Encrypting NFSv4 with Stunnel TLS



*NFS clients and servers push file traffic over clear-text connections in the default configuration, which is incompatible with sensitive data. TLS can wrap this traffic, finally bringing protocol security. Before you use your cloud provider's NFS tools, review all of your NFS usage and secure it where necessary.*

The Network File System (NFS) is the most popular file-sharing protocol in UNIX. Decades old and predating Linux, the most modern v4 releases are easily firewalled and offer nearly everything required for seamless manipulation of remote files as if they were local.

The most obvious feature missing from NFSv4 is native, standalone encryption. Absent Kerberos, the protocol operates only in clear text, and this presents an unacceptable security risk in modern settings. NFS is hardly alone in this shortcoming, as I have already covered [clear-text SMB in a previous article](https://www.linuxjournal.com/content/smbclient-security-windows-printing-and-file-transfer). Compared to SMB, NFS over stunnel offers better encryption (likely AES-GCM if used with a modern OpenSSL) on a wider array of OS versions, with no pressure in the protocol to purchase paid updates or newer OS releases.

NFS is an extremely common NAS protocol, and extensive support is available for it in cloud storage. Although Amazon EC2 supports [clear-text and encrypted](https://docs.aws.amazon.com/efs/latest/ug/mounting-fs.html) NFS, Google Cloud makes no mention of data security in its [documented procedures](https://cloud.google.com/launcher/docs/single-node-fileserver), and major initiatives for the protocol recently have been launched by Microsoft Azure and [Oracle Cloud](https://blogs.oracle.com/linux/2017dec-oraclelinux-storage-appliance-in-oci) that raise suspicion. When using these features over untrusted networks (even within the hosting provider), it must be assumed that vulnerable traffic will be captured, stored and reconstituted by hostile parties should they have the slightest interest in the content. Fortunately, wrapping TCP-based NFS with TLS encryption via stunnel, while not obvious, is straightforward.

The performance penalty for tunneling NFS over stunnel is surprisingly small—transferring an Oracle Linux Installation ISO over an encrypted NFSv4.2 connection is well within 5% of the speed of clear text. Even more stunning is the performance of fuse-sshfs, which appears to beat even clear-text NFSv4.2 in transfer speed. NFS remains superior to sshfs in reliability, dynamic idmap and resilience, but FUSE and OpenSSH delivered far greater performance than expected.

Installation

Most of the NFS client and server code is already present in the Linux kernel, including implementations compatible with Sun's original v2 and v3 servers, as well as v4. A running NFS server does require several nfsd processes that are launched by the tiny /usr/sbin/rpc.nfsd binary, which takes few arguments and runs principally as a userspace placeholder to schedule file server threads within the kernel. The stunnel binary will be needed on both the clients (where the TCP data stream will be emitted) and the server. Some clients also will need to run the rpc.portmap dæmon, but most can now do without it.

On Oracle Linux 7.5 and its peers (CentOS, Scientific Linux, Red Hat), you can install the utilities with the following command (the nfs-utils package is likely already installed):

yum install nfs-utils stunnel

Ubuntu appears to [require](https://help.ubuntu.com/lts/serverguide/network-file-system.html.en) the installation of the full nfs-kernel-server even to run a client.

If you want NFS services to start on boot, use systemd to enable them with the following commands:

systemctl enable rpcbind

systemctl enable nfs-server

systemctl enable nfs-lock

systemctl enable nfs-idmap

You can launch the services with the corresponding start commands (don't launch them now):

systemctl start rpcbind

systemctl start nfs-server

systemctl start nfs-lock

systemctl start nfs-idmap

If you want to allow clear-text NFS over TCP and UDP into the server, reconfigure the firewall with the commands below. If you only intend to allow encrypted NFS over stunnel TLS or clear-text TCP (but not UDP), don't run these commands:

firewall-cmd --permanent --zone=public --add-service=nfs

firewall-cmd --reload

As an alternative, if you'll be testing clear-text NFS over TCP port 2049, run this command instead:

iptables -w -I INPUT -p tcp --dport 2049 --syn -j ACCEPT

The iptables call will not survive a reboot and will not allow UDP transport, but the firewall-cmd changes will be persistent and provide full-featured NFS access.

Clear-Text NFSv4

I should begin my coverage of NFSv4 with the admission that the protocol is not universally admired. Although [general criticism of NFS](https://www.kernel.org/doc/ols/2006/ols2006v2-pages-59-72.pdf) from Linux kernel developers points out a number of major flaws in several versions, Theo de Raadt, leader of the OpenBSD project, had [this commentary](http://openbsd-archive.7691.n7.nabble.com/nfsv4-td18690.html) on the status of v4 within the OpenBSD distribution:

NFSv4 is a gigantic joke on everyone....NFSv4 is not on our roadmap. It is a ridiculous bloated protocol which they keep adding [expletive] to. In about a decade the people who actually start auditing it are going to see all the mistakes that it hides.

The design process followed by the NFSv4 team members matches the methodology taken by the IPV6 people. (As in, once a mistake is made, and 4 people are running the test code, it is a fact on the ground and cannot be changed again.) The result is an unrefined piece of trash.

Many times, one man's trash is another man's treasure. Although Theo de Raadt is a great visionary and we owe our usage of OpenSSH to him, NFSv4 is the easiest NFS implementation to run over stunnel TLS.

NFSv3 and earlier are "stateless" file servers—the server only records read and write operations, and retains no status about client usage. NFS makes extensive use of Sun ONC RPC (Open Network Connectivity Remote Procedure Call), which is coordinated by the rpc.portmap dæmon with several other supporting processes to implement file locking, status reporting, crash recovery and ID mapping—these are distinct server processes running on separate ports that maintain client state information separate from the file server. The issue of using stunnel on v3 and below was [raised](https://marc.info/?t=122089506800034&r=1&w=2) in a discussion thread in 2008, and one of the thread participants [mentioned](https://web.archive.org/web/20120801000000*/https:/w3.physics.uiuc.edu/physwiki/doku.php?id=pcs:unix:nfs_over_stunnel) a document that he had written on the subject that has since been archived. The procedure for tunneling v3 is quite complex.

NFSv4 brings these stateful activities into the main protocol, and a client using it does not need to connect with the older v3 lockd, statd or any other separate RPC service. A local rpc.idmapd is required for the proper ownership and permissions maintenance, but idmapd does not need remote network connectivity beyond the channels already provided by the TCP connection maintained by the v4 client.

