Building Secure Robot Applications

Murph Finnicum
University of Illinois

Samuel T. King
University of Illinois

Abstract

This position paper recognizes that general purpose robots will become increasingly common and argues that we need to prepare ourselves to deal with security for robot applications in an intelligent way. We discuss ways that robots are similar to traditional computing devices and ways that robots are different, and we describe the challenges that arise. We propose a framework for providing security for robot applications and we discuss three potential robot applications: a "fetch coffee" app, a "pretend to be a Labrador" app, and a "is my advisor in his office and available" app. We discuss some of the security needs of these applications and propose a few potential ways to address those security needs.

1 Introduction

Robots are being used for an increasingly wide range of applications. Much of the pioneering work on robotics focused on building industrial robots where robots helped automate portions of the manufacturing process. More recently, robots have seen increasing use for military applications [13], cleaning floors [8], mowing lawns [12], driving cars [17], and to help with rehabilitation by acting as realistic pets [11].

It is our position that robots should be viewed as general purpose computing devices and should be capable of running robot apps. By robot apps we *do not* mean Emacs and gcc, but rather *robots should support a secure and general purpose programming environment for controlling the robot itself.* In our vision we hope to make robots as easy to program as mobile phones, support multiple robot apps at the same time, and provide this functionality with appropriate security mechanisms in place from the beginning. Although many of the same security techniques we use on more traditional computing environments will certainly apply to robots as well (e.g., software least privilege and modularity), robots are fundamentally different than traditional computer systems in interesting and novel ways.

We argue that robot policies and interfaces are best described in terms of higher-level abstractions, and that the application framework should embrace these abstractions and deal with them as first-order concepts.

In this paper we discuss ways that robots are different than traditional computer systems and the security challenges that arise from these differences, and we posit a few directions that this new research area might take.

2 Challenges

Robotic platforms are built on top of traditional computing systems, and thus are vulnerable to the same security issues that they are [3]. However, robotics provide a fundamentally different platform than traditional computer systems that brings with it an entirely new set of security issues.

In this section we discuss ways that robots are similar to traditional computing platforms, and ways that robots are different than traditional computing platforms and the challenges that arise from these differences.

2.1 Similarities

Many robotic platforms use commodity hardware and software as components. For example, our robot, Isaac (Figure 1), uses a netbook running Linux as its main control unit. In order for the robot to be secure, it is necessary that that machine be secure as well. Compromising a robot's operating system via a kernel-level remote code execution exploit would render the robot entirely compromised as well. One of our main sensors is a Kinect that provides video and depth information and uses device drivers to enable software to interact with the sensor data. Robotic systems also contains components outside of the robot itself: a remote server for storing logs or performing computationally expensive calculations, or for

Figure 1: Our experimental robot platform. To experiment with secure robots we built a robot using an iRobot Create, a netbook running Linux, and a Kinect for extra sensing.

providing commands to the robot. These components communicate with each other via traditional networks, such as WiFi. These remote systems and networks need to be accounted for when building secure robot applications.

2.2 Differences

Robots differ from traditional computer systems in two key ways. First, robots can move autonomously and they have manipulators (e.g., arms) that can affect the physical world. Second, their underlying algorithms are fundamentally probabilistic. These differences provide a number of key challenges for designers of more secure robot systems.

2.2.1 Robot mobility and actuation

Robots have the ability to directly interact with the world around them on a level that traditional computer systems do not. This both makes it possible for their security measures to be compromised, and gives an attacker much more ability if they are successful. While a non-robotic system might have some ability to affect its environment, it is usually very limited well defined. A robotic system, given an manipulator like a robotic arm, is capable of doing almost anything a human can. Robots can even modify themselves - using an actuator to reposition one of their cameras, reconnecting an intentionally disconnected sensor, perhaps even intentionally sabotaging their safety devices. In most situations, it is accepted that if you have physical access to a computer, then you can gain access to its information. A robot potentially has this access to itself.

