Combining statistical network data, probabilistic neural networks and the computational power of GPUs for anomaly detection in computer networks

Sascha Bastke and **Mathias Deml** and **Sebastian Schmidt**
Institut for Internet-Security
University of Applied Sciences Gelsenkirchen
{bastke, deml, schmidt}@internet-sicherheit.de

Abstract

To improve anomaly detection in fast computer networks we analyze the usage of Graphics Processing Units (GPU) to detect attacks and threats. We operate on statistical network data to protect privacy rights and to reduce the amount of collected data. Because of the statistical nature of the data we use an anomaly detection based method to detect threats and attacks.To get a detailed overview of the network we monitor a large number of network parameters at many different sensor locations. To achieve this we propose a solution using GPU's. For the implementation we focus on probabilistic neural networks which can be massively parallelized. This short paper presents ideas and first results of this approach.

1. Introduction

Since the birth of the internet the bandwidth has grown from year to year. Also the number of attacks and threats has grown so that network monitoring and intrusion detection has become more important and challenging. For this reason the process of detecting threats and attacks consumes more computational power to analyze the data. One approach to deal with this problem is mentioned in (Hesse and Pohlmann 2008). It describes a method collecting statistical network data concerning privacy rights and the growth of network speed. Although the amount of data was reduced, there is still a huge number of network parameters to monitor in real-time at many different sensor locations to get a detailed overview. In this paper we analyze the usage of Graphic Processing Units (GPU) to improve the detection process.

Because of the statistical nature of the data we use an anomaly detection based method to detect threats and attacks. For the implementation on the GPU we focus on probabilistic neural networks which can be massively parallelized.

The remainder of this paper is organized as follows: Section 2. gives an overview of related work. After this the used sensor technology is introduced. Section 4. describes the statistical method used for the anomaly detection. Then in section 5. the implementation on the GPU is discussed. The last two sections provide first results, a conclusion and further work.

2. Related Work

The computational power of GPUs is used in serveral domains. One example in the field of network monitoring is speeding up the misuse detection process for observing fast networks as mentioned in (Vasiliadis et al. 2008), (Smith et al. 2009) and (Huang et al. 2008). Similar works concerning GPUs and anomaly detection is not known to us so far.

There are many different methods for anomaly detection in network traffic. The use of neural networks is described in (Ryan, Lin, and Miikkulainen 1998) and (Zanero). Time series approaches were shown in (Throttan and Ji 1998) and (Basu, Klivansky, and Mukherjee 1996). An approach that uses Support Vector Machines can be found in (Mukkamala, Janoski, and Sung).

A solution that works on data similar to ours is implemented in the open source project rrd to detect anomalies in network parameters like bandwidth measured by the tool ntop. This approach called “Abberrant Behaviour Detection” and is described in (Brutlag 2000) .

Two papers that discuss the implementation of neural networks on GPUs are (Luo, Liu, and Wu 2005) and (Jang, Park, and Jung 2008).

3. Internet-Analysis-System Sensors

The sensor-technology of the Internet-Analysis-System (see (Hesse and Pohlmann 2008)), called IAS-Probe, is able to provide a continuous view of the traffic behaviour at the sensor location. It can be placed in all IP-based communication infrastructures ranging from local home networks over corporate networks up to the level of autonomous systems. It uses a statistical approach to describe the complete network behaviour. Figure 1 shows how this is done.

The figure is divided up into three sections. (i) The Internet is represented on the left. In this example packets of three different application sessions are shown: a HTTP session, a FTP session and a SMTP session. (ii) The probe is located in the middle of Fig. 1. The packets of the three applications are evaluated by the probe one after another in their random order. They are channelled through several analysis modules, which are responsible for different protocols. These evaluate strictly defined communication parameters in the protocol header at the various communication levels, which are not concerned by data protection laws. (iii) The

