Inductive Intrusion Detection in Flow-Based Network Data using One-Class Support Vector Machines

Philipp Winter, Eckehard Hermann, Markus Zeilinger
Department of Secure Information Systems
Upper Austria University of Applied Sciences
4232 Hagenberg / Softwarepark 11, Austria
{philipp.winter, eckehard.hermann, markus.zeilinger}@fh-hagenberg.at

***Abstract*—Despite extensive research effort, ordinary anomaly detection systems still suffer from serious drawbacks such as high false alarm rates due to the enormous variety of network traffic. Also, increasingly fast network speeds pose performance problems to systems which base upon deep packet inspection.**

In this paper, we address these problems by proposing a novel inductive network intrusion detection system. The system operates on lightweight network flows and uses One-Class Support Vector Machines for analysis. In contrast to traditional anomaly detection systems, the system is trained with malicious rather than with benign network data. The system is suited for the load of large-scale networks and is less affected by typical problems of ordinary anomaly detection systems.

Evaluations brought satisfying results which indicate that the proposed approach is interesting for further research and perfectly complements traditional signature-based intrusion detection systems.

***Keywords*-network intrusion detection; machine learning; support vector machine; netflow;**

I. Introduction

In the area of network intrusion detection, the research community usually focuses on either misuse or anomaly detection systems. While the former is meant to detect precisely specified attack signatures, the latter is supposed to detect patterns deviating from normal network operations. Both concepts exhibit disadvantages. Misuse detection systems struggle with steadily increasing network speeds and attack signatures while anomaly detection systems suffer from high false alarm rates and the lack of representative training data, just to name a few.

The network intrusion detection system (NIDS) proposed in this paper embraces concepts of both worlds. It operates on network flows rather than on entire network packets. Incoming flows are analysed using One-Class Support Vector Machines (OC-SVM). The learning algorithm is trained solely with malicious rather than with benign data. By this means, the NIDS is supposed to recognize previously learned attacks including attack variations instead of detecting anomalies. The proposed concept is entitled “inductive NIDS”.

The remainder of this paper is organized as follows. Section II gives a brief overview of similar contributions. Section III introduces the proposed approach. Section IV describes the process of model and feature selection while Section V evaluates the proposed approach and discusses the results. Finally, Section VI provides a conclusion and discusses future work.

II. Related Work

Gao and Chen designed and developed a flow-based intrusion detection system in [1]. Karasaridis et al. [2], Shahrestani et al. [3] and Livadas et al. [4] proposed a concept for the detection of botnets in network flows. Finally, in [5] Sperotto et al. provided a comprehensive survey about current research in the domain of flow-based network intrusion detection.

A sound evaluation of a NIDS is a nontrivial task and requires high-quality training and testing sets. Unfortunately, the de facto standard is still the DARPA data set created by Lippmann et al. in [6]. Despite its severe weaknesses and the critique published by McHugh [7] it is still used. The KDD Cup ’99 data set can be regarded as another popular data set [8]. Finally, Sperotto et al. contributed the first labeled flow-based data set [9] intended for evaluating and training network intrusion detection systems. This data set is used in this paper.

Finally, in [10] Gates and Taylor examine whether the established anomaly detection paradigms are still valid. Similar work is contributed by Sommer and Paxson in [11]. They point out why anomaly detection systems are hardly used in operational networks.

III. Proposed Approach

We decided to train the NIDS with malicious network data only. This approach is diametric to the way ordinary anomaly detection systems work. As pointed out in [11], machine learning methods perform better at recognizing inliers than at detecting outliers. Further advantages of this approach are discussed in Section V-B.

In short, the proposed NIDS receives network flows and analyzes them with a OC-SVM. The following two sections briefly introduce the concepts behind network flows and OC-SVMs respectively.

