Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 180 (2021) 666–675

Procedia Computer Science

www.elsevier.com/locate/procedia

International Conference on Industry 4.0 and Smart Manufacturing

Smart Factory Security: A Case Study on a Modular Smart Manufacturing System

Federico Maggi[a,*], Marco Balduzzi[a], Rainer Vosseler[a], Martin Rösler[a], Walter Quadrini[b], Giacomo Tavola[b], Marcello Pogliani[b], Davide Quarta[b], Stefano Zanero[b]

[a] *Trend Micro Italy, Viale T. Edison 110, 20099 Sesto S. Giovanni, Italy*
[b] *Politecnico di Milano, Piazza Leonardo Da Vinci 32, 20133 Milan, Italy*

Abstract

Smart manufacturing systems are an attractive target for cyber attacks, because they embed valuable data and critical equipment. Despite the market is driving towards integrated and interconnected factories, current smart manufacturing systems are still designed under the assumption that they will stay isolated from the corporate network and the outside world. This choice may result in an internal architecture with insufficient network and system compartmentalization. As a result, once an attacker has gained access, they have full control of the entire production plant because of the lack of network segmentation.

With the goal of raising cybersecurity awareness, in this paper we describe a practical case study showing attack scenarios that we have validated on a real modular smart manufacturing system, and suggest practical security countermeasures. The testbed smart manufacturing system is part of the Industry 4.0 research laboratory hosted by Politecnico di Milano, and comprises seven assembly stations, each with their programmable logic controllers and human-computer interfaces, as well as an industrial robotic arm that performs pick-and-place tasks.

On this testbed we show two indirect attacks to gain initial access, even under the best-case scenario of a system not directly connected to any public network. We conclude by showing two post-exploitation scenarios that an adversary can use to cause physical impact on the production, or keep persistent access to the plant.

We are unaware of a similar security analysis performed within the premises of a research facility, following a scientific methodology, so we believe that this work can represent a good first step to inspire follow up research on the many verticals that we touch.

Keywords: Cybersecurity; Malware; ICS Security; Smart Manufacturing; Industry 4.0.

* Corresponding author. Tel.: +39 02-925931
E-mail address: federico_maggi@trendmicro.com

1877-0509
10.1016/j.procs.2021.01.289

1. Introduction

Smart manufacturing has become a major technology trend. For example, 10% of the 2021 edition of Hannover Messe—one of the largest industry exhibitions in the world—will be dedicated to digital factories, with more than 600 exhibitors, about 300 of which exclusively target the manufacturing industry. Within the manufacturing industry, smart manufacturing affects sectors including those that deal with: minerals; metals; oil; food; chemicals; textiles and clothing; electric products and electronics; furniture; glass and ceramics; leather, rubber and plastics; paper, cardboard, and related products; pharmaceutical products; tobacco; cars and automotive products, components, and equipment.

The complexity of smart manufacturing systems makes it futile to provide any comprehnsive definition of the concept of smart manufacturing itself. We can say that smart manufacturing systems are the modern implementation of Siemens' totally integrated automation (TIA) concept [14]. From a cybersecurity research standpoint, smart manufacuring represents the frontier of industrial control systems (ICSs). It is quite challenging to obtain access to a sufficiently generic, fully functioning system, deployed within realistic conditions, because the concept of a "reference" smart manufacturing system does not really exist. However, security assessments of these systems are of paramount importance for both the cybersecurity and ICS communities.

We obtained access to a real "smart factory in a box," featuring a fully modular and programmable manufacturing system with 7 stations (each with a programmable logic controllero, or PLC, and a human-machine interface, or HMI), plus an industrial robot and a circular conveyor belt. This system gave us the unique opportunity to perform a thorough security analysis without having to worry about causing downtime, which is instead a very crucial factor in a production floor. Rather than focusing on obvious security flaws—which could be fixed by patching the software—we looked at the system as a whole, striving to think how an attacker would compromise it.

We looked at what an advanced attacker would be able to accomplish, given the technology-specific attack opportunities. Each component of a smart manufacturing system has already been analyzed in previous research: industrial robots, PLCs, HMIs, industrial endpoints or networking appliances, and the like have all been subject to scrutiny. Instead, we focused on how an advanced attacker, with access to one or more of the system's components, would be able find their way into other parts of the system. In other words, this research focuses on the system in its entirety as a set of entry points and targets, under various attacker model assumptions.

