ComSEC: Secure communications for baggage handling systems

Filipe Apolinário[1][0000−0002−2067−3232], João Guiomar[1], Éric Hervé[2], Sven Hrastnik[3], Nelson Escravana[1], Miguel L. Pardal[4][0000−0003−2872−7300], Miguel Correia[4][0000−0001−7873−5531]

[1]INOV, [2]Alstef Group, [3] Zagreb Airport, [4]INESC-ID, IST, ULisboa
{filipe.apolinario,nelson.escravana}@inov.pt, eric.herve@alstefgroup.com, {miguel.pardal,miguel.p.correia}@tecnico.ulisboa.pt

Abstract. Throughout the years, the number of network attacks targeting industrial control systems (ICS) has increased. A notable example targeting airport infrastructures is false data injection attacks, where attackers try to impersonate parts of the ICS system using spoofing techniques, sending unauthorized commands to hinder the quality of service. This article presents ComSEC, a bump-in-the-wire technology for detecting attacks against integrity and replays. The article also describes the development and deployment on the simulation platform of two ComSEC prototypes for monitoring airport baggage handling systems (BHSs): 1) a virtualized version crafted for monitoring virtual machines; 2) a physical hardware version crafted for monitoring airport physical hardware systems. ComSEC was evaluated on a digital twin BHS, available on the SATIE simulation platform[1] and integrated with the Zagreb airport BHS.

Keywords: integrity verification, critical infrastructures, airport security

1 Introduction

Airport infrastructures [16,6], as other industrial infrastructures, are comprised of two main classes of assets: physical assets and technological assets. The realm of *airport physical assets* (such as control towers, hangars, terminals, and baggage conveyors) supports the core business of these transportation infrastructures and provides a wide type of services (air traffic control, ground control, baggage and passenger boarding, etc.) to end users who rely on the activities of these organizations (e.g., passengers, airline companies) [20]. On the other hand, the *technological assets* that comprise airport infrastructure monitoring and control of physical assets provide airport personnel with technological tools to manage and supervise the work performed on the physical assets of the airport and automate the scheduling of the tasks required to provide airport services.

Figure 1 shows that, similarly to other industries (transportation, energy supply and distribution, manufacturing, etc.), airports are cyber-physical infrastructures, typically organized as *industrial control system* (ICS) architectures [18] where

[1] https://satie-h2020.eu/

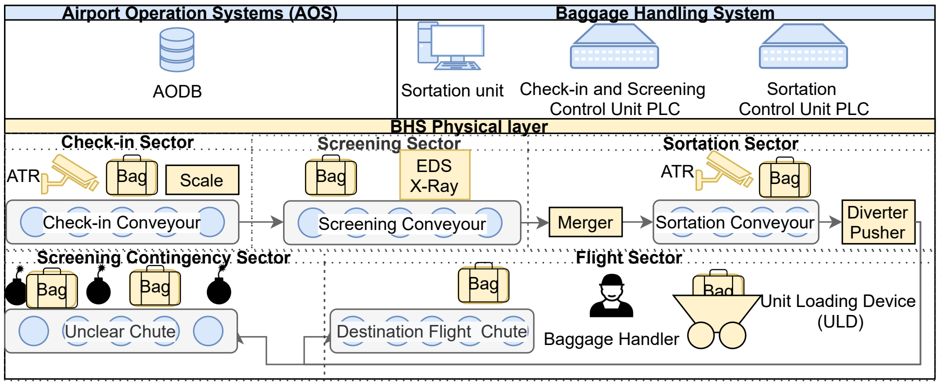

Fig. 1: Example of an airport baggage handling system (BHS) architecture.

physical assets connect to technological assets using different network protocols (e.g., Profinet, MODBUS, DNP3).

Although industrial control systems are often *air gapped* with very strict access control policies, these cyber-physical environments have been subjected to successful cyber-attacks that were able to bypass ICS security mechanisms and issue unauthorized commands to ICS critical systems, resulting in physical equipment and safety failures. A notable example of a cyber-physical attack, Stuxnet, was able to infiltrate and shutdown an Iranian uranium enrichment plant through the Supervisory control and data acquisition (SCADA) system. From there, Stuxnet gained access to the programmable logic controllers (PLCs) that controlled the plant centrifuges. Stuxnet exploited unpatched Windows vulnerabilities and sabotaged centrifuges by making them spin up to 40% faster than normal velocity, while falsely reporting normal velocity to supervisory servers, resulting in permanent equipment damage without being noticed by operators. Another notable example of a cyber-attack on ICS systems is the 2016 Shamoon attack that, among others, targeted Saudi Arabia's General Authority of Civil Aviation (GACA). This attack was carried out using a worm malware that started by obtaining administrator access to a remote computer at the aviation center, destroyed the data on the computer and proliferated to other devices on the network. The Shamoon attack was able "to destroy critical data and hinder operations to a halt for several days". Another example is the 2019 Kemuri attack targeting a water treatment facility, where attackers were able to access from the internet the ICS system of this facility, obtained remote access to the PLCs and performed malicious operations to alter the chemical dosage used in the water treatment procedure, putting at risk the whole treatment procedure.