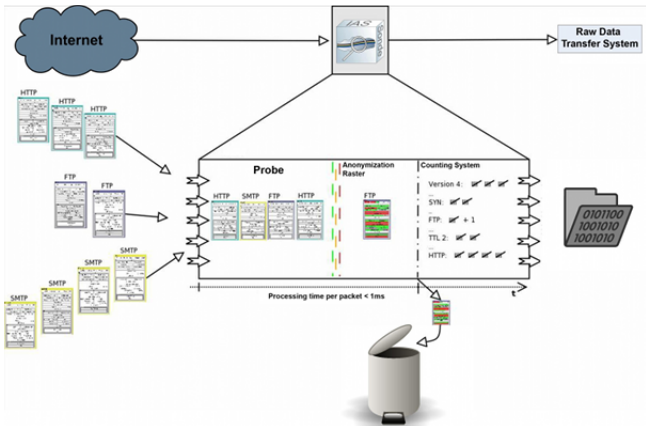


Figure 1: Raw data packet capturing

counters, in the Internet-Analysis-System called descriptors, allocated in the counting system are incremented according the header information of the packet. The frequency of certain header information is recorded in the same way as on a tally sheet. Let us take a look at the following example: In Fig. 1 we see the analysis of a FTP packet, which results in an incrementation of the FTP descriptor by one. The raw data is an aggregation of counters, i.e. counters of communication parameters that have appeared at the various communication levels over a defined period. The packet - in Fig. 1 a FTP packet - is immediately deleted physically, i.e. irreversibly and without trace, by the probe (Pohlmann 2007) . Protocols monitored by the sensors are for instance : IP, ICMP, TCP, UDP, HTTP, SMTP, FTP, DNS, EMULE, IRC, RTP, SIP, Skype, etc (Ricci 2008) . For each protocol a different number of monitored parameters is implemented in the sensor technology and observed by it. For example we count 1,624 for SMTP, 1,123 for HTTP, more than 9,000 for DNS, only 47 for RTP and more than 500,000 different parameters for TCP. The number for TCP is that high, because it includes the observation of various ports as well. These counters sum up to a total of about 1,300,000 different parameters, occurrence is registered by the sensor (Ricci 2008). Another way to explain the collection process more formal is a finite state machine (see (Bastke 2009)).

$$A = (\Sigma, S, \delta, s_0, F) \tag{1}$$

Σ is the input alphabet. In case of the IAS-Probe it is the set of all possible packets which could be sent over a network. The set of the possible states of the machine is characterized by S. Figure 2 shows a subset of states. Every state represents one possible assignment of counters. That means that one state s_i is described by $s_i = (d_1, d_2, \ldots, d_n)$. In which d_j represents a descriptor. Every packet analyzed by the probe causes a transition to another state. The state-transition function δ tells to which state a transition occurs, based on the actual state and input value . This specifies the counting functionality of the probe. The initial state for one sample interval is $s_0 = (0, 0, \ldots, 0)$. F is the set of final states.

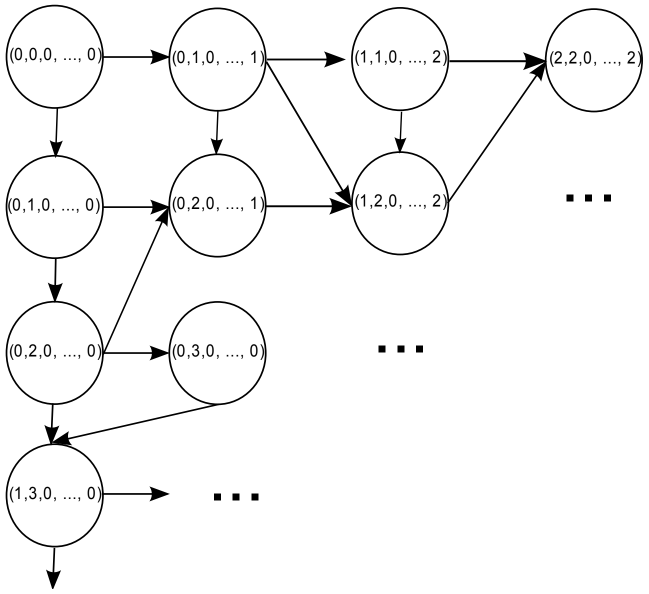


Figure 2: Subset of states

At the end of every sample interval the probe sends the actual state to the raw data transfer system and resets the state to s_0 again.

