Issue 9 vol 3 ISSN: 2321-8134

IJFEAT

INTERNATIONAL JOURNAL FOR ENGINEERING APPLICATIONS AND TECHNOLOGY

KEY EXCHANGE PROTOCOL TO REDUCE THE WORKLOAD ON METADATA SERVER : A REVIEW

Manjeeri D Nawghare[1], Monika S Bora[2], Krutika A Mandape[3], Prof. Aditya P Bakshi[4]

[1]*Student, Department of CSE, JDIET, Maharashtra, India,* ***manjeerinawghare@gmail.com***
[2]*Student, Department of CSE, JDIET, Maharashtra, India,****monikabora98277 @gmail.com***
[3]*Student,Department of EXTC, JDIET, Maharashtra, India,****krutikaashokmandape@gmail.com***
[4]*Assistant Professor, Department of CSE, JDIET, Maharashtra ,India,* ***bakshi.aditya.ab@gmail.com***

Abstract

There is a problem of establishing key in order to secure many-to-many communications as there is a increase in the use of large-scale distributed file systems that supports parallel access to multiple storage devices. The focus is on the current Internet standard for such file systems, that is, parallel Network File System , which makes use of Kerberos to establish parallel session keys between clients and storage devices. The review of the existing Kerberos-based protocol shows that a metadata server facilitating key exchange between the clients and the storage devices has heavy workload that restricts the scalability of the protocol. Here, a authenticated key exchange protocol is proposed capable of reducing the workload of metadata server.

Index Terms: *Parallel Sessions ,Authenticated Key Exchange, Network File Systems, Metadata Server.*

---------- *** ----------

1. INTRODUCTION

With the increase in the use of internet ,data security is the major concern. The data of internet clients is being stored in the storage devices and in order to access that data of multiple clients parallel applications are used. This is usually used in technologies that focuses on high performance and reliable access to large datasets. That is, higher I/O bandwidth is achieved through concurrent access to multiple storage devices within large data storages; while data loss is protected through data mirroring . These are usually required for advanced scientific or data-intensive applications where hundreds or thousands of file system clients share data and generate very high aggregate I/O load on the file system supporting terabyte-scale storage capacities.

Here, the problem considered is of securing many to-many communications in large-scale network file systems that support parallel access to multiple storage devices. Here, we consider a communication model where there are a large number of clients (potentially hundreds or thousands) accessing multiple remote and distributed storage devices in parallel. The focus is on how to exchange key materials and establish parallel secure sessions between the clients and the storage devices in the parallel network file systems that is the current Internet standard in an efficient and scalable manner.

The primary goal in this work is to design efficient and secure authenticated key exchange protocol that meet specific requirements of parallel network file systems. Particularly, we attempt to meet the following desirable property which either have not been satisfactorily achieved by the current Kerberos-based solution: The metadata server facilitates access requests from a client to multiple storage devices should have little workload as possible such that the server is capable of supporting a very large number of clients. The main result here is a new secure authenticated key exchange protocol. The protocol is proposed to achieve the scalability property. The proposed protocol can reduce the workload of the metadata server compared to the current Kerberos-based protocol, while achieving the desired security property and keeping the computational overhead at the clients and the storage devices at a low level. An appropriate security model is defined and shown that the protocol is secure in the model.

2. LITERATURE REVIEW

[2], designed Maat, a security protocol to provide strong, scalable security to these systems. Maat encompasses a set of protocols that facilitate (i) authenticated key establishment between clients and storage devices, (ii) capability issuance and renewal, and (iii) delegation between two clients. The authenticated key establishment protocol allows a client to establish and re-use a shared (session) key with a storage device. However, Maat do not come with rigorous security analysis.*[3]*, focussed on scalable security. This proposal assumed that a metadata server shares a group secret key with each distributed storage device. The group

key is used to produce capabilities in the form of message authentication codes. However, compromise of the metadata server or any storage device allows the adversary to impersonate the server to any other entities in the file system. This issue can be alleviated by requiring that each storage device shares a different secret key with the metadata server. Nevertheless, such an approach restricts a capability to authorizing I/O on only a single device, rather than larger groups of blocks or objects which may reside on multiple storage devices. *[4]*, proposed SFS based on public key cryptographic techniques, that was designed to enable inter-operability of different key management schemes. Each user of these systems is assumed to possess a certified public/private key pair. However, these systems were not designed specifically with scalability and parallel access in mind.

