# Forensic Investigation of A Hacked Industrial Robot

Yanan Gong, Kam-Pui Chow, Yonghao Mai, Jun Zhang, Chun-Fai Chan

Chapter 1

# FORENSIC INVESTIGATION OF A HACKED UNIVERSAL ROBOT

Yanan Gong, Kam-Pui Chow, Yonghao Mai, Jun Zhang and Chun-Fai Chan

**Abstract** Industrial robots play a key role in Industry 4.0. This generation of robots is connected to the network and works together with humans. In addition to convenience, security risks also have increased as data, systems and people are being connected digitally. If robots are connected to the Internet, hackers will have more opportunities for attack. Hacked industrial robots not only bring economic losses to manufacturers, such as damage to production lines, but also cause injuries to workers, even result in death. In this paper, we analyzed the Universal Robots UR3 by identifying vulnerabilities of UR3, and constructed an attack model. Attacks are divided into network and physical attacks, according to the attackers' access privilege of the robot operating system. We further launched two types of attacks to the UR3 robot separately. After that, a forensic investigation of the hacked robot was conducted, which includes forensic image acquisition and detailed data analysis.

**Keywords:** Industrial Robot, Security, Forensics, Universal Robots (UR3)

## 1. Introduction

Robot technology is rapidly expanding upon different fields of the defense, industry, medical or household, and robots adapted to the needs of different fields are being deeply researched and developed. The rise of robots is rapidly changing our society, bringing endless benefits of the development of human society. Nowadays, with the trend of automation and smart factories, industrial robots play a key role in Industry 4.0. Industrial robots ease the strain on human resources, and our development cannot be separated from the assistance of industrial robots. Industrial robots have been deployed in many industries. And in order to maintain efficient and flexible manufacturing, these industrial robots

can be easily integrated into the plant's machine and system network. The International Federation of Robotics (IFR) predicts that by 2022, the number of industrial robots in the world's factories will increase to about 4 million units [11].

As the demand for robotics will grow, so will the risk associated with robotics [12]. The increasing influence of robotics in industry will inevitably lead to robot incidents. Based on existing work, many security researchers have evaluated the security status of industrial robots and discovered some vulnerabilities. For example, unauthorized access by hackers can change the behavior of robots and cause damage to the robot's environment and even harm nearby people. A series of robot-related accidents have already occurred, and some have even led to death. In 2016, at the Ajin factory in the United States, a robot on a machine assembly line suddenly resumed work and pressed a worker named Regina Elsea against the wall of the cage, causing her to be killed by the robot [2]. In most cases, forensic analysis is a key technology for preserving and recording robot-related crime evidence to diagnose and prevent robot-assisted crime. However, in comparison to robot security, forensic investigation of robots is relatively under-studied. Forensic investigation of the robots, as a branch of digital forensics, is a new field of computer forensics [1].

In this article, the authors performed an attacker model and forensically investigated over a Universal Robot (Version 3.5.1.10661). The first phase is to identify the vulnerabilities of UR3, then exploit these problems to launch attacks to the robot. The second stage is to collect evidence of the compromised robot, using FTK (Forensic Toolkit) and Autopsy for data analysis. Our work provides detailed processes of the acquisition and analysis of digital evidence from a robot. The goal of this research is to demonstrate a specific process of the forensic investigation for robots through the detailed analysis of a hacked industrial robot, and give some recommendations for the security of industrial robots as well as forensic investigations of robots.

## 2. Universal Robot

Typically, traditional industrial robots perform repetitive programming tasks that are dangerous or unsuitable for workers in manufacturing and production environments. Therefore, they operate in isolation from humans and other valuable machines [3]. Today, vendors are introducing various collaborative robots ("cobot") which are designed to work together with humans in a common working space [21]. To facilitate programming and maintenance of robots, this generation of robots com-

plies with safety standards, and can be connected to computer networks. Cobots usually include multiple mechanical actuators, controllers, sensors, and human interaction devices [10]. Universal Robots is a Danish manufacturer of smaller flexible industrial collaborative robot arms, based in Odense, Denmark [22]. The UR3 collaborative robot is a smaller collaborative table-top robot, perfect for light assembly tasks and automated workbench scenarios. The compact table-top cobot has 360-degree rotation on all wrist joints, and infinite rotation on the end joint. These unique features make the UR3 cobot the most flexible, lightweight collaborative table-top robot to work side-by-side with employees in the market today. UR3 robot applications span manufacturing industries from medical devices to circuit boards and electronic components [9].

The overall structure of the UR3 collaborative robot can be seen from Figure 1. The version number of the studied robot is 3.5.1.10661. The components of the UR3 cobot are as follows:

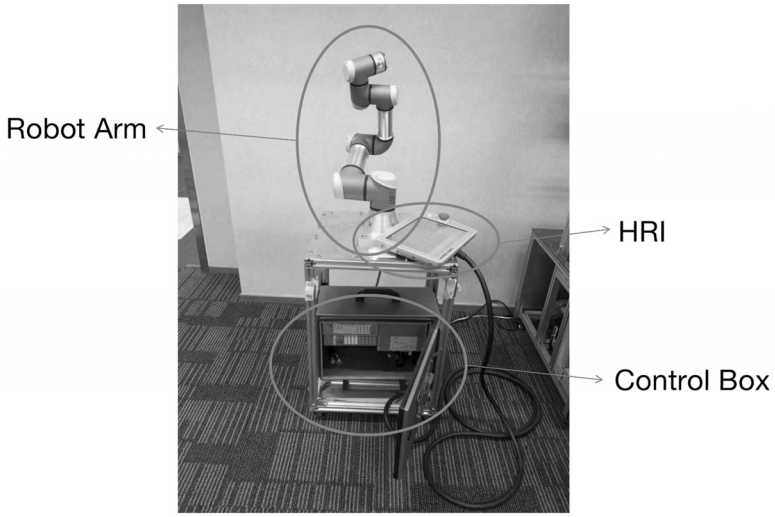



*Figure 1.* Universal Robot (Version 3.5.1.10661)

— Robot Arm: The arm is composed of extruded aluminum tubes and joints, terminated by an end effector such as pliers and laser beam welder that interacts with the environment. It is a multi-purpose robot that can be automatically controlled on three or more axes, and move flexibly according to the instructions [17].

