Encapcap: Transforming Network Traces to Virtual Networks

Daniel Spiekermann
Faculty 1 - Criminal Sciences
Polizeiakademie Niedersachsen
Oldenburg, Germany
Daniel.Spiekermann@polizei.niedersachsen.de

Jörg Keller
Faculty of Mathematics and Computer Science
FernUniversität in Hagen
Hagen, Germany
Joerg.Keller@FernUni-Hagen.de

***Abstract*—Valid and complete network captures are a valuable source when detecting network based attacks and adversarial data exfiltration techniques like covert channels or performing network forensic investigation Also in training, testing, benchmarking and algorithm development, the availability of pre-recorded, entire packet captures is eminent. Such a packet capture contains the entire packet stream with all incoming and outgoing network packets recorded over a defined period of time. Whereas a large number of recorded packet captures with well-known protocols from physical networks exists, the number of available files focused on virtual networks is low. Yet, virtual networks are taking on an ever greater role in modern environments. The creation of such network traces is a time-consuming and error-prone task, and the inherent behaviour of virtual networks eradicates a straight-forward automation of trace generation in comparison to common networks. In this paper we analyze relevant conditions of modern networks which hamper the generation of valid test captures and propose *Encapcap*, a tool that transforms given network packets stored in a capture file to virtual network packets. This improves the process of generating real-life packet captures for testing or training in virtual networks. We evaluate *Encapcap* with several experiments to demonstrate its correctness, usefulness and applicability.**

***Index Terms*—virtual networks, packet generation, packet transformation**

I. Introduction

The analysis of packets in modern networks is a complex and time-consuming task. Such analyses aim to identify attacks, anomalies and unwanted behaviour inside the network, help to troubleshoot the environment and are a valuable resource in digital investigations of suspicious events in the environment. A common way to perform this is a process of capturing, recording and analyzing the network packets. The capturing and recording of the data is done in an online-phase which creates a capture file with all or at least with all relevant network packets. This file is later analysed in an offline phase, in which the investigator uses different tools to extract relevant information like IP addresses, conversation details or suspicious behaviour. A network conversation is defined as the traffic between two specific endpoints and summarizes aspects like bytes sent and received, port addresses or the duration of flows [1]. The increasing performance of modern networks hamper this mostly manual analysis, so modern implementations use machine learning algorithms to filter or classify network packets in order to detect anomalies or outliers. The training of these algorithms is based on captured network data of the productive network. As shown in [2], changes inside virtual networks have an impact on the detection rate of the algorithms. Therefore, possible changes in the infrastructure demand for a new training of the model, based on new data. Unfortunately, the creation of a new data-set is a time-consuming and error-prone task. Every mistake when creating the new data-set might confuse the algorithm and distort the results.

Not only in the field of machine learning, but also in different use cases there is a need of high-quality network packet captures, which is a raising challenge in nowadays research and development area. Different use cases like education and training, tool testing, performance measurement or troubleshooting demand capture packets in different formats and protocols. This is needed not only in the field of information security, but also in digital investigation. The creation of these test data is a critical problem not only in network forensics, but in nearly every branch of digital investigation [3]. There are different packet captures as well as data-sets publicly released. Most of these files are limited in scope. Either they contain only a small number of packets, e. g. for a special protocol investigation, or they are flow-based captures. Various network captures with a bigger number of packets are available, but typically they are limited. There are also other examples of digital forensic test data like test images, SQLite databases, machine learning data-sets with images or textual data. Nonetheless, the overall diversity of these corpora is insufficient for many of the aforementioned use cases. [4] analysed 715 digital forensic articles published between 2010 and 2015 and discovered that many of the associated data-sets used in these experiments are not publicly available. Existing packet capture tools do not cover the specialties of virtual networks, such as VM migration or protocol changes inside the environment.

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.

