A cooperation between Alias Robotics and Kaspersky

Securing robot endpoints in Operation Technology (OT) enviroments

Extending KICS with the Robot Immune System (RIS)

ALIAS ROBOTICS
Robot Cybersecurity

kaspersky

Publisher

Alias Robotics

ABSTRACT

Are robot endpoints secure in industrial environments? Current industrial security solutions monitor network interaction and detect unexpected traffic and cyber-threats. Robot-specific protocols and tools are commonly unknown to these solutions and moreover, detection is far from what robots require since most are fully exposed to simple attacks according to recent research. Given the physical interaction these robots have with humans and our environments, attackers targeting robots should be blocked before interacting with their controllers, preventing safety hazards. Cybersecurity thereby becomes a strong pre-requirement to safety in robotics.

Through a cooperation between Alias Robotics and Kaspersky, we launched a research effort to shed some light on the status of security for robot endpoints in OT environments. Our research showed evidence that simple attacks were feasible and that specialized security controls are necessary for capturing the complexity of modern robot interactions and preventing safety hazards.
We deployed KICS and RIS, the Robot Immune System on selected robots. RIS is a security certified software solution that protects robots and robot components against malware, a Robot Endpoint Protection Platform (REPP). We confirmed how both solutions together successfully managed to protect and detect attacks targeting the robots.

INDEX

Start reading →

Securing robot endpoints in Operational Technology (OT) enviroments
Extending KICS with the Robot Immune System (RIS)

Key terms and definitions

- **Architecture dataflow:** A diagram that displays and interrelates the different components, actors and assets that play a relevant role on a given system
- **Attack library:** A set of attack tools, either proof-of-concept code or fully developed (“weaponized”) exploit code that can help you understand the attacks.
- **Attack surface:** A trust boundary and a direction from which an attacker (often captures with another trust boundary) could launch an attack
- **Attack tree:** A method to find threats, a way to organize threats found with other building blocks, or both.
- **Attack vector:** An attack vector is a path that an attacker could follow to perform an attack on the system typically involving an entry point.
- **Entry point:** Specific areas in your architecture from where an actor could initiate attacks.
- **Trust boundary:** Anyplace where various principals come together—that is, where entities with different privileges interact.
- **Threat:** A risk that might exploit a vulnerability to breach security and therefore cause possible harm.
- **Endpoint Protection Platform (EPP):** an integrated suite of endpoint protection technologies that detects and stops a variety of threats at the endpoint.
- **Information Technology (IT):** the technology and use of computers to store, retrieve, transmit, and manipulate data or information throughout and between organizations.
- **Kaspersky Industrial CyberSecurity (KICS):** a holistic solution for industrial infrastructures.
- **Operational Technology (OT):** the technology that manages industrial operations by monitoring and controlling specific devices and processes within industrial workflows and operations, as opposed to administrative (IT).
- **Robot Immune System (RIS):** a security certified software solution that protects robots and robot components against malware. An EPP for robots.
- **Robotics:** A robot is a system of systems. One that comprises sensors to perceive its environment, actuators to act on it and computation to process it all and respond coherently to its application (could be industrial, professional, etc.). Robotics is the art of system integration. An art that aims to build machines that operate autonomously.
- **Zero trust (ZT):** A security model that makes no trust assumptions and demands strict identity verification for every person, device or component trying to access resources on a network,regardless of whether they are sitting inside or outside of the network perimeter

1 Motivation

Introduction

Robot cybersecurity reviews [9], [10] criticize the current status of cybersecurity in robotics and reckon the need of further investing on securing these technologies. Previous attempts to review the security of robots via offensive exercises or tools include [11], [12], [13], [14], [15], [16] which mostly focus on proof-of-concept attacks and penetration testing, detecting flaws in the Robot Operating System (ROS) [18]. A recent study [17] mentions the identification of several flaws within the ROS-Industrial codebase however it does not explicitly describe exploitable ROS-specific flaws. Considerations are made with regard to the open and insecure architecture predominant in ROS-Industrial deployments throughout its open source drivers. From interactions with the authors of [17] it was confirmed that the reported security issues were made generic on purpose, further highlighting the need for further investment on understanding the security landscape of ROS-Industrial setups.

More recent research [8] showed that while ROS is rapidly becoming a standard in robotics (including its growing use in industry), the commonly held assumption that robots are to be deployed in closed and isolated networks does not hold any further and while developments in ROS 2 show promise, the slow adoption cycles in industry will push widespread ROS 2 industrial adoption years from now. Authors at [8] claim that ROS will prevail in the meantime and they wonder whether ROS be used securely for industrial use cases even though its origins did not consider it. [8] Analyzes this question experimentally by performing a targeted offensive security exercise in a synthetic industrial use case involving ROS-Industrial and ROS packages. The scenario is configured following common OT and ICS practices (including segmentation, segregation and an overall hardening). The authors of [8] select one of the most common industrial robots with ROS-Industrial support and configure an industrial environment applying security measures and recommendations. The exercise results into 4 groups of attacks which all manage to compromise the ROS computational graph and all except one take control of most robotic endpoints at desire.

Figure 1. depicts the current landscape of OT environments when augmented with a security specific solution, namely Kaspersky's Kaspersky Industrial CyberSecurity (KICS). KICS is able to monitor and detect security threats at the network level efficiently, complying with the demands of many industrial automation processes. However the robot endpoints, if present, are often left aside. As discussed in [19], "the lack of security has safety repercussions" and both manufacturers as well as industry-owners will need to apply a security-first approach.

Securing robot endpoints in Operational Technology (OT) enviroments
Extending KICS with the Robot Immune System (RIS)

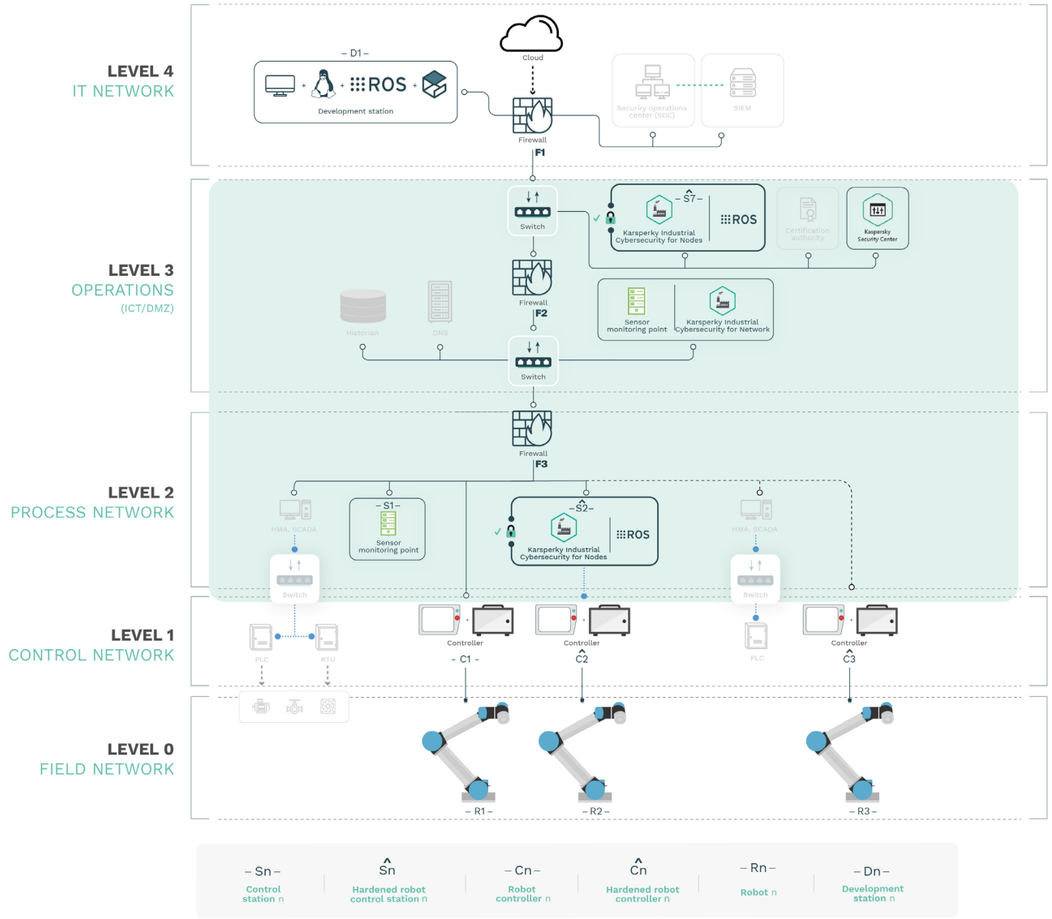


Figure 1. Use case architecture diagram of an OT environment when augmented with a professional cybersecurity solution.

Safety standards in robotics require security

IEC 61508 “Functional Safety of Electrical/Electronic/Programmable Electronic Safety-related Systems” is a meta-standard for safety and from where most functional safety norms grow. This is the case for ISO 26262 (automotive), IEC 61511 (industrial processes), IEC 61513 (nuclear industry) or EN 50126/8/9 (railways), among others.

IEC 61508 indicates the following in section 7.4.2.3:

> *“If the hazard analysis identifies that malevolent or unauthorised action, constituting a security threat, as being reasonably foreseeable, then a security threats analysis should be carried out.”*

Moreover, section 7.5.2.2 from IEC 61508 also states:

> *“If security threats have been identified, then a vulnerability analysis should be undertaken in order to specify security requirements.”*

which translates to security requirements. Note these requirements are complementary to other security requirements specified in other standards like IEC 62443, and specific to the robotic setup in order to comply with the safety requirements of IEC 61508. In other words, safety requirements demand for security requirements, which are specific to the robot and influenced by security research. Periodic security assessments should be performed and as new vulnerabilities are identified, they should be translated into new security requirements.

“safety requirements demand for security requirements, which are specific to the robot and influenced by security research. Periodic security assessments should be performed and as new vulnerabilities are identified, they should be translated into new security requirements”.

Securing robot endpoints in Operational Technology (OT) enviroments
Extending KICS with the Robot Immune System (RIS)

More importantly, the fulfillment of these security requirements to maintain the robot protected (and thereby safe) will demand pushing the measures to the robot endpoint. To meet functional safety standards demand and prevent safety hazards from happening, the effective measures, along with network-based monitoring solutions and endpoint protection for PC-based industrial hosts, should include a security mechanism that protects the robot endpoints, a Robot Endpoint Protection Platform (REPP).