Being mobile, a robot can even position itself wherever it needs to be. If you're working on a robot in a safe environment (e.g., workshop), it is likely capable of relocating itself to an unsafe environment (e.g., the children's playroom, or outdoors). It is possible for a robot to just drive away and be lost somewhere. If your robot has a microphone, it can reposition itself to be able to hear you when you think you're out of listening range. An attacker could even steal your robot (and any possessions it grabs) by remotely controlling it to simply drive away from your house.

2.2.2 Probabilistic algorithms

Research from the last decade has shown how robot designers can use probabilistic techniques to build robust mobile robots. Researchers have developed the mathematical underpinnings and algorithms for reasoning about robot locomotion and for determining where a robot is currently located (localization) in unknown environments and with noisy sensor data [16]. They have applied these techniques to robots for leading tours in museums [15] and building cars that drive themselves [17].

However, the probabilistic nature of these systems complicates security and raises new questions. Because the robot never knows for sure who it is interacting with or where it is located, security decisions that use this information need to take the probabilistic nature of this data into account. Additionally, these systems are robust in unknown and noisy environments, but have been largely untested with an active adversarial model where an attacker is trying to impersonate a legitimate user.

3 A security framework for robot apps

As general purpose robots become more common, we will begin to see a rise in commodity software for robots. There will be apps to enable your robot to interact with specific shops, perform specific chores, or generate specific hilarious noises. Software developers will require robot-platform-independent frameworks to accommodate their applications. Inevitably, we will see some form of "robot app store".

Each robot platform will be able to implement high-level functionality in different ways – movement can be implemented with wheels, legs, treads, etc. To this end, the software will communicate with the framework at a higher level than traditional framework/application interfaces. Commands like "Move to location X" instead of "Draw an 'OK' button".

Security, as well, will have to be implemented through higher level abstractions. Instead of sockets and files and processes, we will have to work with people, places, and actions. It will be important to figure out the correct abstractions that enable security to be defined at this level. It is important that we also respect privacy and do not allow the applications access to more personal information than is required.

In this section we discuss three aspects of our proposed framework for building secure robot applications. First, we discuss the notion of user identification in the context of a robot that buys coffee for users. Second, we propose a robot "app store" and discuss ideas on identifying abstractions that one could use to expose security relevant information. Third, we describe notions of privacy and how this can build off of the work done on distributed social network privacy to help users make decisions about security and privacy with robot applications.

3.1 Identifying users securely

One fundamental aspect of robot security will be identifying users. An example of this comes from a system we call CLASS, where we programmed our robot, Isaac, to go fetch coffee for a user. The application required a concept of who it was fetching coffee for so it could find them again, but there was no need for it to actually know the user's personal details. To implement this, we added an API that identified a user and returned a handle that represented them (essentially a pseudonym) without actually informing the application who they were. Later, the application could query "Is this person the same as the one I interacted with earlier" and the security framework would take care of the details. When the application had trouble finding the user, it would query the CLASS framework "where is this user likely to be found" – and it would get a location back. However, there is no need for a robot to even know the details of this position – since the software framework is in high-level concepts, the application just calls "move to location" and does not know where it is.

3.2 Exposing privileges

Most smartphone app stores provide the user with a summery of privileges that are allowed/denied of the application, and the user has to OK them. While such an idea has merit, the increased capabilities of the robot (and thus increased consequences of attack) and the increased number of possible options makes this very difficult. For example, let's say you downloaded an app "pretend to be a Labrador and fetch a ball"[1]. In order to recognize a tennis ball, it requires the "use the camera" privilege, and in order to fetch it requires the "move the robot" privilege and the "use robot arm" privileges. Additionally, it grabs the "use speakers" privilege so that it can simulate barking. Unfortunately, this app is malicious and it spends its days watching you, hoping to catch sight of your credit card and read the numbers on it. You haven't given it the "create a connection over the internet" privilege, so you think you are safe anyways. However, one day when you are not home the robot simply picks up the phone, calls a number, and reads it the numbers that it had memorized.

This shows why low-level abstractions like currently defined on phones or traditional computer systems are not sufficient to define our security policies. A much better policy would grant this application "locate object (tennis ball), identify/locate user anonymously (via a pseudonym), move to {tennis ball / user}. Speak only in the presence of the user." The challenge here is to expose the key abstractions that are meaningful from a security perspective and to map these to something that the platform can enforce.

