HAL open science

Machine Learning Approach for IP-Flow Record Anomaly Detection

Cynthia Wagner, Jérôme François, Radu State, Thomas Engel

To cite this version:

Cynthia Wagner, Jérôme François, Radu State, Thomas Engel. Machine Learning Approach for IP-Flow Record Anomaly Detection. 10th IFIP Networking Conference (NETWORKING), May 2011, Valencia, Spain. pp.28-39, 10.1007/978-3-642-20757-0_3 . inria-00613602

HAL Id: inria-00613602

https://inria.hal.science/inria-00613602v1

Submitted on 5 Aug 2011

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Distributed under a Creative Commons Attribution 4.0 International License

Machine Learning Approach for IP-Flow Record Anomaly Detection

Cynthia Wagner, Jérôme François, Radu State, and Thomas Engel

University of Luxembourg - SnT,
Campus Kircherg, L-1359 Luxembourg, Luxembourg
{cynthia.wagner,jerome.francois,
radu.state,thomas.engel}@uni.lu
http://www.securityandtrust.lu

Abstract. Faced to continuous arising new threats, the detection of anomalies in current operational networks has become essential. Network operators have to deal with huge data volumes for analysis purpose. To counter this main issue, dealing with IP flow (also known as Netflow) records is common in network management. However, still in modern networks, Netflow records represent high volume of data. In this paper, we present an approach for evaluating Netflow records by referring to a method of temporal aggregation applied to Machine Learning techniques. We present an approach that leverages support vector machines in order to analyze large volumes of Netflow records. Our approach is using a special kernel function, that takes into account both the contextual and the quantitative information of Netflow records. We assess the viability of our method by practical experimentation on data volumes provided by a major internet service provider in Luxembourg.

Keywords: Netflow Monitoring, Kernel Functions, Machine Learning Techniques, Intrusion Detection

1 Introduction

In today's business areas, it is nearly impossible to imagine network architectures without monitoring tools, and this for several reasons. Network monitoring has become one of the most important tools in the detection of malicious or unusual events of different genres, and gathering network traffic information is the main requirement for taking appropriate countermeasures. A realistic assumption is to say that nearly all commercially available routers can export monitored data, called Netflow records. These IP flow records, chronologically ordered ranges of packet sequences, are the most important source for the analysis of unidentifiable events, even if a lot of storage capacity and processing time is needed for an accurate analysis.

In this paper, we present a new method for processing Netflow records by referring to Machine Learning techniques. In a first step, we have developed a new kernel function that operates over Netflow records by analyzing its contextual and quantitative information. For the detection of IP flow record anomalies,

kernel functions have proven their usefulness, but to achieve good classification results, we apply in a second step, our kernel function to Support Vector Machines.

The structure of the paper is as follows: In section 2, we present the architecture of the model. We briefly present fundamental background information and present the anomaly detection part of the tool where we describe its main components. The more we describe the kernel function that has been used for the evaluation of Netflow records. In section 3 we describe the data set and the different attacks, used for the experiments and in section 4 we present our evaluation methods and discusses experimental results. Section 5 discusses related work and conclusions are given in section 6.

2 The Architecture of the Anomaly Detector

In the following section, we present the architecture of our model. The Netflow records, which are exported by the routers from the network to the Netflow collector module are used as input for our Anomaly Detector tool, represented in Figure 1. A Netflow record is a series of packets sent between two entities (hosts) in a chronological order. A Netflow record is composed of a source and destination IP-address, source and destination port numbers, traffic volume, packets and protocol for a session between the previous two endpoints. An argument for using Netflow records is, that they include all relevant network traffic information in a compressed version. It is composed of different modules, which are responsible for the pre-processing, evaluation and interpretation of the results, which means to answer the question, is there an attack or not.