— Control Box: It is a complex device and enclosed in one chassis. The control box mainly contains the physical electrical Input/Output that connects the robot arm, the Teach Pendant and

any peripherals [18]. And it also includes computer systems and multiple interconnected subsystems. It can supervise the activities of the robot and is a critical component. The user must turn on the control box to be able to power on the robot arm.

— HRI (Human-Robot Interface): It is a touch sensitive screen connected to the control box via a wired connection. This handheld device called teach pendant can monitor the status of the robot and program it. PolyScope is the Graphical User Interface (GUI) on the Teach Pendant that operates the robot arm, control box and executes programs [17].

The UR3 cobot's operating system is based on a Linux platform. URControl is the low-level robot controller running on the Mini-ITX PC in the controller cabinet [19]. The controller has 8 I/O ports on the controller board and 2 I/O at the tool point, all of which can control external equipment [24]. When the PC boots up URControl starts up as a daemon (like a service), PolyScope User Interface connects as a client using a local TCP/IP connection and a process named URControl will automatically run in the robot system. The rationale of our experimental setup consisting of Universal Robot is twofold. First, most robotic systems are now supported by Linux or by a POSIX-compliant Unix variant [20]. And more and more robotic components are running embedded Linux. Second, Universal Robots is one of the industrial robotics companies that dominate the global robotics industry in 2019 according to Technavio's Industrial Robotics Research report [16].

## 3. Attacking Universal Robot

The attacker model is based on the Industry 4.0 scenario, industrial robots are connected to a dedicated subnet and isolated from other endpoints. Within the boundaries set by this scenario, the authors have considered what attackers can or cannot do, and divided them into two types of attackers, according to their access level to the robot operating system.

### 3.1 Network Attack

The detailed research of scanning the Internet to identify a number of robots which are accessible and controllable from the public Internet have been presented by Nicholas et al [7]. Industrial robots can be connected to factory local area networks or remote service facilities that may have misconfigurations or vulnerabilities even if the robots are not directly exposed on the Internet. Attackers often make use of the various

entry points to compromise computers within the factory network. The study revealing that such factory subnets and robots can be accessible from the outside in the context of Industry 4.0 was conducted by Davide et al [13]. The strategies and techniques for getting network access are excluded from the scope of this work.

This type of attacker can only communicate with the robot via a network connection. Due to password lock, many functions in the PolyScope User Interface are locked by the use of password protection, such as creating programs for the robot, modifying safety settings, etc. However, the authors discover that an attacker can bypass authentication to change the safety configuration and move the joints remotely once the attacker compromised a computer on the network that the UR3 cobot is connected. This network attack is achieved by exploiting authentication issues of the robot, and modifying safety settings to the status of no safety limits, then sending malicious movement scripts to the robot.

**Attack Implementation Step 1.** Find the robot IP Address. The robot IP address is easy to find, because the default host name can easily announce the existence of the robot [5], which can be realized in many ways.

**Attack Implementation Step 2.** Different safety configurations of UR System are saved as various installation files stored in the robot. UR robot script programming is in the form of "xxx.urp". Using Netcat command can bypass the password protection in the robotic arm and connect the Dashboard Server which is running on port 29999 on the robot's IP address. The basic information and real-time status of the UR3 cobot can be known from remote by sending specific commands to the PolyScope GUI over a TCP/IP socket (Figure 2).

1) The version number of the cobot can be obtained by issuing the command "PolyScopeVersion", the version of the cobot will be known.
2) Command "Robotmode/running" is used to check the current status of the robot.
3) The state and the path of the program loaded by the cobot can be known by sending "programState/get loaded program".
4) Command "unlock protective stop/brake release" can close the current popup of PolyScope and unlock protective stop, then release the brakes.
5) Load a specified installation file via the command "load installation xxx.installation". The "default.installation" file records the initial status of the cobot which has not any safety boundaries, speed


*Figure 2.* Commands sent to the Dashboard server

limits, etc. The safety configuration of the cobot will change to the default status after loading the default installation file.

6) In order not to attract attention of operators, sending command "close popup" to close any popup shown on PolyScope.

7) Command "power on" is to make sure the robot is on.

**Attack Implementation Step 3.** There are 3 ports 30001, 30002 and 30003 on the UR that can be used to receive raw scripts from an external device [23]. These ports can be accessed without authentication by any user on the network. After changing the safety settings to the default state, the attacker is able to keep sending random URScript commands to these ports (Figure 3).

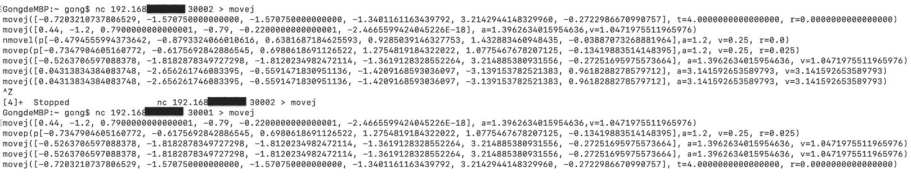



*Figure 3.* URScript commands

**Attack Result.** Unauthorized access by the attacker will change the behavior of the UR3 cobot. Once a worker clicks the "start" button on PolyScope, the robot arm will suddenly move with a custom speed and acceleration. In this scenario, the safety configuration has changed to default initial status, which has no safety limits, and random movement of the robot arm can be carried out. The uncontrolled robot could damage the robot's environment, or cause injuries to people nearby.

## 3.2 Physical Attack

Since industrial robots interact with company's workers, physical access is acceptable and expected. A physical attack is possible when an attacker can access the robot's hardware or mechanical devices to change its behavior or set up persistent threats. The human operator needs to monitor, start, and stop the operations of the robot, and the programmer is responsible for writing task programs. Usually operators and programmers control the movement of the robot through PolyScope. The robot programs can be written by recording the action sequences through the joystick, or programming through simulators or text editors. The simplest and most common situation is that a physical attacker is the robot operator/programmer (insider) who uses HRI to program it manually. A slightly more sophisticated circumstance is that attackers plug an external device into the robot's exposed connectivity ports [13]. The UR3 robot has several exposed accessible ports, there is one open port on HRI panel. The control box also has two available USB ports, and the box can be easily opened with a key. Mouse and keyboards are also supported on Universal Robots Controller's USB interface [14]. Therefore, configurations and robot actions may be modified over USB ports. For instance, a special USB device acting as a keyboard can change settings on the robot or manipulate actions. In order to facilitate control and maintenance, industrial robots often can be remotely controlled by computers. Universal Robots allow SSH access, and there are no enforcement measures to force users to change the default password. The default username "root" is fixed and default password can be easily searched on the Internet.

