



# Rogue Robots: Testing the Limits of an Industrial Robot's Security

Federico Maggi  
Trend Micro Forward-Looking Threat Research

Davide Quarta, Marcello Pogliani, Mario Polino,  
Andrea M. Zanchettin, and Stefano Zanero  
Politecnico di Milano



## TREND MICRO LEGAL DISCLAIMER

The information provided herein is for general information and educational purposes only. It is not intended and should not be construed to constitute legal advice. The information contained herein may not be applicable to all situations and may not reflect the most current situation. Nothing contained herein should be relied on or acted upon without the benefit of legal advice based on the particular facts and circumstances presented and nothing herein should be construed otherwise. Trend Micro reserves the right to modify the contents of this document at any time without prior notice.

Translations of any material into other languages are intended solely as a convenience. Translation accuracy is not guaranteed nor implied. If any questions arise related to the accuracy of a translation, please refer to the original language official version of the document. Any discrepancies or differences created in the translation are not binding and have no legal effect for compliance or enforcement purposes.

Although Trend Micro uses reasonable efforts to include accurate and up-to-date information herein, Trend Micro makes no warranties or representations of any kind as to its accuracy, currency, or completeness. You agree that access to and use of and reliance on this document and the content thereof is at your own risk. Trend Micro disclaims all warranties of any kind, express or implied. Neither Trend Micro nor any party involved in creating, producing, or delivering this document shall be liable for any consequence, loss, or damage, including direct, indirect, special, consequential, loss of business profits, or special damages, whatsoever arising out of access to, use of, or inability to use, or in connection with the use of this document, or any errors or omissions in the content thereof. Use of this information constitutes acceptance for use in an "as is" condition.

# Contents

## 4 State of Industrial Robots

## 11 Industrial Robot Architecture


## 15 Attacking Industrial Robots

## 35 Possible Threat Scenarios for Robot Users

## 39 Toward a Brighter Future

## 44 Conclusion



A background image of an industrial robot arm in a factory setting, with sparks visible at the bottom. The robot arm is blue and white, and the background is dark with some lights.

Vulnerabilities in protocols and software running industrial robots are by now widely known, but to date, there has been no in-depth, hands-on research that demonstrates to what extent robots can actually be compromised. For the first time, with this research—a collaboration between Politecnico di Milano (POLIMI) and the Trend Micro Forward-Looking Threat Research (FTR) Team—we have been able to analyze the impact of system-specific attacks and demonstrate attack scenarios on actual standard industrial robots in a controlled environment.

In industrial devices, the impact of a single, simple software vulnerability can already have serious consequences. Depending on the actual setup and security posture of the targeted smart factory, attackers could trigger attacks that could amount to massive financial damage to the company in question or at worst, even affect critical goods. Almost all industry sectors that are critical for a nation are potentially at risk.

Unfortunately, the Industry 4.0 revolution is just bringing industrial robots closer to the forefront. As improvements in the way industrial robots work and communicate increase their complexity and interconnectedness, the industrial robots sector unlocks a broader attack surface. In our security analysis, we found that the software running on these devices is outdated; based on vulnerable OSs and libraries, sometimes relying on obsolete or cryptographic libraries; and have weak authentication systems with default, unchangeable credentials. Additionally, the Trend Micro FTR Team found tens of thousands of industrial devices residing on public IP addresses,<sup>1</sup> which could include exposed industrial robots, further increasing the risk that an attacker can access and hack them.

The impact of vulnerabilities, for example on robots, is what makes our findings a very loud wake-up call for the industrial control systems (ICS) sector. To quantify such impact, our security analysis revealed that industrial robots must follow three fundamental laws—accurately “read” from the physical world through sensors and “write” (i.e., perform actions) through motors and tools, refuse to execute self-damaging control logic, and most importantly, echo one of the “Laws of Robotics” (devised by Isaac Asimov, a popular science writer) to never harm humans. Then, by combining the set of vulnerabilities that we discovered on a real, standard robot installed in our laboratory, we demonstrated how remote attackers can violate such fundamental laws up to the point where they can alter or introduce minor defects in the manufactured product, physically damage the robot, steal industry secrets, or injure humans. We then considered some threat scenarios on how attackers capitalized on these attacks, as in an act of sabotage or a ransomware-like scheme.

On the one hand, industrial devices are designed according to strict physical security and safety standards in order to work in rough conditions with extreme temperature ranges, vibrations, and electromagnetic noise. On the other, because of the ubiquity and flexibility demanded by the Industry 4.0 trend, industrial devices are being designed to be flexible, easy to deploy, and to not necessarily require any special security or IT skills. These opposing design requirements make producers very prone to introducing software bugs.

Rather than concluding this paper with a classic checklist for ICS vendors, we reflected on reasons why the situation has not changed much over the years. Thus, we provided a series of research and engineering challenges that we believe will make a difference in the journey to secure the Industry 4.0 ecosystem. On this journey toward improving the security posture of robots in the Industry 4.0 setting, we also began reaching out to vendors, among whom ABB Robotics stood out in that it readily welcomed suggestions we had to offer and even started working on a response plan that will affect its current product line without losing time.

More exhaustive technical details of our research, including the vulnerabilities we discovered and subsequently reported to vendors, will be published in the upcoming Institute of Electrical and Electronics Engineers (IEEE) Security and Privacy Conference in San Jose this coming May.<sup>2</sup>



# State of Industrial Robots

Industrial robots (see Figure 1) are mechanical multi-axis “arms” used in modern industries for automating various operations such as welding, packaging, food processing, or die casting. They thus play a key role in Industry 4.0, which is characterized by automation trends and smart factories. The International Federation of Robotics forecasts that by 2018, approximately 1.3 million industrial robot units will be employed in factories globally, and the international market value for robotic systems will reach approximately US\$32 billion.<sup>3</sup>



Figure 1. An industrial robot at work

(Source: RobotWorx, “KUKA Packing Robot at PackExpo 2015”, youtube.com)

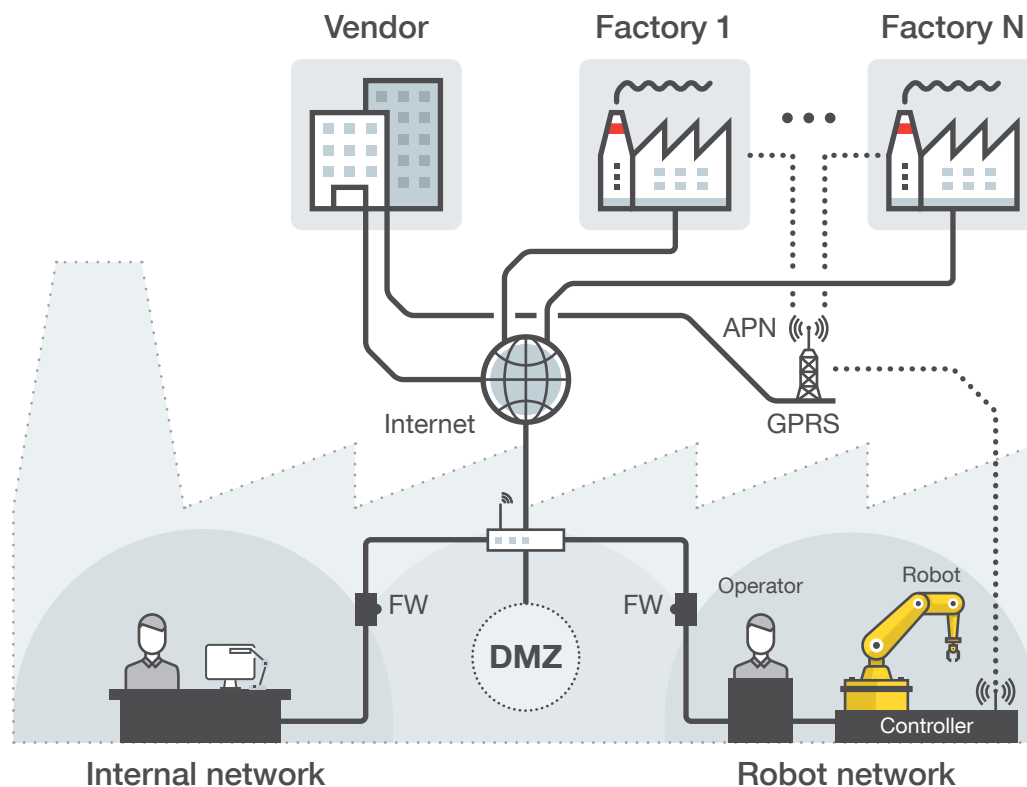


Figure 2. High-level depiction of an Industry 4.0 ecosystem

## Broad Attack Surface

In all of their forms, robots are ultimately complex cyberphysical systems (CPSs) that include multiple mechanical actuators, controllers, sensors, and human interaction devices. In this context, the growing integration of computerized monitoring of physical production processes leads to robots that are interconnected with other robots and external services. In the Industry 4.0 vision, an enterprise management system can track and automatically place an order to suppliers any part needed to complete a scheduled production as well as reconfigure robotized production lines and receive updates on their operational status.<sup>4</sup>

Nowadays, industrial robots are connected to computer networks primarily for programming and maintenance purposes, but we can already see some emphasis on richer and complex programming to integrate robots with the factory IT ecosystem. For example, ABB exposes a so-called Robot Web Service,<sup>5</sup> which allows external software or devices to “speak” with the robot controller by means of HTTP requests. Easy-to-use application programming interfaces (APIs) allow robots to be controlled via smartphones.<sup>6, 7</sup> In fact, robot app stores are becoming increasingly available for both consumer and industrial robots.

Taken together, these translate to major improvements in terms of safety and efficiency, and reduction in production lead time. However, increasing the complexity and interconnection of industrial and robotic systems offers a novel attack opportunity with consequences ranging from the simple compromise of controlling machines to impacting the quality or availability of the production chain. This could be motivated by simple economic competition or in a cyberwarfare scenario, attacking factories that manufacture critical goods.<sup>8</sup>

Taking advantage of new interconnections to compromise devices originally designed to work in isolation is a pattern already observed in other contexts (e.g., automotive<sup>9, 10</sup>; Stuxnet,<sup>11</sup> the German steel mill attack, ICS incidents<sup>12</sup>).

## Vulnerable Software Base

Even superficial analysis revealed that the software that run on these devices is outdated, mainly for back-compatibility reasons; based on vulnerable OSs and libraries (e.g., Linux 2.6, .NET SDK 3.5), sometimes relying on obsolete or otherwise broken cryptographic libraries; and have weak authentication systems with default, unchangeable credentials. Even though some vendors do apply custom patches to outdated software stacks, this does not seem a good enough approach to software security.

Other researches have already found a sheer number of vulnerabilities on these devices in the past, confirming a long-standing and inadequate security posture. After about a year, it seems that the situation has not changed much in the ICS space.

## Software-Defined Safety

Industrial robots are traditionally designed to operate in a cage, physically separated from where humans work. However, vendors are introducing various models of collaborative robots (co-bots) that are able to work in physical proximity to humans (e.g., ABB's YuMi, FANUC's CR-35iA,<sup>13</sup> and various models by Universal Robots; see Figure 3).

This, alongside the gradual shift of safety device implementations from hardwired logic to more flexible software-based implementations, increases the relevance of safety concerns. By exploiting security issues, an attacker may disrupt operations and indirectly pose safety threats to human operators.



Figure 3. Co-bot working closely with a human operator with no physical safety measures

(Source: [enterprisetech.com](http://enterprisetech.com))

## Industrial Robots in Critical Sectors

Nowadays, the vast majority of industry sectors take advantage of robots. Our market search covered 10 major vendors of industrial robots. Among their customers, we found large and mid-sized enterprises operating in the following sectors, most of which are critical for every nation.

- Automotive
- Aerospace
- Alternative energy
- Defense
- Plastics
- Electrical and electronics
- Glassmaking
- Welding
- Food and beverages
- Metal fabrication
- Wood industries
- Paper and printing
- Pharmaceutical
- Packaging and palletizing
- Distribution centers
- Die cast
- Off-road vehicle manufacturing
- Railway
- Foundry and forging

Attackers can take advantage of the presence of industrial robots in critical sectors that distribute public services in order to amplify the impact of their attacks, making these devices an attractive attack vector.

## Industrial Routers Exposed on the Internet

Although direct Internet exposure may seem unrealistic, we did find industrial robots that are reachable from the outside. We ran searches for about two weeks on Shodan, ZoomEye, and Censys—three search services that index data from Internet-wide scans. We were looking for connected robots from the top vendors (see details in Table 1) and found several ones, some of which even provided unrestricted access using anonymous credentials (i.e., the authentication system was disabled). For ethical reasons, we did not directly attempt to connect to those but leveraged the data provided by the search services to filter out false positives.