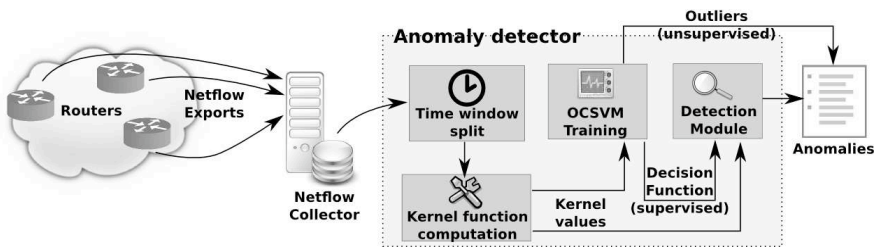

Fig. 1. Anomaly detector architecture

Detecting anomalies is related to classifying data in two different classes: benign and anomalies. Regarding the current context, the range of anomalies is very huge including Spam, Denial of Service (DoS), Scanning, Botnets, etc. Therefore, the anomaly class should regroup many different kinds of data points and so usual classifiers are not well fitted. For this purpose, multi-class classifier are potential solutions, but their main drawback is the necessity to have labeled

samples for each class of anomaly you expect to detect. It also means that a system based on such classifier is unable to detect new anomalies which is usually the kind in the current networks and Internet. Therefore, we have decided to use a very specific kind of algorithm which is called *one-class* classification. The aim is to build a classifier that is able to detect new anomalies, i.e. data points which do not follow a general traffic profile which is used for training the classifier.

More precisely, we consider Support Vector Machine (SVM) which provide good accuracy with a low complexity in many domains [22]. Besides, there is a specific one-class method which was initially proposed in [15]. This specific method has also proven its accuracy for intrusion detection in different contexts such as the system calls on a UNIX system. We propose to use OCSVM (One-class SVM) [23] for analyzing Netflow records based on a new kernel function.

2.1 OCSVM

In OCSVM, the main requirement is to have a dataset of samples assumed to represent the single class of data points. This set is denoted X. The general way to define the problem is the following one: assuming that the points from an original space S follow an underlying probability P, the goal is to define a subset space of this original space S such that the probability that a point from P lies outside S is maximized by a given value between 0 and 1 [15]. This means that during the learning phase, the goal is to determine a function which is positive when applied to a point from S and negative otherwise. Therefore, during the testing phase, the sign of this function indicates if the point is classified to the single class or not.

Assume, a labeled sample $X = \{x_1, \ldots, x_n\}$, the objective is to capture a small region enclosing these points by projecting them into a higher dimensional space, such that a better separation between a defined proportion of X and the origin, is obtained. The goal is to find a hyperplane with maximum margin, *i.e.*, by maximizing its distance from the origin. The projection is performed thanks to the function $\phi(x)$ and the proportion of points to separate is defined by $1-\nu$ with $\nu \in [0, 1]$. This problem is usually defined as follows:

$$min_{w,\rho,\xi_i \ldots n} \frac{1}{2}||w||^2 + \frac{1}{\nu n}\sum_{i=1}^{n}(\xi_i - \rho) \tag{1}$$

subject to:

$$< w.\phi(x_i)) > \geq \rho - \xi_i, \xi_i \geq 0 \tag{2}$$

The optimization problem has to be solved to identify two variables: w and ρ. ξ_i variables represent slack variables for allowing some points of S to not be located on the right side of the hyperplane. This avoids to bias the problem with very particular and maybe erroneous points. Since defining a projection function is not obvious, support vector methods traditionally rely on kernel functions. From a general point of view, they can be considered as similarity measure which have to be finitely positive semi-definite functions. The kernel function $K(x_i, x_j)$ is

equal to $< \phi(x_i), \phi(x_j) >$. Based on this function and by transforming the problem into its dual form, we obtain:

$$min_{\alpha_i \ldots n} \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j K(x_i, x_j) \tag{3}$$

subject to:

$$0 \leq \alpha_i \leq \frac{1}{\nu n}, \sum_{i=1}^{n} \alpha_i = 1 \tag{4}$$

Once this optimization problem is resolved, the decision function is defined by:

$$f(x) = sgn(\sum_{i=1}^{n} \alpha_i K(x_i, x) - \rho) \tag{5}$$

Therefore, this function is applied to each data point to test and if the sign is positive, it means that the point belongs to the class otherwise it is an anomaly. Obviously, many details were omitted due to space limits: the interested readers should refer to [15, 22]. In fact, the main parameter ν represents the maximal proportion of points of X lying on the wrong size of the hyperplane.

