Raspberry Pi NAS with Encryption

Evan Chapman
School of Computing
Clemson University
Clemson, SC
ejchapm@clemson.com

Joe Francesconi
School of Computing
Clemson University
Clemson, SC
france2@clemson.com

ABSTRACT

This paper aims to detail the process behind setting up a Raspberry Pi to function as an encrypted Network Attached Storage (NAS) system. An NAS allows users to access storage and data across a network without needing to use a cloud or a physical device on their computer. The report below details our specific implementation of the NAS device and its encryption methods.

INTRODUCTION

The goal of our project was to create a network-attached storage system with added physical encryption to the drive to reinforce security measures for safe data storage on a network. In our project, we utilized a Raspberry Pi device, SMB3 encryption, and eCrypt file system to make a device that could safely store files on a network. We installed an Ubuntu server onto the device, installed packages, and configured permissions to allow for the device to function properly as an NAS. The NAS device setup resulted in a success in which all data stored on the Pi was encrypted both over the network via SMB3 and physically through eCryptfs.

BACKGROUND

A Network Attached Storage (NAS) system is a storage device for data storage and retrieval over a network. It is like having a private cloud storage solution accessible only on one network. A well-designed NAS should be scalable, meaning you can always add more storage if you need it. Not only is an NAS cheaper and faster than a public cloud solution, it also gives you full control over your data[1].

A Raspberry Pi is a relatively low-cost single-board computer around the size of a playing card. It functions as a complete computer that supports a multitude of operating systems ranging from lightweight operating systems utilizing the Linux kernel to Windows 10[2]. The advantages of a Raspberry Pi are its form factor and cost, which allow professionals and hobbyists alike to utilize it for creative and relatively robust IOT solutions.

Ubuntu Server is a lightweight, GUI-less version of the Ubuntu operating system running the Linux kernel. The current LTS version that we are using for this project is Ubuntu 24.04 LTS, which runs the Linux 6.8 kernel with low latency kernel features enabled by default[3]. Ubuntu is a widespread and popular Linux distribution that is praised for its ease of use, customization, and vast community support.

MOTIVATIONS & OBJECTIVES

The two main methods of storing large amounts of data across devices, thumb drives and cloud storage solutions, both have vulnerabilities and weaknesses that we wanted to find an answer to in this project.

In the case of thumb drives, the storage and accessibility of data tied to a physical device that must always be plugged in can be a burden or complication for many people or scenarios. If multiple people or devices need the drive's data simultaneously, it becomes tedious to transfer and maintain.

With cloud drives, third parties are hosting the storage space on their servers, and the data we upload is not entirely in our own hands. The data is susceptible to widespread attacks targeting the hosting company or server and cloud outages that may render our data inaccessible or lost for an arbitrary period of time. On top of all of this, cloud storage solutions usually charge a monthly or yearly subscription to continue using their service.

With this in mind, we wanted to find a storage solution that was easily accessible without the need to mount or plug in a device every single time a computer needs the data from the drive. We also needed a solution that gives us full ownership and control over how our data is stored and accessed, who can access it, and when it can be accessed, as well as ensuring the option only has a minimal, upfront cost or one-time fee to be operational in the long term. The Raspberry Pi NAS provides all of these things, with the storage being accessible remotely across an entire network, allowing complete control to only the users that have authentication into the device and only requiring the purchase of a Pi device and a storage and drive for it.

METHODOLOGY & DESIGN

To setup this NAS, first, we acquired these materials:

- Raspberry Pi
- microSD card for a boot drive

- External storage device for our NAS storage
- Ethernet cable and stable internet connection
- HDMI cable and monitor
- Keyboard

First, we wrote Ubuntu Server 24.04 to our boot microSD. To do this, we visited the official Raspberry Pi website to download the Raspberry Pi Imager. Then, using the tool, we flashed the OS to the SD card. After inserting the newly flashed microSD card into the Raspberry Pi, we set it up with a monitor and keyboard and connected the ethernet and power. Next, we configured our network and forced a static IP so that we can easily connect to the NAS. To do this, we edited our netplan config file.

