Dataset Analysis for Anomaly Detection on Critical Infrastructures

German Lopez-Civera and Enrique de la Hoz
Computer Engineering Department, University of Alcala, Edificio Politécnico, 28805 Alcalá de Henares, Madrid, Spain

Keywords: Intrusion Detection, Dataset evaluation, Machine Learning, Decision Tree.

Abstract: Anomaly Detection techniques allow to create robust security measures that provides early detection and are able to identify novel attacks that could not be prevented otherwise. Datasets represent a critical component in the process of designing and evaluating any kind of anomaly detection method. For this reason, in this paper we present the evaluation of two datasets showing the dependencies that arise between the techniques employed and the dataset itself. We also describe the characteristics that have to be taken into account while selecting a dataset to evaluate a detection algorithm in a critical infrastructure context.

1 INTRODUCTION

During the last few years the dependency of our society tot the organizations and infrastructures that supports the services that provides the most basic has risen to a level that has force all countries to take measures to protect this kind of facilities. The protection of the systems that allow this organizations to continue working is a requirement that no government can ignore (Rinaldi et al., 2001). The impact that an attack on this kind of institutions could have on the economy, security and health of our society demands adequate responses and a joint effort of government institutions and private companies that operate the facilities that provides these services.

One of the most important assets inside these facilities is the telecommunication network that supports the processes that are performed in a critical infrastructure. The disruption of the communications in a critical infrastructure could have multiple cascading effects that can end up with the complete infrastructure compromised by an attacker. The early detection of attacks at this kind of organizations is a procedure that must be implemented at each layer of its core modules.

Intrusion detection and prevention systems (IPS/IDS) are the most used tool to deploy this kind of detection on real infrastructures. This kind of system can be classified in two categories depending on the method they use to detect the attacks. Misuse detection techniques are based on processing network traffic data looking for known malicious behaviour. Every packet / flow is compared with known malicious patterns that are continuously updated by vendors and experts. While most of the commercial options are mainly based on Misuse detection methods, anomaly detection techniques provide many advantages versus novel attacks.

Anomaly detection tries to find deviations from the normal behaviour. These irregularities can show early stages of attacks and also some kind malfunctions due to hardware or software errors. As these techniques do not use any kind of signature that describes the attacks, they can adapt their behaviour to different kind of topologies and applications. They are also able to detect attacks that have not a signature yet to detect them, being helpful during the initial phase after a vulnerability discovery.

Network anomaly detection is a research field on itself and the interest on it has raised during the last years among the research community. The proliferation of machine learning tools that help to test and evaluate the performance of different algorithms against network data samples have boost the research on this field. As a consequence of this growth, network traffic datasets that contains normal and malicious behaviour are highly demanded. These datasets have to comply with different requirements to become an appropriate alternative to evaluate anomaly detection algorithms. In the following section we will describe the main characteristics that have to be evaluated before choosing a dataset.

Lopez-Civera, G. and Hoz, E.
Dataset Analysis for Anomaly Detection on Critical Infrastructures.
DOI: 10.5220/0006017701510158
In *Proceedings of the 13th International Joint Conference on e-Business and Telecommunications (ICETE 2016) - Volume 1: DCNET*, pages 151-158
ISBN: 978-989-758-196-0

2 DATASET CHARACTERIZATION

As the tool used to evaluate the performance of attack detection techniques, datasets must include a wide variety of data inside them. This data should be composed by both, normal and anomalous samples. This way, the evaluation can provide meaningful metrics that show how the algorithm performs against different environments. A good anomaly detection method should not only detect most of the malicious behaviour (i.e. a good True positive rate (TPR)) but also should not confuse normal behaviour with malicious one (i.e. a high False positive rate (FPR)). The dataset composition can help to detect bad performance from both points of view, but it is crucial to evaluate their content before trusting the results they can output.

