Digital Investigation 20 (2017) S66–S74

Contents lists available at ScienceDirect

Digital Investigation

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

DFRWS 2017 Europe — Proceedings of the Fourth Annual DFRWS Europe

Network forensic investigation in OpenFlow networks with ForCon

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

[a] FernUniversität in Hagen, Germany
[b] HS Ravensburg-Weingarten, Germany

ARTICLE INFO

Article history:
Received 26 January 2017
Accepted 26 January 2017

Keywords:
Virtual networks
Digital investigation
Network forensic
OpenFlow

ABSTRACT

To resolve the challenges of forensic investigation in virtual networks, we present a new forensic framework called "Virtual Network Forensic Process". Based on this framework we present the design, implementation and workflow of ForCon — a forensic controller to implement network investigation in OpenFlow controlled networks using Open vSwitch. Current trends bear out that virtualization techniques are no longer limited to computers as virtual machines. Thus cloud service providers try to offer greater value to their customers by implementing virtual networks and storage. Virtual environments have the same requirements for forensic investigation, however to fulfil these new tools and workflows to resolve new challenges like virtual machine migration or user customization are needed. ForCon uses dislocated agents in the network to monitor the virtual environment for changes and adapt the installed capture process without the need for any further interaction by an investigator. Thus, the network forensic investigation in virtual networks becomes flexible and valid evidence of the network data is gathered.

Introduction

With the virtualization of computers, providers began to change their processes and infrastructure in their datacenters. The virtualization provides higher flexibility, improves the automation of installation, configuration and provisioning of services and offers new possibilities and cost reduction at the same time. Aspects like security, network management and customer requirements were difficult to satisfy with virtual computers only.

The next step in the evolution of datacenters was the implementation of virtual networks, which resolved the issues in the virtual environment and improved the cumbersome manual administration of network devices.

The virtualization of networks offers a logical implementation of separated networks, each isolated from the others. The isolation of the networks is independent of the underlying physical network, all data is still transmitted over the physical network switches, routers and cable connections. But with the use of new network protocols like Virtual eXtensible LAN (VXLAN) (Mahalingam et al., 2014), Stateless Transport Tunneling (STT) (Davie and Gross, 2014) or Network Virtualization using Generic Routing Encapsulation (NVGRE) (Garg and Wang, 2015) different implementations of tunneling and encapsulation techniques enable the creation of separate logical networks, using the same underlying physical infrastructure. Therefore these logical networks are called overlay network. Anderson et al. (2005) characterizes overlay networks as follows:

> *With virtualization, nodes can treat an overlay as if it is the native network, and multiple overlays can simultaneously use the same underlying overlay infrastructure.*

Despite all these changes in the infrastructure of the datacenter or the implementations of the networks, the need for investigations in networks still exists. Digital investigation methods are used to solve crimes committed with computers (Garfinkel, 2010) and to analyse unusual behaviour in digital systems like computers, networks or mobile phones. Digital investigation in virtual environments faces issues like data location, lifecycle of virtual machines (VMs), multitenancy and a valid chain-of-custody (Spiekermann et al., 2015), but this wide area of forensic in cloud-computing environments is still being researched (Dykstra and Sherman, 2013 or Ruan et al., 2013). Additionally network forensic investigations needs to resolve different arising issues like VM migration, user customization of the assigned virtual systems or the lack of physical network interface cards (NIC) (Spiekermann and Eggendorfer, 2016a). The area of network forensic investigation in

* Corresponding author.
E-mail addresses: daniel.spiekermann@fernuni-hagen.de (D. Spiekermann), joerg.keller@fernuni-hagen.de (J. Keller), tobias.eggendorfer@hs-weingarten.de (T. Eggendorfer).

http://dx.doi.org/10.1016/j.diin.2017.01.007

these environments is still under-explored, so this paper presents a framework for digital investigation in virtual networks and an implementation developed by the authors called Forensic Controller (ForCon) to provide a valid and consistent capture process in virtual networks.

The development of ForCon is based on a detailed analysis of new challenges in the virtual networks a forensic investigator is faced with. VMs are migrated from one host to another, or users customize their internal logical network which changes the encountered installation and prevents the further capture of network packets. We assume that these changes do not have to be malicious by the customer, more likely the change in the network is initiated by the cloud environment. These changes occur without any administrative input by the cloud service provider (CSP), only because of reaching predefined usage limits of the hardware resources like cpu-time or storage capacity.

Based on this assumption we discovered the need of implementing a new forensic framework to describe the network forensic investigation in virtual networks. Current frameworks exhibit a static implementation with only one identification process at the beginning of the investigation. We argue that this limited identification at the beginning is not sufficient in virtual networks. Even the migration of the target VM requires a renewed identification of the updated location of the VM. This led to the evolution of our framework called *Virtual Network Forensic Process* (VNFP), which implements a repetition of different tasks to ensure the consistent capture process.

The rest of this paper is structured as follows: Section Related work lists previous and related works related to virtual networks, network forensics and digital forensics in virtual environments. Section Virtual networking describes the different implementations of virtual networks, with a focus on software defined networks and network function virtualization. Network forensic investigation, the limitations of the current techniques and the virtual network forensic process framework are described in Section Network forensic investigation. In Section ForCon the implementation of ForCon is explained in detail. Section Evaluation provides an initial evaluation of our prototype. Section Conclusion concludes and gives an outlook to future work.

Related work

ForCon implements a workflow for network forensic investigation in virtual networks. Each of the current techniques is based on the capture, record and the subsequent analysis of the obtained data (Lazzez, 2013; Pilli et al., 2010a; Hunt and Zeadally, 2012 or Corey et al., 2002). Khan et al. (2016a) presents an overview of current research, taxonomy and open challenges of network forensics.

