Scanning the Internet for ROS: A View of Security in Robotics Research

Nicholas DeMarinis, Stefanie Tellex, Vasileios P. Kemerlis, George Konidaris, and Rodrigo Fonseca
Department of Computer Science, Brown University
Email: {ndemarin, stefie10, vpk, gdk, rfonseca}@cs.brown.edu

***Abstract*— Security is particularly important in robotics, as robots can directly perceive and affect the physical world. We describe the results of a scan of the entire IPv4 address space of the Internet for instances of the Robot Operating System (ROS), a widely used robotics software platform. We identified a number of hosts supporting ROS that are exposed to the public Internet, thereby allowing anyone to access robotic sensors and actuators. As a proof of concept, and with the consent of the relevant researchers, we were able to read image sensor information from and actuate a physical robot present in a research lab in an American university. This paper gives an overview of our findings, including our methodology, the geographic distribution of publicly-accessible platforms, the sorts of sensor and actuator data that is available, and the different kinds of robots and sensors that our scan uncovered. Additionally, we offer recommendations on best practices to mitigate these security issues in the future.**

I. Introduction

Security is particularly important in robotics. A robot can sense the physical world using sensors, or directly change it with its actuators. Thus, it can leak sensitive information about its environment, or even cause physical harm if accessed by an unauthorized party. Existing work has assessed the state of industrial robot security and found a number of vulnerabilities [1, 2]. However, we are unaware of any studies that gauge the state of security in *robotics research.*

To address this problem we conducted several scans of the whole IPv4 address space, in order to identify unprotected hosts using the Robot Operating System (ROS) [3], which is widely used in robotics research. Like many research platforms, the ROS designers made a conscious decision to exclude security mechanisms because they did not have a clear model of security threats and were not security experts themselves. The ROS master node trusts all nodes that connect to it, and thus should not be exposed to the public Internet. Nonetheless, our scans identified over 100 publicly-accessible hosts running a ROS master, shown in Figure 1. Of those we found, a number of them are connected to simulators, such as Gazebo [4], while others appear to be real robots capable of being remotely moved in ways dangerous both to the robot and those around it. We present both a quantitative and qualitative overview of our findings.

Quantitatively, we assessed the number of topics that appear to be sensors and actuators of various types. We noticed a roughly Zipfian distribution, with a few common types and a long tail of one-off sensors and actuators. We also observed that many robots are online for a relatively short period of time (hours or days) and then go offline again.

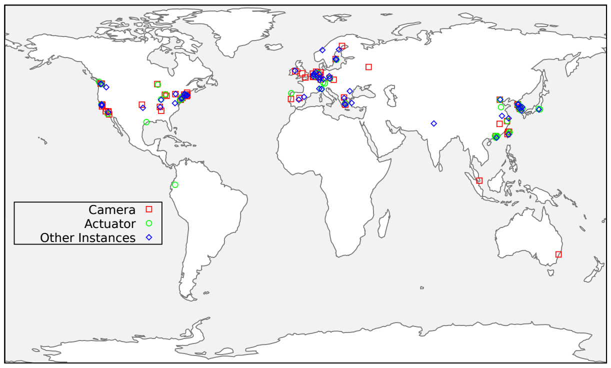


Fig. 1: Approximate locations (slightly jittered to illustrate multiple points) of ROS masters identified across all scans. Red indicates a host publishing camera information. Green indicates a host that showed evidence of a robot which could be actuated. Other hosts are in blue.

As a qualitative case study, we also present a proof-of-concept "takeover" of one of the robots (with the consent of its owner), to demonstrate that an open ROS master indicates a robot whose sensors can be remotely accessed, and whose actuators can be remotely controlled.

This scan was eye-opening for us, too—we found two of our own robots as part of the scan, one Baxter [5] robot and one drone. Neither was intentionally made available on the public Internet, and both have the potential to cause physical harm if used inappropriately.

Our goal is not to single out any researchers or robot platforms, but to promote security as an important consideration—not just in production systems, but in research settings as well. We aim to provide information about a concerning situation and guidance on how roboticists can improve their security. Note that prior to the release of this work, we reached out to the owners of all affected robots and provided them with a summary of our findings.[1]

II. Related Work

Anecdotally, we are aware of a number of compromised robots. For example, Baxter [7] robots use an SSH server with a known default username and password that cannot be changed by the owner [8]. While the SSH account does not directly give administrative access, it is still a significant risk

[1]Further details on our findings and recommendations for ROS users are available at `https://systems.cs.brown.edu/robotsecurity` and in an extended version of our paper [6].

as it can be used as a stepping stone in a multistage attack. That said, we are unaware of any specific Baxters that were compromised as a result of this issue.

