Hindawi
Journal of Computer Networks and Communications
Volume 2018, Article ID 4127487, 12 pages
https://doi.org/10.1155/2018/4127487

*Research Article*

# An SDN-Based Approach to Ward Off LAN Attacks

**René Rietz,[1] Radoslaw Cwalinski,[1] Hartmut König,[1] and Andreas Brinner[2]**

[1]*Brandenburg University of Technology, Department of Computer Science, Group Computer Networks, PF 101344, Cottbus, Germany*
[2]*Genua GmbH, Domagkstraße 7, 85551 Kirchheim Near Munich, Germany*

Correspondence should be addressed to Hartmut König; hartmut.koenig@b-tu.de

Received 24 May 2018; Revised 20 September 2018; Accepted 14 October 2018; Published 21 November 2018

Guest Editor: Ting Wang



The detection of attacks on large administrative network domains is nowadays generally accomplished centrally by analyzing the data traffic on the uplink to the Internet. The first phase of an infection is usually difficult to observe. Often attackers use e-mail attachments or external media, such as USB sticks, hardware with preinstalled malware, or contaminated mobile devices to infect target systems. In such scenarios, the initial infection cannot be blocked at the network level. The lateral movement of attack programs (exploits) through internal networks and the exfiltration of data, however, which are the main purpose of targeted attacks, run always over the network. Security measures against such internal network attacks require a comprehensive monitoring concept that spans the entire network to its edge. Especially for preventive measures, this means providing a security concept for local area networks (LANs). In this paper, we propose based on an analysis of typical LAN-based attacks an approach for preventing these attacks for both IPv4 and IPv6 networks. It applies the software-defined networking (SDN) paradigm for centralizing the related network decisions in a central authority—the SDN controller—that manages all network connections and hence the associated data flows.

## 1. Motivation

Capturing traffic to detect attacks on larger administrative network domains, e.g., an enterprise network composed of multiple local area networks (LANs), is nowadays typically centralized by picking up and analyzing traffic on the uplink to the Internet. This approach allows one to identify attacks from the Internet, but it has though a couple of important disadvantages. Insider attacks are not detected regardless of whether they are initiated deliberately or triggered by compromised devices. External attacks are equally difficult to detect because the initial compromise often takes place via mail by means of unknown vulnerabilities in file attachments or by simple social engineering, which tricks the recipient to run the executable code from mail attachments. In addition, threats, such as references to external web-based content as they are used for phishing and attacks using web-based content, are not recognized by existing preventive and reactive security measures.

The problems that current monitoring technologies have in detecting such attacks is that these attacks consist of different phases. The first phase of an infection (*phase of initial compromise*) is often carried out by an e-mail attachment or a contaminated USB stick on a PC in one of the local subnets. As these activities take place only locally on this PC, they are outside the viewing range of network-based monitoring systems. In order to attack servers in other subnets, an escalation of privileges has to be performed to bridge the intermediate system between two network segments. This can be done, for instance, using a domain-controller-based login on a PC in the subnet of interest with locally captured login information from the first infected PC—a step that is difficult to detect because the login may be legal. Next, the attack program needs information about the respective network segment. For this, scans of the link layer are often deployed to collect information about other systems (*internal reconnaissance*). Existing monitoring methods, e.g., the flow analysis, use primarily accounting information of the network and transport layer [1] and are thus unable to detect actions at the data link layer. Therefore, an attacker can use any link layer attack to propagate into the

target segment (*lateral movement*) to collect more data from all servers and to move the data within the network domain. Moreover, these attacks are not limited to the data link layer. Due to the only slowly growing deployment of the next Internet protocol (IP) standard, IPv6-based attacks are often overlooked by current monitoring systems. In a last step, the collected data must be moved out of the network (*exfiltration of data*). According to analyses of targeted attacks, such as the Regin framework [2], this step can also be performed quite stealthily, e.g., by means of the server message block (SMB) protocol for intermediate stations (data-link-layer variant) and a Transport Layer Security (TLS) socket for the final move out of the network.

In order to solve these monitoring issues, additional security measures have to be introduced into such corporate/private networks. This leads to new challenges. However, preventive security measures for the Internet uplink, e.g., packet filtering, proxies, or an application level gateway (ALG), are easy to implement because there is only one data path, and a monitoring concept has to be developed that ensures an equivalent security level for a domain network consisting of several local networks connected by switches with multiple ports and data paths. In this paper, we present an approach to prevent layers 2 and 3 (L2/3) attacks on local area networks which are directed against the switching elements and the synchronization at these layers and which often are deployed to circumvent higher-level monitoring measures. We apply software-defined networking (SDN) [3, 4] as a vehicle for centralizing information on related network activities in a central controller authority. SDN provides the ability to separate the control and the data plane in a switch. As a result, the switch logic can be outsourced to a separate controller. Thus, decisions regarding packet forwarding are no longer made autonomously in the switch but can be passed to the central controller [5]. The approach requires no changes to the host systems and allows one to effectively prevent attacks at layers 2 and 3. The SDN paradigm has accelerated the discussion about new efficient methods for controlling and managing computer networks, for making them "programmable" [6]. It has attracted much attention. Regarding security, SDN possesses two facets. It offers a lot of advantages to make networks more secure, but it also enables new vulnerabilities and attacks, as discussed only recently [7]. Possible attack vectors exploit vulnerabilities across controller, switches, and their communication channel [8]. On the contrary, with its holistic control of the network devices and a standardized interface to interact with them, SDN enables new, cross-platform security mechanisms to prevent, detect, and react to network attacks [9]. Our approach considers the usage of SDN for improving network security. The reminder of the paper is organized as follows. In Section 2, we exemplarily introduce some typical attacks on local area networks. Existing approaches to ward off these attacks are discussed in Section 3. Thereafter, in Section 4, we introduce the principle of our approach for preventing L2/3 attacks on local area networks using the software-defined networking paradigm. In the subsequent Section 5, we introduce several security services that can be included in the SDN controller to disable the attacks described in Section 2. Some final remarks conclude the paper.

## 2. Threats to Local Area Networks

In local area networks, there are plenty of vulnerabilities that allow a traffic redirection with the possibility of reading and overwriting content [10]. Related attacks focus on layers 1 and 2 of the TCP/IP stack. One indication of the actual use of these attacks is the application of appropriate techniques in the Archimedes framework (https://wikileaks.org/vault7/#Archimedes). In this section, we present some examples of these attacks in the context of both the traditional IPv4 and the more modern IPv6 networks.

*2.1. IPv4-Based Attacks.* Typical examples for these attacks are ARP scan, ARP spoofing [11], port stealing, DHCPv4 starvation [12], and DHCPv4 spoofing. As representative examples, we describe in more detail ARP scan and port stealing here.

*2.1.1. ARP Scan.* In a first attack step, an attacker or an attack program has to explore the local network. The attacker can already obtain first information from existing hosts or even compromised hosts. The current IP address, the subnet mask, and/or the default gateway address may reveal the maximum size of the network because gateway addresses are usually reserved to the upper end of the network range. Then, the attacker can scan the network range based on the previously acquired information. The main scan variant used is the Address Resolution Protocol (ARP) scan (Figure 1).

