Sign in

vanhauser-thc / **thc-hydra** Public

Notifications | Fork 2k | Star 9.7k

Code | Issues 53 | Pull requests 4 | Actions | Projects | Security | Insights

master | Go to file | Code

- .github
- hydra-gtk
- .clang-format
- .gitignore
- .travis.yml
- Android.mk
- CHANGES
- CITATION.cff
- Dockerfile
- INSTALL
- LICENSE
- LICENSE_OPENSSL
- Makefile
- Makefile.am
- Makefile.orig

About

hydra

bruteforce hydra penetration-testing brute-force-attacks brute-force pentesting pentest password-cracker network-security bruteforce-attacks password-cracking pentest-tool bruteforcing brute-force-passwords thc bruteforcer

Readme

AGPL-3.0 license

Activity

9.7k stars

385 watching

2k forks

Report repository

Releases 13

hydra 9.5 Latest
on Jun 12, 2023

+ 12 releases

- Makefile.unix
- PROBLEMS
- README
- TODO
- _config.yml
- bfg.c
- bfg.h
- configure
- crc32.c
- crc32.h
- d3des.c
- d3des.h
- dpl4hydra.sh
- dpl4hydra_full.csv
- dpl4hydra_local.csv
- hmacmd5.c
- hmacmd5.h
- hydra-adam6500.c
- hydra-afp.c
- hydra-asterisk.c
- hydra-cisco-enable.c
- hydra-cisco.c

Packages

No packages published

Contributors 78

+ 64 contributors

Languages

● C 94.5% ● Makefile 2.5%
● Shell 2.2% ● Other 0.8%

- hydra-cobaltstrike.c
- hydra-cvs.c
- hydra-firebird.c
- hydra-ftp.c
- hydra-http-form.c
- hydra-http-proxy-url...
- hydra-http-proxy.c
- hydra-http.c
- hydra-http.h
- hydra-icq.c
- hydra-imap.c
- hydra-irc.c
- hydra-ldap.c
- hydra-logo.ico
- hydra-logo.rc
- hydra-memcached.c
- hydra-mod.c
- hydra-mod.h
- hydra-mongodb.c
- hydra-mssql.c
- hydra-mysql.c
- hydra-ncp.c

- hydra-nntp.c
- hydra-oracle-listener.c
- hydra-oracle-sid.c
- hydra-oracle.c
- hydra-pcanywhere.c
- hydra-pcnfs.c
- hydra-pop3.c
- hydra-postgres.c
- hydra-radmin2.c
- hydra-rdp.c
- hydra-redis.c
- hydra-rexec.c
- hydra-rlogin.c
- hydra-rpcap.c
- hydra-rsh.c
- hydra-rtsp.c
- hydra-s7-300.c
- hydra-sapr3.c
- hydra-sip.c
- hydra-smb.c
- hydra-smb2.c
- hydra-smtp-enum.c

