DANAK: Finding the odd!

Cynthia Wagner, Jérôme François, Radu State, Thomas Engel

DANAK: Finding the odd!

Cynthia Wagner, Jérôme François, Radu State, Thomas Engel
University of Luxembourg,
SnT - Interdisciplinary Centre for Security, Reliability and Trust
Campus Kirchberg
L-1359 Luxembourg
Email: {cynthia.wagner, jerome.francois, radu.state, thomas.engel}@uni.lu
http://www.securityandtrust.lu

***Abstract*—With the growth of network connectivity and network sizes, the interest in traffic classification respectively attack and anomaly detection in network monitoring and security related activities have become very strong. In this paper, a new tool called DANAK has been developed for the detection of anomalies in Netflow records by referring to spatial and temporal information aggregation in combination with Machine Learning techniques. Spatially aggregated Netflow records are fed in a new designed kernel function in order to analyze Netflow records on context and quantitative information. To strengthen the analysis of large volumes of Netflow records, support vector machines are applied. The proposed method has been validated by extensive experimentation on real data sets, including numerous attack strategies of different roots.**

I. Introduction

The need for network security and monitoring activities has become very strong with the growing sizes of networks and the increasing requests for permanent network connectivity. The demand for new and easy-to-use network analysis methods, abstracting complex processes for counter-fighting attacks or detecting network anomalies, is still present. Interesting challenges in networking monitoring are, for example, the automation of security monitoring, like the interaction of different monitoring modules, or to reduce the necessity of interaction between network operator and monitoring framework. Another issue is the large amount of available data that is monitored, where it is nearly impossible to store and evaluate all data. One kind of available information on network borders are Netflow [7] records, which can be exported by mostly all routers today, but storing and analyzing these large quantities of Netflow records instantly is a real problem.

A question that arises here is, if it is really necessary to evaluate all Netflow records or if abstractions of Netflow records are sufficient to provide the same outcomes. In recent research, it has often been referred to condensed forms of packets or sampling methods, where only a selected portion of Netflow records were used for evaluation, but a more novel approach is to use an aggregation-based technique.

In this paper, a new monitoring framework called DANAK (**D**etecting **A**nomalies in **N**etflow records by spatial **A**ggregation and **K**ernel methods) is described, which aims to detect anomalies and attacks in Netflow records. The analysis of all individual Netflow records is nearly impossible, therefore a spatial and temporal Netflow aggregation based technique is used. For the analysis of the aggregated Netflow records, it has been referred to kernel functions, a subdomain of Machine Learning Techniques.

A new kernel function is described, which captures topological changes and traffic evolution in aggregated Netflow records and by this, deduce if network traffic is anomalous or normal. To strengthen the results of the implemented approach, in addition to the kernel function, a Phase Space Analysis function has been implemented and the different profiles are evaluated by referring to a classification algorithm in order to gain a better understanding of the traffic evolution.

The paper is structured as follows: In section II, the modules of the monitoring framework are presented. The section describes the main components of DANAK and presents theoretical background information about the spatial aggregation of Netflow records, the kernel function and the Phase Space Analysis method. Section III describes the attack injection module as well as the different evaluated attacks and their outcomes. Section IV presents relevant research work in this domain and section V describes future work and presents the conclusions about DANAK.

II. The architecture

In the following section presents the most relevant theoretical components including background information and the implemented modules of the monitoring framework DANAK. An overview of DANAK is given in Figure 1. The routers, with Netflow record exporting functionality, capture packets on the network and forward the records to the Netflow Collector module, which stores them. After the capture of Netflow records, DANAK comes into play.

The first module of DANAK, the *S-aggregation*-module, performs the task of spatial - temporal aggregation of Netflow records and is responsible for the generation of traffic profiles. These profiles represent traffic volumes, aggregated by IP-address prefixes within a time window, having a tree-like representation. Therefore, they are of use to track traffic changes in the network.

The second module, *PS-Embedding and Kernel Function*, is the main component of DANAK. This module performs a Phase Space Analysis, the processing and the evaluation of the profiles by applying a new kernel calculus model. This method refines profiles by leveraging historical knowledge.

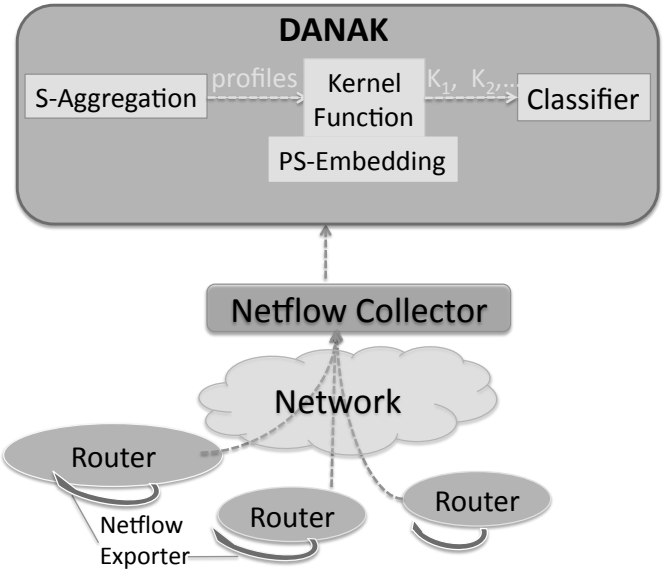

Fig. 1. General View of the DANAK Framework Architecture

The results obtained from the kernel function are applied to a Machine Learning module, the *Classifier*. This module groups the previously obtained kernel values into anomalies or conventional traffic.