To prevent safety hazards from happening, the effective measures should include a security mechanism that protects the robot endpoints, a Robot Endpoint Protection Platform (REPP)

In order to tackle these security requirements and protect robots from inside-out, we will use the first instance of such REPP systems, namely, Alias Robotics’ Robot Immune System (RIS).

Objectives and Goals

In light of the results of all these past studies and the requirements from standards, it appears that results do not favour the secure use of robots in OT environments today unless additional security measures are implemented.

The present study enhances a simplified version of the environment selected at [8] with additional security measures, namely Kaspersky’s Kaspersky Industrial CyberSecurity (KICS) and Alias Robotics’ Robot Immune System (RIS). The main objective is to study how to further enhance the cybersecurity measures of an OT environment with commercial cybersecurity solutions to guarantee that robots can conduct their operation securely.

We will proceed as follows:

- We will describe in more detail the use case and all the components involved, including a short description of the virtualization used throughout the study to accelerate the production of results.

- We will then perform a series of attacks that will allows us to determine whether the proposed security mechanisms

- Finally, we will provide a series of recommendations for future work that will further optimize the overall security posture of OT environments and its lifecycle.

2 Use case

Figure 2. depicts the use case considered for this study which includes KICS and RIS cybersecurity solutions:

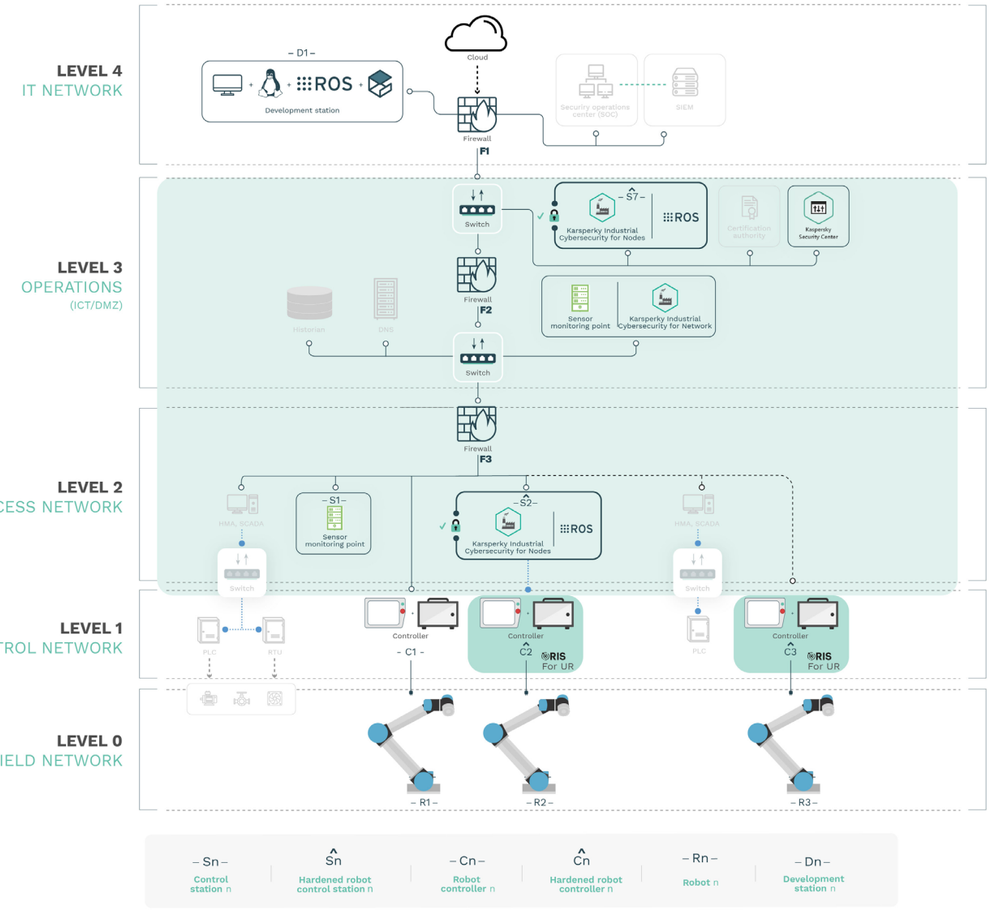


Figure 2. **Use case architecture diagram of this study.** The synthetic scenario presents a network segmented in 5 levels with segregation implemented following recommendations in NIST SP 800-82 and IEC 62443 family of standards. There are three identical collaborative robots from Universal Robots presenting a variety of networking setups and security measures, each connected to their controller. The scenario is augmented with two professional industrial cybersecurity solutions: a) Kaspersky's Kaspersky Industrial CyberSecurity (KICS) and b) Alias Robotics' Robot Immune System (RIS).

Components

Robot Immune System (RIS)

The Robot Immune System (RIS) is a security certified software solution that protects robots and robot components against malware. Inspired by nature, it gets installed directly into the selected robotic system. More particularly, RIS is a Robot Endpoint Protection Platform (REPP), an integrated suite of endpoint protection technologies for robots —including a next-gen antivirus, hardening for known flaws, data encryption, intrusion prevention mechanisms, data loss prevention, etc.— that detects, prevents, stops and informs on a variety of threats that affect the robotic system. Its main capabilities include:

- An adaptive firewall that blocks unexpected communications. Simple adaptation on-the-go.
- Robot-specific hardening. Enforces security policies, patches known vulnerabilities and removes unnecessary communications, often left in robotic systems for development or debugging purposes.
- Secure logging. A forensics-oriented module that registers and records all incoming communications and internal interactions.
- A robot-specific visualization module that allows seamless interaction with RIS. Re-train and control the security notifications with and within the robot.
- A series of machine learning techniques grouped as an Artificial Immune System (AIS) that continuously adapt the response of RIS to incoming security threats.

Kaspersky Industrial CyberSecurity (KICS)

Kaspersky Industrial CyberSecurity (KICS) is a holistic solution for industrial infrastructures. The solution consists of three components:

- KICS for Nodes – a component for industrial network endpoint protection (such as engineering stations, operator stations, SCADA servers)
- KICS for Networks – a component for industrial network monitoring with network integrity checking and deep application protocol inspection capabilities (IEC 60870-5-104, IEC 61850, etc. for electric power infrastructures
- Kaspersky Security Center – a centralized security management software that provides capabilities for systems management, policy management, reporting, notification and SIEM integration.

The three components are treated in more detail below:

Kaspersky Security Center

From [1], Kaspersky Security Center (KSC) is designed for centralized execution of basic administration and maintenance tasks in an organization's network. The application provides the administrator access to detailed information about the organization's network security level and allows configuring all the components of protection built using Kaspersky Lab applications.

KCS provides the following functionality:

- Systems management
  - Centralized system data collection
  - Centralized software deployment
  - Vulnerability detection & patch management
  - Extended client management capabilities
- Policy management
  - Centralized security policy management
  - Remote task scheduling and execution
- Reporting and notification
  - Event logging
  - Dashboards and reports
  - SMS/Email notifications
- SIEM integration
  - ArcSight, Splunk, QRadar
  - Syslog Server

Securing robot endpoints in Operational Technology (OT) enviroments
Extending KICS with the Robot Immune System (RIS)

In addition, KSC supports centralized deployment and management of the several Kaspersky Lab applications [4] summarized in Appendix D. The API for integration with KSC is included in the distribution kit. Particularly and as described at [1], the file kscopenapi.chm contains the description of the API which can be used for integration of third party applications.

One must note that KSC is aimed at **basic administration and maintenance tasks of the overall Kaspersky ICS deployment** but it is not a requirement for the use of individual selected applications. This being the case, for simplicity, we decided **NOT** to use KSC within our setup.

Kaspersky Industrial CyberSecurity for Networks (KICS for Networks)

According to [2], Kaspersky Industrial CyberSecurity for Networks is an application designed to protect the infrastructure of industrial enterprises from information security threats, and to ensure uninterrupted process flows. Kaspersky Industrial CyberSecurity for Networks analyzes industrial network traffic to monitor the activity of devices in the industrial network, detect prohibited system commands transmitted or received by devices, and detect attempts to set incorrect process parameter values. The application is part of the solution known as Kaspersky Industrial CyberSecurity (KICS). It performs the following functions:

- Asset discovery. Passive OT assets identification and inventory
- Deep packet inspection. Almost real-time technical process telemetry analysis
- Network integrity control. Detects unauthorized network hosts and flows
- Intrusion detection system. Alarms signs of offensive network actions
- Command control. Inspects commands over industrial protocols
- External systems. External detection technologies API integration
- Machine learning for anomaly detection (MLAD). Finds cyber or physical violations through real-time telemetry & historical data mining (Recurrent Neural Network)

The API for interacting with KICS for Networks is available at [3].

Kaspersky Industrial CyberSecurity for Nodes

Kaspersky Industrial CyberSecurity for Linux Nodes (or KICS for Nodes) [5] protects computers running Linux operating systems against malware providing the following capabilities.
The ones highlighted have been enabled:

ID	Task name	Description
1	File Threat Protection	File Threat Protection task prevents infection of the file system of the computer. The task resides in the computer's RAM and scans all opened, saved, and active files.
2	Virus scan	A virus scan is a one-time full or custom scan of files on a computer performed by Kaspersky Industrial CyberSecurity for Linux Nodes.
3	Custom scan	Similar to "Virus scan" but targeted to specific files.
4	Boot sector s can	Boot sector scan task lets you scan boot sectors while not specifying a scan scope.
5	Process memory scan	Process memory scan task lets you scan the process memory and kernel memory while not specifying a scan scope.
6	Update	Updating the databases and application modules of Kaspersky Industrial CyberSecurity for Linux Nodes to ensure up-to-date protection on your computer.
7	Rollback	A task that helps to roll back the databases to their previous versions.
8	License	Manage Kaspersky Industrial CyberSecurity for Linux Nodes keys and activation codes
9	Storage management	Storage is a list of backup copies of files that have been deleted or modified during the disinfection process. Backup copy is a file copy created at the first attempt to disinfect or delete this file. Backup copies of files are stored in a special format and do not pose a threat.
10	System Integrity Monitoring	The System Integrity Monitoring task is designed to track actions performed with the files and directories in the monitoring scopes specified in the task settings.
11	Firewall Management	Firewall Management detects all network connections of the user's computer and provides a list of IP addresses, as well as an indication of the status of the default network connection.
12	Anti-Cryptor	The Anti-Cryptor task allows you to protect your files in the local directories with network access by SMB/NFS protocols from remote malicious encrypting.
13	Web Threat Protection	Scans the inbound traffic and prevents downloads of malicious files from the Internet while also blocks malicious, phishing, adware, or other dangerous websites.

