Proceedings of the 6th Győr Symposium and 3rd Hungarian-Polish
and 1st Hungarian-Romanian Joint Conference on Computational Intelligence

Software-Defined Networks for Secure Distributed Industrial Communications

C. Costache[1], T. Balan[1], F. Sandu[1], D. Robu[1]

[1]"Transilvania" University, Electronics and Computers Department
Bd Eroilor nr 29A, 500036 Brasov, Romania
E-mail: costache.cosmin@unitbv.ro

Abstract: The security of distributed Ethernet-based industrial networks is critical, as vulnerabilities of these systems could affect national resources, civilians and the environment. Next-generation firewalls combine application awareness and deep packet inspection to give companies more control over applications while also detecting and blocking security threats. While most industry solutions are physical deployments of industrial switches with firewall capabilities, we propose a network security implementation of distributed middleboxes, dynamically deployed as Linux Containers and centrally managed by a controller, based on Software Defined Networks.

Keywords: SDN, Middleboxes, Linux containers, virtualization

1. Introduction

The Internet today carries lots of packets with little concern about their content. When it comes to mission-critical industrial applications, some recent famous examples (e.g. Stuxnet virus designed to attack industrial Programmable Logic Controllers, national electricity grid vulnerabilities) revealed the importance of the security actions in Industrial Ethernet networks. As it is difficult to implement security tools on the industrial end-devices, the solution would be to deploy a service-aware firewall on the communication link between each end-device and the network, based on distributed middlebox elements.

Middleboxes (MBs) are a crucial part of many enterprise LANs, data centres, and clouds, enabling enterprises to ensure security and improve performance [1]. With the help of middleboxes, the traffic can be further analysed and routing decisions can be taken based on the content carried by the packages [2]. Industrial middleboxes should understand the industrial automation protocol being used by the application, so they can monitor the detailed communication flow and verify it according to the application logic. So far in the industrial environment there are examples of physical implementations of security elements (like for example the RADiFlow service-aware Industrial Ethernet switches or the Cisco Industrial Ethernet 3000 Series Switches). These Industrial Ethernet management systems have advanced features like network

elements auto-discovery, topology configuring (IP sub-nets routing, Ethernet rings, VPN clouds), service groups provisioning and service group security rules planning.

Our proposal is a dynamic software solution that could deploy middleboxes on demand in the distributed Industrial Ethernet Network, based on SDN – Software Defined Networks and the Linux Docker containers. SDN has enabled MBs to be deployed at arbitrary locations in LANs and data centres [3]

The Linux containers represent an emerging technology for fast and lightweight process virtualization. Because the containers require fewer resources to run by sharing the operating system kernel, a higher density of containers can be achieved on the same host, opposed to other virtualization solutions like hardware or para-virtualization.

2. Software-Defined Networking for Industrial Communications

The Software Defined Networking (SDN) is a new architectural concept that aims to decouple the network control and forwarding functions [4]. This separation enables the network control layer to be programmable.

A typical SDN architecture consists of 3 layers. The top layer is the application layer which includes applications delivering services, in the presented case the security rules that represent the middlebox logic. The applications interact with the SDN controller which facilitates automated network management. At the bottom is the physical network layer composed of plain network elements. The network elements are simplified and they concentrate only on the forwarding functions. All the decisions, route calculations and policies are implemented in the controller [5]. Furthermore, by the use of Linux containers at network switch level, other specific security operations (like multi-site VPN using GRE tunnelling) could be deployed on demand in the network.

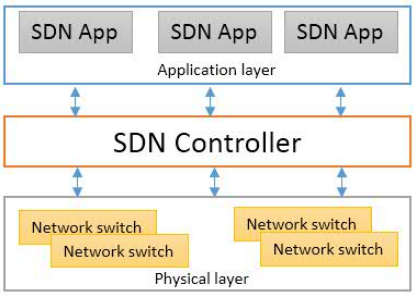

Figure 1 The typical SDN Architecture

In an SDN environment the controller is the central point, providing an abstract, centralized view of the entire network.

