UNICORE UFTPD SERVER

UNICORE Team

Document Version:	1.0.0
Component Version:	3.0.0
Date:	27 04 2021

This work is co-funded by the EC EMI project under the FP7 Collaborative Projects Grant Agreement Nr. INFSO-RI-261611.

Contents

This is the UFTPD user manual providing information on running and using the UNICORE UFTP server *uftpd*. Please note also the following places for getting more information:

UNICORE Website: https://www.unicore.eu

Support list: unicore-support@lists.sf.net

Developer's list: unicore-devel@lists.sf.net

UFTP issue tracker: https://sourceforge.net/p/unicore/uftp-issues

Source code repo: https://github.com/UNICORE-EU/uftp

1 UNICORE UFTP

UFTP is a data streaming library and file transfer tool.

UFTP is best used using the *uftp* client-side application (available from https://sourceforge.net/-projects/unicore/files/Clients/UFTP-Client), but is easily integrated into custom applications due to its FTP compliance.

It can be also integrated into UNICORE, allowing to transfer data from client to server (and vice versa), as well as providing data staging and third-party transfer between UFTP-enabled UNICORE sites.

A full UFTP server installation consists of two parts

- the "uftpd" file server
- either a UNICORE/X server, or a standalone authentication service.

This manual covers the UFTP file server "uftpd" version 3.0 and higher, written in Python.

1.1 UFTP features

- dynamic firewall port opening using passive FTP. UFTPD requires only a single open port
- standalone command line client available
- supports third-party FTP clients such as *curl* or *ftp*, after getting a one-time password via the Auth server (RESTful API)
- integrated into UNICORE clients for fast file upload and download
- integrated with UNICORE servers for fast data staging and server-to-server file transfers
- optional encryption of the data streams using a symmetric key algorithm
- optional compression of the data streams (using gzip)

- optional multiple TCP streams per data connection Based on code from the JPARSS library, Copyright (c) 2001 Southeastern Universities Research Association, Thomas Jefferson National Accelerator Facility
- partial reads/writes to a file. If supported by the filesystem, multiple UFTP processes can thus read/write a file in parallel (striping)
- command port protected by SSL

1.2 How does UFTP work

The UFTP file server, called *uftpd*, listens on two ports (which may be on two different network interfaces):

- the command port receives control commands
- the listen port accepts data connections from clients.

The uftpd server is "controlled" (by a UNICORE/X or Auth server) via the command port, and receives/sends data directly from/to a client machine (which can be an actual user client machine or another server). The client connnects to the "listen" port, which has to be accessible from external machines. The client opens additional data commection(s) via the passive FTP protocol.

The sequence for a UFTP file transfer is as follows:

- the client (which can be an end-user client, or a service such as a UNICORE/X server) sends an authentication request to the Auth server (or another UNICORE/X server)
- the Auth server sends a request to the command port of UFTPD. This request notifies the UFTPD server about the upcoming transfer and contains the following information
  - a "secret", i.e. a one-time password which the client will use to authenticate itself
  - the user and group id which uftpd should use to access files
  - an optional key to encrypt/decrypt the data
  - the client's IP address
- the UFTPD server will now accept an incoming client connection, provided the supplied "secret" (one-time password) matches the expectation.
- if everything is OK, an FTP session is created, and the client can use the FTP protocol to open data connections, list files, transfer data etc. Data connections are opened via "passive FTP", which allows the firewall to dynamically open the requested ports (which can by any port, see below if you want to a fixed port range).
- for each UFTP session, UFTPD will fork a process which runs as the requested user (with the requested primary group)

[NOTE] .IMPORTANT SECURITY NOTE

The UNICORE UFTPD server is running with root privileges. Make sure to read and understand the section below on protecting the command socket. Otherwise, users logged on to the UFTPD machine can possibly read and write other user's files.

2 Installation and operation

2.1 Prerequisites

- Python 3.4.0 or later
- the server's "listen" port needs to be accessible through your firewalls, declaring it an "FTP" port (FTP connection tracking). Alternatively a fixed range of open ports can be configured and used
- the server's command port needs to be accessible from the Auth server(s)
- the UFTPD server needs access to the target file systems
- a server certificate for the UFTPD server is a MUST for production use in a multi-user environment (see the section on SSL below)

A functional UFTP installation requires either an Auth server or a full UNICORE/X server.

