A Pre-clustering Method To Improve Anomaly Detection

Denis Hock[1], Martin Kappes[1] and Bogdan Ghita[2]
[1]*Frankfurt University of Applied Sciences, Frankfurt am Main, Germany*
[2]*Plymouth University, Plymouth, U.K.*

Keywords: Computer Networks, Network Anomaly Detection, Clustering.

Abstract: While Anomaly Detection is commonly accepted as an appropriate technique to uncover yet unknown network misuse patterns and malware, detection rates are often diminished by, e.g., unpredictable user behavior, new applications and concept changes. In this paper, we propose and evaluate the benefits of using clustering methods for data preprocessing in Anomaly Detection in order to improve detection rates even in the presence of such events. We study our pre-clustering approach for different features such as IP addresses, traffic characteristics and application layer protocols. Our results obtained by analyzing detection rates for real network traffic with actual intrusions indicates that our approach does indeed significantly improve detection rates and, moreover, is practically feasible.

1 INTRODUCTION

IT-Systems and their underlying networks have pervaded business and private life and are ubiquitously used in virtually every area. The paramount economic and social impact raised the protection of IP-based networks and associated devices to an increasingly important matter. Lately, these systems are often subject of malicious software attacks that aim to cause damage to users and may affect the functions of devices and network. Network intrusions, defined by (Heady et al., 1990) as "any set of actions that attempt to compromise the integrity, confidentiality, or availability of a resource", can cause lost revenue, damage to the reputation or corrupt critical processes and sensitive data.

A countermeasure to detect these malicious actions in time are Intrusion Detection Systems (IDS). However, traditional IDS require extensive knowledge of attack signatures and are not capable of detecting new attacks, due to the time lag between publication and implementation of an attack signature. With noticeable rise in the number of threats and sophistication, the challenges and requirements of intrusion detection increase as well. These reasons advanced the interest in Anomaly Detection Systems, with the ability to discover even unknown and new attacks by analyzing statistical deviations from a defined normal behavior. But what sounds simple is rather complicated in practice: Statistical noise and inconsistency, attributed to the On/Off behavior of users, traffic peak times and the workings of underlying protocols (e.g. TCP Congestion Control) aggravate to separate between irregular patterns, noise and real outliers. Our aim is to reduce this noise by analyzing subsets of network traffic with more deterministic and predictable traffic. Clustering, the task of sub-dividing a data set into statistically similar groups, can be used to discover patterns, relationships, and structures in large amounts of data. In this paper, we use clustering as data preprocessing technique to reduce the amount of network traffic noise and simplify the detection of anomalous pattern by analyzing statistically related network traffic.

In the remainder, we present an overview of related literature and the theoretical advantages of pre-clustering. Subsequently, we study the feasibility of different metrics for efficiently clustering network traffic into suitable subsets. We provide a profound theoretical explanation and analyze the complexity, noise reduction and characteristics of the resulting subsets. Our experiment, consulting an Anomaly Detection System on malicious network traffic, shows indeed that clustering is well suited to preprocess network traffic and can uncover networks attacks which remain undetected when the identical Anomaly Detection method is applied to cumulative network traffic without clustering.

Hock, D., Kappes, M. and Ghita, B.
A Pre-clustering Method To Improve Anomaly Detection.
DOI: 10.5220/0005953003910396
In *Proceedings of the 13th International Joint Conference on e-Business and Telecommunications (ICETE 2016) - Volume 4: SECRYPT*, pages 391-396
ISBN: 978-989-758-196-0

2 RELATED WORK

Since Dorothy Denning published the initial paper on Anomaly Detection (Denning, 1987), a large variety of techniques based on machine learning, statistics and data mining have been published. Several publications (Leung and Leckie, 2005; Münz et al., 2007; Izakian and Pedrycz, 2014) utilize clustering-based methods to classify malicious network traffic.

Some early Anomaly Detection approaches, such as PHAD (Mahoney and Chan, 2001) and SnortAD (Szmit et al., 2012) use very simple methods to split network traffic into TCP, UDP and ICMP as part of the Anomaly Detection algorithm. But, they did not analyze the effect of the splitting on noise or utilize the split groups to improve their results.

(Bouzida et al., 2004) uses PCA as preprocessing to derive a new set of significant features as data preprocessing. Further common data preprocessing methods used for Anomaly Detection (e.g. selection, transformation and cleaning) have been reviewed by (Davis and Clark, 2011).