In the Industrial Communications [6] implementing SDN for would have a lot of advantages, some of them illustrated in [7]. Leading SCADA protocols, like Modbus or IEC 61850, used for automation, could be implemented in a distributed SDN network.

The advantage of implementing security with SDN is the flexibility of the system: some middleboxes could be deployed at the edge of the network, some security

functions need to be centralised, as visible in fig.2 [8]. Security policies could be adapted on demand or based on application and the migration and the easy nesting of migrated security appliances throughout the network are enhanced by using Linux containers.

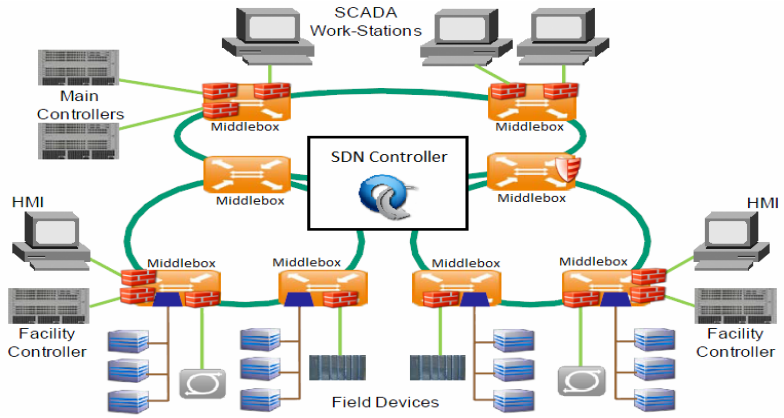

Figure 2 The typical SDN Architecture

3. Middleboxes

A middlebox (MB) is defined as any intermediary device performing functions other than the normal, standard functions of an IP router on the datagram path between a source host and destination host [2].

MBs present several challenges in the implementation. On one hand the specialized hardware is very expensive and it is difficult to extend it in the means of implementing new features. Introducing new features often means deploying new hardware. This leads to other problems like managing the MBs and making them fault-tolerant. With the current deployment model the MBs cannot be scaled on demand. A solution could be the use of virtualized MBs but this approach raises several requirements. The virtualized MBs must run in complete isolation in the case of a multi-tenant environment. Also they have to deliver high throughput with low delay. In order to scale on demand the virtualized MBs must be quickly instantiable.

In this paper we present a solution to virtualized MBs based on Linux containers (LXC) that can be provisioned and configured on demand. The Linux containers offer several advantages: they are running on common “of the shelf” (COTS) hardware with less resource requirements and they do not require any specialized hypervisor software. The LXC technology is included by default in the Linux kernel starting from version 3.4.

4. Implementing Middleboxes using Linux containers

LXC represent a different method of OS-level virtualization. It allows multiple isolated Linux systems (containers) to be run on a single host operating system. The host kernel provides process isolation and performs resource management. This means that even though all the containers are running under the same kernel, each container is a virtual

environment that has its own file system, processes, memory, devices, etc. In the research presented hereby, we used an open source implementation of the LXC technology called Docker.

Docker is an open-source platform for the management of Linux containers. Docker containers can be seen as extremely lightweight virtual machines (VMs) that allow code to be run in isolation from other containers. A Docker container can boot extremely fast making it the best candidate for on demand provisioning scenarios.

Linux containers are lighter and provide better performance compared to classical virtual machines. A full virtual machine can take up to several minutes to be provisioned, whereas a container can be instantiated and started in seconds. Because the containers do not run on top of a hypervisor, the applications they contain offer superior performance close to bare-metal performance.