4. Approach to anomaly detection

Anomaly detection tries to describe the normal behaviour of a network and to identify attacks or failures of the network based on divergences from this normal behaviour, which is described by a model. The predictions taken from the model will be compared to the real behaviour of the network. If divergences are found, an anomaly is detected. Let M be the model of the network behaviour and R be the real network behaviour. If the difference between model M and real behaviour R exceeds an threshold ϵ, an anomaly is detected. This is expressed in equation 2.

$$|M - R| > \epsilon \tag{2}$$

Based on the description of the data which is collected by the IAS sensor technology (see section 3), we can identify two methods for anomaly detection [1].

Time Series modelling The descriptor data over the time can be viewed as time series. This can be modelled with methods from the field of time series analysis. One example is an ARIMA model respectively AR model that is used in (Throttan and Ji 1998) and (Basu, Klivansky, and Mukherjee 1996).

Feature Vector based Detection This approach is based on the use of feature vectors. Similar to pattern recognition, a feature vector $\vec{v} = \{d_1, d_2, \cdots, d_n\}$ composed of descriptors d_i is used. One example for analysing this feature vectors is to estimate the distribution $p(\vec{v})$. Based

[1] More methods can be identified, but they are less important for this work.

on the estimated probability distribution, anomalous values can be identified. Examples for the use of this kind of methods are given in (Nguyen 2002) and (Mukkamala, Janoski, and Sung).

In the remainder of this work we use a statistical approach for anomaly detection. For this we estimate the probability distribution of the parameters. In the literature different approaches for estimation of probability density are known. They can be divided in parametric and non parametric approaches. In case of an parametric approach the form of density is known and only the specific parameter for this density function must be estimated. An example is the assumption of a normal distribution and the estimation of the parameters for mean and variance. If no assumption about the form of density can be made, a non parametric approach must be used so that the structure can be estimated. One example is the parzen window method (see (Duda, Hart, and Stork 2000)). This method uses a set of example data to estimate the probability density by superimposing kernel functions, which are adjusted at the example data. Let $X = \{x_1, x_2, \cdots, x_N\}$ be a set of independent and identical-distributed samples. Then the approximation of density is given by equation 3:

$$p(x) = \frac{1}{Nh}\sum_{i=1}^{N} K(\frac{x - x_i}{h}) \quad (3)$$

In this equation K is the kernel function which is used for approximation. Often a standard gaussian function with zero mean and unit variance is used. The parameter h is a smoothing parameter called bandwith. An example for an approximation of a density function is shown in figure 3.

This method can be used for anomaly detection on the collected data. For this we combine descriptors to feature vectors of the form $\vec{X} = \{d_1, d_2, \cdots, d_k\}$. Which descriptors should be combined is out of scope of this article.

Let E_i be a specific feature vector configuration. For every configuration E_i a set of training data $T_i = \{\vec{X}_1, \vec{X}_2, \cdots, \vec{X}_N\}$ must be collected which reflects the behaviour of the data. Based on this training data we can use equation 3 to approximate the density function.

The approximated density is the model of data that can be used for the decision whether the actual data point $\vec{X}$, measured by the sensor, is normal or anomalous. The actual data point is used as input to equation 3. If the result value is high, we can guess with high probability that the actual point is normal. The closer the result value tends to zero, the higher the probability that the actual measured value is anomalous. To get a decision boundary we must define a threshold value TH. This value can be given by a user or determined automatically.

$$p(\vec{X}) < TH \quad (4)$$

The distribution of data can change over time. For this reason it is important to update the training data for the computation of density function regularly. We can use a time window, from which we take the training data. Let t be the actual point in time, then the interval from which we take