NFS originally ran over UDP (the Unreliable Datagram Protocol) on port 2049 in the expectation that packet loss on a local network would not severely interfere with NFS traffic. NFS over UDP can and does suffer badly when high traffic causes packet loss. NFSv3 added the ability to run over TCP (Transmission Control Protocol), and TCP transport on port 2049 is the default in Linux due to its greater tolerance to adverse conditions. There are usage scenarios where UDP is more efficient (see man 5 nfs for details), but UDP does not work with stunnel, so I don't address it here.

Let's begin by configuring a directory to be offered to clients by an NFS server. Create and populate the directory on the server machine with the following commands:

mkdir /home/share

chmod 777 /home/share

cp /etc/services /etc/nsswitch.conf /etc/hosts /home/share

Edit the file /etc/exports so that it offers a read/write share for the IP address of the client:

/home/share 5.6.7.8(fsid=0,rw)

The fsid is very helpful for NFSv4 mounts and is explained in the man exports manual page: "For NFSv4, there is a distinguished filesystem which is the root of all exported filesystem[s]. This is specified with fsid=root or fsid=0 both of which mean exactly the same thing." Establishing a root fsid will make your exports work more smoothly.

For purposes of instruction, define a small shell function and use it to check for rpc processes. After confirming that none of the well-known NFS programs are running, start the NFS server, and then observe what else is started:

# function pps { typeset a IFS=\| ; ps ax | while read a

do case $a in \*$1\*|+([!0-9])) echo $a;; esac; done }

# pps rpc

PID TTY STAT TIME COMMAND

598 ? S< 0:00 [rpciod]

# systemctl start nfs-server

# pps rpc

PID TTY STAT TIME COMMAND

598 ? S< 0:00 [rpciod]

15120 ? Ss 0:00 /usr/sbin/rpc.statd --no-notify

15131 ? Ss 0:00 /usr/sbin/rpc.idmapd

15143 ? Ss 0:00 /sbin/rpcbind -w

15158 ? Ss 0:00 /usr/sbin/rpc.mountd

It's apparent that the v3-related dæmons are started by the main file server unit under Oracle Linux 7. Don't be surprised by their presence.

On the client, you can add an entry to the /etc/fstab file defining a remote mount—it must contain the hostname or IP address of the server and (for later usage) the TCP port number:

1.2.3.4:/ /home/share nfs noauto,vers=4.2,proto=tcp,port=2049 0 0

The above fstab entry will allow you to mount the server, assuming that any and all firewalls allow the traffic and they can ping one another:

# mount /home/share

# ls -l /home/share

total 664

-rw-r--r--. 1 root root 158 May 16 11:34 hosts

-rw-r--r--. 1 root root 1746 May 16 11:34 nsswitch.conf

-rw-r--r--. 1 root root 670293 May 16 11:34 services

# cp /etc/yum.conf /home/share

# ls -l /home/share

total 668

-rw-r--r--. 1 root root 158 May 16 11:34 hosts

-rw-r--r--. 1 root root 1746 May 16 11:34 nsswitch.conf

-rw-r--r--. 1 root root 670293 May 16 11:34 services

-rw-r--r--. 1 nfsnobody nfsnobody 841 May 16 12:02 yum.conf

The nfsnobody above is an example of "root squash", where the server translates the activity of the client root account into an unprivileged user. There are several types of squashing, and they are usually an unexpected accident.

The following is an example from (the discontinued and unsupported) Oracle Linux 5, where all permissions get squashed:

# ll /some/share

total 44604

-rwxr-xr-x 1 nobody nobody 1638192 Jul 28 2016 7za.16.02

-rw-r--r-- 1 nobody nobody 57280 Oct 18 2017 fuse-sshfs-2.4-1.el5.i386.rpm

-rwxr--r-- 1 nobody nobody 233066 May 2 2017 Oracle\_LMS\_Collection\_Tool.zip

This is happening because an idmap "domain" must be specified in the /etc/idmapd.conf file. By default, the NFS domain is taken from the Fully Qualified Domain Name (FQDN) by removing the hostname prefix. If two servers are in separate DNS domains, their NFSv4 mounts always will be completely squashed. To correct this, specify the NFS domain manually:

# service rpcidmapd stop

Stopping RPC idmapd: [ OK ]

# grep ^Domain /etc/idmapd.conf

Domain = master\_nfs\_domain.yourco.com

# service rpcidmapd start

Starting RPC idmapd: [ OK ]

# umount /some/share

# mount /some/share

# ls -l /some/share

total 44604

-rwxr-xr-x 1 cfisher grp 1638192 Jul 28 2016 7za.16.02

-rw-r--r-- 1 cfisher grp 57280 Oct 18 2017 fuse-sshfs-2.4-1.el5.i386.rpm

-rwxr--r-- 1 root root 233066 May 2 2017 Oracle\_LMS\_Collection\_Tool.zip

Note that NFSv3 and below did not work this way. By default, numerical user and group IDs were preserved on a plain mount without idmap access. Although it's still important to maintain uid/gid synchronization, NFSv4 no longer allows numeric mapping, so don't be surprised by aggressive squashing.

Older Linux kernels used slightly different fstab syntax for NFSv4 mounts. Under Oracle Linux 5, note below the (deprecated) nfs4 mount type and the lack of a vers option:

server:/ /share nfs4 noauto,proto=tcp,port=2049 0 0

Before closing this section, I'd like to return to the fstab entry on the client:

1.2.3.4:/ /home/share nfs noauto,vers=4.2,proto=tcp,port=2049 0 0

The vers=4.2 requests the very latest version of the NFS protocol, which fails if it's not available on the server. Reduce this version if you're working with an older server. The client is chiefly responsible for determining the protocol version and feature settings of the connection (although the server can enable/disable specific NFS versions and some features system-wide in /etc/nfs.conf and /etc/sysconfig/nfs).

The noauto above prevents boot delays due to unreachable NFS servers by not mounting them by default at startup. My advice is *always to use noauto* to avoid a boot hung on an "NFS server not responding". There is a "background mount" option, which is useful, but I prefer a reboot entry in the (Vixie) crontab for root, which guarantees that NFS will not interfere in obtaining a login or otherwise bringing local services up. You can accomplish this with the following crontab entry:

@reboot /sbin/mount /home/share

More appropriate is to place all of your custom startup into a single script, then add the script as the reboot entry. Be sure to place your NFS mounts at the very end, in order of preference and tolerance for delay (you can launch particularly problematic mounts as background processes).