By combining all of our findings we demonstrated a multi-step attack scenario, which we validated by testing the feasibility of each attack. This attack scenario, presented in Section 3, allows the readers to (1) see the impact of an unconventional attack flow and (2) understand how attackers think when they look at a complex system.

1.1. Research Drivers

In this work, we had three guiding research drivers:

- **Under which conditions are certain attacks possible, and what is their impact?** Like any security analysis, we wanted to create a map of the entry points for a potential attack, the possibilities for lateral movements (i.e., attacker moving to other machines after initial access), and impact on the physical assets (e.g., manufacturing machines, finished goods).
- **Are there any overlooked attack vectors?** Traditional attacks (e.g., ransomware) can have repercussions on the operational environment, even if that wasn't the primary goal of the attacker. However, we're more interested in attacks leveraging the specific smart-manufacturing technology, which are more likely to pass undetected by current security solutions.
- **What is the security impact of the current software-development practices?** It's easy to develop custom firmware and integrations thanks to the flourishing ecosystem of open libraries, which make it harder to manage the security of the complex chain of interdependencies: A missing integrity

Figure 1. Picture of the modular smart manufacturing system we used with close ups on the right-hand side.

check in a library included in the firmware of an industrial device may lead to full compromise of the entire manufacturing plant.

1.2. Findings and Lessons Learned

The outcome of our analysis can be summarized as follows:

- In the mindset of an advanced attacker, a smart manufacturing system is just one of the many parts of a *larger ecosystem* made up of hardware and other components, such as software, libraries, developers, and business relations, creating an intricate network of interdependencies that can facilitate indirect attacks.
- In the absence of direct attack vectors, malicious actors can still reach a smart manufacturing system *indirectly* by compromising the software supply chain.
- Attack vectors and vulnerabilities must always be contextualized within the entire system: There are protection layers and humans in the loop that can easily prevent a concrete attack from occurring.
- If the manufacturing execution system (i.e., the supervisor) blindly trusts inputs coming from its database or the enterprise resource planning (ERP) system, an attacker could alter the digital twin to cause unintended product outcomes.
- Security-sensitive features in automation task programs—which drives complex industrial machines such as robots—can create ground for vulnerabilities and maliciously written logic that can pass undetected.
- Mobile HMIs can have weaknesses that can facilitate the access to the physical machines they interact with.
- Add-ins for programming environments could be used as attack vectors if proper vetting, sandboxing, and code signing are not in place.

We have released an extensive, non-peer-reviewed white paper [9] that describes the details of the findings summarized in the present paper.

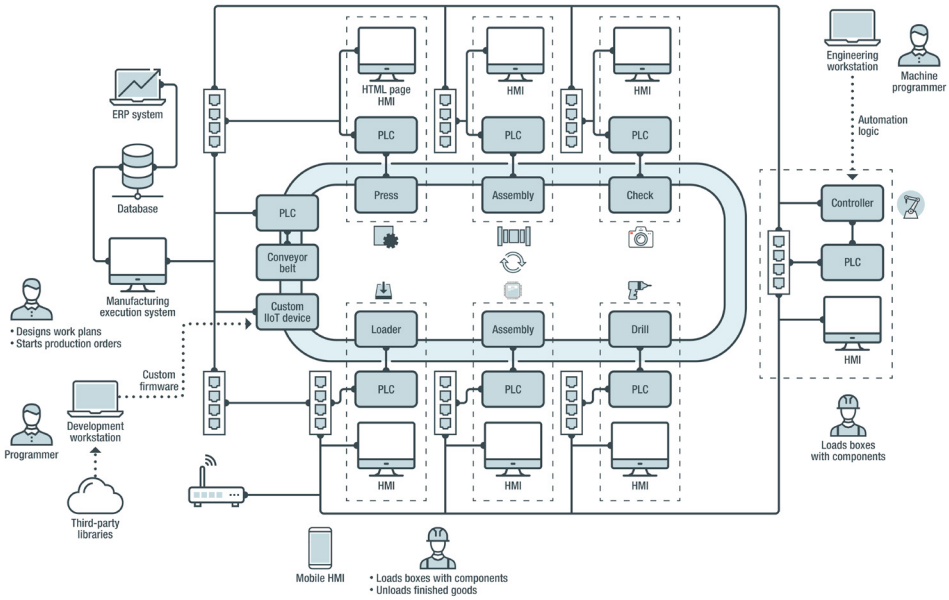


Figure 2. Architecture of the modular smart manufacturing system we used in this research.

