Mingle

A Secure Distributed Search System

Yinglian Xie

Table of Contents

1 Mingle Introduction

Mingle is a secure distributed search system for personal computing. Mingle precomputes an inverted index of local files on each participating host. A query can be processed by the local host, or routed through the participating hosts using peer-to-peer communication to locate all of the desired data.

Here are some general features of the Mingle system:

Indexing

Mingle precomputes an inverted index table of local files on each participating host for fast search. Mingle will then update the index table regularly.

Searching

Mingle supports keyword search at one or more participating Mingle hosts. A query can be processed by the local host, or routed through remote hosts to locate all of the desired data.

Advanced search options include:

1. Search in file names and file contents
2. Support logic operations `"OR"`, `"AND"` in a query
3. Support wildcard matching (e.g., use `"a*"` as a keyword).

Security

Mingle allows file owners to grant search permissions to other Mingle users on a file-by-file basis. Mingle servers authenticate each user request to protect data from being accessed by unauthorized users.

2 Beginning to use Mingle

A Mingle cluster consists of one or more Mingle hosts. On each host, there is a Mingle server running as a daemon. A user may issue a request from any host to any of the servers by launching a Mingle client program. In many cases, a user may wish to issue request to all hosts in a Mingle cluster without specifying host identities. To enable this, you will need to configure a master server for each Mingle cluster (see Section 2.1 [Master server configuration], page 2).

A Mingle server can work in single user mode or multiple user mode. In single user mode, the server will not be used by other local users that have accounts on the same computer. While in multiple user mode, only one copy of server will be running on a host and the server is shared by all the local users (see Section 2.2 [User mode set up], page 3).

Before using Mingle, each user will need to sign on a Mingle ID that identifies the user to Mingle servers in a uniform way. Each Mingle ID is globally unique within a Mingle cluster. A Mingle ID is assigned to a user via a registration process called Mingle single sign-on (see Section 2.3 [Mingle single sign-on], page 3).

If you wish to index files on a host or specify access control decisions, a Mingle ID is usually not enough. The corresponding Mingle server has to know which local user you are to perform such operations. Therefore, you will need to present both your local identity and your Mingle ID to the corresponding servers by a process called "Mingle init", when your local user ID is associated with your Mingle ID (see Section 2.4 [Mingle init], page 3).

To protect data from being searched by unauthorized users, a Mingle server will authenticate each user request. The local request authentication is performed by the operating system when the user logs in to the host. The remote request authentication is based on conventional digital signature using the RSA algorithms. Each Mingle user is associated with a pair of RSA keys (a public RSA key and a private RSA key) that are generated by the Master server at single sign-on (see Section 2.5 [RSA key distribution], page 4).

2.1 Master server configuration

Master server is a normal Mingle server that maintains the list of host names inside the cluster. Upon reception of a user request that needs to be routed through the cluster, a local Mingle server first fetches the host list from the master server, and then forwards the request to each remote host in the list.

You can configure the master server by using the "IsMaster" parameter in the Mingle configure file (A default Mingle configure file is contained in the Mingle server package). For example,

```
IsMaster = 1
```

will set the local Mingle server to be the master server.

For a master server, Use the parameter "ClusterHosts" to specify the list of host names (and port numbers) inside the cluster. For example,

```
ClusterHost = "area51", "grassyknoll:3000"
```

Note that you must specify all the hosts in one line, separating two adjacent hosts by a comma followed by one or more blank spaces.

For a normal server, use the parameter "MasterHost" and "MasterPort" to specify the master server from which a list of host names can be fetched. For example,

```
MasterHost = area51
MasterPort = 3000
```

2.2 User mode set up

A Mingle server can run in single user mode or multiple user mode. When a server runs in single user mode, only the local user who starts the daemon can submit local requests to the server. When running in multiple user mode, the server will be shared by multiple local users. Use the parameter "UserMode" in the configure file to set up user mode.

For example,

```
UserMode = 1
```

will allow multiple local users to submit local requests to the server. In such case, you usually need to run the server in root mode. Note that whether in single user mode or multiple user mode, the server will accept remote requests from any user.