Figure 4. Industrial routers by Robustel, InHand, and Digi that feature one or more cellular modems, Ethernet and USB ports, and serial connection

An increasing number of industrial robots embed remote access devices already used by the vendor for remote monitoring and maintenance. Such devices are essentially industrial routers, often dubbed “service boxes” by vendors. They allow remote users to connect to a robot as if they were on a local network. The connection between the service box and the remote service center can employ various technologies—over the Internet, through a virtual private network (VPN), or through a General Packet Radio Service (GPRS) network that can use commodity carrier-provided Access Point Names (APNs) for data connectivity and machine-to-machine (M2M) subscriber identity module (SIM) cards configured to use vendor-specific APNs. In the second option, if not properly configured, all factories that use robots from the same vendors will share such APNs and “see” each other through the network.



Industrial routers provide an interesting attack surface to gain access to a robot controller and other industrial machines. For example, attackers could target a widespread vendor of such appliances whose products are also resold by various robotics original equipment manufacturers (OEMs) as part of their support contracts. Our market search revealed about a dozen vendors who specialize in building industrial routers for various applications (see Table 1).

Manufacturer	Number of Search-Pattern-Exposed Robots
ABB Robotics	5
FANUC FTP	9
Kawasaki E Controller	4
Mitsubishi FTP	1
Yaskawa	9
<b>TOTAL</b>	28

Brand	Exposed Devices	No Authentication	Known Vulnerabilities	New Vulnerabilities
Belden	956		4	1
Eurotech	160			
eWON	6,219	1,160	10	
Digi	1,200		2	1
InHand	883			
Moxa	12,222	2,300	30	1
NetModule	886	135		1
Robustel	4,491		1	
Sierra Wireless	50,341	220	4	
Virtual Access	209		1	
Welotec	25			
Westermo	6,081	1,200	7	2
<b>TOTAL</b>	83,673	5,105	59	6

Table 1. Industrial robots exposed via their FTP servers and exposed industrial routers, according to Censys, ZoomEye, and Shodan search results as of late March 2017

(Note: We obtained the number of “known vulnerabilities” by querying [Vulners.com](#) while the number of “new vulnerabilities” is based on our own findings.)

Not all of the aforementioned vendors were as ready to respond and interact with us. However, we want to stress that ABB Robotics has been very open to hear our suggestions for security improvements and already started creating an improvement and response plan that will influence their current product line.

The increased connectivity of computer and robot systems is and will continue to expose robots to cyber attacks. Indeed, industrial robots—originally conceived to be isolated—have evolved and are now exposed to corporate networks and the Internet.

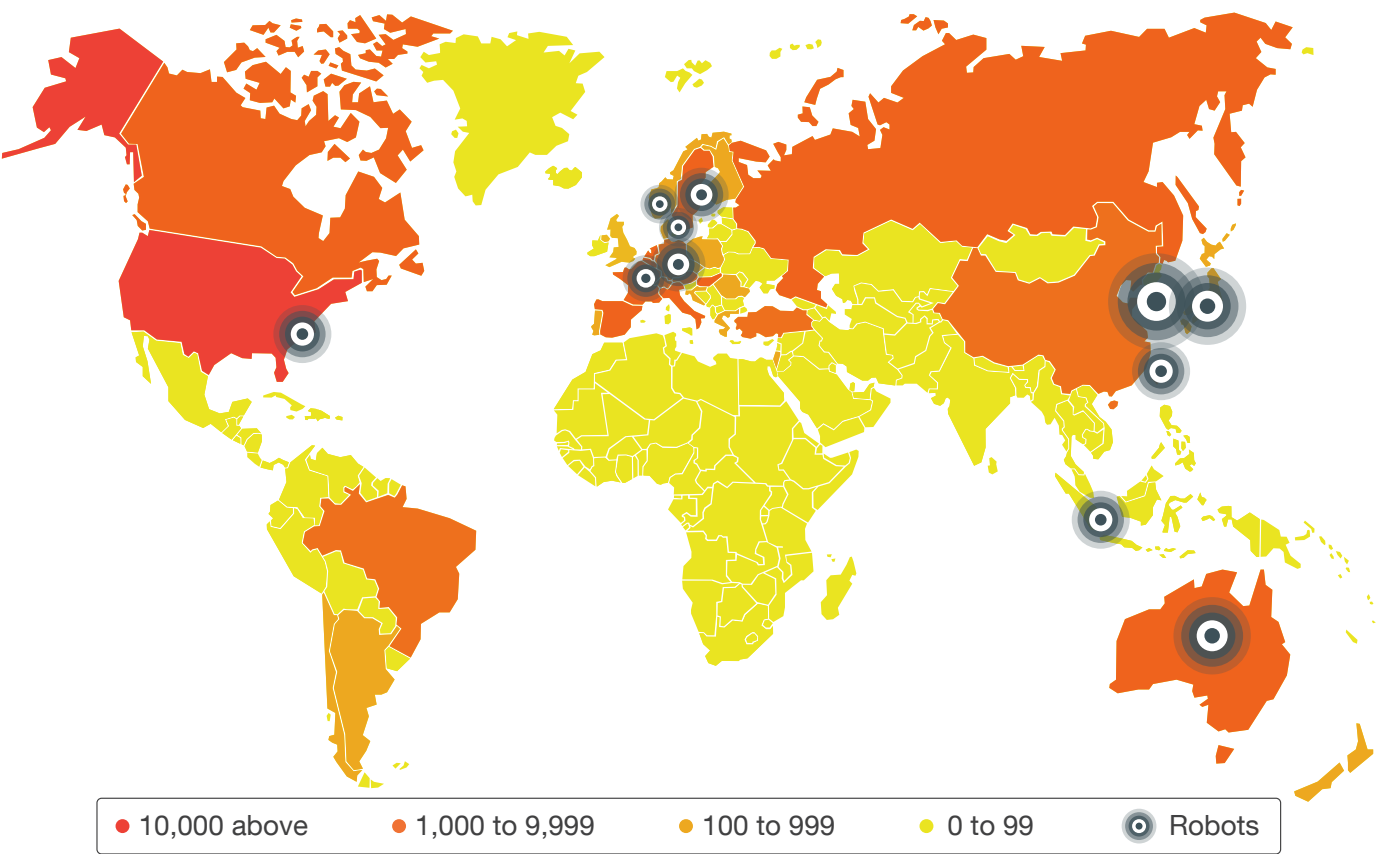


Figure 5. Map showing volume of exposed industrial routers and presence of exposed robots



# Standard Industrial Robot Architecture

It is important to understand the different components of the industrial robot architecture before we present attack scenarios. Since industrial robots are extensively standardized,<sup>14</sup> they are architecturally, functionally, and technically similar across vendors, and share a minimum set of requirements.

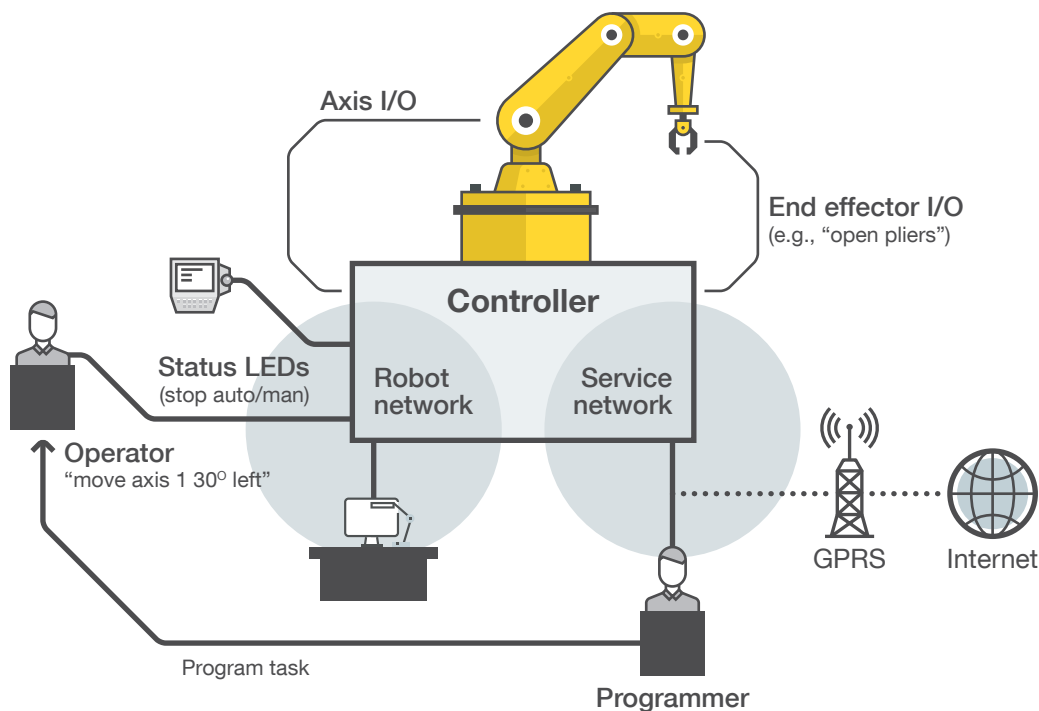


Figure 6. Black box view of a standard industrial robot architecture

In the standard industrial robot architecture, the programmer and/or operator issues high-level commands to the controller through the network (e.g., via a REST API by loading a program from the human-robot interaction [HRI] to move the joystick). The controller translates such commands into low-level inputs for the actuators (e.g., end effectors and servo motors) through dedicated input/output (I/O) interfaces. The controller is also reachable through a remote access interface.

## Robot

An industrial robot is an “automatically controlled, reprogrammable, multipurpose manipulator programmable in three or more axes, which can be either fixed in place or mobile for use in industrial automation applications.”<sup>15</sup> Mechanically, an industrial robot looks like an arm with two or more joints terminated by an end effector (e.g., pliers, a cutter, or a laser beam welder) that interacts with the environment.



Figure 7. Caged robotic arm used in our demonstration

## Controller

The robot controller is a complex device, typically enclosed in one or more chassis. It is composed of multiple interconnected subsystems and computer systems. A robot controller is designed with emphasis on efficiency, complex motion description, nonlinear control, and interaction with human operators.

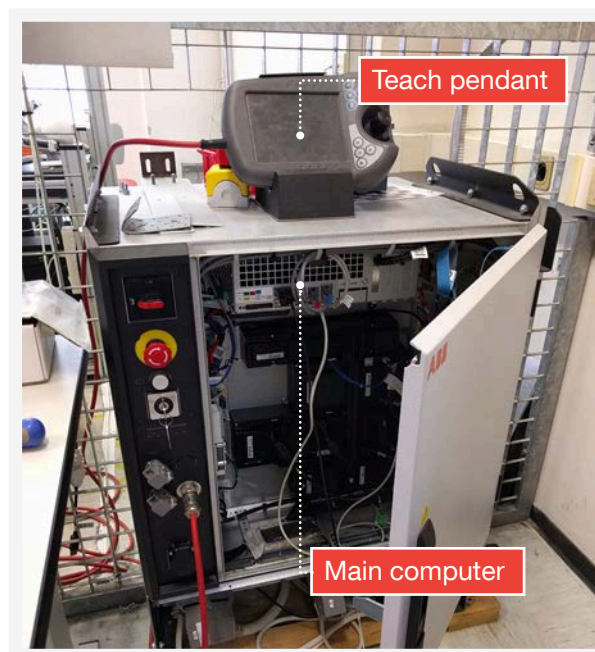


Figure 8. Controller and teach pendant



The controller can work in automatic or manual mode. In automatic mode—intended for regular operations of the robot in production—the controller loads and executes task programs from the teach pendant’s storage or the controller’s memory. In manual mode, the robot performs movements according to inputs issued by the operator through the HRI. This mode allows both a reduced-speed mode, used for programming the robot, and a high-speed mode, used for testing.