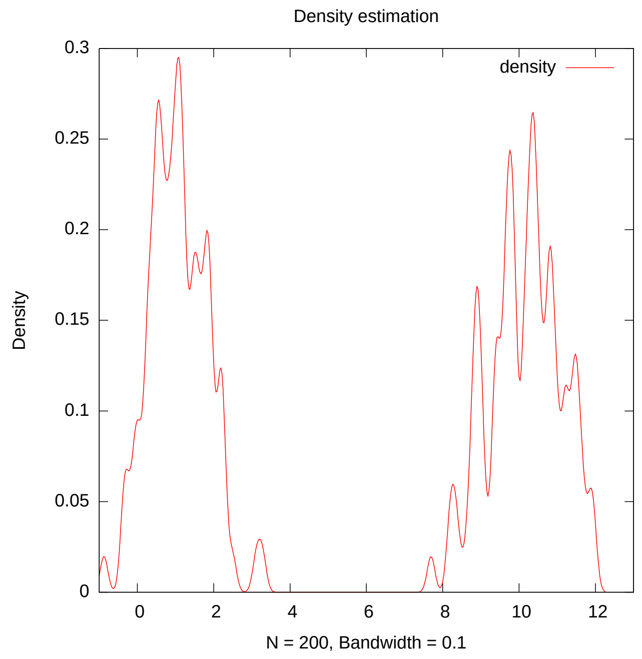


Figure 3: Density estimation for two overlapped normal distributions.

the training data is defined by $(t - L, t - 1)$. It follows that the interval is moved one step forward each time, so that the data for model identification is up to date. In this case the parameter L is important, which defines the length of the interval. If this parameter is too small, the data is not representative. if L is too large, the model adopts very slowly to changes.

Finally a simple method for identifying the threshold TH should be described. Let $G = \{g_1, g_2, \cdots, g_V\}$ be a set of result values calculated with equation 3. A value for the threshold TH can than be calculated by equation 5. This is the rule for identifying outliers based on boxplots. For this we must calculate the quantiles Q_{25} and Q_{75} of G.

$$TH = Q_{25} - 1.5|Q_{75} - Q_{25}| \quad (5)$$

5. Using GPU computation power through Probabilistic Neural Networks

The approach shown in the last section allows a detailed statistical description of data which is collected by sensors, but it is relatively expensive because of the high computational power needed. The usage of modern Graphics Processing Units (GPU), that can reach up to one Teraflop, seems to be a way to deal with this problem. The usage of this GPUs should allow an efficient implementation of analysis methods at relative low cost that can work effectively even in fast computer networks.

Today's Graphic Processing Units (GPU) work massive parallel based on a SIMD-Model (Single Instruction Multiple Data). They execute tha same commands on thousands

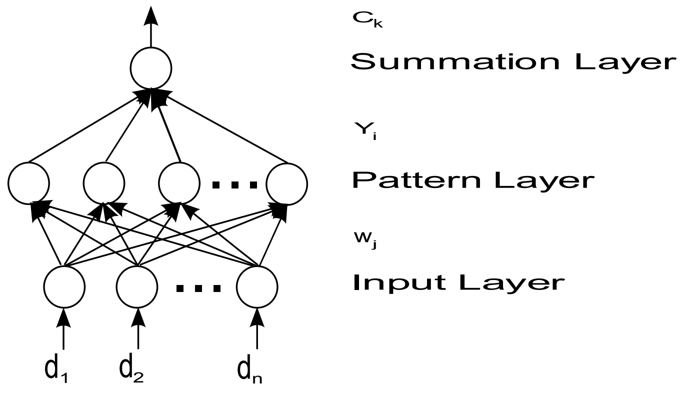


Figure 4: Probabilistic Neural Network

of datasets in parallel. This description discard many details. You should only become an idea.

To use the computational power of the GPU we must parallelize the approach shown in section 4. A parallelization of this approach is Probabilistic Neural Network (see (Duda, Hart, and Stork 2000)) which is describe in detail in the remainder of this section.

A Probabilistic Neural Network (PNN) consists of four layers. One is the input layer where the actual input vector is given. The second one is the pattern layer with one neuron for every training example. Input layer and pattern layer are fully connected. The next one is the summation layer, where the results of the pattern layer will be added . In this layer we have one neuron for each class we want to distinguish. The neurons of the pattern layer are connected to the neurons of the summation layer based on the class of the neuron in the pattern layer. So every neuron in the pattern layer of the same class is connected to the same neuron in the summation layer. The last layer is the output layer where the estimated class of the input data is shown. This is realized by an argmax operator over the outputs of the summation layer. For the use in this article we have no output layer and only one neuron in the summation layer. This structure is shown in figure 4.

