Understanding the Internet-Wide Vulnerability Landscape for ROS-based Robotic Vehicles

Wentao Chen*, Sam Der*, Yunpeng Luo, Fayzah Alshammari, and Qi Alfred Chen
University of California, Irvine
Email: {wentac4, sder, yunpel3, fayzaha, alfchen}@uci.edu

***Abstract*—Due to the cyber-physical nature of robotic vehicles, security is especially crucial, as a compromised system not only exposes privacy and information leakage risks, but also increases the risk of harm in the physical world. As such, in this paper, we explore the current vulnerability landscape of robotic vehicles exposed to and thus remotely accessible by any party on the public Internet. Focusing particularly on instances of the Robot Operating System (ROS), a commonly used open-source robotic software framework, we performed new Internet-wide scans of the entire IPv4 address space, identifying, categorizing, and analyzing the ROS-based systems we discovered. We further performed the first measurement of ROS scanners in the wild by setting up ROS honeypots, logging traffic, and analyzing the traffic we received. We found over 190 ROS systems on average being regularly exposed to the public Internet and discovered new trends in the exposure of different types of robotic vehicles, suggesting increasing concern regarding the cybersecurity of today's ROS-based robotic vehicle systems.**

I. Introduction

The rapid development and deployment of robotic vehicles throughout the world has necessitated the use of a programming framework that developers can rely on to ease their development life cycles. Robot Operating System (ROS) [21] is one such framework that enables easy programming, distribution, and reuse of modules for developing robots for a variety of purposes and has become the de facto choice of many developers.

With the popularity of ROS for use in robotic vehicles and the cyber-physical nature of such systems, an attack on ROS can lead to physical damage, harm, and potentially loss of life. Among many methods of attack, infiltrating vulnerable ROS systems exposed to the Internet is one of the easiest to perform while remaining immensely powerful [16]. Thus, it is important to determine how common publicly exposed ROS hosts are and understand the exact patterns attackers may follow when attempting to gain control of them. DeMarinis et al. [16] performed several scans for ROS hosts across the entire IPv4 space and analyzed the data obtained from the scans, including ROS topic and parameter names. These scans took place nearly six years ago (2017 to 2018), which necessitates revisiting this vulnerability landscape, especially considering the unfortunate rise of cyberattacks on robotic technologies across countless environments including industry, transportation, and the home [14]. As discovered in our analysis of our scan results, there is indeed a substantial increase in both the number and variety of robotic vehicles found among ROS instances exposed to the public.

*The first two authors contributed equally.

When it comes to discovering and understanding Internet-wide attack patterns of adversarial actors, past studies have involved setting up intentionally exposed *honeypots* to attract attackers so that researchers can investigate specific protocols involved in such attacks, such as Secure Shell (SSH) [15], Remote Desktop Protocol (RDP) [15], and Telnet [20]. However, to our knowledge, so far there have not been any honeypot studies performed specifically covering ROS-based systems.

To address these research gaps, we conduct a two-part measurement study. In the first part, we conduct multiple scans over the entire IPv4 network space to search for publicly exposed ROS hosts, categorize their robotic system type and characteristics, and discuss potential security implications. In the second part, we immerse ourselves in the perspective of ROS hosts that may fall victim to cyberattacks by developing and deploying custom ROS honeypots. Specifically, we set up ROS-based honeypots that log all received traffic and analyze the packets to identify any adversarial and ROS-related payloads. From these measurements, our key findings are:

- On average, we find that there are over 190 ROS hosts exposed to the public Internet in each scan, which is a nearly 60% increase compared to six years ago [16]. This substantial increase is likely attributable to a new type of ROS device we discovered in our scans: cleaning robots (e.g., ILIFE robot vacuum cleaners); if we exclude them, the number of exposed ROS hosts becomes more comparable to the prior work [16].
- Besides cleaning robots, we also observe significant exposure of other types of robotic vehicles such as drones and autonomous vehicles.
- We did find active ROS-targeted scanners, but so far do not believe any of them have malicious intent. We also captured various potentially malicious scans that hit our honeypots but do not appear to be specifically targeted towards ROS-based systems.

In summary, this work makes the following contributions:

- We revisit the vulnerability landscape of publicly exposed ROS systems six years after the last reported scans [16]. The results of our new scans show a significant increase in the number of regularly exposed ROS

Symposium on Vehicles Security and Privacy (VehicleSec) 2024
26 February 2024, San Diego, CA, USA
ISBN 979-8-9894372-7-6
https://dx.doi.org/10.14722/vehiclesec.2024.23018
www.ndss-symposium.org

hosts, largely attributable to the new and increasing exposure of various types of robotic vehicles such as cleaning robots, drones, and autonomous vehicles.

- Besides identifying exposed ROS hosts, we further perform the first measurement of ROS scanners in the wild by deploying custom ROS honeypots. We found active ROS scanners and potential non-ROS-targeted malicious scans, but have not yet seen evidence of malicious scans specifically targeting ROS.

