HOW TO RUN YULL.EXE

This is an introduction to how to run YULL.EXE, the command-line version of YULL. There is also the GUI version, YULLG.EXE. The parameters and switches discussed below are also valid for YULLG but that application is not discussed here.

Please see https://YullEncryption.com/Guide for more help with this product.

As a command-line app, Yull is run from a DOS prompt or Powershell. It can be run as a scheduled job in a corporate context, or from a batch file.

As an encryption app, Yull needs at least three pieces of data: a file to encrypt, a key to use for that encryption, and where to write the encrypted file.

Yull has a lot of parameters and switches. To run YULL, however, you need only specify a key, the file you want to encrypt, and the file or folder you want to store the encrypted data, like this:

YULL in=M:\test.txt key=M:\mykeyfile out=M:\test.outfile

In this case, YULL will attempt to open M:\test.txt; it will try to find the file M:\mykeyfile and use that as the encryption key and it will write the output to M:\test.outfile. If you choose to keep the in-file's name unchanged you can do that as well, either by specifying that on the command line or simply using

out=M:\encrypteddata (as an example) where encrypteddata is a folder on the M: drive.

Yull takes 20 possible parameters and 12 possible switches.

Parameters are arguments that take a value like **in=file.txt** or **keysize=590**

Switches are preceded by a forward slash / and take no other value. **/decrypt** is an example of a switch.

Their very presence evaluates to true for that switch.

All arguments are case-insensitive.

PARAMETERS CAN BE PLACED INTO AN OPTIONS FILE SO THE

COMMAND LINE CAN BE AS SIMPLE AS

Yull in=file @options

Here, "options" is the name of the options file. In fact, the options file can be any legal name you want to use and be located anywhere a local application can access. The @ sign tells Yull that what follows is the options file.

HOW YULL PROCESSES COMMAND LINE PARAMTERS

Command line parameters are the words or phrases, separated by spaces, that appear on a line after the name of the executable. Each program has its own way of handling command line parameters. There are very few rules.

YULL processes command line parameters in the order in which they are encountered, starting with the left most parameter.

```
M:\>yull in=file.txt maxbuffsize=80 minbuffsize=12 maxrounds=60 minrounds=18
```

Here, the first parameter that YULL processes is “IN” and it sets the source file (or in-file) to “file.txt”. The next parameter is the **MAXBUFFSIZE.** This is the suggested max size of the reads, so when YULL runs, it will read the in-file in this case File.Txt at most 80 bytes at a time.

All the numerical parameters are tied to the encryption level, discussed below.

Yull might not read 80 bytes at a time. As with all numerical parameters, the **MAXBUFFSIZE** is only a **suggestion** to YULL. Same for the **MINBUFFSIZE.** This is 12 here. Again, Yull treats it as a suggestion. Chances are, YULL will not use 12 but it will probably not go lower than 12.

MAXROUNDS is the next parameter. Like MAXBUFFSIZE and MINBUFFSIZE, MAXROUNDS is a suggestion to YULL. In this case it is set to 60 and MINROUNDS to 18. What happens is that YULL determines how many reads are required and based on that, creates an array (like a list) of numbers, the ROUNDS. Each ROUND is calculated based partly on the MAX and MINROUNDS parameters. There will be the same number of ROUNDS values as there are reads.

With these parameters, if there were 5 reads, YULL might submit each read , meaning YULL will run each read through the encryption routines a series of times based on the number of reads and some calculations for the MAXROUNDS, based on the encryption LEVEL. The MAXROUNDS value here is a suggestion to that calculation.

This does not mean 50 encrypts. The encryption routines can be called hundreds of thousands to millions or even billions of times. This just means that the reads are submitted to the encryption code a number of times derived from a series of numbers calculated with the “suggestion” of 50 rounds.

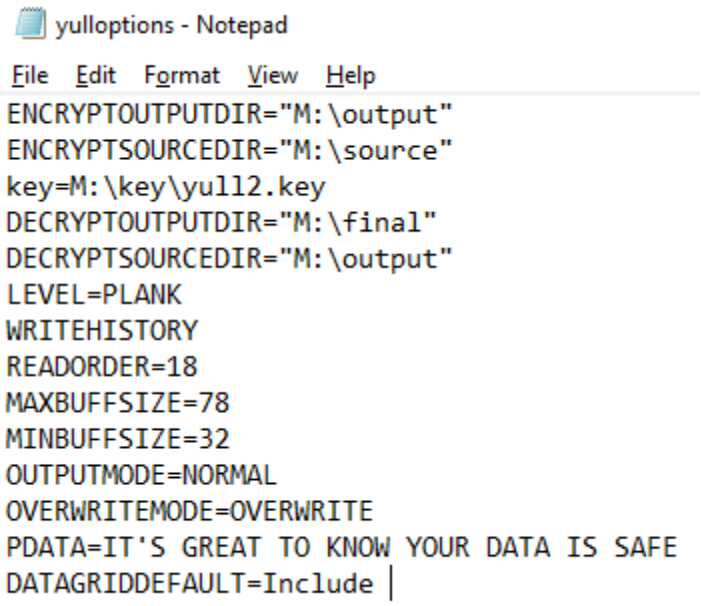
If the read size is, say, 50, and the file is 6000 bytes long, then there will be 120 reads (120 x 50 = 6000). In this case, Yull will create a series of 120 values for the ROUNDS, based on the MAXROUNDS suggested value, the MINROUNDS value (if there is one) and other factors like the LEVEL.