2.2 Unsupervised classification

OCSVM is usually used in a supervised manner as described in the previous section. However, the training can identify outliers thanks to the slack variables ξ_i. Therefore, only by applying this step to an entire dataset, detecting outliers and anomalies is possible. In this case, the parameter ν defines the maximum number of potential anomalies. Thus, it can be regarded as a threshold in a standard method for detecting a deviation from a profile. The main difference is that OCSVM benefit from a better detection, based on a better separation of data points in the high dimensional space.

Thus, we investigate both supervised and unsupervised method in this paper. The main advantage of the supervised one is that it should be more accurate due to the learning stage. Its main drawback is the need of sample data which are free of anomalies while the normal traffic is well represented regarding the different context that may happen. This completeness may be hard to obtain and the unsupervised technique can counter this requirement.

2.3 The Kernel Function for Netflow Processing

Kernel functions have proved their potential in the classification of high dimensional data. A kernel function calculus is homologous to a similarity measure for small data portions, where a mapping of an input space onto a higher dimensional space is performed as already explained in the previous sections. This makes it possible to separate dissimilar data and to calculate the distances, which are derived from a dot product, in this new space. We refer to Vapnik [18], who defines

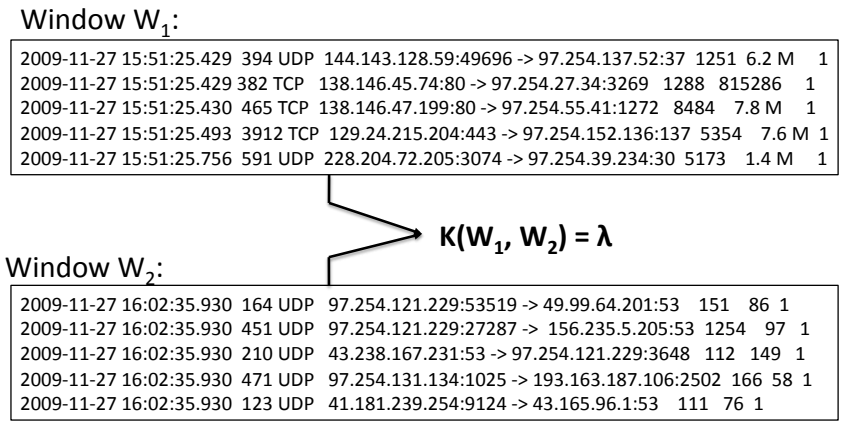

Fig. 2. Principle of kernel function and example of Netflow windows

a kernel function K as a mapping of $K : X \times X \in [0, \infty[$ from an original input space X to a similarity score $K(x, y) = \sum_i \phi_i(x)\phi_i(y) = \phi(x) \cdot \phi(y)$, where $\phi_i(x)$ describes a feature function over a data snippet x. A general property of a kernel function is *symmetry*, such that $[K(x, y) = K(y, x)]$ and *positive-definitness*.