The purpose of the ARP scan is to look for active devices in the subnet. For this, the attacker (e.g., station A) usually generates a list of all possible host addresses and checks them using ARP requests (*ares_op$REQUEST*) (Step 1–4). The addresses are shuffled (*ar$tpa*) to request them in a random order. (Step 5) If there is a host that matches an address, it responds with an ARP reply (*ares_op$REPLY*), containing its MAC address ($ar\$sha = \mathrm{MAC}(< \mathrm{Host} >)$). To perform this scan, the attacker requires a large number of requests ($\#\mathrm{Requests} \gg \#\mathrm{Hosts}$). After determining potential targets of interest, e.g., routers or servers, the attacker is capable of kidnapping individual or even all links in the network in a further step. In this respect, one has to distinguish between attacks with half- and full-duplex capabilities. Half-duplex attacks kidnap only one direction of the communication (e.g., to the Internet), e.g., by spoofing the hardware address of the gateway. Data that are routed from the client through the gateway to the Internet may be intercepted and manipulated by the attacker. The responses in the reverse direction, however, are sent directly to the client. Full-duplex attacks can manipulate communications in both directions.

*2.1.2. Port Stealing.* The purpose of port stealing is to "steal" traffic that is directed to another port of a LAN switch (Figure 2). If an attacker A wants to directly take over packets addressed to another host C1 from the switch he/she

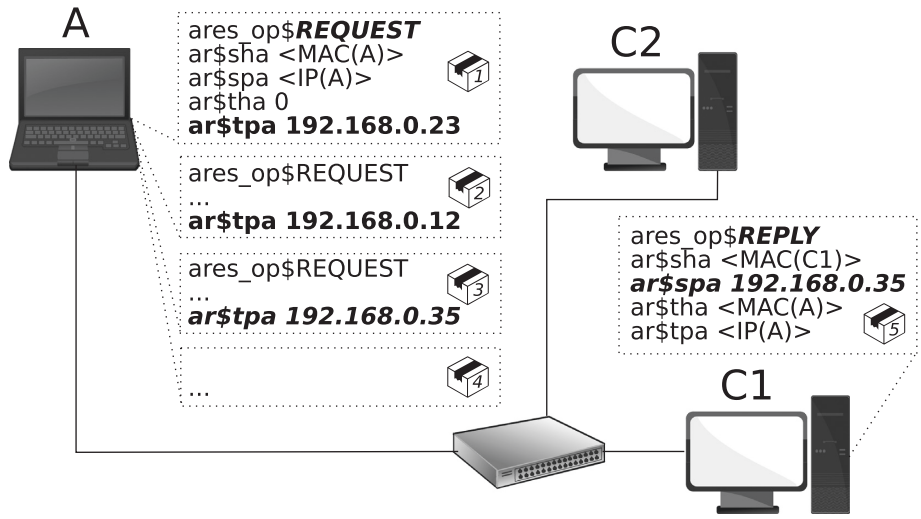


Figure 1: ARP scan.

first must delete the port registration of C1. This can be achieved by repeatedly sending ARP request frames to the switch in which the source MAC address is the one of C1 and the destination address is the MAC address of A (Step 1–3). The switch assigns the MAC address of C1 to the port of A, but it does not forward the ARP frames because A has addressed itself, i.e., the attack is stealthy. If the attacker receives a packet with destination C1 (in the example from C2) (Step 4), he/she sends an ARP request via broadcast to C1 and asks for its IP address (Step 5). After receiving the ARP reply (Step 6), the attacker knows that the switch has registered the MAC address of C1 again to the original port and can forward the intercepted (and possibly manipulated) packet to C1.

*2.2. IPv6-Based Attacks.* Due to the slow spreading of IPv6, a situation has arisen in which each installed network device (router, host) is IPv6-capable, while the protocol is often not used actively. Because of the intended transition from IPv4 to IPv6, the latter has automatically priority in the case of a simultaneous configuration of IPv4 and IPv6 parameters. Attackers can use this fact to examine the network using IPv6 methods and hijack individual connections. Existing monitoring methods are often not able to analyze the IPv6 protocol—a situation that makes IPv6 attacks particularly attractive. Typical examples for IPv6-related attacks are IPv6 multicast alive scan, ICMPv6 neighbor discovery spoofing, ICMPv6 router advertisement spoofing, and firewall circumvention with IPv6 fragment headers. As representative examples, we describe the multicast alive scan and the router advertisement spoofing here.

*2.2.1. IPv6 Multicast Alive Scan.* IPv6 does not use ARP anymore to find the MAC address for a given IP address. In IPv6, active addresses can be determined through a network discovery using multicast alive scans (Figure 3). The attacker sends only a single ICMPv6 EchoRequest packet with an invalid IPv6 destination option to the all-nodes multicast address (FF02 : 1) (Step 1). If the attacker is only interested in local routers, he/she can choose the all-routers multicast

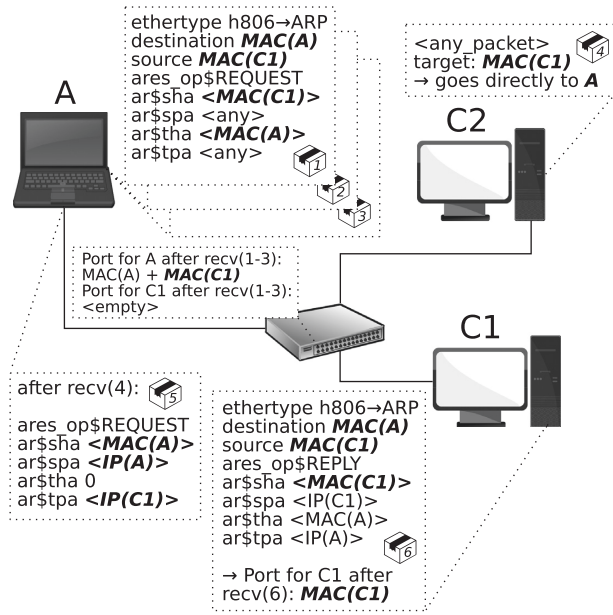


Figure 2: Port stealing.

address (FF02 : 2). All nodes in the network reply with an error message (the ICMPv6 parameter problem) that contains their address in the header because of the incorrect option (Steps 2-3). A similar network scan is possible by using a multicast listener general query [13] to address FF02 : 1. In this case, the hosts respond with a multicast listener report for each network interface. An advantage of the second approach is that the packet rate of the responding hosts to the multicast queries decreases due to random delays of the response which enables an evasion of possible anomaly detection systems.

