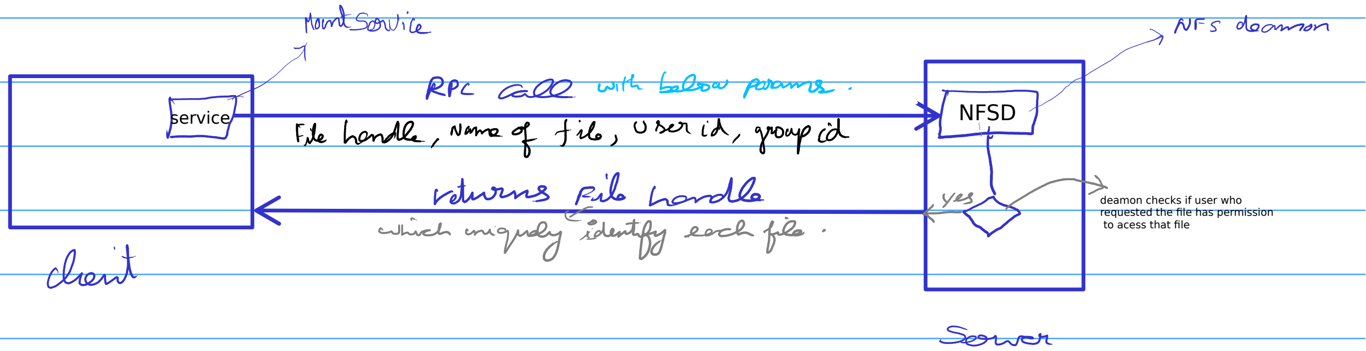
Default port - NFS totally uses 7 ports

Port 111 (TCP and UDP) and 2049 (TCP and UDP) for the NFS server.

NFS :

- Network file System.
- mount file systems into local system.

NFS stands for "Network File System" and allows a system to share directories and files with others over a network. By using NFS, users and programs can access files on remote systems almost as if they were local files. It does this by mounting all, or a portion of a file system on a server. The portion of the file system that is mounted can be accessed by clients with whatever privileges are assigned to each file.


Mount Service
service
client
RPC call with below params.
File handle, Name of file, User id, group id
returns File handle
which uniquely identify each file.
NFS deamon
NFSD
yes
Server

deamon checks if user who requested the file has permission to acess that file

- NFS works in b/w different os.

Enumeration

NFS-Common: Should be installed which helps in NFS enumeration.

- helps in enumeration
- includes programs like lockd, statd, showmount, nfsstat, gssd, idmapd and mount.nfs.

Showmount helps in listing the NFS shares.

Usage: Showmount <option><ipaddress>

man showmount → for more info

After knowing the shares name, we can try to mount it using normal mount command.

To Mount the NFS volume:

usage: mount <option> <ip>:<share name> <mount point>

→ create dir, if needed.

```
eg:) mount -t nfs 192.168.1.2:share /tmp/mount/ -nolock
```

-t ↓ type of device to mount. Here (NFS)

192.168.1.2 → ip

share → share name

/tmp/mount/ → mount point

-nolock → (optional) say not to lock.

Exploitation:

Before going to that, we need to understand about root squash and SUID

Root squash:

↳ NFS allows us to use the files present in shares as our local files present in our system.

↳ So the problem is the root user of the client machine who accessing the NFS shares is treated as root while accessing the shares.

↳ But, It allows a remote user to modify/delete the file he want.

↳ So, To avoid it in NFS, ROOT SQUASHING is Turned on by default in NFS

↳ If it is turned on, remote root user who uses the shares are assigned as NFSNOBODY, which has least privilege.

NFSNOBODY → username like that

Note: If it is turned off attacker may add a file with SUID bit set to the NFS share and run it to get shell.

SUID, GUID, STICKY Bit → Special permission given to files/Dir.

Normally, In linux when file executed, it inherits its permission from logged in user.

This gives Temporary special permission

↳ SUID - set owner user ID upon execution

It helps others to run the file with the permission of the owner of the file

↳ SGID - set group id up on execution.

It helps others to run the file with the permission of the group.

SGID is defined as giving temporary permissions to a user to run a program/file with the permissions of the file group permissions to become member of that group to execute the file. In simple words users will get file Group's permissions when executing a Folder/file/program/command.

↳ Sticky bit - when it is set to the directory, then people can delete the files present in directory which belong to them only not other users. irrespective of the permission

Note :

To set this special permission

↳ chmod U+S <file/dir> SUID

↳ chmod G+S <file/dir> SGID

↳ chmod O+t <Directory> Sticky

Steps :

↳ Assume you have rights to mount & upload file to NFS.

↳ Assume The scenario, where you have the non-privileged shell access to the system. (To do privilge escalation, follow the below steps).

i) First Mount The NFS to your local system. (Since Root squash is turned off If the remote user is root, then That user is Treated as root in NFS also).

Doing it in the NFS Share you mounted

ii) Copy/upload the bash shell to the NFS. (u may download shell from internet)

iii) change the ownership of the file to root

↳ chown root:root shell

user ↑ group ↑

iv) change the permission to executable and set SUID bit.

↳ sudo chmod +x shell

↳ sudo chmod +s shell

v) After doing the above steps, Then Login to systems as low privileged user (In this example, we already have a low privileged SSH access).

↳ In low privilged shell, execute the executable file we uploaded in NFS share. while executing use ./shell -p → Add this flag which runs the file with defined permissions.

↳ Then, boom! you got a root shell.