The training process of a PNN is very easy. For each training vector in the training data set we create a neuron in the pattern layer. The weights ω_j of this neuron are set to the values of the training vector. After this the PNN is trained and can be used. The processing of an input vector $\vec{x}$ can be described as follows:

- The input vector $\vec{x}$ is given to the input layer and then processed by the neurons of the pattern layer. Every neuron does the calculation in equation 6 to calculate the output value y_i.

$$y_i = e^{-\frac{(\vec{x}-\vec{w_i})^T(\vec{x}-\vec{w_i})}{2\sigma^2}} \tag{6}$$

- The output values y_i will be transferred to the summation layer and the output value c_k will be calculated by equation 7.

$$c_k = \frac{\sum_{y_i \in K_k} y_i}{|K_k|} \tag{7}$$

The expression $|K_k|$ is the cardinality of the training set. This is the equivalent of N from equation 3. The values of the neurons of the pattern layer are averaged.

- In the last step we identify the maximum value of c_k, so we know the class of the input vector[2].

$$\text{argmax } c_k \tag{8}$$

This structure can be implemented in parallel on the GPU. A simple approach is to execute every neuron in the pattern layer in a thread. These threads execute in parallel and the results must be put together. In pseudo code this is:

1. Load the training set of cardinality N to GPU memory.
2. Load the input vector to GPU memory
3. Execute calculation of equation 6 N times parallel
4. Execute the calculation of equation 7
5. Copy back result to CPU memory
6. Compare result to threshold TH

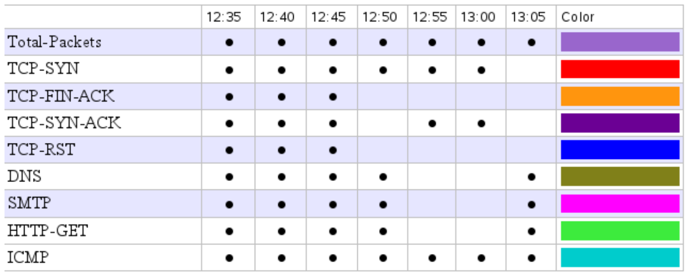
Another approach for the parallelization is to process a set of PNNs in parallel on the GPU. Instead of executing only one PNN on the GPU we can execute Z in parallel. The value of Z is only limited by the memory of the GPU.

6. Some Results

The algorithms, explained in section 4, are able to reveal a number of attacks, which are common for the internet. Some examples of these attacks are scans, dictionary attacks against different services, internet worms and spam waves. Especially the execution and impact of distributed denial-of-service attacks (DDoS attacks) which are subject of the following example are easy to detect. The data of this example was collected during a real DDoS attack towards a network, monitored by an IAS-Probe. The attack started with a significant increase of the amount of TCP-SYN packets and ICMP-Echo-Requests. This led to the conclusion, that the attack was a ping flood combined with a syn flood. The increase in the measurands of the monitored descriptors resulted into a value of zero for c_k of the PNN, in a time frame of one sample interval. Therefore the algorithms provided a warning with a reaction interval of at least five minutes which could be used for counteractive measures before the the attacked systems wouldn't have been reachable any more. This decrease generated another series of anomalies which allowed to trace the impacts of the DDoS-attack. Figure 5 shows the measurands in logarithmic scale during the attack and provides a list with anomalies which occurred in the different intervals for the set of the most important descriptors or descriptor vectors in this example.

Another example acts as a demonstration for the monitoring of company policies with the introduced algorithms

[2]Not used in this application.