A. Network flows

A set of unidirectional network packets sharing certain characteristics together form a so called network flow. This set

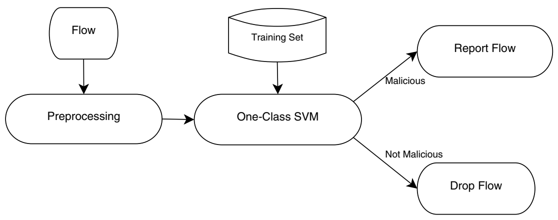

Fig. 1. The high-level view on the proposed approach for an inductive NIDS.

of characteristics is defined as $\{IP_{src}, IP_{dst}, Port_{src}, Port_{dst}, \textit{IP Protocol}\}$, i.e., source and destination IP addresses, ports and the IP protocol number. Additional information can be derived such as the amount of packets or bytes transferred in a flow. All this information is summed up to form a flow record (often just referred to as flow) consisting of several flow attributes. This flow record is then sent from a router to a so called flow collector. Network flows only provide meta information and do not carry any packet payload. We decided to use NetFlow in version 5, the most popular and widespread protocol for dealing with network flows.

Several reasons led to the decision of working with network flows: The highly lightweight nature, the broad availability (only a NetFlow-enabled router is necessary) and the possibility to analyze even encrypted network traffic. A more comprehensive discussion of the advantages of flow-based network intrusion detection is provided in [5].

B. One-Class Support Vector Machines

OC-SVMs as proposed by Schölkopf et al. [12] are an unsupervised learning method. Roughly speaking, unsupervised learning methods are aware of only a single class of data: OC-SVMs distinguish between vectors which are referred to as either in-class (inside the trained distribution) or outliers (outside the distribution). In the context of network intrusion detection, a OC-SVM as used in this paper distinguishes between "malicious" (in-class) and "not malicious" (outlier) network data. We decided to use the OC-SVM implementation provided by the popular library libsvm [13].

The choice fell on OC-SVMs for the following reasons: First of all, SVMs combine accurate detection rates with acceptable training time. Also, by the use of so called kernels SVMs are able to perform nonlinear classification. Finally, researchers already achieved promising results with SVMs in intrusion detection [14].

C. High-level view

Figure 1 provides a high-level view on the proposed NIDS. All flows received by the NIDS are first preprocessed. This step involves scaling the flow to a predefined numeric range and selecting only the relevant flow attributes (see Section IV).

Afterwards, the scaled flow is passed on to the analysis engine. This engine makes use of a OC-SVM which classifies the incoming flow as either malicious (in-class) or not malicious (outlier). In the latter case the flow is ignored while in the former case the flow is to be reported to the network operator.

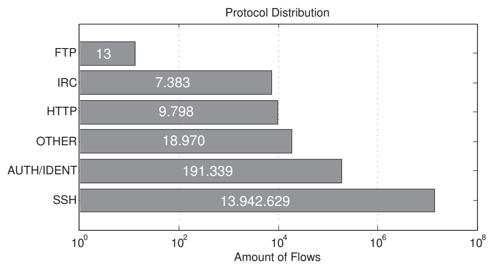

Fig. 2. The protocol distribution of the original training set.

Figure 1 also shows the training set which is used in order to train the OC-SVM. As already mentioned, the set contains only malicious flows. By this means, the OC-SVM is only aware of how malicious flows look like. Benign flows are supposed to be classified as outliers, accordingly. The structure of the malicious training set is covered in Section III-D.

In addition to the malicious set, a data set consisting only of benign flows was created. This set is not part of Figure 1, though. The following two sections now discuss how and why both sets are generated.

D. The malicious data set

The training set for the OC-SVM is a subset of the data set contributed by Sperotto et al. [9]. The authors created the set by setting up a honeypot which was exposed to the Internet for 6 days. The honeypot featured services for three protocols: HTTP, SSH and FTP. The authors identified attacks by monitoring the log files of these three services. That way, the authors gathered around 14 million flows which are mostly of malicious nature. For the protocol and attack semantics, it is referred to the original contribution [9].

Figure 2 illustrates the protocol distribution of the set. By far the most flows have been collected for SSH. The 13 collected FTP flows turned out to be not malicious. Furthermore, the collection process yielded flows belonging to IRC and AUTH/IDENT. According to the authors, these flows can be considered as side effect traffic and are not malicious per se. So the IRC flows are not part of botnet activity. Rather, an attacker just installed an IRC server on the honeypot. Thus, malicious flows in the original training set are effectively limited to SSH and HTTP since there were no malicious FTP flows.

Figure 3 illustrates the type of attacks present in the original training set. The most interesting attacks from the perspective of a network operator are manual and succeeded attacks. These account for only 6 and 144 flows, respectively.

