unnecessary inter-segment communications. To this end, machine learning is a promising tool to classify network endpoints based on their communication patterns. In this paper we applied ML and traffic classification to group network endpoint in Industry 4.0 networks. We analysed the data collected from a large segment of Infineon’s network, within the realization of ECSEL research project SemI40. Using DPI tools, we constructed labelled datasets with 22 traffic features and applied several supervised algorithms. The results reveal that DT and k-NN demonstrate outstanding performance, with an accuracy reaching up to 99.4%. Moreover, applying PCA algorithm can reduce the number of feature remarkably from 22 features down to only six features, with a negligible loss of 0.05% in the accuracy of the learning algorithm. This work can be extended in the following directions. First, determining different groups of network endpoint in a CPPS, based on the traffic flows that they generate, is an important research topic. Second, integrating the proposed grouping intelligence into a conventional network device, e.g., Firewall, is another critical research problem.

8 Acknowledgment

The authors would like to thank Infineon Technologies, especially Christian Zechner and Stephan Spittaler for their great support in data acquisition and identifying the addressed challenges. It is also acknowledged that this work has been developed within Power Semiconductor and Electronics Manufacturing 4.0 (SemI40) project, under grant agreement No. 692466, co-funded by grants from Austria, Germany, Italy, France, Portugal (through Fundação para a Ciência e Tecnologia ECSEL/0009/2015) and Electronic Component Systems for European Leadership Joint Undertaking (ECSEL JU).

References

1. Esfahani, A., Mantas, G., Matischek, R., Saghezchi, F.B., Rodriguez, J., Bicaku, A., Maksuti, S., Tauber, M., Schmittner, C. and Bastos, J.: A lightweight authentication mechanism for M2M communications in industrial IoT environment. IEEE Internet of Things Journal (2017).
2. Finsterbusch, M., Richter, C., Rocha, E., Muller, JA., Hanssgen, K.: A survey of payload-based traffic classification approaches. IEEE Communications Surveys & Tutorials 16(2), pp. 1135–56 (2014).

3. Shi, H., Li, H., Zhang, D., Cheng, C., Cao, X.: An Efficient Feature Generation Approach based on Deep Learning and Feature Selection Techniques for traffic classification. Computer Networks 132, pp. 81–98 (2018).
4. Zhang, J., Chen, C., Xiang, Y., Zhou, W., Xiang, Y.: Internet traffic classification by aggregating correlated naive Bayes predictions. IEEE Transactions on Information Forensics and Security. 8(1), pp, 5–15 (2013).
5. Valenti, S., Rossi, D., Dainotti, A., Pescapè, A., Finamore, A., Mellia, M.: Reviewing traffic classification. In: Biersack, E., Callegari, C., Matijasevic, M. (eds.) Data Traffic Monitoring and Analysis, pp. 123–147. Springer, Berlin, Heidelberg (2013).
6. Zhang, J., Chen, X., Xiang, Y., Zhou, W., Wu, J.: Robust network traffic classification. IEEE/ACM Transactions on Networking (TON) 23(4), pp. 1257–1270 (2015).
7. Zhang, J., Chen, C., Xiang, Y., Zhou, W., Xiang, Y.: Internet traffic classification by aggregating correlated naive Bayes predictions. IEEE Transactions on Information Forensics and Security 8(1), pp. 5-15 (2013).
8. Kim, H., Claffy, KC., Fomenkov, M., Barman, D., Faloutsos, M., Lee, K.: Internet traffic classification demystified: myths, caveats, and the best practices. In: Proceedings of the 2008 ACM CoNEXT conference, pp. 11:1–11:12, ACM (2008).
9. Williams, N., Zander, S., Armitage, G.: A preliminary performance comparison of five machine learning algorithms for practical IP traffic flow classification. ACM SIGCOMM Computer Communication Review 36(5), pp. 7–15 (2006).
10. Nguyen, TT., Armitage, G.: A survey of techniques for internet traffic classification using machine learning. IEEE Communications Surveys & Tutorials. 10(4), pp. 56–76 (2008).
11. McGregor, A., Hall, M., Lorier, P., Brunskill, J.: Flow clustering using machine learning techniques. In: International Workshop on Passive and Active Network Measurement, pp. 205–214. Springer, Berlin, Heidelberg (2004).
12. Erman, J., Arlitt, M., Mahanti, A.: Traffic classification using clustering algorithms. In: Proceedings of the SIGCOMM workshop on Mining network data, pp. 281-286. ACM (2006).
13. Moore, AW., Zuev, D.: Internet traffic classification using Bayesian analysis techniques. In: ACM SIGMETRICS Performance Evaluation Review 33(1), pp. 50–60. ACM (2005).