Review

Securing IoT networks through SDN technologies

Jose Miguel-Alonso *

Posted Date: 26 July 2023

doi: 10.20944/preprints202307.1781.v1

Keywords: Internet of Things; Software Defined Networking; Cybersecurity.

Review

Securing IoT Networks through SDN Technologies

Jose Miguel-Alonso *

Department of Computer Architecture and Technology, Faculty of Informatics, University of the Basque Country UPV/EHU, 20018 Donostia-San Sebastian, Spain; j.miguel@ehu.eus

Abstract: Security in IoT systems is extremely important, as an intrusion into an IoT device or network can affect not only our domestic lives, but also industrial assets, with the potential to cause enormous damage. We discuss IoT security issues as defined by the OWASP Foundation, focusing on network related aspects. After a brief description of SDN technologies, with focus on OpenFlow standards, we discuss how SDN technologies can greatly help in designing and deploying more secure IoT networks by enhancing the cryptographic capabilities of devices, isolating devices or networks, blocking access to unwanted services, redirecting traffic to deep inspection systems, monitoring packet flows and devices, etc. These capabilities can be implemented using open-source OpenFlow controllers such as Faucet.

Keywords: internet of things; software defined networking; cybersecurity

1. Introduction

The Internet of Things (IoT) refers to physical devices, or networked sets of devices, that are equipped with sensors, processors, software, and other technologies that connect to and exchange data with other systems and equipment using the Internet or other networks [1]. This technology has changed the way we interact with devices at home or in a building, and also the way devices intercommunicate in a smart factory (industrial IoT). The advantages of IoT come together with some risks, some of them related to the fact that devices are connected and, therefore, exposed to attacks from external parties.

Software-defined networking (SDN) technologies can be viewed as a centralized way to monitor, configure, and manage a network infrastructure. With the right configuration, an SDN-enabled device can act as an Ethernet switch, router, firewall, load balancer, and so on. Even better, it can perform all of these functions simultaneously. This is because SDN technologies provide very flexible ways of forwarding (and blocking) packets. Furthermore, SDN technologies also provide a global view of the infrastructure, as monitoring is an integral part of the SDN specifications [2]. For the remainder of this paper, we are going to use the the OpenFlow standards published by the Open Networking Foundation [3] as a way to implement SDN concepts. OpenFlow is not the only possible implementation of SDN, but it is a popular one and allows us to provide specifics about the realization of security-related operations.

This paper ties the two concepts together by providing guidance on how SDN technologies in general (and OpenFlow standards in particular) can help to enhance an IoT network (i.e., a network used for device-to-device, device-to-edge, device-to-cloud, edge-to cloud communications) with security measures to reduce the risk of successful attacks. We begin by defining the major IoT security risks (Section 2) and the basics of the OpenFlow specifications (Section 3). In Section 4, we explain how SDNs can help improve network security in general, and continue in Section 5 with guiding tips for securing IoT networks. Section 6 discusses a possible implementation of these ideas using Faucet, an open-source controller. We end with some conclusions in Section 7.

2. Security Issues of IoT Systems

OWASP (The Open Worldwide Application Security Project) is a non-profit foundation that works to improve software security [4]. The work of OWASP is divided into open source projects, which are collections of related tasks that have a dedicated team of volunteer members and a well-

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 26 July 2023 doi:10.20944/preprints202307.1781.v1

defined roadmap. Currently, OWASP maintains more than 100 active security-related projects, several of them including the words "Top 10": these are lists of security risks of a particular technology. Some examples are "Top 10 Web Application Security Risks", "Top 10 Kubernetes Risks", and "Top 10 Privacy Risks".

The project we are focusing on is called "Internet of Things" [5]. It aims to *"help manufacturers, developers, and consumers better understand the security issues associated with the Internet of Things, and to enable users in any context to make better security decisions when building, deploying, or assessing IoT technologies*". It is structured into several subprojects, one of which is in charge of maintaining a top 10 list of security issues associated to IoT. The most recent list was published in 2018, see it in Figure 1. There is a previous list from 2014. A mapping between the two lists available in [6].