Some people express affinity for NFS automounters from various sources. I don't have enough mounts to justify the maintenance of multiple client automount configurations, so I've omitted such discussion here. With luck, the stunnel modifications that you're about to make if you're following along here should be compatible with most automounters.

NFSv4 over TLS with Stunnel

The interception of the TCP connection before it leaves the client will require a port for a local endpoint. A new port must also be selected on the server for TLS services. For reference, the following ports appear to be related to NFS:

# egrep -i '([^a-z]nfs|nfs[^a-z])' /etc/services

nfs 2049/tcp nfsd shilp # Network File System

nfs 2049/udp nfsd shilp # Network File System

nfs 2049/sctp nfsd shilp # Network File System

picknfs 1598/tcp # picknfs

picknfs 1598/udp # picknfs

3d-nfsd 2323/tcp # 3d-nfsd

3d-nfsd 2323/udp # 3d-nfsd

mediacntrlnfsd 2363/tcp # Media Central NFSD

mediacntrlnfsd 2363/udp # Media Central NFSD

winfs 5009/tcp # Microsoft Windows Filesystem

winfs 5009/udp # Microsoft Windows Filesystem

enfs 5233/tcp # Etinnae Network File Service

mountd 20048/tcp # NFS mount protocol

mountd 20048/udp # NFS mount protocol

nfsrdma 20049/tcp # (NFS) over RDMA

nfsrdma 20049/udp # (NFS) over RDMA

nfsrdma 20049/sctp # (NFS) over RDMA

For the simple ease of reading a netstat, I open port 2363 on the server and redirect client mounts to their local port 2323. You might not want your NFS traffic easily identified—if so, choose unrelated ports.

At a minimum, the stunnel TLS server must present a keypair. I actually generate and distribute a single self-signed keypair valid for ten years to the server and all the clients, which will act as a "local protocol key" that all members must both present and verify correct of all connected participants. Here I generate an example key:

$ openssl req -newkey rsa:4096 -x509 -days 3650 -nodes \

-out nfs-tls.pem -keyout nfs-tls.pem

Generating a 4096 bit RSA private key

.................................................++

...................................++

writing new private key to 'nfs-tls.pem'

-----

You are about to be asked to enter information that will be incorporated

into your certificate request.

What you are about to enter is what is called a Distinguished Name or a DN.

There are quite a few fields but you can leave some blank

For some fields there will be a default value,

If you enter '.', the field will be left blank.

-----

Country Name (2 letter code) [XX]:US

State or Province Name (full name) []:IL

Locality Name (eg, city) [Default City]:Chicago

Organization Name (eg, company) [Default Company Ltd]:NFS-TLS

Organizational Unit Name (eg, section) []:CHI

Common Name (eg, your name or your server's hostname) []:nfs-tls

Email Address []:foo@bar.org

The above command generates a key similar to the following output. Move your file to the /etc/stunnel directory, and set it to 400 read-only permission for root. *Do not copy the content below; it's for demonstration purposes only—you must generate your own*:

# f=nfs-tls.pem; cat $f ; chmod 400 $f ; mv $f /etc/stunnel

-----BEGIN PRIVATE KEY-----

MIIJQwIBADANBgkqhkiG9w0BAQEFAASCCS0wggkpAgEAAoICAQDMNL69ML5CX63O

d1kIeLYRjaKcxjH8s8vSv1REUOvs55h6cvIQBMFoRgabjD+cxzSvNuz+fbXzPlB5

QpsqyfZhq5LX48MvPBxmqoK4BcJWH0Vejo/kfkBPC+SSZd/QOKBHYxjvNBD0CGF+

/YqdEW8KSgVwFzQCKN28Rn2xfh/GBS564B3jwqsTGoL+gIXIeSuyozG1uLfD+nVS

N0zCfLwmNDQoRyVqhPK/r3ALNthpNzhQoFShoRxt0+pMgnhHexEezAMAUjEhZ22H

1iA5hlzO7jO7w0pmvIUb0zkFEYaIY1E/xKd5be4cf5cYvksohiwVvTKK66iNPcbW

fUTO9OeZ0jNRo8bI90LDYbZhoDS75vbNMlNON0YqtElhjE70s/3PAFkaAlMb3EeD

g4WXfbOzb0L5T8/8lgfFs/+DIa3lajJ81lbI/OO2gBfvVnzM5y2pSxROL+5I21cY

CtJolWA27vZWSvNbE4SGzW7Y4MhOg2uX+5Bln5Zqo7UDoXVSe6hlz7M5x1P6mKsX

+1YkjKGe4xi2ySLrWofHLqgtTTs+tI4hEWxFcCHu/ea5z2c3tEks6921VSyQc8Ak

cvuWVKqSBG04zqd3b+42JLZZg5mtdeaN3k2YiDWG0JUgh5qfu3UwiFUwFIPZRLEm

vPHT5iMNNvN9CpJqH1BkF9QF7XhNSwIDAQABAoICAEW2N+tUSY9VJHuYiL94ngcu

B/ZnPsdbBdkDUhwkV/Y/NfGPbg2D4hbb2QOfBFRcOSMbqBpVBhltC4Hp+BjKa576

OJ4U9hwY9EUkLo3uAWLvN/pIxtylMQULNVO5DYgC3MyiCvAWITd96PK2UWy/d93W

WTbj5PBbzR6qHdzLBsPOHwj5m5qWaVqTMWb6rzE6FG3egmjcD3gK96RClqTKely8

c5XQe/h6PHitxp09cvGwVTxJD7tByffAYXsPC0qzu6t80AV7CaSyr1SxB707nlFS

RjzyNWMPNo3CNPQDAJ9s8F7Jnra4jZITCJz80aGa9E/Tj/6W5qqZDVlJ2ISiXLGt

FWfynwUMZr1fqLmYV2W8kBdpzVva37iHq5TVErQZT9SHw+etAmaFUmPLbzwZm1JK

XPG1V4XNUG1V2YHzIFW0HUeFDhk16I9svwo/u8dK8HJyvW+cDBIsPeUWEhcR4qIp

XYx/rNZiU0qFVtnlpedDvDJf/ma2DyA3iDxS6YLpzK+RtDjnbznfglj2iVilnuCw

MMVzWTdIqs0VJ4iRL8+rV6wxO3kV++sXI0KQsJPbondVjX/FikbUkx7WRQ2OgbqJ

qjXL5hjrY4Bb2iC7gsIKuvfG4oMyS6O2amJ/V/YlO0nWQkVQZyqtn7z9iOTyQlay

MezX9XfF5zITnD9PDS9JAoIBAQDxjdUbdEVepIaXTnzkOj46uHdULJraop3bY3//

61CsU0LIzAN9/toCjAJWm8RxAME6weUZ+UZB3XRM0jfmAJnNT3a3I2s1+f8pJigE

zpvkPJjRRB/wpWBwMfIjDnMFD10gA0ChgcdvXdFtOS4v9nHxUaZyJC0xrofEQnh9

JEEWkmvPRq7VbfQUtFpEbpeWn16hdBNIC0V4MaVS17f3pQTYRoPWC4pT4SyN2pDF

pbmejkX58ahsnuql7Mv0pJhkwl/Cb5pkH3BdDIDZFOmmJMlCwghJvR9wvR92xuPy

hzSlATueePfLYAxarqhtEkeGxCWlYWGUD+W92q6MGTLnudIHAoIBAQDYax5cjj85

JTyu39dEEAZIneb+E/ZDQMxHfLVig/akxUpTNro2XChn56Lus27IMFI+lQ52hQ7Q

ftLnj+IyR41DlFDqsi3SbTU/dZsqYxVetl8+MDlOcxfmmJMrOkWLz5jrND0uZmt0

Kmf48xHKyOc6SZC7c4kUzlUPYsE0kRQaZ/fkTRG9aTJ65iH/JeXhROwQDt+qtkoD

xSMyqo2Pnj+u0LjPIw2MH/nuuM5bosCHPBBazf/CvFnlpi2Oq1jXHp2d8cVLyXUH

gM5CNT4kBBvw/ocAOORpbCMtM8EZdXB/a5SBXgnSbmdapMMQ6EAebpqfw3sK1Wie

BkuLxZetzcmdAoIBAQCk/GYxkVIMWb3gPOjLDgkRHIvMv4apjObbQXPc/gIlId18

vvQnq9mGYdD7DPu433YbxvHPstZNCJB2JCOwAnsKo5sHbba9sFqa5Yfx+Ji75LPQ

Q4K5YIulNkgXr7faHetSgUY0yirJI0B3JNYqRl7/H/DbB2CjDX2IDIq1lvyqCSp/

8dxaxPYw6hq5oPwDEimVh23gCGrTtL0h/1uVV24ettM3cLxznFpNLZsylIZbCPw8

wtVyE31cBYgtOfso3yZ+7LF8b4jU1URwgXsxUvDwmw0EKJv/6f1CqIhrT/QiO9xX

2nINxDXL/n3ludWG9BRuiDwY4F7gNSyBXnjJk78jAoIBAQCel9EGDo+yNuGDXTGJ

BR01tdECvGoo2qFYecEKUp46HQHcfSx0jZBmpE64EfHK7e43Qk/49oTmsSmo273t

DpYswdGSS8Rcgf8VY/+zTizo3UhqcDhujtUi/QhME0XHsPfk1MFI8XEpDbJnsuiE

7DjWc/aGB6KbBqE6xynCddZ/i1UTjo7DeQWvHlonegQ90p4THnM1zKPso1ip1mYq

qtMMLpRf5tYUq5IiKHfAm0HvWEq74F3evNw7+E1GUbam3h6vEe99HEKQnwmHZzEE

f6ZiMoOH3Ck2QDJ++4A0QeWQ2qtXKiyUcqd2u2rfRvNF2dOh5ESUqdMiioZuBPyk

NzvZAoIBAHdUEMDydPF6qBoknEAP9csaMZZcVmBfkIGcKyumzCiznF34VsE7FG9C

SuxdIShP/9/BVBAL4wKwVUYjRArJg0aIRTnOMRZC95GCq5YspozwPCJPxXYUWZuX

r0SfsXHuO6GhzvLjqUxguAbxAlHl7lI+cWiBM9xRbXxNG9jA8Yf1wq/8x3YGzad/

rMkTUL61i8xk6OwQA4exAH3PxtflooqVDHDnoL0Ukm57mddtoqBDA1NwZ4g149op

dwbERXBvnjJgn6m3kEQ/VoKKWzQY+y0Fu5OlHeVw9A2fcCWaCj4kp/pK7a860clR

NqwdAo0hNa3SsNtiM4Z3TM0RzDLw6fw=

-----END PRIVATE KEY-----

-----BEGIN CERTIFICATE-----

MIIFxzCCA6+gAwIBAgIJAI0iFv1oP1G9MA0GCSqGSIb3DQEBCwUAMHoxCzAJBgNV

BAYTAlVTMQswCQYDVQQIDAJJTDEQMA4GA1UEBwwHQ2hpY2FnbzEQMA4GA1UECgwH

TkZTLVRMUzEMMAoGA1UECwwDQ0hJMRAwDgYDVQQDDAduZnMtdGxzMRowGAYJKoZI

hvcNAQkBFgtmb29AYmFyLm9yZzAeFw0xODA1MjIwMDQzMTZaFw0yODA1MTkwMDQz

MTZaMHoxCzAJBgNVBAYTAlVTMQswCQYDVQQIDAJJTDEQMA4GA1UEBwwHQ2hpY2Fn

bzEQMA4GA1UECgwHTkZTLVRMUzEMMAoGA1UECwwDQ0hJMRAwDgYDVQQDDAduZnMt

dGxzMRowGAYJKoZIhvcNAQkBFgtmb29AYmFyLm9yZzCCAiIwDQYJKoZIhvcNAQEB

BQADggIPADCCAgoCggIBAMw0vr0wvkJfrc53WQh4thGNopzGMfyzy9K/VERQ6+zn

mHpy8hAEwWhGBpuMP5zHNK827P59tfM+UHlCmyrJ9mGrktfjwy88HGaqgrgFwlYf

RV6Oj+R+QE8L5JJl39A4oEdjGO80EPQIYX79ip0RbwpKBXAXNAIo3bxGfbF+H8YF

LnrgHePCqxMagv6Ahch5K7KjMbW4t8P6dVI3TMJ8vCY0NChHJWqE8r+vcAs22Gk3

OFCgVKGhHG3T6kyCeEd7ER7MAwBSMSFnbYfWIDmGXM7uM7vDSma8hRvTOQURhohj

UT/Ep3lt7hx/lxi+SyiGLBW9MorrqI09xtZ9RM7055nSM1Gjxsj3QsNhtmGgNLvm

9s0yU043Riq0SWGMTvSz/c8AWRoCUxvcR4ODhZd9s7NvQvlPz/yWB8Wz/4MhreVq

MnzWVsj847aAF+9WfMznLalLFE4v7kjbVxgK0miVYDbu9lZK81sThIbNbtjgyE6D

a5f7kGWflmqjtQOhdVJ7qGXPsznHU/qYqxf7ViSMoZ7jGLbJIutah8cuqC1NOz60