NOTE ON PATHS

The UNICORE UFTPD server is distributed either as a platform independent and portable tar.gz or zip bundle, or as an installable, platform dependent package such as RPM.
Depending on the installation package, the paths to various files are different. If installing using distribution-specific package the following paths are used:

```
CONF=/etc/unicore/uftpd
BIN=/usr/share/unicore/uftpd/bin
LIB=/usr/share/unicore/uftpd/lib
```

If installing using the portable bundle, all UFTPD files are installed under a single directory. Path prefixes are as follows, where INST is the directory where UFTPD was installed:

```
CONF=INST/conf
BIN=INST/bin
LIB=INST/lib
```

These variables (CONF, BIN and LOG) are used throughout the rest of this manual.

Note that after installation UFTPD is NOT automatically enabled as a systemd service, since you will need to edit the configuration and provide a server certificate.

2.2 Starting and stopping the UFTPD server

If using the Linux packages, uftpd is integrated as a service via systemd, and you can stop/start it via *systemctl*. Also, logging is (by default) done via systemd, and you can look at the logs via *journalctl*

To do things manually, you can use the start/stop and status scripts that are provided in the BIN directory.

- `unicore-uftpd-start.sh` starts the server
- `unicore-uftpd-stop.sh` stops the server
- `unicore-uftpd-status.sh` checks the server status

The parameters such as server host/port, control host/port, and others are configured in the `CONF/uftpd.conf` file

In a production scenario with multiple users, the uftpd server needs to be started as root. This is necessary to be able to access files as the correct user/group and set correct file permissions.

To enable UFTPD as a systemd service (after configuring and adding a server certificate), you can use *systemctl* :

```
sudo systemctl add-wants multi-user.target unicore-uftpd
```

2.3 Configuration parameters

The following variables can be defined in the configuration file (uftpd.conf):

```
CMD_HOST        : the interface where the server listens for ↩
    control commands

CMD_PORT        : the port where the server listens for  ↩
   control commands

SERVER_HOST     : the interface where the server listens for ↩
    client data
                  connections

SERVER_PORT     : the port where the server listens for  ↩
   client data
                  connections
```

```
ADVERTISE_HOST  : (optional) (Only used in the PASV  ↩
   implementation)
                  Advertise this server as having the  ↩
                     following IPv4 address in the
                  control connection. This is useful if the  ↩
                     server is behind
                  a NAT firewall and the public address is  ↩
                     different from
                  the address(es) the server has bound to

SSL_CONF        : File containing SSL settings for the  ↩
   command port

ACL             : File containing the list of server DNs  ↩
   that are allowed
                  access to the command port

MAX_CONNECTIONS : the maximum number of concurrent control  ↩
   connections per user (default: 16)

MAX_STREAMS     : the maximum number of parallel TCP streams ↩
    per FTP session (default: 4)

PORT_RANGE      : (optional) server-side port range in the  ↩
   form 'lower:upper' that will be
                  used to accept data connections. By  ↩
                     default, any free ports will be used.
                  Example: set to '50000:50050' to limit the ↩
                      port range

DISABLE_IP_CHECK : (optional) in some situations, the client ↩
    IP can be different from
                   the one that was sent to the UFTPD server ↩
                       by the Auth server.
                   This will lead to rejected transfers.  ↩
                      Setting this variable to "true"
                   will disable the IP check. Only the one- ↩
                      time password will be checked

UFTP_KEYFILES    : (optional) list of files (relative to  ↩
   current user's $HOME) where uftpd will
                   read public keys for authentication. List ↩
                       is separated by ":". This defaults
                           to ".ssh/authorized_keys"
```

```
UFTP_NO_WRITE    : (optional) ":"-separated list of file  ↩
   name patters that uftpd should
                  not write to

LOG_VERBOSE      : set to 'true' to get (much) more detailed ↩
    logging

LOG_SYSLOG       : set to 'false' to print logging output to ↩
    stdout
```

As usual if you set the SERVER_HOST to be "0.0.0.0", the server will bind to all the available network interfaces.