Securing robot endpoints in Operational Technology (OT) enviroments
Extending KICS with the Robot Immune System (RIS)

14	Device Control	Manages user access to devices that are installed on or connected to the computer (for example, hard drives, smart card readers, or Wi-Fi modules).
15	Removable drivers scan	Scans a connected device and its boot sectors for viruses and other malware. The following removable drives may be scanned: CDs, DVDs, Blu-ray discs, flash drives (including USB modems), external hard drives, and floppy disks.
16	Network Threat Protection	Scans inbound network traffic for activity that is typical of network attacks. Particularly, this task scans inbound traffic only for 80, 139, 445, and 8080 TCP ports.
17	Container scan	Scans Docker containers, images, and name spaces for viruses and other malware.
18	Custom Container scan	Scans Docker containers and images for viruses and other malware. You can run multiple custom Container Scan tasks simultaneously. A custom Container Scan uses the same parameter values as "Container_Scan", except for the parameter ScanPriority=Normal. The custom task does not scan name spaces.
19	Behavior Detection	The Behavior Detection task monitors malicious activity in the operating system. If the malicious activity is detected, Kaspersky Industrial CyberSecurity for Linux Nodes terminates that process.

KICS for Linux Nodes can be managed by using the following methods:

From the command line using the application control commands [6].

Via Kaspersky Security Center.

Using graphical user

PLC

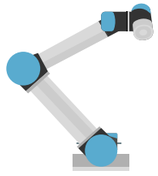
PLC

A Programmable Logic Controller (PLC) or programmable controller is an industrial digital computer which has been ruggedized and adapted for the control of manufacturing processes. PLCs operate such as assembly lines, or robotic devices, or any activity that requires high reliability, ease of programming and process fault diagnosis. PLCs are connected to sensors and actuators in the control process and are networked to the supervisory system (SCADA). In factory automation, PLCs typically have a high speed connection to the SCADA system. In remote applications, such as a large water treatment plant, PLCs may connect directly to SCADA over a wireless link, or more commonly, utilise an RTU for the communications management. PLCs are specifically designed for control and were the founding platform for the IEC 61131-3 programming languages.

Robot

Robot n (**R**n)

The robot (generally only the mechanical side of it and the embed sensors). In this case, given the use case the robots will represent CB3.1 series Universal Robots robots (UR3s, UR5s or UR10s). Communication with the controller happens over an industrial bus. No security measures are enabled within the hardware as far as our inspection went.



Hardware:
UR3, UR5 or UR10

Entry points:
- Fieldbus
- Physical attacks

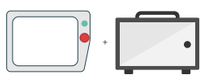
Security measures:
None

Securing robot endpoints in Operational Technology (OT) enviroments
Extending KICS with the Robot Immune System (RIS)

Robot controller

Robot controller n (**C**n)

The robot controller accessible locally via physical means (e.g. USB ports or Ethernet ports) or its local network connections. A simulated version of the robot controller will be developed to speed up testing. Such simulation will be used throughout the exercise and will expose the same services (with the same software versions) and networking ports that the real robot controller does. The controller includes by default no security measures enabled. It must be noted that past work reported several flaws affecting this controller which have yet to be patched. Each controller is assumed to run firmware version 3.13.0 from Universal Robots.



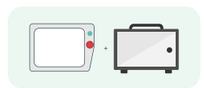
Hardware:
Universal Robots controller CB3.1

Entry points:
- Teach pendant
- Ethernet port
- USB port (in the teach pendant)
- Local area network

Security measures:
None

Hardened robot controller n (**C**n)

A hardened version of the robot controller. The hardening is implemented via the deployment of the Robot Immune System (RIS) and includes patches for known flaws in the controller's services and processes, strict access control, an embedded adaptative firewall, an Intrusion Detection System (IDS), a secure logging mechanism, and a series of techniques that learn from usual interactions (by capturing network and system's information) while developing a pattern for detecting common and uncommon behaviors.



Hardware:
Universal Robots controller CB3.x

Entry points:
- Teach pendant (hardened)
- Ethernet port (hardened)
- USB port (in the teach pendant) (hardened)
- Local area network (filtered)

ROS driver: None

Security measures:
Access control, security patches, IDS, adaptative IDS, secure logging, network filtering

Control station

Control station n (**S**n)

Linux-based control station from where to operate the robot controller (and coherently, the robot mechanics). The station will be based on Ubuntu Bionic (18.04 LTS), include ROS Melodic Morenia and the ROS Industrial drivers for Universal Robots, communicating with the robot controller via a local area network. No wireless connectivity is assumed. Control stations are simulated with limited resources. Particularly, we assign each 4 CPUs and 4096 MB of RAM. Beyond the defaults, no particular security measures are applied into the control stations.

Hardware:
- Industrial-grade PC
- CPU: 4 cores
- RAM: 4096 MB

Entry points:
- Physical access (digital I/O, local area network interfaces, storage devices, etc.)
- Local area network

ROS driver:
- ur_modern_driver
- Universal_Robots_ROS_Driver

Security measures:
None

Hardened control station n ($\hat{\mathbf{S}}$n)

A hardened Linux-based control station from where to operate the robot controller (and coherently, the robot mechanics). The station will be based on Ubuntu Bionic (18.04 LTS), include ROS Melodic Morenia and the ROS Industrial drivers for Universal Robots, communicating with the robot controller via a local area network. Security measures applied follow the recommendations of Canonical's report on how to secure ROS robotics platforms in Ubuntu Bionic 18.04 Linux distribution. On top of these measures, the configuration of the hardened stations was further enhanced using. No wireless communications are assumed to be enabled in the hardened controls stations.

Hardware:
- Industrial-grade PC
- CPU: 4 cores
- RAM: 4096 MB

Entry points:
- Physical access (digital I/O, local area network interfaces, storage devices, etc.)
- Local area network

ROS driver:
- ur_modern_driver
- Universal_Robots_ROS_Driver

Security measures:
See sections below for more details.

Central control station

Central control station n (**C**n)

Linux-based central control station from where to command other ROS-enabled enpoints (such as the ROS drivers enabled on each sub-control station). The station will be based on Ubuntu Bionic (18.04 LTS), include a ROS Melodic Morenia and ROS-Industrial packages, communicating with the robot controller via a local area network. Technical specifications and security measures of the central control station are the same as of hardened control stations S^ n above. The central control station is assumed unique in the networking setup and wherein the ROS Master process will be running (in other words, all other ROS-enabled machines will be acting as slaves).

Hardware:
- Industrial-grade PC
- CPU: 4 cores
- RAM: 4096 MB

Entry points:
- Physical access (digital I/O, local area network interfaces, storage devices, etc.)
- Local area network

ROS driver:
- ur_modern_driver
- Universal_Robots_ROS_Driver

Security measures:
See sections below for more details.

Development machine

Development station n (**D**n)

Linux-based development station from where to develop additional features, monitor and/ or introspect the robotic setup. The station will be based on Ubuntu Bionic (18.04 LTS), includes ROS Melodic Morenia, Gazebo 9 and the ROS Industrial drivers for Universal Robots. A Gazebo simulated instance of the robot will be used for development purposes. Beyond the defaults, no particular security measures are applied into the development station.

Hardware:
- General purpose PCCPU:
- CPU: 4 cores
- RAM: 4096 MB

Entry points:
- Physical access (digital I/O, local area network interfaces, storage devices, etc.)
- Local area network

ROS driver:
- ur_modern_driver
- Universal_Robots_ROS_Driver

Security measures:
None

Virtualization

Due to the expensive price of building a complete real industrial setup including the control stations, controllers, robots, HMIs, PLCs and other equipment, frameworks started appearing which propose a high-fidelity computer virtualization designed to support rapid, parallel experimentation with the automated design of software agents in mind. By leveraging virtualization, we can create a high-fidelity clone of the OT environment. With this security researchers can distribute the work and perform experiments to see how the environment and its security measures respond to certain kinds of attacks.

Virtualization is a commonly used technique in other areas allowing developers and security practitioners to more easily test, reuse and ship systems. When using virtualization, there are often two approaches, emulation and simulation. Emulation is generally implemented through hardware-virtualization, typically using full-virtualization with type two hypervisors (e.g. VirtualBox, VMWare Workstation or QEMU), commonly referred to as Virtual Machines (VMs). This provides complete isolation between guest kernels and the host, while allowing to run many different operating Systems (OS) within the same physical host.

Simulation is often implemented with simpler abstractions such as state machines or through OS-virtualization, using a shared kernel across both the host and guests. OS-virtualization is typically referred to as “containers” in Linux, and is widely considered the most efficient virtualization method for its highest performance and fastest “start-up” time.

In this study we use the alurity toolbox [8] to create a virtual OT environment as depicted in Figure 2. The virtualization happens at both hardware and software levels using emulation and simulation, respectively. The whole virtualization of the scenario can be reproduced with a simple YAML file which is shared and available in Appendix A.

For simplicity and focus, the virtual environment does not include KSC. As noted above, KSC is focused on basic administration and maintenance tasks across Kaspersky Lab applications. Since the purpose of this exercise is to test and evaluate the security offered by KICS in relation to the robotic endpoints only KICS for Networks and Nodes have been used.

3 Attacking an OT environment with robots

In order to analyze the security of the robotic endpoints, the following subsections describe different attack vectors considered on the synthetic OT environment constructed Figure 2.

ATTACK VECTOR 1: Process network insider

In this vector we consider a path that begins from an attacker with access to a machine that has network connectivity with endpoints in the Process Network (Level 2). This could be a device either inside of the Process Network itself or in any other segment with the capability of reaching the Process Network. Details of the particular entry point and method for scaling privileges to gain network capabilities on it are beyond the scope of this study.

In our proof-of-concept attack, the attacker leverages RVD#1489 and launches an exploit that exfiltrates intellectual property.

