IDA Teams Administrator Guide

Table of Contents

Last updated on October 16, 2023 — v8.3

1. Introduction

This manual describes the installation, management, and interaction with the key server-side components of an IDA Teams deployment.

It is primarily intended for administrators, and will focus on the different servers that are part of IDA Teams:

1. The The Hex-Rays Vault server
2. The The Lumina server

While we will (at least superficially) make use of the command-line clients that are used to access/manage those servers, this manual will not offer a detailed explanation of their usage: there are dedicated documents for that (e.g., the `hv` user manual, the `lc` user manual, …).

1.1. Let’s get started

The first server to install, and the one that is at the center of an IDA Teams deployment, is the Hex-Rays Vault server.

It is recommended to have the `hv` user manual ready before proceeding.

2. Installing the Hex-Rays Vault server

2.1. Prerequisites

After your purchase of IDA Teams licenses, you have received an e-mail that contains links to a download area where you will find:

- an installer for the IDA Teams server (also called the "Hex-Rays Vault server")
- this guide
- an installer for IDA
- an `ida.key`

All those will be necessary, so please go ahead and download them.

You will also need `root` access on the host where you will be installing the server.

2.2. Installation

This chapter explains how to install two parts of IDA Teams: the vault server, and a client.

We recommend installing a client first, to be able to connect to the server immediately after installation. The very first user to connect to the server becomes the administrator.

2.2.1. Installing clients

There are 2 Hex-Rays Vault clients:

1. `hv`: a command-line client (which we'll use in this document)
2. `hvui`: a GUI interface to the server

Vault clients are bundled with IDA Teams installers: simply run the IDA installer and follow the instructions. That will install IDA, and the 2 clients next to it.

2.2.2. Installing the server

The Hex-Rays Vault server can be installed on Linux servers. We have tested it on Debian and Ubuntu, but other major flavors of Linux should be fine too.

To install the server, run the Hex-Rays Vault installer as `root` and follow the instructions (the server will not require `root` permissions; only the installer does.)

TIP If your Linux system is based on `systemd` (e.g., Debian/Ubuntu, Red-Hat, CentOS, …), it is recommended to let the installer create systemd units so that the server will start automatically at the next reboot.

Once the server is installed, it will be necessary to activate its license.

Activating the server license

In order for the Hex-Rays Vault server license to be activated, it must be bound to a Host ID (an Ethernet MAC address.)

From a command prompt, run `/sbin/ifconfig`, and lookup the "ether" address for the network interface through which the server will be accessible.

```
>/sbin/ifconfig
enp4s0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        [...snipped...]
        ether bf:e2:91:10:58:d2  txqueuelen 1000  (Ethernet)
        [...snipped...]
```

In this case, our mac address is: `bf:e2:91:10:58:d2`

You will be able to activate both Hex-Rays Vault server and Lumina server in one activation if you have also the Host ID of your Lumina server.

Go to https://hex-rays.com/activate , and submit both the `ida.key` file and your MAC address. You will then receive another e-mail with instructions to download the following files:

- `hexvault.crt`
- `hexvault.key`
- `hexvault.lic`

Those need to be copied in the Hex-Rays Vault installation directory. As `root`:

```
>cd /opt/hexvault
>cp .../path/to/hexvault.crt .
>cp .../path/to/hexvault.key .
>cp .../path/to/hexvault.lic .
>chown hexvault:hexvault hexvault.crt hexvault.key hexvault.lic
>chmod 640 hexvault.crt hexvault.key hexvault.lic
```

Creating the initial database

At this point, the server should be ready to run.

CAUTION If your system is already in production and hosts files, skip this section. Using the `--recreate-schema` option as in the example below, will re-create an empty database and lose all history.

On the first install, you will need to initialize the database the server will use:

```
>sudo -u hexvault ./vault_server --config-file hexvault.conf \
                                 --vault-dir ./files \
                                 --recreate-schema
>2022-04-14 14:30:28 Vault Server v1.0 Hex-Rays (c) 2022
>2022-04-14 14:30:28 Database initialized; exiting.
```

Testing the server

Now that the server is installed and has a database to work with, we can test that it works:

```
>sudo -u hexvault ./vault_server --config-file hexvault.conf \
                                 --certchain-file hexvault.crt \
                                 --privkey-file hexvault.key \
                                 --license-file hexvault.lic \
                                 --vault-dir ./files
>2022-04-14 14:35:47 Vault Server v1.0 Hex-Rays (c) 2022
>2022-04-14 14:35:47 Using a license with 5 seats
>2022-04-14 14:35:47 Listening on 0.0.0.0:65433...
```

Good, the server appears to run! (If you are observing more worrying messages than this one, please refer to the troubleshooting section.)

At this point, you may want to either let the server run, or stop it (`Ctrl+C` will do) and restart it using systemd:

```
>systemctl restart hexvault.service
```

…and make sure it runs:

```
>ps aux | grep vault_server
hexvault  58246  0.0  0.0 ...
```

If you don’t see a running `vault_server` process, please refer to the `systemd` diagnostic tools (e.g., `journalctl`) for more info.