However, to the best of our knowledge, clustering has not been proposed as method to reduce noise and optimize thresholds during the data preprocessing, in previous works.

3 METHOD

In the following, we will briefly outline the rationale behind our approach. This section demonstrates the optimization of thresholds and impact of clustering on noise.

3.1 Advantages of Pre-clustering

In contrast to other approaches, Anomaly Detection does not depend on the existence of malicious examples. Instead, it aims to detect statistical deviations from normal network traffic, which results in the capability to detect previously unseen events. Anomaly Detection Systems can assign a certain probability p (or alternatively a numerical score) to each event indicating its abnormality, where p is generated by a function which compares the current traffic to reference values from historical data.

The binary result $r \in \{0,1\}$ (where 0 denotes normal and 1 abnormal) depends on a threshold θ.

$$r = \begin{cases} 1 & \text{if } p \text{ is} > \theta \\ 0 & \text{if } p \text{ is} \leq \theta \end{cases} \tag{1}$$

If an event was assigned a probability p above the threshold θ, but was actually legitimate traffic, we call the result a false positive. If an event was assigned a p below θ, but was actually unwanted traffic, we call the result a false negative.

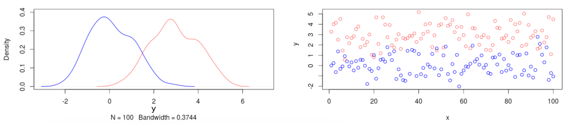

Figure 1: Classification problem on two normal distributions.

Figure 1 illustrates the concept of classification based on thresholds with an artificial sample. Our example includes two distinct set of data (light and dark), which are both normal distributed with $N = 100$. The right-hand side shows the density curves, the left-hand side a plot with dark dots ($\mu = 0, \sigma = 1$) that symbolize normal data and light dots are considered intrusive ($\mu = 3, \sigma = 1$). The area where both sets overlap (which is 7,3%) is a region where an algorithm can not reliably predict a value as intrusive or normal - we can only decide to favor false positives or false negatives by adapting the threshold.

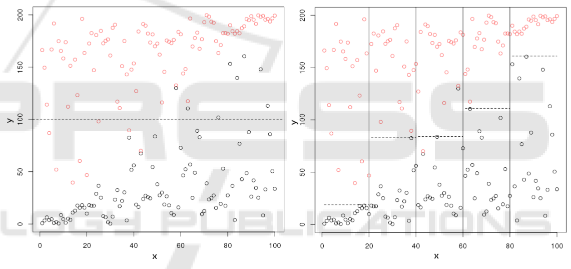

Figure 2: Classification problem on a heavy-tailed distribution.

Figure 2 presents a heavy-tailed distribution, as they occur on several network traffic features such as byte-size. The left-hand side shows the classification with a single threshold, the right-hand side uses several different thresholds to optimize the amount of false positives (FP) and negatives (FN) as well as the true positives (TP) and negatives (TN). Each range has been optimized using the threshold θ with maximal accuracy A.

$$A_{max} = \frac{TP + TN}{TP + FP + TN + FN} \tag{2}$$

where TP, FP, TN, and FN depend on θ.

While the figure is a simplified case for illustration and unlikely to occur in actual network traffic, we can transfer our approaches to the real world: If we cluster network traffic into classes with distinct groups of normal and malicious traffic, we can optimize each threshold using training data and outperform a cumulative threshold. Setting up different thresholds for

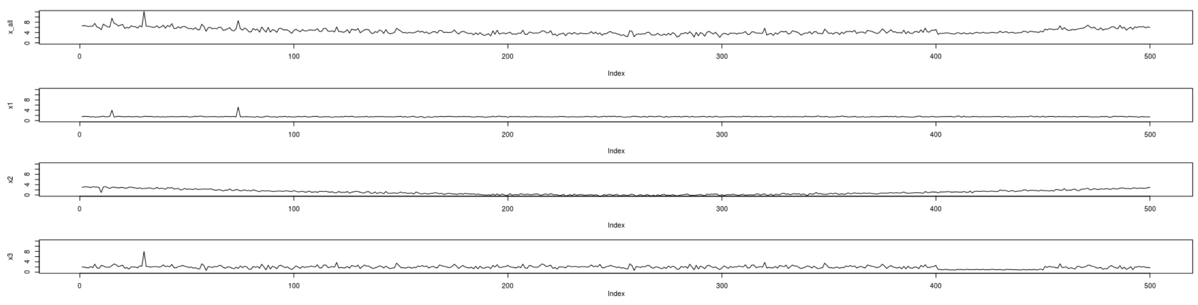

Figure 3: Time-series clustered and cumulative.

