Flow-based benchmark data sets for intrusion detection

Markus Ring[1], Sarah Wunderlich[1], Dominik Grüdl[1], Dieter Landes[1], Andreas Hotho[2]
[1]Coburg University of Applied Sciences, Coburg, Germany
[2]University of Würzburg, Würzburg, Germany
markus.ring@hs-coburg.de
sarah.wunderlich@hs-coburg.de
dominik.gruedl@stud.hs-coburg.de
dieter.landes@hs-coburg.de
hotho@informatik.uni-wuerzburg.de

Abstract: Anomaly based intrusion detection systems suffer from a lack of appropriate evaluation data sets. Often, existing data sets may not be published due to privacy concerns or do not reflect actual and current attack scenarios. In order to overcome these problems, we identify characteristics of good data sets and develop an appropriate concept for the generation of labelled flow-based data sets that satisfy these criteria. The concept is implemented based on OpenStack, thus demonstrating the suitability of virtual environments. Virtual environments offer advantages compared to static data sets by easily creating up-to-date data sets with recent trends in user behaviour and new attack scenarios.
In particular, we emulate a small business environment which includes several clients and typical servers. Network traffic is generated by scripts which emulate typical user activities like surfing the web, writing emails, or printing documents on the clients. These scripts follow some guidelines to ensure that the user behaviour is as realistic as possible, also with respect to working hours and lunch breaks. The generated network traffic is recorded in unidirectional *NetFlow* format.
For generating malicious traffic, attacks like Denial of Service, Brute Force, and Port Scans are executed within the network. Since origins, targets, and timestamps of executed attacks are known, labelling of recorded *NetFlow* data is easily possible. For inclusion of actual traffic, which has its origin outside the OpenStack environment, an external server with two services is deployed. This server has a public IP address and is exposed to real and up-to-date attacks from the internet.
We captured approximately 32 million flows over a period of four weeks and categorized them into five classes. Further, the chronological sequence of the flows is analysed and the distribution of normal and malicious traffic is discussed in detail. The main contribution of this paper is the demonstration of a novel approach to use OpenStack as a basis for generating realistic data sets that can be used for the evaluation of network intrusion detection systems.

Keywords: data set generation, intrusion detection, anomaly detection, netflow, openstack

1. Introduction

Network-compatible devices permeate nearly all areas of modern society. This does not only hold for the increasing number of the devices due to the internet of things but also for the sensible data stored in every company's network. Simultaneously, the number of cyber threats against company data and critical infrastructures is increasing. However, company data are a valuable asset which must be protected against loss and manipulation by unauthorized parties (Landes et al., 2013). Further, critical infrastructures are essential for day-to-day operations of economy and government (Ralston et al., 2007). Consequently, it is necessary to protect them against cyber threats. Various security mechanisms like network intrusion detection systems (NIDS), security information and event management systems (SIEM), or firewalls are in use for protecting company networks and critical infrastructures against criminal activities (Ring et al., 2017a).
In this work, we focus on anomaly-based NIDS. Anomaly-based systems try to model normal network behaviour based on representative training data. If incoming network traffic differs significantly from learnt behaviour, it is marked as malicious. As a result, anomaly-based NIDS are able to detect novel and obfuscated attacks (Otto et al., 2016). However, realistic and labelled data sets are rare and often do not reflect current trends (Shiravi et al., 2012).
In order to overcome these problems, we propose an approach for creating labelled data sets for training and evaluating anomaly-based NIDS. This approach builds upon the software OpenStack. We emulate a small business environment in OpenStack and capture the generated network traffic of the virtual machines in

unidirectional *NetFlow* format. Network traffic is recorded in flow-based format instead of packet-based since this bypasses the problem of encrypted connections, leads to less privacy concerns and reduces the amount of data to be analysed. Anomaly-based systems are better at finding similarities than finding outliers (Sommer and Paxson, 2011). Therefore, normal and malicious user activities are considered when creating the data sets. Normal user behaviour is generated by executing Python scripts on the clients which follow some self-defined guidelines. Malicious network traffic is captured by explicitly executing attacks within the OpenStack environment and by capturing the traffic of a server which is exposed to real and up-to-date attacks from the internet.
The main contribution of this paper is two-fold: first, we demonstrate the suitability of OpenStack as a tool to generate labelled and realistic benchmark data sets for NIDS, and second, the exemplarily generated data set CIDDS-001 (Coburg Intrusion Detection Data Sets) (Ring et al., 2017b) as well as the Python scripts (Ring et al., 2017c) are made publicly available for use by other researches.

The rest of the paper is organized as follows: Related work on network-based data sets is discussed in Section 2. Section 3 describes our concept for the generation of labelled flow-based data sets using OpenStack. In Section 4, we analyse the created data set in more detail. The last section summarizes the paper and provides an outlook for future work.