[1] We expect this app to be a hit.

3.3 Privacy

Another simple application we think would be useful would be an app for having our robot check if a professor is in their office and available to meet with a graduate student. There are many privacy issues hidden in this seemingly simple app. While it is acceptable for a graduate student to ask if his or her advisor in their office, it is not acceptable for everyone.

3.3.1 A social network

The list of people who should be able to ask if a professor is in his or her office is hard to come up with. Students they advise, students in his or her classes, students in his or her department (during office hours), other professors, building administrators, spouses, etc. Even with a well thought-out list, there are sure to be additional people – like if a child's schoolteacher came looking for a professor, there is probably a good reason for it.

Privacy decisions cannot be made accurately by creating a central directory of "groups." Instead, it seems to lend itself to a network of relationships – essentially, a social network. This could easily be implemented through a Facebook plugin. Most of the information could even be automatically determined without user input – people you interact with frequently are more trusted. Relationships such as "sibling" and "friend" and "friend of friend" are also well defined already on such a network. It would be reasonable to trust your friends' opinions as well – if someone that you trust strongly trusts someone, then you probably can too (given no

other information). Trust, in this case, would not be a binary value but a measure that can be adjusted over time.

Rather than have a user attempt to quantify their exact opinion of what privacy entails, we believe that it can be accomplished through feedback and learning mechanisms. The user can view a summary of recent privacy related decisions, and then provide input if they feel that they were too strict or too harsh.

3.3.2 Acceptable mistakes

While a method like this will occasionally make mistakes, they will generally be small ones – it will not go and tell your entire personal calendar to a stranger, but it might give someone slightly too much or too little information – perhaps it overestimated the values of a friends' ratings. We do not think that small mistakes like this will be a problem – you could expect as much from any similar system, human or robot.

However, privacy goes both ways. Perhaps a user could indicate that they do not wish for the professor to know that they were looking for him (what student would really want the professor's privacy dashboard to pop up "Student A came by 20 times while you were away"). The robot should be expected for respect their privacy as well – though it would be much more likely that the professor will not tell information to an anonymous user.

3.3.3 Users need control of their privacy

In order to be sure that we respect every user's privacy, we came up with a strong guarantee for the framework that we are designing. Every user will have a master "node" that is responsible for making their privacy-related decisions, and *all* data containing their private information will go through that node. You can use public key cryptography to sign messages and prove that they are indeed from the correct node and thus represent the will of the user.

Here is an example. A student, Alice, wants to know if her professor, Bob, is in his office. She sends her robot off to go have a look. When the robot arrives in the office, it looks through the window and recognizes Bob. However, it cannot immediately tell Alice this as the information has not been signed off by Bob's privacy node (could just be an application running on his desktop, or his smartphone, or wherever). Instead, the robot must send a message to Bob's node asking if it is okay for Alice to know where Bob is. And since that message contains information about Alice, it should be signed by her node (luckily it was obvious that this message would be needed when the request was given, so the robot has it handy). Now, at a later time, Bob can review his logs and see that Alice was attempting to find him – and Alice can review her logs and know that Bob was told she was looking for him.

Having a log to keep everyone accountable is very important for privacy. If a mistake is made, knowing who was involved gives the affected people a lot of peace of mind. It also allows you to hold people accountable – someone had a robot watching you all day long? You probably want to have a talk with them (and adjust your privacy thresholds).

4 The software architecture

The key challenge with the software architecture is defining the security relevant robot abstractions and exposing the communication between different modules that operate on these abstractions. Then, the framework should be decomposed around these abstractions.

As we have shown, high-level abstractions are well suited for defining the policies and APIs for robot apps. Recent work on building secure web browsers has also used high-level abstractions as first-order concepts, so we can borrow a few lessons on architecture from them. Like the OP web browser [4, 5], we believe that separating the main components of the framework and defining a rigid interface between them can make reasoning about security much simpler. All inter-component communications go through a common message-passing interface that is easy to inspect.