For analyzing monitored windows of Netflow records, we have introduced a new kernel function $K(W_1, W_2)$, which enables to determine the similarity between two windows of IP flow records W_1 and W_2 of n seconds, in order to detect anomalies. An illustrating example of input Netflow windows are shown in Figure 2. As input space we use the Netflow records log files and define two metrics for our kernel function, provided by these flow log files. The first parameter in our kernel function are the IP-addresses from the source (src) and destination (dst) in CIDR[1] format, such that $IP = (prefix, suffixlength)$. While comparing IP-addresses, we consider the $prefix$ as the longest common sequence of two IP-addresses and the remaining bits as the $suffixlength$. As second parameter we have the quantitative factor of the captured Netflow records, which uses the traffic volume, called vol in Bytes. By this, we can model a Netflow record window for our kernel function as a set of IP flows $W = \{f_1, \ldots, f_n\}$ and a IP flow f defined as a f_i =$(prefix(src)_i$, $suffixlength(src)_i$, $prefix(dst)_i$, $suffixlength(dst)_i$,$vol_i)$. Our kernel function for Netflow windows $K(W_1, W_2)$ returns as output a similarity score given by the sum over four functions over all flows in a window, considering source and destination information as well as traffic volume and is composed of, a similarity function $s(a_i, b_j) \in [0, \infty[$ for the source and destination information and a matching function $v(a_i, b_j) \in [0, 1]$ for the traffic information. A higher similarity score means the more similar two

[1] CIDR — Classless InterDomain Routing: A standard system for the IP-address allocation and IP packet routing, where IP addresses are described by a network address part and a host identifier part within that network

windows are. Our kernel function K for two windows W_1 and W_2 is defined as

$$K(W_1, W_2) = \sum_{i \in N_{W_1}, j \in N_{W_2}} s_{src}(a_i, b_j) \times v_{src}(a_i, b_j) \times s_{dst}(a_i, b_j) \times v_{dst}(a_i, b_j) \tag{6}$$

where the windows of flows denoted by W_1 and W_2 are represented by N_{W_1} and N_{W_2}. We define the similarity measure parts for source and destination, $s_{src}(a_i, b_j)$, $s_{dst}(a_i, b_j)$ between flows a_i and b_j by

$$s_{src,dst}(a_i, b_j) = \begin{cases} \frac{2^{suffixlength_j}}{2^{suffixlength_i}} \text{ if } prefix_i \text{ prefix of } prefix_j \\ \frac{2^{suffixlength_i}}{2^{suffixlength_j}} \text{ if } prefix_j \text{ prefix of } prefix_i \\ 0 \text{ otherwise} \end{cases} \tag{7}$$

and the matching function for source and destination $v_{src,dst}(a_i, b_j)$ by

$$v_{src,dst}(a_i, b_j) = exp\big(- \frac{\mid vol_i - vol_j \mid^2}{\sigma^2} \big) \tag{8}$$

σ is the width scaling factor for the Gaussian kernel [3] and can be estimated on different ways, as for example by a grid search, where a range of values are used to find the optimal values for the kernel. To identify anomalous events, we use the different successive windows $K(W_i, W_{i+1})$ which we compare to each other. To illustrate the effectiveness of our kernel function without referring to

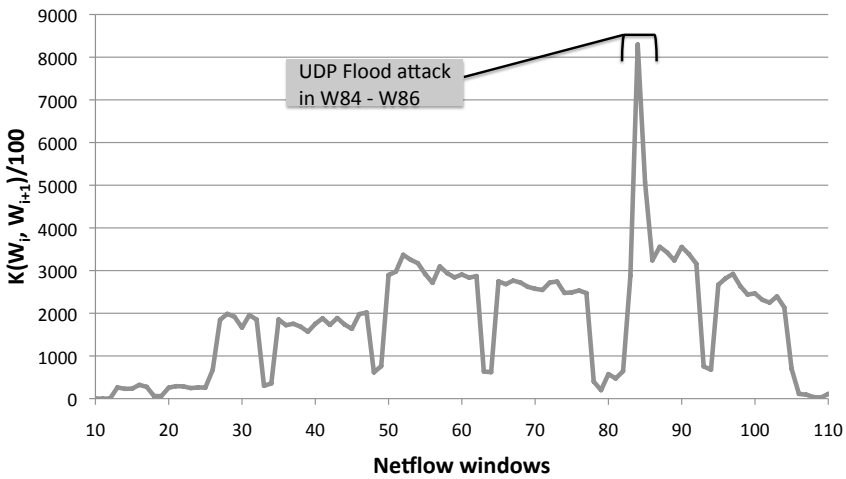

Fig. 3. Visualizing an attack by using the kernel function

classification, we show on a real example in Figure 3 that the used kernel function detects in this case a UDP flooding attack.