Code/data release. The code/data from this study are released at our project website **https://sites.google.com/view/cav-sec/roboscan** [12] to benefit future researchers.

II. Background

A. Robotic Vehicles and ROS

Contrary to what its name may imply, the Robot Operating System (ROS) [21] is not actually an operating system, but rather an open source software framework and middleware that facilitates the process of building robotic applications. ROS provides a publisher-subscriber model of communication in which nodes, or individual components of an application, send data to topics, which provide a medium for other nodes to receive this data [8]. Nodes can publish and subscribe to such topics, and when a node publishes data to a particular topic, that data will be received by every node that is subscribed to that topic [8]. ROS currently has two major versions for developers to use:

1) ROS 1: To manage communication between nodes, ROS 1 [21] utilizes a master node running on port 11311 that keeps track of all nodes in an application. A node that wishes to subscribe to a topic sends a request to the master node via the lightweight HTTP-based *XML-RPC* protocol, which in response, provides the addresses of nodes that publish to that topic. The original node would then send requests to those nodes to communicate with them directly.

2) ROS 2: ROS 2 [19] takes a different approach from ROS 1, doing away with both the master node architecture and XML-RPC. It instead turns to a different service, the Data Distribution Service (DDS), which acts as a middleware for ROS 2 by providing the publish-subscribe functionality over UDP using the Real Time Publish Subscribe protocol (RTPS).

B. Previous Works on Publicly Exposed ROS Systems

Previous works have shown that an exposed ROS application can in fact be taken advantage of. Teixeira et al. [23] exploited the fact that by default, ROS 1 does not provide any encryption of the messages sent between nodes: by performing a man-in-the-middle attack, they gained the ability to view and modify the data packets being transmitted between them. This has major implications, as nodes could fail after having their packets intercepted and modified to produce errors such as segmentation faults.

To discover how many ROS 1 applications are regularly exposed to the Internet, DeMarinis et al. [16] ran three scans between December 2017 to January 2018 across the entire IPv4 internet space, each time observing over 100 instances spanning 28 countries [16]. However, it has been nearly six years since their last scan, and as the results of our new scans reveal, both the count and variety of robotic vehicle systems observed have substantially increased since then.

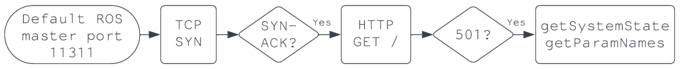


Fig. 1: Overview of ROS 1 scanning procedure. `501` refers to the expected response of error code `501 Not Implemented` to `HTTP GET /` requests sent to the ROS 1 master port [16]. `getSystemState` and `getParamNames` are the *passive commands* (§III-A) used to retrieve information from ROS 1 instances [16].

Other researchers have taken a different approach to examining application security by deploying intentionally exposed applications called "honeypots" to investigate patterns that attackers may follow. Pa et al. [20] set up Internet of Things (IoT) honeypots to examine Telnet-based attacks and discovered that there are at least 4 varieties of DDoS attacks targeting Telnet-enabled IoT devices. Meanwhile, Dang et al. [15] set up honeypots to investigate protocols involved in attacks on IoT applications, including SSH and RDP. However, to the best of our knowledge, no such honeypot studies have been performed for either ROS 1 or ROS 2. Thus, it is important to determine whether any such patterns can be observed in ROS-related attacks and be able to mitigate them if necessary.

III. Internet Scan for ROS Hosts

By default, ROS systems have no security mechanisms, with the ROS master accepting connections and commands from any and all sources [16, 21]. As such, exposure to the public Internet opens up critical security vulnerabilities, with attackers able to arbitrarily connect to open ROS hosts. Once connected, attackers could then retrieve potentially sensitive information and inject their own data into the system, including sending commands that could compromise the integrity of the system as well as the real-world space a physical robot may inhabit [16]. Thus, as described in this section, we perform new scans to revisit the vulnerability landscape of these publicly exposed ROS systems six years since the last reported scans, conducted by DeMarinis et al. [16].

A. Methodology and Setup

Scanning method. In an effort to minimize disturbances to scanned networks and systems, we followed the general scanning procedure described by DeMarinis et al. [16] in our ROS host isolation and scanning pipeline. We used ZMap [17] to asynchronously scan the IPv4 address space for hosts listening and responding on TCP port 11311, and as outlined in Fig. 1, after an initial `TCP SYN` scan to identify hosts responding on port 11311, we sent an `HTTP GET /` request to the port, expecting a response of HTTP error code `501 Not Implemented` [16]. Upon receiving such response, we retrieved non-sensitive host information using *passive commands* that would not alter the state of the ROS system in any way, allowing us to collect information on topics, services, and parameters while minimizing the risk of operational disruptions [16]. As with DeMarinis et al. [16], we made conscious efforts to ensure no sensitive information (e.g., camera feed) was collected from ROS hosts in any capacity.