2. Related work

This section reviews related work on available data sets for intrusion and insider threat detection. Intrusion detection data sets can be categorized into network-based, host-based and application-based. Since the proposed approach is network-based, the following review does not consider data sets from the other two types. We subdivide network-based intrusion detection data sets further into packet-based (category I) and flow-based (category II).

Packet-based data sets usually contain packet-headers and payloads. The well-known DARPA98 and DARPA99 data sets from the MIT Lincoln Laboratory are representatives of this category. Both data sets capture traffic from a simulated environment. The KDD CUP 99 data set is a modified version of DARPA98 and one of the most widely used data sets for NIDS evaluation. However, weaknesses of the KDD CUP 99 data set include amongst other the high number of duplicates. Tavallaee et al. (2009) proposed the NSL-KDD data set which is based on KDD CUP 99 and tries to eliminate identified weaknesses. However, the mentioned data sets are based on data which was captured more than 17 years ago. Consequently, it is questionable if they contain network traffic which reflects up-to-date scenarios of both, malicious and normal traffic.
The MAWI repository provides up-to-date packet-based data sets. Those data sets are created by capturing the network traffic of an internet backbone and are labelled by combining various anomaly detectors (Fontugne et al., 2010). Despite their topicality, it is questionable if these data sets are suitable for training and evaluation of anomaly-based NIDS, since the characteristics of network traffic from internet backbones differs from the one in company networks.
Vasilomanolakis et al. (2015) published a packet-based data set generator (ID2T). This generator uses real input data and mixes them with attacks. Attacks are inserted either by script-based attack generation or by pcap modification.

Data sets from category II contain only aggregated information about the connections within the network. Sperotto et al. (2009) contributed one of the first flow-based data sets. The authors collected flow-based data from a honeypot with several services and analysed the log files to label the corresponding flows. However, nearly all of the 14 million flows are malicious since real background traffic is missing.
The CTU 13 Malware data set (García et al., 2014) is another representative of this category. It contains normal traffic and traffic from different malware scenarios. Labelling of malicious traffic is based on the IP addresses used by the botnets. Shiravi et al. (2012) proposed a systematic approach to generate labelled flow-based data sets for IDS. The authors describe various profiles which describe normal user activities as well as attack scenarios. These profiles can be combined into new data sets. The data sets published by (Shiravi et al., 2012) and (García et al., 2014) contain bidirectional flows while we aim at using unidirectional flows. Converting bidirectional into unidirectional flows might be viable, but would require some effort for re-labelling. Nevertheless, we would still be limited to a fixed number and kind of attacks within these data sets.
Wheelus et al. (2014) and Zuech et al. (2015) proposed flow-based data sets. The SANTA data set of (Wheelus et al., 2014) contains real traffic and different attack scenarios. Labelling of the SANTA data set was done by

manual analysis and heuristics. Zuech et al. (2015) present a data set for IDS evaluation called IRSC. The authors collected network flows as well as full packets and spent much effort for correct labelling. So far, however, neither of the two data sets is publicly available.

It may be concluded that evaluation of NIDS is challenging due to the lack of publicly available data sets. Data sets that are discussed in literature are often outdated or not publicly available. As a countermeasure, we propose an approach to generate flow-based data sets similar to (Shiravi et al., 2012). In contrast to (Shiravi et al., 2012), we also capture real and up-to-date network traffic and threats by deploying a server in the internet. Further, our emulated small business environment includes additional services like network printers or file synchronization servers and we use a different labelling approach which provides additional information about attacks and normal user behaviour.

3. Data set generation

This section introduces our approach for generation of labelled flow-based data sets. Section 3.1 discusses characteristics of good data sets. Section 3.2 provides an overview of our setting and presents the underlying ideas. Sections 3.3 and 3.4 explain the generation of normal and malicious traffic. Labelling and anonymization of the flow-based data are described in Sections 3.5 and 3.6.

3.1 Requirements for good data sets

When generating new data sets, it is important to note how good data sets should look like. Małowidzki et al. (2015) define the following features for good data sets:

- contain recent data
- be realistic
- contain all typical attacks met in the wild
- be labelled
- be correct regarding operating cycles in enterprises (working hours)
- should be flow-based

We agree to all identified features. Further, we contend that a good data set should be comparable with real traffic and therefore has more normal than malicious traffic, since most of the traffic within a company is normal and only a small part is malicious.

3.2 Overview of our data set generation approach

The overall objective of this work is the generation of realistic labelled flow-based data sets conformant to the requirements discussed in Section 3.1. First of all, we claim that realistic flow-based network traffic should be recorded from actual networks instead of simulating network traffic by mathematical models. This assures the consideration of all parameters which could influence the timing behaviour of flow-based traffic such as response times of servers, CPU usage, or intermittent bottlenecks of the internet connection.