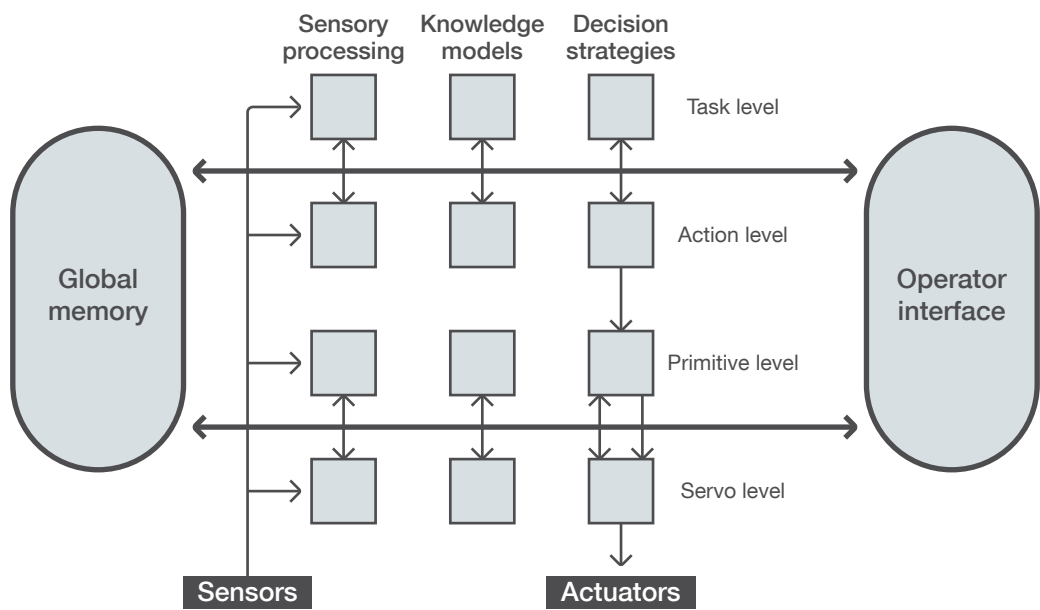


Figure 9. Abstract representation of the control system architecture of a robot controller

The arrows in Figure 9 represent the data flow while the blocks represent “generic” modules that comprise the control system. In the control system above, the controller acquires measurements from the environment, uses this knowledge and its internal state to process inputs, makes a decision, and actuates it. The functionality of the robot can be expressed at various levels of abstraction, from the highest level at the top (task such as what the operator wants the robot to do) to the lowest level at the bottom (servo such as current to apply to the motor’s driver).

# Control System

The controller implements a control system to supervise the activities of the robot, making it one of the most safety- and security-critical components. The control system is split into different modules and hierarchies that allow it to measure the state of the system and translate high-level tasks into specific actions.

# Human-Robot Interface

The human operator is the person in charge of monitoring, starting, and stopping the operations of the robot whereas the programmer is in charge of writing task programs (i.e., the set of instructions that define the specific tasks of the robot). Operators and programmers can monitor the status of the robot and program it through a handheld unit connected to the controller via a wired or wireless connection called the “teach pendant” (Figure 8 shows a real pendant by ABB). The teach pendant resembles a “heavy-duty” tablet augmented with physical buttons and joysticks.

## Robot Programming

The controller accepts task specifications written through a domain-specific programming environment. These programs can be written online (recording action sequences through the joystick) or offline (programming actions either through text editors or more commonly, simulators that can then be saved).

Robot programs are stored in the controller’s global memory. It holds the functional blocks needed to exchange information between levels and modules, and maintain estimations of the state of the whole system (e.g., known position on each axis) and the environment (e.g., temperature or size of the piece held by the pliers).

## Requirements

Through the calculated interaction of the components described above, industrial robots are expected to execute with the following requirements:

- **Accuracy:** The robot should read precise measurements from sensors, and issue correct and accurate commands to actuators so movements are performed within acceptable error margins.
- **Safety:** The robot should expose sufficient and correct information to enable operators to take safe and informed decisions, allow human operators to engage emergency procedures, and execute them quickly and safely.
- **Integrity:** The robot controller should minimize the possibility that a badly written control logic could result in damaging the robot’s physical parts.

The next section shows how a violation of any of these requirements, if initiated through a digital attack, can lead to a robot being controlled by an attacker.



# Attacking Industrial Robots

## Attacker Model

Our security assessment of industrial robots begins with profiling an attacker model, which allows us to think about what an attacker can or cannot do within the boundaries set by the Industry 4.0 scenario schematized earlier.

In this scenario, industrial robots are connected to a dedicated subnet (in the best case) and isolated from other endpoints. However, we found that such subnets are often remotely accessible, either through a connection to the Internet or via a dedicated remote access box (e.g., via GPRS). Robots are physically connected to a network via a controller, which in the first instance, can be approximated as a general-purpose computer that controls their operations, and its subsystems through which operators interact (e.g., joysticks, switches, or I/O and diagnostic ports). We purposely designed this scenario to be vendor agnostic and rather simple while representing a complete deployment.

## Access Level

We broadly distinguish between network and physical attack vectors. We explicitly skirt attacks that involve breaking or tampering with the physical security of the robot controller's case, focusing exclusively on accessible hardware components that allow access to the digital attack surface.





A slightly more sophisticated profile is the “casual” attacker (e.g., malicious contractor or technician) who is able to plug a device into the openly accessible robot controller’s RJ-45 (or equivalent) port. This grants full access to the robot controller’s computer via Ethernet or other proprietary I/O interfaces. The physical attacker can leverage further vectors such as internal I/O interfaces that the controller uses to communicate directly with the robot’s components (e.g., DeviceNet over CANbus, as in ABB’s controllers). Thus, this profile is strictly more powerful than a network attacker.

## Technical Capabilities

We assume attackers are familiar with the structure of the target industrial robot and possess the technical skills to perform reverse engineering without exploiting any insider technical knowledge. This means they rely on publicly available information (e.g., controller software and firmware available for download from vendors’ websites) or information obtained by reverse engineering publicly available software. We learned most of the details described in this paper by reading freely available technical documentation. An attacker can do the same thing.

## Access to Equipment

We consider it trivial for an attacker to obtain a copy of the executable binaries of the robot controller firmware. In fact, some vendors make the controller firmware freely available for download from their websites (e.g., the controller firmware for ABB robots is included in RobotWare, part of the RobotStudio software suite). This allows an attacker to easily access and reverse engineer the software and discover vulnerabilities.

Depending on the attackers’ budget, they may or may not be able to test exploits before carrying out an attack because this would require access to a full-fledged deployment. Even without being able to access a real industrial robot, many vendors distribute simulators of some controller parts (as in the case of ABB’s software suite), which allows attackers to run simulated versions of the software running on the robot’s controller.

Some vendors (e.g., COMAU and Kuka) provide their software only to customers. The attacker needs at least temporary physical access to a robot controller to dump the firmware from the controller itself. In many cases, to ease maintenance and upgrades, the firmware is loaded from a removable memory card such as a MultiMediaCard (MMC), which can be leaked via or stolen from the company that has access to the robot software as part of a support contract. Note that this does not necessarily require an insider profile. Knowledge about obtaining that information and short-term access to the factory’s robot network is sufficient.

## Attacker's Budget

As in other specialized domains (e.g., avionics and ICS), used or reconditioned industrial robot parts are available for sale without restrictions. The cost of such parts can vary greatly and a complete robot with its controller has a relatively high price tag. However, we do not consider such price out of reach for our attacker. For example, a search on online marketplaces (e.g., [globalrobots.com](https://www.globalrobots.com), [ebay.com](https://www.ebay.com), and [alibaba.com](https://www.alibaba.com)) revealed that the IRB 140 manipulator from ABB, which matches the reference setup we used in our experimental analysis, can be purchased for US\$13,000–28,950 together with an outdated S4C controller while other manipulators with an IRC5 controller can be found and bought for US\$24,999–35,500.

Access to specific features (e.g., the GPRS remote service box of the robot that we analyzed) is more complex, as the service box is only available directly from the vendor as part of support contracts.

## Robot-Specific Attacks

We systematically analyzed the feasibility of attacking a modern industrial robot and succeeded in doing so. Due to the architectural commonalities of most modern industrial robots and the existence of standards, the robot chosen for our case study is representative of a large class of industrial robots.

We explored concrete attack vectors that when exploited, allowed us to subvert the interaction between a robot and its physical environment, violating its basic operational requirements. We found that certain combinations of software vulnerabilities enable specific classes of attacks that are unique to industrial robots (e.g., circumventing safety measures or impairing movement precision) and evaluated the potential impact of such attacks. More specifically, we found five classes of attacks based on the observation that a robot working under normal circumstances should be able, at least, to read accurately from sensors, execute its control logic, perform precise movements, and not harm humans. Instead, the effect of our attacks is that none of these properties are assured.

Attack Class and Description	Concrete Effects	Requirements Violated
<b>Attack 1: Altering the Control-Loop Parameters</b> The attacker alters the control system so the robot moves unexpectedly or inaccurately.	Defective or modified products	Safety Integrity Accuracy
<b>Attack 2: Tampering with Calibration Parameters</b> The attacker changes the calibration to make the robot move unexpectedly or inaccurately.	Robot damages	Safety Integrity Accuracy
<b>Attack 3: Tampering with the Production Logic</b> The attacker manipulates the program executed by the robot to stealthily introduce a flaw into the workpiece.	Defective or modified products	Safety Integrity Accuracy
<b>Attack 4: Altering the User-Perceived Robot State</b> The attacker manipulates the status information so the operator is not aware of the true status of the robot.	Operator injuries	Safety
<b>Attack 5: Altering the Robot State</b> The attacker manipulates the true robot status so the operator loses control or can get injured.	Operator injuries	Safety

Table 2. Summary of robot-specific attacks and their effects

We analyzed industrial-robot-specific attack opportunities to violate accuracy, safety, and integrity requirements in this section. Below, we showed how an attacker can leverage digital vulnerabilities to carry out such attacks.

## Attack 1: Altering the Control-Loop Parameters

This attack targets the servo motor level (see Figure 9). It relies on the fact that for flexibility and code reusability purposes, kinematics and configuration parameters are read from a file or otherwise defined at runtime. This means that an attacker that can access a configuration file can modify these parameters.

For the attacker, the most interesting parameters to modify are the ones that affect robot movements—an extreme modification of which can completely violate functional and safety requirements.

A full video demo of this attack type is available at <https://www.trendmicro.com/vinfo/us/security/news/internet-of-things/rogue-robots-testingindustrial-robot-security>.



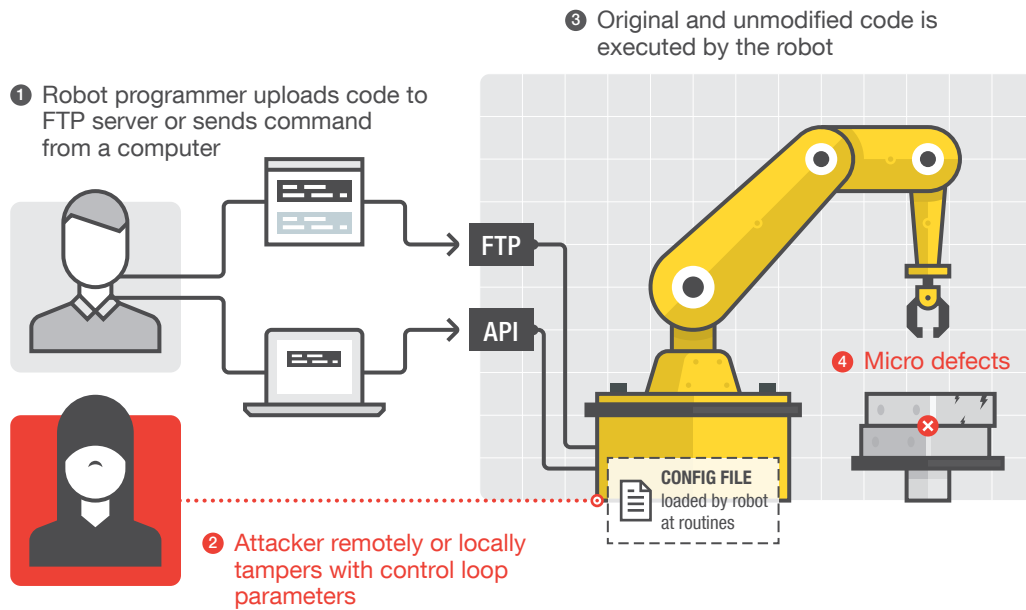


Figure 11. Attacker tampers with controller parameters to alter the execution of a program or commands issued by the programmer

## Closed-Loop Control Detuning

Precise control of the position of each link is crucial to ensure that the robot closely follows the desired trajectory, especially over long periods of time. To do this, industrial robots adopt closed-loop control techniques such as industry-standard Proportional Integral and Derivative (PID) and Proportional Position Integral and Proportional Velocity (PIV) control systems, for the angular position of each joint axis.