	12:35	12:40	12:45	12:50	12:55	13:00	13:05	Color
Total-Packets	●	●	●	●	●	●	●	🟪
TCP-SYN	●	●	●	●	●	●		🟥
TCP-FIN-ACK	●	●	●					🟧
TCP-SYN-ACK	●	●	●		●	●		🟪
TCP-RST	●	●	●					🟦
DNS	●	●	●	●			●	🟫
SMTP	●	●	●	●			●	🟪
HTTP-GET	●	●	●	●			●	🟩
ICMP	●	●	●	●	●	●	●	🟦

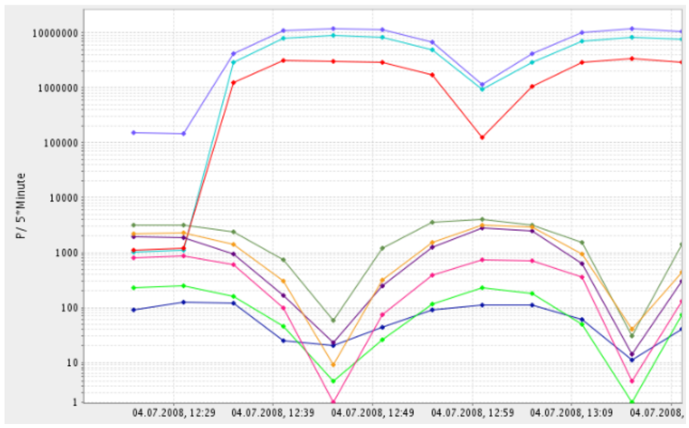


Figure 5: DDoS events

and probes. An anomaly in the total amount of measured network packets was led back by the involved analysts to a traffic-increase on port 15000 of the UDP-protocol. With the help of descriptors, describing the udp-length-field, the volume of the traffic increase was approximated to 4.2 GByte which is nearly the size of a DVD-5. In further investigations we found out that this port is used by the P2P file-sharing client Thunder Network prevalently used in china and combined with malware in many cases. Figure 6 shows the increase of the involved descriptor measurands for the probes monitoring the inbound and outbound traffic.

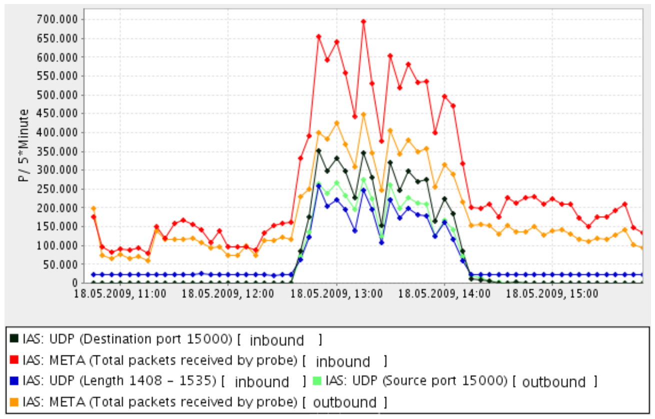


Figure 6: UDP events

7. Conclusions and Further Work

Many tests demonstrated that the introduced methods can be used to detect anomalies and problems in network traffic at different probe locations for the domains of network management and network security. In combination with expert knowledge in these domains the anomalies provide useful hints to running attacks, compromised systems in local-networks, invalidly configured services, other general network problems or violations of policies. The combination of statistical network data, selflearning algorithms and computational power of GPUs causes the system to adapt independently to different environments and enables it to reveal problems which are not detected by current intrusion detection or network monitoring systems.

In further work we are going to analyze the strengths and weaknesses of the collected data and detection algorithm for different kinds of threats and attacks. To achieve this we will develop a set of tagged test data containing common attacks and threats to local networks. To achieve a highly realistic result and to measure the false-positive rate it is necessary to include normal network traffic. First results have revealed two problems. The first problem is related to the sample interval of five minutes which turned out to be too long because even global botnet actions can be completed within minutes (Li et al. 2005) . The decrease of the sample interval with simultaneous increase of computation capacity of the system through the usage of GPUs or the deployment of the algorithms on the probes for realtime monitoring are only two of many possible approaches. The second problem results from the current circumstance that the probe does not analyze the payload of the application layer because of anonymization reasons. Especially for the detectoion of exploits this is a problem, because in most cases they are contained in this part of the packet. An extension of the probe which is able to model the normal structure of different application layer payloads with methods like the one described in (shan Zhao et al. 2008) could help to fill this gap.

