Journal of
Cybersecurity and Privacy

Article

Requirements for Crafting Virtual Network Packet Captures

Daniel Spiekermann [1] and Jörg Keller [2,*]

[1] Polizeiakademie Niedersachsen, 31582 Nienburg, Germany; daniel.spiekermann@polizei.niedersachsen.de
[2] Faculty of Mathematics and Computer Science, FernUniversität in Hagen, 58084 Hagen, Germany
* Correspondence: joerg.keller@fernuni-hagen.de

Abstract: Currently, network environments are complex infrastructures with different levels of security, isolation and permissions. The management of these networks is a complex task, faced with different issues such as adversarial attacks, user demands, virtualisation layers, secure access and performance optimisation. In addition to this, forensic readiness is a demanded target. To cover all these aspects, network packet captures are used to train new staff, evaluate new security features and improve existing implementations. Because of this, realistic network packet captures are needed that cover all appearing aspects of the network environment. Packet generators are used to create network traffic, simulating real network environments. There are different network packet generators available, but there is no valid rule set defining the requirements targeting packet generators. The manual 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. Hence, we analyse relevant conditions of modern virtualised networks and define relevant requirements for a valid packet generation and transformation process. From this, we derive recommendations for the implementation of packet generators that provide valid and correct packet captures for use with virtual networks.

Keywords: virtual networks; packet generation; packet transformation; network forensic investigation

check for updates

Citation: Spiekermann, D.; Keller, J. Requirements for Crafting Virtual Network Packet Captures. *J. Cybersecur. Priv.* **2022**, *2*, 516–526. https://doi.org/10.3390/jcp2030026

Academic Editors: Mario Antunes and Carlos Rabadão

Received: 9 May 2022
Accepted: 4 July 2022
Published: 6 July 2022

1. Introduction

Virtual machines, virtual storage and virtual networks are the fundamentals of modern infrastructures, providing highly flexible applications, services and environments such as Software-as-a-Service (SaaS), Platform-as-a-Service (PaaS) and Infrastructure-as-a-Service (IaaS). Cloud environments as well as modern company networks use virtualisation to provide dynamic, secure and highly customisable environments to internal and external customers.

These environments have to cover various, partly conflicting challenges. Inter-networking and connections to internal and external networks are as important as the isolation and secure separation of different networks categories such as customers, management and backup. The implementation of network virtualisation overlay (NVO) is a common technique to create flexible overlay networks on top of common and hardware-based underlay networks. The software-based overlay networks use different virtual network protocols such as VXLAN [1], GENEVE or NVGRE [2] to create a flexible network structure for the different tenants. These protocols facilitate the separation of different networks on top of a single and independent underlay network by encapsulating a given network packet with new protocol information as shown in Figure 1.

The use of encapsulating protocols provides various benefits to the providers. The encapsulation provides flexibility to static networks by adding new network traffic information to a given network packet. The idea of encapsulating network protocols has been known for a long time and is not limited to Ethernet or IP [3]. In addition to this, encapsulating protocols are not limited to local area networks (LAN), even in wide area networks (WAN) protocols such as Point-to-Point Protocol (PPP) [4] or ATM (Asynchronous Transfer Mode) use encapsulation.

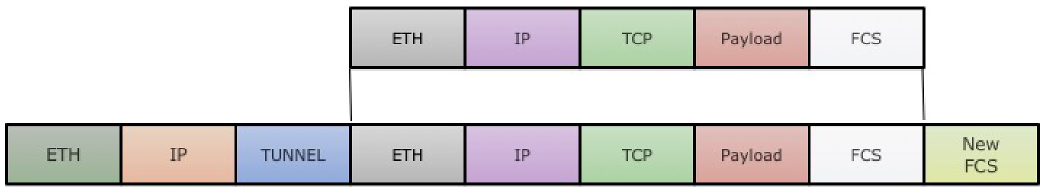


Figure 1. Network packet encapsulation.

In modern networks, protocols such as VXLAN use a special field in the header that identifies an isolated subnet under the administration of a single customer or for a special purpose. In this way, subnets belonging to the same entity 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.

The administration of these networks differs from the administration of pure hardware-based environments. Now, additional attack vectors such as denial-of-service attacks against network controllers [5], insider attacks against virtualised devices [6] or software crashes and buffer overflows of the software implementations are gaining increased importance.

To manage the different challenges, modern techniques such as machine learning-based intrusion detection or network packet classification with deep learning are implemented, but tasks such as prediction occur as well. To train the involved algorithms, a huge amount of network traffic is needed. A common approach is the use of available datasets, but this information differs sometimes from the given network environment. To obtain proper network traffic, [7] describes six different techniques to generate network packets or traffic collection:

1. Use of real networks,
2. Creation of a honeynet,
3. Use of a network simulator,
4. Existing traffic dumps,
5. Use of network traffic generators,
6. Combining the aforementioned techniques.

The use of real network traffic seems to be a common possibility to get relevant network traffic, but this traffic lacks in attack-based network traffic or malicious packets. The creation of a honeynet as well as the network simulator is easier in virtual environments as in hardware-based environments but still a complex task. The process of implementing such an environment is done in various steps. In [8], the authors define significant planning as the first step, along with discussions of trained personnel to preserve the realistic scenario. The technical implementation is based on the installation and configuration of VMs running on compute nodes as well as the wanted network structure. If dynamic aspects such as VM migration have to be covered, a reconfiguration of the entire environment is necessary.