The IBOS operating system [14] shows that promoting these high-level abstractions to first-order concepts allows for security invariants to be verified directly at the system kernel level. In effect, this massively reduces the amount of code that you're trusting when you make security decisions and allows you to withstand attacks that compromise the framework level libraries and modules.

Figure 2 shows our proposed architecture. This overall architecture resembles a microkernel where there is a thin layer of software responsible for passing messages between different modules running above. The layers above implement hardware-specific features, robot abstractions, and abstractions for applications. The applications run at the top and use the abstractions exported by lower layers.

Not only does the real world present a massive state space in terms of possible robot actions and encounters, but it provides one that we cannot modify easily. For a computer system, we regularly define standards - all web servers must accept requests in this form, all file systems must support these features, etc. You can not however release the IEEE spec on coffee shop ordering, or the standard kitchen cabinet interface. To this end, you'll need to have a modular system wherein modules can provide capabilities to one another. If you make an app that fetches coffee from the shop, it might depend on the Starbucks

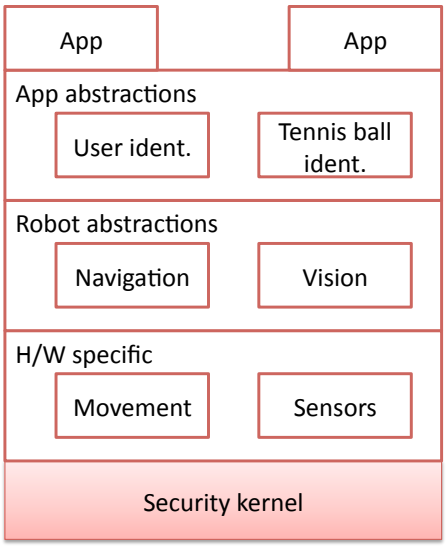

Figure 2: Proposed software architecture. This figure shows the layers of our proposed software architecture with a few example services in each layer.

app to be able to order coffee from Starbucks. Many coffee shops could realize the benefit of a robot knowing how to order their coffee, and they could each write a module providing functionality for their shop.

5 Additional related work

In one related project, O'Kane proposes privacy enhancements for robots [10]. In this work, O'Kane suggests approaching robot privacy from a hardware perspective by crippling sensors intentionally to avoid allowing the robot to learn too much about the objects it is sensing. However, our position is that robots should have the highest quality hardware that makes economic sense for the robot and that privacy should be enforced by the more flexible software layer.

Tomatis *et al.* proposed writing robot algorithms in type safe software to avoid common programming pitfalls (e.g., buffer overflows). However, the sheer number of libraries needed to run reasonable robot applications makes this approach infeasible.

Several projects from the robotics area look at developing robots that avoid hurting people [18, 9, 6]. Their key techniques supplement the basic collision avoidance techniques used by most robots to take into account soft tissue. When their algorithms detect soft tissue, they prevent damaging it. This type of measure is fundamental to the robot itself and does not have much to do with robot apps.

Several recent studies in privacy are related to our work. Hong *et al.* defined a model for privacy with ubiquitous systems [7], and Beresford and Stanjano look at anonymity for applications that track location [2]. Finally, recent work on decentralized social networks [1] shows mechanisms we could use to implement some of the social-network-based privacy mechanisms we propose.

6 Conclusions

Given the success of "app stores" with smartphones, it is likely that a similar platform will show up for robot applications. These generally programmable robots will enable a new class of applications that use robots in novel and interesting ways. However, robots are fundamentally different than traditional computer systems, so it will be challenging to design systems that allow users to download and install relatively untrustworthy applications without compromising the user's or the robot's security and privacy. We have some promising initial ideas for improving security and privacy for robot apps, but this new area of security research will evolve with time as people build more robots and more robot apps.

Acknowledgment

This research was funded in part by NSF grant CNS 0953014 and AFOSR MURI grant FA9550-09-01-0539. Any opinions, findings, conclusions or recommendations expressed in this paper are solely those of the authors.

References

[1] Backes, M., Maffei, M., and Pecina, K. A security API for distributed social networks. In *Proceedings of the 18th Annual Network and Distributed System Security Symposium* (February 2011).