This type of attack is mainly deployed by operators/programmers (insider). Assuming a physical attacker wants to stop a working robot to create some unnecessary troubles. The easiest way is to kill the URControl process, then the URControl controller cannot run normally. And to avoid suspicion, the attacker can also let the robot back to normal again after stopping. Actually a physical attacker can do a lot of things to the robot. The author chooses some typical operations, such as in-

serting an external device, deleting files and so on. Put all of them into one physical attack scenario. This is the reason why making this attack assumption.

**Attack Implementation Step 1.** A physical attacker inserts a USB device to an exposed USB port of the UR3 cobot.

**Attack Implementation Step 2.** Using one computer to connect the robot via the free SSH tool SmarTTY. Uploading a python file called "restartp.py" from this computer which can start a new URControl process after a preset time.

**Attack Implementation Step 3.** Execute the "restartp.py" file and kill the running URControl process. In the robot system, a directory called programs is used to store all safety configuration files and URScript programs. The attacker downloads some files from this directory to the plugged USB. After the python file runs to completion, delete it from the robot.

**Attack Result.** The log screen in the PolyScope User Interface shows that the URControl controller was disconnected, the robot stopped working. After a while, the controller was connected automatically and the robot was back to normal. Compare the PID (process identifier) of URControl process before and after the execution of the "restartp.py" file. The PID value of URControl process has changed.

Actually, the authors learn most of the details of the UR3 cobot through free and available technical documentation on the Internet. And vendors' websites usually provide controller software available for download. Similarly, relying on these publicly available information, attackers can even perform reverse engineering and discover more vulnerabilities without exploiting any insider technical knowledge. Even without being able to test their attacks to a real industrial robot, they can use simulators (URSim) provided by vendors, which allows attackers to run simulated versions of the software running on the robot's controller, and prepare an attack payload.

## 4. Forensic Investigation

Digital forensics is the process to obtain, preserve and document evidence. This section presents the method for acquiring the robot image and the analysis of the data after the attacking section. In general, to ensure the safety of robots, robot systems sacrifice functionality and do not have the full functions of Linux. For instance, commands like "make" or "tcpdump" are not supported by the UR3 cobot. Therefore, in order to maintain the originality and integrity of the UR3 cobot as much as possible, extra commands or applications will not be installed

during the process of the forensic investigation. And different forensic tools will be used.

## 4.1 Image Acquisition

The hacked robot is the direct sources of evidence. First of all, use "fdisk" command to ensure the actual usage of the hard disk in the UR3 cobot. All data is stored in the first drive "/dev/sda". The total size of this drive is 2059403264 bytes, and the system is Linux. There are various tools can be used for making forensic images. The Data Dump(dd) command is available on all Linux distribution, and is able to read and write to an unmounted drive because it is not bound by a logical file system. The "dd" command will capture all files, unallocated data and slack space [6]. Remotely connect the robot with a computer, type command "dd if = /dev/sda of = /media/USB/ursda.img conv=noerror, sync" in a terminal window. This is a byte-to-byte disk cloning. Where "/dev/sda" is the drive we are acquiring the image of and "capture.img" is the chosen name and extension of the acquisition file. Using "conv=noerror, sync" switch ensures "dd" will not skip over any sectors or blocks on the source device. The second part of the switch, "sync" provides the zero padding and also ensures that the sectors on the target device are aligned with those from the source device, thus ensuring an accurate replication of the original media [8]. The robot does not support remote file transfer. There is not enough space to store the image file because of the limited storage of the robotic system, the image file will be stored in a plugged USB device. Once the process is finished, calculate MD5 hash value of the image to maintain the integrity of the data collected.

## 4.2 Image Analysis

Image analysis is the process of discovering abnormal traces, saving data and analyzing data in compliance with a forensically sound manner. The robot image is the Linux EXT3 file system. Forensic Toolkit (FTK) and Autopsy are used. Both FTK and Autopsy are commonly used forensic tools with powerful forensic functions. FTK has an intuitive GUI, and Autopsy can be used through the Autopsy Forensic Browser. The version number of FTK is 4.2.0.13, and it runs on Windows 10. Autopsy is in Kali Linux, and the version is 2.24.

### 4.2.1 Network Attack Analysis

The log screen in the PolyScope User Interface provides useful information about the real-time status of the robot and URControl controller

(Figure 4). For instance, information about controller temperature, consumption, power supply output and joint status is available [25]. However, these log entries will be cleaned from the screen once restarting the UR3 cobot. All log entries appeared will also be recorded in the robot, and the corresponding log file is "log_history.txt" which is located in the "root" directory in the robot system. The only difference is that the log information in the PolyScope User Interface is more human-readable than the file "log_history.txt".

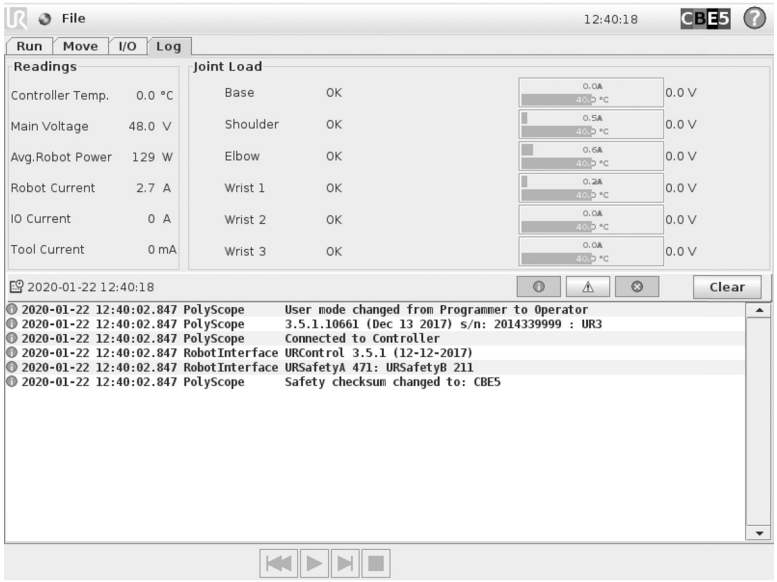



*Figure 4.* PolyScope's log screen

In the UR3 cobot, different safety configurations are recorded in the safety configuration files named "safety.conf". The UR3 cobot uses the CRC (STM-32) algorithm to generate a CRC checksum to ensure the integrity of the safety configuration file. A safety parameter is found in the "safety.conf" file. Therefore, each safety configuration file has the corresponding CRC checksum and safety parameter. The safety parameter is in the robot system. The first four digits of the CRC checksum value are called the safety checksum value and displayed in the upper right corner of the PolyScope User Interface.

