ROSploit: Cybersecurity tool for ROS

Sean Rivera†, Sofiane Lagraa†, and Radu State†
† SnT, University of Luxembourg
firstname.lastname@uni.lu

***Abstract*—Robotic Operating System(ROS) security research is currently in a preliminary state, with limited research in tools or models. Considering the trend of digitization of robotic systems, this lack of foundational knowledge increases the potential threat posed by security vulnerabilities in ROS. In this article, we present a new tool to assist further security research in ROS, *ROSploit*. *ROSploit* is a modular two-pronged offensive tool covering both reconnaissance and exploitation of ROS systems, designed to assist researchers in testing exploits for ROS.**

***Index Terms*—Robotics, ROS, Security, Threat Model, Cybersecurity**

Introduction

As robotic systems become standard throughout industry and research, there is a greater concern about security vulnerabilities. One of the primary frameworks for robotic development, Robotic Operating System (ROS) [1], is used widely in the field. With the addition of ROS-Industrial (ROS-I) [2], ROS has moved from a purely research tool into an industry standard middleware system, with various versions of ROS systems found in the field. As a tool that was initially developed for academic research, security was not part of the initial ROS design [3]. As such the core implementation of the ROS system is highly insecure and the previous research into the security of ROS systems has been stymied by the lack of a consistent set of tools [4]. A consistent set of security tools is crucial for security research as it provides foundation for further development. Additionally, the existence of such tools allows a developer to test their systems against similar real-world attacks and more effectively develop secure reliable systems.

This paper has the following major contributions:

- We propose a new model of security for ROS systems, that future work can use as a foundation for ROS security analysis.
- We propose *ROSploit* a new exploitation tool for simulating attacks against ROS systems and we compare it with ROSPenTo, a similar new exploitation tool under development.

Security for ROS

In SROS: Securing ROS over the wire, in the graph, and through the kernel [5], the authors proposed SROS, a library for ROS ecosystem to support modern cryptography and security measures to address existing vulnerabilities. In SROS, all network communication is encrypted using Secure Sockets Layer (SSL), or more specifically TLS. The encryption is done through the use of Public Key Infrastructure (PKI), where each ROS node is provided an x.509 certificate, equivalently an asymmetric key pair, signed by a trusted certificate authority. These results indicate security vulnerabilities in ROS, requiring additional libraries to ensure security in vital ROS systems.

However, the most common vulnerabilities of ROS, are presented in Security for the Robot Operating System [3]. The authors highlighted the security issues in ROS with several possible attack vectors on a ROS application such as unauthorized Publishing (Injections), unauthorized data access, and denial of service (DoS) attacks on specific ROS nodes. They showed how to secure ROS on an application level and describe a solution which is integrated directly into ROS core. Additionally they built a tool to assist them in looking for additional attack vectors, ROSPenTo.

Threat Model

In order to conduct a security analysis of ROS applications, an extended threat model was constructed based on the attack vectors discussed in Security for the Robot Operating System [3], mainly focusing on the following attacks: Unauthorized publishing, Unauthorized data access, and a packet flooding DOS as a first pass analysis into the security of ROS. For our research, we extended these attacks into broader categories and explored what other attacks would be possible for those categories.

For this threat model, two distinct types of threats, internal and external, and their primary risks were considered. For the purposes of this paper, internal threats are defined as attackers who have access to the underlying Linux system, with access to legitimately spawn new nodes onto the system and have at least partial network access. Insider threats can come in a wide variety of forms including; malicious manufactures, compromised nodes, disgruntled developers with accessor even simple software bugs. For the purpose of this paper, we define external threats as any sort of threat that does not have partial access to the underlying Linux system. External threats cannot launch new nodes on their own.

By referencing similar threat models for normal Internet-connected systems [6], it is evident that external threats will likely be the vast majority of threats that ROS systems will face. When analyzing external threats, several areas of known potential vulnerabilities for ROS system were discovered. These include sensor input tampering with available IP flows [3]. Given the existence of remote parameter control for ROS services, an attacker may be able to escalate their attacks to remote code execution. This would allow the attacker to load

malware on the robot, which would elevate the attacker to be an internal threat instead of external.

ROSPLOIT

ROSploit is designed to assist security researchers in their analysis of ROS systems and compare it to the other tool currently in development: ROSPenTo [7]. This system can be split into two separate components: reconnaissance and exploitation.