[2] Beresford, A. R., and Stajano, F. Location privacy in pervasive computing. *IEEE Pervasive Computing 2* (January 2003), 46–55.

[3] Denning, T., Matuszek, C., Koscher, K., Smith, J. R., and Kohno, T. A spotlight on security and privacy risks with future household robots: attacks and lessons. In *Proceedings of the 11th international conference on Ubiquitous computing* (New York, NY, USA, 2009), Ubicomp '09, ACM, pp. 105–114.

[4] Grier, C., Tang, S., and King, S. T. Secure web browsing with the OP web browser. In *Proceedings of the 2008 IEEE Symposium on Security and Privacy* (May 2008), pp. 402–416.

[5] Grier, C., Tang, S., and King, S. T. Designing and implementing the OP and OP2 web browsers. In *ACM Transactions on the Web (TWEB)* (2011).

[6] HADDADIN, S., ALBU-SCHANDFFER, A., AND HIRZINGER, G. Soft-tissue injury in robotics. In *Robotics and Automation (ICRA), 2010 IEEE International Conference on* (may 2010), pp. 3426 –3433.

[7] HONG, J. I., NG, J. D., LEDERER, S., AND LANDAY, J. A. Privacy risk models for designing privacy-sensitive ubiquitous computing systems. In *Proceedings of the 5th conference on Designing interactive systems: processes, practices, methods, and techniques* (New York, NY, USA, 2004), DIS '04, ACM, pp. 91–100.

[8] IROBOT, INC. `http://www.irobot.com/`.

[9] LUCA, A. D., ALBU-SCHAFFE, A., HADDADI, S., AND HIRZINGER, G. Collision detection and safe reaction with the dlr-iii lightweight manipulator arm. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS2006)* (2006).

[10] O'KANE, J. M. On the value of ignorance: Balancing tracking and privacy using a two-bit sensor. In *Proc. International Workshop on the Algorithmic Foundations of Robotics* (2008).

[11] PARO ROBOTS USA, INC. Paro therapeutic robot. `http://www.parorobots.com/`.

[12] ROBOMOW. `http://www.robomow.com/`.

[13] SINGER, P. *Wired for war: robotics revolution and conflict in the 21st century*. Penguin Press, 2009.

[14] TANG, S., MAI, H., AND KING, S. T. Trust and protection in the Illinois browser operating system. In *Proceedings of the 9th USENIX Conference on Operating Systems Design and Implementation* (Berkeley, CA, USA, 2010), OSDI'10, USENIX Association.

[15] THRUN, S., BEETZ, M., BENNEWITZ, M., BURGARD, W., CREMERS, A., DELLAERT, F., FOX, D., HÄHNEL, D., ROSENBERG, C., ROY, N., SCHULTE, J., AND SCHULZ, D. Probabilistic algorithms and the interactive museum tour-guide robot minerva. *International Journal of Robotics Research 19*, 11 (2000), 972–999.

[16] THRUN, S., BURGARD, W., AND FOX, D. *Probabilistic robotics*. Intelligent robotics and autonomous agents. MIT Press, 2005.

[17] THRUN, S., MONTEMERLO, M., DAHLKAMP, H., STAVENS, D., ARON, A., DIEBEL, J., FONG, P., GALE, J., HALPENNY, M., HOFFMANN, G., LAU, K., OAKLEY, C., PALATUCCI, M., PRATT, V., STANG, P., STROHBAND, S., DUPONT, C., JENDROSSEK, L.-E., KOELEN, C., MARKEY, C., RUMMEL, C., VAN NIEKERK, J., JENSEN, E., ALESSANDRINI, P., BRADSKI, G., DAVIES, B., ETTINGER, S., KAEHLER, A., NEFIAN, A., AND MAHONEY, P. Winning the DARPA grand challenge. *Journal of Field Robotics* (2006). accepted for publication.

[18] ZINN, M., KHATIB, O., ROTH, B., AND SALISBURY, J. K. Playing it safe - human-friendly robots. In *IEEE Robotics and Automation Mag* (2002).