

Enhancing the Security of IoT Enabled Robotics: Protecting TurtleBot File System and Communication

Michael Horton
Georgia Southern University
Statesboro, GA
Mikep.horton@gmail.com

Lei Chen
Georgia Southern University
Statesboro, GA
lchen@georgiasouthern.edu

Biswanath Samanta
Georgia Southern University
Statesboro, GA
bsamanta@georgiasouthern.edu

Abstract—The recently emerged Internet of Things (IoT) has already become largely prevalent in today's society. However, strong and fundamental security practices have not been applied to fully protect these systems, partially negating the benefits of IoT. The concept has grown so much in recent years that this lack of a security infrastructure can no longer be ignored. Recent developments in the field of IoT and robotics have proven invaluable in building a solid framework for what lies ahead. This paper focuses on the examination and enhancement of security between IoT enabled robots, specifically in this project, TurtleBots, and the cloud infrastructure supporting them by providing a combined set of security best practices for robotic file systems and communications.

Keywords—Security; Cloud Robotics; Linux; TurtleBot; IoT; Ubuntu; ROS; UFW

I. INTRODUCTION

The Internet of Things (IoT) has become the subject of much research in recent years. IoT can be loosely described as an interconnection between all physical items that have been given the capacity to communicate. However, IoT is a concept that is so large and ever changing, that even experts in the field do not always agree on a universal definition [1, 2]. One can understand why research has dramatically increased in this field. In IoT, real world physical objects become virtual entities in the cyber world and are enhanced with the sensing, processing, and self-adapting abilities to interact [3].

IoT is already prevalent in today's society, from thermostats, to smart homes and security systems, to wearables and vehicles, and the concepts have entered the lives of everyday citizens. It can also be found in intelligent transportation, smart grids and smart buildings. However, because of the lack of standards, specifications, and architecture development, it hasn't fully reached its intended scale [4].

One of the most prolific uses of IoT today is in robotics. The recent developments in this field that have enabled the creation of more sophisticated robots and robotic devices

have proven invaluable in building a framework for the IoT and showcasing what the possibilities are. As robotics become more reliable and less cost prohibitive, they will continue the trend and enter everyday lives just as much as other IoT devices. Nevertheless, without strong security and privacy foundations, any benefits provided by IoT will be outweighed by security breaches, attacks and malfunctions [5, 6].

The IoT concept as a whole has grown exponentially recently, so much so that the need for security is urgent. IoT security however, does not have a simple solution. It is a multidimensional subject that combines information security, network security, infrastructure security, and management security [3]. Furthermore, IoT security encompasses a wide range of tasks, which can include data confidentiality, integrity, availability, privacy, etc. [7]. This issue is further enhanced because of its highly distributed nature, plus its use of fragile technologies in public areas which create weak links [5]. With the IoT, each connected device is a potential doorway into the infrastructure [7].

The Internet of Things is a hybrid and heterogeneous network, which requires multi-faceted security solutions [7]. Because of this, there has been research in many related areas such as algorithms, authentications, access control, trust, and governance frameworks [3]. Under the research umbrella, this particular project focuses on the examination, development, and enhancement of a secure relationship between a private cloud-based server infrastructure and its associated IoT enabled robots. This will require not only security at the robotics level, but infrastructure and communication security, and governance as well.

The rest of this paper is arranged as follows. Research questions are raised and the investigation methodology is presented in Section 2. Section 3 addresses the results and analysis in terms of the security in three main components: IoT robots, the cloud infrastructure, and the communications between them. These results are discussed in Section 4. Conclusions are drawn and plans for future work are in Section 5.

II. RESEARCH QUESTIONS AND METHODOLOGY

A. Introduction to Methods

IoT security research is an endless variety of topics covering a multidisciplinary framework. This specific study focuses on IoT enabled robots and a cloud processing environment. The aim of this study is to answer the following questions:

1. How can the data located on the robot itself be protected in the event of theft or other unauthorized physical access?
2. How can the cloud infrastructure be protected from malicious attacks from inside or outside of the organization?
3. How can we ensure that the robots only accept connections and instructions from approved servers and administrative devices?

