Computer Networks 192 (2021) 108017

Contents lists available at ScienceDirect

Computer Networks

journal homepage: www.elsevier.com/locate/comnet

Unsupervised packet-based anomaly detection in virtual networks ☆

Daniel Spiekermann [a,*], Jörg Keller [b]

[a] Polizeiakademie Niedersachsen, Germany
[b] FernUniversität in Hagen, Germany

ARTICLE INFO

Keywords:
Virtual networks
Anomaly detection
Machine learning
Virtual environment

ABSTRACT

The enormous number of network packets transferred in modern networks together with the high speed of transmissions hamper the implementation of successful IT security mechanisms. In addition, virtual networks create highly dynamic and flexible environments which differ widely from well-known infrastructures of the past decade. Network forensic investigation that aims at the detection of covert channels, malware usage or anomaly detection is faced with new problems and is thus a time-consuming, error-prone and complex process. Machine learning provides advanced techniques to perform this work faster, more precise and, simultaneously, with fewer errors. Depending on the learning technique, algorithms work nearly without any interaction to detect relevant events in the transferred network packets. Current algorithms work well in static environments, but the highly dynamic environments of virtual networks create additional events which might confuse anomaly detection algorithms. This paper analyzes highly flexible networks and their inherent on-demand changes like the migration of virtual machines, SDN-programmability or user customization and the resulting effect on the detection rate of anomalies in the environment. Our research shows the need for adapted pre-processing of the network data and improved cooperation between IT security and IT administration departments.

1. Introduction

Nowadays IT environments are a key factor in our modern life. Modern data centers form the basis for our everyday digital life. Digital services play an important role, in the private life (i.e. as a backup for photos, videos and files or as a shared online calendar) as well as in many professional areas such as the financial sector, development & research or the office environment. With the evolution of cloud computing, ubiquitous use of computers, digital services and resources become more and more usual in our everyday life, which has led to a demand for faster connections and higher data rates. To fulfill these demands, modern data centers require a highly flexible infrastructure, which is adaptable without any further administrative work. The introduction of various virtualization layers like virtual machines (VM), virtual networks (VN) and virtual storage provides such an on-demand infrastructure. This evolution led to the implementation of advanced techniques like container-based environments, which create a dynamic infrastructure like the different cloud services Software-as-a-Service, Platform-as-a-Service or Infrastructure-as-a-Service [1].

Cloud service providers (CSP) use the virtualization inside their data centers to comply with the demands of their customers. Especially the use of VMs or containers improves the on-demand provision of new systems. These systems create additional flexibility in the environment. Not only the life cycle of containers or swarms have an impact on the internal dynamic, even the migration of a VM increases this adaptability. In addition, user customization for internal changes inside the VM and its assigned networks create extra benefits. But the connection of these systems with a hardware-based network hamper the necessary adaptability. Only with use of VNs the data center plays to its strength. These VNs work on an additional layer in the environment which led to the designation of an underlay and an overlay network as shown in Fig. 1.

The overlay networks perform various tasks. On the one hand, a VM needs access to the internet and maybe to different internal or external networks. On the other hand, the same VM has to be separated from VMs of other customers to ensure an isolated environment. Typically, this separation is done with virtual networks, which run on top of the hardware-based underlay network.

Protocols that create the overlay network are so-called virtual network protocols, which are used for the interconnection of the different VMs in modern infrastructures. With these protocols, VMs of one customer are connected together in a logical subnet, which is separated from other subnets of different customers. While the hardware-based

☆ This paper is an extended version of Spiekermann and Keller (2020). Revised December 2020.

* Corresponding author.
E-mail addresses: daniel.spiekermann@polizei.niedersachsen.de (D. Spiekermann), joerg.keller@fernuni-hagen.de (J. Keller).

https://doi.org/10.1016/j.comnet.2021.108017
Received 31 July 2020; Received in revised form 17 January 2021; Accepted 11 March 2021
Available online 18 March 2021

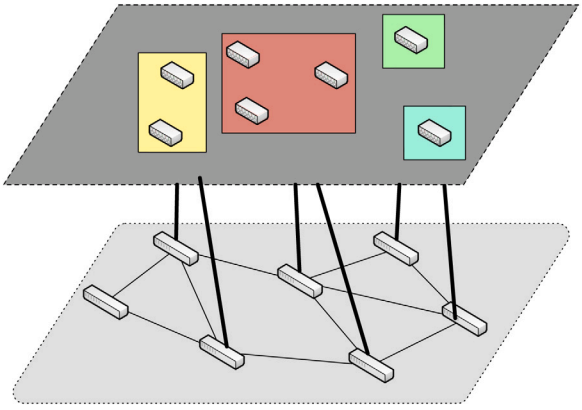

Fig. 1. Underlay and overlay networks.

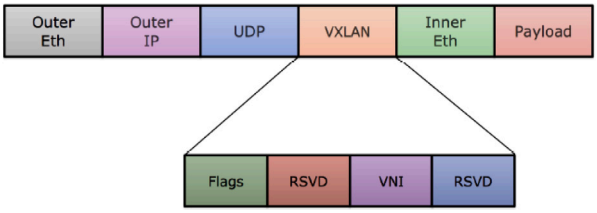

Fig. 2. VXLAN header and encapsulation.

underlay network and its addressing scheme, routing rule-set or security features could be implemented in a static manner, the virtual network provides the flexibility and dynamic needed to interconnect the VMs. Various protocols exist to implement a VN, each of them with a different focus. The first technique to implement a virtual subnet was the use of Virtual LAN (VLAN) [2]. The increasing demand in a data center led to the development of adapted protocols Generic Network Virtualization Encapsulation (GENEVE) [3], Network Virtualization using Generic Routing Encapsulation (NVGRE) [4], Stateless Transport Tunneling (STT) [5] and Virtual Extensible LAN (VXLAN) [6].

The easiest way to implement a virtual network is by using the well-known VLAN protocol that is however limited to only 4096[1] subnets, which is not sufficient in modern networks. The most notable protocol to implement these virtual networks is VXLAN, which is similar to the VLAN protocol, but expands its features and adds some new. VXLAN increases the maximum number of subnets to $2^{24} = 16,777,216$ networks by using a 24 bit virtual network identifier (VNI). Fig. 2 shows the VXLAN-header, its position in the entire frame and the use of UDP as the encapsulating protocol.

Due to the evolution of these dynamic environments, the protection of the network and the internal services gains more and more in importance. Therefore, the detection of attacks or the occurrence of anomalies in the environment is a relevant part of the IT security implementations. A modern network is target of various attacks, either from inside or from outside attackers. The types of attacks vary, from exploited misconfigurations like in 2018, when Amazon S3 buckets with more than 70 million records were leaked due to poor configuration [7], to ransomware attacks like 2019, when one of the biggest US data center providers *CyrusOne* was attacked [8]. A virtual network facilitates new vulnerabilities on its own, therefore various attacks against the virtual infrastructure exist. [9] presents *Network Harvester*, an implementation of attacking the isolation of network devices in SDN environments. They evaluated their attack with common SDN controllers like *ONOS* and *Floodlight*. [10] categorized different attacks against SDN devices and network function virtualisation (NFV) and summarized them as related to *Network function virtualisation, virtual layer, orchestrator manager* and *virtualized infrastructure manager*. [11] describes the implementation of covert channels inside a virtual environment and analyzes the possibility of data hiding in network protocols like VXLAN or GENEVE. These attacks show the need for an effective protection of the environments. By using different security mechanisms like firewalls or intrusion detection systems, providers try to increase their overall IT security and counter these kinds of attacks. But the increasing number of network packets transferred inside the environment in combination with the high speed of connections as well as the huge amount of attacks make this task complex and expensive.

Advanced techniques like machine learning (ML) try to support the provider by the detection of anomalies in the network traffic which might be an indicator for unknown attacks against their infrastructure.