The last parameter processed is the one that is used.

```
M:\>yull in=file.txt maxbuffsize=80 minbuffsize=12 maxrounds=60 minrounds=18 maxrounds=45
```

Notice, that in this example **MAXBUFFSIZE** appears again at the end of the command line. Now it is set to 45, so 45 is what YULL will use. In the case of duplicate parameters, the last parameter “wins”.

This is not very useful when YULL is run like this. BUT you can package your parameters into an **OPTIONS** file, like this:



The Options file is specified by being preceded with the “at” sign: @.

yull in=file.txt @yulloptions

This will instruct YULL to encrypt file.txt (encrypt is the default) with the parameters found in the yulloptions file in the local directory. The options file, like any of the other file parameters, can be anywhere reachable by the local system.

You could put the @optionsfile (that is, @<options file path> as the first parameter on the command line, like this:

yull @yulloptions <rest of commandline>

You could include a key in your options file but supersede that key with a different key later on the command line. That way if someone steals your options file they might assume that the key referenced in it was THE key when it’s not.

The most important thing to note about running YULL is that the parameters used for encryption must be used for decrypting.

Here is the list of parameters and switches and what they mean:

PARAMETER	**MEANING AND USE**
IN	The source file or folder IN=M:\accounts*.xlst is perfectly valid as is In=M:\accounts in which case YULL will encrypt all files in accounts. Wildcards are OK

OUT	The destination file or folder. Generally Yull does not change the name of the destination file. If you are encrypting a file like this: in=myfile.txt out=R:\test\output (a folder) then Yull will encrypt myfile.txt to R:\test\output\myfile.txt. Assuming R: is a valid drive spec and a folder \test\output exists. The input and output files do not have to be on the same disk. Just reachable by an ordinary desktop app. For the IN and OUT parameters, if you specify a full qualified path name, Yull will use that. If you specify a source or output dir for use (see just below) you do not need to use a FQPN for the file.
ENCRYPTSOURCEDIR	Specifies the source directory. If used like: yull in=myfile.txt encryptsourcedir=L:\temp then Yull will look for myfile.txt in the L:\temp directory.
ENCRYPTOUTPUTDIR	As with the ENCRYPTSOURCEDIR, Yull will use this value to store the encrypted file. So you can use a command line like this: yull in=c:\test\myfile.txt encryptoutputdir=M:\output Yull will store the encrypted file myfile.txt in M:\output. This makes more sense when it is part of an options file.
DECRYPTSOURCEDIR	With this option YULL will look into the folder this points to as the source of files to be decrypted.
DECRYPTOUTPUTDIR	With this option, Yull will store the decrypted output files in the folder this value points to.
DIRECTION	Either ENCRYPT or DEDCRYPT. You can also use /encrypt or /decrypt. If none are specified, encrypt is assumed.

/ASKFORKEY	YULL will pause it's operation until you type in a complete path to a key. This information is NOT stored for later use.
KEY	This is the FQPN to the key. Yull's keys are between 100 and 50000 bytes long. Yull can use any file for a key, even text files, files of all one value, system files, whatever you want. No key is harmed while Yull runs.
KEYSIZE	IF you have no key and you want YULL to create one you can specify a size for the key Yull will create, between 100 and 50000 bytes.
KEYDIR	If you have no key and want YULL to create one, this will instruct YULL to create a key and put it in this folder (FQPN).
MAXBUFFSIZE	This an integer between 30 and 250. This will instruct YULL to use a number no greater than the MAXBUFFSIZE you've selected for reading. That is, if you have a MAXBUFFSIZE of 175, YULL will read your data in chunks no greater than 175 bytes at a time. This does not mean the size is 175, just that it is no greater than 175. If omitted, this value will be determined by the encryption LEVEL (see below).
MINBUFFSIZE	Same as MAXBUFFSIZE. Yull will not read data in sizes smaller than the MINBUFFSIZE. Again, this too is determined by the LEVEL. And as with all parameters, the Yull Rules of Precedence apply. WHATEVER the MAXBUFFSIZE and MINBUFFSIZE values are, MAXBUFFSIZE cannot be smaller than or equal to MINBUFFSIZE.
MAXROUNDS	When Yull encrypts (or decrypts) it works on data ROUNDS number of times. If the rounds is, say, 10, then YULL will encrypt (or decrypt) that data 10 times. Valid values are between 5 and 150, inclusive. Again, this too is determined by the LEVEL. And as with all parameters, the Yull Rules of Precedence apply. And again, note: This doesn't mean that your data