A. Spatial-Temporal Aggregation of Netflow Records

Spatial and temporal aggregation of IP related information was first presented by a tool called Aguri in [6], [12]. Aguri is a near real-time tool, which uses full packet captures as input and then spatially aggregates network data into traffic profiles. A clear advantage of this approach is that due to aggregation, overviews on subnet basis can be presented instead of single IP flow basis. In [6], the authors are able to detect different attacks, like flooding or denial of service attacks.

```
Total =  11008757
0.0.0.0/0  540220 (4.91% / 100.00%)
└96.0.0.0/3  560312 (5.09% / 27.21%)
  101.0.0.0/8  550880 (5.00% / 22.12%)
  │101.138.64.0/20  754834 (6.86% / 11.99%)
  │ 101.138.74.115/32  564682 (5.13% / 5.13%)
  │101.176.128.0/19  564851 (5.13% / 5.13%)
 144.0.0.0/4  771170 (7.01% / 67.88%)
  144.115.176.0/20  552664 (5.02% / 5.02%)
  145.213.132.0/22  590328 (5.36% / 5.36%)
  145.213.144.0/20  712268 (6.47% / 6.47%)
  147.186.128.0/18  737222 (6.70% / 17.24%)
   147.186.144.0/21  599586 (5.45% / 5.45%)
   147.186.160.0/21  561074 (5.10% / 5.10%)
  147.186.192.0/18  559543 (5.08% / 26.78%)
   147.186.192.0/20  629860 (5.72% / 5.72%)
   147.186.208.0/21  561773 (5.10% / 5.10%)
   147.186.240.0/21  608873 (5.53% / 5.53%)
   147.186.248.0/21  588617 (5.35% / 5.35%)
```

Fig. 2. Generated Tree of Aggregation Task for a Destination Profile

In this paper, a similar method to Aguri has been implemented. A major differences to Aguri is, DANAK needs Netflow records and Aguri full packet captures as input format. In Figure 1, this module is denoted by *S-Aggregation*. The logged Netflow records are spatially aggregated into tree-like structures by using a user-defined aggregation-threshold α over a time interval of η seconds. DANAK outputs traffic profiles in tree-like form by spatially assembling subnet, host and quantitative traffic information by extraction from the captured Netflow records. Netflow records are aggregated into a tree representation, having a Patricia tree as primal structure [20], with a bounded size for scalability reason. Each node in the tree represents a subnet. When a record has to be inserted, it is compared with other nodes using a pre-order traversal. Netflow records are compared based on their longest common IP prefix. If a match is found, the volume part (called vol_i and expressed in bytes or packets) is updated, otherwise a new branching point with a new node is created. At the end of the aggregation task, only nodes having a volume $vol_i > \alpha$ are kept. The output generated by DANAK are traffic profiles for source and destination information. For a detailed description of the aggregation task, we refer the reader to [6].

The traffic profiles contain a summary of the contextual and quantitative evolution of network traffic for a small time interval and serve as input information to the kernel function. An extract for a destination traffic profile is represented in Figure 2. This traffic profile is generated in DANAK with an aggregation threshold of $\alpha = 5\%$. The used time interval is $\eta = 5$ seconds. The first line in Figure 2 represents the overall transmitted traffic information in bytes for a traffic profile. At the second line starts the generated traffic profile with information about the network traffic. Each line is a node and is divided as follows,

- The first column in a traffic profile line represents the IP-address subnet.
- The second column holds the traffic load in bytes.
- The third column represents the percentage of traffic load in a node (including aggregated nodes).
- The last column holds the percentage of traffic load of the current subtree.

B. The Kernel Function for Traffic Profiles

For the evaluation of the generated traffic profiles, it has been referred to a commonly used method in Machine Learning, the kernel functions [25], [23]. A kernel function can be described as a similarity function for for complex input data and the distances can directly be derived without exhaustive calculations. Vapnik [25] defines a kernel function K as a similarity mapping $K : X \times X \rightarrow [0, \infty[$, where X is an input space and $K(x, y) = \sum_i \phi_i(x)\phi_i(y) = \phi(x) \cdot \phi(y)$ a similarity score, with $\phi_i(x)$ being a feature function over a sample x. The more, a kernel function respects symmetry $[K(x, y) = K(y, x)]$ and positive-definiteness.

In this paper, a new kernel function is introduced for the evaluation of the generated traffic profiles. It calculates the similarity between two input profiles, T_n and T_m, by counting the amount of common nodes and not by performing an

exhaustive computation over the entire space [25]. A schematic figure of the kernel function process can be seen in Figure 3.