3. INTERNET STANDARD--NFS

Network File System (NFS) is currently the basic file system
standard supported by the Internet Engineering Task Force (IETF). The NFS protocol is a distributed file system protocol originally developed by Sun Microsystems that allows a user on a client computer to access files over networks in a manner similar to how local storage is accessed .It is designed to work different machines, operating systems, network architectures, and transport protocols. NFS is used in environments where performance is a major factor. The most recent version of NFS, parallel network file system that allows direct, concurrent client access to multiple storage devices to improve performance and scalability.

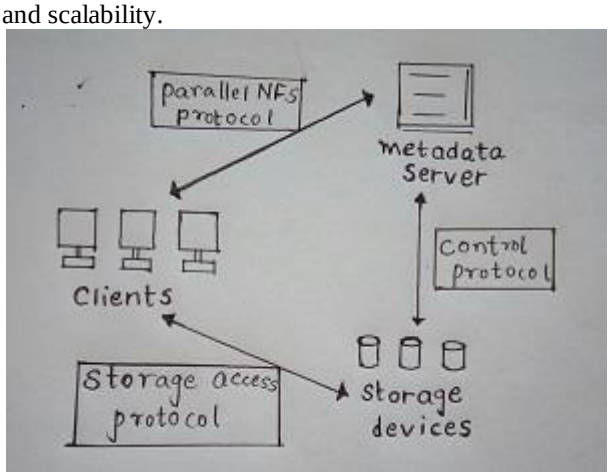


Fig-1:Conceptual view of parallel network file systems

When file data for a single NFS server is stored on multiple storage devices (by comparison to the server's throughput capability), the result can be significantly better file access performance. Parallel network file systems processes regular files data by stripping and storing across storage devices or servers. Data striping occurs in two ways: on a file-by-file basis for smaller files and, for sufficient large files, on a block-by-block basis. In parallel NFS, a read or write of data managed ,is a direct operation between a client node and the storage system itself.

More specifically, parallel NFS comprises a collection of three protocols (fig-1)(i) the parallel NFS protocol that transfers file metadata, also known as a layout, between the metadata server and a client node; (ii) the storage access protocol that specifies how a client can access data from the associated storage devices according to the corresponding metadata; and (iii) the control protocol that synchronizes state between the metadata server and the storage devices.

3.1 Security Concern

Earlier versions of NFS focused on simplicity and efficiency, and were designed to work well on intranets and local networks. The later versions i.e parallel NFS aim to improve access and performance within the Internet environment. However, security has become a major concern. Among many other security issues, user and server authentication within an open and distributed environment are a complicated matter. Key management can be fatigue and expensive, but an important thing to ensure security of the system. Moreover, data privacy may be critical in high performance and parallel applications, for example, those associated with biomedical information sharing , financial data processing & analysis. Hence, distributed storage devices possess greater risks to various security threats, such as illegal modification or stealing of data that resides on the storage devices, as well as change of data in transit between different nodes within the system. NFS, therefore, has been mandating that implementations support end-to-end authentication, where a client mutually authenticates to an NFS server. Moreover, major concern should be given to the integrity and privacy of NFS requests and responses.

