Analysis Report for: BackdoorMalware.c

Overall Functionality

This C code implements a malicious program that connects to a remote server, executes commands received from the server, and attempts to self-delete. It uses several layers of obfuscation including custom mathematical functions and network communication to hide its true purpose. The program appears to be a backdoor or a component of a more complex malware system.

Function Summaries

* **`sub_401000`, `sub_401040`, `sub_401080`**: These functions perform bitwise operations on input integers (`a2`), influenced by a control integer (`a1`). They seem to be custom obfuscation routines, modifying the input values in a non-obvious way.

* **`sub_4010C0`**: This function takes an integer pointer (`a1`) and a character array (`a2`) as input. It performs byte-level manipulations on `a2` to generate three integers that are stored in the memory locations pointed to by `a1`. It then returns a single byte from `a2`. This function seems designed to obfuscate data transformation.

* **`sub_401130`**: This function uses the previously described bitwise manipulation functions (`sub_401000`, `sub_401040`, `sub_401080`) and `sub_401190` to modify an array of three unsigned integers (`a1`). The modifications are complex and appear to be designed for obfuscation. It returns 0 or 1.

* **`sub_401190`**: This function checks if at least two out of three input values have their high-order bit set. It returns `TRUE` if at most one value has its high-order bit set, and `FALSE` otherwise. This is likely used as a control mechanism in other functions.

* **`sub_4011E0`, `sub_401220`**: These functions iterate through a memory block, applying the `sub_401130` function to modify the data. The difference seems only stylistic, with `sub_401220` simply calling `sub_4011E0`.

* **`WinMain`**: The entry point of the program. It loads a string (likely a configuration string), calls `sub_4012C0`, and exits.

* **`sub_401270`**: Sends data (`buf`) over a socket (`s`). It handles sending in chunks and returns the number of bytes sent or -1 on error.

* **`sub_4012C0`**: Creates two event handles (`hObject`, `hEvent`), initializes a data structure (`unk_4030E0`), calls `sub_401320`, waits on `hObject`, and closes `hObject`. Its role is setting up the main program functions.

* **`sub_401320`**: This function is the core network communication and command execution component. It parses a configuration string, connects to a remote server, receives and executes commands. It uses threads and pipes to manage this process.

* **`sub_401600`**: Creates a pipe for communication between threads. It returns a handle to the pipe or NULL if an error occurs.

* **`StartAddress`**: Creates two threads (`sub_401940`, `sub_401A70`) and a process using `sub_401860`, managing their execution via a pipe, and then cleans up.

* **`sub_401860`**: Creates a new process (`cmd.exe`), redirecting its input/output to pipes. This is likely the mechanism for running commands on the victim machine.

* **`sub_401940`**: A thread function that reads data from a named pipe, processes it with `sub_4010C0` and `sub_4011E0` and sends it to the remote server.

* **`sub_401A70`**: A thread function that receives data from the remote server, processes it with `sub_4010C0` and `sub_401220`, and writes it to a named pipe.

* **`sub_401B50`**: Attempts to delete itself using `cmd.exe` and `del`.

* **`UserMathErrorFunction`**: A dummy function, returning 0. Likely a placeholder or remnant of development.

Control Flow

The program's control flow is complex due to the extensive obfuscation. The primary flow is:
1. `WinMain` initializes and calls `sub_4012C0`.
2. `sub_4012C0` sets up event handles and calls `sub_401320`.
3. `sub_401320` handles network connection and command execution. This function's internal logic involves parsing a configuration string, connecting to a server specified in that string, receiving commands, and executing those commands within the created process. A key conditional branch determines what happens depending on the commands received.
4. The `StartAddress` function manages the threads and child processes.
5. Threads (`sub_401940`, `sub_401A70`) handle communication and data transformation through pipes.
6. `sub_401B50` attempts self-deletion.

Data Structures

The code uses several important data structures:

```
* **`unk_4030E0`**: A large, uninitialized structure of unknown type. Its purpose is not clear, but it is likely related to the overall obfuscation strategy.
* **Pipes**: Used for inter-process and inter-thread communication. This helps the malware hide its activity.
* **Sockets**: Used for communication with the command and control (C&C) server.


**Malware Family Suggestion**

Based on its functionality, this code strongly resembles a **remote access trojan (RAT)** or a backdoor. Its ability to connect to a remote server, receive and execute commands, and its self-deletion attempt are clear indicators of malicious intent. The heavy obfuscation is a common characteristic of sophisticated malware designed to evade detection. The use of multiple threads and pipes further complicates analysis and makes removal more difficult. More specific classification would require further analysis and identifying the C&C server and the commands executed.
```