Configuration vulnerability in SNORT for Windows Operating Systems

[1]Luca Faramondi, [1]Marta Grassi, [1]Simone Guarino, [1]Roberto Setola, and [2]Cristina Alcaraz

[1]Unit of Automatic Control, University Campus Bio-Medico di Roma,

Via Alvaro del Portillo 21, Rome, Italy 00128

[2]Computer Science Department,University of Malaga,

Campus de Teatinos s/n, 29071, Malaga, Spain

Abstract

Cyber-attacks against Industrial Control Systems (ICS) can lead to catastrophic events which can be prevented by the use of security measures such as the Intrusion Prevention Systems (IPS). In this work we experimentally demonstrate how to exploit the configuration vulnerabilities of SNORT one of the most adopted IPSs to significantly degrade the effectiveness of the IPS and consequently allowing successful cyber-attacks. We illustrate how to design a batch script able to retrieve and modify the configuration files of SNORT in order to disable its ability to detect and block Denial of Service (DoS) and ARP poisoning-based Man-In-The-Middle (MITM) attacks against a Programmable Logic Controller (PLC) in an ICS network. Experimental tests performed on a water distribution testbed show that, despite the presence of IPS, the DoS and ARP spoofed packets reach the destination causing respectively the disconnection of the PLC from the ICS network and the modification of packets payload.

Keywords: Digital Twin Network, 6G, Industrial Internet of Things, Industry 5.0, Cybersecurity.

1 Introduction

Critical infrastructures are the physical and cyber systems so vital for the wellness of population to the point that their destruction or disability would generate an impairing impact in terms of injury to people, environmental pollution or physical damage to equipment. In this framework, Industrial Control Systems (ICS) can be considered as the "core" of such infrastructures where the Programmable Logic Controllers (PLC) are the most used interface between cyber domain and the physical plant [1]. ICSs are largely employed in several fields such as water treatment [2], smart grids [3], oil and natural gas pipelines [4], chemical process control [5], power generation and distribution systems [6].

Actually, ICSs have increased interaction with the Internet to exploit its wide range of valuable functions in order to support business needs, allowing remote control and facilitate intelligent decision making. As a result of the strengthening connectivity, ICSs became vulnerable to cyber attacks to the point that on the dark Web there are several tools and information to support malicious actions against ICS [7], [8].
The research of a specific and well-defined software or hardware configuration to be infected is often one of the first steps for an attacker. An authoritative example is Stuxnet which was studied to sabotage Iranian Natanz uranium enrichment base in 2010. The goal was to infect a specific PLC (Siemens Simatic S7-300) dedicated to the control of centrifuges for separating nuclear material after taking advantage of zero-day vulnerabilities in Windows Operating Systems [9].
Later on, in 2015, the malware BlackEnergy3 has been used to attack the Ukrainian power grid, causing circuit disconnections and a large-scale blackout [10].
The relevance of cyber risk imposed to Critical Infrastructure operators' the adoption of specific security initiatives in order to protect their ICSs with respect to malicious actions and overcome the old (and ineffective) security by obscurity strategy [11]. In this framework a cornerstone element of such strategies is the use of an Intrusion Prevention System (IPS), i.e, a system designed to monitor real time network flow in order to filter and block not-legitimate traffic [12]. To this end, IPSs use information and models about the legitimate flow to label eventually deviations as abnormal, hence an alert is raised. Due to its effectiveness, IPSs itself become attractive targets for malicious attackers, which make use of several types of attacks as described in [13] and [14]. It is important to realize that IPSs are software programs based on a set of default parameters, following a specific policy, and thus subjects to vulnerability, by definition [15]. However, from the best of our knowledge, a studying model that exploits this kind of configuration vulnerability against IPSs is still missing.
In this paper, IPSs' vulnerabilities, such as readable and writable plain-text configuration files, are inquired. We focus on SNORT [16], the foremost IPS, considering Windows environment as it is the most used OS in ICSs [17]. With the aim to demonstrate the relevance of such vulnerabilities in Windows OS, we elaborate a batch script able to exploit configuration vulnerabilities in order to disable the ability of SNORT to detect and reject Denial of Service (DoS) attacks and ARP poisoning in an ICS network. Such script works in a stealth manner without compromising the proper functioning of Windows OS or enabling host remote control. In this way, an attacker can only compromise Snort ability to detect specific attacks making the system vulnerable to these kinds of threats. Specifically, such a strategy can be considered as the preliminary step for a destructive cyber attack as shown by experimental tests where undetected DoS and ARP poisoning-based Man-In-The-Middle (MITM) attacks induce respectively the disconnection of PLC from the ICS network and the redirect to the attacker of legitimate traffic. In this second case, we show that the attacker can stealthily modify packets payload changing tanks water level values sent to

the Human Machine Interface (HMI) by the PLC.

The paper is organized as follows. Section 2 briefly describes some preliminary aspects about IPSs, DoS attack and ARP poisoning hacking technique. Section 3 focuses on essential aspects of SNORT configuration vulnerabilities. Section 4 presents the steps for the implementation of the batch script. Section 5 analyzes the experimental setup while Section 6 reports experimental results and possible countermeasures. Some conclusive remarks are collected in Section 7 which concludes the paper.

2 Preliminaries

2.1 Intrusion Prevention Systems: SNORT

SNORT is one of the most important signature based IPSs on the market [18]. It basically verifies if packets taken from the network to be protected are malicious or not relying on a list of rules written by the user [19]. If a packet matches a particular rule, SNORT returns an alarm and eventually blocks the suspecting traffic and log the event in an output file.
The syntax of a SNORT rule is reported below [20]:

```
[action][protocol][sourceIP][sourceport]
->[destIP][destport]([Rule options]).
```

At beginning, the rule defines the *action* to be executed when a packet matches the rule criteria defined in *options*. The three main actions are: *alert* which generates an alert and logs the packet in a file; *drop* which blocks and logs the packet and *reject* which blocks the packet, logs it, and then sends a TCP reset if the protocol is TCP or an ICMP port unreachable message if the protocol is UDP.
The rule distinguishes packets in terms of the protocol and the source and destination IP and MAC addresses. Finally, some options can be added in order to enable more power and flexibility in the detection, searching, for example, for specific bytes in the packet payload. When a rule fires, an alert is triggered and the event is logged in an output file.