While a huge number of datasets of common networks such as CICDDoS2019 [9], CSE-CIC-IDS 2018 [10] and INSecS-DCSl [11] exist, to our knowledge, there is no publicly available data set with different virtual protocols such as VXLAN or GENEVE.

The use of network traffic generators provides various benefits to the provider. By defining coherent parameters, a packet generator is able to create proper network traffic which fits the involved network. In [12], the authors define packet generation as

> *the result of time-stamped series of packets arriving and departing from particular network interfaces with realistic values.*

We define network packet generation as the synthetic creation of network packets regarding different parameters. These parameters depend on the purpose of the creation process and cover various fields such as the testing of new implementations [13], troubleshooting [14], education and training [15,16] or advanced aspects such as network forensic investigation [17,18].

However, ad hoc traffic generation is cumbersome and error-prone and normally does not take virtualised networks into account. Packet generators are used in various fields of network management, security and performance optimisation. In [19], the authors describe the use of different network packet generators to implement training platforms for teaching defence techniques against denial of service attacks. With *Encapcap* [20], we have presented a first attempt to transform network traces so that they contain virtualised network traffic for specific scenarios. In the present research, we investigate systematically the requirements to be met by packet generators for virtualised networks and thus to see which direction to evolve *Encapcap* in for use with more general scenarios. We also present some use cases that illustrate the necessity of these requirements and demonstrate the use of *Encapcap* in application scenarios. While those use cases comprise the use of test sets for virtual networks generated by *Encapcap*, our goal is not to present a particular test set but the possibility to generate it.

Our main research contributions are as follows:

- An analysis of relevant features in traffic from virtualised networks and subsequently the derivation of a set of requirements, which is as complete as possible, as a baseline for a packet generation process targeted towards virtual environments;
- The extension of the *Encapcap* tool to meet the above requirements;
- The presentation and evaluation of different use cases with the help of (extended) *Encapcap*.

The remainder of this article is structured as follows. Section 2 discusses related work in the fields of virtual networks and packet capture generation. In Section 3, we define pertinent requirements for a valid packet generation process targeted towards virtualised networks. Section 4 presents and analyses use cases, while Section 5 presents conclusions and an outlook to future work.

2. Related Work

The creation of network packets is discussed in various research works. In [21], the authors propose Moongen as a high-speed packet generator able to saturate a 10GBit/s connection. Pktgen [22] is another software-based packet generator that aims to cover high-speed communication. In addition to this, different hardware-based generators such as [23] exist.

Encapsulating protocols are able to tunnel given protocol information to transfer this information without any further interaction from a given point in a network to its intended destination. This is done for security reasons such as IPSec and Encapsulating Security Payload (ESP) [24], virtual private networks [25] or virtual environments [26]. Virtual environments such as software-defined networks (SDN), Docker [27] or Kubernetes [28,29] are heavily based on encapsulating protocols [30]. Network packet captures freeze the transferred network traffic and, depending on the captured traffic, provide additional opportunities for training, learning and testing.

The generation of own packet captures is done for various reasons such as testing of new network protocols, fuzzing, IoT design [31] or security implementations [32]. Because of this, [33] differentiates application-level, flow-level and packet-level traffic generators.

A common technique is the replay of the network packets inside a virtual environment with tools such as tcpreplay, TCPivo [34] or OFRewind [35]. The tools provide great flexibility in repeating captured network traces but focus on the implementation or evaluation of high-performance networks. In contrast to these, tools with a forensic purpose such as [36] focus on making the replay as accurately as possible. When crafting network packets with encapsulation protocols as used in SDN, only *Encapcap* [20] is available. The definition of requirements typically depends on the intended purpose of the generator; e.g., in [37], the authors describe bandwidth, accuracy and precision as relevant requirements for network packet generators, but this research focuses on the live injection of the crafted network packets into a network or network analysis tool. In [38], the authors define five requirements supporting the reproducible traffic generation. In [39], the authors define seven requirements in virtual networks, but they focus on benchmarking and measurement.

3. Requirements

The synthetic creation of valid packet captures, typically stored as pcap files, (https://en.wikipedia.org/wiki/Pcap, accessed on 27 June 2022) is complex and faced with challenges. Errors, misconfigurations or missing considerations of existing standards might lead to incorrect and futile capture files. To facilitate valid and correct capture files, we present the following requirements, which help to overcome these challenges. Thus, we also give recommendations for generators with respect to these requirements.

To ensure a systematic process and achieve a list of requirements that is as complete as possible, we took software testing as an example and role model [40]. In addition, we used our multi-year experience in this field. Yet, we are aware that—as always—such a list of requirements remains a best effort approach. Software testing has to ensure the *correctness* of the different algorithms implemented in the software, e.g., in the form of unit tests. IN addition, the different parts must work together and must be *compatible* with the outside world, e.g., checked by integration tests. Testing needs to cover as many program paths as possible, i.e., to show *flexibility* to address different circumstances, and need to rerun when circumstances change, e.g., in regression testing; i.e., they must be *adaptable*. Tests must be *reproducible*, but, e.g., in black-box testing, *randomisation* is needed. Tests should not only consider functionality but must also address *performance*. Especially for embedded and real-time systems, also predictable and acceptable temporal behaviour like response time must be checked, which we cover under the term *precision*. Our last requirement refers to appropriate use case design: a generator must be *aware* of virtual networks to address their specific features.

