# Deep Packet Inspection for intelligent Intrusion Detection in Software Defined industrial networks: A proof of concept

Markel Sainz, Iñaki Garitano, Mikel Iturbe, and Urko Zurutuza

Electronics and Computing Department
Mondragon University, Goiru 2, 20500 Arrasate-Mondragón, Spain,
{msainzo,igaritano,miturbe,uzurutuza}@mondragon.edu

**Abstract** Specifically tailored Industrial Control Systems (ICSs) attacks are becoming increasingly sophisticated, accentuating the need of ICS cyber security. The nature of these systems makes traditional IT security measures not suitable, requiring expressly developed security countermeasures. Within the last decades research has been focused in network based Intrusion Detection Systems (IDSs). With the appearance of Software Defined Networks (SDNs), new opportunities and challenges have shown up in the research community. This paper describes the potential benefits of using SDNs in Industrial Networks with security purposes and presents the set up and results of a pilot experiment carried out in an scaled physical implementation. The experimental set up consists in the detection of ICMP Flood and packet payload alteration based on signature comparison. Results point to the potential viability of the technology for intrusion detection and the need of researching in architectural scalability.

**Keywords:** Software Defined Networking, Industrial Control Systems, Security, Anomaly Detection

## 1 Introduction

Industrial Control Systems (ICSs) are a collection of special elements used for the control and supervision of industrial processes. ICSs are large scale, geographically dispersed heterogeneous, life-critical systems that encompass a variety of sensors, actuators and control and network devices [1]. Since the interconnection of ICSs to the Internet, Cyber Physical Systems (CPSs) security has become an important issue. Nowadays, most of the ICSs are composed of legacy hardware equipment designed decades ago, without following any security by design concept neither considering future existence of Internet and the many threats that would bring with it [2]. ICSs have been the objective of many cyber attacks used in IT environments and also, attacks designed specifically to cause damage in Operational Technology (OT) scenarios. In 2010, Stuxnet [3] worm demonstrated how sophisticated an attack could be by uploading malicious code to Programmable Logic Controllers (PLCs) and hiding the modifications. A set

of causes that make ICSs vulnerable are described by Graham *et al.* [4] such as the long hardware replacement periods and their limited computing power, the delay or non-existence of software or firmware updates and patches, the use of insecure communication protocols and the long lasting conviction that security can be enhanced through obscurity.

Consequently, after realizing the potential threat, the scientific community has been working on different approaches to enhance cyber security in ICSs, including Critical Infrastructures (CIs). Due to their availability constraints, upgrading ICS hardware to include security enhanced devices becomes difficult and thus, the development of technologies able to cope with legacy devices has been necessary, such as Intrusion Detection Systems (IDSs) [1].

Software Defined Networking (SDN) consists in an alternative networking paradigm that separates control and data plane, in order to ease the management and maintenance of IT networks [5]. SDN has demonstrated benefits in Traffic Engineering (TE) and security in traditional IT networks as stated in [6]. However, SDN has been barely used with cyber security purposes in ICSs, which could be considered a good test candidate due to their communication periodicity and predictability. Research in the area is getting more and more active with new approaches emerging both for IT and OT environments. This work presents a pilot experiment carried out in an scaled industrial scenario using Deep Packet Inspection (DPI) for the detection of network attacks.

The article is structured as follows. Section 2 presents the related work. Section 3 explains the experimental set up and the process used to obtain the data. Section 4 is used to expose the results. Finally Section 5 outlines identified research challenges and Section 6 extracts the final conclusions.

## 2 Related work

As it has been previously mentioned, the research in SDN based ICS security cannot be considered as mature as IT related, but it is getting more and more active with new publications showing up frequently. This section describes the approaches that we consider more relevant and lined up with our work.