For the traffic profile evaluation, the kernel function needs the source and destination IP-addresses and the traffic information as input metrics.

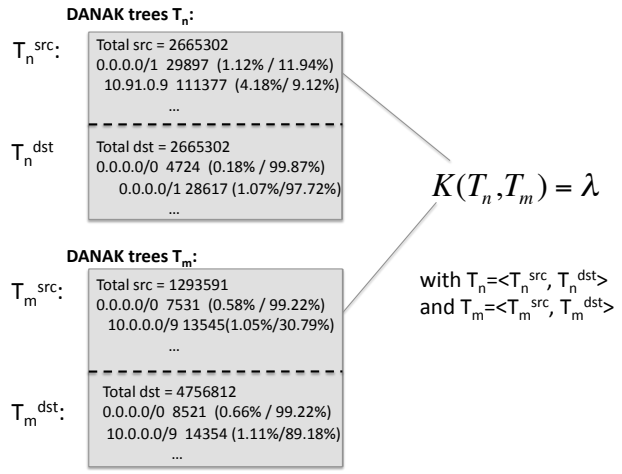

Fig. 3. Traffic Profiles in Tree Form as Input for the Kernel Function

The first kernel function metric uses source and destination IP-address information (first column in Figure 4) respecting the CIDR[1] scheme [22]. In the kernel function for profiles, an IP-subnets can be described as, *IP = ($prefix_i$, $prefixlength_i$)*, where $prefix_i$ is the IP network part and $prefixlength_i$ the length of the host identifier part (shown in Figure 4).

The second metric for the kernel function handles the traffic volume information (bytes) for a node, expressed in percent. This percentage represents the traffic volume in a node including other aggregated nodes. This parameter is called vol_i (third column in Figure 4). To illustrate the selection of the relevant kernel function parameters, an example is shown in Figure 4.

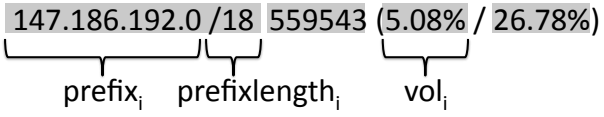

Fig. 4. Parameter Selection for the Kernel Function

A spatially-aggregated profile can be described as $T_i =< T_i^{src}, T_i^{dst} >$ with T_i^{src} for a source profile and T_i^{dst} for a destination profile. Each profile type is defined as a set of nodes $T_i = \{n_1, ..., n_j\}$, where a node n_x is defined as n_x = (prefix$_x$, prefixlength$_x$, vol$_x$).

To complete the definition of spatial-temporal aggregation, the temporal aggregation can be defined as the timeline of traffic profiles and can be described as $\{< T_1^{src}, T_1^{dst} >, \ldots, < T_M^{src}, T_M^{dst} >\}$ with T_i^{src} being the aggregation tree for time window i for the source information and T_i^{dst} for destination information.

Consider two traffic profiles denoted by $T_n =< T_n^{src}, T_n^{dst} >$ and $T_m =< T_m^{dst}, T_m^{dst} >$. The output of the kernel function about these two profiles $K(T_n, T_m)$ gives a similarity score being the sum over all aggregation tree nodes and can be defined as,

$$K(T_n, T_m) = \frac{1}{2} \sum_{i \in T_n^{src}, j \in T_m^{src}} s(i,j) \times v(i,j) + \frac{1}{2} \sum_{i \in T_n^{dst}, j \in T_m^{dst}} s(i,j) \times v(i,j) \quad (1)$$

with s measuring the similarity between subnets and v catching the similarity between volumes. The similarity score s is the measure for the similitude of traffic profiles. For this, the similarity function compares traffic profiles and calculates the similarity score based on their longest common IP prefix. The higher the similarity score, the more similar the traffic profiles are. The similarity function part s for source and destination traffic profiles can be defined as follows:

$$s(i,j) = \begin{cases} \frac{2^{prefixlength_j}}{2^{prefixlength_i}} & \text{if } prefix_i \text{ prefix of } prefix_j \\ \frac{2^{prefixlength_i}}{2^{prefixlength_j}} & \text{if } prefix_j \text{ prefix of } prefix_i \\ 0 & \text{otherwise} \end{cases} \quad (2)$$

The matching function part v is known as a classical Gaussian kernel where σ scales the width of this kernel function. The matching function part v is computed for source and destination traffic profiles and performs the volume calculi. The matching function v can be defined as:

$$v(i,j) = exp\big(- \frac{\mid vol_i - vol_j \mid^2}{\sigma^2} \big) \quad (3)$$

A good choice of σ is therefore necessary for the results. The optimization of σ can be realized by a grid search [27] or by applying the Kruskal algorithm [15].

C. The Phase Space Method with delayed coordinates

In this section, the Phase Space Analysis with delayed coordinates is briefly presented. Traffic profiles summarize network activity for a period of η seconds. In the case of an attack, it generally operates over more than one time period (η seconds). The same is valid for benign traffic. The principle of the Phase Space Analysis with delayed coordinates allows to reconstruct missing dimensions by using previous and delayed information (function values) as additional coordinates [24],

[1]CIDR: Classless Inter-Domain Routing, Standard scheme for IP-address allocation and IP packet routing. IP addresses are decomposed into network part and host identifier part. Example: $192.168.0.0/16$ means $192.168.0.0$ is the network part and $/16$ the routing prefix size.