jiERbEVwIe795rnPZze0SSzr3bVVLJBzwCRy+5ZUqpIEbTjOp3dv7jYktlmDma11

5o3eTZiINYbQlSCHmp+7dTCIVTAUg9lEsSa88dPmIw02830KkmofUGQX1AXteE1L

AgMBAAGjUDBOMB0GA1UdDgQWBBQOE2cR4iZyEFHtuFd8uknFrzkUZDAfBgNVHSME

GDAWgBQOE2cR4iZyEFHtuFd8uknFrzkUZDAMBgNVHRMEBTADAQH/MA0GCSqGSIb3

DQEBCwUAA4ICAQAI6hgJ4p+ySxFxotUZXvzxN02D04FspLNBpoOc+4XI5KyGRGCg

0RKVuKjpVCEqsM1N4g+JMIqLPy9rvzfpcSbTnwJVPdE4VefU/EuUCSml5wY6sbll

7pbBAP7y2GOfpYRjAQLMsPTc6HxFDSOMc9F0kFe/OPU6GlH1ZF1NiOsEiDAE/bAO

D9GCFygrEaZyrlze5t5WRHx1dwKL3G+7hdOYqj2qPjvABhH2eWdzkWXN9Pwjdgz+

h8Mum1Ks7CWREMsJOxZqmMB/iQzsQBf7anAlxxyhmFkHK2M8H6TfvS/GZQdMdJFQ

xcmaWOQi+7GeN4aDO6Z+UO32mRY9rknUpTWVwaq8lekU8TGtKBIPloqThsH5700o

DeoUfjfRt08f5xR6vJgzeHbhYIdSvMtLlZ6avP1DOoSyMy13zbZuAf3CSrwRkRhE

ov7WvKSyv8BTO3WWQwasRqRE5ZkC0Fwhm48mWbNhV6HTYs1ISqNpBncOw6/w1hnZ

v1+w3/jtitg6awSFsJFFKdAWY0Wt4E7POVKjXQgj0pgXRWp1hxKPQD0T/UCxbTpu

ex2xm/udPy5AVCqq0wp1tgbUmF5sJtqpGtsh0p6iW/D7HP/cS/3ClyUgK7S8RM3p

jLjajrq+yGElf+/9E6gycpJfUIBJn71N6q3nu15Gh6NDDx4qA/p32k58IA==

-----END CERTIFICATE-----

On the file server, add an export for the same share to localhost. Set the insecure option, which will allow for connections from client ports above 1024 (I discuss the consequences of this momentarily). If you want to remove the clear-text export, make sure the client has unmounted first:

$ cat /etc/exports

/home/share 5.6.7.8(fsid=0,ro)

/home/share 127.0.0.1(fsid=0,ro,insecure)

Run the following command to activate the share to localhost:

exportfs -a

Add an inetd-style socket activation unit on port 2363 to launch stunnel with a timeout of ten minutes:

$ cat /etc/systemd/system/MC-nfsd.socket

[Unit]

Description=NFS over stunnel/TLS server

[Socket]

ListenStream=2363

Accept=yes

TimeoutSec=600

[Install]

WantedBy=sockets.target

Configure the socket to launch stunnel with a settings file that you'll define shortly:

$ cat /etc/systemd/system/MC-nfsd@.service

[Unit]

Description=NFS over stunnel/TLS server

[Service]

ExecStart=-/bin/stunnel /etc/stunnel/MC-nfsd.conf

StandardInput=socket

Start the socket and enable it for automatic start at boot with the following commands:

systemctl start MC-nfsd.socket

systemctl enable MC-nfsd.socket

Open port 2363 to allow encrypted NFS through your firewall:

iptables -w -I INPUT -p tcp --dport 2363 --syn -j ACCEPT

Create the following stunnel control file for the NFS server:

$ cat /etc/stunnel/MC-nfsd.conf

#GLOBAL#######################################################

TIMEOUTidle = 600

renegotiation = no

FIPS = no

options = NO\_SSLv2

options = NO\_SSLv3

options = SINGLE\_DH\_USE

options = SINGLE\_ECDH\_USE

options = CIPHER\_SERVER\_PREFERENCE

syslog = yes

debug = 0

setuid = nobody

setgid = nobody

chroot = /var/empty/stunnel

libwrap = yes

service = MC-nfsd

; cd /var/empty; mkdir -p stunnel/etc; cd stunnel/etc;

; echo 'MC-nfsd: ALL EXCEPT 5.6.7.8' >> hosts.deny;

; chcon -t stunnel\_etc\_t hosts.deny

curve = secp521r1

; https://hynek.me/articles/hardening-your-web-servers-ssl-ciphers/

↪ciphers=ECDH+AESGCM:DH+AESGCM:ECDH+AES256:DH+AES256:ECDH+

↪AES128:DH+AES:RSA+AESGCM:RSA+AES:!aNULL:!MD5:!DSS

#CREDENTIALS##################################################

verify = 4

CAfile = /etc/stunnel/nfs-tls.pem

cert = /etc/stunnel/nfs-tls.pem

#ROLE#########################################################

connect = 127.0.0.1:2049

Create the chroot() directory where stunnel will drop privileges:

# mkdir /var/empty/stunnel

Attempt a local clear-text socket connection to port 2363; stunnel configuration problems will appear here:

# nc localhost 2363

Clients allowed=500

stunnel 4.56 on x86\_64-redhat-linux-gnu platform

Compiled/running with OpenSSL 1.0.1e-fips 11 Feb 2013

Threading:PTHREAD Sockets:POLL,IPv6 SSL:ENGINE,OCSP,FIPS

↪Auth:LIBWRAP

Reading configuration from file /etc/stunnel/MC-nfsd.conf

FIPS mode is disabled

Compression not enabled

Snagged 64 random bytes from /dev/urandom

PRNG seeded successfully

Initializing inetd mode configuration

Certificate: /etc/stunnel/nfs-tls.pem

Error reading certificate file: /etc/stunnel/nfs-tls.pem

error queue: 140DC002: error:140DC002:SSL

routines:SSL\_CTX\_use\_certificate\_chain\_file:system lib

error queue: 20074002: error:20074002:BIO

routines:FILE\_CTRL:system lib

SSL\_CTX\_use\_certificate\_chain\_file: 200100D:

error:0200100D:system library:fopen:Permission denied

Service [MC-nfsd]: Failed to initialize SSL context

str\_stats: 11 block(s), 355 data byte(s), 638 control byte(s)