Processes and practices will be tested and implemented to answer these questions during this study. The security practices tested here will also need to ensure that not only individual objects are managed, but the group as a whole will be accounted for.

B. Hardware and Equipment

A virtual test lab was set up prior to implementation. This lab consists of four Oracle VirtualBox virtual machines (VM). Three of them are used to house the Robot Operating System (ROS) implementation. Each VM consists of 1 GB RAM and an 8 GB hard drive. The VMs were named NSF-TurtleBot, NSF-Processor, and NSF-Controller. Each VM was set up with a 64-bit version of the Ubuntu 14.04 Operating System, codename trusty. A fourth VM, set up with the same specifications and named NFS-Intruder, is used for attacking the other ROS enabled devices and for logging and reporting the successes and failures.

ROS Indigo is installed on all of the virtual machines, one as the robot simulator, one to handle the commands and processing, and one as the master to run the ROSCORE. This enables the testing of security features in a similar setup as the production environment before rollout to the actual units.

The robots used in this study are the open hardware and open software Kobuki TurtleBot 2's. These TurtleBots consist of a wheeled platform, a Microsoft Kinect camera, and an Asus netbook running Ubuntu Linux and ROS [8, 9]. The netbooks have 3.7 GB of RAM and a dual core, 1.1 GHz processor.

The test setup was built to partially mimic the production lab, which consists of four of these TurtleBots operating wirelessly, using the IEEE 802.11n protocol. There are 5 VMWare ESXi hosts and one Network Attached Storage (NAS) appliance in the lab. These hosts contain 25 VMs total, though not all are included in this particular study.

Physical equipment between the server farm and the robots includes an Ethernet switch and a wireless router. The switch manages multiple VLANs to contain traffic to their respective subnets. Only one of these VLANs will be examined during this project.

C. Local Data Protection

The first step in protecting the local data on the robots is the operating system. Ubuntu was installed in the test lab using an encrypted version, which means it protects the data if the computer is lost or stolen. The user is required to enter a passphrase each time the computer starts before the OS boots. The home folder of the main user is also encrypted by a separate passphrase.

By default, Ubuntu does not allow OS logins as root. The root user remains disabled and a strong passphrase is required for the enabled user. Only a single user will be enabled at this time and any non-essential users will be removed, including the Guest Session account. This account was removed by creating a lightDM (the display manager running in Ubuntu Linux) configuration file with the Ubuntu user session and unity greeter enabled, and the tag, allow-guest, set to false. The remote login option was also removed by setting the tag, greeter-show-remote-login to false.

The next step taken for protecting local data was to prevent single user access and recovery mode access from GRUB (GNU Grand Unified Bootloader). This required editing the `/etc/default/grub` file and setting the `GRUB_DISABLE_RECOVERY` to true by removing the comment qualifier. As an added protection, any attempts to edit the GRUB on startup will also require a password. This was enabled by adding a user and password to the superusers group in the `00_header` file of `/etc/grub.d`. Finally, a password was set on each GRUB menu entry. This was accomplished by navigating in a terminal window to `/etc/grub.d` and running the command `sudo sed -I -e '/^menuentry /s/ {/ --users username {/ *' . This command uses sudo to give the user temporary super user rights and runs the sed stream editor. The -I trigger indicates that the file should be edited in place while the -e adds the script to the commands. /^menuentry is the text entry to look for with /s/ making it explicit. The line to add is specified by --users username and finally, the * indicates that this should be run for each file in the directory.`

A single TurtleBot was chosen in the production lab for initial testing, and the setup from the test lab was recreated on the attached netbook. All tested settings except full disk encryption were successfully implemented. Full disk encryption cannot be enabled after the operating system has been installed and since this would require a rebuild of the robot, it was decided to wait until a more acceptable time frame.