Another remaining aspect is the usage of the GPU to improve the detection process. For this different questions must be answered. First we must analyse how we can port the detection algorithm to the GPU in an efficient way. If there are different ways, what kind of advantages and disadvanteges can be identified for them. By imlementing the code there could be discovered different issues like a slow transfer between graphics card memory and host memory.

References

Bastke, S. 2009. Beschreibung des Netzwerkverkehrs unter Nutzung von Markow-Ketten. In *DACH Security 2009*.

Basu, S.; Klivansky, S.; and Mukherjee, A. 1996. Time series models for internet traffic.

Brutlag, J. D. 2000. Aberrant behavior detection in time series for network monitoring. In *LISA*, 139–146.

Duda, R. O.; Hart, P. E.; and Stork, D. G. 2000. *Pattern Classification (2nd Edition)*. Wiley-Interscience.

Hesse, M., and Pohlmann, N. 2008. Internet situation awareness. In *eCrime Researchers Summit, 2008*, 1–9.

Huang, N.-F.; Hung, H.-W.; Lai, S.-H.; Chu, Y.-M.; and Tsai, W.-Y. 2008. A gpu-based multiple-pattern matching algorithm for network intrusion detection systems. In *Advanced Information Networking and Applications - Workshops, 2008. AINAW 2008. 22nd International Conference on*, 62–67.

Jang, H.; Park, A.; and Jung, K. 2008. Neural network implementation using cuda and openmp. In *Computing: Techniques and Applications, 2008. DICTA '08.Digital Image*, 155–161.

Li, J.; Ehrenkranz, T.; Kuenning, G.; and Reiher, P. 2005. Simulation and analysis on the resiliency and efficiency of malnets. In *PADS '05: Proceedings of the 19th Workshop on Principles of Advanced and Distributed Simulation*, 262–269. Washington, DC, USA: IEEE Computer Society.

Luo, Z.; Liu, H.; and Wu, X. 2005. Artificial neural network computation on graphic process unit. In *Neural Networks, 2005. IJCNN '05. Proceedings. 2005 IEEE International Joint Conference on*, volume 1, 622–626 vol. 1.

Mukkamala, S.; Janoski, G.; and Sung, A. Intrusion Detection: Support Vector Machines and Neural Networks. In *Smart Engineering System Design: Neural Networks, Fuzzy Logic, Evolutionary Programming, Data Mining and Complex Systems*.

Nguyen, B. V. 2002. Self-Organizing Map (SOM) for Anomaly Detection. Advanced Topics in Artificial Intelligence. Spring.

Pohlmann, N. 2007. Probe-based internet early warning system. *ENISA Quarterly*.

Ricci, G. 2008. Evaluation of the relevance for the detection of abnormalities and attacks of the communication parameters collected by the internet analysis system. Master's thesis, University of Applied Sciences Gelsenkirchen.

Ryan, J.; Lin, M.-J.; and Miikkulainen, R. 1998. Intrusion Detection with Neural Networks. In Jordan, M. I.; Kearns, M. J.; and Solla, S. A., eds., *Advances in Neural Information Processing Systems*, volume 10. The MIT Press.

shan Zhao, T.; zhi Li, Z.; ming Wang, Z.; and fen Lin, X. 2008. An approach to model normal network behaviors based on entire network packets. In *ICNSC*, 1405–1408. IEEE.

Smith, R.; Goyal, N.; Ormont, J.; Sankaralingam, K.; and Estan, C. 2009. Evaluating gpus for network packet signature matching. In *Performance Analysis of Systems and Software, 2009. ISPASS 2009. IEEE International Symposium on*, 175–184.

Throttan, M., and Ji, C. 1998. Adaptive Thresholding for Proactive Network Problem Detection.

Vasiliadis, G.; Antonatos, S.; Polychronakis, M.; Markatos, E. P.; and Ioannidis, S. 2008. Gnort: High performance network intrusion detection using graphics processors. In *RAID '08: Proceedings of the 11th international symposium on Recent Advances in Intrusion Detection*, 116–134. Berlin, Heidelberg: Springer-Verlag.

Zanero, S. Analyzing TCP Traffic Patterns Using Self Organizing Maps.