

# Kerberized Network File System for Clusters

## Evaluating the Impact of Increased Data Security

Chris Hoffman • Ian Burns • Paige Ashlynn      Mentor: David Kennel

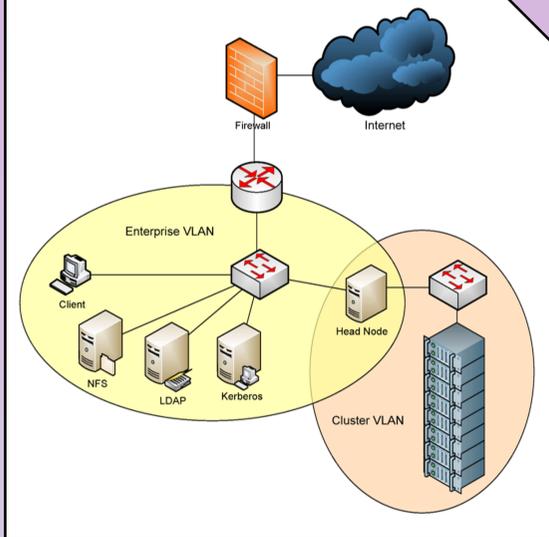
### Abstract

With constantly looming cyber-security threats, protecting production data has become an increasingly important issue. Though clustered environments traditionally have not required internal security, the landscape is changing rapidly. At LANL and other critical sites, the need to adopt proactive measures is apparent; however, implementation of security protocols should not compromise user experience or system performance. The Kerberos protocol provides a high level of security while minimizing overhead. A central Kerberos server provides authentication for a variety of services distributed over any number of connected (enterprise or cluster) networks.

It is important for any authorized person to be able to access their data from whatever computer they must use for their work. This could be a simple desktop workstation, or a large supercomputer. There needs to be a single, secure method of accomplishing this sharing for all environments. A Kerberized Network File System can be used to address this need for data mobility in a trusted manner. However, the performance impact that Kerberos will have on NFS in a clustered setting is still largely unknown. Factors such as level of security and types of encryption affect performance and usability greatly, potentially critically in a High Performance Computing environment.

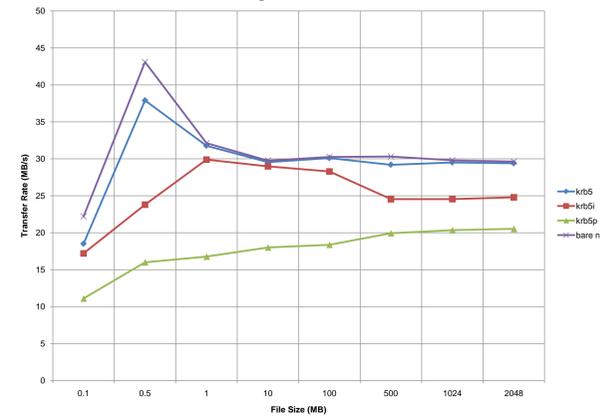
We will evaluate these impacts and make a general recommendation for suitable security levels and feasibility for possible deployments in current and future LANL systems.