In general, the aim of a closed-loop control system is to make the controlled variable follow as closely as possible a reference signal (set point). The values of its parameters affect “how well” the controlled variable is able to track the set point, along with the voltage of the servo motors. Via suboptimal tuning, the controller will slowly reach the desired position, violating accuracy requirements. This, in turn, affects machining quality (i.e., the workpiece can be milled too much or welding accuracy can be reduced to the point of compromising the structural integrity of goods). Furthermore, wrong parameters may lead to controller instability, causing frequent overshoots over the desired set point. This can result in a violation of safety properties and mechanical solicitations that could induce robot breakage.

## Open-Loop Control Parameters Tampering

Speed and position control is usually implemented with additional open-loop actions, employing filters to smooth the signal generated by the closed-loop control. This means that any change to the configuration of this part will directly and immediately affect the output (position and speed). This could severely amplify resonance effects, violating the integrity requirements of the robot, or cause joint overshoots, bypassing safety boundaries.

## Robot Arm and Workpiece Configuration Tampering

Since a single model of controller must drive different robots, the physical characteristics of the manipulator (i.e., arm and joints) are configurable.

This configuration will be part of the knowledge model of the robot in the functional model (see Figure 9) and affect the overall dynamic of the system. The workpiece and manipulator are part of the system because they have characteristic weight, shape, and center of gravity and mass, which change over time (e.g., the workpiece is cut in half or another manipulator is installed).

Any unexpected modification of these parameters can result in a quantity of generated force exceeding the safety limits or simply destroying the workpiece or the surrounding environment.

In the case of co-bots, which operate with no physical fencing,<sup>17</sup> this aspect raises safety concerns to the point that standards (e.g., International Organization for Standardization [ISO]/DTS 15066)<sup>18</sup> define the maximum force and pressure levels that a co-bot can apply against each relevant part of the human body.

## Safety Limits Tampering

Control loop parameters comprise the speed limits (i.e., when in manual mode) and characteristics of brakes (i.e., minimum activation time). Despite the fact that safety measures are mechanically actuated, since control loop parameters can be configurable at runtime, the attacker may be able to bypass safety measures or change the precision of a robot's movements.

## Attack 2: Tampering with Calibration Parameters

This attack targets the sensory processing and knowledge model levels of the control system (see Figure 9). It is essential for any control system to know the precise axes position and compute for errors. The first time a robot is connected to a controller or after any configuration change, the sensing equipment must be calibrated. The controller uses the calibration data to compensate for known measurement errors when triggering servo motors.

The calibration data initially stored in the sensing equipment is transmitted to the controller during system boot. The controller then uses its local copy of the data.

When the robot is not moving, an attacker can manipulate the calibration parameters on the controller. When the robot starts moving, such manipulation can force a servo motor to move erratically or unexpectedly because the true error on the measured signal (e.g., joint position) is different from the error that the controller knows. This has the concrete consequence of violating all of the requirements.

If such a malicious manipulation happens while the robot is moving, two possible outcomes may ensue. If the controller does not supervise speed and positions, the outcome is the same as in the previous case with the additional effect that not only the final position of a joint is affected, but also the maximum speed. Instead, if the controller supervises speed and positions, it can detect unexpected movements, engaging stopping procedures. While the latter does not result in a violation of any requirement, if an attacker repeatedly triggers such manipulations at “runtime,” it can lead to a DoS attack. The robot will persist in the stop status.

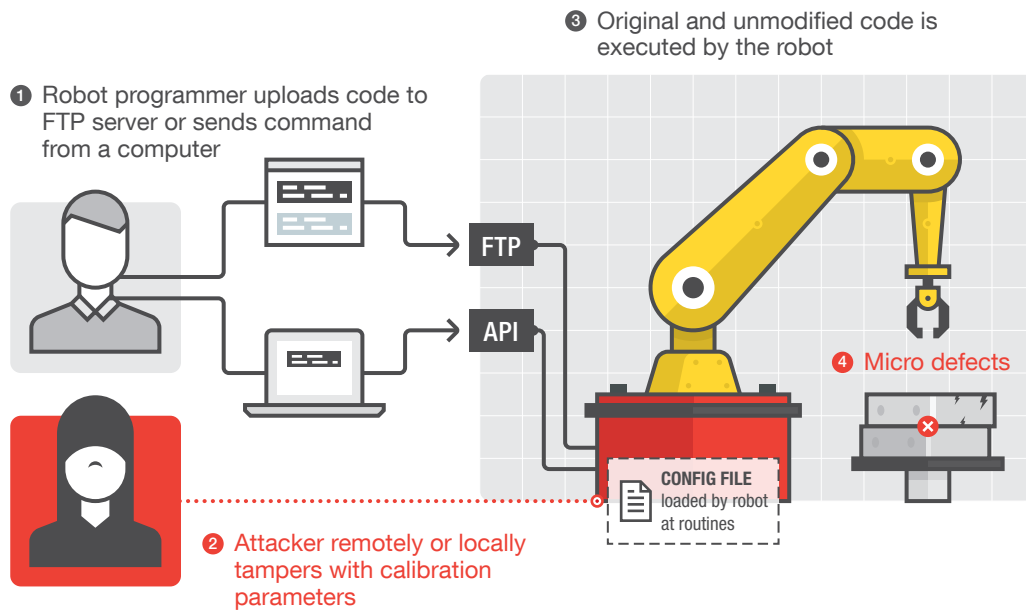


Figure 12. Attacker tampers with the calibration parameters to alter the execution of a program or command issued by the programmer

## Attack 3: Tampering with the Production Logic

This attack refers to the task level of the reference functional model (see Figure 9) such as the task program that embeds the production logic. If the controller does not enforce end-to-end program integrity, an attacker can leverage a file system or an authentication-bypass vulnerability to arbitrarily alter a program task. For example, an attacker could insert small defects, modify a workpiece, or fully compromise a company's manufacturing process.



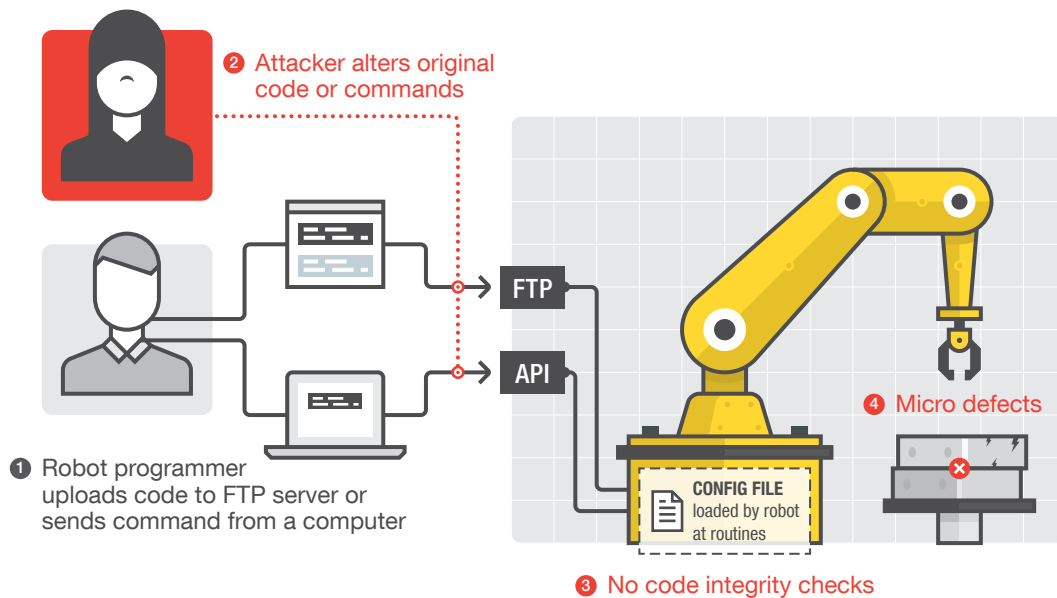


Figure 13. Attacker modifies the original program or commands in transit; lack of integrity check on the robot's side makes any modification accepted and executed blindly

## Attack 4: Altering the User-Perceived Robot State

The operator interface must provide timely information at least on the motor state (on/off) and operational mode (manual/automatic). Moreover, standards<sup>19</sup> mandate that safety-critical conditions (e.g., restarting a robot from the stop status) require deliberate user confirmation. Unfortunately, some of these conditions are communicated and require user interaction via software, not through electrical components (e.g., LED buttons). This is the case of current models of co-bots.

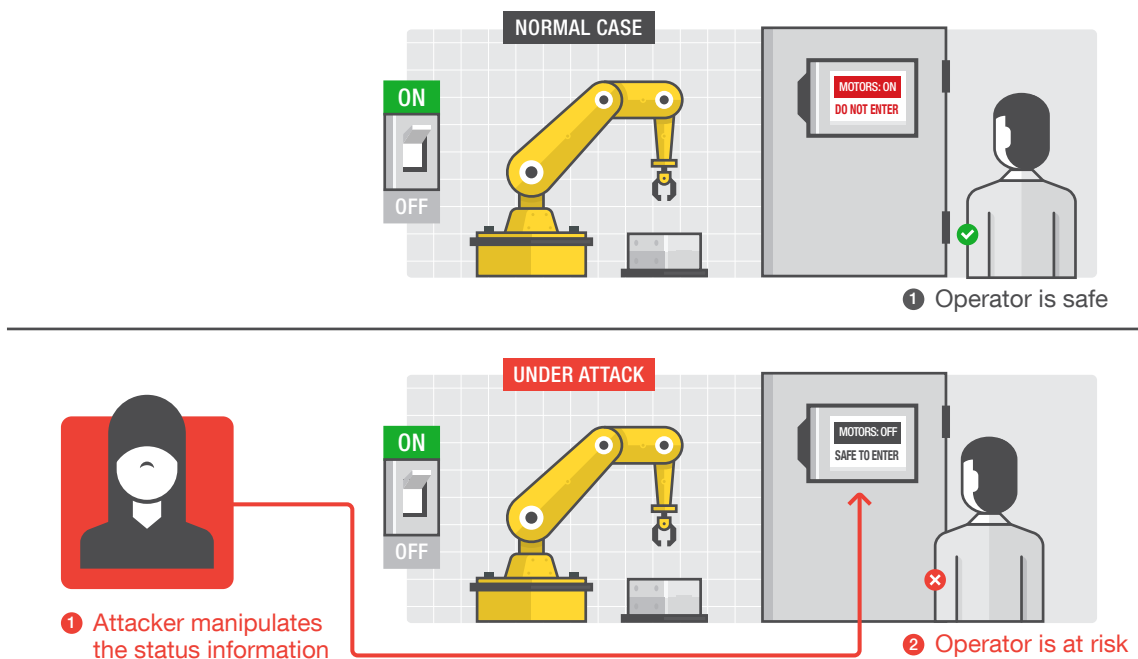


Figure 14. Attacker tampers with the user-perceived robot state to put the operator at risk

Thus, the impact of a mere user interface (UI)-modification attack is remarkable. Altering the UI could hide or change the true robot status, fooling operators into making wrong risk evaluations and consequently, creating a substantial safety hazard.

## Attack 5: Altering the Robot State

The attacker can go beyond altering the robot's perceived state. Under some conditions, the attacker can alter the true state of the robot while the operator remains unaware. This attack could be combined with other attacks for greater impact. For instance, a workpiece could be altered without even the controller noticing.