The process of a digital investigation is separated in different stages leading the investigator from the initial start to the concluding reporting of the analysis. Different frameworks describe this process for network forensic investigation. The first framework for network forensics was presented in Palmer (2001). Pilli et al. (2010b) lists 10 frameworks and summarizes the different phases, which led to the development of the generic framework for network forensics. Rasmi et al. (2013) lists current frameworks for network forensic investigation.

ForCon is a tool to analyse, monitor, identify, capture and, if necessary, adapt the network forensic process in OpenFlow networks. OpenFlow is explained in detail in McKeown et al. (2008) and Azodolmolky (2013), the evolutionary change of networks by Software Defined Networks (SDN) and the notability of OpenFlow is discussed in Kreutz et al. (2015). The proposed agents of ForCon interact with Open vSwitch as the involved vswitch. The implementation of Open vSwitch (OVS) is described in Pfaff et al. (2015), the communication between Open vSwitch and the SDN controller is handled with the OpenFlow protocol.

Different research is done in the field of using OpenFlow as a controlling unit for network traffic to implement IT security approaches or forensic investigations. Achumba et al. (2015) uses OpenFlow as a virtual appliance to implement a security monitoring interface. Bates et al. (2014) implements a security infrastructure based on OpenFlow by using middleboxes to monitor and analyse the transmitting network traffic. Khan et al. (2016b) proposes with a framework named FML, a forensic management layer to analyse the controlling tiers in SDN. Bremler-Barr et al. (2014) steers traffic by communicating with the native controller to implement deep-packet-inspection of network data.

Virtual networks raise different issues for digital investigation, which are discussed in great in Spiekermann and Eggendorfer (2016b) and E. T. S. I (2016). Most of the common tools are implemented to extract network information, not to capture all transmitted data transferred from or to a suspicious system. ForCon provides an entire packet capture process targeted to one system using distributed agents. A similar approach is discussed in Ren and Jin (2005), which uses agents to capture all traffic in a local net and transmit it to a network forensic server. ForCon extends this scope by following the target system and capturing only the relevant data without capturing all network traffic in the whole segment.

Virtual networking

The implementation of VMs in a datacenter infrastructure poses new problems to the provider. The administration of new VMs, the reconfiguration of running VMs or resource management of the hardware used was getting easier and more flexible. But the isolation of VMs of different customers or the interconnection between VMs of the same customer still requires a cumbersome and error-prone manual administration like reconfiguring current access control lists (ACL), VLAN[1]-tagging or routing information.

The limitation of current network infrastructures impedes the implementation of highly dynamic, flexible, secure and automated environments that might reduce the costs and provide a customizable network. These limitations are based on the one hand on the deployed network protocols like VLAN, which limits the number of different, logically separated networks, or spanning-tree-protocols, which reduce the number of usable interconnections between switches to only one, even if more links were available. On the other hand, the installed network devices do not provide interfaces, which allow the automated configuration of the connected devices based on previously defined rules or by analysing states depending on the current network situation.

These circumstances led to the development of new network protocols like Virtual eXtensible LAN (VXLAN), Stateless Transport Tunneling (STT) or Network Virtualization using Generic Router Encapsulation (NVGRE), and to the implementation of new paradigms like Software Defined Networks (SDN) and Network Function Virtualization (NFV).

Network protocols

The new network protocols try to eradicate different limitations of currently used network protocols like VLAN[2] or spanning-tree-protocols.[3] These implementations do not provide enough

[1] Virtual local area network.
[2] E. g. implemented by IEEE 802.1q.
[3] E. g. the Rapid Spanning Tree Protocol (RSTP).

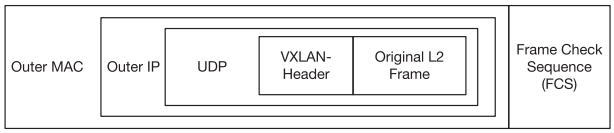

Fig. 1. VXLAN encapsulation.

flexibility, which is needed in virtual environments or have a limited hardware utilization.

VXLAN tries to increase the number of possible separated subnets in the datacenter from 2^{12} to 2^{24} by using an extended, 24 bit long tag in the ethernet frame. All hosts with the same tag-id belong to the same layer2-subnet. The implementation of VXLAN facilitates the spreading of this subnet all over the physical infrastructure. If one host wants to communicate with another host located in the same subnet but on another physical host in the datacenter, the network packet is encapsulated via VXLAN. This process of encapsulating is divided in the trailing VXLAN-header, a new UDP-header and then a new header with IPv4 or IPv6 to transmit the encapsulated packet to the next VXLAN tunnel endpoint (VTEP). Fig. 1 shows the encapsulated network packet.

The VTEP decapsulates and forwards the packet to the given destination of the original ethernet frame.

Other protocols like STT[4] or NVGRE[5] work in a similar way but provide the same generic technology of transmitting network packets inside other protocols, even with different means to attain the same goal.

Software defined networks

SDN provide a new flexibility in the design of networks. The main idea behind the development of SDN was the improvement of testing new network protocols on existing network devices without threatening the current network configuration. SDN provides different advantages like:

- **Separation of control and forwarding plane**
  A network switch uses an internal flow-table to forward the incoming network packets to the correct destination (e.g. only one port or by flooding the packet to all ports). This flow-table is a vendor-specific implementation, which decides the correct destination of network packets by using an internal table. This process is implemented at the control plane. The fast processing of the network packets to the correct output is performed by the forwarding or data plane.
  By decoupling these two planes and moving the forwarding plane to a separate device, the network management gets more centralized, which might reduce the error-prone and time-consuming manual configuration of each network switch.
- **Centralized configuration**
  This separation of planes and moving the forwarding plane to the SDN controller offers a centralized configuration of all connected switches. By configuring different rules for packet processing, the SDN controller transmits these rules to all connected switches. If a switch is not able to forward a network packet by its known rules, it forwards the packet or packet information to the controller. The controller analyses this data and informs the involved switch of the further processing of this kind of packets.
- **Programmability of the network**
  The central configuration improves the management of the devices. The communication between the SDN controller and the switch is realized by the southbound-API, the communication between the SDN controller and the higher applications is implemented with the northbound-API. These APIs provide the programmability of the network. Implementations of different applications, which act on the higher layer via the northbound-API, control the forwarding of network packets.