Rules can be listed in order to detect many attacks such as *scanning attacks*, *ARP poisoning* and *Denial of Service attacks*. In this article we focus on the rules adopted to detect and prevent DoS attack and ARP poisoning; however, the approach can be easily generalised in order to prevent different types of attack.

2.2 DoS attack

As reported by Kasperky ICS CERT in 2019 [21], 14.6% of the vulnerabilities identified in an industrial control system could lead to a DoS attack. If not effectively recognized, this kind of attack can cause the disabling of legitimate accesses to the target system compromising its availability [22]. An attacker can carry out a DoS attack sending to the victim a huge number of packets in a small time slot: the victim saturates its resources compromising its functioning. There are several types of DoS attacks such as *Smurf attack* in which the attacker sends ICMP packets and *SYN flood*, in case of TCP packets [23].

2.3 Detection of SYN flood using SNORT

As stated above, one of the most common DoS attack is the SYN flood attack. In this view, a SNORT rule can be set to recognize and avoid SYN flood attacks against a particular victim in the network. As described in [24], the rule has to identify if hosts have received an high number of packets in a small time period (for example three seconds). According to the syntax presented in Section 2.1, an example of a SNORT rule for the identification of SYN flood attacks is reported below:

```
reject tcp any any -> $DST_IP_ADDRESS $DST_PORT (msg :" TCP flood
     ";sid:1000003; rev :1; detection_filter : track by_dst,
    count 5000 , seconds 3;)
```

This rule is able to detect a SYN flood attack if the victim, identified by the ip address $DST_IP_ADDRESS, receives on a specific port, identified by $DST_PORT, more than 5000 TCP packets in a time period of 3 seconds from any host of any network. The only way to provide this behavior is the use of the option *detection_filter* with its fields *count* and *seconds*. In this case, the rule tracks the packets in relation to their destination regardless of their source. The result is the rejection of these packets with the closing of the TCP connection.

2.4 ARP poisoning

ARP poisoning is an hacking technique that exploits leak of authentication in the ARP protocol which is widely used to resolve Internet layer addresses to link layer ones in a switched LAN. The main goal is to modify IP and MAC addresses pair information saved in the hosts ARP tables. To this end, the attacker sends spoofed ARP reply messages onto a local area network in order to associate the attacker's MAC address with the IP address of another host, such as the default gateway, causing any traffic meant for that IP address to be sent to the attacker instead [25].

2.5 Detection of ARP poisoning using SNORT

The detection of ARP poisoning requires SNORT to compare Ethernet addresses with those defined in the ARP packets. To this end, a simple detection rule is not sufficient: a specific SNORT component, called preprocessor, has to be used. A preprocessor is a type of plugin which enables SNORT to manipulate traffic patterns in order to defeat those attacks which cannot be detected by signature

matching via simple detection rules. Specifically, as described in Section 2.1, SNORT detection engine checks if a single aspect of one packet, like the protocol or a particular value in the payload, verifies criteria defined in a SNORT rule [26]. On the contrary, preprocessors can handle data stretched over multiple packets and can perform more complex operations like TCP stream reassembly, IP defragmentation and comparing fields of multiple protocols in a single packet [27].
The `arpspoof` preprocessor is able to detect an ARP poisoning attack by determining if source MAC address is different from the one included in the ARP message, indicating a spoofed reply.
This preprocessor has to be activated in SNORT configuration file by adding the following line:

```
preprocessor arpspoof
```

Actually, in order to enable preprocessor events and to define the type of action to be executed when an ARP spoofed packet is detected, specific rules are required. `Arpspoof` preprocessor rules are predefined in SNORT and are uniquely identified by the `gid` keyword number 112. An example is reported below.

```
reject ( msg: "ARPSPOOF_ETHERFRAME_ARP_MISMATCH_SRC"; sid: 2; gid
    : 112; rev: 1; metadata: rule-type preproc ; classtype:bad-
    unknown; )
```

Notice that these rules define only the action (*reject*) without performing the detection which is previously executed by the preprocessor. Without these specific rules, even if `arpspoof` preprocessor is still working, malicious packets will not be dropped and no event will be shown compromising SNORT effectiveness.

Thus, the result is the rejection of ARP spoofed packets which will be logged in a `csv` extension file as described in Section 2.3.

3 Analysis of SNORT configuration vulnerability

This section investigates configuration vulnerability [28, 29] about IPSs, focusing in particular on SNORT installed on Windows Operating Systems.

The main vulnerability of SNORT for Windows environment is strictly related to the installation phase and the configuration of default parameters of the tool. This vulnerability is properly defined as *configuration vulnerability* and in SNORT is due to: readable and writable plain-text parameters files and predefined folders structure.
By default, in SNORT, all files, including configuration ones, can be easily read and written due to the lack of any type of security measure, such as encryption or authentication.
Moreover, these files are always located in the same sub-folder shown in Figure 1. By default, the main configuration file, `SNORT.conf`, is located at `C:\SNORT\etc\`.