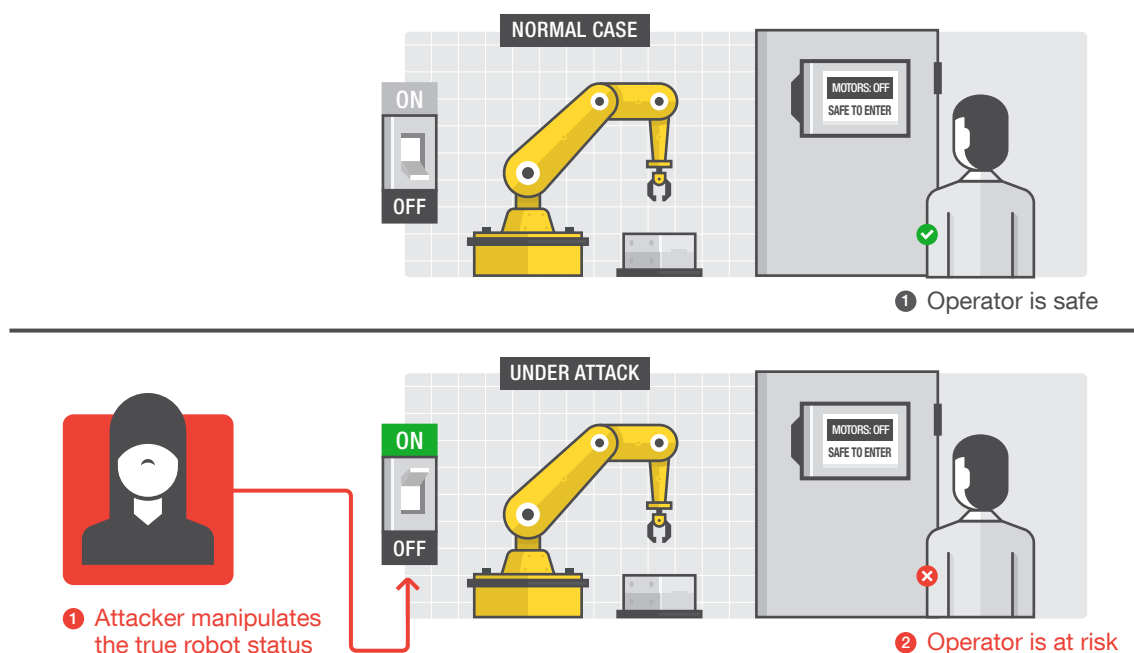


Figure 15. Attacker tampers with the actual robot state to put the operator at risk

## Motor State

In some co-bot controllers, the switch between “manual” and “automatic” mode is triggered via a software panel implemented in the teach pendant, not via a hardwired physical switch. As a result, the human operator would trust the robot when in manual mode and operate near it while an attacker could be silently changing the mode of operation and move the robot arm at full speed, causing physical harm to the nearby human. Indeed, co-bots do not operate inside a safety cage.

## Software-Defined or Wireless-Triggered Safety Features

Some vendors implement safety features such as emergency stop (e-stop) buttons in software. Worse, modern teach pendants, which of course must include an e-stop button, are wireless (e.g., COMAU's wireless teach pendant).

Therefore, safety features are subject to man-in-the-middle (MitM) or interface-manipulation attacks. For example, an MitM attacker can cause a denial of service (DoS) (i.e., forcefully stop the robot during normal operation). Moreover, an attacker can disable safety features, thus preventing the legitimate user from triggering the e-stop procedure in case of an emergency, with clear implications to the safety of the operator.

## Case Study: Demonstrating an Actual Attack

Our setup consists of ABB's six-axis IRB140 industrial robot that is capable of carrying a payload up to 6kg, equipped with the widely deployed IRC5 controller, and runs RobotWare 5.13.10371\* (based on the VxWorks runtime) and FlexPendant (teach pendant) based on Windows® CE. The IRB140 must operate in a cage and relies on default, standard safety measures.

To demonstrate how much an attacker can learn and in order to carry out the attacks we presented, we reverse engineered the RobotWare control program (i.e., the software running on the FlexPendant) and the RobotStudio software suite, which are freely available for download from the vendor's website.

The full technical details of our setup, including a thorough security analysis of the main computer and FlexPendant, and the vulnerabilities we found in them, can be found in our academic paper that will be presented at the IEEE Symposium on Security and Privacy.<sup>20</sup>

## Entry Points: Industrial Routers and Remote Access

To verify the possibility of remote attacks, we started by querying Shodan for the default banner of the embedded web server of each of the top industrial router brands. We started with eWON, which is ABB's choice for remote access boxes (see Figure 16) to remotely maintain its robots. These routers' embedded web server can be found using the "Server: eWON" search pattern. Other vendors such as Moxa, Sierra Wireless, Westermo, and so on (see Table 1) are equally easy to find (e.g., putting the 'Moxa' string in the raw response header or the 'Westermo' string in the 'Basic realm=' HTTP response header). Even though this search is conservative because it does not include customized banners, we found several devices that expose the configuration web interface to the Internet, mostly through a cellular network. The web-based console, when enabled also for the interface, provides an easily "fingerprintable" attack surface to

---

\* Although the latest version is the 6.x series, our controller supports only the 5.x series: However, we verified with ABB and by manual reverse engineering that our findings apply to the latest version of the 6.x series at the time of running the experiments.



exploit vulnerabilities or misconfigurations in the router and gain access to an industrial robot (as well as other industrial devices connected to them).

Vulnerabilities in this kind of systems are quite common (see Table 1). For example, we analyzed ABB's Service Box device in a black box fashion and discovered a severe authentication-bypass vulnerability that allows an attacker to read the configuration and some device information (e.g., event logs) without knowing the administrator password. We disclosed this vulnerability to the vendor (through ABB), which fixed the issue in the latest firmware revision (11.2s2).

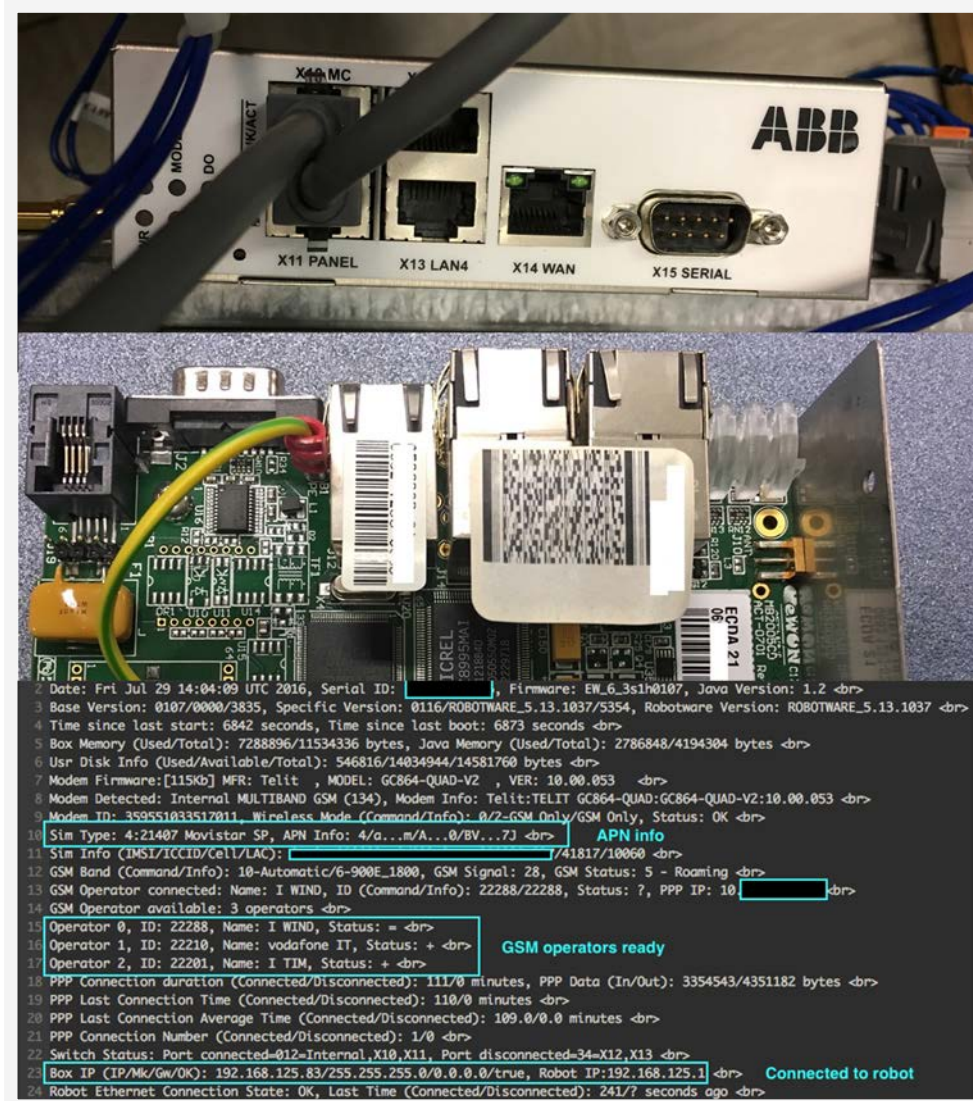


Figure 16. ABB's Service Box (a rebranded eWON industrial cellular router) attached to our robot with system logs that show the connection to the robot via the service APN

Without going into too much detail, we summarized the issues we found with the following taxonomy. Note that the exploitability of the vulnerabilities strongly depends on the context and custom solutions adopted by vendors. For instance, web interfaces are not necessarily exposed, although we found many instances of exposure.

- **Information disclosure and “fingerprintability”:** It was too easy for us and so can be for an attacker, to find and recognize devices mainly because of the following issues:
  - Sensitive technical materials (documentation and software) are available on vendors’ websites. Although we are by no means supporting the concept of “security through obscurity,” all of our findings on industrial routers were possible, thanks to the ample and easily accessible technical materials (including firmware images) available on almost all of the top vendors’ websites.
  - The same certificates are reused across all product instances, most of the time self signed. In some cases, the country and distinguished name on the certificate were all we needed to find exposed devices of a certain brand.
  - Network service banners include detailed information such as the vendor’s name, media access control (MAC) address, firmware version, CPU model, and sometimes even CPU frequency.
- **Outdated software components:** As simple as this might seem, some software components that run on these devices were outdated, sometimes based on unsupported versions despite having a recently built firmware image. In the best cases, although not certainly a good practice, vendors apply custom patches to eliminate known vulnerabilities from the original (outdated) software to maintain back compatibility. In the worst cases, outdated libraries were used “as is.”
  - Application-level libraries and tools (e.g., Busybox 1.6, which is not even downloadable from the main site anymore)
  - Compiler (e.g., GCC 3.3)
  - Kernel (e.g., Linux 2.4)
  - Cryptography libraries
- **Default credentials or poor authentication:** Despite this being a common class of vulnerability, we are still surprised to find it, given the ease of fixing it.
  - Null default passwords are not a big deal as long as there is a procedure that forces a user to change them at first boot. However, given that most consumer-level routers nowadays come with randomized credentials, we believe that industry-grade devices should follow the same sane approach.

- Private or static OpenVPN keys were found in a couple of firmware images available on the vendors' websites. If not paired with a mandatory key reset procedure, we consider this a critical finding because, while users have grown accustomed to changing default passwords, they are not necessarily familiar with the concept of shared or secret keys and might underestimate risks.
- Weak hashing functions allowed us to recover default credentials pretty easily.
- **Poor transport encryption (OWASP I4):**<sup>21</sup>
  - Outdated cryptographic libraries or ciphers, as explained above, also fall in this category.
  - Misconfigured cryptographic software refers, for example, to the use of shared, symmetric keys for VPNs. This is not a good practice because it does not allow access auditing with the obvious revocation issues in case it gets lost.
  - Web-based administration interfaces are not always on HTTPS despite being the main access point for management.
- **Insecure web interface (OWASP I1):**<sup>21</sup>
  - Poor input sanitization is used. We found one vendor's web interface accepting straight GET parameters without any sanitization.
  - Unmaintained open source codes are used "as is." The same vendor included a large portion of PHP code coming from a blog post of a novice developer, explaining how he implemented a REST layer in PHP. Needless to say, that software is unmaintained and the vendor did not change it (it indeed included the very same vulnerabilities).
- **Poor software protection:** On top of easily accessible firmware images, we found that custom binaries (e.g., the embedded web server) built by some vendors include all debug information (i.e., unstripped). Generally, all but one vendor's firmware images were easy to open with Binwalk's default settings.

At the moment, we are coordinating with other top vendors of industrial routers (see Table 1) to whom we disclosed our findings.

## Other Vulnerabilities in Robot Components

Access	Channel
Physical	USB port
	Industrial bus access via after-market end effectors, for instance
	Ethernet: LAN service port
	Direct access to internal devices (e.g., axis computer)
Local	LAN port
Remote	WAN access to (unfirewalled) LAN port
	WAN access to remote service facilities such as a service box
Wireless	Wireless (e.g., GSM) access to remote service facilities

Table 3. Attack surfaces by channel

We were able to identify several weaknesses in how industrial robots are designed to connect to the outside world. Among these are optional user authentication and how, in the boot process, any firmware or application file coming from the File Transfer Protocol (FTP) server is implicitly trusted.

The robot's main computer and flex pendant expose a number of network services that are essential for the robot's operation. Standard services such as FTP are used to share files and system information between a robot and an internal network, and custom services. RobAPI is a custom service that offers the most extensive network attack surfaces with its several commands and inputs. Moreover, the main computer exposes a User Datagram Protocol (UDP)-based discovery service used by FlexPendant and RobotStudio to automatically discover robot controllers on a network. FlexPendant uses broadcast UDP packages to send debug information and messages.