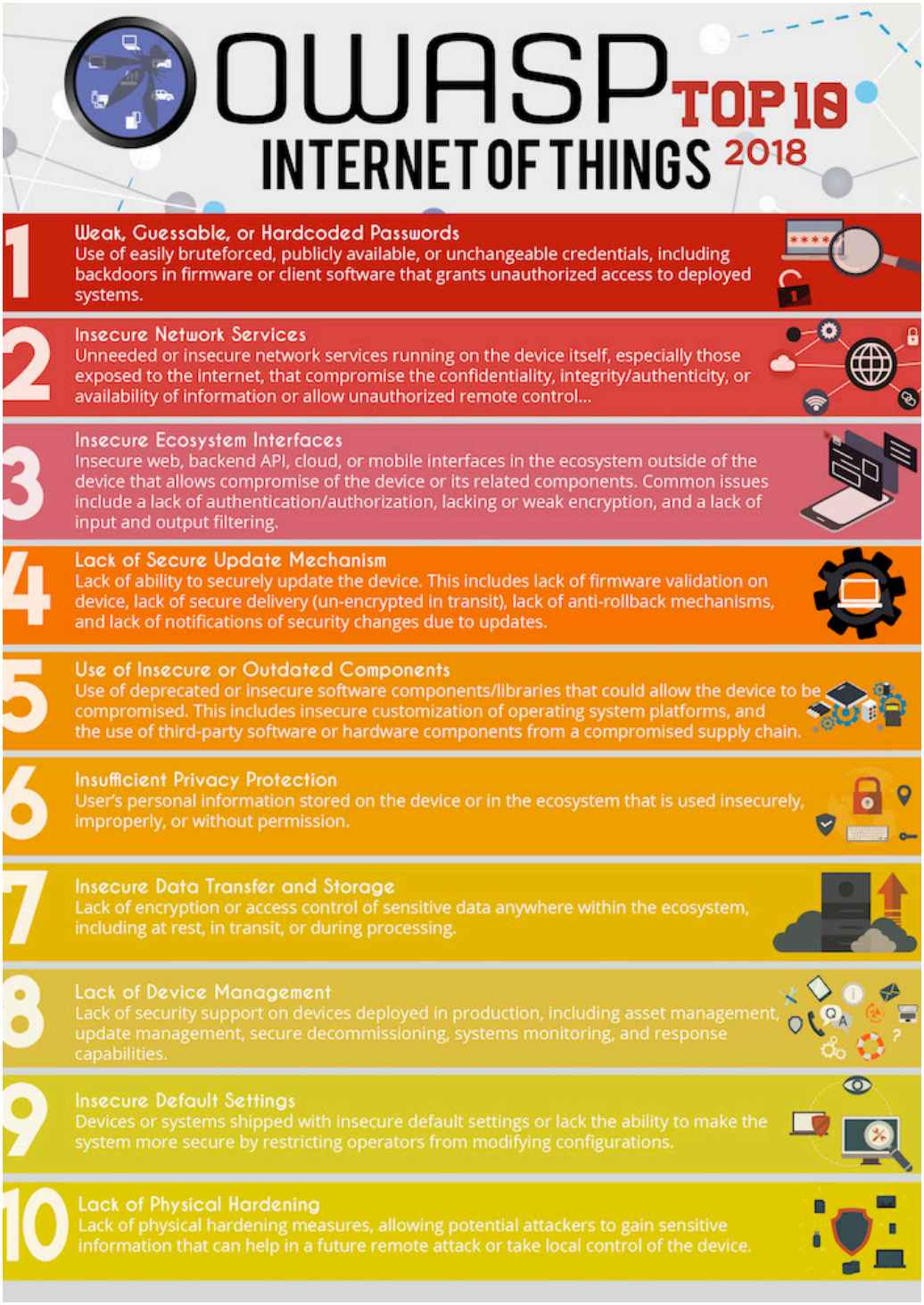

Figure 1. Top ten common pitfalls in IoT system design, deployment, and management [5].

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 26 July 2023 doi:10.20944/preprints202307.1781.v1

Most of the issues in the list are not strictly network related. For example, lack of physical hardening is not really a network issue, although access to a device can open the door to attacks on other systems. However, some issues are directly related to the network infrastructure and its use (e.g., insecure data transfer and insecure network services). The focus of this paper is to examine how an SDN-based network infrastructure can help secure IoT networks.

3. OpenFlow

The OpenFlow (OF) standards, introduced in [7], provide an implementation of the SDN concept. They define a switch specification that describes how an OF-compliant switch (packet forwarder) must behave, and a protocol that describes how a centralized controller can monitor and manage a collection of OF-compliant switches. OF has evolved from its original specification, adding many new features. The latest versions of these standards, published by the Open Networking Foundation, are defined in [8].

OF has enabled a large collection of research on networking technologies, see for example [9] (which focuses on datacenter networking). The purpose of this paper, as stated in the Introduction section, is to explore how OF can also help secure IoT networks.