ML and its impact on cyber security is a fast growing research area, which results in the definition of different algorithms and an improved analysis of unknown data. One of the most important parts for ML in networks is the detection of anomalies, which [12] defines as

> *... the problem of finding patterns in data that do not conform to expected behavior.*

The detection of anomalies in the network is part of classification problems [13]. This type of problem can be described by classifying data points to given categories [14]. A wide-spread classification in IT security is the detection of SPAM [15]; here an incoming e-mail is checked against a set of features. To analyze an e-mail correctly, the classifier has to be trained with benign and malicious data, hereby it learns parameters which indicate SPAM mails.

Anomaly detection in networks tries to find changes concerning e.g. the mix of packets in a network. This might be an indicator for the beginning of an attack or a current data leakage [16]. The detection of covert channels with the help of ML is a recent research area [17]. Modern malware uses covert channels to transfer their payload or to exfiltrate sensitive data [18]. The detection of such attacks requires good knowledge of the traffic which is typical in this environment. An outlier of this known traffic might be an indicator for a security issue, therefore it calls for additional investigation of such traffic.

Because of this, machine learning algorithms heavily depend on an environment with some mostly static parameters. Common classifiers use network flows or parts of protocol headers to create a benchmark data-set, which is used to train the algorithm. If a certain level of deviations is reached, e.g. if a threshold is exceeded, the classifier will detect an anomaly and start a pre-defined process like logging or alerting.

Unfortunately, changes are inherent in virtual networks, so a machine learning algorithm will produce various false-positive messages, which therefore demand a reconfiguration of the training sets. This is a common task in company networks and not a new problem in IT security [19]. But changes in virtual networks occur much more frequently. Additionally, these changes are mostly unpredictable, because both administrator and customer are able to adapt their part of the network. The administrator might change some parameters of the virtual environment like the deployed protocol or only some protocol fields. The customer might change the internal IP addresses used in the assigned logical subnet. Because of the separation of the layers, both changes should not have any impact to the other part of the network.

Hence, we investigate the impact of virtual networks on the detection capabilities of machine learning algorithms for malware detection by finding network anomalies. We do this by creating a cloud computing environment based on OpenStack with various virtual network protocols that transfer specific information in this test environment. We capture and analyze the traffic with different ML algorithms focusing on anomaly detection.

Our contributions can be summarized as follows:

[1] VLAN headers use 12 bit VLAN IDs to implement up to $2^{12} = 4096$ subnets.

- We point out challenges for machine learning algorithms when used for forensic anomaly detection in *virtual* networks.
- We identify changes in virtual overlay and underlay networks that may point to anomalies like malware.
- We perform simulations of virtual networks to create data sets of captured network packets with above changes that can be used to parameterize, train and evaluate forensic machine learning algorithms for virtual networks.
- We evaluate forensic machine learning in virtual networks with above data sets.

The remainder of this paper is structured as follows. In Section 2 we summarize related work from virtual network forensics and machine learning for anomaly detection. Section 3 describes the methodology for data collection and analysis, together with relevant changes in virtual networks that define the data to collect. The implementation of the deployed ML algorithm is discussed in Section 4. In Section 5 we evaluate the different situations and their impact on anomaly detection based on IsolationForest and LocalOutlierFactor. Section 6 concludes this paper and gives an outlook to our future research.

2. Related work

Virtual networks and modern data centers necessitate a change of the well-known methods of digital investigation in hardware-based networks as discussed in [20,21]. [22] describes the arising problems of network forensic investigation in virtual networks. [23] defines an SDN model usable to perform secure network forensic investigation in nowadays data centers, especially when they are distributed over different locations. The need for a special process to implement valid investigation in modern environments is discussed in [24]. A further discussion about the problems of packet captures for law enforcement in modern data centers is discussed in [25]. An important task in a data center's security strategy is the detection of abnormal behavior inside the transmitted network packets [26]. So, the anomaly detection is a part of forensic investigation, but in some cases the results of such a monitoring has to be accessible much faster. [27] discusses the use of machine learning aspects as an implementation of *automated network forensics*. [28] discuss problems and countermeasures of anomaly detection in big data networks. The most notable protocol in modern data centers apart from ethernet is the *Internet Protocol*. [29] analyzes various sources of network data like routing or management protocols and special network probes.

The research of anomaly detection in virtual networks is thin. [30] describes the detection of distributed denial of service (DDoS) attacks in virtual networks with the help of the network analyzer *Bro*. The analysis results are used to configure parts of the virtual network with the help of OpenFlow. Anomaly detection in modern networks is discussed in various papers. [31] describes the detection of anomalies in a small test environment based on pre-defined network packets and measures the impact of different changes in the network. [32] proposes the use of IP-flow records for anomaly detection with Support Vector Machines (SVM), which improves the performance in high speed networks. The detection of anomalies or outliers is a crucial task in modern networks, and as an internal component a detection system might be faced with adversarial attacks. [33] presents a survey of adversarial attacks against intrusion detection systems. The authors define six different goals, namely *evasion, poisoning, over-stimulation, denial of service, response hijacking* and *reverse engineering*. [34] defines *evasion* and *poisoning* as the most relevant attack models against machine learning algorithms.

In contrast to the aforementioned research, our present research tries to identify changes in virtual networks originating from anomalies such as malware in a systematic manner, and to create data sets of captured networks packets in virtual networks. We use those data sets to evaluate forensic machine learning in virtual networks.

3. Anomaly detection in virtual networks

Anomaly detection is a critical task in modern networks, either to detect advanced attacks like covert channels or DDoS. Occurring anomalies in the network might be an indicator for malicious behavior, so using ML to detect such behavior is a common technique. Whereas anomaly detection in traditional networks is well researched, detection algorithms are faced with new challenges in virtual networks.

For a structured procedure, we first introduce the process model that we follow. Our research focuses on the impact of virtualization in a network and the inherent changes on ML algorithms that are used to detect anomalies in the network. A virtual environment provides a huge flexibility on different layers. Both overlay and underlay might create relevant changes in the network infrastructure, which therefore leads to a measurable impact for anomaly detection. Hence, we also analyze possible changes in overlay and underlay networks, which allows to define the data collection in Section 4.

3.1. Process model

A successful implementation of ML for network forensic investigation like anomaly detection, malware analysis [35] or event reconstruction [36] depends on various parameters like valid packet captures, correct data extraction and the use of a suitable ML algorithm. An established method in digital investigationo ensure the correctness of a digital investigation is the use of so-called process models or frameworks which define the necessary steps, mostly separated in different phases. Whereas different frameworks for anomaly detection in networks exist [37,38], there is no specific framework with a special view regarding the dynamic of a virtual network. As shown in [39], digital investigations in a virtual network require adapted frameworks which are able to manage the flexibility of the environment.

We propose the use of the process model defined in [40]. The authors define six steps to implement ML for network analysis:

- **Problem formulation**
  In this phase the investigator defines various parameters of the analysis. ML algorithms are often time-consuming, so a detailed definition of necessary input data and ML categories is quite relevant for the subsequent steps.
- **Data collection**
  In this phase the relevant packets are captured. In traditional networks, this step is easy to implement [20,22], but virtual networks increase the complexity of network packet capture processes [41].
- **Data analysis**
  This phase comprises all necessary steps to transform the captured packets into a usable format for the subsequent steps. The captured data might be stored in raw, pcap or pcap-ng-formats[2] and transformed into formats like *Netflow*, *sFlow*, *csv*, *json* or other user-defined structures. Typically, these techniques do not store the entire network packet but various header information from different layers of the OSI model. Whereas Netflow and sFlow use layer 3 and layer 4 protocols, the other formats might extract information from all other layers. The definition of the necessary data depends on the intended analysis.
- **Model construction**
  The construction involves the training, testing and tuning of the learning model.
- **Model validation**
  In this phase the model is validated to ensure its quality. If errors occur or improvements are needed, all prior tasks are involved to eradicate these issues.

[2] In addition to these formats, various vendor-dependent file formats exist.