2.3. Initial configuration

This chapter explains how to perform the initial configuration of the vault server.

For the sake of the examples below, we'll imagine the following fictional group of users:

- Jane Smith, the department admin/IT head
- Fred Bloggs, senior reverse engineer

In addition, we'll assume:

- the company name is `Acme`
- the Hex-Rays Vault server has been installed on the company's LAN, on the host `hexvault.acme.com`

2.3.1. Creating the administrator

IMPORTANT The very first user to log into the server becomes the first administrator. S/he can create new administrators and otherwise manage the server.

Once the server is up and running, login to server using a username and password of your choice (`hv` is the vault client utility, it is installed as part of the client package.)

NOTE We will assume Jane installed IDA (and thus `hv`) in `/home/jane/idateams`.

```
>cd /home/jane/idateams
>./hv -hhexvault.acme.com -ujane -psecr3t info

Hex-Rays Vault Server v1
Vault time: 2022-04-14 15:28:03, up since 2022-04-14 15:17:25
License user : Jane Smith, IDA Ultimate
License email: jane@acme.com
License: IDAULTTM; 1 users out of 5; expires on 2023-04-05
MAC address: xx:xx:xx:xx:xx:xx
Vault directory: /opt/hexvault/files
Client name: jane *ADMIN*
Client site:
Client host: 127.0.0.1
Client root:
Login time : 2022-04-14 15:28:03
Last active: 2022-04-14 15:28:03
```

TIP Please note that there is no space between the command line switches and values.

Since Jane is the first user to login to the server, the credentials she provided, will be used to create the server's primary administrator.

You can verify that you are the only user by checking the user list:

```
>./hv -hhexvault.acme.com -ujane -psecr3t users

LastActive Adm    Login       License          Email
---------- --- ------------ --------------- ------------
2022-04-14  *  jane          <>
```

You may also add information (like your real name) to your user record by issuing:

```
>./hv -hhexvault.acme.com -ujane -psecr3t user edit jane "Jane Smith" jane@acme.com 1 "" 48-XXXX-XXXX-XX
>./hv -hhexvault.acme.com -ujane -psecr3t users

LastActive Adm    Login       License          Email
---------- --- ------------ --------------- ------------
2022-04-14  *  jane         48-XXXX-XXXX-XX Jane Smith <jane@acme.com>
```

However, note that having to pass a user name, host name and a password on the command line each time will get very tedious very fast. The next chapter will show how we can make our lives easier.

2.3.2. hv credentials

In order to connect to the vault server, hv must at least have:

- a username
- a password
- a hostname

For example:

```
$ hv -hhexvault.acme.com:65433 -uadmin -psecret users
LastActive Adm    Login        Email
---------- --- ------------ ------------
2022-06-27  *  admin
2022-06-22     alice        Alice <alice@acme.com>
Never          bob          Bob <bob@acme.com>
...
```

There are 3 ways to specify credentials (in decreasing order of priority):

- providing them as command-line arguments (as in the example above)
- storing them in environment variables
- storing them in the registry+keychain (recommended)

IMPORTANT All credentials, including usernames, are case-senstive, meaning that "Joe" and "joe" would be different users.

Command line

Passing credentials on the command line will always take precedence over environment variables and registry+keychain.

`-uUSERNAME` specify username

`-pPASSWORD` specify password

`-hHOST` specify host (server:port) (if port is omitted, defaults to 65433)

`-sSITENAME` specify site

`--set` remember credentials. This option doesn’t require the credentials to be passed through the command line, credentials passed through environment variables will work as well

Environment variables

Credentials can also be passed through environment variables. They will take precedence over those possibly found in the registry+keychain.

`VAULT_HOST` the server host name

`VAULT_PORT` the server port

`VAULT_USER` the username to connect to the server

`VAULT_PASS` the user’s password

`VAULT_SITE` the site to use (most commands need a site to operate)

Registry + keychain

Unless environment variables or command-line arguments are provided, hv will look for credentials in the registry (and the OS’s keychain for passwords.)

Credentials can be stored in the registry (and keychain) like so:

```
alice@alice_PC$ hv --set -ualice -palice -hvaultserver -salice_on_alicepc
```

The user, host (and optional site) will be persisted in the registry, while the password will be saved to the OS’s keychain.

NOTE For this operation to succeed, at least a user and host must be provided

TIP In order to keep the various commands' syntax as clear as possible, we will assume that the user has stored credentials (in either the registry+keychain or environment variables) for the rest of this manual.

Best practices

We recommend persisting credentials using the registry+keychain method.

Once that is done, commands will become cleaner:

```
>./hv info

Hex-Rays Vault Server v1
Vault time: 2022-04-14 15:36:29, up since 2022-04-14 15:17:25
...
```

TIP if you login to the server using `hvui` and save the login information, it will end up in the the registry+keychain method, and thus hv will then be able to use that information as well.