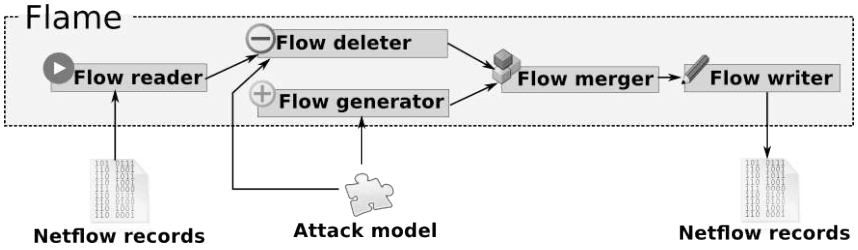

Fig. 4. Dataset generation

3 Dataset

Anomaly detection is a quite challenging topic and evaluating new innovative solutions is a hard task due to the lack of labeled datasets. There were some works aiming at providing such datasets. A very well-known dataset is the Lincoln data set[2] but it is quite out-dated now. A recent dataset is provided in [17] based on Netflow records. However, this dataset is only limited in attacks since they use a honeypot. Furthermore, this kind of architecture collects attack traces from end-user point of view and not from the network point of view. The dataset is generated by injecting synthetic attack traces into real traces using the tool Flame [2]. This tool is based on Netflows and is well suited in our case. It takes as input (flow reader) Netflow record files containing traces without malicious activities and creates attacks by generating (flow generator) or deleting (flow deleter) flow records based on attack models. The main idea is that attacks will generate traffic but also affect normal traffic like DDoS. That is why there is a need to merge these activities using the flow merger. Figure 4 highlights this process. A key part of Flame are the attack models which have to be created for the simulation of attacks.

Flow #	1,371,194
IP addresses #	128,781 (source), 125,723 (destination)
Duration	13min 31sec
Bytes #	7.5GB
Avg. bytes/flow	5492
Packets #	11.5M
Avg. packets/flow	8.36
UDP Flows #	983,511
TCP Flows #	375,132
ICMP Flows #	11,347
Other protocols Flows #	1204

Table 1. Data set statistics

[2] `http://www.ll.mit.edu/mission/communications/ist/index.html`

We use as starting point a real dataset provided by a network operator from Luxembourg and assume that it is free of malicious Netflow records. This assumption is made based on a secondary semi-automated screening of the traffic, where an ISP specific solution was used. The general characteristics of this data set are provided in Table 1. Even if the duration of the data set is small, it is sufficient for evaluating our scheme, since we have limited all attacks to last only 30 seconds. The Flame website also provides some attack models which were derived from real observations. At first we used this described attack model and later on, we extended it to create more stealthy attacks. Our data set contains the following attacks:

