Protecting Cyber Physical Production Systems using Anomaly Detection to enable Self-adaptation

Giuseppe Settanni, Florian Skopik, Anjeza Karaj, Markus Wurzenberger, Roman Fiedler
Center for Digital Safety and Security
AIT Austrian Institute of Technology - Vienna, Austria
firstname.lastname@ait.ac.at

***Abstract*—The industrial world is going through its fourth revolution also known as Industry 4.0. Modern industrial processes leverage advanced IT technologies to increase productivity and often combine multiple system concepts such as Internet of Things (IoT), Cyber Physical Systems (CPS) and Cloud Computing. Cyber Physical Production Systems (CPPS) are key enablers of this revolution. In CPPS, raw materials, machines, and operations are interconnected to form a sophisticated network. Protecting them against advanced cyber-threats is a priority concern for the future implementation of Industry 4.0 applications. Any impairment of such systems can lead, in fact, to catastrophic damages resulting in a substantial financial loss for governments, companies, as well as endanger the safety of the society. The need for high availability and reliability of these systems is therefore the pillar guiding our research. This paper proposes the adoption of anomaly detection as a method to support self-adaptation in CPPS and to ensure flexibility, reliability, and protection of industrial environments against modern cyber threats. An anomaly detection mechanism can be employed to monitor, and learn the normal behavior of an industrial system, and to generate alerts when the observed events indicate abnormal activities. On this concept we base our work, and we demonstrate how timely identifying critical security events can enable, through the self-adaptation (e.g., triggering automatic configuration changes), an efficient protection of the CPPS against advanced threats, and an effective containment of their effects.**

***Index Terms*—Cyber Physical Production Systems, Self-adaptation, Anomaly Detection, Industry 4.0.**

I. Introduction

Cyber-Physical Production Systems (CPPS) will become the backbone of modern industry. Protecting them against sophisticated cyber-threats is a priority concern for the future implementation of Industry 4.0, as numerous challenges related to data protection, safety, and security, become critical in this domain [1]. Reliability and availability of CPPS can be dangerously affected by the increasing number of cyber-attacks specifically targeting these systems. Traditional means of handling attacks and failures, relying on human intervention in industrial processes, do not provide appropriate protection in this new setup, where real-time operations and high availability are crucial requirements [2]. Moreover, the communication flows in this environment are highly heterogeneous and dynamic, because of the coexistence of legacy devices and systems embedding newer technologies. The industrial Internet is, in fact, more than a simple combination of Operational Technology (OT) and Information Technology (IT) within the digital industry. It involves Internet of Things, autonomous production, and CPS. This synergy creates numerous opportunities and advantages for industrial environments [3], however, a great complexity and sophistication implies an increasing attack surface and exposes this environment to several security challenges.

According to the Industrial Internet Consortium [4], a broad range of vulnerabilities can expose industrial endpoints (e.g., Programmable Logic Controllers (PLCs) and sensors) to security threats in areas spanning from change and configuration management, software development, and access control management. The American National Institute of Standards and Technology (NIST) catalogs threats potentially affecting industrial control systems in: control logic manipulation, malware, denial of control actions, and spoofed system status [5]. Additionally, security threats can be categorized according to the system layers they affect [6]. The *sensor and actuator* layer is usually targeted by brute force attacks, dictionary attacks, and power consumption attacks; the *network* layer is exposed to security threat related to communication processes such as replay attacks, data eavesdropping, collision attacks, jamming, flooding and wormhole attacks. On the *Control* layer security threats against the PLCs, remote terminal units or other devices can provoke de-synchronization, leading the systems to take undesired control decisions. Finally, on the *Information* layer eavesdropping and traffic analysis can be put in place to acquire sensitive data from the system.

These security threats need to be considered while designing the protection measures of a network infrastructure for Industry 4.0. Consequently, a comprehensive approach is required to protect such an infrastructure, which takes advantage of the distributed nature of CPPS, and leverages the integrated IT technologies not only to increase productivity, but also to enhance security [7]. Sophisticated mechanisms are to be employed to timely adjust the overall system's configuration and regulate the behavior of the different components deployed, in order for the CPPS to operate in its optimal condition. CPPS of the future will be able to self-configure, self-protect, self-heal and self-optimize. According to [8], self-adaptation is indeed one of the five areas that will have high priority in the future research for Industry 4.0. Self-adaptive CPPS flexibly and timely configure themselves and swiftly adjust to adverse and suboptimal conditions, guaranteeing the system to always operate above a predefined performance level.

In this work we demonstrate how anomaly detection methods allow to opportunely detect critical threats, and, based on a series of defined security metrics, it permits to instantiate the self-adaption process, hence containing the detected attack, and mitigating its impact on the CPPS.

The remainder of this paper is organized as follows. Section II presents a state of the art analysis of self-adaptation control methods, and gives a brief overview on existing anomaly detection techniques. In Section III we outline the application of self-adaptation to CPPS, illustrating the four phases of the MAPE-K cycle; moreover, we propose the adoption of anomaly detection mechanisms to support the monitoring and analysis phase of the self-adaptation process. In Section IV we describe the testbed we developed, as proof of concept, to validate the effectiveness of the proposed approach. Section V concludes the paper providing an outlook on future work.