Figure 2 depicts the logical view of an OF switch. Let us describe briefly its components:

1. The **OpenFlow channels** allow the switch to communicate, using the OF protocol over either TCP or TLS, with the controller or controllers.
2. The **Ports** are the physical input/output sockets where Ethernet-compatible devices are connected.
3. **Flow** and **Group Tables** are used to perform packet lookups and forwarding. A flow table consists of a set of flow entries, each of which is composed of match fields, counters, and a series of instructions to be applied to matching packets.
4. The **Meter Table** contains meters, a mechanism for performing basic QoS and traffic shaping operations.

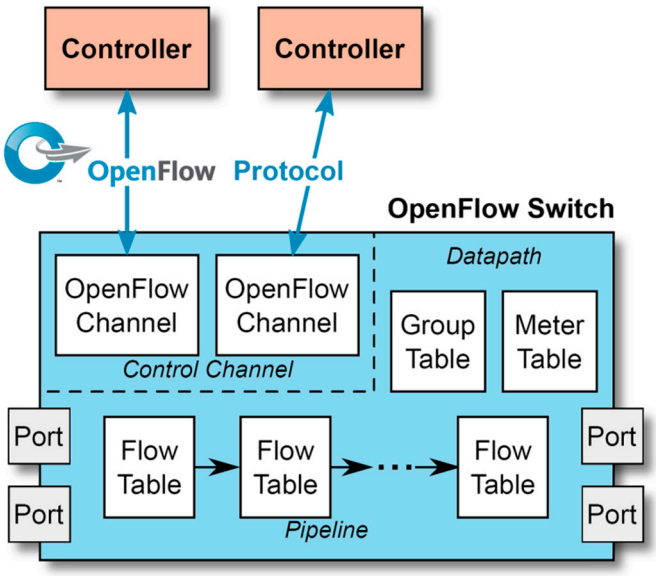

Figure 2. Logical sketch of an OF switch [8]. Note that the controllers are external entities that interact with the switch utilizing the OF protocol.

By utilizing the OF protocol, the controller has the capacity to efficiently manage flow and group tables, as well as meters, by adding, updating, and removing flow entries. These actions can be undertaken in a reactive manner, responding to the properties of incoming packets, or in a proactive manner, such as when configuring a new switch.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 26 July 2023 doi:10.20944/preprints202307.1781.v1

As packets enter an input port, the switch attempts to match their headers (which belong to different protocols) against the flow entries that have been set up. Matching begins with the first flow table and can be pipelined to additional tables. A basic example of a flow table is shown in Figure 3. For a given packet, if there is a match with a flow entry, the instructions linked to that entry are run. If more than one match is encountered, the entry with the highest priority is picked. If no match is identified in a flow table, the table-miss setup determines what happens. The packet may be transmitted to the controller via the OF channel, discarded, or passed on to the next flow table. A match specification may seek up matching values in a variety of sections (headers) or packets, including TCP/UDP, IP, ICMP, ARP, Ethernet, VLAN tags (802.1q) and others.

Match									Instructions
Port #	Src. MAC	Dst. MAC	Ethertype	Src. IP	Dst. IP	IP proto	Src. Port	Dst. Port	Actions
*	*	00:1f:ee:12:23:ac	*	*	*	*	*	*	Output 4
3	00:30:ab:25:ef:3d	00:1f:ee:12:23:ac	IP (0x0800)	192.168.1.2	10.0.0.1	TCP (6)	*	80	Drop
*	*	*	IP (0x0800)	*	10.0.0.0/16	UDP (17)	*	22	Output 3
*	*	*	ARP (0x0806)	*	*	*	*	*	ALL
*	*	*	*	*	*	*	*	*	CONTROLLER

Figure 3. Simplified example of an OF flow table, showing some entries with matching fields (headers of packets) and actions [9]. Each entry also has a priority and a list of counters that gather usage statistics, not shown. An "*" means "don't care".

Each flow entry's instructions either include actions or alter pipeline handling. Instructions convey actions that define modification and/or forwarding of packets, as well as group table processing. When the instruction set associated with a matched flow entry does not indicate a next table, pipeline processing ceases; at this point, the packet is normally deleted or altered and transmitted through a port. The port can be a physical one, a logical one (defined by the manager to implement link aggregation groups, loopback interfaces or tunnels), or a reserved one. Examples of reserved ports are:

- *CONTROLLER*: send the packet, using the OF protocol, to the controller, for further processing. This is commonly the port used in the event of no match.
- *ALL*: perform flooding, sending copies of the packet through all the output ports except the incoming one.
- *NORMAL*: in a switch that implements the usual Ethernet forwarding, use this process instead of the OF forwarding.

