Lab #4 Report: 2FA with Google Authenticator

CSC432 – Computer & Network Security

Franklin Nuth

13 February 2018

Abstract

In this lab, I will upgrade the security of my router by adding a Google two-factor authorization. I will install their Google Authenticator software in my router and in my Samsung phone to add another layer of security besides my already complex password. A few files in the router will be configured so that the Google Authenticator app will have the authorization it need to forward randomly-generated tokens to my phone and my router. I will then check the success of the whole process by logging into my router from the router itself, my web server, and my Kali to see if the two-factor authentication stays active through reboots.

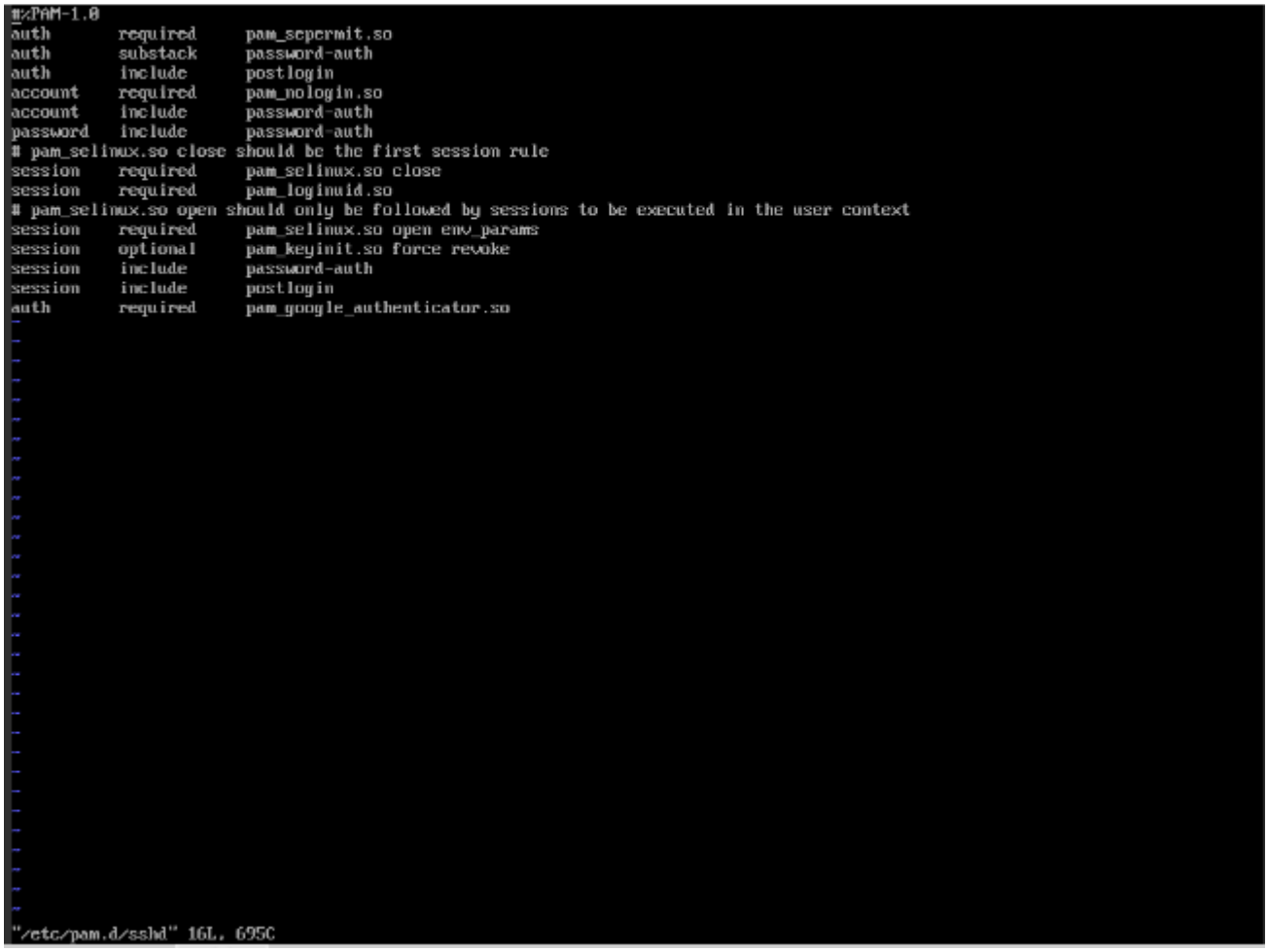
Introduction

Passwords are one of the most common forms of login credentials to date. Everybody and their parents know what passwords are since everyone has a digital device for looking up e-mail or texting others through apps. Unfortunately, no matter how hard we all try to protect our passwords with tight lips and minimal writing, hackers continue to successfully tap into their powers of innovation and create an impressive plethora of password-cracking software that most credentials are not safe from. To further add into the already chaotic situation, it is often the fault of the user for being hacked from using one password for all sites or writing