different clusters, as it is suggested here, might seem to lack in its scalability because setting up thresholds manually would be a tedious task. But using equation 2, it is obviously possible to automatize the finding and setting of those. Moreover, it is unlikely to have a vast amount of clusters, since the number of clusters heavily depend upon the limited number of network characteristics with the ability to split the traffic in a meaningful way.

Clustering applies statistical properties to rearrange related data into predefined smaller classes. Naturally, smaller classes also have a smaller standard deviation, which is noise. The optimal case would be statistically independent groups, when we let a and b denote any two curves where a and b are independent, then $var(a+b) = var(a) + var(b)$. A commonly used method to detects malicious actions is time-series analysis in combination with volume-based metrics such as the amount of bytes. Figure 3 shows an artificial time-series dataset with a x-axis ranging from 0 to 500. The three bottom graphs $(x1, x2, x3)$ show the clustered data, while the top graph $(xall)$ contains the cumulative data - we could archive such a setup e.g. by splitting the traffic using the layer 4 protocol. At first glance, the cumulative data is not only much more noisy, it also contain the trends of graphs $x1, x2$ and $x3$. To measure the impact of the reduced noise, we located four outliers on different positions of the x-axis when we generated the data: In graph $x1$ on point 15 and 74, in graph $x2$ on 10 and $x3$ on 30.

An outlier detection algorithm - following Chen and Liu (Chen and Liu, 1993) proposed procedure - showed, that it is much more difficult to reliably detect the injected outliers within the cumulative data. The result showed, that the algorithm returned one false negative and several false positives on the cumulative data, while archiving perfect results for each individual graph.

3.2 Metrics to Cluster Network Traffic

In the following, we propose three metrics to split network traffic and outline their properties: IP addresses, flow characteristics and a combination of Principle Component Analysis (PCA) and k-means clustering. The proposed metrics are only a few possible examples and their capability may vary for different anomaly detection methods and environments (e.g. splitting by IP addresses would fail if our network has only a single subnet).

The following examples have been achieved with the network traces over one day, including approximately one hundred thousand flows to the Internet and other networks, captured on a university network. The network consisted of several personal computers, servers and other network devices such as VoIP phones, printers and managed switches.

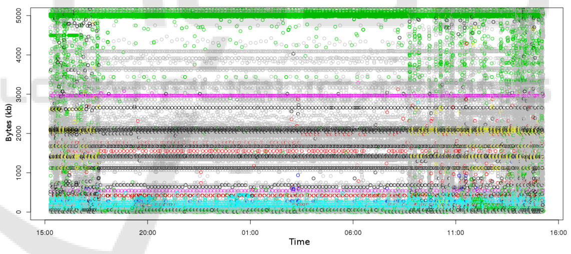

Figure 4: Byte size for each netflow.

Figure 4 displays the entire network traces, which are clustered later. Each dot depicts the byte size (y-axis) of a connection with different colors for each destination port (<1024). The figures aims to show the tremendous amount of variation, which prevents us to distinguish regular pattern from noise, detect deviations or find network pattern. The latter approaches try to split this traffic into smaller logical classes.

3.2.1 Splitting Traffic by IP Addresses

IP addresses can reveal a lot of information when used to compare the traffic of different subnets. We aim to depict subnets of the local area network with our clusters, because they may logically separate smaller networks used for server applications, virtual private networks or network administration.

Network traces do not contain any information on subnets, and there is a possibility that we have to many small subnets to reflect each in a single cluster. Therefore, we strongly simplified our operation by using the first 24 bit to determine the classes. In addition, we assume for our dataset, that the local area network is represented by the class with the highest number of participants.