II. Related Work

Security is a major concern for CPPS as they become more intelligent, interconnected, and coupled with physical devices. For various activities from security analysis to designing security controls and architectures, a systematic and structured view of security-related information is required. Approaches to establish a security viewpoint in the CPPS reference architecture model have been proposed by the scientific community, mostly based on the Reference Architecture Model for Industry 4.0 (RAMI 4.0) standard [9].

The risk architecture level of the RAMI 4.0 model includes vulnerabilities and threat catalogs, as well as safety and security components. We demonstrate in this paper that, within such components, anomaly detection can be adopted to monitor and analyze events generated by the CPPS and trigger alarms whenever suspicious activities are revealed, with the ultimate goal of timely reacting to cyber threats.

This approach enables CPPS to implement the self-adaptation paradigm [10]. A self-adaptive system is a system able to modify its behavior to meet predefined performance objectives [11]. In order to handle unexpected events, such as undesired changes in the system environment, failures, security-threats, etc., self-adaptive systems are able to monitor themselves, analyze the observed events, and autonomously put in place specific reaction procedures when necessary. Different approaches have been developed to achieve self-adaptive capabilities in control systems such as in [12] and [7]. Our research focuses on a self-adapting mechanism whose essential functions are described in the Monitor-Analyze-Plan-Execute over Knowledge-based (MAPE-K) reference model [13]. In particular we investigate, how anomaly detection techniques can support the monitoring phase and implement the analysis phase in the MAPE-K model.

There exist generally three detection methods usually applied in intrusion detection systems (IDS): signature-based detection (SD), stateful protocol analysis (SPA), and anomaly-based detection (AD) [14]. SD and SPA can only detect previously known attack patterns using signatures and rules that describe malicious events and thus are also called black-listing approaches [15]. AD approaches are more flexible and able to detect novel and previously unknown attacks. They establish a baseline of normal system behavior and therefore are also called white-listing approaches [16].

The rapidly changing cyber threat landscape demands for flexible and self-adaptive IDS approaches. One solution are self-learning AD based approaches that automatically learn the system behavior, and continuously adapt the corresponding model to reflect any system change; this serves as ground truth to detect anomalies that reveal attacks and especially intruders.

Generally, there are three ways to realize self-learning AD: *supervised*, *semi-supervised*, and *unsupervised* [17]. Unsupervised methods do not require any labeled data and are able to learn distinguishing normal from malicious system behavior during the training phase. Semi-supervised methods are applied when the training set only contains anomaly-free data; they are also known as 'one-class' classification. Supervised methods require a fully labeled training set containing both normal and malicious data.

In this paper, we propose a semi-supervised self-learning anomaly detection method (introduced in [18]) as means to reveal critical security events occurring in the CPPS, to allow the definition of relevant security metrics, and to enable the monitoring and analysis phases in the self-adaption cycle.

III. Self-adaptive CPPS

In this section we recall the phases comprising the MAPE-K cycle, we propose the adoption of this reference model as a suitable approach to facilitate self-adaptation in CPPS, and we illustrate how anomaly detection methods facilitate the phases of monitoring and analysis of this cycle.

A. MAPE-k Cycle

Cyber-physical systems (CPPS) seamlessly integrate computational and physical components. Adaptability, realized through feedback loops, is a key requirement to deal with uncertain operating conditions in CPPS. Among the existing models, the MAPE-K feedback loop (shown in Figure 1) is the most influential reference control model for autonomic and self-adaptive systems [13].

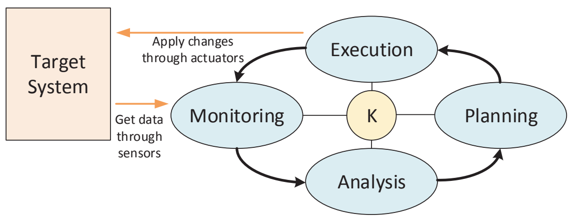


Fig. 1. MAPE-K cycle.

The *Knowledge* base (K) maintains data related to the target system and its environment, the adaptation goals, and other relevant information that are shared between all the MAPE phases. The *Monitoring* function (M) gathers particular data from the underlying target system and from the operational environment, through sensors deployed in the infrastructure. It stores the collected data in the shared knowledge base. The *Analysis* function (A) examines the data to check whether an adaptation is necessary. If so, it triggers the *Planning* function (P) that, following some predefined policies, composes a workflow of adaptation actions necessary to achieve the systems goals. These actions are then carried out by the *Execution* function (E), through effectors (or actuators) installed on the managed system. All these functions can communicate directly with one another or indirectly by sharing information in the knowledge base. The operations performed by M, A, P, and E may be executed by multiple components that coordinate with one another to adapt the system when needed, i.e., they may be decentralized throughout the multiple MAPE-K loops.

The main task of the monitoring function is collecting data captured by different sensors. The process of generating events requires data aggregation and filtering to determine what needs to be analyzed in the subsequent phase. Since hundreds of sensors can be placed in a production plant, it is crucial that unnecessary data, or data that does not carry any relevant information, is filtered out and not used for further analysis.

The analysis function is responsible to observe and analyze the output of the monitoring phase and determine if any change is required. Performance metrics are adopted to define the state of the system. If such metrics indicate that the system is operating in a sub-optimal condition, a change request, describing the modifications that need to be applied to the system, is generated and delivered to the planning phase.

