A Comprehensive Model for Revealing Anomaly in Network Data Flow

Maher Salem, Ulrich Buehler

Department of Applied Computer Science
University of Applied Sciences Fulda
Marquardstr 35
36043 Fulda
maher.salem@informatik.hs-fulda.de
u.buehler@informatik-hs-fulda.de

Abstract: Large computer and communication networks lead to the generation of massive data flows. The difficulty of analyzing and managing these data in network security degrades the online detection of intrusion and suspicious connections. To overcome this problem, we present a comprehensive model that handles the traffic of computer networks and uncovers intrusions in real time. The model consists of dataset generator and intrusion detector. The dataset generator captures, analyzes and manages the live traffic using a dynamic queuing concept. It continuously constructs connection vectors from the live traffic and exports them either as datasets or sequentially into a pipe for further processing. The intrusion detector is based on an enhanced growing hierarchical self organizing map which classifies exported vectors to normal, anomaly or unknown connections. The model has been evaluated using synthetic and realistic data sources. It is able to process data flows within significant time and classifies the connections in the online mode effectively.

1 Introduction

The online operation mode of Intrusion Detection System (IDS) has become a real challenge since the amount of generated heterogeneous and non-stationary data and the interconnection between communication networks are increasing rapidly [HB11],[LYW13],[GJP12]. Recent IDS models have been improved to classify network traffic in online operation mode, but the improvement could achieve limited targets such as detecting anomalous traffic in http request [LLZ10] or detecting only DoS attacks [XC07]. However, these models are not able to detect anomaly in networks which generate massive data flows.

Moreover, the detection models suffer from the inability to detect new attacks or attacks with updated signature in real time, because they adhere to the original offline training dataset and they are not superior in recognizing unknown data flow. Therefore, most recent IDSs still struggle to adapt with the current heterogeneous networks. They should professionally be improved so that they can handle massive data flows and provide preciseness in detecting anomalous traffic. More specifically, they should be competent of converting massive data flows to connection vectors on 1 Gbps to 10 Gbps networks. These connection vectors are used for training and evaluating IDS models. Hence, IDS models need a state-of-the-art method to handle the previous challenges. Figure 1 depicts a general diagram of the major steps of an IDS model.

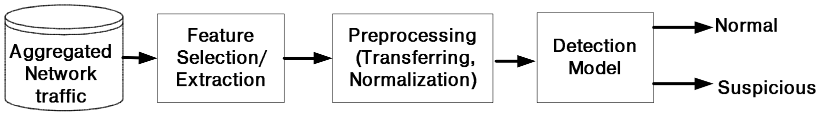

Aggregated Network traffic
Feature Selection/ Extraction
Preprocessing (Transferring, Normalization)
Detection Model
Normal
Suspicious

Figure 1: Major steps of an IDS model

Artificial Neural Networks are considered one of the superior methods in ensuring significance of network management. They have been using in IDS and then become the most efficient method in network security [ZCL11]. Growing Hierarchical Self Organizing Map (GHSOM) is considered as the most successful detection method based on neural network. It is a modified approach from the Kohonen Self Organizing Map (SOM) that can discover knowledge and abstract relations of high-dimensional data and map these data into a two-dimensional representation space [TK95]. GHSOM approach has improved the original SOM by structuring several SOMs in a hierarchical growing form [RMD02].

In this paper, we present a novel method that consists of two major parts: datasets construction using dynamic queuing concept and reinforcing the security over networks using an intelligent intrusion detection model. The first part continuously captures network packets and hosts' events in real time, processes and analyzes them in a queuing concept of dynamic window size, and then constructs sequential connection vectors based on selected features. These features are 22 valuable features, which are accurately selected based on our proposed feature selection method in [SBR11]. The second part is complementary to the first one, which is an enhanced GHSOM detection model (EGHSOM) that is able to uncover known and unknown anomaly from the constructed connection vectors [SB13]. The result of performance evaluation demonstrates the feasibility of the first part of the framework and shows a high performance of the second part.

The rest of this paper is organized as follows. Section 2 discusses previous approaches in the scope of our research. Our proposed method is described in section 3. While, section 4 presents the performance evaluation. Finally, section 5 concludes our work.

