A Cyber Security Detection Framework for Supervisory Control and Data Acquisition Systems

Abstract—This paper presents a Distributed Intrusion Detection System (DIDS) for Supervisory Control and Data Acquisition (SCADA) Industrial Control Systems (ICS) that was developed for the CockpitCI project. Its architecture was designed to address the specific characteristics and requirements for SCADA cyber security that cannot be adequately fulfilled by techniques from the Information Technology (IT) world, thus requiring a domain-specific approach. DIDS components are described in terms of their functionality, operation, integration and management. Moreover, system evaluation and validation are undertaken within a specially designed hybrid testbed emulating the SCADA system for an electrical distribution grid.

I. INTRODUCTION

Cyber security is currently one of the main concerns for SCADA ICS operators, as a result of a series of recent successful cyber attacks against several targets, such as electric power substations and distribution grids [1], sewage processing units or even nuclear power plants [2]. These have had far reaching impact, affecting a substantial number of persons, potentially causing significant damage and ultimately threatening human lives [3]. Post-event investigation has frequently linked these attacks to the exploitation of vulnerabilities deeply rooted in the ICS design philosophy and related technologies.

The first generations of SCADA systems relied on air gapped isolation, together with a security-by-obscurity approach, based on proprietary or poorly documented technologies to protect the infrastructure. Later on, the introduction of features from the information and communication technologies (ICT) world, such as Ethernet networks and Transmission Control Protocol/Internet Protocol (TCP/IP) stacks, encouraged the interconnection of the ICS and ICT network domains, as well as to the exterior (mostly for remote management) [4].

Increased connectivity, together with the adoption of open and documented protocols exposed the SCADA ICS "black box" to a variety of new threats. Due to their importance, those systems rapidly became popular targets for all sorts of frivolous, ego-driven or vindictive attacks [5]. They have even subject to Advanced Persistent Threats (APTs) operating over extended time periods as part of cyber warfare strategies designed to probe and profile potential targets, with the aim of extracting valuable intelligence data which can be later used to attack civilian or military infrastructures. The Stuxnet [5] worm is such an example, which has raised awareness for the problems of Critical Infrastructure (CI) protection.

A. Motivation and Contributions

While empirical reasoning might suggest otherwise, the two aforementioned domains are significantly different, mainly due to the increasing number of ICT-related technologies assimilated within ICS. ICS operation and design principles follow an order of priorities which places availability and reliability first, followed by information integrity and, finally, confidentiality and security, while ICT follows the exact reverse order [6]. This is considered one of the primary reasons behind SCADA infrastructure security issues, also explaining the need for domain-specific cyber security mechanisms.

Recognizing the need to improve upon existing security solutions for SCADA ICS, the CockpitCI project was set up in order to address the problems of CI interdependence and security in an integrated way. Its objectives included the development and implementation of a cyber security detection framework for SCADA ICS, which is the main subject of this paper. This framework departs from the state-of-the-art in ICS cyber security, by providing a series of benefits, namely:

- **Diversified and innovative probing techniques**, supporting different types of security agents, tuned for each specific ICS domain, together with the development of innovative network, device and process-level security probes, aimed at reinforcing the detection capabilities beyond the existing solutions;  
- **Hybrid analysis capabilities**, through integration of both signature and anomaly-based techniques, providing the ability to deal with both known and rogue threats;  
- **Integration, scalability and reliability**, adopting a distributed multi-layered design, geared towards scalability and reliability, using message queues to relay normalized events between components.

This solution was also designed with existing infrastructures in mind, by supporting mature technologies and taking advantage of already deployed management and operations support systems to provide topology and asset information.

This paper is organized as follows: the next section will provide an overview of SCADA security, also addressing proposals and initiatives to deal with existing issues. Section 3 presents the CockpitCI cyber detection architecture for SCADA ICS, followed by Sections 4 and 5, detailing the analysis mechanisms and detection agents within the platform, respectively. Section 6 addresses the validation of the proposed solution, the architecture of the testbed used for that purpose, the test methodologies and results. Finally, Section 7 concludes the paper, also presenting directions for future developments.

II. SCADA SECURITY: AN OVERVIEW

SCADA ICS security encompasses aspects as diverse as the protection of physical infrastructures and processes, communications protocols, asset management or software and hardware
lifecycle management, which cannot be handled in the same way as their ICT counterparts.