The planning function selects one or more Self-Adaptation Policies (SAPs) to trigger a required action on the target system. Based on the incoming results of the analysis phase a change plan is generated, and delivered to the execution phase. SAP can be Event-Condition-Action (ECA) policies, goal policies, or utility function policies [12].

The execution function carries out the actions defined in the planning phase through effectors or actuators on the target system. The *Autonomic Manager*, responsible for the coordination of the execution phase, selects a self-adaptation policy corresponding to a change request, and specific actions are executed to opportunely modify the state of the system. The execution phase could involve updating the shared knowledge as part of the execution of the planned change.

The knowledge base is used to extend the capabilities of the self-adaptive mechanism. It can include policy knowledge, topology knowledge or problem determination knowledge.

B. Anomaly Detection to Enable Self-Adaptation

As previously mentioned, this work focuses specifically on the monitoring and analysis phase of the MAPE-K cycle, proposing anomaly detection as appropriate method to observe security events, and evaluate security metrics to steer the self-adaptation process of a target CPPS, with the ultimate goal of achieving flexibility while protecting the system against modern cyber threats. This section shows how the process presented in the previous section can be implemented to realize self-adaptation in CPPS and achieve greater security.

Monitoring events occurring in the CPPS is essential to capture necessary data and allow to detect suspicious activities, mitigate security threats, and assure the safety and security of the system. It is fundamental to select the optimal vantage points that provide monitoring of the CPPS with minimal redundancy. Normally, the monitoring process consists in monitoring the network traffic (e.g., passing through a switch), inbound and outbound connections traced by firewalls, database access, activity of end device controllers (such as PLCs), commands send from HMIs and workstations.

Event data generated and processed in a CPPS can be collected in different ways, depending on the data type and the collection mode. Different tools and standards are available for this purpose, such as TCPdump[1], Wireshark[2], Syslog logging[3], and Security Information and Event Management (SIEM) solutions (e.g., OSSIM[4]).

The main outcome of the monitoring phase is a series of events observed by the sensors, employed at diverse locations within the target system, which necessitate to be examined in order to identify potential security issues, and trigger the invocation of an adaptation policy. The following step consists, hence, in analyzing the acquired data from the CPPS through anomaly detection mechanisms. This provides information related to the security status of the system, with the intention to timely identifying malicious activities within the system. During the analysis phase, security metrics are observed based on the alerts triggered by the employed anomaly detection methods. If the security metric indicates a non-secure CPPS, a change request is generated and forwarded to the planning phase, as an input for selecting the most appropriate self-adaption policy.

Several approaches can be adopted to analyze the data acquired in the monitoring phase; applicable solutions include rule-based Network and Host Intrusion Detection Systems (NIDS), anomaly detection mechanisms leveraging machine learning and/or clustering algorithms, event correlation tools and the like [16].

C. Security Metrics and Self-adaptation Policies

In order to accurately identify and examine security metrics derived from the analysis phase, it is important to distinguish between a security metric and a security event. A *security event* is an event that triggers an alarm during the detection to indicate an abnormal data instance; a *security metric*, instead, considers the occurrence of security events to evaluate the security state of a system. As illustrated in Figure 2, the data collected during the monitoring phase can be analyzed by anomaly detection mechanisms, whose alarms are then interpreted considering predefined security metrics. The security metrics are therefore evaluated and, if necessary, specific actions are triggered depending on the existing self-adaptation policies.

[1]http://www.tcpdump.org/
[2]https://www.wireshark.org/
[3]https://tools.ietf.org/html/rfc5424
[4]https://www.alienvault.com/products/ossim

These response actions are performed through the actuators, deployed in the target system, during the execution phase.

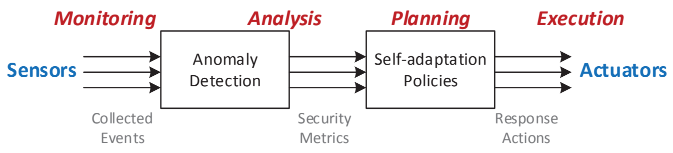


Fig. 2. Anomaly Detection to support monitoring and analysis in the MAPE-K adaptation model.

Examples of security metrics that can be examined in a CPPS to indicate abnormal behavior may include, the amount of alarms triggered by intrusion detection systems within a time interval, the amount of failed attempts to access a PLC, an unusual event sequence in the access process to a SCADA system, the presence of traffic generated from or destined to unknown MAC addresses in the SCADA network, exceptionally large packets transmitted between a PLC and a SCADA system, and many more.

In the planning phase a mitigation strategy is selected among self-adaptation policies (SAPs) to enable a required action in the target system. Based on the input coming from the analysis phase a change plan is generated, consisting of a set of necessary changes (e.g., in the system configuration), and delivered to the execution phase. The identified security metrics are the required inputs for enabling self-adaptation policies.

Let us consider a security metric reflecting the validity of the events sequence during the access to a PLC in the production network. If the observations indicate that the control commands issued to a PLC, which are normally sent from the SCADA MTU, are now being sent from an unknown device in the network, and the anomaly detection tools detect events that prove this process irregularity, a security alarm will be triggered. This will indicate an abnormality in the security metric potentially caused by a connection hijacking attempt. A possible response action in this case could be: *if the anomaly detection reveals a connection hijacking attempt, targeting a PLC, block every PLC connection attempt coming from unknown MAC addresses*.