In a broader sense, a high number of *attacks on cyber-physical environments* can often be classified as attacks on data integrity, often through *false data injection attacks*. In these attacks, attackers impersonate parts of the ICS system using spoofing techniques to send unauthorized commands to critical systems in order to hinder the quality of service. Common attacks often include injection of

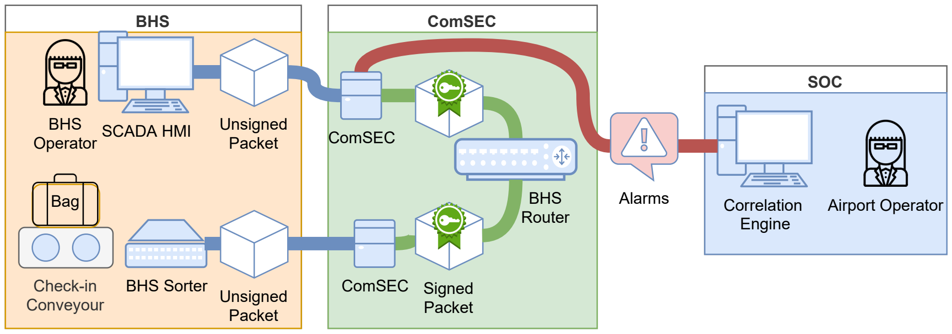

Fig. 2: Deployment of ComSEC on a BHS sending alarms to a SOC.

privileged operations to change a control unit state (e.g., shutdown operations or tampering state variables), or issuing operations to change the properties of the physical systems the unit is connected to (e.g., stopping baggage handling systems conveyors or shutting down explosive detection systems). Other common attacks often include, sending false messages to a SCADA or HMI system on the behalf of control units, in order to gain a false impression of the system current state for the infrastructure administrator to trigger the wrong contingency measures. Common examples of SCADA attacks include falsifying control unit readings to circumvent malicious activity detection, or falsifying readings for administrators to perform countermeasures that can hinder the quality of service (such as, shutting down control unit equipment, changing physical properties, like stopping or slowing down conveyors, etc.).

Protection against false data injection attacks is often difficult in ICS systems, since on one hand they are typically constructed with specific network protocols for each domain (e.g., airport systems use specific network protocols for the aviation domain, such as IATA message types or INFOPAX, while energy sectors use iec61850 network protocol suite) that typically do not include cyber security measures to assure the message integrity protection required to protect against these attacks; and on the other hand most systems are developed for long lifespan and any efforts to add additional network protection to those systems often require development costs to equip these legacy systems with the necessary cryptography means for integrity assurance. Several *security mechanisms* protect against false data injection attacks exist and can be divided into two categories, secure *tunnelling approaches* [3,4,13,14] that create a secure channel capable of providing security assurances on all data packets shared between ICS devices and *bump-in-the-wire approaches* which are cryptographic units placed between ICS devices and the local network, to update security guarantees of network protocols. However, current approaches to improving security often *introduce latency* in communication due to encryption and decryption operations, which can compromise the quality of services of time critical ICS operations.

This article describes ComSEC, a mechanism aimed at providing *integrity* and *authentication* assurances to all IP network communications. As illustrated in Figure 2, ComSEC was developed as a bump-in-the-wire, that inspects network traffic exchanged between an ICS device and the network and digitally signs outgoing traffic. Incoming traffic is inspected and validated according to other ComSEC signatures. ComSEC is equipped with an alerting system that sends in real-time events to other intrusion detection and Security Information and Event Management (SIEM) systems, whenever integrity and authentication incidents are detected. ComSEC reduces network latency often imposed by bump-in-the-wire technology, by proposing an innovative mechanism based on Linux Netfilter kernel modules [2] that inspects outgoing traffic exchanged between ICS devices and sends the network packet digital signatures on a separate packet asynchronously through the ICS network. ComSEC has correlation capabilities to filter out network packet signatures from ICS network traffic, forward the traffic to the ICS device and asynchronously validate integrity. ComSEC is designed to be plug-and-play transparent (i.e., it has no IP address; it mimics the IP address of ICS devices connected to ComSEC) and does not require configuration on the network or hosts. During the installation, ComSEC automatically infers the necessary configuration parameters based on the information collected from the network. ComSEC can be inserted into ICS networks, supporting several ICS network protocols (tested with several protocols, including Profinet, BHS-specific network protocols, MODBUS protocols) and requires no modifications to ICS devices to accommodate ComSEC interaction.

For evaluation, ComSEC was installed at the Zagreb airport to validate the integrity of BHS communications. During this installation, we simulated false data injection attacks that were detected by ComSEC with 100% accuracy, and a low false positive rate (3%). We also integrated ComSEC with a SIEM developed in the H2020 SATIE project, named Correlation Engine, which received the ComSEC integrity alerts and correlated with other alerts coming from a detection system that monitored BHS processes, named Business Process Intrusion Detection System (BP-IDS)[12]. Thanks to ComSEC and the Correlation Engine, it was possible to detect the false data injection attack (based on ComSEC alerts), and also identify the consequences of the attack to the physical processes of the BHS (based on BP-IDS alerts). Thus showing that the mixture of different cyber and physical detection sensors can be instrumental for an holistic and complete view of the cyber-physical attack surface.

2 Related Work

Approaches to increase the security of unprotected industrial control system network communications have been studied over the years. These approaches can be divided into two main categories: secure tunneling approaches for creating secure links between ICS components; and bump-in-the-wire approaches that use network taps to intercept network traffic between devices and ICS network, and upgrade the network packet security assurances.

Secure tunneling approaches [3,4,13,14] allow upgrading network protocol security using secure shell tunnels (SSH) or IPSec virtual private networks (VPN).