2. Smart Manufacturing System Overview

We used a smart manufacturing system hosted in the Industry 4.0 Lab [6] at Politecnico di Milano, installed by an external supplier. The system is programmed to produce (dummy) cell phones composed of a two-parts case plus an internal circuit board. Despite being an educational, lab-oriented deployment, it is based on the same equipment used in real modern factories.

As shown in Figure 1 and 2, the smart manufacturing system comprises seven stations, each with PLCs and HMIs, various physical actuators (e.g., drills, presses), Arduino-based sensors (e.g., temperature, pressure), an inspection camera, a conveyor belt, and an industrial robot. The PLCs are Siemens Simatic DP CPU 1510SP-1 PN units, the HMIs are Siemens Simatic HMI TP800 Comfort units, the industrial robot is a Mitsubishi Melfa V-2AJ, and the networking is managed with Siemens Scalance X208 switches. The SIMATIC Series 7 is considered a de-facto standard for industrial control systems. We complemented the system by including a mobile HMI, to understand their role in the overall attack surface. Mobile HMIs are gaining traction because they allow convenient interaction with the system, up to controlling machines such as robots. However, there are concerns to their adoption, including issues regarding integration, risk management, and cybersecurity. In particular, we included an HMI app by COMAU, which aids the design and execution of pick-and-place tasks.

Despite this system is not the exact replica of what is normally found in a real (smart) factory, all of the parts are widely used in the industry, and design and deployment have been performed by professional system integrators specialized in industrial automation. Given that it is rare if not impossible to have unconstrained access to a production system for research purposes because of the direct and indirect costs (e.g., production losses due to downtimes), we consider our testbed a good approximation of a real system, with the benefit of having complete access to it within a research facility.

Production Flow. The operator loads the supply parts as needed and picks up the finished goods: All the rest is completely automated. The first station loads the first part of the phone case from the magazine onto the conveyor belt. The next stations prepare the case and drill 4 holes, and the industrial robot performs

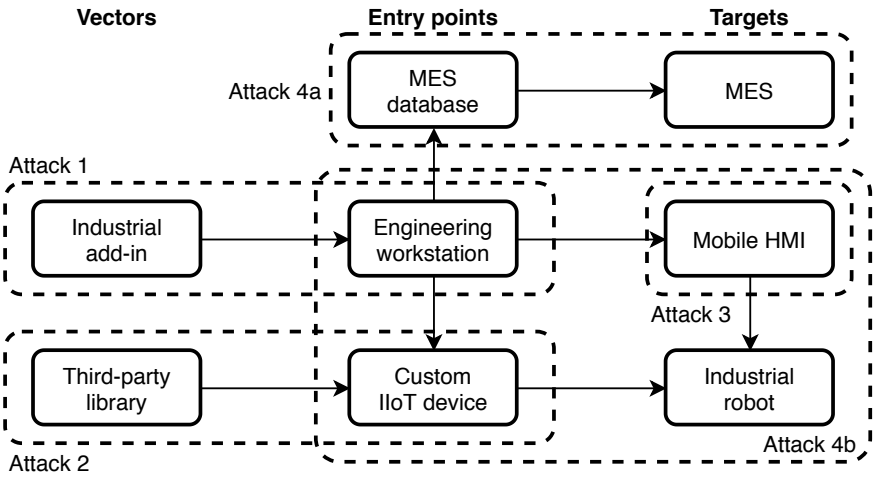


Figure 3. High-level overview of the attack scenario.

pick-and-place operations to position the circuit board inside. Before the second part of the case is positioned and pressed, the camera station performs a visual check on the circuit board.