It is important to notice that static mechanisms, such as invariable access control lists, permitting only a limited set of hosts with specific IP address to communicate with field devices, are not effective when considering highly flexible and volatile environments like CPPS. Therefore agile methods (e.g., based on the MAPE-K model) are required.

As a second example, let us consider a security metric that reflects the number of triggered errors during the PLC login procedure. The generated alerts from the detection tool show 50 failed login attempt per minute. This is considered a violation of the aforementioned security metric, and indicates a possible unauthorized attempt to access the PLC. A possible response action could be: *if the security metric reveals more than 10 failed PLC login attempts per minute, reset the PLC, block the connections coming from the identified attempting MAC address, and request a password reset.*

D. Executing Response Actions

Finally, the execution phase is responsible to carry out the mitigation actions defined in the planning phase, through the adoption of effectors or actuators deployed on the target system. Once the autonomic manager has selected a self-adaptation policy corresponding to a change request form a security metric, tailored mitigation actions will be put in place to modify the security state of the CPPS and counter the identified threat. Response actions may include restarting system components through a control command, initiating the procedure of a password update, adding rules to the firewall to block suspicious connections, etc.

IV. Proof of Concept

In order to evaluate the approach described in the previous section, we setup a test environment to reproduce a simplified manufacturing process. Each step necessary to prove the effectiveness of the proposed concept is outlined in this section.

A. Testbed

The testing environment replicates some of the properties, requirements and processes in place in a manufacturing plant. In particular, it reflects a simplified version of a CPPS, deployed in a semiconductor manufacturing plant, to manage a liquid tank used for cooling down production machinery. As depicted in Figure 3, the testbed consists of a PLC (Siemens S7 1200), an HMI (Siemens Simatic), and a laptop PC hosting a web server to control and configure the PLC (Siemens Totally Integrated Automation (TIA) portal); these components are connected to one another through a gigabit network switch (Netgear ProSAFE Plus GS108E).

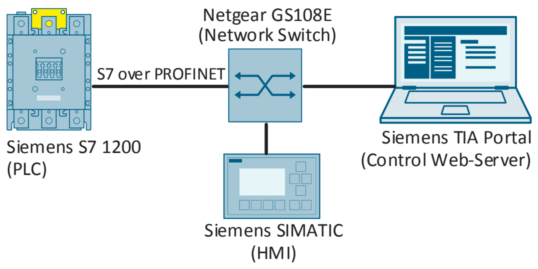

Netgear GS108E (Network Switch)
S7 over PROFINET
Siemens S7 1200 (PLC)
Siemens TIA Portal (Control Web-Server)
Siemens SIMATIC (HMI)

Fig. 3. Testbed architecture diagram.

The components deployed in the testbed communicate using the *S7* communication protocol. S7 is a proprietary protocol developed by Siemens to support secure data transmission over PROFINET[5], and to prevent attacks such as *Man in the Middle* (MitM) and replay. The connections to the web-server are secured using TLSv1.2, enabled by default.

[5]http://us.profinet.com/technology/profinet/

B. Simulated Industrial Process

The testbed simulates a simplified process that could be deployed in a semiconductor manufacturing environment to manage the level of cooling liquid in a tank used to control the temperature of a manufacturing machine. The PLC is programmed to open or close three valves (*fill*, *drain* or *cool*) on the tank, depending on specific configurable conditions.

Figure 4 shows the inputs of the PLC responsible to control the aforementioned industrial process based on the programmed logic. The PLC's ladder logic is configured so that the PLC's outputs (attached to the three tank valves) can assume a binary value (*open/close*), depending on: i) the level of the liquid in the tank, ii) on the buttons pressed by the operator on the HMI, and iii) on a temperature sensor connected to the PLC.

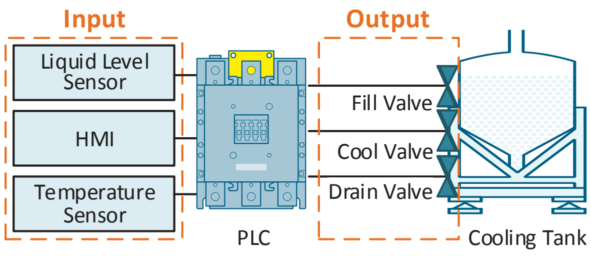


Fig. 4. PLC input/output.

Every second, the manufacturing machine communicates its state to the PLC through a temperature sensor. If the temperature is higher than $90°C$, the PLC opens the cool valve of the liquid tank and the liquid starts flowing out of the tank to cool down the manufacturing machine. The cooling process takes 50 seconds and requires an amount of $25dm^3$ of liquid to decrease the temperature of the manufacturing machine to the desired value of $70°C$. According to the chip manufacturing process, it takes 140 seconds for the temperature of the manufacturing machine to go again above the threshold ($90°C$), this means that the cool valve is opened every 2 minutes and 20 seconds. Figure 5 illustrates this process and shows the variation of the liquid level in the tank over time.