In this case, SELinux is enabled, and the type on the key is preventing stunnel from reading it. A chcon command is required to fix this:

# cd /etc/stunnel

# ls -lZ

-rw-r--r--. root root XXX:XXX:stunnel\_etc\_t:s0 MC-nfsd.conf

-r--------. root root XXX:XXX:user\_home\_t:s0 nfs-tls.pem

# chcon -t stunnel\_etc\_t nfs-tls.pem

# ls -lZ

-rw-r--r--. root root XXX:XXX:stunnel\_etc\_t:s0 MC-nfsd.conf

-r--------. root root XXX:XXX:stunnel\_etc\_t:s0 nfs-tls.pem

When you can run the netcat without error, you're ready to move to the client. Add the inetd-style socket activation unit on the NFS client:

$ cat /etc/systemd/system/3d-nfsd.socket

[Unit]

Description=NFS over stunnel/TLS client

[Socket]

ListenStream=2323

Accept=yes

TimeoutSec=300

[Install]

WantedBy=sockets.target

Configure the socket to launch stunnel with a settings file that you'll define shortly:

$ cat /etc/systemd/system/3d-nfsd@.service

[Unit]

Description=NFS over stunnel/TLS client

[Service]

ExecStart=-/bin/stunnel /etc/stunnel/3d-nfsd.conf

StandardInput=socket

Create a stunnel control file for the NFS client:

$ cat /etc/stunnel/3d-nfsd.conf

#GLOBAL#######################################################

sslVersion = TLSv1.2

TIMEOUTidle = 600

renegotiation = no

FIPS = no

options = NO\_SSLv2

options = NO\_SSLv3

options = SINGLE\_DH\_USE

options = SINGLE\_ECDH\_USE

options = CIPHER\_SERVER\_PREFERENCE

syslog = yes

debug = 0

setuid = nobody

setgid = nobody

chroot = /var/empty/stunnel

libwrap = yes

service = 3d-nfsd

; cd /var/empty; mkdir -p stunnel/etc; cd stunnel/etc;

; echo '3d-nfsd: ALL EXCEPT 127.0.0.1' >> hosts.deny;

; chcon -t stunnel\_etc\_t hosts.deny

curve = secp521r1

; https://hynek.me/articles/hardening-your-web-servers-ssl-ciphers/

↪ciphers=ECDH+AESGCM:DH+AESGCM:ECDH+AES256:DH+AES256:

↪ECDH+AES128:DH+AES:RSA+AESGCM:RSA+AES:!aNULL:!MD5:!DSS

#CREDENTIALS##################################################

verify = 4

CAfile = /etc/stunnel/nfs-tls.pem

cert = /etc/stunnel/nfs-tls.pem

#ROLE#########################################################

client = yes

connect = nfs-server.yourco.com:2363

Note: I've referred to the server by the IP address 1.2.3.4 previously, but above it is nfs-server.yourco.com—use whatever form of your hostname you prefer.

The latest Ubuntu is equipped with a "stunnel4", which is actually stunnel version 5.44. It does not run with the NO\_SSLv2 or either of the SINGLE\_\*\_USE options (you must remove them), and the group "nogroup" should be used there for the setgid option above.

Modify the fstab entry for /home/share to connect to the local stunnel:

$ grep share /etc/fstab

localhost:/ /home/share nfs noauto,vers=4.2,proto=tcp,port=2323 0 0

Mount the volume, and check for a stunnel process, and then examine the active network connections:

# mount /home/share

# pps stun

PID TTY STAT TIME COMMAND

5870 ? Ss 0:00 /bin/stunnel /etc/stunnel/3d-nfsd.conf

# netstat -ap | grep nfsd

tcp 0 0 localhost:860 localhost:3d-nfsd

↪ESTABLISHED -

tcp 0 0 squib:48804 192.168.:mediacntrlnfsd

↪ESTABLISHED 5870/stunnel

tcp6 0 0 [::]:3d-nfsd [::]:\*

↪LISTEN 1/init

tcp6 0 0 localhost:3d-nfsd localhost:860

↪ESTABLISHED 1/init

# ls -l /home/share/

total 676

-rw-r--r-- 1 root root 158 May 21 18:58 hosts

-rw-rw-r-- 1 cfisher cfisher 5359 May 21 19:22 nfs-tls.pem

-rw-r--r-- 1 root root 1760 May 21 18:58 nsswitch.conf

-rw-r--r-- 1 nobody nogroup 1921 May 21 19:17 passwd

-rw-r--r-- 1 root root 670293 May 21 18:58 services

Also, examine the server's stunnel process and network status:

# pps stun

PID TTY STAT TIME COMMAND

16282 ? Ss 0:00 /bin/stunnel /etc/stunnel/MC-nfsd.conf

# netstat -ap | grep nfsd

tcp6 0 0 [::]:mediacntrlnfsd [::]:\*

↪LISTEN 1/systemd

tcp6 0 0 192.168.:mediacntrlnfsd 192.168.0.24:48824

↪ESTABLISHED 1/systemd

Squashed permissions may be recorded in your syslog:

rpc.idmapd[4321]: nss\_getpwnam: name 'cfisher@yourhost'

does not map into domain 'localdomain'

To remedy this, you'll need to set the domain in /etc/idmapd.conf manually.

A major problem on the NFS client is the ability of any local users to connect to the NFS endpoint via SSH or other port-forwarding tools. They can forward this to a server of their choosing (and under their control) to mount and manipulate the remote file server. Any local user on the client is able to:

# telnet localhost 2323

Trying 127.0.0.1...

Connected to localhost.

Escape character is '^]'.

Connection closed by foreign host.

The ability to connect to the endpoint grants the ability to control it.

