1 Introduction to KERBEROS/OpenAFS

— last modified 2017/09/26 20:30:59 —

Contents

1 Introduction to KERBEROS/OpenAFS 1
1.1 Abstract . . . 1
1.2 The Building Blocks of LIONS - Kerberos, OpenAFS, LDAP . . . 1
1.3 Kerberos Tickets and AFS Tokens . . . 2
1.3.1 Viewing Ticket Information . . . 2
1.3.2 What To Do When Tickets Expire . . . 3
1.4 ACL Permissions . . . 3
1.4.1 Types of ACL Permissions . . . 4
1.5 Useful Commands . . . 5

1.1 Abstract

If you are new to the Old Dominion University LIONS environment and you would like to know more about the technology used in the building of LIONS, this is the document for you!

LIONS was created with redundancy and fault tolerance as it's primary objective. It features fault tolerant servers, a fully mirrored Storage Area Network (SAN), fault tolerant routers for the SAN, a security master server with replica servers, a file system master server (with replica servers), an Enterprise Level backup/restore solution, and a uniform environment for all academic UNIX users.

LIONS also provides centralized services such as FTP, UNIX software license management, printing, Enterprise Level web and documentation servers, and features a web/e-mail based help request system.

To assist us with running such a large environment requires a world-class Enterprise Security and File systems.

1.2 The Building Blocks of LIONS - Kerberos, OpenAFS, LDAP

The underlying structure which makes up the LIONS architecture are:

- Kerberos is a network authentication protocol. It is designed to provide strong authentication for client/server applications by using secret-key cryptography.

LIONS uses the free implementation of this protocol available from the Massachusetts Institute of Technology.

- OpenAFS is a distributed filesystem that enables co-operating hosts (clients and servers) to efficiently share filesystem resources across both local area and wide area networks. AFS is a distributed filesystem product, pioneered at Carnegie Mellon University and supported and developed as a product by Transarc Corporation (now IBM Pittsburgh Labs). It offers a client-server architecture for file sharing, providing location independence, scalability and transparent migration capabilities for data. IBM branched the source of the AFS product, and made a copy of the source available for community development and maintenance. They called the release OpenAFS.

- LDAP is used as the naming services "glue" which binds the Kerberos and OpenAFS information together.

1.3 Kerberos Tickets and AFS Tokens

Kerberos credentials or *"tickets"* are one of the essential concepts all users of **LIONS** need to understand. When users successfully log in to a UNIX workstation they are authenticated by the Kerberos KDC (Key Distribution Center) and given a ticket. The first ticket you obtain is a *ticket-granting ticket*, which permits you to obtain additional tickets. This TGT is presented to the OpenAFS server to obtain a *"token"*. Both the ticket and the token grant access to all of a user's autorized resources (ex., departmental software, home directories, web content etc.). Tickets are only valid for **one day**, or **24 hours**. Once a user's tickets and tokens expire they will have to be renewed for continued access to any LIONS resources.

1.3.1 Viewing Ticket Information

To view ticket information, type the **"klist"** (no quotes) command at the command prompt. To view token information, type **"tokens"** command. A listing similar to the following will appear. (In the example, the $ character represents the UNIX command prompt).

```
$ klist
Ticket cache: FILE:/tmp/krb5cc_1120_m11027
Default principal: jtest@AUTH.ODU.EDU

Valid starting     Expires            Service principal
02/01/17 08:18:21  02/01/17 18:18:21  krbtgt/AUTH.ODU.EDU@AUTH.ODU.EDU
02/01/17 08:18:23  02/01/17 18:18:21  afs/lions.odu.edu@AUTH.ODU.EDU
$ tokens
```

```
Tokens held by the Cache Manager:

User's (AFS ID 1120) tokens for afs@lions.odu.edu [Expires Feb  1 18:18]
   --End of list--
```

The ticket cache is the location of your ticket file. In the above example, this file is named `FILE:/tmp/krb5cc_1120_m11027`. The default principal is your Kerberos *principal*.

The "valid starting" and "expires" fields describe the period of time during which the ticket is valid. The *service principal* describes each ticket. The ticket-granting ticket has the primary krbtgt, and the instance is the realm name.

1.3.2 What To Do When Tickets Expire

When tickets expire there are 2 simple remedies. The first is to log out and log back in. The second is to use the combination of **"kinit"** and **"aklog"**. The second method is useful if you stay logged into a **LIONS** system for long periods of time, or if your credentials expire as you're using your account. Here's how it works....