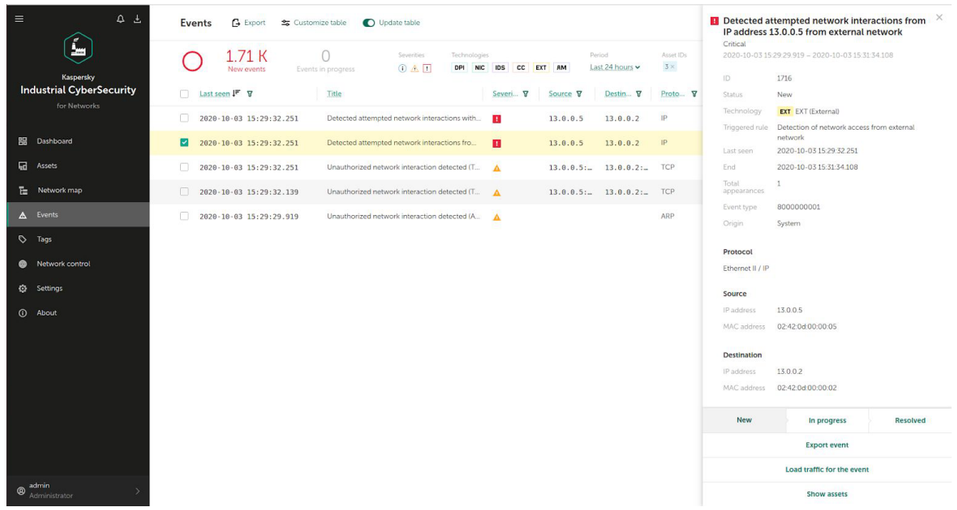


Figure 3. **Detection of an incident as a result of an attacker leveraging RVD#1489.** KICS for Networks shows how after a previous training phase (“Learning” mode) and once it has entered its “Monitoring” mode, it successfully detects unexpected network attempts. The menu on the right allows to handle the status of the event and/or introspect the impacted assets or the network traffic involved.

KICS for Networks was previously trained (in Learning mode) and configured to monitor the network (in Monitoring mode). Correspondingly, since there’s a KICS sensor placed in that Process Network segment, KICS for Networks successfully manages to detect the unusual behavior and raises an event indicating “Detected attempted network interactions from IP …”. The attacker is able to retrieve the intellectual property leveraging vulnerability RVD#1489. The following trace retrieved from KICS from Networks shows this particular network exchange:

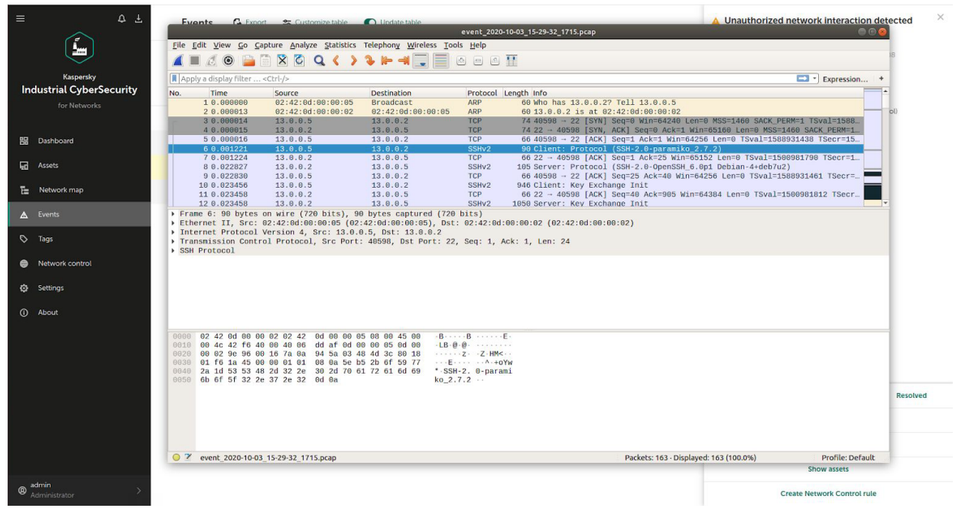


Figure 4. **Network traffic resulting from an attacker exfiltrating Intellectual Property from robot leveraging RVD#1489.** KICS for Networks allows to easily download the traffic that corresponds to a particular event or incident which simplifies significantly the aftermath research of a security breach or cyberattack (incident response).

The same situation was confirmed with other vulnerabilities affecting the selected robotic endpoints including RVD#1406, RVD#1410, RVD#1412 or RVD#1413, among others.

Based on these results we argue that in order to not only detect timely but also to prevent network attacks on robotic endpoints and safety, additional security solutions should be in place which demands for additional security solutions

ATTACK VECTOR 2: Controller insider

In this attack vector we consider a compromised controller that includes no additional security measures. In particular, we assume the attacker compromises C1 as depicted in Figure 5.:

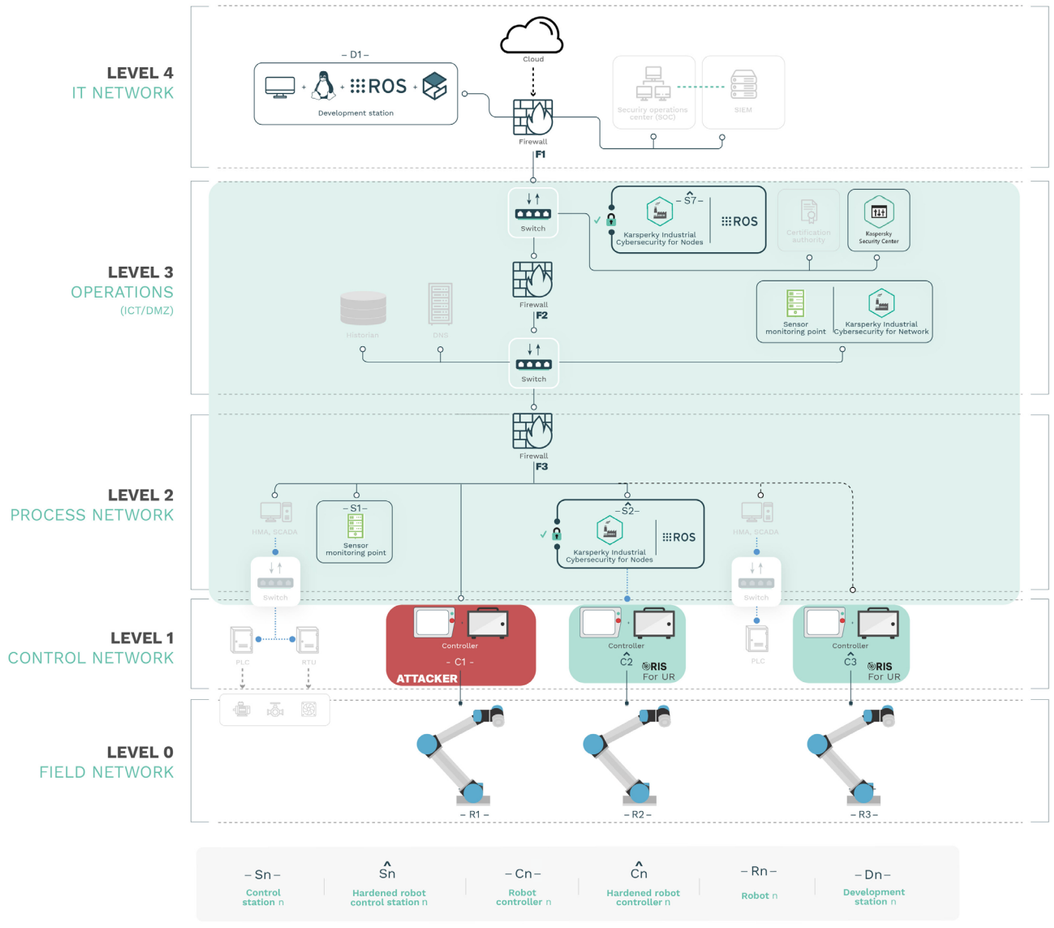


Figure 5. **Figure 5. Entry point of an attacker acting as a controller insider and illustrating how KICS gets no visibility.** The attacker could gain access to the controller by either a) exploiting existing and known vulnerabilities in the controller directly or b) moving laterally using previously mentioned flaws affecting networking exposed services.

Compromising C1 can happen in a variety of ways and either by exploiting known and previously mentioned network vulnerabilities (while moving laterally) or by leveraging other flaws including the widely known RVD#672 or RVD#1408, an attacker could easily gain control of C1 with root privileges. From this point on, the robot could be commanded as desired and safety could be compromised easily.

In our case KICS for Networks has no sensors installed at the Level 2 network, and doesn't monitor traffic coming out of robot controllers towards the robot mechanics and control stations and is thereby unable to detect attacks that originate from the following endpoint pairs of our environment affecting control stations, controllers and robots: S2-C2, C2-R2, C3-R3

The lack of sensors monitoring traffic coming out of robot controllers needs to be complemented with additional solutions that protect robot endpoints. That's where Alias Robotics' RIS comes to play

Access to the controller implies complete control of the robot in most cases. The lack of security measures in robot controller endpoints, together with no detection capabilities makes them an interesting target for attackers, especially given their strategic value in an OT production environment. Attackers will leverage these weaknesses to deploy malware targeting robots. An example of such was illustrated in [20] with the Akerbeltz robot ransomware, where a robot-specific ransomware was developed and demonstrated to block the robot, encrypting IP and demand a ransom to unlock the machine.

Based on the evidence collected, it becomes clear the need of securing the robot controllers. This was validated by RIS deployed in C2 and C3, protecting them proactively and deploying security measures directly at the controller endpoint. With RIS, robot controllers C2 and C3 remained resilient to the same attacks, preventing any safety hazards while informing operators of any suspicious activity as illustrated in Figure 6.

With RIS complementing KICS, robot controllers C2 and C3 remained resilient to the same attacks, preventing any safety hazards

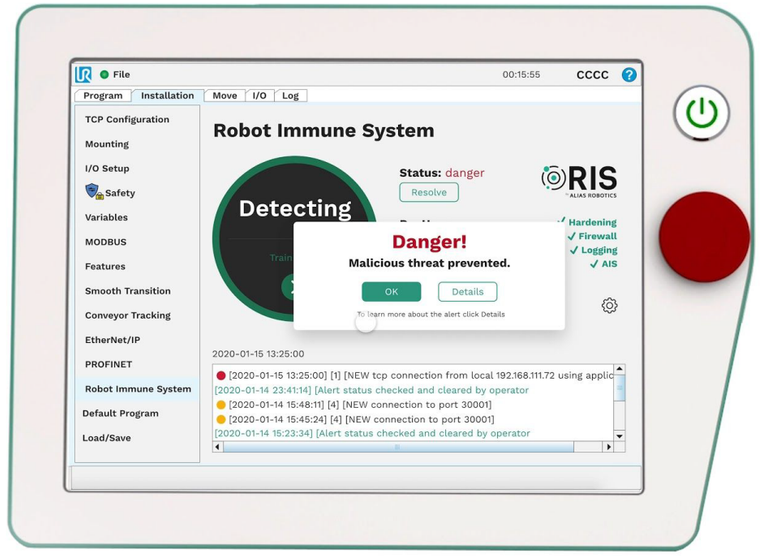


Figure 6. **Robot Immune System (RIS) visualization depicting how it detects an attack attempt in the targeted robot.** The RIS hardening module mitigates flaws and prevents attackers from exploiting them. In the figure the visualization module displays the attack attempt and provides operators with additional information and options on how to handle the incident.