- **Nachi scan**: the Nachi/Welchia worm was released in 2003. First, it tests the reachability of hosts using ICMP scan which is considered in the attack model. One host of our original dataset sends single ICMP packets of 92 bytes to destination IP addresses following several properties: a shift between 40 to 45 between two consecutive scans, a positive or negative shift of 400 every 200 scans and a shift between 45 to 110 every 800 flows. The inter-arrival time of flows is generally around 2 milliseconds but there is a break of around 61 milliseconds after 5 scans.
- **Netbios scan**: it is a traditional scan for finding vulnerabilities. The corresponding UDP flows contain a single packet to port 137 where the inter-arrival time is generally between 60 and 70 milliseconds. The destination hosts are scanned sequentially while keeping one IP address sometimes. Besides, between 100 and 200 scans there is a shift in the destination IP address between 60 and 70.
- **DDoS UDP flood**: one host of our dataset receives UDP packets on various ports and also sends from multiple IP addresses with various ports. The attack operates by a burst of 40 flows. Then, there is a break between 60 and 120 milliseconds.
- **DDoS TCP flood**: this denial of service attack is against web servers running on TCP port 80 with 3 packets and 128 bytes. There are bursts of 10 packets before a break between 60 and 120 milliseconds.
- **stealthy DDoS UDP flood**: different to a normal UDP flood, we apply here more randomness on flows characteristics (number of packets, size, duration). The inter-arrival time can reach 1 second which represents a very stealthy characteristic for a flooding attack. This attack is more generic in order to improve the completeness of our tests.
- **DDoS UDP flood + traffic deletion**: this is equivalent to the DDoS UDP flood, but each additional flow originated by the victim has a probability of 0.2 to be deleted due to the victim overload.
- **Popup spam**: this kind of spam is similar of sending undesired Windows Messenger popups by using UDP port 1026 and 1027. Only one packet of 925 bytes is needed. The victims IP addresses do not highlight a regular pattern because two consecutive IP addresses have a gap of 200 addresses. The inter-arrivaltime is generally lower than 1 millisecond except every 200 flows where it is around 64 milliseconds and every 550 where it is 250 milliseconds.

– **SSH scan + TCP flood**: the goal of TCP scan is to probe an SSH server by trying to log in. This is by far the most popular attack occuring in the wild.Each flow contains between 1 and 4 packets. The inter-arrival time oscillates between 1 to 50 milliseconds. The destination IP addresses are scanned in a sequential way until 400 scans are executed approximately. After, there is a shift between 200 to 400 IP addresses. In order to test our approach in a real scenario, 5% of the attacks are considered successful and the corresponding victims trigger a TCP flood attack.

4 Evaluation

In the following section we describe the experimental part of this work. We have generated the data sets following the data set generation procedure, as shown in Figure 4. Generated Netflow windows have a duration of 5 seconds each. Then, we have applied our kernel function for the contextual and quantitative evaluation of Netflow record windows. On the similarity values obtained by the kernel function, the OCSVM has been applied in order to see, if our method can detect the different attacks or not. For each data set, we have used for the training phase about 20% of our Netflow record windows and for the testing phase the remaining 80%. The outcomes for our different simulations can be seen in Table 2.

Type of Attack	Results		
	Accuracy	False Positive rate	True Negative rate
Nachi scan	0.896	0.004	0.996
Netbios scan	0.938	0.000	1.000
Popup Spam	0.915	0.023	0.97
SSh scan + TCP flood	0.917	0.011	0.989
DDoS UDP flood	0.915	0.022	0.978
DDoS TCP flood	0.907	0.033	0.967
stealthy DDoS UDP flood	0.938	0.000	1.000
DDoS UDP flood + traffic deletion	0.934	0.000	1.000

Table 2. Classification results by using one class Support Vector Machines

The results by using the OCSVM are very promising, we can see that we have only a very low false positive rate and that we have an average accuracy over all attack classes of around 92% for the classification results. To compare our work with others, refer to the work of Yuan et al. [24] which used SVM for classification of network traffic of different types having attacks like worms, scans, etc. Yuan et al. achieved a similar classification accuracy (92,8%) as we did, which validates our approach, but there are no indications about the complexity of their algorithms.