too many passwords and forgetting them all. Two-factor authorization is the best and most recent solution to combat this predicament. Google provides tokens for anyone opting for two-factor authentication on their personal devices. As for me, I will be setting it up in my virtual router. I look forward to seeing how effective this new security measure will be.

Processes & Screenshots

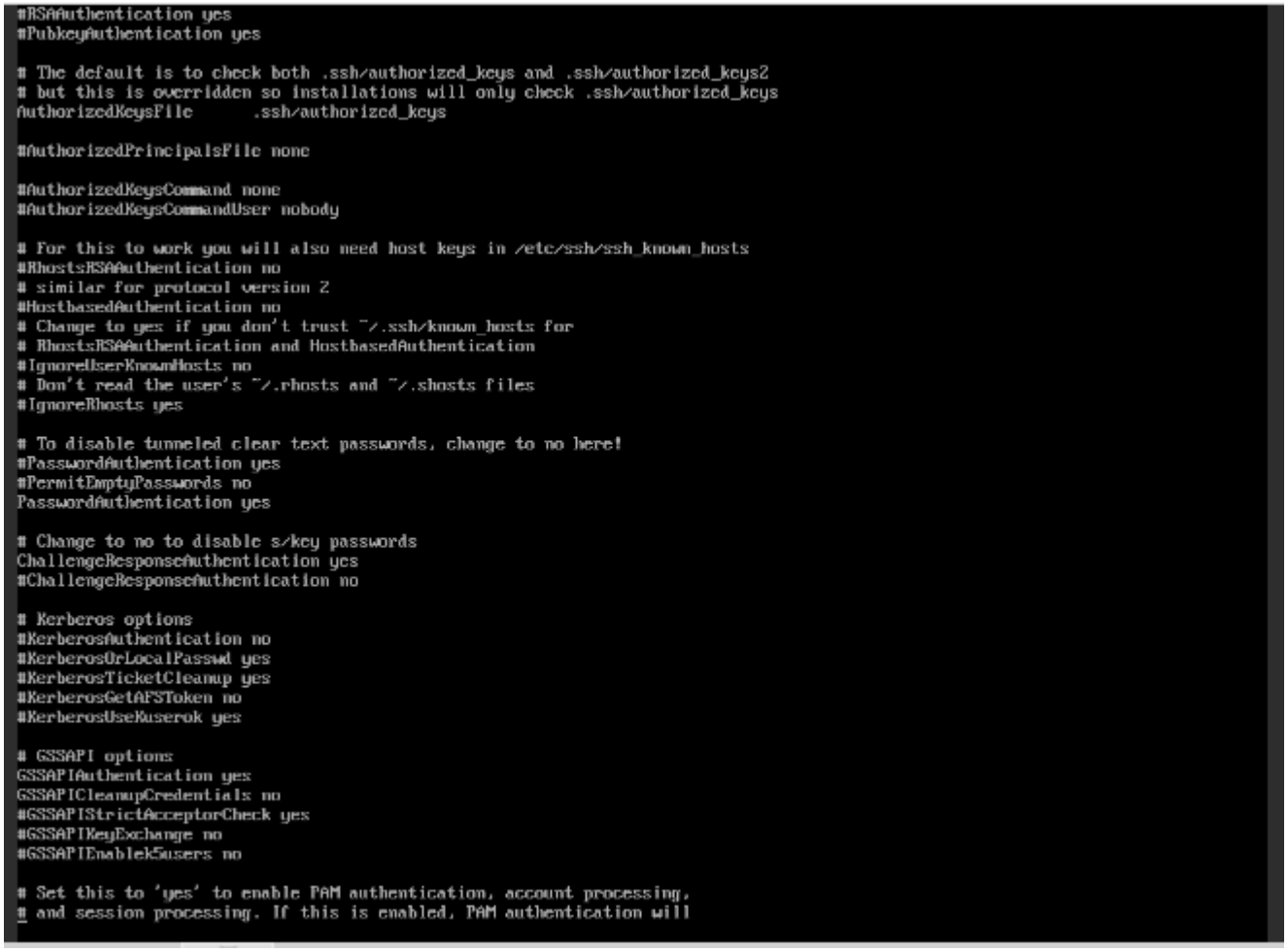
I first accessed my router through the ProxMox Virtual Environment. In the command line I typed “yum install google-authenticator”. This command installs all packages needed so the router can have what it needs to run the Google Authenticator service. After letting my router install the components, I