*2.2.2. ICMPv6 Router Advertisement Spoofing.* The basic idea behind ICMPv6 router advertisement spoofing [14] is the same as for DHCP and DNS spoofing in traditional IPv4

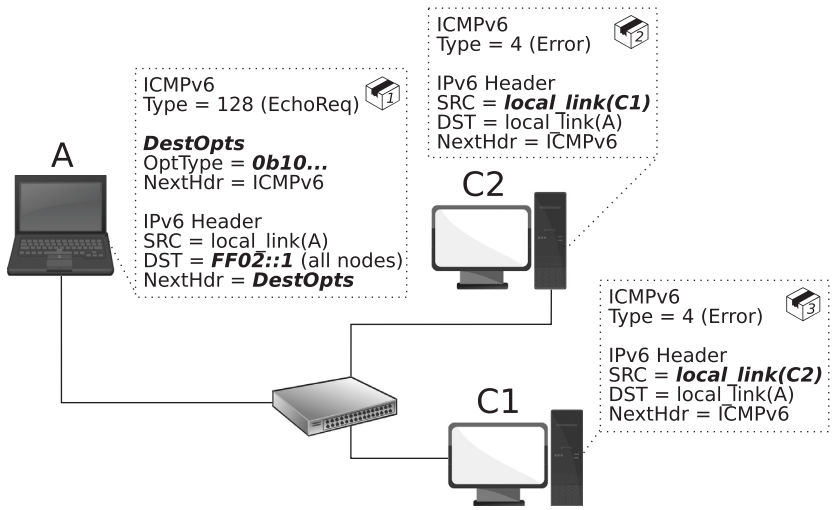


Figure 3: IPv6 multicast alive scan.

networks to assign a DNS server to all local hosts that is under control of the attacker. Figure 4 depicts the procedure for the IPv6 variant. First, the attacker sends a fake router advertisement with its own source link layer address (slla) and a randomly selected local network prefix (unique local address–ULA) which prompts the clients to start autoconfiguration ($A = 1$) with the option to obtain other parameters ($O = 1$) via DHCP (Steps 1–3). The receiving client selects an address that contains the ULA prefix and its own MAC address or a random 64-bit postfix (Step 4). The client validates the uniqueness of the selected address via IPv6 neighbor solicitation and sends a DHCPv6 solicit request to obtain other parameters, in particular DNS servers (Step 5). In response, the attacker provides its own IPv6 address via DHCPv6 as the DNS server (Step 6). The DHCPv6 part is similar to that of the DHCPv4 attack, but there are detailed differences in the following step. Since the attacker is usually the only IPv6 router on the network, he/she can respond to DNS queries with any IPv6 address (Step 7–8). The subsequent IPv6 traffic is then routed through the attacking computer.

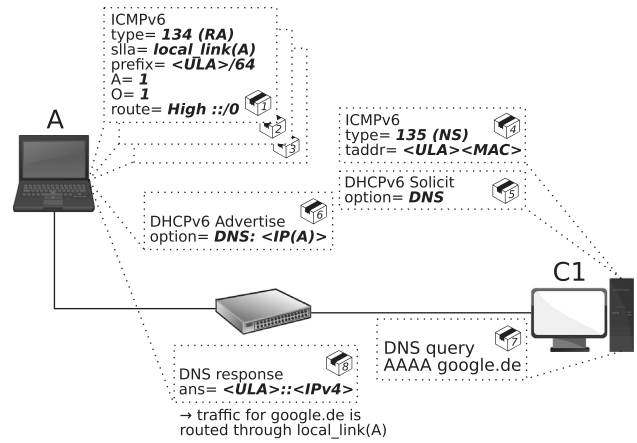


Figure 4: ICMPv6 router advertisement spoofing.

## 3. Approaches to Ward Off LAN Attacks

There are various approaches to protect physical networks, at least partially, against these attacks. In this section, we present some of these approaches.

In order to secure classical physical systems, the networks are often divided into smaller segments. This can be done on a logical level by configuring IP subnets, but this is only a very weak division or physically by an appropriate Ethernet wiring of the systems. At the transition points, e.g., routers or switches, data can then be analyzed by packet filters. Packet filters are the basis of traditional network- and host-based security measures, but they are inconvenient to manage in large network infrastructures—a weakness that distributed firewalls [15] do not have due to a centralized policy design and distributed policy enforcement. Distributed firewalls demand, however, implementations for each operating system and are themselves an attractive attack target.

Virtual LANs (VLANs) represent another possible security measure. They can be used for separating clients, so that only the systems that are associated with the same VLAN can communicate with each other. However, there are also attacks that aim at the so-called VLAN hopping (http://www.ciscopress.com/articles/article.asp?p=1681033&seqNum=3). Moreover, the use of VLANs is quite inflexible because a client is either assigned to a particular VLAN or to none at all.

Another security measure is the use of intelligent switches that provide some protection against rogue DHCP servers and ARP attacks. Smarter L3 switches use, for instance, DHCP snooping against DHCP attacks to enforce a fixed mapping between IP, MAC, and switch addresses/ports. ARP spoofing is prevented in this context by discarding nonapproved source addresses. On some switches, this security measure can be circumvented by another attack which uses the spanning tree protocol (STP) to redirect traffic to an attacker. Such attacks can be avoided

by limiting STP to ports which are explicitly used for switch coupling. Currently, no security measures are known that reliably protect against similar IPv6-based attacks. In addition, there are problems with port-stealing attacks that override the internal cache of the switches with fake MAC/port pairs.

With the advent of software-defined networking (SDN), projects like SANE [16] and Ethane [17] tried to concentrate complex functions, such as routing, naming, and firewall policing, in a central controller. The main focus of these projects was on access control and enforcement of communication relations rather than on preventing layer 2/3 attacks. In particular, the Ethane approach [17] for local area networks focused on improving security. It couples flow-based Ethernet switches with a centralized controller that knows the global network topology and grants access by explicitly enabling permitted flows within the network switches along a centrally computed route. The controller enforces a strong binding between a packet and its source by restricting the port access to a switch on the IP addresses assigned via DHCP. One of the biggest problems that were reported in this implementation is the handling of broadcast traffic. Most broadcast traffic is caused by address resolution protocols, e.g., ARP, which generate a huge load on the controller. Other address resolutions, such as the IPv6 neighbor discovery, were apparently not implemented resulting in further shortcomings with regard to the spoofing of Ethernet/IP addresses. Accordingly, there is still a need for research on a method that simultaneously limits broadcast traffic and implements the address assignment and resolution in a secure manner for all major protocols (IPv4 and IPv6 including auxiliary protocols).

Beside these approaches for attack prevention, some approaches for anomaly detection with SDN capabilities were proposed. Mehdi et al. suggest several anomaly detection algorithms [18] that were implemented on the NOX controller (https://github.com/noxrepo/nox). Zhang published an adaptive flow counting method to detect additional anomalies in SDN-based networks [19]. The projects FRESCO [20] and OrchSec [21] provide beyond anomaly detection additional signature-based analysis methods that can detect ARP cache poisoning, DDoS attacks, and DNS amplification attacks. Due to the central management of the controller, port-stealing attacks cannot run in a software-defined network. Methods that are able to cope with IPv6-based attacks have not been reported, yet.

## 4. An SDN-Based Approach to Protect Switched LANs

We now present an approach to protect switched LANs against attacks at layers 2 and 3 using the software-defined networking (SDN) paradigm. It does not require any changes in the stations/host systems.

Software-defined networking provides the ability to separate the control and the data plane in a switch. As a result, the switch logic can be outsourced to a separate controller with the aim to control network flows from this centralized control application, running on a server or virtual machine. Decisions regarding packet forwarding are no longer be made autonomously in switches or routers but are passed to the central controller. In a certain way, they become "slaves" of this controller. The controller can contain various applications, among them dedicated security services, which define the rules how to handle and route network traffic, data packets, and frames in the network. Admins can readily write/rewrite them. Thus, SDN-enabled switches can support user requirements for a wide range of applications (service level agreements, quality of service management, policy enforcements, etc.). A further key advantage is the ability to define routing choices at a much finer granularity level, i.e., per application flow, than at the usual IP level.