- **Deployment and inference**
  This phase comprises all relevant steps to implement the ML process in the operational environment with a focus on resource usage, accuracy and performance.

Problem formulation is covered in the next subsections. Data collection is described in detail in Section 4. Model construction and validation is done in Section 5. We did not focus on the last phase of deployment and inference in depth, because we limit our approach to the analysis of different changes without an optimization of the deployed algorithm or the selection of the best algorithms.

3.2. Changes in overlay network

The part of the overlay protocol is defined by the internal networks, which are under administration of the user. By this the customer is able to change different settings of a VM or the assigned network on his own, which might result in relevant changes.

- **Internal IP addressing**
  Typically, the internal network uses private IP addresses from a predefined subnet as described in [42]. The user of this network is able to change this internal addressing scheme without involving the CSP or any administrator of the cloud environment.
- **VM Life Cycle**
  A VM runs inside the virtual environment and is under control of the customer. So, the customer is able to start and stop virtual machines on his own, which leads to an unpredictable behavior affecting every IT security feature focused on this part of the network.
- **Addition or Deletion of VMs**
  A customer is able to start new VMs within seconds and to connect them to the internal network. Typically, this task is initiated by using the web interface of the cloud environment. If a new VM is started, the internal network changes.

3.3. Changes in underlay network

A cloud service provider (CSP) is free to change the underlying network whenever needed, which might lead to a fully different network behavior without affecting the virtual network of the users.

- **Overlay protocol**
  Overlay protocols like VXLAN, GENEVE or STT are used to create the different isolated and separated networks of the different customers. If the CSP changes the separation protocol of the internal virtual networks from VXLAN to GENEVE, this only needs a quick change of the internal transfer mechanism. A possible change is the use of an improved protocol like STT, which implements the use of special features of the network interface card. By this the resource usage of the CPU is reduced, which improves the overall performance of the network.
- **Migration**
  A huge benefit of virtual environments is the migration of VMs. This can be done for availability reasons, e.g. when the hosting server has a critical problem which requires a reboot, or for providing an improved environment, e.g. when different VMs on a single host demand for a higher CPU usage. If a VM is migrated, the system is moved from one hosting server to another. In case of such a migration, some parameters of the VM (like the IP address or MAC address) do not change, but the underlying structure of the network needs some adaptations to create this flexibility. This might include the creation of a separate tunnel depending on the protocol as well as the deletion of existing ones.

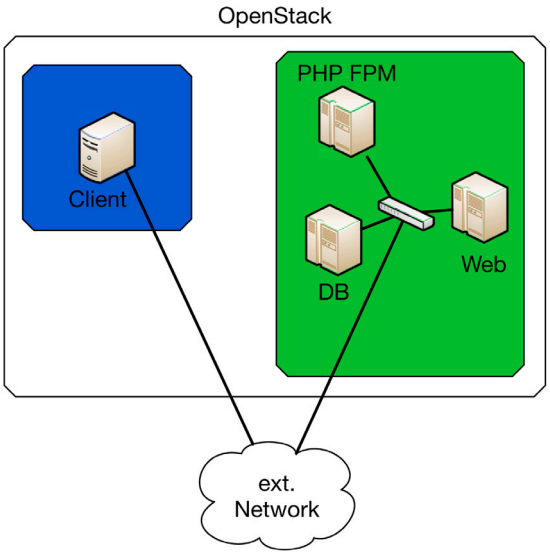

Fig. 3. Logical OpenStack installation. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

- **Programmability of the network**
  The use of SDN inside a virtual environment provides additional flexibility in the infrastructure. By implementing new or adapted features, the network is able to change its behavior on demand. SDN decouples the traffic management from the traffic forwarding. An SDN controller manages the flows of network traffic inside the environment by communicating with connected devices via the so-called southbound-API. The most notable protocol for this communication is OpenFlow. By adding or deleting OpenFlow rules, a packet flow inside the network is adjusted, which provides a huge flexibility and dynamic in the environment. This programmability is used to implement different applications like network management [43], security [44], quality of service [45] or network forensic [39].

4. Implementation

This section describes the collection of relevant network data, the definition of our packet based feature set and the algorithms used for anomaly detection.

4.1. Data collection

In [31] we analyzed the impact of virtual networks when using machine learning for anomaly detection inside a simple network infrastructure. To measure the impact in a more realistic scenario, we extend the original test environment and created a cloud environment based on OpenStack with two user networks as shown in Fig. 3. This environment provides all relevant aspects which facilitate the analysis of the aforementioned occurrences. The virtual environment is installed on three Dell Poweredge R6415, each with 32GB RAM and three network interface cards (NIC). One NIC of a server is dedicated to the tenant network, the connection between the servers is done with a Cisco WS-C2960X-24PS-L gigabit ethernet switch. We installed Ubuntu 18.04 LTS as the underlying operating system with OpenStack release Train. As the SDN controller, we use OpenDaylight. The deployed protocols VXLAN, GRE and GENEVE are the relevant ones of the most notable cloud environment solution OpenStack [46].

In the green network in Fig. 3, we installed a webserver based on *nginx* as the frontend and a *MySql* database as the backend. The webserver hosts two different files, a single html document and a php script

that collects some information from the mysql database running on the different system. To increase the number of packets in the network, we added a dedicated VM providing the *php-fpm* functionality. By this setup, a single request for the php script results in the communication with the php-fpm service, and after that with the database.

The blue network contains a single VM that repeatedly, with random intervals in-between, requests an internet site from the top 500 websites[3] via curl. This produces common network traffic, which is normal in cloud environments.

All administrative work is done via the web interface of the OpenStack installation. We focus on the so-called tenant networks which define the networks for the VMs and the customers. All other networks like backup or management are not in the scope of this paper. This infrastructure provides a low-level environment, which helps to focus on the impact of the changes. We assume that an implementation of networks with more VMs or connections would not result in different detection rates, but hamper the analysis due the existence of more network packets stored during the collection process.

At first, we validated the usability of the environment by capturing the network data on two different layers of the network. A capture process in the overlay network results in a packet capture without any overlay information, whereas a packet capture performed in the underlay network gathers all involved protocols, which includes the overlay protocol.

Fig. 4 shows a packet capture without any encapsulation information. In this case there are information of layer 2 (starting with Ethernet II), layer 3 (starting with Internet Protocol Version 4 (IPv4)) and the transmitted data (in this case the Internet Control Message Protocol) available. Fig. 5 displays a packet capture of the same network packet, but this time encapsulated with VXLAN, which is used by UDP. By this, there are two ethernet headers and two IPv4 headers in one captured packet.

Whereas some connection-specific details are different, the timestamp of the ICMP packet is the same in both captures. So, the capture processes create two files with the same network information encapsulated in different layers. For the further processing of the data, we performed some data sanitization for the per-packet analysis. To remove irrelevant packets, we removed all ARP packets from the capture files due to the fact that these do not contain any relevant information.

To create some anomalies, we injected some crafted network packets, which use illegal combinations of network headers. For creating those packets, we used scapy,[4] a python framework for the creation, manipulation and capturing of network packets. The packets were designed with the following variations (alone and in combination).

- Irregular value of IP version (set to 5)
- Random IP protocol number
- Unusual combination of TCP flags (RST, ACK, FIN) / (SYN, PSH, RST)
- Unused IP addresses

We injected these packets into the network using *tcpreplay*, so that they are recorded during the capture process.

The use of the crafted network packets provides a simplified implementation of the possible changes as mentioned in Section 3.2. Table 1 shows the implementation details, which are used to create a specific event to produce a change inside the network.

[3] https://www.alexa.com/topsites.

[4] www.scapy.net.

Table 1
Implementation of events.

