Integration of Network Services in Tactical Coalition SDN Networks

Anders Fongen and Mass Soldal Lund
Norwegian Defence University College, Cyber Defence Academy (FHS/CIS)
Lillehammer, Norway
Email: anders@fongen.no

***Abstract*—In order for Software Defined Network (SDN) technology to work in a military network, several identified problems need to be solved. This paper reports from experimental efforts to extend the application of SDN to a multi-domain, coalition, mobile network with wireless links and with end systems belonging to several Communities Of Interest (COI). The paper also demonstrates how SDN technology allows different network services to be integrated with a single class of Network Elements (NE). Considerations related to authentication, COI separation and intrusion prevention is given special attention during the discussions.**

Keywords*—*authentication; intrusion prevention; software defined networks; tactical networks; trust management

I. INTRODUCTION

Software Defined Networking (SDN) [1] offers an unprecedented flexibility in network configuration and operation. An important potential is how a spectrum of specialized Network Elements (NE), often called *middleboxes*, may be replaced with a single class of Network Elements called *switches*. The configuration and run-time operation of the NE is controlled by a single piece of programming logic running in a separate computer called the *SDN controller* (SDNC).

The SDN paradigm grew out of data center operations where links are abundant, have high capacity and low error rates. In a mobile and temporary military network used for military operations (named *tactical networks*), however, the links are often radio based. Radio links are few, costly, vulnerable, and with high error and packet loss rates. Consequently, the use of SDN in a tactical environment must consider the scarcity, latency and error rates of links in the system design [2].

SDN reduces the *complexity* of configuration, improves the *cooperation and integration* of network functions, extends the flexibility and dynamicity of traffic policing, and increases the link efficiency. The configuration of an SDN network involves fewer routine operations, but requires more software insight and programming skills.

The task at hand is to investigate the potential advantages offered by SDN in a tactical coalition network. The current problems related to this class of networks are identified as:

- They are based on Internet Protocol (IP) version 4 protocols, adding address planning, subnetting and frequent configuration changes even in small network enclaves.
- Virtual Local Area Network (VLAN) configuration in switches are weakly related to the IP layer, yet must be coordinated with the subnetting structure.
- Intrusion detection and protection is most often done in a single point, e.g., in the network backbone connection point. A compromised end system may have unrestricted access to services and end systems on the same network.
- Coalition partners wish to keep their traffic separate, but still need to coordinate their IP address plans, since separation takes place in link layer (VLAN) while sharing IP routes.
- Traffic policing becomes complicated since it requires coordinated use of IP Type Of Service (TOS) field (DiffServ) values across management domain borders.
- Authentication of end systems is based on MAC addresses, if used at all. Authentication on user level is done by application level services, e.g., MS Active Directory. No credential based scheme for authentication of *end systems* is in use.

The efforts presented in this paper address these listed problems and suggest an SDN-based configuration (based solely on SDNC software) which is purely a link-layer network. The network layer may be independently organized and the address plan does not need to be coordinated between Communities Of Interest (COIs). Any network layer protocol can be used (most likely to be IPv4 or IPv6).

The design has been prototyped in a virtualized environment and evaluated for functional correctness, performance and efficacy. Problems related to scalability are also being addressed.

The remainder of the paper is organized as follows: In Section II, a requirement analysis for a tactical SDN network will be discussed. The technology chosen for the experiment is presented in Section III and the actual network configuration is shown in Section IV. The new network functions added during this part of the study are discussed in Section V. The evaluation of the network functions is presented in Section VI, followed by a presentation of related research in Section VII. The paper concludes with a summary in Section VIII, where also topics on future research are presented.

II. DESIGN ANALYSIS

Strong coupling between the link layer and the network layer complicates their configuration. Where MAC-learning switches are being used, the connection between the two address structures is solved by the Address Resolution Protocol

 ISBN: 978-1-61208-821-1

(ARP) protocol. Where VLAN separation is used, the separation will normally reflect the IP subnet separation. During splitting or joining of subnets the VLAN configurations must be configured accordingly.

Many of these problems may be solved by configuring the network purely based on link layer mechanisms, over which any network layer structure can be built. Scalability problems related to multicast distribution can be alleviated through COI separation, besides that a tactical network enclave is not expected to grow to a large scale. Functions related to load balancing and traffic policing are not easily offered in link layer network, but may be provided through SDN flow mechanisms.