Since SDN is not limited to network devices of a certain vendor, it can be applied to devices from various vendors if the same protocol is used. Most SDN solutions rely on the widely-used OpenFlow (OF) protocol (for current specification, see https://www.opennetworking.org) [22] for the communication between the controller and the switches. OpenFlow is a vendor-independent standard that allows for interoperability between heterogeneous devices. Version 1.5 of the protocol supports 44 different types of header fields to match a packet against, to choose the flow it belongs to, and, thus, to determine the route it should follow. The protocol is also applied here.

*4.1. SDN-Based Secure Switching.* In order to prevent the above-introduced attacks on switched LANs, the interactions (or a part thereof) of the mentioned protocols (ARP, DHCP, etc.) have to be forwarded to a security unit that proves the meaningfulness and correctness of the interactions. Such a redirection can easily and efficiently be implemented by the software-defined networking paradigm using OpenFlow as the capture protocol, for instance. Figure 5 shows the principle of the approach. It uses an OpenFlow-enabled switch as the data forwarding component. The data plane switch is connected via the OpenFlow protocol with the control plane, in which various applications to control the data flows may be installed, and among them security services enforce a correct protocol behavior. By shifting the switch logic into this separate controller, packet forwarding decisions are made in a policy-based software switch inside this controller and not in the data plane switch. Thus, the software switch gains complete control over the network and the data routing.

One of the big problems in local area networks is that address configuration and resolution protocols, such as ARP and DHCP, broadcast packets on the network, and thus, each station can listen in and also respond. Therefore, it makes sense to proactively set rules in the OpenFlow switches that ensure that all ARP and DHCP packets are routed to the controller and no longer distributed. To be able to answer legitimate inquiries, an appropriate ARP and DHCP server should be integrated as application into the OF-controller. Most of the vulnerabilities can be resolved with these simple measures alone. ARP spoofing and poisoning are prevented by no longer broadcasting (malicious)

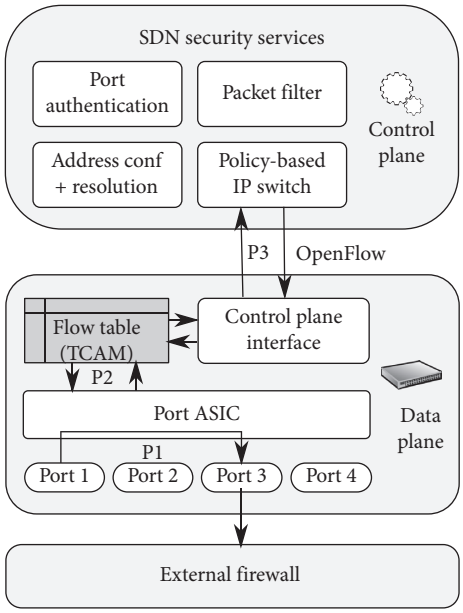


Figure 5: Openflow-based secure switching.

ARP packets in the network, but instead being answered or rejected by the OF-controller only. The same applies to DHCP packets to block rogue DHCP servers. An attacker does not even see the DHCP requests of other systems necessary to send a malicious reply at the right moment.

This approach also allows us to address the target systems with IP instead of MAC addresses. Many of the mentioned vulnerabilities are based on the fact that two different levels are responsible for addressing a system (layers 2 and 3), which must be adjusted. For example, ARP is used to obtain the MAC address associated with an IP address. At this point, an attacker can begin to bring the system into an inconsistent state and to intercept or block connections. To avoid this, the ARP server should be implemented in the OF-controller, so that it can answer all the ARP requests of the clients with a virtual, nonexistent MAC address. When sending Ethernet frames, clients use these MAC addresses together with the IP addresses of the target computer. The OF switches forward this information to the controller which uses the IP address to determine a suitable route and to create corresponding match entries in the switches. The last switch on this route is given the task of rewriting the MAC addresses in such a way that they are valid for the target system. Due to the use of virtual MAC addresses, each client knows only its own MAC and the IP addresses of the other systems. The MAC addresses of the other systems and the actual network topology remains hidden from them. Thus, each system knows only the information which is essential for a successful communication. The corresponding concept is referred to as IP switching. In Section 5, we describe such a security service more in detail.

*4.2. Authentication at Switch Ports.* Port authentication is an important problem of this approach for uniquely identifying hosts that are accessing the switch and thus the network. We implemented it as a service in the OF-controller using the extensible authentication protocol (EAP) [23], which is often applied in wireless networks and point-to-point communications. The existing authentication infrastructure of most corporate networks in the form of RADIUS servers and LDAP [24] directory services can directly be used for a compatible SDN-based authentication service based on EAP. The EAP standard is a port-based access restriction for switch ports that only unlocks after a successful authentication. The switch acts as an intermediary between the host (supplicant) and the authentication server—usually a RADIUS server—that optionally queries a directory service (Figure 6). The problem to be solved is the integration of two different authentication formats in a SDN-based network. Although all parties, i.e., the hosts, the switch, and the RADIUS server, communicate with each other via EAP, the EAP packets are encapsulated differently, e.g., in Ethernet frames (EAP over LAN (EAPoL)) or UDP packets (RADIUS over UDP). The solution is that only the EAPoL packets are allowed at a nonauthenticated switch port (ether type = 0x888E, address = 01 : 80 : C2 : 00 : 00 : 03 multicast). These packets are identified by the OF switch and redirected to the OF-controller. The OF-controller sends the EAP messages as RADIUS packets to an authentication server. If authentication succeeds, the controller transfers appropriate rules to the OF switch allowing the host to participate in the communication. Hostapd (http://w1.fi/hostapd/) is used as the authentication server. It is a daemon running in the user mode with software WLAN access point functionality. In addition, it provides an EAP authenticator as well as a RADIUS client and server. The EAP standard supports various authentication methods. For a prototype of this approach, the EAP-MD5 protocol was selected which does not use a protected communication channel. In practical use, more secure alternatives, such as EAP-TLS, EAP-TTLS, or EAP-PEAP, should be deployed. In the SDN-controlled network, however, the broadcast traffic is blocked and the communication partners communicate in a circuit-switched manner. Clients are unable to analyze or modify packets that are not addressed to them (even in the promiscuous mode). Thus, EAP vulnerabilities that primarily relate to networks that use a shared medium, such as the IEEE 802.11 wireless networks or the classic Ethernet, cannot be exploited.

*4.3. Further Services.* In addition to these security services, further security functionality, e.g., packet filters, deep packet inspection, or application-level gateways, can be integrated into the controller. Here, however, the following problem occurs. The OpenFlow data plane works internally flow- and not packet-oriented, i.e., each incoming packet for which there is no flow rule in the flow tables of the data plane switch is redirected to the OF-controller (path P3 in Figure 5) which causes a high overhead. Only after the controller has stored an appropriate flow rule in the flow tables of the affected OpenFlow-enabled switch, subsequent packets can be forwarded to the respective ports (see paths P2 and P1 in Figure 5). For a packet filter, the high overhead for the first packet of a flow is not critical. A deep packet inspection, in

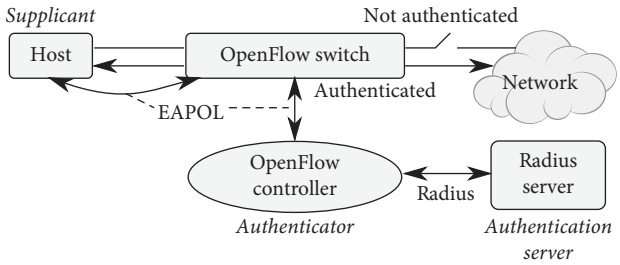


Figure 6: EAP/RADIUS authentication using SDN/Openflow.

contrast, must examine each packet or a sequence of consecutive packets of a flow. To avoid the expensive sending of packets to the controller, a specific rule can be stored in the flow table of the data plane switch that directs security-critical packets to a separate external analysis unit.