[2] `https://www.netfilter.org/`

For example [3] uses SSH to secure machine manufacturing messages (MMS) exchanged between a SCADA server and ICS control units. In this case, each ICS control unit has an SSH server application, and SCADA server establishes an SSH tunnel where MMS messages are sent to a local port of the SCADA server and redirected via SSH tunnel to control unit machine. Although this approach raises confidentiality, integrity, and authentication, the approach also imposes several maintenance difficulties. First, it requires changing software applications to use the SSH tunnels instead of directly connecting to control unit devices (which can be often difficult to manage since each network protocol will have a different local port). Second, firewalling cannot be done at the network level, as all network communications are done through SSH tunnels. Thus, firewalling needs to be done at the host level to forbid network traffic sent over the SSH tunnels. Another way to perform secure tunneling while reducing the amount of configuration needed on each ICS host is to use IPSec [13,14]. In this approach, each ICS device establishes a secure channel to other devices using IPSec transport mode. In IPSec transport mode, each device establishes a secure connection to another machine where all network protocols exchanged are upgraded with two possible modes: (1) authentication headers, where the IPSec application alters outgoing packets and appends an authentication header to ensure integrity of all data present in the internet protocol (IP); (2) encapsulating security payload (ESP), where IPSec application alters outgoing traffic and replaces IP data with an encapsulation payload that contains the original packet cyphered and digitally signed. Although this approach solves confidentiality, integrity and authentication and requires less configuration than SSH tunnels (requires setting up one IPSec connection for each ICS device the machine communicates, instead of one SSH tunnel for each network protocol and host the machine communications) the approach continues to require installing IPSec software and demands from ICS devices a high level of computation requirements to ensure cryptographic properties (this can be a particular difficulty in ICS since most control unit devices do not have the computational requirements to install and run IPSec), several maintenance difficulties can also arise in ESP, namely requires firewalling cannot be done at network level since all network communications are done via IPSec, it requires firewalling at host level to forbid network traffic sent over the IPSec tunnel (which can be difficult to perform given the control units computational requirements). Aside from the aforementioned problems, tunneling approaches also introduce latency in the communication due to encryption and decryption operations, which can compromise the quality of services of time-critical ICS operations.

To reduce the amount of configuration and computational requirements on the ICS control unit devices, recent articles studied the introduction of *bump-in-the-wire devices* placed between the ICS devices and the local network. These devices are often viewed as proxies or network bridges that intercept network communications between devices, and upgrade the communications. Bump-in-the-wires improve security by *offloading computational requirements* from ICS devices. An approach [17] is to create a bump-in-the-wire to authenticate transport communication between SCADA and autonomous vehicle control units, in order to create a secure channel for the legacy machine manufacturing protocol (MMS).

Another approach [19] is to design a bump-in-the-wire methodology to cypher data and provide authentication in communications. Another possible approach [9] is constructing a bump-in-the-wire to convert legacy network protocols (like the unsecure power grid network protocol IEC 61850-90-5) into the upgraded and more secure version of this protocol (IEEE C37.118.2) which provides confidentiality and integrity assurances, and can also convert it back to IEC 61850-90-5 at the receiving ICS device for backward compatibility on legacy systems. Another bump-in-the-wire approach [11] created a stealth bump-in-the-wire network bridge firewall based on Netfilter ebtables that is hidden from attackers by not having an IP address. Another approach [10] customizes bump-in-the-wire firewall to inspect MODBUS network packets and offers features for ICS administrators to create custom rules to inspect proprietary network protocols. Bump-in-the-wires [7] were also created as a bridge based on ebtables that intercepts network packets and appends authentication with message authentication codes and freshness parameters to all outgoing traffic exchanged by ICS devices for validating freshness of communication exchanged between entities. A lightweight bump-in-the-wire [5] was also proposed using hash-chaining based on a key sharing protocol exchanged between ICS devices to protect against integrity and replay attacks in a multi-hop environment. Another bump-in-the-wire [8] validated coherence between the MODBUS messages received by the control units and the physical signals issued by the control unit. Although the various approaches address integrity and replay attacks, current approaches to improve security are often *focused on particular network protocols* or need to be configured with information about hosts on the network and therefore cannot be adapted to dynamic environments where new hosts are introduced into the network.

3 ComSEC

ComSEC is a bump-in-the-wire technology to ensure integrity in ICS communication networks. It is designed to be physically connected between a device and a network router/switch, to digitally sign all network traffic sent by the device, and to validate network traffic before it reaches the device. ComSEC is able to detect and block the effects of man-in-the-middle attacks (such as false message injection and message tampering). It supports any network protocol that works over IP (e.g., TCP, UDP, MODBUS, Profinet, etc.). ComSEC notification engine sends security alerts in real time to Correlation Engine. As seen in Figure 3, ComSEC is organized into a three-layer architecture:

1. *Physical layer* – provides the bump-in-the-wire and alert exportation. It contains two ethernet interfaces configured in bridge mode, one connected to the device and one connected to the router. These interfaces provide the bump-in-the-wire feature. The layer also has an alarm interface to export alerts to other systems.
2. *Netfilter layer* – provides a way to intercept the communications coming from and to the device. This is made using two ComSEC Kernel modules (ComSEC KMod): (1) Netfilter Outbound Module that intercepts packets sent from the