KDD99 dataset is the most used dataset in the academic field as it has become a standard evaluation benchmark. Despite this fact, this dataset is highly outdated and does not represent the current threats that can face a critical infrastructure. Moreover several works (Brown et al., 2009) (McHugh, 2000) have highlighted some deficiencies in this dataset that can bias the results of the algorithms applied to it. This fact arise the issue of searching another dataset that meets the requirements of our scenario and that can be used as a benchmark for anomaly detection algorithms.

The process of creating a dataset that provides a realistic scenario, while providing as much data as possible and preventing biased information means a significant challenge (Shiravi et al., 2012). In this section we will briefly describe the main characteristics that we have found more important during our work with different kinds of datasets and the impact they can have on algorithm performance evaluation.

- **Generation Method:** Dataset generation can be either synthetic or real capture based. Real capture datasets are built by employing real traffic collected from a real institution like a university, a research facility or a private organization. On the other hand synthetic datasets are manually created by injecting malicious traffic into normal traffic samples. These normal can also be synthetically generated or be part of a real traffic capture. Real capture datasets are inherently better as they model real network behaviour and therefore can offer the most realistic information about the actual characteristic of an attack.

  Moreover the normal part of the dataset show the real use of the network without needing to model it via any kind of traffic generation pattern. Despite of all these advantages this kind of datasets are really hard to find due to the complexity of capturing real attacks and the privacy issues that can arise from publicly share network traffic of an organization. The method employed to generate the dataset has to be taken into account when translating the results of the performance metrics into actual conclusions.

- **Network Data Format:** The format in which the dataset is presented determines the quantity of information that is offered by it. As the data represented is network traffic, the format are mainly based on different standardized network traffic representations. The traffic can be offered raw or after performing some level of aggregation. For sharing raw network traffic, PCAP format is the most used one. It is a standardized format that contains a direct copy of the traffic that travels through a network, therefore it is a way to avoid losing any kind of information when sharing network traffic data. The main disadvantage is the size of the data (i.e. It takes up the same size as the actual data collected from the network) and as a no-loose format, the privacy issues of sharing a raw copy of the data. As a consequence real data capture of a critical infrastructure is extremely hard to find as it would represent a huge threat for the organization itself.

  As opposed to PCAP , Netflow-like formats offer a summarized view of the traffic collected in the dataset. Their information unit is the traffic flow, that is, a sequence of messages exchanged between two network nodes. Each one of these flows could be composed by different traffic packets but it is summarized as a single flow and characterized by its duration, size, number of packets, etc. This kind of formats solve some privacy and size issues while retaining most of the core behaviour of the network and as a consequence are widely employed for dataset generation.

- **Anonymization Level:** To solve the privacy issues mentioned above different techniques are employed to reduce the amount of private information provided in the dataset. These techniques try to preserve most of the actual behaviour of the network, so attacks can still be detected and distinguished from normal traffic. The most basic anonymization method is the aggregation offered by the format itself. As we mentioned in the previous characteristic, if the dataset is offered in a flow summarized format, the payload of the packets is removed. There exist datasets in raw format that offer Pcap files without the data payload of the packets. Both techniques prevent leaking the data that travelled through the network but the actual origin and destination of the communication is still present in the dataset. As the actual IP addresses or timestamps could allow to infer the context and intention of the communication further anonymization methods are usually performed (Coull et al., 2009). These techniques include the truncation of the data fields (e.g remove the last octet of every IP address), regenerate data fields based on the statistical model that best fit them, replace IP addresses by pseudonyms or quantization fields like the communication timestamps. Each one of these techniques affect in a different way the dataset and a balance between privacy and utility must be reached during the design phase of the dataset.