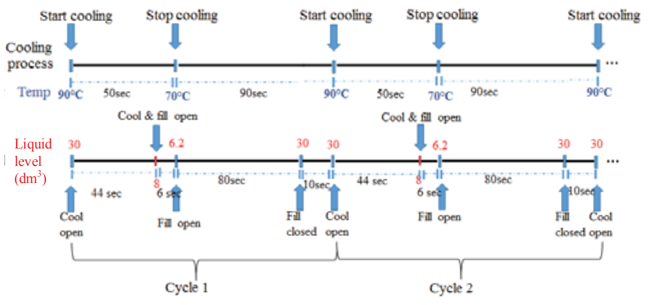


Fig. 5. Simulated industrial process: manufacturing machine cooling.

Before the cool valve is enabled, the tank is filled until the liquid level reaches $30dm^3$. The time interval between the end of the filling process and the beginning of the following cooling cycle is 10 seconds. As soon as the cool valve is activated, the liquid level decreases with a rate of $-0.5dm^3$ per second. After 44 seconds the fill valve is opened because the level of the liquid reaches the lower threshold ($8dm^3$). The cooling process finishes 6 seconds after the fill valve is enabled; at this point the level is $6.2dm^3$. Since the level of the liquid is below $8dm^3$, the PLC will let the filling process continue until the liquid level reaches $30dm^3$. The fill valve allows the liquid to enter the tank at rate of $+0.3dm^3$ per second, so it takes around 80 seconds to reach $30dm^3$. The variation of the liquid level in the tank follows the same trend as long as no change is applied to the PLC logic and no action is performed by the operator through the HMI. If an operator activates the fill valve via the HMI, the cooling liquid will flow in the tank until the level reaches a value of $46dm^3$, then the PLC will automatically open the drain valve and let the water flow out the tank. In case the drain command is not executed (e.g., due to a failure in the PLC system, or because the HMI control commands are not delivered to the PLC) the fill valve will remain open as long as the liquid does not exceed the safety limit of $47dm^3$, hence preventing overflows.

In our testbed, no real liquid tank is deployed. The change of liquid level (N_{lev}) is, in fact, not measured by a sensor, but is calculated following the expression: $N_{lev} = p_l + t_s * r$, where p_l is the liquid level at the previous stage, r is the rate at which the liquid is flowing in or out the tank, and t_s is the period of time that a valve remains open.

C. Monitoring and Analysis

In order to observe the system behavior, several data sources can be monitored in our testbed, including the PLC logs, the PLC's diagnostic buffer, the HMI logs, and the network traffic crossing the switch.

For the sake of simplicity, in our validation we focused our attention on log messages produced by the PLC and by its diagnostic buffer. The log messages generated by the PLC, report the status of the PLC's inputs and outputs, as well as the liquid level measured in the tank. By analyzing the PLC's log messages it is possible to observe if the logic of the PLC is altered i.e., there is a mismatch between input, expected output, and liquid level, indicating a potential PLC corruption. The PLC's diagnostic buffer, instead, records the latest 50 system events, including transitions of the CPU operating mode, errors detected by the PLC's CPU, as well as connections established between the control server and remote clients.

Logs produced by the two selected data sources, and collected through continuous monitoring, are consequently examined by an anomaly detection system during the analysis phase. Considering the implemented test setup and the monitored data, we selected ÆCID[6] as the most suitable anomaly detection tool.

ÆCID follows a lightweight event-based white-listing approach, which processes log data in order to reveal any

[6]https://aecid.ait.ac.at/

deviations from the self-learned system behavior (see [18] for further details). ÆCID consists of two components: *ÆCID Central* and the anomaly miner (*AMiner*). An AMiner instance can be installed on distributed nodes, or deployed on a central node and collects logs from distributed monitored systems. The AMiner parses log lines and checks white-listing rules to identify if the normal system behavior is violated. In cases of an anomaly the AMiner triggers an alarm, notifies the administrator (via e-mail or SIEM alert), and reports the alarm to ÆCID Central. ÆCID Central manages all the AMiner instances deployed in the monitored network. It continuously learns and adapts the internal system model by collecting and analyzing unparsed log lines from the AMiner instances; it generates, updates and distributes sets of rules to be checked by the AMiner instances; moreover, it correlates events reported by different AMiner instances to detect suspicious activities spanning multiple components in the network.

In the test setup we deployed ÆCID in its most light-weight configuration: only one AMiner instance was indeed installed, stand-alone, on the control server machine. The log messages, produced by the PLC and its diagnostic buffer, were therefore collected and analyzed off-line by the AMiner instance running on this machine. In order to keep the resource requirements as low as possible, thus accurately reflecting a real CPPS scenario, no ÆCID Central was installed in our tests. In this operational mode the AMiner needs to be opportunely configured by the system administrator to: i) parse the different input log messages, and ii) distinguish abnormal log lines which do not match the rules reflecting the normal system behavior.

Using the set of data model elements available in ÆCID, the administrator can define multiple parser models to instruct the AMiner on how to correctly interpret the different log messages. Moreover, since the behavior of the PLC and the cooling process is known a priori, a precise set of rules can be defined by the administrator to white-list every expected event recorded by the logs. For example, every log line generated by the PLC reporting: *liquid level = 6* (between $0dm^3$ and $8dm^3$), *inputs = 0* (i.e., all valves closed), and *outputs = 1* (i.e., open fill valve), is to be considered normal, and therefore white-listed.

D. Detecting Security Threats

In this section we demonstrate how a cyber threat, targeting the simplified CPPS implemented in our testbed, can be revealed by ÆCID, and contained by employing the self-adaptation approach described in Section III.

