pychat Documentation

Release v2.0.0

bensoer & kbohlen

Mar 11, 2017

Contents

PyChat is a UDP based terminal chatting program that allows multiple devices to communicate together over a secure connection using an algorithm of their choice. The purpose of the project is to explore encryption algorithms and sending data back and forth securely and minimizing options for external sources to decrypt the data. The project is maintained by Ben Soer and Kurtis Bohlen.

CHAPTER 1

Installation

Prerequisites

You must have python 3.4 installed on your system. On linux this is available on most distros as *python3*. You can check by typing *python3 –version* into your linux bash

You will need to have pip installed aswell in order to download the *pycrypto* library needed for some of the algorithms

Installation

For the rest of the app:

1. Download the latest release from https://github.com/bensoer/pychat/releases
2. Execute `sudo pip3 install -r requirements.txt` to install dependencies

Quickstart

Start the program by calling

```
python3 pychat.py -h localhost -p 7000 -lp 8000 -u bert -a CaesarCipher
```

This will start PyChat on *localhost* calling another user on port *7000* and listening for responses on *8000*. The converstion will be encrypted with a *CaesarCipher*. Your username for the other user will appear in this example as *bert*. Note that the CaesarCipher has additional optional parameters. Since we did not use them, default CaesarCipher setings were used. For details on these optional parameters to this example, see the *CaesarCipher* section in the *Available Encryption/Decryption Algorithms* section of the readme.

See the *Parameters* section for all common valid parameters

CHAPTER 2

Usage

The documentation for usage has been spread out into two sections. Below lists all global information that can be configured regardless of the algorithm chosen with pychat. Depending on the selected algorithm, additional parameters may be required ontop of those listed below. For algorithm specific parameters see the *Algorithms* section

For quickstart documentation see the bottom of the *Installation* section

Parameters

Below lists all of the global parameters that can be used with any algorithm.

Parameter	Description
-h	Set the host pychat will be communicating with
-p	Set the port pychat will be communicating with
-lp	Set the port pychat will listen for incoming messages from
-u	Set the username for this user. Default is a random number
-a	Set the encryption / decryption algorithm used to secure messages in transit
-v	Set optionally the verification algorithm to be used on each message

Flags

On top of the global parameters there are also flags that can be set. These will enable certain modes of pychat and change how it will operate

Flag	Description
—DEBUG	Enable Debug Mode. Debug logging will be written to console

Console Execution

Running pychat uses the same setup for all end systems wanting to communicate through pychat. Starting a pychat client is initiated in the console by simply running pychat with the appropriate parameters above along with any required parameters with the selected algorithm in the following format:

```
python3 ./pychat.py -h <host> -lp <listeningport> -p <connectionport> [-u <username>]␣
↪-a <algorithm> [-v <verificationalgorithm>] [<FLAGS>]
```

Note that by executing the application with no parameters will print a simple help information to the console and terminate

CHAPTER 3

Algorithms

Below lists all available algorithms on pychat including additional parameters and usages

CaesarCipher

Parameters listed below are **optional**

Parameter	Description
-o	Set the offset value of how many letters down in the caesarcipher translation the algorithm should go

Example

Set the offset value to 13 (Commonly known as ROTL13 encryption)

```
python3 pychat.py -h localhost -p 7000 -lp 8000 -u bert -a CaesarCipher -o 13
```

RandomCaesarCipher

Parameters listed below are **optional**

Parameter	Description
-s	Set the seed value for the rand. Used for generating the scrambled alphabet
-o	Set the offset value of how many letters down in the caesarcipher translation the algorithm should go

Example

Set the offset value to 5 and a randomization seed to 20

```
python3 pychat.py -h localhost -p 7000 -lp 8000 -u bert -a RandomCaesarCipher -o 5 -s␣
↪20
```

TranspositionCipher

Parameters listed below are **required**

Parameter	Description
-k	Set the key used for generating the transposition table

Example

Set the key value to 'spaghetti'

```
python3 pychat.py -h localhost -p 7000 -lp 8000 -u bert -a TranspositionCipher -k␣
↪spaghetti
```

VigenereCipher

Parameters listed below are **required**

Parameter	Description
-k	Set the key word used for encryption

Example

```
python3 pychat.py -h localhost -p 7000 -lp 8000 -u bert -a VigenerCipher -k spaghetti
```