- **Attack Diversity:** The development of new attack techniques and the sustained growth in the complexity of threats forces to continuously adapt and update the detection methods employed. As a consequence when developing a dataset a wide variety of attacks must be included. If we only focus on one kind of attacks, the algorithms evaluated with them can have a good performance, while completely ignoring other kind of threats, leading to inconsistent results. However we are aware that an actual compilation of all kind of attacks is not feasible, so a good selection must be performed. It should include attacks that covers all layers of the communication network, from TCP/IP flood techniques to Web application attacks like SQL injection or Cross-site Scripting. In the case of critical infrastructure particular attention is needed to model Denial of Service (DoS) attacks and application specific attacks to provide a heterogeneous environment for the detection methods evaluated.
- **Normal/Malicious Distribution:** Depending on the composition of the dataset and the ratio between normal and malicious samples the dataset may lead to confusing results and to overtraining the algorithms. Datasets are usually biased due to the fact that normal traffic is much more frequent than malicious. Despite that we can say that most of the time the network will behave following this principle an attacker could try to abuse this assumptions by mimic his attack into normal traffic patterns. For example, DoS attacks drastically change the dynamic of the network traffic and can be launched in different phases that make the traffic to evolve step by step, while attempting to avoid statistical detection. Therefore a good balance between both kinds of samples must be reached. Very few malicious samples could lead to a bad model of attack behaviour but too much attack samples can also increase the false positive rate.
- **Labelling Process:** Depending on the algorithm employed to detect the anomalies a labelled dataset may be a requirement to run the algorithm itself (i.e. Supervised machine learning algorithms) but independently of the type of algorithm employed the labels are the only way to actually evaluate the accuracy of a detection method. Without labels the dataset can only be used as reference data but not as an evaluation benchmark. The labelling process is complex and highly dependent on how the dataset is generated.

  While manually labelling the dataset is the most accurate method it is usually infeasible due the size of the data and the difficult that involves detecting attacks on the data collected. Synthetic dataset are easier to label, often the dataset publishers know the source of the attacks (they have launched them) and they can automatize the labelling process. In other cases where this manual process is not viable, the dataset is labelled according to the output of multiple anomaly detection tools that are already evaluated in the past. This output offer a relative level of confidence and can help to label massive datasets that could not be label otherwise. The quality and completeness of the dataset labelling will have an impact on the evaluation and is one of the most critical aspect while selecting a dataset for our purposes.

3 DATASET EVALUATION

Following the principles and characteristic described in the previous section we have selected two datasets to test machine learning techniques against them and evaluate their performance. The datasets chosen are TORPEDA (Torrano-Gimenez et al.,) and CTU-13.

3.1 TORPEDA and CTU13 Datasets

TORPEDA is a synthetic labelled dataset shared in XML format. It is composed by HTTP requests made against a custom vulnerable web application. The main objective of the creators was to provide a standardized dataset that helps to evaluate the performance of Web Application Firewall products. The actual data it contains includes the HTTP method, all HTTP headers and the path part if the URL.

CTU-13 is also a synthetic dataset focused on Botnet network traffic but also contains a wide range of attack types including DDoS or port scanning. The dataset is composed by thirteen samples. In each sample a different botnet malware is deployed in a controlled environment. The dataset comes in two formats: a labelled bidirectional Netflow file and a full Pcap of all malicious packets. Recently they have added Pcap samples with both, normal and malicious packets but removing the data payload to prevent privacy issues. Among these thirteen samples we can found different combinations of attacks and different normal/malicious ratios. The combination of these two formats and the diversity shown in the thirteen samples makes CTU 13 a multi-purpose dataset that can fit many research requirements.

3.2 Classification Algorithms

During our research on anomaly detection datasets we have used multiple algorithms to evaluate their performance against different datasets. Depending on the type of anomalies we are trying to find there exist different approaches to solve the problem. We can organize anomalies as volume-based or content-based anomalies. Volume-based anomalies are the ones that produce a change in the quantity or frequency of information exchanged in the network. Examples of this kind of anomalies are DoS attacks or Port scanning. On the other hand content-based anomalies focus on detecting attacks in the payload of the network traffic and are able to detect application specific attacks like SQL injection.

Both approaches have their own strengths and weaknesses. Content-based anomaly detection can offer better results against application layer attacks but as they must inspect the payload of the traffic, depending on the volume of information that travels through the network, the computational cost could make them impossible to use. In contrast volume-based anomaly detection have less computational requirements when analysing the same amount of traffic. It is a more attack-agnostic method as they do not try to find specific characteristic for each type of attack, instead they look for the consequences the attack have on the whole behaviour of the network.