The most relevant protocol of the southbound-API is OpenFlow (Braun and Menth, 2014). The development started at Stanford University in 2008 (McKeown et al., 2008). OpenFlow enables the programmability of the network by providing an interface between a central network controller and the connected ethernet switches. These so-called OpenFlow-switches do not use an internal ruleset to forward the packets any more. Instead they use the central controller by transmitting packet information to the controller, which analyses this information and transmits the valid rule to the OpenFlow-switch. The communication protocol for exchanging this information is OpenFlow.

Network function virtualization

Network function virtualization (NFV) is another branch of virtual network implementations. In contrast to SDN there is no new protocol or technique to implement virtualization in networks, but different services like switching, routing, firewalls or intrusion detection service (IDS) which are typically realized by separated hardware devices are now provided as virtual appliances. These virtual appliances might be executed on standard hardware, which provides new advantages like vendor independence, cost reduction or improved orchestration of these devices.

OVS (Pfaff et al., 2009) is an implementation of an ethernet switch implemented in software. OVS provides layer2-switching, VLAN tagging or quality-of-service and runs on current linux systems. OVS is frequently used in virtual environments to realize the interconnection of VMs. Cloud environments like OpenStack (Sefraoui et al., 2012) use OVS as a virtual switch (vswitch), which interconnects the VMs independent from the hypervisor used.

Network forensic investigation

Network forensics is the science of digital investigation in networks (Pilli et al., 2010a). The network forensic investigation (NFI) is separated in two phases, which are classified as online and offline. The online phase comprises the capture and recording of relevant network packets, the subsequent analysis of this data is performed in the offline phase.

The recording of the data depends on the encountered network infrastructure (Nelson et al., 2014). Depending on the implementation, three different techniques to retrieve the data are common:

- **TAP**
  A tap is a special network device, which is interleaved in the connection between a system of interest and the next network component. The tap duplicates the transferred data and transmits it to the receiver and additionally to a second output.
- **SPAN**
  A span-port (or port-mirror) on a physical switch is a specially configured port which enables the mirroring of all received and transmitted data of the given port to another output port.
- **Bridge**
  If it is not possible to configure a port-mirror and taps are not available, the third technique is to interleave a specially

[4] Which uses an encapsulation with TCP.
[5] Implemented with an encapsulation with GRE.

configured system which acts nearly invisible in the network and stores all traversing network packets.

All three aforementioned possibilities are well-defined for the use in traditional networks, which have a static wiring without changing the hardware frequently. The captured data is stored on a separate storage medium, which is able to handle all incoming packets and write them to disk fast enough to avoid bottlenecks concerning storing and receiving new files.

After capture and recording of the relevant network data, the subsequent analysis entails the detailed investigation to reveal who has communicated with whom, how long this communication took, when it was performed and possibly which data was transferred.

Challenges

Different problems exist for network forensic investigation in virtual networks. Spiekermann and Eggendorfer (2016b) presents different problems of digital investigation in virtual networks, separated in online, offline and organizational problems.

There are two main critical problems, that impede the network forensic investigation.

- **Migration**
  Migration describes the process of moving a VM from one physical host to another (Clark et al., 2005). If this happens, the used network connections, connected links and switch ports get invalid and are reconfigured on the new physical host.
- **Customization**
  The customization process defines the change of the network or system implementation employed by the user without any additional influence by the CSP. The rate of possible changes depends on the environment and is typically predefined by the CSP. If the logical network of a VM is reconfigured by the customer, it might happen that the installed capture process fails and the recorded data gets incomplete.

Current processes in the online phase use static workflows and tools, which are designed and implemented for traditional networks. The static implementations fail when investigating virtual networks. These networks are defined by a highly flexible context, with VMs migrating from one physical host to another or a customization of the logical networks by the customer without any further administrative help of the provider.

The current recording techniques require a static wiring to get all incoming or outgoing packets from the suspicious system. As long as this wiring is not changed, the recording and capture process will retrieve all network packets.

If the suspicious system is migrated to another physical host, the capture process fails, the now used NIC has changed, which requires a manual and cumbersome reconfiguration of the recording technique, e.g. disconnecting the current switch ports, reconfiguring the port-mirror on another switch, delete the port-mirror on the now unused switch and reconnect the storage system with the new NIC.

But the migration process of the suspicious system is unpredictable. Based on external events like high performance demands of other VMs, installed on the same compute node, or a hardware failure, the migration of the VMs might be started without any further interaction by the provider.

New NFI framework

The current NFI process is defined by a NFI model which describes different tasks, performed sequentially without any adaptation within a phase. Some models like the Generic Network Forensic Process model (GNFP) (Pilli et al., 2010a) describe a revision of tasks, activated by new information, which lead to a complete new capture process. Adeyemi et al. (2012) analyses 23 different frameworks, which are designed for digital or network forensic investigation. But none of these frameworks are valid for the digital investigation in virtual networks. Some phases are still valid like the identification, record, storage and analysis of the network data, but there is no phase to recognize the critical actions.

Therefore the network forensic investigation process is not divided in four parts any more. The network forensic investigation process in virtual networks has to be extended to implement a subroutine of monitoring and adapting the capture process to guarantee a valid capture file, even with migration or customization. Fig. 2 illustrates the new process called Virtual Network Forensic Process (VNFP).