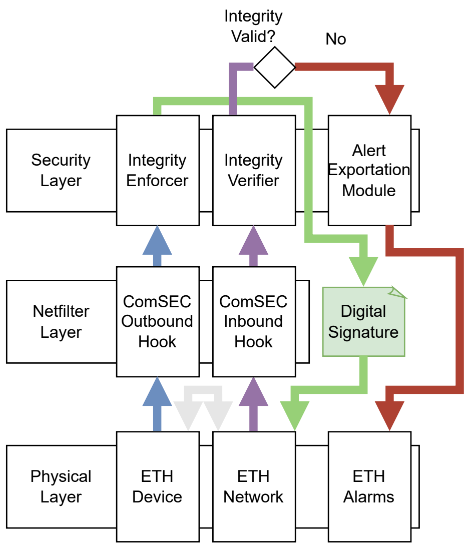

Fig. 3: Internal software architecture diagram of one ComSEC.

device ComSEC is protecting; and Netfilter Inbound Module that intercepts network packets before they are received by the machine ComSEC is protecting.

3. *Security layer* – handles the security of the communication. It has three Linux user space modules: (1) Integrity Enforcer (IE) that signs network packets coming from the device (intercepted by Outbound Module); (2) Integrity Verifier (IV) that verifies integrity of the communication reaching the device (received by the Inbound Module); (3) Alert Exportation Module, issues alarm notifications to monitoring systems when IV identifies integrity problems.

As also seen in Figure 3, ComSEC functioning is organized into four main workflows. First is the signing incoming packets (represented by the blue lines in the figure). Here the Ethernet (ETH) Device interface receives packets and forwards them to IE module, using the Outbound module installed in the Netfilter. The second is forwarding ComSEC signatures into the network (represented by the green lines in the figure). Here the IE module sends the signatures to the ETH Network interface. The third is validating integrity of the received network traffic (purple lines in the figure). Here the ETH Network interface receives packets and signatures and forwards them to IV module, using the Inbound module installed in the Netfilter. Forth is publishing alerts when integrity is compromised (Red lines). Here the IV notifies the alert exportation module, which sends alerts using a dedicated ETH Alarms interface.

3.1 Bridged Interfaces

To be physically installed between a device and a network router/switch, ComSEC operates at layer 2, as a virtual bridge, with two interfaces with no IP addresses (it is transparent to layer3). This way, ComSEC is as a bump-in-the-wire between one computer and the network. For validating the integrity on communications between two devices (e.g, two servers, Server1 and Server2), two ComSECs need to be installed. The ComSECs will intercept network traffic sent between the machine and the network (ComSEC 1 for Server1 and ComSEC 2 for Server2) and signs the network traffic to allow the other ComSEC to validate. Also, each ComSEC will intercept network traffic received by the machine and validate it using the signatures received by the other ComSECs in the network (i.e., to validate packets received from Server2, ComSEC 1 will use the ComSEC 2 signatures to validate the packet).

3.2 Netfilter modules

ComSEC uses Netfilter to capture and analyze packets that pass through the network bridge. Netfilter is a framework for packet filtering and mangling. As can be seen in Figure 4, each protocol defines a series of hooks, at various points (five for IPv4) in a packet's traversal of a protocol stack. At each of these points, the protocol will call the Netfilter framework with the packet and the hook number:

PRE: NF_IP_PRE_ROUTING hook, called for packets that come in after simple sanity checks, and before any routing code.

IN: NF_IP_LOCAL_IN hook, Netfilter framework is called again, after routing, if the packet was destined for a local process, and before being passed to the intended process.

FWD: NF_IP_FORWARD hook, called after routing if the packet was destined to another interface instead.

POST: NF_IP_POST_ROUTING hook, Netfilter framework is called again before being put on the network interface again.

OUT: NF_IP_LOCAL_OUT hook, called for packets that are created locally.

Kernel modules can register to listen to the different hooks for each protocol, specifying the priority of the function within the hook. When that Netfilter hook is called, each module registered at that point is called in the order of priority, being free to manipulate the packet. The module can then tell Netfilter to: **NF_DROP**: Discard the packet.

NF_ACCEPT: Continue traversal as normal.

NF_STOLEN: Forget the packet.

NF_Queue: Delegate the decision on packets to a user space software.

NF_REPEAT: Call the current hook again.

The ComSEC Netfilter kernel module (ComSEC KMod) registers to listen to the IPv4 protocol (PF_INET) on the $PRE-ROUTING$ hook with the highest possible priority. The hook used is $NF\_IP\_PRE\_ROUTING$.

When ComSEC KMod is called, first it needs to identify whether it is an inbound or outbound packet. Inbound packets are packets in which the destination is the device to which ComSEC is connected, and outbound packets are those that come from that same device, that is, outbound packets are those in which the device is the source of the packets. ComSEC performs different actions depending on the packet destinations, inbound packets are signed by ComSEC, using a user space module named Integrity Enforcer, while outbound packets are validated

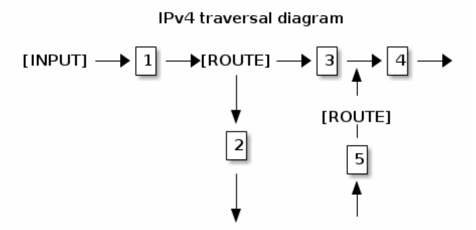

Fig. 4: ComSEC's Netfilter hooks for inspecting communications.