D. Cloud Infrastructure Security

In the virtual test lab, the test server VMs are installed with the same encrypted version of Ubuntu as the test lab TurtleBot node. This step is not as important here because the production servers will not likely be physically removed from the labs; though, it is a best practice overall. The production lab does not currently have any mobile workstations, but these will need to be considered if they are added in the future.

Essentially, the same steps were taken on the test lab servers as the test lab TurtleBot itself. The servers were configured with encrypted file systems and home folders. Any extraneous users were also removed, and finally, GRUB passwords were implemented and the recovery option disabled.

These same settings, minus the encryption, were also mirrored successfully on the server housing the ROSCORE and a VM sending instructions to the TurtleBot in the production laboratory.

The server farm is protected from malicious external access by the university's firewall and Intrusion Detection / Intrusion Prevention Systems (IDS/IPS). Furthermore, the production ESXi hosts and NAS are physically protected by a locked server cabinet in a locked laboratory.

E. TurtleBot Connectivity Security

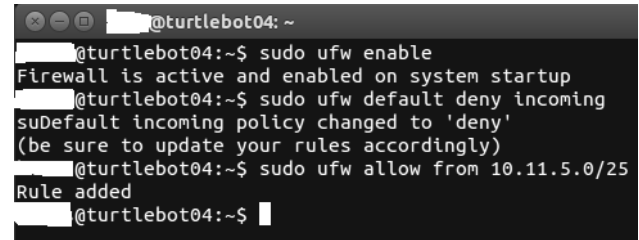
ROS itself does not provide much in the way of security. Therefore, the bulk of the connectivity safeguards will be handled on the Ubuntu OS. When an ROS Master is started in the default setup, it is bound to TCP port 11311. The individual nodes use TCPROS or UDPROS to communicate with the master and the source port varies.

The first step in securing this connection is to implement Uncomplicated Firewall (UFW) to block connections from any machine other than the designated servers involved. These include the ROS master and any command or processing machines. Each of the other servers that communicate with the bot will also need UFW enabled with entries for the individual nodes, other servers, and workstations, to protect them from any malicious connections. The test lab implemented the command 'sudo ufw default deny incoming' to reject all but the explicitly allowed traffic. The firewall was then configured for access from the ROS Master using the commands 'sudo ufw allow from IP Address' to allow the ROS traffic and 'sudo ufw allow from IP Address port 22', which allows traffic from the specified IP address over TCP port 22. Port 22 is specified for Secure Shell (SSH) traffic, providing remote management.

To further protect the incoming traffic, MAC address filtering was enabled using the command 'sudo iptables -A INPUT -m mac --mac-source mac-address -j ACCEPT'. This command, run as a super user calls the iptables command using -A to append the specified rule to a table, in this case, INPUT. The -m parameter is used to match the

rule that we wanted. --mac-source specifies what machine we are allowing and -j states that if the pattern is matched, we jump to the action 'Accept' the traffic.

Fig. 1. UFW Rules Setup



```
@turtlebot04: ~
@turtlebot04:~$ sudo ufw enable
Firewall is active and enabled on system startup
@turtlebot04:~$ sudo ufw default deny incoming
suDefault incoming policy changed to 'deny'
(be sure to update your rules accordingly)
@turtlebot04:~$ sudo ufw allow from 10.11.5.0/25
Rule added
@turtlebot04:~$
```

Finally, the ROS file bashrc, was edited with a persistent means of setting up the ROS_MASTER_URI environment variable to prevent any misconfigurations.

The test lab settings were checked for basic reliability using the fourth virtual machine, NSF-Intruder, prior to implementation in the production environment. These tests were not all-inclusive, but served as a baseline to show the settings were correctly implemented.

After successful simulation in the test environment, these firewall and iptables commands were executed on the production machines (see Fig. 1). A spare desktop computer was used to attack the TurtleBot in order to test the security in place. Finally, Wireshark was installed on the TurtleBot to capture traffic.

The production lab contains a wireless router for network connectivity between the robots and the server farm, as well as any laptops or other devices that may be needed. The examined security settings were Wi-Fi Protected Access II (WPA2) and Temporal Key Integrity Protocol (TKIP) [10]. The router was set to broadcast on Wireless-N only and the Service Set Identifier (SSID) broadcast was set to disabled. The local firewall on the router was also enabled and multicast traffic filtered out.