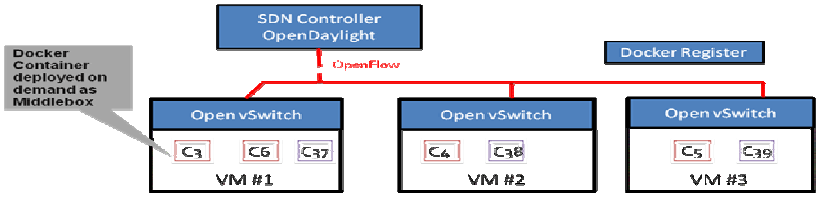

Figure 3 Architecture of the test SDN network

To simulate multiple network nodes hosting virtualized MBs, we have built a test environment composed of 3 virtual machines running Linux and each VM hosting multiple containers. Because we had only one physical machine available, we decided to use VMs to simulate a network topology with 3 nodes. All the VMs run on top of a Linux OS using the Kernel Virtualization Module (KVM). The host OS is a 64bit Ubuntu distribution (12.04 LTS). The virtual machines are running a guest OS based on the Ubuntu 14.04 distribution and each has allocated 1GB of RAM.

Because the Docker containers are lightweight, each VM will host multiple Docker containers. Additionally, on each VM we have installed a virtual switch module. After creation, all the containers on a VM will be attached to its local switch. The switches will be linked using GRE (Generic Routing Encapsulation) tunnels.

For the virtual switch we have chosen the open source software called “open vSwitch”. To enable the communication between the containers located on different virtual machines, we have created GRE tunnels between the 3 open vSwitch instances. The tunnel configuration is depicted in fig. 4.

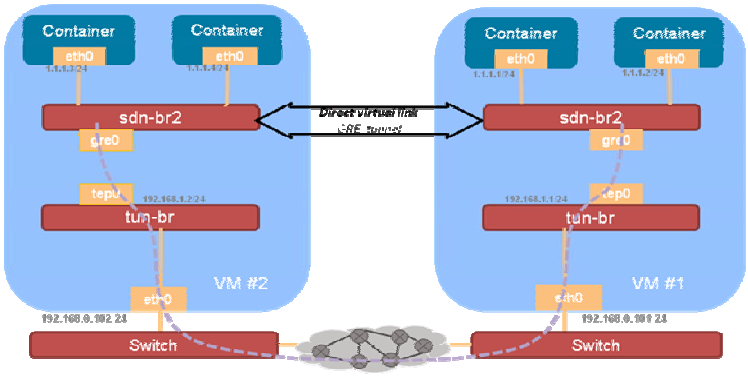

Figure.4 Direct virtual link between two bridges using GRE

Each open vSwitch instance is connected using GRE tunnels with its peers from the other VMs. The MB functionality is enclosed in a Docker container that can be provisioned on demand. After the container is started with the help of the SDN controller, flow entries can be inserted into the virtual switches and traffic routed to the middlebox.

The content and runtime configuration of a Docker container is stored in a repository as a template also called "Docker image". A Docker image can be downloaded from a public or private repository. For our test scenario we have configured a private Docker repository and made it available on the network. The repository has been populated with several Docker preconfigured containers. To facilitate the search and retrieval from the repository, each container has an associated unique ID.

When the container is started, the Docker daemon will automatically assign MAC and IP addresses and the container will be attached to the default docker0 bridge. An example with 2 containers connected to the default bridge is shown in fig. 4.

After the switches are registered with the controller, all the packets received by the switches, that do not match any entry in the flow table, are sent to the controller which takes the appropriate decisions. The controller can decide to insert a rule in the flow table of the switch or to drop the package. Besides the GUI interface, the controller exposes a set of APIs that can be used to automatically configure the flows between containers. This will allow the SDN applications to dynamically provision containers and configure the data flow as response to user requests.