Result analysis method. As in DeMarinis et al. [16], the data collected from each scan was stored in a local database.

We first manually inspected the data (e.g., hostnames, topic types, and topic, service, and parameter names) for common or meaningful identifiers that indicated different sensors, capabilities, simulators, robot types, and libraries. Then, we constructed queries using these identifiers to automate the process of categorizing the scanned ROS instances. For each ROS instance, we also sought to discern whether it was running within a simulation or in a physical space, to account for differences in potential security implications between instances with capabilities likely in the physical world and those with such capabilities likely only within simulations. Specifically, we categorize a ROS instance as being a *Simulation* (as in Tables I and II) if the instance shows evidence of running a simulator (e.g., Gazebo, playback, or simtime) by identifying topic and service names such as `gazebo`, implying use of the Gazebo simulator [18]; services such as `playback`, signaling the playing back of recorded data; or the parameter `use_sim_time` with a value set to `true`, indicating use of simulated time. Instances not showing any such evidence, including those with `use_sim_time` set to `false`, revealing the potential to run with simulated time inputs [16] but indicating that such simulated inputs were not used at the time of observance [3], are classified as *Likely-Physical*.

Scanner setup. In total, we performed three scans for ROS 1 hosts, each performed approximately one month apart between the months of September and November of 2023. In line with DeMarinis et al. [16] and the guidelines and best practices for Internet scans utilizing ZMap as laid out in Durumeric et al. [17], we performed our scans from a network within our institution, with each scan performed over the span of multiple days to lower the amount of traffic received by any single network at any given time. Additionally, on the host from which we performed our scan, we provided a web page with contact information outlining our work and the purpose of our scans, offering to exclude parties that did not wish to be included in any future scans [16, 17].

B. Results

Overall statistics. Table I shows the aggregated results of the three scans and Table II shows the breakdown of the detected instances based on ROS node characteristics (§III-A). Note that since the deployment period of our honeypots (§IV) overlapped with our scans, the raw scanning results were processed to ensure that all data and results presented in this paper do not include our own honeypots. In total we detected $\sim$194 hosts per scan, which is a nearly 60% increase compared to the results six years ago ($\sim$123) [16]. Of the total instances detected across the three scans, 61 appeared across all three scans, 80 appeared across two of the scans, and 240 appeared in only one scan. Meanwhile, the percentage of hosts observed in addresses belonging to research institutions, as identified through the Autonomous System (AS) name associated with each address, dropped from $\sim$70% [16] to $\sim$35%. These results fall in line with a prediction made by DeMarinis et al. [16] postulating that the number of potentially vulnerable robotic systems, particularly in home and industry settings, would likely only increase as time went on and robots are deployed out of laboratory settings.

Case study: Cleaning robots. One of the major new robot types observed in our scans that is completely absent from the results of DeMarinis et al. [16] is that of cleaning robots. In particular, we observed two major types of cleaning robots: *Rockchip/TinaLinux* and *ILIFE*, identified as such based on the hostnames observed in their corresponding ROS instances. Apart from the specific hostname of individual instances varying between *Rockchip* and *TinaLinux*, instances of the *Rockchip/TinaLinux* cleaning robot type are nearly identical, sporting nodes with names such as `clean_the_room` and `clean_the_area` as well as sensors and capabilities including lidar, IMU, odometry, gyro, and mapping. The *ILIFE* type refers to the ILIFE robot vacuum cleaner [4]. Instances of this type possessed topic names including `clean_mode` and `slam_map_for_cleaning` along with sensors and capabilities such as lidar, odometry, and mapping. All cleaning robot instances contained no evidence of running as simulations at the time of identification and are thus classified as *Likely-Physical*, suggesting that their public Internet exposure may expose possible safety risks and lead to potential direct physical damage.

TABLE I: Summary of the number and robot types of publicly exposed ROS instances found in our scans. The number of exposed instances is substantially higher ($\sim$60% more) compared to the prior scan $\sim$6 years ago [16].

Type	Scan 1	Scan 2	Scan 3	Average
Likely-Physical				
- Cleaning robot	56	65	50	57
- Autonomous vehicle	13	9	8	10
- Drone	8	9	7	8
- Tutorial	9	6	5	7
- Unclassified	83	79	84	82
Simulation	33	28	31	31
Total	202	196	185	194
Prior scan in 2017-18 [16]	144	122	102	123

In fact, this new exposure appears to be the most important contributor to the significant increase in the number of publicly exposed ROS instances in recent years. If we exclude these cleaning robots from the total number of identified hosts, our totals would be 146, 131, and 135 exposed hosts per scan respectively, which is substantially closer to the host numbers identified by the prior work (144, 122, and 102, respectively) [16]. Note that six years ago, this ROS instance exposure was contained largely within university and research settings [16], whereas now, this newly observed type of ROS-based robot is more likely to be found in industrial and home environments (e.g., as marketed by ILIFE [4]). This suggests that this trend of increasing public exposure of robotic systems, if exploited, may result in more damage at a broader societal level than would have easily been possible before.