As an example, lifecycle management procedures that are commonplace in the ICT world, such as patching and updating a system, require a different approach for ICS. With availability being a top priority (components are frequently kept in operation for years without re-initialization), the impediment/high cost of stopping production in case of (un)planned downtime has prompted some ICS managers to neglect routine updates or patching [7]. Moreover, since software releases are subject to prior certification by equipment manufacturers and solution providers, it might not be possible to deploy an operating system patch or upgrade any other key component, as it has not yet been tested for compliance. This is not even an option for products beyond their support lifespan.

Another distinguishable issue has to do with the deployment of security-related components. Due to the time-sensitive nature of SCADA systems, several equipment and software providers advise against installing anti-virus software in computers hosting Human-Machine Interface (HMI) or server hosts, due to latency concerns. Conventional IT Network Intrusion Detection Systems (NIDS) face similar issues, being considered unsuitable for inline deployment, as they constitute an unwanted point-of-failure, which can also contribute to deteriorate communications latency (a critical aspect, especially for systems dealing with strict, real-time, control).

Also, most mature SCADA protocols, that support the communication between devices, such as Programmable Logic Controllers (PLCs) or Remote Terminal Units (RTUs) and the supervisory stations, are insecure. For instance, the Modbus protocol, which was introduced by Modicon in 1979, is still being widely used and supported for new product lines. Its simplicity, which is one of the main reasons for its popularity, also makes it insecure, as it does not support encryption or any other protection mechanisms. Attempts to update this protocol with encryption or authentication features were unsuccessful, owing to increased hardware and computational overhead on both the PLCs/RTUs [8] and master stations [9], while also being incompatible with existing equipment.

These examples illustrate some of the reasons behind the proliferation of unprotected, unpatched or unsupported systems within SCADA ICS, which are potentially vulnerable to cyber threats. Together with the constraints on the implementation of counter-measures [10] this situation has made evident the need for a domain-specific approach for ICS security, a subject next discussed.

A. Towards cyber security awareness

At first, several SCADA operators tended to overlook security issues, mainly because it was deemed too costly and complex to update entire infrastructures, just to protect them from seemingly far-fetched threats. This applied all the more, when considering the potential implications of these updates, such as going through planned downtime procedures and further expenditure or abandoning proven and mature platforms, valued for their reliability.

Once cyber security awareness started to grow among the ICS community (a situation partly owing to the increase of ICS-related security incidents), organizations in industry, researchers and governments began working towards improving security in SCADA systems. For example, the National Institute of Standards and Technology (NIST) has published a reference document [11] that provides an overview of ICS threats and vulnerabilities, recommending adequate countermeasures and policies. The Industry Standards Organization (ISA) also published similar guidelines [6], as well as other organizations such as the North American Energy Reliability Committee (NERC) and the International Electrotechnical Commission (IEC), which have published recommendations for infrastructure protection for electric production and distribution [12], as well as the U.S. Nuclear Regulatory Committee (USNRC) regulations for nuclear infrastructures [13], just to mention a few. ICS cyber security issues were also addressed as part of wide scale-efforts developed within European FP7 projects such as CRISALIS, PRECYSE and VIKING [10].

There has also been considerable amount of work regarding SCADA intrusion and anomaly detection, including Intrusion Detection Systems (IDS) [14], device-level anomaly detection and classification [15]. IDS solutions combining network traces and physical process control data [16], detection based on traffic and protocol models [17] and approaches based on machine-learning techniques [18], among others. Acknowledging the fact that each detection technique has its own merits and scope, the CockpitCI architecture was built on the idea that the whole is greater than the sum of the parts, by bringing several of these techniques together within an inclusive detection framework, which is described next.

III. THE COCKPITCI DISTRIBUTED IDS ARCHITECTURE

One of the primary goals of the CockpitCI project was building effective cyber detection capabilities for SCADA ICS, integrated within a Dynamic Perimeter Intrusion Detection System (PIDS), which would be responsible for continuously assessing the electronic security perimeter of the CI.

