Network File System (NFS)

Coleman Kane
Coleman.Kane@ge.com

February 3, 2016

Network File System (NFS)

One of the most common schemes for providing network-based file shares within a network is NFS. The protocol is ubiquitous and supported generally by every UNIX-like variant, as well as being supported by operating systems by Microsoft and Apple. The system provides a system for permission management, as well as user and role-based access controls. The protocol dates itself to the 1980's, with the current version released in 2000 and an update in 2010.

Components

Like many network servers, NFS is implemented as a suite of function-specific components that together build a functioning NFS implementation. This includes:

- RPC services to provide service identification, availability, and NFS file-locking
- NFS daemon to provide the file access
- Server configurations which define access controls and what portions of the server to export

Service Access

In many cases, the following two components will be network-attached services executing with `root` permissions on the server.

- RPC services
- NFS Daemon

In the above, the RPC dependency would be considered unituitive out-of-the-box, and as such can introduce an unexpected vulnerability into the system.

RPC Services

In many NFS configurations, there are four RPC services responsible for supporting the system:

port mapper Typically either `rpcbind` or `portmap`, this service is running on a known port (`111`). NFS was designed to publish itself to this system as a listening service.

lock daemon Typically `rpc.lockd`, designed to manage file-locking operations across the network mounts on different clients.

stat daemon Works in tandem with *lock daemon* to facilitate crash/reboot recovery

mount daemon Facilitates mount / unmount requests (session management for NFS)

The above list is not comprehensive, and there exist other services providing additional features as well.

RPC Vulnerability: statd & lockd

Both of these services are provided for supporting additional features to clients. In most cases, clients will support blocking lock operations if they are running. Buggy versions of these tools can introduce vulnerabilities into the system:

- Client denial of service after crash or deadlock
- Server-side unauthorized `root` access and even RCE

Apple OS X <= 10.6.6 has a good example in `CVE-2011-0183`

RPC Vulnerability: statd SIGPIPE

In 2004, `rpc.statd` commonly installed on many UNIX systems and Linux systems was identified to contain a vulnerability where it incorrectly handled SIGPIPE.

PIPE signal is typically sent when a file handle is closed outside the current process. It tells the current process that the other end of the handle closed, such as the remote side of connections.

Access Controls

Access control is managed through a text file named `/etc/exports`. This file allows you to control many access options.

- Host-based (domain or IP) access control
- Restrict to parts of filesystem hierarchy
- Remapping of user Ids (prevent root access, etc.)