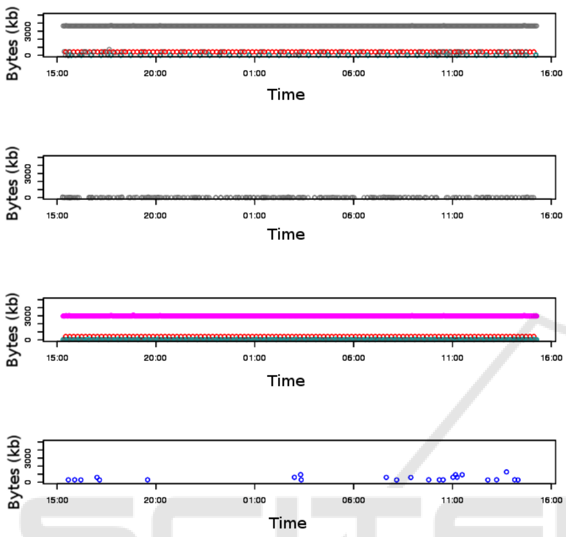

Figure 5: Byte size for the different subnets.

Figure 5 shows the first four classes we created. Each figure displays the transfered byte per flow for a class. The x axis show the time and the y axis the flow size in byte. The color visualizes the used destination port - all ports greater than 1024 are displayed in gray. Even if one of the subnets (not included in Figure 5) contains a major part of the complete traffic with eighty thousand flows, compared to the graph with all flows, we can see clear patterns in our classes.

3.2.2 Splitting Traffic by Flow Characteristics

Flow characteristics have been previously used to predict applications. Here, we split traffic using the flow duration, byte size and number of packets per flow. Figure 6 shows a 3D visualization of the mentioned metrics. The x axis shows the byte size from 0 - 6000 byte, the y axis the amount of packets from 0 to 30 packets and the z axis the duration from 0 to 1 second. A small amount of flows outside these ranges (with extreme values such as 100.000 bytes) were removed as outliers.

The color for each dot symbolizes the destination port as in the previous diagrams, the cross in the diagram shows our 8 different classes. Analyzing the classes, we concluded, that the characteristics split

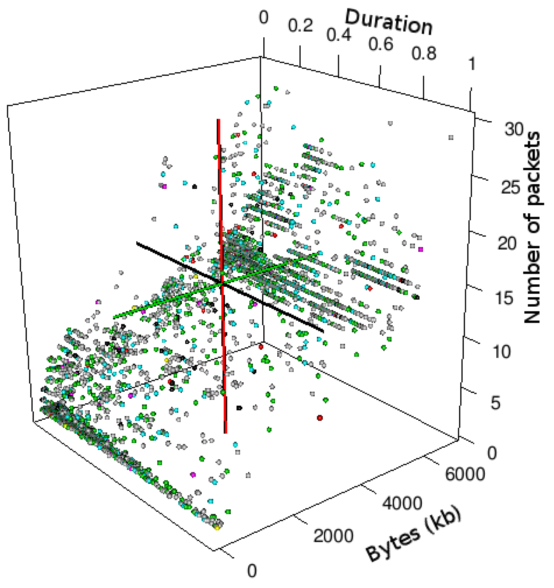

Figure 6: Byte size vs. packets vs. duration.

our flows into constant network activity, which are most likely network services, and user dependent activity which occured intensified on certain times of the day.

3.2.3 Splitting Traffic by Application Layer Protocols

Our last method utilizes Principle Component Analysis (PCA) and a k-means clustering, which delegates all logic entirely to a machine learning algorithm and aims to conclude for a human obscure higher-level information. PCA is a procedure which aims to create a set of linearly uncorrelated variables, defined by the largest possible variance. The input data used for our PCA model are the layer 4 protocol, destination port, number of packets and byte size for each flow. We derived four principle components and generate eight clusters with the k-means algorithm. Figure 7 shows the byte size of every flow for the first four of the eight clusters. In contrast to the approach using IP addresses, we obtain classes with a more equal number of flows. The color represents the destination port.