It is recommended that the lab make the following changes: keeping WPA2 but enabling Advanced Encryption Standard (AES) instead of TKIP, lowering the power settings, and filtering proxy and abnormal Secure Shell (SSH) or Telnet traffic. Further recommendations are to only allow HTTPS access to the router and to designate the acceptable IP addresses for remote management. Lastly, Telnet should be removed as a router management option and SSH enabled in its place. These changes were agreed upon; however, the timing of the project prohibited immediate implementation.

III. RESULTS AND ANALYSIS

A. Protection of TurtleBot Data

The local operating system encryption in the test lab showed a complete lockdown of the system. Without the initial passphrase, the OS would not post at all. Furthermore, bypassing this by calling the boot menu also proved fruitless to an attacker as the recovery mode could

not be accessed once the GRUB protections were in place, and changes could not be made on the fly without the main user's password. Disabling the guest user session provided extra security even though the guest user does not have full control of the system by default.

Moving these potential security fixes to the production lab's TurtleBot showed the same level of inaccessibility, minus the encryption. The only bypass this researcher found was that using a USB drive or CD-ROM could still allow access to the physical components of the netbook. Since full-disk encryption was not implemented at this time on live TurtleBot, the home folder and file systems could be read by a bootable media OS.

B. Cloud Infrastructure Security

All security settings tested on the virtual machines in the production lab, as with the TurtleBots, showed a successful implementation. We did not have access to the University's IDS or Firewall, so no data could be obtained on potential attacks from outside of the organization. Fortunately, though, no issues have been brought to the attention of the lab's personnel.

C. TurtleBot Connectivity Security

The connectivity security testing was the main focus of this implementation. UFW setup on the TurtleBots, the ROS Master, and the other virtual machines involved in the project proved essential in blocking unwanted access. In Fig. 2, details are provided from the Syslogs of UFW showing unauthorized traffic being denied from reaching its intended destination. This traffic, SSH on port 22, was initiated from an unknown host without permissions to the system.

Uninterrupted communication is a key component of robotics; and as such, a concern in our research was the effect that running UFW would have on the performance of the machine while ROS was running tasks. System monitoring was used to carefully inspect CPU and network utilization. The CPU usage over time is detailed in Fig. 3, where a high intensity ROSTOPIC, a connection mechanism for sending ROS Messages, was running, along with UFW, while attempts were made to access the system from the attacking computer. The increase at 20:12:19 was caused by the instance of the ROSTOPIC beginning. The first connection attempt took place at approximately 20:12:34.

Fig. 2. Syslogs details showing unauthorized traffic denied by UFW

```
Apr 27 19:11:03 turtlebot04 kernel: [ 1269.301126] [UFW BLOCK] IN=wlan0 OUT=MAC=6c:71:d9:26:0a:7d:00:1f:3b:62:62:d7:08:00 SRC=10.11.5.216 DST=10.11.5.129 LEN=52 TOS=0x00 PREC=0x00 TTL=128 ID=600 DF PROTO=TCP SPT=55044 DPT=22 WINDOW=8192 RES=0x00 SYN URGP=0
Apr 27 19:11:09 turtlebot04 kernel: [ 1275.299093] [UFW BLOCK] IN=wlan0 OUT=MAC=6c:71:d9:26:0a:7d:00:1f:3b:62:62:d7:08:00 SRC=10.11.5.216 DST=10.11.5.129 LEN=48 TOS=0x00 PREC=0x00 TTL=128 ID=601 DF PROTO=TCP SPT=55044 DPT=22 WINDOW=8192 RES=0x00 SYN URGP=0
```

In Fig. 4, the specific processes and their percentage of use during this test can be found. The ROSTOPIC was running at the highest percentage with 9.69%. Compiz, a GUI rendering process, measured at 6.05%. X, which

includes system processes, including the UFW, averaged at 5.51% with the Gnome-System-Monitor just behind it at 4.56%. Lastly, an instance of Mozilla Firefox was running and averaged at 2.61% processing power. It is imperative that the security processes do not overconsume the machines resources to the detriment of the communications. Overall, the total percentage of processing power in use was 28.42%.