Case study: Drones. Another robotic vehicle type briefly mentioned but not specifically counted nor discussed in DeMarinis et al. [16] is that of drones, marked by topics like `quadrotor` and `mavros` [7]. Instances of this type observed in our scan revealed sensor capabilities including lidar, IMU, odometry, camera, GPS, and compass. Notably, 24 out of the 27 instances across the three scans with evidence of being drones were *Likely-Physical* with no signs of running simulations, highlighting another type of publicly exposed robotic vehicle with the potential for direct physical damage and potential safety risks. Especially with drones, security is a particularly important concern due to their flying capabilities and increased mobility, posing potential threats to aircraft, in-

TABLE II: Categorization of the publicly exposed ROS instances from our scans based on common topics, services, parameters, nodes, and hostnames. In each row, we list the number of ROS instances of a specific type with a breakdown based on whether it is likely running in a simulator (*Simulation*) or in the physical world (*Likely-Physical*) (classification method described in §III-A). For each category, the different types are ranked based on number of discovered instances.

		Scan 1		Scan 2		Scan 3	
Category	Type	Likely-Physical	Simulation	Likely-Physical	Simulation	Likely-Physical	Simulation
Sensors	Lidar	99	22	122	17	98	22
	IMU	87	19	101	16	75	21
	Odometry	84	19	102	16	72	16
	Camera	40	21	49	17	53	21
	Gyro	51		55		45	
	GPS	22	12	20	8	16	5
	Compass	14	3	14		14	2
	Radar	6	1	5	1	5	2
Capabilities	Movable	75	20	93	13	73	17
	Mapping	56		64	2	50	1
	Nuvo	7	1	5		5	
	CAN bus	5	1	2		2	2
	Navigation	1	2	2	2	1	3
	Traffic sign/light	3	1	2		1	
Simulators	Simtime	7	25	11	21	10	25
	Gazebo		19		16		20
	Playback		8		7		6
Robot types	Cleaning robot	56		65		50	
	- Rockchip	30		31		26	
	- TinaLinux	21		24		16	
	- ILIFE	5		9		6	
	Autonomous vehicle	13	1	9		8	2
	Drone	8	3	9		7	
	Tutorial	9	1	6	1	5	
Libraries	RViz	15	9	17	9	18	10
	Rosbridge	18	4	14	5	18	3
	MAVROS	6	3	8		7	
	RMS	2		2		2	
	darknet_ros	1	1	1			1

frastructure, property, and people [13]. Should a drone exposed to the public Internet be the target of an attack, not only might an attacker be able to obtain information such as that from a camera or GPS, but they may also be able to interfere with its flight systems, leading to potentially disastrous consequences including potential loss of life.

Case study: Autonomous vehicles. We also observe public exposure of a third major robotic vehicle type not discussed in DeMarinis et al. [16]: autonomous vehicles, categorized by the *CAN bus*, *Nuvo*, and *Traffic sign/light* types in Table II. Instances of the *CAN bus* type were identified by topics indicating usage of the CAN protocol, with topic names such as `vehicle_can` (or `VehicleCAN`). The presence of CAN messages in a ROS instance signals a high likelihood that such an instance is an autonomous automobile with computation, communication, and control provided by ROS. Instances of the *Nuvo* type were categorized based on evidence showing usage of Nuvo vehicle AI computing platforms such as the Nuvo-8108GC [6], with hostnames such as `discovery-Nuvo-8108GC-Series`. Such Nuvo vehicle AI computing platforms are commonly used for autonomous vehicle applications such as in Baidu Apollo [2]. Instances of the *Traffic sign/light* type were identified by evidence of traffic sign and light/signal processing capabilities, with topics including `trafficsign`, `trafficlight`, and `trafficsignal`. Two out of seven such *Traffic sign/light* instances also belonged to the other *CAN bus* and *Nuvo* types, further suggesting the likely identity of instances across these three types as autonomous vehicles.

The autonomous vehicle instances we discovered also possessed previously discussed sensor capabilities including lidar, IMU, odometry, camera, gyro, GPS, compass, and radar. As with the identified drone instances, the majority (9 of 12 across the *CAN bus* type, 17 of 18 across the *Nuvo* type, and 6 of 7 across the *Traffic sign/light* type) of these autonomous vehicle instances across the three scans were *Likely-Physical*, showing no evidence of being simulated, highlighting another major publicly exposed robotic vehicle type with potentially disastrous real-world consequences should they be targeted for attack. This becomes especially urgent as autonomous driving continues to develop and evolve at a rapid pace, with related vulnerabilities in autonomous driving technologies having already been extensively demonstrated [22].