ATTACK VECTOR 3: compromising the ROS network

The Robot Operating System (ROS) [21] is the de facto framework for robot application development. At the time of writing, the original ROS article has been cited more than 7600 times, which shows its wide acceptance for research and academic purposes. ROS' popularity has continued to grow beyond research and into industry while supported by projects like ROS-Industrial (ROS-I for short), an open-source initiative that extends the advanced capabilities of ROS software to industrial relevant hardware and applications.

ROS was not designed with security in mind, but as it started being adopted and deployed into products or used in government programs, more attention was placed on it however, at the time of writing, none of the past security efforts remain actively maintained and no official ROS security solution exists today. Moreover, recent research reports [22] indicate that as-is, ROS can't be used securely.

With the advent of ROS in industry and professional use, security of ROS networks becomes a relevant matter. We deployed a ROS network in our scenario Figure 2. wherein S7 acts as the ROS Master and other stations, including S2 (both hardened by KICS for Nodes) act as slaves deploying ROS nodes. Figure 7. depicts a simplified ROS computational graph where the interactions between the ROS Master and the ROS Slave APIs are illustrated.

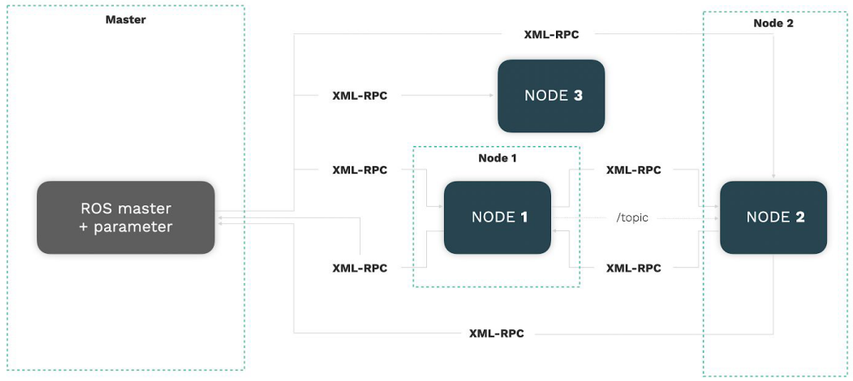


Figure 7. **ROS computational graph.** The computational graph models the nodes executing tasks with their topics, services and other abstractions.

In ROS, nodes intercommunicate through an API (the ROS API) via XML-RPC, a remote procedure call protocol using XML encoding, as well as message/service data exchanges using transport libraries such as ROSTCP or ROSUDP for serialization over IP sockets. All network traffic is transmitted in clear text and in addition, no integrity check is performed on received. Overall, this makes ROS a target for packet sniffing and man-in-the-middle attacks, resulting in an absence of native network confidentiality and data integrity.

The ROS communication model imposes strong restrictions on how network interaction between nodes work and the exchange of data between both. More particularly, the ROS API (Master, Slave and Parameter sub-APIs) via XMLRPC followed by the UDP or TCP sockets (through ROSUDP or ROSTCP transport libraries) require network visibility between each nodes and the Master, as well as between endpoints exchanging data. Figure 8. depicts this while showing a simple talker/listener example, common in the ROS community.

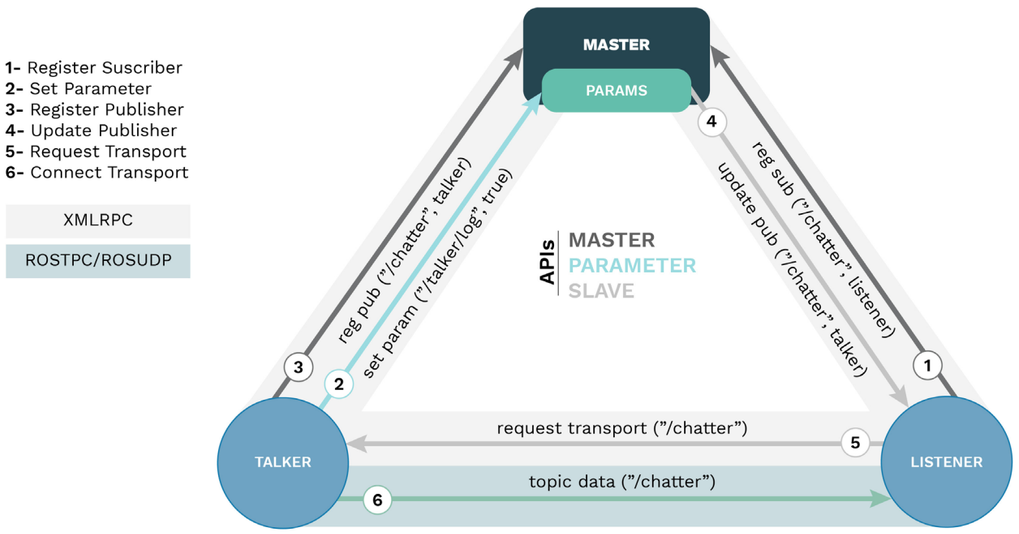


Figure 8. **ROS computational model and APIs.** The three APIs are exemplified using the common talker/listener computational graph which involves two nodes (a talker and a listener) and the Master.

To date KICS doesn’t support the dynamic nature of the ROS networks (new connections on arbitrary TCP/UDP ports). The support will be considered in the future work, that might include a deep packet inspection module for ROS communications in KICS for Network and a submodule in KICS for Nodes that leverages the ROS API, becoming a first class participant of the ROS computational graph and thereby becoming capable of relying relevant information to the centralized security core for autonomous rule generation on-the-go. Such an approach was implemented as part of the RIS for ROS solution available for Kinetic Kame and Melodic Morenia distros

It is technically non-trivial to protect ROS Networks via host-based firewalls or network monitoring solutions. The rules generated by KICS while ‘Learning’ are mostly static and do not capture the complexity of the ROS computational graph. Future versions of KICS might extend the current capabilities in a follow up collaboration.

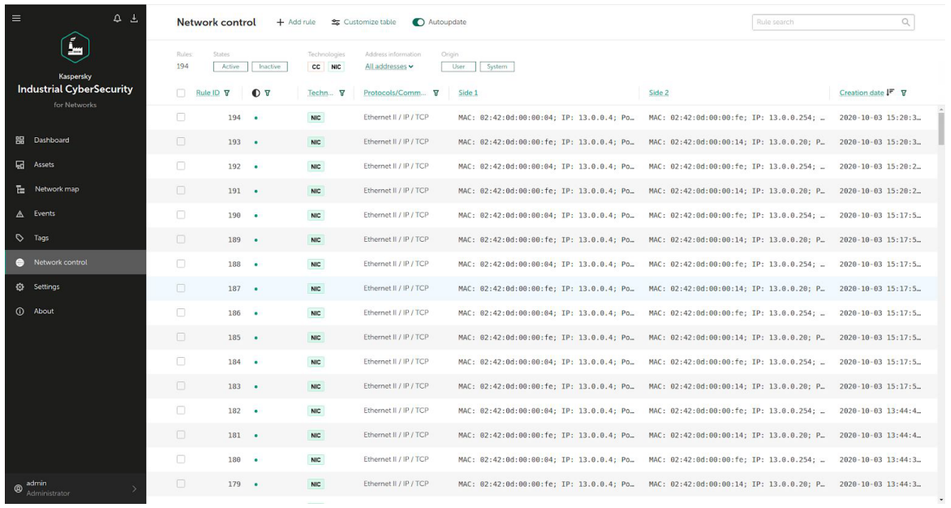


Figure 9. **Rules generated by KICS for Networks while learning the interactions of a simplified ROS computational graph.** The amount of rules grows rapidly as ROS nodes in the graph exchange information. For each new connection, new rules are created. The rules generated are static and specific to each connection, which conflicts with the dynamic nature of the ROS API and its computational graph.

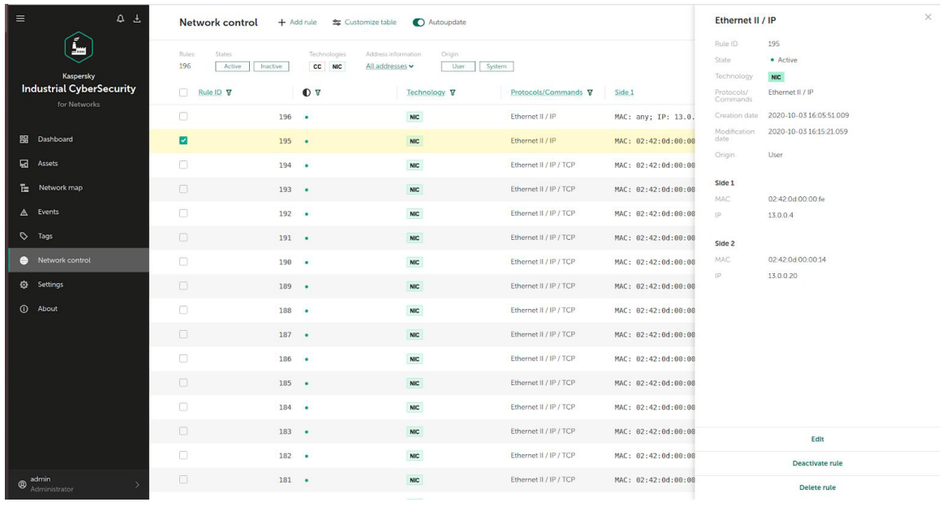


Figure 10. **Example of a KICS for Networks rule that permits ROS interactions without false positives.** The rules that permit fluent ROS communications are at the time of writing generic.

4 Conclusions and future work

The results acquired through past sections led us to conclude that the combination of the Robot Immune System (RIS, on each robot endpoint) together with the Kaspersky Industrial Cybersecurity (KICS) managed to jointly secure robot endpoints in OT environments.

KICS (and particularly KICS for Networks and Nodes) qualifies nicely as an industrial-grade solutions to monitor environments with robotic endpoints; however beyond monitoring general industrial networks, additional security means must be used in combination with KICS to prevent robots from being compromised and cause safety hazards. We confirmed this with attacks targeting robot controllers and ROS networks. We also verified that the addition of RIS as a complementary solution successfully protects robot endpoints, mitigating the cybersecurity flaws of the selected robot endpoints and fulfilling the safety requirements.

the addition of RIS as a complementary solution (to KICS) successfully protects robot endpoints, mitigating the cybersecurity flaws of the selected robot endpoints and fulfilling the safety requirements.