Our main research contribution is *Encapcap*, a tool for:

- Creation of valid packet capture filessuited for virtual networks, based on existing data-sets from physical networks.
- Dynamic encapsulation with different virtualization protocols like NVGRE, VXLAN and GENEVE.
- Simulating effects of inherent virtual network behaviour like VM migration or protocol changes.

Our tool uses Scapy to parse and write original and transformed trace files, resp., but the logic to represent the virtualized network and compute transformations and actions within that network is original contribution of *Encapcap*.

The remainder of this article is structured as follows. In Section II, we provide background information on network virtualization, packet capture generation, and discuss related work. In Section III, we present *Encapcap* to transform packet traces from physical to virtual networks, and how to apply it to different parts. In Section IV we report on our experimental results, evaluate the proof-of-concept and discuss open challenges, while Section V presents conclusions and an outlook to future work.

II. Background

A. Virtual protocols

Modern environments heavily base on dynamic and flexible infrastructures to fulfill the demands of users and services. In the past the environments were mostly static, but due to increased enquiries for an easy and customizable usage of the infrastructure, virtualisation came apart. At the beginning, virtual machines (VM) were used, but these systems require an adaptable and programmable environment to play to the strengths. With the evolution of virtual network protocols a flexible and dynamic connection of network devices got possible. Whereas VLAN as the first implementation of such a protocol allows the use of 4096 different logical subnets[1], modern implementations like Virtual eXtensible LAN (VXLAN), Generic Networking Virtualization Encapsulation (GENEVE) or Network Virtualization using Generic Routing Encapsulation (NVGRE) facilitate more than 16.000.000 different subnets[2]. The separation of the different networks is done with a special field in the header, which identifies an isolated subnet under the administration of a single customer. By this, subnets can be spread all over the physical infrastructure. If a host wants to communicate with a host located in the same subnet, but on a different compute host, the network packet is encapsulated as shown in Figure 1. The new header fields of the encapsulation protocols add information for the transport [5].

The encapsulation process is done by a tunnel endpoint (TE). The TE residing on the other compute node removes the additional headers and forwards the original packet to the intended destination.

[1]Size of tag-field of VLAN is 12 byte, which led to $2^12 = 2096$ values.

[2]The protocols use a 24 bit sized header field, which creates $2^24 = 16.777.215$ subnets.

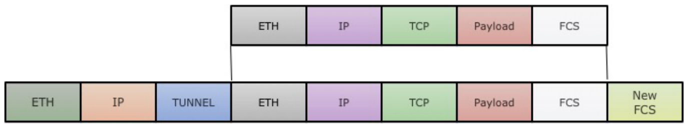

Fig. 1. Encapsulation of network packets.

B. Related Work on Packet generation

A lot of network traffic captures are publicly available, software vendors like Netresec[3] and Wireshark[4] provide packet captures as well as different web pages[5]. The number of files is high, but mostly the number of captured packets is too small for any advanced usage. Different researchers provide the download of huge data-sets, partly with additional information like labels or statistics. Mostly the available network captures are labeled data-sets containing only flows without a full packet capture like *CTU-13* [6] or *UGR'16* [7]. These files have a high value in the field of machine learning, but the use for packet trace transformation is limited. Data-sets with a full packet capture are *CICDDoS2019* [8], *CSE-CIC-IDS 2018* [9] and *INSecS-DCSl* [10], but there is no publicly available data-set with different virtual protocols. A common technique is the replay of the network packets inside a virtual environment with tools like *tcpreplay*, *TCPivo* [11] or *OFRewind* [12]. The tools provide a great flexibility in repeating captured network captures, but focus on the implementation or evaluation of high-performance networks. In contrast to these, a forensic purpose like [13] focus on the replay as accurately as possible. Because of this, none of the tools is able to create virtual network captures. In addition to this, the tools listed in II-B are powerful, and provide the creation of different network captures with VXLAN, but do not provide a flexible generation of network packets with virtual encapsulation based on a given network capture.

