Networked Storage

Networked Storage

- For at least 20 years, some computers have accessed disks over the net
- Initially, that was because disks were too expensive to put on every small computer; now, it's for distributed access, large file storage, and *manageability*

Risks

- Confidentiality — spy on disk files
- Integrity — modify files
- Availability
- Note the special concern: unauthorized access can violate assumptions based on operating system file permissions

Types of Networked Storage

- Remote file system
- Remote disk
- For both, is the storage reasonably local to the client or accessed across the Internet?

Remote File System

- Access is to files, directories, etc.
- Must match OS file semantics
- Must implement — and honor — OS file permissions
- Consequence: must have some notion of OS userids
- Complexity: what happens if a single storage device is serving multiple computers with different userids?

Remote Disk

- Access is to disk blocks
- Simpler to implement; more portable
- Harder to share between computers — can two (or more) computers access the same "disk drive" at the same time?
- That can be done — and has been done, for at least 35 years — but it requires special OS-level support for shared drives. (No Unix or Windows system I know of has such support.)

Locking

- Locking mechanisms are crucial to either scheme
- For remote file systems, OS-type locking is needed, i.e., the Unix `flock()` system call
- For remote disks, the lock protocol is more subtle, and may involve OS access to file system metadata such as the free block list
- The lock mechanism itself can be a source of vulnerability

Major Networked Storage Systems

- NFS (Unix remote file system)
- CIFS (Windows remote file system)
- iSCSI (SCSI disks over IP)
- FCIP and iFCP (Fibre Channel disks over IP)

NFS

NFS

- Originally developed by Sun Microsystems
- Intention: support diskless workstations
- Now supported by all Unix variants; also available for Windows
- Large storage appliances implement it, too

NFS Technology

- Based on *Remote Procedure Calls* (RPC)
- (As we'll see in a few days, this is a source of a lot of security trouble in some environments)
- Original version ran over UDP only (again, a source of security trouble)
- Server was stateless (except for locking); all state kept on the client
- More recent versions use TCP

Traditional Flow of Control

- RPC call to find *mount* server
- RPC call to mount file system
  - Authentication happens at mount time
  - Credential returned is mediates all further access
- RPC operations to (kernel-resident) NFS server for I/O
- RPC operations to (user-level) lock daemons

Finding the Mount Daemon


```
$ rpcinfo -p clic.cs.columbia.edu
   program vers proto   port  service
    100024    1   udp  32768  status
    100024    1   tcp  32772  status
    100003    2   udp   2049  nfs
    100003    3   udp   2049  nfs
    100003    4   udp   2049  nfs
    100003    2   tcp   2049  nfs
    100003    3   tcp   2049  nfs
    100003    4   tcp   2049  nfs
    100005    1   udp    848  mountd
    100005    1   tcp    860  mountd
    100005    2   udp    848  mountd
    100005    2   tcp    860  mountd
    100005    3   udp    848  mountd
    100005    3   tcp    860  mountd
```

rpcinfo

- Many versions of many protocols available
- Access over TCP and UDP
- Services live on random port numbers
- The `rpcinfo` command queries the *portmapper* daemon to learn what's available on what port

The Mount Daemon

- Authenticates the client (but how?)
- Returns the *file handle* of the root i-node of the exported file system
- *File handles are at the heart of NFS operation and NFS security*

Querying the Mount Daemon


```
$ showmount -e mineral.cs.columbia.edu
Exports list on mineral.cs.columbia.edu:
/vol/vol2/admin1                   cs-nfs
/vol/vol1/faculty1/angelos         cs-nfs nyarlathot
/vol/vol2/research                 cs-nfs
/vol/vol2/reserve                  templar
/vol/vol2/micedata                 chihiro.cs.columb
/vol/vol2/proj_class               cs-nfs
/vol/vol2                          templar raphael
/vol/vol1                          raphael templar
...
```

File Handles

- File handles are random-seeming opaque strings
- Actually, generally composed of device number, i-node number, and a random value
- Every file and every directory has a file handle
- File operations present a file handle; directory lookups return a handle for the new file
- If you know the file handle for a single directory, you can read the entire disk. . .

File-Handle Guessing Attack

- Where does the random value come from?
- Initial value supplied when the file system is initialized
- Where do random numbers come from?
- If the PRNG seed is taken from too small a space, the "random" numbers are guessable
- This once happened; see `http://www.cert.org/advisories/CA-1991-21.html`
- For better advice on random number generation, see RFC 4086

A Digression on Randomness

- Many cryptographic and security systems require unpredictable random numbers
- Computers are not very good at true randomness — ideally, one should use a hardware source, such as a Geiger counter
- Most computers don't have Geiger counters. . .