Concerning DPI based solutions, several novel research articles can be cited. Wan *et al.* [7] present an event based anomaly detection system. The authors have created a solution that installs a DPI module in a switch using NFV and inspects packets matching them with a Hidden Markov Model designed adhoc. Ha *et al.* [8] describe the creation of an external IDS connected to the controller that samples part of the mirrored network traffic directly from the switches. The amount of traffic mirrored is dependent of the sampling rate established for each flow rule, that can be altered dynamically if suspicious behaviour is detected in a type of transmission. It is interesting that they have developed an algorithm to reduce the false negative occurrence as a consequence of low sampling rates. Murillo *et al.* [9] present an efficient solution composed of an external signature based IDS connected to an SDN controller. In case a threat is detected in the network, the controller gets notified and changes the needed flow

rules to deviate the transmission to a reliable ICS honeypot, in order to register precisely the attack. The approach published by Mantur *et al.* [10] consists of an external IDS connected to Opendaylight [11], a well known SDN controller which changes network behaviour based on inspected packets. The intrusion detection process is signature based and is carried out by using Hadoop. Sultana *et al.* [12] describe in a rich way several SDN based intrusion detection solutions using Machine Learning (ML) and Deep Learning (DL) algorithms, emphasizing the detection accuracy of DL based approaches. Niyaz *et al.* [13] have presented a work for the detection of DDoS attacks in Software Defined Network by the use of DL algorithms with an accuracy rate of 95%. They have been able of detecting seven types of DDoS attacks with different communication protocols.

Apart from DPI based solutions, there are others focused on the flow-rule statistics analysis. Braga *et al.* [14] present a solution that recollects flow rule statistics and detect DDoS attacks by the use of Self Organizing Maps. They use as input relevant OpenFlow features as packet counts, byte counts, flow pairs, augmentation of single flows and augmentation of used ports. Abubakar *et al.* [15] have built an hybrid solution consisting on an Snort IDS and a traffic statistics based anomaly detection system. The solution is based in flow counts and uses neural networks to detect misbehaviours with a 97% accuracy rate. The used neural network have been trained with NLS-KDD dataset, which can be considered quite obsolete and they do not present any solution for payload based zero day attacks.

The presented approach bases its efficacy in the centralization of the DPI process by performing it on the SDN Controller. This leads to the enlargement of the visibility of the network and the possible categorization of traffic in an smarter way. Moreover, implementing the detection logic in a high level device enables the development of complex logic for intrusion detection.

## 3 Experimental set up

In order to test the viability of developing SDN based IDSs, an experimental environment has been set up where Deep Packet Inspection is used to detect illicit network packets and prevent them from reaching their destination. To test the suitability of the strategy, the experimental phase has been divided into two test cases.

The experimentation testbed consists in an emulation of a bottle filling plant deployed in the eMUlab laboratory of Mondragon Unibertsitatea. eMUlab laboratory sits on top of emulab software [16], which allows the construction of a wide range of testing environments. Figure 1 shows the topology built for this experiment. The plant simulation runs in an ABB AC800M proprietary PLC, which stands in the control network of the topology and is connected directly to an OpenFlow Switch. On the other side, an ABB SCADA Server located in the supervisory network, pulls data from the mentioned PLC. ABB devices communicate with each other by default using the Manufacturing Message Specification (MMS) protocol [17]. In the middle of the link between the SCADA Server and

the OpenFlow switch, an Ubuntu machine has been set with the purpose of performing a transparent Man in The Middle (MITM) attack using two bridged Ethernet ports and capturing traffic with an application based on Scapy [18]. This machine is used in order to modify licit packets to alter their functionality or create and inject new malicious packets into the network. Lastly, the OpenFlow switch is attached to an Opendaylight controller, running in an isolated subnet. The controller is not accessible for any device except for the switch, which is connected to it using a dedicated port that is unreachable from the rest. A bundle has been developed in controller native language Java that implements listening capabilities to capture network packets and process them. The experimentation phase has been divided into two different test cases: MMS packet alteration and ICMP Flood.