Occurrence	Event	Implementation
Overlay	Internal IP addressing scheme	Adaptation inside the VM
Overlay	VM life cycle	Via web interface
Overlay	Addition or deletion of VMs	Crafted network packet
Underlay	Migration	Via web interface
Underlay	Change of the protocol	Adaptation of overlay protocol
Underlay	SDN programmability	Adaptation of overlay protocol

4.2. Feature set

The definition of the correct features to train the ML algorithms heavily depends on the goal of the analysis. So, the selection of significant parameters which differ benign from malicious traffic is still a difficult process, which depends on various parameters and aspects of the network. Different research uses feature extraction algorithms based on the network traffic to define a usable feature set. [47] describe the feature selection with the help of the Poisson Moving Average (PMA), [48] describes the use of Linear Discriminant Analysis (LDA) and Principal Component Analysis (PCA). But the use of these algorithms might have some drawbacks, especially the small size of the data set might result in an insufficient data set. Because of this, we focus on a manual selection of features to be used.

The information used in the feature set is variable, and might contain various information like packet details of the different layers, results of deep packet inspection [49] and time or flow-specific values [50].

The use of network flows improves the results and the time needed for the analyses by limiting the amount of information used for the algorithm. A flow is a group of packets with common protocol details like IP addresses or port numbers. So, a flow does not contain any application data, thus this limits the traffic that has to be analyzed in contrast to a per-packet analysis [51]. Common analysis based on network flows use the following five fields [52]:

- Source IP address
- Destination IP address
- Source port number
- Destination port number
- Layer 3 protocol type

In contrast to this, [53] add statistical parameters to the feature set. Especially the number of transmitted bytes are considered as relevant. We use a mostly predefined communication scheme, so the amount of traffic is static to align the different scenarios. Timing parameters are relevant in productive networks, but our test bed is based on a internal LAN, so timing parameters like duration of the flow or delta between packets are negligible. Because of this, we focus on the use of packet details.

In [54] a first feature set of 23 parameters is considered to detect DDoS attacks. After measuring the importance a set of eight parameters were defined as necessary. In contrast, [55] uses only four features consisting only of the parameters source and destination IP address and combinations of them. Advanced attacks like covert channels do not only use application protocols like HTTP or DNS, but implement their information in lower level protocols like IP or TCP.

So, all aspects of *Ethernet*, *IP* and *TCP/UDP* as well as generic values like the length of the frame are relevant. These changes cover modifications on the underlay network, while changes inside the virtual networks typically concern changes of the IP addresses used. The use of length parameters of the protocols like *frame.len* and *ip.len* is useful for the detection of various covert channels as discussed in [56]. So, there is no defined list of relevant protocol fields to be used for anomaly detection in network forensic investigation. In contrast to our research in [31], we changed the deployed feature set. The first approach was

```
▶ Frame 17: 98 bytes on wire (784 bits), 98 bytes captured (784 bits)
▶ Ethernet II, Src: 2e:99:39:18:2d:17 (2e:99:39:18:2d:17), Dst: 0a:31:b6:b8:a5:bc (0a:31:b6:b8:a5:bc)
▶ Internet Protocol Version 4, Src: 192.168.1.2, Dst: 192.168.1.1
▼ Internet Control Message Protocol
    Type: 8 (Echo (ping) request)
    Code: 0
    Checksum: 0x3f6b [correct]
    [Checksum Status: Good]
    Identifier (BE): 7414 (0x1cf6)
    Identifier (LE): 63004 (0xf61c)
    Sequence number (BE): 107 (0x006b)
    Sequence number (LE): 27392 (0x6b00)
    [Response frame: 18]
    Timestamp from icmp data: Jul 27, 2020 11:38:58.000000000 CEST
    [Timestamp from icmp data (relative): 0.417511000 seconds]
  ▶ Data (48 bytes)
```

Fig. 4. Packet without VXLAN encapsulation.

```
▶ Frame 34097: 148 bytes on wire (1184 bits), 148 bytes captured (1184 bits)
▶ Ethernet II, Src: Apple_b0:6d:4c (9c:f3:87:b0:6d:4c), Dst: Vmware_5b:cd:71 (00:0c:29:5b:cd:71)
▶ Internet Protocol Version 4, Src: 192.168.10.157, Dst: 192.168.10.215
▶ User Datagram Protocol, Src Port: 41394, Dst Port: 4789
▶ Virtual eXtensible Local Area Network
▶ Ethernet II, Src: 2e:99:39:18:2d:17 (2e:99:39:18:2d:17), Dst: 0a:31:b6:b8:a5:bc (0a:31:b6:b8:a5:bc)
▶ Internet Protocol Version 4, Src: 192.168.1.2, Dst: 192.168.1.1
▼ Internet Control Message Protocol
    Type: 8 (Echo (ping) request)
    Code: 0
    Checksum: 0x3f6b [correct]
    [Checksum Status: Good]
    Identifier (BE): 7414 (0x1cf6)
    Identifier (LE): 63004 (0xf61c)
    Sequence number (BE): 107 (0x006b)
    Sequence number (LE): 27392 (0x6b00)
    [Response frame: 34098]
    Timestamp from icmp data: Jul 27, 2020 11:38:58.000000000 CEST
    [Timestamp from icmp data (relative): 0.417351000 seconds]
  ▶ Data (48 bytes)
```

Fig. 5. Packet with VXLAN encapsulation.

Table 2
Deployed feature set.

Feature	Layer	Description
frame.len	–	Length of the entire frame
*e*th.src	2	MAC address of the source
*e*th.dst	2	MAC address of the destination
*e*th.type	2	Protocol ID of the upper layer
ip.version	3	Version (4 or 6) of the IP packet
ip.len	3	Length of the IP packet
ip.flags	3	Flags of the IP protocol header
ip.fragment	3	Fragmentation of the packet
ip.dst	3	IP address of the destination
ip.src	3	IP address of the source
ip.proto	3	Protocol ID of the upper layer
ip.tos	3	Type of service
udp.srcport	4	Destination port of the UDP datagram
udp.dstport	4	Source port of the UDP datagram
tcp.srcport	4	Destination port of the TCP datagram
tcp.dstport	4	Source port of the TCP datagram
tcp.flags	4	Flags of the TCP datagram
tcp.urgent_pointer	4	Urgent flag of the TCP datagram

reduced to application information related to the used ICMP protocol. The analyses in this research focus more on a realistic environment, so we need to adapt the used features. Similar to our first analysis our network bases on an internal LAN without any effects from external networks like packet loss, changing transmission times or different routes between the involved hosts, so all timing parameters as well as the Time-to-live of the packets were discarded. Table 2 lists our feature set, the name of the feature derives from the display-filter name used by *Wireshark*.

We implement a packet-based anomaly detection in contrast to a flow-based approach due to the fact that we want to measure the impact of small changes in the network. A reconstruction of the flows in a virtual network requires the focus on the traffic inside the tenant network, otherwise the UDP-based encapsulation of the VXLAN protocol led to similar flows, which might irritate the algorithms.

4.3. Outlier detection

Anomaly detection with ML is part of supervised learning and can be done by a classification of the different values [57]. The algorithms use different methods to calculate the anomaly score of the different changes. Common algorithms are *Decision Trees*, *Support Vector Machine*, *k-nearest Neighbors* (k-NN) and *probabilistic methods* [14,58]. The detection of outliers in network traffic can be performed in two different ways.

- The first step to detect outliers in network traffic is the definition of normal or benign traffic and later the comparison of the network data to this baseline of the data. k-NN algorithms implement this kind of calculation by computing a local density deviation of a given data point with respect to its neighbors. An outlier has a lower density than its neighbors. LocalOutlierFactor (LOF) [59] is an algorithm which uses this density.
- Algorithms like IsolationForest (IF) [60] define anomalies as

  > *. . . 'few and different', which make them more susceptible to isolation than normal points.*

  . Instead of trying to build a model of normal instances, it explicitly isolates anomalous points in the data set. IF algorithms as a branch of *Random Forests* detect outliers by randomly selecting features and isolating them by a value between the minimum and the maximum values of this feature.

Table 3
Packet count for each scenario.