3.1. Correctness

The correctness of data is a critical aspect in all digital investigation [41] as well as testing and development. Thus, the correctness of synthetically created network packets is crucial to the overall process. Incorrect or partial network packet captures are invalid and should not be used in the further process of analysis. Missing or distorted network packets have an unpredictable effect on the subsequent steps, which might result in incorrect or misleading results. Common errors are missing values, incomplete packet flows and wrong check-sums resulting in new errors when investigating or analysing corrupt 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. Neglecting the correctness is possible if payload content does not matter and only header information is analysed [42].

3.2. RFC Compatibility

The process of generating network packets is mostly done for a specific purpose. As mentioned in the previous part, the correctness of the packets is a crucial aspect. In addition to this, the compatibility to the most recent Request-for-Comments (RFC) is important. These standards define relevant aspects such as protocol structures, valid and invalid entries, default values and recommendations for processing the information. If these standards or parts of them get ignored, different follow-up errors might arise. These errors might detract from the intended purpose of the packet capture. Hence, it is crucial that the generated network packets consider the valid specifications. This is a time-consuming part of the generation, because the different specifications and their expansions have to be analysed in detail. Some specifications depend on each other, expand older versions and add new functions. As an example, GRE as defined in RFC 2784 [43] does not provide the relevant header fields for NVGRE, but the extension of GRE defined in RFC 2890 [44] adds two optional header fields to the original header with a length of 32 bits each. NVGRE uses 24 bits to store the virtual subnet ID (VSID). The remaining 8 bits are used as a FlowID, which provides per-flow entropy for flows in the same VSID.

3.3. Flexibility

Packet captures of hardware-based networks provide mostly static information, as the number of changes inside this environment is very low. Connected systems such as servers or computers do not change in these environments, which results in a predictable network design. A major benefit of virtual networks and virtual environments is the flexibility and dynamic of the infrastructure. Typically, packet captures of virtual environments contain a huge number of changing network information such as IP addresses, protocols and involved hosts. This is a result of VMs that are started and stopped inside the environment, or suspended systems copied to different locations inside a user network. A packet generation process in this environment should provide the same flexibility as the environment; otherwise, the generation process is too static for a modern environment. As a result, the generator should be able to add additional systems into the capture file or change information of existing systems.

3.4. Adaptability

In all fields of information security or digital investigation, the use of valuable test data is a perpetual challenge [8]. In contrast to traditional hardware-based environments, it is not applicable to use purchased second-hand hardware to collect different user data [45]. Unfortunately, network data are volatile and not stored on hardware after the transmission; thus, purchasing old hardware is not a reasonable approach. However, the need for different scenarios based on various data is crucial. In [46], the authors define the need for realistic training as follows: "In order to prevent or detect cyber-attacks, people soon realised that the best way is practising hands-on training, where trainees work in a testing environment that mimics real-life situations." Without this variety of scenarios, the effect of training is limited and does not guarantee an ongoing improvement of the involved personnel [47]. With different scenarios and various levels of difficulty, based on flexible generated network captures, the overall acceptance of training and testing can be improved. The packet generator has to create adaptable capture files based on given files. The adaptability can be reached by a useful software design like a graphical user interface or a parameter-based command line interface.

3.5. Reproducability

In addition to the correctness of the created packet captures, it is necessary that the results of packet transformation are repeatable and resilient. The reproducability is a critical requirement of digital investigations [48,49] and ensures the equality of two (or more) generated packet captures based on the same original file and that are manipulated in the same manner. The generation process should be automatised and perform the same steps for every run. Use of parameter sets to characterise and store all relevant parameter settings for a particular generation run in one place can be of help here.

3.6. Randomisation

On the other hand, the generator should be able to create randomised values, which change different packet parameters if needed. By using pseudo-random number generators of good quality [50] and by recording the seed value as part of the parameter set, the requirements of reproducability and randomisation can be fulfilled together.

3.7. Performance

The necessary speed of the generation process depends on the intended purpose. If the generation process injects the packets into a real network in real time, the performance of the process should touch 10 GBit/s [22]. A limiting factor of a generation process is the deployed hardware, but the design of the software heavily affects the overall performance. In addition to this, the memory allocation of the creation process should be suitable for the created capture file.

3.8. Precision

A packet generator has to create precise packet captures with defined gaps or intervals between network packets, which results in accurate packet captures [21,51]. In particular, the pcapng-format is able to manage network packets with a timestamp of 64-bit accuracy, so a packet generator should be able to create correct timestamps in a packet.

However, a common issue of different network packet generators is a varying rate of packet transmission. Software-based solutions depend on the CPU load. If a generator such as *tcpreplay* is used multiple times to replay a given capture file, the time between sent network packets might differ from run to run [52]. If there are competitive processes on a single-core system, the replay process is not eligible to use the CPU core in time.

3.9. Awareness

Virtual networks differ from physical networks in various aspects which have to be handled during the transformation. Techniques such as migration or on-demand changes in the infrastructure, often initiated by a user, are common in virtual environments. A packet generator for virtual environments has to be aware of these situations and has to be able to create valid scenarios based on a given packet capture.