An action can also send packets to a group for further processing. Groups are collections of actions that implement flooding and other types of non-trivial forwarding operations, including fault-tolerant link aggregation, multipath forwarding, and fast rerouting. The meter table is made up of meter entries that allow OF performing rate-limiting, a basic QoS and traffic shaping action that limits a flow or a group to a specified bandwidth.

Table, flow, and port statistics maintained by the switches can be retrieved by a control application using the OF protocol. If there are problems in the switch (such as a physical port going down), the device will notify the controller. These two capabilities constitute the core of OF monitoring.

Despite the simplicity of the OF specification (especially the early versions; they are not simple anymore), it allows managers to implement complex behaviors in switches that go beyond the capabilities of traditional Ethernet switches. With appropriate control applications, an OF-enabled switch can perform these operations, alone or in combination: packet forwarding based on MAC addresses (Ethernet forwarding); packet forwarding based on IP addresses (IP forwarding); multipath routing; broadcast, multicast, anycast forwarding; link grouping with fault tolerance; NAT (Network Address Translation); bandwidth control for QoS management; traffic shaping; device isolation; network segmentation (firewalling) based on a variety of protocol headers; traffic redirection; detailed monitoring, etc.

Figure 4 represents the architecture of an SDN network particularized for OF. We already know the bottom part: the collection of switches that make up the Data Plane. These are centrally managed by an external piece of software, called the controller, via a Southbound interface: one or more

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 26 July 2023 doi:10.20944/preprints202307.1781.v1

management and monitoring protocols such as OF. The controller exposes SKDs and APIs that form its Northbound Interface and allow the implementation of the actual control applications, which are on the Application Plane. Southbound Interfaces are often standardized to work across hardware from different vendors. This is not the case for Northbound Interfaces.

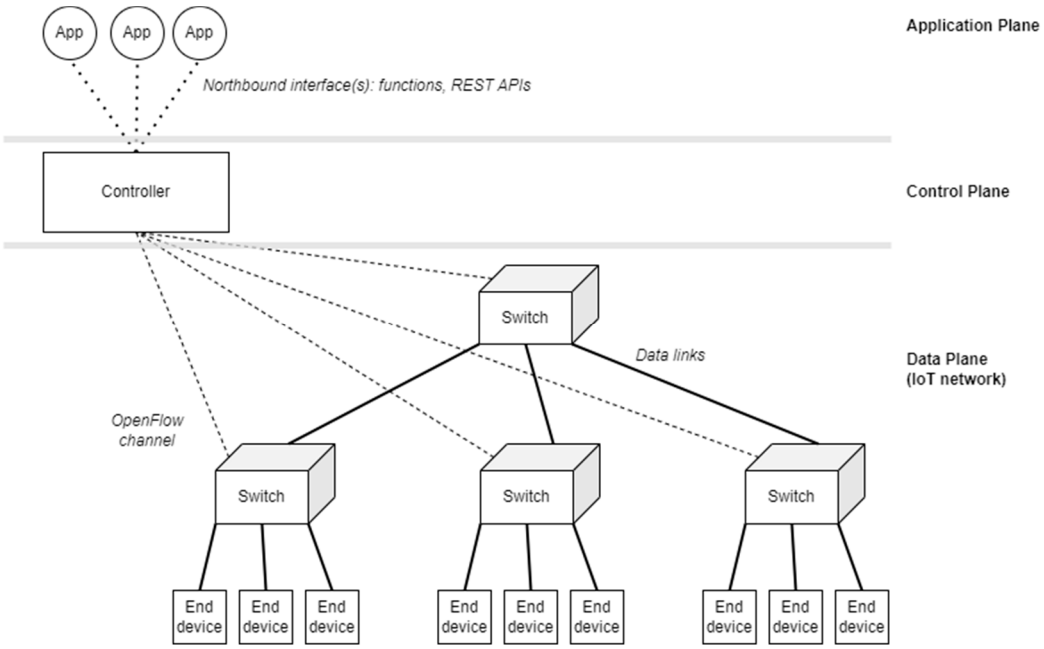

Figure 4. Architecture of an OpenFlow-based SDN network: planes and interfaces.

As mentioned above, OF is not the only southbound interface available. There are many other network management and monitoring interfaces and standards available. However, describing these alternatives is beyond the scope of this paper. It is also worth noting that not all switches in the Data Plane are hardware devices. In virtual environments, a popular option is Open vSwitch (OVS) [10], a powerful virtual switch with programmatic capabilities that allow, for example, network automation [11]. For management and monitoring, Open vSwitch implements OpenFlow, sFlow, and IPFIX.