Occurrence	No. of packets		
	Before change	After change	Overall
Internal IP addressing	5,612	4,699	10,311
VM life cycle	6,897	6,302	13,199
Addition or deletion of VM	4,918	5,523	10,441
Change of overlay protocol	6,494	6,392	12,886
Migration	6,883	7,318	14,201
SDN-programmability	8,893	8,022	16,915

5. Evaluation

To evaluate our results, we used capture files produced by the data collection phase. The number of packets in each scenario is listed in Table 3.

We use the *IsolationForest*-algorithm of the scikit-learn-framework,[5] which provides the function *IsolationForest.predict()* as an indicator whether a packet is classified as an anomaly or as a normal behavior. This value marks an anomaly with the score of *-1* and a normal behavior with *1*.

As an application of the first approach, we implement LOF, which calculates a value of *1* if it is a normal behavior, and a value between *1.2 and 2.0* if it is an anomaly. *sklearn* provides *LocalOutlierFactor.negative_outlier_factor_* as an indicator for the outlier detection. IF and LOF are unsupervised algorithms, so we do not need any training data, and each packet of each capture file was analyzed by *IsolationForest.predict()* and *LocalOutlierFactor.negative_outlier_factor_*. In addition to this, we focus on a detection of changes in the network and their identification as an anomaly. The impact of changes in a virtual network and the detection as an anomaly as a result of such a change is the main part of this evaluation, therefore we did not focus on the improvement of the detection algorithms and did not evaluate the correct classification of every network packet.

To validate the detection, we repeated the research of [31], and analyzed the detection of outliers based on the ICMP messages without any overlay protocols. Both algorithms, IF shown in Fig. 6 as well as LOF shown in Fig. 7, detect the ICMP messages successfully as outliers, which validates our implementation. Fig. 6 shows the result of the *IsolationForest.predict()*, the bar around $-0,2$ defines the existence of abnormal packets. Fig. 7 defines the existence of abnormal packets with a red circle, which marks values that have a lower density than their neighbors. The x-axis represents the network packets based on its values, and the y-axis defines a value of anomaly based on the density. The radius r of the red circle is the outlier score calculated by LOF. The calculation is done with

$$r = \frac{(LOF\_Scores.max() - LOF\_Scores)}{(LOF\_Scores.max() - LOF\_Scores.min())}$$

IF provides two scores to classify a value as an anomaly. By using the *IsolationForest.predict()*-function, IF marks an anomaly as *-1*. The use of *IsolationForest.score_sample* results in a negative value. The larger the absolute value, the more abnormal is the value. The analysis of the ICMP packets shows an anomaly score of *-0.59377377*, which defines an abnormal network packet. As the value is not close to -1, it might be an indicator that this network traffic is not easily detected as an anomaly.

LOF calculates anomalies with a score of *-1* when using *LocalOutlierFactor.negative_outlier_factor_* [61], thus the network packets with the IP protocol number of 1 (which defines ICMP packets) are marked as an anomaly. The ICMP packets are marked with *-1*, whereas the TCP datagrams as the layer 4 protocol are marked as normal packets with a value of *1*.

[5] Details can be found at: https://scikit-learn.org/stable/.

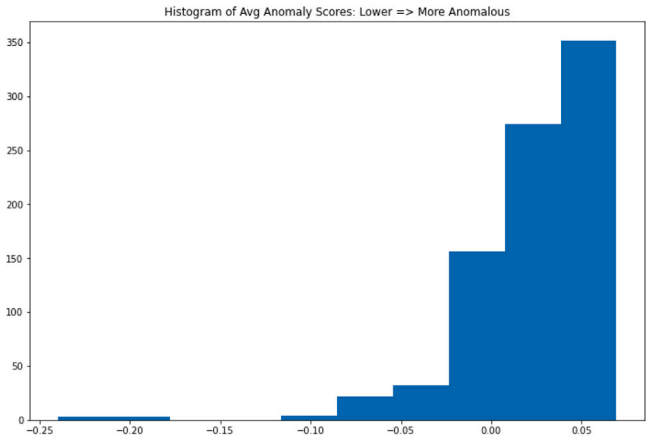

Fig. 6. IsolationForest.

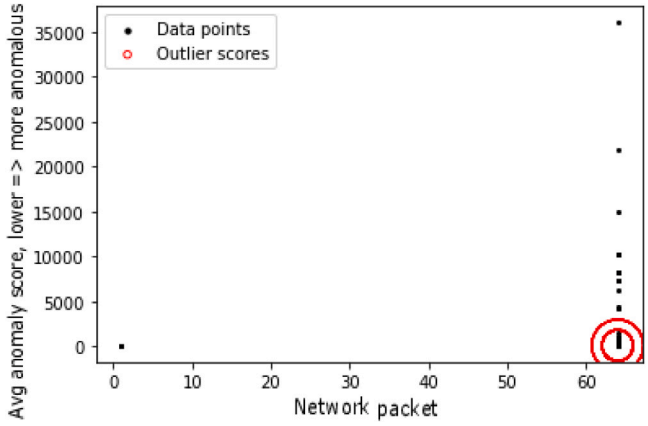

Fig. 7. LocalOutlierFactor.

Both algorithms detect the anomalies in our data set correctly. In addition to this, the crafted network packets were successfully identified as an anomaly by both algorithms, too. Therefore, we focus on the use of IF and the resulting graphs for the next steps.

The next step was the analysis of the data with overlay protocols, but without any changes in the structure as discussed in Section 3.2. Both techniques successfully detect the ICMP packets and the crafted packets as an anomaly, both with small changes to the aforementioned analysis, shown in Fig. 8 (the bars around $-0.2 \rightarrow -0.15$ define the existence of abnormal packets) and Fig. 9.

The next experiments analyze the different changes as discussed in Section 3.2.

- **VM migration**
  Due to the fact that OpenStack does not support any kind of live migration, we created a snapshot of the VM and restarted the frozen VM on another host. There were no detectable changes in the overlay network, but the underlay network creates a new VXLAN tunnel, whereby a new virtual tunnel endpoint (VTEP) is created. This new VTEP acts as an additional system appearing in the network capture of the underlay network. This results in a small impact on the lower values shown in Fig. 10. As described for Fig. 6, bars at lower values define the existence of anomalous packets.
  The injection of the crafted network packets has a smaller impact of the detection as shown in Fig. 11, because some of the values in these network packets intentionally collide with the effects of the additional system. Especially the additional IP addresses used by the crafted packets might irritate the algorithm.

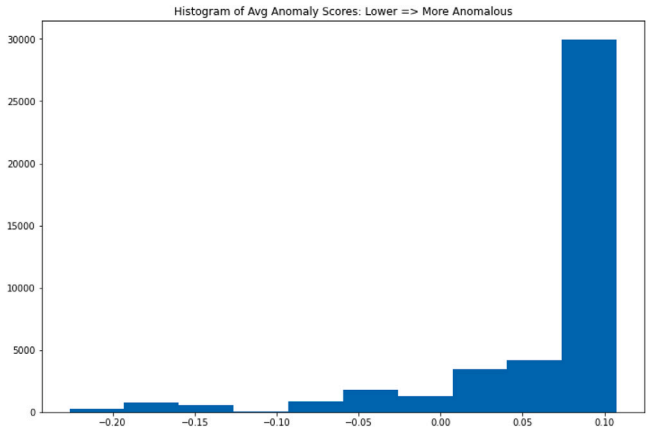

Fig. 8. IsolationForest — virtual network.

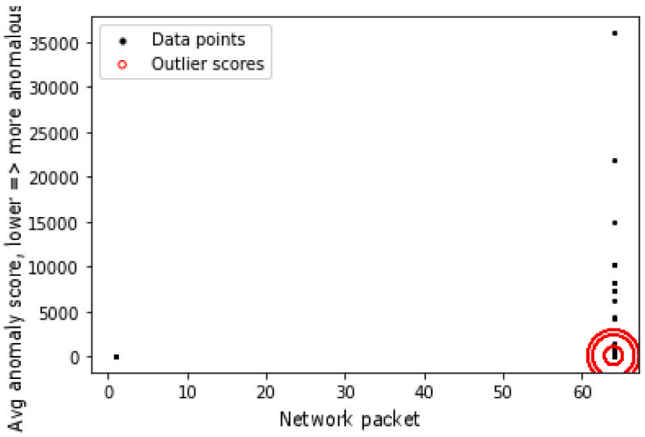

Fig. 9. LocalOutlierFactor — virtual network.

