OS (Server)

Lecture 6
Network Services in Linux: Sharing Files

Dr. Kevin Farrell

Lecture Overview

- Sharing *Non-System* Files: NFS Theory:
  - Intro
  - Components
  - Protocol Versions
  - Stateless Mounting
  - Security
- Sharing *System* Files: Brief Overiew of:
  - NIS
  - NIS+
  - LDAP
  - References

Introduction

- The Network File System (NFS) allows you to share filesystems among computers
- NFS is almost transparent to users and “stateless”, meaning that no information is lost when an NFS server crashes. Clients can simply wait until the server returns and then continue as if nothing had happened.
- Introduced by Sun Microsystems in 1985.
- Originally implemented as a surrogate filesystem for disk-less clients
- But protocol proved to be well designed and very useful as a general file-sharing solution
- Nearly all modern Linux distributions have at least minimal support for NFS.

NFS Components and General Info

- NFS consists of a number of components:
  - a mounting protocol
  - mount server
  - daemons that coordinate basic file service, and
  - several diagnostic utilities
- A portion of both the server-side and client-side software resides in the kernel.
- However, these parts of NFS need no configuration and are largely transparent from an administrator's point of view.

NFS Protocol Versions

- NFS protocol has been remarkably stable over time.
- Now two versions of the protocol available, simply:
  - Version 2, and
  - Version 3
- Version 2 is the original public release of NFS
- Version 3 appeared in the early 1990s
- Version 4 is still under development and is not yet supported by Linux

NFS Protocol Version 2

- Original public release of NFS
- NFS clients cannot assume that a write operation is complete until they receive an acknowledgement from the server =>
  - servers must commit each modified block to disk before replying to avoid discrepancies in the event of a crash.
  - This constraint introduces a significant delay in NFS writes, since modified blocks would normally be written only to the in-memory buffer cache.

NFS Protocol Version 3

- Appeared in the early 1990s
- A collection of changes were integrated into the protocol, which significantly increases performance, and provides better support for large files:
  - eliminates the bottleneck of version 2 (on previous slide) with a coherency scheme that makes writes safely ***asynchronous***
  - It also updates several other aspects of the protocol that were found to have caused performance problems.

Choice of Transport

- NFS runs on top of Sun's RPC (Remote Procedure Call) protocol:
  - a system-independent way for processes to communicate over a network.
  - Side effect of this architecture => possible to use either UDP or TCP as the underlying transport protocol.
- NFS originally used UDP because that was what performed best on the LANs and computers of the 1980s.
- UDP lacks the congestion control algorithms that are essential for good performance on a large IP network.
- Therefore, consensus is that TCP is usually the best option for both local and Internet NFS traffic

Cookies and Stateless Mounting 1

- A client must explicitly mount an NFS share before using it, just as a client must mount a filesystem stored on a local disk.
- However, because NFS is stateless, the server does not keep track of which clients have mounted each filesystem.
- Instead, the server simply discloses a secret "cookie" at the conclusion of a successful mount negotiation.
- The cookie identifies the mounted directory to the NFS server and so provides a way for the client to access its contents.

Cookies and Stateless Mounting 2

- Unmounting and remounting a filesystem on the server normally changes its cookie.
- As a special case, cookies persist across a reboot so a server that crashes can return to its previous state.
- Once a client has a magic cookie, it uses the RPC protocol to make requests for filesystem operations such as creating a file or reading a data block.
- Because NFS is stateless, the server doesn't care what requests the client has or hasn't made before.
- In particular, the client is responsible for making sure that the server acknowledges write requests before it deletes its own copy of the data to be written.

Security and NFS 1

- NFS provides a convenient way to access files on a network => great potential to cause security problems.
- Protocol originally designed with essentially no concern for security.
- Access to NFS volumes is granted by a file called **/etc/exports** which enumerates the hostnames or IP addresses of client-systems that should have access to a server's filesystems. This has problems:
  - This is a weak form of security because the server trusts the clients to tell it who they are.
  - It's easy to make clients lie about their identities, so this mechanism cannot be fully trusted.
- **Rule**: Only export filesystems to clients that you trust
- **Rule**: Always check that you have not accidentally exported filesystems to the whole world.