2 Related Work

Data flow in computer and communication networks is increasing rapidly which lead to the generation of voluminous amount of data. This issue makes network monitoring and security more complicated and demands more robust frameworks and methods to preserve the management. A framework that addresses this issue, by presenting a graph differential anomaly visualization tool, is proposed in [LS12]. This kind of solution requires a persistent monitoring from the human side and reasonable reaction from the machine side as well. Therefore, it is very difficult to reach a satisfactory trade-off between these two sides.

Generally, IDS approaches often depend on available offline datasets for training and evaluating their models, and ignore the way of aggregating them. However, an acceptable framework (SSENet-2011) that synthetically generates datasets is addressed in [VHS11]. The framework has been installed in a similar simulation environment to that in DARPA intrusion detection evaluation program in 1999 [LYW13]. On the other hand, synthetic datasets cannot represent online traffic; hence, they are not appropriate for online IDS evaluation. In contrast, C. Hsu and S. Wang propose a packet capture module, called Flow Ring, to avoid packet loss and to enhance intrusion detection performance [HW11]. The proposed work consists of flow controller and buffer ring to ease packet capturing and to decrease losses as well. The experimental study demonstrates that the Flow Ring is a promising positive result especially when integrating to Snort IDS [MR99], [SN13] in computer networks.

In this regard, A. Biswas and P. Sinha outperform the Flow Ring method by proposing an efficient packet capturing method to improve the IDS performance [BS06]. The architecture performance is compared with Linux, NAPI, and PFRING against a comprehensive set of criteria. The result indicates that the proposed architecture DMA ring can be used with tcpdump or snort to improve their performance. The architecture can be helpful to dump packets without losses in high-speed networks. A similar and more scalable architecture has also been presented from C. Morariu and B. Stiller [MS08]. Most of the proposed works investigate only packet capturing and overlook other data sources such as hosts events. In this paper, dataset generation will be conducted for network traffic and hosts events in real time.

3 Proposed Method

The proposed method interconnects the above major steps of an IDS model and consists of two parts: The *Dataset Generator* and *Intrusion Detector* as illustrated in figure 2.

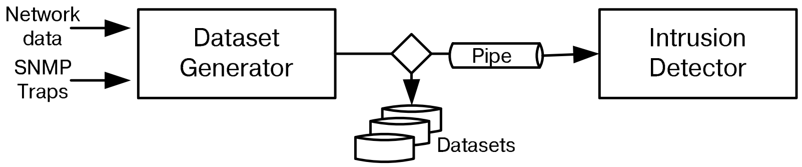
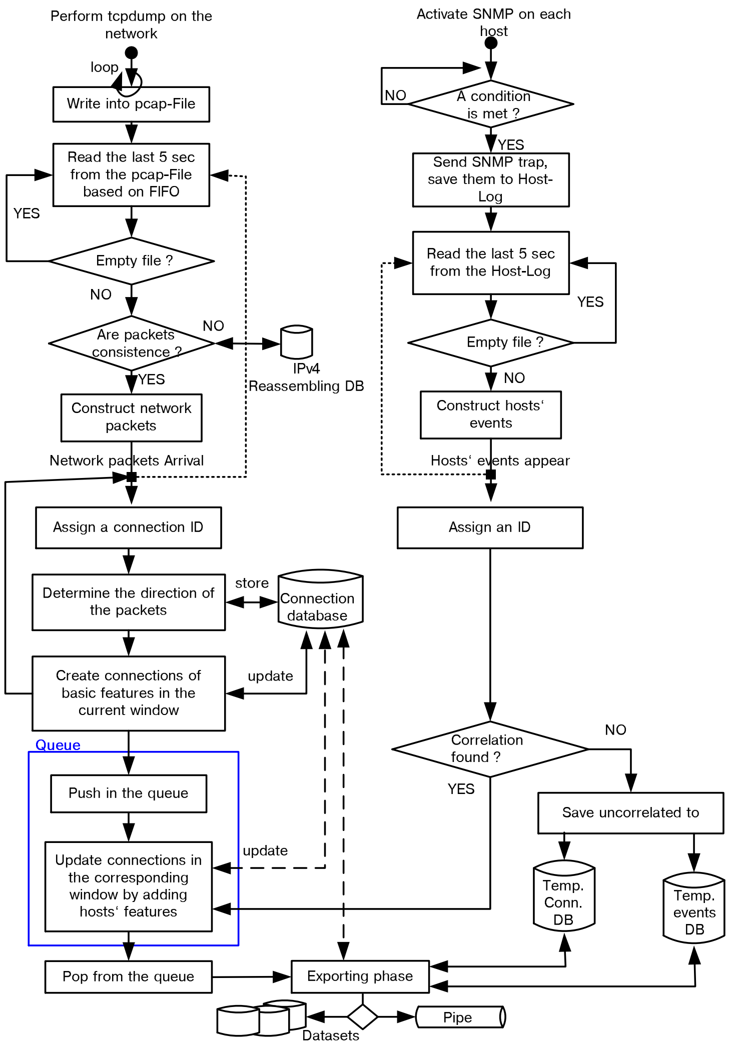