5 Related work

Network monitoring techniques as, for example the work by Bahl et al. [1], are based on Netflow records or related data. One of the main interests of Netflow records is not only their compactness, but they can be used without respecting a complete TCP state machine and are available in most commercial available routers. For example, Karpilovsky et al. [8] analyzed the IPv6 deployment by referring to a Netflow analysis. A major drawback of using Netflow records is storage capacity. Large amounts of Netflow records (30 000 flows/second) are normal for ISP networks or networks with high link loads. In [16], the authors have described the effect of Netflow capture on the accuracy analysis. To reduce the aspect of storage or to perform near real time analysis, it is often referred to Netflow sampling as described for example by C. Estan [5] or Paredes-Oliva [13], but the challenge remains to identify good sampling rates. In the domain of information security and intrusion detection, a lot of relevant work has already been performed for evaluating and processing Netflow related data [10] [7], as for example statistics on packet information.

A new observed trend is to refer to Machine Learning Techniques for evaluating Netflows or IP flow related data, like Flow Mining [11], where main issue relies on the selection of good parameters for achieving high quality. Another aspect of using Machine Learning are the kernel methods, strong mathematical tools, which have found their utility in the evaluation of large and complex data sets. Kernel methods have already been introduced as a mathematical tool in early 1900's by Hilbert and were first introduced as part of Machine Learning Techniques in the late 1990's by Vapnik [18] and further developed in the early 2000's by Schölkopf et al. [14]. In recent past, kernel functions found their applicability in most different various domains, i.e. Genetics [12], Bioinformatics [19], Natural Language Processing [4], and can be applied to most different input formats, as structured format, i.e. graphs, trees, or unstructured format, i.e. texts.

A work that is similar to our work is the evaluation of temporal and spatial IP-flow records by kernel methods presented by Wagner et al. in [20] and [21]. The main difference is that the authors in [20] [21] refer to spatial and temporal aggregated IP-flows, using the Aguri [6] tool and apply a kernel function to detect anomalies. By referring to the aggregation tool, it is only possible to get a view from the source or destination side, but no global view of both sides. Using Netflow record windows without aggregation, we can keep a global view of the whole traffic information. Then, we go further and apply Support Vector Machines in order to detect and classify benign traffic from attacks. Supervised learning has become a common tool for evaluating large data sets on common patterns, as for example by using Support Vector Machines (SVM) [14] [18]. SVMs have proven their utility in most different domains as in natural language processing [4] or bioinformatics [19]. Since a few years, SVMs are also used in computer security, where they are used for intrusion detection [9] [25].

6 Conclusion

In this paper, we have presented a new approach for the evaluation of Netflow records on most various attacks on a real data set obtained from a large ISP in Luxembourg. For validating our approach, we have generated different types of attacks by referring to the trace modification tool called Flame. Our contribution first consists in the evaluation of Netflow records in their quantitative and contextual context, where we have developed a new kernel function for calculating the similarities between Netflow windows. Second, we have applied a new SVM algorithm to our developed kernel function in order to detect different attacks in data sets. We have implemented our tool in a distributed way by using Hadoop[3]. The classification results are very promising, such that most attacks can be identified.

Acknowledgments. This project has been supported by the SnT - Interdisciplinary Centre for Security, Reliability and Trust. Furthermore, we want to address our special thanks to RESTENA Luxembourg for their support.