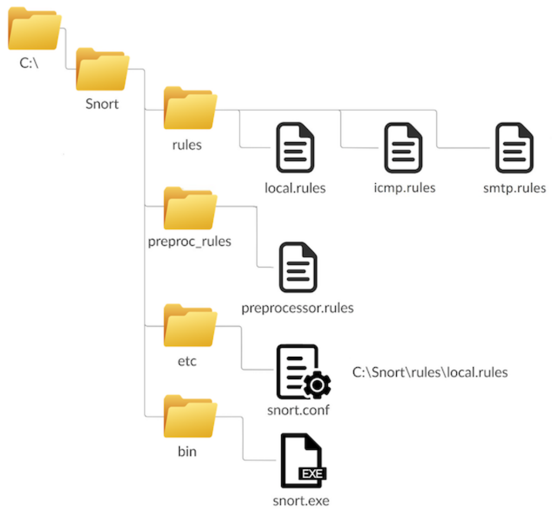


Figure 1: SNORT folder tree

It is used to define the correct functioning of SNORT and contains both configuration of active preprocessors and paths of all support files. In particular, it contains absolute paths of rules files and of specific-preprocessors rules files which are located respectively at `C:\SNORT\rules\` and at `C:\SNORT\preproc_rules\`. Files located in `rules` folder, such as `local.rules`, `icmp.rules` and `smtp.rules`, contain the default and custom list of rules which define how SNORT identifies malicious events in network traffic. Thus, as described in Section 2.3, they are fundamental for attack detection and prevention. Files located in `preproc_rules` folder, such as `preprocessor.rules`, contain the default rules specific for SNORT preprocessors. As described in Section 2.5, they are essential to enable preprocessor events and to define the action to be executed when malicious packets are detected by the preprocessor itself.
Unfortunately, despite their critical role, an attacker can change or disable specific rules because rules files are readable and writable plain-text. Therefore, on the basis of these information about default configuration parameters, an attacker is able to create a sequence of malicious instructions, described in Section 4, which can degrade SNORT effectiveness in detecting one or multiple attacks. Thus, the neglect and the avoidance of security choices in initial configuration phase can lead to significant consequences in terms of possible attacks which will be stealthily launched against the vulnerable network.

4 Batch Script for SNORT Configuration Vulnerabilities Exploitation

In this Section, we describe the fundamental steps to develop a batch script to manipulate SNORT configuration files in order to degrade its effectiveness. Notice that the proposed batch script is based exclusively on the knowledge of default parameters and configuration of SNORT. The proposed procedure exploits SNORT configuration vulnerability in terms of knowledge of default directory structure and the presence of readable and writable plain-text files. We assume that the batch script is accidentally run by the user who has Administrator privileges on the OS, e.g. in the presence of phishing attack via email link as in the case of BlackEnergy3 [30] or via USB flash drive as it was for Stuxnet [31].
The main objective of the script is to corrupt SNORT rules files. In particular, we want to identify and disable the rules devoted to the recognition of DoS SYN flood attack and of ARP poisoning in order to make the system vulnerable to these kinds of threats. Moreover, the script has been implemented in order to work in a stealth manner without leaving traces of its activity according to the following requirements:

- **Filtering process**: SNORT will regularly work in an unchanged fashion except for its ability to manage DoS and ARP poisoning attacks.
- **Resource access:** there will be no traces of the script effects. To this end, after rules files corruption, the last modification date and the file size are set to the values prior to the attack.
- **Background execution:** the user will not have any visual effect of the execution of malicious code.
- **Restoration attempt:** the rules files will be corrupted at any restart of the Operating System despite the user's attempt to restore them.

Notice that, the script is developed to corrupt and disable specific rules related to the filtering of the DoS and ARP poisoning traffic exclusively, without stopping the entire filtering process or modifying the rest of the rules or preprocessors that define the behaviour of SNORT.

We now describe in detail the actions performed by the proposed script and its architecture.
A main batch file with Powershell ISE support has been developed for the implementation of the entire project. A batch file is a type of file in Microsoft Disk Operating System (MS-DOS) containing a sequence of commands executed in the command prompt one by one, in the exact order they are written in a file.

As illustrated in Figure 2, the batch file is being carried out in three phases. At first, the script performs a dynamic research on victim's entire PC, looking for the presence of SNORT installation files. Once they are found, it saves

`SNORT.conf` allocation path; in fact, even if this file is typically present in the directory `C:\SNORT\etc\` folder, it could still be placed elsewhere by the user. In this way, the batch script is unaffected by any sort of modifications in terms of file allocation made by the user.

During the second phase, the script reads SNORT configuration file searching for the path of all rules files which are included during SNORT execution, such as `local.rules`, `icmp.rules` and `preprocessor.rules` described in Figure 1. In this way, the attacker knows the path of rules files which are used by SNORT to inspect network traffic. Typical file inclusion lines in `SNORT.conf` are reported below:

```
include $RULE_PATH/icmp.rules
include $RULE_PATH/local.rules
include $RULE_PATH/smtp.rules
include $PREPROC_RULE_PATH/preprocessor.rules
```

where, `$RULE_PATH` and `$PREPROC_RULE_PATH` are variables whose values are the absolute paths of rules files and preprocessors rules files folders respectively. Thus, the script seeks all lines which include a file name with the extension `.rules` and the word `include`.
In the third and last phase, the main batch file is ready to corrupt rules files: this procedure follows four main stages.
During the first one, rules files are corrupted in order to disable specific rules which detect SYN flood DoS attacks and ARP poisoning. For each selected rules file, a keyword research is performed. According to the syntax of a SNORT rule reported in Section 2.1 and as described in Section 2.3, the only way to detect SYN flood DoS attack is to use the rule option `detection_filter`; thus, the batch file will search for all rules containing this keyword. If it is present, the rule is disabled adding, at the beginning of the line, the comment symbol #.

On the contrary, as described in Section 2.5, preprocessor rules are uniquely identified by the keyword `gid` number 112; thus, the batch file will comment all rules which contain this identification number. Recall that, in this way, `arpspoof` preprocessor is still working but, disabling its rules, SNORT will not be able to show alerts and to block effectively ARP spoofed packets.
During the second stage, the batch file is used to restore the last modification date of rules files. In this way the corruption procedure does not leave any temporal traces on such files.
The third stage ensures that the size of the corrupt rules file remains unchanged. The number of added characters is considered in order to keep the total number unchanged; thus, the file size is restored to the value prior to the attack.
The fourth and last stage ensures the restoration attempt requirement. It creates a shortcut of the main batch file inside the Windows "StartUp" folder, which can be found with a parametric research of the path in the victim's PC. Creating the shortcut in this specific folder ensures that script automatically runs every time victim restarts or shuts down the operating system, without typing or selecting any commands. It also ensures that if victim has restored rules files previously corrupted by the script, these will be compromised again.