Fig. 3. ROSTOPIC TurtleBot CPU Usage over Time

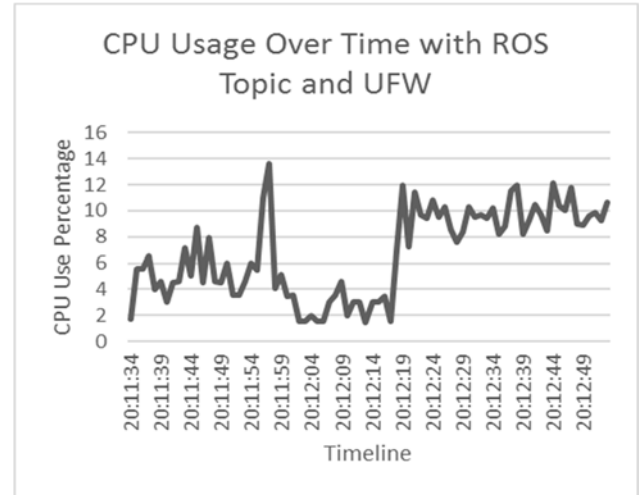
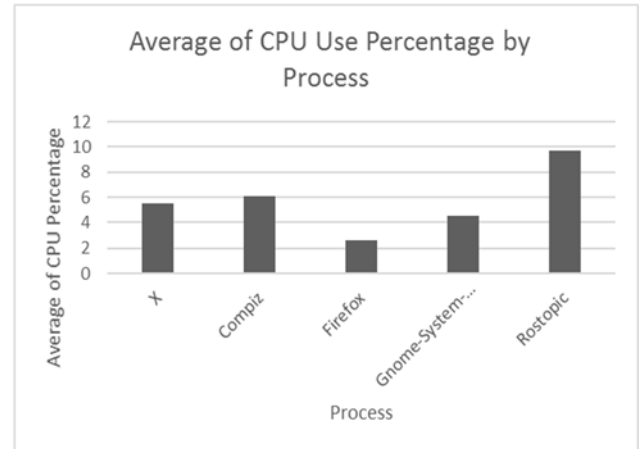


Fig. 4. Average TurtleBot CPU Usage by Process



IV. DISCUSSION

The setup explored in this project returned some very meaningful results. Encrypting the file system did not add much overhead to the OS, and though it was a slight nuisance and an added step to booting up the system, the security provided seems well worth the price. Along this same path, locking down GRUB's editing and recovery mode provides a means of protecting the data locally stored on the robots in case of theft or misuse. Disabling the Guest Session was not a step that brings much security enhancement overall, since the account is very limited in its access privilege. Yet, aside from accessing the file system, this did prevent any unauthorized users from

potentially damaging the equipment by running code and packages improperly.

The planning and setup of the test lab and the commands and security practices were sufficiently laid out so that they were easily migrated to the production lab. It is recommended that when time allows, the TurtleBot operating systems be rebuilt with the OS encrypted upon install and the home folders encrypted as well. The inability to have this piece in place may have caused a slightly less accurate CPU summary, though this setup was not proven to have any adverse effect in the test lab.

There was a conflict of timing when these tests were performed, as the wireless security settings could not be adjusted except in a scheduled maintenance period. The first available window for these adjustments was outside of the timeframe of this research paper. However, the security settings suggested are best practices and should be implemented as soon as possible. Furthermore, already having WPA2 in place is sufficient enough for the immediate future.

The performed firewall testing showed very low processor overhead and did not appear to impede communication between the controller, the master, and the TurtleBot. At the time of the attack, there was a 2% increase in CPU utilization; however, the initialization of the ROSTOPIC had already shown a peak usage of 12%, which was the same as when the attack occurred. There is not enough evidence from the tests performed to conclusively say that UFW had an impact on the CPU.