A. Broadcast free operation

A link layer structure may not contain cycles, since the forwarding of broadcast frames will cause endless loops. The *spanning tree protocol* (STP) may prune a cyclic structure into a spanning tree, leaving the redundant links available only for fail-over purposes, not for load balancing. The scarcity and capacity of radio based links in a tactical network renders this limitation to be unacceptable.

It is possible, however, to command SDN switches to forward multicast frames along the links of a spanning tree with root in the originating switch, rather than to every output port in the switch. For this to be possible the SDNC need a topology map of the link structure in the network, something that has been accomplished with a link discovery protocol.

Also, the broadcast operation during the MAC-learning process of a link layer switch can be avoided through the same topology map, through which the next hop in the path towards any other switch is known. The association between the MAC address of an end system and its connected switch port is known by the SDNC from the first frame transmitted by the end system.

Frames need an extra header to convey information about the originating switch (in multicast frames) or the destination switch (in unicast frames), in addition to COI membership information. Header extensions like MPLS and 802.1Q are both candidates, possibly a combination of both. The choice will be made based on the ability of OpenFlow to set, mask and test these data elements.

B. Whitelisted flows

An obvious application of SDN flow processing is to protect end systems. Switches can block or allow traffic based on a blacklist or a whitelist made of flow rules. A whitelist is the more aggressive protection, where an end system (client or server) is allowed to transmit/receive frames only if they are related to known protocols, identified by transport level port numbers. Restrictions on IP addresses may also be applied, e.g., to specific subnets. Every end system can have different whitelists since they match individual ports or MAC addresses. Flows rejected by the whitelist may be discarded, passed on to an Intrusion Detection System (IDS) or a Honeypot system. The efficacy of this mechanism has been investigated and will be reported later in the paper.

C. Authentication of end systems

End systems should be authenticated, in particular end systems which are temporarily connected through public access networks. This mechanism must provide a link layer tunnel over a network layer connection, and *bind the authenticated connection to the link layer tunnel*. The authenticated identity of the end system should be communicated to the SDNC which will install flows enforcing the permissions granted to this end system, e.g., in the form of a flow whitelist.

On end system platforms with sufficient separation of user spaces and storage areas, user credentials can be applied to the authentication process, so that the trust relation shifts from the end *system* to one end *user*.

III. TECHNOLOGY PLATFORM

In this section, the choice of technology components will be described. The components are all software, including operating system, hypervisor and system-level components.

The study of a medium sized networks with more than 10 nodes is best conducted in a virtualized environment. The hypervisor of choice is Oracle's VirtualBox, which is free, easily configured, and offers the right degree of scalability. The limit of four ports per VM was the most limiting factor during the experiments.

For the Network Elements, complete instances of Linux were chosen. The reason for this choice is that the experimental network is used for testing several services and protocols auxiliary to the OpenFlow protocol, and a general computing platform offers the necessary flexibility and software availability, contrary to Mininet [3]. The Linux instances do not need a GUI and were installed with a text-only console interface for the sake of saving memory.

The chosen OpenFlow switch (the NE) implementation is OpenVswitch [4], which is easily installed, relatively easy to configure, and offers the necessary inspection and logging mechanisms for testing and debugging purposes.

As the network controller (SDNC), the Ryu framework was used [5]. Ryu is very popular as an experimental platform with a relatively low abstraction level: OpenFlow statements are generally not automatically generated, but individually constructed through Python programming code. For the experimentation at hand, Ryu performs well and with good stability, although the API and the required design patterns takes some time to learn.

For all the chosen technology components, an important convenience point is the community support offered. Most problems are easily solved through these support resources.

IV. EXPERIMENTAL NETWORK

The network used in the experiment is shown in Figure 1. The network consists of a number of green switching nodes (NEs), a number of yellow and brown end systems and a number of server nodes for serving OpenVPN, Dynamic Host Configuration Protocol (DHCP), Domain Name Services (DNS), Hypertext Transfer Protocol (HTTP), Server Message Block (SMB), Network Address Translation (NAT), etc. End systems are separated in two COIs indicated by their