Denning et al. [9] studied the security and privacy risks of home robots. They investigated three specific household robots in 2008, uncovered a number of vulnerabilities, and surveyed implications for the future. McClean et al. [10] identified various ROS security issues after setting up a honeypot at DEFCON-20 (2012), including plaintext communications, open ports, and unencrypted storage. Cerrudo and Apa [11] identified vulnerabilities in several home and research robotics platforms, demonstrating weak or non-existent authentication procedures that allow an attacker to control robots, perform firmware updates, or gain access to sensor data.

Quarta et al. [1] and Maggi et al. [2] conduct a security assessment on an industrial robot controller, combined with a practical exploit of an arm. They surveyed domain experts from both academia and industry, and found that 30% had robots accessible from the Internet, while 76% had never performed a professional cybersecurity assessment. They also used Internet search engines, like Shodan and ZoomEye, to find FTP servers matching industrial devices, identifying 28 robots and thousands of "industrial routers" enabling remote access to devices. Our results, instead, quantitatively assess the availability of potentially-vulnerable ROS robots on the public Internet.

The ROS community has been making efforts to develop security extensions for ROS systems—we refer readers to [12] for a more comprehensive survey. Most notably, the SROS project [13, 14], under development by the Open Source Robotics Foundation (OSRF), updates the ROS API with TLS [15] to secure connections between nodes and provides a key distribution service to manage chains of trust and access controls. The work of Dieber et al. [12] propose an architecture adding authentication, authorization, and key management services for individual ROS nodes, as well as extensions to SROS' transport layer. However, as these projects are still in development, neither are yet in wide use.

Rosbridge [16] provides a WebSocket interface to ROS, which can be used as part of Robot Web Tools [17] to make robots accessible from the Internet. To facilitate this, Rosbridge provides TLS support for WebSocket connections, access controls to limit available topics, and an authorization mechanism to restrict API calls [17]. However, all of these features are optional and only apply to clients using the WebSocket API.

ROS2 [18] is the next iteration of the ROS architecture, which is currently in development. ROS2 is based on the Data Distribution System (DDS) standard [19] for data exchange, a flexible middleware interface to allow users to customize the underlying transport mechanisms for different applications. The DDS standard contains optional security extensions [20], which are in development for ROS under the SROS2 project [21]. While these are promising steps for ROS' future, ROS1 is currently the dominant platform.

III. Robot Operating System (ROS)

The Robot Operating System (ROS) [3] was introduced by Willow Garage in 2007. ROS operates as a publish-subscribe service to distribute data among *nodes* in a system. A central master service is responsible for tracking published and subscribed topics and provides a parameter server for nodes to store various metadata. Nodes can publish data as *topics* by advertising to the ROS master service. Other nodes can subscribe to these topics by querying the master, which provides the IP address and TCP port number of any nodes publishing a given topic, allowing the subscriber to contact the publishers directly to establish further connections.

ROS has a distributed architecture: nodes may run on the same machine as the master, or on different machines. The ROS master API is implemented using the XML-RPC protocol, which is built over HTTP, and typically listens on TCP port 11311 [22]. Each node runs its own XML-RPC server that allows the node to advertise the topics it publishes other nodes, update parameters, publish or subscribe to topics on other nodes, and receive parameter and publisher updates from the ROS master [23].

IV. Identifying Robots using ROS

We searched for ROS masters connected to the IPv4 Internet address space by scanning all public addresses (roughly 3.7 billion IPs) on TCP port 11311, the default ROS master port. Our scans were performed using ZMap [24], a research tool for performing Internet-wide port scans. Port scans operate by asynchronously sending probe packets (`TCP SYN`) to a set of addresses to gather information about the hosts that respond. We note that our scans only considered the *default* master port—hosts using other ports would not be observed by our broad scans, but could still be detected by more targeted scans that probe more ports on each host.

While port scans are very common on the public Internet, conducting Internet-wide scans poses some inherent risks. First, and foremost, the volume of traffic sent by a scan could overwhelm destination networks. For this reason, we chose ZMap as our scan apparatus because it selects addresses to scan using a pseudorandom permutation, rather than sending probes in sequential order, to greatly reduce the number of packets sent to a network at one time. Second, sending ROS commands to unknown hosts also has the potential to cause disruption: *e.g.*, if the host is running a service other than a ROS master on port 11311. When designing our scanning framework, *we made efforts to minimize potential disruptions by using a series of minimally-invasive probes to confirm the presence of a ROS master before sending commands.* Specifically, our scans were conducted in four stages, each run on successively fewer hosts:

1) A `TCP SYN` scan to identify hosts accepting connections on the default ROS master port.
2) A `TCP SYN` scan on a high-numbered, normally-closed port (*e.g.* 58243), to rule out addresses that may respond on *any* port to deter scanning [25].