Altogether, we can conclude that each cluster has the potential to simplify the detection of certain attacks. Each approach to cluster showed several classes with unique characteristics (e.g. only TCP traffic above Port 1024, only UDP packets to Port 51, or only a small set of different protocols with fixed byte size) and all clusters with a small amount of good defined traffic can support in finding anomalies, which are unexpected, rare and previously unseen events. It was not in the scope of this work to test

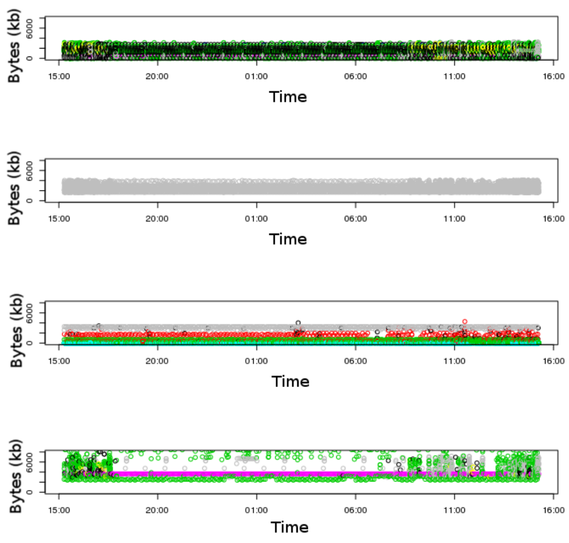

Figure 7: Byte size for each PCA cluster.

or justify a different number of clusters, which would be interesting. Here, we used the maximum number of clusters, where we could constantly observe traffic in each class, which is a tradeoff between usability and performance.

4 RESULTS

The aim of our study was to evaluate the effect of pre-clustering on Anomaly Detection. Here, we briefly presented a prototypical evaluation with and without prior clustering. While some state of the art Anomaly Detection Systems aim to unveal Advanced Persistant Threats, we use a strongly simplified Anomaly Detection algorithm and only focus on a particular subset of Network Anomaly Detection, namely network intrusions which visibly alter the statistical properties of network traffic. However, the contribution of this paper (preprocessing) can be applied to improve any modern detection alogrithm.

Our data consists of a training set, which is the previously introduced flow data, and a smaller test set captured on the same network. The test set contains three labeled attacks. The first two attacks aimed to compromise a managed switch from inside the network: First, a stealthy port scan, executed with nmap, discovered running services and open ports. Then, we manually tried six wrong passwords to log-in via SSH. The third attack was a SYN-flood on port 80 with random source addresses, executed with the tool hping3. The data has been acquired over a period of approximately 10 minutes and contains about 5500 flows and also contains regular background traffic. However, most flows belong to the attack, since every random IP address started a new flow.

We used an Anomaly Detection scheme which assigns, based on the number of occurrences o in historical data, a probability for being abnormal to each connection. Thus, previously unseen and rare events are marked as suspicious and receive a higher probability to be anomalous. Our historic (training-) data consists of the traces introduced in the previous chapter. We extracted vectors using the amount of packets, delay, destination port, bytes and layer 4 protocol and converted the values to categorical data (by rounding the each value to the next 1000) to prevent to many unique occurrences. We counts the number of occurrences in the historic data and calculated the probability p with $p = (100 - o)/100$. The binary result (decision) is made classifying each event as positive or negative based on a threshold. We calculated the trade-off between true positive (benefits) and false positive (costs) at various threshold settings using ROC-Curves. The clustering has been applied as introduced in the previous chapter. We used the mean value of all probabilities to create a ROC curve for the Anomaly Detection with clustering. The area under curve (AUC), which is **0.47** without cluster and **0.83** with cluster, is equal to the probability that our Anomaly Detection System assigns a higher anomaly score to abnormal traffic than normal traffic. Our reasoning for the good performance of the clusters is, that the stealthy scan and SSH intrusion (which is normally very hard to detect with just flow data) appeared unusual in the classes which contains traffic of the managed switch. The denial of service on the other hand appeared in a class very common for HTTP traffic and has been found because it was unusual in comparison to earlier traffic with similar characteristics.