Security and NFS 2

- The TCP wrappers package (**tcpd**) can help limit the hosts that can access NFS filesystems.
- Recommended that you deny access to the **portmap** service to everyone:
  - Edit the **/etc/hosts.deny**
  - Enable access for trusted hosts and subnets in **the /etc/hosts.allow**
  - Never put anything but IP addresses on the portmap lines in these files because the hostname lookups themselves may require access to the portmap daemon, resulting in a loop!

Security and NFS 3

- As on local filesystems, file-level access control on NFS filesystems is managed according to UID, GID and file permissions.
- But once again, the NFS server trusts the client to tell it who is accessing files. This again has problems:
  - If two users share the same UID on two separate clients, they will have access to each other's NFS files.
  - Users that have root access on a system can change to whatever UID they want; the server will happily give them access to the corresponding files.
- For these reasons, it is strongly recommended to use globally unique UIDs and the **root_squash** option; the latter which "squashes" root down to the status of an ordinary user, with ordinary user abilities, and no rootly powers!

Security and NFS 4

- You can use the **all_squash** option to map ***all*** client UIDs to the ***one*** UID on the server.
- This configuration eliminates all distinctions among users and creates a public-access filesystem of sorts.
- Vulnerabilities still exist in this scenario, because system logins such as “bin” and “sys” aren't UID-mapped, so that any files they own, such as the occasional system binary or third-party application may be vulnerable to attack.
- One real advantage of UID mapping is that it prevents access to files that are owned by root and which are not world-readable or world-writable.

Sharing System Files

- Introduction
- Brief Overview of:
  - NIS
  - NIS+
  - LDAP

Introduction

- A properly functioning system depends on maintaining a considerable number of configuration Files
- In a networked environment, manual maintenance of these configuration files on each host becomes an impossible task
- Can often combine hosts into groups that share configuration Info. This can be done in many ways:
  - Keep a master copy of each configuration file in one place, and distribute it to hosts in a group whenever it changes
  - Eliminate text configuration files entirely, storing them instead in a central database server; each host obtains its configuration info. from this server.

Network Information Service (NIS)

- Released by SUN in the 1980s
- First “prime time” administrative database
- Originally called Sun Yellow Pages, but renamed for legal reasons
- NIS commands still begin with letters “yp” for this reason
- It's purpose is to serve administrative files across a network from a server to all or selected clients in a client-server system.

NIS: Overview 1

- Unit of sharing is the record <=> one line in a config. File (usually)
- A master server maintains authoritative copies of system files, in their original locations (mainly /etc)
- A server process makes the contents of the files available over the network
- Server and clients constitute an NIS “domain”
- Data files are pre-processed into databases by the GNU Database Manager, gdbm.
- Gdbm is a hashing library = a disk file format database which stores key/data-pairs in single files. The actual data of any record being stored is indexed by a unique key, which can be retrieved in less time than if it was stored in a text file.

NIS: Overview 2

- After editing files on the master server, you tell NIS to convert them to their hashed format using **make**
- Only one key can be associated with each entry, so a system file may have to be translated into several “maps”. For eg:
  - The /etc/passwd file is translated into two different maps called passwd.byname and passwd.byuid. One map is used to look up entries by username, and the other to look up entries by UID.
- NIS allows replication of network maps on slave servers (which the clients do not distinguish as different from the master)=>
  - Load balancing
  - Redundancy

NIS+

- NIS+ = son of NIS!!!
- Sun released NIS+ in the early 1990s
- Designed to correct the deficiencies of NIS
- From an implementation perspective, NIS and NIS+ are different
- But, from an administrative perspective, they appear similar.
- Although Linux can act as an NIS+ client, there is no NIS+ server for Linux

LDAP

- The letters LDAP denote Lightweight Directory Access Protocol, the current IETF standards-track system designed to fill the role as a directory service for distribution of both administrative and non-administrative data.
- There are five basic assumptions that characterise a database for inclusion into a directory service. These are:
  - Data objects are relatively small
  - The database will be widely replicated and cached
  - The information is attribute-based
  - Data are read often but written infrequently
  - Searching is a common operation

References

- “Linux Administration Handbook”, by Nemeth, Snyder & Hein (Prentice Hall, 2002)
- Hashing Library: http://www.whatis.com
- GDBM: http://www.linuxfromscratch.org