---

Figure 1. High-level view of the CockpitCI PIDS architecture

The PIDS (see Figure 1) is organized along the distinct network domains of the CI, each one with a well-defined
security scope: Control, Process Control, and IT (information technology). It is able to enforce security policies customized to the particular characteristics of each domain, each having its own analysis mechanisms, fed by several detection agents. This architecture was designed to integrate different detection strategies, namely:

- **Detection agents**, comprising specialized network probes (such as NIDS), domain-specific honeypots [19], host-level agents (using HIDS–Host IDS and other agents) and specialized device monitoring components;

- A **distributed multi-zone, multi-level correlation structure**, processing the event feed from the detection sensor layer, complemented by machine-learning capabilities, in the form of One-Class Support Vector Machine (OCSVM) adaptive anomaly detection modules;

- **Usage of topology and system-specific detection mechanisms**, due to the fact that the role, placement and behaviour of ICS components tend to remain unchanged over larger time spans than their ICT counterparts. Therefore, the analysis layer is also provided with information from topology databases or asset management systems.

The PIDS operation is orchestrated and managed by means of a Security Management Platform (SMP – see Figure 2), which oversees the maintenance and management of detection agents and analysis components (correlators, OCSVM analysis), whilst also performing regular in-place security auditing of the protected infrastructure. The SMP adjusts the detection agent sensitivity threshold according to the overall risk level of the infrastructure (adjusted to the estimated probability of an attack event occurring) and also feeds other CockpitCI system components with information from the detection layer, through a secure mediation gateway.

Figure 2. CockpitCI PIDS orchestration

Event messages exchanged between PIDS components are encoded using the Intrusion Detection Message Exchange Format (IDMEF - RFC4765)², which establishes a neutral format for intrusion detection related events. Also, the baseline IDMEF schema is encoded using the Extensible Markup Language (XML), allowing it to be updated or customized without disrupting compatibility.

IDMEF event messages generated by detection agents within each domain are published to a broker that routes them to a message queue subscribed by a corresponding local correlator. Once correlation is finished, each local correlator publishes the results to a different queue subscribed by the main correlator - the same queue is also fed by OCSVM components. Finally, the events generated by the main correlator are queued to be sent to the SMP. Each queue (an "Event Bus") is implemented using the Advanced Message Queuing Protocol (AMQP), which defines a neutral (IDMEF-compatible) message encoding method, whilst also providing secure and reliable transport with high-availability features.

### IV. Analysis Components

The PIDS analysis capabilities are provided by two types of components: the correlators and OCSVM modules. Together, they bring the benefits of both signature-based and anomaly detection techniques within the cyber detection framework. This section analyzes these mechanisms in detail.

#### A. IT-OCSVM

When designing an IDS for ICS, developers typically have to face a difficult situation in that it is not easy to find datasets that contain malicious traffic in order to train a typical machine learning algorithm properly. On the other hand, the datasets that can be collected from a real Industrial network may contain traffic that is not legitimate, while the system is agnostic of this. In order to overcome these issues, the One-Class SVM (OCSVM) has become a widely applied method for outlier detection [20]. It can be trained with datasets that contain only legitimate traffic. Once the model is properly trained, it can detect any misbehavior of the system that is due to possible attacks. The OCSVM classification technique can detect novel attacks, as long as they produce network traffic, and is capable of performing real time classification, due to its low computational time.

The OCSVM method has some drawbacks though, like production of high false positives, difficulty to cope with different nature of attacks, induction of high overhead and the inability to classify the attacks. All these issues reveal the need for the creation of more sophisticated detection methods [21] for SCADA systems. Based on these findings we have developed and validated a new integrated detection mechanism, namely the IT-OCSVM [18].

Figure 3. Architecture of the IT-OCSVM detection mechanism

The novel IT-OCSVM method is capable of performing outlier detection with high accuracy and low overhead within a temporal window, adequate for the nature of SCADA systems. In order to achieve these goals, the proposed method consists...
of several distinct stages that need to be carried out, as shown in Figure 3. The first, similar to any other SVM methods, consists of choosing the most appropriate features for the training of the OCSVM models and the pre-processing of raw input data. In IT-OCSVM alongside the central OCSVM a cluster of split OCSVM models is created.