4. SDN and Network Security

The previous section on the OF protocol and OF switches provided some guidance on implementing security-related functions. This section provides a more detailed description of techniques that can be used to strengthen network security in general (not limited to IoT).

- **Total blocking** of a device or a network segment. It is possible to insert flows with "drop" actions in one or more switches to completely isolate a part of the network. The device or segment can be easily reconnected by modifying the flow tables. Physical ports, VLAN tags, MAC addresses and IP addresses/masks can be used to identify the device(s) to be blocked or authorized.
- **Bandwidth limitation**. The meter feature can be used to limit the bandwidth associated with a flow. For example, if a device has been compromised and is being used in a botnet, a meter can drastically reduce its attack potential without completely blocking the connection.
- **Restricted communication**. Using IP addresses/masks or VLAN tags, different network segments can be configured and communication between these segments can be tightly controlled: only packets belonging to some specific source-destination pairs are allowed, other packets are dropped. The granularity of the segments can be coarse (network-based) or fine (device-specific).
- **Port-based protection**. Traffic to a specific TCP/UDP port (i.e. service) can be easily controlled. Ports can be "opened" and "closed" as required by inserting/removing flow entries in the switches, without the need to manage the protected device. Note that we are referring to transport layer (TCP/UDP) ports, not the physical ports of a switch.

 doi:10.20944/preprints202307.1781.v1

- **Traffic redirection**. By installing the right flow entries and rewriting packet fields (such as VLAN tags, MAC addresses or IP addresses), it is possible to redirect specific classes of traffic to devices that perform specialized network functions. Note that when these functions are implemented in virtual machines or containers, we have *network function virtualization*, a term often associated with SDN. For example, a procedure to authenticate and authorize a device could be offloaded to a specific virtual network function (VNF). If authorized, the controller installs the necessary flows. If not, the device is blocked. Similarly, traffic (or a copy of it) can be redirected to an intrusion detection system.
- **Monitoring for security**. A control application can collect information from all devices and maintain control of port usage and flow entries. This information can feed visualization panels and anomaly detectors, giving the network manager both global and detailed views of the health of the network.

As discussed, with the right control applications, a network built with OF switches (or other similar SDN technologies) can implement multiple functionalities without the use of specialized equipment. Network capabilities can also be enhanced through the use of VNFs.

5. Securing the Communications of IoT devices

A team of the OWASP foundation's Internet of Things project is working in a document called "The OWASP Internet of Things Security Verification Standard" (ISVS) [12]. It is defined as "*a community effort to establish an open standard of security requirements for Internet of Things (IoT) ecosystems*". These requirements are intended to be used at different stages of the lifecycle of IoT ecosystems, including design, development, and testing.

5.1. The ISVS

The ISVS document is structured into an introductory chapter "Using the ISVS" and five chapters covering specific security requirements. These requirements can be represented as a stack. From bottom to top, these are the layers (see Figure 5):

- **V5**: Requirements for the **Hardware Platform**, the physical implementation of the IoT device.
- **V4**: **Communication** requirements.
- **V3**: Requirements for the **Software Platform**, which together with the Communication Layer provides interfaces to develop rich applications to run in the IoT device.
- **V2**. **User Space Applications** requirements.
- **V1**. Requirements for the **Ecosystem**, that is, the interactions of the IoT device with surrounding devices and services.

The document also describes three levels of security checks. **Level 1** is the least stringent; requirements at this level are intended to protect against software-only threats (attacks that do not require physical contact with the device's hardware). **Level 2** requirements aim to protect against attacks that target the hardware. Finally, **Level 3**, the most stringent, lists requirements for devices that must never be compromised.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 26 July 2023 doi:10.20944/preprints202307.1781.v1