2.3.3. Adding users

To be able to connect to the vault server, users need to be added to the server. That can be done with the `user add` command:

```
>./hv user add fred "Fred Bloggs" fred@acme.com 0 ""
>./hv users

LastActive Adm    Login        Email
---------- --- ------------ ------------
Never          fred         Fred Bloggs <fred@acme.com>
2022-04-14  *  jane         Jane Smith <jane@acme.com>
```

Setting the new user’s password

Then, we need to set the user’s password, using the `passwd` command:

```
>./hv passwd stea1thy fred
```

2.3.4. Adding groups

To facilitate user management, sometimes it makes sense to make user groups. All users of a group then can be granted or denied access to certain files on the server.

Let’s add a few groups:

```
>./hv group add org
>./hv group add analysts
```

Using the `groups` command, we can see the new groups are still empty:

```
>./hv groups
analysts:
org:
```

We can now add group members:

```
>./hv group edit org jane 1
>./hv group edit org fred 1
>./hv group edit analysts fred 1
>./hv groups
analysts: fred
org: fred jane
```

Groups are especially useful for managing permissions.

2.4. Management

This chapter explains in detail how to perform regular administrator tasks.

2.4.1. Managing permissions

If you want to limit access to the files that will be stored on the vault server, you can specify who can access what. By default, the permission table grants all users access to all files:

```
>hv perm get
# The permission for each vault file is determined as the result of applying
# all matching lines, from the beginning of the permission table to the end.
# An empty permission table grants all access to everyone.
# A non-empty permission table starts by denying all access to everyone.
```

You will need to prepare a new permission table and put it into a file. The permission table consists of lines with the following format:

```
ACTION CATEGORY WHO PERM PATH
```

where:

ACTION

one of "grant" or "deny"

CATEGORY

one of "user" or "group"

WHO

name of the user or group to match

PERM

one of "list", "read", "write"

PATH

path pattern that the rule is for

Below is a sample permission table:

NOTE We'll assume the server has been in use for a while, and holds some files in the directories `subdir-for-fred/`, `local-secret/`, and `subdir/for/idbs/`.

```
# The permission for each vault file is determined as the result of applying
# all matching lines, from the beginning of the permission table to the end.
# An empty permission table grants all access to everyone.
# A non-empty permission table starts by denying all access to everyone.

# Fred can freely list, read, and modify all files inside "subdir-for-fred"
grant user fred write //subdir-for-fred/

# The "remote" group cannot even see "local-secret":
deny group remote list //local-secret

# The analysts can work on IDBs:
grant group analysts write //subdir/for/idbs/

# Everyone else may read them:
grant user * read //subdir/for/idbs/
```

The permissions have the following order:

- Adding the `read` permission also adds the `list` permission.
- Adding the `write` permission also adds the `list` and `read` permissions.
- Removing the `read` permission also removes the `write` permission.
- Removing the `list` permission also removes the `read` and `write` permissions.

Once the permission table is ready and stored in a file, we can install it:

```
>hv perm set @path/to/permission-file
```

After setting the permissions, it is a good idea to verify them. For example, this is how we can get a full list of files that `fred` can see, with the `rw` or `r-` prefixes, depending on the permissions:

```
>hv perm check fred //
rw //subdir-for-fred/afile
rw //subdir-for-fred/anotherfile
r- //subdir/for/idbs/malware.idb
```

Or we could limit our check to a particular file:

```
>hv perm check fred //local-secret
```

The empty output means that `fred` cannot see `local-secret` even though it exists.

2.4.2. Backup and restore

Currently, there is no dedicated procedure to back up the vault contents. It can be done by temporarily stopping the vault server and making a copy of the sqlite3 database as well as the files. The server must be stopped only during the backup of the sqlite3 database and then can be immediately restarted. It is ok to let the server run when making copies of the vault files. In the worst case some additional files will get copied in the backup, which normally will not cause problems. Since we never modify vault files but always create new revisions, there is no danger of copying inconsistent data.

Alternatively, it is possible to use sqlite3 backup functionality to make a backup of the database. Vault files can be copied using any Linux command (e.g. rsync or tar).

2.4.3. Upgrading the server

Switching to the newest versions of the Hex-Rays Vault server is recommended in order for the team to benefit from its improvements and new features.

The upgrade procedure consists of the following steps:

1. stopping the server. E.g., `sudo systemctl stop hexvault` if you are using `systemd` to manage the server.
2. performing a backup of the database
3. putting the new server instead of the old one
4. making sure the new server runs, upgrading the database schema to the new version if needed
5. restarting the server. E.g., `sudo systemctl start hexvault`

Verifying the database schema

Right after putting the new Hex-Rays Vault server binary into place, it is recommended to make sure it runs fine. To do that, we'll run the server manually just like we did the first time we installed it:

```
>sudo -u hexvault ./vault_server --config-file hexvault.conf \
                                 --certchain-file hexvault.crt \
                                 --privkey-file hexvault.key \
                                 --license-file hexvault.lic \
                                 --vault-dir ./files \
>2022-10-07 11:13:55 Vault Server v1.0 Hex-Rays (c) 2022
>2022-10-07 11:13:55 Using a license with 30 seats
>Error: obsolete database schema (5); use --upgrade-schema to upgrade it
```

In this case, the server complains that the database schema is outdated. This may happen as we will keep improving the Hex-Rays Vault server, and new versions might require an upgrade of the database schema in order to be able to work correctly.