Another event that can occur in virtual networks is the change of internal network packet information and addresses such as the IP range of a user network. A customer is able to change the assigned network and might change IP addresses, routes to some destinations, packet forwarding, firewalling or add additional NFV devices such as VPN gateways or load balancers. All these changes have effects on the network and have to be managed by a packet generator, i.e., by changing internal network information of the transferred network packets.

A further difference to traditional hardware-based networks is the possibility to change protocols on the fly. This protocol swap is done to implement additional features scgh as TCP-offloading [53]. A packet generator should be able to manage such a protocol change by providing the ability of swapping from one protocol to another, which results in different NVO information such as protocol types or port numbers. A packet generator has to analyse and change the necessary information on the fly to create valid and usable capture files.

3.10. Extending Encapcap

Given the systematic analysis of requirements above, we analysed how far our own tool *Encapcap* [20] already fulfills these requirements and where it can be improved.

The analysis resulted in the following extensions. The first release of *Encapcap* as discussed in [20] was able to create virtual network packet captures from a given capture file containing arbitrary network packets. A check for correctness and RFC compatibilitiy of the initial trace has been added. When started with a list of various parameters *Encapcap* is able to provide flexible and adaptable but also reproducible packet captures. A special parameter *--rand* creates randomised values, which prevents the correct reproducability of a created packet capture. As a consequence, the use of *--rand* has been extended to include a mandatory seed value in the current release of *Encapcap*. The extension of *Encapcap* provides a tool that covers all aforementioned requirements with the exception of performance. *Encapcap* is implemented in Python3, so the performance of the creation process is limited due to the interpreting speed of the Python language. Implementing a packet generator in a compiling language such as C will heavily improve the performance of *Encapcap* [54] and is thus targeted as future work.

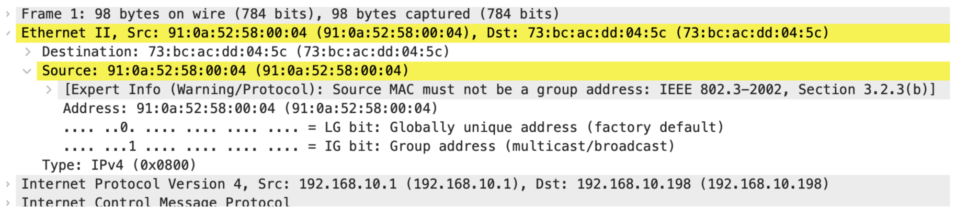
4. Use Cases

In this section, we list three different use cases, which illustrate the proposed requirements. Based on the scenario, the requirements have different relevance. Not every requirement is critical for every use case, but with three use cases, we illustrate that the complete set of requirements is necessary; e.g., the performance as discussed in Section 3.7

is mostly irrelevant, when the need for having adapted network captures is not urgent or the process of creating network packets does not have to be conducted in a short time. On the other hand, a flexible adaptation of the resulting network capture as discussed in Section 3.3 is important for network security. This area has to manage different network traffic, and various attacks use an unusual combination of protocol header fields [55] or network internal aspects such as fragmentation [56]. In addition to this, various network based covert channels use different header fields to exfiltrate the intended information [57,58].

4.1. Training

The implementation of new network protocols might result in new or unknown errors, e. g., in the case of misconfiguration of involved systems or misinterpretation of specifications. If a problem occurs, a common technique is network packet analysis [59]. Such an analysis aims to detect irregular values in packet headers or the incorrect implementation of given standards. Packet captures with pre-defined values related to a given network help to train network engineers to learn new protocols in order to prevent the aforementioned issues. As discussed in Section 3.2, the creation of the requested packet captures needs to consider valid specifications; otherwise, emerging errors might hamper the intended analysis. Figure 2 shows such an intentionally misconfigured capture file that violates the addressing scheme for MAC-addresses defined in [60]. Such a misconfiguration might confuse the user of the tool and distract from other, more relevant issues or the intended purpose. Thus, such misconfiguration should not occur within training sets for network engineers. So, considering specifications and creating correct captures are critical requirements for network packet generations. Otherwise, there would be a need for additional checks after the creation of the transformed dataset before using it in an actual training session. The use of *Encapcap* allowed us to create traces covering 1 hour of packets for training faster.



```
› Frame 1: 98 bytes on wire (784 bits), 98 bytes captured (784 bits)
⌄ Ethernet II, Src: 91:0a:52:58:00:04 (91:0a:52:58:00:04), Dst: 73:bc:ac:dd:04:5c (73:bc:ac:dd:04:5c)
  › Destination: 73:bc:ac:dd:04:5c (73:bc:ac:dd:04:5c)
  ⌄ Source: 91:0a:52:58:00:04 (91:0a:52:58:00:04)
    › [Expert Info (Warning/Protocol): Source MAC must not be a group address: IEEE 802.3-2002, Section 3.2.3(b)]
      Address: 91:0a:52:58:00:04 (91:0a:52:58:00:04)
      .... ..0. .... .... .... .... = LG bit: Globally unique address (factory default)
      .... ...1 .... .... .... .... = IG bit: Group address (multicast/broadcast)
    Type: IPv4 (0x0800)
› Internet Protocol Version 4, Src: 192.168.10.1 (192.168.10.1), Dst: 192.168.10.198 (192.168.10.198)
› Internet Control Message Protocol
```

Figure 2. Example of incompatibility to RFC.