In this paper we will employ machine learning algorithms to try to detect both types of anomalies inside both chosen datasets. Among all types of machine learning methods anomaly detection can be seen as a subcategory of classification problems. Classification methods aim to create rules based on the data provided to them that helps to classify the data into a set of categories. Once these rules are established the algorithm is used to predict the category where new data samples best fit. The performance of this kind of algorithms is measured based on the accuracy they have when predicting the categories of the samples in a test benchmark where the categories are already known.

To develop this analysis we have employed Python programming language with Scikit-learn (Pedregosa et al., 2011) and Matplotlib (Hunter, 2007) libraries that helps to quickly evaluate different types of machine learning algorithms and easily manage the datasets. Among all available classification algorithms we have chosen Decision Trees. This kind of algorithm is a well-known method to perform recursive partition on the data it is applied to. The decision tree consist on a set of nodes (with a root node as origin), at each one of them the data is split in different classes depending on the feature and threshold evaluated at it. The most frequent approach is to ask one single question to the data coming to the node, and depending on the answer, the data is divided into two different sub-spaces. This process is repeated recursively until the tree reaches particular depth or until the leaf nodes contain less samples than an specific threshold. Each one of this leaf nodes indicates the predicted class (or the probability of being on it) of the data that is inside it. Despite the fact it has not the best accuracy of all tested algorithms, it is one of the few algorithms that allows to see its inner working through decision tree graphical representations.

The majority of classification algorithms can be customized through different parameters to adjust different kinds of thresholds but decision trees also allow to see which part of the data is most important to guide the choices the algorithm made through its different steps. This feature will help us to identify outliers that could be leading to bad decisions allowing us to adapt the dataset if needed. This way we can quickly diagnose overfitting problems. Overfitting arise when the algorithm does not generalize enough the underlying relationship in the data. This problem leads to poor prediction performance as the model will overreact to changes and therefore it must be properly addressed.

3.3 Data Preprocessing

Prior to launch this kind of algorithms a preliminary data preprocessing is needed to model each dataset as a set of features that can be input to the anomaly detection process.

Both selected datasets needed different kinds of preprocessing. TORPEDA comes in XML format and the information is represented as strings. A typical approach is to deal with the dataset as it is a normal text, splitting it into sequences of tokens of different sizes called n-grams. After that the n-grams are processed to count the frequency of appearance in the n-gram corpus, generating statistics like term frequencyinverse document frequency (Tf-idf) that help the anomaly detection algorithm to process the dataset, as they usually only understand numeric data.

CTU-13 dataset labelled flow format comes directly in a CSV-like format that can be easily loaded by the analysis tools. Nevertheless due to the size and heterogeneity of the features that comes with it, some design decisions must be carefully taken prior to launch the anomaly detection phase. The first decision was to remove timestamps and IP addresses from the data that is actually sent to the anomaly detection algorithm. The reason was to avoid overfitting problems due to detecting specific IP addresses as malicious instead of looking for the actual attack characteristics. The timestamps was removed afterwards after noticing it adds noise during the detection phase and did not help to improve the accuracy. The final set of features employed was:

- Dur: The total time employed by each flow to send the information.
- Proto: Protocol used (i.e. TCP, UDP, ICMP)
- Sport, DPort: Source and destination port.
- TotPkts, TotBytes: The total amount of packets and bytes exchanged in the flow.

Duration and source/destination ports were also converted to numeric format as its order can be meaningful to detect attack patterns.

After selecting good features two aggregation processes were applied to the dataset aiming to reduce the computational cost. First, the dataset was re-sampled to five minutes slots and then only the thirty percent of each feature that retains the highest variance is saved for processing.