The software platform OpenStack allows the creation of virtual networks and virtual machines. A virtual environment based on OpenStack allows us to meet most of the above mentioned features. A major advantage of generating data sets in a virtual environment is the possibility of regular adjustments of the behaviours of server or clients. This way, new attacks or new trends in user behaviour can be included easily to constantly generate recent and up-to-date data sets. Also, new ideas for improving the quality of the emulation can be easily integrated and tested.

As another important advantage, full control over the environment including network devices like routers and switches is warranted. Thus, the richness of the data set, operating cycles as well as the ratio of normal to malicious behaviour may be set as necessary. Currently, we configure OpenStack to use *neutron* with *OpenVSwitch* and capture the whole network traffic within the virtual networks in unidirectional *NetFlow* format. We set up a small business environment with typical servers and some clients. Figure 1 shows an overview of our small business environment.

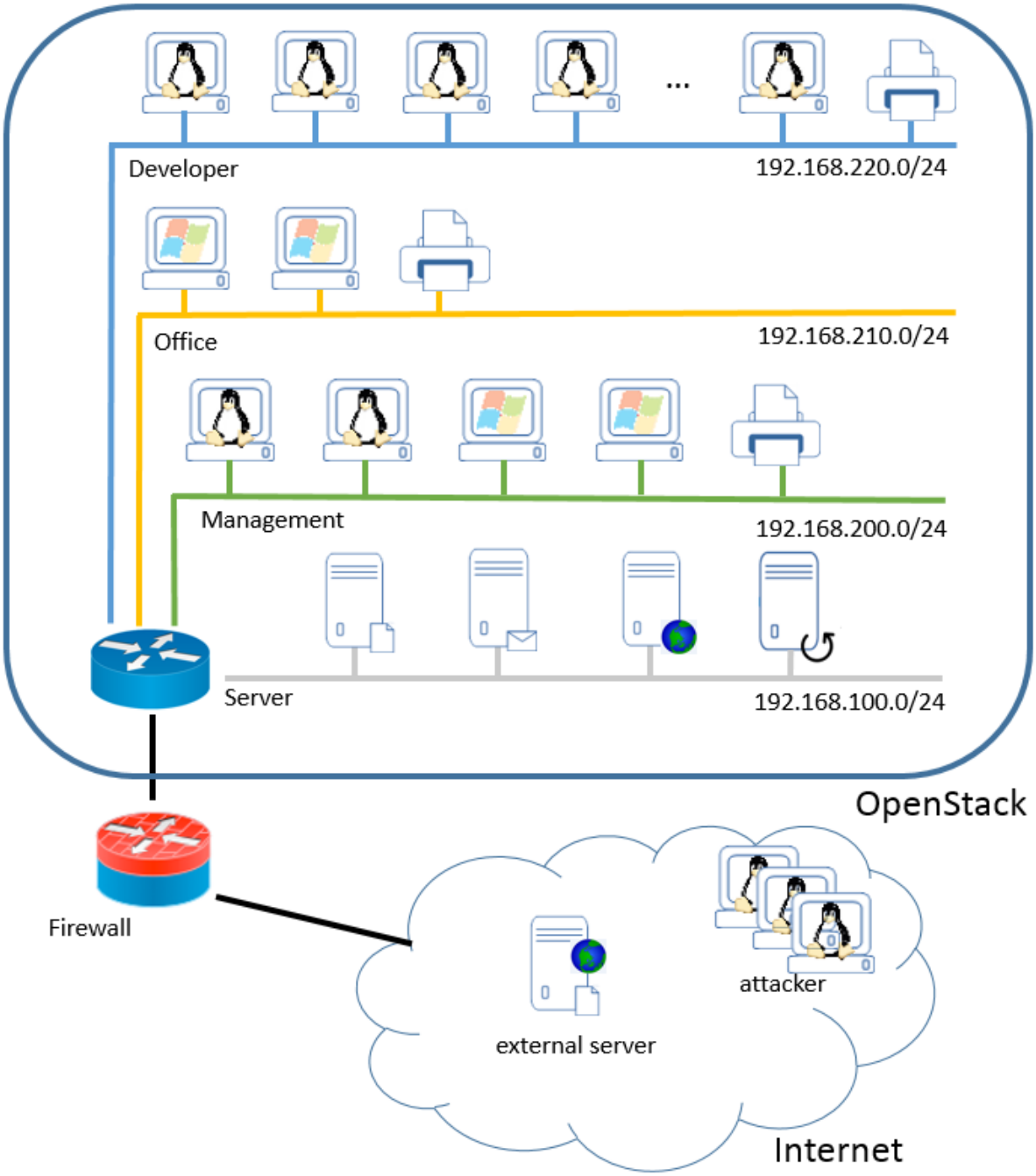

Figure 1: Overview of the simulated environment.