Other requirements such as adaptability, as discussed in Section 3.4, or reproducability, discussed in Section 3.5, help training scenarios to create a clean training environment that provides an identical basis for different training scenarios

4.2. Testing

The network is a fundamental and crucial element in modern environments. The high-speed connection of involved systems as well as the correct transport of the information is one of the most relevant tasks. Bringing new devices into the network has to be tested in order to minimise the risk of unwanted results. Such devices, hardware-based as well as virtual, change the internal packet flow, which results in a different behaviour. So, testing a new device in a real-world scenario is necessary in modern infrastructures, but most of the environments do not provide such an opportunity. Creating network packets with real-world information in real time supports the testing and implementation of new devices. The generated packets, either by a tool such as *Encapcap* or by a different tool that is able to provide virtual network packet captures, can be replayed or sent in real time into a given network device, and the resulting behaviour can be analysed in detail in a simulated network environment to ensure the intended processing. A slow or dissimilar packet flow might not produce the same behaviour as high-speed packet transfers with valid network data, so the precision and performance as well as the awareness are crucial requirements in this use case. We evaluated this process by replaying a generated packet capture into a firewall implementation to analyze the forwarding process of the device. By this, we were able to detect minor misconfigurations in the complex ruleset of the firewall. Without such a testing process, these errors would have persisted into production mode. As a result, this testing opportunity is similar to gray box testing [61].

4.3. Security

Network security and protection is relevant in modern environments. Due to the increasing speed and diversity of network packets, the detection of network-based attacks is a complex task. Preventing adversarial attacks is typically done with techniques such as machine learning, but the deployed algorithms need realistic and correct network packets to train and improve the models. Incorrect, static or incongruous information leads to wrong results and therefore useless models. So, training the models with appropriate information is crucial for a high level of security. As shown in [18], the shift from hardware-based to virtual networks impacts every involved network device that analyses the transferred network traffic. Whereas performance issues or a huge resource load might be detected in an easy way, the detection of misconfigured or incomplete rules for the protection of the network is critical but extremely difficult. Because of this, the process of implementing new network devices or new algorithms for the classification of network packets in virtual networks needs a valid baseline of well-known information. By using a known data-set of the traditional network and transferring this capture to a capture file with adapted information of the intended virtual network, the overall security benefits result in a clean and correct process of packet generation. The re-configuration of the firewall mentioned in Section 4.2 improved the security of a virtual network.

4.4. Relation to Requirements

The requirements defined in Section 3 cover the relevant aspects of the aforementioned realistic use cases. Table 1 summarises the listed use cases and their corresponding requirements. An *x* marks the need of the requirement for the given use case, whereas – defines a lower significance for the use case.

Table 1. Use case analysis.

Requirements	Use Case 1 Training	Use Case 2 Testing	Use Case 3 Security
Correctness	x	x	x
RFC compatibility	x	-	-
Flexibility	x	-	x
Adaptability	x	x	x
Reproducability	x	x	x
Randomisation	-	-	x
Performance	-	x	-
Precision	x	x	-
Awareness	-	x	x

Table 1 illustrates that not every requirement is critical for every use case, but that the complete set of requirements is necessary already for the three given use cases. Because of the variety of the different reasons for packet generation, a different emphasis in other scenarios is possible. On the other hand, Table 1 details that the correctness is the most relevant requirement when creating adapted virtual network packets. If the generation of virtual packet traces produces irregular or incorrect data, every subsequent step such as flow creation, feature selection, in-depth analysis or performance calculation will result in misleading data, errors or unwanted behaviour.

5. Conclusions

Virtual networks are on the rise, and hence packet traces of virtual networks are needed for a variety of use cases ranging from machine learning-based firewall configuration to realistic training sessions for network engineers.

The process of creating such a dataset for virtual networks is a time-consuming and error-prone task, which is faced with different issues. If the creation process is not fully applicable to the intended purpose, subsequent errors might be undetected. In this paper, we therefore derived, in a systematic manner and as completely as possible, a set of requirements for the creation of datasets in modern networks, with an eye on virtualisation techniques.

We have extended our own tool *Encapcap* [20] with the knowledge from the systematic analysis in this research to meet all requirements except performance. We have applied this tool in several use cases and have illustrated that the set of requirements is indeed necessary for successful application.

Future work comprises the re-implementation of *Encapcap* in a compiled programming language to improve performance as the last missing requirement.