5 CONCLUSIONS AND FUTURE WORK

In this paper, we presented an approach for reducing noise of network traffic and optimizing thresholds for Anomaly Detection Systems. These first results introduced the output of three different methods to cluster network traffic into smaller subsets. The clustered traffic shows a significantly reduced noise and clearly visible patterns in byte size, flow amount and destination port. We showed that pre-clustering can improve Anomaly Detection through noise reduction and independent threshold setting - The concept was evaluated with a prototype implementation and showed acceptable results. The optimal cluster can vary depending on the network infrastructure, used applications and Anomaly Detection method. But we can conclude that a good cluster needs to contain intrusive and normal traffic in order to optimize the threshold cluster by cluster and is at best statistical independent to reduce as much noise as possible. Our experiment pointed out that a proficient clustering can uncover abnormal traffic which is difficult to detect in the cumulative data. Especially our manual SSH intrusion attempt and the stealthy scans are normally difficult to detect by Anomaly Detection methods, because they do not transparently change volume or shape of the network traffic. However, we were easily able to detect them. It is likely that both, Anomaly Detection with and without clustering, could be equally improved by using more complex detection methods. There is a wide range of, here unused, Anomaly Detection schemes (e.g. Local Outlier Detection) and metrics (packets per second) available, which proved useful in other publications. However, we deliberately wanted to keep the experiment simple and focus on the effect of our preprocessing instead Anomaly Detection. We are aware that our approach heavily depends on the network characteristics. Our experimentation provided good results, because it was possible to separate networks into sub-networks with unique characteristics. It remains an open question whether it is possible to always reach this goal with any network. We belief that the task of finding significant features to split the traffic is event simpler when the network is larger than the network used for our experiments. In this work, we only performed tests on a single data set and artifical data, because we avoided the use of obsolete public data sets. As future prospects we would like to extend our experiments to a variaty of networks, evaluate more parameters and extend our work with ways to cluster obfuscated protocols and encrypted traffic.

REFERENCES

Bouzida, Y., Cuppens, F., Cuppens-Boulahia, N., and Gombault, S. (2004). Efficient intrusion detection using principal component analysis. In *3éme Conférence sur la Sécurité et Architectures Réseaux (SAR), La Londe, France*.

Brauckhoff, D., Salamatian, K., and May, M. (2009). Applying pca for traffic anomaly detection: Problems and solutions. In *INFOCOM 2009, IEEE*, pages 2866–2870. IEEE.

Chen, C. and Liu, L.-M. (1993). Joint estimation of model parameters and outlier effects in time series. *Journal of the American Statistical Association*, 88(421):284–297.

Chhabra, P., Scott, C., Kolaczyk, E. D., and Crovella, M. (2008). Distributed spatial anomaly detection. In *INFOCOM 2008. The 27th Conference on Computer Communications. IEEE*. IEEE.

Cretu-Ciocarlie, G. F., Stavrou, A., Locasto, M. E., and Stolfo, S. J. (2009). Adaptive anomaly detection via self-calibration and dynamic updating. In *Recent Advances in Intrusion Detection*, pages 41–60. Springer.

Davis, J. J. and Clark, A. J. (2011). Data preprocessing for anomaly based network intrusion detection: A review. *Computers & Security*, 30(6):353–375.

Denning, D. E. (1987). An intrusion-detection model. *Software Engineering, IEEE Transactions on*, (2):222–232.

Heady, R., Luger, G., Maccabe, A., and Servilla, M. (1990). *The architecture of a network-level intrusion detection system*. Department of Computer Science, College of Engineering, University of New Mexico.

Izakian, H. and Pedrycz, W. (2014). Anomaly detection and characterization in spatial time series data: A cluster-centric approach.

Kim, J. and Bentley, P. J. (2002). Towards an artificial immune system for network intrusion detection: an investigation of dynamic clonal selection. In *Evolutionary Computation, 2002. CEC'02. Proceedings of the 2002 Congress on*, volume 2, pages 1015–1020. IEEE.

Kim, M.-S., Kong, H.-J., Hong, S.-C., Chung, S.-H., and Hong, J. W. (2004). A flow-based method for abnormal network traffic detection. In *Network Operations and Management Symposium, 2004. NOMS 2004. IEEE/IFIP*, volume 1, pages 599–612. IEEE.

Leung, K. and Leckie, C. (2005). Unsupervised anomaly detection in network intrusion detection using clusters. In *Proceedings of the Twenty-eighth Australasian conference on Computer Science-Volume 38*, pages 333–342. Australian Computer Society, Inc.

Li, Z., Gao, Y., and Chen, Y. (2005). Towards a high-speed router-based anomaly/intrusion detection system.

Mahoney, M. and Chan, P. K. (2001). Phad: Packet header anomaly detection for identifying hostile network traffic.

Münz, G., Li, S., and Carle, G. (2007). Traffic anomaly detection using k-means clustering. In *GI/ITG Workshop MMBnet*.

Szmit, M., Adamus, S., Bugała, S., and Szmit, A. (2012). Anomaly detection 3.0 for snort. *Snort. AD Project*.