Checking the "log_history.txt" file with FTK, it was found that the safety checksum recorded was initially CBE5 when the robot was just started on January 17, 2020. After a while, the safety checksum became to the number CCCC at 11:40:54. After a few minutes, some programs, "movej" and "movep" are executed successfully (Figure 5).

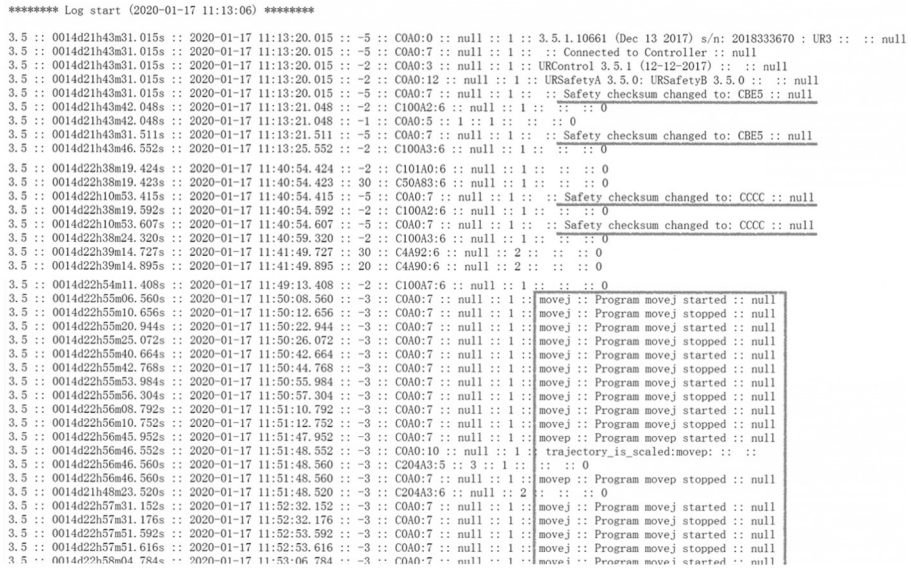



*Figure 5.* The content of "log_history.txt"

Under the "/root/.urcontrol" directory, the file "safety.conf" saves the current safety settings of the robot, and the safety parameter recorded is the initial default value 1216962 which corresponds to the default safety checksum CCCC (Figure 6). Therefore, it is surely that the safety settings have been changed to initial status at 11:40:54 on January 17, 2020.


*Figure 6.* The safety parameter of "safety.conf"

The "programs" directory saves all installation files and URScript programs. After checking the image, there is no program called "movej" or "movep" in the robot, nor are they in the deleted files. Under normal circumstances, while running an existed program via PolyScope User

Interface, the last save time of this loaded program will also be displayed in the log file. Therefore, no program stored in the robot was executed by the attacker. Figure 7 shows the log records of the robot program "goodpunch". When the program starts, the program's last saved time shows in brackets on the right.

```
3.5 :: 0014d21h15m38.559s :: 2020-01-16 18:25:23.559 :: -5 :: C0A0:7 :: null :: 1 ::  :: Program goodpunch starting... (Last saved: 2018-08-27 19:47:36) :: null
3.5 :: 0014d22h44m21.040s :: 2020-01-16 18:25:23.040 :: -3 :: C0A0:7 :: null :: 1 :: goodpunch :: Program goodpunch started :: null
3.5 :: 0014d22h44m37.120s :: 2020-01-16 18:25:39.120 :: -3 :: C0A0:7 :: null :: 1 :: goodpunch :: Program goodpunch stopped :: null
```

*Figure 7.* Running records of "goodpunch" in "log_history.txt"

The folder "flightreports" under the "root" directory includes compressed files at five different times. When safety settings are modified, the robot will automatically save compressed files like these. And each of them consists of the key information and the status of the robot at that time. For instance, the data provided are about controller state, safety settings, loaded program and log. In the "recording20200117_11_51_54.zip" file, there is a file called "summary.log" which records detailed information about OS (operating system), memory, the thread dump, etc. From the "PolyScope Probe" part of this file, it can be known that the name of the installation file at that time was "default" and the loaded program was still "a.urp". This means programs "movej" and "movep" were not executed in the format of regular robot programs, and the safety settings have been changed to initial status.

There is also a file called "URControl.log.0" in this compressed file (Figure 8). One parse error of TCPReceiver is found in this file, which contains the suspicious URScript "nmovel" commands, such as the URScript "nmovel(p [ -0.4794555994373642, -0.8793324066010616, 0.6381687184625593, 0.9285039146327753, 1.432883460948435, -0.03887073268881964], a=1.2, v=0.25, r=0.0)". URScript commands like these can only be sent to specific ports 30001/30002/30003 and executed. And before the parse error of TCPReceiver, multiple "movej" programs were executed. After this parse error, a program "movep" was also found. This indicates that URScript commands have been sent to these ports and were successfully executed. Therefore, the programs "movej" and "movep" were in the form of URScript commands and received by the robot via open socket ports. From this zip file, the presence of the "default.installation" file represents the default installation file was loaded by the robot at that time. The safety parameter in the "safety.conf" file of the compressed file is 1216962, which corresponds to the default safety checksum value CCCC. This is capable of proving again the safety settings have been changed to initial status.

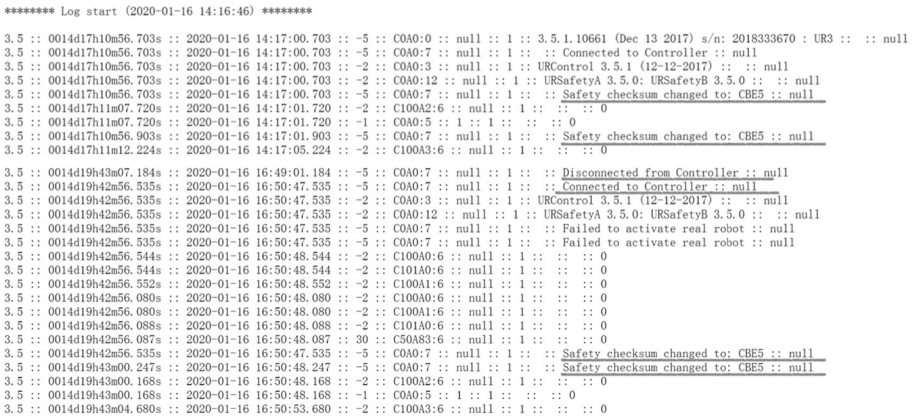

*Figure 8.* The content of "URControl.log.0"

**Analysis Conclusion.** In summary, it is certain that the safety configuration was changed to the initial default status at 11:40:54 on January 17, 2020. And after that random movement commands are able to be executed successfully without safety limits. URScript "movej" and "movep" commands have been received through ports 30001/30002/30003 and executed by the robot.