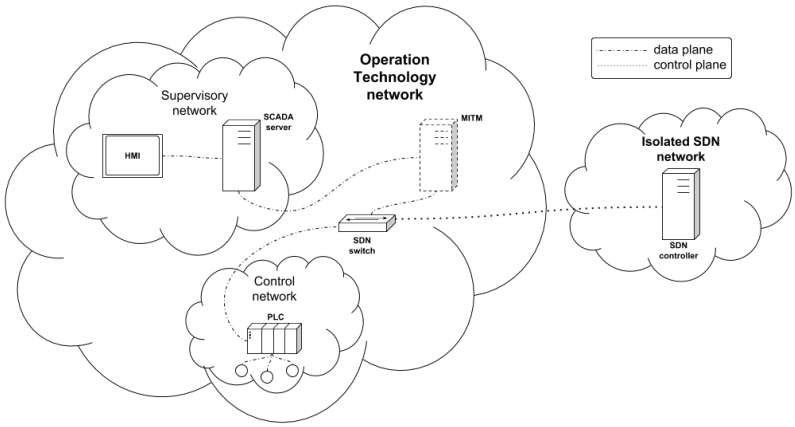



Figure 1: Implemented topology

### 3.1 Test Case 1: MMS packet alteration

The bottle filling plant simulation running in the PLC manages a set of variables and processes that make possible the operation. This variables can be seen from an HMI connected to the SCADA Server. To enable this action, SCADA server periodically pulls available data from PLC and refreshes its database. As stated before, this communication happens under the MMS protocol, a proprietary industrial network protocol. MMS integrates a collections of message types or PDUs with different purposes. Under normal condition with no errors, the following four PDU types can be identified if the transmissions are captured:

- *Initiate-RequestPDU*
- *Initiate-ResponsePDU*

– *Confirmed-RequestPDU*
– *Confirmed-ResponsePDU*

The first two are essential to establish a TCP tunnel among two devices. Once established, they will not be sent again unless a connection reset happens. The other two are associated with the periodical request and response of variable data. SCADA Server will send a *Confirmed-RequestPDU* packet to PLC whenever it wants to refresh its database and if transmission is correct, PLC will response with a *Confirmed-ResponsePDU* that will contain the actual value of the variables. In this scenario, this information exchange occurs once every second, as it has been set in the SCADA Server configuration. The attack focus on this implementation has been the request packet. Request packets are expected to be the same during all the operation cycle, while responses will change continuously due to process variations. In the future, possible responses will be mapped and used to detect anomalies. Figure 2a shows a fragment of a *Confirmed-RequestPDU* packet captured with Wireshark [19], specifically the section where the address of requested variable values is stored. This packet is firstly intercepted by the MITM machine, where it is identified and altered. Figure 2b shows exactly the same packet after being altered. It can be observed that bytes six and seven have exchanged positions. Taking into account that the purpose of this work is demonstrating the capability of SDN to detect illicit packets in industrial environments, the alteration of the packet has been performed in an arbitrary way, due to the little relevance of the attack complexity in this particular case. By altering the packet this way, the pointer to the PLC variables has been changed, causing a malfunction in the value retrieval process. Once the packet has been altered by the MITM machine, it is again pushed to the network in the original direction.

```
00a0  82 26 03 32 a7 01 17 52  33 36 32 37 35 34 33 31
00b0  34 41 70 70 6c 69 63 61  74 69 6f 6e 5f 31 02 00
00c0  1e 03 00 24 00 28 00 1c
```

(a) MMS Request licit packet

```
00a0  82 26 03 32 a7 17 01 52  33 36 32 37 35 34 33 31
00b0  34 41 70 70 6c 69 63 61  74 69 6f 6e 5f 31 02 00
00c0  1e 03 00 24 00 28 00 1c
```

(b) MMS Request altered packet

Figure 2: MMS Request packets

As mentioned earlier in this section, the switch is attached to an SDN controller, where an application bundle has been developed to perform DPI. When the packet arrives to the switch, its header values are checked looking for a match with installed flow rules. If a match happens, the associated action will be performed for that packet. If no flow rule matches with the incoming transmission, the packet is sent to the controller for further processing tagged as *PACKET_IN* under the OpenFlow encapsulation. In our approach, packets are sent directly to controller, where the developed application handles them and applies the appropriate routing logic after extracting the payload from the OpenFlow income and inspecting it. The application acts as a learning switch, with a dynamic MAC table that performs routing transparently.