Coordination and Programming. A manufacturing execution system (MES) coordinates the production by talking to PCLs and HMIs via OPC UA and Modbus, while the PLCs and the HMIs are interfaced via the proprietary Siemens S7. The MES also receives feedback from the PLCs, including tracking information of each workpiece, collected via RFID tags. Although in this particular setting the MES is backed only by a simple database system, in production deployments the MES is typically connected to an ERP for higher-level management tasks. Like in every industrial control system (ICS), also in this manufacturing system there is an engineering workstation, used to develop and deploy program logic, or to connect to field devices for maintenance, diagnostics, or programming, typically via so-called offline programming (OLP) and simulation applications.

The MES allows the operators to write "recipes" that specify the production steps. These can be seen as work templates (i.e., sequences of generic actions), which are translated into network packets to the various stations (i.e., the PLC and HMI) once they receive input parameters (e.g., the number and variant of items to produce). When in execution, the MES receives feedback from the stations, ensures that the sequence of instructions in the work order is followed, and takes corrective actions if needed.

3. Case Study Attack Scenario

Figure 3 shows a high-level overview of the possible attacks that we considered in our case study. Instead of looking at attack vectors such as infected USB sticks [8, 11] or phishing emails—which have been extensively used in targeted ICS attacks (Stuxnet being only the first case)—we consider two attack vectors that are very specific to the manufacturing vertical: A software component for OLP applications, and a third-party library for the custom industrial IoT (IIoT) devices, both present on our testbed system. Regardless of the attack vector or entry point (i.e., OLP application or custom IIoT device), the type of attacker that we consider in our scenarios wants to reach core production machines (e.g., MES, robot), affect the physical environment, steal sensitive data, or mount extortion based attacks.

3.1. Attacker Profile and Capabilities

In this research we considered an advanced attacker with enough resources and capabilities to compromise at least one machine, directly or indirectly connected to the target manufacturing system. We consider

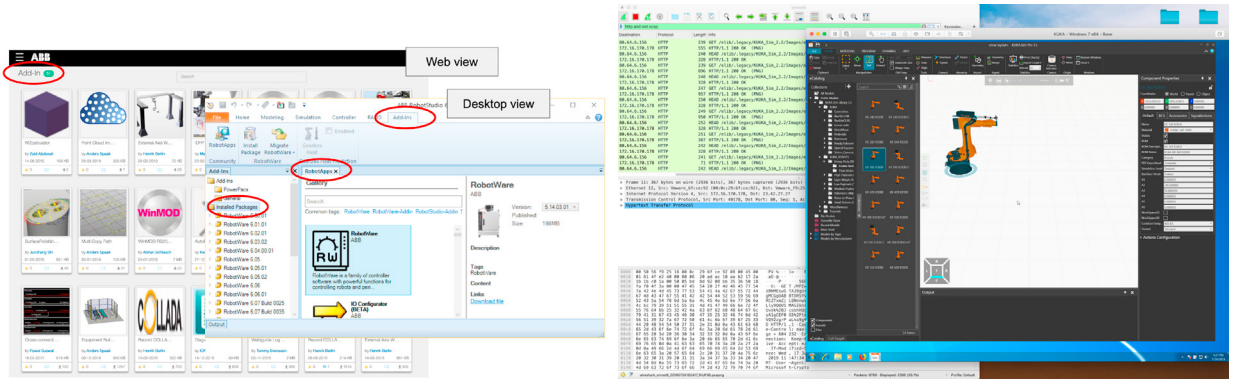


Figure 4. Attack 1: The role of industrial add-ins and digital twins.

attackers who want to cause malfunctions, damage the produced goods, or alter the workflow such that it would manufacture defective products.

Depending on their profiles, attackers may or may not have a foothold in the smart manufacturing system network already. For example, if an attacker is on an adjacent network, they may be able to alter the database of the MES because the MES is often a "bridge" between the enterprise and the factory floor networks. A remote attacker with no access to the factory network may attempt to create a malicious program for the industrial robot and use an insider or other software-based attacks to make it run on the robot. If the attacker knows that the system integrator or the target factory is using a specific piece of development software, they will likely target either the software itself or some third-party extensions for that software. There has been a number of supply chain attacks on software development tools or libraries, especially open-source ones[7, 4, 5, 3, 2]. Interestingly, 42% of attacks on the manufacturing industry reportedly do not directly target facilities, but rather target some of the systems along the supply chain[13, 1].

3.2. Attack 1: Industrial Add-in or Digital Twin Compromise

The development environments that are specific to industrial automation (e.g., OLP software) offer an interesting entry point to reach the engineering workstation, for example to steal secrets from it and propagate further by compromising any software developed on such machine (e.g., task programs for robot automation).