A reduction of the original set is necessary for two reasons: First, the training of 14 million flows would take a vast amount of time. Second, the original set holds many flows which should not be trained such as side effect traffic which is not malicious per se.

The reduction consists of three steps: 1) selecting only the relevant flow attributes, 2) deleting irrelevant flows and 3) performing random sampling to gain a smaller, yet representative subset:

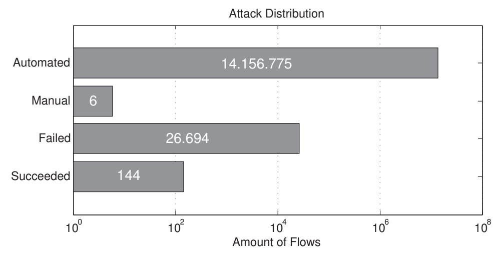

Fig. 3. The attack types of the original training set.

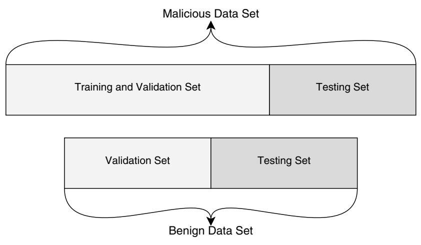

Fig. 4. Partitioning of the malicious and the benign data set.

1) The IP addresses of the original data set were discarded since they have been anonymized. Furthermore, all time information was derived to a single attribute entitled "duration". After all, the following flow attributes remained in the reduced training set: packets/flow, octets/flow, duration per flow, source port, destination port, TCP flags, IP protocol.
2) The deletion of irrelevant flows followed these rules: First of all, all 5.968 unlabeled flows of the original data set (i.e. flows whose nature could not be verified) were deleted. Next, all flows belonging to protocols other than SSH and HTTP were deleted; 215.123 flows were affected by this deletion.
3) Random sampling is necessary because the intermediate data set after step 1) and 2) still contained the vast amount of almost 13 million flows. Training that many flows would require an unacceptably high training time. The random sampling process selected every flow of the intermediate data set with a probability of 1/600. Finally, this step yielded a total of 22.924 flows. These flows now represent the reduced training set which will be used for model and feature selection in Section IV.

E. The benign data set

A data set holding only benign flows was created from scratch. The purpose of this set is to aid in the process of model and feature selection (see Section IV). In the end it is also used to evaluate the NIDS (see Section V). This set was created by manually generating network traffic inside a virtual machine based on grml linux [15]. All the network traffic was captured with tcpdump [16] and then converted to flow format using softflowd [17]. The benign data set embraced flows belonging to the following protocols: HTTP, SSH, DNS, ICMP and FTP. Overall, the set consists of 1.904 flows.

F. Data set partitioning

Preliminary to the process of model and feature selection, the malicious and benign data sets were further divided into a so called testing and validation set, respectively. The partitioning is illustrated in Figure 4.

The two validation sets are only used for model and feature selection whereas the testing sets are used for the final evaluation described in Section V. One-third of the malicious set and half of the benign set were randomly sampled and added to the testing set. The remainder of both sets was chosen to be the validation set, respectively.

IV. Model and Feature Selection

The sole purpose of model and feature selection is the optimization of the classifiers classification capability, i.e., the best possible adaption of the classifier to the custom classification problem.

Feature selection is the process of selecting a subset of features out of the original feature set. Within the scope of this paper this means that the original feature set $\mathcal{F}$ consisting of the seven flow attributes described in Section III-D is to be reduced to a subset $\mathcal{S}$ where $\mathcal{S} \subseteq \mathcal{F}$.

Model selection or optimization is the process of finding parameters for the respective machine learning method which lead to optimal classification results. OC-SVMs require the parameter $\nu, \{\nu \in \mathbb{R} \mid 0 < \nu \leq 1\}$ which controls the fraction of the outliers in the training set. For ν, we tested the values $\{0.001, 0.201, 0.401, 0.601, 0.801\}$. This linearly increasing set was chosen to approximately cover the possible range for ν. Furthermore, a radial basis function (RBF) kernel was chosen for the OC-SVM due to its general applicability [18]. The RBF kernel requires a parameter $\gamma, \{\gamma \in \mathbb{R} \mid 0 \leq \gamma\}$ which specifies the width of the RBF kernel. For γ, the values $\{0.1, 0.3, 0.5, 1, 2, 5, 10\}$ are tested. This set increases nonlinearly and is also supposed to cover a small but realistic range for γ.