Figure 2: General overview of the comprehensive model

The constructed connection vectors from the *Dataset Generator* represent the link between these two parts. The output of the *Dataset Generator* is used as input for the *Intrusion Detector*. Finally, the classification of the constructed vectors delivers information about the security state of the observed computer and communication network (i.e. normal, anomaly or unknown).

3.1. Dataset Generator

The model addresses network data flows and hosts' activities effectively by using the *Dataset Generator* that analyzes, processes, and correlates them based on a flow diagram as shown in figure 3.



Perform tcpdump on the network
loop
Write into pcap-File
Read the last 5 sec from the pcap-File based on FIFO
YES
Empty file ?
NO
Are packets consistence ?
NO
IPv4 Reassembling DB
YES
Construct network packets
Network packets Arrival
Assign a connection ID
Determine the direction of the packets
store
Connection database
Create connections of basic features in the current window
update
Queue
Push in the queue
Update connections in the corresponding window by adding hosts' features
update
Pop from the queue
Exporting phase
Datasets
Pipe
Activate SNMP on each host
NO
A condition is met ?
YES
Send SNMP trap, save them to Host-Log
Read the last 5 sec from the Host-Log
YES
Empty file ?
NO
Construct hosts' events
Hosts' events appear
Assign an ID
Correlation found ?
NO
YES
Save uncorrelated to
Temp. Conn. DB
Temp. events DB

Figure 3: Flow diagram of the dataset generator

We use benefits of the well-known tool TCPDUMP to aggregate the online massive traffic by mirroring it on a specific port. Moreover, we exploit the SNMP mechanism to send traps from network hosts when a certain activity is matched, e.g. by a failure login of a user. To ease the illustration, we assume that the aggregator handles the dump data and traps into an appropriate time slot, e.g. every 5 seconds as figure 3 shows.

The generator analyzes and processes aggregated dump traffic and constructs corresponding network packets. Similarly, it constructs hosts' events from the received SNMP traps at the server side. It correlates the packets and traps upon ID match to construct connection vectors, based on certain features such as *protocol_type, service, count* [SBR11], inside the queue and export them as datasets.

Specifically, the generator performs the following main steps:

- Capture network traffic and dump it to a certain pcap file. Similarly, receiving SNMP traps from network hosts and saving them to a Host-log file.
- Read continuously from the pcap and Host-log files each certain time slot window (e.g. 5 seconds) based on First Input First Output (FIFO) concept.
- Check the packets consistency. If the packet flow is not consistence, it will be saved in an IPv4 reassembling database, else the generator constructs network packets for the current time slot window and assigns each packet with a unique-ID which is {*Timestamp, Protocol_Type, IP_Src, Port_Src, IP_Dst, Port_Dst*}.
- The generator performs the same procedure on Host-log file for the current time slot window, but without any reassembling because the SNMP traps send only three main users' activities which are logging status of each user, number of log in per user, and if the user uses root shell prompt. Likewise, each host event will be assigned by a unique ID as mention in the previous step.
- For the current time slot, the generator uses the network packets to determine the direction (i.e. from initiator to responder or vise versa) then constructs the corresponding connection from the following features {*timestamp, src_ip, dst_ip, src_port, dst_port, protocol_type, services, src_byte, flag, wrong_fragment*}. It then saves the constructed connections for the current time slot into a connection database and concurrently pushes them in the first window inside the dynamic queue.
- Meanwhile, the generator examines whether any host event and network packet have the same ID for the current time slot. In other word, when the generator pushes the constructed connections inside the queue, it checks the Host-log file if any host events is received. If there are any, the generator examines the ID for the host event and synchronizes it with the same constructed connections and updates the connection vectors of the current window inside the queue. If not, it saves uncorrelated events and packets to a temporary events and connections databases, respectively.
- While the window is still inside the queue, the generator keeps looking for any update in the connection database and adjusts all values of the connection vectors of the concerned window.