# Flows	3 907 859
# IP Addresses	source addresses : 250 314 destination addresses: 235 120
# bytes	24.1 GB
Avg. bytes/Flow	6 829
# Packets	38 132 130
Avg. Packets/Flow	9.76
# UDP Flows	2 756 321
# TCP Flows	1 097 030
# ICMP Flows	50 914
# Other Protocol Flows	3 594

TABLE I
NETWORK OPERATOR DATA SET STATISTICS

[18], [28]. By this, a historical profile knowledge (including missing dimensions) can be explored.

In [28], Phase Space Analysis is explained as a n-dimensional space that fully describes a n-variable system, in this case, the generated traffic profiles. A three-dimensional representation of one-dimensional data is generated by calculating the first-order differences between traffic profiles. By this, the function dynamics become more suitable and useful for further processing. The obtained coordinates are also known as function attractors [18], which can be used for further processing in the classifier module.

The Phase Space Analysis with delayed coordinates can be defined as follow: Consider an input set s, and x, y, z as the point coordinates, then the equation for a sample n can be defined as,

$$x[n] = s[n-2] - s[n-3] \quad (4)$$

$$y[n] = s[n-1] - s[n-2] \quad (5)$$

$$z[n] = s[n] - s[n-1] \quad (6)$$

In this paper, $s[n] = K(T_n, T_{n-1})$ and the first-order difference equations are applied to previous, delayed and actual traffic profiles kernel function values in order to obtain an accurate and extended view on the evolution of the network topology and traffic. This allows to catch the dynamism of changes unlike direct historical comparisons of one sample with prior ones. The obtained values are used as input for the evaluation with the classification method.

III. EXPERIMENTAL RESULTS

This section highlights the major outcomes by using DANAK for the classification of traffic profiles. A major problem for the validation of the approach was to find a labeled dataset. A lot of previous works used the Lincoln data set[2], but nowadays it is considered outdated. Therefore in this work, it has been referred to a real data set provided by a large network operator in Luxembourg and the dataset is assumed to be free from malicious network traffic. This assumption is based on the fact that a secondary semi-automated traffic screening has been realized with aid of a network operator specific solution [1].

A technical description of an average sized data set of a duration of 15 minutes is given in Table I.

The parameters for the aggregation task of the Netflow records have been set to a minimal value of $\alpha = 5\%$ for the aggregation threshold, in order to have a finest profile granularity and the traffic profile time window set to $\eta = 5$ seconds. The generated data set included 180 traffic profiles, where one profile represents a monitoring of 5 seconds.

A. Injection attacks

For the validation of the monitoring framework DANAK, the provided network operator data set has been extended by injecting synthetic attack traces into real traces using the tool Flame [4]. Flame is a realistic Netflow-based attack injection tool. The input for the *flow reader* of Flame are the cleaned Netflow record files from the network operator. In the *flow generator* and *flow deleter*, Netflow records can be generated or deleted, depending of the attack typ. The main idea of Flame is to generate traffic by injecting different attacks, but also to affect normal traffic scenarios, as for example in DDoS, where traffic is reduced to the overload. An overview of the Flame architecture is shown in Figure 5.

The injected attacks have been limited to time periods of 5 minutes. This duration is necessary to evaluate the attack detection with various background traffic. In [4], different attack models derived from real-world scenarios have been described. In this paper, some of these attack models have been applied to evaluate the performance of DANAK. Additionally to this performance measure, some existing attack models of Flame have been extended in order to create new attack models. The list below shows the attacks that have been generated for the validation of DANAK.

- **Nachi scan**[3]: The Nachi worm first tests the hosts' reachability by ICMP scanning. The attack model considers this, such that one host of the original data set sends single ICMP packets of 92 bytes to destination IP addresses respecting the following properties: a shift of 40 to 45 IP-addresses between two consecutive scans, a positive/negative IP-address shift of 400 each 200 scans and a shift between 45 to 110 IP-addresses each 800

[2]http://www.ll.mit.edu/mission/communications/ist/corpora/ideval/data/index.html, last accessed: march 2011

[3]http://www.mcafee.com/threat-intelligence/malware/default.aspx?id=101013, last accessed: march 2011

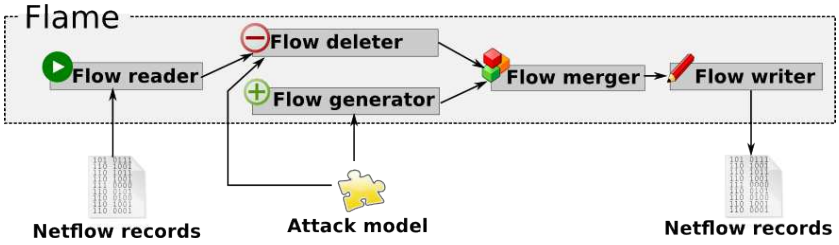

Fig. 5. The Flame Architecture

flows. The flow inter-arrival time is approximately 2 milliseconds but, a break around 61 milliseconds after 5 scans has been observed.