There are no native stunnel options to restrict client access to privileged ports, but you can write a wrapper of your own to restrict this access—it calls an exec() function to start stunnel after verifying that the incoming port is privileged, passing the active file descriptors to the replacement process. To engage this wrapper, place the following file:

# cat /bin/pstunnel.c

#include <stdio.h>

#include <unistd.h>

#include <arpa/inet.h>

int main(int argc, char \*argv[], char \*envp[])

{

struct sockaddr\_storage addr;

socklen\_t len = sizeof addr;

int port = 65535, bad = 0;

if(getpeername(fileno(stdin), (struct sockaddr \*) &addr, &len)) bad = 1;

else if(addr.ss\_family == AF\_INET) //IPv4

{

struct sockaddr\_in \*s = (struct sockaddr\_in \*) &addr;

port = ntohs(s->sin\_port);

}

else if(addr.ss\_family == AF\_INET6) //IPv6

{

struct sockaddr\_in6 \*s = (struct sockaddr\_in6 \*) &addr;

port = ntohs(s->sin6\_port);

}

else bad = 1;

if(!bad && port < IPPORT\_RESERVED) execve("/bin/stunnel", argv, envp);

else printf("Nope.\n");

}

Compile the privileged wrapper with the following commands:

# cd /bin

# cc -s -O2 -DFORTIFY\_SOURCE=2 -Wall -o pstunnel pstunnel.c

Modify the socket unit file to call the privileged wrapper:

# cat /etc/systemd/system/3d-nfsd@.service

[Unit]

Description=NFS over stunnel/TLS client

[Service]

ExecStart=-/bin/pstunnel /etc/stunnel/3d-nfsd.conf

StandardInput=socket

Then reload systemd to recognize the modified unit:

# systemctl daemon-reload

Connections from non-privileged clients are now blocked, but mount requests still will pass:

# telnet localhost 2323

Trying 127.0.0.1...

Connected to localhost.

Escape character is '^]'.

Nope.

Connection closed by foreign host.

# mount /home/share

# pps stun

PID TTY STAT TIME COMMAND

2483 ? Ss 0:00 /bin/pstunnel /etc/stunnel/3d-nfsd.conf

# umount /home/share

Note that argv[0] will retain the name of the wrapper.

Rather than simply print "Nope", you might adjust your wrapper to trigger notifications that unprivileged users are abusing your endpoint—a matter of some seriousness.

The pstunnel.c wrapper doesn't work quite as expected under Oracle Linux 5. Any active NFS mount will be reported by netstat as originating from a privileged client port, but mount attempts will fail after moving to the privileged wrapper in xinetd. An observed workaround is to mount without the wrapper, switch the xinetd configuration to pstunnel, then allow the stunnel timeout to expire, causing new stunnels spawned to service the existing connection to enforce privileged ports. It appears that the cause of this problem is a preliminary non-privileged client connection when the mount is established (perhaps the STATD\_OUTGOING\_PORT parameter in nfsconf is the culprit). This workaround might be useful on other operating systems, so I've included it here even though Oracle Linux 5 is out of support.

If you're on a system that doesn't block remote connections to the 2323 endpoint with a firewall, you should use the libwrap feature documented above in the client stunnel controlfile to restrict access to localhost. The libwrap features are less useful on the server, where the RSA keypair must be presented before access is allowed.

Be advised that Microsoft Windows has NFS clients available, but the platform does not observe limitations on the privileged ports under 1024—any Windows user is allowed to originate connections from these restricted ports, so low port filtering will not be an effective security control. If you export NFS volumes to a windows client, you must trust *all* of the client's users.

Note also that the insecure option on the NFS server will allow local users there to do similar mischief. Linux iptables has an owner match module that can be locked to root, which may be able to protect the server's vulnerable port 2049 similarly. If you cannot protect the NFS server from users establishing subversive local connections, you shouldn't have any untrusted local users on it.

Finally, be aware that the following socket options in your stunnel control files might be very useful for NFS:

socket = a:TCP\_NODELAY=1

socket = a:SO\_KEEPALIVE=1

The NODELAY option disables the Nagle algorithm, which prevents delays in your NFS traffic at the expense of (potentially) sending "tinygrams"—stunnel will not wait in the hope of a full packet to send, which should make operations on small amounts of data more responsive. If you will be exchanging large amounts of data constantly, this option might not be as helpful.

NFSv4 has deep file locking and "delegations" where a client can "check out" a file from a server for an indefinite time. The server must be able to contact the client to cancel the delegation and obtain the current contents if the file is requested by another client, which will not occur if the stunnel connection shuts down. The client can restart the connection automatically if/when it has activity for the server, but the reverse is not true, which might impact locks and delegations. Although the server can disable delegations system wide with the command echo '0' > /proc/sys/fs/leases-enable, the KEEPALIVE option might be a helpful alternative, and is left as a topic of research for the reader.

Performance Benchmarks

For those with real data security concerns, performance is irrelevant; sensitive information cannot be allowed over clear-text connections. Still, it's important to understand the price that must be paid for the encryption overhead, so I've performed a few simple tests involving NFSv4 to make the penalty clear.

Linux once had an NFS server implemented entirely in userspace, but this was moved into the kernel for Linux 2.2 to improve performance (there is still a [userspace NFS server](https://github.com/nfs-ganesha/nfs-ganesha/wiki) under active development that is useful for specific applications, notably FUSE). I had expected a heavy speed penalty in forcing a trip back into userspace for the stunnel on each side, but the impact was far less than anticipated.

My test was performed on two HP DL360 G9 servers running recent releases of the Oracle Unbreakable Enterprise Kernel v4 (UEK). The test involved pushing a copy of the Oracle Linux 7.5 install ISO to the server, under both clear-text NFS and TLS.

I made an attempt to clear the caches on both the client and server before sending any data over NFS:

# sync && echo 3 > /proc/sys/vm/drop\_caches

I removed any copy of the ISO on the server from the previous test:

# rm /home/share/V975367-01.iso

rm: remove regular file '/home/share/V975367-01.iso' y

I then verified the Oracle-supplied sha256 ISO hash on the client side in an effort to get the ISO's contents into the client's buffer cache:

# tail -1 sha256

D0CC4493DB10C2A49084F872083ED9ED6A09CC065064C009734712B9EF357886

↪V975367-01.iso

# sha256sum -c < sha256

V975367-01.iso: OK

At this point, I mounted the server over a clear-text NFSv4.2 connection:

# tail -1 /etc/fstab

1.2.3.4:/ /home/share nfs noauto,vers=4.2,proto=tcp,port=2049 0 0

# mount /home/share

Then I ran three iterations of the copy, clearing caches between each run:

# time cp V975367-01.iso /home/share

real 0m39.697s

user 0m0.005s

sys 0m2.173s

# time cp V975367-01.iso /home/share

real 0m39.927s

user 0m0.005s

sys 0m2.159s

# time cp V975367-01.iso /home/share

real 0m39.489s

user 0m0.001s

sys 0m2.218s

The average wall clock time to move the ISO over a clear-text connection was 39.70 seconds. I then reconfigured to use stunnel:

# tail -1 /etc/fstab

localhost:/ /home/share nfs noauto,vers=4.2,proto=tcp,port=2323 0 0

# mount /home/share

And ran the tests again:

# time cp V975367-01.iso /home/share

real 0m39.476s

user 0m0.002s

sys 0m2.265s

# time cp V975367-01.iso /home/share

real 0m40.376s

user 0m0.005s

sys 0m2.189s

# time cp V975367-01.iso /home/share

real 0m41.971s

user 0m0.001s

sys 0m2.894s

The average time taken for the encrypted connection was 40.61, a difference of 2.2% (hardly a high price to pay).