### Test Set-Up

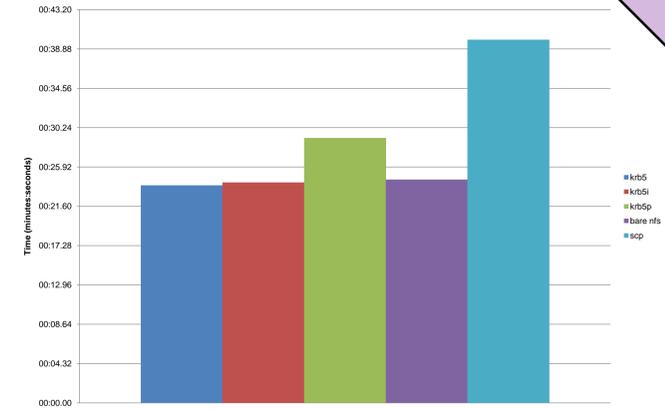


### Performance

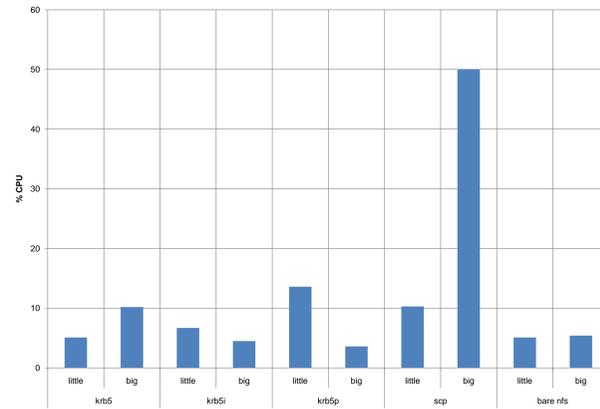
Average File Transfer Rate



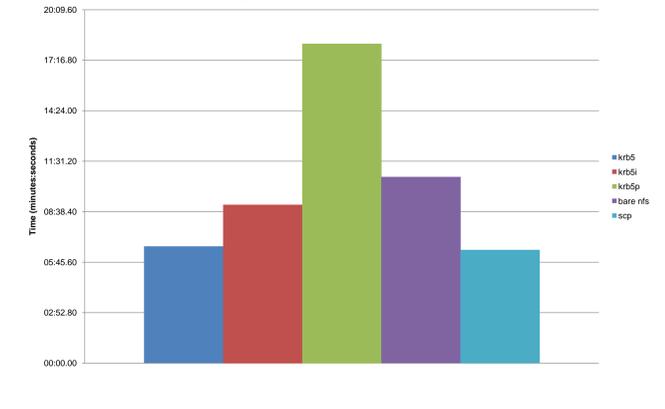
Average Time to Copy 10,000 5 kB Files



CPU Utilization by Protocol



Average Time to Copy 20 500 MB Files



### Kerberos 5 & NFS 4

#### Network File System

Initially developed by Sun Microsystems in the mid 1980s, the **Network File System (NFS)** protocol has evolved to become an open, industry-standard means for data exchange, employed on virtually all UNIX-like operating systems and available on most other systems as well. Through the use of remote procedure calls (RPC), the NFS protocol is able to deliver files and file systems across the network in much the same way that local files on local media are accessed. Historically, NFS presented several serious drawbacks including a poor security model; however, these concerns have been addressed in the most recent major revision of the protocol, version 4 (NFSv4). Operating simultaneously at multiple layers of the Internet protocol stack, NFSv4 is a stateful protocol which expects an accompanying security mechanism. Several alternative security measures are available.

#### Kerberos

Initially developed by the Massachusetts Institute of Technology in the early 1980s, **Kerberos (KRB)** has been widely adopted as a robust, reliable, and adaptable authentication mechanism. An open standard since its inception, the protocol has been revised numerous times to keep up with the changing security world, and has become an integral element in virtually every major operating system. Through the use of symmetric key cryptography and a trusted third-party security server, Kerberos allows both clients and servers to mutually authenticate by proving their identities. The current revision of Kerberos (**KRB5**) includes additional functionality such as the ability to verify the integrity of data transmitted over the network. While the user need authenticate only once, each subsequent transaction is silently validated. Kerberos is a complex protocol that relies on a sophisticated security model. As such, implementing it on an already-complex system like a supercomputer is not trivial. Nevertheless, the three security levels available in KRB5 allow policy makers to weigh features against requirements.

#### Levels of Security

- **Plain Kerberos (KRB5)** — The most basic level of security provided by Kerberos allows for clients and servers to prove to one another machine, application, and user identity in a manner that prevents a variety of network misuses with minimal overhead. This level of security prevents most forms of mounted-NFS abuse.
- **Kerberos Integrity (KRB5i)** — Has all the features of the basic KRB5, but in addition employs a checksum technique to verify the integrity of the RPC data transmitted. This level of security prevents transmission alteration.
- **Kerberos Privacy (KRB5p)** — Has all the features of KRB5i, but additionally employs encryption to protect the RPC data. This level of security prevents intermediaries from reading the RPC packets.