Now that we have our machine up and running Ubuntu Server, it is time to set up our hard drive or storage device for NAS storage. First, we created and formatted a partition on our external storage device. We formatted our partition in NTFS since we will access this NAS from Windows machines, and we want the format to be compatible. After we formatted our partition, we set the mount point inside our user's home directory by finding the UUID of our hard drive and editing the automount settings in fstab.

We used Samba for network file transfer to maintain compatibility with Windows. First, we installed Samba and configured Samba shares. To force SMB3 encryption with data transfer, we set `smb encrypt = mandatory` in the config file. Then, we configured the share folder by setting its path and specific masks. Finally, we created a new Samba user account to connect to the NAS through SMB and access the files on the shared drive.

After setting up Samba, we decided to add physical encryption to the device for an extra layer of security. We did this so that even if someone were to get a hold of the physical device, they would not be able to access the data in the share folder since it resides in the home directory. To encrypt the home directory, we utilized eCryptfs. First, we created a new, temporary admin account with which to work. Using this account, we encrypted our main account's home directory with eCryptfs. After encrypting the primary user, we logged back into that account and deleted the temporary admin account. With all this done, the home folder is encrypted and only decrypts when the user logs in, re-encrypting all the data every time the user logs out or turns the machine off. This protects from people attempting to access the drive's information via dual booting into the machine.

Now that the NAS is completely set up, we can connect to it from a Windows device. We do this by mapping a network drive to the static IP of the NAS and browsing for the folder defined in the Samba config file. Once we start the connection, we are prompted with a user login; here, we sign in with the Samba user that we created in the previous steps. If the login is successful, we will have access to the shared folder on the NAS and can add, copy, edit, or delete any file in the shared storage. For more information on the specific steps to set up the NAS, please consult our GitHub readme[4].

The NAS is designed so that anyone on the network (provided they have a valid account to connect with) can access the storage. The flow of network traffic might look something like this:

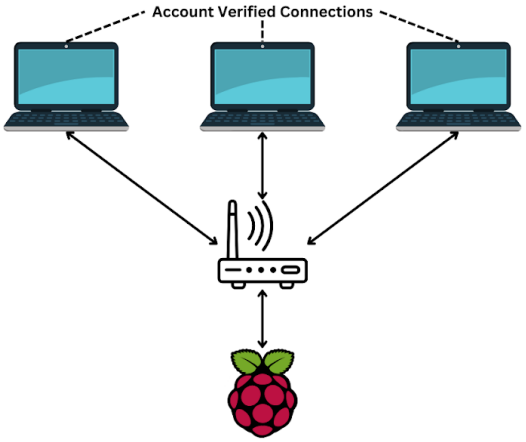

Figure 1: A demonstration of how a Raspberry Pi NAS connects to the network and communicates with other authorized machines on the network.

ANALYSIS & RESULTS

To demonstrate how this NAS works, first, a user must connect to it with an authorized account. On Windows, we map a network drive to the IP and shared folder and then login like so.

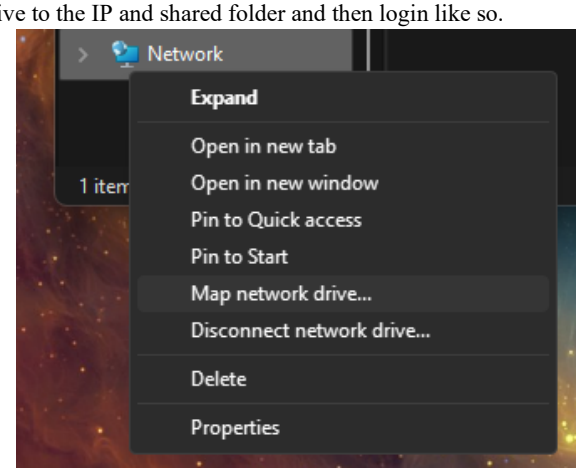

Figure 2: Process to start mapping a network drive in Windows.