We collected evidence in a synthetic scenario and shared implementation details in Appendix A. From all of it, we learned the following:

- KICS for Networks succesfully detects network attacks, even those targeting robotic endpoints if the attack uses a channel sensed by KICS.
- KICS for Nodes hardens control stations successfully and prevents them from being affected by attacks. ROS can be embedded into such hardened-stations.
- RIS enhancing KICS prevents attacks and protects robot endpoints.
- RIS for UR protects the robot endpoints against targeted attacks and can be used in combination with KICS for securing OT environments with robot endpoints.
- The protection of ROS networks requires an extension of KICS for Networks that grasps the complexity of the ROS API. RIS for ROS follows such an approach and delivers an alternative protection for control stationsagainst targeted attacks.

The general posture of KICS aligns nicely with the specific characteristics of RIS. Note that RIS is available for different robots and robot components, always targeting specific versions. At the time of writing the following options are publicly available for robot endpoints:

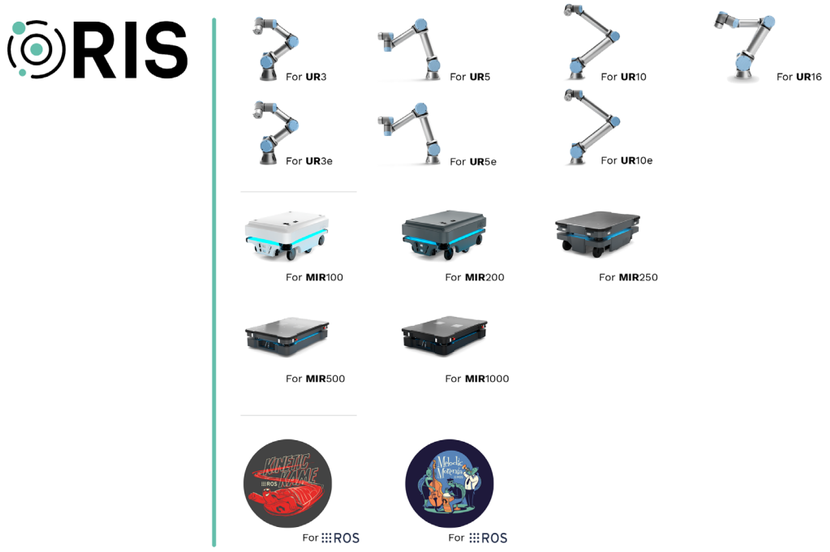


Figure 11. **Robots and robot components supported by RIS.** RIS currently supports several robots and robot components including collaborative robot manipulators (cobots), Autonomous Mobile Robots (AMRs) and frameworks like ROS.

For each one of these endpoints, RIS provides a security-in-depth solution that collects a significant amount of information, currently not being pushed to a centralized security management system. Future work may look into this aspect and leverage this information at the endpoint for an overall enhanced security posture.

Figure 11. summarizes the synthetic OT environment studied and includes some additional interactions specific to KICS, as well as some desired communication paths between KICS and RIS for enhanced security.

Aligned with these thoughts of an enhanced security posture, future research may focus on the following items:

- Interoperability between RIS and KICS for Nodes to harden the communication paths in between control stations and controllers.

- Interoperability between RIS, KICS for Nodes and KICS for Networks to register network interactions without the need of placing additional endpoint-specific KICS sensors.

- Security management of RIS via KSC to deliver updates, patches and new vulnerability mitigations.

- Integrate RIS for ROS into KICS for Nodes, enabling KICS to secure ROS computational graphs.

5 References

[0] Kaspersky. *Kaspersky Support Center 11. Retrieved 22-08-2020* from https://support.kaspersky.com/KSC/11/en-US/5022.htm

[1] Kaspersky. *Kaspersky Support Center 11. Retrieved 22-08-2020* from https://support.kaspersky.com/KSC/11/en-US/5022.htm

[2] Kaspersky. *Kaspersky Industrial CyberSecurity for Networks.* User Guide Application version: 2.8

[3] Kaspersky. *Industrial CyberSecurity for Networks API Developer's Guide.* Retrieved 22-08-2020 from https://support.kaspersky.com/KICSforNetworks/2.9API/en-US/55937.htm

[4] Kaspersky. *List of supported Kaspersky Lab applications.* Retrieved 23-08-2020 from https://support.kaspersky.com/KSC/11/en-US/172903.htm

[5] Kaspersky. *Kaspersky Industrial CyberSecurity for Linux Nodes 1.0.* Retrieved 03-09-2020 from https://support.kaspersky.com/KICS4Linux/1.0/en-US/192756.htm

[6] Kaspersky. *Managing Kaspersky Industrial CyberSecurity for Linux Nodes tasks using command line.* Retrieved 03-09-2020 from https://support.kaspersky.com/KICS4Linux/1.0/en-US/161263.htm

[7] Kaspersky. *Kaspersky Industrial CyberSecurity for Networks.* Retrieved 03-09-2020 from https://support.kaspersky.com/KICSforNetworks/2.9/en-US/83112.htm

[8] Mayoral-Vilches, V., Abad-Fernández, I., Pinzger, M., Rass, S., Dieber, B., Cunha, A., ... & Gil-Uriarte, E. (2020). alurity, a toolbox for robot cybersecurity. *arXiv pre print arXiv:2010.07759.* https://arxiv.org/pdf/2010.07759.pdf

[9] L. Alzola Kirschgens, I. Zamalloa Ugarte, E. Gil Uriarte, A. Muñiz Rosas, and V. Mayoral Vilches, "Robot hazards: from safety to security," ArXiv e-prints, Jun.2018.

[10] G. Lacava, A. Marotta, F. Martinelli, A. Saracino, A. La Marra, E. Gil-Uriarte, and V. M. Vilches, "Current research issues on cyber security in robotics," 2020.

[11] J. McClean, C. Stull, C. Farrar, and D. Mascareñas, "A preliminary cyber-physical se-curity assessment of the robot operating system (ros)," in Unmanned Systems Technolo-gy XV, vol. 8741. International Society for Optics and Photonics, 2013, p. 874110.

Securing robot endpoints in Operational Technology (OT) enviroments
Extending KICS with the Robot Immune System (RIS)

[12] G. Olalde Mendia, L. Usategui San Juan, X. Perez Bascaran, A. Bilbao Calvo, A. Hernández Cordero, I. Zamalloa Ugarte, A. Muñiz Rosas, D. Mayoral Vilches, U. Ayucar Car-bajo, L. Alzola Kirschgens, V. Mayoral Vilches, and E. Gil-Uriarte, "Robotics CTF (RCTF), a playground for robot hacking," ArXiv e-prints, Oct. 2018.

[13] T. Olsson and A. L. Forsberg, "Iot offensive security penetration testing."

[14] V. Mayoral-Vilches, L. U. S. Juan, U. A. Carbajo, R. Campo, X. S. de Cámara, O. Urzelai, N. García, and E. Gil-Uriarte, "Industrial robot ransomware: Akerbeltz," arXiv preprint arXiv:1912.07714, 2019.

[15] S. Rivera, S. Lagraa, and R. State, "Rosploit: Cybersecurity tool for ros," in 2019 Third IEEE International Conference on Robotic Computing (IRC). IEEE, 2019, pp. 415–416.

[16] B. Dieber, R. White, S. Taurer, B. Breiling, G. Caiazza, H. Christensen, and A. Cortesi, "Penetration testing rros," in Robot Operating System (ROS). Springer, 2020, pp. 183–225.

[17] F. Maggi and M. Pogliani, "Rogue automation: Vulnerabile and malicious code in industrial programming," Trend Micro, Politecnico di Milano, Tech. Rep, 2020.

[18] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "Ros: an open-source robot operating system," in ICRA workshop on open source sof-tware, vol. 3, no. 3.2. Kobe, Japan, 2009, p. 5.

[19] Kirschgens, L. A., Ugarte, I. Z., Uriarte, E. G., Rosas, A. M., & Vilches, V. M. (2018). RoboT hazards: from safety to security. arXiv preprint arXiv:1806.06681.

[20] Mayoral-Vilches, V., Juan, L. U. S., Carbajo, U. A., Campo, R., de Cámara, X. S., Urzelai, O., ... & Gil-Uriarte, E. (2019). Industrial robot ransomware: Akerbeltz. *arXiv preprint arXiv:1912.07714.*

[21] Quigley, M., Conley, K., Gerkey, B., Faust, J., Foote, T., Leibs, J., ... & Ng, A. Y. (2009, May). ROS: an open-source Robot Operating System. *In ICRA workshop on open source software* (Vol. 3, No. 3.2, p. 5).

[22] Mayoral-Vilches, V., Pinzger, M., Rass, S., Dieber, B., & Gil-Uriarte, E. (2020). Can ROS be used securely in industry? Red teaming ROS-Industrial. *arXiv preprint arXiv:2009.08211.*