 ISBN: 978-1-61208-821-1

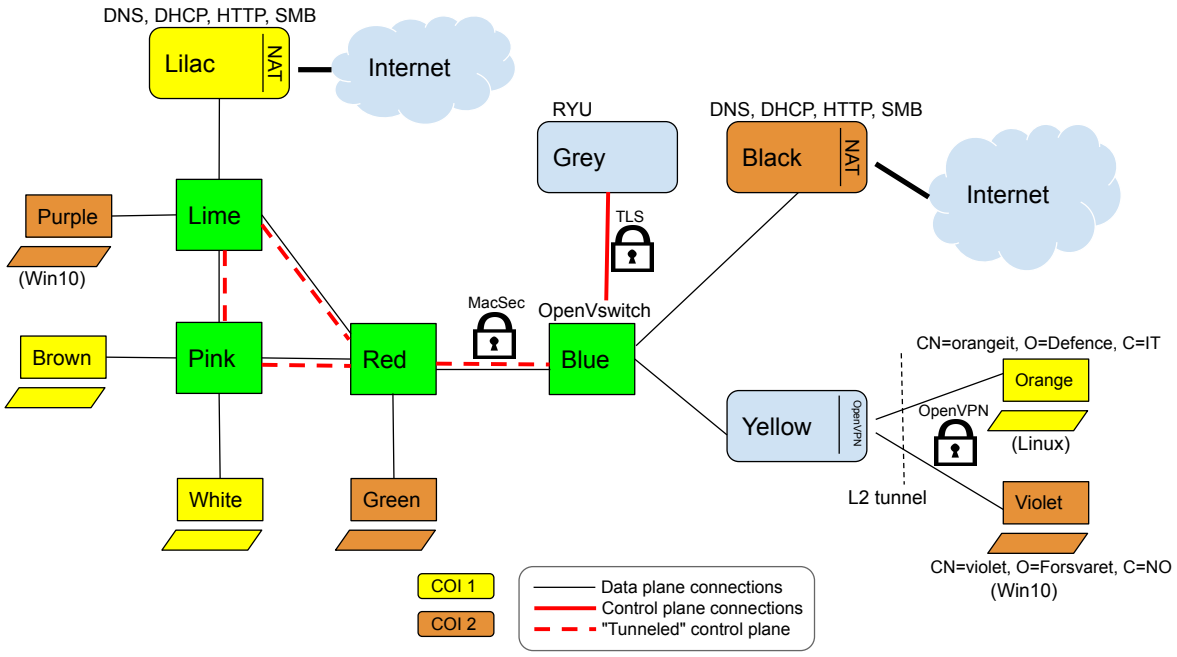

Figure 1: Current SDN laboratory configuration

brown/yellow color. The nodes are arbitrarily given names after colors, which should not be confused with the coloring codes of the diagram.

The links between the NEs Lime/Pink/Red are redundant and consequently form a loop. The redundant links are essential for the study of fail-over mechanisms and load balancing services [6]. The experimental network was used for investigating security mechanisms in an SDN based environment [7], for which reason the presence of OpenVPN, MACsec and Transport Layer Security (TLS) is indicated in the figure.

A. *Existing network functions*

From previous iterations of the SDN laboratory experiment, security functions and control plane redundancy has been investigated [6] [7]. These earlier efforts have shown:

- The links between NEs can be protected from a range of attacks using MACsec encryption. OpenFlow does not easily assist in the key management though, so static keys were installed during the NE configuration.
- NEs connect to the SDNC using TLS authentication and protection, using public key certificates and private keys for bidirectional authentication. Since the network uses in-band control plane a robust cryptographic separation between control plane and data plane was found to be mandatory. Certificate information is not made available to the Ryu application (nor the OpenVswitch code, for that matter) so the authentication control is restricted to the checking for certificate validity without revocation control.
- End systems connecting temporarily through an access network are authenticated by a Virtual Private Network (VPN) service before allowed access to the data plane. A Virtual Extensible LAN (VXLAN) tunnel through an IP Security (IPSec) connection was used in [7], but later replaced with a better solution based on OpenVPN.
- The control plane is constructed as an overlay network on top of the data plane, for more efficient use of the links available. The control plane was also constructed to automatically find alternative paths through the data plane if links were broken and NEs were isolated from the SDNC [6].

V. New network functions

Two new network functions have since been introduced and are subject to presentation in this paper: *COI separation* and *traffic whitelisting*.