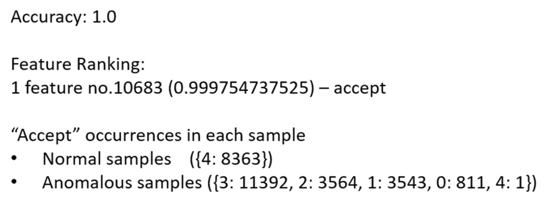
4 RESULTS

In this section we will show the results we obtained after launching several experiments with decision tree algorithms against the data coming from TORPEDA and CTU-13 datasets.

The first dataset we analysed was TORPEDA. The dataset was split in training and test sequences that allow to evaluate the performance of the decision tree classifier. With this set-up we obtained a 99.99% Accuracy score that made us suspect of suffering overfitting issues. To find the source of this behaviour we explored the internal data structures employed by the decision tree classifier, to see which tokens had more weight when guiding the decisions made by the classifier. At this point we realize that some tokens had the most part of the weight. These tokens were completely unrelated with attacks, but with specific characteristics of the context where the malicious requests were made. Some types of these tokens are detailed below:

- The Web application session ID assigned to the attacker during the attack generation is present in every malicious request.
- Some HTTP Header parameters like ”gzip”, ”deflate” only appears on attack samples.
- The User-Agent employed by the attacker was also constant on most of the requests.
- Some application specific parameters that were more frequent in malicious requests than in normal traffic, due to its corresponding endpoint was attacked more times than present in benign traffic.

To try to solve this issue we started removing this tokens from the data fed to the algorithm. After each experiment new tokens that biased the result were found. The most representative tokens of this issue were ”sqlmap” and ”Accept”. The first one was included in some attacker request and it had the 100% of the decision weight before removing it. Despite every request that includes this token in the User-Agent header should be labelled as suspicious, an attacker can easily change this header and he would bypass the detection of our system. The second token was harder to diagnose as all requests included at least one ”Accept” HTTP header. As we can observe in Figure 1 the accuracy was still biased and the token had 99.97% of the decision weight. But as shown in the figure, the underlying reason was that all normal samples include four different accept headers, while the majority of malicious ones contains less than four.


```
Accuracy: 1.0

Feature Ranking:
1 feature no.10683 (0.999754737525) – accept

"Accept" occurrences in each sample
•    Normal samples    ({4: 8363})
•    Anomalous samples ({3: 11392, 2: 3564, 1: 3543, 0: 811, 4: 1})
```

Figure 1: TORPEDA dataset result metrics.

The distribution of the *Accept* occurrences in the requests was biasing the results and forcing the decision tree to make bad choices. Even after combining multiple decision tree classifiers in a random forest or randomizing the dataset split in training/test samples, the output was the same.This kind of issues illustrates the complexity inherent in the task of generating a dataset. Little details can produce a huge impact in detection rate if not properly identified.

To compare this results we chose CTU-13 dataset as a good benchmark. It provides a much bigger data sample and different kinds of features that will help us to measure the impact they can have applying the same algorithms.

Before launching experiments with the classifier algorithm we found interesting to plot some features of the dataset trying to find specific patterns that could help us to guide the decision tree. To illustrate this process we selected Duration and Total Packets features. In Figure 2 we can easily identify a pattern in which the plot of these two features clearly cluster the malicious behaviour in three different sets on sample 10 of CTU-13. This result may mislead our research if we do not double check them against other samples. As we can see in Figure 3 the same plot made with sample 9 of the dataset behaves really different. No cluster can be identified at clear sight and the malicious flows are evenly distributed among the most frequent points of the plot. Therefore we can conclude that despite this kind of manual analysis can help in some cases to identify clear patterns in the datasets, it must be carefully performed as it can mislead the decisions we made while choosing which features to use in the analysis.

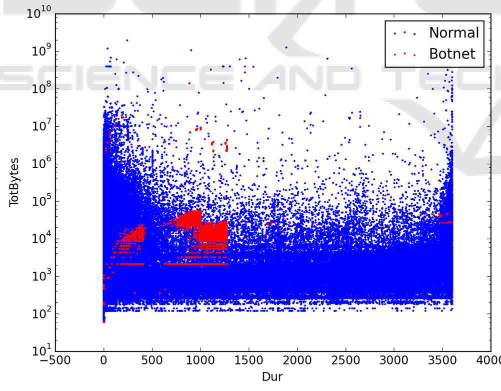


Figure 2: Duration vs Total Packets in sample 10 of CTU-13 dataset.