VNFP uses well-known and proved concepts of other digital forensic framework models. Phases like capture, recording or analysis are adopted from the GNFP. In addition to these reused models the phases monitoring, evaluation and adaptation are new in this model.

- **Monitoring**
  The aim of the monitoring phase is to implement an option to observe the network environment and send out an information about the change.
- **Evaluation**
  Not all changes pertain the running investigation process. By analysing the occurring events only relevant changes are noticed which instigate the subsequent adaptation.

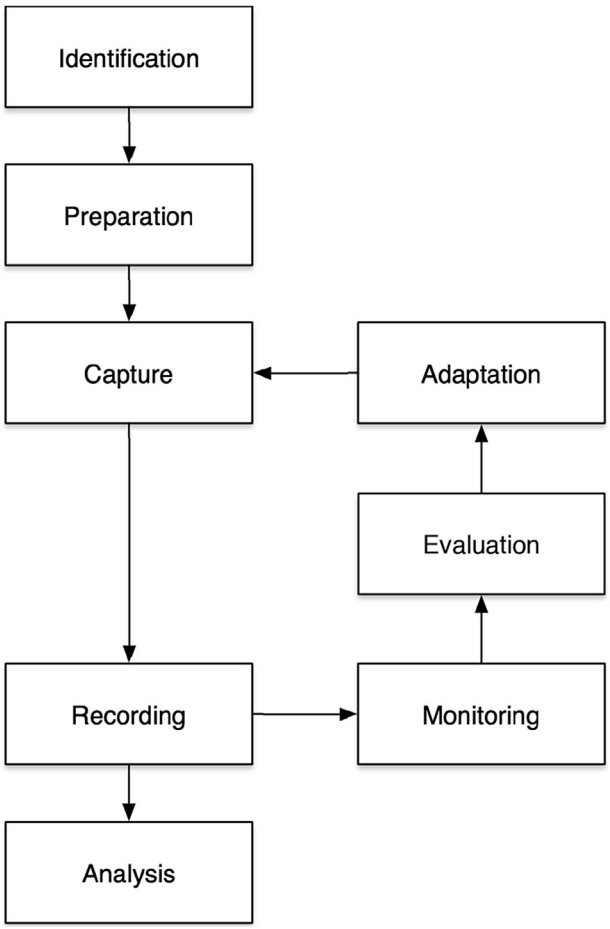

Fig. 2. Virtual network forensic process framework.

- **Adaptation**
  If the change of the network pertains the record process, an adaptation of this process is initiated. This adaptation has to edit the current capture process (e. g. by deleting the current installation and inserting new rules, filters or parameters or by reconfiguring the running process). If the changes of the network structure are too comprehensive, a restart with the new identification is necessary.

After adapting the capture process, the monitoring process is started again to monitor the network environment repetitive.

ForCon

The challenges discussed in Section Challenges arise from the evolution of highly dynamic, flexible and customizable virtual networks, which impede the network forensic investigation with current processes and techniques. These techniques are inflexible and rigid, as they are not focused on investigation in this fast-moving environment.

To achieve a successful digital investigation with a valid capture file, the capture process has to be flexible as well, which led to the need to virtualize the whole capture process.

The development of the tool ForCon was driven by the need to implement a capture process which is able to monitor the virtual environment and to provide a fast and valid reaction towards network changes.

The multitude of different implementations of virtual networks limits the possible approaches to a predefined environment. We analysed the OpenFlow protocol in combination with vswitches running with OVS, because it is widely used in practice.

Implementation

ForCon is implemented in Python and needs no additional setup routine on a given virtual or physical machine in the network. ForCon has a command-line interface. By passing the appropriate parameters ForCon controls its distributed agents to monitor the environment, ensuring a stable and correct forensic process. The parameters are

- Valid identifier of the suspicious VM[6]
- Port number listening for connections

ForCon uses a special message format, called ForCon Protocol (FCP), which implements seven different message types as described in Table 1.

These message types are exchanged with two types of agents, which implement different tasks of monitoring and manipulating of network flows. ForCon uses the management network of the environment to communicate with the agents, the customer network is not swayed by these messages.

There are two types of agents to fulfil the capture process.

- **SDN agent**
  These agents run on the compute nodes which host the different VMs and provide the connection to the vswitches. Each agent of a compute node analyses the installed vswitches and the connected devices of each vswitch. These information of the connected devices are transmitted via the *Informational*-message to ForCon. If necessary, ForCon informs the agent to manipulate the flows of the system, submitted within a *Manipulation*-message.

[6] This can be a mac-address, VID (VLAN-Identifier) or a combination.

Table 1
ForCon protocol message types.

Type	Identifier	Task
Informational	I	Transmission of identifiers
Flow request	F	Command to extract flows
Manipulation	M	Command to manipulate flows
Update	U	Update the information
Connection	C	Successfully connected to ForCon
Monitoring	X	Relevant event occurred
Delete	D	Command to delete the forensic flows
Tunnel	T	Create tunnel between involved vswitches

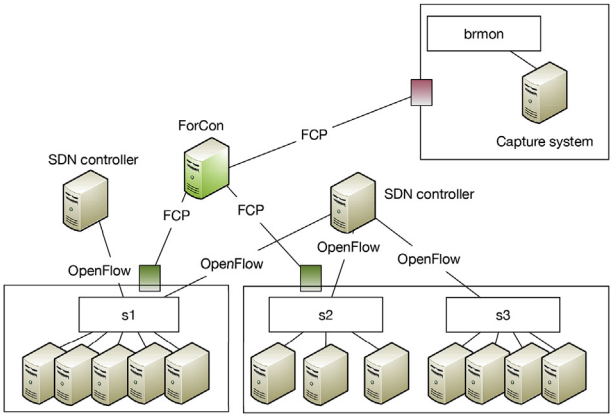

Fig. 3. ForCon architecture.

