Security Risks Involved in Raspberry Pi Development

Skyler Tom

Abstract

Raspberry Pis have become increasingly popular over the past few years. In addition to teaching student's basic coding skills, Raspberry Pis can be used for home servers. Projects such as home servers and torrent boxes are extremely useful for the hacker, but are vulnerable to attacks. There are many Raspberry Pi projects on the web providing step by step instructions on how to do the project; however, the security risks involved in the projects is not well documented. Many of these projects instruct the user to modify settings or use frameworks that aid in security but do not explain what risks these procedures are mitigating. This paper aims to cover some techniques used to protect Raspberry Pis. Further research could result in useful, fully comprehensive documentation of risks and prevention methods associated with Raspberry Pis.

Introduction

Raspberry Pis have become increasingly popular over the past few years. There are a wide variety of applications including but not limited to: teaching students to code, building gaming consoles, and setting up home servers. The rise in popularity of Raspberry Pi's has led to an increase in creativity and documentation of potential projects for Raspberry Pis. These projects are a productive way to learn and express creativity (no one is arguing that turning Raspberry Pi's into DVD players is dangerous); however, some of the more "useful" projects can lead unaware Raspberry Pi hackers to expose themselves to attackers. Projects such as home servers and headless torrent boxes leave the user susceptible to attack.

To the Community

In today's world, it is essential to be technologically savvy. Furthermore, there are rapidly increasing numbers of computer science majors across the country. Building technology is increasingly popular and the barriers to entry are rapidly decreasing. Raspberry Pis are a perfect example of the improvements in access to building technology. They are both relatively cheap (the standard model is consistently $35 and there is even is a $5 mini version) and are easy to modify. In COMP 116 at Tufts we have discussed how the increase in coders will likely also lead to an increase in security flaws in code (as the number of applications produced goes up, the number of security flaws will at increase proportionally at the very least). With the increase in documentation of projects on the Internet and increase in interest of Raspberry Pi's there will undoubtedly be an increase in insecure Raspberry Pi projects.

Much of the documentation available online for projects includes the installation of some security measure without a full explanation of the risks involved if one doesn't install these measures correctly. The goal of this paper will to be illuminate some of the flaws in common Raspberry Pi projects so that potential Raspberry Pi hackers will be able to learn about why and how important various security measures are. Furthermore, the purpose of this paper is to gather existing information about Raspberry Pi security into one resource.

Action Items

First, a point almost not worth discussing: Raspberry Pi users must immediately change the default password for the *Pi* or *root* user. In the Raspberry Pi documentation it is listed that the default username and password are "Pi" and "raspberry" [1]. It is too simple for attackers to sudo in using these defaults. While not complex nor unpublicized, there are plenty of careless users who do not make this a priority or do not know to do this.

One security concern with Raspberry Pis is the *PermitRootLogin* feature. One Raspberry Pi project page described enabling this feature as: "Whilst this opens up security concerns its sometimes very useful" [4]. Users should not enable this feature unless absolutely necessary. Too often tutorials instruct users to flip switches that could have serious security ramifications.

To safe guard against brute-force attacks, users can utilize intrusion prevention systems to ban malicious IPs. Fail2ban is a framework that bans IP's after multiple unsuccessful logins. By default, it bans an IP after 6 unsuccessful login attempts for 10

minutes, but is Fail2ban is easily configurable to ban after any number of logins and time period (indefinitely is the best option). It should be noted that Fail2ban does not mean that the Raspberry Pi will be immune to brute-force attacks. Attackers can use different IPs and use other techniques; however, this makes it much harder for generic brute-force attacks to succeed at a low cost to the user.

The easiest way to secure a Raspberry Pi's SSH is to disable the password login and use SSH keys instead. A common way to attack Raspberry Pi's is to attempt to SSH in. Many Raspberry Pi users report that there were brute force attempts after a relatively short amount of time with their server up and running. Even the most secure password can be brute-forced, it just depends on the amount of time and processing power the attacker uses [3]. Users are only human and in general choose bad passwords (further enabling brute-force attacks). To use SSH keys, you must have a public and private key pair. Once generated, the public key is stored in a collection of authorized SSH keys on the Raspberry Pi and the private key is password protected on the local machine [2]. You may store as many public keys in the authorized collection of keys on the Raspberry Pi (one key per machine requiring SSH access). On Unix operating systems these keys can be generated with *ssh-keygen*. Now when SSHing in, you will be prompted for the password to access the private key and then signed in. After this successful completion, you should disable password based authentication on the Raspberry Pi. Brute force attacks will not work because the password that SSH is requesting is to unlock the private key on your local machine. It's not the password for your private SSH key that logs you in, it's the private key itself. Other users would have to have a copy of an approved public-private key pairing to log in.

Some websites simply recommend changing your port number from the default for SSH; however this is not an effective method. Changing the port merely gives the false sense of security. It is true that many automated programs simply hit the default ports. It is also extremely easy for attackers to scan an IP's open ports to find the corresponding SSH port. Instead of changing ports, it is much more effective that users use SSH keys for authentication.

Conclusion

The techniques covered all apply to SSH security regarding Raspberry Pis. The goal of this paper was only partially accomplished by providing in multiple ways to modify SSH access. Further research is recommended to compile a more comprehensive list of security recommendations for Raspberry Pi users. As the number of documented projects for Raspberry Pis and interest in doing these projects grows, the need for documentation regarding the security of different projects grows.

References

[1] Raspberry Pi Configuration:

https://www.raspberrypi.org/documentation/configuration/raspi-config.md

[2] Use Public Key Authentication with SSH

https://www.linode.com/docs/networking/ssh/use-public-key-authentication-with-ssh/

[3] Cryptography

http://tuftsdev.github.io/DefenseOfTheDarkArts/notes/cryptography.html

[4] Root user privileges:

http://www.raspberry-projects.com/pi/command-line/root-user-privileges

[5] Fail2ban

http://www.fail2ban.org/wiki/index.php/Main_Page