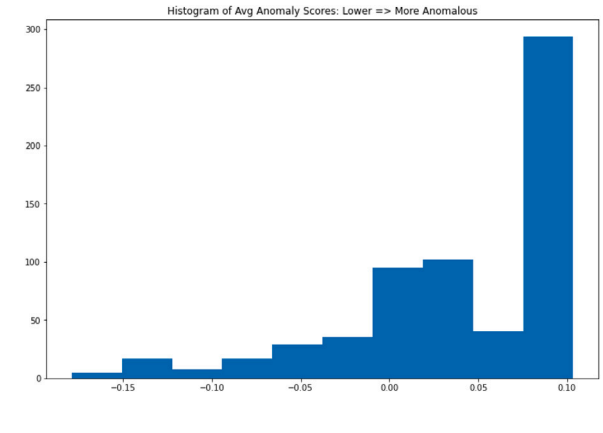

Fig. 10. Additional VTEP.

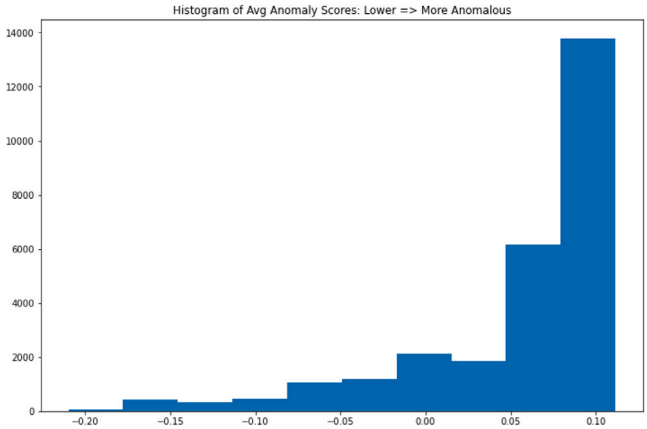

Fig. 11. Additional VTEP and crafted network packets.

- **VM Life Cycle**
  The life-cycle of a VM changes the internal structure of the virtual customer network. Even when a VM is stopping or rebooting, internal identifiers like the IP address or MAC address are still the same. The only necessary adaptation is done in the overlay network, when the VM is connected again to the network. In this case, the network traffic is similar to the traffic related to VM migration. Due to the fact that a VM after a shutdown or reboot might appear as the same device concerning the IP address and MAC address, there is no detectable impact of this process.
- **Addition or Deletion of VMs**
  In contrast to the VM life cycle, the starting or permanent deletion of a VM results in a change within the network. A deletion might lead to a change in the mapping between MAC address and IP address; if another VM is started this system might get the previously assigned IP address. This process is similar to the VM migration, because the location of the new VM is randomly in the network. In detail, there are some different aspects compared to the VM migration process, which affect different parameters of the communication. For example, a new location of a migrated VM might result in changed network details like timing parameters or traffic volume. But these values are not detectable by our packet-based approach.
- **Change of the protocol**
  To simulate the change of the deployed overlay protocol, we changed from VXLAN to GENEVE during the packet transmission on the fly. As shown in [31], such a change has a measurable impact on anomaly detection. The result is again a detection of anomalies in the underlay network, shown in Fig. 12. To clarify the anomaly, Fig. 13 shows the detection of anomalies when the protocol is changed back from GENEVE to VXLAN on the fly. The different heights of the bars might emerge from the processing of IF, which uses different values for the beginning of the calculation.
- **SDN-programmability**
  The programmability of the network based on SDN provides a huge flexibility of the network. As a result of a new programming of the network, flows inside the network might be redirected, reconfigured or replaced with other packets. To evaluate the SDN functionality, we implement a simple configuration change, which redirects the traffic originally sent to the web server to an additional web server, which acts as a simple proxy system. This alters the network traffic, but the arising changes are the same as VM addition or deletion, e.g. a new system is available in the network.

Different scenarios as discussed in Section 3.2 might appear in virtual environments and on different layers of the network, but the impact on the process of anomaly detection is sometimes similar.

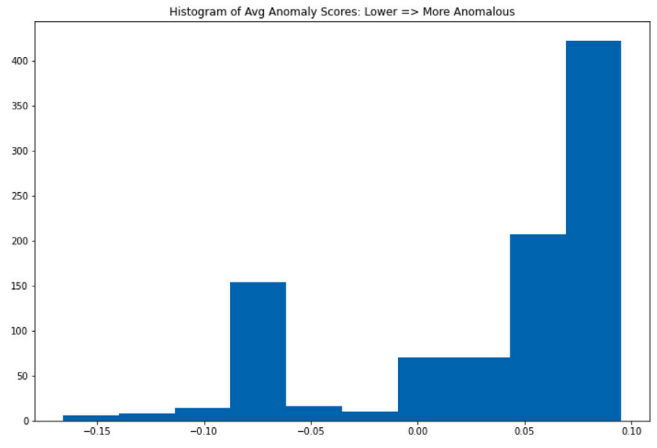

Fig. 12. Change of underlay protocol from VXLAN to GENEVE.

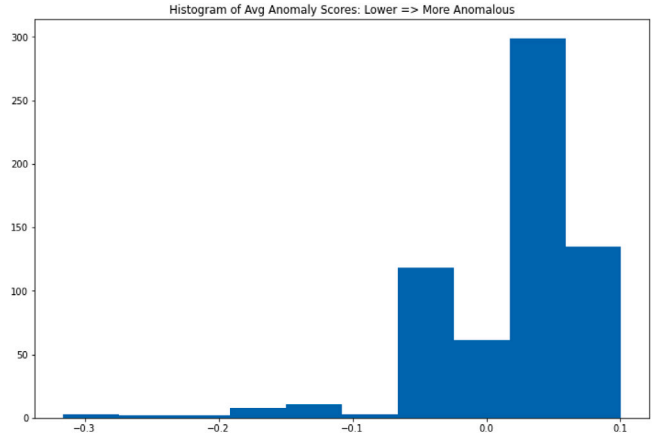

Fig. 13. Change of underlay protocol from GENEVE to VXLAN.

Especially all processes that create additional systems (VMs as well as VTEPs) have a similar impact on the ML algorithms. As we focus on the detection of anomalies, we did not calculate any further metrics like precision, recall and F-measure, which are typically used in the evaluation of machine learning algorithms [62,63].

All changes in virtual networks result in a modification of the environment. So, an occurring change is, depending on the intended analysis, detected as an anomaly which demands for an adapted method of anomaly detection. Only this way might eradicate the alleged anomalies and thereby do not hamper the detection of relevant anomalies.

6. Conclusion and future work

Cloud environments are highly used networks and provide a great flexibility, either for the user as well as for the provider. Modern networks provide various virtualization techniques to create a highly flexible and dynamic environment. Whereas this customizability creates benefit for the administrator or the user, IT security processes are hampered when using ML. These ML processes require a valid data set used for training the ML algorithms, but this is not guaranteed in modern networks. The appearance of various changes in the environment might lead to different effects like false positives or false negatives. Changes like VM migration or user customization cover other issues in the network and endanger the detection of real anomalies.

This paper analyzes unsupervised packet-based anomaly detection with two different algorithms. IsolationForest as well as LocalOutlierFactor detect arising changes in the network and are therefore a suitable technique to detect outliers in highly used networks. We defined possible changes on two different layers of the virtual environment and evaluated the algorithms in a realistic scenario based on an OpenStack cloud.