Besides capturing live packet traces, network packet traces can be generated synthetically. The generation of network packets is defined as the result of time-stamped series of packets arriving and departing from particular network interfaces with realistic values [14]. This process is performed in various disciplines like IT-security [15], network troubleshooting [16], educational and training [17], [18], testing [19] and network forensic investigation [2], [20]. The generated network captures provide a valid way to repeat different tasks in order to train, simulate, troubleshoot or learn from the results. Different tools allow design and creation of network packets. Common tools are *Cisco TRex*[6], *Ostinato* [21] and *Genesids* [22].

[23] describes six different techniques to generate network packets or traffic collection:

1) Use of real networks

[3]https://www.netresec.com/?page=pcapfiles

[4]https://wiki.wireshark.org/SampleCaptures

[5]Cf. e.g. https://www.unsw.adfa.edu.au/unsw-canberra-cyber/cybersecurity/ADFA-NB15-Datasets/ or https://www.kdd.org/kdd-cup/view/kdd-cup-1999/Data.

[6]Details can be found at https://trex-tgn.cisco.com.

2) Create a honeynet
3) Use of a network simulator
4) Existing traffic dumps
5) Use of network traffic generators
6) Combining the aforementioned techniques.

In the field of digital investigation, the use of valuable test data is a perpetual challenge [24]. [25] describe the use of purchased second hand hardware to collect different user data. Unfortunately, network data is volatile and not stored on hardware after the transmission, thus purchasing old hardware is not a reasonable approach. Unfortunately, these works do not consider virtual networks.

III. Packet Trace Virtualization

A. Challenges

Virtual networks differ from physical networks in various aspects which have to be handled during the transformation. Virtual networks provide a high dynamic to fulfil different demands of modern networks like migration or on-demand changes in the infrastructure, often initiated by user customization. A common process in a virtual environment is the migration of the internal virtual machines (VM). This is done to provide benefit for load-balancing and high availability. Such a migration changes the internal structure of the network by adding or deleting tunnel endpoints, routing tables, IP addresses and existing firewall rules [26]. Another difference to traditional hardware-based networks is the possibility to change used protocols on-the-fly. This protocol swap is done to implement additional features like TCP-offloading [27].

Because of this, the creation of packet captures with virtual protocols and the aforementioned challenges is a time-consuming task which is nowadays done in various testbeds. The process of implementing such an environment is done in various steps. [24] defines a significant planning as the first step, along with discussions of trained personnel to preserve the realistic scenario. The technical implementation bases on the installation and configuration of VMs running on compute nodes and the underlying network structure. On top of the underlay network the different implementations for the overlay networks (e. g. VXLAN, GENEVE, NVGRE) have to be installed and configured. To capture network traffic, suitable positions have to be determined, which have a huge impact on the quality of the network capture [28]. After the implementation of all aspects, different scripts are needed that perform actions which result in the network traffic to be captured. After creating the packet capture file, the configuration of the network has to be changed in order to capture a different protocol. If dynamic aspects like VM migration is covered, a reconfiguration of the entire environment is necessary.

B. Methodology

Based on a given network packet trace, we use *Encapcap*[7] to transform the captured data of a non-virtualized network to a packet capture with virtual network packets.

As discussed in Section II the use of existing captured files is one of the most relevant techniques, but the results are typically static with a simple rewriting of the packets [23]. Because of this, we combine the static transformation of given capture files with dynamic change of values to create more realistic packet capture files, which covers the flexibility and dynamic with constantly recurring changes in the infrastructure of virtual networks. This demands for a deeper packet analysis of the existing capture file. An appropriate transformation has to cover aspects of involved systems like IP addresses, MAC addresses, packet flow and, when modifying further values, the existing standards like RFCs.