- **Netbios scan**[4]: A well known scanning method for vulnerability detection. The corresponding UDP flows contain a single packet to port 137 where the inter-arrival time is mostly around 60 to 70 milliseconds. Destination hosts are scanned sequentially while keeping one IP-address. Furthermore, between 100 to 200 scans there is an IP-address shift in the destination range of 60 to 70 IP-addresses.

- **SSH scan + TCP flood**: TCP scans aim to probe SSH servers by trying to log in. Each flow is built of 1 to 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 approximately executed. Then a shift of 200 to 400 addresses has been observed. In order to simulate a real scenario, 5% of the attacks are considered successful, triggering the victims TCP flood attack.

- **DDoS UDP flood**: One host receives UDP packets which have been sent from various ports and from multiple IP addresses having different ports. This attack operates by bursts of 40 flows and has a break for 60 to 120 milliseconds.

- **DDoS TCP flood**: This DoS attack is directed against web servers running on TCP port 80 with 3 packets and 128 bytes. Bursts can be observed 10 packets before a break of 60 to 120 milliseconds.

- **Popup spam** [11]: This kind of spam can be compared to undesired Windows Messenger popups by using UDP port 1026/1027. Only one packet of 925 bytes is needed. The victims IP addresses do not show up a regular pattern because two consecutive IP-addresses have a gap of 200 IP-addresses. The inter-arrival time is generally lower than 1 millisecond except each 200 flows where it is 64 milliseconds and every 550 flows where it is 250 milliseconds.

[4]http://www.iss.net/security_center/reference/vuln/Netbios_Name_Scan.htm, last accessed: march 2011

- **DDoS UDP flood + traffic deletion**: The attack is similar to the DDoS UDP flooding attack, but each additional flow originated by the victim has a deletion probability of 0.2 due to the victim overload.

B. Interpretation of the Kernel Function

In Figure 6, the results of the kernel function for n traffic profiles are graphically represented. In Figure 6, the z-coordinate of the Phase Space Analysis (given in Equation 6) is represented, where a violent DDoS UDP Flood attack has been injected into the data set. The attack has been injected between traffic profiles 60 and 120, which represents a 5 minute attack injection period. It can be observed that the anomalies can clearly be detected by the kernel function and be visualized in the graph.

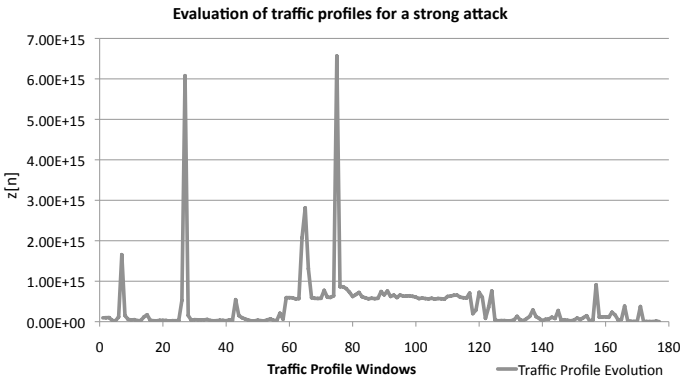

Fig. 6. Kernel Function Evaluation of a Violent DDoS UDP Flood Attack

In Figure 7, a realistic DDoS Flood attack has been injected into the data set. Also in this case, the attack occurs between traffic profile 60 and 120, but this time the attack cannot be easily detected by kernel function, as shown in the graph. This result really shows that in this case, the kernel function is not sufficient to identify the anomalies and that it cannot be seen by human expertise per se.

C. Interpretation of the Phase Space Embedding Analysis

The following section discusses the utility of using Phase Space Analysis in the detection of anomalies in Netflow records as it was shown that the kernel function (via the z coordinates) by itself, is not strong enough to identify