Every packet that reaches the controllers is inspected throughout different analysis stages. If the matching conditions of an stage are met, the packets is further processes in the next stage. Otherwise, is routed or dropped, depending on the result. First stage consists in the extraction of destination IP direction. Packets matching the PLC destination IP direction are further processed, while the rest are normally routed. The second stage looks for TCP port 102 and byte traces belonging to the Connection Oriented Transport Protocol (COTP) based on ISO 8073 which is used by the MMS protocol [17]. Each TCP packet matched in the previous stage is checked looking for MMS request distinctive bytes in the payload. As it has been previously described different MMS packet types can be found in the set up, and the *requestPDU* type packets are isolated in this stage. If a packet matches the previous requisite, it is processed. The DPI process looks for the variable access address looking for anomalies. In this stage the required bytes are extracted from the packet payload and compared to the corresponding bytes in a licit sample packet. If the byte sequence does not match the one in Figure 2a, the packet is dropped, preventing it to reach its destination and cause a malfunction in the PLC or a connection reset.

### 3.2 Test Case 2: ICMP Flood

In this case, the performed attacks are agnostic to the process itself and the protocols used for the normal operation of the designed (ICS). To overcome with this test case, two attacks have been launched and a single process has been developed to detect them.

Both attacks have been launched from the MITM machine transparently without interfering in the normal network operation. The generated packets have been ICMP packets with PLC IP direction as destination point and a random IP direction from the licit subnet as source point. The first attack consists in the launching of a ping of the death offensive. With that aim, an ICMP packet has been created exceeding the maximum length of 65536 bytes. This has been accomplished by creating normal Ethernet/IP/ICMP header and oversizing the payload with "X" characters. The generated packet has been then fragmented into diverse size packets. The obtained packets are sent to the the PLC in order to saturate the buffer when the fragments are recomposed [20]. The second attack consists in a traditional ICMP flood attack [21]. The key concept in this attack is the generation of a massive quantity of ICMP request packets that are sent to a device, aiming to collapse its network card when trying to cope with them. With this objective, MITM machine executes a thread that generates and sends continuously ICMP packets to the PLC until user stops the attack on demand. This packets are sent to the PLC as they are generated, throughout the network switch.

Referring to the detection process, the DPI performance strategy implemented in this scenario is the same as in Test Case 1, where the network switch is left without flow rules, and all the packets are sent to the controller. The controller inspects the packets and manages the routing process by dropping or routing them correctly. To detect the attack an algorithms has been designed. Firstly, a

filter is used to detect relevant packets. This filter inspects the header values in search of the destination IP direction, and the IP protocol, matching the ones which belong to ICMP. After the filtering process, the designed algorithm is executed in order to decide whereas the detection of an ICMP packets consists in a potential attack, or in an ordinary PING operation. It is necessary to note, that PING requests use ICMP protocol and it is assumed that normal PING operations must be allowed to happen, in order to check connectivity among devices. As machines used to control and manage the PLC are Windows machines, which perform 4 ICMP requests when pinging, the maximum number of ICMP requests has been set to 4. The developed algorithm uses two counters, one for ICMP packets and one for other traffic type packets. When an ICMP packet incomes, the ICMP counter is increased by one and the other counter is set to 0. Once ICMP counter reaches 4, all the incoming ICMP traffic is dropped. To reset this counter, at least ten packets have to income which are not ICMP protocol. When this happens, all the counters are reset. The maximum values for the counters have been adjusted based in the nature to the experimentation environment and can be configured on demand with adequacy purposes.

## 4 Results

This section describes the results obtained in the experimentation phase, divided into the two same sections presented in Section 3, according to each of the test cases.

### 4.1 Test Case 1: MMS packet alteration

The presented approach has been able of detecting malicious request packets satisfactorily. To test the response of the implementation, a single packet alteration has been performed by the MITM machine on user demand during normal operation of the environment. After that attack, normal condition has been restored.

Wireshark has been used with the appropriate filters to monitor only the relevant MMS traffic. To achieve this, the traffic has been filtered by MMS protocol and frame length, due to the infrequent change on the length of the relevant frames, that increases only when ID size is exceeded. Opendaylight controller as it has been explained in the previous sections, applies the routing logic of the packets, so it is expected to capture an associated *PACKET_OUT* frame for each *PACKET_IN* it processes, unless the packet is dropped by the implemented logic. Figure 3 shows a graphical representation of the capture made with Wireshark in the moment when the attack is performed, having captured also previous and latter normal condition traffic. Each *PACKET_IN* frame has an immediate *PACKET_OUT* associated response except for the fifth frame, where the altered packet is processed and dropped due to its illicitness. For the sake of simplicity, time units in the figure are not the same as in the real capture, having used natural numbers in the image. Nevertheless, it can be

appreciated that frames are captured every second with a little delay, product of the full forwarding process. To conclude, it can be observed that the developed environment is able of preventing malicious transmissions by dropping only illicit packets before they reach their destination without interfering in the correct operation of harmless transmissions.