We examined the leading OLP applications and focused on ABB and Kuka (Figure 4), which have the most advanced features. ABB offers an online platform for software extensions and Kuka has a cloud-based catalog for digital twins.

Malicious Industrial Add-in. ABB has an application store[1] (similar in spirit to mobile app stores) from where users can automatically install extensions for (RobotStudio) ABB's OLP software or for their industrial robot controller platform.

Add-ins can be developed and uploaded by anyone: We verified that the sign-up process does not enforce any strict checks. However, uploaded add-ins undergo a vetting procedure and, if successful, they are approved and made available for 1-click installation right into the OLP application. However, we have discovered a vulnerability—promptly fixed by ABB after we reported it—that would allow anyone to bypass the vetting procedure completely and upload an add-in. We verified that a malicious actor could have exploited such vulnerability to upload a malicious add-in that would have given them (indirect) access to the engineering workstation.

[1] https://robotapps.robotstudio.com/

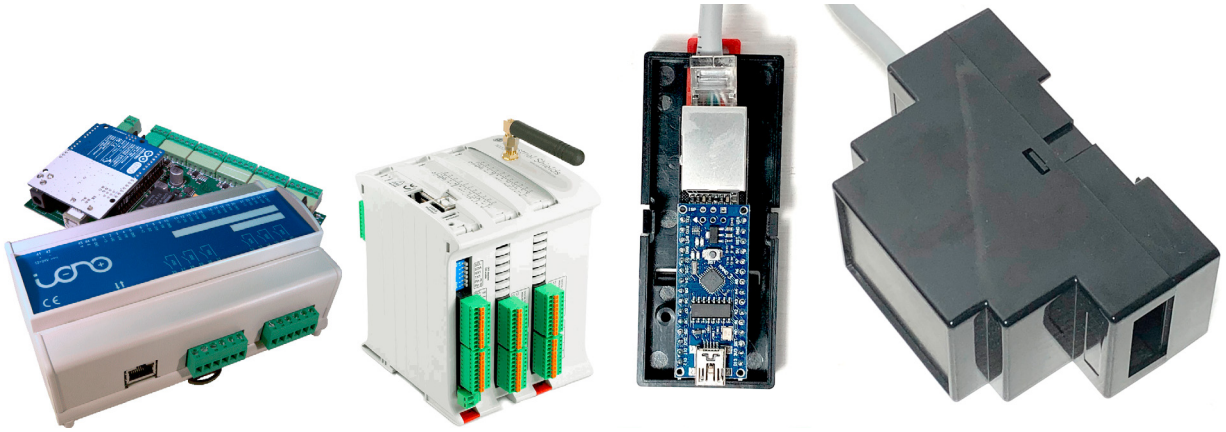


Figure 5. Attack 2: The role of 3rd-party libraries on industrial IoT firmware security.

Digital Twin Compromise. We found and reported a slight variation of the above issue in Kuka.Sim (Kuka's OLP tool). The US ICS Cyber Emergency Response Team (ICS-CERT) has issued an alert[2] based on our findings, so that users can take appropriate countermeasures.

A digital-twin file is a 3D model along with the code that defines the dynamic properties of the modeled object (e.g., weight, velocity). Any alteration to the digital-twin file is reflected in the (modeled) physical object. This means that if anyone changes the 3D model definition or the code, the digital twin will represent *a different* object. We discovered that, before our report to Kuka, their cloud-based digital-twin catalog did not enforce any integrity check of the digital twin files. This means that an attacker between the remote, cloud-based catalog and the engineering workstation that is downloading a digital-twin file, could alter the latter without the OLP software noticing.

Countermeasures. Malicious industrial add-ins are files that contain executable code. Since executable files alone (e.g., DLLs or EXEs) do not necessarily represent a threat, they could not simply be blocked a-priori. The industry best practice in these cases is to employ a behavioral-based detection systems, which can recognize when any software component is performing generically suspicious activities (e.g., suddenly modify many files). Digital twins are more complex, custom objects, which are thus harder to analyze for compromise with generic solutions. The only countermeasure to prevent that a manipulated digital twin is used instead of the original one consists in ensuring end-to-end integrity (e.g., with cryptographic signatures) of the digital twin files, throughout their entire development, deployment, and production process.