hydra-smtp.c

hydra-snmp.c

hydra-socks5.c

hydra-ssh.c

hydra-sshkey.c

hydra-svn.c

hydra-teamspeak.c

hydra-telnet.c

hydra-time.c

hydra-vmauthd.c

hydra-vnc.c

hydra-wizard.sh

hydra-xmpp.c

hydra.1

hydra.c

hydra.h

libpq-fe.h

ntlm.c

ntlm.h

README AGPL-3.0 license

H Y D R A

```
                        (c) 2001-2023 by van Hauser / THC
             <vh@thc.org> https://github.com/vanhauser-
thc/thc-hydra
       many modules were written by David (dot) Maciejak @
gmail (dot) com
                 BFG code by Jan Dlabal <dlabaljan@gmail.com>

                    Licensed under AGPLv3 (see LICENSE file)

           Please do not use in military or secret service
organizations,
                          or for illegal purposes.
      (This is the wish of the author and non-binding. Many
people working
       in these organizations do not care for laws and ethics
anyways.
            You are not one of the "good" ones if you ignore
this.)

           NOTE: no this is not meant to be a markdown doc!
old school!


Hydra in the most current github state can be directly
downloaded via docker:
```
docker pull vanhauser/hydra
```



INTRODUCTION
------------
Number one of the biggest security holes are passwords, as
every password
security study shows.
This tool is a proof of concept code, to give researchers and
security
consultants the possibility to show how easy it would be to
gain unauthorized
access from remote to a system.

THIS TOOL IS FOR LEGAL PURPOSES ONLY!

There are already several login hacker tools available,
however, none does
either support more than one protocol to attack or support
parallelized
connects.

It was tested to compile cleanly on Linux, Windows/Cygwin,
Solaris,
FreeBSD/OpenBSD, QNX (Blackberry 10) and MacOS.

Currently this tool supports the following protocols:
 Asterisk, AFP, Cisco AAA, Cisco auth, Cisco enable, CVS,
Firebird, FTP,
 HTTP-FORM-GET, HTTP-FORM-POST, HTTP-GET, HTTP-HEAD, HTTP-
POST, HTTP-PROXY,
 HTTPS-FORM-GET, HTTPS-FORM-POST, HTTPS-GET, HTTPS-HEAD,
HTTPS-POST,
 HTTP-Proxy, ICQ, IMAP, IRC, LDAP, MEMCACHED, MONGODB, MS-SQL,
MYSQL, NCP, NNTP, Oracle Listener,
 Oracle SID, Oracle, PC-Anywhere, PCNFS, POP3, POSTGRES,
Radmin, RDP, Rexec, Rlogin,
 Rsh, RTSP, SAP/R3, SIP, SMB, SMTP, SMTP Enum, SNMP v1+v2+v3,
SOCKS5,
 SSH (v1 and v2), SSHKEY, Subversion, Teamspeak (TS2), Telnet,
VMware-Auth,
 VNC and XMPP.

However the module engine for new services is very easy so it
won't take a
long time until even more services are supported.
Your help in writing, enhancing or fixing modules is highly
appreciated!! :-)



WHERE TO GET
------------
You can always find the newest release/production version of
hydra at its
project page at https://github.com/vanhauser-thc/thc-
hydra/releases
If you are interested in the current development state, the
public development
repository is at Github:
  svn co https://github.com/vanhauser-thc/thc-hydra
 or
  git clone https://github.com/vanhauser-thc/thc-hydra
Use the development version at your own risk. It contains new
features and
new bugs. Things might not work!

Alternatively (and easier) to can pull it as a docker
container:
```
docker pull vanhauser/hydra
```
```

```
HOW TO COMPILE
--------------
To configure, compile and install hydra, just type:

```
./configure
make
make install
```

If you want the ssh module, you have to setup libssh (not
libssh2!) on your
system, get it from https://www.libssh.org, for ssh v1
support you also need
to add "-DWITH_SSH1=On" option in the cmake command line.
IMPORTANT: If you compile on MacOS then you must do this - do
not install libssh via brew!

If you use Ubuntu/Debian, this will install supplementary
libraries needed
for a few optional modules (note that some might not be
available on your distribution):

```
apt-get install libssl-dev libssh-dev libidn11-dev libpcre3-
dev \
                 libgtk2.0-dev libmysqlclient-dev libpq-dev
libsvn-dev \
                 firebird-dev libmemcached-dev libgpg-error-
dev \
                 libgcrypt11-dev libgcrypt20-dev
```

This enables all optional modules and features with the
exception of Oracle,
SAP R/3, NCP and the apple filing protocol - which you will
need to download and
install from the vendor's web sites.

For all other Linux derivates and BSD based systems, use the
system
software installer and look for similarly named libraries like
in the
command above. In all other cases, you have to download all
source libraries
and compile them manually.
```

```
SUPPORTED PLATFORMS
-------------------
- All UNIX platforms (Linux, *BSD, Solaris, etc.)
- MacOS (basically a BSD clone)
- Windows with Cygwin (both IPv4 and IPv6)
- Mobile systems based on Linux, MacOS or QNX (e.g. Android,
iPhone, Blackberry 10, Zaurus, iPaq)



HOW TO USE
----------
If you just enter `hydra`, you will see a short summary of the
important
options available.
Type `./hydra -h` to see all available command line options.

Note that NO login/password file is included. Generate them
yourself.
A default password list is however present, use "dpl4hydra.sh"
to generate
a list.

For Linux users, a GTK GUI is available, try `./xhydra`

For the command line usage, the syntax is as follows:
 For attacking one target or a network, you can use the new
"://" style:
  hydra [some command line options]
PROTOCOL://TARGET:PORT/MODULE-OPTIONS
 The old mode can be used for these too, and additionally if
you want to
 specify your targets from a text file, you *must* use this
one:

```
hydra [some command line options] [-s PORT] TARGET PROTOCOL
[MODULE-OPTIONS]
```


Via the command line options you specify which logins to try,
which passwords,
if SSL should be used, how many parallel tasks to use for
attacking, etc.

PROTOCOL is the protocol you want to use for attacking, e.g.
ftp, smtp,
http-get or many others are available
TARGET is the target you want to attack
MODULE-OPTIONS are optional values which are special per
```

```
PROTOCOL module

FIRST - select your target
 you have three options on how to specify the target you want
to attack:
 1. a single target on the command line: just put the IP or
DNS address in
 2. a network range on the command line: CIDR specification
like "192.168.0.0/24"
 3. a list of hosts in a text file: one line per entry (see
below)

SECOND - select your protocol
 Try to avoid telnet, as it is unreliable to detect a correct
or false login attempt.
 Use a port scanner to see which protocols are enabled on the
target.

THIRD - check if the module has optional parameters
 hydra -U PROTOCOL
 e.g. hydra -U smtp

FOURTH - the destination port
 this is optional, if no port is supplied the default common
port for the
 PROTOCOL is used.
 If you specify SSL to use ("-S" option), the SSL common port
is used by default.


If you use "://" notation, you must use "[" "]" brackets if
you want to supply
IPv6 addresses or CIDR ("192.168.0.0/24") notations to attack:
  hydra [some command line options] ftp://[192.168.0.0/24]/
  hydra [some command line options] -6
smtps://[2001:db8::1]/NTLM

Note that everything hydra does is IPv4 only!
If you want to attack IPv6 addresses, you must add the "-6"
command line option.
All attacks are then IPv6 only!

If you want to supply your targets via a text file, you can
not use the ://
notation but use the old style and just supply the protocol
(and module options):
  hydra [some command line options] -M targets.txt ftp
You can also supply the port for each target entry by adding
":<port>" after a
target entry in the file, e.g.:
```

```
```
foo.bar.com
target.com:21
unusual.port.com:2121
default.used.here.com
127.0.0.1
127.0.0.1:2121
```

Note that if you want to attach IPv6 targets, you must supply
the -6 option
and *must* put IPv6 addresses in brackets in the file(!) like
this:

```
foo.bar.com
target.com:21
[fe80::1%eth0]
[2001::1]
[2002::2]:8080
[2a01:24a:133:0:00:123:ff:1a]
```

LOGINS AND PASSWORDS

You have many options on how to attack with logins and
passwords
With -l for login and -p for password you tell hydra that this
is the only
login and/or password to try.
With -L for logins and -P for passwords you supply text files
with entries.
e.g.:

```
hydra -l admin -p password ftp://localhost/
hydra -L default_logins.txt -p test ftp://localhost/
hydra -l admin -P common_passwords.txt ftp://localhost/
hydra -L logins.txt -P passwords.txt ftp://localhost/
```

Additionally, you can try passwords based on the login via the
"-e" option.
The "-e" option has three parameters:

```
s - try the login as password
n - try an empty password
r - reverse the login and try it as password
```
```

~~~~
```

If you want to, e.g. try "try login as password and "empty
password", you
specify "-e sn" on the command line.