- At the end of the queue, the generator pops the window out of the queue and calculates some advanced features [SB14] for each connection vector in the window.
- Finally, the generator exports the connection vectors in a comma separated value (CSV) format either as a dataset or in sequentially in a pipe.

3.2. Intrusion Detector

Constructed connections by the generator can be exported via a pipe concept to the anomaly detector and so they can be classified as normal, suspicious, or unknown in real time. The anomaly detector is based on our enhanced growing hierarchical self organizing map (EGHSOM). Basically, the GHSOM construes high dimensional data on several layers with several maps to explore supplementary details [RMD02]. In the training process, the input vectors are presented to fixed number of neurons on the map using certain number of iterations. The final best matching units on the maps are called the GHSOM model. However, there are several shortcomings on the GHSOM model, which have been addressed on our work [SB13], major enhancements are:

- Meaningful map initialization
- Splitting threshold technique to boost the final topology
- Merging best matching units to robust the final GHSOM model
- Dynamic classification-threshold confidence to detect unknown attacks

Further detailed information about our EGHSOM is available in [SB13]. The main key aspect of the EGHSOM is using a dynamic classification-confidence distance threshold to classify the incoming connection vectors via the pipe as normal, suspicious or unknown. The EGHSOM is able to update its detection model in real time and hence it is able to uncover new attacks. Figure 4 illustrates the *Intrusion Detector* components.

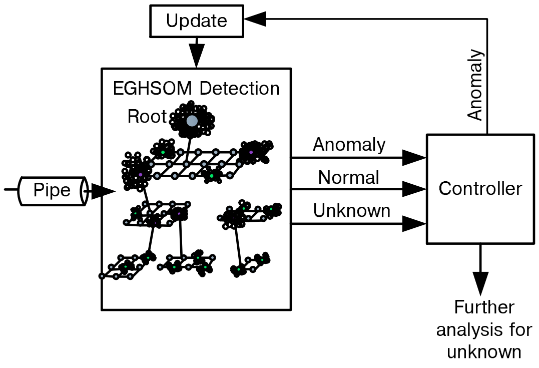


Figure 4: Internal components of the intrusion detector

The detection model classifies the connections and sends the detection result to the controller, which manages and distributes the connections as follows:

- It gathers the anomaly connections and sends them to the update model, which uses them to update the current EGHSOM detection model.
- It sends the unknown connections for further analysis by an expert.

The entire processes of EGHSOM perform the detection and update during the real time, which avoid degrading the network performance and emerging an overhead throughout the detection process.

Both parts of our model operate in the online operational mode in real time. The proposed model aggregates massive data flows of large scale network, analyzes and processes them to construct connection vectors, and sends them via a pipe for classifying the security state of the network system.

4 Performance Evaluation

We have evaluated the proposed model using synthetic and realistic data sources. In the university campus, we have installed a test network as seen in figure 5 and synthetically generated network traffic with a maximum bit rate of 200 Mbps. We have generated the traffic based on realistic network of our industrial partner. We capture the traffic at specific point (the bridge) and use it to evaluate the proposed model. There are several services running on the test network such as HTTP, SSH, FTP, SMTP, Rsync, Telnet.

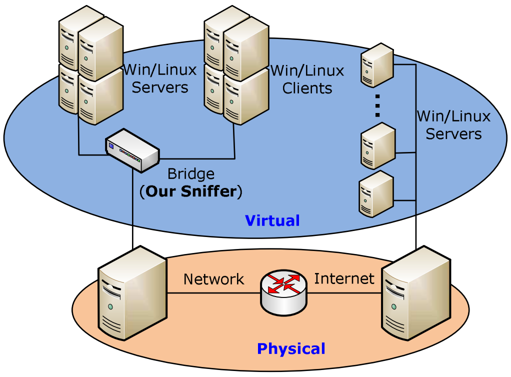


Figure 5: Architecture of the test network

Traffic generation in the test network has been performed using an automotive script. In addition, we have evaluated the model using live network traffic of an industrial firm. We have installed the model on network segmentation which has more than 10 servers and 300 clients connected. We have tried to evaluate the model while almost all users surf and browse the internet, sending emails, do other activities like sending or receiving requests. Results are summarized in the following sections.