### Installation

NFS requires at least one machine to act as a fileserver. Similarly, Kerberos requires at least one machine responsible for performing authentications and providing session or service tickets and keys. For optimal performance each machine should be a dedicated server. In addition, both protocols allow for scaling via redundant servers, and the various stages of the Kerberos protocol suite can be spread across multiple machines. In our environment, we simulated an enterprise network and a High Performance Computing (HPC) cluster all within a single physical cluster by dividing cluster nodes into multiple virtual networks. On our simulated enterprise network, one machine was dedicated to all Kerberos authentication activities, a second to NFS file-serving, and a third to providing other network services (NTP, DNS, etc). The remainder became the HPC cluster, with a Head node providing a gateway to the outside network and several Compute nodes operating behind the Head. Each Compute node acts as a Kerberos principal and an NFS client. Accessing the enterprise network via the Head (which performs Network Address Translation), the clients are able to authenticate and retrieve data in each of the three security levels. To deploy a similar setup on a production system would necessitate several infrastructure and network-topology choices. Ideally, the NFS/KRB servers would probably reside outside the cluster on dedicated machines; however, the optimum solution is application-dependant.

### Future Work

An outstanding problem for adoption of these technologies in a cluster environment is integration of Kerberized NFS with a **job scheduler**. Currently Torque (part of the Portable Batch System [PBS] project) partly supports Kerberized NFS, but support is incomplete. In absence of ready solutions, the simplest route may involve rewriting some of the code. Another area for future research is integration with the Lightweight Directory Access Protocol (LDAP). LDAP has been successfully integrated with Kerberized NFS in the past, but not to our knowledge in a cluster environment. Finally, analysis of the impact of server- and client-side caching is desirable. If and how caching impacts Kerberized NFS depends upon a number of operating system and hardware factors that could greatly alter the test results.

### Test Procedure

The performance of Kerberized NFS was measured using three tests over a gigabit Ethernet network. For comparison purposes, we also measured the performance of the standard UNIX/Linux remote file transfer command line utility Secure Copy (SCP). First, the average transfer rate for a range of file sizes from 100 kB to 2 GB was measured. The next test measured the average time to copy 10,000 5kB files using each form of NFS and SCP. The final test was similar but used 20 500 MB files. The CPU utilization was also measured during these final 2 tests. Each test was completed 10 times, and the average result of each completion was taken. All tests were performed from the same NFS client with no other activity on any of the machines involved. The average transfer rate test wrote files consisting entirely of zeros to a mounted NFS directory, while the other tests wrote files consisting of random values to a temporary folder before copying them to an NFS directory.

### Conclusions

At its simplest, Kerberized NFSv4 introduces minimal overhead to and can even accelerate an NFS system while providing greatly improved security. **Simple Kerberos 5** authentication provides equal performance for a variety of use cases and little to no difference from a bare NFS system. **Integrity-checking KRB5i** security changes little when handling a vast number of small files, but decays in performance as file size increases. **Encrypted KRB5p** also scales well for small files, introducing expected though minimal overhead. For large files, however, it decays more significantly in performance than KRB5i. CPU load remained below 10% for all forms of NFS with the exception of small file transfers using the encrypted KRB5p. So, while the actual transfer performance of KRB5p with small files scales well, it does use a significant amount of additional CPU time. For file transfers, SCP provides much improved transfer rates for large files at the expense of massive CPU usage. On the other hand, NFS provides better transfer rates and lower CPU usage for small files, even with encrypted Kerberized NFS. Inside a cluster, Kerberized NFS behaves normally with the default addressless tickets. If a central node image is used, as is the case with systems like Perceus, a keytab file using wildcards can be inserted into the image to allow this normal operation. A significant limitation for a cluster, however, is that the user must authenticate — a problem for scheduled jobs that run longer than the ticket lifetime.