Requirements for Using Pseudo-Random Number Generators

- Unpredictable initial seed
- Too large a search space to be brute-forced (at least 64 bits, preferably 128 bits)
- PRNG algorithm (and pattern) that does not permit guessing the next output from having seen the previous one
  - Non-cryptographic generators (i.e., `rand()` or `random()`) aren't adequate
  - $R_i = \mathrm{SHA1}(R_{i-1})$ is bad;
    $R_i = \mathrm{SHA1}(i || \mathrm{seed})$ is good
  - $R_i = \mathrm{HMAC}(\mathrm{seed}, i)$ is better

Random Seeds

- Low-order bits of disconnected microphone input (turn up the gain)
- Low-order bits of disk timing
- Interpacket or interkeystroke arrival times (*sometimes*)
- All of these sources require post-processing

Authentication and NFS

- Traditional NFS used address-based system authentication
- That is, the IP address was used to authenticate a system
- The remote system was trusted to enforce userids in I/O requests
- NFSv4 uses cryptographic authentication of individual users, via Kerberos-protected RPC calls — much safer

UID Mapping

- Originally, both systems needed identical UIDs
- Remember — this is a kernel-level activity, where UIDs are used, not user names
- One early exception: `root` was mapped to some other ID
- Today, general UID maps can be loaded

Risks of Traditional NFS

- Full trust in remote system
- Full trust in LAN — eavesdropping on a LAN is trivial
- Arguably reasonable 20 years ago — but far from acceptable today

File-Locking

- File-locking is done by a separate process
- Again, RPC is used
- A user-level process is used to permit easy disk I/O — lock information is written to disk, because the main path of an NFS server is stateless and won't remember locks after a reboot

NFSv4

- NFSv4 fixed many of the problems
- TCP is the primary transport, easing some of the firewall problems
- Locking is done in-band, again to simplify life with firewalls
- There's real authentication, on a per-user basis

Three Different Attack Vectors

- Fool authentication (impersonate host)
- Abuse the network medium
- Exploit implementation flaw

CIFS

Common Internet File System

- Developed by Microsoft
- Internet version of old NetBIOS protocol
- Primarily for Windows, though there's a popular open source server (Samba)
- Provides access to more than just files: printers, named pipes, and more
- Sometimes called the SMB — Server Message Block — protocol, which proves that I should have filed for a trademark years ago. . .

Finding Shared Resources

- On a LAN, servers broadcast their offerings
- There are remote name services to help find remote share offerings
- Partly integrated with basic Windows name service

Security Model

- Two types: share-level and server-level
- Share level: an entire disk is shared, read-only or read-write, to anyone who knows the name and password
- User-level permits fine-grained authentication of individual users and sharing of particular files or directories, rather than entire disk drives

Authentication

- Many forms of authentication possible
- Must adapt to many historical schemes in different verions of Windows
- Often, servers consult separate authentication servers for validation
- In any case, an opaque credential is returned after login; this is passed along with future requests

A Digression on Storing Passwords

- Systems generally do not store plaintext passwords; instead, they store $H(P)$, where H is some slow, non-invertible function
- But that requires that the client send the password in the clear to the server — probably acceptable (for modest security threats) on a phone line, but not over the Internet
- Using a challenge/response protocol requires that passwords (or at least the equivalent for purposes of this authentication) be stored in the clear, creating other risks

Never Store Plaintext Passwords

- For challenge/response, store $H(P)$ on the server; let the client calculate $H(P)$ from the entered password and use that as the key for the challenge/response
- Rationale: make it harder to steal the password for use on other systems
- A better variant:
  Server stores $S, H(P, S)$, where S is a random salt.
  Challenge: N, S
  Both sides calculate $F(N, H(P, S))$
- Why is that better?

Remote Disks

iSCSI and FCIP

- IP transport of existing command sets
- Originally for hardware devices — SCSI for small machines; Fibre Channel for mainframes
- Original protocols had *no* authentication — they were implemented over dedicated wires

Bandwidth Requirements

- Very high speed
- Intended target is full line speed over Gigabit Ethernet "with rapid migration to 10 GbE"
- Expected to require implementation of much of TCP and IP in hardware
- Direct data placement — copy data directly from wire into proper memory location, with no intermediate copies

What Kind of Crypto?

- TLS is processed after TCP, which makes it harder to do in hardware
- Obvious choice is IPsec
- 3DES-CBC is secure enough, and (marginally) fast enough in hardware — but it has to be rekeyed too often
- Other choice: AES in counter mode (why not CBC?)

Authentication

- IKE is used to provide authentication
- Manual keying can't be used, because of the need for rekeying
- iSCSI has its own authentication protocol — how do they combine?

IPsec Protection

- IKE is generally machine-level authentication
- IPsec provides per-packet protection — again, at machine granularity
- The iSCSI layer provides user-level authentication
- Crucial role for the OS: keep other users away from the iSCSI socket

Commonalities

- Note again: authentication is a weak spot
- We're trusting the OS even more, if the iSCSI remote disk is shared
- iSCSI got the packet protection model correct from the start — with, of course, the benefit of about 20 years more experience

Storage and the Internet

Off-site Disks and File Systems

- The same protocols can be used over the Internet
- Are there any new security issues?

The Obvious

- It's over the Internet, not local
- You need strong authentication and strong protection of the server host

User Population

- Who are the users?
- If it's a commercial service, with a heterogeneous user base, good authentication becomes crucial
- There's less of an issue if you're accessing your own normal file server, over an IPsec VPN

Encryption

- This is a good environment for encrypted storage
- Usually, file encryption is a bad idea — it provides little extra protection compared with the OS, but raises the risk of losing your data if you lose the key
- File encryption is useful when there's a physical threat
- You don't know who has access to a remote server