Encapcap bases on Python3 in combination with scapy [8] 2.4.4. Scapy is used to parse the original pcap file and to extract relevant fields like IP addresses. After analysing these information, *Encapcap* uses scapy to create a valid network packet capture file which is written to disk. The configuration of *Encapcap* is done with arguments submitted via the command line and parsed at start. *Encapcap* needs at least three parameters to start. *--file* defines the input pcap file with the original data. *--net* defines the network to be used as internal network. The last parameter needed by *Encapcap* is the intented type of encapsulation. Based on this information, *Encapcap* determines relevant packets for encapsulation. With parameter *--geneve*, *Encapcap* is started to create an encapsulation with the *GENEVE* protocol, the parameter *–vxlan* will create a VXLAN-encapsulation header, *–nvgre* will result in a NVGRE-based pcap file. The result of the transformation will be stored in a new file with the prefix of the encapsulation protocol followed by the old name. The example below creates a new capture file named *geneve_anomalies.pcap* from input file *anomalies.pcap* using GENEVE encapsulation:

```
>>> encapcap.py
>>>   --file anomalies.pcap
>>>   --net 192.168.10
>>>   --geneve
```

The first step of *Encapcap* is to analyze the packet capture to retrieve information of the conversations, endpoints and protocols. Based on this information, a set of relevant IP addresses is created. This set contains all systems which have to be handled during the transformation. As a protocol for the connection of VMs running in different layer3 domains, the internal address scope is important. Therefore, *Encapcap* uses a *HOME* parameter when started to detect and encapsulate only packets which are transmitted between devices of this network. The encapsulation process adds additional header information to the packets as shown in Figure 1.

These additional header information depend on various aspects of the conversation of two systems. Based its configuration, *Encapcap* detects parameters like source and destination of the conversation and adds the correct header information. If this process fails, the resulting network packet contains invalid packets, which might impede or hamper the subsequent

[7]This is an acronym of "**Encap**sulating **Pcap**"

[8]Details can be found at https://scapy.net.

analysis. Especially analyses with deep packet inspection, network troubleshooting or performance investigations might be confused when processing such malformed capture files. *Encapcap* pays attention to standards like ISO/IEC 10039 and the addressing scheme of MAC addresses with a special focus on the correct usage of group addresses.

Encapcap provides the use of static or randomized values to create a realistic content, the control of the insertion of these values is done by various parameters given when starting the transformation process. By this, repeated transformation of packet captures can be identical or varying, based on the decisions and intended purpose of the creation. The randomized values cover fields such as IP and MAC addresses or port numbers used.

After analyzing the network packets and implementing the correct encapsulation, *Encapcap* is able to simulate the dynamic changes of a virtual network. When a VM is migrated inside the virtual environment, it is moved to another compute node of the environment, and tunnel endpoints are created to provide a connection between the involved compute nodes hosting the communicating VMs. The parameter *--migrate* commands *Encapcap* to change the MAC and IP addresses of the encapsulating packet, which simulates the result of a migration of a VM. This change starts at a random time after 40% to 60% of the processed packets. The migration of a VM in production environment depends on various circumstances, and is mostly unpredictable. We used the aforementioned values to simulate a migration, which results in a balanced number of network packets before and after the migration. This separation creates a usable packet capture, otherwise the capture file might be unbalanced, which might result in unwanted aftereffects. The following example illustrates this:

```
>>> encapcap.py --file test.pcap
>>>   --net 10.0.0.
>>>   --vxlan
>>>   --migrate

 Will simulate migration
 Will use 36:3A:B5:42:D2:85 with
    IP 192.168.0.18 as additional systems
 Will use E8:CB:6B:DB:AB:42 with
    IP 192.168.0.248 as additional systems
 Will start after 55% of processed packets
```

The result of this simulation is a change of the IP addresses of the encapsulation during the packet flow in the packet capture. The following listing shows the change of the outer header information from subnet *172.16.0.x* with the tunnel endpoint *172.16.0.132* and *172.16.0.183* to new IP addresses in subnet *192.168.0.x*[9]. The internal IP addresses *10.0.0.2* and *10.0.0.12* remain unchanged during the entire transformation process, because these are the IP addresses used in the original packet capture.

```
172.16.0.132,10.0.0.2 172.16.0.183,10.0.0.12
192.168.0.18,10.0.0.2 192.168.0.248,10.0.0.12
```

[9]The simulated tunnel endpoints have the IP addresses 192.168.0.18 and 192.168.0.248.