In the core of the novel IT-OCSVM mechanism lies the OCSVM algorithm which is used in order to map the input data, the network traffic, into a high dimensional feature space (using a Kernel) and through iterations tries to find the maximal margin hyperplane in order to separate the training data from the origin in an optimum way. The OCSVM can be viewed as a two class SVM where all the training data lie on the one class and the origin is the only member of the second class. The hyperplane that is created using the following classification rule:

$$f(x) = <w,x> + b$$  \hspace{1cm} (1)

where w is the normal vector and b is a bias term. By solving an optimization problem, OCSVM tries to find the rule f that creates the maximal margin between the origin and the training data. Using the same rule on real time test data x we can assign a label f(x), which categorizes the data as anomaly when the value of f(x) is negative. On the IT-OCSVM detection agent several OCSVM algorithms run in parallel on separate training datasets, labeling the data as normal or anomalous.

The next stage is the assembling of the information gathered by the discrete OCSVMs. The outcomes of both the central OCSVM and the split OCSVMs are combined by using a simple mean value method. They are then given additional weights through the use of Social Network Analysis (SNA) metrics (Equation 2).

$$q_e(i,j) = \sum_{n=1}^{N} w_n d_e(i,j)$$  \hspace{1cm} (2)

where, d_e(i,j) is the outcome of each individual classifier n for sample data i that originates from node j and w_n is the weight given to each classifier.

More specific, Spearman’s rank correlation coefficient is used in order to add weight to alerts produced from different sources based on the protocols that each source uses during the training and the testing phase (Equation 3).

$$p = 1 - \frac{6 \sum d_i^2}{n(n^2-1)}$$  \hspace{1cm} (3)

The new outcomes are further aggregated and classified into categories, using a two-stage K-means clustering algorithm. During the first stage of fusion individual alerts are grouped together according to their origin. Each alert i has an initial value q_{i,j} (and a source j) based on the procedures described above. Using the equations from (4), each aggregated alert j is assigned two values qa_{j} and qb_{j} that represent its severity. This severity comes from both the sum of the values of the initial alerts and the number of attacks that originate from the same node, thus indicating that this node probably misbehaves.

$$qa_j = \sum_i q_e(i,j), \hspace{0.5cm} qb_j = \sum_i 1, \forall q_e(i,j) \text{ with source } j$$  \hspace{1cm} (4)

During the second stage of fusion K-means clustering is used in order to divide the alarms into: possible, medium and severe (See Equations 5 and 6).

$$SSE = \sum_{k=1}^{K} \sum_{j=1}^{N_k} ||qa_j - \mu_k||^2$$  \hspace{1cm} (5)

where, N_k is the number of instances belonging to cluster k and \mu_k is the mean of cluster k, calculated as the mean of all the instances belonging to the cluster i

$$\mu_{k,i} = \frac{1}{N_k} \sum_{q=1}^{N_k} qa_{q}, j \forall j$$  \hspace{1cm} (6)

We use two k-means algorithms that run in parallel and each alarm is assigned a final classification: possible, medium or severe. These alarms are sent to the correlator using IDMEF messages containing information such as the nature of the alert, the location of the attacker in the system and the time of detection. The proposed IT-OCSVM analysis mechanism runs in a distributed way and can be used in any SCADA system after proper training. The performance of the mechanism is enhanced by the combination of SNA metrics with machine learning techniques and can be easily incorporated into a soft real time detection system, due to its low overhead.

**B. Correlation**

As shown in Figure 1, local correlators provide the first event processing stage, being fed by the detection agents deployed within their network domain and sending their output to the main correlator in the form of IDMEF messages. Their role is to filter, process and relate security events within a network domain, also delivering noise filtering and event reduction capabilities, aggregating alerts produced by several detection agents or multiple events from a single source.

This approach provides context separation, whilst at the same time allowing for increased scalability and efficiency, as the main correlator can concentrate on multi-step, attack focus recognition correlation, as well as alert prioritization.

The generic PIDS correlator architecture (see Figure 4) encompasses two event interfaces: one to receive events (which may arrive from detection agents, on local correlators or from the local correlators, on the main correlator), and another to send resulting events (to the main correlator, on local correlators or to the SMP, on the main correlator), both using an AMQP-based Event Bus.

The correlator components use the Esper Complex Event Processing (CEP) tool, a flexible and mature event processor, which exhibits a good equilibrium between CPU, memory usage and execution speed in demanding situations. It natively accepts XML-encoded events, which is convenient as IDMEF is an XML-based format. Moreover, it is able to perform correlation over sliding time windows using several data sources,
such as input events, persisted and cached information as well as external data sources such as asset management platforms and topology databases. Correlation rules are encoded using Event Processing Language (EPL) statements.