Let us consider the case in which the CPPS is targeted by a multi-stage threat. The main purpose of the intruders is to cause damage to the production facility. In particular, they aim to compromise the cooling process of the manufacturing machine and, at the same time, to flood the surrounding area with the cooling liquid overflowing from the tank. To perform this attack, the intruders intend to attack the PLC's ladder logic modifying its control sequence. To achieve this they put in place an advanced persistent threat [19], consisting of four stages. After acquiring relevant information using social engineering methods, in the reconnaissance phase (stage I), the attackers carry out a spear phishing campaign to gain access to the enterprise network. In the initial compromise, they infect the victim employee's workstation with *DarkComet*[7] (a sophisticated remote administration tool) and enable a back door to allow remote access (stage II). Consequently the attackers manage to infect other hosts in the local network, performing the so called lateral movements (stage III), and obtain administrative privilege on an engineering workstation deployed in the SCADA network. To intrude the production network, the attackers exploit a vulnerability of a network switch and establish a communication with the field devices, including PLCs (stage IV). The attackers are now able to modify the PLC configuration through the TIA portal and customize its logic. In particular they apply specific changes to the ladder logic, to disable the cool and the drain valve, change the fill limit values in the PLC memory, and deny the HMI to send any overriding control command to the PLC. Subsequently, they upload the altered configuration settings to the PLC.

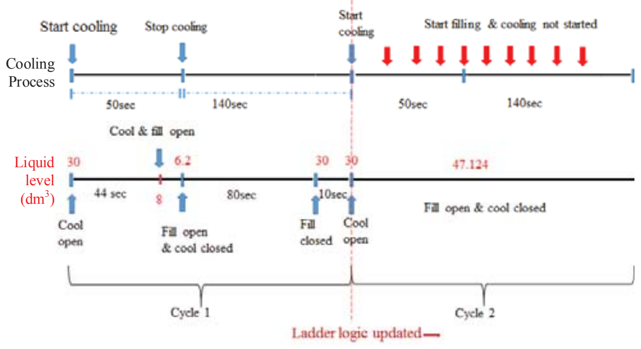


Fig. 6. Cooling process altered after malicious PLC logic update.

Figure 6 illustrates how the cooling process changes after the PLC starts operating following the new logic (compare this diagram with Figure 5). Supposing that the new logic is uploaded before a new cooling cycle starts (as illustrated with the red dashed line), although the temperature sensor will keep sending high temperatures measurements to the PLC, the cool valve will not be open, and the manufacturing machine will not be cooled down. Assuming that no further safety precautions are enabled, this will continue until the manufacturing machine will overheat and stop operating. Meanwhile, the liquid level in the tank will increase because the fill valve will remain open, making the liquid overflow the tank, and subsequently flood the area around the tank.

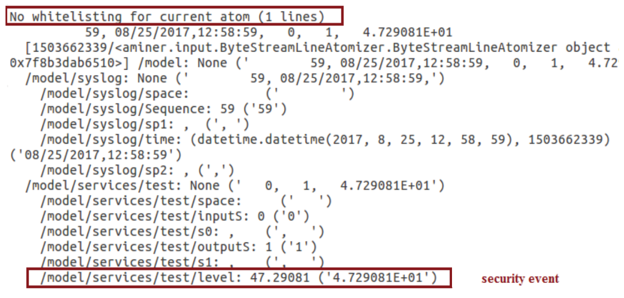
To detect these irregularities in the cooling process, we installed an AMiner instance on the control web server. We collected the PLC logs and the diagnostic buffer messages, captured during the same time interval the PLC was compromised, and we let the AMiner analyze them. Differently from the anomaly-free data, every log line not respecting the normal

[7]http://www.darkcomet-rat.com/

TABLE I
DETECTED SECURITY EVENTS AND CORRESPONDING SECURITY METRICS

Security Event	Security Metric
SE01: Liquid level out of range	SM01: Amount of security events indicating a liquid level higher than $46dm^3$
SE02: Cool valve erroneously disabled	SM02: Amount of security events indicating cool valve disabled when should be enabled
SE03: Unauthorized access to control server	SM03: Presence of unauthorized IP address accessing the control server
SE04: Ineffective HMI control command	SM04: Amount of security events indicating failed HMI command

process, corresponding to the previously defined white-listing rules, triggered an alert. For example, a log line reporting a liquid level over the maximum allowed limit, for a given combination of input and output values, would be classified as anomalous, and trigger an alert, as shown in the AMiner output report depicted in Figure 7.



```
No whitelisting for current atom (1 lines)
          59, 08/25/2017,12:58:59,   0,   1,   4.729081E+01
  [1503662339/<aminer.input.ByteStreamLineAtomizer.ByteStreamLineAtomizer object
0x7f8b3dab6510>] /model: None ('        59, 08/25/2017,12:58:59,   0,   1,   4.72
  /model/syslog: None ('        59, 08/25/2017,12:58:59,')
    /model/syslog/space:          ('        ')
    /model/syslog/Sequence: 59 ('59')
    /model/syslog/sp1: ,  (', ')
    /model/syslog/time: (datetime.datetime(2017, 8, 25, 12, 58, 59), 1503662339)
('08/25/2017,12:58:59')
    /model/syslog/sp2: , (',')
  /model/services/test: None ('   0,   1,   4.729081E+01')
    /model/services/test/space:     ('   ')
    /model/services/test/inputS: 0 ('0')
    /model/services/test/s0: ,    (',   ')
    /model/services/test/outputS: 1 ('1')
    /model/services/test/s1: ,    (',   ')
    /model/services/test/level: 47.29081 ('4.729081E+01')
```

Fig. 7. Anomalous event detected by the AMiner.