So, roughly speaking model selection is the task of determining the best tradeoff between ν and γ for the OC-SVM.

A. Joint optimization

Model and feature selection was regarded as one joint rather than two independent optimization problems. As pointed out in [19] and [20], this methodology can lead to better results.

As argued by the authors of [11], limiting the false alarm rate of an anomaly detection system should have top priority. Hence, our proposed NIDS was optimized with respect to its false alarm rate.

The small search space enables the use of the popular grid search method for joint optimization. In the scope of this paper, the grid spans over three dimensions:

1) The feature subset consisting of $2^7 - 1$ possible subsets. The exponent stands for the 7 feature candidates, i.e. flow attributes.
2) The model parameter ν for which 5 values are tested as discussed in the previous section.

TABLE I
RESULTS OF THE COARSE GRAINED OPTIMIZATION.

False alarm rate	Miss rate	γ	ν	Feature subset
0%	22.53807%	0.1	0.201	SP, DP, TF, PR
0%	22.53807%	0.1	0.201	SP, DP, TF
0%	22.61685%	0.3	0.201	SP, DP, TF, PR
0%	22.61685%	0.3	0.201	SP, DP, TF
0%	22.66281%	0.1	0.201	P, SP, DP, TF, PR

TABLE II
RESULTS OF THE FINE GRAINED OPTIMIZATION.

False alarm rate	Miss rate	γ	ν
0%	4.71376%	0.26	0.02
0%	4.72689%	0.25	0.02
0%	4.74658%	0.29	0.02
0%	4.75315%	0.27	0.02
0%	4.76628%	0.28	0.02

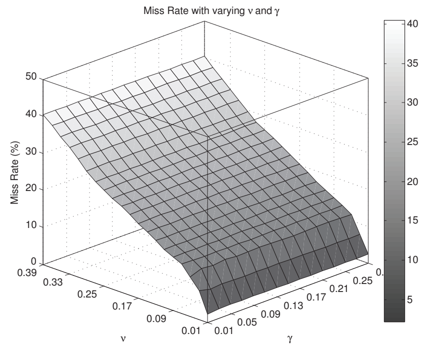

Fig. 5. Relationship between the varying parameters ν and γ and the miss rate during the fine grained grid search.

3) The kernel parameter γ for which 7 values are tested.

Overall, the grid holds a total of 127 * 5 * 7 = 4.445 points. For each grid point, an 8-folded[1] cross-validation is performed to determine the error rates consisting of the false alarm rate and the miss rate.

The optimization was divided into two consecutive steps.

1) The first step is entitled "coarse grained optimization". It is meant to determine the best feature subset as well as the best tradeoff between ν and γ.
2) Afterwards, "fine grained optimization" is performed. This step bases upon the results of the coarse grained optimization and further explores the surrounding region of the best ν/γ-combination to achieve even better results.

B. Coarse grained optimization

The best five results of the coarse grained optimization with respect to the false alarm rate are listed in Table I. The abbreviations in the last column are as follows: P = packets per flow, SP = source port, DP = destination port, TF = TCP flags and PR = IP protocol.

The results reveal that none of the best five combinations causes any false alarms. The best one scores only marginally better than the remaining four. For all five points, the model parameter ν is 0.201. The parameter γ, on the other hand exhibits some variance in the range between 0.1 and 0.3.

Since all of the five points yield an identical false alarm rate, the point with the lowest miss rate is chosen for further optimization. In fact, the first two points of Table I have identical error rates. The difference is that one of these points contains the IP protocol feature whereas the other does not. We chose to take the one including this feature since we believe that the additional feature enhances the generalization capability of the OC-SVM.

C. Fine grained optimization

The coarse grained optimization resulted in the point characterized by γ = 0.1, ν = 0.201 and the feature subset holding source port, destination port, TCP flags and IP protocol.