3) An `HTTP GET /` request on the ROS master port. The ROS master responds to this request with a specific error code, ruling out other services.
4) A series of *passive* ROS commands to collect host information.

We also scanned for Rosbridge instances, which run on TCP port 9090 and communicate using a JSON-based WebSocket protocol. The scanning process for Rosbridge followed similar stages, with the exception of using a different test in stage 3 to identify WebSocket-capable HTTP servers before sending commands.

Critically, sending commands to active robots may pose a safety hazard. We selected a minimal set of *passive* commands designed to confirm that the host was in fact running ROS and gather data on the topics and parameters available on each ROS instance. *At no time did we attempt to modify the state of the ROS master, or connect to any nodes,* with the exception of the experiments discussed in Section VI, which were performed with express permission from the robot's operators. Specifically, we called `getSystemState` to retrieve the list of publishers, subscribers, and services, and called `getParamNames` to get the list of all named parameters (but not their value). Additionally, we retrieved the value of the parameter indicating the ROS version, as well as the `robot_description` parameter which returns the URDF (Unified Robot Description Format) if present [26].

Our scans were conducted from a host located on our (university campus) network. Each scan was performed over a period of 7 weekdays, to ensure a low rate of probe traffic at destination networks. In accordance with a set of best practices outlined by the ZMap authors [24], we made efforts to provide information to any users observing our scans. This included publishing a web page on our scanning host, with a description of the scan and a contact email address for more information or to request removal from further scans. (Note that over the course of our scanning period, we received only one *automated* request to cease probing an organization's network, and complied with the request.)

V. Quantitative Results

We conducted three scans on the ROS master port between December 2017 and January 2018. We refer to these scans as Master 1–3, respectively. Each ROS master scan observed over 100 ROS instances, spanning 28 countries, with over 70% of the observed instances using addresses belonging to university networks or research institutions. We performed one scan for Rosbridge instances in November 2017 and identified 15 total instances, with 11 instances located in networks recognizable as cloud service providers.

Table I provides a summary of our results, organized into types based on their topic data. We define a simulator as a host that showed evidence of one of the robot simulator topics listed in Section V-A. We define a robot as a host that shows evidence of a sensor and an actuator, but is not a simulator. Each type is mutually exclusive, so identified sensors, actuators, and robots, in this table, did not show evidence of being a simulator. These results must be taken as approximate, since we did not actually subscribe to any of the topics, consider their type, or verify connectivity with real hardware. However, given the standardization of topic names, it seems likely that many of the hosts we found were indeed running sensors, actuators, and other software. Empty ROS cores showed only the base `rosout` topics and services. Some of these hosts additionally had parameters set (perhaps indicating some nodes had been running previously, and later shut down). Unclassified nodes did not fit into any of our other categories.

TABLE I: Scan results summary.

Category	**Master 1**	**Master 2**	**Master 3**	**Rosbridge**
Identified robots	19	13	12	4
Simulation only	37	32	21	2
Empty ROS cores	37	29	26	0
Only sensors	24	28	18	2
Only actuators	2	1	3	0
Only services	11	8	12	6
Unclassified	14	11	10	1
Total Instances	**144**	**122**	**102**	**15**

We do not combine the results of each scan to form a "grand total" for each type, as many hosts appeared in more than one scan and returned different topic data each time. We discuss these observations in more detail in Section V-B.

Table II shows the number of hosts for every type of sensor, actuator, and service, for each scan. Each host may appear in more than one row of this table, for example, if it contained evidence of a camera, an IMU (Inertial Measurement Unit), or a joystick. We separate results into hosts that showed evidence of being a simulator vs. all others. This way, we can separate hosts that showed evidence of exposing physical sensors compared to hosts that are likely only exposing simulated sensors and actuators. We inferred information about robots from the `robot_description` parameter, as well as some canonical topic names. The most common type of robot in our search was the Turtlebot [27]. We also found Baxter robots [5], WAM arms [28], a Da Vinci research kit [29], drones, vehicles, and one flying insect.

A. Identifying topics

We classified the number of sensors and actuators in different categories. For privacy and safety reasons, our scans retrieved only the *names* of the topics and parameters available on each ROS master. As we did not subscribe or publish to any topics, we cannot state with certainty whether we have found active sensors or actuators. However the list of topic names provides substantial evidence exposed resources.