Encapcap calculates random IP addresses of the tunnel endpoints based on RFC 1918 [29] to create realistic packet captures.

Whereas a traditional network and the deployed hardware remain mostly unchanged for a period of time, virtual networks provide a great flexibility and allow changes of the environment during run-time. This provides a change of the deployed protocols as well as user customization. *Encapcap* is able to handle these aspects and allows the change of the virtual network packet by command. The following listing shows the frame number and the protocols available in the frame. Thus, there is a change between packets with indices 6 (using VXLAN) and 7 (using GENEVE).

```
6 eth:ethertype:ip:udp:vxlan:eth:
7 eth:ethertype:ip:udp:geneve:eth:
```

Whereas some changes are relevant in case of packet transformation, other values should not change in order to reduce possible errors when the traffic is analyzed. Timing parameters like time-stamps, time-to-live or the delta of packets should not change, as well as the internal payload or ports and IP addresses. These are therefore maintained in *Encapcap*.

IV. Experimental Results

The most critical aspect in the generation of network packets is the correctness of the data. Especially in the field of machine learning the use of valid and appropriate test data is crucial for the effective detection of unwanted traffic.

To validate our approach, we use various network packet traces which are publicly available. We submit the relevant information about the internal network structure with the *--net* parameter. We assume that only packets defined by the subnet are encapsulated with the relevant information.

A. Anomaly detection

The detection of anomalies or novelties is an important part of modern network security. Whereas anomaly detection is part of unsupervised learning, the analysis of the novelty is part of supervised learning. As shown in [2], virtual networks have an impact on the detection of anomalies, because the changes in the network might confuse the algorithms. Therefore, the models have to be trained again to learn aspects of virtual networks and their inherent dynamic [30]. If this training is done without a valid data-set, the quality of the analysis might be low. To evaluate *Encapcap* in the field of machine learning with a focus on the creation of data-sets, we analyze a given packet capture with benign and malicious network traffic. We use Decision Trees, Logistic Regression and Naive Bayes as algorithms for classification of the data. The original data-set contains 699.218 network packets, and we use the *sklearn train_test_method()*[10] to split the data into a training and a test data-set with a rate of 70% for training and 30% for testing. As shown in Table I, the trained models achieve a prediction rate of $> 99\%$ based on a given data-set. We first

[10]Details can be found at: https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html.

analyzed the prediction rate based on common network traffic, e. g. without any virtual network protocols in the capture file. We stored the model (named *old model* in Table I) for the subsequent analysis of the network data after the transformation with *Encapcap*. After the transformation, the network capture contains the same payload, but now encapsulated with virtual network information. We performed the same analysis as before, now the prediction rate is significantly lower, which might be critical in productive environments.

TABLE I
PREDICTION RATE IN PERCENTAGE

Algorithm	**Old model**	**Transformed, old model**	**Transformed, new model**
Decision Tree	100	39,83	99,98
Naive Bayes	99,98	27,68	99,97
Logistic Regression	99,99	86,61	99,97

Afterwards we trained the algorithms with the transformed packets[11], which results in a good prediction rate of 99%.

The adaptation of existing models for traditional networks to highly dynamic virtual networks is faced with different issues. Because of this, existing implementations are not usable for the reconfiguration of trained models to achieve a fast and simplified use. Thus, the training of the deployed machine learning algorithm and the evaluation of the data-sets are mostly done in productive networks.