4.1. Results of Dataset Generator

We have configured the window time slot in the dataset generator to be 5 seconds. For the gathered packets and events, we have monitored two relevant performance metrics, which are the number of processed packets inside each window and the required time to process these packets and export them as a dataset (connection vectors) based on the exporting phase.

Figure 6 illustrates the number of processed packets inside each window in the dynamic queue for the test network data sources.

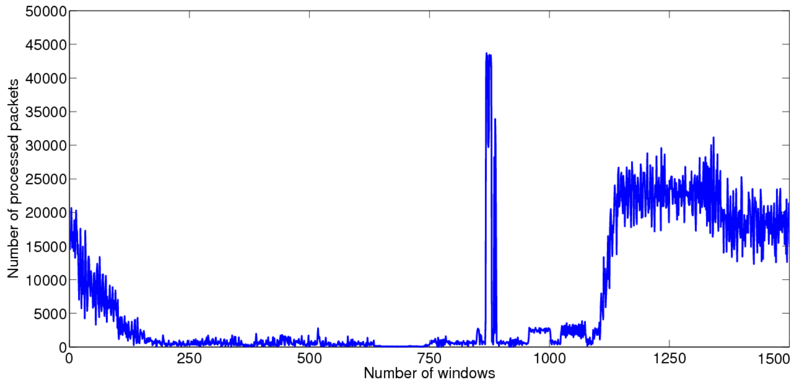


Figure 6: Number of processed packets per window for the test network

Our traffic generator generates network traffic randomly and so the number of processed packets is varying from window to another. Therefore, some windows could processed up to 45000 packets whereas others less than 1000 packets. Each window needs a specific time to process packets and events and then export them accordingly. This time should be not greater than the length of the window itself; otherwise the packet loss will increase rapidly. Figure 7 shows the processing time needed for each window.

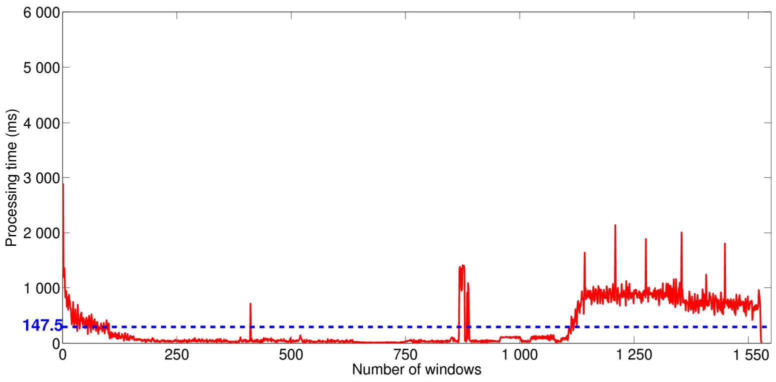


Figure 7: Processing time per window for the test network

The processing time was less than the window length and the packets could be processed without any loss. Similarly, the model could process the live traffic of a realistic network of industrial firm within the length of the window as shown in figures 8 and 9.

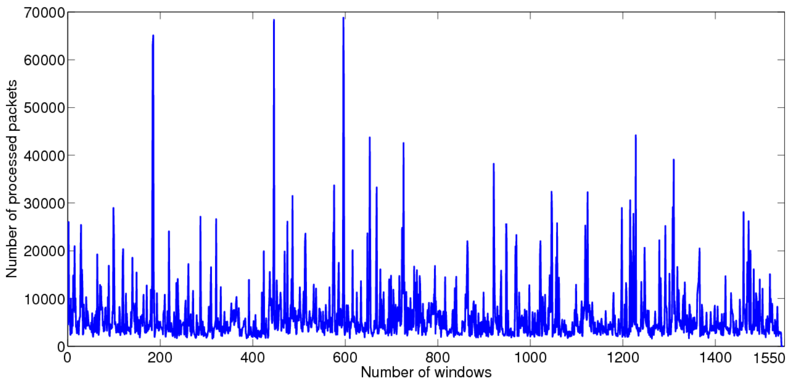


Figure 8: Number of processed packets per window of live traffic

The maximum number of packets has achieved 70000 packets per window within sufficient time as shown in figure 9.