- **Mirror agent**
  Mirror agents run on physical hosts which connect the capture system to the network. This connection is based on an OVS instance, which is initiated without any further links to other systems. This guarantees the correct isolation of the capture system. If a *Tunnel*-message is received, the Mirror agent extracts the information of the tunnel endpoint and creates a tunnel with a VXLAN-encapsulation from the OVS instance to the transmitted destination vswitch. Only now the capture system is connected via the tunnel to the network. By manipulating the flows, the mirrored data is transmitted from the target system via the tunnel to the capture system. If the network topology changes, the Mirror agent has to dismantle the current tunnel and reconstruct another tunnel to the new involved vswitch. The use of tunnels based on VXLAN facilitates the installation of the capture system somewhere in the network. Because of this, only one mirror agent is needed to perform the network forensic investigation in the network.

Fig. 3 describes the architecture with ForCon running in a network with two SDN controllers managing three vswitches. SDN agents are marked in green (in the web version), the mirror agent is marked in red.

After initializing all information, ForCon is waiting for a connection (via a *Connection*-message) of an agent which is signalized by the IP-address of the agent.

```
Connection from Agent 172.16.40.129 established
```

ForCon orders the installed flows of the switches installed on the compute node via a *Flow-Request*-message. The agents receive the message and transmit the information to ForCon.[7]

[7] In this example the *Informational*-message uses a format with *vswitch-identifier-SEPARATOR-source-mac-address-SEPARATOR-destination-mac-address*.

```
[172.16.40.1] ->F
Received F > extract Flows
I;s1;00:00:00:00:00:01;00:00:00:00:00:03
I;s1;00:00:00:00:00:03;00:00:00:00:00:01
I;s1;00:00:00:00:00:03;00:00:00:00:00:02
I;s1;00:00:00:00:00:04;00:00:00:00:00:03
```

ForCon analyses the information and, if necessary, sends a *Tunnel*-message to the mirror agent running on a separated host and the involved SDN agent.

```
T;172.16.12.140;brmon;172.16.12.129;s1
```

The mirror agent establishes a tunnel connection to the vswitch on the submitted IP-address and the vswitch used. In this example a tunnel between the mirror agent running on the system identified by the IP-address *172.16.12.140* creates the tunnel on the vswitch named *brmon* with the target *172.16.12.129*. The SDN agent running on *172.16.12.129* creates the tunnel on vswitch *s1* vice versa.

After exchanging these messages, the tunnel between the vswitch connected to the target VM and the vswitch connected to the capture system is established. The next step is to manipulate the assigned flows.

Initial capture

To capture the data of the suspicious system, all flows containing information of the target system have to be manipulated.

The manipulation of the flows is initiated by a *Manipulation*-message, transmitted from the server to the agent. This message contains the valid identifier of the target system, which facilitates the receiving agent to manipulate the flows.

```
Received M -> M;00:00:00:00:00:03;s1
```

As described in Section Software Defined Networks, the structure of OpenFlow messages is pre-defined, but each controller uses vendor specific additional information. In our research, we use *Floodlight* and *Ryu* as the central SDN controllers and analyse the structure of the messages to find coinciding elements.

There are two types of flows that have to be manipulated to gather all network data of the suspicious system. The incoming packets with the destination address of the target system are defined as *Ingress*, all data leaving the target system are defined as *Egress* data.

To capture the ingress data, the flow manipulation is easier, it is sufficient to add the id of the tunnel interface as an additional output port to the action part.

```
cookie=0x0, duration=11965.378s, table=0,
n_packets=10120,n_bytes=975632,priority=1,
in_port=1,dl_dst=00:00:00:00:00:03
actions=output:2
```

is manipulated to

```
cookie=0x0, duration=1362.132s, table=0,
n_packets=10120, n_bytes=975632,priority=2,
in_port=1,dl_dst=00:00:00:00:00:03
actions=output:2,99
```

Additionally the priority field is incremented to guarantee the use of this flow.

The capture of the egress data is more complicated because of the lack of usable information. Because of this, ForCon adds additional flows to the flow-table which use the identifier[8] as source. Now the agent adds a flow for each mac-address found in the same layer2-broadcast domain, which uses the mac-address of the suspicious VM as source and the mac-address found as destination, and inserts the current output port for the destination and additionally the output port to the capture system.

```
cookie=0x0, duration=1604.682s, table=0,
n_packets=0, n_bytes=0,
priority=2,dl_src=00:00:00:00:00:03,
dl_dst=00:00:00:00:00:01
actions=output:1,output:99

cookie=0x0, duration=1604.690s, table=0,
n_packets=0, n_bytes=0,
priority=2,dl_src=00:00:00:00:00:03,
dl_dst=00:00:00:00:00:02
actions=output:2,output:99

cookie=0x0, duration=1604.699s, table=0,
n_packets=0, n_bytes=0,
priority=2,dl_src=00:00:00:00:00:03,
dl_dst=00:00:00:00:00:04
actions=output:4,output:99
```

Each new flow is tagged with an incremented priority to ensure the use of this flow instead of using an original flow inserted by the SDN controller. This implementation guarantees that every packet to or from the suspicious system is copied to the capture system, too.

Monitoring

The initial capture implementation realizes the first step to record all relevant network data to or from the suspicious system. But the dynamic nature of the virtual environment results from the possibility to migrate VM from on host to another or by the customization of the infrastructure. This leads to the need of a monitoring process after implementing the initial capture process.

This monitoring process is not limited to only one physical host, because the migration process of a VM is typically not predictable. So each compute node in the datacenter is a possible host for the VM to be moved to.

