Snowshill, Fotolia

NFS 3 and the specter of the spoof attack

GHOST ON THE LOOSE

Host-based authentication is easy to configure, but it doesn't do much to stop uninvited guests. **BY NILS MAGNUS**

The CIO of a famous credit union was fairly sure he had thought of everything: state-of-the-art antivirus tools, separate zones for desktops and servers, firewalls at strategic points, and regular operating system updates. He also favored “stable versions” of operating systems, applications, and protocols for added assurance.

Unfortunately, this CIO decided to take a short cut in setting up the file server: NFS version 3 [1] with the legacy “host-based access control” and no special hardening. Although NFS 4 has been around for several years, many networks continue to use NFS 3 because it is simpler and it is already up and running.

Ease of use is certainly an important consideration; however, in the case of NFS, this simplicity sometimes comes with a cost. The host-based scheme used with NFS 3 has several significant problems. Also, NFS 3 transmits unencrypted data blocks (see Figure 1), it only performs rudimentary authentication, and its authorization mechanisms are easily avoided (see the “NFS Security Functions” box).

In this article, you will get a glimpse of one of the many catastrophes that can happen if you use NFS 3 without paying attention to security. Ultimately, the best solution to these sorts of problems is to upgrade to version 4, but for readers who don’t have the option of an upgrade, I offer some thoughts on how to avoid these kinds of shenanigans.

Compromise

To explore some of the security issues related to NFS, *Linux Magazine* staged a typical attack on an NFS configuration, during which, we looked at the individual steps an attacker would take to grab the crown jewels of an unwary IT department. This scenario assumes the blackhat can access the internal network. It is unlikely that an attacker would be able to launch such an attack from across the Internet because most enterprise firewalls will succeed in preventing the intrusion.

But think about those spare network sockets in your conference room or the systems left on overnight with cleaning staff (or anyone else) poking around. Another technique gaining favor with attackers is to attach a small WLAN router to the network and leave it hidden in an empty cubicle. In other cases, the intruder might not need to sneak around at all because the attack is launched by an employee of the company.

Finding a Server

The first thing an attacker needs to do is search for NFS servers that offer their service on the enterprise network. If the IP address is unknown, an attacker will probably opt for the Nmap scanner [2], the tool of choice for over a decade: After identifying the possible IP range of the target for the attack, by analyzing a sniffer log, for example, the attacker will launch a targeted search. The command *nmap -n -v -sS -P0 -p111 address_range* searches for RPC port mappers behind TCP port 111.

Compared with a full scan with several thousand ports per system, the following search is fairly quick. Matches are stored in the *upips.txt* file and are then available for closer inspection:

```
nmap -n -v -sSVR -O -i upips.txt
```

The scan method *-s* requested here is a half-open scan (*S*); if this scan is successful, a version scan (*V*), and a query to the RPC calls (*R*) will follow. The results for an NFS server will look something like the output in Listing 1.

Going to Work

NFS uses RPC for remote function invocation, introducing separate data encoding in the form of External Data Representation (XDR). The Nmap scan at least gives a clear view of the RPC calls. The *rpcbind* in line 5 forwards RPC queries to other ports, such as the ports for *mountd* (line 7), which is responsible for setting up a connection, or the *nfs* daemon (line 8), which delivers or accepts data. Figure 2 shows this configuration.

An attacker can glean valuable information from these services. In fact, the attacker doesn't even need special hacking tools: The admin tools provided with the distribution are just fine. One of them is called Showmount. Typing *showmount -e alpha* will output the NFS server's exports:

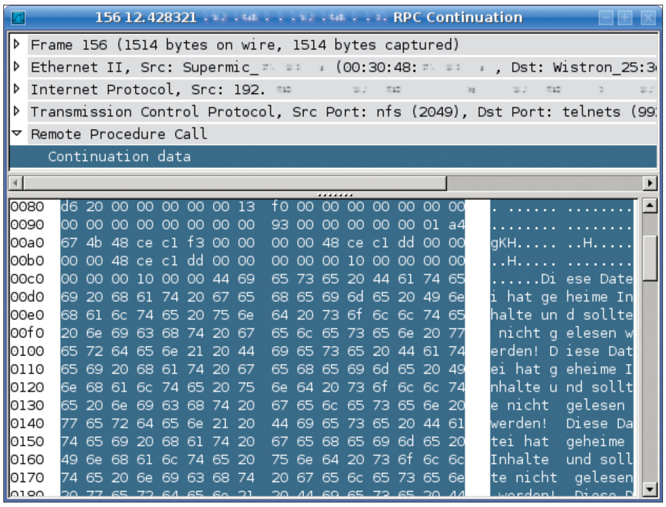
```
Export list for alpha:
/opt/someapp    *
/mnt/cdrom      everyone(ro)
/home          *.example.com,↩
192.168.1.66
```

NFS Security Functions

When mounting a volume, legacy NFS servers only authenticate by referring to the client IP address and a privileged port, both of which are easy to spoof. NFS provides a *handle* to an entity that presents credentials. Any entity presenting the handle can access the data. It is left up to the client to check access privileges. Because the protocol does not encrypt, both the handles and data are easily sniffed (see Figure 1). However, an attacker can't leverage a handle without some effort because it involves some programming. The method described in this article is far easier, and it uses only standard Linux tools.

To improve NFS, Sun originally planned on securing RPC connections via Secure RPC. But documents that cast doubt on its viability were published as early as 1995. The attack, which took a mainframe a couple of days back then, takes a modern PC just a couple of hours.

The NFS architects finally dropped Secure RPC and focused on NFS 4, which offers more in the line of security functions. Because it relies entirely on TCP, it is easier to protect with firewalls and VPNs. The RPCSEC_GSS API that NFS 4 uses also supports the use of tried-and-trusted security architectures like Kerberos for authentication purposes.



```
156 12.428321 [redacted] RPC Continuation
▷ Frame 156 (1514 bytes on wire, 1514 bytes captured)
▷ Ethernet II, Src: Supermic_[redacted] (00:30:48:[redacted], Dst: Wistron_25:3
▷ Internet Protocol, Src: 192.[redacted]
▷ Transmission Control Protocol, Src Port: nfs (2049), Dst Port: telnets (99
▽ Remote Procedure Call
    Continuation data

0080  d6 20 00 00 00 00 00 13  f0 00 00 00 00 00 00 00   . ...... ........
0090  00 00 00 00 00 00 00 00  93 00 00 00 00 00 01 a4   ........ ........
00a0  67 4b 48 ce c1 f3 00 00  00 00 48 ce c1 dd 00 00   gKH..... ..H.....
00b0  00 00 48 ce c1 dd 00 00  00 00 00 10 00 00 00 00   ..H..... ........
00c0  00 00 00 10 00 00 44 69  65 73 65 20 44 61 74 65   ......Di ese Date
00d0  69 20 68 61 74 20 67 65  68 65 69 6d 65 20 49 6e   i hat ge heime In
00e0  68 61 6c 74 65 20 75 6e  64 20 73 6f 6c 6c 74 65   halte un d sollte
00f0  20 6e 69 63 68 74 20 67  65 6c 65 73 65 6e 20 77    nicht g elesen w
0100  65 72 64 65 6e 21 20 44  69 65 73 65 20 44 61 74   erden! D iese Dat
0110  65 69 20 68 61 74 20 67  65 68 65 69 6d 65 20 49   ei hat g eheime I
0120  6e 68 61 6c 74 65 20 75  6e 64 20 73 6f 6c 6c 74   nhalte u nd sollt
0130  65 20 6e 69 63 68 74 20  67 65 6c 65 73 65 6e 20   e nicht  gelesen
0140  77 65 72 64 65 6e 21 20  44 69 65 73 65 20 44 61   werden!  Diese Da
0150  74 65 69 20 68 61 74 20  67 65 68 65 69 6d 65 20   tei hat  geheime
0160  49 6e 68 61 6c 74 65 20  75 6e 64 20 73 6f 6c 6c   Inhalte  und soll
0170  74 65 20 6e 69 63 68 74  20 67 65 6c 65 73 65 6e   te nicht  gelesen
0180  20 77 65 72 64 65 6e 21  20 44 69 65 73 65 20 44    werden!  Diese D
```

Figure 1: NFS 3 transmits unencrypted data. In this figure, Wireshark has just sniffed NFS access to a text file.