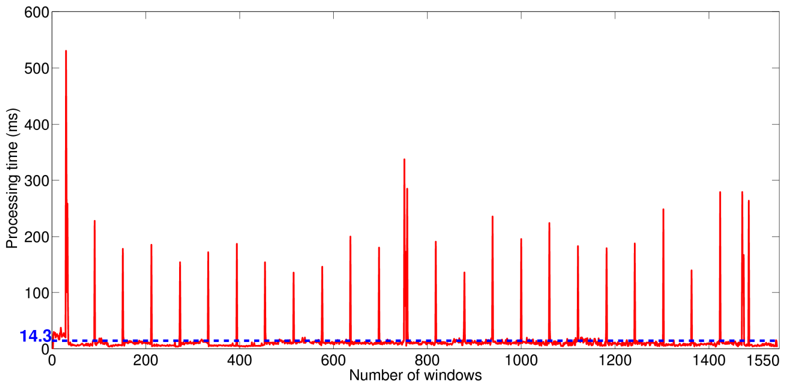


Figure 9: Processing time per window of live traffic

We notice that at the beginning the processing time is a bit more than the window length. This is related to the start phase of the model. If we consider the number of processed packets at the beginning we see that the number is less than 3000 packets; this number means that the model could process these packets surely in less than the window length but as we mentioned beforehand, at the start up of the model can cause some extra processing time.

4.2. Results of the Intrusion Detector

In this section we will briefly show that the detector can effectively classify the exported traffic from the dataset generator. The detector as shown in figure 4 is based on the proposed EGHSOM model [SB13] which is able to classify the constructed connections from the dataset generator to three labels, normal, anomaly and unknown.

We have evaluated the model in a realistic live network. The evaluation parameters can be summarized in table 1.

Characteristic	**Value**
Size of aggregated packets und events	4 GB
Total time of operation	3 hours
Number of constructed connections	303604

Table 1: Parameters of evaluation in real time

The final detection result throughout the evaluation in real time is shown in table 2.

Label of detected connections	**Number of connections**	**Percentage**
Normal	303544	99.9 %
Anomaly	57	0.0188 %
Unknown	3	0.0009 %

Table 2: Evaluation result of EGHSOM in real time

The detector could detect anomaly connections, which most likely related to advertisement servers, as our search confirmed. Other connections could be classified as unknown because they did not fall into the threshold classification margin (see [SB13]). These connections are considered suspicious and need to be further analyzed.

On both cases of synthetic and real time data flows, the model could effectively aggregate, process the traffic and reveal anomaly connections. However, the network speed of both cases was not large enough to proof the feasibility of the model. In the test network we have reached about 250 Mbps although we have installed more than 500 virtual clients on it. The network firm operates on 1 Gbps to 10 Gbps but our model is installed on network segmentation where about 300 users work on their daily activities. However, in the live test, the bit rate was almost 200 Mbps. Therefore, evaluating the model on a bit rate of 1Gbps is still not practically performed.

We have evaluated the proposed model on other synthetic and realistic environments and obtained almost the same result, which states that the generator can effectively aggregate network traffic and construct connection vectors continuously. The detector can precisely classify the constructed connections and update the EGHSOM model appropriately.

5 Conclusion

This paper presents a comprehensive model that consists of two main parts, the dataset generator and the intrusion detector. The former aggregates network packets and hosts' events, processes them and performs a correlation process to construct connection vectors using a dynamic queuing concept, and then it exports these connections in a dataset form or directly into a pipe for the detector. The detector is based on an intelligent EGHSOM model which is able to classify these connections to normal, anomaly or unknown based on a threshold margin. The model has been evaluated on a test network and on a real network firm. It could handle a large amount of packets for each window and effectively reveal anomaly connections. However, in the live test we could not reach a bit rate of 1 Gbps or even 10 Gbps to examine the plausibility, capability and scalability of the model in the high speed networks. This shortage refers to the limited number of users and small number of running services as well. In the future, the model will be enhanced to be able to classify the unknown connections. Moreover, it will be further evaluated on a large scale computer networks up to 10 Gbps.

