Enigma over IP (EoIP)

Written by Owen Davies

October 22, 2012

TL;DR

Send messages to other networked machines, using a simulated Enigma machine to encrypt the messages. It's like World War II but with computers! Credit for the stupid name goes to Daniel Hertz.

Usage

EoIP is run from the commnad line, using the following command line argument structure:

```
./enigma [*.rot] .pb [-n server|(client [hostname|IP]) port [-v]]
```

- The `-n` flag tells the program that we want to run it with networking enabled, and either create a client or a server;
- The next argument is either `server` or `[hostname|IP]` , which tells the program whether we want to set up a server which will listen for a client or whether we want to set up a client who will look for a server on the machine called `hostname` or with IP address `IP`;
- The `port` argument tells the server which port to listen on, and tells the client which port to look on;
- Finally, you can add the `-v` argument if you want to start the program in verbose mode (which shows diagnostic network messages and more detailed information about the encryption and decryption done by the Enigma machine).

Like in World War II, the server and the client machines must use the same Enigma machine rotor and plugboard configuration in order for the messages recieved to be decrypted correctly. Obivously a server `instance` must be running before a `client` instance.

Implementation Details

The networking aspect of this program is implemented using the C/C++ standard socket networking library, `socket.h`. It uses the `sockaddr_in` structure to configure the server, and then calls the `listen` function to wait for a connection to be made by a client.

Once the server is listening for connections, the client can then be run. The user must give the client the correct hostname or IP of the server, and the port which the server is listening for connections on. Once it has configured the server address settings, it uses the `accept` function from the `socket.h` library to attempt a connection to the server.

Once the server and client are connected they can alternate sending messages to eachother. The message system is implemented in this alternate fashion as this is how messages would have been sent back in the war (definitely NOT because it's quite a lot easier to code!). For example, the following output for a server instance (in verbose mode) of EoIP:

```
./enigma rotors/II.rot rotors/III.rot plugboards/null.pb -n server 12345 -v
Setting up socket...
Configuring server address settings...
Waiting for connection...
Accepting connection...
Connected to 127.0.0.1
Reading encrypted message:
LSHCV
Unencrypting...
127.0.0.1> HELLO
Write message:
> HELLO
Encrypting message as:
VPCHY
Sending message to client...
Message sent.
```

Is mirrored by the following output from a client instance of enigma (also in verbose mode):

```
./engima rotors/II.rot rotors/III.rot plugboards/null.pb -n client localhost 12345 -v Setting
up socket...
Configuring server address settings...
Attemping to connect...
Connected to 127.0.0.1
Write message:
> HELLO
Encrypting message as:
LSHCV
Sending message to client...
Message sent.
Reading encrypted message:
VPCHY
Unencrypting...
127.0.0.1> HELLO
```

This exmaple shows how the server and the client are run with the same enimga configuration settings, and are run on the same port. Here I use the loop back IP to talk between two instances of engima on the same machine.