Changes in this network are relevant and might occur quite often. As shown in Section 5, these benign changes are detected as anomalies and therefore impede the detection of real attacks or anomalies in the network. To cover this problem, the provider needs to adapt its implemented algorithms and focus on the possible changes. As the changes might occur on all levels of the OSI model, a limitation to specific parameters is not usable. Thus, a periodic redefinition or sanitization of the network traffic is necessary to overcome these issues.

Our future work will be focused on the analysis of large packet captures like *UNSW-NB15* [64] in virtual environments and the comparison of packet-based and flow-based feature sets. Modern networks provide high speed connections, therefore a real time packet-based anomaly detection has to be fast enough, to capture and analyze each network packet in time. In our scenario, the network speed was 1 gigabit per second, but the network traffic was limited to only a few involved systems. In addition to this, we firstly captured the data in the so-called online phase, and analyzed this data in a subsequent offline phase. By this, the performance of the analyzing system is not relevant for the intended process. We will improve the analyzing part of the approach by further investigations to filter the network traffic and analyze only relevant information. Furthermore, as a simulated scenario, our approach is not protected against any model of adversarial attacks, so we will harden future implementation against attacks like evasion or poisoning.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgment

This work is supported by project SIMARGL (Secure intelligent methods for advanced recognition of malware, stegomalware & information hiding methods, https://simargl.eu), which has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 833042.

References

[1] M. Guzek, P. Bouvry, E.-G. Talbi, A survey of evolutionary computation for resource management of processing in cloud computing, IEEE Comput. Intell. Mag. 10 (2) (2015) 53–67.

[2] Institute of Electrical and Electronics Engineers, IEEE standard for local and metropolitan area network–bridges and bridged networks, 2018, 802.1Q-2018 (Revision of IEEE Std 802.1Q-2014).

[3] J. Gross, GENEVE: Generic network virtualization encapsulation, 2019, http://tools.ietf.org/html/draft-davie-stt-06.

[4] P. Garg, Y. Wang, NVGRE: Network virtualization using generic routing encapsulation, 7637, 2015, http://www.rfc-editor.org/rfc/rfc7637.txt.

[5] B. Davie, J. Gross, A stateless transport tunneling protocol for network virtualization (STT), 2014, http://tools.ietf.org/html/draft-davie-stt-06.

[6] M. Mahalingam, D. Dutt, K. Duda, P. Agarwal, L. Kreeger, T. Sridhar, M. Bursell, C. Wright, Virtual extensible local area network (VXLAN): A framework for overlaying virtualized layer 2 networks over layer 3 networks, 7348, 2014, http://www.rfc-editor.org/rfc/rfc7348.txt.

[7] Symantec, Internet security threat report 24, Tech. rep., Symantec, 2019, https://www.symantec.com/content/dam/symantec/docs/reports/istr-24-2019-en.pdf.

[8] C. Cimpanu, Ransomware attack hits major US data center provider, 2019, https://www.zdnet.com/article/ransomware-attack-hits-major-us-data-center-provider/.

[9] R. Xiao, H. Zhu, C. Song, X. Liu, J. Dong, H. Li, Attacking network isolation in software-defined networks: New attacks and countermeasures, in: 2018 27th International Conference on Computer Communication and Networks (ICCCN), 2018, pp. 1–9, http://dx.doi.org/10.1109/ICCCN.2018.8487340.

[10] F. Reynaud, F. Aguessy, O. Bettan, M. Bouet, V. Conan, Attacks against network functions virtualization and software-defined networking: State-of-the-art, in: 2016 IEEE NetSoft Conference and Workshops (NetSoft), 2016, pp. 471–476, http://dx.doi.org/10.1109/NETSOFT.2016.7502487.

[11] D. Spiekermann, J. Keller, T. Eggendorfer, Towards covert channels in cloud environments: a study of implementations in virtual networks, in: International Workshop on Digital Watermarking, Springer, 2017, pp. 248–262.

[12] V. Chandola, A. Banerjee, V. Kumar, Anomaly detection: A survey, ACM Comput. Surv. 41 (3) (2009) 15.

[13] M.H. Bhuyan, D.K. Bhattacharyya, J.K. Kalita, Network anomaly detection: methods, systems and tools, IEEE Commun. Surv. Tutor. 16 (1) (2013) 303–336.

[14] S.B. Kotsiantis, Supervised machine learning: A review of classification techniques, in: Emerging Artificial Intelligence Applications in Computer Engineering, IOS Press, 2007, pp. 3–24.

[15] T.S. Guzella, W.M. Caminhas, A review of machine learning approaches to spam filtering, Expert Syst. Appl. 36 (7) (2009) 10206–10222.

[16] A.A. Diro, N. Chilamkurti, Distributed attack detection scheme using deep learning approach for Internet of Things, Future Gener. Comput. Syst. 82 (2018) 761–768.

[17] T. Sohn, J. Seo, J. Moon, A study on the covert channel detection of TCP/IP header using support vector machine, in: International Conference on Information and Communications Security, Springer, 2003, pp. 313–324.

[18] W. Mazurczyk, L. Caviglione, Information hiding as a challenge for malware detection, IEEE Secur. Priv. 13 (2) (2015) 89–93.

[19] M. Rehák, E. Staab, V. Fusenig, M. Pěchouček, M. Grill, J. Stiborek, K. Bartoš, T. Engel, Runtime monitoring and dynamic reconfiguration for intrusion detection systems, in: International Workshop on Recent Advances in Intrusion Detection, Springer, 2009, pp. 61–80.

[20] E.S. Pilli, R.C. Joshi, R. Niyogi, Network forensic frameworks: Survey and research challenges, Digit. Investig. 7 (1–2) (2010) 14–27.

[21] S. Garfinkel, Network forensics: Tapping the internet, IEEE Internet Comput. 6 (2002) 60–66.

[22] D. Spiekermann, T. Eggendorfer, Challenges of network forensic investigation in virtual networks, J. Cyber Secur. Mobil. 5 (2) (2016) 15–46.

[23] A.H.R. Al Awadi, Dual-layer SDN model for deploying and securing network forensic in distributed data center, Curr. J. Appl. Sci. Technol. (2017) 1–11.

[24] M.T. Nashnosh, A.E. Abuhamra, M.M. Alkabiir, T.A. Shaladi, M.M. Abdunnabi, H.M. Hamedan, Design and implementation of a forensic logger for software defined networks, in: International Conference on Technical Sciences (ICST2019), Vol. 6, 2019, p. 04.

[25] D. Spiekermann, J. Keller, T. Eggendorfer, Improving lawful interception in virtual datacenters, in: Central European Cybersecurity Conference 2018, ACM, 2018, p. 8.

[26] M.H. Bhuyan, D.K. Bhattacharyya, J.K. Kalita, Network Traffic Anomaly Detection and Prevention: Concepts, Techniques, and Tools, Springer, 2017.

[27] L.D. Merkle, Automated network forensics, in: 10th Annual Conference Companion on Genetic and Evolutionary Computation, ACM, 2008, pp. 1929–1932.

[28] J. Zhang, H. Li, Q. Gao, H. Wang, Y. Luo, Detecting anomalies from big network traffic data using an adaptive detection approach, Inform. Sci. 318 (2015) 91–110.

[29] M. Thottan, C. Ji, Anomaly detection in IP networks, IEEE Trans. Signal Process. 51 (8) (2003) 2191–2204.

[30] M.A. Lopez, D.M. Ferrazani Mattos, O.C.M.B. Duarte, An elastic intrusion detection system for software networks, Ann. Telecommun. 71 (11) (2016) 595–605, http://dx.doi.org/10.1007/s12243-016-0506-y.

[31] D. Spiekermann, J. Keller, Impact of virtual networks on anomaly detection with machine learning, in: 2020 6th IEEE Conference on Network Softwarization (NetSoft), 2020, pp. 430–436, http://dx.doi.org/10.1109/NetSoft48620.2020.9165325.

[32] C. Wagner, J. François, T. Engel, et al., Machine learning approach for ip-flow record anomaly detection, in: International Conference on Research in Networking, Springer, 2011, pp. 28–39.