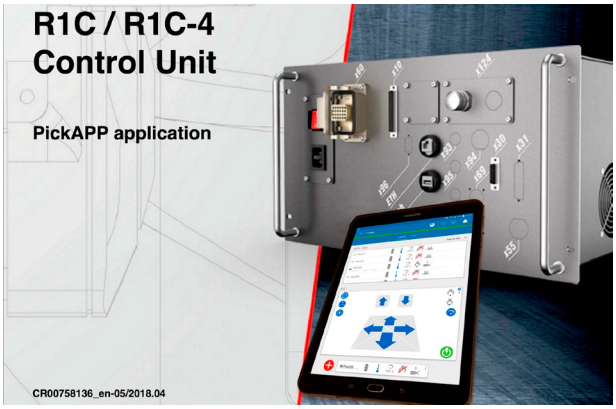
3.3. Attack 2: Compromise via 3rd-party Firmware Libraries

We noticed a number of custom IIoT devices in the system under analysis. Indeed, thanks to the active IoT community, it is nowadays very easy to quickly provision custom sensors and actuators using pre-made components (e.g., software libraries, hardware modules) such as those shown in Figure 5. This is useful to test new features and extend the functionalities of smart factories in general.

Once again, the integrity of the software supply chain (i.e., from the original developer down to the hardware) is key to ensure that the firmware code of the IIoT device is authentic has not been tampered with.

While Arduino provides its own development environment with an online catalog of about 80 official, vetted libraries, there exit over 7,000 (and counting) unofficial libraries available through 3rd-party catalogs (e.g., PlatformIO), with no integrity guarantees no vetting at the source. Not only in the past there have

[2] https://www.us-cert.gov/ics/advisories/icsa-20-098-05




```
public EDPValue login(String paramString1, String paramString2, MessageParame
{
  MessageObject localMessageObject = MessageObject.MessageObjectFactory(paran
  localMessageObject.m_TxStream.edp_encode_int(513);
  localMessageObject.m_TxStream.edp_encode_int(4);
  localMessageObject.m_TxStream.edp_encode_string(new EDPstr(paramString2));
  localMessageObject.m_TxStream.edp_encode_int(513);
  localMessageObject.m_TxStream.edp_encode_int(4);
  localMessageObject.m_TxStream.edp_encode_string(new EDPstr(paramString1));
  localMessageObject.m_TxStream.edp_encode_int(540);
  localMessageObject.m_TxStream.edp_encode_int(531);
  return transmit(localMessageObject);
}
```

Figure 6. Attack 3: The role of mobile human-machine interfaces.

been deliberate attempts to tamper with the open-source supply chain (e.g., [7, 4]), on May 28, 2020, GitHub has discovered a malware purposely written to automate the infection of open source projects [10].

To show the impact of a compromised library we created a custom firmware for a temperature-sensing IIoT device, based on the most popular open-source library for the BME280 ambient sensor. We simulated an attacker that altered the code of the library in transit, and we verified that, during development, we received no alerts that would warn us about the presence of extra code in addition to the original one wrote by the library developer. We altered the original library's source code by adding two malicious routines: one to transmit spoofed network packets (to disrupt network communication) and one to alter the measured temperature. In both cases we successfully verified that the attacks had the desired outcome: None of the PLCs and HMIs was able to contact the MES anymore, and the temperature profile showed unexpected peaks.

Countermeasures. This is still an open research area, because the only way to prevent these cases is to have full visibility over the software supply chain, including all the components developed externally and internally. While this problem is not unique to *industrial* IoT development, the impact of an attack is higher than on consumer IoT devices.

3.4. Attack 3: Vulnerable Mobile HMI Exploitation

Despite HMIs' being a niche market, there are over 170 HMI apps on the Google Play Store, with more than 40 having over 1,000 installations—up to 100,000 in some cases. We expect that future HMIs will rely more on mobile devices, with the security benefits of their modern operating system design, where each HMI app can be sandboxed.