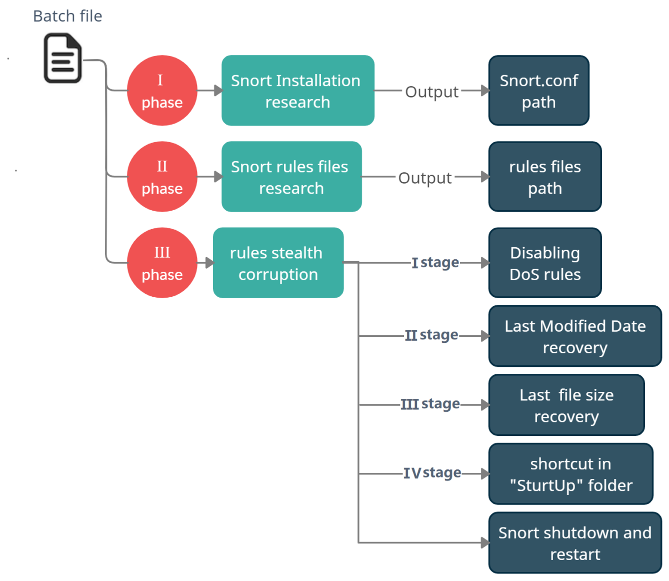


Figure 2: Script actions on victim PC

Finally, the batch file performs an automatic shutdown of SNORT process followed by an automatic restart of such process on all active network interfaces, ensuring the correct compromise of SNORT which is now unable to recognize DoS attacks and ARP poisoning due to the corrupted rules file.

5 Experimental set-up

5.1 WDT Testbed

The effectiveness of the proposed attack strategy has been experimentally evaluated against the ICS in the WDT (Water Distribution Testbed) testbed [32] which reproduces a real plant consisting of 5 tanks, 20 solenoid valves and 4 pumps which control the water passage (see Figure 3). Water level in tanks is measured by pressure sensors placed under them.

Two PLC MODICON M340, equipped with BMX P342020 processor, DDM16025 discrete I/O and AMM0600 mixed analog I/O modules, are used to control the plant. The PLCs communicate with the SCADA (Supervisory Control and Data Acquisition) system Movicon 11.6 installed in a Windows Server 2012 machine with the following characteristics: Intel® Xeon® CPU E5-2620 v2 @2.10 GHz with a RAM of 16 GB.
The SCADA includes the HMI which shows the real time status of the system in terms of water level in tanks, if the valves are open or closed and if pumps are in action, as shown in Figure 4.

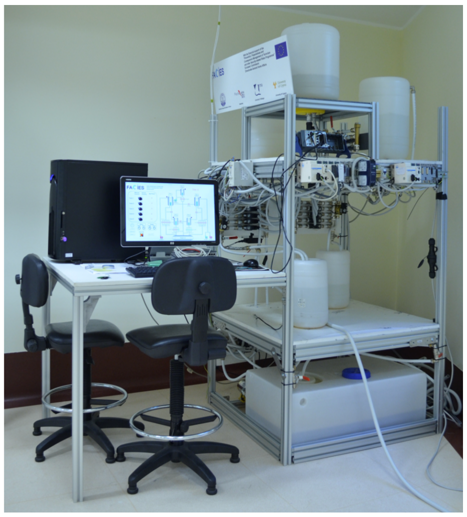


Figure 3: WDT testbed

The communication protocol between PLCs and SCADA is Modbus TCP/IP which uses port 502 [33].

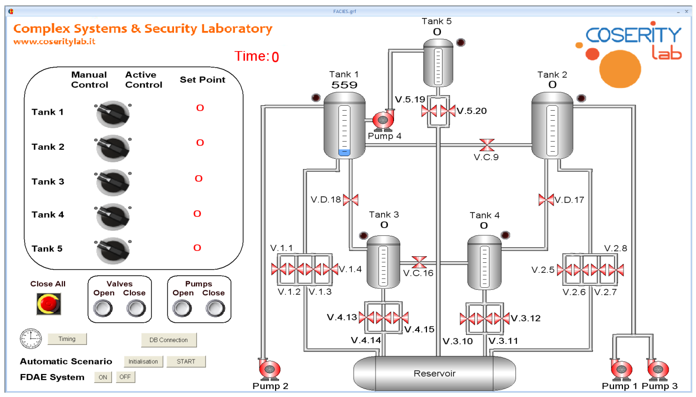


Figure 4: HMI interface

5.2 Network architecture

The network architecture consists of two network segments, as required by NIST (National Institute of Standards and Technology) best practice [34]. The first, on the left of the schema described in Figure 5, represents the most critical side, the ICS, which is directly involved in the control of the water treatment plant. On the other side, in the right portion, the SCADA with its HMI is connected to the IT monitoring network segment. Network segregation and segmentation allow to minimize the unauthorized access to sensitive and critical information ensuring that ICS can continue to operate effectively. This approach requires the adoption of a rule-set which can control the network traffic between the two segments. In our case, this function is provided by SNORT implemented as IPS [35] which allows only legitimate traffic to flow from the less protected IT segment to the most critical ICS. Thus, all the traffic exchanged between these two segments goes through the SNORT machine which consequently can block or let packets pass.