With the obtained data stored in a local database, we constructed queries for various topics of interest in robotics, and divide our results into sensors and actuators, common libraries, and simulators. In constructing queries, we manually examined the topic list in order to identify topics that map to different sensors/actuators. As we found indicators, we added these to the list of queries run on all data to classify the object. We describe the most prevalent queries below, the complete list can be found in the extended version [6].

TABLE II: Search results for common robotic system components from topic and parameter names

Category	Parameter	Master 1		Master 2		Master 3		Rosbridge	
		Phys. HW	**Sim./Log**	**Phys. HW**	**Sim./Log**	**Phys. HW**	**Sim./Log**	**Phys. HW**	**Sim./Log**
Sensors	Camera	29	22	29	11	22	11	5	1
	Camera + Depth	13	15	13	6	9	8	2	1
	Camera + RGB	12	7	6	5	9	5	1	1
	Camera + Stereo	1	3	1	4		3		
	Kinect	3	2	3	2	2	2		
	IMU	14	16	9	16	6	11	2	
	Lidar	12	13	5	9	2	11	3	2
	Motion Capture	4	2	3	1				
	Compass		3		2		1		
	Odometry	8	12	6	14	5	11	3	2
	Pressure	1	3	1	2	1	1		
	Contact	4	3	2	3	2	4		
	Velodyne	4	5	4	2	3	1		
	`point_cloud`	1	4		1	3	1		1
	Radar	1	2	1	1	1	1		
	Geolocation	4	9	6	8	3	3		
	Temperature		2	2	1	1			
	Battery Monitor	4	3	3	2	2	1	5	
	Joystick	5	2	3	9	2	3	3	
		Phys. HW	**Sim./Log**	**Phys. HW**	**Sim./Log**	**Phys. HW**	**Sim./Log**	**Phys. HW**	**Sim./Log**
Actuators	Movable base	11	12	8	13	9	9	4	2
	Servo	1	1	2					
	Lights	1	12	1	9	1	6		
	Arm	4	6		7	2	3		1
	Gripper	4	3	1	5	2	2		1
		Phys. HW	**Sim./Log**	**Phys. HW**	**Sim./Log**	**Phys. HW**	**Sim./Log**	**Phys. HW**	**Sim./Log**
Simulators	Gazebo		19		18		15		1
	Unity		1		1				
	Playback		3		2				
		Phys. HW	**Sim./Log**	**Phys. HW**	**Sim./Log**	**Phys. HW**	**Sim./Log**	**Phys. HW**	**Sim./Log**
Robot Types	Baxter	1		1		1			
	PR2		2		3		2		
	WAM	1	1		1				
	JACO	1							
	Turtlebot	1				1			
	DaVinci			1					
		Phys. HW	**Sim./Log**	**Phys. HW**	**Sim./Log**	**Phys. HW**	**Sim./Log**	**Phys. HW**	**Sim./Log**
Libraries	Rosbridge	7	3	8	3	9	2	12	2
	RViz	27	15	19	7	15	1		
	MoveIt!	1	4		2	1			
	Transform Library (tf)	39	28	32	17	26	15	4	2
	Fiducial Libraries	1	2	1	2		1		
	`web_video_server`	2	1	2	1	1	2	2	1

Sensors: Sensors found in our scan included cameras, laser range finders, barometric pressure sensors, GPS devices, tactile sensors, and compasses. While camera topics appear to have the most standardization in terms of topic names (*e.g.* `camera_info` or `image`), we found other viable search terms as well. Some example sensors and their search terms included depth cameras (`depth_registered`), LIDAR (`velodyne`, `point_cloud`), biometric pressure sensors (`baro`) used on drones to control altitude, and various odometry sensors (`odom`, `odometry`).

Actuators: Actuator topic names are much more eclectic. We found several standard topics to indicate an arm or joint that moves (`joint_trajectory`), as well as grippers (`gripper`), and libraries to play sounds (`sound_play`). As moving robots often use heartbeat signals for safety reasons, we also considered topics matching `heartbeat` to indicate the presence of an actuator. We also found a number of one-off actuator topics: for example, the topic `inceptor_command` seemed likely to refer to a topic that sends flight commands; the topic `flystate2phidgetsanalog` seems to send voltages for controlling a faux-fruit fly used in biology experiments [30].

Simulators: We identified many ROS instances that appeared to be connected to simulated robots. The most obvious were connected to the Gazebo simulator [4]. We also found groups that appeared to be using the Unity Game Engine [31], the game TORCS [32] (`torcs_ros`), and the Stage simulator [33]. In this category we included hosts with parameters `use_sim_time` or `fake`, which indicate simulated inputs.