B. Network-based attacks

The detection of malicious network packets is not limited to machine learning algorithms. Web application firewalls and intrusion detection systems (IDS) have to be able to analyse various network protocols to detect unwanted payloads. As discussed in [31], virtual protocols change the position of strings inside the stream which might confuse the analysing systems. *Encapcap* allows the quick analysis of such changes as shown in the following listing.

```
xxd cleartext.pcap |grep Hel
00001c00: 6c6c 6f20 776f 726c &Hello worl
xxd geneve_cleartext.pcap |grep Hel
00002914: 6c6c  ......`....&Hell
```

We send a clear text message *Hello World!* from one host to another, intercept the communication and store the packets in the file *cleartext.pcap*. After transforming this capture file with *Encapcap*, the shift of the starting position of the string is detectable[12], which helps to adjust the configuration of firewalls and IDS.

C. Covert channels

The detection of network-based covert channels is a complex task in modern networks [32]. Available countermeasures as discussed in [33] depend on the detection of the covert channel [34]. The detection of covert channels in virtual networks is discussed in [35] with a special focus on the possibilities of the virtual protocols. *Encapcap* supports the analysis of these covert channels by providing a fast and reliable generation of different captures from physical networks, which can be used to detect possible covert channels.

D. Software testing

[36] noted that *the production of error-free protocols or complex process interactions is essential to reliable communications.* The quality of new software tools depends on the quality of available data for developing and testing. Especially when new protocols arise, tools need to be updated and adjusted to detect, disassemble, analyze and classify the new protocol information. Without knowing internal structures of the protocol, this process is hampered and results in an error-prone task with a high number of trial-and-error approaches.

[37] discuss often underestimated issues of packet generation and criticize a lack of accuracy in the generation of the network packets. *Encapcap* encapsulates given network traffic in a predefined manner, which ensures the level of accuracy. In addition to this, the transformation is under full control of the user, which provides a high level of flexibility considering the original structure of the packets.

E. Education and training

A critical part in the fields of information security and digital investigation is the preparation of involved personnel [38]. The training and education bases on the availability of usable training material, test data and use cases. In combination with the aforementioned aspects, the education and training benefit from the creation of various virtual protocol captures without the need of complex test scenarios as discussed in Section III-A. The simplified generation of virtual network packets improves the diversity of training data of packet captures is needed in training of beginners as well as experts.

V. CONCLUSIONS

This paper proposes *Encapcap*, a tool for creating network packet traces with virtual packet encapsulation based on existing capture files. Different use cases, which demand for a flexible and on-demand creation of predefined, but partly randomized network traffic captures, exist in various fields like information security, digital investigation and network troubleshooting.

The creation of valid data for testing, learning or developing tools in all of the aforementioned branches is a complex, error-prone and time-consuming task. The generation of network data might be an easy process, but the creation of virtual network data is hampered by different challenges which derive from the high flexibility of these networks. Especially the customizability and migration processes are typical in modern, highly virtualized networks. *Encapcap* is able to manage these challenges and simulate an appropriate behaviour. The predictability of the results increases because of the knowledge

[11]The labels of the packets remain unchanged.

[12]The position in the intercepted traffic is at offset *0x1c00*, while the position in the virtual transformation is at offset *0x2914*.

of the origin network traffic, which is transformed to a virtual capture. Our future research is focused on the expansion of our tool, which helps researchers to add more encapsulation like Point-to-Point-protocol (PPP) or Encapsulated Remote SPAN (ERSPAN).

References

[1] J. H. Baxter, *Wireshark essentials*. Packt Publishing Ltd, 2014.

[2] D. Spiekermann and J. Keller, "Impact of virtual networks on anomaly detection with machine learning," in *6th IEEE Conference on Network Softwarization (NetSoft)*, 2020, pp. 430–436.

[3] S. Garfinkel, "Forensic corpora: a challenge for forensic research," *Electronic Evidence Inf. Cent. 1e10*, 2007.

[4] C. Grajeda, F. Breitinger, and I. Baggili, "Availability of datasets for digital forensics–and what is missing," *Digital Investigation*, vol. 22, pp. S94–S105, 2017.