IV. ROS Scanners in the Wild

Given the vast number of ROS hosts exposed to the public Internet, it is essential to further understand if real world attackers are already attempting to actively scan and exploit them, along with any patterns they may follow when doing so. To achieve this, we deploy ROS honeypots around the world and report our observations in this section. To the best of our knowledge, this presents the first study to measure ROS scanners in the wild.

A. Methodology and Setup

Honeypot instance setup. We take a similar approach to Dang et al. [15], setting up honeypots on public cloud servers

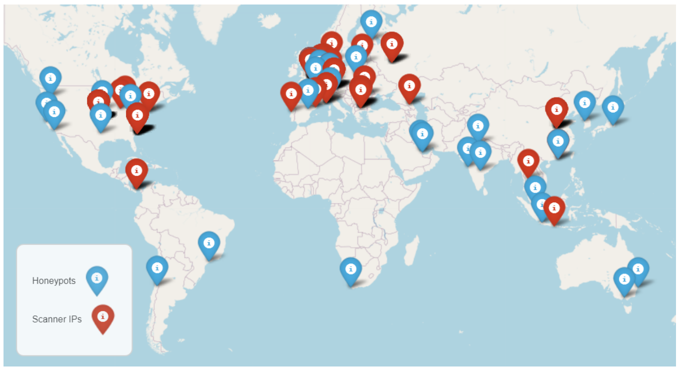


Fig. 2: Geographical distribution of our ROS honeypots. The blue points represent the locations of our honeypots and the red points represent the locations of all the IPs that sent traffic to our honeypots.

and logging network traffic on specific ports. The ports under investigation are TCP port 11311, the default port used for ROS 1 master nodes, and UDP ports 7400 and 7401, where 7400 is the DDS default port and 7401 is a port one of our ROS 2 nodes runs on. If there are packet payloads that use the XML-RPC or RTPS protocols, then there is a high probability that a ROS-related scan or attack is taking place.

Each server runs two copies of the talker-listener application, one on ROS 1 Noetic and the other on ROS 2 Iron. The talker-listener application is the most classic example application in the official ROS tutorial [9], in which a “talker” node sends a simple string every second to a “listener” (receiver) node. To ensure that the honeypots themselves are not corrupted, we run the application on Docker containers, enabling port forwarding on the necessary ports so that they are still publicly exposed. Using the `scapy` library, we intercept traffic to these ports on each server and store them in PCAP files. We then set up an AWS cloud server that retrieves these files using `rsync` and stores each packet in a database, using an open-source visualization tool called Grafana to provide improved insights into this traffic. We wrote a shell script to automate the process of setting up all the honeypot software, and the code of this entire pipeline, including the script, is released at our project website [12].

Honeypot deployment. We deployed 38 honeypots on cloud servers running Ubuntu 22 Jammy, distributed as evenly as possible around the world from three cloud providers: Amazon Web Services (AWS), Google Cloud Platform (GCP), and Microsoft Azure, making sure to open TCP port 11311 (ROS 1) and UDP ports 7400 and 7401 (ROS 2) for all honeypots. We then left the honeypots running from November to December of 2023. Table III and Fig. 2 display information related to each honeypot, including provider and location. We aimed to diversify both the IPs of our honeypots and their geographical locations to maximize the potential traffic received by them, although we were still limited due to the lack of cloud provider availability in select regions.

B. Results

Overall statistics. From the 38 honeypots we set up, we received over 4,000 packets in total from more than 220 IPs over this period. On average, we received around 2.8 packets per honeypot per day. The traffic to ROS 1 took up the majority of the average (approximately 2.7 packets per honeypot per day) while ROS 2 occupied the remaining 0.1, mainly because we deployed our ROS 2 honeypots later than our ROS 1 honeypots. The red points in Fig. 2 indicate the locations of the IPs these packets were sent from and Fig. 3 shows the average number of packets received per honeypot by day.

TABLE III: IP distribution of our 38 ROS honeypots. To maximize the potential traffic our honeypots would receive, we set them up in different locations around the world on different cloud providers.

Provider	IP Address	
AWS	Eastern US	Brazil
	United Kingdom	Bahrain
	South Africa	Japan
	Western India	Singapore
	Spain	Central India
	Western US (2)	Korea
	Italy	Indonesia
	Australia	Germany
	Sweden	
GCP	Central US (2)	Finland
	Australia	Hong Kong
	Qatar	Chile
	Northern India	Poland
	Belgium	Western US
Azure	Western US	Canada
	United Arab Emirates	France
	The Netherlands	South Africa
	Switzerland	Norway
	Central India	

Overall takeaway: No malicious ROS scans observed. We received packets hitting our ROS ports from regions all around the world, including Great Britain, Belize, Belgium, and China, but we do not believe any of them were malicious ROS scans. Specifically, for scans targeted at ROS 1 instances, we expect to receive an XML-RPC payload, since this protocol is needed to interact with the master node. However, for the packets hitting our ROS 1 honeypots, we did not identify any received payloads containing such information. For the packets hitting ROS 2 ports, we received payloads from likely ROS 2-specific scanners, but those were either from known legitimate Internet scanners or from a university, which we elaborate further below. Meanwhile, we did manage to capture likely malicious, but non ROS-targeted, real-world scanners.