A. *COI separation*

Coalition members do not trust each other completely, so their network traffic need to be robustly separated. Similar to VLAN functionality, both unicast and multicast frames should

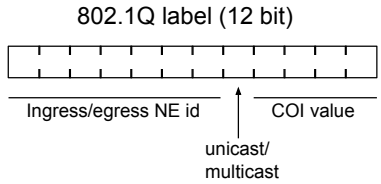

Figure 2: Encoding of NE id and COI value in 802.1Q label

only reach destinations which belong to the same *Community Of Interest* (COI) as the sender. Contrary to the well-known Ethernet switch, the presented solution does not need any direct configuration of NEs, the configuration is controlled by the SDNC software. The COI value of a frame is contained in the frame header, as described in Section V-B.

The assignment of a COI value to an end system can be based on its MAC address or its VPN authentication certificate. The latter alternative requires that the end system connects through a VPN server in a public access point and provides credentials in the form of a public key certificate. The VPN server will pass the certificate identifier to the SDNC which will decide the COI value accordingly. In both cases, the COI value of an end system is represented as a flow rule in the connected NE, the end system has no information about this value and is not able to modify it.

OpenVPN was chosen for the VPN service. OpenVPN offers a link layer tunnel as a core service, which also binds the MAC address of the tunnel adapter on the end system to the authenticated network layer connection. The end system will not be able to modify the MAC address in order to circumvent the access control. An IPSec connection can also contain a link layer tunnel (using, e.g., VXLAN), but no method to bind this tunnel to the authenticated IPSec connection was found.

For scalability reasons, multicast distribution may affect only the subset of NEs necessary to reach all end systems with the same COI value as the sender. The present implementation has a simpler implementation and distributes multicast frames to every NE.

B. Choice of link header extension

The link layer frame needs extra header information carrying its COI affiliation and the identifier of the egress (for unicast frames) or ingress (for multicast frames) NE. Since these information elements are independently processed, they need to be stored in one maskable element or two separate elements. MPLS, 802.1Q, MPLS-over-802.1Q or Q-in-Q are candidate structures for this purpose.

The chosen structure was to use one 802.1Q header for both elements. The 802.1Q label has 12 bits, which are divided into 7 bits NE designation, 1 bit for multi/unicast distinction, and 4 bits for COI value, as shown in Figure 2. The low number of bits limits the scale of the SDN network, but is sufficient for the experiment at hand.

Other type of candidate link headers were considered: The MPLS header contains more bits and could accommodate more COIs, but the MPLS header is not maskable in OpenFlow and therefore not useful for use with forwarding information (Cf. Section II-A). A combination of an outer MPLS header for forwarding information and and inner 802.1Q header for COI separation is not supported by OpenFlow. Two 802.1Q headers (called Q-in-Q or 802.1ad) is now supported by OpenVswitch, and may be considered for use in the future.

The COI relation of an end system is expressed as a numeric value 0-15 as 4 bits in the 802.1Q VLAN label of the Ethernet frame. The VLAN label value is added to the frame in the ingress NE after the whitelist control has been passed. In the egress NE, the COI value is again checked with the COI value of the MAC address associated with each port before passing the frame to the receiving end systems.

C. Traffic whitelisting

An SDN NE lends itself well to simple filtering of traffic based on flow matching, for reasons of end system protection. A client system need to connect to a set of server ports, possibly a small set of known IP addresses, in addition to services like DNS, DHCP and ARP. It should never receive a TCP segment with the flag ACK=0, since that indicates an inbound connection attempt. For a service provider end system, the opposite is the case, one would not see an outbound TCP segment with ACK=0, except to a small number of subordinate services. Through the chosen table structure (described in Section V-D) it is possible to pass both outbound and inbound frames through a set of flow rules which will submit the approved frames to the next flow table or output port, otherwise pass the frames on to an intrusion detection system (IDS) or to a Honeypot system, or to discard the frame. For the experimental evaluation, a realistic set of whitelist entries were made: ARP, DHCP (UDP/67,UDP/68), DNS (UDP/53, TCP/53), HTTP (TCP/80, TCP/443), SMB2 (TCP/445) and LLMNR (UDP/5355), which allows the client end-system to operate on the majority of web and file sharing services.

This simple arrangement does not inspect the application layer payload, and it does not aspire to replace an Intrusion Detection System (IDS). The main advantage is that the whitelist control takes place in every port connected to an end system, so it will also contribute to the internal protection in the LAN, whereas an IDS is usually seen as a single instance inspecting the traffic across a WAN connection point. Besides, an IDS do not *protect* systems, it only *detects* attacks.