In order to avoid unauthorized access to the control server, and therefore prevent the modification of the PLC's ladder logic, only IP addresses within a permitted range are allowed to get access to the server. One of the rules added to the AMiner configuration ensures that the IP addresses of the clients connecting to the server, belong to the allowed range. The attack performed in our test violated this rule, since the intruders gained access to the TIA Portal from an unusual host machine, whose IP address does not belong to such range. The AMiner was indeed able to detect these anomalies by analyzing the messages of the diagnostic buffer; it revealed that the IP address of one of the clients accessing the web-server during the attack, was not part of the authorized IP addresses range (i.e., 192.168.1.0 - 192.168.1.20), and hence triggered an alert.

Considering the detection performed by monitoring and analyzing the PLC logs and the diagnostic buffer logs, the following 4 different security events were generated:

- *SE01 - Liquid level out of range:* the level of the liquid in the tank was above the allowed maximum. According to the ladder logic, it is impossible to have a liquid level higher than $46dm^3$, because the drain valve would be automatically opened. Any event indicating a violation of the PLC logic is considered as an alarm.
- *SE02 - Cool valve erroneously disabled:* the cool valve was disabled when, according to the ladder logic, it should have been enabled.
- *SE03 Unauthorized access to control server:* access to the server performed from an unauthorized IP address.
- *SE04 Ineffective HMI control command:* commands sent by the operator from the HMI could not override the PLC logic.

These events are used as input to the subsequent planning and execution phases foreseen by the self-adaptation cycle, and described in the following section.

E. From Security Metrics to Mitigation Actions

A security metric (SM) describes, in a measurable way, the security status of the CPPS, evaluates if the requirements for self-adaptation are fulfilled, and triggers a change request to the planning phase if adaptation is necessary. For each security event (SE) triggered by the AMiner we defined and observed a corresponding security metric. Table I highlights the correspondence between each security event observed by the AMiner (in the analysis phase), and its respective security metric.

The metric *SM01* observes the amount of sequential alarms indicating that the volume of liquid in the tank overcomes the maximum allowed level. *SM02* refers to the amount of consecutive alarms triggered because of an interruption of the cooling process, i.e., because the cooling valve is closed although it should be open. *SM03* indicates if the control server is accessed from a host using an authorized IP address. *SM04* counts the number of sequential security events indicating that commands sent by the operator, via the HMI, are not being executed, preventing the override of the PLC logic.

It is a crucial property of self-adaptive control mechanism, to ensure that systems keep a certain security state. If the value assumed by any of the metrics listed above implies that the security of the system is compromised, a change request is to be triggered in the planning phase. Actions to overcome anomalies in the process, are selected by the autonomic manager, according to specific predefined self-protection policies, and forwarded to actuators. These actions can be simple commands or complex scripts. The actuators deployed in the CPPS call specific functions that modify system configuration and appropriately adjust settings to mitigate the effects of the detected anomaly, and restore the secure operation of the system.

Table II lists examples of significant deviations in the system behavior, which would correspond to abnormal values of the security metrics, and therefore require the adoption of self-adaptation policies (SAP). Considering the self-adaptation policies listed in the table, a number of possible mitigation actions can hence be derived to contain the detected anomalous behavior:

TABLE II
SECURITY METRICS AND CORRESPONDING SELF-ADAPTATION POLICIES

Security Metric	**Self-adaptation Policy**
SM01: Amount of security events indicating a liquid level higher than $46dm^3$	SAP01: If SM01 is higher than 2 events per hour, send a control command to disable the fill valve and activate the drain valve
SM02: Amount of security events indicating cool valve disabled when should be enabled	SAP02: If SM2 is higher than 4 events per minute, reset the PLC and switch to backup cooling tank to cool the machine
SM03: Presence of unauthorized IP address accessing the control server	SAP03: If a possible unauthorized connection is observed, prevent the identified IP address from accessing control server by adding a denying firewall rule
SM04: Amount of security events indicating failed HMI command	SAP04: If SM04 is higher than 2 events per minute, reset the PLC

- in case the policy SAP01 is invoked, a series of control commands can be sent to the PLC from the control server to: i) enable the drain valve, and ii) disable the fill valve;
- in case SAP02 is invoked, a reset command can be issued from the control server to the PLC, and a backup cooling system, controlled by a secondary PLC, can be enabled to cool down the manufacturing machine;
- in case SAP03 is invoked, a rule can be added to the firewall, to blacklist the discovered unauthorized IP address;
- in case SAP04 is invoked, a reset command can be issued from the control server and executed on the PLC.

Some of these mitigation action would need to be executed manually by system administrators, others will be automatically performed by dedicated software tools.

V. CONCLUSION

Cyber-physical Production Systems (CPPS) are one of the technical driving forces behind the transformation of industrial production towards the digital factory of the future in the context of Industry 4.0. Security is a major concern for such systems as they become more intelligent, interconnected, and coupled with physical devices.