In a local shell type `kinit` followed by `aklog`:

```
$ kinit
password: (type in your password)
$ aklog
$
```

kinit will create a new TGT after authentication, aklog then uses this ticket to generate an AFS authentication token from one of our AFS database servers. Once you have the new Kerberos TGT and AFS token you're good to go, without having logged out and back in.

Your credentials are now refreshed for another 24 hours. To check, type `klist` and view the date/time stamp of ***Expires*** field.

To avoid having credentials expire when least expected, enter the `klist` command and check the Expires field .

1.4 ACL Permissions

OpenAFS uses ACLs (Access Control Lists or 'ackuls') to control access to files and directories. Being knowledgeable about regular UNIX mode bits (r,w,x) and their application will assist you in understanding and appreciating the flexibility of AFS ACLs.

As the name implies, ACLs control access to files and directories. Unlike standard UNIX permissions, AFS ACLs uses seven permissions to provide greater flexibility for access control. With ACLs you can restrict access on an files or directories to a single user, a single group, many individual users, or many groups.

1.4.1 Types of ACL Permissions

There are 3 types of permissions in AFS. Directory, File and Auxiliary. We will be discussing only directory and file permissions in this document.

Lets start with a brief explanation of ACL Permissions:

Four Directory Permissions

`l = lookup` (functions as something of a gate keeper for access to the directory and its files)

`i = insert` (enables a user to add new files to the directory)

`d = delete` (enables a user to remove files and subdirectories from the directory or move them into other directories)

`a = administer` (enables a user to change the directory's ACL)

Three File Permissions

`r = read` (enables a user to read the contents of files in the directory)

`w = write` (enables a user to modify the contents of files in the directory)

`k = lock` (enables a user to run programs that issue system calls to lock files in the directory)

Operation	Required Permissions
Change to a directory	`rl` on the directory itself `rl` on all directories that lead to the directory
List contents of a directory	`l` on the directory itself `rl` on all directories that lead to the directory
List information about objects in a directory	`rl` on the directory itself `rl` on all directories that lead to the directory
Create an object	`rli` on the directory in which the object is to be created `rl` on all directories that lead to the directory
Delete an object	`rld` on the directory in which the object is to be deleted `rl` on all directories that lead to the directory
Rename an object	`rldi` on the object's current directory `rli` on the object's new directory `rli` on the object

Read or read lock a file	`rlk` on the file itself `rl` on all directories that lead to the file
Write or write lock a file	`rlik` on the file itself `rl` on all directories that lead to the file
Execute a binary file	`rl` on the file itself `rl` on all directories that lead to the file
Execute a shell script	`rl` on the file itself `rl` on all directories that lead to the file
List ACLs on an object	`rl` on all directories that lead to the object
Change object's ACLs or modify mode bits	`a` on the object `rl` on all directories that lead to the object

The official **OpenAFS User Guide** include a section on ACLs which contain some **examples of their use**.

1.5 Useful Commands

When you obtain a **LIONS** account you will receive an account with a directory in AFS space. In most cases, these directories function just like any other UNIX file system. You use standard UNIX commands to create subdirectories and to move, copy and delete files. You will, however, need to familiarize yourself with a few special Kerberos and OpenAFS commands:

- fs listacl

  The **fs listacl** or **fs la** command displays the access control list (ACL) associated with each specified file, directory, or symbolic link.

- fs setacl

  The**fs setacl** or **fs sa** command adds the access control list (ACL) entries specified with the -acl argument to the ACL of each directory named by the -dir argument.

- fs listquota

  The **fs listacl** or **fs lq** command displays information about the volume containing each specified directory or file (its name, quota, and amount of disk space used), along with an indicator of the percentage of space used on the host partition.

- kpasswd

  The **kpasswd** command changes the password recorded in an Kerberos Database entry. BE CAREFUL WHEN USING THIS COMMAND! MIDAS controls your

password normally, so if you change it with kpasswd and you subsequently update your MIDAS password, MIDAS will overwrite your changes.

- k5start

  The command **k5start** normally acts like the Kerberos kinit command, but it has the options to renews tickets and tokens for the **LIONS** environment.

- mkpublic_html

  A locally written command, **mkpublic_html** creates a web directory in your home directory. This does not automatically give you permission to serve web pages, please see the the **Web Services** section of the UNIX/LIONS pages for more information.