## 5. SDN-Based Security Services

The attacks presented in Section 2 are basically caused by the decentralized management of network services in LANs. In the following, we outline several security services that can be installed in the centralized OF-controller and describe their implementation by indicating the respective flow rules.

*5.1. Spoofing and Scan Resistance.* As argued above, many of the LAN security issues result from the fact that two different addressing schemes are used in layers 2 and 3 that have to be reconciled. To obtain the corresponding MAC address of a system with an IP (v4/v6) address, ARP and ICMPv6 Neighbor Discovery (ND) are deployed, respectively. One of the major problems is that address configuration and address resolution frames are broadcasted within the networks. These frames can be monitored from any host and responded. This enables an attacker to bring the system into an inconsistent state, to eavesdrop on running network connections, or to suppress further communication. The following address configuration and address resolution schemes implement the necessary functionality to prevent various aspects of these attacks.

*5.1.1. IPv4 Address Configuration Service.* Countermeasures against spoofing can mainly be avoided by means of address configuration services. The IPv4 address configuration service proposed here is based on a controller-internal DHCPv4 service that redirects all DHCP client packets according to the following flow rule to the OF-controller (see rule DHCP4c in Table 1).

The contents of the rule can be interpreted as shown in Figure 7. The rule header corresponds to the matching and action data structures defined in the OpenFlow standard that are initially transferred to the controller (lower left part of the figure). The prefixes OFPXMT_OFB and OFP_ACTION have been omitted in the rule; the suffixes *eth_type, ip_proto, udp_src*, and *udp_dst* correspond to the grey-marked fields of the protocol stack shown in the upper part of the figure. The internal representation of the matching data structures

Table 1: IPv4 address resolution service rules.

| Rule | eth_type | ip_proto | udp_src | udp_dst | out_port |
|---|---|---|---|---|---|
| DHCP4c | 0x0800 (IPv4) | 17 (UDP) | 68 (client) | 67 (server) | Controller |
| ARP | 0x0806 (ARP) | | | | Controller |

in the flow tables is not defined in the standard. In physical switches, it is typically converted into a sort of bit mask for a TCAM (Ternary Content Addressable Memory) (see lower middle part of the figure). If a packet matches this mask, it is sent to the specified *out_port* defined in the rule. The internal DHCPv4 service of the controller assigns IP addresses or renews leases for the connected systems. Allocated address entries are kept in the configuration of the controller for the address resolution services which are discussed below. This measure prevents attacks from rogue DHCPv4 servers because a spoofing is not possible without knowledge of the 32 bit DHCP transaction ID and the exact time of the client requests. An attacker does not even see the DHCP requests of the other systems required to send the malignant answer in the right moment.

*5.1.2. IPv4 Address Resolution Service.* Countermeasures against network scans and further antispoofing measures can be implemented in the IPv4 address resolution service which is based on the knowledge of the DHCPv4 address configuration service. The ARP service part installs rules in all switches that redirect Ethernet frames with the *eth_type* 0x0806 (ARP) to the controller (see rule ARP in Table 1). Thus, no ARP frames are forwarded in the switches. The OF-controller sends accordingly ARP responses based on the knowledge of the DHCPv4 service. In addition, ARP requests that are not related to the gateway address in the appropriate subnet of the requesting client are ignored. These measures effectively prevent attacks, such as ARP scans, ARP spoofing, and ARP flooding.

*5.1.3. IPv6 Address Configuration Services.* IPv6 address configuration is provided by an internal ICMPv6 router advertisement (RA) service and a DHCPv6 service linked to the internal DHCPv4 server. The former enforces configuration of IPv6 addresses using DHCPv6. This is done through regular flooding on all switch ports using router advertisements with the managed address configuration flag set. Initially, the DHCPv6 service works similar to the DHCPv4 service by redirecting DHCPv6 messages to the controller (Table 2). DHCPv6 requests are answered exclusively using the IPv4-mapped IPv6 address of the requesting client based on the knowledge from the DHCPv4 service [25]. The IPv4 address 192.168.120.63, for instance, is mapped to IPv6 address : FFFF : 192.168.120.63. DHCPv6 spoofing is therefore even more limited as in DHCPv4 because in addition to the lack of the transaction ID and request time, there is also no possibility to exhaust the address pool. Another advantage is that the communication can be processed by existing firewall logic derived from IPv4

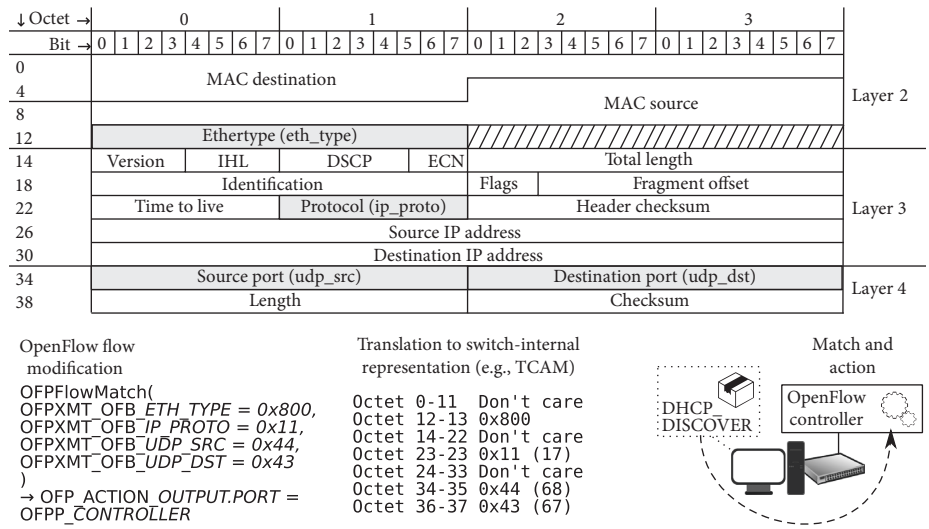


Figure 7: Matching packet header fields for rule DHCP4c of Table 1.

Table 2: IPv6 address configuration rule.

| Rule | eth_type | ip_proto | udp_src | udp_dst | out_port |
|---|---|---|---|---|---|
| DHCP6c | 0x8DD (IPv6) | 17 (UDP) | 546 (client) | 547 (server) | Controller |

firewall rules. The disadvantage of nonroutable IP addresses associated with the mapping may be compensated by a very simple form of NAT which applies a direct (IPv6) address to (IPv6) address translation (static one-to-one address assignment without protocol/port translation) at the edge of the network.

*5.1.4. IPv6 Address Resolution Service.* ICMPv6 neighbor discovery is based on the same idea as ARP for IPv4. It installs two flow rules on all switches to redirect ICMPv6 neighbor discovery solicitations and advertisements to the controller (Table 3). All solicitation requests except those to the gateway's IPv4-mapped address in the subnet of the requesting client are ignored. This prevents standard IPv6 scans as discussed in Section 2.