The router within the OpenStack environment connects four subnets with each other and forwards traffic to the internet. The four subnets reflect the organizational structure of our virtual company. In particular, the Management, Office, and Developer departments have their own subnets (/24). The fourth subnet (192.168.100.0/24) contains four internal servers: mail, web, backup and file server. An external server provides two additional services via the internet: a file synchronization service (Seafile) and a public webserver.

Randomized and parameterized python scripts emulate a variety of network activities on the clients and can be adopted to specific scenarios. The traffic caused by these scripts is recorded at the router within the OpenStack environment. Python scripts emulate typical user activities following some guidelines described below in Section 3.3. For the generation of malicious traffic, different types of attacks are executed within the virtual network (see Section 3.4). To make the generated data even more realistic, we also record the network traffic of our external server at its network card and merge it into the other traffic. The external server offers two services which are correctly used by the clients, but is simultaneously exposed to real and up-to-date

attacks from the internet. Other than a honeypot, this approach enables recording both normal and malicious traffic at the external server.
Overall, the whole environment has five servers, three printers, four windows clients and fifteen Linux clients. The Developer subnet exclusively contains Linux clients, the Office subnet encompasses only Windows clients while the Management subnet is a mix of both.

3.3 Generation of normal data

Since we emulate normal user behaviour through scripts on the clients, we define two guidelines for generation of realistic flow-based network traffic. The first guideline focuses on the realistic emulation of user behaviour whereas the second guideline takes the heterogeneity of operating systems into account. To fulfil these guidelines, the scripts consider the following features:

1.) Runnable on different operating systems
2.) Consider typical computerized activities of employees
3.) Consider different tasks and working methods of employees
4.) No periodic repetition of user activities
5.) Consider typical working hours and breaks

The first feature is met by using the platform independent language Python for writing the user behaviour scripts.
Employees have a wide range of activities during their daily work like writing emails, creating documents and presentations, browsing (private or business searches), printing, sharing files and so on. For emulating such activities with respect to potential different characteristics of different employees, each client has its individual configuration file. The configuration file controls activities and their frequency for each client. Thus, different user profiles may be assigned to different clients. For transferring files and printing documents, it is important to ensure that the corresponding files vary in terms of types and sizes. Further, when sending emails, the number of attachments should change.
Realistic user behaviour cannot be characterized by repeating a list of activities periodically. Instead, the temporal sequence of user activities should be randomized and the kind of activities should vary. Still, activities should not be totally random and follow a probability distribution which is based on typical working hours.
Typically, employees are not permanently performing tasks which cause network traffic. It is important to consider meetings, offline work or coffee breaks (see feature 5 above). Scripts should emphasize working hours and stop activities in breaks and in the evening.

Considering these requirements lead to data sets that are as realistic as possible, even though they do not reflect perfect user behaviour. Since the configuration file is modular, new restrictions and ideas may be integrated easily. We could even try to learn the parameters for the individual configuration files from real settings to get better emulated traffic.

3.4 Generation of malicious traffic

A comprehensive IDS data set consists of normal as well as malicious network traffic. Normal network traffic is generated by Python scripts as described above. The generation of malicious network traffic is based on a two-pronged solution.
On the one hand, two internal clients perform insider attacks (Ping-Scans, Port-Scans, Brute-Force and Denial of Service attacks) using Linux tools like *nmap* and various Python scripts. The list of attacks can be easily extended. Further, the attacker computers (see Figure 1) attack the external server with Port-Scans and Brute-Force attacks. All these attacks can be easily labelled for our dataset. On the other hand, the external server is directly accessible from the internet and consequently exposed to actual and up-to-date attacks from the web which are not initiated by ourselves. Labelling of this traffic is more difficult.

3.5 Labelling

For evaluation of NIDS, a fully labelled data set is indispensable. Therefore, we perform a four-stage labelling process. The first label attribute is called *class* and categorizes flows in five categories: *normal*, *attacker, victim suspicious* and *unknown*. The second label attribute is called *attackType* and provides the type of attack, if the label attribute *class* contains the value *attacker* or *victim.* A third label attribute called *attackID* assigns a

unique ID to all flows that belong to the same attack. The fourth label attribute is called *attackDescription* and gives a more detailed explanation of the attack.