The same architecture is used for both the local and main correlators. Not only does this provide design uniformity, but it also eases management and maintenance, as all correlators share the same rule syntax, thus allowing for rules to be reused across different contexts.

V. DETECTION AGENTS

Detection agents constitute the lowest layer of the PIDS architecture, whose role is to collect evidence of suspicious activity directly from the ICS infrastructure elements used for security analysis. They gather information, normalizing and feeding it to the local correlator for its network domain. The PIDS includes a diversity of security probes and detection agents, including (but not restricted to) the following:

- **Network IDS**: each network domain perimeter is monitored using separate NIDS instances for IT, Process Control, and Control Networks; feeding security events to the respective zone correlator. Snort was used for this purpose, but other NIDS can be integrated;
- **Host IDS**: Host IDS are deployed in the ICS stations/servers, providing host-level anomalous behaviour detection. OSSEC was adopted for this purpose, although different HIDS can also be used;
- **Honeypots**: these components act as decoys, to lure and uncover attackers. The PIDS encompasses three types of honeypots, one for each ICS network domain [22];
- **Shadow Security Unit**: a security and safety device that is able to monitor a PLC or RTU continuously [23];
- **Exec Checker**: by passively intercepting network traffic, the Exec Checker is able to capture and reassemble a suspicious binary file, in order for it to be analyzed;
- **Output Traffic Controls**: this tool analyses network usage profiles in order to detect the presence of Remote Access Trojans, regularly scanning system components to search for known toolbox signatures;
- **Configuration Checker**: this tool implements system configuration monitoring, checking for unauthorized modification or corruption of configuration data sources.
- **Behavior checker**: this tool monitors and analyzes the behavior of low-level software and hardware indicators, such as temperature and processor activity, in order to avert or anticipate accidental or intentional disruption.

All detection agents publish IDMEF security events on the Event Bus used by its local correlator and since each detection tool has its own eventing format, a generic embedded adaptor was developed to perform normalization. A uniform interface provides agent management capabilities, via the SMP.

Among these detection agents, the Shadow Security Unit and the SCADA Honeypot deserve particular attention, due to their innovative nature. Both will be next analyzed in detail.

A. Shadow Security Unit

The Shadow Security Unit (SSU) [23] was developed to address the protection of RTU and PLC devices within existing SCADA systems. This was deemed a critical PIDS requisite, due to reasons such as the need to deal with zero-day vulnerabilities or because mature device families with a poor security record often have a large deployed base, the updating or replacement of which is considered unfeasible.

The SSU (see Figure 5) is a device attached in parallel to RTU/PLCs that is able to intercept and decode inflight SCADA Protocol Data Units (PDU), correlating extracted data with the state of the physical I/O channels interfacing with field actuators and sensors. It is a security and safety probe, providing continuous behavior and health device monitoring.

B. SCADA Honeypot

One of the innovations of the CockpitCI PIDS is the SCADA honeypot [22], conceived for deployment in the process control network of a SCADA ICS, operating side-by-side with the PLCs, RTUs and other devices, using the network’s unallocated IP addresses.
The SCADA honeypot constitutes a decoy for potential attackers, attracting their attention and reporting any suspicious interactions. This is achieved by emulating both the functionality and service set of a commercial PLC, with the purpose of engaging the attacker in intensive probing activities, whose traces can be used for profiling purposes. Its components comprise a fully functional Modbus TCP emulator, including register banks and functions, further complemented by other protocol modules providing file transfer and management services, which are commonplace in commercial PLCs, as well as network portscan detection capabilities.

VI. DEPLOYMENT AND EVALUATION

This section details the PIDS validation process, comprising the testbed architecture, IT-OCSVM performance, functional validation use cases, and the event processing capacity tests undertaken to estimate the scalability of the solution.

A. IT-OCSVM initial validation

In order to validate the performance of the IT-OCSVM mechanism that is a core part of the PIDS, a small-scale testbed was created, providing the means to mimic a SCADA system operating both in normal conditions and under the influence of cyberattacks. The testbed architecture was comprised by process control and control network elements, which included a HMI station (for process monitoring), a managed switch (with port monitoring capabilities for network traffic capture), and two PLC units, for process control.