[5] M. Mahalingam, D. G. Dutt, K. Duda, P. Agarwal, L. Kreeger, T. Sridhar, M. Bursell, and C. Wright, "Virtual extensible local area network (vxlan): A framework for overlaying virtualized layer 2 networks over layer 3 networks." *RFC*, vol. 7348, pp. 1–22, 2014.

[6] S. Garcia, M. Grill, J. Stiborek, and A. Zunino, "An empirical comparison of botnet detection methods," *computers & security*, vol. 45, pp. 100–123, 2014.

[7] G. Maciá-Fernández, J. Camacho, R. Magán-Carrión, P. García-Teodoro, and R. Therón, "Ugr '16: A new dataset for the evaluation of cyclostationarity-based network idss," *Computers & Security*, vol. 73, pp. 411–424, 2018.

[8] I. Sharafaldin, A. H. Lashkari, S. Hakak, and A. A. Ghorbani, "Developing realistic distributed denial of service (ddos) attack dataset and taxonomy," in *International Carnahan Conference on Security Technology (ICCST)*. IEEE, 2019, pp. 1–8.

[9] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, "Toward generating a new intrusion detection dataset and intrusion traffic characterization." in *ICISSp*, 2018, pp. 108–116.

[10] N. Rajasinghe, J. Samarabandu, and X. Wang, "Insecs-dcs: a highly customizable network intrusion dataset creation framework," in *2018 IEEE Canadian Conference on Electrical & Computer Engineering (CCECE)*. IEEE, 2018, pp. 1–4.

[11] W.-c. Feng, A. Goel, A. Bezzaz, W.-c. Feng, and J. Walpole, "Tcpivo: A high-performance packet replay engine," in *Proceedings of the ACM SIGCOMM Workshop on Models, Methods and Tools for Reproducible Network Research*, ser. MoMeTools '03. New York, NY, USA: Association for Computing Machinery, 2003, p. 57–64. [Online]. Available: https://doi.org/10.1145/944773.944783

[12] A. Wundsam, D. Levin, S. Seetharaman, and A. Feldmann, "Ofrewind: Enabling record and replay troubleshooting for networks," in *USENIX Annual Technical Conference*. USENIX Association, 2011, pp. 327–340.

[13] J. Parry, D. Hunter, K. Radke, and C. Fidge, "A network forensics tool for precise data packet capture and replay in cyber-physical systems," in *Proceedings of the Australasian Computer Science Week Multiconference*, 2016, pp. 1–10.

[14] K. V. Vishwanath and A. Vahdat, "Realistic and responsive network traffic generation," in *Proceedings of the 2006 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, ser. SIGCOMM '06. New York, NY, USA: Association for Computing Machinery, 2006, p. 111–122. [Online]. Available: https://doi.org/10.1145/1159913.1159928

[15] S. Ghazanfar, F. Hussain, A. U. Rehman, U. U. Fayyaz, F. Shahzad, and G. A. Shah, "Iot-flock: An open-source framework for iot traffic generation," in *2020 International Conference on Emerging Trends in Smart Technologies (ICETST)*. IEEE, 2020, pp. 1–6.

[16] Y. Li, R. Miao, M. Alizadeh, and M. Yu, "DETER: Deterministic TCP replay for performance diagnosis," in *16th USENIX Symposium on Networked Systems Design and Implementation (NSDI 19)*, 2019, pp. 437–452.

[17] V. Padman and N. Memon, "Design of a virtual laboratory for information assurance education and research," in *Workshop on Information Assurance and Security*, vol. 1, 2002, p. 1555.

[18] J. Son, C. Irrechukwu, and P. Fitzgibbons, "Virtual lab for online cyber security education," *Communications of the IIMA*, vol. 12, no. 4, p. 5, 2012.

[19] A. G. Voyiatzis, K. Katsigiannis, and S. Koubias, "A modbus/tcp fuzzer for testing internetworked industrial systems," in *20th conference on emerging technologies & factory automation (ETFA)*. IEEE, 2015, pp. 1–6.