The DL380 servers have CPUs that implement AES-NI native machine instructions, which likely boosted performance. Configuring stunnel for high logging (setting debug=debug), the reported cipher used was ECDHE-RSA-AES256-GCM-SHA384. Systems without AES-NI recognized by OpenSSL will not perform this well.

I also tested this activity with fuse-sshfs from the [EPEL](https://fedoraproject.org/wiki/EPEL) repository. I unmounted NFS, installed the RPM, then reconnected to the remote target:

# sshfs cfisher@1.2.3.4:/home/share /home/share

The authenticity of host '1.2.3.4 (1.2.3.4)' can't be established.

ECDSA key fingerprint is

↪4c:90:f8:48:2e:03:f5:31:30:c1:73:a3:5e:da:42:d3.

Are you sure you want to continue connecting (yes/no)? yes

cfisher@1.2.3.4's password:

I then reran the tests:

# time cp V975367-01.iso /home/share

real 0m38.727s

user 0m0.039s

sys 0m4.733s

# time cp V975367-01.iso /home/share

real 0m39.498s

user 0m0.035s

sys 0m4.751s

# time cp V975367-01.iso /home/share

real 0m39.536s

user 0m0.030s

sys 0m4.763s

The average for sshfs was 39.25 seconds, 3.3% faster than NFSv4 over stunnel. There have been [other tests](http://prokop.uek.krakow.pl/projects/fs_benchmark.html) that indicate NFS to be faster, but I did not see that behavior, although this test might not have performed enough activity under sufficiently rigorous conditions to reveal the discrepancy.

NFS is preferable to sshfs in several scenarios, despite any performance differences. More filesystem features are supported (such as the df command as mentioned in the FAQ), NFS implements dynamic id mapping (sshfs accepts only static maps), and NFS clients with or without stunnel will restart broken TCP connections automatically, allowing long-term mounts to be maintained reliably in adverse network conditions. OpenSSH is a tool focused on interactive use; the client was not intended to run out of inetd as stunnel does, and stunnel is more suited to basic automated services for these reasons.

Conclusion

In the decades of NFSv4 development, it is astonishing that a simple symmetric cipher was overlooked in the stampede of new features into the protocol. Version 4.2, published in November 2016 in [RFC 7862](https://tools.ietf.org/html/rfc7862), was recent enough for the authors to be painfully aware of the abuse of plain-text traffic. The omission was likely intentional, and an AEAD suite in common use (that is, AES-GCM and/or ChaCha20-Poly1305) should be retrofitted immediately upon all versions of NFS in supported products.

The sec=krb5p option will encrypt NFSv4 traffic in a Kerberos realm, but requiring this infrastructure is inappropriate in hosted environments and is generally far from helpful. Basic access to symmetric cryptography does not and should not mandate such enormous baggage.

It is increasingly obvious that we cannot trust our networks. [Cisco](https://www.theregister.co.uk/2018/05/18/network_roundup_17_may_2018) has again been found with hard-coded back doors in its products. As I write this, an [FBI advisory](https://www.ic3.gov/media/2018/180525.aspx) is in effect [requesting](https://arstechnica.com/information-technology/2018/05/fbi-tells-router-users-to-reboot-now-to-kill-malware-infecting-500k-devices) a reboot of home and small office routing equipment due to malware penetration by unknown vectors. The internet is awash in compromised devices, because we don't patch the software that runs this infrastructure. Assuming compromise and encrypting all traffic has become the only reasonable stance.

While the crusade against telnet may have been largely won, Linux and the greater UNIX community still have areas of willful blindness. NFS should have been secured long ago, and it is objectionable that a workaround with stunnel is even necessary. Sensitive data should not be shared with unknown sources. Until protocol and kernel architects take this to heart, use stunnel to wrap your NFS.

Addendum

An alternate approach for encrypting NFS traffic has [recently](https://marc.info/?l=linux-netdev&m=153306429108040&w=2) been submitted for inclusion into the Linux kernel. [Wireguard](https://www.wireguard.com/) is a VPN implementation that strives for a minimal code footprint (a mere 4,000 lines) to simplify auditing for assurances of implementation correctness.

Wireguard's cryptography is intentionally limited to DJB's Curve25519 and ChaCha20-Poly1305. The documentation [expounds](https://www.wireguard.com/papers/wireguard.pdf) that "WireGuard is cryptographically opinionated. It intentionally lacks cipher and protocol agility. If holes are found in the underlying primitives, all endpoints will be required to update." Pairs of systems exchange 25519 public keys, which are recorded in a simple configuration file (in exactly the way that we would hope to see in /etc/fstab for NFS mounts). For those with concerns of Shor's Algorithm running on a capable quantum computer, "WireGuard also supports a mode in which any pair of peers might additionally pre-share a single 256-bit symmetric encryption key between themselves" (at the potential cost of forward secrecy).

Once encrypted, Wireguard converts all traffic to UDP prior to transmission. Since all processing occurs inside the kernel, performance is very good and reportedly beats OpenVPN even on systems with native AES machine instructions. In theory, it would be safe to go back to telnet and FTP over Wireguard connections.

Wireguard is not and cannot be a complete solution to NFS security. While implementations also [exist](https://www.theregister.co.uk/2018/08/02/linux_kernel_wireguard/) for macOS, Windows and other platforms, stunnel will be far more portable to a larger subset of the POSIX community since it runs entirely in userspace. Many will prefer the TCP transport of stunnel to UDP Wireguard traffic for a myriad of reasons. Some may also prefer the cryptographic flexibility of TLS, especially when combined with the chroot() capability of stunnel.

Still, it is encouraging that Curve25519 and ChaCha20-Poly1305 are likely to have a key role in the Linux kernel. Perhaps the NFSv4.3 RFC will mandate them and force these tools into the kernels of commercial UNIXen and other NFS implementors.

Wireguard and stunnel will likely be complementary security tools. Each can serve markets that are inaccessible to the other, and both will likely have some role to play in NFS security for the immediate future.