Tests evaluated the performance of the IT-OCSVM mechanism using three types of attacks (later described in more detail in this chapter), namely: network scan, Man-in-the-Middle (MITM) using ARP spoofing and Denial-of-Service (DoS). In all three scenarios the IT-OCSVM module was fed with network traffic and performed data analysis in order to detect malicious behavior. The testing data consisted of normal and attack data and the composition of the data sets is as follows:

- Testing set-A: Normal data
- Testing set-B: ARP spoofing attack + Network scan
- Testing set-C: Flooding DoS attack + Network scan
- Testing set-D: MITM attack

The proposed IT-OCSVM detection mechanism exhibited adequate performance in all attack scenarios, while managing to achieve both high detection accuracy (DA) and low false alarm rates (FAR), as shown on Table I.

```
<table>
<thead>
<tr>
<th>Dataset</th>
<th>DA</th>
<th>FAR</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>98.81%</td>
<td>1.18%</td>
</tr>
<tr>
<td>B</td>
<td>94.6%</td>
<td>5.4%</td>
</tr>
<tr>
<td>C</td>
<td>95.29%</td>
<td>4.71%</td>
</tr>
<tr>
<td>D</td>
<td>96.37%</td>
<td>3.63%</td>
</tr>
<tr>
<td>All</td>
<td>96.3%</td>
<td>2.5%</td>
</tr>
</tbody>
</table>
```

Based on the performance exhibited by the IT-OCSVM module, it was decided to deploy it along with the rest of the PIDS components on the platform validation testbed, described on the following subsection.

B. The HEDVa testbed

The Hybrid Environment for Development and Validation (HEDVa) was designed by the Israeli Electric Company for the development and validation of CIs as well as security research projects, being used to implement the attack use cases for the PIDS validation effort. Its architecture (as Figure 6) constitutes a hybrid approach, because of the existence of real/physical SCADA and network/telecom infrastructure components, which are used to implement a simulation model of the electric grid elements.

![Figure 6. HEDVa networking architecture](image)

The CockpitCI emulated HEDVa scenario corresponds to an energy distribution grid, as shown in Figure 7. Its implementation required the development and testing of different modeling techniques, also including related key performance indicators, fine-tuned with data from production scenarios. Grid elements such as breakers and feeders are controlled by real PLCs in a simulation model, with voltage and current values for segments being calculated accordingly using a mathematical model of the physical grid. In case of failure, the scenario is reconfigured accordingly with the operator’s Fault Isolation and Service Restoration procedures.

C. Functional validation

This subsection presents the most relevant use cases for PIDS validation, which were implemented on the HEDVa.

1) Layer 2/3/4 attacks: These scenarios correspond to malicious activities targeting the SCADA network resources associated to the data-link (2), network (3) and transport layers (4) of the OSI model. For this category, two validation use cases were chosen: a FIN/SYN Scan and a SYN Flooding.

A port scan is a typical first step for an attacker to obtain information about the network’s hosts and its topology, using...
techniques such as SYN, ACK or FIN scans (among others). This use case was validated on the HEDVa using the nmap tool to send a series of SYN and FIN probes (see Figure 8) to devices on a network scope, to scan for open TCP/IP ports.

![Figure 7. HEDVa grid scenario, with breakers and substation feeders](image)

**The SYN flooding** Denial-of-Service (DoS) attack abuses TCP connections, by sending a high volume of SYN packets to the PLC/RTU. This creates a large number of half-open connections on the target device, leading to resource exhaustion and the inability to answer new requests. For validation purposes, SYN flood attacks were implemented using the hping tool, against a PLC and an HMI, causing loss of process visibility.

Functional validation demonstrated the effectiveness of several PIDS components against both use cases:
- the NIDS and the OCSVM modules, which detect the corresponding abnormal traffic patterns;
- the SSU authorized device validation and communications stream analysis modules, that detect the anomalous traffic patterns, over a time window (see Figure 8);
- the SCADA Honeypot, which flags up abnormal interactions.

Moreover, in the event that the FIN/SYN packets are forged from a legitimate host, it becomes possible to detect the attack, because of the SSU auto-similar flow pattern analysis features.