The main computer is the most sensitive entry point, as it exposes various services to a network, and gaining unauthorized access to it leads to the complete compromise of the controller. It performs sensitive operations on its own and the communication between internal components of the robot is implicitly trusted. The following summarizes the vulnerabilities we found in the robot's main computer:

- **Vulnerability 1: Unsecured Network and Command Injection (ABBVU-DMRO-124642):** Network-exposed services are an important attack vector. Later in the case study, we show that an attacker with read and write access to an FTP-exposed file system can abuse network services to directly control the robot's actions.



- **Vulnerability 2: Weak Authentication (ABBVU-DMRO-124644):** We discovered that an attacker can bypass the User Authentication System (UAS) because of several implementation flaws: 1) disabled authentication during system boot, 2) use of a default user name (without a password) that cannot be changed or removed, and 3) the use of a specific user that comes with a set of unchangeable hardcoded credentials.
- **Vulnerability 3: Naive Cryptography:** An attacker with read-only file system access can tamper with the UAS configuration, changing the privileges of existing accounts and changing or retrieving all of the users' passwords.
- **Vulnerability 4: Memory Corruption (ABBVU-DMRO-124645, ABBVU-DMRO-128238):** We found an exploitable memory error (a textbook stack-based buffer overflow) in the code that receives RobAPI requests for the DHROOT handler.

The flex pendant, on the other hand, also contains vulnerabilities:

- **Vulnerability 5: Missing Code Signing:** The boot image that the flex pendant downloads from the main computer is not signed and can be easily modified by an attacker who knows how to reverse engineer the file format.
- **Vulnerability 6: Memory Corruption:** We found a memory error in the executable, TpsStart.exe, executed during the system startup process, which allows an attacker to trigger a stack-based buffer overflow if he can modify a file to make its file name longer than 512 bytes.
- **Vulnerability 7: Poor Runtime Isolation:** We found some issues in the compliance tool that comes with the FlexPendant software development kit (SDK). The tool does not actually enforce certain important restrictions, including preventing the use of namespaces that allow access to raw file system and RobAPI capabilities.

## Attack Implementation Step 1: Controller Compromise

We will now show how an attacker can leverage vulnerabilities similar to the ones we found to create a multistep attack and take complete control of the controller. Here, we consider the most general adversary—a remote network attacker that can reach the RobAPI server or FTP without knowing valid UAS credentials.

We implemented the following exploitation sequence:

1. **Main computer compromise:** We used the FTP static credentials we discovered to access the / command driver. Alternatively, we could have exploited the memory errors found in the RobAPI to gain initial access.

2. **Authentication bypass:** We temporarily disabled the UAS and triggered a full system reboot. To this end, we invoked `uas_disable` via the shell function of the FTP `/command` and alternatively, via exploiting the remote code execution vulnerability (e.g., the RobAPI memory error).
3. **Payload upload:** With full FTP access without credentials, we permanently disabled UAS by editing the obfuscated configuration file then uploaded custom, malicious, .NET dynamic-link libraries (DLLs) (or RAPID code) to the running system directory on the controller.
4. **Persistent access:** We triggered another reboot via the `/command` shell reboot FTP function, causing FlexPendant to auto execute the uploaded malicious .NET libraries or the main computer to load the malicious RAPID files.

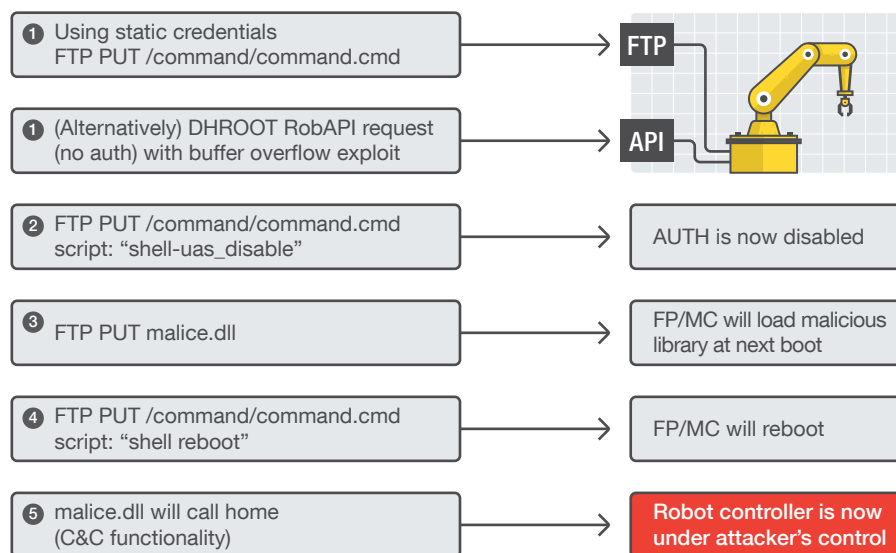


Figure 17. Multistep attack to gain control of the robot's controller through a remotely reachable FTP server

In the FlexPendant context, due to lack of proper sandboxing, an attacker can gain complete access to OS resources, and any arbitrary remote code execution vulnerability in the RobAPI would imply unrestricted control of the executed code. Another possibility is to obtain full unrestricted access to the FTP to upload a maliciously crafted RobotWare image, allowing an attacker to compromise all other components because the firmware images are not signed.

## Command and Control

Although not essential, we exploited the FTP-shared file system as a simple command-and-control (C&C) server to exchange data with the compromised FlexPendant without needing a direct network connection to it. Another possibility is to load a RAPID module that is executed on the main controller to act as relay between a bot and a remote attacker. For this reason, we considered that, at this point, an attacker has completely compromised the robot controller. We started from this assumption to describe the implementation of an attack.

## Attack Implementation Step 2: Robot Compromise

We will hereby now explain how we implemented robot-specific attacks to show that an adversary can affect the production outcome after compromising a robot controller.

Some attacks presented can have a very costly impact, possibly up to the entire cost of the deployment, or violate safety policies and regulations. When unable to run an attack for this reason, we discussed our implementation with a domain expert who confirmed its feasibility and potential destructive effects.

### Accuracy Violation (With Attack 1)

The robot we analyzed uses a closed-loop controller to control the joint position. For each joint, a PID controller ensures that its angular position follows as closely as possible the reference trajectory needed to complete a task. As mentioned in attack 1, if an attacker can “detune” the controller, he can reduce the accuracy of a movement and ultimately, impair the machining precision.

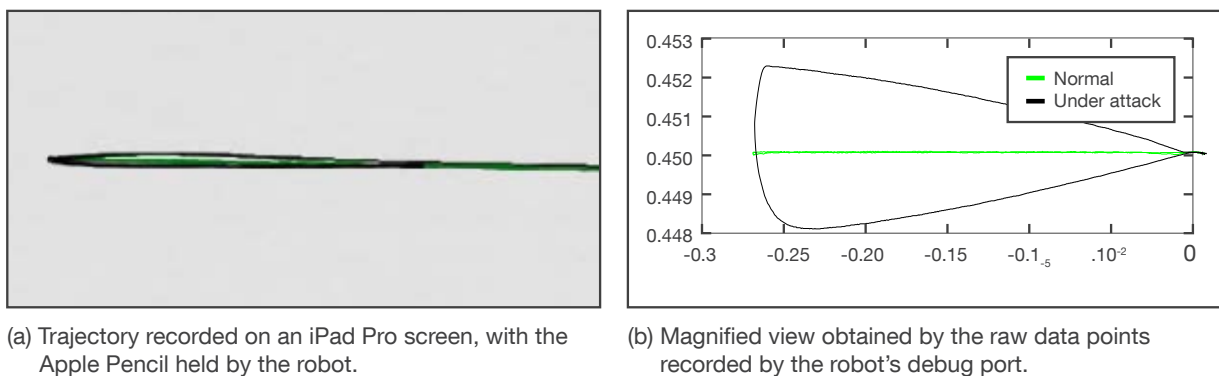


Figure 18. The trajectory followed by the robot when programmed to move along a straight line

In both cases (see Figure 18), the green (normal) and black (under attack) lines when the programs were executed appear to be the same. During the attack, an operator standing by the robot would not notice any difference in the arm's movements. However, while under the effect of attack 1, the robot's trajectory is not precise because the attacker modified the open-loop control parameters. A full video demo is available at <https://www.trendmicro.com/vinfo/us/security/news/internet-of-things/rogue-robots-testing-industrial-robot-security>. In the attack shown in this video demo, we use a combination of the vulnerabilities listed in pp. 29-30 and Figure 17.

In our security analysis, we observed that the PID parameters are stored in a robot-specific configuration file on the file system of the main computer. This file is naively obfuscated. For example, in an IRB140 robot, it is located in the directory under robots/irb1\_140/irbcfg/sec\_140\_0.81\_5\_typea\_1.cfg.enc.

To precisely measure the trajectory of the end effector under nominal and attack conditions, we decoded such data from the RS232 service port and real-time Ethernet by enabling a debug functionality in the main computer, which allowed us to collect the coordinates of the position at fast sampling intervals. While doing so, we did not interfere with the attacker's capabilities.

By leveraging the remote code execution vulnerability, we modified the control-loop configuration files, which are naively obfuscated and thus easily modifiable. In particular, we changed the proportional gain of the first joint's PID controller, setting it to 50% of its original value. Then we programmed the robot to perform a straight horizontal movement. Figure 18 shows the trajectory of the end effector projected on the horizontal plane, which has been notably altered. Although the maximum difference between the position under normal conditions and under attack is small (less than 2mm), according to the specific machining that the robot is performing, it can be enough to destroy the workpiece.

## **Safety Violation (With Attack 4)**

We implemented this attack using the user-perceived robot state alteration approach to trick the operator into thinking that the robot is in manual/motor-off mode when it was in auto/motor-on mode. Recall that in auto/motor-on mode, an attacker can load a program task that pilots the robot while the operator believes that it is in manual/motor-off mode and it is safe to get close to it.

We implemented this attack by leveraging the lack of code signing in the flex pendant firmware, which we reverse engineered to find routines that implement the UI. We modified the strings that were shown on the display to show false information to the operator and repackaged the binary firmware. Then we used the authentication bypass on the FTP to load the binary onto the main computer's file system and waited for the next reboot that had the effect of loading the malicious firmware onto the teach pendant. As an alternative, we could have leveraged the command-injection vulnerability (shell reboot) to force the main computer to reboot remotely. Figure 19 shows the modified UI.

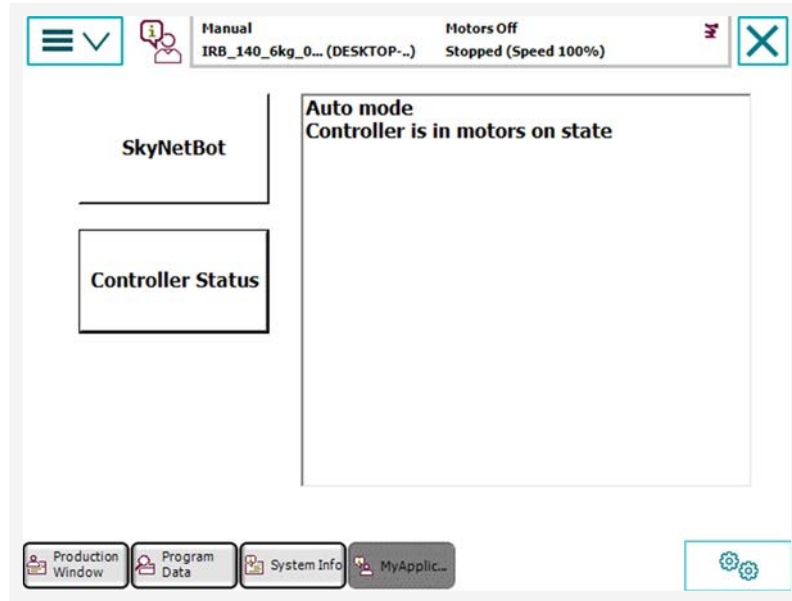


Figure 19. Modified FlexPendant screen showing the robot is in “motors off” (or “manual mode”) while it is in in “auto” (or “motor on”) mode, which means that it will not respond to manual commands issued through the HRI joystick

## Integrity Violation (With Attack 2)

It is possible to violate a robot’s integrity through the control-loop alteration and calibration parameters tampering approaches described earlier. We wanted to overshoot the joints in order to collapse the robot on itself and force the servo motors beyond their physical, structural limits. Note that this attack is costly and potentially destructive because its goal is to damage the robot.