by ComSEC using a user space module named Integrity Verifier. In order to identify the traffic flow, that is, the inbound and outbound packets, and decide the destination of the packets, to the Integrity Enforcer or to the Integrity Verifier respectively, it is necessary to set in the ComSEC configuration which of the ComSEC bridged interfaces is connected to the device and which is connected to the network router/switch. Therefore, if a packet received on the ComSEC network bridge comes from the interface connected to the device, this packet is outbound traffic and will be sent to the Integrity Enforcer. Otherwise, if the packet comes from the interface connected to the network router/switch, this packet is inbound traffic and will be sent to the Integrity Verifier. Sending a packet to user space applications, Integrity Enforcer or Integrity Verifier, means sending the packet to the corresponding Netfilter queue (NF_Queue) target, by the ComSEC KMod. Netfilter queue (NF_Queue) allows user space modules to subscribe to Netfilter kernel events, and this way offer an interface between kernel and user space models. When a packet reaches an NF_Queue target it is placed in the queue depending on whether it should be handled by the Integrity Enforcer or Integrity Verifier. The packet queue is a chained list with each element being the packet and metadata. The packets in the queue are then handled by the user space software asynchronously.

3.3 Integrity Enforcer

The Integrity Enforcer (IE) is a user space daemon that subscribes to a given Netfilter queue. When an outbound packet is enqueued, the Integrity Enforcer is notified by the kernel using an nfnetlink formatted message causing the execution of the Integrity Enforcer handler, a callback function that is subscribing Netfilter queue events. For each packet received by the Integrity Enforcer handler, the handler signs the packet and sends it in a raw UDP packet. This raw UDP packet functions as a control packet and is injected into the network through the interface connected to the network router/switch. The control packet will be used by the Integrity Verifier of the other ComSECs present in the network, to validate the integrity of inbound network traffic. The decision was made for ComSEC to send packet signatures over UDP raw packets instead of TCP, to facilitate its deployment in critical infrastructures without the need to keep track of the connections to other ComSEC on the network, and also not to be exposed to other devices on the network. ComSEC devices do not have IP addresses, nor have any sockets exposed to the monitored network, they function as a stealth network bridge that injects the packet signatures on behalf of the monitored devices (i.e., raw packet containing the signature has the same source and destination address of the original packet). This design makes it harder for attackers to identify the ComSEC module, but requires ComSEC signatures to be sent over sessionless communication channels (i.e., UDP) in order to be aligned with its stealth design. This makes ComSEC possibly susceptible to verification errors, due to possible control packet loss in the network. If we consider an estimated bit error rate (BER) in the network and recall that ComSEC sends a control packet for each network packet in the network, the probability of ComSEC to have verification errors is given by the following formula: $P(Error) = \frac{BER \times \sum_{i=0}^{n} |controlPacket_i|}{\sum_{i=0}^{n} |controlPacket_i| \sum |packet_i|}$

As can be seen in this formula, since control packets have almost the same size (size of the hash and metadata does not vary significantly), the probability for ComSEC to display errors depends on the BER and on the size of the packet. Given that BER in lossy networks (e.g., wireless) is typically estimated to be 10^{-5} the error rate would be negligible on the ComSEC integrity detection results giving 1 alarm every 10^5 packets. Also, since control packets are very small compared to the packet size, the probability for a bit error would be smaller than 1 out of 10^5 packets.

3.4 Control packets

ComSEC control packets contain the necessary information for integrity validation.Control packets are sent as UDP control packets, in which the headers contain the following information required for network routing: Identification: ID of the IP header used for all control packets (defined in configuration for debugging purposes); Source and Destination Address: Same as in the packet that originated the control packet; Source and Destination Port: Control packet source and destination port (defined in configuration).

Control packet payloads are used as signatures, for integrity verification and contain the following information:

- Validations fields: Prefix: Used to differentiate control packets from other packets. It is a fixed string (e.g., "ComSEC _PACKET") that is introduced in the configuration. ComSEC ID: ComSEC identification number. Since multiple ComSECs are deployed on the same network, this ID allows the identification of the ComSEC responsible to issue each control packet. This ComSEC is defined in configuration and should be different from other ComSECs installed on the network. NFQUEUE Packet Hash: Cryptographic hash-based message authentication code (HMAC) used to evaluate the integrity of the packet that originated the control packet. The data used to calculate covers the packet IP header, transport header and data. This HMAC uses a pre-shared key present on the ComSEC file system that is shared between all ComSECs during installation. Control Packet MAC: Control packet hash-based message authentication code (HMAC). It can be used to validate the control packet (the data used to calculate is Prefix, ComSEC ID, Timestamp Nonce and Packet Hash).

– Freshness fields: Timestamp: UNIX timestamp of the creation of the control packet. Nonce: Number that can be used just once in a communication.

3.5 Integrity Verifier

Like the Integrity Enforcer, the Integrity Verifier (IV) is a user space daemon that subscribes to a given Netfilter queue. When an inbound packet is enqueued, the Integrity Verifier is notified with an nfnetlink formatted message causing the execution of a callback function (the Integrity Verifier handler). The Integrity Verifier handler will perform a set of validations in order to assess the integrity of each inbound packet. The main objective of the Integrity Verifier is to receive a network packet and the corresponding control packet from the Netfilter queue and validate integrity. Since both packets are received separately from the queue, Integrity Verifier uses a temporary knowledge database, to store information about packet or control packet until it has received both packets and has the necessary information to decide. Since ComSEC may experience data loss or intentional packet drops, at the moment the Integrity Verifier receives the first packet (network or control packet), a timer is set in the database to limit the time frame for the Integrity Verifier to decide. If this timer is reached, ComSEC raises the alert that the packet has been tampered and integrity cannot be assured.