Table 3: IPv6 address resolution rules.

| Rule | eth_type | ip_proto | icmpv6_type | out_port |
|---|---|---|---|---|
| ICMPv6nds | 0x8DD (IPv6) | 58 (ICMPv6) | 135 (solicit) | Controller |
| ICMPv6nda | 0x8DD (IPv6) | 58 (ICMPv6) | 136 (advertise) | Controller |

*5.2. IP Switching with Topology Hiding and Routing Enforcement.* The security services introduced above provide a secure initialization of the network configuration but no secure message routing. They prevent that data connections are kidnapped by means of ARP spoofing or corresponding IPv6 attacks, but packets can still be hijacked if a false identity (MAC address) is adopted by a host and only a simple MAC-learning switch is applied in connection with the OF-controller. In addition, broadcasts like in ARP are a major problem as already mentioned in the Ethane approach [17] because they flood the controller too much. Moreover, broadcasts and ARP can be used to explore the network topology at the link. Therefore, a service is required that (1) binds the identity of a system to a switch port, (2) limits the number of possible requests for other systems, and (3) implements the route of the data connection between two systems in a secure way.

Historically, broadcasts have always been limited by separation of networks. The binding of a system to a switch port is also a special case of network separation in some way. Therefore, objectives (1) and (2) can be achieved by network separation with address configuration. For this purpose, the address configuration service assigns a private/30 subnet to each host using DHCPv4. This addressing scheme allows one to hold four IP addresses for each subnet. The lowest address is the network address. The next two addresses are assigned to the host and to a virtual gateway that references the OF-controller. The upper address remains for further use, e.g., as the IPv4 broadcast address. The hosts in the subnets are forced by this addressing scheme to use routes via the virtual gateway instead of direct communication to reach the target systems. The only possible broadcast request—an ARP request for the virtual gateway address—is intercepted by the OF-controller and not flooded.

Figure 8 exemplifies this address configuration. The address resolution service in the controller assigns host C1 to the network X.Y.Z.0/30. The resulting address configuration (IP(C1) = X.Y.Z.1, GW(C1) = X.Y.Z.2) is deposited together with the host's MAC address in a database of the controller. The next available subnet (X.Y.Z.4/30) is assigned to host C2,

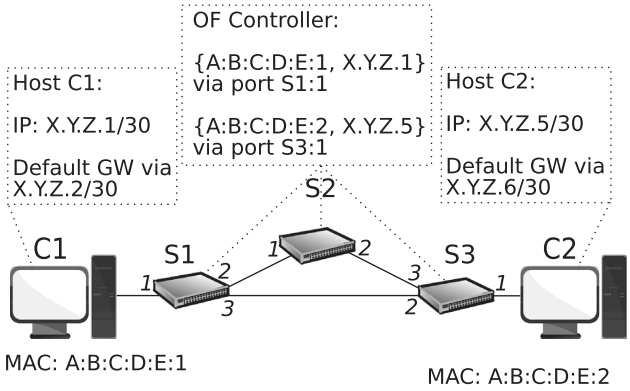


Figure 8: IP switch configuration with virtual gateway addresses.

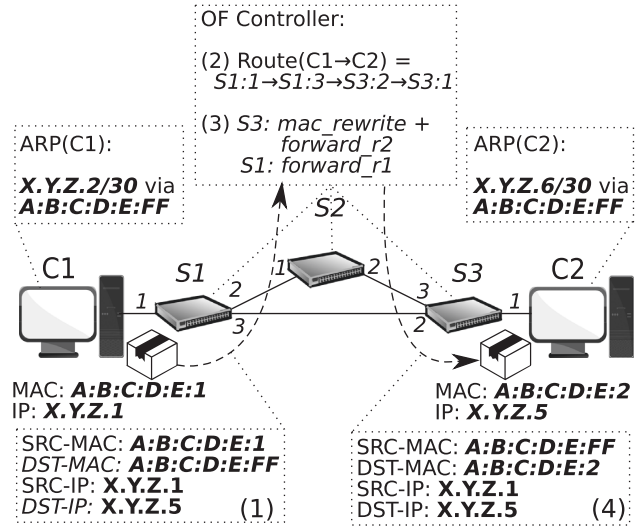


Figure 9: IP switching (a.k.a. routing) with virtual MAC addresses.

and its configuration is also stored in the database. The hosts know only their own IP address and the respective virtual gateway, thus limiting their ability to scan for other hosts with layer-2 protocols. Only the IP addresses are known of the other systems, and even the actual network topology remains hidden.

Based on the information of the (subnet) address configuration service, a secure routing service uses reactive flow instantiation to bind the identity of the participating systems for the duration of a connection to their source and destination switch ports to prevent data kidnapping. The routing process is exemplified in Figure 9 (based on the address configuration of Figure 8). The virtual gateway address is the only address that is resolved via ARP or ICMPv6-ND (MAC address A : B : C : D : E : FF in the ARP caches ARP(C1) and ARP(C2) of the figure). (1) When sending IP packets, the hosts use this address as the destination MAC address (DST-MAC) together with the IP address (DST-IP) of the target computer. The OpenFlow-enabled switches forward the first packet of a connection to the OF-controller. (2) The controller determines a suitable route based on the source and target IP addresses (switch 1, port 1 (S1 : 1) via switch 1, port 3 (S1 : 3), via switch 3, port 2 (S3 : 2) to switch 3, port 1 (S3 : 1)). (3) Appropriate match entries are created in the switches (Table 4, nw_src = SRC-IP, nw_dst = DST-IP).

The rules are installed in the reverse transport direction of the packet (first *mac_rewrite* + *forward_r*2 on switch S3, then *forward_r1* on switch S1) to avoid multiple redirects of the same packet by the subsequent switches to the controller. The last switch on the route is given the task to rewrite the MAC addresses (*set_field* part of the *mac_rewrite* + *forward_r*2 rule), so that they are valid for the target system. (4) The queued packet is released (by forwarding to switch 1, port 3–S1 : 3) and reaches the target in accordance with the defined rules. A glance at the chain created in this way in Table 4 illustrates the binding of each packet to its source and destination. In switch S1 the source IP address X.Y.Z.1 is bound by the rule *forward_r*1 for the exemplary data flow to input port (*in*) 1. The rule *mac_rewrite* + *forward_r*2 for switch S2 rewrites the virtual destination MAC address to the MAC address of host C2 and binds the data flow and the destination IP address to output (*out*) port 1. Thus, a spoofing of IP addresses is prevented.

*5.3. In-Network Firewalling and Robustness against Firewall Bypassing.* Physical switches often implement simple firewall logic in the form of access control lists–a concept that is also feasible with SDN. Firewalls can be implemented as a combination of SDN-based rules on the switches and controller-based firewall logic. Simple packet filtering can be implemented directly by respective rules on the switches. In addition, firewalls with stateful connection management can be implemented on the switches with some help of the controller and the dynamic creation of rules. IP switching is basically a flow-based switching with packet filtering functionality. For each new connection, a request is sent to the OF-controller which then decides whether this connection should be permitted or not. Then, a specific rule is stored in the switches ensuring that the other packets of this connection no longer needs to be treated by the controller. Thus, there is no additional overhead for the subsequent communication. All these security rules can individually be stored in the OF-controller for each client and are enforced independently of the switch port to which the client is connected to. Thus, each port on each SDN-enabled switch becomes a firewall.