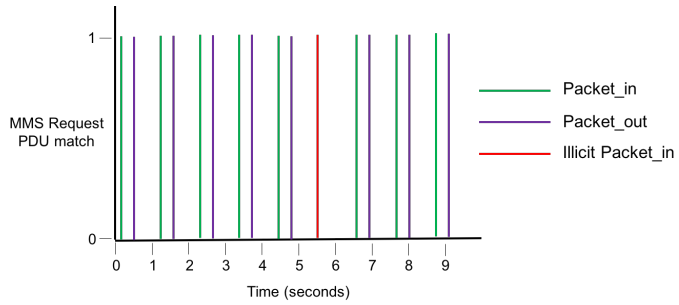


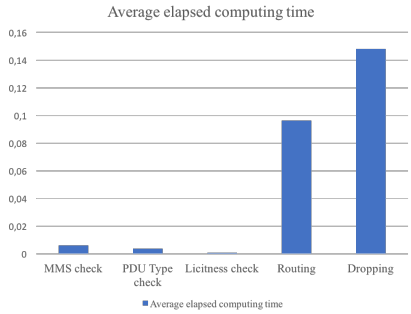

Figure 3: MMS Request PDU match captures

To complement the results obtained by the network analyser and measure precisely the efficiency of the developed algorithms, average elapsed times have been obtained for different stages of the packet inspection process. Figure 4 shows a graphical representation of the numerical values present in the table. It can be appreciated, that functions used to identify the type of a packet require more computing time than matching the payload with the stored signature. This is likely to happen due to the diversity of packets inspected and the necessity of checking different parameters and specific byte values. Once a packet has been identified as *Confirmed-RequestPDU*, matching it with the signature takes less computing time. Regarding the full time of packet processing, it can be appreciated that dropped packets involve more computing time than accepted ones. Accepted packets are not only MMS packets, but many other type of them. This means that many packets are just accepted due to the lack of MMS filter match, without further processing. Dropped packets are always the same type of MMS packet, forced to complete all the inspection process before being accepted or rejected.

### 4.2 Test Case 2: ICMP Flood

The designed algorithm has been satisfactorily used to deny both attacks using the same detection criteria. This has been achieved without interfering in the normal network operation. In order to capture the relevant data of the test case, Wireshark has been used.

Concerning to the impact of the attacks, although ping of death has been traditionally used with good results [20], it has been ineffective in this imple-




| DPI process stages | Avg time (ms) |
|---|---|
| MMS check | 0,00633 |
| PDU Type check | 0,00398 |
| Licitness check | 0,00099 |
| Full routing process | 0,0963 |
| Full dropping process | 0,148 |

Figure 4: Elapsed time for DPI process

mentation due to the existence of required inbuilt countermeasures in the PLC. On the other hand, ICMP Flood attack has been highly effective, collapsing completely the network capability of the PLC and leaving it out of communication for a considerable amount of time after stopping the attack. The attack has been able of interfering in the control communication and leaving the PLC isolated from the OPC Server.

As for the detection process itself, the captures from Wireshark have shown the absolute efficacy for the detection of the launched attacks. A summary table has been created for each of the attacks with the relevant data from their corresponding Wireshark captures. To understand the information shown in the tables, it has to be taken into account that if a packet is routed correctly, a PACKET_IN-PACKET_OUT pair has to exist for the transmission, corresponding to the income to the controller and the outgoing from this device to its respective destination.

Table 1 corresponds to the results obtained in the Ping of Death attack. This table shows that the first 4 packets are routed correctly and the rest of the packets are dropped by the controller. The entries in the table show the fragments of the large generated packets, that is why they are oversized. The same thing happens with the ICMP Flood, as it can be seen in Table 2. In this particular case, the packets are normal ICMP packet and their size is smaller. In both cases it can be clearly seen that the first 4 packet pairs correspond to accepted transmissions and the rest are dropped. This behaviour lasts as long as the attack is kept on. Additionally, a PING request has been made from the OPC Server to the PLC under normal conditions to test the viability of doing such an operation, with satisfactory results.