3.2. Kerberos & LIPKEY

In parallel NFS, the Kerberos and the Low Infrastructure Public Key (LIPKEY) mechanisms are used. Kerberos is used particularly for user authentication, while LIPKEY particularly used for server authentication in the Internet environment. Kerberos, a widely deployed network authentication protocol supported by all major operating systems, allows nodes communicating over a non secure network to perform mutual authentication. It works in a client-server model, in which each domain is governed by a Key Distribution Center (KDC), acting as a server that authenticates and provides ticket-granting services to its users (through their respective clients) within the domain. Each user shares a unique password with its KDC and is authenticated through a password-derived symmetric key that is known only between the user and the KDC. However, one security weakness of this authentication method is that when a weak password is used to derive a key that encrypts a protocol message transmitted between the client and the KDC ,there is threat of an off-line password guessing attack. Hence, LIPKEY authenticates the client with a password and the metadata server using its public key certificate, and there by establishes a secure channel between the client and the metadata server. Through LIPKEY, a client with no public key certificate accessing a server with a public key certificate, for this the client in NFS :

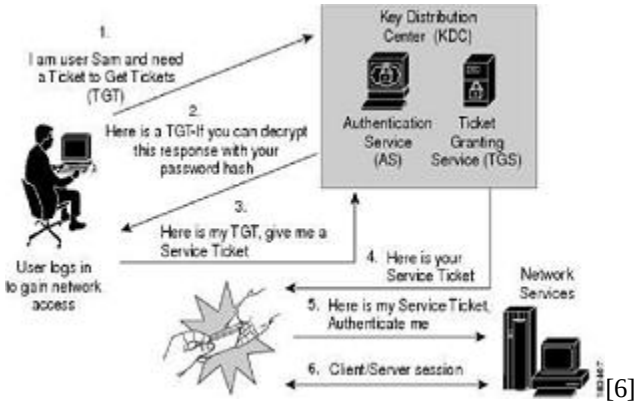
- obtains the metadata server's certificate
- verifies that it was signed by a trusted Certification Authority (CA)
- generates a random session symmetric key
- encrypts the session key with the metadata server's public key and
- sends the encrypted session key to the server.

At this point, the client and the authenticated metadata server have set up a secure channel and now the client can

provide a user name and a password to the server for user authentication.

4.KERBEROS-BASED PNFS PROTOCOL

The key establishment protocol recommended for parallel NFS between a client and n storage devices , through a metadata server is considered. Here the efficiency of the parallel NFS protocol is compared with the new proposed protocol. During the setup phase, assume that metadata server establishes a shared secret key with each the storage device. Here, a key is derived from client's password, that is also known by metadata server; while there exists the role of a ticket-granting server (it is part of metadata server). Also, prior to executing the protocol in , the client and metadata have already setup a secure channel through LIPKEY.



[6]

Fig-2: Kerberos-Based parallel NFS protocol

An algorithm for the existing kerberos based parallel NFS protocol:

Step 1*:* Client requests the KDC(Key Distribution Center) asking TGT to get the service tickets.

Step 2: In response, the metadata server sends the TGT to the client and asks the client to decrypt it using user's password.

Step 3: Now the client passes on the TGT to Ticket Granting Server(TGS) after decrypting and asks for service tickets.

Step 4: As the ticket granting server receives the request from the client, it sends the service tickets, the layout, session keys in response.

Step 5*:* Client passes the service tickets to the respective storage devices asking to authenticate itself, and thus creating a session between the client and storage devi*ces.*

Step 6: Storage device convince the client that, storage device is using the same session key that client uses. This is a key confirmation step.

Once client has been authenticated by metadata server and granted access to n storage devices, it receives a set of service tickets, session keys, and layouts from ticket granting server of the protocol. Clearly, the client is able to uniquely extract each session key from authentication token. Since the session keys are generated by metadata server and transported to storage device through client, no interaction is required between client and storage devices (in terms of key exchange) in order to agree on a session key. This keeps the communication between the client and each storage device to minimum where key exchange is required. The computational overhead for the client and each storage device is very low since the protocol is mainly based on symmetric key encryption technique. The message in step (6) serves as key confirmation, that is to convince client that the storage device is in possession of the same session key that client uses.