Finally after preparing CTU-13 dataset for the analysis, we launched a random forest classifier to compare the results with TORPEDA dataset. As we can see in Figure 4 in this case we also obtained a high accuracy score of 0.98 and the weight ranking was less biased. Nevertheless the decision was still dominated by source and destination port features.

Despite of the good results obtained such high accuracy metrics should be double-checked, therefore our next step was to analyse the decisions that the tree was making. To do so, we make use of the features

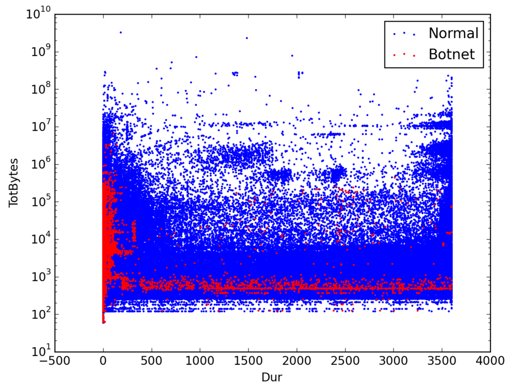


Figure 3: Duration vs Total Packets in sample 9 of CTU-13 dataset.

```
Accuracy: 0.98419742583

Feature Ranking:
1 feature no.1 (0.575403961314) - Sport
2 feature no.2 (0.37743475438) - Dport
3 feature no.0 (0.0232711252271) - Dur
4 feature no.18 (0.0172043689734) - tcp
5 feature no.4 (0.00668579010523) - TotBytes
```

Figure 4: CTU-13 dataset result metrics.

that Scikit-learn library provides to graph the decision tree. In Figure 5 we can see the result of this analysis. In that graph if the condition showed on top of each node is met, then go to the child node on the left. The amount of samples that goes on each direction is shown at the bottom of each node.

If we examine the resulting tree we can see that the first check that the tree makes to each data sample is that if the source port is smaller or bigger than 5000. With that single test, 83% of the dataset is already classified as benign (i.e. all samples on the left of the value array) as the right child node is a leaf node. Despite the fact that small source port values would be less likely malicious as they are usually allocated for specific applications and may need high privileges to use them, a small source port cannot be directly linked to benign traffic as any attacker can avoid this filter by customizing its attack source port.

Further analysis of the decision tree shows that most of the nodes tend to look for application specific destination port (like destination port smaller than 54.5 for DNS protocol). Moreover generic features like Duration, Total Packets or Total size have very little weight in the decision process.