## 5 Research challenges and future work

The extracted data in this work points to the possibility of exploiting the technology with security purposes to develop solutions that may require less investment due to the lack of necessity of adhoc hardware equipment and even in some

| TS | IPsrc-IPdst | Size | OF tag |
|---|---|---|---|
| 1.300 | 172.17.84.131-172.17.84.124 | 1598 | P_IN |
| 1.301 | 172.17.84.131-172.17.84.124 | 1604 | P_OUT |
| 1.304 | 172.17.84.131-172.17.84.124 | 1598 | P_IN |
| 1.304 | 172.17.84.131-172.17.84.124 | 1604 | P_OUT |
| 1.305 | 172.17.84.131-172.17.84.124 | 1598 | P_IN |
| 1.306 | 172.17.84.131-172.17.84.124 | 1604 | P_OUT |
| 1.307 | 172.17.84.131-172.17.84.124 | 1598 | P_IN |
| 1.307 | 172.17.84.131-172.17.84.124 | 1604 | P_OUT |
| 1.310 | 172.17.84.131-172.17.84.124 | 1598 | P_IN |
| 1.310 | 172.17.84.131-172.17.84.124 | 1598 | P_IN |
| 1.314 | 172.17.84.131-172.17.84.124 | 1598 | P_IN |

Table 1: Ping of death attack

| TS | IPsrc-IPdst | Size | OF tag |
|---|---|---|---|
| 1.149 | 172.17.84.131-172.17.84.124 | 626 | P_IN |
| 1.150 | 172.17.84.131-172.17.84.124 | 632 | P_OUT |
| 1.151 | 172.17.84.131-172.17.84.124 | 626 | P_IN |
| 1.151 | 172.17.84.131-172.17.84.124 | 632 | P_OUT |
| 1.152 | 172.17.84.131-172.17.84.124 | 626 | P_IN |
| 1.152 | 172.17.84.131-172.17.84.124 | 632 | P_OUT |
| 1.154 | 172.17.84.131-172.17.84.124 | 626 | P_IN |
| 1.154 | 172.17.84.131-172.17.84.124 | 632 | P_OUT |
| 1.155 | 172.17.84.131-172.17.84.124 | 626 | P_IN |
| 1.161 | 172.17.84.131-172.17.84.124 | 626 | P_IN |
| 1.164 | 172.17.84.131-172.17.84.124 | 626 | P_IN |

Table 2: ICMP Flood attack

cases, reusing existing compatible network elements. Anyway, further research is needed and will be done in order to prove the viability of developing intelligence on top of the DPI modules using artificial intelligence approaches. This section presents some thoughts and future tasks focused on the continuation of the transmitted research.

Regarding the actual implementation, improvement strategies can be designed. It is necessary to have in mind that it has been developed in an scaled real environment with physical equipments, with the limitations that this fact may impose. In this particular case, the DPI process occurs entirely in the controller, with the need of routing all traffic through it. The topology designed is composed of a single switch and two hosts, with no excessive traffic and complexity. It has been demonstrated that efficiency has not been a problem but it must be considered that increasing inspection process and topology complexity will lead to a penalization in efficiency. The increase on the amount of traffic that the controller will have to manage in the actual implementation is proportional to the number of OpenFlow devices attached to it, due to the lack of installed flow rules on them. Moreover, developing intelligent algorithms based in for example, machine learning, would slow down the packet processing time and as a consequence, the full forwarding process. Research in the subject has conducted the team to the discovery of possible enhancements that could be applicable.

One of the main problems stands in the fact that all the traffic is processed and has to pass through a single node, in this case, the controller. This may suppose not only a congestion bottleneck but also a threat vector, because if controller got compromised, the network would go down. Some solutions found in the literature such as [9] [8] [10] use an external IDS that warns the controller in case an anomaly is detected in order to update forwarding rules in affected switches. Approaches such as the one proposed by Braga *et al.* [14] use OpenFlow statistics collected from the forwarding counts in the installed flow rules to detect anomalies. Both kind of previously mentioned solutions are able of detecting network intrusions but they are not able of responding just in the moment when an illicit packet is detected.