### 4.2.2 Physical Attack Analysis

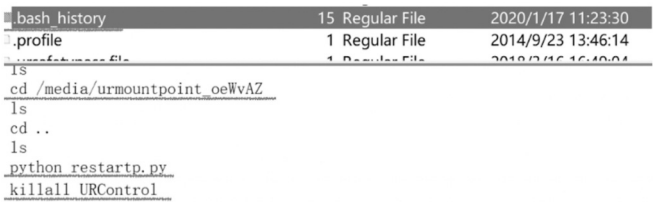
Figure 9 shows the content of the "log_history.txt" file, it can be seen that the URControl controller was disconnected at 16:49:01, and after a while, the controller was connected automatically and the robot was back to normal. The recorded value of the safety checksum in this log file was CBE5. Checking all other safety settings related things, such as the "safety.conf" file, the safety parameter and files under the folder "flightreports". It shows that the value of safety checksum was same with the one when the UR3 cobot just started on January 16, 2020, the loaded installation file and the loaded robot program at that time were both not changed.

Under the "root" directory, by checking the file ".bash_history", a device shown as "urmountpoint_oeWvAZ" was once mounted to the robot. A python file called "restartp.py" that does not initially belong to the robot was executed, then the running URControl process was killed (Figure 10). Using the tool Autopsy, the file "restartp.py" was found to have been deleted from the "media" directory in the robot system (Figure 11).



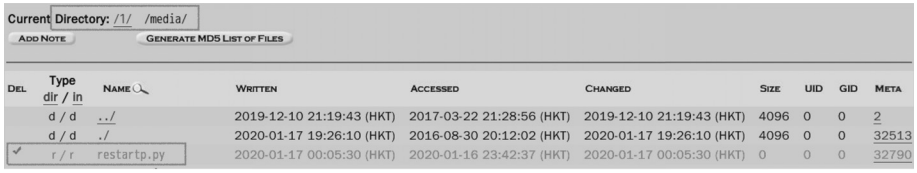
```
******** Log start (2020-01-16 14:16:46) ********

3.5 :: 0014d17h10m56.703s :: 2020-01-16 14:17:00.703 :: -5 :: C0A0:0 :: null :: 1 :: 3.5.1.10661 (Dec 13 2017) s/n: 2018333670 : UR3 ::  :: null
3.5 :: 0014d17h10m56.703s :: 2020-01-16 14:17:00.703 :: -5 :: C0A0:7 :: null :: 1 ::  :: Connected to Controller :: null
3.5 :: 0014d17h10m56.703s :: 2020-01-16 14:17:00.703 :: -2 :: C0A0:3 :: null :: 1 :: URControl 3.5.1 (12-12-2017) ::  :: null
3.5 :: 0014d17h10m56.703s :: 2020-01-16 14:17:00.703 :: -2 :: C0A0:12 :: null :: 1 :: URSafetyA 3.5.0: URSafetyB 3.5.0 ::  :: null
3.5 :: 0014d17h10m56.703s :: 2020-01-16 14:17:00.703 :: -5 :: C0A0:7 :: null :: 1 ::  :: Safety checksum changed to: CBE5 :: null
3.5 :: 0014d17h11m07.720s :: 2020-01-16 14:17:01.720 :: -2 :: C100A2:6 :: null :: 1 ::  ::  :: 0
3.5 :: 0014d17h11m07.720s :: 2020-01-16 14:17:01.720 :: -1 :: C0A0:5 :: 1 :: 1 ::  ::  :: 0
3.5 :: 0014d17h10m56.903s :: 2020-01-16 14:17:01.903 :: -5 :: C0A0:7 :: null :: 1 ::  :: Safety checksum changed to: CBE5 :: null
3.5 :: 0014d17h11m12.224s :: 2020-01-16 14:17:05.224 :: -2 :: C100A3:6 :: null :: 1 ::  ::  :: 0

3.5 :: 0014d19h43m07.184s :: 2020-01-16 16:49:01.184 :: -5 :: C0A0:7 :: null :: 1 ::  :: Disconnected from Controller :: null
3.5 :: 0014d19h42m56.535s :: 2020-01-16 16:50:47.535 :: -5 :: C0A0:7 :: null :: 1 ::  :: Connected to Controller :: null
3.5 :: 0014d19h42m56.535s :: 2020-01-16 16:50:47.535 :: -2 :: C0A0:3 :: null :: 1 :: URControl 3.5.1 (12-12-2017) ::  :: null
3.5 :: 0014d19h42m56.535s :: 2020-01-16 16:50:47.535 :: -2 :: C0A0:12 :: null :: 1 :: URSafetyA 3.5.0: URSafetyB 3.5.0 ::  :: null
3.5 :: 0014d19h42m56.535s :: 2020-01-16 16:50:47.535 :: -5 :: C0A0:7 :: null :: 1 ::  :: Failed to activate real robot :: null
3.5 :: 0014d19h42m56.535s :: 2020-01-16 16:50:47.535 :: -5 :: C0A0:7 :: null :: 1 ::  :: Failed to activate real robot :: null
3.5 :: 0014d19h42m56.544s :: 2020-01-16 16:50:48.544 :: -2 :: C100A0:6 :: null :: 1 ::  ::  :: 0
3.5 :: 0014d19h42m56.544s :: 2020-01-16 16:50:48.544 :: -2 :: C101A0:6 :: null :: 1 ::  ::  :: 0
3.5 :: 0014d19h42m56.552s :: 2020-01-16 16:50:48.552 :: -2 :: C100A1:6 :: null :: 1 ::  ::  :: 0
3.5 :: 0014d19h42m56.080s :: 2020-01-16 16:50:48.080 :: -2 :: C100A0:6 :: null :: 1 ::  ::  :: 0
3.5 :: 0014d19h42m56.080s :: 2020-01-16 16:50:48.080 :: -2 :: C100A1:6 :: null :: 1 ::  ::  :: 0
3.5 :: 0014d19h42m56.088s :: 2020-01-16 16:50:48.088 :: -2 :: C101A0:6 :: null :: 1 ::  ::  :: 0
3.5 :: 0014d19h42m56.087s :: 2020-01-16 16:50:48.087 :: 30 :: C5A83:6 :: null :: 1 ::  ::  :: 0
3.5 :: 0014d19h42m56.535s :: 2020-01-16 16:50:47.535 :: -5 :: C0A0:7 :: null :: 1 ::  :: Safety checksum changed to: CBE5 :: null
3.5 :: 0014d19h43m00.247s :: 2020-01-16 16:50:48.247 :: -5 :: C0A0:7 :: null :: 1 ::  :: Safety checksum changed to: CBE5 :: null
3.5 :: 0014d19h43m00.168s :: 2020-01-16 16:50:48.168 :: -2 :: C100A2:6 :: null :: 1 ::  ::  :: 0
3.5 :: 0014d19h43m00.168s :: 2020-01-16 16:50:48.168 :: -1 :: C0A0:5 :: 1 :: 1 ::  ::  :: 0
3.5 :: 0014d19h43m04.680s :: 2020-01-16 16:50:53.680 :: -2 :: C100A3:6 :: null :: 1 ::  ::  :: 0
```