Case study: ROS 2 scans. The majority of the packets our ROS 2 honeypots received were from the Shodan search engine, a reputable Internet scanning service [10]. They all consisted of the exact same payload (depicted in Fig. 4) as the one demonstrated by Vilches et al. [24]: an RTPS packet used to discover ROS 2 nodes. By crafting such RTPS packets and sending them to a ROS 2 node on the same subnet, they can trigger a response from another ROS 2 node.

We received a similar RTPS payload from the Technical University of Denmark, shown in Fig. 5. From the payload itself, we were unable to determine what exactly their intentions were; perhaps they were scanning for ROS 2 hosts. Nonetheless, this was one of the only non-Shodan payloads our ROS 2 honeypots received at the time of writing.

Case study: Potentially malicious RDP scans. One of the most notable observations from our study so far was that over 400 of the packets we received on our ROS 1 honeypots contained the text `Cookie: mstshash`. This is most likely a probe attempting to connect to a Remote Desktop Protocol (RDP) instance, rather than targeting ROS. One of

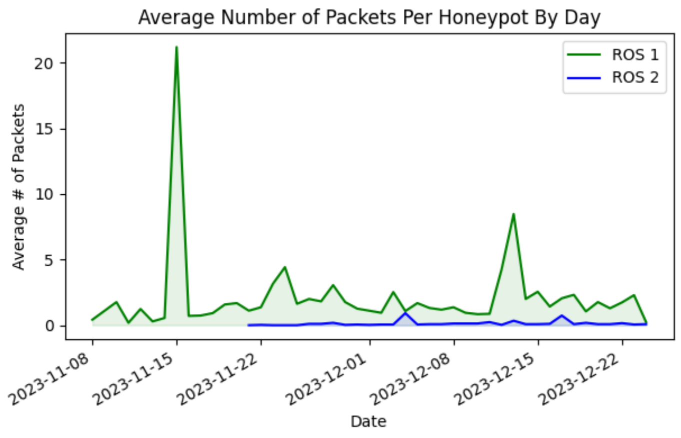


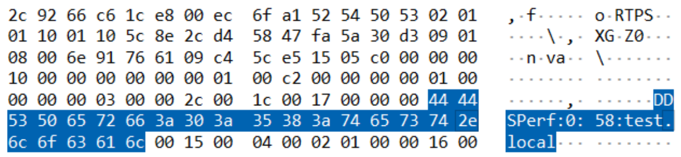
Fig. 3: Number of packets hitting our ROS honeypots per honeypot per day. The number on November 15 was much larger because of an aggressive HTTP scan (§IV-B). The ROS 2 line begins later than the ROS 1 line as we started deploying our ROS 2 honeypots at a later time than the ROS 1 honeypots.

the parameters RDP accepts when connecting to an instance is `mstshash`, which is used for load balancing in RDP servers [11]. We have seen multiple values provided to this parameter including `Administr` and `Domain`, which are usernames truncated to nine characters [11].

At this moment, we are unsure of whether these payloads are part of an attack because we have seen evidence for both possibilities. For example, several developers noted that their public Flask servers received the same `Cookie: mstshash=Administr` payload, characterizing this as a "connection attempt" or an "attack" [5]. Upon further examination of the RDP connection specification however, we discovered that this `Cookie` parameter is optional when sending a connection request to an RDP server, so it is most likely not related to the login process, except for the fact that the value contains an RDP username [11]. In fact, the specification states that `Cookie` can only be present if a separate parameter called `routingToken` [11], used for load balancing, is not present, which implies that `Cookie` is involved more with load balancing rather than with logging in. Thus, we were not able to confidently conclude whether this payload represents an attack.

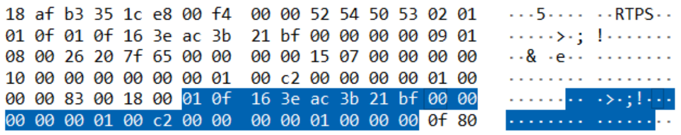
Case study: Other potential malicious scans. Besides the RDP scans, we received one packet from an IP that attempted to exploit the Log4Shell vulnerability in order to gain LDAP access to our honeypot. The same IP then proceeded to perform an aggressive HTTP scan, sending HTTP requests to a variety of paths and files from our honeypots such as `/api/hpe-restapi.json`, `/Wsusadmin/Errors/BrowserSettings.aspx`, and `/api/sonicos/is-sslvpn-enabled`. We checked AbuseIPDB [1], a widely used and trusted website that collects reports on abusive IPs, and found that this IP has been reported previously for aggressive web scans and web app attacks.