We use different labelling processes for the captured network traffic within the OpenStack environment and the external server. Network traffic captured at the OpenStack router can easily be labelled. Since origins, targets, and timestamps of executed attacks are known, *attack* traffic can be easily identified and assigned with the corresponding labels (*attacker* or *victim)*. The remaining traffic is labelled as *normal*.
Labelling traffic of the external server is more time-consuming. All clients from the OpenStack environment communicate with the same public IP address to the external server. This traffic can be labelled as *normal* traffic. Further, we have control over some other machines that are directly connected to the internet (see *attacker* in Figure 1). For these computers, we also know the IP addresses and only exploit attacks to the external server. Hence we know the origins, target and timestamps of these attacks. Consequently, we are able to label the corresponding flows with *attacker* and *victim*. However, we are not able to determine unequivocal labels to the other flows. Therefore, we use the following two rules. (1) All requests on port 80 and 443 are labelled as *unknown,* since we do not know if requests are normal customer visits or attacks. (2) All other requested ports on the external server are labelled as *suspicious*, since no other ports are offered for public users.

3.6 Anonymization

For privacy reasons, all public IP addresses are anonymized according to the following approach: the public IP address of OpenStack is replaced with "OPENSTACK_NET", the IP address of the DNS server is replaced with "DNS" and the IP address of the external server is replaced with "EXT_SERVER". IP addresses of the attackers are replaced by "ATTACKER1", "ATTACKER2" and "ATTACKER3".
For all other public IP addresses, the first three bytes of each IP address are replaced with a randomly generated number. The anonymization process ensures that the same IP address is always mapped to the same generated number. This allows anonymization of public IP addresses while preserving information about subnets. For example, possible transformations could look like:

- "8.102.3.251" to "4711_251"
- "8.102.3.233" to "4711_233"
- "6.204.34.23" to "2342_23"
- "201.133.175.87" to "9721_87"

4. Analysis of the generated data

This section analyses the captured network traffic of the scenario described in Section 3. The main characteristics of the CIDDS-001 data set are provided in Section 4.1. Section 4.2 and Section 4.3 describe the *OpenStack* and *ExternalServer* related traffic in more detail.

4.1 Overview

Proper evaluation of the resulting data set requires a deeper analysis of the captured flows. The network traffic was captured over a period of four weeks. Table 1 gives an overview regarding the sources of the flows.

Table 1: Main characteristics of the CIDDS-001 data set.

	OpenStack	ExternalServer	Overall
Number of flows	31287934	671241	31959175
Number of exploited attacks	70	22	92

The resulting data set consists of two parts: *OpenStack* and *ExternalServer*. Table 1 shows the number of flows and exploited attacks for each part while the last column contains the sum of both. As one can easily see, the OpenStack environment generates the main portion of traffic. Nearly 32 million flows were captured from which around 31 million flows were captured in the *OpenStack* environment. Overall, we exploited 92 attacks within the four weeks.
Each flow within the data set contains the typical *NetFlow* attributes, namely: *Source IP Address, Source Port, Destination IP Address, Destination Port, Proto, Date first seen, Duration, Bytes, Packets, Flows,* and *TCP-Flags.* The labelling process (see Section 3.5) adds four more attributes to each flow: *class*, *attackType*, *attackDescription* and *attackID*.

4.2 OpenStack

In this section, we analyse the in *OpenStack* captured flow-based network traffic in more detail.

Table 2: *Class* distribution of the flow-based network traffic captured in OpenStack.

Number of flows	31287934 (100.0%)
Number of *normal* flows	28051907 (89.66%)
Number of *attacker* flows	1656605 (05.29%)
Number of *victim* flows	1579422 (05.05%)
Number of *suspicious* flows	0 (00.00%)
Number of *unknown* flows	0 (00.00%)

Table 2 shows the distribution of *class* labels in the flows captured in the *OpenStack* environment. The first column indicates the measured parameter whereas the second column contains the number of flows and their proportional percentage (in brackets). Over four weeks, about 31 million flows were captured most of which are caused by normal user behaviour. Since the exact timestamps of the executed attacks are known, labelling each flow corresponding to its meaning is easily possible. Consequently, the number of suspicious and unknown flows is zero.