Note that the Hex-Rays Vault server will not perform that upgrade automatically. That is on-purpose, to give you a chance to backup the database before proceeding.

Let’s tell the server to upgrade to the latest schema:

```
>sudo -u hexvault ./vault_server --config-file hexvault.conf \
                                 --certchain-file hexvault.crt \
                                 --privkey-file hexvault.key \
                                 --license-file hexvault.lic \
                                 --vault-dir ./files \
                                 --upgrade-schema
>2022-10-07 11:15:59 Vault Server v1.0 Hex-Rays (c) 2022
>2022-10-07 11:15:59 Upgrading the database schema from 5 to 8...
>2022-10-07 11:15:59 Database schema is up to date; exiting.
```

WARNING Please ensure you performed a backup of the database before issuing this command.

Once this is done, you should be able to restart the server in a normal way, and resume work.

2.4.4. Managing vault files

We plan to introduce additional functionalities like:

- obliteration of files
- periodic vault self-verification
- periodic backups
- usage stats

2.5. Hex-Rays Vault server command-line options

-p … (--port-number …)	Port number (default 65433)
-i … (--ip-address …)	IP address to bind to (default to any)
-c … (--certchain-file …)	TLS certificate chain file
-k … (--privkey-file …)	TLS private key file
-v (--verbose)	Verbose mode
(--upgrade-schema)	Upgrade database schema; then quit
-C … (--connection-string …)	Connection string
-l … (--log-file …)	Log file
-L … (--license-file …)	License file
-f … (--config-file …)	Config file
(--recreate-schema)	Drop & re-create schema; then quit **THIS WILL ERASE ALL DATA**
(--set-admin …)	Set admin password; requires username:password
-d … (--vault-dir …)	Directory for vault files
-e … (--log-level …)	Log level

2.6. Troubleshooting

This chapter explains how to solve typical problems with the vault server.

2.6.1. Connection issues

By default, the vault server listens on the TCP port 65433 on all interfaces. Please ensure that this port is enabled in your firewalls.

The vault server uses secure TLS connections with the clients. The TLS layer requires the certificate (.crt) and private key (.key) files. Usually, they are attached to the email message with the activation information.

2.6.2. Lost admin password

A lost admin password can be reset by following these steps:

- Stop the running server
- Launch the server with the `--set-admin` command line switch
- Start the server

In practice it may look like this:

```
>systemctl stop hexvault.service
>vault_server --config-file hexvault.conf --set-admin USERNAME:PASSWORD
>systemctl start hexvault.service
```

The uppercase USERNAME and PASSWORD placeholders should be replaced by the desired values. The user name and the password are separated by a colon.

The specified user must exist. If sh/e was not an admin before, s/he will be promoted to an admin by this command.

TIP If you do not know any valid users of the vault, use the `sqlite3` command line utility to list the users. They are stored in the `users` table.

2.6.3. Site verification

The following command:

```
>hv md5 PATH REVISION
```

can be used to retrieve MD5 checksums of the specified files.

PATH
path pattern to retrieve checksums from

REVISION
optional file revision. If not specified, the checksum of the last revision is reported

2.6.4. The server complains about a "world-accessible" file, and exits

The following files shouldn't be readable by everyone on the system, but only by `root` and `hexvault`:

- `hexvault.conf`: this file file holds the connection string to the database the server will use, and might contain credentials.
- `hexvault.crt`: the certificate chain
- `hexvault.key`: the private key file
- `hexvault.lic`: the license file

As a precaution, the Hex-Rays Vault server will refuse to start if these files are readable by unauthorized users.

Please make sure they:

- have `hexvault:hexvault` ownership: `chown hexvault:hexvault hexvault.crt hexvault.key hexvault.lic hexvault.conf`
- are not world-accessible: `chmod 640 hexvault.crt hexvault.key hexvault.lic hexvault.conf`

2.6.5. Licensing

The hexvault.lic file is tied to the MAC address of the first network interface. If they do not match, the server will not start. To change the MAC address, please contact support@hex-rays.com

2.6.6. Restoring from backups

There are no special precautions to take: restoring the sqlite3 database and vault files from a backup should be enough.

3. IDA Teams Lumina server

In addition to the Hex-Rays Vault server, IDA Teams users can make use of a private, on-the-premises Lumina server (unlike the Hex-Rays Vault server, the Lumina server isn't a strictly necessary component of an IDA Teams deployment.)

A Lumina server will have to be configured to point to the Hex-Rays Vault server

3.1. Installing the Lumina server

3.1.1. Prerequisites

After your purchase of IDA Teams licenses, you have received an e-mail that contains links to a download area where you will find, among other things:

- an installer for the Lumina server
- this guide
- a `lumact.key` , if you didn't activate Lumina server during the Hex-Rays Vault server activation
- `lumina.crt`, `lumina.key` and `lumina.lic` you recieved during the activation of Lumina server with Hex-Rays Vault server

NOTE `ida.key` and `lumact.key` may contain the same licenses information.