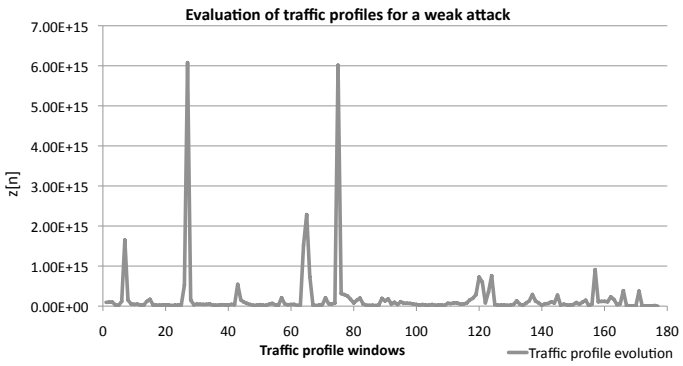

Fig. 7. Kernel Function Evaluation of a Weak DDoS UDP Flood Attack

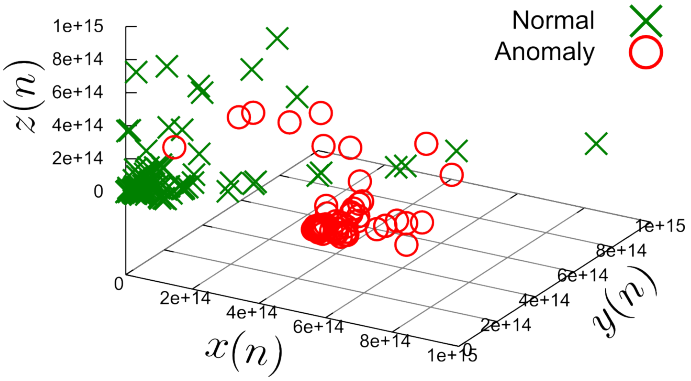

Fig. 9. DDoS UDP Flood Attack with Phase Space Embedding Analysis

anomalous network behaviours. Phase Space Analysis with delayed coordinates allows to reconstruct missing dimensions by using previous and delayed information (function values) as additional coordinates as illustrated in section II-C. In this paper, it is used to built a historical profile knowledge (including missing dimensions) which shows to be very effective. In Figure 8, a DDoS UDP Flooding attack is visualized without using Phase Space Embedding. The figure represents a simple history of the actual traffic profile T_i (expressed as kernel value) compared to the three previous profiles T_{i-1}, T_{i-2} and T_{i-3}.

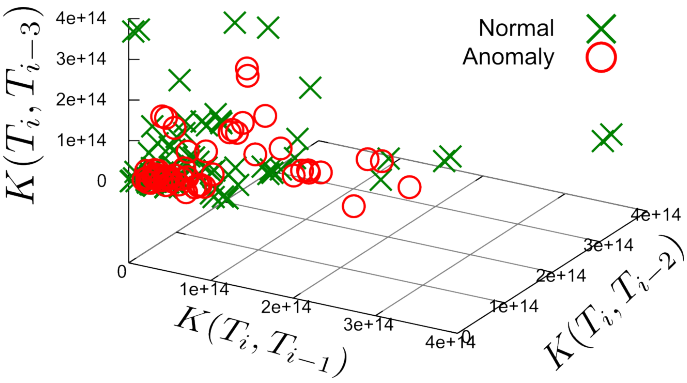

Fig. 8. DDoS UDP Flood Attack without Phase Space Embedding Analysis

In Figure 9, the same DDoS UDP attack is visualized, but this time by using Phase Space Analysis. The axis in the graph represent the x, y, z -coordinates obtained by applying the first order difference equations, presented in section II-C.

By comparing the two figures (Figure 8 and 9), it can been seen that the anomalies can be visualized by applying Phase Space Analysis. In Figure 9, anomalies are now are regrouped and disjoint compared to the case in Figure 8. By combining kernel function and Phase Space Analysis different types of attacks can be visually detected, but to strengthen the validation of the presented approach, a classification algorithm is used in order to identify anomalous and benign network activities.

D. Classification results

For the classification task, it has been referred to the open-source Machine Learning tool WEKA version 3-6-4[5]. This tool implements different Machine Learning algorithms (supervised and unsupervised) for real scenarios.

To evaluate the Phase Space Analysis with the kernel function values obtained by the traffic profile processing, the SimpleCART algorithm has been used. The CART algorithm [3] is a supervised learning algorithm and stands for Classification and Regression Trees. It relies on a binary tree-building algorithm and aims to identify the optimal splitting parameter for the binary splitting. The SimpleCART of WEKA implements a minimal cost pruning method [3]. To fine-tune the experiments, a cross-validation parameter of 5 folds has been set. Cross-validation is the equivalent of dividing the data set into n equally sized sub-data sets (in this case $n = 5$). The decision tree is generated from subsets where one of the subsets is left out. The performance of these others ones is assessed. This is repeated for all subsets and the average is taken. Each injected attack has been evaluated by using the SimpleCART algorithm. Table II summarizes the classifier results for each attack by the True-Positive and the False-Positive Rate. Finally, it is important to note that, it has been assumed that the data set is free of malicious traffic, but there is no ground truth. Hence, some False Positives may reveal as real attacks in the original data set.

It can be observed that this classifier can accurately detect the different attacks which have been injected into the data set. This highlights that the use of kernel functions for traffic profile evaluation is valid, even if weak attacks could not be detected visually in the graph (see Figure 7). The same conclusions hold for the Phase Space Analysis. The True Positive rates vary around 88% to 94% for the attacks. The False Positive rates are acceptable (2 - 20%), even if for one attack, the rate is higher than 30%. The previous paper by Wagner et al. [26] shows that False Positive rates can be lowered. However in [26], Netflow records were used as input (without aggregation) and compared in time windows, which

[5]http://www.cs.waikato.ac.nz/~ml/weka/index.html, last accessed: march 2011

Type of Attack	Results	
	True Positive Rate	False Positive Rate
Nachi scan	0.912	0.222
Netbios scan	0.941	0.185
Popup Spam	0.882	0.361
SSh scan + TCP flood	0.882	0.028
DDoS UDP flood	0.923	0.077
DDoS TCP flood	0.887	0.027
DDoS UDP flood + traffic deletion	0.932	0.072