Author Contributions: Conceptualization, D.S. and J.K.; methodology, D.S.; software, D.S.; validation, D.S.; formal analysis, D.S. and J.K.; investigation, D.S.; resources, D.S.; data curation, D.S.; writing—original draft preparation, D.S. and J.K.; writing—review and editing, D.S. and J.K.; visualization, D.S.; All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Mahalingam, M.; Dutt, D.; Duda, K.; Agarwal, P.; Kreeger, L.; Sridhar, T.; Bursell, M.; Wright, C. Virtual eXtensible Local Area Network (VXLAN): A Framework for Overlaying Virtualized Layer 2 Networks over Layer 3 Networks. *RFC* 7348, 2014. Available online: https://datatracker.ietf.org/doc/rfc7348/ (accessed on 8 May 2022).
2. Garg, P.; Wang, Y.S. NVGRE: Network Virtualization Using Generic Routing Encapsulation. *RFC* 7637, 2015. Available online: https://datatracker.ietf.org/doc/rfc7637/ (accessed on 8 May 2022).
3. Kantor, B. Internet Protocol Encapsulation of AX.25 Frames. *RFC* 1226, 1991. Available online: https://datatracker.ietf.org/doc/rfc1226/ (accessed on 8 May 2022).
4. Simpson, W.A. The Point-to-Point Protocol (PPP). *RFC* 1661, 1994. Available online: https://datatracker.ietf.org/doc/rfc1661/ (accessed on 8 May 2022).
5. Mousavi, S.M.; St-Hilaire, M. Early detection of DDoS attacks against SDN controllers. In Proceedings of the 2015 International Conference on Computing, Networking and Communications (ICNC), Garden Grove, CA, USA, 16–19 February 2015; pp. 77–81.
6. Aljuhani, A.; Alharbi, T. Virtualized Network Functions security attacks and vulnerabilities. In Proceedings of the 2017 IEEE 7th Annual Computing and Communication Workshop and Conference (CCWC), Las Vegas, NV, USA, 9–11 January 2017; pp. 1–4.
7. Kotenko, I.; Chechulin, A.; Branitskiy, A. Generation of source data for experiments with network attack detection software. In *Proceedings of the Journal of Physics: Conference Series*; IOP Publishing: Bristol, UK, 2017; Volume 820, p. 012033.
8. Du, X.; Hargreaves, C.; Sheppard, J.; Scanlon, M. TraceGen: User Activity Emulation for Digital Forensic Test Image Generation. *Forensic Sci. Int. Digit. Investig.* **2020** , *38*, 301133. [CrossRef]
9. Sharafaldin, I.; Lashkari, A.H.; Hakak, S.; Ghorbani, A.A. Developing realistic distributed denial of service (DDoS) attack dataset and taxonomy. In Proceedings of the International Carnahan Conference on Security Technology (ICCST), Chennai, India, 1–3 October 2019; pp. 1–8.
10. Sharafaldin, I.; Lashkari, A.H.; Ghorbani, A.A. Toward generating a new intrusion detection dataset and intrusion traffic characterization. In Proceedings of the ICISSp, Funchal, Portugal, 22–24 January 2018; pp. 108–116.
11. Rajasinghe, N.; Samarabandu, J.; Wang, X. INSecS-DCS: A highly customizable network intrusion dataset creation framework. In Proceedings of the 2018 IEEE Canadian Conference on Electrical & Computer Engineering (CCECE), Quebec City, QC, USA, 13–16 May 2018; pp. 1–4.