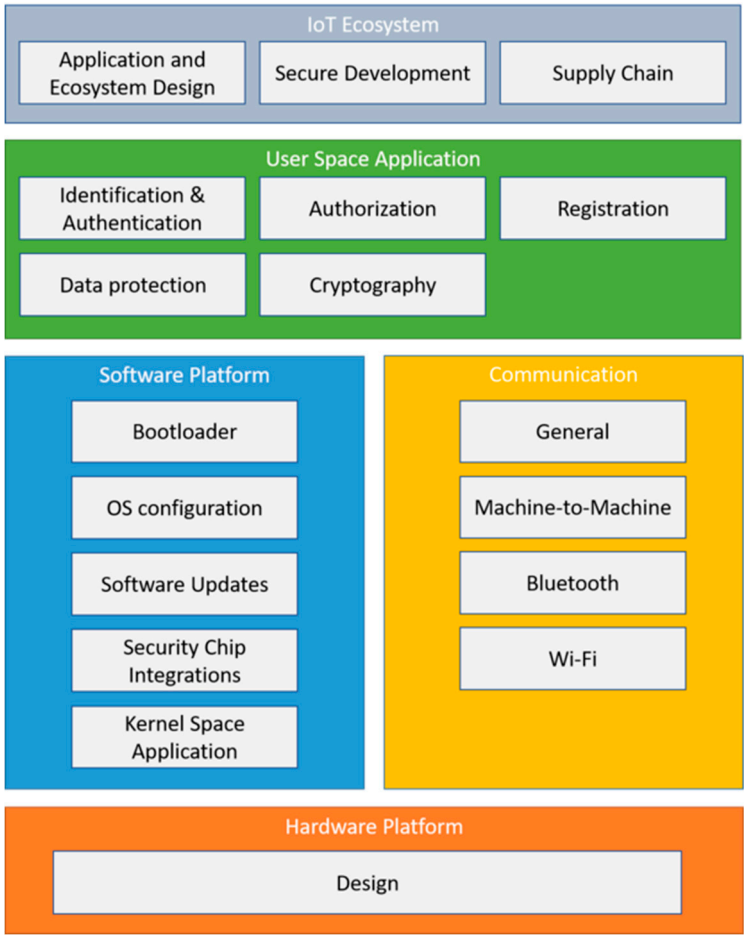

Figure 5. Layers of the ISVS [12].

Our focus is on **Layer V4: Communication**. It is assumed that an IoT device exchanges data and commands with its ecosystem (other devices, edge/cloud systems). Trust between parties is essential, so authentication of parties, communication integrity and confidentiality (to prevent data exfiltration) must be ensured. This requires the use of the latest and strongest security protocols and cryptographic algorithms, with appropriate configuration settings. The industry relies on standards such as TLS, Bluetooth and Wi-Fi, so the configuration of these protocols must be reviewed frequently to ensure their effectiveness in providing secure communications. In summary, the ISVS recommends:

- Use TLS or equivalent strong encryption and authentication, no matter the perceived sensitivity of the information that is exchanged.
- Implement certificate-based mutual authentication mechanisms.
- Use appropriate configurations to set the preferred collection of cryptographic algorithms and ciphers when establishing a communication channel. Disable obsolete or known insecure options.
- Use the latest security standards with the strongest security settings for Bluetooth and Wi-Fi connections.

Communication requirements are grouped into four blocks (general, machine-to-machine, Bluetooth, Wi-Fi) and assigned to the appropriate threat level(s). They are summarized in Appendix A. Note the strong focus on cryptographic protocols for confidentiality, authentication and data integrity. Note also that the type of IoT networking described fits best into a specific class of IoT systems: those in which devices are connected to the Internet, directly or through a gateway, with minimal or no device-to-device interaction.

In this context, the responsibility for meeting the listed requirements lies primarily with the hardware, firmware and software implemented in the devices. What can an SDN network do to help

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 26 July 2023 doi:10.20944/preprints202307.1781.v1

meet these requirements? It can help by providing protocol extensions through external support. For example, if a device cannot be updated to use the latest cryptographic protocol, a proxy VNF can be set up and traffic redirected through it. Similarly, authentication and authorization functions, such as those involving the use of certificates, may be offloaded to a VNF.

5.2. Beyond the ISVS

As stated before, the ISVS view of networking is very device-centric. In a network of IoT devices, for example in an industrial IoT environment, additional communication security capabilities beyond cryptographic confidentiality and authentication are required, and SDN can help implement them.

5.2.1. Network Segmentation

This is a fundamental requirement in industrial IoT networks [13] to control device-to-device, device-to-edge, device-to-cloud and edge-to-cloud communications. As explained in the previous section, per-device/network rules can be used to control both incoming and outgoing communications, for example restricting access to external servers, access from external clients, and the use of unauthorized transport ports. OpenFlow can work with VLAN tags, so they can be part of a segmentation policy.

5.2.2. Access Control

The ISVS requirements focus on end-to-end authentication-authorization. Finer-grained rules can be implemented to allow a device to access the network (as a whole) or some specific segments. An SDN solution can easily implement forwarding and filtering rules based on ports and/or MAC addresses. MAC addresses can be spoofed internally, but not easily from outside the IoT network. The SDN controller can maintain and enforce a list of device pairs that are allowed to communicate.

5.2.3. Network Monitoring

It is important to be able to control the status of the network and the devices connected to it from a centralized console. A control application can use OpenFlow (or other protocols) to monitor the network activity of all devices, looking for unexpected patterns. Alarms can be raised if a device is accessing a device or service for which it is not authorized, if a device is generating or consuming an unusual amount of data, if packets with unexpected protocols/ports/destinations are being injected into the network, and so on. Note that threats do not always come from external devices: a compromised internal device can be used to attack other devices or the network as a whole.