2.3 Mingle single sign-on

In a Mingle cluster, each user is identified by a global, unique Mingle ID. The Mingle ID has to be signed on explicitly once before using Mingle. To do this, you will have to log in to one of the Mingle host, and submit a sign-on request:

```
mingle signon mingle-id
```

In the above command, '`mingle-id`' is a text string that you select. You will then be prompted to input a Mingle password to the client program, which conveys both the Mingle ID and the password to the local Mingle server. The local Mingle server sends this pair to the master server for this Mingle cluster. If the Mingle ID has not previously been registered, the master server generates an RSA key pair for you, which will later be retrieved by the mingle servers in the same cluster to authenticate your requests whenever necessary (see Section 2.5 [RSA key distribution], page 4).

After signing on, you have established your identity represented as your Mingle ID in the cluster. Use the command '`mingle passwd`' if you wish to change your Mingle password:

```
mingle passwd mingle-id
```

2.4 Mingle init

After Mingle sign-on, you are uniquely identified as your Mingle ID in the corresponding cluster. However, if you wish to index files on a host or specify access control decisions, you will need to let the corresponding Mingle server know which local user you are via the '`mingle init`' command. This Mingle init process also presents your Mingle ID to the server so that the server can sign the request on your behalf before forwarding the request to the remote servers. Note that the '`mingle init`' request has to be submitted locally, meaning that you will need to log in to the desired Mingle host and submit a request there.

The syntax of the '`mingle init`' request is the following:

```
mingle init mingle-id
```

where '`mingle-id`' is the Mingle ID that you have signed on. Next, the Mingle server interactively prompts you for your Mingle password for authentication, creates an entry for you, and associate your local user name and your Mingle ID together so that you no longer need to enter your Mingle ID for subsequent requests. The local Mingle server will also fetch your RSA private signing key from the master server using your password and cache it (see Section 2.5 [RSA key distribution], page 4).

After the cached private key expires, this Mingle init process can be repeated to obtain a refresh copy of key for caching. Therefore, the private key cache expiration time determines the frequency of executing such "init" command. Modify the parameter "PrivateTTL" to adjust the expiration time of caching private keys.

After Mingle init, use the command '`mingle display`' to check your current status. For example, if your local user name is '`bovik`' and your Mingle ID is '`mbovik`', then

```
mingle display
```

will return

```
Mingle ID     = mbovik
Local user ID = bovik
```

2.5 RSA key distribution

By default, mingle servers authenticate each user request in order to protect data from being accessed by unauthorized people. You can enabled or disabled the permission checking by modifying the parameter "SecurityMode" in the Mingle configure file. For example,

```
SecurityMode = 0
```

disables the permission checking for all user requests.

When permission checking is enabled, a Mingle server will authenticate remote requests using the RSA digital signature scheme.

Each Mingle user has a pair of RSA signing keys (generated by the master server at Mingle single sign-on (see Section 2.3 [Mingle single sign-on], page 3)). Once a local Mingle server obtains your private RSA key at Mingle init (see Section 2.3 [Mingle single sign-on], page 3), the server can sign your request using your private key, and send the signed request to the relevant remote Mingle servers.

Each remote Mingle server that receives this request will retrieve your public key from the master server, and then verify the digital signature using it. The public keys can also be cached for better performance. Modify the parameter "PublicTTL" to adjust the cache expiration time for public keys.

For example,

```
PublicTTL = 24
```

sets the public key cache expiration time to be 24 hours.

Both the key distribution and the request authentication processes are transparent to the user, so that you do not need to manage your own RSA keys. However, each Mingle server also has a pair of RSA keys to encrypt and decrypt passwords and private keys to protect them from being eavesdropped during transmission. You can manually specify the pair of keys used by a server by modifying the "PublicKey" and "PrivateKey" parameters in the configure file, or let the server create and handle them for you.

2.6 An example

Suppose you wish to set up a Mingle cluster consisting of two hosts: 'area51.cmcl.cs.cmu.edu' ('area51' in short) and 'grassynoll.cmcl.cs.cmu.edu' ('grassyknoll' in short). In both hosts, there will be one Mingle server running at port number 7018. You would like to configure the host 'area51' to be the master server. Your local user name at host 'area51' is 'abovik' and your local user name at host 'grassyknoll' is 'gbovik'. Your preferred Mingle ID is 'mbovik', and you wish to cache RSA keys for one day.