ForCon provides the monitoring of the compute nodes by using its distributed SDN agents, each SDN agent has to monitor the compute node it is assigned to. If the suspicious VM is migrated, the central SDN controller has to change the relevant flows on the involved vswitches. The SDN agent on the prior compute node recognizes the change, and a *Monitoring*-message is sent to ForCon. ForCon reacts to this change by sending out an *Update*-message to all connected SDN agents in the network.

The monitoring of the SDN agent depends on the installed vswitch. In our approach, the agents are responsible for OVS, so it is possible to use given monitoring implementations of OVS. Information of network changes are provided in several ways. They are logged in various locations of the installation of the infrastructure or are provided by APIs. The relevant information can be extracted from

- Logfiles of
  - compute node
  - vswitch
  - cloud controller
  - SDN controller
- Database of the vswitch
- Controller-to-vswitch communication
- Monitoring method

[8] In this example we use the mac-address as the valid identifier.

The logfiles are typically vendor specific, which leads to a difficult and cumbersome analysis of the structure. Each update or newer version of the software may invalidate the previous analysis. The controller-to-switch communication transfers all relevant data between SDN controller and vswitch, but the communication is encrypted and realized with a so-called secure channel (McKeown et al., 2008). The database of the vswitch stores all relevant information for a consistent workflow (Pfaff and Davie, 2013). The extraction of relevant information is a possible way to retrieve the data, but to notice the changes, lots of entries of the database have to be stored duplicated in a special data structure of the agent.

Because of this, the SDN agent of ForCon uses a monitoring-method provided by OVS. This method is used by a connecting listener, which gets information about changes of the OVS database (OVSDB). With this implementation a timely reaction of changes in the network is possible.

After initializing, the SDN agent connects to the monitor-method of the running vswitch-instance. After connecting, the agent gets all information submitted to the monitor method. This comprises keep-alive messages, which are sent periodically between vswitch and controller to inform about the availability. The agent has to parse the variety of messages to filter out the relevant events. If a possibly relevant event occurs, the SDN agent checks the significance. If the message notifies about a port-deletion of the targeted system, a migration might start, and ForCon gets informed by the agent with the *Monitoring*-message, which is sent to the controller.

Changes

The *Update*-message from ForCon causes the agents to extract all connected local vswitches again and transmit all extracted information again back to ForCon. ForCon analyses this data and informs the newly involved agent about the needed manipulation via a *Tunnel*-message and a *Manipulation*-message. The other agents get the *Delete*-message to remove the manipulated flow entries and to delete the tunnel between the storage system and the former vswitch.

Evaluation

To demonstrate the correct capture of all relevant data, we evaluated ForCon in different scenarios. At first we validated ForCon in a separate testbed which consists of three compute nodes hosting the VMs, a cloud controller running OpenStack with Neutron as the controlling unit for the network infrastructure. Ryu was installed as the SDN controller which interacts with OVS instances running on the compute nodes. ForCon was imported to this testbed and started with the mac-address of a running VM as the initial parameter. The agents extracted the flows and informed ForCon to analyse the data. ForCon identified the VM and instructed the involved agent to manipulate the flows. The captured data was recorded with the capture system running on a separate host. After that the target VM was migrated manually to another host. The agent of the prior host noticed this change and informed ForCon about the need of a update of the network status. ForCon informed all agents to extract the flows again, identified the VM successfully and reconfigured the capture process.

Different values might affect the mirroring and capture process and might hamper the further investigation. We assume the following three parameters as the most relevant.

- **Cpu load**
  A high cpu load might hamper the process by discarding packets which leads to an incomplete capture file.
- **Number of entries in the CAM**
  The content addressable memory (CAM) stores the mapping of port and mac-address. If the CAM has to handle lots of entries, the search for the correct ports might take longer. Additionally, the flow manipulation might be affected by this elongation.
- **High network utilization**
  The hosting system has to manage all network traffic by calculating the relevant checksums, port mappings and the transfer of packets from or to the VMs. An increase of the overall network usage might slow down this calculation process, whereby the capture process might be constrained.

We adjusted the values with common tools and scripts, i. e. the cpu load is manipulated with the linux tool *stress* and the high network usage is performed with by using the tool *iperf*. The high number of entries in the CAM table is simulated by configuring hundreds of vNICs inside the hosting system. The values are gradually increased to examine a possible disruption of the capture process.

We captured the network traffic on the suspicious and the capture system simultaneously to compare the entire number of captured packets. By using predefined network captures with a known number of packets we were able to validate ForCon.

We injected six different capture files in the testbed, each containing a defined number of network packets. Table 2 lists the average values of a total of three runs with of 174,772 network packets transmitted from or to the target system. The overall duration of processing the packets slowed down, but no packets got lost. The column P_t lists the number of packets captured on the target system and the column P_c lists the number of packets received by the capture system.

The successful capture of all packets depends on different circumstances. ForCon uses only one mirror port installed on a special vswitch managed by the mirror agent. Therefore known problems such as a loss of performance while increasing the number of vswitches as discussed in Ohira (2014) do not eradicate the ForCon process. The implementation in software manages the entire process of forwarding, therefore the system knows how and whereto the packets have to be forwarded. Even if the calculation takes longer, the cpu achieves the valid transmission to all given destinations.

ForCon manipulates only flows which are relevant for the capture process and ignores flows assigned to other systems. This limitation provides the fast processing of submitted flows via the *Informational*-message.

The evaluation demonstrates the correct processing of network packets, even in environments with a high utilization. Being still a proof-of-concept ForCon provides a valid process of capturing network packets of a given target VM.

Conclusion

Network forensic investigation in virtual networks has several issues. Most critical are techniques like migration and user customization, which change the circumstances for the capture process. New network protocols impede the subsequent offline

Table 2
Evaluation of ForCon

Scenario	P_t	P_c	Duration
No manipulation	174,772	174,772	4.1 s
Cpu load	174,772	174,772	4.4 s
CAM	174,772	174,772	4.2 s
Network usage	174,772	174,772	4.4 s

analysis of the captured data, because of the lack of usable tools.