The reconnaissance components of the system integrate with the existing research tool NMAP [8] as a set of NSE scripts which can be enabled when scanning a suspected ROS system. As of the publishing of this article, there are two high-level scripts implemented as part of this system. The first script is a master node scan, which focuses on pulling information from the ROS master, while the second is an addition to the normal NMAP wide port scan, allowing NMAP to identify various ROS nodes during the course of the port scan. The master scan script runs only for the master port(normally 11311) of ROS. This scan calls the *getSystemState()* function, a part of every ROS system. Once it calls the function it is given a list of every single running node topic and service on the ROS system, which it then parses for further user examination. The wide port scan can be run as a part of any scan of the system. It can determine if an open TCP port is a ROS node, part of ROS master, or a ROS service by sending normal XMLRPC requests to the nodes and monitoring the responses it receives. If it fails to receive a response from the XMLRPC requests, it attempts to send a TCPROS subscription request to the open port. Using the response to that subscription request, the system can determine if the node is a topic or a service and if it is running TCPROS. Unlike ROSPenTO we are able to analyze a system with an unknown master node, and able to partially analyze a ROS system based on port numbers. In Figure 1 we demonstrate the node scan portion of ROSploit and compare it with the results of the rosgraph tool in order to demonstrate the effectiveness of the scanning tool. The system correctly identifies the open ports as well as the name of the node running. The script flags these by describing them as topic ports, and then naming the publisher. This indicates that we can effectively scan the whole system, including the debugging add-ons.

The exploitation component of the system is developed in Python as a set of modular exploit components. It is similar to Metasploit in design as it is a modular system of scripts that contains canned exploits to be run against an already scanned target, and it depends on the reconnaissance half of the system to determine which parameters are needed. In order to facilitate easy development of various exploits, we provide interfaces to the entire ROS middleware. This allows the user to run *ROSploit* without having to fully install ROS, and it opens up new potential exploit areas where attacks can target the underlying system by, as an example, sending malformed TCPROS messages. A user can select which attacks they with to run as command-line arguments. As an example, in order to run a MiTM attack, a user would give the name of the topic and the two nodes to the script. The script would then insert itself into the communication between the two nodes on the selected topic. From there use user is free to modify the communication as needed. As compared to ROSPenTo we are capable of performing every action that it can related to topics and nodes, though we do not have scripts to interact with the parameter server yet.

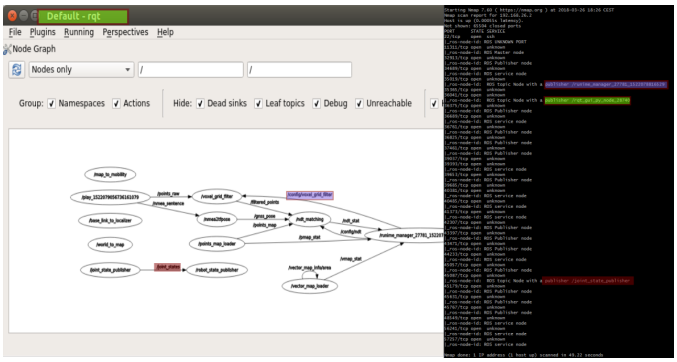

Fig. 1. An example version of the NMAP port scan compared to the plot generated by the internal ROS graph tool. The highlighted names were included to demonstrate that they nodes were successfully scanned while the rest of the names were omitted due to brevity.

CONCLUSION AND FUTURE WORKS

In this paper we introduce *ROSploit*, a new research tool modeled after NMAP and Metasploit for modeling and exploiting vulnerabilities in ROS. This tool is designed to provide a framework for further research into security for ROS systems allowing providing a flexible platform for the development of future research. For future work, *ROSploit* will be extended to support additional functionality, newer attacks, as well as allow multi-stage exploits. Additionally, we will extend the tool to support ROS2.

REFERENCES

[1] "Community metrics report 2017," http://download.ros.org/downloads/metrics/metrics-report-2017-07.pdf, accessed: 2018-06-29.

[2] "Ros-industrial overview," http://wiki.ros.org/Industrial, accessed: 2018-06-29.

[3] 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. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0921889017302762

[4] A. J. A. Wang, "Information security models and metrics," in *Proceedings of the 43rd Annual Southeast Regional Conference - Volume 2*, ser. ACM-SE 43. New York, NY, USA: ACM, 2005, pp. 178–184. [Online]. Available: http://doi.acm.org/10.1145/1167253.1167295

[5] R. White, H. I. Christensen, and M. Quigley, "SROS: securing ROS over the wire, in the graph, and through the kernel," *CoRR*, vol. abs/1611.07060, 2016.

[6] M. Jouini, L. B. A. Rabai, and A. B. Aissa, "Classification of security threats in information systems," *Procedia Computer Science*, vol. 32, pp. 489 – 496, 2014, the 5th International Conference on Ambient Systems, Networks and Technologies (ANT-2014), the 4th International Conference on Sustainable Energy Information Technology (SEIT-2014). [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1877050914006528

[7] "Hacking a mir robot with rospento." [Online]. Available: https://bernharddieber.com/post/mir-hacking-video/

[8] Fyodor, "The art of port scanning," https://nmap.org/p51-11.html, accessed: 2018-06-29.