Important security measures that should be contained in every firewall are static firewall rules which are directed against the circumvention of the firewall logic. The concept described above, for instance, allows a filtering on packet basis, but fundamental problems that affect pure packet-based firewall logic, e.g., the processing of fragmented oversized packets, as supported by IPv4 and IPv6, cannot be solved. Past experience has shown that it is not possible to implement packet fragmentation in a secure manner. For this reason, the presented approach installs rules on all switches to discard IPv4 and IPv6 packets with fragmentation options/headers (see rule fragment in Table 5). Another IPv6 issue to bypass the OF-controller is the

Table 4: Routing rules for Figure 9.

| Rule | In | eth_type | nw_src | nw_dst | set_field | Out |
|---|---|---|---|---|---|---|
| forward_r1 | 1 | 0x800 (IP) | X.Y.Z.1 | X.Y.Z.5 | | Port 3 |
| mac_rewrite + forward_r2 | 2 | 0x800 (IP) | X.Y.Z.1 | X.Y.Z.5 | eth_dst = A:B:C:D:E:2 | Port 1 |

problematic IPv6 extension headers. The following headers should be redirected through rules on the switches to the controller (to log attack attempts) or discarded: *hop-by-hop options, routing header, destination options*, and *mobility header* (the mobility header supports Mobile IPv6) (Table 5). Although these are almost all IPv6 extension headers, these rules are in line with recent observations in transit networks (https://tools.ietf.org/html/draft-ietf-v6ops-ipv6-ehs-in-real-world-02) which discard packets with the same headers. They represent special use cases that are often not supported or configured but are, however, used within the IPv6 attack suites as an evasion method for IPv6 security measures. Another packet type that should be blocked is ICMPv6 redirects (Table 5). This packet type allows another attack in which a client can be redirected to a fake gateway address.

*5.4. Evaluation.* As part of the research for the described concept, an OF-controller that realizes the described functionality was implemented with the help of the Ryu framework (http://osrg.github.io/ryu/). During the development phase, the controller was tested extensively with Mininet [26]. The proposed defensive measures against ARP scans, ARP spoofing, port-stealing, and DHCPv4 spoofing with DNS spoofing worked as expected. There were, however, mixed results for IPv6. Scans of the IPv6 test network, neighbor discovery spoofing, router advertisement spoofing, and DHCPv6 with DNS spoofing were blocked successfully. The defensive capabilities against malformed IPv6 packets, however, depend on the used IPv6 stack, e.g., the Linux IPv6 stack discussed below.

In order to test the defenses against malformed IPv6 packets, two virtual clients were evaluated with the penetration test suite from the THC IPv6 toolkit (https://github.com/vanhauser-thc/thc-ipv6). One of the clients was acting as a test server to check which packets have passed through the firewall, and the other client was used to send manipulated packets. The test suite consisted of 56 test cases that were first executed without the SDN-enabled switch to test the Linux IPv6 stack and then with it including the OF-controller prototype with all services presented above. The purpose of this comparison was to evaluate the additional benefit of the SDN-based architecture compared to the default stack with decentralized packet management. The test purpose of the penetration test suite was to prove whether the firewall can be bypassed (pass) or whether the packets are blocked (fail). 38 tests failed (were successfully blocked) with the standard Linux stack and also with the Linux-based SDN-enabled switch. Among the tests that failed in both approaches were also those with overlapping fragment headers. The structure of the test suite is interesting in itself. Approximately half of the circumvention test cases started with a single extension header type and increased the number of types up to three. As expected, all standard test cases with *hop-by-hop, destination options*, and *source routing options* passed the Linux stack (i.e., they were forwarded without complaints) and failed with the controller prototype (i.e., they were successfully blocked). However, there were test cases with multiple destination option headers in a packet. They passed the firewall policy of the controller for unknown reasons. Three more test cases passed through the firewall because they were not covered by a policy and were also not efficient to implement: ICMPv6 echo requests (ping6) with bad checksum, zero checksum, and with hop count 0. All other test cases failed in the controller. A direct neighbor solicitation test was successful on the standard stack but failed in the prototype because of the defenses against network scans.

Table 5: Static rules against firewall bypass.

| Rule | eth_ Type | ipv6_ exthdr | ip_ Proto | icmpv6_ Type | Output |
|---|---|---|---|---|---|
| Fragment | 0x8DD | 44 | | | Controller |
| hop_by_hop | 0x8DD | 0 | | | Controller |
| Routing | 0x8DD | 43 | | | Controller |
| Destination | 0x8DD | 60 | | | Controller |
| Mobility | 0x8DD | 135 | | | Controller |
| icmpv6_redirect | 0x8DD | | 58 | 137 | Controller |

To sum up, the default stack successfully blocked 38 attacks (67.86% of the malicious traffic), while the SDN-based approach blocked 53 attacks (94.64%). Thus, the SDN security architecture blocks about 39% more attack cases. The remaining 3 cases still require further research. One probable reason is that the SDN-enabled switch (Open vSwitch) does not correctly analyze nested IPv6 extension headers because these extensions should be blocked by our static rule set.

## 6. Conclusions

In this paper, we have considered security measures against attacks on switched LANs in the context of IPv4 and IPv6 networks which are often deployed to circumvent monitoring measures in domain networks. A significant similarity of these attacks is the spreading of the malicious code in the internal networks and the extraction of data from compromised subnets and hosts. As part of a potential mitigation strategy against these attacks, software-defined networking (SDN) has been proposed as a vehicle for centralizing information about all network activities in a central authority—the SDN controller—that manages all network connections and hence the associated data flows. The SDN technology allows us to provide networks with security services that perform basic tasks, such as address

configuration, address resolution, and firewalling within the network, much more efficient than on scattered individual systems. The resulting secure networks are based on switched routing which uses auxiliary information from the address configuration services to enforce a strong binding between a packet and its origin as well as its target that disables these attacks. Our approach can also be integrated into existing policy-based security frameworks, such as OpenSec [27].

We are currently extending the approach to secure the data exchange between virtual machines that represent blind spots in network monitoring. Conventional firewall systems cannot protect virtual machines because communication between virtual machines runs only within the virtualization server/host. The network interfaces to wireless local area networks (WLAN) represent an additional blind spot for network monitoring [28]. Often, the WLAN access points are not part of the monitoring and security infrastructure but represent only the last hop to the devices. As a consequence, sensitive information of the communication between access point and device gets lost. A software-defined networking approach, in which the access point passes authentication and monitoring information to the SDN-based security services to analyze incoming data streams, seems more reasonable [29].

## Data Availability

The source code of the SDN controller that has been implemented to evaluate the described concept is available from the authors upon request.

## Disclosure

This paper is based on a thesis submitted by the first author in partial fulfillment of the degree of Doctor of Engineering to the faculty of Mathematics, Computer Science, Physics, Electrical Engineering, and Information Technology of the Brandenburg University of Technology and defended on November 17, 2017 [30].

## Conflicts of Interest

The authors declare that they have no conflicts of interest.

## Acknowledgments

This research was supported by the Bundesministerium für Bildung und Forschung (Grant 01BY1204A).