*Figure 9.* Records of "log_history.txt"



```
.bash_history          15  Regular File    2020/1/17 11:23:30
.profile                1  Regular File    2014/9/23 13:46:14
ls
cd /media/urmountpoint_oeWvAZ
ls
cd ..
ls
python restartp.py
killall URControl
```

*Figure 10.* Records of ".bash_history"



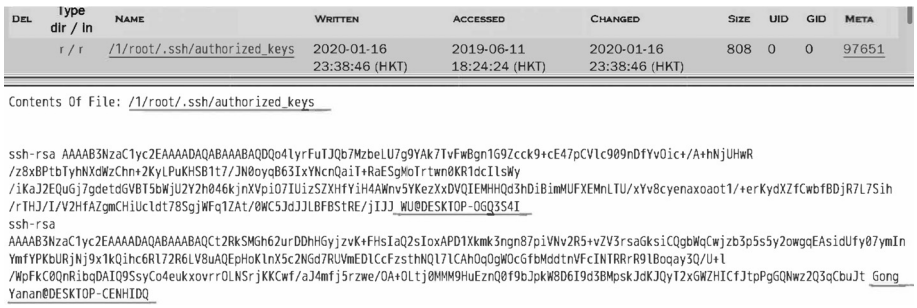
Current Directory: /1/ /media/

ADD NOTE | GENERATE MD5 LIST OF FILES

| DEL | Type dir / in | NAME | WRITTEN | ACCESSED | CHANGED | SIZE | UID | GID | META |
|---|---|---|---|---|---|---|---|---|---|
| | d / d | ../ | 2019-12-10 21:19:43 (HKT) | 2017-03-22 21:28:56 (HKT) | 2019-12-10 21:19:43 (HKT) | 4096 | 0 | 0 | 2 |
| | d / d | ./ | 2020-01-17 19:26:10 (HKT) | 2016-08-30 20:12:02 (HKT) | 2020-01-17 19:26:10 (HKT) | 4096 | 0 | 0 | 32513 |
| ✓ | r / r | restartp.py | 2020-01-17 00:05:30 (HKT) | 2020-01-16 23:42:37 (HKT) | 2020-01-17 00:05:30 (HKT) | 0 | 0 | 0 | 32790 |

*Figure 11.* Deleted file "restartp.py"

In SSH, the authorization keys configured for each user are usually recorded in a file named authorized_keys. Authorized keys specify which users are allowed to log into a server using public key authentication in SSH [15]. After searching, there is a file named "authorized_keys" in the ".ssh" folder under the "root" directory. Two users "WU" and "Gong Yanan" once connected to the robot (Figure 12). And "Gong Yanan" is the known computer used to remotely control the robot. "WU" is an unknown user.



| DEL | Type dir / in | NAME | WRITTEN | ACCESSED | CHANGED | SIZE | UID | GID | META |
|---|---|---|---|---|---|---|---|---|---|
| | r / r | /1/root/.ssh/authorized_keys | 2020-01-16 23:38:46 (HKT) | 2019-06-11 18:24:24 (HKT) | 2020-01-16 23:38:46 (HKT) | 808 | 0 | 0 | 97651 |

Contents Of File: /1/root/.ssh/authorized_keys

ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQDQo4lyrFuTJQb7MzbeLU7g9YAk7TvFwBgn1G9Zcck9+cE47pCVlc909nDfYvOic+/A+hNjUHwR
/z8xBPtbTyhNXdWzChn+2KyLPuKHSB1t7/JN8oyqB63IxYNcnQaiT+RaESgMoTrtwn0KR1dcIlsWy
/iKaJ2EQuGj7gdetdGVBT5bWjU2Y2h046kjnXVpiO7IUizSZXHfYiH4AWnv5YKezXxDVQIEMHHQd3hDiBimMUFXEMnLTU/xYv8cyenaxoaot1/+erKydXZfCwbfBDjR7L7Sih
/rTHJ/I/V2HfAZgmCHiUcldt78SgjWFq1ZAt/0WC5JdJJLBFBStRE/jIJJ WU@DESKTOP-OGQ3S4I
ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAABAQCt2RkSMGh62urDDhHGyjzvK+FHsIaQ2sIoxAPD1Xkmk3ngn87piVNv2R5+vZV3rsaGksiCQgbWqCwjzb3p5s5y2owgqEAsidUfy07ymIn
YmfYPKbURjNj9x1kQihc6Rl72R6LV8uAQEpHoKlnX5c2NGd7RUVmEDlCcFzsthNQl7lCAhOqOgWOcGfbMddtnVFcINTRRrR9lBoqay3Q/U+l
/WpFkC0QnRibqDAIQ9SsyCo4eukxovrrOLNSrjKKCwf/aJ4mfj5rzwe/OA+OLtj0MM9HuEznQ0f9bJpkW8D6I9d3BMpskJdKJQyT2xGWZHICfJtpPgGQNwz2Q3qCbuJt Gong
Yanan@DESKTOP-CENHIDQ

*Figure 12.* The content of "authorized_keys"

**Analysis Conclusion.** The user "WU" connected to the robot without authorization via SSH, and a device shown as "urmount-point_oeWvAZ" was inserted to the robot. The python file "restartp.py" was once uploaded to the "media" directory in the robot system and deleted by the attacker after execution. The URControl process was killed by the attacker at 16:49:01 on January 16, 2020, but the controller became normal automatically at 16:50:47, so it may be due to the execution of the suspicious file "restartp.py" that leads to the restored connection of URControl controller.