If possible, use an "internal" interface for the Command socket. If that is not possible, make sure the Command socket is protected by a firewall!

We STRONGLY recommend enabling SSL for the Command socket. Please refer to the next section.

2.4 Protecting the Command socket

Using SSL for the Command port ensures that only trusted parties (i.e. trusted Auth and/or UNICORE/X servers) can issue commands to the UFTPD server. To further limit the set of trusted users, an access control list (ACL) file is used.

In production settings where users can log in to the UFTPD server machine, SSL MUST be enabled to prevent unauthorized data access!

[NOTE] .IMPORTANT SECURITY NOTE

Without SSL enabled, users logged in to the UFTPD server can easily create exploits to read or write files with arbitrary user privileges (except *root*).

2.4.1 SSL setup

To setup SSL, you need a PEM file containing the UFTPD server's credential, and a PEM file containing certificate authorities that should be trusted.

The following properties can be set in the `CONF/uftpd-ssl.conf` file.

```
credential.path=path/to/keyfile.pem
credential.password=...

truststore=path/to/ca-cert-file.pem
```

If the `credential.path` property is NOT set, SSL will be disabled.

Backwards (in)compatibility to previous versions
UFTPD 2.x SSL config is NOT supported
If you already have a p12 keystore for UFTPD 2.x, you can use *openssl* to convert it to PEM format

2.4.2 ACL setup

The access control list contains the distinguished names of those certificates that should be allowed access.

The "ACL" setting in `CONF/uftpd.conf` is used to specify the location of the ACL file

```
export ACL=conf/uftpd.acl
```

The default ACL contains the certificate DN of the UNICORE/X server from the UNICORE core server bundle. In production, you need to replace this by the actual DNs of your UNICORE/X server(s) and UFTP Authentication server(s).

The ACL entries are expected in RFC2253 format. To get the name from a certificate in the correct format using openssl, you can use the following OpenSSL command:

```
$> openssl x509 -in your_server.pem -noout -subject -nameopt ↩
    RFC2253
```

The ACL file can be updated at runtime.

2.5 Firewall configuration

UFTPD requires * an open TCP port for accepting FTP connections * additional open TCP ports for accepting data connections

The data connections can either be openend dynamically using "FTP connection tracking", or you can use a dedicated port range and permanently open those in the firewall.

Note
Please consult the firewall documentation on how to enable an "FTP" service on your firewall (or operating system).

With Linux iptables, you may use rules similar to the following:

```
iptables -A INPUT -p tcp -m tcp --dport $SERVER_PORT -j ACCEPT
iptables -A INPUT -p tcp -m helper --helper ftp-$SERVER_PORT -j  ↩
    ACCEPT
```

where $SERVER_PORT is the SERVER_PORT defined in uftpd.conf. The first rule allows anyone to access port $SERVER_PORT. The second rule activates the iptables connection tracking FTP module on port $SERVER_PORT.

On some operating systems it may be required to load additional kernel modules to enable connection tracking, for example on CentOS:

```
modprobe nf_conntrack_ipv4
modprobe nf_conntrack_ftp ports=$SERVER_PORT
```

If you cannot use connection tracking, you will need to open a port range, and configure UFTPD accordingly.

For example, in `uftpd.conf` :

```
export PORT_RANGE=21000:21010
```

and the iptables rule

```
iptables -A INPUT -p tcp -m tcp --dport 21000:21010 -j ACCEPT
```

would allow incoming data connections on ports 21000 to 21010.

A fairly small range (e.g. 10 ports) is usually enough, since these are server ports.

2.6 Logging

By default, UFTPD writes to syslog, and you can use *journalctl* to read log messages.

To print logging output to stdout, set "export LOG_SYSLOG=false" in the `uftpd.conf` file.

3 UNICORE integration

Please refer to the UNICORE/X manual for detailed information on how to configure UFTP based data access and data transfer.

4 Testing the UFTPD server

You should use the *uftp* client to run tests, which contains many options such as the number of concurrent FTP connections, and can use /dev/null and /dev/zero as data source/sink.