Alternatively, an attacker could use the robot state alteration approach to repeatedly and abruptly start and stop a servo motor, causing electromechanical components, the brakes, and the servo motor to wear.

We implemented the software stages of this attack using two vulnerabilities. We exploited the remote code execution bug in the main computer to load the configuration files, holding the calibration parameters. Since these files were weakly obfuscated through “encryption,” we leveraged this vulnerability and reversed the logic, which allowed us to write arbitrary, valid content on such files, which were correctly loaded by the main computer.

At this point, instead of starting the robot, we showed the modified configuration files to a domain expert (lab technician who normally operates the robot) who confirmed the destructive effects on the robot. Additionally, we checked the effect of such configuration change on the sections of code that implement control and supervision routines, which we reverse engineered manually. The only integrity checks that we found in the speed and position supervision routines can be clearly bypassed using any of the vulnerabilities that we found for modifying the main computer’s control logic.



# Possible Threat Scenarios for Robot Users

Threat actors are motivated by different goals, which could be monetary, but also political in nature. They may act to further a personal idea or on behalf of a larger network that pursues business goals. Given the safety-critical and economic characteristics of targets, nation-state level interest is also imaginable. An actor who would be able to reach the same access level and target knowledge of an insider would be able to launch sophisticated attacks from remote endpoints and not be constrained by cost.

In the particular case of exploiting industrial robots, anyone could have an interest; even an insider threat cannot be underestimated in this context. Since industrial robots are key to manufacturing products or delivering services in different industries, including critical sectors, threat actors who are studying possible avenues of attack may realize the promise that various devices hold for being hacked, especially if they intend to physically sabotage their targets. Enterprises that have not undergone adequate security reviews with regard to their use of connected robots may be at risk.

Given the vulnerabilities that we found in our security analysis and the robot-specific attacks summarized in the previous section, the threat scenarios below seem possible.

## Threat Scenario 1: Production Outcome Alteration or Sabotage

By exploiting robot control, we were able to inject faults and microdefects into the workpiece, as shown on the right-hand side of Figure 20. A full video demo of the attack is available at <https://www.trendmicro.com/vinfo/us/security/news/internet-of-things/rogue-robots-testing-industrial-robot-security>.

While it would take a complete breakdown of every step of the production line for these defects to reach the end user and cause significant financial loss, this attacker capability is an important entry point that industrial robot operators must consider. A certain volume of products, for instance, may be rendered unusable, as large batches are tested by the quality control department and do not make it to the customer. Taken to the extreme, however, should microdefects successfully evade detection by a vendor's multiple checks and depending on the nature of the goods themselves, injury or fatality could occur.

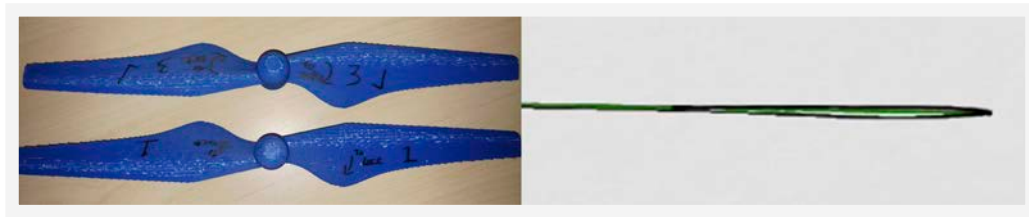


Figure 20. Left: 3D-printed drone rotor (top: non-sabotaged, bottom: sabotaged), as demonstrated by Belikovetsky, et al.<sup>22</sup>; Right: a microdefect (in the trajectory followed by a robot in production) that we introduced using one of our attacks without altering the source code of the program executed by the robot.

In Figure 20, during the attack, the robot's arm overshot a couple of millimeters (black line) and the operator did not notice any unusual movement of the arm until close inspection of the workpiece.

## Threat Scenario 2: Ransomware Attacks on Altered Products

Although the ransomware scheme may sound like a “hype” at the moment, there is a unique monetization opportunity for an attacker here. Depending on the thoroughness and nature of quality control that is implemented in the target facility and what other checking mechanism is in place, an attacker may be able to stealthily introduce microdefects in the production chain, keep track of which products are affected by such microdefects, and later on contact the manufacturer asking for ransom to reveal which lots were affected.

If an attacker successfully does so and the value of disposing finished goods or having masses of faulty products on their hands may be deemed more expensive than the ransom demand or worse, vendors may consider paying. This is a hypothetical scenario, as many factors need to be in place and attackers may opt for quicker ways to profit from victims. However, if successful or worth it, an attacker could get to be in a position to provide a “sample” of damaged goods, which can act as a strong lever.

## Threat Scenario 3: Physical Damage

An attacker that can control a robot can damage its parts or even cause injuries to people who work closely with it, for instance, by disabling or substantially altering safety devices. We demonstrated two attacks that can make this threat scenario a reality. First, we showed that an attacker can alter a robot's state as perceived by a human operator by altering the status information displayed on the HRI device (see Figure 21). Second, we showed that an attacker can instead alter the true robot state without its operator noticing. In both cases, the operator thinks it is safe to walk or stand near the robot even if in that very moment, an attacker is controlling its movements.



Figure 21. Left: HRI screen used to manually control the movements of a robot's arm from abb.com; right: modified HRI screen showing that the robot is in “motor off” (or “manual mode”) when it is in “auto” (or “motor on”) mode, which means it will not respond to manual commands issued through the HRI joystick

## Threat Scenario 4: Production Line Process Interference

Depending on the extent of damages caused by a cyber attack, production may or may not be promptly restarted (e.g., the amount of time for repairs varies greatly). On the one hand, an attacker may cause a robotic arm to behave erratically but only requires a quick reboot by the operator if the intrusion's source is immediately detected. On the other hand, an attacker may cause a robotic arm to overextend or overexert force, damaging parts of the production line, forcing an operator to halt the production for the affected step, thereby causing short-term bottlenecks and some production delays that, depending on the type and size of a target company, can have varying financial ramifications.

## Threat Scenario 5: Sensitive Data Exfiltration

A robot is “just” another computer attached to a factory network. If not properly patched, it represents another entry point to a network. Moreover, because of tasks that it has to accomplish, it does sometimes store sensitive data (e.g., source code or information about production schedules and volumes), including, for instance, industry secrets. Industry secrets include the calibration parameters of the robot, which are protected by both vendors and users to prevent unauthorized access. However, the protection mechanisms in place, are often insufficient. Clearly, an attacker that finds a vulnerable robot has direct access to such data, which is highly valuable on the market or simply for a competitor.

# Toward a Brighter Future

While we consider the discovery and subsequent resolution of the vulnerabilities we discovered important, we want our research to highlight how easily and effectively those vulnerabilities can be used to carry out complex attacks in industrial robot architectures. What we have seen led us to believe that the answer to a more secure robotics ecosystem lies not exclusively in improving the quality of the embedded software running on robots nor more disclosures and faster patching, but in more a holistic approach.

## Cybersecurity and Safety Standards

Industrial robot standards emphasize safety requirements,<sup>23</sup> such as stopping functions, e-stop features, required pendant controls, and speed bounds. Unfortunately, none of the standards explicitly account for cybersecurity threats when our research clearly shows the feasibility and possible impact of such. Some safety standards have mild security implications without explicitly accounting for adversarial control during risk assessment. These standards should be reexamined in the same way that cybersecurity issues in the ICS<sup>24</sup> and automotive<sup>25</sup> sectors have been by their respective standardization bodies.

## Security Measures and Challenges

Any industrial robot vendor's already hard task of designing a secure architecture without sacrificing functionality is exacerbated by challenges in providing timely security updates.<sup>26</sup> Compared with other information and communication technology (ICT) systems, industrial robots have a very long lifetime, which increases the burden on vendors to support several deployed devices, leading to so-called "forever-day" vulnerabilities (i.e., well-known vulnerabilities that are never patched). Moreover, lack of system-level hardening features renders the exploitation of a vulnerability easier than on a mainstream piece of OS. This is amplified by the "patching problem" where customers may be worried by downtimes or potential regressions carried by software updates and thus refrain from timely patching their systems.

In addition to the overarching issue described above, the following sections show other challenges that arise when applying even textbook-level security practices in the industrial robots' domain.



## Human Interaction

Security mechanisms must be put in place to ensure that human operators can override unexpected or unwanted robot behaviors in a safe manner. Human interaction, for instance, could modify the outcome of some of the attacks we presented. If emergency safety stops are correctly implemented by using electromechanical switches, an operator could completely stop the robot should he notice anything amiss. This is not, however, applicable to less-visible attack scenarios.

## Attack Detection and System Hardening

Effective and readily applicable attack detection approaches are needed to provide a short- to medium-term solution for threat mitigation.

### Attack Detection (Short Term)

Research and industry efforts should provide short-term solutions to mitigate the impact of vulnerabilities. For example, detection and correction of software and network anomalies must be explored, rather than focusing on access control techniques. As for automotive systems,<sup>10</sup> a delicate balance between detection and prevention exists and should be considered since prevention measures may interfere with the production chain and induce downtimes.

### System Hardening (Short and Medium Term)

As it is hard to patch all vulnerabilities in a complex piece of software, medium- to long-term solutions include system hardening to make reliable exploitation more expensive. Implementing these techniques in legacy embedded platforms is challenging, as they may require hardware support and design changes.

In fact, the exploitation of memory corruption vulnerabilities in the industrial robot we analyzed is far easier than in mainstream OSs. Industrial robots, like many embedded systems, employ real-time OSs (RTOSs) with poor or no hardening features. Surprisingly, although RTOSs are used in critical tasks, research in this area is more focused on determinism, efficiency, and safety, rather than on system security.

An effective, short-term way to harden an RTOS is by enabling OS- and compiler-based mitigations such as address space layout randomization (ASLR), data-execution prevention (DEP), and canaries. This set of functionalities is common, but not always supported in embedded systems, especially those based on outdated software toolchains.

Another effective containment measure is to enforce privilege separation at the OS level by physically separating critical functionality across different subsystems. The challenging part here is the trade-off between security and real-time requirements. For instance, the VxWorks 5.x OS used in ABB's RobotWare

5.x executes all the code in kernel mode and built as a monolithic ELF binary. Although VxWorks 6 supports user- and kernel-mode demarcation since 2004 for platforms with memory management unit (MMU), which enables the so-called “real-time process model,” this functionality is not used in newer versions of RobotWare (12 years later).

This reasoning applies also to functionality aimed at executing custom code. Even though the flexibility of industrial robots requires the execution of customized software, they do not need access to any functionality of the underlying OS. Custom applications and robot programs can be limited to a sandboxed environment, following the principle of least privilege. For example, Microsoft .NET’s Application Domains<sup>27</sup> can be used as a light sandboxing mechanism in .NET-based embedded systems such as a teach pendant.

## **Software Design and Deployment Challenges**

We identified some steps that vendors can adopt to reduce the impact of security issues. Although well-known in the security community, applying them in this domain is challenging due to time-consuming patching processes and nontrivial changes in the controller design.

### **Program Protection (Short Term)**

Removing or making it easy for users to disable detailed debug outputs and symbols can play a relevant role in increasing the cost of an attack, especially for casual attackers. In the system we analyzed, the detailed debug output was readily available from the serial console, which eased reverse engineering and exploit development. We also observed that the ABB’s flex pendant broadcasts detailed debug information through UDP packets, making them accessible to a physical attacker connected to the main computer service port.

### **Secure Software Development Life Cycle (Long Term)**

In general, enforcing secure software engineering practices can improve code robustness and harden an underlying platform. Such practices range from forbidding the use of unsafe C library functions (e.g., strcpy, strcat, and sprintf) to the use of static-analysis tools to find potentially problematic code regions and unsafe implementation patterns. These practices must be a part of a comprehensive process that takes security into account during the whole software development life cycle. Unfortunately, the fact that we found a textbook vulnerability in one of the most widely deployed robot controllers is an indicator that not even basic static checking was put in place.

## **Secret Management (Medium Term)**

In our case study, we observed frequent use of static, wired-in credentials, shared among devices of the same model; obfuscation of passwords (as opposed to hashing and salting); and naive “encryption” of configuration files. These measures create a false sense of security. For example, a superficial review of the software source code may reveal that a certain functionality is authenticated when it is instead accessible with a password readily available in the firmware executable.