	gets encrypted ROUNDS number of times. It means that each read of your data gets submitted to the encryption package ROUNDS number of times. In that encryption package, your data can be encrypted thousands to billions of times.
MINROUNDS	As with MAXROUNDS, YULL will not encrypt the data less than MINROUNDS number of times. The values are 1 to 50. WHATEVER the MAXROUNDS and MINROUNDS values are, MAXROUNDS cannot be smaller than or equal to MINROUNDS. Again, this too is determined by the LEVEL. And as with all parameters, the Yull Rules of Precedence apply.
DUMMYDATAAMOUNT	A number between 20 and 100. This is the amount of dummy (i.e., random data) YULL adds to each read. Again, this too is determined by the LEVEL. And as with all parameters, the Yull Rules of Precedence apply.
READORDER	This is a number between 3 and 254. The numbers are NOT ORDERED. In no sense for this parameter is 10 greater than, say, 5. They are just values used in calculating how YULL reads the data. **Yull reads and writes the data out of disk order.** This number is one of the values used to determine the read order. NOTE: Yull will ALWAYS read and write the data out of disk order.
PDATA	This is a string of text data. It is NOT a password. It can contain spaces, High Order bit characters, anything you can type into a box, even using ALT-CTRL+numeric keypad combinations. YULL uses this (among other data items) to encrypt your key. Examples of PDATA are: **Now is the time for all good programmers to ˝"⌡ς of their country.** (the weird characters were inputted with the alt+numeric key pad. You can use as many as 200 characters for the PDATA. You there have a much larger space than the upper and lower alphabetic characters plus numbers and other normal keyboards characters.

/ASKPDATA	YULL will ask you for your PDATA at runtime.
LEVEL	This is the encryption level. The default is PLANK. There are six levels: PLANK (PLANK, the fastest, is the default, meaning if no LEVEL is specified, YULL will use PLANK), TINY, NEURO, FAST, NORMAL, MAX. The default values for the other parameters will change depending on the level. Plank is fast, meaning, small read buffer sizes, fewer rounds per read. MAX is slower, meaning larger read buffer sizes (meaning, more encryptions per read), more rounds per read, etc. You can see how these interact in when you run YULLG (the GUI version of YULL). As you change the LEVEL, the other values will change. You can leave these values as they are or you can alter them. BUT, if you do alter them, please save your changes in an Options file. There will be no way you can recover your data if you alter one of these parameters and then forget what you did.
OUTPUTMODE	NORMAL, MINIMUM, SILENT. This determines how much data Yull (the command line program) outputs at the end of a run. For NORMAL, Yull will output a lot of details about what it has done. For MINIMUM, YULL will just output that it ran and the run has ended. For SILENT, YULL will output nothing.
/USEDEFAULTKEY	If you have created a default key, then this switch will instruct YULL to use it.
/CREATEDEFAULTKEY	If you have not created a default key, Yull will create one using your KEY parameters if you have any. If you instruct YULL to create a default key, YULL will also create a default folder. This will be (by default) in your C:\users\<username>\AppData\Local\YullEncryption folder. Yull will create an encrypts folder to hold encrypted files, history.txt, and a YullOptions file, along with a <username>.key file. As with most everything with YULL, you can use this, change it, use something else. With YULL it doesn't matter.
/SHRED	Yull will SHRED your source files after encrypting them by overwriting the file with random data and then truncating them and then deleting them. This will NOT affect files in your Shadow Copies (assuming that is enabled).

	See the web site for more details about this.
/USEDEFAULTOPTIONS	If you have created a default options folder and key, you can specify this on the command line.
/encryptOptions	Use this to encrypt your options file, like this: yull in=optionsfile /encryptoptions Yull will ask you for a password and then will encrypt the options file. After this, each time you run YULL with this options file, YULL will ask you for the password. NOTE: You can use any legal file name you wish for the options file and it can be located anywhere reachable from your local system.
/HELP	The help text. Or just Yull<return> on the command line.
-LEGAL	Yull displays the license agreement. Note the dash.

Contact help@yullencryption.com for more information and to report any issues.

Or visit https://YullEncryption.com

Once you have set up your default folder and key,

all you need do is something like this:

yull in=inputfile <return>

As Yull will assume encryption and use default options.

You could encrypt an entire folder like this:

yull in=p:\secretdocumentsfolder

It’s THAT easy!