The security features here are recommended to be expanded to all applicable virtual machines and robots in the production lab on a permanent basis. New benchmark tests should also be run during that implementation to verify that there is not an increase in overhead when the entire system is secure.

V. CONCLUSIONS AND FUTURE WORK

Unfortunately, at its current version, ROS does not provide any security in and of itself. This is a major downfall that should be addressed by the developers. However, many security practices and processes can be implemented both at the operating system level and the communications level that will provide peace of mind to the operators.

There were a few things that this research found could benefit the production lab's security in the near future. Currently, bootable media poses a problem for the hardware in the lab. This needs to be addressed on the operating systems in place but authorized use of these devices must not be impeded. Also, even though SSH is used throughout the lab, no additional security, other than the default options, has been configured for its use. This should be examined in more detail in the near future.

Furthermore, an LDAP (Lightweight Directory Access Protocol) server can and should be set up as part of this lab's network to remediate the need for weakly secured local user accounts. This can provide a more rigorous environment where user access can be tightly controlled.

Lastly, a control website would be a beneficial feature for the lab, enabling the public and/or interested students the ability to interact with these robots securely and easily. This type of interface uses Linux-Apache-MySQL-PHP (LAMP) and HTTPS to access the system and many safeguards are available for protecting this type of data access.

Further future work will include expanding these and other security practices to cover the full production lab, which includes multiple IoT instances of varying types.

The security procedures detailed in this paper can be viewed as a combined set of best practices for protecting a basic Internet of Things network involving many nodes, virtual machines, multiple VLANs, and wired and wireless communication. Security is of the utmost importance when dealing with IoT, and the settings here can and should be scaled to the larger networks of the future.

ACKNOWLEDGMENTS

M. H. would like to thank Georgia Southern University for the use of their laboratory and Chris Reid for his knowledge of, and assistance in, the production lab environment. Also, Jenna, James, and Gwen for their love and never-ending support.

REFERENCES

- [1] K. Hodgson, "The Internet of [Security] Things," *SDM: Security Distributing & Marketing*, vol. 45, pp. 54-72, 2015.
- [2] P. de Leusse, P. Periolellis, T. Dimitrakos, and S. K. Nair, "Self Managed Security Cell, a security model for the Internet of Things and Services," 03/02/ 2012.
- [3] H. Ning, Liu, Hong, "Cyber-Physical-Social Based Security Architecture for Future Internet of Things," *Advances in Internet of Things*, vol. 2, pp. 1-7.
- [4] D. Yu and G. Junhua, "Research on Malicious Security Issues of Internet of Things for Mobile Internet," *Applied Mechanics & Materials*, vol. 687-691, pp. 1888-1891, 11// 2014.
- [5] R. Roman, P. Najera, and J. Lopez, "Securing the internet of things," *Computer*, p. 51, 2011.
- [6] G. Broenink, J.-H. Hoepman, C. v. t. Hof, R. van Kranenburg, D. Smits, and T. Wisman, "The Privacy Coach: Supporting customer privacy in the Internet of Things," 01/25/ 2010.
- [7] S. Li, T. Tryfonas, and H. Li, "The Internet of Things: a security point of view," *Internet Research*, vol. 26, p. 337, 03// 2016.
- [8] F. H. Nordlund, "Enabling Network-Aware Cloud Networked Robots with Robot Operating System : A machine learning-based approach," ed. Sweden, Europe: KTH, Radio Systems Laboratory (RS Lab), 2015.
- [9] L. D. Riek, "Embodied Computation: An Active-Learning Approach to Mobile Robotics Education," *IEEE Transactions on Education*, vol. 56, p. 67, 02// 2013.
- [10] D. Tepšić, M. Veinović, and D. Uljarević, "PERFORMANCE EVALUATION OF WPA2 SECURITY PROTOCOL IN MODERN WIRELESS NETWORKS," *Singidunum Journal of Applied Sciences*, pp. 600-605, 2014.