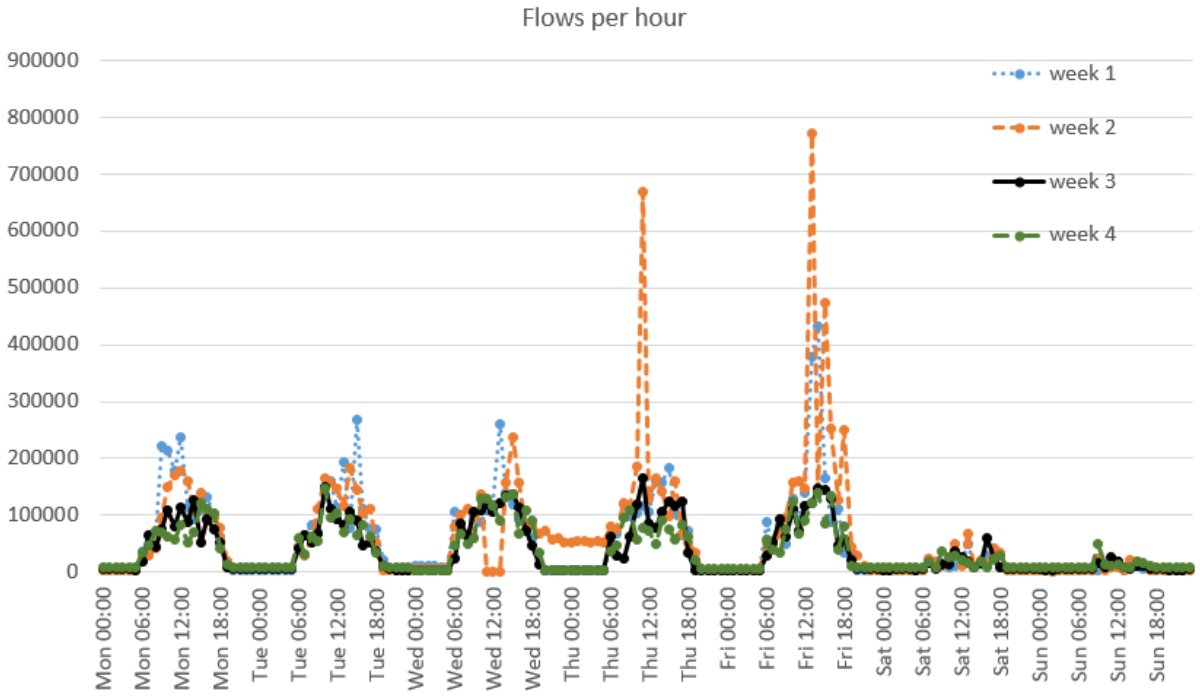

Figure 2: Temporal sequence of the captured traffic at the OpenStack router. The y-axis counts the flows per hour and each week is displayed as a line of different colour and structure.

Figure 2 shows the temporal sequence of captured network traffic. Each line represents a week while the y-axis indicates the number of flows per hour. Typical working hours can be easily recognized in Figure 2. Considering Monday, we can observe an increasing number of flows at 6:00, a small decrease around lunch time (12:00) and an increase one hour later. Between 16:00 and 18:00, when most employees leave from work, the amount of network flows decreases. Further, we can identify the typical working days from Monday to Friday with much more traffic than on Saturdays and Sundays, which entails a smaller extent of daily fluctuations of the flows. The nightly backup of the servers causes only a small number of flows which is not recognizable in Figure 2. In non-working hours, we observe an equal distribution of the flows which is caused by default requests, network drives and Seafile synchronizations. The peaks in *week2* for Thursday and Friday are caused by *DOS* attacks. Further, we can observe a network failure in week2 at Wednesday around 12:00.

4.3 ExternalServer data

Besides *OpenStack*, the *ExternalServer* is used as a second source for flow-based traffic. As already mentioned, the server is deployed on the internet and is hence open to everyday exploits. Consequently, the labels are more diverse in comparison.
Table 3 shows the label distribution of the flows. Again, the first column indicates the measured parameter whereas the second column contains the number of flows and their proportional percentage (in brackets). We captured over 0.6 million flows within four weeks. *OpenStack* clients generated nearly one fifth of the flows. Attacks exploited by us from the internet contribute over three percent of the flows. Since a publicly available webserver is deployed on the external server, the corresponding traffic (Port 80 and 443) could be normal customer requests or attacks. This traffic is labelled as *unknown* and causes about 11 percent of the traffic. The remaining traffic, which is nearly two thirds of the whole traffic, is labelled as suspicious.

Table 3: *Class* distribution of the flow-based network traffic captured at the external server.

Number of flows	671241 (100.0%)
Number of *normal* flows	134240 (20.00%)
Number of *attacker* flows	12260 (01.83%)
Number of *victim* flows	8907 (01.32%)
Number of *suspicious* flows	437911 (65.24%)
Number of *unknown* flows	77923 (11.61%)

Figure 3 shows the temporal sequence of the flows at the external server.