APPENDICES

A Alurity YAML file to reproduce use case

```
# This file allows to reproduce the scenario used for the study titled as
# “Securing robot endpoints in Operational Technology (OT) environments” which
# studies how arbitrary robot endpoints are affected by known malware and how
# existing industrial security mechanisms need to be extended to detect and
# stop these new security threats.
#
# To launch it:
#   alurity start -- --privileged ; alurity enter --user root; alurity stop
# or:
#   alurity flow --user root --kill; alurity stop; alurity start -- --privileged; alurity flow --user root
#

############
# Networks
############
networks:

# Level 0: Field Networks
 #  for each robot-controller pair, there should be a field-level network
 #  however in the context of this study, they won't be used.

 # - network:
 #   - name: field-network_r1_c1
 #   - driver: overlay
 #   - internal: true
 #   - encryption: false
 #   - subnet: 11.1.0.0/24
 # - network:
 #   - name: field-network_r2_c2
 #   - driver: overlay
 #   - internal: true
 #   - encryption: false
 #   - subnet: 11.2.0.0/24
 # - network:
 #   - name: field-network_r3_c3
 #   - driver: overlay
 #   - internal: true
 #   - encryption: false
 #   - subnet: 11.3.0.0/24

# Level 1: Control Networks, connect controllers and control stations
#  for each controller, we expect a dedicated control-network
  - network:
    - name: control-network_c2_s2
    - driver: overlay
    - internal: false
    - encryption: false
    - subnet: 12.2.0.0/24

# Level 2: Process Network, where dedicated control stations and robots
 #   interconnect for automation of processes

 # # overlay version, isolating it from the rest
 # - network:
 #   - name: process-network
 #   - driver: overlay
 #   - internal: true
 #   - encryption: false
 #   - subnet: 13.0.0.0/24

# bridge version, to interface with VMs
 - network:
   - name: process-network
   - driver: bridge
   - subnet: 13.0.0.0/24

# Level 3: DMZ 2 sub-network
#   NOTE: used used to interface Process Network with machines in DMZ 2
#   (e.g. KICS for Networks, Sensor nodes, a historian, additional servers and related)

 # - network:
 #   - name: dmz2
 #   - driver: overlay
 #   - internal: true
 #   - encryption: false
 #   - subnet: 14.0.0.0/24

 - network:
     - name: dmz2
     - driver: bridge
     - subnet: 14.0.0.0/24

# Level 3: DMZ 1 sub-network
#   NOTE: used used to interface IT Network with central control station
 - network:
   - name: dmz1
   - driver: overlay
   - encryption: false
   - internal: true
   - subnet: 16.0.0.0/24

# Level 4: IT Network
 - network:
   - name: it-network
   - driver: overlay
   - encryption: false
   - internal: true
   - subnet: 15.0.0.0/24

# Beyond Level 4: Cloud
 - network:
   - name: cloud-network
   - driver: overlay
   - encryption: false
   - internal: false
   - subnet: 17.0.0.0/24

#################################
# Firewalls and network elements
#################################
firewalls:
    - container:
      - name: firewall-it-dmz1
      - ingress: it-network
      - egress: dmz1
      - rules:
        - iptables -t nat -A POSTROUTING -o eth1 -j MASQUERADE
        - iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
        - iptables -A FORWARD -i eth1 -o eth0 -m state --state RELATED,ESTABLISHED -j ACCEPT
        - iptables -A FORWARD -i eth0 -o eth1 -j ACCEPT
        - iptables -t nat -A POSTROUTING -o eth2 -j MASQUERADE
        - iptables -A FORWARD -i eth2 -o eth0 -m state --state RELATED,ESTABLISHED -j ACCEPT
        - iptables -A FORWARD -i eth0 -o eth2 -j ACCEPT
        - route add 13.0.0.20 gw 16.0.0.254 eth2
    - container:
      - name: firewall-process-dmz2
```

```
      - ingress: process-network
      - egress: dmz2
      - rules:
       - iptables -t nat -A POSTROUTING -o eth1 -j MASQUERADE
        - iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
        - iptables -A FORWARD -i eth1 -o eth0 -m state --state RELATED,ESTABLISHED -j ACCEPT
        - iptables -A FORWARD -i eth0 -o eth1 -j ACCEPT

############
# Containers
############
containers:

 #
 # Robots
 #
 # Robot mechanics could be represented by a machine, by a simulator or by both.
 # in this case and to optimize resources, they are not captured since the
 # end goal of the attacks finalizes with access to the controller.
 # # R1
 # - container:
 #   - name: "r1"
 #   - modules:
 #       - base: registry.gitlab.com/aliasrobotics/offensive/alurity/alurity:latest
 #       - network: field-network

 #
 # Controllers
 #
 # C1
 - container:
   - name: "c1"
   - modules:
       - base: registry.gitlab.com/aliasrobotics/offensive/alurity/robo_ur_cb3_1:3.10
       - network:
         - process-network
         # - field-network_r1_c1
   - cpus: 4
   - memory: 2048
   - extra-options: ALL

 # C^2
 - container:
   - name: "c2"
   - modules:
       - base: registry.gitlab.com/aliasrobotics/offensive/alurity/robo_ur_cb3_1:3.10
       - network:
         - control-network_c2_s2
         # - field-network_r2_c2
   - ip: 12.2.0.20  # assign manually an ip address
   - cpus: 4
   - memory: 2048
   - mount: PWD/installer_ur_220.ris:/installer_ur_220.ris
   - extra-options: SYS_PTRACE

 # C^3
 - container:
   - name: "c3"
   - modules:
       - base: registry.gitlab.com/aliasrobotics/offensive/alurity/robo_ur_cb3_1:3.10
       - network:
         - process-network
         # - field-network_r3_c3
   - cpus: 4
   - memory: 2048
   - mount: PWD/installer_ur_220.ris:/installer_ur_220.ris
   - extra-options: ALL

 #
 # Control stations
 #
 # S^2
 - container:
   - name: "s2"
   - modules:
       - base: registry.gitlab.com/aliasrobotics/offensive/alurity/comp_ros:melodic-scenario  # non-hardened
version
       #  - base: registry.gitlab.com/aliasrobotics/offensive/alurity/comp_ros:melodic-scenario-hardened
       - volume: registry.gitlab.com/aliasrobotics/offensive/alurity/comp_ros_ur:melodic-official-scenario
       - network:
         - control-network_c2_s2
         - process-network
   - ip:
     - 12.2.0.50  # ip for control-network_c2_s2
     # - 13.0.0.6  # ip for process-network
   - extra-options: ALL
 # S7
 - container:
   - name: "s7"
   - modules:
       - base: registry.gitlab.com/aliasrobotics/offensive/alurity/comp_ros:melodic-scenario
       - volume: registry.gitlab.com/aliasrobotics/offensive/alurity/comp_ros_ur:melodic-official-scenario
       - network:
         - dmz1
         - process-network
   - ip:
     - 16.0.0.20  # ip in dmz1
     - 13.0.0.20  # ip in process-network
   - extra-options: ALL

 #
 # Development stations
 #
 # D1
 - container:
   - name: "d1"
   - modules:
       - base: registry.gitlab.com/aliasrobotics/offensive/alurity/comp_ros:melodic-scenario
       #  - base: registry.gitlab.com/aliasrobotics/offensive/alurity/comp_ros:melodic-scenario-hardened
       - volume: registry.gitlab.com/aliasrobotics/offensive/alurity/comp_ros_ur:melodic-official-scenario
       - network:
         - it-network
         - cloud-network
         # - process-network  # bypass firewall restrictions by connecting directly
   # - ip:
   #     - 13.0.0.9
   - extra-options: NET_ADMIN

 #
 # Attackers
 #
 - container:
   - name: attacker_cloud
   - modules:
       - base: registry.gitlab.com/aliasrobotics/offensive/alurity/alurity:latest
       - network:
         - cloud-network
   - extra-options: ALL

 - container:
   - name: attacker
   - modules:
       - base: registry.gitlab.com/aliasrobotics/offensive/alurity/alurity:latest
       - volume: registry.gitlab.com/aliasrobotics/offensive/alurity/reco_nmap:latest
       - volume: registry.gitlab.com/aliasrobotics/offensive/alurity/expl_robosploit/expl_robosploit:latest
       - volume: registry.gitlab.com/aliasrobotics/offensive/alurity/expl_icssploit:latest
       - network:
         - dmz1
         - process-network
   - extra-options: ALL
```

```
#
# extra elements
#

# connector of
#  - it-network
#  - dmz2
#  - dmz1
 - container:
   - name: firewall-it-dmz1
   - modules:
     - base: registry.gitlab.com/aliasrobotics/offensive/projects/rosin-redros-i:firewall-three-net
     - network:
       - it-network
       - dmz2
       - dmz1
   - extra-options: NET_ADMIN
   - ip:
     - 15.0.0.254
     - 14.0.0.254
     - 16.0.0.254

 # DMZ machine
 - container:
   - name: dmz-server
   - modules:
     - base: registry.gitlab.com/aliasrobotics/offensive/projects/rosin-redros-i:dmz
     - network: dmz2
   - extra-options: NET_ADMIN
   - ip: 14.0.0.20

 # Connector of process-network and dmz2
 - container:
   - name: firewall-process-dmz2
   - modules:
     - base: registry.gitlab.com/aliasrobotics/offensive/projects/rosin-redros-i:firewall-two
     - network:
       - dmz2
       - process-network
   - extra-options: NET_ADMIN
   - ip:
     - 14.0.0.253
     - 13.0.0.254


############################
# Virtual Machines (VMs)
############################
vms:
 #
 # KICS for Networks
 #   connected to dmz2
 - vm:
   - name: kics4nets
   - path: $(pwd)/vms/kics4nets
   - network: dmz2
   - ip: 14.0.0.100
   - cpus: 8
   - memory: 8192

 #
 # KICS for Networks sensor 1
 #   connected to process-network
 - vm:
   - name: sensor1
   - path: $(pwd)/vms/sensor1
   - network: process-network
   - ip: 13.0.0.101
   - cpus: 4
   - memory: 4096

 #  #
 #  # KICS for Networks sensor 2
 #  #   connected to the process-network
 #  - vm:
 #    - name: kics4nets_sensor2
 #    - path: $(pwd)/vms/kics4nets_sensor2
 #    - network: process-network
 #    - ip: 12.2.0.4
 #    - cpus: 4
 #    - memory: 4096

 - vm:
   - name: kics4nodes
   - path: $(pwd)/vms/kics4nodes
   - network: process-network
   - ip: 13.0.0.100
   - cpus: 4
   - memory: 4096

#####################################################################
                      FLOWS
#####################################################################

####################
# Flow 1: Demonstrate basic capabilities and main elements of the environment
#  this flow does not perform any specific attack and only displays a basic
#  setup for experimentation and research
####################

flow:
 - container:
   - name: “s2”
   - window:
     - name: routing-ros-dmz
     - commands:
       - command: “sudo route add -net 13.0.0.0/24 gw 13.0.0.254 eth2”
       - split: “horizontal”
       # wait for a signal in port 9999, to start launching the ROS UR driver in the designated IP address
       #  , this is done so that the controller can first set up and install RIS
       - command: “apt-get update; apt-get install -y netcat && \
nc -l -p 9999 && source /opt/ros_ur_ws/devel/setup.bash && \
export ROS_MASTER_URI=’http://13.0.0.20:11311’ && roslaunch ur_robot_driver ur3_bringup.launch robot_
ip:=12.2.0.20”
   # - select: ros-driver

 - container:
   - name: “s7”
   - window:
     - name: routing-ros-dmz
     - commands:
       - command: “sudo route add -net 13.0.0.0/24 gw 13.0.0.254 eth1”  # capture all traffic in the firewall
       - command: “sudo route add -net 14.0.0.0/24 gw 13.0.0.254 eth1”  # reach dmz network
       - command: “sudo route add -net 14.0.0.0/24 gw 16.0.0.254 eth0”
       - command: “route add -net 15.0.0.0/24 gw 16.0.0.254 eth0”  # establish bidirectional comms. with IT Ne-
twork
       - command: “source /opt/ros/melodic/setup.bash && roscore”  # act as ROS master

 - container:
   - name: “d1”
   - window:
     - name: dmz-ros-it-network
     - commands:
       - command: “sudo route add -net 14.0.0.0/24 gw 15.0.0.254 eth0”
       - command: “sudo route add 13.0.0.20 gw 15.0.0.254 eth0”
   - select: dmz-ros-it-network

 # C1
 - container:
   - name: “c1”
   - window:
```

```
      - name: auto-run
      - commands:
        - command: 'echo -e "easybot\neasybot" | passwd'  # define default password
        - command: "mkdir /var/run/sshd && /usr/sbin/sshd"  # enable default SSH port
        - command: "echo '2013333333' > /root/ur-serial && truncate -s -1 /root/ur-serial"
        - command: "cd /root/.urcontrol && ln -s urcontrol.conf.UR3 urcontrol.conf"
    - select: auto-run

  # C^3 - Install RIS
  - container:
    - name: "c3"
    - window:
      - name: auto-run
      - commands:
        - command: 'echo -e "easybot\neasybot" | passwd'  # define default password
        - command: "mkdir /var/run/sshd && /usr/sbin/sshd"  # enable default SSH port
        - command: "echo '2013333333' > /root/ur-serial && truncate -s -1 /root/ur-serial"
        - command: "cd /root/.urcontrol && ln -s urcontrol.conf.UR3 urcontrol.conf"
        - command: "/installer_ur_220.ris"
    - select: auto-run

  # C^2 - Install RIS and launch services, including GUI for ROS services interoperability
  - container:
    - name: "c2"
    - window:
      - name: auto-run
      - commands:
        - command: "echo '2013333333' > /root/ur-serial && truncate -s -1 /root/ur-serial && cd /root/.urcontrol
&& ln -s urcontrol.conf.UR3 urcontrol.conf"
        - command: "/installer_ur_220.ris && source /root/run_gui.sh && $RUN_GUI"
        - split: "horizontal"
        - command: "/bin/sleep 1 && tail --pid=$(pidof installer_ur_220.ris) -f /dev/null && /bin/sleep 10 && cd /
root/.urcontrol/daemon/ && ./run"
        - split: "vertical"
        # signal 20 seconds after ris installer finishes S2 and indicate that that controller's ready for the ROS
driver
        - command: "/bin/sleep 1 && tail --pid=$(pidof installer_ur_220.ris) -f /dev/null && /bin/sleep 20 && echo
-n 'start' | nc -q1 12.2.0.50 9999"
    - window:
      - name: ssh-service
      - commands:
        - command: "/etc/init.d/ssh start"
    - select: auto-run

  - container:
    - name: dmz-server
    - window:
      - name: routing-ros-dmz
      - commands:
        - command: "sudo route add -net 13.0.0.0/24 gw 14.0.0.253 eth0"  # add route enable bidirectional comms
from DMZ 2, to process network (stations)

  # - container:
  #   - name: kics4nodes
  #   - window:
  #     - name: kics4nodes
  #     - commands:
  #       - command: "ls -l /"

  - container:
    - name: kics4nets
    - window:
      - name: kics4nets
      - commands:
        - command: "ls -l /"

  - container:
    - name: sensor1
    - window:
      - name: sensor1
      - commands:
        - command: "ls -l /"

  - container:
    - name: attacker
    - window:
      - name: attacker
      - commands:
        - command: "ls -l /"

  - attach: "c2"
```