The whitelist does not replace a firewall. A firewall will effectively protect an inside network (LAN) from attacks coming from outside (WAN), and stop any connection attempts to computers on the LAN, while allowing any outbound activity from end systems on the LAN. This is not the purpose of the whitelist, which will also block connection attempts to/from non-approved ports or to non-approved IP addresses.

Since a whitelist also blocks outgoing traffic and connections from server end systems, it also stops malware payloads (resulting from the exploitation of a vulnerability) from connecting back to an attacker, thus allowing for a security-in-depth arrangement.

The efficacy of the whitelist has been evaluated and will be reported in Section VI-B.

 ISBN: 978-1-61208-821-1

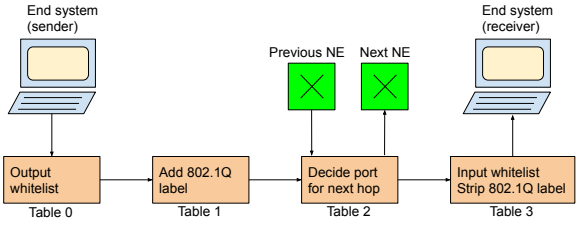

Figure 3: SDN flow table table structure in the NEs

D. Flow table structure

The flow arrangement in the NEs has been divided into several tables as shown on Figure 3, and will be presented in this section. Flows related to link and topology discovery, and path discovery in in-band control plane has been left out for reasons of clarity.

- Table 0 Contains outgoing whitelist which will inspect frames from end systems connected to this NE. Frames with an 802.1Q label (added by a previous NE in the path) will be submitted directly to Table 2.
- Table 1 Will assign an 802.1Q label with ID of the egress switch (ID of ingress switch in case of multicast frame) and COI value. Will then submit the frame to Table 2.
- Table 2 Will determine next hop for the frames based on the 802.1Q label value, and forward accordingly. If the frame should be distributed locally, then the frame wil be submitted to Table 3.
- Table 3 Strip off the 802.1Q label and apply the COI control and incoming whitelist control. If the frame passes both controls it is forwarded to the port connected to the receiving end system.

It should be pointed out that the whitelist arrangement breaks the desired isolation of the link layer since it introduces flows specific to network and transport protocols. On the other hand, the whitelist is structurally independent from the rest of the control program, and can be removed or modified at will without affecting other functionality. For the present experimentation only IPv4 packets will pass, but additional rules for IPv6 can be added with little efforts.

VI. Laboratory evaluation

Besides the testing for functional correctness two series of experiments were conducted to evaluate the presented arrangements:

- The cost of the extensive set of flow rules which has to be evaluated for every forwarded frame, in terms of NE throughput.
- The efficacy of the traffic whitelist protection.

A. Cost of flow processing

The OpenFlow protocol is designed with execution by specialized hardware in mind. *Ternary Content Addressable Memory* (TCAM) is a memory structure which is able to match byte strings in parallel in one clock cycle and therefore will not be penalized by complicated matching criteria. When OpenVswitch runs in normal computing hardware, the opposite is expected to be true. This section will report on simple throughput measurements on traffic passing 1, 2 or 3 NEs between the communicating end systems. Throughput between end systems separated by a VPN tunnel was also measured. The MACsec protection of the Blue-Red link was disabled on one run to measure its cost. References to nodes with color names are according to the colors used in Figure 1.

In order to establish a baseline for performance, throughput through the localhost adapter (row 1 in Table I) was measured as well as through the VirtualBox internal network (row 2). The OpenVswitch was tested in *standalone fail-mode* (row 3) where it behaves like a MAC-learning switch as well with a single flow rule to make it behave like a MAC-learning switch (*action:NORMAL*, row 4) or an Ethernet hub (*action:FLOOD*, row 5).

The `iperf` program was used to measure TCP throughput between the nodes Black (running `iperf` in server mode and a TCP receive window of 85.3 kBytes) and Green, White and Violet respectively. The node Green was temporarily connected to Blue in order to measure a connection passing only one NE (rows 3-6). The measurement involving Violet determines the performance of the VPN tunnel.