To implement a successful investigation, we first examined current network forensic frameworks to determine the ability to apply one of the proved frameworks in virtual networks. Current frameworks describe the need of identification, capture, record and analysis. But the critical actions require additional steps with a repetition of defined tasks. These tasks have to cover the monitoring and identification of the changes, and the adaptation to the changed environment. None of the analysed frameworks provide a suitable model, so we presented VNFP as a new forensic framework for investigation in virtual networks. This framework implements three additional phases named monitoring, evaluation and adaptation to handle the flexible behaviour in virtual networks.

Based on VNFP we developed ForCon as an implementation of a specialized OpenFlow controller. ForCon is able to implement a network forensic investigation in OpenFlow networks by extracting and analysing the valid flows installed on the layer2-switches. Based on submitted parameters ForCon is able to find the suspicious target system in the virtual environment and to configure flow entries, which mirror all relevant data to a given destination. To realize this workflow, ForCon uses two different types of distributed agents running on the physical hosts. One type of agents extracts the information of connected VMs and the flows of each vswitch and transfers this information to ForCon. ForCon gathers all data from the connected agents and informs the agent which monitors the vswitch with the suspicious target system. This agent manipulates the flows to mirror all incoming and outgoing network traffic to the capture system. The capture system is connected via the second type of agent, which establishes a tunnel between the target vswitch and the prepared vswitch interconnected with the capture system, which stores all incoming traffic.

If an event occurs in the network (e. g. a migration or customization by the user), the central OpenFlow controller is informed and manipulates the assigned flows. The distributed agents notice this change of flows and transmit the new data to ForCon. ForCon checks whether this change is relevant and informs the involved agents to adapt their settings if necessary.

This separation of tasks and central information treatment provides a fast and easy to review workflow, which implements a fully automatic network forensic investigation in virtual networks. The agent-based implementation allows an adaptation for other vswitches. We use OVS as the running vswitch system, other implementations of vswitches can be monitored and configured by implementing a special agent which is able to communicate with the vswitch-API.

ForCon was evaluated and validated in different simulated scenarios. Hereby the valid processing was demonstrated. Running ForCon in highly loaded systems does not affect the overall load, at the same time the manipulation and mirroring actions of ForCon are not pertained by this utilization. As a further improvement, the overall speed might be increased by using a filter mechanism processed by the distributed agents, but this requires an adapted implementation which still needs to be evaluated.

ForCon implements successfully the three new steps *Monitoring*, *Evaluation* and *Adaptation* of VNFP, which eradicates the possible breakdown of a running capture if a VM migrates or is customized.

Even being a proof-of-concept, ForCon is the first implementation which guarantees a valid packet capture process in virtual networks. With the submitted parameters like mac-address or VLAN-tag, it is able to find, monitor and follow the target system and mirror all assigned network traffic to a capture system. A validation in a separate testbed approved the valid workflow of ForCon. Nevertheless a detailed validation in networks with more hosts, higher throughput and different configurations should be pursued to ensure the process of identifying the target VM, installing the needed recording and capturing process and, if necessary, adapting this installed capture process.

This capture system is connected via the so-called Mirror-agent within in VXLAN-tunnel to the relevant vswitch, where the target VM is connected.

Future implementations of ForCon should provide a dynamic choice of possible tunnel implementations like GRE or VLAN, which offers an additional benefit to guarantee a valid capture.

Virtual networks do not only interconnect VMs or establish an internet connection to the web, they also provide a connection to storage subsystems (Wu et al., 2010). A customer of a virtual environment is not limited to use only a VM, the provider supplies storage capacity, which might be accessed without the use of a VM. If a customer uses this kind of object storage, ForCon is not able to monitor this traffic because of the lack of OpenFlow connections and missing valid parameters to detect the connection. Future work has to be done to implement a network forensic investigation to storage-as-a-service environments.

References

Achumba, I.E., Okafor, K.C., Ezeh, G.N., Diala, U.H., 2015. Openflow virtual appliance: an efficient security interface for cloud forensic spyware robot. Int. J. Digit. Crime Forensics (IJDCF) 7 (2), 31–52.

Adeyemi, I.R., Razak, S.A., Azhan, N.A.N., Identifying Critical Features for Network Forensics Investigation Perspectives. CoRR.

Anderson, T., Peterson, L., Shenker, S., Turner, J., 2005. Overcoming the internet impasse through virtualization. Computer 38 (4), 34–41. http://dx.doi.org/10.1109/MC.2005.136.

Azodolmolky, S., 2013. Software Defined Networking with OpenFlow. Packt Publishing Ltd.

Bates, A., Butler, K., Haeberlen, A., Sherr, M., Zhou, W., 2014. Let SDN be your eyes: secure forensics in data center networks. In: Proceedings of the NDSS Workshop on Security of Emerging Network Technologies (SENT 14).

Braun, W., Menth, M., 2014. Software-defined networking using openflow: protocols, applications and architectural design choices. Future Internet 6, 302–336.

Bremler-Barr, A., Harchol, Y., Hay, D., Koral, Y., 2014. Deep packet inspection as a service. In: Proceedings of the 10th ACM International on Conference on Emerging Networking Experiments and Technologies. ACM, pp. 271–282.

Clark, C., Fraser, K., Hand, S., Hansen, J.G., Jul, E., Limpach, C., Pratt, I., Warfield, A., 2005. Live migration of virtual machines. In: Proceedings of the 2nd Conference on Symposium on Networked Systems Design & Implementation, vol. 2. USENIX Association, pp. 273–286.