But there are two more modes for trying passwords than -p/-P:
You can use text file which where a login and password pair is
separated by a colon,
e.g.:

```
admin:password
test:test
foo:bar
```

This is a common default account style listing, that is also
generated by the
dpl4hydra.sh default account file generator supplied with
hydra.
You use such a text file with the -C option - note that in
this mode you
can not use -l/-L/-p/-P options (-e nsr however you can).
Example:

```
hydra -C default_accounts.txt ftp://localhost/
```

And finally, there is a bruteforce mode with the -x option
(which you can not
use with -p/-P/-C):

```
-x minimum_length:maximum_length:charset
```

the charset definition is `a` for lowercase letters, `A` for
uppercase letters,
`1` for numbers and for anything else you supply it is their
real representation.
Examples:

```
-x 1:3:a generate passwords from length 1 to 3 with all
lowercase letters
-x 2:5:/ generate passwords from length 2 to 5 containing only
slashes
-x 5:8:A1 generate passwords from length 5 to 8 with uppercase
~~~~

~~~~
and numbers
-x '3:3:aA1&~#\\ "\'<{([-|_^@)]=}>$%*?./§,;:!`' -v generates
lenght 3 passwords with all 95 characters, and verbose.
```

Example:

```
hydra -l ftp -x 3:3:a ftp://localhost/
```

SPECIAL OPTIONS FOR MODULES
---------------------------
Via the third command line parameter (TARGET SERVICE OPTIONAL)
or the -m
command line option, you can pass one option to a module.
Many modules use this, a few require it!

To see the special option of a module, type:

  hydra -U <module>

e.g.

  ./hydra -U http-post-form

The special options can be passed via the -m parameter, as 3rd
command line
option or in the service://target/option format.

Examples (they are all equal):

```
./hydra -l test -p test -m PLAIN 127.0.0.1 imap
./hydra -l test -p test 127.0.0.1 imap PLAIN
./hydra -l test -p test imap://127.0.0.1/PLAIN
```

RESTORING AN ABORTED/CRASHED SESSION
------------------------------------
When hydra is aborted with Control-C, killed or crashes, it
leaves a
"hydra.restore" file behind which contains all necessary
information to
restore the session. This session file is written every 5
minutes.
NOTE: the hydra.restore file can NOT be copied to a different
platform (e.g.
from little endian to big endian, or from Solaris to AIX)
~~~~

```
HOW TO SCAN/CRACK OVER A PROXY
------------------------------
The environment variable HYDRA_PROXY_HTTP defines the web
proxy (this works
just for the http services!).
The following syntax is valid:

```
HYDRA_PROXY_HTTP="http://123.45.67.89:8080/"
HYDRA_PROXY_HTTP="http://login:password@123.45.67.89:8080/"
HYDRA_PROXY_HTTP="proxylist.txt"
```

The last example is a text file containing up to 64 proxies
(in the same
format definition as the other examples).

For all other services, use the HYDRA_PROXY variable to
scan/crack.
It uses the same syntax. eg:

```
HYDRA_PROXY=
[connect|socks4|socks5]://[login:password@]proxy_addr:proxy_po
rt
```

for example:

```
HYDRA_PROXY=connect://proxy.anonymizer.com:8000
HYDRA_PROXY=socks4://auth:pw@127.0.0.1:1080
HYDRA_PROXY=socksproxylist.txt
```

ADDITIONAL HINTS
----
* sort your password files by likelihood and use the -u option
to find
  passwords much faster!
* uniq your dictionary files! this can save you a lot of time
:-)
    cat words.txt | sort | uniq > dictionary.txt
* if you know that the target is using a password policy
(allowing users
  only to choose a password with a minimum length of 6,
containing a least one
  letter and one number, etc. use the tool pw-inspector which
comes along
  with the hydra package to reduce the password list:
    cat dictionary.txt | pw-inspector -m 6 -c 2 -n >
passlist.txt

RESULTS OUTPUT

The results are output to stdio along with the other
information. Via the -o
command line option, the results can also be written to a
file. Using -b,
the format of the output can be specified. Currently, these
are supported:

* `text`   - plain text format
* `jsonv1` - JSON data using version 1.x of the schema
(defined below).
* `json`   - JSON data using the latest version of the schema,
currently there
             is only version 1.

If using JSON output, the results file may not be valid JSON
if there are
serious errors in booting Hydra.