Running a simulator with an open ROS master does not pose the same physical risk as a real robot. However, given the complexity of a ROS system, and the ability to connect to other nodes, in many programming languages, it seems likely that this configuration still poses a threat: the machine can be compromised through a remote exploit, such as a buffer overrun in a ROS node.

Common Libraries: We also report evidence of use of the most common libraries. Examples include MoveIt! [34] (`move_group`), Rosbridge [16], fiducial marker libraries such as AprilTags [35] or AR Tags [36].

B. *Persistent Hosts vs. Temporary Hosts*

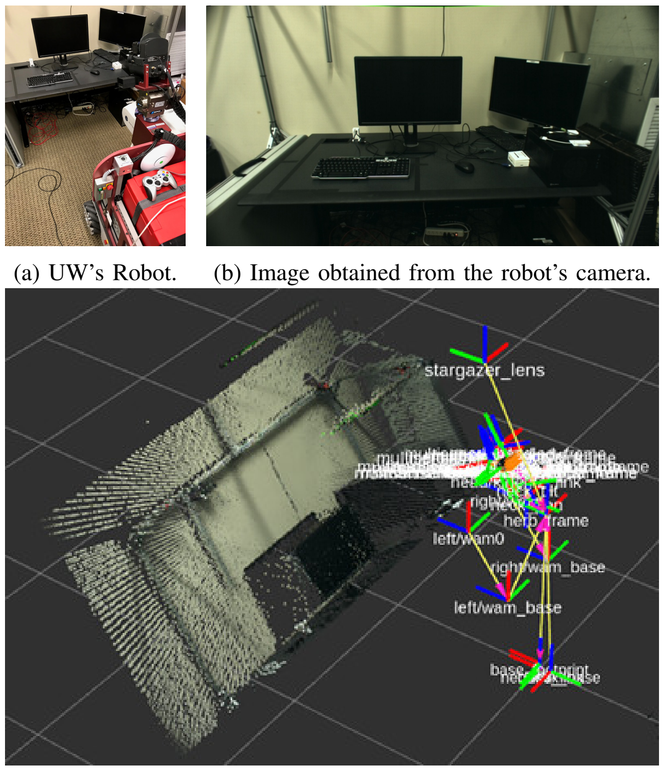
We hypothesized that many ROS instances are not always online, and instead are only available (and visible to scans) during intermittent periods of usage. Since each of our scans took place over 7 days, intermittent hosts may not have been detected if they were not online when their IP address was scanned. Conversely, a single host may have been detected multiple times in one scan if its IP address changed during the scan interval. We examined our results to find hosts that appeared in multiple scans and those appearing in only one scan. Comparing, however, host responses between scans is non-trivial. A single ROS instance may use different IP addresses, or DNS names, over time due to dynamic IP allocations (via DHCP) or physical movement between networks, and may provide different topic and parameter data based on the host's activity when scanned.

We devised two methods to provide an approximation of the number of hosts appearing in multiple scans. First, some results could be matched to a single host using its local machine hostname, usually found in the parameter list. When a machine name was provided, we also matched hosts with the same IP addresses if they contained a percentage of similar topics. Further discussion of our matching techniques is provided in the extended version [6].

We found that 41 hosts were present in all three ROS master scans, 48 hosts were found in two scans, with the remaining hosts only observed once. Of the hosts appearing multiple times, 25 were grouped into different categories (cf. Table I) in each scan. This demonstrates how a single ROS instance may have different usage patterns over time. While this poses a challenge to classification, it also highlights a further privacy risk, as the topic data can demonstrate a researcher's usage patterns. To provide one example, we found one machine name that appeared three different times in the scan with three distinct IPs: one with a DNS name for a university research lab, which showed topics indicating a simulator; one in another network at the same university, also with simulator topics; and another IP in a consumer ISP located near the university, which displayed no topics, indicating the ROS core was empty.

VI. Case Study: Mobile Manipulator

We present one case study of one of the robots identified in our scans to provide a sense of the potential damage that could occur if a robot was compromised. Further case studies are available in the extended version [6].



(a) UW's Robot. (b) Image obtained from the robot's camera.

(c) View from RVIZ showing 3D point cloud and TF tree.

Fig. 2: Images obtained from the robot's camera.

Our scans identified the mobile manipulator robot Herb2, at the University of Washington. This robot was running topics that indicated the presence of a multisense RGB-D sensor, a node that appears to perform speech production/generation, as well as a service for moving its neck. The robot also had parameters indicating the presence of controllers for arms, but the arm nodes were not available during our scans.