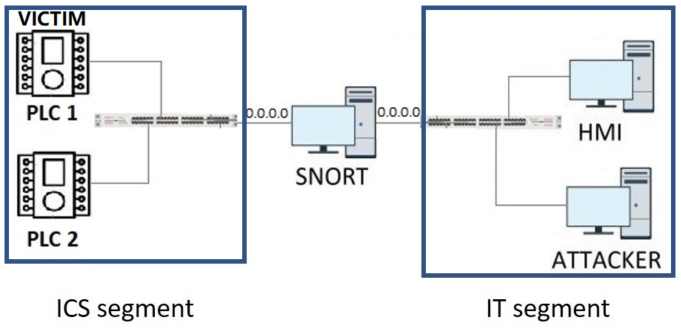


Figure 5: SCADA network

Hence, SNORT machine has two active network interfaces in promiscuous mode in order to get all the packets regardless of what their MAC address is. Moreover they have not an own IP address in order to be "transparent" for the two network segments.
SNORT 2.19.16 is installed in a Windows Server 2012 machine with the following characteristics: Intel® Xeon® CPU E5-2620 v2 @ 2.10 GHz with a RAM of 16 GB.
The attacker is implemented in a Kali Linux machine v. 2019.4 with the following characteristics: Intel® Core™ i7-8750H CPU @ 2.20GHz with a RAM of 2.4 GB. Assuming PLC1 is the victim, we define two attack scenarios:

- a SYN flood DoS attack performed with *hping3* tool in order to saturate PLC1's network resources;
- an ARP poisoning-based MITM attack performed with *ettercap* tool. The aim is to redirect to the attacker Modbus traffic sent to the HMI by PLC1 in order to change tanks water level values in packets payload.

In order to detect and block DoS attack against PLC1 in this specific network architecture, we developed a custom rule starting from the general one described in Section 2.3.

```
reject tcp $HOME_NET any -> $IP_PLC1 502
(msg :" TCP flood "; sid:1000003; rev :1;
detection_filter : track by_dst,
count 5000, seconds 3;)
```

The rule recognizes SYN flood attacks against PLC1, with the IP address $IP_PLC1, and starting from any host in the network $HOME_NET.

In order to detect ARP poisoning against PLC1, SNORT `arpspoof` preprocessor has been activated in `snort.conf` file as described in Section 2.5. The corresponding rules were already defined in `preprocessor.rules` file.

6 Results

We demonstrate the practical effectiveness of the batch script by analyzing network traffic circulating in the ICS and IT network segments during the two attack scenarios described in Section 5.1. Specifically, we show the effects of such attacks before and after SNORT impairment.
Wireshark software is used to measure packets circulation rate and to analyze Modbus packets payload.

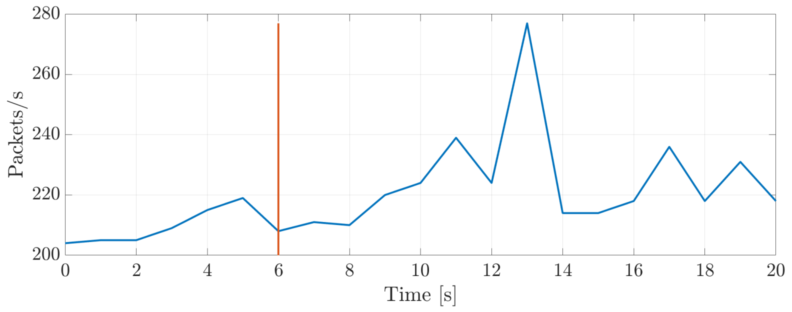
6.1 First attack scenario: SYN flood DoS attack

Figure 6 shows network traffic circulating within ICS and IT network segments when SNORT is not compromised. We observe that packet circulation rate remains in the vicinity of 450 packets/s during the first 6 seconds in the IT segment whereas in the ICS segment it assumes values between 200 and 280 packets/s. After 6 seconds, the attacker starts a SYN flood DoS attack against PLC1. As a result, packet circulation rate in the IT segment grows to 100000 packets/s value; but, no effects are induced in the ICS segment because these packets are recognized and effectively blocked by SNORT machine.

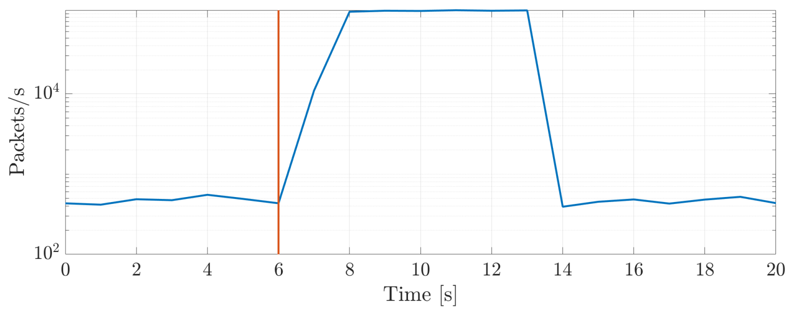
On the contrary, Figure 7 shows network traffic circulating within the two network segments when SNORT is compromised by our batch script. In this case, we observe that the attack successfully reaches the ICS network segment at time *t=6s* causing PLC1 network resources saturation. Indeed, Figure 8 shows the arrival rate of packets received by PLC1 normalized respect to its maximum throughput. We observe that, when the attack is launched, the PLC1 bandwidth is totally saturated causing its disconnection from the network.

6.2 Second attack scenario: MITM attack

Figure 9 shows the water level values of tank number 2 sent to the HMI by PLC1 over time. Specifically, we show the difference between values read from Modbus packets as they are sent by PLC1 and as they are received by the HMI. In this way, we can observe if the attacker has successfully launched an ARP poisoning-based MITM attack and if he has changed water level values in packets payload.