While finishing the forensic analysis, recompute MD5 hash value of the image with FTK and Autopsy separately. And compare them with the original hash value. After comparison, all hashes are exactly the same. The purpose of this is to prove that the image has not been broken or modified during the forensic process. Ensure the effectiveness and reliability of the forensic investigation.

## 5. Discussion

In this section, the authors will discuss and summarize the work done. In addition, some recommendations have come up for the robot security as well as robot forensics.

### 5.1 Network Attack

Unique log history files generated by robots are crucial for the forensic investigation. For instance, the "log_history.txt" file from the UR3 cobot. But it also contains various event codes and other symbols. It would be better if the log entries are more human-readable, like the format shown in the PolyScope User Interface. And just relying on this log file is not enough. It is also necessary to check other essen-

tial information about the robot status at that time, such as safety settings. The information about the robot system is also very important for forensics. For the network attack, compressed files under the path "/root/flightreports" are very useful for analyzing attacks related to modifying the safety configuration. Because these files consist of the key information and the status at that time of the robot. When an attacker launches the attack to the robot through the open ports, the commands sent over a TCP/IP socket are hard to be obtained in the forensic process. Firstly, the commands used to get feedback from the Dashboard server cannot be gotten while doing the forensic analysis. To put it simply, all commands sent to the port 29999 cannot be recorded by the robot. Even using SSH to connect to the robot in advance, then set up a connection to the Dashboard server, the bash history file is only able to save the first command which is used for establishing TCP connection with the port 29999. Furthermore, there are three interfaces capable of accepting raw URScript commands from an external device. The transmission of URScript commands via ports 30001/30002/30003 is certain, but on which port these commands are sent, and the specific content of the successfully executed URScript is unknown.

## 5.2 Physical Attack

For the physical attack, using SSH tools which support the SCP protocol such as SmarTTY, files can be downloaded or uploaded silently between the robot and the computer. These tools provide functionality that can be done without the need to type any commands, just with mouse clicks. For instance, uploading a directory. There will be no records about these operations, although the bash history file can preserve all instructions entered during the SSH connection. While conducting the physical attack, SmarTTY was used to establish the SSH connection. The python file "restartp.py" was uploaded from the computer and deleted from the robot after execution. In the process of forensic analysis, using Autopsy, we can see that this file "restartp.py" was deleted from the robot. However, in the ".bash_history" file, there are no records about uploading or deleting this python file. This file has also been tested to see if it can record commands sent to those open ports. Using SSH to connect to the robot remotely in advance, then the Netcat command is adopted to build connections with ports respectively. For Dashboard server, the ".bash_history" file only saves the Netcat command sent to the port 29999, and there is no command about getting feedback from this server. For interfaces, there is not even the first "nc" command which is used for establishing a TCP connection

in this file, let alone URScript commands. Only when the "echo" and "nc" commands are both used to transmit URScript commands to the robot, the file has the corresponding command records.

## 5.3 Recommendation

Industrial robots have a very long lifetime [13], it is hard to patch all vulnerabilities in complex software, but multiple measures can be adopted to improve the security of robots.

**Authentication and Authorization.** It's critical to identify the users authorized to access robots [4]. Weak authentication means attackers may easily use certain services without a valid username and password and control the robots. Vendors must adopt essential methods, like multiple password lock, to make sure only valid users have access to robots' functionality and services. And implementing code-signing mechanisms with strong authentication can control the execution of custom code. The operator must be able to sign code for the prescribed robots (not arbitrary robots). In this way, attackers cannot execute arbitrary code, although they upload arbitrary files to the robot.

**Security Audits.** Before robots are put into use, comprehensive security assessment tests should be performed. For instance, check all kinds of servers and open ports and test parts, as far as possible to lower the safety hidden trouble. Security audits can also be handed over to the professional robot safety evaluation agency to conduct the safety risk analysis. Robot manufacturers can develop the standard safety operation procedures for robots. The factories should provide the safety operation training of robots to restrict the behavior of operators and maintainers. Enhance safety awareness and ensure the safe operation of robots.

**Standards.** A complete framework of the forensic investigation for robots is necessary. Robot forensics can discover and record robot-related crime evidence. Setting standards and methodologies can facilitate the development of robot forensics and make sure the investigation process is forensically sound.

**Forensic tools.** In the process of forensics, it is recommended to comprehensively use multiple tools to identify the evidence and analyze the data. While conducting forensics, it is important to ensure there is no modification to the robot's operating system. Many forensic tools may not be suitable for forensic investigations of robots because of the special nature of the robot operating system. In order to ensure the security, the robot generally sacrifices functions and does not have all the functions of Linux. For instance, Linux Memory Extractor is often

used for Linux memory acquisition, but the command "make" does not be installed in the UR3 cobot. Network forensics is also capable of obtaining digital artifacts from a networked environment. However, there is also no "tcpdump" command in the robot. And some general forensic tools require to be downloaded and run in the robot for memory acquisition. Note not to destroy the integrity and originality of the robotic system is of great importance. Therefore, dedicated forensic tools can be developed for forensic investigations of robots.

Robot technology is increasingly being used in different areas such as the defense, space, medical or household. These fields often involve sensitive information and complex tasks, so robot security is of paramount importance. The current state of robotics is susceptible to many security risks, and it is hard to patch all vulnerabilities of all kinds of robots. This can lead to a variety of criminal acts, whether in domestic relations, competitor relations or cyber warfare, that these vulnerabilities make it possible. Therefore, robot forensics is equally important. In dealing with robot-related crimes, robot forensics plays a key role in investigating and recording criminal evidence, diagnosing and preventing related crimes. Robot forensics should be paid more attention as well as robot security.

## 6. Conclusions

In an attempt to provide a specific process of forensic investigations of robots and make some suggestions for the security of industrial robots, in this paper, the authors performed studies of the Universal Robot (UR3). Firstly, identify the vulnerabilities of the UR3 cobot and set up an attack model. Attacks are divided into network attack and physical attack, according to the attackers' access level to the robot operating system. Perform these two types of attacks to the UR3 cobot separately. Then conduct a forensic investigation of the hacked robot. The forensic process consists of image acquisition and image analysis.