To address the most critical security challenges, we outlined in this paper how CPPS can benefit from the adoption of anomaly detection techniques to facilitate self-protection. We illustrated the main security threats CPPS need to be able to detect and react to, we recalled the phases comprising the self-adaptation process, and we introduced the concept of anomaly detection as enabler of the monitoring and analysis phases in the MAPE-k control loop. Finally, we demonstrated, through an illustrative example implemented in a laboratory testbed, how anomaly detection methods (e.g., ÆCID) can allow a CPPS to timely reveal and react to a complex cyber threat.

The application of the approach illustrated in this paper is currently being validated in the context of the European research project SemI40 (Power Semiconductor and Electronics Manufacturing 4.0). In this project, we intend to further develop, evaluate, and demonstrate the effectiveness of our solution into an operational industrial environment for semiconductor manufacturing.

ACKNOWLEDGMENT

This work was funded by the European project SemI40 (ECSEL 692466) and the FFG project BAESE (852301).

REFERENCES

[1] N. Jazdi, “Cyber physical systems in the context of industry 4.0,” in *2014 IEEE International Conference on Automation, Quality and Testing, Robotics*, 2014, pp. 1–4.

[2] T. Bartman and K. Carson, “Securing communications for scada and critical industrial systems,” in *Protective Relay Engineers (CPRE), 2016 69th Annual Conference for*. IEEE, 2016, pp. 1–10.

[3] i–SCOOP. Industry 4.0: the fourth industrial revolution guide to Industry 4.0. [Online; accessed 10 April 2017].

[4] Industrial Internet Consortium and others, “Industrial Internet of Things Volume G4: Security Framework.” pp. 55–80, 2016.

[5] K. Stouffer, J. Falco, and K. Scarfone, “Guide to industrial control systems (ics) security,” *NIST special publication*, vol. 800, no. 82, 2011.

[6] S. Han, M. Xie, H.-H. Chen, and Y. Ling, “Intrusion detection in cyber-physical systems: Techniques and challenges,” *IEEE Systems Journal*, vol. 8, no. 4, pp. 1052–1062, 2014.

[7] M. Tauber, G. Kirby, and A. Dearle, “Self-adaptation applied to peer-set maintenance in chord via a generic autonomic management framework,” in *Self-Adaptive and Self-Organizing Systems Workshop (SASOW), 2010 Fourth IEEE International Conference on*. IEEE, 2010, pp. 9–16.

[8] M. Hankel and B. Rexroth, “The reference architectural model industrie 4.0 (rami 4.0),” *ZVEI*, 2015.

[9] Z. Ma, A. Hudic, A. Shaaban, and S. Plosz, “Security viewpoint in a reference architecture model for cyber-physical production systems,” in *Security and Privacy Workshops (EuroS&PW), 2017 IEEE European Symposium on*. IEEE, 2017, pp. 153–159.

[10] H. Muccini, M. Sharaf, and D. Weyns, “Self-adaptation for cyber physical systems: a systematic literature review,” in *Proceedings of the 11th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*. ACM, 2016, pp. 75–81.

[11] A. Musil, J. Musil, D. Weyns, T. Bures, H. Muccini, and M. Sharaf, “Patterns for self-adaptation in cyber-physical systems,” in *Multi-Disciplinary Engineering for Cyber-Physical Production Systems*. Springer, 2017, pp. 331–368.

[12] J. O. Kephart and D. M. Chess, “The vision of autonomic computing,” *Computer*, vol. 36, no. 1, pp. 41–50, 2003.

[13] P. Arcaini, E. Riccobene, and P. Scandurra, “Modeling and analyzing mape-k feedback loops for self-adaptation,” in *Proceedings of the 10th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*. IEEE Press, 2015, pp. 13–23.

[14] H.-J. Liao, C.-H. Richard Lin, Y.-C. Lin, and K.-Y. Tung, “Intrusion detection system: A comprehensive review,” *Journal of Network and Computer Applications*, vol. 36, no. 1, pp. 16–24, Jan. 2013.

[15] M. E. Whitman and H. J. Mattord, *Principles of information security*, 4th ed. Stamford, Conn.: Course Technology, Cengage Learning, 2012.

[16] P. Garcia-Teodoro, J. Diaz-Verdejo, G. Maci-Fernandez, and E. Vazquez, “Anomaly-based network intrusion detection: Techniques, systems and challenges,” *Computers & Security*, vol. 28, no. 1-2, pp. 18–28, 2009.

[17] M. Goldstein and S. Uchida, “A Comparative Evaluation of Unsupervised Anomaly Detection Algorithms for Multivariate Data,” *PloS one*, vol. 11, no. 4, p. e0152173, 2016.

[18] M. Wurzenberger, F. Skopik, G. Settanni, and R. Fiedler, “Aecid: A self-learning anomaly detection approach based on light-weight log parser models,” in *Proceedings of the 4th International Conference on Information Systems Security and Privacy - Volume 1: ICISSP,*, INSTICC. SciTePress, 2018, pp. 386–397.

[19] M. Ussath, D. Jaeger, F. Cheng, and C. Meinel, “Advanced persistent threats: Behind the scenes,” in *Information Science and Systems (CISS), 2016 Annual Conference on*. IEEE, 2016, pp. 181–186.