A substantial contribution can be done in the research of different packet inspection strategies using available technology. Firstly, as it has been presented in this work, DPI can be implemented directly on top of the controller with no installed flow rules on forwarding devices, forcing all the traffic to pass through the controller node. A better implementation of this would combine the use of flow rules in network switches with the ability of them to clone every packet and send it to the controller. This way, intrusion detection can be performed without interfering in the normal operation of the network, and still act as fast enough to stop an attack if necessary. Other solutions differ from the actual one in terms of the node that processes the traffic, delegating this task to the switches themselves or to specific adhoc devices. The distribution of detection logic throughout various network switches seems to be a promising research area also [22].

It is necessary to gather information about existing network operating systems in the market that can be installed in switches, providing both, OpenFlow forwarding capabilities, and DPI engines. Last mentioned solution can be also achieved using Network Function Virtualization. This technology allows to develop virtual networking devices on top of common IT hardware such as hosts or servers, providing higher computing power and extended capabilities [23]. Virtualized Network Functions (VNFs) are implemented in hardware and are managed by an SDN controller, with whom they can exchange data of the results obtained from packet inspection. It is evident that a deep research must be done in order to enlighten what can be done with existing technology and how to use it for intrusion detection purposes.

## 6 Conclusions

Software Defined Networks are gaining recognition in research community and are increasingly being implanted in many IT environments. On the other hand, the adoption of the technology in industrial scenarios is being slower. This work has firstly analysed the potential distinctive features of both, industrial networks and SDN technology and the benefits of using them in conjunction for the creation of intelligent security solutions. Related with the previous assertion, a pilot experiment has been carried out in order to test the technology's viability

in performing DPI for Intrusion Detection in an scaled industrial environment, obtaining satisfactory results.

The experimental set up proposed in this paper has demonstrated the viability of performing DPI on top of an SDN controller to detect malicious network packets satisfactorily. However, the current implementation has been designed having as a reference an small ICS with reduced network traffic. Presumably, this implementation may not scale correctly for bigger network scenarios, so future research path will be focused on the deployment of more efficient DPI performing architectures and detection strategies, by testing distributed machine learning based approaches.

**Acknowledgements** Iñaki Garitano is partially supported by the INCIBE grant "INCIBEC-2015-02495" corresponding to the "Ayudas para la Excelencia de los Equipos de Investigación avanzada en ciberseguridad". This work has been developed by the intelligent systems for industrial systems group supported by the Department of Education, Language policy and Culture of the Basque Government. It has been partially funded by SEKUTEK project. This project has received funding from the Department of Economic Development and Infrastructures under the grant agreement KK-2017/00044. Moreover, it has been partially funded by the POSIC project, funded by the Gipuzkoa Provincial Council under the grant agreement 93/17.

## References

1. R. Mitchell and I.-R. Chen, "A survey of intrusion detection techniques for cyber-physical systems," *ACM Computing Surveys*, vol. 46, no. 4, pp. 1–29, mar 2014. [Online]. Available: http://dl.acm.org/citation.cfm?doid=2597757.2542049
2. M. Krotofil and D. Gollmann, "Industrial control systems security: What is happening?" *2013 11th IEEE International Conference on Industrial Informatics (INDIN)*, no. July 2013, pp. 670–675, 2013. [Online]. Available: http://ieeexplore.ieee.org/document/6622964/
3. S. Karnouskos, "Stuxnet worm impact on industrial cyber-physical system security," in *IECON 2011-37th Annual Conference on IEEE Industrial Electronics Society*. IEEE, 2011, pp. 4490–4494.
4. J. Graham, J. Hieb, and J. Naber, "Improving cybersecurity for Industrial Control Systems," *IEEE International Symposium on Industrial Electronics*, vol. 2016-Novem, pp. 618–623, 2016.
5. M. Dabbagh, B. Hamdaoui, M. Guizani, and A. Rayes, "Software-defined networking security: Pros and cons," *IEEE Communications Magazine*, vol. 53, no. 6, pp. 73–79, 2015.
6. M. Sainz, M. Iturbe, I. Garitano, and U. Zurutuza, *Software defined networking opportunities for intelligent security enhancement of industrial control systems*, 2018, vol. 649.
7. M. Wan, J. Yao, Y. Jing, and X. Jin, "Event-Based Anomaly Detection for Non-Public Industrial Communication Protocols in SDN-Based Control Systems," vol. 55, no. 3, pp. 447–463, 2018. [Online]. Available: www.techscience.com/cmc