## **Component Interconnection Hardening (Medium Term)**

In industrial robots as well as in the ICS and automotive domains, the threat model assumes that the internal (i.e., factory) network is a trusted element. This assumption is not realistic in scenarios where a component is compromised. Even if such a compromised component is not critical to the operation of the system, it may grant an attacker access to the trusted network. An effective hardening measure is to move toward an Industry-4.0-compliant threat model where the internal network is not trusted, and the boundary between internal and external is much more dynamic. This implies, for instance, appropriately filtering inputs coming from connected components as if they were coming from an untrusted party and taking into account eavesdropping and tampering of messages.

Attacks similar to the UI alteration proposed in attack 4 can be mitigated by not trusting the teach pendant software. For example, if the dead-man switch is hardwired to the robot controlled, it can be used to request user confirmation.

## **Code and Configuration Signing (Medium Term)**

An effective system-level form of mitigation is implementing code-signing mechanisms with strong authentication.

Broadly speaking, we can distinguish among three types of executable code—vendor-provided firmware, program task code, and custom software developed on top of a vendor-provided SDK. It is expected that only the vendor can develop and run updated firmware for robot components. For custom code, a customer must be able to sign code for his own robot (but not on other customers’). That way, although an attacker uploads arbitrary files to the robot file system, they would not gain arbitrary code execution.

Solving the problem on a global scale is challenging due to the trade-off between safety and security, and development and deployment time and effort. A possible way to tackle it is using a public key infrastructure (PKI) with the vendor as a certification authority, issuing per-customer certificates.

Fortunately, commercial embedded OSs and hardware platforms are moving toward supporting secure boot capabilities (e.g., the Security Profiles in VxWorks 7), which would allow the implementation of a full code-signing chain.

A mechanism to keep the flexibility of the teach pendant programming model intact, if sandboxing mechanisms are in place to limit the privileges of user-provided program task codes, is to enforce the code-signing policy only when the robot is running in automatic mode. Before switching to automatic mode, the code can be signed offline or even online (e.g., via smart card devices connected to the teach pendant).

In the system we analyzed, we found various encrypted configuration files. It is unclear if the intended use of encryption for these configuration files is to avoid casual tampering of the configuration or the developers had a more complex threat model in mind. However, these configuration files control safety parameters and signal routings, and are critical for the operation of the robot. If the latter is the case, requiring that the configuration files be signed (with the same infrastructure in place for custom code) will address this problem effectively.

# Conclusion

This paper represents the first step in a broader exploration of security issues in the industrial robotics ecosystem. We explored, theoretically and experimentally, the challenges and impacts of the security of modern industrial robots. Our security assessment allowed us to reach the conclusion that robot-specific attacks are well within the realm of possibility and as such, must be considered seriously by industrial robot operators, some of whom include critical sectors.

We were able to demonstrate in a laboratory setting using a widespread model of industrial robots (representative of a de facto standard architecture) attacks that modify the robot's controller and its calibration parameters, attacks that manipulate the robot's user-perceived status and its actual status, and attacks that modifies the task logic executed by the robotic arm. Threat actors may use any of these attack classes to reach a specific goal, which can be monetary, political, or otherwise. Impact-wise, they may range from sabotage and business disruption to data exfiltration. We also simulated an entire attack algorithm from an entry point to infiltration and compromise to demonstrate how an attacker would make use of existing vulnerabilities in order to perform the attacks we discussed.

Ultimately, the call for security extends beyond industrial robot operators but to all players involved in bringing these devices into production and the market. Between cybersecurity standards makers, robot software developers, robot vendors, and network defenders, the overall objective should be to make reliable exploitation of existing and future vulnerabilities more expensive for attacks.



# References

1. Numaan Huq, Stephen Hilt, and Natasha Hellberg. (15 February 2017). *Trend Micro Security News*. “U.S. Cities Exposed: Industries and ICS.” Last accessed on 5 April 2017, <https://www.trendmicro.com/vinfo/us/security/news/internet-of-things/us-cities-exposed-in-shodan>.
2. Davide Quarta, et al. “An Experimental Security Analysis of an Industrial Robot Controller.” In the Proceedings of the 38th IEEE Symposium on Security and Privacy, vol. (to appear), S&P 2017. San Jose, CA: ACM, May 2017.
3. IFR. (2017). *International Federation of Robotics*. “Highlights.” Last accessed on 5 April 2017, [http://www.ifr.org/index.php?id=59&df=2016FEB\\_Press\\_Release\\_IFR\\_Robot\\_density\\_by\\_region\\_EN\\_QS.pdf](http://www.ifr.org/index.php?id=59&df=2016FEB_Press_Release_IFR_Robot_density_by_region_EN_QS.pdf).
4. Detlef Zuehlke. (April 2010). *ScienceDirect*. “SmartFactory—Toward a Factory of Things.” Last accessed on 5 April 2017, <http://www.sciencedirect.com/science/article/pii/S1367578810000143>.
5. ABB. (2016). *ABB*. Last accessed on 5 April 2017, [http://developercenter.robotstudio.com/Index.aspx?DevCenter=Robot\\_Web\\_Services](http://developercenter.robotstudio.com/Index.aspx?DevCenter=Robot_Web_Services).
6. Jens Lambrecht, Moritz Chemnitz, and Jörg Krüger. (21 April 2011). *IEEE Xplore*. “Control Layer for Multivendor Industrial Robot Interaction Providing Integration of Supervisory Process Control and Multifunctional Control Units.” Last accessed on 5 April 2017, <http://ieeexplore.ieee.org/document/5753492/>.
7. Jens Lambrecht, Moritz Chemnitz, and Jörg Krüger. (18 August 2011). *YouTube*. “iPhone Industrial Robot Control—KUKA KR 6.” Last accessed on 5 April 2017, <https://www.youtube.com/watch?v=yFi7UL70zTo>.
8. Ahmad-Reza Sadeghi, Christian Wachsmann, and Michael Waidner. (27 July 2015). *IEEE Xplore*. “Security and Privacy Challenges in Industrial Internet of Things.” Last accessed on 5 April 2017, <http://ieeexplore.ieee.org/document/7167238/>.
9. Stephen Checkoway, Damon McCoy, Brian Kantor, Danny Anderson, Hovav Shacham, and Stefan Savage. “Comprehensive Experimental Analyses of Automotive Attack Surfaces.” Last accessed on 5 April 2017, <http://www.autosec.org/pubs/cars-usenixsec2011.pdf>.
10. Karl Koscher, Alexei Czeskis, Franziska Roesner, Shwetak Patel, Tadayoshi Kohno, Stephen Checkoway, Damon McCoy, Brian Kantor, Danny Anderson, Hovav Shacham, and Stefan Savage. “Experimental Security Analysis of a Modern Automobile.” Last accessed on 5 April 2017, <http://www.autosec.org/pubs/cars-oakland2010.pdf>.
11. Martin Brunner, Hans Hofinger, Christoph Krauß, Christopher Roblee, Peter Schoo, and Sascha Todt. (December 2010). *Chair for IT Security*. “Infiltrating Critical Infrastructures with Next-Generation Attacks: W32.Stuxnet as a Showcase Threat.” Last accessed on 5 April 2017, <https://www.sec.in.tum.de/christoph-kraus/publication/189>.
12. Homeland Security. (2015). “NCCIC/ICS-CERT Year in Review: National Cybersecurity and Communications Integration Center/Industrial Control Systems Cyber Emergency Response Team: FY 2015.” Last accessed on 5 April 2017, [https://ics-cert.us-cert.gov/sites/default/files/Annual\\_Reports/Year\\_in\\_Review\\_FY2015\\_Final\\_S508C.pdf](https://ics-cert.us-cert.gov/sites/default/files/Annual_Reports/Year_in_Review_FY2015_Final_S508C.pdf).
13. James R. Hagerty. (13 April 2015). *The Wall Street Journal*. “New Robots Designed to Be More Agile and Work Next to Humans: ABB Introduces the YuMi Robot at a Trade Fair in Germany.” Last accessed on 6 April 2017, <https://www.wsj.com/articles/new-robots-designed-to-be-more-agile-and-work-next-to-humans-1428945644>.
14. Seungbin Moon and Gurvinder S. Virk. (13 November 2009). *IEEE Xplore*. “Survey on ISO Standards for Industrial and Service Robots.” Last accessed on 6 April 2017, <http://ieeexplore.ieee.org/document/5334289/>.
15. ISO. (March 2012). *International Organization for Standardization*. “ISO 8373:2012: Robots and Robotic Devices—Vocabulary.” Last accessed on 6 April 2017, <https://www.iso.org/standard/55890.html>.
16. Matthew Tischer, Zakir Durumeric, Sam Foster, Sunny Duan, Alec Mori, Elie Bursztein, and Michael Bailey. “Users Really Do Plug in USB Drives They Find.” Last accessed on 6 April 2017, <https://zakird.com/papers/usb.pdf>.

17. Ilian Bonev. (7 April 2014). *CoRo*. "Should We Fence the Arms of Universal Robots?" Last accessed on 6 April 2017, <http://coro.etsmtl.ca/blog/?p=299>.
18. ISO. (February 2016). *International Organization for Standardization*. "ISO/TS 15066:2016: Robots and Robotic Devices—Collaborative Robots." Last accessed on 6 April 2017, <https://www.iso.org/standard/62996.html>.
19. ISO. (July 2011). *International Organization for Standardization*. "ISO 10218-2:2011: Robots and Robotic Devices—Safety Requirements for Industrial Robots—Part 2: Robot Systems and Integration." Last accessed on 6 April 2017, <https://www.iso.org/standard/41571.html>.
20. IEEE. (2017). *38th IEEE Symposium on Security and Privacy*. "Program." Last accessed on 11 April 2017, <http://www.ieee-security.org/TC/SP2017/program.html>.
21. OWASP. (14 February 2017). "IoT Security Guidance." Last accessed on 11 April 2017, [https://www.owasp.org/index.php/IoT\\_Security\\_Guidance](https://www.owasp.org/index.php/IoT_Security_Guidance).
22. Sofia Belikovetsky, Mark Yampolskiy, Jinghui Toh, and Yuval Elovici. (1 September 2016). Cornell University Library. "dr0wned—Cyberphysical Attack with Additive Manufacturing." Last accessed on 6 April 2017, <https://arxiv.org/abs/1609.00133>.
23. ISO. (February 2016). *International Organization for Standardization*. "ISO 10218-1:2011: Robots and Robotic Devices—Safety Requirements for Industrial Robots—Part 1: Robots." Last accessed on 6 April 2017, <https://www.iso.org/standard/51330.html>.
24. Keith Stouffer, Victoria Pillitteri, Suzanne Lightman, Marshall Abrams, and Adam Hahn. (May 2015). "Guide to Industrial Control Systems (ICS) Security." Last accessed on 6 April 2017, <http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-82r2.pdf>.
25. SAE International. (14 January 2016). *SAE International*. "Cybersecurity Guidebook for Cyberphysical Vehicle Systems." Last accessed on 6 April 2017, [http://standards.sae.org/j3061\\_201601/](http://standards.sae.org/j3061_201601/).
26. Dorottya Papp, Zhendong Ma, and Levente Buttyan. (3 September 2015). *IEEE Xplore*. "Embedded Systems Security: Threats, Vulnerabilities, and Attack Taxonomy." Last accessed on 6 April 2017, <http://ieeexplore.ieee.org/document/7232966/>.
27. Microsoft. (2017). *Microsoft Developer Network*. "Application Domains Overview." Last accessed on 6 April 2017, [https://msdn.microsoft.com/en-us/library/2bh4z9hs\(v=vs.90\).aspx](https://msdn.microsoft.com/en-us/library/2bh4z9hs(v=vs.90).aspx).

A large industrial robotic arm is shown in the foreground, welding a metal component. Bright sparks are flying from the welding point. The background is a blurred industrial setting with metal beams and lights.

Created by:

**TrendLabs**

The Global Technical Support and R&D Center of TREND MICRO

**TREND MICRO™**

Trend Micro Incorporated, a global cloud security leader, creates a world safe for exchanging digital information with its Internet content security and threat management solutions for businesses and consumers. A pioneer in server security with over 20 years experience, we deliver top-ranked client, server, and cloud-based security that fits our customers' and partners' needs; stops new threats faster; and protects data in physical, virtualized, and cloud environments. Powered by the Trend Micro™ Smart Protection Network™ infrastructure, our industry-leading cloud-computing security technology, products and services stop threats where they emerge, on the Internet, and are supported by 1,000+ threat intelligence experts around the globe. For additional information, visit [www.trendmicro.com](http://www.trendmicro.com).



Securing Your Journey  
to the Cloud

[www.trendmicro.com](http://www.trendmicro.com)