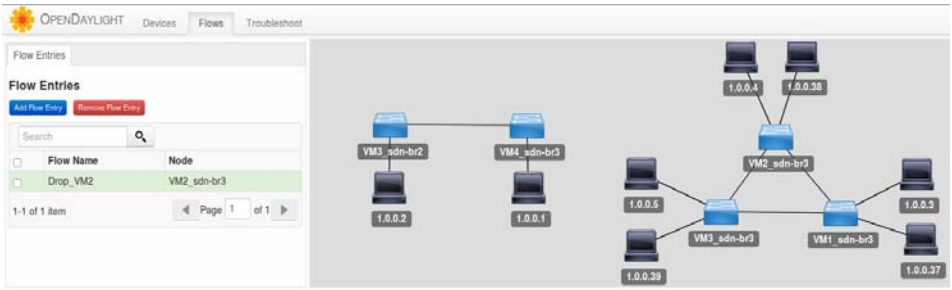

Figure.5 Using the same OpenDaylight SDN Controller for overallaping bridges

Using scripts we can now instantiate Docker containers into any of the available VMs and in the same time using the APIs provided by the SDN controller, to configure the underlying network to interconnect the containers. This creates the premises of dynamic provisioning of middleboxes. Industrial application-awareness can be obtained at switch level, by implementing industrial protocols in containers. The open implementations libmodbus [9] or libiec61850[10] can be deployed at container level with the benefit of excluding the OS interoperability problem, that is solved by Docker.

Conclusions

We identified the following typical functions of distributed industrial security elements, studying the existing legacy implementations [8]:

- Deployment of distributed firewall rules could be performed centralised at SDN controller level and deployed on demand on the switches/middleboxes. SDN allows flexibility and application-aware configurability of the firewall rules, while Linux containers allow the migration of policies configured at some network level to another location.
- The configuration of ACL (Access List) rules per port, to allow only devices with specific MAC or IP addresses to be connected to this port
- Detailed service-aware inspection of leading SCADA protocols (ModBus, IEC 101/104, DNP-3, IEC 61850) is not yet implemented at SDN level, but could be very efficiently implemented as part of the Linux containers using open source implementations [9-10]
- Inter-Site VPN, in case a distributed operational network has to use public transport links to connect between the sites. Our proposal is to use GRE tunnels, as shown in the previous example – fig. 4

As the industrial implementations are not easy to adapt and deploy in terms of new security policies, the solution for security in Industrial Ethernet networks would be the implementation of application-aware security rules at network level. Software Defined Networks and Linux Containers represent an optimal solution for a flexible, programmable and fast adaptable secure industrial network with awareness of specific industrial protocols.

Acknowledgement

This paper is supported by the Sectoral Operational Programme Human Resources Development (SOP HRD), ID134378 financed from the European Social Fund and by the Romanian Government.

References

[1] A. Gember, P. Prabhu, Z. Ghadiyali, A. Akella: *Toward Software-Defined Middlebox Networking*
[2] B. Carpenter: *Middleboxes: Taxonomy and Issues*, IBM Zurich Research Laboratory, http://tools.ietf.org/html/rfc3234

[3] M. Casado, M. J. Freedman, J. Pettit, J. Luo, N. McKeown, and S. Shenker: *Ethane: taking control of the enterprise*, In SIGCOMM, 2007
[4] Nadeau T., Gray K.: *SDN – Software Defined Networks*, O'Reilly, 2013, ISBN: 978-1-449-34230-2
[5] Jain R., Paul S.: *Network virtualization and software defined networking for cloud computing: a survey*, IEEE Communications Magazine, vol.51, no.11, pp.24-31
[6] Zurawski, R., editor – *Industrial Communication Technology Handbook*, Second Edition, CRC Press 2014, 1756 pag., ISBN 978-1-4822-0732-3
[7] Abhilash Gopalakrishnan, *Applications of Software Defined Networks in Industrial Automation*, http://www.academia.edu/2472112
[8] RAD Data Communications, *Secure Distributed Industrial Networks WP*, 2011
[9] Open Modus library: http://libmodbus.org/
[10] Open IEC61850 library: http://libiec61850.com/libiec61850/