3.6 Alert Exportation Module

ComSEC Alert Exportation Module implements alarmistic functions with automatic export of alerts. Thus, with this module, it becomes possible to configure ComSEC to send alerts to a SOC and have quick access to the incidents reported.

4 Implementation

To demonstrate ComSEC a prototype was created. ComSEC is built as a network bridge appliance that intercepts network communication packets exchanged between a host and a network switch/router. ComSEC acts as a validator component that intercepts all network traffic and validates the integrity of each network packet. Whenever integrity validation fails to be verified for a given network packet, ComSEC generates integrity alerts.

As depicted in Figure 5, in essence, ComSEC is designed as an appliance running software composed of kernel modules and application modules.

The kernel module, installed on the Netfilter of the ComSEC appliance, intercepts communications between the monitored BHS device and network. The kernel module consists of two components: ComSEC Outbound Module that intercepts network packets sent by the BHS device to the network; and the ComSEC Inbound Module that intercepts network packets received from the network to the BHS device.

The application modules that receive the packets, inspect the network packets intercepted by the kernel modules, and perform the necessary operations to ensure

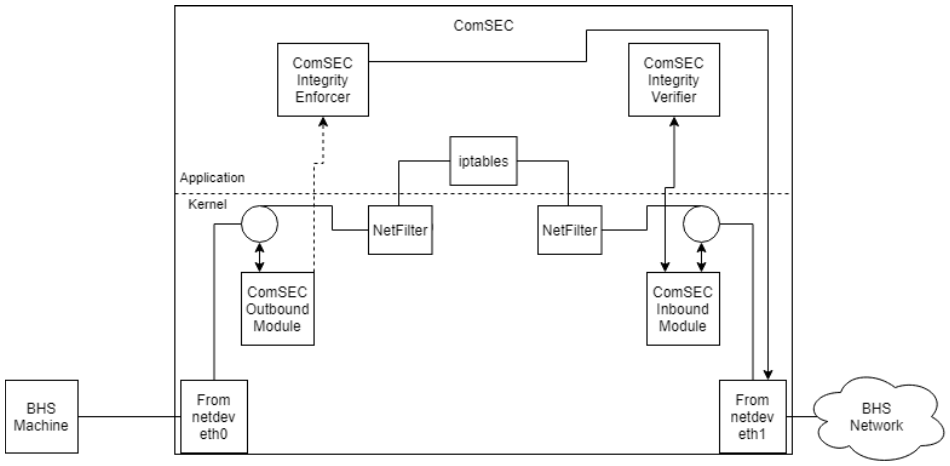

Fig. 5: ComSEC architecture installed between one BHS machine and network.

the integrity of the communication. The two modules are: ComSEC Integrity Enforcer that generates message authentication codes (MACs) of all network packets sent by the BHS machine to the BHS network and forwards them to the BHS network to allow other ComSECs to validate integrity; and the ComSEC Integrity Verifier that compares network packets with the MACs produced by other ComSEC present in the BHS network and detects integrity attacks (e.g., unauthorized command execution, denial of service, man-in-the-middle and replay attacks).

The ComSEC prototype functions as a network bridge, allowing all traffic to and from the monitored device to pass transparently through the ComSEC network bridge. Each ComSEC will also have an isolated connection to the SATIE Correlation Engine to send alerts to the Correlation Engine.

5 Evaluation

ComSEC was integrated with a BHS. Specifically, the evaluation focused on three airport systems represented in the simulation platform: the flight information management system (FIMS), airport operation database (AODB1) and the BHS manufactured by Alstef. Under this setting, the experiments conducted to evaluate ComSEC monitored the Airbus simulation platform of the BHS operating continuously for 24 hours. This simulation included a virtualized airport database that provided to the BHS sortation unit identifiers of fictitious bags and the corresponding fictitious flights assigned to physical locations of the BHS. The simulation also included, real physical equipment namely explosive detection systems (EDSs), automatic tag readers (ATRs) and conveyors. The equipment was managed by real control units connected to the BHS sortation unit in the simulation platform. The simulation used for this experiment, represented a high-fidelity with real BHS available on airports by using Emulate3D, a testing platform commonly used by BHS service providers to test their systems on contractual operating conditions before they are installed on airports. During the tests FIMS sent flight information

Type	Alerts	TN	TP	FN	FP	Accuracy	FPR
Integrity	220	18736131	213	0	7	100.0%	3.2%
Uniqueness	14656	18721695	14527	0	129	100.0%	0.88%
Both	14876	18721475	14740	0	136	100.0%	0.91%

Table 1: ComSEC integrity detection results for the 18736351 packets evaluated.

messages. AODB issued bag check-in operations, and received messages about the bags sortation and screening decisions made by the BHS. The network traffic (24 hours) used for monitoring the BHS, included a total of 1956 bags monitored and a total of 18.736.351 monitored packets. During two hours, the simulation platform was used to force the BHS to two abnormal situations. The first one, *screening anomaly*, had a duration of 22 minutes where the EDS screening results of 42 bags were changed by the simulation platform to route unclear bags to flight instead of the contingency chute (7 unclear bags), and clear bags to the contingency chute (33 clear bags). The second abnormal situation, "sortation anomaly", the simulation platform overwrote the BHS sortation messages to falsely route bags to different destinations. The abnormal situation had a duration of 30 minutes when wrong sortation orders were given for 36 bags, where bags assigned to "flight 1" were routed to "flight 2" (20) and vice-versa (16).