TABLE I: THROUGHPUT EVALUATION OF OPENVSWITCH

#	Client end system	NEs	Throughput
1	Black (localhost comm)	0	23 Gbps
2	Green (directly connected)	0	1116 Mbps
3	Green (connected to Blue in standalone mode)	1	853 Mbps
4	Green (connected to Blue with action:NORMAL)	1	811 Mbps
5	Green (connected to Blue with action:FLOOD)	1	250 Mbps
6	Green (connected to Blue, WL in effect)	1	835 Mbps
7	Green (connected to Red, WL with MACsec)	2	250 Mbps
8	Green (connected to Red, WL w/o MACsec)	2	561 Mbps
9	White (connected to Pink, WL with MACsec)	3	260 Mbps
10	Violet (though VPN)	1	42 Mbps

Using the VirtualBox hypervisor, the upper limit of the network throughput was estimated in row 2 (1116 Mbps). Furthermore, the simplest configuration of OpenVswitch, operating it as a MAC-learning switch yielded a throughput of 853 and 811 Mbps, respectively (rows 3 and 4). Operating it as an Ethernet hub gave a significantly lower performance (row 5), unsurprisingly since this mode involves a larger traffic volume to be processed. Row 6 reports the performance when Blue was operating with whitelist in effect (WL), which is only marginally lower than in standalone mode.

The traffic via Red and Blue (both with WL enabled) was tested both with MACsec protection turned on and off (rows 7 and 8), and the numbers (250 and 561 Mbps) indicate the high cost of MACsec protection. The traffic over three NEs (White to Black, row 9) is statistically equal to traffic across two NEs (row 7) when MACsec is enabled, while rows 6 and 8 indicate a significant drop in performance when extending the path from one to two NEs.

The lower number for traffic across more NEs may partly be due to the fact that all activities in the virtual network compete for the same pool of computing resources, and when more NEs are in action the each process gets a smaller fraction.

 ISBN: 978-1-61208-821-1

For the purpose of design evaluation these results are encouraging, since a more comprehensive set of flow rules does not seem to impose a significant performance penalty, by comparing rows 3 and 6.

B. Efficacy of whitelist protection

The whitelist is a simple protection mechanism with its scope limited to the stateless inspection of link-, network- and transport header elements. The chosen design is to associate a filter with a MAC address, so that several end systems may share the same switch port if needed (although the whitelist protection will not apply to traffic between these end nodes), and to list the approved UDP and TCP ports for this MAC address. The flow rules also inspect the ACK-flag in the TCP header to ensure that TCP connections are opened in the allowed direction.

By employing whitelist protection, vulnerabilities in end systems may only be exploited through approved ports, and payloads deployed through a successful exploit will meet the same restrictions. The list of approved ports is expected to reflect the services in actual use, so the ports are likely to be occupied by running services and not available for allocation by payload scripts. Delivered payloads which do not communicate are not restricted by the whitelist protection.

IP addresses may also be subject to restrictions, although this has not been demonstrated yet. Such restrictions can avoid fake DNS and DHCP services to be accepted by end systems.

Exploits that exclusively use approved ports are not expected to be stopped by the whitelist protection. SQL injection and other attacks on poorly written web service software, EternalBlue, Heartbleed, etc. are examples of this category. General cyber hygiene for OS platform and applications should therefore still be in place in end systems.

C. Evaluation of whitelist protection

A number of known vulnerabilities were examined in the SDN laboratory which is shown in Figure 1. The Blue NE was configured as a MAC-learning switch and a full SDN switch with whitelist protection, respectively, while the same set of exploits where run. The focus of interest was to find exploits that could pass through the whitelist protection. Only exploits successful through the MAC-learning switch were tested on the whitelist protected NE.

Kali Linux [8] running in a virtual machine was connected to a port on Blue and given whitelist protection as a client, i.e., was only allowed to make *outbound* TCP connections on the approved ports. Also, virtual machines running Windows7, WindowsXP and Metasploitable Linux [9] were connected to other ports on Blue and were given whitelist protection as servers, where only *inbound* TCP connections are allowed.

For the evaluation we used Metasploit [9] installed on the Kali Linux virtual machine. Metasploit is a penetration testing framework shipped with a database of scripted exploits for known vulnerabilities, and various payloads to be combined with the exploits. The Windows7, WindowsXP and Metasploitable virtual machines acted as targets. Metasploitable is a deliberately vulnerable Linux server, while the Windows7 and WindowXP virtual machines were unpatched installations with Windows Firewall disabled. All three targets thus had known vulnerabilities.