JSON Schema

Here is an example of the JSON output. Notes on some of the
fields:

* `errormessages` - an array of zero or more strings that are
normally printed
   to stderr at the end of the Hydra's run. The text is very
free form.
* `success` - indication if Hydra ran correctly without error
(**NOT** if
   passwords were detected). This parameter is either the
JSON value `true`
   or `false` depending on completion.
* `quantityfound` - How many username+password combinations
discovered.
* `jsonoutputversion` - Version of the schema, 1.00, 1.01,
1.11, 2.00,
   2.03, etc. Hydra will make second tuple of the version to
always be two
   digits to make it easier for downstream processors (as
opposed to v1.1 vs
   v1.10). The minor-level versions are additive, so 1.02
will contain more
```

```
   fields than version 1.00 and will be backward compatible.
Version 2.x will
   break something from version 1.x output.

Version 1.00 example:
```
```
{
    "errormessages": [
        "[ERROR] Error Message of Something",
        "[ERROR] Another Message",
        "These are very free form"
    ],
    "generator": {
        "built": "2021-03-01 14:44:22",
        "commandline": "hydra -b jsonv1 -o results.json ...
...",
        "jsonoutputversion": "1.00",
        "server": "127.0.0.1",
        "service": "http-post-form",
        "software": "Hydra",
        "version": "v8.5"
    },
    "quantityfound": 2,
    "results": [
        {
            "host": "127.0.0.1",
            "login": "bill@example.com",
            "password": "bill",
            "port": 9999,
            "service": "http-post-form"
        },
        {
            "host": "127.0.0.1",
            "login": "joe@example.com",
            "password": "joe",
            "port": 9999,
            "service": "http-post-form"
        }
    ],
    "success": false
}
```
```


SPEED
-----
through the parallelizing feature, this password cracker tool
can be very
fast, however it depends on the protocol. The fastest are
generally POP3
```

```
and FTP.
Experiment with the task option (-t) to speed things up! The
higher - the
faster ;-) (but too high - and it disables the service)

STATISTICS

Run against a SuSE Linux 7.2 on localhost with a "-C FILE"
containing
295 entries (294 tries invalid logins, 1 valid). Every test
was run three
times (only for "1 task" just once), and the average noted
down.

```
                        P A R A L L E L    T A S K S
SERVICE 1       4       8       16      32      50      64
100     128
------- ------------------------------------------------------
--------------
telnet  23:20   5:58    2:58    1:34    1:05    0:33    0:45*
0:25*   0:55*
ftp     45:54   11:51   5:54    3:06    1:25    0:58    0:46
0:29    0:32
pop3    92:10   27:16   13:56   6:42    2:55    1:57    1:24
1:14    0:50
imap    31:05   7:41    3:51    1:58    1:01    0:39    0:32
0:25    0:21
```

(*)
Note: telnet timings can be VERY different for 64 to 128
tasks! e.g. with
128 tasks, running four times resulted in timings between 28
and 97 seconds!
The reason for this is unknown...

guesses per task (rounded up):

  295   74   38   19   10   6   5   3
3

guesses possible per connect (depends on the server software
and config):

  telnet   4
        ftp   6
        pop3   1
        imap   3

BUGS & FEATURES

Hydra:
Email me or David if you find bugs or if you have written a
new module.
vh@thc.org (and put "antispam" in the subject line)