TABLE II
CLASSIFICATION RESULTS BY USING A CLASSIFIER

is equivalent to compute pairwise distances of approximately 22 000 flows per window. This paper shows the benefit of combining Phase Space Analysis with kernel function to Netflow records, which is only possible if Netflow records are aggregated, as done in DANAK. By inspecting the data set more deeply, it can be observed that the size of traffic profiles varies between 23 to 52 nodes per tree. In the worst case this means to compute 52^2 kernel functions for the DANAK method which is very low in comparison of 22000^2 for the simple Netflow record method. Obviously, the aggregation process needs some additional optimization, as proposed in [6], [12], especially for the bounding of the size of a tree during its construction phase. By introducing tree-optimization, trees can be constructed on the fly. In other words, aggregation is a good tradeoff for improving the scalability of the presented approach by maintaining a high True Positive rate with only a low cost in terms of False Positives.

IV. RELATED WORK

The main interest for using Netflow records in the monitoring task is the compact format and the fact that most commercially available routers support Netflow record export. In the domain of Network monitoring and security, a lot of research has been done yet and a lot of different techniques been presented [2], [16], [29], either by introducing dependency models or using statistics. A more recent approach is to refer to rule-based methods, as for example in [13], [8]. In [8], the authors use packet based techniques to generated signatures, whereas in [13], the authors try to classify traffic by applications.

Netflow records contain relevant information about traffic on the network and the evaluation insights to the operators in case of suspicious events, but require strong storage requirements. In Luxembourg, peak Rates of 60 000 flows /second have been regularly observed at ISP networks supporting high link loads. Therefore, different authors have described the Netflow captures by referring to Netflow sampling [9], [21], but finding good sampling rates remains challenging and precious information is not considerd. Another interesting approach was presented in [10], where the authors have implemented a new column-oriented storage infrastructure. In [10], the authors have been inspired by spatial and temporal aggregation for their tool. Spatial-temporal aggregation was first presented in [6], [12]. A major advantage of this technique is that key information is preserved and less relevant information aggregated into so called traffic profiles.

Phase Space Analysis has first been in [24] for chaotic dynamical systems to reconstruct attractors from observations by a generic function. This analysis technique is commonly used for analyzing dynamic systems, like nonlinear systems or indeterministic chaos theories [18], [24]. The method of using Phase Space Analysis with delayed coordinates in computer science has first been presented by [28]. In [28], Phase Space Analysis has been used for measuring TCP/IP sequence number generation quality, used for the estimation of attack feasibility and for the Pseudo Random Number Generator function behaviour analysis.

Machine Learning Techniques are accurate approaches for evaluating Netflows or IP flow related data, lass presented in the works for Flow Mining [17], [5]. A sub-domain of Machine Learning Techniques are Kernel methods, which have first been presented in computer science in the 1990's [25]. An advantage of using kernel methods is that complex data can be evaluated without complex calculus techniques. Besides the kernel functions in computer security, classification techniques are commonly used nowadays for the evaluation of complex data [5], [14], [19].

V. CONCLUSION

In this paper, a new approach for the evaluation of Netflow records has been introduced by referring to spatial - temporal aggregation. Various attacks on a real data set, obtained from a large network operator in Luxembourg have been analyzed in this new monitoring tool for validating approach. The contribution of DANAK is twofold. First, a new tool based on spatial aggregation has been implemented for processing large quantities of Netflow records. Second, the evaluation of Netflow records in their quantitative and contextual context, where a new kernel function including a Phase Space Analysis for calculating the similarities between traffic profiles has been presented. It has been shown that scalable and efficient flow anomaly detection is possible and corresponding performance metrics well chosen. As future work is planned to optimize the spatial-temporal method for the traffic profile generation and to find kernel function parameters such that attacks can be observed more easily.

ACKNOWLEDGMENT

We especially acknowledge Prof. T. Duhautpas from Restena Luxembourg for his counsel.

REFERENCES

[1] Arbor Networks, *Peakflow SP: Traffic anomaly detection*. http://www.arbornetworks.com/en/peakflow-sp-traffic-anomaly-detection-4.html, last accessed: 29 March 2011.

[2] P. Bahl, R. Chandra, A. Greenberg and S. Kandula and D.A. Maltz and M. Zhang, *Towards highly reliable Enterprise Network Services via Inference of Multi-level Dependencies*. ACM SIGCOMM'07, Kyoto, Japan, 2007.

[3] L. Breiman, J.H. Friedman, and R.A. Olshen and C.J. Stone, *Classification and Regression Trees*. Belmont, California, Wadsworth International Group, 1984.

[4] D. Braickhoff, A. Wagner and M. May, *FLAME: a flow-level anomaly modeling engine*. Proceedings of the conference on Cyber Security experimentation and test, USENIX Association, San Jose, CA, 2008.

[5] A. Caracas, A. Kind, S. Fussenegger and D. Dechouniotis, *Mining semantic relations using NetFlow*. 3rd IEEE/IFIP International Workshop on Business-Driven IT Management, pp.110-111, Salvador, Brazil, 2008.