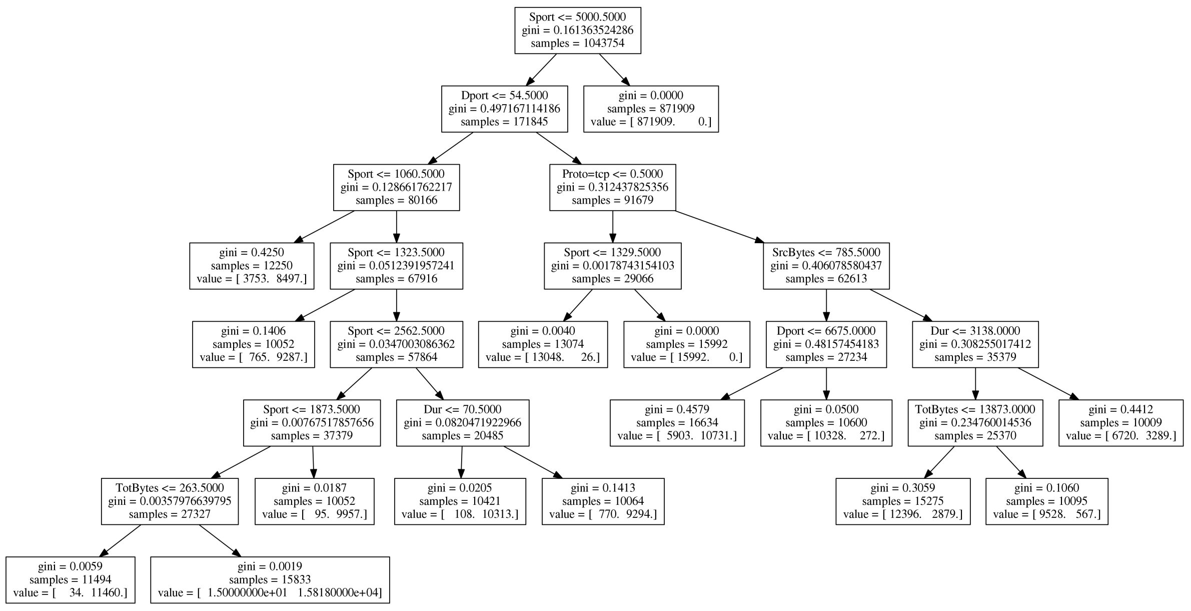


Figure 5: Decision tree computed over CTU-13 dataset.

5 CONCLUSIONS

Examining the results obtained with these two datasets we can conclude that a direct process over the dataset data even if we properly adapt it to reduce the amount of overfitting issues is not an adequate approach. Classifier algorithms have shown a good behaviour during the evaluation of the datasets and has helped us to identify features that can mislead the anomaly detection process.

Nevertheless using only classifier algorithms to perform actual anomaly detection to raw datasets has been shown as an unsuccessful approach.

In future analysis we would like to explore the impact of generating new features via feature engineering techniques. This will allow to feed the detection algorithms with features that offer more discriminating power and that help them to infer the inner structure of the data and detect the underlying anomalies.

Further analysis must be performed against bigger datasets that try to combine multiple types of datasets (Bhuyan et al., 2015), offering a wider variety that will prevent the overfitting problem or at least make it less likely to happen. We also consider that hybrid approaches that combine different kinds of detection techniques applied to flow and raw capture formats will be more resistant to overfitting and produce better predictions.

ACKNOWLEDGEMENTS

This work was partially supported by SCOUT, a research project supported by the European Commission under its 7th Framework Program (contract-no. 607019).

REFERENCES

Bhuyan, M. H., Bhattacharyya, D. K., and Kalita, J. K. (2015). Towards generating real-life datasets for network intrusion detection. *I. J. Network Security*, 17(6):683–701.

Brown, C., Cowperthwaite, A., Hijazi, A., and Somayaji, A. (2009). Analysis of the 1999 darpa/lincoln laboratory ids evaluation data with netadhict. In *2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications*, pages 1–7.

Coull, S. E., Monrose, F., Reiter, M. K., and Bailey, M. (2009). The challenges of effectively anonymizing network data. In *Conference For Homeland Security, 2009. CATCH'09. Cybersecurity Applications & Technology*, pages 230–236. IEEE.

Hunter, J. D. (2007). Matplotlib: A 2d graphics environment. *Computing In Science & Engineering*, 9(3):90–95.

McHugh, J. (2000). The 1998 lincoln laboratory ids evaluation. In *Recent Advances in Intrusion Detection*, pages 145–161. Springer.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Rinaldi, S. M., Peerenboom, J. P., and Kelly, T. K. (2001). Identifying, understanding, and analyzing critical infrastructure interdependencies. *IEEE Control Systems*, 21(6):11–25.

Shiravi, A., Shiravi, H., Tavallaee, M., and Ghorbani, A. A. (2012). Toward developing a systematic approach to generate benchmark datasets for intrusion detection. *Comput. Secur.*, 31(3):357–374.

Torrano-Gimenez, C., Perez-Villegas, A., and Alvarez, G. Torpeda: Una especificacion abierta de conjuntos de datos para la evaluacion de cortafuegos de aplicaciones web.