First, you log in to 'area51', and modify the following parameters in the configure file:

```
MinglePort = 7018
IsMaster = 1
ClusterHosts = "LOCAL", "grassyknoll"
UserMode = 1
PublicTTL = 24
PrivateTTL = 24
```

Then, start the Mingle server at 'area51':

```
[abovik@area51]$mingle -c mingle.config &
Mingle server 0.1
Local ip: 128.2.220.239  local port: 7018
Start accepting requests now...
```

And perform the Mingle single sign-on and Mingle init:

```
[abovik@area51]$mingle signon mbovik
Select your Mingle password: ********
Reenter password: ********
Success: Sign-on successfully.

[abovik@area51]$mingle init mbovik
Please enter your password: ********
Success: Mingle init successfully.

[abovik@area51]$mingle display
Mingle ID     = mbovik
Local user ID = abovik
```

Next, you log in to 'grassyknoll', and modify the following parameters in the configure file there:

```
MinglePort = 7018
IsMaster = 0
MasterHost = area51
UserMode = 1
PublicTTL = 24
PrivateTTL = 24
```

Start the Mingle server at 'grassyknoll':

```
[gbovik@grassyknoll]$mingle -c mingle.config &
Mingle server 0.1
Local ip: 128.2.220.241  local port: 7018
Start accepting requests now...
```

Finally, perform Mingle init at 'grassyknoll':

```
[gbovik@grassyknoll]$mingle init mbovik
Please enter your password: ********
Success: Mingle init successfully.
[gbovik@grassyknoll]$mingle display
Mingle ID     = mbovik
Local user ID = gbovik
```

From now on, you are done with the initial set up, you can start submitting requests from either '`area51`' or '`grassyknoll`' as you would like.

3 Sending remote requests

You can send a request from any Mingle host to any remote Mingle server using the '`-r`' option followed by a remote host list:

```
mingle command -r host-list
```

where the host-list is of the following format:

```
host1:port1  host2:port2 ...
```

For example,

```
mingle search bovik -r area51 grassyknoll:3000
```

will forward the request of searching keyword '`bovik`' to the host '`area51`' at the default Mingle port '`7018`' and the host '`grassyknoll`' at port '`3000`'.

As another example,

```
mingle index /home/bovik/test -r area51
```

will request the remote host '`area51`' to index the directory '`/home/bovik/test`' (Note: the '`/home/bovik/test`' directory should be a valid directory at **area51** before indexing).

If the host-list is specified as '`all`', then the local Mingle server forwards the corresponding request to all the Mingle servers inside the Mingle cluster. For example,

```
mingle search bovik -r all
```

4 Indexing

When a Mingle server is started on a host for the first time, none of the files on that host are indexed. You must make explicit requests to a Mingle server to index directory trees or files that you own. Thus Mingle is "opt-in" in the sense that users on Mingle hosts must issue explicit requests before their data is indexed and made available to Mingle clients.

Here is the syntax of an index request:

```
mingle index dir1/file1 dir2/file
```

A Mingle server stores the computed index table in the local disk. You can specify where to store the index table by setting the parameter "IndexDir" in the Mingle configure file. For performance optimization, Mingle maintains a cache in memory for frequently accessed terms and their indexes. Adjust the cache size using the parameter "CacheSize".

Mingle servers will update the index tables regularly. By modifying the parameter "IndexInterval", you can customize the frequency of such updates.

5 Searching

The syntax of a search request is:

```
mingle search keywords [-ts] [-a AND|OR] [-n num]
```

By default, Mingle performs search in file contents. Use '`-t`' option to search keywords in file names.

Use a pair of quotation marks "" to enclose all the keywords if multiple keywords are specified. Mingle also supports wildcard matching where you can use a "*" to match one or more characters.

If you use the '`-s`' option, then Mingle will display all the matching results in a file. By default, Mingle only displays the first result in each file.

Use the '`-a`' option to specify the logic operations among keywords. The default logic operation is "OR".

Use the '`-n`' option to specify the number of results to be displayed. By default, the first 10 results are displayed for each query.