## References

[1] B. Claise and B. Trammell, *Information Model for IP Flow Information Export (IPFIX)*, RFC 7012 (Proposed Standard), 2013, http://www.ietf.org/rfc/rfc7012.txt.

[2] R. Symantec, "Top-tier espionage tool enables stealthy surveillance," Technical report, August 2015, http://securityresponse.symantec.com/content/en/us/enterprise/media/security_response/whitepapers/regin-analysis.pdf.

[3] M. Casado, *Architectural Support for Security Management in Enterprise Networks*, Ph.D. thesis, Stanford, CA, USA, 2007.

[4] D. Kreutz, F. M. V. Ramos, P. E. Verssimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig, "Software-defined networking: a comprehensive survey," *Proceedings of the IEEE*, vol. 103, no. 1, pp. 14–76, 2015.

[5] J. Xie, D. Guo, Z. Hu, T. Qu, and P. Lv, "Control plane of software defined networks: a survey," *Computer Communications*, vol. 67, pp. 1–10, 2015.

[6] B. A. A. Nunes, M. Mendonca, X. N. Nguyen, K. Obraczka, and T. Turletti, "A survey of software-defined networking: past, present, and future of programmable networks," *IEEE Communications Surveys Tutorials*, vol. 16, no. 3, pp. 1617–1634, 2014.

[7] S. Scott-Hayward, S. Natarajan, and S. Sezer, "A survey of security in software defined networks," *IEEE Communications Surveys and Tutorials*, vol. 18, no. 1, pp. 623–654, 2016.

[8] I. Alsmadi and D. Xu, "Security of software defined networks," *Computers and Security*, vol. 53, pp. 79–108, 2015.

[9] M. C. Dacier, H. König, R. Cwalinski, F. Kargl, and S. Dietrich, "Security challenges and opportunities of software-defined networking," *IEEE Security and Privacy*, vol. 15, no. 2, pp. 96–100, 2017.

[10] T. Kiravuo, M. Sarela, and J. Manner, "A survey of ethernet LAN security," *IEEE Communications Surveys Tutorials*, vol. 15, no. 3, pp. 1477–1491, 2013.

[11] N. Hubballi, S. Biswas, S. Roopa, R. Ratti, and S. Nandi, "LAN attack detection using discrete event systems," *ISA Transactions*, vol. 50, no. 1, pp. 119–130, 2011.

[12] H. Mukhtar, K. Salah, and Y. Iraqi, "Mitigation of DHCP starvation attack," *Computers and Electrical Engineering Special Issue on Recent Advances in Security and Privacy in Distributed Communications and Image processing*, vol. 38, no. 5, pp. 1115–1128, 2012.

[13] R. Vida and L. Costa, *Multicast Listener Discovery Version 2 (MLDv2) for IPv6*, RFC 3810 (Proposed Standard), updated by RFC 4604, 2004, http://www.ietf.org/rfc/rfc3810.txt.

[14] T. Chown and S. Venaas, *Rogue IPv6 Router Advertisement Problem Statement*, RFC 6104 (Informational), 2011, http://www.ietf.org/rfc/rfc6104.txt.

[15] S. Ioannidis, A. D. Keromytis, S. M. Bellovin, and J. M. Smith, "Implementing a distributed firewall," in *Proceedings of 7th ACM Conference on Computer and Communications Security CCS 2000*, pp. 190–199, Athens, Greece, November 2000.

[16] M. Casado, T. Garfinkel, A. Akella et al., "A protection architecture for enterprise networks," in *Proceedings of 15th USENIX Security Symposium*, Vancouver, BC, Canada, July-August 2006, https://www.usenix.org/conference/15th-usenix-security-symposium/sane-protection-architecture-enterprise-networks.

[17] M. Casado, M. J. Freedman, J. Pettit, J. Luo, N. McKeown, and S. Shenker, "Ethane: taking control of the enterprise," in *Proceedings of ACM SIGCOMM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, pp. 1–12, Kyoto, Japan, August 2007.

[18] S. A. Mehdi, J. Khalid, and S. A. Khayam, "Revisiting traffic anomaly detection using software defined networking," in *Proceedings of Recent Advances in Intrusion Detection-14th International Symposium*, pp. 161–180, RAID 2011, Menlo Park, CA, USA, September 2011.

[19] Y. Zhang, "An adaptive flow counting method for anomaly detection in SDN," in *Proceedings of Conference on emerging Networking Experiments and Technologies*, pp. 25–30, CoNEXT'13, Sydney, Australia, June 2013.

[20] S. Shin, P. A. Porras, V. Yegneswaran, M. W. Fong, G. Gu, and M. Tyson, "FRESCO: modular composable security services

for software-defined networks," in *Proceedings of 20th Annual Network and Distributed System Security Symposium*, NDSS 2013, San Diego, CA, USA, February 2013.

[21] A. Zaalouk, R. Khondoker, R. Marx, and K. M. Bayarou, "OrchSec: an orchestrator-based architecture for enhancing network-security using Network Monitoring and SDN Control functions," in *Proceedings of 2014 IEEE Network Operations and Management Symposium*, pp. 1–9, NOMS 2014, Krakow, Poland, May 2014.

[22] N. McKeown, T. Anderson, H. Balakrishnan et al., "OpenFlow: enabling innovation in campus networks," *Computer Communication Review*, vol. 38, no. 2, pp. 69–74, 2008.

[23] B. Aboba, L. Blunk, J. Vollbrecht, J. Carlson, and H. Levkowetz, *Extensible Authentication Protocol (eap)*, RFC 3748, Proposed Standard, 2004, http://www.ietf.org/rfc/rfc3748.txt.

[24] K. Zeilenga, *Lightweight Directory Access Protocol (ldap): Technical Specification Road Map*, RFC 4510 Proposed Standard, 2006, http://www.ietf.org/rfc/rfc4510.txt.

[25] R. Hinden and S. Deering, "IP version 6 addressing architecture," RFC 4291 draft standard, updated by RFCs 5952, 6052, 7136, 7346, 7371, 2006, http://www.ietf.org/rfc/rfc4291.txt.

[26] B. Lantz, B. Heller, and N. McKeown, "A network in a laptop: rapid prototyping for software-defined networks," in *Proceedings of 9th ACM Workshop on Hot Topics in Networks*, p. 19, HotNets 2010, Monterey, CA, USA, October 2010.

[27] A. Lara and B. Ramamurthy, "Opensec: policy-based security using software-defined networking," *IEEE Transactions on Network and Service Management*, vol. 13, no. 1, pp. 30–42, 2016.

[28] H.-J. Liao, C.-H. R. Lin, Y.-C. Lin, and K.-Y. Tung, "Intrusion detection system: a comprehensive review," *Journal of Network and Computer Applications*, vol. 36, no. 1, pp. 16–24, 2013.

[29] R. Cwalinski and H. König, "RADIator—an approach for controllable wireless networks," in *Proceedings of IEEE NetSoft Conference and Workshops, NetSoft 2016*, pp. 260–268, Seoul, South Korea, June 2016.

[30] R. Rietz, "Optimization of network intrusion detection processes," Doctoral thesis, Faculty of Mathematics, Computer Science, Physics, Electrical Engineering and Information Technology of the Brandenburg University of Technology, 2018.