You should use PGP to encrypt emails to vh@thc.org :

```
-----BEGIN PGP PUBLIC KEY BLOCK-----
Version: GnuPG v3.3.3 (vh@thc.org)
```


```
mQINBFIp+7QBEADQcJctjohuYjBxq7MELAlFDvXRTeIqqh8kqHPOR018xKL09p
ZT
KiBWFBkU48xlR3EtV5fC1yEt8gDEULe5o0qtK1aFlYBtAWkflVNjDrs+Y2BpjI
TQ
FnAPHw0SOOT/jfcvmhNOZMzMU8lIubAVC4cVWoSWJbLTv6e0DRIPiYgXNT5Quh
6c
vqhnI1C39pEo/W/nh3hSa16oTc5dtTLbi5kEbdzml78TnT0OASmWLI+xtYKnP+
5k
Xv4xrXRMVk4L1Bv9WpCY/Jb6J8K8SJYdXPtbaIi4VjgVr5gvg9QC/d/QP2etmw
3p
lJ1Ldv63x6nXsxnPq6MSOOw8+QqKc1dAgIA43k6SU4wLq9TB3x0uTKnnB8pA3A
CI
zPeRN9LFkr7v1KUMeKKEdu8jUut5iKUJVu63lVYxuM5ODb6Owt3+UXgsSaQLu9
nI
DZqnp/M6YTCJTJ+cJANN+uQzESI4Z2m9ITg/U/cuccN/LIDg8/eDXW3VsCqJz8
Bf
lBSwMItMhs/Qwzqc1QCKfY3xcNGc4aFlJz4Bq3zSdw3mUjHYJYv1UkKntCtvvT
CN
DiomxyBEKB9J7KNsOLI/CSst3MQWSG794r9ZjcfA0EWZ9u6929F2pGDZ3LiS7J
x5
n+gdBDMe0PuuonLIGXzyIuMrkfoBeW/WdnOxh+27eemcdpCb68XtQCw6UQARAQ
AB
tB52YW4gSGF1c2VyICgyMDEzKSA8dmhAdGhjLm9yZz6JAjkEEwECACMCGwMCHg
EC
F4AFAlIp/QcGCwkIAwcCBhUKCQgLAgUWAwIBAAAKCRDI8AEqhCFiv2R9D/9qTC
JJ
xCH4BUbWIUhw1zRkn9iCVSwZMmfaAhz5PdVTjeTelimMh5qwK2MNAjpR7vCCd3
BH
Z2VLB2Eoz9MOgSCxcMOnCDJjtCdCOeaxiASJt8qLeRMwdMOtznM8MnKCIO8X4o
o4
qH8eNj83KgpI50ERBCj/EMsgg07vSyZ9i1UXjFofFnbHRWSW9yZO16qD4F6r4S
Gz
```
```

```
dsfXARcO3QRI5lbjdGqm+g+HOPj1EFLAOxJAQOygz7ZN5fj+vPp+G/drONxNyV
Kp
QFtENpvqPdU9CqYh8ssazXTWeBi/TIs0q0EXkzqo7CQjfNb6tlRsg18FxnJDK/
ga
V/1umTg41bQuVP9gGmycsiNI8Atr5DWqaF+O4uDmQxcxS0kX2YXQ4CSQJFi0pm
l5
slAGL8HaAUbV7UnQEqpayPyyTEx1i0wK5ZCHYjLBfJRZCbmHX7SbviSAzKdo5J
Il
Atuk+atgW3vC3hDTrBu5qlsFCZvbxS21PJ+9zmK7ySjAEFH/NKFmx4B8kb7rPA
OM
0qCTv0pD/e4ogJCxVrqQ2XcCSJWxJL31FNAMnBZpVzidudNURG2v61h3ckkSB/
fP
JnkRy/yxYWrdFBYkURImxD8iFD1atj1n3EI5HBL7p/9mHxf1DVJWz7rYQk+3cz
vs
IhBz7xGBz4nhpCi87VDEYttghYlJanbiRfNh3okCOAQTAQIAIgUCUin7tAIbAw
YL
CQgHAwIGFQgCCQoLBBYCAwECHgECF4AACgkQyPABKoQhYr8OIA//cvkhoKay88
yS
AjMQypach8C5CvP7eFCT11pkCt1DMAO/8Dt6Y/Ts10dPjohGdIX4PkoLTkQDwB
DJ
HoLO75oqj0CYLlqDI4oHgf2uzd0Zv8f/11CQQCtut5oEK72mGNzv3GgVqg60z2
KR
2vpxvGQmDwpDOPP620tf/LuRQgBpks7uazcbkAE2Br09YrUQSCBNHy8kirHW5m
5C
nupMrcvuFx7mHKW1z3FuhM8ijG7oRmcBWfVoneQgIT3l2WBniXg1mKFhuUSV8E
rc
XIcc11qsKshyqh0GWb2JfeXbAcTW8/4IwrCP+VfAyLO9F9khP6SnCmcNF9EVJy
R6
Aw+JMNRin7PgvsqbFhpkq9N+gVBAufz3DZoMTEbsMTtW4lYG6HMWhza2+8G9Xy
aL
ARAWhkNVsmQQ5T6qGkI19thB6E/T6ZorTxqeopNVA7VNK3RVlKpkmUu07w5bTD
6V
l3Ti6XfcSQqzt6YX2/WUE8ekEG3rSesuJ5fqjuTnIIOjBxr+pPxkzdoazlu2zJ
9F
n24fHvlU20TccEWXteXj9VFzV/zbPEQbEqmE16lV+bO8U7UHqCOdE83OMrbNKs
zl
7LSCbFhCDtflUsyClBt/OPnlLEHgEE1j9QkqdFFy90l4HqGwKvx7lUFDnuF8LY
sb
/hcP4XhqjiGcjTPYBDK254iYrpOSMZSIRgQQEQIABgUCUioGfQAKCRBDlBVOdi
ii
tuddAJ4zMrge4qzajScIQcXYgIWMXVenCQCfYTNQPGkHVyp3dMhJ0NR21TYoYM
C5
Ag0EUin7tAEQAK5/AEIBLlA/TTgjUF3im6nu/rkWTM7/gs5H4W0a04kF4UPhaJ
UR
gCNlDfUnBFA0QD7Jja5LHYgLdoHXiFelPhGrbZel/Sw6sH2gkGCBtFMrVkm3u7
tt
x3AZlprqqRH68Y5xTCEjGRncCAmaDgd2apgisJqXpu0dRDroFYpJFNH3vw9N2a
62
0ShNakYP4ykVG3jTDC4MSl2q3BO5dzn8GYFHU0CNz6nf3gZR+48BG+zmAT77pe
TS
```