All those will be necessary, so please go ahead and download them.

You will also need `root` access on the host where you will be installing the Lumina server (to install the server, not to run it).

MySQL server

The Lumina server stores its data in a MySQL DBMS server. It is therefore necessary to have valid credentials to such a server, as well as a fresh, empty database.

NOTE The Lumina server requires a MySQL server version 5.8 or newer.

For illustration purposes, let's assume the MySQL database the server will use, is called: `"lumina_db"`.

Hex-Rays Vault server

In an IDA Teams setup, the Lumina server must have access to a running Hex-Rays Vault server in order to work: indeed, all user management is done through the Hex-Rays Vault server

Please make sure the Hex-Rays Vault server is properly installed & running.

3.1.2. Installation

At installation-time, the Lumina server installer will need information about the MySQL instance the Lumina server will be using (host, port, username, password). Eventually, that information will end up in the `lumina.conf` file, sitting next to the `lumina_server_teams` binary:

```
CONNSTR="mysql;Server=127.0.0.1;Port=3306;Database=lumina;Uid=lumina;Pwd=<snipped>"
```

In addition to information about the MySQL database in which the Lumina server will store the metadata, the installer asks for the `host`+`port` pair where to find a running Hex-Rays Vault server to which user authentication will be delegated.

That information will also end up in the `lumina.conf` file:

```
VAULT_HOST="vault.acme.com:65433"
```

Supported platforms

The Lumina server can be installed on Linux servers. We have tested it on Debian and Ubuntu, but other major flavors of Linux should be fine too.

To install the server, run the Lumina installer as `root` and follow the instructions (the server will not require `root` permissions; only the installer does.)

TIP If your Linux system is based on `systemd` (e.g., Debian/Ubuntu, Red-Hat, CentOS, …), it is recommended to let the installer create systemd units so that the server will start automatically at the next reboot.

Activating the server license

NOTE If you already requested Lumina server activation with Hex-Rays Vault server, you can skip this part.

In order for the Lumina server license to be activated, it must be bound to a Host ID (an Ethernet MAC address.)

From a command prompt, run `/sbin/ifconfig`, and lookup the "ether" address for the network interface through which the server will be accessible.

```
>/sbin/ifconfig
enp4s0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        [...snipped...]
        ether bf:e2:91:10:58:d2  txqueuelen 1000  (Ethernet)
        [...snipped...]
```

In this case, our mac address is: `bf:e2:91:10:58:d2`

Go to https://hex-rays.com/activate , and submit both the `ida.key` file and your MAC address. You will then receive another e-mail with instructions to download the following files:

- `lumina.crt`
- `lumina.key`
- `lumina.lic`

Installing the server license

Those need to be copied in the Lumina installation directory. As `root`:

```
>cd /opt/lumina
>cp .../path/to/lumina.crt .
>cp .../path/to/lumina.key .
>cp .../path/to/lumina.lic .
>chown lumina:lumina lumina.crt lumina.key lumina.lic
>chmod 640 lumina.crt lumina.key lumina.lic
```

Creating the initial database schema

At this point, the server should be ready to run.

CAUTION If your system is already in production and hosts files, skip this section. Using the `--recreate-schema` option as in the example below, will re-create an empty database and lose all data.

For the Lumina server to work, it needs to have a proper database schema to work with (at this point, the MySQL database (i.e., `"lumina_db"`) must already exist but is still empty.)

That is why, on the first install, you will need to initialize the database the server will use:

```
>sudo -u lumina ./lumina_server_teams --config-file lumina.conf --recreate-schema
Hex-Rays Lumina Server Teams v8.0 Hex-Rays (c) 2022
```

```
2022-09-02 10:28:30 Database has been initialized; exiting.
```

If you see "Error: Cannot connect to lumina db" please refer to troubleshooting section.

Testing the server

Now that the server is installed and has a database to work with, we can test that it works:

```
>sudo -u lumina ./lumina_server_teams --config-file lumina.conf \
                      --certchain-file lumina.crt \
                      --privkey-file lumina.key
Hex-Rays Lumina Server Teams v8.0 Hex-Rays (c) 2022
2022-09-22 12:14:37 Listening on 0.0.0.0:443...
```

Good, the server appears to run! (If you are observing more worrying messages than this one, please refer to the troubleshooting section.)

At this point, you may want to either let the server run, or stop it (`Ctrl+C` will do) and restart it using systemd:

```
>systemctl restart lumina.service
```

…and make sure it runs:

```
>ps aux | grep lumina_server_teams
lumina  78812  0.0  0.0 ...
```

If you don’t see a running `lumina_server_teams` process, please refer to the `systemd` diagnostic tools (e.g., `journalctl`) for more info.

3.1.3. Initial configuration

Hex-Rays Vault authentication

The Lumina server delegates authentication to the Hex-Rays Vault server. That is where the primary source of users information is located.

Consequently, it is the exact same set of users, with the exact same credentials as those that are able to use the Hex-Rays Vault server, that will be able to make use of the Lumina server.