5.2.4. Device Vigilance and Isolation

Once a device has been identified as having suspicious communication behavior, a number of actions can be taken, including (but not limited to):

- Completely or partially block the device (drastically reducing its allowed use of bandwidth) to reduce the risks associated with a potential attack.
- Divert the traffic to a device (or VNF) to carry out a deeper analysis of what is going on (intrusion detection)

6. Implementation

The ideas outlined in this paper can be implemented using one or more control applications developed using the northbound interface of an OpenFlow controller (as discussed in Section 3). This is not a straightforward task and can be greatly simplified by using a readily available controller such as Faucet [14, 15]. This controller uses OpenFlow SDN technology (v1.3) and supports a wide range of switches (from Allied Telesys, HPE-Aruba, Lagopus, Cisco, etc.). The list includes the software-based OVS (Open vSwitch). Faucet is built on top of Ryu [16], an open-source, Python-based SDN framework to build control applications.

 doi:10.20944/preprints202307.1781.v1

The configuration of Faucet is done by beans of YAML configuration files. Through the right configurations (the documentation is available in [17]), the network manager can:

- Set up the properties of "datapaths" (OF switches)
- Define VLANs
- Enable 802.1X authentication
- Define allowed VLAN-to-VLAN forwarding, and other IP-based forwarding rules (routing)
- Deploy BGP
- Deploy meters, that are associated to ACLs (see below)
- Use Access Control Lists (ACLs) that not only implement block/allow rules. They can also
    - Rewrite protocol fields, including VLAN tags
    - Forward a packet to several ports (to implement anycast, multicast, broadcast)
    - Define a failover output port
    - Define tunnels
    - Redirect traffic to VNFs

Clearly, the most distinctive feature of Faucet is the ability to define complex ACLs, which allows the implementation of most of the mechanisms described in Section 5 for securing IoT networks.

Faucet comes with a companion powerful monitoring tool called Gauge, which is also configured using YAML files. Gauge can monitor at different levels of granularity (switch, port, flow) and export the captured data to real-time monitoring environments (such as Prometheus and InfluxDB) or log files. With these tools, it is possible to have a constantly updated view of the network, store data for future use, perform sophisticated checks on the captured data (e.g. for anomaly detection), set alerts, etc. Gauge is therefore the tool that can be used to implement the monitoring and vigilance requirements set out in Section 5.

7. Discussion and Conclusions

In this paper, we have reviewed the security-related communication requirements of IoT devices/networks, using the OWASP IoT project as the main reference. We then explained the basics of SDN, focusing on one particular implementation of this concept: OpenFlow. This paves the way for answering the question: what can SDN do to implement more secure IoT communications? We have seen that SDN-OF can really help by enabling the implementation of features such as (1) comprehensive network monitoring; (2) implementation of different segmentation policies based on services, devices, VLANs, IP addresses... or any combination of these fields; and (3) redirection of traffic, when needed, to specialized devices or VNFs to implement additional actions, such as intrusion detection, using a single type of device (OF-enabled switches). These are not theoretical ideas: we have been careful to discuss functionality that can be implemented using an OF controller, such as the open source Faucet.

Funding: This work has received support from the following programs: PID2019-104966GB-I00AEI (Ministerio de Ciencia e Innovación, Spain), IT-1504-22 (Basque Government), KK-2022/00106 (Elkartek project financed by the Basque Government).

Conflicts of Interest: The authors declare no conflict of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript; or in the decision to publish the results.

Appendix A. ISVS V4: Communication Requirements

As explained in Section 5.1, the OWASP IoT Security Verification Standard (ISVS) aims to specify an open standard for security requirements for IoT ecosystems [12]. In the ISVS layered architecture, the communication requirements form layer V4. They are divided into four groups, which are presented in the following subsections. The numbers in parentheses indicate the threat level to which they apply.