The first column contains the exported directory on server *alpha* – NFS refers to this as a volume. The second column tells which clients are permitted to access the volume. Different NFS servers have different ways of writing permitted client names, but whatever the syntax, the address will include IP addresses and DNS names. An asterisk or the *everyone* string indicates no restrictions for the volume. In many cases, admins use this option for CD-ROMs or static, read-only data. (The *(ro)* entry specifies read only.)

The NFS server resolves DNS names itself and thus authenticates querying clients on the basis of IP addresses. Unfortunately, it is really easy to spoof IP addresses on the local network. In this example, any DNS name from the *example.com* domain is permitted.

Querying the DNS Server with *dig example.com axfr* could return a complete list of all known hostnames and matching addresses. As an alternative, the attacker can just make an intelligent guess if a couple of hostnames are known or if some hosts have been identified with the use of a sniffer. In any case, the intruder will put together a list of matching IP addresses and check which ones they can hijack.

The addresses need to be in the segment the attacker has access to. Often this is the case if a larger number of clients mount home directories on the NFS server.

The next step is to find an authorized IP address that is currently unused. Many users switch off their computers at the end of the day. The command

```
ping -c3 192.168.1.66
```

checks to see whether the specified host is active. To do so, the tool sends three ICMP packets to the client (see Listing 2). If you want to be completely certain that the target is not simply blocking ICMP, you can additionally check whether *arp -na* resolves the address:

```
? (192.168.1.1) to ↩
00:31:42:42:42:13 eth0
? (192.168.1.66) to ↩
<incomplete>  eth0
```

Although a MAC address is known for 192.168.1.1, ARP was unable to resolve the target. In other words, the address is free.

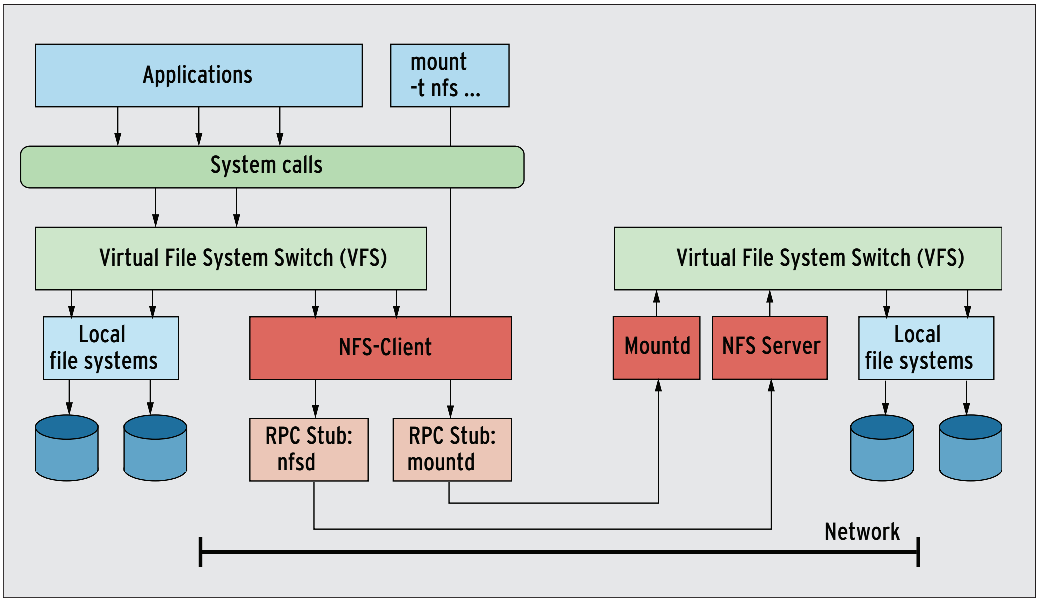


Figure 2: NFS 3 security suffers because the filesystem relies on authenticated identities for access. The process relies on the credentials presented by the querying client.