Assuming Alice was registered in the Hex-Rays Vault server (and has admin rights), she should be able to use the `lc` utility to perform operations on the Lumina server.

NOTE `lc` is the Lumina command-line administration client, which comes with the Lumina server installer. We will assume the server has been installed in `/opt/lumina/`, and thus `lc` is present in `/opt/lumina/lc`.

```
>cd /opt/lumina
>./lc -hlumina.acme.com -ualice -psecr3t info
    Hex-Rays Lumina Server v8.0
    Lumina time: 2022-09-01 14:28:02, up since 2022-09-01 14:27:58
    MAC address: <snipped macaddr>
    Client name: alice *ADMIN*
    Client host: 127.0.0.1
```

```
>./lc -hlumina.acme.com -ualice -psecr3t users
    LastActive          Adm Login License         User name Email
    ------------------- --- ----- --------------- --------- --------------
    2022-09-01 14:28:04 *   alice AA-A11C-AC8E-01 Alice     alice@acme.com
    # Shown 1 results
```

"shadow" copy of users in the Lumina server

Although the authority for user management & authentication is the Hex-Rays Vault server, the Lumina server will need to "shadow" that information for its own database to be in a coherent state.

Every time a user opens a new connection to the Lumina server it will in turn perform a call to the Hex-Rays Vault server to authenticate the user against the provided credentials.

Depending on the success of that latter call, a "shadow" of the user will be recorded, updated, or deleted from the Lumina server's database.

NOTE Not everything is copied to the Lumina server's shadow users table; in particular, the password hash isn't.

Useful environment variables

To facilitate using `lc`, you may consider defining the following environment variables:

```
export LUMINA_HOST=lumina.acme.com
export LUMINA_USER=alice
export LUMINA_PASS=secr3t
```

After that, you can connect to the server effortlessly. For example, this command will print information about the server and the client:

```
>./lc info
    Hex-Rays Lumina Server v8.0
    Lumina time: 2022-09-01 14:28:02, up since 2022-09-01 14:27:58
    MAC address: <snipped macaddr>
    Client name: alice *ADMIN*
    Client host: 127.0.0.1
```

```
...
```

3.1.4. Lumina server command-line options

-p … (--port-number …)	Port number (default 443)
-i … (--ip-address …)	IP address to bind to (default to any)
-c … (--certchain-file …)	TLS certificate chain file
-k … (--privkey-file …)	TLS private key file
-v (--verbose)	Verbose mode
(--recreate-schema …)	Drop & re-create schema. Note that **THIS WILL ERASE ALL DATA**
(--upgrade-schema)	Upgrade database schema; then quit. Only necessary when upgrading to a newer version of the Lumina server.
-C … (--connection-string …)	Connection string
-l … (--log-file …)	Log file
-f … (--config-file …)	Config file
-D … (--badreq-dir …)	Directory holding dumps of requests causing internal errors

3.1.5. Troubleshooting

The server complains about a "world-accessible" file, and exits

The following files shouldn't be readable by everyone on the system, but only by `root` and `lumina`:

- `lumina.conf`: this file file holds the connection string to the database the server will use, and might contain credentials.
- `lumina.crt`: the certificate chain
- `lumina.key`: the private key file
- `lumina.lic`: the license file

As a precaution, the Lumina server will refuse to start if these files are readable by unauthorized users.

Please make sure they:

- have `lumina:lumina` ownership: `chown lumina:lumina lumina.crt lumina.key lumina.lic lumina.conf`
- are not world-accessible: `chmod 640 lumina.crt lumina.key lumina.lic lumina.conf`

MySQL

Before the first `--recreate-schema` command can succeed, it is necessary to create the MySQL database, as well as the user that it will be accessed as.

"Authentication plugin 'caching_sha2_password' cannot be loaded"

Some recent Linux distributions ship versions of MySQL that use the "caching_sha2_password" password-checking plugin (as opposed to the traditional "native password" one.)

In order for the Lumina server to able to use this strategy, it would have to ship with a `libmysqlclient.so` file that is linked against `libssl.so.3`.

Unfortunately, this library is not yet available on all of the various [versions of] distributions that we currently support, and introducing this dependency would significantly reduce the diversity of platforms on whith the Lumina server can run.

Consequently (for now) we have opted for another approach: the "lumina" MySQL user should use MySQL's "native password" strategy. This can be accomplished by issuing the following command in a MySQL prompt:

```
ALTER USER lumina@localhost IDENTIFIED WITH mysql_native_password BY '<luminauserpassword>';
```

Once `libssl.so.3` is more generally available, Lumina will not require this particular fine-tuning anymore.

MySQL TLS connections

For the same reasons as the `caching_sha2_password` issue, the Lumina server does not use TLS connections and thus will negociate a plain TCP connection to the MySQL server.