[6] K. Cho, R. Kaizaki and A. Kato, *Aguri: An aggregation-based traffic profiler*. QofIS2001, Lecture Notes in Computer Science 2156, pp.222-242, Springer, 2001.

[7] Ed.B. Claise, *Cisco Systems Netflow Services Export Version 9*. http://tools.ietf.org/html/rfc3954, 2004.

[8] N. Duffield, P. Haffner and B. Krishnamurthy and H. Ringberg, *Rule-based Anomaly Detection on IP Flows*. In Proceedings of IEEE INFOCOM, pp. 424 - 432, Rio de Janeiro, Brazil, 2009.

[9] C. Estan, *Building a better NetFlow*. In Proceedings of the conference on Applications, technologies, architectures, and protocols for computer communications, pp. 245 - 256, 2004.

[10] P. Giura and N. Memon, *NetStore: An efficient storage infrastructure for network forensics and monitoring*. In 13th Recent Advances in Intrusion Detection (RAID) Conference, Lecture Notes in Computer Science, Springer, 2010.

[11] G. Hulten, A. Penta and G. Seshadrinathan and M. Mishra, *Trends in Spam Products and Methods*. Proceedings of CEAS, http://ceas.cc/2004/165.pdf, 2004.

[12] R. Kaizaki, K. Cho and O. Nakamura, *Detection of Denial of Service attacks using AGURI*. International Conference Telecommunications ICT2002, Beijing, China, 2002.

[13] T. Karagiannis, K. Papagiannaki and M. Faloutsos, *BLINC: Multilevel Traffic Classification in the Dark*. ACM SIGCOMM'05, Philadelphia, Pennsylvania, USA, 2005.

[14] L. Kahn, M. Awad and B. Thuraisungham, *A new intrusion detection system using support vector machines and hierarchical clustering*. The VLDB Journal, vol.16, iss. 4, pp. 507-521, Springer, 2007.

[15] J.B. Kruskal, *On the shortest spanning subtree of a graph and the traveling salesman problem*. In Proceedings of the American Mathematical Society, vol. 7, pp. 48-50, 1956.

[16] A. Lakhina, M. Crovella and C. Diot, *Mining Anomalies Using Traffic Feature Distributions*. ACM SIGCOMM'05, Philadelphia, Pennsylvania, USA, 2005.

[17] W. Lee, S.J. Stolfo and K.W. Mok, *Mining in a data-flow environment: experience in network intrusion detection*. 5th International Conference on Knowledge Discovery and Data Mining (SIGKDD), pp. 114-124, San Diego, US, 1999.

[18] B.B. Mandelbrot, *The Fractal Geometry of Nature*. W.H. Freeman and Company, NY, 1982.

[19] A. McGregor, M. Hall and P. Lorier and J. Brunskill, *Flow Clustering using machine learning techniques*. Passive and Active Network Measurement, Lecture Notes in Computer Science, Springer, 2004.

[20] D.R. Morrison, *PATRICIA- - Practical Algorithm To Retrieve Infromation Coded in Alphanumeric*. ACM Journal, vol 15, iss. 4, pp. 514-534, 1968.

[21] I. Paredes-Oliva, *Portscan Detection with Sampled NetFlow*. Lecture Notes in Computer Science, vol. 5537, pp. 26-33, 2009.

[22] Y. Rekhter and T. Li, *An Architecture for IP Address Allocation with CIDR*. RFC1518, http://tools.ietf.org/html/rfc1518, September 1993.

[23] B. Schoelkopf, J.C. Platt and J.C. Shawe-Taylor and A.J. Smola and R.C. Williamson, *Estimating the Support of a High-Dimensional Distribution*. Neural Computation, vol. 13, pp..1443-1471, MIT Press, 2001.

[24] F. Takens, *Detecting Strange Attractors in Turbulence*. In Dynamical Systems and Turbulence, Lecture Notes in Mathematics, vol. 898, Berlin, pp. 366-381, 1981.

[25] V. Vapnik, *Statistical Learning Theory*. Wiley-Inderscience, 10-9780471030034, 1998.

[26] C. Wagner, J. François and R. State and T. Engel, *Machine Learning Approach for IP-Flow Record Anomaly Detection*. In IFIP Networking 2011, Lecture Notes in Computer Science , vol. 6640, Springer, 2011.

[27] C. Wagner, G. Wagener and R. State and A. Dulaunoy and T. Engel, *Game Theory driven monitoring of spatial-aggregated IP-Flow records*. 6th International Conference on Network and services Management, Niagara Falls, Ontario, Canada, 2010.

[28] M. Zalewski, *Strange Attractors and TCP/IP Sequence Number Analysis*. http://www.lcamtuf.coredump.cx/tcpseq.html, 2001, last accessed: January 2011.

[29] H. Zang, *Traffic monitor deployment in IP networks*. In Computer Networks: The International Journal of Computer and Telecommunications Networking, vol. 53 , Issue 14, pp. 2491-2501, 2009.