From the design we expect all exploits which use destination ports other than the allowed ports (53,80,443,445) to be stopped by the whitelist protection. Exploits that depend on outbound connections from the attackee are not expected to succeed either.

These exploits were tested (names refer to their designation in Metasploit):

Unreal_ircd_3281_backdoor utilizes the IRC service port which is blocked by the whitelist. The attack is therefore not able to deploy a payload, and the attack is unsuccessful, even though Metasploitable is vulnerable to this exploit.

Ms08_067_netapi utilizes the SMB service port which is not blocked by the whitelist. A payload may be deployed to WindowsXP, but the *shell_reverse_tcp* is not allowed to make outbound TCP connections since this virtual machine is protected by a server-side whitelist. On the other hand, the *shell_bind_tcp* payload communicates over an incoming TCP connections which was bound to port 443, which is open in the whitelist. The attack was therefore successful.

Samba_symlink_traversal utilizes the SMB service port on a Linux computer and a poorly configured Samba service. The attack creates a symbolic link from a writeable share and opens every world-readable file for read access to an SMB client. Since the SMB service port is open in the whitelist, this exploit is successful.

Ms17_010_eternalblue utilizes the SMB port and exploits a bug in the server code in Windows7. It successfully deploys a payload. The chosen payload was *meterpreter_bind_tcp* which was instructed to bind to port 443 and wait for incoming connections. The exploit was successful.

Beside the Metasploit scripts, SQL injection and command injection were demonstrated on a web application on Metasploitable Linux (Mutillidae) deliberately coded for demonstration of the OWASP top ten web application vulnerabilities [10]. As long as these vulnerabilities are exploited through the normal service port, protection based on whitelists will have little effect.

The exploits shown above were carefully chosen for the demonstration of the limitation of whitelist protection mechanisms. Many possible exploits were not tested since they would obviously not succeed. It should also be noted that unpatched WindowsXP, Windows7 and Metasploitable Linux have obvious security flaws and would never be put in service in real life. And even a well maintained OS platform cannot protect a poorly programmed application service.

Some of the exploits succeeded only because there were whitelisted ports unoccupied by running services, in these cases TCP port 443. The whitelist should closely reflect the running services on the individual end system.

VII. Related research

The SDN architecture lends itself well to a range of techniques for intrusion detection and -prevention (IDS/IPS). The techniques differs on matters like:

 ISBN: 978-1-61208-821-1

- Does it offer prevention in addition to detection?
- Is the detection signature based or anomaly based?
- How much traffic does it create in the control plane?
- To what extent does it involve centralized computational resources?

In [11], Jankowski and Amanowicz demonstrate a IDS mostly targeted on attacks on the SDN controller and NEs, and are employing a range of machine learning techniques to detect anomalies. They base their evaluation on the KDD99Cup reference dataset for intrusions, which is commonly regarded to be obsolete [12]. Machine learning does not take place in NEs, so the design involves the SDNC to a large extent in the communication with centralized computational resources.

Intrusion prevention using SDN would involve dynamic updates of flow statements as a result of a positive intrusion detection. False positives (something anomaly based IDS is known for) will unnecessarily block suspected flows of traffic and obstruct legitimate use of the network. There are no known examples of such arrangement in the academic litterature, only a GitHub project which demonstrates this design [13].

The OpenFlow matching function is limited to the inspection of link- network- and transport headers, although an OpenFlow switch can also report traffic volumes associated with match statements as well as volumes across ports. Intrusion detection can base its decisions on matching function alone, in combination with traffic volume counters, or through inspection by the SDNC of the entire network frame. These approaches represent different observation horizons and different traffic load on the control plane links.

Several studies on anomaly based detection are known, they often limit their sensing to the reading of traffic volume counters and some even apply machine-learning algorithms for this purpose. For a survey of these reports, see [14].

Other approaches have been to raise suspicion on the basis of traffic counter values or the matching function, and to take in suspected flows in its entirety to the SDNC for deeper and stateful inspection of the payloads [11] [15] [16].

Signature based IDS is not seen as an SDN application, probably because the detection rules are way too complicated for the SDN matching functions, and bringing all network frames to the SDNC for stateful inspection would create a performance bottleneck in the control plane links.