We also received multiple attempts to connect to our honeypots via the Secure Shell (SSH) protocol using different versions of OpenSSH from multiple different IPs. Similar findings were also observed in past honeypot studies, including Dang et al. [15].



```
2c 92 66 c6 1c e8 00 ec  6f a1 52 54 50 53 02 01   ,·f·····  o·RTPS··
01 10 01 10 5c 8e 2c d4  58 47 fa 5a 30 d3 09 01   ····\·,·  XG·Z0···
08 00 6e 91 76 61 09 c4  5c e5 15 05 c0 00 00 00   ··n·va··  \·······
10 00 00 00 00 00 00 01  00 c2 00 00 00 00 01 00   ········  ········
00 00 00 03 00 00 2c 00  1c 00 17 00 00 00 44 44   ······,·  ······DD
53 50 65 72 66 3a 30 3a  35 38 3a 74 65 73 74 2e   SPerf:0:  58:test.
6c 6f 63 61 6c 00 15 00  04 00 02 01 00 00 16 00   local···  ········
```

Fig. 4: The ROS 2 scan payload we received from Shodan. Packet content related to RTPS and DDS can be seen in the decoded portion.



```
18 af b3 35 1c e8 00 f4  00 00 52 54 50 53 02 01   ···5····  ··RTPS··
01 0f 01 0f 16 3e ac 3b  21 bf 00 00 00 00 09 01   ·····>·;  !·······
08 00 26 20 7f 65 00 00  00 00 15 07 00 00 00 00   ··& ·e··  ········
10 00 00 00 00 00 00 01  00 c2 00 00 00 00 01 00   ········  ········
00 00 83 00 18 00 01 0f  16 3e ac 3b 21 bf 00 00   ········  ·>·;!···
00 00 00 01 00 c2 00 00  00 00 01 00 00 00 0f 80   ········  ········
```

Fig. 5: The RTPS packet we received from the Technical University of Denmark. Notice how the majority of the content of the two payloads differs, but both are still RTPS packets as indicated by the RTPS header.

V. Conclusion

In this paper, we perform a two-part measurement study to systematically understand the vulnerability landscape for ROS-based robotic vehicles exposed to the public Internet. By performing new scans over the entire IPv4 address space, we find that various classes of robotic vehicles, such as cleaning robots, drones, and autonomous vehicles, are newly and more commonly exposed to the public compared to the previous scans performed six years ago [16]. Although there is always a possibility of such robotic systems being used in research settings (e.g., used as a base for research prototyping), these types of robotic vehicles (in particular cleaning robots) are much more likely to be used in industrial and home environments. As such, the compromising of such vehicle systems by a remote attacker could result in significantly more damage at a broader society-wide level. Meanwhile, by setting up ROS honeypots, we performed the first study on ROS scanners in the wild. Fortunately, although we did find ROS-targeted scans, we did not find any evidence of malicious intentions up to this point. Nevertheless, considering that we did find some (∼190) systems being regularly exposed to and accessible on the public Internet, it becomes both urgent and timely for developers and users of robotic vehicles to start becoming more aware of such problems and take steps towards addressing them. We hope our efforts in this paper can help foster such change, and thus contribute to the security improvements of these critical cyber-physical systems.

Future work. Due to lack of time, in this work we did not get the chance to perform scans for ROS 2 hosts in addition to the ROS 1 hosts. We thus leave this to future work.

Acknowledgements

This research was supported in part by the National Science Foundation under grants CNS-2145493, CNS-1929771, and CNS-1932464 as well as the United States Department of Transportation University Transportation Center Program under grant 69A3552348327.

References

[1] "AbuseIPDB - IP address abuse reports - Making the Internet safer, one IP at a time." [Online]. Available: https://www.abuseipdb.com/

[2] "Apollo." [Online]. Available: https://developer.apollo.auto/platform/hardware.html

[3] "Clock - ROS Wiki." [Online]. Available: https://wiki.ros.org/Clock

[4] "ILIFE® Robot Vacuum Cleaner — Global Offical Site." [Online]. Available: https://www.iliferobot.com/

[5] "mstshash=administr explained · Issue #221 · olipo186/Git-Auto-Deploy." [Online]. Available: https://github.com/olipo186/Git-Auto-Deploy/issues/221

[6] "Nuvo-8108GC NVIDIA RTX 30 Series GPU Computing Platform - Neousys Technology." [Online]. Available: https://www.neousys-tech.com/en/product/product-lines/edge-ai-gpu-computing/nuvo-8108gc-intel-9th-gen-nvidia-rtx-2080-250w-gpu-computing-platform?utm_source=datasheet&utm_medium=web&utm_campaign=Nuvo-8108GC