On a Linux system, IP addresses are easily spoofed: root can just configure them on the interface:

```
ifconfig eth0 192.168.1.66 up
```

This hack will not work with virtual interfaces such as *eth0:1*: NFS clients use the first address on their interface for requests and do not configure any additional IPs. After completing the preparatory steps, the attacker can become root and type

```
mkdir /tmp/MNT
mount -t nfs ⮐
192.168.1.200:/home /tmp/MNT
```

to create a local mount point and then mount the */home* volume exported by NFS. If you just see a prompt after this, the connection is working.

An intruder operating as local root probably won't be able to access this volume directly. Although the superuser can typically read anything on the local disk, the NFS volume suddenly blocks read access. This effect is typically referred to as a *root_squash* configuration, but the setback is temporary and easy to work around.

Typing *ls -ln* outputs the numeric user IDs for the file owners. The attacker can then create entries for these users in the local */etc/passwd* file and type *su - victim* to assume their identity. The intruder is now free to browse user folders, exchange SSH keys, or poke around in mail directories. In the case of the credit union, the attacker found customer data backup scripts, including clear text passwords for the database.

Conclusions

What should the CIO do differently next time? The best solution is to upgrade to NFS 4, which comes with much more sophisticated security features, such as Kerberos authentication and GSS-API support. Configuring these additional components definitely takes some effort, but the result is a much more secure environment.

If, for whatever reason, you have a need to continue with NFS 3, keep the following tips in mind:

- Make sure you don't export rw-volumes to everyone.
- Keep reasonable control over your IP addresses. For instance, use a physically separate address space and make sure no one has logical access to it, impose some form of Layer 2 authentication (such as IEEE 802.1x) for all clients on the segment, or use VLANs with IEEE 801.1q tagging for communication between NFS servers and clients.
- If feasible, use dedicated VPN tunnels to protect and authenticate NFS traffic.

Of course, if you add all these additional security structures to your NFS 3 configuration, your system could end up much more complex than if you had simply upgraded to NFS 4, but at least you'll sleep better knowing you have patched some of the cracks in NFS. ■

Listing 1: Scan Results for an NFS Server

```
Interesting ports on 192.168.1.200:
PORT     STATE SERVICE                    VERSION
21/tcp   open  ftp                        ProFTPD 1.3.0
22/tcp   open  ssh                        OpenSSH 4.3p2 Debian 8ubuntu1.4 (protocol
2.0)
111/tcp  open  rpcbind (rpcbind V2)        2 (rpc #100000)
143/tcp  open  imap                       Cyrus IMAP4 2.2.13-Debian-2.2.13-1
869/tcp  open  mountd (mountd V1-3)        1-3 (rpc #100005)
2049/tcp open  nfs (nfs V2-4)             2-4 (rpc #100003)
MAC Address:   00:aa:ff:ee:01:23           (Fortytwo Systems)
Service Info:  Host: alpha.example.com; OSs: Unix, Linux
```

Listing 2: Checking the Target Address

```
# ping -c3 192.168.1.66
PING 192.168.1.66 (192.168.1.66) 56(84) bytes of data.
From 192.168.1.175   icmp_seq=1   Destination Host Unreachable
From 192.168.1.175   icmp_seq=2   Destination Host Unreachable
From 192.168.1.175   icmp_seq=3   Destination Host Unreachable

--- 192.168.1.66 ping statistics ---
3 packets transmitted, 0 received, +3 errors, 100% packet loss, time 2002ms
```