Table 1 shows the detection results for ComSEC to identify integrity anomalies and replays. ComSEC monitored 18.736.351 packets and raised a total of 14747 alerts. Of the alerts raised, 220 were related to integrity problems, while 14527 were related to retransmission of network packets (uniqueness). The number of true positives (TP) is the number of attack packets that ComSEC detected correctly. True negatives (TN) are packets correctly verified by ComSEC as legitimate. False positives (FP) are packets wrongly judged to be an attack. False negatives (FN) are packets that were able to evade ComSEC detection. The metrics used were accuracy as $\frac{(TP+TN)}{(TP+TN+FN+FP)}$ and false positive rate (FPR) as $\frac{FP}{(TP+FP)}$.

During the evaluation, ComSEC had 0 false negatives, showing that ComSEC detected all anomalies and that it is a very accurate solution to assess communication integrity. This is due to the fact that ComSEC is deterministic when assessing integrity. For each packet sent from a device ComSEC is connected to, it will send a control packet with the signature of the packet. The non-correspondence of the signature or absence of the control packet on a receiving ComSEC, will always generate an alert. Regarding false positives, ComSEC shows a false positive rate of 3.2% for detecting integrity problems and 0.88% for detecting packet retransmission. Regarding integrity detection, this FPR is related to data loss during the transmission of control packets between BHS devices. The control packets are UDP packets which is a connectionless protocol. They have no guarantee of delivery, ordering, or duplicate protection. However, taking into account the reliability of current networks, a high UDP packet loss rate is unlikely (in such a case, ComSEC may display some FPR).

ComSEC was also integrated in the Zagreb airport and inspected the BHS network traffic. To this end, the Zagreb demonstration encompassed two stages: the installation and public demonstration.

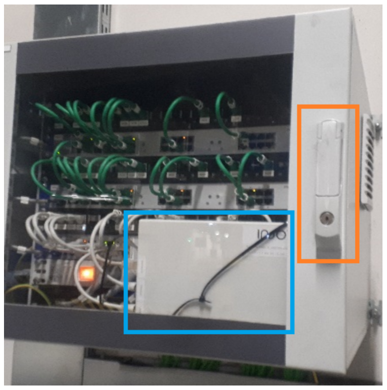

Fig. 6: ComSEC (in blue) on the locked airport switch cabinet (orange).

Two ComSEC appliances were connected on the BHS, one for each PLC. ComSEC was placed with one network interface connected to one PLC, and another network interface connected to the BHS switch, and one network interface connected to the simulation platform. During the period in which the ComSECs were deployed, they served as a bump-in-the-wire for all traffic that reached the PLCs. This deployment allowed ComSECs to completely monitor BHS activities and forward any detected integrity failures to the SOC. BHS security adminstrators used the SOC impact assessment tool called Business Impact Assessment (BIA) [15][3], to interpret ComSEC alerts and react accordingly.

During 1 month, ComSEC operated continuously, including during the normal airport operation period. Throughout the time ComSEC was active, there was no downtime or the need to intervene to do troubleshooting. This shows that ComSEC is compatible with the BHS infrastructure and capable of withstanding the normal operation conditions of the airport. A picture of the deployment can be seen in 6. The picture shows ComSEC installed on the airport cabinet. In the deployment ComSEC was strategically placed in the switch cabinet since the switch is protected by lock, showing that in an airport installation it would be very difficult to have physical access ComSEC, since it would be as difficult as accessing a network switch (which only a limited number of authorized personnel have access to).

During the Zagreb airport experiments, two staged attacks were carried out using the simulation platform: 1) Ransomware attack; 2) Change sortation messages.

The Ransomware attack starts in one of the simulation platform devices (representing the airport check-in counters computers) and automatically propagates to the simulated SCADA system through the network. This step is detected by ComSEC system which raises an alert of anomalous communication targeting the SCADA system. The alert is raised since the machine where the Ransomware was placed sent an unauthenticated network packet to the SCADA system.

[3] BIA is also published as Critical infrastructure impact assessment [2]

Regarding the change of sortation messages, ComSEC detected a man-in-the-middle between BHS and PLCs; performing the change of sortation orders. This is due to ComSEC packet replay protection. This means that due to the man-in-the-middle the PLC's ComSEC will receive two network packets, the original one (sent from the PLC) and one sent by the attacker machine with the same signature, and will thus raise an alert.

6 Conclusion

This article described ComSEC, a network bridge device that validates integrity and detects replay attacks on ICS infrastructures, while imposing no significant overhead on network communications. ComSEC provides real-time alarms, as demonstrated in our evaluation. The deployment of ComSEC in a simulation platform and the Zagreb airport were also discussed in this article, showing that it can be easily deployed on current airport systems, with little configuration needed, maintenance and good detection results. From the SATIE project, it was possible to see ComSEC adds security without any additional deployment requirements, it can also be used alongside intrusion detection systems (such as BP-IDS used in the SATIE project [15,12]) to correlate alarms and identify the reasons for integrity violations. A closer look into the BP-IDS evaluation conducted on the SATIE project [1] shows ComSEC could provide an early warning packet integrity violations, which can then be correlated with a BP-IDS alarm containing the physical impact of the anomaly. The correlation of both alarms as it was done in the SATIE project provides a rich trace of the anomalous behaviour [4].