12. Vishwanath, K.V.; Vahdat, A. Realistic and Responsive Network Traffic Generation. In Proceedings of the 2006 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications, Pisa, Italy, 11–15 September 2006; Association for Computing Machinery: New York, NY, USA, 2006; pp. 111–122. [CrossRef]
13. Voyiatzis, A.G.; Katsigiannis, K.; Koubias, S. A Modbus/TCP fuzzer for testing internetworked industrial systems. In Proceedings of the 20th Conference on Emerging Technologies & Factory Automation (ETFA), Luxembourg, 8–11 September 2015; pp. 1–6.
14. Li, Y.; Miao, R.; Alizadeh, M.; Yu, M. DETER: Deterministic TCP Replay for Performance Diagnosis. In Proceedings of the 16th USENIX Symposium on Networked Systems Design and Implementation (NSDI 19), Boston, MA, USA, 26–28 February 2019; pp. 437–452.
15. Padman, V.; Memon, N. Design of a virtual laboratory for information assurance education and research. In Proceedings of the Workshop on Information Assurance and Security, West Point, NY, USA, 1–3 June 2002; Volume 1, p. 1555.
16. Son, J.; Irrechukwu, C.; Fitzgibbons, P. Virtual lab for online cyber security education. *Commun. Iima* **2012**, *12*, 5.
17. Corey, V.; Peterman, C.; Shearin, S.; Greenberg, M.S.; Van Bokkelen, J. Network forensics analysis. *IEEE Int. Comput.* **2002**, *6*, 60–66. [CrossRef]
18. Spiekermann, D.; Keller, J. Impact of Virtual Networks on Anomaly Detection with Machine Learning. In Proceedings of the 6th IEEE Conference on Network Softwarization (NetSoft), Ghent, Belgium, 29 June–3 July 2020; pp. 430–436. [CrossRef]
19. Trabelsi, Z.; Alketbi, L. Using network packet generators and snort rules for teaching denial of service attacks. In Proceedings of the 18th ACM Conference on Innovation and Technology in Computer Science Education, England, UK, 1–3 July 2013; pp. 285–290.
20. Spiekermann, D.; Keller, J. Encapcap: Transforming Network Traces to Virtual Networks. In Proceedings of the 2021 IEEE 7th International Conference on Network Softwarization (NetSoft), Tokyo, Japan, 28 June–2 July 2021; pp. 437–442.
21. Emmerich, P.; Gallenmüller, S.; Raumer, D.; Wohlfart, F.; Carle, G. Moongen: A scriptable high-speed packet generator. In Proceedings of the 2015 Internet Measurement Conference, Tokyo, Japan, 28–30 October 2015; pp. 275–287.
22. Olsson, R. Pktgen the linux packet generator. In Proceedings of the Linux Symposium, Ottawa, Canada, 22–25 July 2005; Volume 2, pp. 11–24.
23. Sanlı, M.; Schmidt, E.G.; Güran, H.C. FPGEN: A fast, scalable and programmable traffic generator for the performance evaluation of high-speed computer networks. *Perform. Eval.* **2011**, *68*, 1276–1290. [CrossRef]
24. Kent, S.; Atkinson, R. IP Encapsulating Security Payload (ESP). *RFC* 2406, 1991. Available online: https://datatracker.ietf.org/doc/html/rfc2406 (accessed on 8 May 2022).
25. Worster, T.; Rekhter, Y.; Rosen, E. Encapsulating MPLS in IP or Generic Routing Encapsulation (GRE). *RFC* 4023, 2005. Available online: https://datatracker.ietf.org/doc/rfc4023/ (accessed on 8 May 2022).
26. Chowdhury, N.M.K.; Boutaba, R. A survey of network virtualization. *Comput. Net.* **2010**, *54*, 862–876. [CrossRef]
27. Hausenblas, M. *Container Networking*; O'Reilly Media, Incorporated: Sebastopol, CA, USA, 2018.
28. Botez, R.; Costa-Requena, J.; Ivanciu, I.A.; Strautiu, V.; Dobrota, V. SDN-Based Network Slicing Mechanism for a Scalable 4G/5G Core Network: A Kubernetes Approach. *Sensors* **2021**, *21*, 3773. [CrossRef] [PubMed]
29. Spiekermann, D.; Keller, J. Wiretapping Pods and Nodes-Lawful Interception in Kubernetes. *Electron. Commun. EASST* **2021**, *80*.
30. Spiekermann, D.; Keller, J.; Eggendorfer, T. Network forensic investigation in OpenFlow networks with ForCon. *Digit. Investig.* **2017**, *20*, S66–S74. [CrossRef]
31. Al-Hadhrami, Y.; Hussain, F.K. Real time dataset generation framework for intrusion detection systems in IoT. *Future Gener. Comput. Syst.* **2020**, *108*, 414–423. [CrossRef]
32. Belenko, V.; Krundyshev, V.; Kalinin, M. Synthetic datasets generation for intrusion detection in VANET. In Proceedings of the 11th International Conference on Security of Information and Networks, Cardiff, UK, 10–12 September 2018; pp. 1–6.
33. Botta, A.; Dainotti, A.; Pescapé, A. Do you trust your software-based traffic generator? *IEEE Commun. Mag.* **2010**, *48*, 158–165. [CrossRef]
34. Feng, W.c.; Goel, A.; Bezzaz, A.; Feng, W.c.; Walpole, J. TCPivo: A High-Performance Packet Replay Engine. In Proceedings of the ACM SIGCOMM Workshop on Models, Methods and Tools for Reproducible Network Research , Karlsruhe, Germany, 25–27 August 2003; Association for Computing Machinery: New York, NY, USA, 2003; pp. 57–64. [CrossRef]
35. Wundsam, A.; Levin, D.; Seetharaman, S.; Feldmann, A. OFRewind: Enabling record and replay troubleshooting for networks. In Proceedings of the USENIX Annual Technical Conference, Portland, OR, USA, 15–17 June 2011; USENIX Association: Berkeley, CA, USA, 2001; pp. 327–340.
36. Parry, J.; Hunter, D.; Radke, K.; Fidge, C. A network forensics tool for precise data packet capture and replay in cyber-physical systems. In Proceedings of the Australasian Computer Science Week Multiconference, Canberra, Australia, 2–5 February 2016 ; pp. 1–10.
37. Emmerich, P.; Gallenmüller, S.; Antichi, G.; Moore, A.W.; Carle, G. Mind the gap-a comparison of software packet generators. In Proceedings of the 2017 ACM/IEEE Symposium on Architectures for Networking and Communications Systems (ANCS), Beijing, China, 18–19 May 2017; pp. 191–203.
38. Gallenmüller, S.; Scholz, D.; Stubbe, H.; Carle, G. The pos framework: A methodology and toolchain for reproducible network experiments. In Proceedings of the 17th International Conference on emerging Networking EXperiments and Technologies s (CoNEXT '21), Munich, Germany, 7–10 December 2021 ; pp. 259–266.