8. T. Ha, S. Kim, N. An, J. Narantuya, C. Jeong, J. Kim, and H. Lim, "Suspicious traffic sampling for intrusion detection in software-defined networks," *Computer Networks*, vol. 109, pp. 172–182, nov 2016. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1389128616301645
9. A. F. Murillo Piedrahita, V. Gaur, J. Giraldo, A. A. Cardenas, and S. J. Rueda, "Leveraging Software-Defined Networking for Incident Response in Industrial Control Systems," *IEEE Software*, vol. 35, no. 1, pp. 44–50, jan 2018. [Online]. Available: http://ieeexplore.ieee.org/document/8239925/
10. B. Mantur, A. Desai, and K. S. Nagegowda, "Centralized Control Signature-Based Firewall and Statistical-Based Network Intrusion Detection System (NIDS) in Software Defined Networks (SDN)," in *Emerging Research in Computing, Information, Communication and Applications*. New Delhi: Springer India, 2015, pp. 497–506. [Online]. Available: http://link.springer.com/10.1007/978-81-322-2550-8{_}48
11. J. Medved, R. Varga, A. Tkacik, and K. Gray, "Opendaylight: Towards a model-driven sdn controller architecture," in *A World of Wireless, Mobile and Multimedia Networks (WoWMoM), 2014 IEEE 15th International Symposium on*. IEEE, 2014, pp. 1–6.
12. 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, jan 2018. [Online]. Available: http://link.springer.com/10.1007/s12083-017-0630-0
13. Q. Niyaz, W. Sun, and A. Y. Javaid, "A Deep Learning Based DDoS Detection System in Software-Defined Networking (SDN)," nov 2016. [Online]. Available: http://arxiv.org/abs/1611.07400http://dx.doi.org/10.4108/eai.28-12-2017.153515
14. R. Braga, E. Mota, A. P. L. C. N. (LCN), and undefined 2010, "Lightweight DDoS flooding attack detection using NOX/OpenFlow," *ieeexplore.ieee.org*. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/5735752/
15. A. Abubakar, Pranggono, and Bernardi, "Machine learning based intrusion detection system for software defined networks." [Online]. Available: http://shura.shu.ac.uk/16558/
16. K. Emulab, "Network Emulation Testbed Home."
17. Jan Tore Sørensen and Martin Gilje Jaatun, "An Analysis of the Manufacturing Messaging Specification Protocol," vol. 6905, no. May, 2011. [Online]. Available: http://link.springer.com/10.1007/978-3-642-23641-9
18. P. Biondi, "Scapy: explore the net with new eyes." [Online]. Available: http://secdev.org/conf/scapy{_}T2.pdf
19. G. Combs, "Wireshark-network protocol analyzer," *Version 0.99*, vol. 5, 2008. [Online]. Available: https://www.emulab.net/
20. E. Luiijf, "Threats in Industrial Control Systems." Springer, Cham, 2016, pp. 69–93. [Online]. Available: http://link.springer.com/10.1007/978-3-319-32125-7{_}5
21. S. Hansman and R. Hunt, "A taxonomy of network and computer attacks," *Computers and Security*, vol. 24, no. 1, pp. 31–43, 2005.
22. H. Li, H. Hu, G. Gu, G.-J. Ahn, and F. Zhang, "vNIDS: Towards Elastic Security with Safe and Efficient Virtualization of Network Intrusion Detection Systems," in *Proc. of the 25th ACM Conference on Computer and Communications Security (CCS'18)*. New York, New York, USA: ACM Press, 2018, pp. 17–34. [Online]. Available: http://dl.acm.org/citation.cfm?doid=3243734.3243862
23. Yong Li and Min Chen, "Software-Defined Network Function Virtualization: A Survey," *IEEE Access*, vol. 3, pp. 2542–2553, 2015. [Online]. Available: http://ieeexplore.ieee.org/document/7350211/