then went on to execute the Google Authenticator software by typing “google-authenticator” on the command line. At this point in the process, I also went ahead and download the Google Authenticator app on my phone. Upon opening the Google Authenticator program in the router, I have been asked a series of questions. I said yes to time-based tokens, yes to my root authenticator file, yes to disallowing multiple uses of the same tokens throughout half-minute intervals, no to increasing permitted codes, and no for rate-limiting. With Google Authenticator now installed on my phone, I then opened up my pam.d file through “vi /etc/pam.d/sshd” and added the following line “auth required pam_google_authenticator.so”.



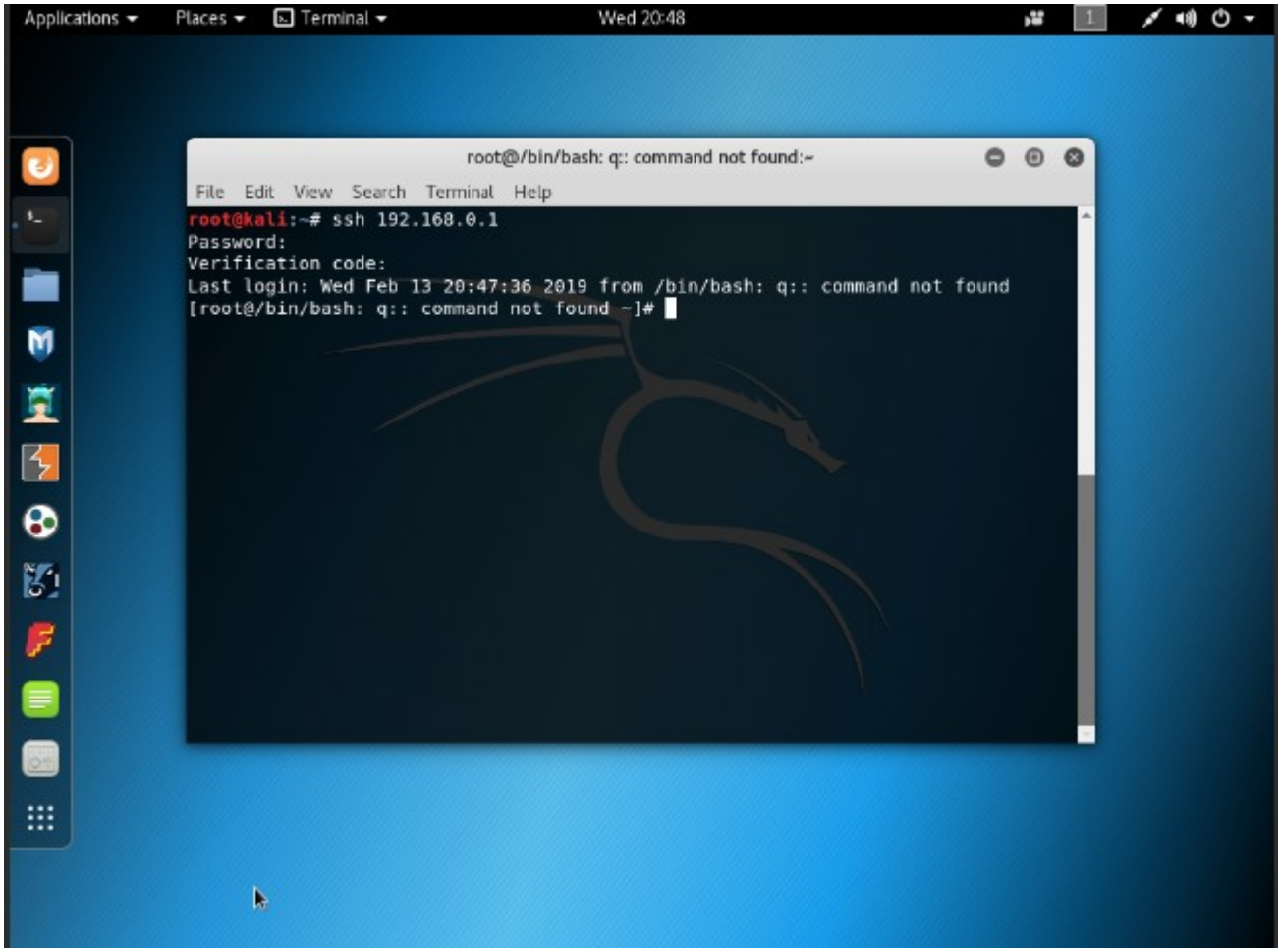
(The current configuration of my pam.d file. I inserted and formatted the last line column style, similar to the code above.)

After that, I entered another file which is “vi /etc/ssh/sshd.config”. This is where I changed “#ChallengeResponseAuthentication yes” to “ChallengeResponseAuthentication yes” and “ChallengeResponseAuthentication no” to “#ChallengeResponseAuthentication no”. I then restart the sshd service with “systemctl restart sshd.service”, and now my router is secure with two-factor authorization.