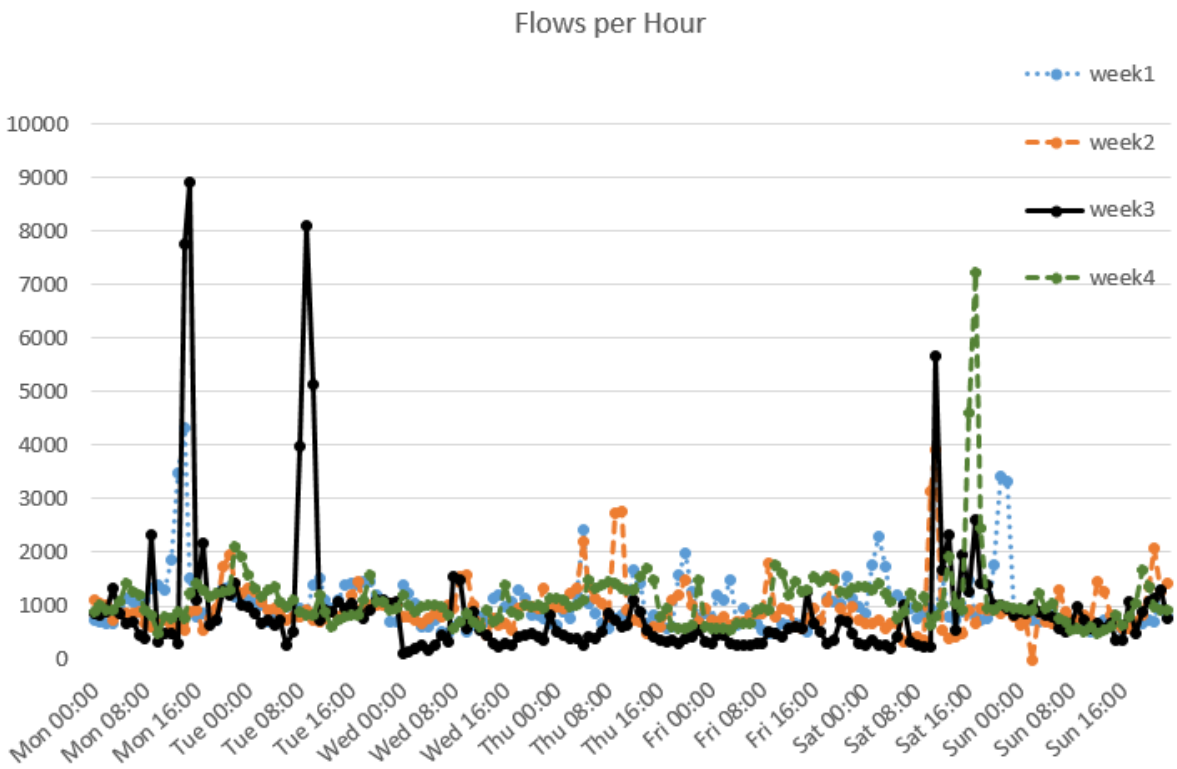

Figure 3: Temporal sequence of the captured traffic at the external server. The y-axis counts the flows per hour and each week is represented by a line of different colour and structure.

The temporal sequence of captured traffic at the external server differs significantly from the traffic of the *OpenStack* network. First of all, we cannot observe an increasing amount of traffic during the working hours, since two thirds of the traffic are suspicious traffic from the web. Most of the peeks in Figure 3 represent scanning attacks executed by ourselves. Further, we take a deeper look at the traffic which is labelled as suspicious in Table 4.

Table 4: Deeper investigation of *suspicious* flows.

Number of suspicious flows	437911 (100.0%)
Number of flows to Port 22 (ssh)	337427 (77.05%)
Number of flows to Port 23 (telnet)	38050 (08.89%)
Number of flows from Port 80	634 (0.14%)
Number of flows to Port 8000 (Seafile)	1462 (0.33%)
Number of remaining flows	60608 (13.84%)

Over three quarters of the traffic is directed to port 22 and represents SSH login attempts not initiated by our clients. Requests to port 23 (telnet) contribute further 9 percent of the traffic. About 1500 flows tried to access our Seafile service. Further, we detected a port scan from origin port 80, which also causes 0.14 percent of the traffic.

5. Summary and Future Work

Anomaly-based network intrusion detection systems (NIDS) are used to protect company data and critical infrastructures against loss and manipulation by unauthorized parties. In this paper, we proposed an approach for the generation of labelled flow-based data sets for training and evaluating NIDS. We used the OpenStack software platform to set up a small business environment and recorded the network traffic in unidirectional *NetFlow* format. Python scripts running on the clients for generating normal network traffic. These scripts underlie some guidelines to ensure as realistic user behaviour as possible. Malicious traffic is added by explicitly executing attacks within the network. Network traffic that originates outside the OpenStack environment is added by an external server. This external server offers two services and is exposed to real and up-to-date attacks from the internet. The resulting data set is analysed in more detail and significant characteristics are highlighted in graphical views.