With the development of Industry 4.0, the number of industrial robots has grown rapidly. Hacked industrial robots are capable of leading to serious consequences. Not only will it have a huge financial impact due to stopping production, it may also cause injuries to people who work closely with it. On the one hand, it is essential to improve the security status of robots. Access authentication to robots needs to be strengthened to prevent any unauthorized access. Manufacturers ought to conduct comprehensive security tests on robots to reduce the existence of safety vulnerabilities. To avoid the execution of arbitrary code, code-signing mechanisms can be adopted. On the other hand, the forensic investigation of robots is a new area of computer forensics field. There

should be a systematic standard to regulate and guide forensic investigations of robots. Due to the particularity of robotic systems, it is also necessary to develop forensic tools specifically for forensic investigations of robots.

Robots have been applied in many fields until now. Vulnerabilities of robots are able to result in the kinds of potential robot-related crimes. And these robot- related criminal acts may invoke information leakage, economic loss and other critical consequences. Therefore, more research on robot forensics are worth being carried out. Robot forensics as well as the robot security are both of paramount importance.

## Acknowledgement

The authors would like to thank HKSAR Logistics and Supply Chain Multi-Tech R&D Centre (LSCM) to allow them to conduct their research on their UR3 robot. The views expressed in this paper are those of the authors, and do not reflect the official policy or position of the HKSAR Logistics and Supply Chain Multi-Tech R&D Centre (LSCM).

## References

[1] I. Abeykoon and X. Feng, A forensic investigation of the robot operating system, *Proceedings of the IEEE International Conference on Internet of Things, Green Computing, Communications, Cyber, Physical and Social Computing, and Smart Data*, pp. 851–857, 2017.

[2] Achieng, 15 most savage deaths caused by robots, *TheRichest*, July 27 2017.

[3] L. Apa, Exploiting Industrial Collaborative Robots, *IOActive*, Seattle, Washington, USA, (`www.ioactive.com/exploiting-industrial-collaborative-robots/`), August 22, 2017.

[4] C. Cerrudo and L. Apa, Hacking robots before Skynet, IOActive, Seattle, Washington, USA (`ioactive.com/pdfs/Hacking-Robots-Before-Skynet.pdf`), 2017.

[5] C. Cerrudo and L. Apa, Hacking robots before Skynet: Technical appendix, IOActive, Seattle, Washington, USA (`ioactive.com/pdfs/Hacking-Robots-Before-Skynet-Technical-Appendix.pdf`), 2017.

[6] V. Codispot, Data Dump(dd) to Create a Forensic Image with Linux, Threat Analysis, (`vcodispot.com/data-dump-dd-create-forensic-image-linux`), 2017.

[7] N. DeMarinis, S. Tellex, V. Kemerlis, G. Konidaris and R. Fonseca, Scanning the Internet for ROS: A view of security in robotics

research, *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 8514–8521, 2019.

[8] Forensic Focus, Linux 'dd' basics, (www.forensicfocus.com/articles/linux-dd-basics/), 2011.

[9] Innovative Total Solutions, UR3 Robot, Midleton, Ireland (itsl.ie/shop/universal-robots/ur3).

[10] F. Maggi, D. Quarta, M. Pogliani, M. Polino, A. Zanchettin and S. Zanero, Rogue robots: Testing the limits of an industrial robot's security, *Trend Micro* (documents.trendmicro.com/assets/wp/wp-industrial-robot-security.pdf), May 2, 2017.

[11] C. Müller, IFR World Robotics Presentation, International Federation of Robotics, Shanghai, China (www.ifr.org/downloads/press2018/IFR%20World%20Robotics%20Presentation%20-%2018%20Sept%202019.pdf), 2019.

[12] I. Priyadarshini, Cyber security risks in robotics, in *Cyber Security and Threats: Concepts, Methodologies, Tools, and Applications*, Information Resources Management Association (Ed.), IGI Global, Hershey, Pennsylvania, USA, pp. 1235–1250, 2018.

[13] D. Quarta, M. Pogliani, M. Polino, F. Maggi, A. Zanchettin and S. Zanero, An experimental security analysis of an industrial robot controller, *Proceedings of the IEEE Symposium on Security and Privacy*, pp. 268–285, 2017.

[14] Z. Skovsgaard, Training manual: Hint and tips Version 1.4, Zacobria, Singapore (www.zacobria.com/universal_robots_zacobria_hints_and_tips_manual_1_4_3.htm), 2012.

[15] SSH.com, Authorized key (www.ssh.com/ssh/authorized-key).

[16] Technavio, Top 21 industrial robotics companies in the world 2019 (blog.technavio.com/blog/top-21-companies-in-the-industrial-robotics-market), February 5, 2019.

[17] Universal Robots, User Manual: UR3/CB3 Version 3.5.5, Odense, Denmark (s3-eu-west-1.amazonaws.com/ur-support-site/32340/UR3_User_Manual_en_Global-3.5.5.pdf), 2018.

[18] Universal Robots, Universal Robots e-Series User Manual: UR10e Version 5.2, Odense, Denmark (https://s3-eu-west-1.amazonaws.com/ur-support-site/46114/UR10e_User_Manual_en_US.pdf), 2018.

[19] Universal Robots, The URScript Programming Language: Version 1.2, Odense, Denmark (s3-eu-west-1.amazonaws.com/ur-support-site/18407/UR-Scriptmanual_en_1.2.pdf).

[20] V. Vilches, L. Kirschgens, E. Gil-Uriarte, A. Hernández and B. Dieber, Volatile memory forensics for the robot operating system, arXiv:1812.09492 (arxiv.org/abs/1812.09492), 2018.

[21] Wikipedia Contributors, Cobot (en.wikipedia.org/wiki/Cobot), 2020.

[22] Wikipedia Contributors, Universal Robots (en.wikipedia.org/wiki/Universal_Robots), 2020.

[23] Zacobria, Introduction to universal-robots script programming, Singapore (https://www.zacobria.com/universal-robots-knowledge-base-tech-support-forum-hints-tips/universal-robots-script-programming).

[24] Zacobria, Linux OS on industrial robot (www.zacobria.com).

[25] Zacobria, Log Window Tab, Singapore (www.zacobria.com/universal-robots-zacobria-forum-hints-tips-how-to/log-%20window-tab), 2017.