As already mentioned, this feature subset is not changed anymore. Hence, only ν and γ remain for further optimization. The fine grained testing range for ν and γ is chosen to lie between the nearest coarse grained values. Since the previous optimization came up with ν = 0.201 and γ = 0.1, the set to be further explored starts with 0.01 for ν and γ. The set ends with 0.29 for ν and 0.39 for γ. For both parameters, a step size of 0.01 is chosen. After all, for γ the following set is tested: $\{0.01, 0.02, ..., 0.28, 0.29\}$ whereas for ν the set contains: $\{0.01, 0.02, ..., 0.38, 0.39\}$. With 39 values in the set for ν and 29 values in the set for γ, the fine grained optimization has to test $29 * 39 = 1.131$ points. Again, for each point an 8-folded cross-validation is executed.

The best five results of the fine grained optimization are listed in Table II. All results yield a miss rate of around 4.7% with a false alarm rate of still 0%. The parameter ν is 0.02 for all results whereas γ varies between 0.25 and 0.29.

The fine grained optimization was able to significantly lower the error rates of the coarse grained optimization. The miss rate was originally around 22.5% and could be lowered to 4.7%.

Figure 5 illustrates how the miss rate on the Z-axis varies during fine grained optimization. One can clearly see the correlation between ν and the miss rate. The range of the parameter γ, on the other hand, hardly influences the miss rate.

V. EVALUATION AND DISCUSSION

After model and feature selection determined the best set of features and model parameters, it is crucial to test the final model by making use of the dedicated testing set created in Section III-F.

A. Testing set prediction

For the purpose of testing, the OC-SVM is trained with the malicious training set (see Figure 4), an RBF kernel and the

[1]The fold size was chosen to be a multiple of the four CPU cores.

TABLE III
PREDICTION OF THE TESTING SETS.

	Benign set		Malicious set	
Type	**Predicted**	**Actual**	**Predicted**	**Actual**
In-class	0 (0%)	0	7.540 (98.07492%)	7.688
Outlier	942 (100%)	942	148 (1.92507%)	0

parameters ν = 0.02 and γ = 0.26. Only the features source port, destination port, TCP flags and IP protocol are used for training.

Table III lists the results of the prediction of both testing sets. None of the benign flows was predicted as in-class, i.e., malicious. Rather, the OC-SVM correctly classified all of the benign flows as outliers. This corresponds to a false alarm rate of 0%. Furthermore, around 98% of the flows were correctly predicted as in-class. Almost 2% were classified as outliers, i.e., not malicious. This corresponds to a miss rate of 2%.

B. Discussion

Although the results of the evaluation seem highly promising at first glance, they have to be viewed critically. We identified the following drawbacks in our approach.

First of all, the OC-SVM appears to be highly dependent on source and destination port. This assumption is affirmed by the fact that evaluation results are significantly worse if these two features are omitted. In this case, the false alarm rate increases to 0.7% and the miss rate increases to even 81.1%.

Another drawback lies in the fact that the training set is effectively limited to attacks for SSH and HTTP. Also, the largest fraction in the training set is scan traffic. In fact, many network operators deem probing activity to be uninteresting but it can be an indicator for botnet infections or other compromised hosts when coming from "the inside".

On the other hand, the proposed inductive NIDS embraces several important advantages. First of all, a once trained OC-SVM can be distributed to multiple networks without a prior learning phase as often required by anomaly detection systems. The use of NetFlow alleviates the monitoring of large-scale networks and allows the analysis of encrypted network connections. Finally, we believe that training malicious rather than benign network traffic keeps the false alarm rate low and allows the detection of attack variations. However, further research will be necessary to prove these assumptions.

VI. CONCLUSIONS AND FUTURE WORK

We introduced a novel concept for an inductive NIDS. The NIDS operates on network flows and makes use of One-Class Support Vector Machines for analysis. The NIDS is designed for recognizing attacks and their variations rather than for detecting deviations of normal traffic.

The results of the evaluation suggest that the proposed approach yields quite satisfying results although there is much room for further testing and improvement.

Predominantly, future work will concentrate on enhancing both data sets. A sound and comprehensive evaluation requires multiple representative and realistic data sets collected "in the wild".

The source code and the prepared data sets are available upon email request to the authors.

ACKNOWLEDGMENT

We want to thank the anonymous reviewers who provided us with many helpful comments.