For example,

```
mingle search "fit*"
```

will return the first 10 files that contain any word starting with prefix "fit". For each file, Mingle displays the first line that a keyword appears.

```
mingle search "computer bovik" -s -a AND -n 5
```

will return the first 5 files that contain both the keyword '`computer`' and the keyword '`bovik`'. For each returned file, Mingle displays all the matched lines.

6 Access control

Mingle allows file owners to grant search permissions to other Mingle users on a file-by-file basis. You can specify your access control decisions using an access-right mapping where you map a Mingle user U to a local user name (or group name) u, meaning the Mingle user U is able to search all the files that are searchable by the mapped local user (or group) u.

The syntax of an access-right mapping request is the following:

```
mingle addarp [-ug] mingle-id user-id (group-id)
```

Use the '`-u`' option if the user identified as '`mingle-id`' is mapped to a local user id '`user-id`'. Use the '`-g`' option for a local group id.

For example,

```
mingle addarp -u mbovik bovik
```

maps the Mingle user with Mingle ID '`mbovik`' to a local user name '`bovik`'.

You can delete an access-right mapping entry by using the command '`mingle rmarp`'. For example,

```
mingle rmarp -u mbovik bovik
```

You can use the command '`mingle cleararp`' to delete all access-right mappings associated with a specific Mingle user. For example,

```
mingle cleararp mbovik
```

will delete all access-right mappings created for the Mingle ID '`mbovik`'.

Use the command '`mingle lsarp`' to check the current access-right mappings. For example,

```
mingle lsarp mbovik
```

displays all the access-right mappings associated with Mingle ID '`mbovik`'.

```
mingle lsarp
```

displays the access-right mappings for all the Mingle users that you have granted permissions.

7 Searching AFS files

AFS is a distributed file system that enables sharing files across both local area and wide area networks. It is based on a distributed file system that was called the Andrew File System developed at Carnegie Mellon University.

Here are some general features of AFS:

1. In conjunction with Kerberos, AFS provides a global authentication system (all passwords are verified with a site-wide database).
2. Access Control Lists (ACLs) provide more flexibility in setting file access permissions than traditional Unix file systems.
3. Users can access AFS files at remote sites, if given appropriate permissions.

Mingle treats AFS files as normal files on another "disk". Current implementation supports CMU AFS file system, and may need to be revised to port to other AFS file systems.

AFS related commands include:

1. **mingle init**

   In order to index and search AFS files, you need to have a valid AFS user ID that are conveyed to Mingle via the Mingle init process using the following syntax:

   ```
   mingle init mingle-id -afs afs-id
   ```

   For example,

   ```
   mingle init mbovik -afs afsbovik
   ```

   associates the Mingle ID '`mbovik`' with the AFS user ID '`afsbovik`'.

   The init process interactively prompts you for your Mingle password and your AFS password, which it uses to authenticate to the master server and the Kerberos server. It also obtains a Kerberos ticket for the specified AFS user ID so that further operations such as indexing AFS files will be permitted. However, a Mingle server can hold only one (the latest) AFS user ticket at any time. Therefore, it is a good idea to perform the '`mingle init`' command every time before indexing AFS directories or files.

2. **mingle kauthd**

   Since a mingle server can hold only one AFS user ticket and an AFS ticket lasts for only 24 hours, the server may lose your AFS ticket when updating index tables. You can execute the '`mingle init`' command periodically, or use the '`mingle kauthd`' command to let server automatically periodically reauthenticate to Kerberos on your behalf:

   ```
   mingle kauthd mingle-id -afs afs-id
   ```

   Note that by executing the '`mingle kauthd`' command, you implicitly allow the Mingle server to remember your AFS password. However, the server will be careful not to ever write your password to the file system.

3. **mingle display**

   The '`mingle display`' command shows your current AFS user ID. For example,

   ```
   [abovik@area51]$mingle display
   Mingle ID     = mbovik
   Local user ID = gbovik
   AFS ID        = afsbovik
   ```

4. **mingle addarp/rmarp/lsarp**

   You can use the '`-a`' option to specify access-right mappings for an AFS user ID. For example,

   ```
   mingle addarp -a mfred afsbovik
   ```

   maps the Mingle user with Mingle ID '`mfred`' to AFS user ID '`afsbovik`'

Concept Index