B KASPERSKY lab application

- For workstations
  - Kaspersky Endpoint Security 10 for Windows
  - Kaspersky Endpoint Security 11 for Windows
  - Kaspersky Endpoint Security 10 for Linux
  - Kaspersky Endpoint Security 10.1.2 for Linux Elbrus Edition
  - Kaspersky Endpoint Security 10.1.4 for Linux ARM64 Edition
  - Kaspersky Endpoint Security 11 for Linux
  - Kaspersky Endpoint Security 10 for Mac
  - Kaspersky Endpoint Security 11 for Mac
  - Kaspersky Embedded Systems Security for Windows: 2.1 (2.1.0.441), 2.2 (2.2.0.605), 2.3 (2.3.0.754)
- Kaspersky Industrial Cybersecurity
  - Kaspersky Industrial Cybersecurity for Nodes: 2.5, 2.6
  - Kaspersky Industrial Cybersecurity for Linux Nodes 1.0
  - Kaspersky Industrial Cybersecurity for Networks: 2.7, 2.8, 2.9 (centralized deployment is not supported)
- For mobile devices
  - Kaspersky Security 10 for Mobile (Kaspersky Endpoint Security for Android)
- For file servers
  - Kaspersky Endpoint Security 10 for Windows (file server mode)
  - Kaspersky Endpoint Security 11 for Windows (file server mode)
  - Kaspersky Security 10 for Windows Server
  - Kaspersky Endpoint Security 10 for Linux (Server Protection)
  - Kaspersky Endpoint Security 11 for Linux (Server Protection)
  - Kaspersky Security for Virtualization 5.x Light Agent: 5.0
  - Kaspersky Security for Virtualization 5.0 Agentless
- For mail systems and SharePoint/collaboration servers
  - Kaspersky Security 8.0 for Linux Mail Server
  - Kaspersky Secure Mail Gateway 1.0
  - Kaspersky Security 9.0 for SharePoint Server
  - Kaspersky Security 9.0 for Microsoft Exchange Servers
- For detection of targeted attacks
  - Kaspersky Anti Targeted Attack Platform 3.6
  - Kaspersky Sandbox: 1.0, 1.1

Securing robot endpoints in Operational Technology (OT) enviroments
Extending KICS with the Robot Immune System (RIS)

C KASPERSKY Industrial Cybersecurity - detailed capabilities

KASPERSKY SECURITY CENTER (KSC)

- Create a hierarchy of Administration Servers to manage the organization's network, as well as networks at remote offices or client organizations. The client organization is an organization whose anti-virus protection is ensured by the service provider.
- Create a hierarchy of administration groups to manage a selection of client devices as a whole.
- Manage an anti-virus protection system built based on Kaspersky Lab applications.
- Create images of operating systems and deploy them on client devices over the network, as well as perform remote installation of applications by Kaspersky Lab and other software vendors.
- Remotely manage applications by Kaspersky Lab and other vendors installed on client devices. Install updates, find and fix vulnerabilities.
- Perform centralized deployment of keys for Kaspersky Lab applications to client devices, monitor their use, and renew licenses.
- Receive statistics and reports about the operation of applications and devices.
- Receive notifications about critical events during the operation of Kaspersky Lab applications.
- Manage mobile devices.
- Manage encryption of information stored on the hard drives of devices and removable drives and users' access to encrypted data.
- Perform inventory of hardware connected to the organization's network.
- Centrally manage files moved to Quarantine or Backup by security applications, as well as manage files for which processing by security applications has been postponed.

KASPERSKY Industrial Cybersecurity For Networks (kics for networks)

- Scans communications between industrial network devices to check their compliance with defined Network Control rules.
- Monitors industrial network devices and detects the activity of devices previously unknown to the application, as well as the activity of devices that must not be used in the industrial network or that have not shown any activity in a long time. When monitoring devices, the application can automatically refresh information about devices based on data received in network packets.
- Displays the network interactions between industrial network devices depicted as a network map. Problematic objects are visually distinguished from other displayed objects.
- Extracts the parameter values of the technological process controlled by the Industrial Control System (hereinafter referred to as the “ICS”) from network packets and checks the acceptability of those values based on the defined Process Control rules.
- Analyzes industrial network traffic to see if network packets contain system commands transmitted or received by devices involved in automating an enterprise’s processes (hereinafter referred to as “process control devices”). Monitors traffic to detect system commands or situations that could be signs of industrial network security violations.
- Monitors project read and write operations for programmable logic controllers (PLCs), saves the obtained information about projects, and compares this information to previously obtained information.
- Analyzes industrial network traffic for signs of attacks without affecting the industrial network or drawing the attention of a potential attacker. Uses defined Intrusion Detection rules and preset network packet scan algorithms to detect signs of attacks.
- Registers events and relays information about them to recipient systems and to Kaspersky Security Center (KSC).
- Analyzes registered events and, upon detecting certain sequences of events, registers incidents based on embedded correlation rules. Incidents group events that have certain common traits or that are associated with the same process.
- Can be used to work with both the GUI and API.

Securing robot endpoints in Operational Technology (OT) enviroments
Extending KICS with the Robot Immune System (RIS)

KASPERSKY Industrial Cybersecurity For Nodes (kics for nodes)

- Scan file system objects located on the computer’s local drives, as well as mounted and shared resources accessed via the SMB and NFS protocols.
- Scan file system objects both in real time using File Threat Protection tasks and on demand using virus scan tasks.
- Scan boot sectors.
- Scan process memory.
- Detect infected objects. If an object is found to contain code from a known virus, Kaspersky Industrial CyberSecurity for Linux Nodes considers the object as infected.
- Neutralize threats detected in files automatically choosing what action to perform to neutralize the threat.
- Save backup copies of files before disinfection or deletion and restore files from backup copies.
- Manage tasks and configure their settings.
- Add keys and activate the application by using activation codes.
- Notify the administrator about events occurring during the operation of Kaspersky Industrial CyberSecurity for Linux Nodes.
- Update Kaspersky Industrial CyberSecurity for Linux Nodes databases from Kaspersky update servers, via the Administration Server, or from a user-specified source by schedule or on demand.
- Use anti-virus databases to detect and disinfect infected files. Kaspersky Industrial CyberSecurity for Linux Nodes analyzes each file for threats during the scan process: file code is matched against code that resembles a particular threat.
- Monitor the integrity of the system or specified files, and report changes. System Integrity Monitoring can be performed in a constant monitoring mode, and in on-demand scan mode.
- Manage an operating system firewall and, if necessary, restore a set of the firewall rules that was changed.
- Protect your files in the local directories with network access by SMB/NFS protocols from remote malicious encrypting.
- Scan traffic that arrives to the user computer via the HTTP/HTTPS and FTP protocols, and check whether web addresses are malicious or phishing.

- Configure flexible access restrictions to mass storage devices (such as hard drives, removable drives, CD, DVD), data transmission equipment (such as modems), equipment that converts information (such as printers), or interfaces for connecting devices to computers (such as USB, FireWire).
- Scan removable drives when they are connected to a computer.
- Inspect inbound network traffic for activity that is typical of network attacks.
- Scan Docker containers and images, name spaces.
- Receive information about the actions of applications on a computer.
- Specify encrypted connections scan settings.
- Participate in Kaspersky Security Network.
- Allow non-root users manage basic application functions using the graphical user interface.
- Update the application by using update packages.
- Check the integrity of application components by using the integrity check tool.

Securing robot endpoints in Operational Technology (OT) enviroments
Extending KICS with the Robot Immune System (RIS)

ALIAS ROBOTICS

Robot Cybersecurity