BeaufortCipher

Parameters listed below are **required**

Parameter	Description
-k	Set the key word used for encryption

Example

```
python3 pychat.py -h localhost -p 7000 -lp 8000 -u bert -a BeaufortCipher -k spaghetti
```

AESCipher

Parameters listed below are **required**

This does not do a secure Diffie-Hellman Key Exchange of a randomly generated AES key, it uses SHA256 to hash the password passed by the user

Parameter	Description
-k	Set the password which is then turned into the 256bit key for encryption

Example

Set the key password to 'spaghetti'

```
python3 pychat.py -h localhost -p 7000 -lp 8000 -u bert -a AESCipher -k spaghetti
```

PureAESCipher

PureAESCipher differs from AESCipher in being a pure implementation written by Kurtis Bohlen. Use of the *pycrypto* library is not incorporated in this implementation

Parameters listed below are **required**

Parameter	Description
-k	Set the key for encryption (must be 16,24, or 32 bytes long)
-m	Set the block cipher mode of operation (default is CBC)

Example

Set the key to 'asixteenbyteword' and the mode to 'CBC' (default mode)

```
python3 pychat.py -h localhost -p 7000 -lp 8000 -u bert -a PureAESCipher -k␣
↪asixteenbyteword -m CBC
```

DESCipher

Parameters listed below are **required**

Parameter	Description
-k	Set the key for encryption. It must be 16 characters long. Both users must enter the same key

Example

Set the key password to 'spaghetti'

```
python3 pychat.py -h localhost -p 7000 -lp 8000 -u bert -a DESCipher -k spaghetti
```

PureRSA

PureRSA differs from a normal RSA implementation as it is a pure implementation written by Ben Soer. The *pycrypto* library is not incorporated in the implementation of this algorithm

Parameters listed below are **required**

Parameter	Description
-p1	Set first prime number for generating the public/private key pair
-p2	Set second prime number for generating the public/private key pair

Example

Set the prime numbers to 5 and 11

```
python3 pychat.py -h localhost -p 7000 -lp 8000 -u bert -a PureRSA -p1 5 -p2 11
```

CHAPTER 4

Verification Algorithms

Verification Algorithms are hashing algorithms that are generated and sent with every message from pychat. In order to use verification algorithms, the `-v` flag must be set on all clients along with the same verification algorithm being selected on each system. When each message is sent, a hash will be generated and appended to the beginning of each message. Upon receival, pychat will decrypt and verify the message against the hashing algorithm. If the hashing fails a warning message will be printed to console before the original message is printed. Regardless of whether the hash is valid or not, the original message will always be printed.

Below lists all options that are currently implemented in pychat

MD2

Verify message integrity using MD2 hashing algorithm

Example

```
python3 pychat.py -h localhost -p 7000 -lp 8000 -u bert -a CaesarCipher -v MD2
```

MD4

Verify message integrity using MD4 hashing algorithm

Example

```
python3 pychat.py -h localhost -p 7000 -lp 8000 -u bert -a CaesarCipher -v MD4
```

MD5

Verify message integrity using MD5 hashing algorithm

Example

```
python3 pychat.py -h localhost -p 7000 -lp 8000 -u bert -a CaesarCipher -v MD5
```

SHA1

Verify message integrity using SHA1 hashing algorithm

Example

```
python3 pychat.py -h localhost -p 7000 -lp 8000 -u bert -a CaesarCipher -v SHA1
```

SHA224

Verify message integrity using SHA224 hashing algorithm

Example

```
python3 pychat.py -h localhost -p 7000 -lp 8000 -u bert -a CaesarCipher -v SHA224
```

SHA256

Verify message integrity using SHA256 hashing algorithm

Example

```
python3 pychat.py -h localhost -p 7000 -lp 8000 -u bert -a CaesarCipher -v SHA256
```

SHA384

Verify message integrity using SHA384 hashing algorithm

Example

```
python3 pychat.py -h localhost -p 7000 -lp 8000 -u bert -a CaesarCipher -v SHA384
```

SHA512

Verify message integrity using SHA512 hashing algorithm

Example

```
python3 pychat.py -h localhost -p 7000 -lp 8000 -u bert -a CaesarCipher -v SHA512
```