Creating the user and database

What follows is an example creating a user + database, on a Debian system:

```
>sudo mysql -uroot -p
[sudo] password for aundro:
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 14306
Server version: 10.1.48-MySQL-0+deb9u2 Debian 9.13

Copyright (c) 2000, 2018, Oracle, MySQL Corporation Ab and others.
```

```
    Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

    MySQL [(none)]> create user lumina@localhost;
    Query OK, 0 rows affected (0.00 sec)

    MySQL [(none)]> set password for lumina@localhost = PASSWORD('<snipped>');
    Query OK, 0 rows affected (0.00 sec)

    MySQL [(none)]> grant all on *.* to lumina@localhost;
    Query OK, 0 rows affected (0.00 sec)

    MySQL [(none)]> ALTER USER lumina@localhost IDENTIFIED WITH mysql_native_password BY '<snipped>';
    Query OK, 0 rows affected (0.00 sec)

    MySQL [(none)]> create database test_lumina;
    Query OK, 1 row affected (0.00 sec)

    MySQL [(none)]> [Ctrl+C] Bye
```

"Index column size too large. The maximum column size is 767 bytes."

The Lumina server cannot create its schema due to a particularly stringent limit on "index prefix sizes" in older versions of MySQL.

This limit was increased in MySQL **5.8**, and thus this is the minimum version the Lumina server can work with.

"Error: Cannot connect to lumina db"

In this case, edit the configuration file, by default `/opt/lumina/lumina.conf` and replace `Server=localhost` by `Server=127.0.0.1` in `CONNSTR`.

3.2. Concepts

3.2.1. What is the Lumina server

The Lumina server is a "functions metadata" repository.

It is a place where IDA users can **push**, and **pull** such metadata, to ease their reverse-engineering work: metadata can be extracted from existing projects, and re-applied effortlessly to new projects, thereby reducing (sometimes dramatically) the amount of time needed to analyze binaries.

Lumina server vs Hex-Rays Vault server: what is the difference?

While the workflow with the Hex-Rays Vault server and associated tools (hv, hvui and IDA's diff/merge modes) are extremely powerful for working on multiple revisions of the same binaries, the Lumina server in turn eases the replication of past efforts to new projects.

In effect, the Lumina server offers another "dimension" to collaborative reverse-engineering efforts.

Functions metadata

The Lumina server associates "function metadata" to functions, by means of a (`md5`) *hash* of those functions: whenever it wants to push information to, or pull information from the server, IDA will first have to compute hashes of the functions it wants to retrieve metadata for, and send those hashes to the Lumina server.

Similarly, when IDA **push**es information to the Lumina server, it will first compute hashes for the corresponding functions, extract the metadata corresponding to those from the `.idb` file, and send those hash+metadata pairs to the server.

Metadata contents

Metadata about functions can include:

- function name
- function address
- function size
- function prototype
- function [repeatable] comments
- instruction-specific [repeatable] comments
- anterior/posterior (i.e., "extra") comments
- user-defined "stack points" in the function's frame
- the function frame description and stack variables
- instructions operands representations

Pushing & overriding metadata

When a user pushes metadata about a function whose `md5` hash isn't present in the database, the Lumina server will simply create a new record for it.

However, when a user pushes metadata about a function whose `md5` hash (and associated metadata) is already present in the database, the Lumina server will attempt to "score" the quality of the old metadata and the quality of the new metadata. If the score of the new metadata is higher, the new function metadata will override the previous one.

NOTE When a user asks IDA to push *all* functions to the Lumina server, IDA will automatically skip some functions: those that still have a "dummy" name (e.g., `sub_XXXX`), or that are below a certain size threshold (i.e., 32 bytes) will be ignored.

Metadata history

The Lumina server retains a history of the metadata associated to functions. Using the `lc` utility, it is possible to dig into that history, and view changes (detailed diffs, too.)

File contents

It’s worth pointing out that when pushing metadata to the Lumina server, IDA will not push the binary file itself. Only the following metadata about the file itself will be sent:

- the name of the input file
- the name of the IDB file
- a `md5` hash of the input file

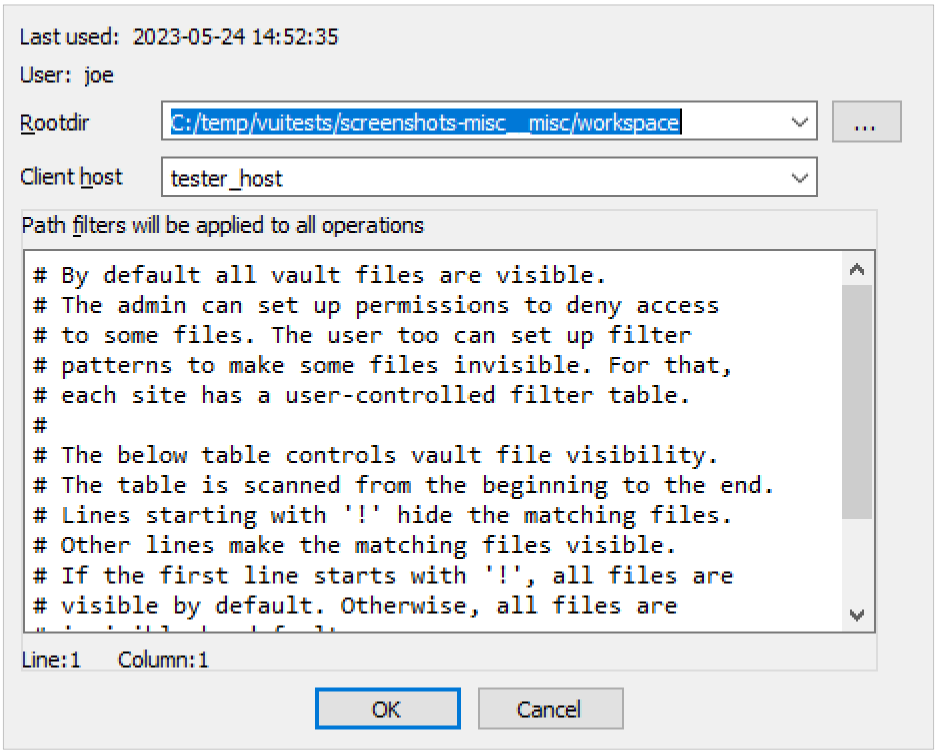
The Lumina server cannot therefore be used as a backup/repository for binary files & IDBs (that would be the role of the Hex-Rays Vault server)