After contacting the robot's operators, we obtained their permission to connect to it remotely. During our test, we attempted to gather data from the robot's sensors and move its actuators. For safety reasons, we coordinated with the robot's operators to ensure the robot was physically monitored during the test. To mimic the capabilities of a remote attacker, we made an effort to utilize only the data we obtained from the ROS master port, as well as public documentation on ROS packages and services, when sending commands.

We were able to use the camera of the ROS master, as shown in Figure 2, to view images at 1.5Hz. Note that camera feed can be obtained without the knowledge, or consent, of the robot owner or operator (although we did have knowledge and consent in this case), potentially violating the privacy of people working with the robot. Additionally, an attacker could perform reconnaissance by viewing sensor information: *e.g.* run a face tracker or a movement sensor on the images, to determine dangerous times to move the robot. We were also able to view joint angles, obtained from the TF tree, as well as a 3D point cloud, visible in Figure 2c.

We tried to actuate the robot by playing sounds and moving its neck. The robot used a custom service for performing speech synthesis. In ROS, calling a service requires knowledge of its *service files*, which describe the data format it accepts. As the source code was publicly available [37], we were able to play sounds on the robot by installing and writing a script to send the action. Many other services have similarly well-known topics, *e.g.* `ros_control` [38], which could be accessed in this way.

Moving the robot's neck presented a more significant challenge: the service to move it used custom software not released by the lab. As a result, although we could see that the service existed, we did not know the format of the service file to specify calls. ROS does not permit service calls unless the MD5 checksums of the service files between the two nodes match. Since MD5 is not a cryptographically secure hash [39], and instead is only useful for checking data integrity, it may be possible for a determined attacker to reconstruct the service files with partial knowledge of the data format. In our test, we left this component to future work and simply asked the lab to provide the service files, which we used to construct a ROS package and actuate the neck. Notably, the lab informed us that we could have damaged the robot by actuating the neck to certain out-of-bounds values.

VII. Recommendations for Improved Security

We contacted the owner(s) of the hosts identified as ROS instances to notify them of our findings and provide recommendations for how users can fix their security. To help promote awareness, we have also contributed a page to the ROS wiki[2] outlining recommendations on how ROS can be used in more secure configurations.

Long term, we advise ROS users–especially those building Internet-facing applications—to investigate migrating to ROS2 with security extensions, which offers robust security features at both the application and transport layers. In the meantime, our recommendations focus on securing ROS1 at the network layer to provide a stopgap solution for the most critical security issues.

Detecting Exposure: At minimum, we recommend that users inspect their network exposure using tools such as `nmap` [40] and `nc` [41] from a host *outside* their organizational network. We advise researchers to consult organizational policies on conducting scans, and potentially coordinate with their network administrators, before scanning many addresses.

Firewall: One way to secure ROS instances against unauthorized access is to use a firewall to prevent exposure of ROS services to the public Internet. We recommend that ROS users define a *trust zone* on a network where ROS traffic is permitted to perform tasks, and take steps to restrict network access to ROS hosts outside this segment.

A common—but insufficient—practice to create a small private network is to use a common wireless router, or other device, to provide Internet access to a private IP range (*e.g.* 192.168.*.*) using NAT. By design, NAT blocks incoming connections from outside the network, meaning a ROS master behind a NAT would not be visible to the Internet by default. While NAT provides a first step to blocking external traffic, *NAT is not a security apparatus*: an internal host that makes external connections can still leak data, and NAT implementations can be misconfigured or, in some cases, exploited to open ports for outside access [42, 43, 44]. Therefore, we recommend using a more complete firewall solution, or an isolated network. We encourage researchers to coordinate with their organization's IT services to develop an effective solution. At minimum, traffic on the ROS master port should not be permitted from outside hosts. ROS traffic on other services is difficult to block, since port numbers are allocated dynamically by the master. For client machines using ROS, we suggest using an OS-level firewall (*e.g.*, Netfilter [45]) to restrict incoming traffic except on trusted networks. *This is especially important for ROS users on laptops or mobile devices*, which may operate from both trusted and untrusted networks.

VPN: In some cases, it may be necessary for a robot to be operated outside a trusted network. Remote access can be provided using a Virtual Private Network (VPN). If available, a VPN provides a comprehensive way to permit only authorized hosts to access a ROS master. VPNs are considered the best practice for securing a ROS system by the Open Source Robotics Foundation and were configured on every PR2 that shipped.

Rosbridge: Rosbridge is an alternative interface to ROS that provides certain security benefits. Rosbridge acts as a proxy for the ROS master API, allowing users to specify *protected* topics that will not be served to clients. Further, it provides an optional authorization mechanism using MACs [46] providing a way to define flexible access control policies.