[7] "ROS Package: mavros." [Online]. Available: https://index.ros.org/p/mavros/

[8] "ROS/Concepts - ROS Wiki." [Online]. Available: https://wiki.ros.org/ROS/Concepts

[9] "ROS/Tutorials - ROS Wiki." [Online]. Available: https://wiki.ros.org/ROS/Tutorials

[10] "Shodan Search Engine." [Online]. Available: https://www.shodan.io/

[11] *[MS-RDPBCGR]: Remote Desktop Protocol: Basic Connectivity and Graphics Remoting — Microsoft Learn*, Sep. 2022. [Online]. Available: https://learn.microsoft.com/en-us/openspecs/windows_protocols/ms-rdpbcgr/5073f4ed-1e93-45e1-b039-6e30c385867c

[12] "AD & CV Systems Security - RoboScan," 2024. [Online]. Available: https://sites.google.com/view/cav-sec/roboscan

[13] R. Altawy and A. M. Youssef, "Security, Privacy, and Safety Aspects of Civilian Drones: A Survey," *ACM Trans. Cyber-Phys. Syst.*, vol. 1, no. 2, nov 2016. [Online]. Available: https://doi.org/10.1145/3001836

[14] A. Botta, S. Rotbei, S. Zinno, and G. Ventre, "Cyber security of robots: A comprehensive survey," *Intelligent Systems with Applications*, vol. 18, p. 200237, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2667305323000625

[15] F. Dang, Z. Li, Y. Liu, E. Zhai, Q. A. Chen, T. Xu, Y. Chen, and J. Yang, "Understanding Fileless Attacks on Linux-based IoT Devices with HoneyCloud," in *Proceedings of the 17th Annual International Conference on Mobile Systems, Applications, and Services*, ser. MobiSys '19. New York, NY, USA: Association for Computing Machinery, 2019, p. 482–493. [Online]. Available: https://doi.org/10.1145/3307334.3326083

[16] N. DeMarinis, S. Tellex, V. P. Kemerlis, G. Konidaris, and R. Fonseca, "Scanning the Internet for ROS: A View of Security in Robotics Research," in *2019 International Conference on Robotics and Automation (ICRA)*, 2019, pp. 8514–8521.

[17] Z. Durumeric, E. Wustrow, and J. A. Halderman, "ZMap: Fast Internet-wide Scanning and Its Security Applications," in *22nd USENIX Security Symposium (USENIX Security 13)*. Washington, D.C.: USENIX Association, Aug. 2013, pp. 605–620. [Online]. Available: https://www.usenix.org/conference/usenixsecurity13/technical-sessions/paper/durumeric

[18] N. Koenig and A. Howard, "Design and Use Paradigms for Gazebo, An Open-Source Multi-Robot Simulator," in *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566)*, vol. 3, 2004, pp. 2149–2154 vol.3.

[19] S. Macenski, T. Foote, B. Gerkey, C. Lalancette, and W. Woodall, "Robot Operating System 2: Design, architecture, and uses in the wild," *Science Robotics*, vol. 7, no. 66, p. eabm6074, 2022. [Online]. Available: https://www.science.org/doi/abs/10.1126/scirobotics.abm6074

[20] Y. M. P. Pa, S. Suzuki, K. Yoshioka, T. Matsumoto, T. Kasama, and C. Rossow, "IoTPOT: Analysing the Rise of IoT Compromises," in *9th USENIX Workshop on Offensive Technologies (WOOT 15)*. Washington, D.C.: USENIX Association, Aug. 2015. [Online]. Available: https://www.usenix.org/conference/woot15/workshop-program/presentation/pa

[21] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Ng, "ROS: an open-source Robot Operating System," in *ICRA Workshop on Open Source Software*, vol. 3, 01 2009.

[22] J. Shen, N. Wang, Z. Wan, Y. Luo, T. Sato, Z. Hu, X. Zhang, S. Guo, Z. Zhong, K. Li, Z. Zhao, C. Qiao, and Q. A. Chen, "SoK: On the Semantic AI Security in Autonomous Driving," 2022.

[23] R. R. Teixeira, I. P. Maurell, and P. L. Drews, "Security on ROS: analyzing and exploiting vulnerabilities of ROS-based systems," in *2020 Latin American Robotics Symposium (LARS), 2020 Brazilian Symposium on Robotics (SBR) and 2020 Workshop on Robotics in Education (WRE)*, 2020, pp. 1–6.

[24] V. M. Vilches, C. Toyama, E. Boasson, F. Maggi, M. Cheng, P. Kuo, and T.-L. Yen, "Hacking ROS 2 ethically," Nov. 2021. [Online]. Available: https://cybersecurityrobotics.net/hacking-ros-2/