Blocking of traffic flows as the result from anomaly detection always runs the risk of blocking legitimate traffic. A whitelisting approach, on the other hand, becomes a part of a service contract, where the end system and the network service supplier agrees on which services are available. E.g., in the particular configuration, e-mail has to be delivered through a web interface, not through IMAP or POP protocols. The whitelist serves as a predictable part of the application service and security planning, and has been shown to thwart a wide range of cyber attacks.

VIII. Conclusion

The presented paper has addressed new network functions in a tactical coalition network and demonstrated how new functions may be integrated into existing Network Elements without requiring new hardware components. The two new network functions, COI separation and whitelist protection, were demonstrated and evaluated for computational requirements and protection efficacy. The protection based on whitelists applies to every end systems in each connection point and becomes a valuable supplement to centralized security functions like intrusion detection and firewalls.

Remaining research topics on tactical SDN include the design and study of distributed SDN controllers. Wireless links are less reliable than wired links, and an in-band control plane arrangement will need to accommodate the event of lost connection between NEs and the SDNC. For this reason, a distributed SDNC design for tactical coalition SDN will be a subject for future research.

References

[1] E. Haleplidis *et al.*, "Software-Defined Networking (SDN): Layers and Architecture Terminology," RFC 7426, Jan. 2015, last accessed Oct 2020. [Online]. Available: https://rfc-editor.org/rfc/rfc7426.txt

[2] J. Spencer and T. J. Willink, "SDN in coalition tactical networks," in *2016 IEEE Military Communications Conference, MILCOM 2016, Baltimore, MD, USA, November 1-3, 2016*, 2016, pp. 1053–1058.

[3] "Mininet," http://mininet.org, Online, Accessed Oct 2020.

[4] "Open vSwitch," http://openvswitch.org, Online, Accessed Oct 2020.

[5] "Ryu SDN Framework," https://ryu-sdn.org/, Online, Accessed Oct 2020.

[6] A. Fongen, "Dynamic path discovery for in-band control plane communication in a tactical sdn network," in *EMERGING 2019, The Eleventh International Conference on Emerging Networks and Systems Intelligence*, Porto, Portugal, 2019, pp. 9–15.

[7] A. Fongen and G. Køien, "Trust management in tactical coalition software defined networks," in *2018 International Conference on Military Communications and Information Systems, ICMCIS 2018*. Institute of Electrical and Electronics Engineers Inc., 5 2018, pp. 1–8.

[8] "Kali Linux," http://kali.org, Online, Accessed Oct 2020.

[9] "Metasploit," http://metasploit.help.rapid7.com/docs/metasploitable-2, Online, Accessed Oct 2020.

[10] "Open Web Application Security Project," http://owasp.org/www-project-top-ten, Online, Accessed Oct 2020.

[11] D. Jankowski and M. Amanowicz, "Intrusion detection in software defined networks with self-organized maps," *Journal of Telecommunications and Information Technology*, vol. nr 4, pp. 3–9, 2015.

[12] A. Özgür and H. Erdem, "A review of kdd99 dataset usage in intrusion detection and machine learning between 2010 and 2015," https://peerj.com/preprints/1954v1/, 01 2016, Not Peer Reviewed. Online, Accessed Oct 2020.

[13] "SDN-Intrusion-Prevention-System-Honeypot," https://github.com/pratiklotia/SDN-Intrusion-Prevention-System-Honeypot, Online, Accessed Oct 2020.

[14] N. Sultana, N. Chilamkurti, W. Peng, and R. Alhadad, "Survey on SDN based network intrusion detection system using machine learning approaches," *Peer-to-Peer Networking and Applications*, pp. 1–9, 01 2018.

[15] Y. Hande, A. Muddana, and S. Darade, "Software-defined network-based intrusion detection system," in *Innovations in Electronics and Communication Engineering*, H. S. Saini, R. K. Singh, and K. S. Reddy, Eds. Singapore: Springer Singapore, 2018, pp. 535–543.

[16] T. A. Tang, L. Mhamdi, D. McLernon, S. A. R. Zaidi, and M. Ghogho, "Deep learning approach for network intrusion detection in software defined networking," in *2016 International Conference on Wireless Networks and Mobile Communications (WINCOM)*, Oct 2016, pp. 258–263.

 ISBN: 978-1-61208-821-1