Zombie Networks via Internet Relay Chat

/braaaiiinnnsss

IRC - Introduction & Basic Terms

Internet Relay Chat (IRC) is an application layer protocol for textual communication. Using a client/server model, it allows for group discussion in forums called channels.

Any user may host a channel, allowing others to join and discuss topics of interest. Many channels have a specific purpose, but some are used as “hang out” spots.

An IRC Bot is a script/program that makes a TCP connection to an IRC server and is controlled from within the channels of IRC by the users. It offers functionality to the people in the channels, most often using APIs for different services (such as Wikipedia, translation software, calculation websites, etc.).

IRC servers aren’t just the resting place of the zombie hordes, but for this lesson we’ll pretend they are.

IRC - Clients & Servers

IRC Client options include:
Irssi - This is what we'll be using! CLI based client for Unix systems.
Chatzilla - Plugin client for Mozilla-based browsers such as Firefox.
Colloquy - Using its own "Chat Core" engine, an open-source client for Mac OS X
mIRC - Popular client for Windows, has an integrated scripting language
Konversation - Built on the KDE platform, one of the popular clients for Linux distros

There are plenty of IRC Servers, but the two most popular are:
Freenode - 74,841 average users, has steadily become the most populated network
QuakeNet - 24,627 average users, held the record of 240,000+ users in 2005

IRC Historic Events - Constitutional Crisis of ‘93

IRC users in Moscow were able to pass info before the major news reporting agencies could broadcast it:

```
<slipper> cnn intl just now confirming report here 5 mins ago that Russ tv off line!
...
<Bravo> Around 16:00 (sorry don't have exact times) group of people around 3-4 thousand
started to move in the direction of Moscow municipal building
...
<Bravo> Currently, first 5 floors of city hall are taken…
...
<geek> Moscow radio on shortwave…
<ginster> i have a sw radio - what is the frequency?
<Bravo> … they have taken the Ostankino Tower, so it is not talking anymore
```

Zombies - Plug & Play

The following files need to remain unmodified for the zombie to operate correctly.

- `bot_connect.py`
- initializes the zombie's TCP connection and handles the data-to-parser loop
-
- `bot_core.py`
- stores the brains of the zombie and handles module organization
-
- `bot_parser.py`
- parses all data received by the zombie and handles any data received

Zombies - Plug & Tinker For A Minute Or Two, Then Play

These files may be modified so that you may better control the zombie.

- `bot_data.py`
- stores the static variables so the zombie knows where to go and whom to obey
-
- `bot_commands.py`
- houses the functions that a zombie's owner has access to

Code Walkthrough - bot_commands.py

```
import commands

command_dictionary = {
    "join":{"code":"bot_core.bot_commands.join_channel(bot_core);" },
    "part":{"code":"bot_core.bot_commands.part_channel(bot_core);" },
    "quit":{"code":"bot_core.bot_commands.quit_server(bot_core);" },
    "debug":{"code":"bot_core.bot_commands.debug_variable(bot_core);" },
    "ping":{"code":"bot_core.bot_commands.ping_server(bot_core);" }
    };

def join_channel(bot_core):
    channel = bot_core.bot_data.command_info[ "args"][0];
    bot_core.send_raw( "JOIN {0}".format(channel));

def quit_server(bot_core):
    bot_core.send_raw( "QUIT :Local kill");
    bot_core.socket_connection.close();
    quit();
```

Code Walkthrough - bot_commands.py

```
def ping_server(bot_core):
    target_server  = bot_core.bot_data.command_info[ "args"][0];
    ping_allowed  = True;

    if len(target_server)  <= 15:
        try:
            for item in target_server.split( "."): item = int(item);
        except: ping_allowed  = False;
    else: ping_allowed  = False;

    if ping_allowed:
        bot_core.send_message( "Sending ten pings, give me around 20 seconds to process."  );
        ping_output  = commands.getoutput( "ping -c 10 {0}" .format(target_server)).split( "\n");
        for item in ping_output:
            item_found  = False;
            if "transmitted"  in item and item_found  != True:
                item_found  = True;
                bot_core.send_message( "Here you go: {0} | {1}" .format(ping_output[ 0], item));
    else: bot_core.send_message( "Sorry, this command is pretty strict. Make sure your IP is IPv4."  );
```

Code Walkthrough - bot_data.py

```
from platform import node, platform, version;

machine_info = {
                    "node":node(),
                    "platform":platform(),
                    "version":version()
                };

BUFFER = [""]; irc_data = {"raw":""}; command_info = {"name":"", "args":[]};
message_info = {"message":"", "length":0, "sender":{"name":"", "respond":""}};
server_info = {"address":"irc.university.edu", "channel":"#cis76", "port":6667};

bot_name = "PodXXBot"; command_symbol = "!";
auth_users =["xxxxxx76", "zombie_lord"];
```