[33] I. Corona, G. Giacinto, F. Roli, Adversarial attacks against intrusion detection systems: Taxonomy, solutions and open issues, Inform. Sci. 239 (2013) 201–225.

[34] N. Pitropakis, E. Panaousis, T. Giannetsos, E. Anastasiadis, G. Loukas, A taxonomy and survey of attacks against machine learning, Comp. Sci. Rev. 34 (2019) 100199.

[35] A. Shalaginov, S. Banin, A. Dehghantanha, K. Franke, Machine learning aided static malware analysis: A survey and tutorial, in: Cyber Threat Intelligence, Springer, 2018, pp. 7–45.

[36] M. Khan, I. Wakeman, Machine learning for post-event timeline reconstruction, in: First Conference on Advances in Computer Security and Forensics, Liverpool, UK, 2006.

[37] T. Shon, Y. Kim, C. Lee, J. Moon, A machine learning framework for network anomaly detection using SVM and GA, in: 6th Annual IEEE SMC Information Assurance Workshop, IEEE, 2005, pp. 176–183.

[38] E. Eskin, A. Arnold, M. Prerau, L. Portnoy, S. Stolfo, A geometric framework for unsupervised anomaly detection, in: Applications of Data Mining in Computer Security, Springer, 2002, pp. 77–101.

[39] D. Spiekermann, J. Keller, T. Eggendorfer, Network forensic investigation in openflow networks with forcon, Digit. Investig. 20 (2017) S66–S74.

[40] M. Wang, Y. Cui, X. Wang, S. Xiao, J. Jiang, Machine learning for networking: Workflow, advances and opportunities, IEEE Netw. 32 (2) (2018) 92–99, http://dx.doi.org/10.1109/MNET.2017.1700200.

[41] D. Spiekermann, T. Eggendorfer, Towards digital investigation in virtual networks: A study of challenges and open problems, in: 11th International Conference on Availability, Reliability and Security (ARES), 2016, pp. 406–413, http://dx.doi.org/10.1109/ARES.2016.34.

[42] Y. Rekhter, R.G. Moskowitz, D. Karrenberg, G.J. de Groot, E. Lear, Address allocation for private internets, (5) 1996, https://tools.ietf.org/html/rfc1918.

[43] R. Cziva, S. Jouët, D. Stapleton, F.P. Tso, D.P. Pezaros, SDN-Based virtual machine management for cloud data centers, IEEE Trans. Netw. Serv. Manag. 13 (2) (2016) 212–225.

[44] H. Farhady, H. Lee, A. Nakao, Software-defined networking: A survey, Comput. Netw. 81 (2015) 79–95.

[45] S. Sharma, D. Staessens, D. Colle, D. Palma, J. Goncalves, R. Figueiredo, D. Morris, M. Pickavet, P. Demeester, Implementing quality of service for the software defined networking enabled future internet, in: 2014 Third European Workshop on Software Defined Networks, IEEE, 2014, pp. 49–54.

[46] OpenStack, Overlay (tunnel) protocols, 2020, Online, https://docs.openstack.org/neutron/rocky/admin/intro-overlay-protocols.html.

[47] B.L. Dalmazo, J. Vilela, P. Simões, M. Curado, Expedite feature extraction for enhanced cloud anomaly detection, in: NOMS 2016 - 2016 IEEE/IFIP Network Operations and Management Symposium, 2016, pp. 1215–1220.

[48] H. Qiao, J.O. Blech, H. Chen, A machine learning based intrusion detection approach for industrial networks, in: 2020 IEEE International Conference on Industrial Technology (ICIT), 2020, pp. 265–270, http://dx.doi.org/10.1109/ICIT45562.2020.9067253.

[49] M. Wallschläger, A. Gulenko, F. Schmidt, O. Kao, F. Liu, Automated anomaly detection in virtualized services using deep packet inspection, Procedia Comput. Sci. 110 (2017) 510–515.

[50] P. Winter, E. Hermann, M. Zeilinger, Inductive intrusion detection in flow-based network data using one-class support vector machines, in: 4th IFIP International Conference on New Technologies, Mobility and Security, IEEE, 2011, pp. 1–5.

[51] Z. Jadidi, V. Muthukkumarasamy, E. Sithirasenan, M. Sheikhan, Flow-based anomaly detection using neural network optimized with gsa algorithm, in: IEEE 33rd International Conference on Distributed Computing Systems Workshops, IEEE, 2013, pp. 76–81.

[52] S. Song, L. Ling, C. Manikopoulo, Flow-based statistical aggregation schemes for network anomaly detection, in: IEEE International Conference on Networking, Sensing and Control, IEEE, 2006, pp. 786–791.

[53] F. Palmieri, U. Fiore, Network anomaly detection through nonlinear analysis, Comput. Secur. 29 (7) (2010) 737–755.

[54] M. Suresh, R. Anitha, Evaluating machine learning algorithms for detecting ddos attacks, in: International Conference on Network Security and Applications, Springer, 2011, pp. 441–452.

[55] S. Zhao, M. Chandrashekar, Y. Lee, D. Medhi, Real-time network anomaly detection system using machine learning, in: 11th International Conference on the Design of Reliable Communication Networks (DRCN), IEEE, 2015, pp. 267–270.

[56] M.A. Elsadig, Y.A. Fadlalla, Packet length covert channel: A detection scheme, in: 1st International Conference on Computer Applications Information Security (ICCAIS), 2018, pp. 1–7, http://dx.doi.org/10.1109/CAIS.2018.8442026.

[57] T. Lane, C.E. Brodley, An application of machine learning to anomaly detection, in: 20th National Information Systems Security Conference, Vol. 377, Baltimore, USA, 1997, pp. 366–380.

[58] C.C. Aggarwal, Data Classification: Algorithms and Applications, CRC Press, 2014.
[59] M.M. Breunig, H.-P. Kriegel, R.T. Ng, J. Sander, LOF: identifying density-based local outliers, in: ACM SIGMOD International Conference on Management of Data, 2000, pp. 93–104.
[60] F.T. Liu, K.M. Ting, Z.-H. Zhou, Isolation forest, in: 8th IEEE International Conference on Data Mining, IEEE, 2008, pp. 413–422.
[61] scikit-learn, Sklearn.neighbors.localoutlierfactor, 2019, Online, https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.LocalOutlierFactor.html#sklearn.neighbors.LocalOutlierFactor.
[62] D. Liu, Y. Zhao, H. Xu, Y. Sun, D. Pei, J. Luo, X. Jing, M. Feng, Opprentice: Towards practical and automatic anomaly detection through machine learning, in: Proceedings of the 2015 Internet Measurement Conference, 2015, pp. 211–224.
[63] A. Gulenko, M. Wallschläger, F. Schmidt, O. Kao, F. Liu, Evaluating machine learning algorithms for anomaly detection in clouds, in: 2016 IEEE International Conference on Big Data (Big Data), IEEE, 2016, pp. 2716–2721.
[64] N. Moustafa, J. Slay, The evaluation of network anomaly detection systems: Statistical analysis of the UNSW-NB15 data set and the comparison with the KDD99 data set, Inf. Secur. J. 25 (1–3) (2016) 18–31.

Daniel Spiekermann is a professor for digital forensics and cyber crime at Polizeiakademie Niedersachsen (Police Academy of Lower Saxony in Germany). He received his Master of science degree in electrical and computer engineering with a thesis in the field of cloud forensics in 2014. He received his Ph.D. on digital forensics in virtual environments at FernUniversität Hagen in 2017. Before he worked as a forensic investigator at North-Rhine Westphalia Police, Dortmund, Germany. His research interests are network forensics, virtual networks and it security.

Jörg Keller received the Ph.D. degree in computer science from Universität des Saarlandes, Saarbrücken, Germany, in 1992 He is a professor at the Faculty of Mathematics and Computer Science o FernUniversität in Hagen, Germany. His research interests include energy-efficient parallel computing, security and cryptography and fault tolerant computing.