Corey, V., Peterman, C., Shearin, S., Greenberg, M.S., Van Bokkelen, J., 2002. Network forensics analysis. IEEE Internet Comput. 6 (6), 60–66.

Davie, E.B., Gross, J., April 2014. A Stateless Transport Tunneling Protocol for Network Virtualization. https://tools.ietf.org/html/draft-davie-stt-00.

Dykstra, J., Sherman, A.T., 2013. Design and implementation of frost: digital forensic tools for the openstack cloud computing platform. Digit. Investig. 10, 87–95.

E. T. S. I, Jan. 2016. Lawful Interception (LI); Cloud/Virtual Services for Lawful Interception (LI) and Retained Data (RD). Tech. Rep. ETSI TR 101 567. European Telecommunications Standards Institute.

Garfinkel, S.L., 2010. Digital forensics research: the next 10 years. Digit. Investig. 7, 64–73.

Garg, P., Wang, Y., September 2015. NVGRE: Network Virtualization Using Generic Routing Encapsulation. RFC 7637 (Informational). https://tools.ietf.org/html/rfc7637.

Hunt, R., Zeadally, S., 2012. Network forensics: an analysis of techniques, tools, and trends. Computer 45 (12), 36–43.

Khan, S., Gani, A., Wahab, A.W.A., Shiraz, M., Ahmad, I., 2016. Network forensics: review, taxonomy, and open challenges. J. Netw. Comput. Appl. 66, 214–235.

Khan, S., Gani, A., Wahab, A.W.A., Abdelaziz, A., Bagiwa, M.A., 2016. FML: a novel forensics management layer for software defined networks. In: 2016 6th International Conference-Cloud System and Big Data Engineering (Confluence). IEEE, pp. 619–623.

Kreutz, D., Ramos, F.M., Verissimo, P.E., Rothenberg, C.E., Azodolmolky, S., Uhlig, S., 2015. Software-defined networking: a comprehensive survey. Proc. IEEE 103 (1), 14–76.

Lazzez, A., 2013. A survey about network forensics tools. Int. J. Comput. Inf. Technol. 2 (1), 74–81.

Mahalingam, M., Dutt, D., Duda, K., Agarwal, P., Kreeger, L., Sridhar, T., Bursell, M., Wright, C., Aug. 2014. Virtual eXtensible Local Area Network (VXLAN): A Framework for Overlaying Virtualized Layer 2 Networks over Layer 3 Networks. RFC 7348 (Informational). http://www.ietf.org/rfc/rfc7348.txt.

McKeown, N., Anderson, T., Balakrishnan, H., Parulkar, G., Peterson, L., Rexford, J., Shenker, S., Turner, J., 2008. Openflow: enabling innovation in campus networks. ACM SIGCOMM Comput. Commun. Rev. 38 (2), 69–74.

Nelson, B., Phillips, A., Steuart, C., 2014. Guide to Computer Forensics and Investigations. Cengage Learning.

Ohira, K., 2014. Performance evaluation of an openflow-based mirroring switch on a laptop/raspberry pi. In: Proceedings of the Ninth International Conference on Future Internet Technologies, CFI '14, ACM, New York, NY, USA, 20:1–20:2.

Palmer, G., 2001. A Framework for Digital Forensic Science, a Roadmap for Digital Forensic Research, pp. 15–20.

Pfaff, B., Davie, E.B., December 2013. The open vswitch database management protocol. RFC 7047 (Informational). https://tools.ietf.org/html/rfc7047.

Pfaff, B., Pettit, J., Amidon, K., Casado, M., Koponen, T., Shenker, S., 2009. Extending networking into the virtualization layer. In: Hotnets.

Pfaff, B., Pettit, J., Koponen, T., Jackson, E., Zhou, A., Rajahalme, J., Gross, J., Wang, A., Stringer, J., Shelar, P., Amidon, K., Casado, M., 2015. The design and implementation of open vswitch. In: 12th USENIX Symposium on Networked Systems Design and Implementation (NSDI 15), pp. 117–130.

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

Pilli, E.S., Joshi, R., Niyogi, R., 2010. A generic framework for network forensics. Int. J. Comput. Appl. 1 (11), 14–27.

Rasmi, M., Jantan, A., Al-Mimi, H., 2013. A new approach for resolving cyber crime in network forensics based on generic process model. In: The 6th International Conference on Information Technology (ICIT 2013).

Ren, W., Jin, H., 2005. Distributed agent-based real time network intrusion forensics system architecture design. In: 19th International Conference on Advanced Information Networking and Applications (AINA'05) Volume 1 (AINA Papers), vol. 1. IEEE, pp. 177–182.

Ruan, K., Carthy, J., Kechadi, T., Baggili, I., 2013. Cloud forensics definitions and critical criteria for cloud forensic capability: an overview of survey results. Digit. Investig. 10 (1), 34–43.

Sefraoui, O., Aissaoui, M., Eleuldj, M., 2012. Openstack: toward an open-source solution for cloud computing. Int. J. Comput. Appl. 55 (3), 38–42.

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

Spiekermann, D., Eggendorfer, T., 2016. Towards digital investigation in virtual networks: a study of challenges and open problems. In: Proc. 11th International Workshop on Cyber Crime (IWCC 2016). IEEE Computer Society, pp. 406–413.

Spiekermann, D., Eggendorfer, T., Keller, J., 2015. Using network data to improve digital investigation in cloud computing environments. In: High Performance Computing & Simulation (HPCS), 2015 International Conference on. IEEE, pp. 98–105.

Wu, J., Ping, L., Ge, X., Wang, Y., Fu, J., 2010. Cloud storage as the infrastructure of cloud computing. In: Intelligent Computing and Cognitive Informatics (ICICCI), 2010 International Conference on. IEEE, pp. 380–383.