References

1. Bahl, V., Chandra, R., Greenberg, A., Kandula, S., Maltz, D., Zhang, M.: Towards highly reliable enterprise network services via inference of multi-level dependencies. In: In SIGCOMM. pp. 13–24 (2007)
2. Brauckhoff, D., Wagner, A., May, M.: Flame: a flow-level anomaly modeling engine. In: Proceedings of the conference on Cyber security experimentation and test. USENIX Association (2008)
3. Burges, C.: A tutorial on support vector machines for pattern recognition. In: Data Mining and Knowledge Discovery 2(2). pp. 121 – 167 (1998)
4. Collins, M., Duffy, N.: Convolution kernels for natural language. In: Advances in Neural Information Processing Systems 14. pp. 625–632. MIT Press (2001)
5. Estan, C.: Building better netflow. In: In Proceedings of the 2004 conference on Applications, technologies, architectures, and protocols for computer communications (2004)
6. Kaizaki, R., Nakamura, O., Murai, J.: Characteristics of denial of service attacks on internet using aguri. In: Information Networking, Lecture Notes in Computer Science, vol. 2662, pp. 849–857. Springer Berlin / Heidelberg (2003)
7. Karagiannis, T., Papagiannaki, K., Faloutsos, M.: BLINC: multilevel traffic classification in the dark. In: ACM Conference on Applications, technologies, architectures, and protocols for computer communications (SIGCOMM) (2005)
8. Karpilovsky, E.: Quantifying the extent of ipv6 deployment. In: Lecture Notes in Computer Science, vol. 5448 (2009)
9. Khan, L., Awad, M., Thuraisingham, B.: A new intrusion detection system using support vector machines and hierarchical clustering. The VLDB Journal 16(4), 507–521 (2007)
10. Lakhina, A., Crovella, M., Diot, C.: Mining anomalies using traffic feature distributions. In: ACM SIGCOMM'05 (2005)

[3] `http://hadoop.apache.org/`

11. Lee, W., Stolfo, S., Mok, K.: Mining in a data-flow environment: experience in network intrusion detection. In: 5th International Conference on Knowledge Discovery and Data Mining (1999)
12. Nguyen, H., Ohn, S., Chae, S., Song, D., Lee, I.: Optimizing weighted kernel function for support vector machine by genetic algorithm. In: Gelbukh, A., Reyes-Garcia, C. (eds.) MICAI 2006: Advances in Artificial Intelligence, Lecture Notes in Computer Science, vol. 4293, pp. 583–592. Springer Berlin / Heidelberg (2006)
13. Paredes-Oliva, I.: Portscan detection with sampled netflow. In: Lecture Notes in Computer Science, vol. 5537 (2009)
14. Schölkopf, B., Smola, A.: Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond. MIT Press, Cambridge, MA, USA (2001)
15. Schölkopf, B., Platt, J.C., Shawe-Taylor, J.C., Smola, A.J., Williamson, R.C.: Estimating the support of a high-dimensional distribution. Neural Comput. 13, 1443–1471 (July 2001)
16. Sommer, R.: Netflow: Information loss or win? In: In Proceedings of the 2nd ACM SIGCOMM Workshop on Internet measurement (2002)
17. Sperotto, A., Sadre, R., van Vliet, D.F., Pras, A.: A labeled data set for flow-based intrusion detection. In: IP Operations and Management (IPOM 2009). Springer (October 2009)
18. Vapnik, V.: In: Statistical Learning Theory. Wiley (1998)
19. Vert, J.: A tree kernel to analyze phylogenetic profiles (2002)
20. Wagner, C., Wagener, G., State, R., Dulaunoy, A., Engel, T.: Game theory driven monitoring of spatial-aggregated ip-flow records. 6th International Conference on Network and services Management (2010)
21. Wagner, C., Wagener, G., State, R., Dulaunoy, A., Engel, T.: Peekkernelflows: Peeking into ip flows. 7th International Workshop on Visualization for Cyber Security pp. 52–57 (2010)
22. Wang, L. (ed.): Support Vector Machines: Theory and Applications, Studies in Fuzziness and Soft Computing, vol. 177. Springer (2005)
23. Wang, Y., Wong, J., Miner, A.: Anomaly intrusion detection using one class svm. In: Information Assurance Workshop, 2004. Proceedings from the Fifth Annual IEEE SMC. pp. 358 – 364 (june 2004)
24. Y., R., Li, Z., Guan, X., Xu, L.: An svm-based machine learning method for accurate internet traffic classification. Information Systems Frontiers 12, 149–156 (April 2010)
25. Zhang, B.Y., Yin, J.P., Hao, J.B., Zhang, D.X., Wang, S.: Using support vector machine to detect unknown computer viruses. International Journal of Computational Intelligence Research 2(1) (2006)