In the future, we want to extend our small business environment. A repository server should be integrated in the environment and additional activities should enhance client scripts e.g. by using Skype. Further, we want to learn the distribution of user activities from real network traffic for even more realistic parameterization of our scripts which control the user behaviour. Additionally, we want to exploit more sophisticated attack scenarios within our OpenStack environment.

Acknowledgements

This work is funded by the Bavarian Ministry for Economic affairs through the WISENT project (grant no. IUK 452/002).

References

Fontugne, R., Borgnat, P., Abry, P., and Fukuda, K. (2010) “MAWILab: combining diverse anomaly detectors for automated anomaly labeling and performance benchmarking”, *Proc. of the 6th Int. Conf. on Emerging Networking Experiments and Technology (Co-Next)*, ACM, pp 8:1-8:12.

García, S., Grill, M., Stiborek, J., & Zunino, A. (2014) "An Empirical Comparison of Botnet Detection Methods", *Computers & Security*, Vol. 45, pp 100-123.

Landes, D., Otto, F., Schumann, S., and Schlottke, F. (2013), „Identifying Suspicious Activities in Company Networks Through Data Mining and Visualization”, In Rausch, P., Sheta, A.F., and Ayesh, A. (eds.), *Business Intelligence and Performance Management,* Springer, London, pp 75-90.

Małowidzki, M., Berezinski, P., and Mazur M. (2015) "Network Intrusion Detection: Half a Kingdom for a Good Dataset." Proc. of NATO STO SAS-139 Workshop, Portugal.

Otto, F., Ring, M., Landes, D., and Hotho, A. (2016) “Creation of specific flow-based training data sets for usage behaviour classification”, Proc. of the 15th European Conference on Cyber Warfare and Security (ECCWS), ACPI, pp 437-440.

Ralston, P.A., Graham, J.H., and Hieb, J.L. (2007). “Cyber security risk assessment for SCADA and DCS networks”, ISA transactions, Vol 46, No. 4, pp 583-594.

Ring, M., Wunderlich, S., Grüdl, D., Landes, D., and Hotho, A. (2017a) „A Toolset for Intrusion and Insider Threat Detection“, In Palomares, I., Kalutarage, H., and Huang, Y. (eds.), *Data Analytics and Decision Support for Cybersecurity: Trends, Methodologies and Applications*, (to appear).

Ring, M., Wunderlich, S., Grüdl, D., Landes, D., and Hotho, A. (2017b). „*Coburg Intrusion Detection Data Sets (CIDDS)"*. [online] Available at: https://www.hs-coburg.de/cidds. [Accessed 28 April 2017].
Ring, M., Wunderlich, S., Grüdl, D., Landes, D., and Hotho, A. (2017c). "*Generation Scripts CIDDS"*. [online] Available at: https://github.com/markusring/CIDDS. [Accessed 28 April 2017].
Shiravi, A., Shiravi, H., Tavallaee, M., and Ghorbani, A. A. (2012) "Toward developing a systematic approach to generate benchmark datasets for intrusion detection", *Computers & Security*, Vol. 31, No. 3, pp 357-374.
Sommer, R. and Vern, P. (2010) "Outside the Closed World: On Using Machine Learning For Network Intrusion Detection", *IEEE Symposium on Security and Privacy (SP)*, IEEE, pp 305-316.
Sperotto, A., Sadre, R., Van Vliet, F., and Pras, A. (2009) "A Labeled Data Set For Flow-based Intrusion Detection", *Proc. of the 9th IEEE Int. Workshop on IP Operations and Management (IPOM)*, Springer, pp 39-50.
Tavallaee, M., Bagheri, E., Lu, W., and Ghorbani, A. (2009) "A detailed analysis of the KDD CUP 99 data set", *IEEE Symposium on Computational Intelligence for Security and Defence Applications*, IEEE, pp 1-6.
Wheelus, C., Khoshgoftaar, T. M., Zuech, R., and Najafabadi, M. M. (2014) "A Session Based Approach for Aggregating Network Traffic Data - The SANTA Dataset", *Proc. of the Int. Conf. on Bioinformatics and Bioengineering (BIBE)*, pp 369-378.
Vasilomanolakis, E., Cordero, C. G., Milanov, N., and Mühlhäuser, M. (2016) "Towards the creation of synthetic, yet realistic, intrusion detection datasets", *Network Operations and Management Symposium* (NOMS), IEEE, pp 1209-1214.
Zuech, R., Khoshgoftaar, T. M., Seliya, N., Najafabadi, M. M., and Kemp, C. (2015) "A New Intrusion Detection Benchmarking System", *Proc. of the 28th Int. Florida Artificial Intelligence Research Society Conference*, pp 252-256.