REFERENCES

[1] Y. Gao, Z. Li, and Y. Chen, "A DoS Resilient Flow-level Intrusion Detection Approach for High-speed Networks," in *Proc. of the 26th IEEE International Conference on Distributed Computing Systems (ICDCS '06)*, Washington, DC, USA, 2006, p. 39.
[2] A. Karasaridis, B. Rexroad, and D. Hoeflin, "Wide-scale botnet detection and characterization," in *Proc. of the first conference on Hot Topics in Understanding Botnets (HotBots '07)*, Berkeley, CA, USA, 2007, p. 7.
[3] A. Shahrestani, M. Feily, R. Ahmad, and S. Ramadass, "Architecture for Applying Data Mining and Visualization on Network Flow for Botnet Traffic Detection," in *Proc. of the International Conference on Computer Technology and Development (ICCTD '09)*, Washington, DC, USA, 2009, pp. 33 – 37.
[4] C. Livadas, R. Walsh, D. Lapsley, and W. T. Strayer, "Using Machine Learning Techniques to Identify Botnet Traffic," in *2nd IEEE LCN Workshop on Network Security (WoNS '06)*, 2006, pp. 967 – 974.
[5] A. Sperotto *et al.*, "An Overview of IP Flow-Based Intrusion Detection," *IEEE Communications Surveys Tutorials*, vol. 12, no. 3, pp. 343 – 356, 2010.
[6] R. P. Lippmann *et al.*, "Evaluating Intrusion Detection Systems: The 1998 DARPA Off-line Intrusion Detection Evaluation," in *Proc. of the DARPA Information Survivability Conference and Exposition*, 2000, pp. 12 – 26.
[7] J. McHugh, "Testing Intrusion detection systems: a critique of the 1998 and 1999 DARPA intrusion detection system evaluations as performed by Lincoln Laboratory," *ACM Transactions on Information and System Security*, vol. 3, no. 4, pp. 262 – 294, 2000.
[8] "KDD Cup 1999: General Information," 1999, http://www.sigkdd.org/kddcup/index.php?section=1999&method=info.
[9] A. Sperotto, R. Sadre, F. Vliet, and A. Pras, "A Labeled Data Set for Flow-Based Intrusion Detection," in *Proc. of the 9th IEEE International Workshop on IP Operations and Management (IPOM '09)*, Berlin, Heidelberg, 2009, pp. 39 – 50.
[10] C. Gates and C. Taylor, "Challenging the anomaly detection paradigm: a provocative discussion," in *Proc. of the 2006 workshop on New security paradigms (NSPW '06)*, New York, NY, USA, 2007, pp. 21 – 29.
[11] R. Sommer and V. Paxson, "Outside the closed world: On using machine learning for network intrusion detection," in *IEEE Symposium on Security and Privacy (S&P)*, May 2010, pp. 305 – 316.
[12] B. Schölkopf *et al.*, "Estimating the Support of a High-Dimensional Distribution," *Neural Computation*, vol. 13, no. 7, pp. 1443 – 1471, 2001.
[13] C.-C. Chang and C.-J. Lin, *LIBSVM: a library for support vector machines*, 2001, software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm.
[14] L. Khan, M. Awad, and B. Thuraisingham, "A new intrusion detection system using support vector machines and hierarchical clustering," *The VLDB Journal*, vol. 16, no. 4, pp. 507 – 521, 2007.
[15] "Grml," http://www.grml.org.
[16] "tcpdump / libpcap," http://www.tcpdump.org.
[17] "softflowd," http://www.mindrot.org/projects/softflowd/.
[18] C.-W. Hsu, C.-C. Chang, and C.-J. Lin, "A Practical Guide to Support Vector Classification," 2000.
[19] Q.-Z. Yao, J. Cai, and J.-L. Zhang, "Simultaneous Feature Selection and LS-SVM Parameters Optimization Algorithm Based on PSO," in *Proc. of the WRI World Congress on Computer Science and Information Engineering (CSIE '09)*, Washington, DC, USA, 2009, pp. 723 – 727.
[20] S.-W. Lin, T.-Y. Tseng, S.-C. Chen, and J.-F. Huang, "A SA-Based Feature Selection and Parameter Optimization Approach for Support Vector Machine," in *IEEE International Conference on Systems, Man and Cybernetics (SMC '06)*, vol. 4, 2006, pp. 3144 – 3145.