![Figure 8. TCP FIN scan](image)

2) **SCADA protocol and process-level attacks:** SCADA protocols can be specifically targeted by corruption, interception or tampering attacks. These can potentially lead to a loss of confidentiality, visibility, or device connectivity, also providing support to implement process-aware attacks, thus compromising safety and security. A Man-in-the-Middle (MITM) attack using Address Resolution Protocol (ARP) poisoning was chosen as the validation use case for this category, exercising protocol and process-level detection capabilities.

The MITM use case starts with an intruder host within the HEDVa Control Network sending unrequested ARP replies to poison the targets’ ARP caches and hijack the Modbus TCP protocol flows between the HMI 1 system and the PLCs; this allows for the attacker to eavesdrop communications and gather information about the SCADA network and the process. Afterwards, the attacker will use the previously gathered information to craft a scripted real-time simulator using the Scapy tool (see Figure 9 - green flows), reproducing the normal operation of the simulated grid and providing nominal status data back to the HMI 1, without any visible supervisory feedback.

![Figure 9. ARP-based, Man-in-the-Middle attack against PLC+SSU](image)

**Figure 9. ARP-based, Man-in-the-Middle attack against PLC+SSU**

Because the HEDVa grid model depends on the coordinated operation of all PLCs, the attacker scripts are able to respond to HMI operations and react accordingly, simulating the correct reaction from the PLCs – as such, when the HMI operator operates a specific breaker (managed by a PLC), the entire view is updated with the correct energy values of affected segments. This requires the attacker to properly handle all the TCP and Modbus protocol semantics in real time, in order to keep communication integrity and remain undisclosed. Functional validation demonstrated the capability of several PIDS components for detecting the MITM attack, namely:

- the NIDS, which detect network traffic coming from a station not registered in the asset management or topology databases, together with an anomalous ARP reply traffic pattern (inter-arrival times below normal thresholds);
- the Control Network OCSVM module, that detects an ARP packet rate above the classification threshold;
- the SCADA Honeypot, which detects unrequested ARP replies on the Control Network domain;
- the SSU station validation and high-level command flow processing modules, which detect Modbus command pat-
terns deviating from normal sequences (modeled using Markov chains), eventually involving unknown peers;
- the SSU message checker, that detects inconsistencies between the HMI 1 interactions and Modbus commands arriving at the PLC (see Figure 9).

3) Compromised HMI/PLC: This scenario corresponds to a situation in which an HMI or a PLC becomes compromised. For functional validation purposes, two cases were considered: an HMI infected with malicious software and a PLC reprogrammed by an attacker to disrupt a controlled process.

The HMI malware infection attack was performed using the metasploit tool to inject a custom malware payload, abusing an Operating System vulnerability allowing for remote execution of binaries. This exploit was used to modify a series of Basic Control Engine script files from the Cimplicity HMI software to hide malicious activity, tampering with the events used to update the HMI, while directly manipulating PLCs.

After the malware infection, the HIDS detected a change on HMI file set signatures, together with an unknown process, while the Exec Checker was able to detect the presence of an executable payload in transit for HMI 1. This was further confirmed by the presence of anomalous traffic patterns after the malware infection, because of Command and Control interaction (reported from the NIDS, OCSVM and output traffic controls). Moreover, the behaviour analysis and command interception features of the SSU detected anomalous command patterns, despite the fact that they were coming from a legitimate source. The SMP security auditing module could have prevented this attack, since it detected the exploited vulnerability, which was left unpatched for test purposes.

The PLC reprogramming attack was implemented using a vulnerability that allows for unauthorized upload and download of ladder logic code on Schneider PLCs [25]. This was used to reprogram one of the PLCs to flip the state of a breaker while the Exec Checker was able to detect the presence of an executable payload in transit for HMI 1. This was further confirmed by the presence of anomalous traffic patterns after the malware infection, because of Command and Control interaction (reported from the NIDS, OCSVM and output traffic controls). Moreover, the behaviour analysis and command interception features of the SSU detected anomalous command patterns, despite the fact that they were coming from a legitimate source. The SMP security auditing module could have prevented this attack, since it detected the exploited vulnerability, which was left unpatched for test purposes.