39. *Network Functions Virtualisation (NFV) Release 3*; Testing; Specification of Networking Benchmarks and Measurement Methods for NFVI. Standard, European Telecommunications Standards Institute: Sophia Antipolis, France, 2020.
40. Pezzè, M.; Young, M. *Software Testing and Analysis—Process, Principles and Techniques*; Wiley: Hoboken, NJ, USA, 2007.
41. Vömel, S.; Freiling, F.C. Correctness, atomicity, and integrity: Defining criteria for forensically-sound memory acquisition. *Digit. Investig.* **2012**, *9*, 125–137. [CrossRef]
42. Soltanian, M.R.K.; Amiri, I.S. Chapter 4—Results and Discussions. In *Theoretical and Experimental Methods for Defending Against DDOS Attacks*; Soltanian, M.R.K., Amiri, I.S., Eds.; Syngress: Waltham, MA, USA, 2016; pp. 47–56. [CrossRef]
43. Li, T.; Farinacci, D.; Hanks, S.P.; Meyer, D.; Traina, P.S. Generic Routing Encapsulation (GRE). *RFC* 2784. Available online: https://dl.acm.org/doi/10.17487/RFC2784 (accessed on 8 May 2022).
44. Dommety, G. Key and Sequence Number Extensions to GRE. *RFC* 2890. Available online: https://datatracker.ietf.org/doc/html/rfc2890 (accessed on 8 May 2022).
45. Moch, C.; Freiling, F.C. Evaluating the forensic image generator generator. In Proceedings of the International Conference on Digital Forensics and Cyber Crime, Dublin, Ireland, 26–28 October 2011 ; Springer: Berlin/Heidelberg, Germany, 2011; pp. 238–252.
46. Tang, D.; Pham, C.; Chinen, K.i.; Beuran, R. Interactive cybersecurity defense training inspired by web-based learning theory. In Proceedings of the 2017 IEEE 9th International Conference on Engineering Education (ICEED), Kanazawa, Japan, 9–10 November 2017; pp. 90–95.
47. Torten, R.; Reaiche, C.; Boyle, S. The impact of security awarness on information technology professionals' behavior. *Comput. Secur.* **2018**, *79*, 68–79. [CrossRef]
48. Pan, L.; Batten, L. Reproducibility of digital evidence in forensic investigations. In Proceedings of the 5th Annual Digital Forensic Research Conference (DFRWS 2005), New Orleans, LA, USA, 17–19 August 2005; pp. 1–8.
49. Spiekermann, D. FAP: Design of an Architecture of a Forensic Access Point to Perform Online Access in a Forensically Sound Manner. In Proceedings of the European Interdisciplinary Cybersecurity Conference, Rennes, France, 18 November 2020; pp. 1–6.
50. Kneusel, R.T. *Random Numbers and Computers*; Springer: Berlin/Heidelberg, Germany, 2018.
51. Gallenmüller, S.; Emmerich, P.; Raumer, D.; Carle, G. *Moongen: Software Packet Generation for 10 Gbit and Beyond*; USENIX NSDI: Oakland, CA, USA, 2015.
52. Covington, G.A.; Gibb, G.; Lockwood, J.W.; McKeown, N. A packet generator on the NetFPGA platform. In Proceedings of the 17th IEEE Symposium on Field Programmable Custom Computing Machines, Napa, CA, USA, 5–7 April 2009; IEEE: New York, NY, USA, 2009; pp. 235–238.
53. Kawashima, R.; Matsuo, H. Implementation and Performance Analysis of STT Tunneling Using vNIC Offloading Framework (CVSW). In Proceedings of the 2014 IEEE 6th International Conference on Cloud Computing Technology and Science, Singapore, 15–18 December 2014; pp. 929–934.
54. Prechelt, L. An empirical comparison of c, c++, java, perl, python, rexx and tcl. *IEEE Comput.* **2000**, *33*, 23–29. [CrossRef]
55. John, W.; Tafvelin, S. Analysis of internet backbone traffic and header anomalies observed. In Proceedings of the 7th ACM SIGCOMM conference on Internet measurement, San Diego, CA, USA, 24–26 October 2007; pp. 111–116.
56. Bittau, A. The fragmentation attack in practice. In Proceedings of the IEEE Symposium on Security and Privacy, Oakland, CA, 8–11 May 2005; IEEE Computer Society: Washington, DC, USA, 2005.
57. Mazurczyk, W.; Szary, P.; Wendzel, S.; Caviglione, L. Towards reversible storage network covert channels. In Proceedings of the 14th International Conference on Availability, Reliability and Security, Canterbury, UK, 26–29 August 2019; pp. 1–8.
58. Spiekermann, D.; Keller, J.; Eggendorfer, T. Towards Covert channels in cloud environments: A study of implementations in virtual networks. In Proceedings of the International Workshop on Digital Watermarking, Magdeburg, Germany, 23–25 August 2017; Springer: Berlin/Heidelberg, Germany, 2017; pp. 248–262.
59. Sanders, C. *Practical Packet Analysis, 3E: Using Wireshark to Solve Real-World Network Problems*; No Starch Press: San Francisco, CA, USA, 2017.
60. *IEEE Std-802.3-2005 (Revision IEEE Std-802.3-2002 including all approved amendments)*; IEEE Standard for Information Technology—Telecommunications and Information Exchange Between Systems—Local and Metropolitan Area Networks—Specific Requirements Part 3: Carrier Sense Multiple Access with Collision Detection (CSMA/CD) Access Method and Physical Layer Specifications. IEEE: Piscataway, NJ, USA, 2005. [CrossRef]
61. Ivanova, M.E.; Dushkin, A.V.; Bryushinin, A.O. Method of Fuzzing Testing of Firewalls Using the Gray Box Method. In Proceedings of the 2021 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (ElConRus), St. Petersburg and Moscow, Russia, 26–28 January 2021; p. 2340. [CrossRef]