(a)

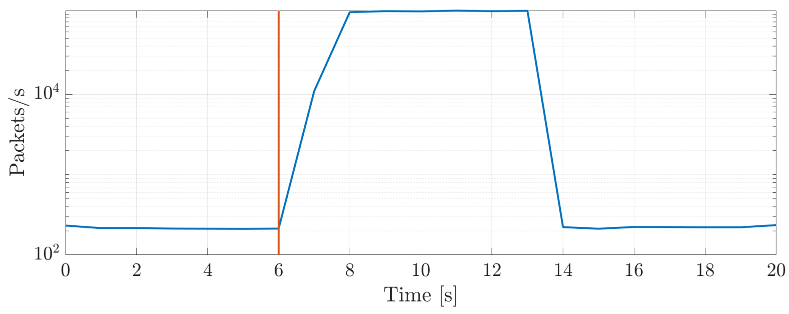


(b)

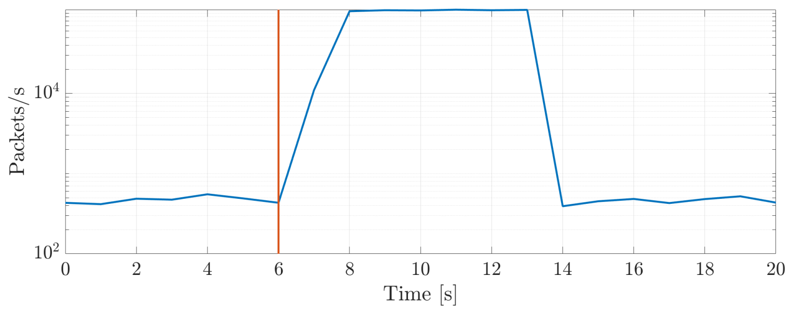
Figure 6: Network traffic in SCADA network when SNORT is not compromised. Red line represents the time when the SYN flood attack is launched. (a) Packets per second in the ICS network segment. (b) Packets per second in the IT network segment.

In this case SNORT machine has not been compromised; thus, even if the attacker attempts to poison PLC1 ARP table, spoofed packets do not reach the destination as they are effectively recognized and blocked by SNORT. As a consequence, the attacker is not able to redirect to itself network traffic sent by PLC1 to the HMI; therefore it cannot modify Modbus packets payload. The right water tank value is received by the HMI as two graphs show the same trend and the same values over time.

On the contrary, Figure 10 shows a significant difference between water level values as sent by PLC1 and as received by the HMI. In this case, SNORT is compromised by our batch script; thus it is not able to detect and block ARP spoofed packets. As a consequence they are received by PLC1 whose ARP table is successfully altered by the attacker. Therefore, the attacker can effectively modify payload of Modbus packets, before they are sent to the HMI, in order to simulate an emptying process of the tank. The effect is that the HMI will show water level values which are different to those actually present in the second tank.



(a)



(b)

Figure 7: Network traffic in SCADA network when SNORT is compromised. Red line represents the time when the SYN flood attack is launched. (a) Packets per second in the ICS network segment. (b) Packets per second in the IT network segment.

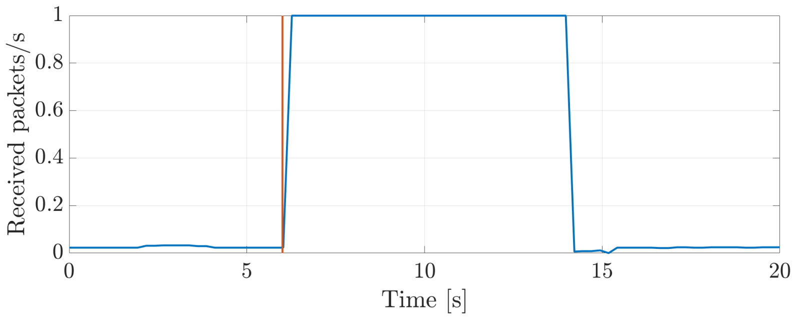
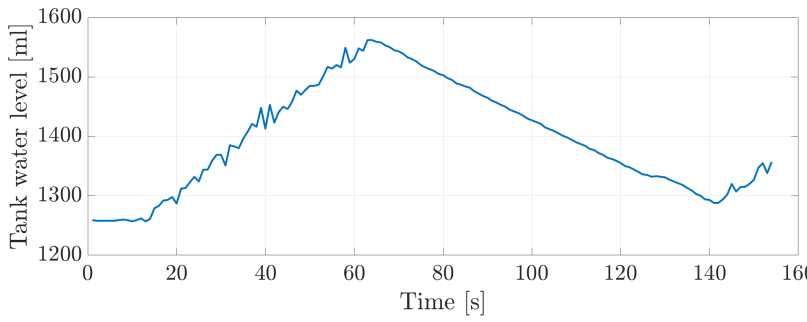
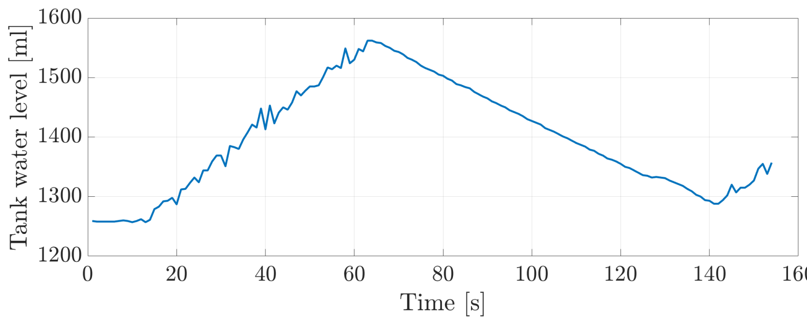


Figure 8: Packets per second received by PLC1 normalized respect to its maximum throughput.



(a)



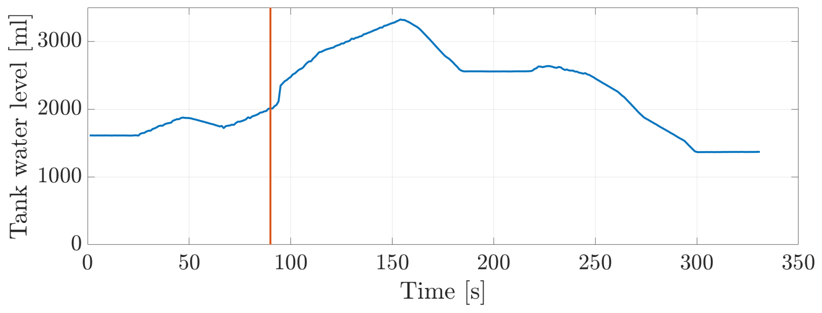
(b)

Figure 9: Tank-2 water level values in Modbus packets as sent by PLC1 (a) and as received by HMI (b) when SNORT is not compromised. Red line represents the time when the MITM attack is launched.