VIII. Conclusion

Though a few unsecured robots might not seem like a critical issue, our study shows that a number of research robots are accessible and controllable from the Internet, demonstrating a risk to user safety and privacy. It is therefore important to develop mechanisms for using these powerful tools, while also securing them from malicious actors. We hope that more roboticists will become aware of the security issues involved as a result of our work, and take steps to secure their platforms. As robots move out of the lab and into industries and home settings, the number of units that could be subverted is bound to increase—recent attacks on the Internet infrastructure from home devices serve as a clear warning. Consequently, it becomes imperative to provide adequate security for devices that can change not just the virtual world, but the physical world as well.

IX. Acknowledgements

We would like to thank Sidd Srinivasa and Youngsun Kim for help with our proof-of-concept "attack." We also thank the network operations and IT staff at Brown University, who provided the network resources for our scans.

[2]`https://wiki.ros.org/Security`

References

[1] D. Quarta, M. Pogliani, M. Polino, F. Maggi, A. M. Zanchettin, and S. Zanero, "An experimental security analysis of an industrial robot controller," in *IEEE Symposium on Security and Privacy*. IEEE, 2017, pp. 268–286.

[2] F. Maggi, D. Quarta, M. Pogliani, M. Polino, A. M. Zanchettin, and S. Zanero, "Rogue robots: Testing the limits of an industrial robot's security," Trend Micro, Politecnico di Milano, Tech. Rep., 2017.

[3] 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 Software*, vol. 3, no. 3.2. Kobe, 2009, p. 5.

[4] N. Koenig and A. Howard, "Design and use paradigms for gazebo, an open-source multi-robot simulator," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2004, pp. 2149–2154.

[5] Rethink Robotics, "Baxter," Accessed: February 27, 2019. [Online]. Available: http://www.rethinkrobotics.com/baxter/

[6] N. DeMarinis, S. Tellex, V. Kemerlis, G. Konidaris, and R. Fonseca, "Scanning the internet for ros: A view of security in robotics research," *arXiv preprint arXiv:1808.03322*, 2018.

[7] Rethink Robotics, "Baxter SDK: SSH," Accessed: February 27, 2019. [Online]. Available: http://sdk.rethinkrobotics.com/wiki/SSH

[8] "Root access and ssh password," Baxter Research Robot Community mailing list, 2016. [Online]. Available: https://groups.google.com/a/rethinkrobotics.com/forum/#!topic/brr-users/acVIkCnk5Ow

[9] T. Denning, C. Matuszek, K. Koscher, J. R. Smith, and T. Kohno, "A spotlight on security and privacy risks with future household robots: attacks and lessons," in *International Conference on Ubiquitous Computing*. ACM, 2009, pp. 105–114.

[10] J. McClean, C. Stull, C. Farrar, and D. Mascareñas, "A preliminary cyber-physical security assessment of the robot operating system (ros)," *SPIE Defense, Security, and Sensing*, vol. 8741, p. 874110, 2013.

[11] C. Cerrudo and L. Apa, "Hacking robots before skynet," in *PacSec Applied Security Conference*, 2017.

[12] B. Dieber, B. Breiling, S. Taurer, S. Kacianka, S. Rass, and P. Schartner, "Security for the robot operating system," *Robotics and Autonomous Systems*, vol. 98, pp. 192–203, 2017.

[13] SROS, "Secure robot operating system," Accessed: February 27, 2019. [Online]. Available: http://wiki.ros.org/SROS

[14] R. White, H. I. Christensen, and M. Quigley, "Sros: Securing ros over the wire, in the graph, and through the kernel," *arXiv preprint arXiv:1611.07060*, 2016.

[15] T. Dierks and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2," Tech. Rep. 5246, August 2008. [Online]. Available: https://tools.ietf.org/html/rfc5246

[16] C. Crick, G. Jay, S. Osentoski, and O. C. Jenkins, "Ros and rosbridge: Roboticists out of the loop," in *ACM/IEEE International Conference on Human-Robot Interaction*. ACM, 2012, pp. 493–494.

[17] R. Toris, J. Kammerl, D. V. Lu, J. Lee, O. C. Jenkins, S. Osentoski, M. Wills, and S. Chernova, "Robot web tools: Efficient messaging for cloud robotics," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2015, pp. 4530–4537.