References

[BS06] A. Biswas and P. Sinha, "On improving performance of Network Intrusion Detection Systems by efficient packet capturing," in *Proc. IEEE/IFIP NOMS*, pp.1-4, April 2006

[GJP12] V. Gulisano, R. Jimenez-Peris, M. Patiño-Martinez,C. Soriente, P. Valduriez, "StreamCloud: An Elastic and Scalable Data Streaming System," in *IEEE Trans. TPDS*, vol.23, no.12, pp.2351-2365, Dec. 2012.

[HB11] A. Hofmann, B. Sick, "Online Intrusion Alert Aggregation with Generative Data Stream Modeling," in *IEEE Trans. TDSC*, VOL. 8, NO. 2, pp. 282-294, MARCH-APRIL 2011.

[HW11] C. Hsu and S. Wang, "Embedded Network Intrusion Detection Systems with a Multi-core Aware Packet Capture Module," presented in *IEEE ICPP*, pp.207-213, Sept. 2011.

[LLZ10] Li Lin, C. Leckie, C. Zhou, "Comparative Analysis of HTTP Anomaly Detection Algorithms: DFA vs N-Grams," International Conference on Network and System Security (*NSS*), pp.113-119, Sept. 2010.

[LS12] Q. Liao and A Striegel, "Intelligent network management using graph differential anomaly visualization," in *Proc. IEEE/IFIP NOMS*, pp.1008-1014, 2012.

[LYW13] Y. Lee, Y. Yeh, Y. F. Wang, "Anomaly Detection via Online Oversampling Principal Component Analysis," in *IEEE Trans. TKDE*, pp. 1460-1470, VOL. 25, NO. 7, JULY 2013.

[MR99] M. Roesch, "SNORT-Lightweight Intrusion Detection For Networks," in *Proc. Of USENIX LISA* '99, pp. 229-238, 1999, Seattle, WA.

[MS08] C. Morariu and B. Stiller, "DiCAP: Distributed Packet Capturing architecture for high-speed network links," presented in IEEE Conf. on Local Computer Networks, pp.168-175, Oct. 2008.

[RMD02] A. Rauber, D. Merkl, M. Dittenbach, "The growing hierarchical self organizing map: exploratory analysis of high-dimensional data," in *IEEE Trans. TNNLS*, Vol. 13, Is. 6, pp. 1331-1341. Nov. 2002.

[SB13] M. Salem and U. Buehler, "An Enhanced GHSOM for IDS," in *Proc. IEEE SMC:Cybernetics*, Manchester, UK, 2013, pp. 1138-1143.

[SB14] M. Salem and U. Buehler, "Persistent Dataset Generation using Real time Operative Framework," in *IEEE Workshop on Computing, Networking and Communications*, Honolulu, Hawaii, USA, 2014, pp. 1023-1027.

[SBR11] M. Salem, U.Buehler, S. Reissmann, "Improved feature selection method using SBS-IG-Plus," in *Proc. of the ISSE Symposium*, Prague, Czech Republic, 2011, pp 352–36.

[SN13] Snort Network Intrusion Prevention and Detection System. Avaliable [Online] http://www.snort.org. Accessed 15 March 2014.

[TK95] T. Kohonen, Self-Organizing Maps. Berlin, Germany: Springer- Verlag, 1995.

[VHS11] A.R. Vasudevan, E. Harshini, and S. Selvakumar, "SSENet-2011: a Network Intrusion Detection System Dataset and its Comparison with KDD CUP 99 Dataset," presented in *AH-ICI*, Kathmandu, Nepal, 2011, pp 1-5.

[XC07] X. Chun, et al. "Research of DoS Intrusion Real time Detection Based on Danger Theory," Presented in the Symposium Data, Privacy, and E-Commerce, *ISDPE*, Nov. 2007, pp.209-211.

[ZCL11] J. Zhao, M. Chen, and Q. Luo, "Research of Intrusion Detection System Based on Neural Networks," in *IEEE-ICCSN*, Xi'an, China, May 2011, pp. 174-178.