While the abuse of function code 90 (0x5a) can be detected on the NIDS (via signature rules or unauthorized flow detection), there is a potential contention problem if the switch monitor port where the NIDS is connected receives traffic from other interfaces, all running at the same speed – this may cause packet loss on the NIDS port. For this specific attack use case, the SSU provides two effective mechanisms: command stream correlation (which captures the anomalous interaction) and also behaviour checking, which compares the behaviour of the PLC (command interactions and physical I/O ports) against a model of its nominal operation mode, enabling it to detect an anomalous switching rate on a physical I/O channel. Further information about the performance of the PIDS can be found on the public deliverables of the CockpitCI project 3 4.

D. Event processing capacity evaluation

To evaluate the capacity of the event transport and correlator components, a series of off-band performance tests were executed on a validation testbed (see Figure 10), composed of three VMWare ESXi 5.5 virtual machines running the CentOS 6.4 Linux OS: an event publisher (simulating a PIDS agent), an event processing node (with the RabbitMQ broker and the PIDS correlator), and a consumer application for events.

The VM host is equipped with an Intel Xeon X5660 6-core CPU running at 2.80GHz, and 32GB RAM. Core and resource affinity parameters were tuned to avoid contention: VMs for the event publisher and consumer used 1 core and 4GB RAM each, while the event processing host was configured with 2 processor cores and 6GB RAM - all VMs have equal priority I/O shares, for a non-SSD datastore. Network connectivity is provided using a virtual switch running at host speed, reducing the physical network latency and speed overhead issues.

The event processing node consumes events from queue A on the broker and then, after processing, publishes results to queue B. The correlator was configured with 20 statement/rules, for testing purposes - all received events trigger a rule and there is no second order correlation. For timing purposes, the publisher records a timestamp on the IDMEF messages, enabling the consumer to assess processing delay by calculating the delta relative to the reception time of the corresponding resulting event. System clocks on all hosts are synchronized using the Network Time Protocol (NTP).

Validation results will be next presented, with a focus on event processing performance and CPU/memory overhead.

1) Event processing performance: The event processing capacity tests for the correlation component were executed using IDMEF event sizes ranging from 1 (minimum size payload, with mandatory attributes) to 20 KB (corresponding to a complex meta-event with several references). Tests also evaluate the overhead for the queue acknowledgment mode, which can either be enabled, so that the publisher receives confirmation that the message was received by the consumer or persisted to disk by the broker, or in AutoAck mode, in which case the broker issues an acknowledgment as soon it receives a message. Encryption overhead was also measured.

Figure 11 shows average results for 15 test rounds (standard deviation between 4 and 11%) of 10,000 event bursts, executed with persistence support so that the broker saves messages to mass storage when they cannot be consumed right away, allowing reliable delivery, albeit with a performance penalty. The results show the event processing rate to decrease with the message size, as larger messages take longer to send and marshall/unmarshall. Also, confirmation and encryption over-

---


head decrease along with the message size, with the former having a bigger (albeit increasingly insignificant) impact, when compared with the latter, especially for event sizes between 5 and 20KB. Results were considered adequate for PIDS usage.

2) CPU and memory overhead: CPU usage for Host 2 was continuously monitored during the tests. For 1, 2 and 5KB event bursts, the CPU capacity of the host was exhausted, a situation that is mainly attributed to the correlator, which alone takes an average 70% of total CPU time for each test case – for 10KB and 20KB, total CPU usage averaged at 82 and 80% respectively. These results can be improved: a physical networking infrastructure will have a mitigating effect on event bursts; also, as correlation is CPU-bound, a more capable VM hardware configuration will have a significant impact. Memory usage was stable at 810MB of RAM, approximately, with the correlator using an average 104MB RAM, with a maximum of 141MB - the rest was used by the RabbitMQ broker.

VII. CONCLUSION

This paper presented the architecture of the CockpitCI domain-specific IDS for SCADA ICS. Its design departs from the state-of-the-art in CI security, incorporating distributed multi-level correlation, together with domain-specific anomaly detection elements, leveraging the benefits of both signature-based and rogue threat detection. Also, the PIDS brings together a set of heterogeneous detection and analysis components within a common framework with normalized eventing, coordination and orchestration capabilities.

Future developments of the PIDS are focused on the evolution of the Honeypot and Shadow Security Units, as well as support for a more diversified set of SCADA technologies and protocols. Research into complementary analysis techniques is also undergoing, in order to improve detection capabilities.

ACKNOWLEDGMENT

This work was funded by the CockpitCI European FP7 project (FP7-SEC-2011-1 Project 285647).

REFERENCES