[20] V. Corey, C. Peterman, S. Shearin, M. S. Greenberg, and J. Van Bokkelen, "Network forensics analysis," *IEEE Internet Computing*, vol. 6, no. 6, pp. 60–66, 2002.

[21] B. R. Patil, M. Moharir, P. K. Mohanty, G. Shobha, and S. Sajeev, "Ostinato-a powerful traffic generator," in *2nd International Conference on Computational Systems and Information Technology for Sustainable Solution (CSITSS)*. IEEE, 2017, pp. 1–5.

[22] F. Erlacher and F. Dressler, "Testing ids using genesids: Realistic mixed traffic generation for ids evaluation," in *Proceedings of the ACM SIGCOMM 2018 Conference*, ser. SIGCOMM '18. New York, NY, USA: Association for Computing Machinery, 2018, p. 153–155. [Online]. Available: https://doi.org/10.1145/3234200.3234204

[23] I. Kotenko, A. Chechulin, and A. Branitskiy, "Generation of source data for experiments with network attack detection software," in *Journal of Physics: Conference Series*, vol. 820, no. 1. IOP Publishing, 2017, p. 012033.

[24] X. Du, C. Hargreaves, J. Sheppard, and M. Scanlon, "Tracegen: User activity emulation for digital forensic test image generation," *Forensic Science International: Digital Investigation (09 2020). Proceedings of DFRWS APAC*, 2020.

[25] C. Moch and F. C. Freiling, "Evaluating the forensic image generator generator," in *International Conference on Digital Forensics and Cyber Crime*. Springer, 2011, pp. 238–252.

[26] D. Spiekermann and T. Eggendorfer, "Challenges of network forensic investigation in virtual networks," *Journal of Cyber Security and Mobility*, vol. 5, no. 2, pp. 15–46, 2016.

[27] R. Kawashima and H. Matsuo, "Implementation and performance analysis of stt tunneling using vnic offloading framework (cvsw)," in *2014 IEEE 6th International Conference on Cloud Computing Technology and Science*, 2014, pp. 929–934.

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

[29] Y. Rekhter, R. G. Moskowitz, D. Karrenberg, G. J. de Groot, and E. Lear, "Address allocation for private internets," Internet Requests for Comments, BCP 5, February 1996. [Online]. Available: https://tools.ietf.org/html/rfc1918

[30] D. Spiekermann and J. Keller, "Unsupervised packet-based anomaly detection in virtual networks," *Computer Networks*, vol. 192, p. 108017, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1389128621001286

[31] D. Spiekermann and 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)*, Aug 2016, pp. 406–413.

[32] W. Mazurczyk and S. Wendzel, "Information hiding: challenges for forensic experts," *Communications of the ACM*, vol. 61, no. 1, pp. 86–94, 2017.

[33] J. Kaur, S. Wendzel, and M. Meier, "Countermeasures for covert channel-internal control protocols," in *2015 10th International Conference on Availability, Reliability and Security*. IEEE, 2015, pp. 422–428.

[34] M. A. Elsadig and Y. A. Fadlalla, "Network protocol covert channels: Countermeasures techniques," in *2017 9th IEEE-GCC Conference and Exhibition (GCCCE)*, 2017, pp. 1–9.

[35] D. Spiekermann, J. Keller, and 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.

[36] P. Zafiropulo, C. West, H. Rudin, D. Cowan, and D. Brand, "Towards analyzing and synthesizing protocols," *IEEE Transactions on Communications*, vol. 28, no. 4, pp. 651–661, 1980.

[37] A. Botta, A. Dainotti, and A. Pescapé, "Do you trust your software-based traffic generator?" *IEEE Communications Magazine*, vol. 48, no. 9, pp. 158–165, 2010.

[38] B. D. Carrier and E. Spafford, "Getting physical with the digital investigation process," *Int. J. Digit. Evid.*, vol. 2, 2003.