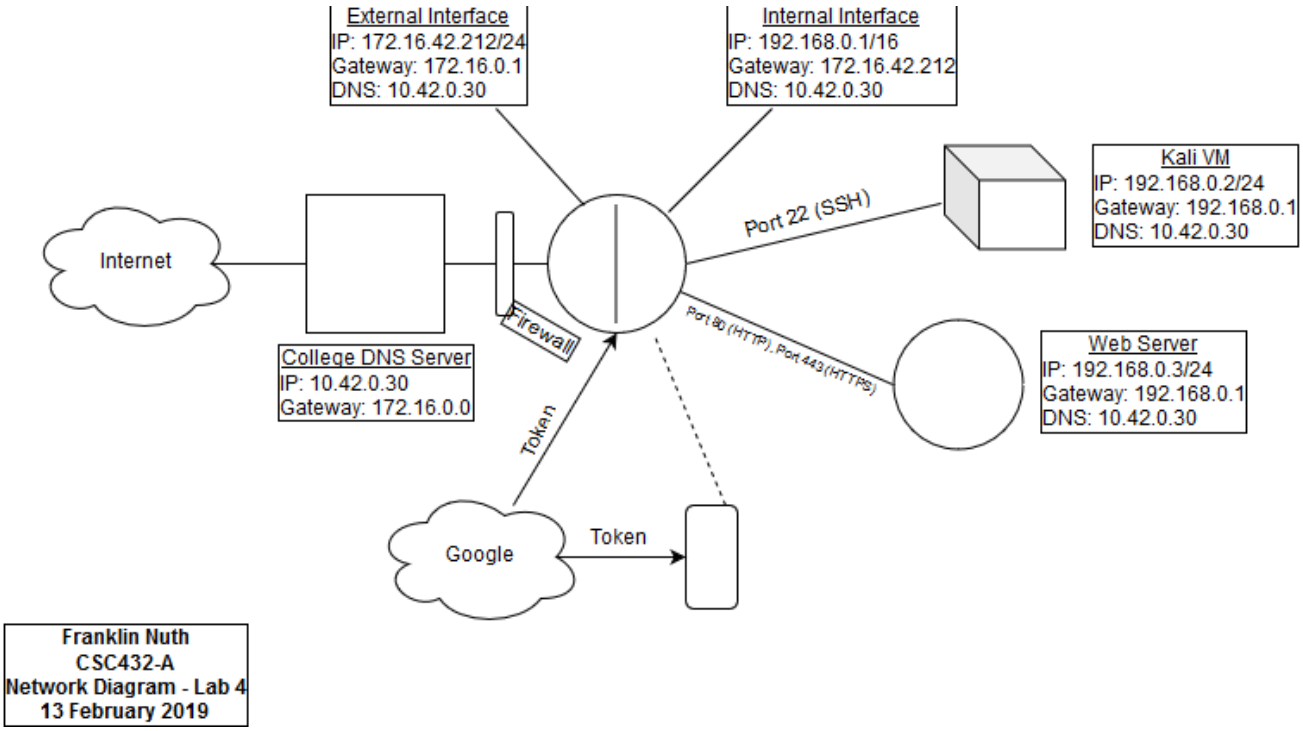
(The lines ChallengeResponseAuthentication properly changed for Google Authenticator.)

(Confirming that the two-factor authentification stays on the router.)



(Confirming that two-factor authentication protects the router from my Kali VM.)

(Checking if two-factor authentication protects the router from the web server.)



(My network diagram, revised and updated up to four times.)

Issues & Resolutions

Although I consider this to be the most straightforward lab thus far, I still ran into issues upon setting the software up and running it. When I tried to install Google Authenticator for the first time, the router was able to see the package, but was not able to download it. After minutes of research, I found out that I didn't install the repository that has the Google Authenticator package through the command "yum install epel-release -y". After downloading the repository, along with "yum install google-authenticator", I have an easy time setting everything up and configuring the files. The only issue that ate up most of my time was making sure my authentication tokens are officially going to my router and Kali. In my first ten attempts, I had a hard time syncing my phone and router by entering the verification code on time. I solved this issue by saying yes when the software asks me if I need a half-minute interval before a random code generates again. This amount of time is what I needed before my router accepts the new two-factor authhentication policy.

Conclusion

I learned from this lab that passwords are quickly becoming useless in our day and age. Not only I can set up two-factor authentication with my router, I can also set the same rules for my phone too. I have observed the internal workings of what it is lke to set up the extra security and become impressed at the level of alertness that Google has for cybersecurity. My personal password is more than 20 characters long, and I have an easy time remembering it. I think if everyone has a personal pass phrase that only they know seperated by underscores, then I think that extra tip will add a lot more security.

References

Wallen, Jack. 2018 April 12. *How to st up two-factor authentication on Cent OS 7.* Rerieved from: https://www.techrepublic.com/article/how-to-set-up-two-factor-authentication-on-centos-7/

Bolling, Josh. 2019 April 7.*How-to: Harden SSH Authentication with Google Authenticator.* Retrieved from: http://joshbolling.com/2017/04/09/how-to-harden-ssh-authentication-with-google-authenticator/