4. Miscellaneous

4.1. What is a "site"?

A site represents a mapping of the server files to the local filesystem. Normally each computer has a site associated with it. A site has the following attributes:

- A site name
- A host name
- The path to a folder on the filesystem (a.k.a., "root directory")
- Path filters (optional)



Last used: 2023-05-24 14:52:35

User: joe

Rootdir: C:/temp/vuitests/screenshots-misc__misc/workspace

Client host: tester_host

Path filters will be applied to all operations

```
# By default all vault files are visible.
# The admin can set up permissions to deny access
# to some files. The user too can set up filter
# patterns to make some files invisible. For that,
# each site has a user-controlled filter table.
#
# The below table controls vault file visibility.
# The table is scanned from the beginning to the end.
# Lines starting with '!' hide the matching files.
# Other lines make the matching files visible.
# If the first line starts with '!', all files are
# visible by default. Otherwise, all files are
```

Line:1 Column:1

4.1.1. Root directory

The root directory is the essential attribute of a site. It denotes where all files from the vault server will be mapped to the local disk. Everything inside the root directory can potentially be uploaded to the vault server and shared with other team members.

The vault server cannot manage files located outside the root directory. However, this limitation is straightforward to overcome: create a symbolic link (or, on Windows, a junction point) from the root directory to the directory of your choice. This will make the target of the symbolic link visible as part of the root directory.

The vault server keeps track of each site’s state: what files have been downloaded to the local disk, what files have been checked out for editing, etc. This simplifies the housekeeping tasks, especially for big repositories with millions of files. Even for them, downloading the latest files or reconciling the local disk with the server, are almost instantaneous.

The host name is a security feature that prevents from using a site on a wrong computer. Since the server keeps track of the files downloaded to each site, using a wrong site may lead to an inconsistent mapping between the server and local disk. However, if the user does not want this protection, it is possible to erase the host name in the site definition.

Sites can be edited from the "Sites" view.

4.1.2. Path filters

By default all server files are visible, but for servers that manage gigabytes of data this can be problematic: it may be undesirable for users to download all files to their local computer.

Site filters provide a mechanism that lets users restrict the set of files their IDA Teams client works with. Users who want to work on some specific projects can set a filter that restricts the visibility only to selected subdirectories.

Each site has its own filters, that con be modified at any time. Filters do not directly affect any files on the local disk, or on the server: they are strictly about visibility.

WARNING Site filters are meant simplify a user's life by letting them focus on specific projects. Since they can be modified by users, they should **not** be considered a security measure: that would be the role of the permissions system, which can only be managed by Hex-Rays Vault server administrators.

NOTE The purpose of site filters is to create a subset of the full set of files provided by the server. Site filters don't directly affect what locally-available files (i.e., present in the site's rootdir, but not tracked by the server) are visible by IDA Teams clients.

There is another mechanism to specify what files should not be added to the vault. See `.hvignore` for more info.

Examples

An empty filter

```
$ cat empty_filter.txt
$
```

Hide all files, except those in `malware/`

```
$ cat only_malware.txt
malware/
$
```

Show all files, except those from the pentesting team

```
$ cat hide_pentest.txt
!pentesting/
$
```

Show all files but those from the pentesting team, except their produced documents

```
$ cat hide_pentest_but_docs.txt
!pentesting/
pentesting/research_docs/
$
```

4.2. The registry

On Microsoft Windows, IDA Teams will store certain bits of information in the registry (host name, user name, site name.)

On macOS and Linux, it will use a pseudo-registry file, located at `$HOME/.idapro/hvui.reg`.

4.3. Passwords storage in the OS's keychain

While hosts, user names & site names are persisted to the registry, passwords are stored securely in the operating system's keychain.

- On Windows, the Windows Credential Store is used (therefore requiring Windows 7 or newer)
- On macOS, the macOS Keychain is used
- On Linux, the "Secret service" is used (through `libsecret-1`)