However, if not designed and developed securely, HMI apps can actually facilitate attacks and thus increase the overall risk. This was the case that we examined, and that we reported to the developer: The COMAU PickApp HMI (shown in Figure 6) had a vulnerability that allowed an attacker to obtain the (supposedly secret) credentials to connect to and interact with the industrial robot. Although the robot that we had in the smart manufacturing system under analysis was not compatible with the COMAU PickApp HMI, we were able to reproduce the vulnerability and report it to COMAU, which confirmed it[3] and removed the app from their website for fixing.

Putting this finding in the context of the overall attack scenario, an attacker able to break into a smart manufacturing plant—via either Attack 1 or 2—could now leverage the credentials found in a weak HMI app (such as PickApp in this example) to control the industrial robot.

[3] MITRE assigned the identifiers CVE-2020-10998 and CVE-2020-10999

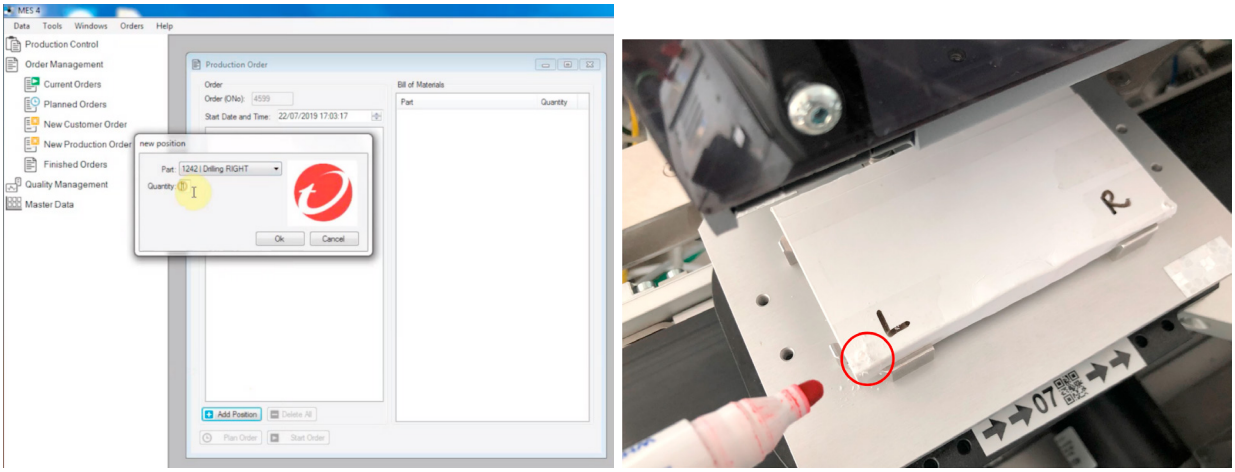


Figure 7. Attack 4: Altering produced products by compromising the MES.

Countermeasures. Detecting vulnerable or malicious mobile apps can prevent the root cause. In this specific case we found that the developers implemented a weak authentication mechanism. We do not think that this case could be detected automatically by streamlined code analyzers. Thus, we think that the best approach is to perform manual analysis of the apps that implement the HMI functionalities.

3.5. Attack 4: Persistency and Production Alteration

At this point the attacker has accessed the manufacturing system and can stay persistent, exfiltrate sensitive data, or affect the production process (e.g., to create the premise for an extortion-based attack).

Vulnerable or Malicious Automation Logic. In our research we confirmed that the attacker can leverage the powerful programming languages used to automate industrial robots. There are two cases. In the first case, if the robot is running a vulnerable automation script (and we have found several instances [12]), the attacker can use it either to move the robot arbitrarily or to load and execute custom code in it (e.g., for persistency). In the second case the attacker can use Attack 1 to alter the automation task program from the OLP software, and hide in it a malicious functionality. While testing this attack we were able to push the robot arm to the limits of the safety system, causing the end effector to detach and fall down into the cage. We had to halt the production for the time required to re-attach the end effector and re-test the robot cycles.

Data Mangling on the MES. Alternatively, the attacker may target the MES, exploiting the implicit trust relation between the MES and its database (or ERP). In particular, we have verified that the MES used in the smart manufacturing system under analysis was trusting any data coming from its database, without performing any validation before taking action. More specifically, we verified that a local attacker could alter the content of the MES database such that the MES would continue working uninterrupted, while altering the workpiece model, and thus the physical object. For example, we were able to change some parameters of the digital twin stored into the MES database, while keeping its metadata unaltered: The operator noticed no difference on the MES user interface, while the drilling station processed the phone case in the incorrect position, as shown in Figure 7.