A.1. General Requirements

- Communication with other components of the ecosystem must be over a secure channel that guarantees the confidentiality and integrity of the data, using a protocol that prevents replay attacks (L1, 2, 3)
- If TLS is used, it must use FIPS-Based cipher suites and the device must verify the certificate (L1, 2, 3)
- Detection or protection against jamming must be provided (L2, 3)
- The implementation of TLS must use its own certificate store, and must pin to the endpoint certificate or public key. Connection with endpoints with different certificates or keys must not be allowed, even if signed by a trusted CA (L2, 3)
- Inter-chip communications must be encrypted (L3)

A.2. Machine-to-Machine Requirements

- Unencrypted communication must be limited to non-sensitive data and instructions (L1, 2, 3)
- Shared secrets required for encrypted communication must not use hardcoded keys (L1, 2, 3)
- MQTT brokers must only allow operations with authorized devices (L1, 2, 3)
- MQTT transactions should use certificate-based authentication, instead of username/password pairs (L1, 2, 3)

A.3. Bluetooth Requirements

- Pairing and discovery must be disabled, except when necessary (L1, 2, 3)
- PIN or PassKey codes must not be easily guessable (L1, 2, 3)
- For devices with old versions of Bluetooth, pairing must require a PIN (L1, 2, 3)
- For devices with modern versions of Bluetooth, SSP authentication must use at least 6 digits (L1, 2, 3)
- Encryption keys must use the maximum allowable size (L1, 2, 3)
- The most secure pairing methods should be used: OOB, Numeric Comparison or PassKey Entry, depending on the capabilities of the device (L1, 2, 3)
- The strongest security mode and level supported by the device must be used (L1, 2, 3)

A.4. Wi-Fi Requirements

- Wi-Fi communications must be disabled unless necessary: devices that do not require a connection or that have a different type of connection (such as Ethernet) should not use Wi-Fi (L1, 2, 3)
- WPA2 or higher security protocols must be used (L1, 2, 3)
- If WPA is used, AES encryption must be selected (L1, 2, 3)
- WPS must be disabled (L1, 2, 3)

References

1. Wikipedia. *Internet of Things*. Available online: https://en.wikipedia.org/wiki/Internet_of_things (accessed on 19 July 2023).
2. Peterson, L et al. *Software-Defined Networks: A Systems Approach*. Publisher: Systems Approach LLC, 2022. Available online: https://sdn.systemsapproach.org/index.html (accessed on 19 July 2023)
3. Open Networking Foundation Web Site. Available online: https://opennetworking.org/ (accessed on 19 July 2023).
4. OWASP Foundation Web Site. Available online: https://owasp.org/ (accessed on 19 July 2023).
5. OWASP Internet of Things Project. Available online: https://wiki.owasp.org/index.php/OWASP_Internet_of_Things_Project#tab=Main (accessed on 19 July 2023).
6. IoT Top 10 2018 Mapping Project. Available online: https://wiki.owasp.org/index.php/OWASP_Internet_of_Things_Project#tab=OWASP_IoT_Top_10_2018_Mapping_Project (accessed on 19 July 2023).
7. McKeown, T et al. *OpenFlow: Enabling innovation in campus networks*. ACM SIGCOMM Comput. Commun. Rev. 2008, vol. 38, no. 2, pp. 69-74. DOI: https://doi.org/10.1145/1355734.1355746

8. Open Networking Foundation. *OpenFlow Switch Specification Version 1.5.1*. Available online: https://opennetworking.org/software-defined-standards/specifications/ (accessed on 19 July 2023).
9. Miguel-Alonso, J. *A Research Review of OpenFlow for Datacenter Networking*. IEEE Access 2023, 11: 770-786. DOI: https://doi.org/10.1109/ACCESS.2022.3233466
10. Linux Foundation. Open vSwitch project site. Available online: https://www.openvswitch.org/ (accessed on 19 July 2023).
11. Pfaff, B et al. *Extending Networking into the Virtualization Layer.* In Proceedings of ACM Workshop on Hot Topics in Networks 2009. Available online: https://www.openvswitch.org/support/papers/ (accessed on 19 July 2023).
12. OWASP IoT Security Verification Standard (ISVS). Available online: https://github.com/OWASP/IoT-Security-Verification-Standard-ISVS (accessed on 19 July 2023).
13. Juniper Networks. *IoT Network Segmentation*. Available online: https://www.juniper.net/content/dam/www/assets/solution-briefs/us/en/iot-network-segmentation.pdf
14. Bailey, J; Stuart, S. *Faucet: Deploying SDN in the Enterprise: Using OpenFlow and DevOps for rapid development*. ACM Queue (2016), vol. 14, no. 5, pp. 54-68. DOI: https://doi.org/10.1145/3012426.3015763
15. Faucet Web Site. Available online: https://faucet.nz/ (accessed on 19 July 2023).
16. RYU Project Team. *RYU SDN Framework, Release 1.0*. Available online: https://ryu-sdn.org/resources.html#books (accessed on 19 July 2023)
17. Faucet Documentation. Available online: https://docs.faucet.nz/en/latest/ (accessed on 19 July 2023).