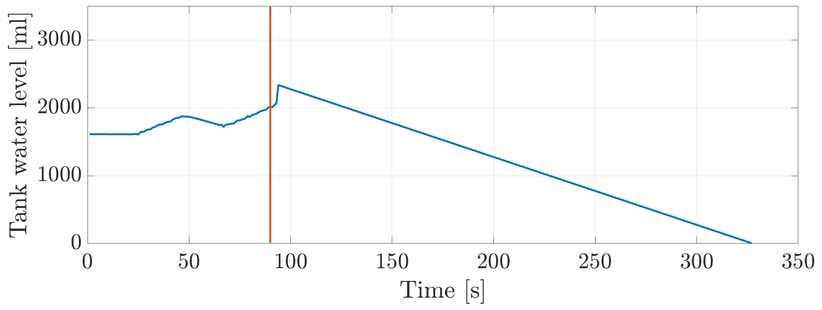
6.3 Possible countermeasures

In order to prevent an attacker to compromise an ICS network by tampering the SNORT functioning, the human operator has to consider specific security measures during SNORT installation. Specifically, the operator has to solve the main configuration vulnerability described in Section 3: the presence of readable and writable plain-text parameters files.

Configuration files of SNORT have to be encrypted in order to prevent any unauthorized attempt to modify the correct operation of the IPS. Thus, it is really important to manage the Administrator permissions as defined in the *Guide to Industrial Control Systems (ICS) Security* [34]: super-user accounts must be disabled or, at least, limited to a specific person; they have not be used for daily activities and each operation has to be password-protected in order to prevent any unauthorized attempt to tamper the OS. Moreover, any system modification has to be automatically logged and sent remotely in order to be aware of the state of the system security. Finally, internal cyber-security policies have to be defined and shared with the employees in order to increase



(a)



(b)

Figure 10: Tank-2 water level value in Modbus packets as sent by PLC1 (a) and as received by HMI (b) when SNORT is compromised. Red line represents the time when the MITM attack is launched.

the cyber-security awareness according to a process of continuous update and improvement.

7 Conclusions

In this work, we described how to exploit configuration vulnerabilities of SNORT, one of the most adopted IPSs, for Windows Operating Systems. We developed a new batch script in order to retrieve and modify the plain-text rules files of SNORT with the aim to disable its ability to detect and block DoS and ARP poisoning-based MITM attacks against a PLC in an ICS network. We assume that the script is accidentally run by the user with Administrator privileges as it was for Stuxnet and Blackenergy3. It works in a stealth manner without compromising the proper functioning of Windows OS or enabling host remote control. The sole purpose is to compromise Snort ability to detect specific attacks making the system vulnerable to these kind of threats. We experimentally evaluated its effectiveness by considering a real scenario consisting of the WDT testbed whose physical process is controlled by two PLCs. These ones communicate with the SCADA thanks to SNORT implemented as an IPS. It ensures communication between two network segments: the ICS one in which there are two PLCs and the IT one where the SCADA is implemented and where the attacker gains access. We analyzed the effects of two attack scenarios by inspecting network traffic in both segments before and after SNORT impairment. Results showed that, if the IPS is not compromised, SNORT is able to detect and block malicious packets which cannot reach the ICS network segment. On the contrary, if the batch script is running on SNORT machine, all the malicious packets gain access to the ICS network causing the disconnection of the PLC, in case of a DoS attack, and the modification of its ARP table, in case of ARP poisoning. Moreover, we showed that, as a consequence of an ARP poisoning technique, the attacker is able to stealthily modify Modbus packets payload changing tanks water level values. In this way, the human operator will see altered values of water levels which do not correspond to those actually present in tanks. Thus, he will take corrective actions, such as closing or opening valves, which could lead the plant to unexpected and potentially dangerous physical states. In conclusion, we demonstrated that the neglect and the avoidance of security choices in the initial configuration phase of an IPS can lead to significant consequences in terms of possible attacks against ICS networks.

Future works will provide a comparative analysis of configuration vulnerabilities for different types of IPSs such as Suricata and Zeek.

Acknowledgment

This work was supported by POR FESR Lazio Region Project RESIM under grant n. 36673 and by Project "DRIVERS: Approccio combinato data- driven ed experience-driven all'analisi del rischio sistemico", funded by the Italian National Institute for Insurance against Accidents at Work (INAIL) under the BRiC 2021 call.

References

[1] M. Endi, Y. Z. Elhalwagy, and A. hashad, "Three-layer plc/scada system architecture in process automation and data monitoring," in *2010 The 2nd International Conference on Computer and Automation Engineering (ICCAE)*, vol. 2, 2010, pp. 774–779.

[2] J. Weiss, *Industrial Control System (ICS) Cyber Security for Water and Wastewater Systems*, 10 2014, pp. 87–105.

[3] Y. Mo, T. H. Kim, K. Brancik, D. Dickinson, H. Lee, A. Perrig, and B. Sinopoli, "Cyber–physical security of a smart grid infrastructure," *Proceedings of the IEEE*, vol. 100, no. 1, pp. 195–209, 2012.

[4] M. r. Akhondi, A. Talevski, S. Carlsen, and S. Petersen, “Applications of wireless sensor networks in the oil, gas and resources industries,” in *2010 24th IEEE International Conference on Advanced Information Networking and Applications*, 2010, pp. 941–948.

[5] A. Cardenas, S. Amin, Z.-S. Lin, Y. Huang, C.-Y. Huang, and S. Sastry, “Attacks against process control systems: Risk assessment, detection, and response,” 01 2011, pp. 355–366.

[6] Y. Wang, Z. Xu, J. Zhang, L. Xu, H. Wang, and G. Gu, “Srid: State relation based intrusion detection for false data injection attacks in scada,” in *European Symposium on Research in Computer Security.* Springer, 2014, pp. 401–418.

[7] L. A. Maglaras, K.-H. Kim, H. Janicke, M. A. Ferrag, S. Rallis, P. Fragkou, A. Maglaras, and T. J. Cruz, “Cyber security of critical infrastructures,” *Ict Express*, vol. 4, no. 1, pp. 42–45, 2018.