```
+C4Mbd6LmMmB0cuS2kYiFRwE2B69UWguLHjpXFcu9/85JJVCl2CIab7l5hpqGm
gw
G/yW8HFK04Yhew7ZJOXJfUYlv1EZzR5bOsZ8Z9inC6hvFmxuCYCFnvkiEI+pOx
PA
oeNOkMaT/W4W+au0ZVt3Hx+oD0pkJb5if0jrCaoAD4gpWOte6LZA8mAbKTxkHP
Br
rA9/JFis5CVNI688O6eDiJqCCJjPOQA+COJI+0V+tFa6XyHPB4LxA46RxtumUZ
MC
v/06sDJlXMNpZbSd5Fq95YfZd4l9Vr9VrvKXfbomn+akwUymP8RDyc6Z8BzjF4
Y5
02m6Ts0J0MnSYfEDqJPPZbMGB+GAgAqLs7FrZJQzOZTiOXOSIJsKMYsPIDWE8l
Xv
s77rs0rGvgvQfWzPsJlMIx6ryrMnAsfOkzM2GChGNX9+pABpgOdYII4bABEBAA
GJ
Ah8EGAECAAkFAlIp+7QCGwwACgkQyPABKoQhYr+hrg/9Er0+HN78y6UWGFHu/K
VK
d8M6ekaqjQndQXmzQaPQwsOHOvWdC+EtBoTdR3VIjAtX96uvzCRV3sb0XPB9S9
eP
gRrO/t5+qTVTtjua1zzjZsMOr1SxhBgZ5+0U2aoY1vMhyIjUuwpKKNqj2uf+uj
5Y
ZQbCNklghf7EVDHsYQ4goB9gsNT7rnmrzSc6UUuJOYI2jjtHp5BPMBHh2WtUVf
YP
8JqDfQ+eJQr5NCFB24xMW8OxMJit3MGckUbcZlUa1wKiTb0b76fOjt0y/+9u1y
kd
X+i27DAM6PniFG8BfqPq/E3iU20IZGYtaAFBuhhDWR3vGY4+r3OxdlFAJfBG9X
DD
aEDTzv1XF+tEBo69GFaxXZGdk9//7qxcgiya4LL9Kltuvs82+ZzQhC09p8d3YS
QN
cfaYObm4EwbINdKP7cr4anGFXvsLC9urhow/RNBLiMbRX/5qBzx2DayXtxEnDl
```