5.PROPOSED WORK

The proposed protocol can be said as a modified version of Kerberos that allows the client to generate its own session keys. That is, the key materials that are used to derive a session key is already being computed by the client for each validity period *v* ,which can be hours, days, months ago and forwarded to the storage devices in the form of an authentication token at time *t* (t is the session time with each storage device). Along with Kerberos, symmetric key encryption is used to protect the confidentiality of secret information used in the protocol.

Specification of proposed protocol:

Firstly, Consider that client and metadata server is authenticated and protected through a secure channel associated with key established using the LIPKEY mechaism. It works in two phases, where the Phase I constitutes of pre-computing the key materials and any request from client to access storage devices is considered part of Phase II of the protocol until *v* expires.

Phase I:For each validity period v,

Step 1: Client first pre-computes a set of key materials before it accesses any of storage devices. These key materials are then send to meta data server asking for the authentication token.

***Step* 2:** Metadata server then issues a authentication token for each key material for the associated storage device.

Phase II: For each access request at time t,

Step 1: Client submits an access request to metadata server, the request contains all the identities of storage devices that Client wishes to access.

Step 2*:* Metadata server issues the layout for each storage devices.

Step 3: Client then forwards the respective layouts, authentication tokens (from Phase I), and encrypted messages consisting of identities of storage devices, along with time slot t for each storage device, to all *n* storage devices.

Step 4: Upon receiving an I/O request for a file object from Client, each Storage device does following checking:

- checks whether the layouts received is correct or not.

- decrypts the authentication token and computes the session keys
- decrypts the encrypted message, and matches the identities send by the client.

When all the check passes, storage device replies the client with key confirmation message using the first session key generated. At the end of protocol, another key generated is set to be the session key for securing communication between client and storage device.

6.ADVANTAGES

- **Key Storage**

Here the key storage requirements for Kerberos based parallel NFS and the described protocol is roughly similar from the client's perspective. For each access request, the client needs to store N or $N + 1$ key materials for n storage devices in their internal states. In contrast to Kerberos-parallel NFS, our proposed protocol do not require to maintain any client key information.

- **Workload on metadata server**

After computing using the proposed protocol , it seems that the workload on the metadata server is decreased as compared to existing kerberos-based parallel NFS protocol.

7. CONCLUSION

The keys are vulnerable to various attacks in a parallel NFS and in order to provide security as well as smooth communication between the client and storage device , a authenticated key exchange protocol is being proposed that has the advantage over the existing Kerberos-based parallel NFS protocol. The proposed protocol can reduce the workload on the metadata server.

REFERENCES

[1] Hoon Wei Lim, Guomin Yang," Authenticated Key Exchange Protocols for Parallel Network File Systems", IEEE Transactions on Parallel and Distributed Systems. IEEE ,2013.

[2] A.W. Leung, E.L. Miller, and S. Jones."Scalable security for petascale parallel file systems". ACM Press, Nov 2007.

[3] M. Factor, D. Nagle, D. Naor, E. Riedel, and J. Satran." The OSD security protocol". IEEE Computer Society, Dec 2005.

[4] David Mazieres, M. Kaminsky, M.F. Kaashoek, and E. Witchel. "Separating key management from file system security", ACM Press, Dec 1999.

[5] Web Available at: Parallel virtual file systems (PVFS) version 2. http://www.pvfs.org.

[6] Web available at: https://www.google.co.in/url?sa=i&rct=j&q=&esrc=s&source=images&cd=&ved=0ahUKEwiCqbHX6bzWAhUFQLwKHeWmAKUQjRwIBw&url=https%3A%2F%2Fdecoder.cloud%2F2017%2F02%2F12%2Fthe-golden solution%2F&psig=AFQjCNGdxGgq4TCnwXblS9t_rmKKr__R-Q&ust=1506308250987412