Work supported by the European Commission through contract 832969 (SATIE) and by national funds by FCT through grant UIDB/50021/2020 (INESC-ID).

References

1. F. Apolinário, N. Escravana, E. Hervé, M. L. Pardal, and M. Correia. FingerCI: Generating specifications for critical infrastructures. In *Proceedings of the 37th ACM/SIGAPP Symposium on Applied Computing*, SAC '22, page 183–186, 2022.
2. O. Carvalho, F. Apolinário, N. Escravana, and C. Ribeiro. Ciia: Critical infrastructure impact assessment. In *Proceedings of the 37th ACM/SIGAPP Symposium on Applied Computing*, SAC '22, page 124–132, New York, NY, USA, 2022. Association for Computing Machinery.
3. M. M. Chowdhury, H. Raddatz, and J. E. Rossebo. Challenges when securing manufacturing message service in legacy industrial control systems. In *Proceedings of the 2014 IEEE Emerging Technology and Factory Automation (ETFA)*, pages 1–6. IEEE, 2014.
4. D. Dzung, M. Naedele, T. Von Hoff, and M. Crevatin. Security for industrial communication systems. *Proceedings of the IEEE*, 93(6):1152–1177, 2005.

[4] An example of correlation of both systems is available on the SATIE youtube channel: `https://www.youtube.com/watch?v=z9bVvnZs6YY`

5. E. Esiner, D. Mashima, B. Chen, Z. Kalbarczyk, and D. Nicol. F-Pro: A fast and flexible provenance-aware message authentication scheme for smart grid. In *2019 IEEE International Conference on Communications, Control, and Computing Technologies for Smart Grids (SmartGridComm)*, pages 1–7. IEEE, 2019.
6. M. Falvo, F. Santi, R. Acri, and E. Manzan. Sustainable airports and nzeb: The real case of rome international airport. In *2015 IEEE 15th International Conference on Environment and Electrical Engineering (EEEIC)*, pages 1492–1497, 2015.
7. B. Genge, H. Piroska, and H. Sándor. PROTECT-G: Protection of communications in natural gas transportation systems. In *2018 6th International Symposium on Digital Forensic and Security (ISDFS)*, pages 1–5. IEEE, 2018.
8. V. Graveto, L. Rosa, T. Cruz, and P. Simões. A stealth monitoring mechanism for cyber-physical systems. *International Journal of Critical Infrastructure Protection*, 24:126–143, 2019. Publisher: Elsevier.
9. R. Khan, K. Mclaughlin, D. Laverty, and S. Sezer. Design and implementation of security gateway for synchrophasor based real-time control and monitoring in smart grid. *Ieee Access*, 5:11626–11644, 2017. Publisher: IEEE.
10. D. Li, H. Guo, J. Zhou, L. Zhou, and J. W. Wong. SCADAWall: A CPI-enabled firewall model for SCADA security. *Computers & Security*, 80:134–154, 2019. Publisher: Elsevier.
11. P. Likhar and R. S. Yadav. Stealth Firewall: Invisible Wall for Network Security. In *Innovations in Computer Science and Engineering*, pages 413–421. Springer, 2020.
12. J. Lima, F. Apolinário, N. Escravana, and C. Ribeiro. BP-IDS: Using business process specification to leverage intrusion detection in critical infrastructures. In *31st IEEE International Symposium on Software Reliability Engineering (ISSRE 2020)*, 2020.
13. P. Plesowicz. A15: Secure signal tunneling for SCADA and PLCs using SSH protocol. *IFAC Proceedings Volumes*, 37(20):88–93, 2004. Publisher: Elsevier.
14. S. Rahimi and M. Zargham. Security Analysis of VPN Configurations in Industrial Control Environments. In *International Conference on Critical Infrastructure Protection*, pages 73–88. Springer, 2011.
15. F. Reuschling, N. Carstengerdes, T. H. Stelkens-Kobsch, K. Burke, T. Oudin, M. Schaper, F. Apolinário, I. Praça, and L. Perlepes. Toolkit to enhance cyber-physical security of critical infrastructures in air transport. *Cyber-Physical Threat Intelligence for Critical Infrastructures Security*, pages 254–287, 2021.
16. K. Sampigethaya and R. Poovendran. Aviation Cyber–Physical Systems: Foundations for Future Aircraft and Air Transport. *Proceedings of the IEEE*, 101(8):1834–1855, Aug. 2013.
17. A. A. Soares, D. M. Mattos, Y. Lopes, D. S. Medeiros, N. C. Fernandes, and D. C. Muchaluat-Saade. An Efficient Authentication Mechanism based on Software-Defined Networks for Electric Vehicles. In *2019 IEEE 28th International Symposium on Industrial Electronics (ISIE)*, pages 2471–2476. IEEE, 2019.
18. K. Stouffer, S. Lightman, V. Pillitteri, M. Abrams, and A. Hahn. Guide to Industrial Control Systems (ICS) Security. Technical Report NIST Special Publication (SP) 800-82 Rev. 2, National Institute of Standards and Technology, June 2015.
19. P. P. Tsang and S. W. Smith. YASIR: A low-latency, high-integrity security retrofit for legacy SCADA systems. In *IFIP International Information Security Conference*, pages 445–459. Springer, 2008.
20. B. Willemsen and M. Cadee. Extending the airport boundary: Connecting physical security and cybersecurity. *Journal of Airport Management*, 12(3):236–247, 2018.