[18] ROS, "ROS2," Accessed: February 27, 2019. [Online]. Available: https://index.ros.org/doc/ros2/

[19] Object Management Group, "Data Distribution Service Specification Version 1.4," Tech. Rep., 2015. [Online]. Available: https://www.omg.org/spec/DDS/1.4

[20] ——, "DDS Security Specification Version 1.1," Tech. Rep., 2018. [Online]. Available: https://www.omg.org/spec/DDS-SECURITY/1.1/

[21] ROS, "SROS2," Accessed: February 27, 2019. [Online]. Available: https://github.com/ros2/sros2

[22] ——, "ROS Master API," Accessed: February 27, 2019. [Online]. Available: http://wiki.ros.org/ROS/Master_API

[23] ——, "Slave api," Accessed: February 27, 2019. [Online]. Available: http://wiki.ros.org/ROS/Slave_API

[24] Z. Durumeric, E. Wustrow, and A. J. Halderman, "Zmap: Fast internet-wide scanning and its security applications." in *USENIX Security Symposium*. USENIX, 2013, pp. 47–53.

[25] A. Mirian, Z. Ma, D. Adrian, M. Tischer, T. Chuenchujit, T. Yardley, R. Berthier, J. Mason, Z. Durumeric, A. J. Halderman *et al.*, "An internet-wide view of ics devices," in *Annual Conference on Privacy, Security and Trust*. IEEE, 2016, pp. 96–103.

[26] ROS, "URDF," Accessed: February 27, 2019. [Online]. Available: http://wiki.ros.org/urdf

[27] Open Source Robotics Foundation, "Turtlebot," Accessed: February 27, 2019. [Online]. Available: https://www.turtlebot.com

[28] Barrett Technology, LLC, "Wam arm," Accessed: February 27, 2019. [Online]. Available: http://www.barrett.com/wam-arm

[29] P. Kazanzides, Z. Chen, A. Deguet, G. S. Fischer, R. H. Taylor, and S. P. DiMaio, "An open-source research kit for the da Vinci® Surgical System," in *IEEE International Conference on Robotics and Automation*. IEEE, 2014, pp. 6434–6439.

[30] florisvb, "Kinefly," Accessed: February 27, 2019. [Online]. Available: https://github.com/ssafarik/Kinefly

[31] U. G. Engine, "Unity game engine-official site," Accessed: Febrary 27, 2019. [Online]. Available: https://unity3d.com

[32] fmirus, "Code for interfacing torce1.3.7 with ros," Accessed: February 27, 2018. [Online]. Available: https://github.com/fmirus/torcs_ros

[33] B. Gerkey, R. T. Vaughan, and A. Howard, "The play-

er/stage project: Tools for multi-robot and distributed sensor systems," in *International Conference on Advanced Robotics*. IEEE, 2003, pp. 317–323.

[34] S. Chitta, I. Sucan, and S. Cousins, "Moveit![ros topics]," *IEEE Robotics & Automation Magazine*, vol. 19, no. 1, pp. 18–19, 2012.

[35] E. Olson, "Apriltag: A robust and flexible visual fiducial system," in *IEEE International Conference on Robotics and Automation*. IEEE, 2011, pp. 3400–3407.

[36] M. Fiala, "Artag, a fiducial marker system using digital techniques," in *IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 2005, pp. 590–596.

[37] personalrobotics, "Talker," Accessed: February 27, 2019. [Online]. Available: https://github.com/personalrobotics/talker

[38] ROS, "`ros_control`," Accessed: February 27, 2019. [Online]. Available: http://wiki.ros.org/ros_control

[39] CERT, "Vulnerability Note VU836068: MD5 vulnerable to collision attacks," 2008. [Online]. Available: https://www.kb.cert.org/vuls/id/836068

[40] G. F. Lyon, *Nmap network scanning: The official Nmap project guide to network discovery and security scanning*. Insecure, 2009.

[41] "netcat," Accessed: February 27, 2019. [Online]. Available: http://netcat.sourceforge.net/

[42] R. Davis, "The myth of network address translation as security," F5 Solutions, Tech. Rep., 2016. [Online]. Available: https://f5.com/resources/white-papers/the-myth-of-network-address-translation-as-security

[43] M. Baugher and V. Lortz, "Home-network threats and access controls," in *International Conference on Trust and Trustworthy Computing*. Springer, 2011, pp. 217–230.

[44] D. Garcia, "Universal plug and play (UPnP) mapping attacks," *DEFCON-19*, 2011.

[45] Netfilter, "iptables documentation," Accessed: February 27, 2019. [Online]. Available: http://ipset.netfilter.org/iptables.man.html

[46] R. Toris, C. Shue, and S. Chernova, "Message authentication codes for secure remote non-native client connections to ros enabled robots," in *IEEE International Conference on Technologies for Practical Robot Applications*. IEEE, 2014, pp. 1–6.