[8] L. Cazorla, C. Alcaraz, and J. Lopez, “Cyber stealth attacks in critical information infrastructures,” vol. 12, no. 2, 2018, pp. 1778–1792.

[9] R. Langner, “Stuxnet: Dissecting a cyberwarfare weapon,” *IEEE Security & Privacy*, vol. 9, no. 3, pp. 49–51, 2011.

[10] R. Khan, P. Maynard, K. McLaughlin, D. Laverty, and S. Sezer, “Threat analysis of blackenergy malware for synchrophasor based real-time control and monitoring in smart grid,” in *4th International Symposium for ICS & SCADA Cyber Security Research 2016 4*, 2016, pp. 53–63.

[11] G. Settanni, F. Skopik, Y. Shovgenya, R. Fiedler, M. Carolan, D. Conroy, K. Boettinger, M. Gall, G. Brost, C. Ponchel *et al.*, “A collaborative cyber incident management system for european interconnected critical infrastructures,” *Journal of Information Security and Applications*, vol. 34, pp. 166–182, 2017.

[12] D. Ding, Q.-L. Han, Y. Xiang, X. Ge, and X.-M. Zhang, “A survey on security control and attack detection for industrial cyber-physical systems,” *Neurocomputing*, vol. 275, pp. 1674–1683, 2018.

[13] I. Corona, G. Giacinto, and F. Roli, “Adversarial attacks against intrusion detection systems: Taxonomy, solutions and open issues,” *Information Sciences*, vol. 239, pp. 201–225, 2013.

[14] S. Patton, W. Yurcik, and D. Doss, “An achilles’ heel in signature-based ids: Squealing false positives in snort,” in *Proceedings of RAID*, vol. 2001. Citeseer, 2001.

[15] R. Gula, “Correlating ids alerts with vulnerability information,” 2002.

[16] “Snort website,” https://www.snort.org/, accessed: 2021-03-16.

[17] A. Ujvarosi, "Evolution of scada systems," *Bulletin of the Transilvania University of Brasov. Engineering Sciences. Series I*, vol. 9, no. 1, p. 63, 2016.

[18] V. Kumar, "Signature based intrusion detection system using snort," *International Journal of Computer Applications Information Technology*, vol. 1, p. 7, 11 2012.

[19] A. Tasneem, A. Kumar, and S. Sharma, "Intrusion detection prevention system using snort," *International Journal of Computer Applications*, vol. 181, pp. 21–24, 12 2018.

[20] "Snort manual," http://manual-SNORT-org.s3-website-us-east-1.amazonaws.com/node29.html, accessed: 2022-04-22.

[21] "Kaspersky ics cert," https://ics-cert.kaspersky.com/media/KASPERSKY_H22019_ICS_REPORT_FINAL_EN.pdf, accessed: 2021-03-16.

[22] T. Mahjabin, Y. Xiao, G. Sun, and W. Jiang, "A survey of distributed denial-of-service attack, prevention, and mitigation techniques," *International Journal of Distributed Sensor Networks*, vol. 13, 2017.

[23] U. Tariq, M. Hong, and K.-s. Lhee, "A comprehensive categorization of ddos attack and ddos defense techniques." Berlin, Heidelberg: Springer-Verlag, 2006.

[24] Z. Hassan, Shahzeb, R. Odarchenko, S. Gnatyuk, A. Zaman, and M. Shah, "Detection of distributed denial of service attacks using snort rules in cloud computing remote control systems," in *2018 IEEE 5th International Conference on Methods and Systems of Navigation and Motion Control (MSNMC)*, 2018, pp. 283–288.

[25] B. Zhu, A. Joseph, and S. Sastry, "A taxonomy of cyber attacks on scada systems," in *2011 International Conference on Internet of Things and 4th International Conference on Cyber, Physical and Social Computing*, 2011, pp. 380–388.

[26] J. Koziol, *Intrusion detection with Snort.* Sams Publishing, 2003.

[27] J. Beale, J. C. Foster, J. Posluns, and B. Caswell, "Preprocessors," in *Snort Intrusion Detection 2.0*, J. Beale, J. C. Foster, J. Posluns, and B. Caswell, Eds. Rockland: Syngress, 2003, pp. 197–265.

[28] C. Simmons, C. Ellis, S. Shiva, D. Dasgupta, and Q. Wu, "AVOIDIT: A cyber attack taxonomy," in *9th Annual Symposium on Information Assurance (ASIA'14)*, 2014, pp. 2–12.

[29] M. Gegick and L. Williams, "Matching attack patterns to security vulnerabilities in softwareintensive designs," *ACM SIGSOFT Software Engineering Notes*, vol. 30, pp. 1–7, 07 2005.

[30] G. Assenza, L. Faramondi, G. Oliva, and R. Setola, "Cyber threats for operational technologies," *International Journal of System of Systems Engineering*, vol. 10, no. 2, pp. 128–142, 2020.

[31] D. Kushner, "The real story of stuxnet," *ieee Spectrum*, vol. 50, no. 3, pp. 48–53, 2013.

[32] L. Faramondi, F. Flammini, S. Guarino, and R. Setola, "A hardware-in-the-loop water distribution testbed dataset for cyber-physical security testing," *IEEE Access*, vol. 9, pp. 122 385–122 396, 2021.

[33] "Modicon modbus protocol reference guide, pi–mbus–300 rev. j," https://modbus.org/docs/PI_MBUS_300.pdf, accessed: 2021-03-16.

[34] K. Stouffer, J. Falco, K. Scarfone *et al.*, "Guide to industrial control systems (ics) security," *NIST special publication*, vol. 800, no. 82, pp. 16–16, 2011.

[35] H. Li and D. Liu, "Research on intelligent intrusion prevention system based on snort," in *2010 International Conference on Computer, Mechatronics, Control and Electronic Engineering*, vol. 1. IEEE, 2010, pp. 251–253.