Countermeasures. The only effective countermeasure to prevent vulnerable or malicious automation logic is manual or automated code review, before and after deployment. Network and host monitoring alone are not effective, first because there are legitimate reasons that a robot, for instance, must receive data from a

machine, and blocking that traffic would result in false positives and consequent downtime; secondly, automation task programs are not common executable files, which available security solutions cannot analyze—and block, if necessary.

The best strategy to prevent attacks against the MES is to prevent the database or ERP system from getting compromised. Indeed, all traffic from the database or the ERP to the MES is normally authorized. So, if an attacker compromises any trusted machine (e.g., ERP or database) which the MES accepts traffic from, it's already too late. An attacker could gain the same level of access without compromising the ERP or the database, yet by spoofing their traffic, effectively impersonating them.

4. Lessons Learned and Conclusions

Despite the trend to integrate and interconnect, smart factory systems are still relatively closed systems. The advantage of this "closed world" security architecture is that there are little chances that conventional mass attacks will hit a smart factory. The main drawback is that a single security flaw may be sufficient for an attacker to successfully gain access to one of the factory's machines: In that case, since the network is designed under the assumption that "what's inside is trusted," the attack can propagate to the rest of the network.

We believe that by showing how non-obvious attack vectors can lead to concrete post-exploitation consequences, we promoted a more compartmentalized security architecture even in those environments where any machine is traditionally considered "trusted."

References

[1] 2467wpczar, 2019. The ominous rise of "island hopping" & counter incident response continues. https://www.carbonblack.com/blog/carbon-blacks-global-incident-response-threat-report-the-ominous-rise-of-island-hopping-counter-incident-response-continues/.

[2] Bertus, 2018. Cryptocurrency clipboard hijacker discovered in pypi repository. https://medium.com/bertusk/cryptocurrency-clipboard-hijacker-discovered-in-pypi-repository-b66b8a534a8.

[3] Cimpanu, C., 2019. Canonical github account hacked, ubuntu source code safe. https://www.zdnet.com/article/canonical-github-account-hacked-ubuntu-source-code-safe/.

[4] Dunn, J.E., 2017. Pypi python repository hit by typosquatting sneak attack. https://nakedsecurity.sophos.com/2017/09/19/pypi-python-repository-hit-by-typosquatting-sneak-attack/.

[5] Franklin, C., 2019. How hackers infiltrate open source projects. https://www.darkreading.com/application-security/how-hackers-infiltrate-open-source-projects-/d/d-id/1335072.

[6] Fumagalli, L., Macchi, M., Pozzetti, A., Taisch, M., Tavola, G., Terzi, S., 2016. New methodology for smart manufacturing research and education: The lab approach.

[7] Garrood, H., 2019. Malicious code in the purescript npm installer. https://harry.garrood.me/blog/malicious-code-in-purescript-npm-installer/.

[8] of Inspector General, O., 2019. Cybersecurity Management and Oversight at the Jet Propulsion Laboratory. Technical Report. NASA.

[9] Maggi, F., Pogliani, M., 2020. Attacks on Smart Manufactururing Systems: A Forward-looking Security Analysis. Technical Report. Trend Micro, Inc.

[10] Muñoz, A., 2020. The octopus scanner malware: Attacking the open source supply chain. https://securitylab.github.com/research/octopus-scanner-malware-open-source-supply-chain.

[11] Nissim, N., Yahalom, R., Elovici, Y., 2017. Usb-based attacks. Computers & Security 70, 675 – 688. doi:https://doi.org/10.1016/j.cose.2017.08.002.

[12] Pogliani, M., Maggi, F., Balduzzi, M., Quarta, D., Zanero, S., . Detecting unsafe code patterns in industrial robot programs, in: Proceedings of the 2020 on Asia Conference on Computer and Communications Security, ACM, New York, NY, USA. p. (to appear).

[13] Seals, T., 2019. Threatlist: Half of all attacks aim at supply chain. https://threatpost.com/half-all-attacks-supply-chain/143391/.

[14] Siemens, 2020. Totally integrated automation – future inside. https://new.siemens.com/global/en/products/automation/topic-areas/tia.html.