Code Walkthrough - bot_connect.py

```
import bot_parser; import bot_core; import bot_data; import bot_commands;

connection_core = bot_core.bot_core(bot_parser, bot_commands, bot_data);
connection_core.send_raw( "JOIN {0}".format(connection_core.bot_data.server_info[ "channel"]));

while True:
    connection_core.bot_data.BUFFER = connection_core.socket_connection.recv( 1024).split("\r\n");
    if connection_core.bot_data.BUFFER != [""]:
        connection_core.bot_parser.filter_errors(connection_core);
```

Code Walkthrough - bot_core.py

```
import socket; import time;
import bot_parser; import bot_commands; import bot_data;

def bot_core(bot_parser, bot_commands, bot_data):
    class bot():
        def __init__(self):
            self.socket_connection = socket.socket(socket.AF_INET, socket.SOCK_STREAM);
            self.bot_data = bot_data; self.bot_commands = bot_commands; self.bot_parser = bot_parser;

            try:
                self.socket_connection.connect((self.bot_data.server_info["address"], self.bot_data.server_info["port"]));
            except socket.error, e:
                print("I failed to connect to the server you provided." );
                quit();

            time.sleep( 1); self.send_raw("NICK {0}".format(self.bot_data.bot_name));
            time.sleep( 1); self.send_raw("USER EH-Zombie 8 * :EHZombie" );
            time.sleep( 1); self.send_raw("MODE {0} +B".format(self.bot_data.bot_name));
            print("Sent my identity to the IRC server." );
```

Code Walkthrough - bot_core.py

```
    def module_rehash(self):
        module = self.bot_data.command_info["args"][0];
        sender = self.bot_data.message_info["sender"]["respond"];
        exec("reload({0});".format(module)) in globals();
        self.send_message("I reloaded {0}.".format(module), sender);

    def send_raw(self, message):
        self.socket_connection.send("{0}\r\n".format(message));

    def send_message(self, message, response=""):
        if response == "": response = self.bot_data.message_info["sender"]["respond"];
        self.socket_connection.send("PRIVMSG {0} :{1}\r\n".format(response, message));
        print("I just send the the message '{0}' to {1}." );

botcore = bot();
return botcore;
```

Code Walkthrough - bot_parser.py

```
from codecs import decode
def filter_errors(bot_core):

    try:
        parse_data(bot_core);
    except:
        error_data  = traceback.format_exc().split( "\n");
        error_data  = error_data[::-1];
        bot_core.send_message( "I just caught an error. Printing data locally." ); print(error_data);
```

Code Walkthrough - bot_parser.py

```
def assign_data(bot_core):
    irc_data  = bot_core.bot_data.irc_data[ "raw"];
    message_info  = {"message":"", "length":0, "sender":{"name":"", "respond":"", "real":""}};
    command_info  = {"name":"", "args":[]};

    message_info[ "message"] = " ".join(irc_data[ 3:])[1:];
    message_info[ "length"] = len(message_info[ "message"]);

    if len(irc_data[ 3:]) >= 1:
        if irc_data[ 3][1:][0] == bot_core.bot_data.command_symbol:
            command_info[ "name"] = irc_data[ 3][2:]; command_info[ "args"] = irc_data[ 4:];

    message_info[ "sender"][ "name"] = irc_data[ 0][1:].split("!")[0];
    message_info[ "sender"][ "real"] = irc_data[ 0][1:].split("!")[1].split("@")[0];

    if irc_data[ 2][0] == "#": message_info[ "sender"][ "respond"] = irc_data[ 2];
    elif irc_data[ 2] == bot_core.bot_data.bot_name:
        message_info[ "sender"][ "respond"] = message_info[ "sender"][ "name"];

    bot_core.bot_data.message_info  = message_info;
    bot_core.bot_data.command_info  = command_info;
```

Code Walkthrough - bot_parser.py

```
def parse_data(bot_core):

    for item in bot_core.bot_data.BUFFER:
        bot_core.bot_data.irc_data[ "raw"] = item.split();
        if len(bot_core.bot_data.irc_data[ "raw"]) == 2:
            if bot_core.bot_data.irc_data[ "raw"][0] == "PING":
                bot_core.send_raw( "PONG {0}".format(bot_core.bot_data.irc_data[ "raw"][1]));

        elif len(bot_core.bot_data.irc_data[ "raw"]) >= 3:
            if search(":.+!.+@.+", bot_core.bot_data.irc_data[ "raw"][0]):
                if len(bot_core.bot_data.irc_data[ "raw"]) >= 4:
                    if bot_core.bot_data.irc_data[ "raw"][1] == "PRIVMSG":
                        assign_data(bot_core);
                        print("{0}".format(" ".join(bot_core.bot_data.irc_data[ "raw"])));
```

EOF