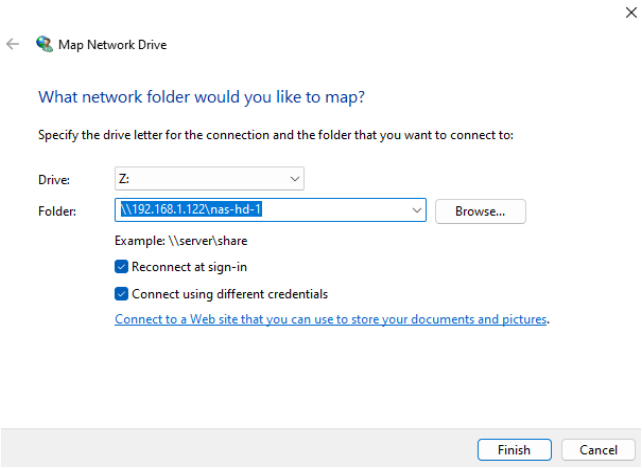

Figure 3: A demonstration of how to map a network drive in Windows File Explorer.

Then we are asked for our login credentials.

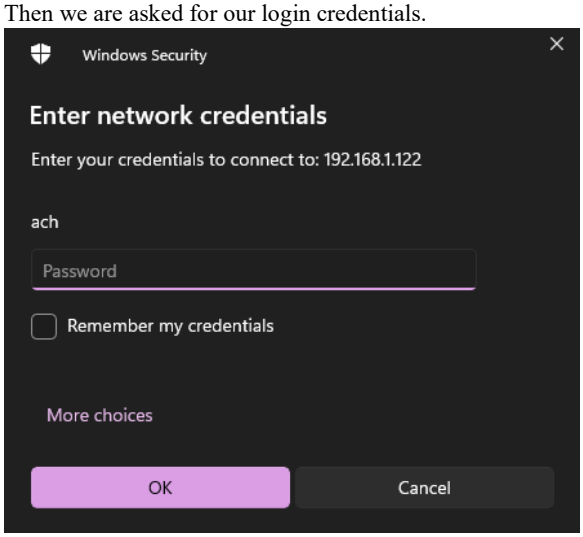

Figure 4: A login screen to login and authenticate the SMB connection.

After logging in with valid credentials, we can access the NAS.

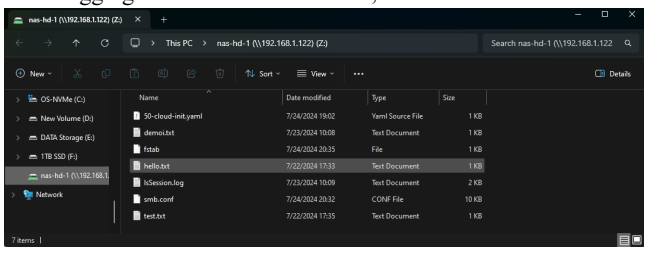

Figure 5: Windows file explorer connected to the network storage.

This NAS is most likely more secure than some public cloud storage solutions; however, the security, reliability, and safety of the NAS are wholly in the hands of the owner and setup. It can be as secure or open as the owner chooses. It is also dependent on Samba, so if there is a security vulnerability in Samba, then the data could be at risk. Also, the device is still vulnerable to social engineering attacks where a bad actor might gain authorized login credentials and access to the network. Additionally, to make the device more secure, we suggest adding a firewall to block all incoming traffic that is not SMB to prevent alternative breach methods.

CONCLUSION

The NAS system accomplished the goal of providing a safer, cheaper, and more easily accessible storage option across a network compared to alternatives. Once everything was installed and configured onto the Pi, we could connect our computers to the drive and access the data after authentication, successfully managing the storage in the system with session logs in the system to verify it.

The biggest weakness of the system compared to other storage devices is that the device's security is entirely dependent on us as the users configuring it. There is a setup process that other storages do not need, and it requires technical knowledge and understanding of Ubuntu/Linux. A lack of knowledge of the system can pose many security risks for storage on a network, so ultimately, the storage is only as safe as the least knowledgeable person using it.

REFERENCES

[1] What is NAS (Network Attached Storage) and Why is NAS Important for Small Businesses. *Seagate*. https://www.seagate.com/blog/what-is-nas-master-ti/. Accessed Jul 20, 2024.
[2] Raspberry Pi. *Raspberry Pi*. https://www.raspberrypi.com/software/. Accessed Jul 22, 2024
[3] Get Ubuntu Server. *Ubuntu*. https://ubuntu.com/download/server. Accessed Jul 22, 2024
[4] Evan Chapman, Joe Francesconi. 2024. 4240Project: Creating an NAS with a Raspberry Pi. https://github.com/ec3459/4240Project/blob/main/README.md.