Analysis Report for: BackdoorMalware.c

Overall Functionality

This C code exhibits characteristics strongly suggestive of malware, specifically a backdoor or remote access trojan (RAT). It establishes a network connection to a remote server, exfiltrates information about the infected system (computer name), and executes commands received from the server. The code heavily obfuscates its functionality through custom mathematical functions (`sub_401000`, `sub_401040`, `sub_401080`, `sub_4010C0`, `sub_401130`), cryptic variable names, and the use of pipes for inter-process communication. The self-deletion routine at the end further confirms malicious intent.

Function Summaries

* **`sub_401000`, `sub_401040`, `sub_401080`**: These functions appear to be custom bitwise manipulation routines used for obfuscation. They take an integer and an unsigned integer as input and return a modified unsigned integer. The exact purpose is unclear without reverse engineering the bitwise operations.

* **`sub_4010C0`**: This function transforms an 8-byte input (likely from network data) into a 12-byte output, further suggesting obfuscation or encryption/decryption.

* **`sub_401130`**: This function uses the previously mentioned functions to perform a series of bitwise operations on a 3-element unsigned integer array, adding another layer of obfuscation.

* **`sub_401190`**: A boolean function that checks bit flags within its three integer parameters, seemingly for some kind of conditional branching within the obfuscation process.

* **`sub_4011E0`, `sub_401220`**: These functions perform a loop-based XOR operation on a buffer using the result from `sub_401130`. This likely applies a key-based transformation to the data.

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

* **`sub_401270`**: A function to send data over a socket, handling potential partial sends.

* **`sub_4012C0`**: Initializes events and calls `sub_401320`, the core network connection and command execution logic.

* **`sub_401320`**: This is the heart of the malware. It parses a command string (potentially command and control IP/port from the loaded string), connects to the remote server, exchanges data, and starts secondary threads. It uses `beginthread()` to create threads for communication and command execution. Failure to connect results in a short sleep before retrying.

* **`sub_401600`**: Creates two pipes for inter-process communication, returns a structure containing handles for them and a process handle.

* **`StartAddress`**: Creates two threads (`sub_401940` and `sub_401A70`) and a process using `sub_401860`, manages these threads and the process using `WaitForMultipleObjects`, and performs cleanup after the threads and the process are finished.

* **`sub_401860`**: Creates a process (`cmd.exe`) with redirected standard input and output/error pipes.

* **`sub_401940`**: Reads data from a named pipe, performs obfuscation, and sends it to the remote server using `sub_401270`.

* **`sub_401A70`**: Receives data from the remote server, performs reverse obfuscation, and writes the data to a named pipe. Exits when receiving the 'exit' command.

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

* **`UserMathErrorFunction`**: A dummy function that always returns 0.

Control Flow

The control flow is complex due to obfuscation. `sub_401320` is crucial, containing nested loops and conditional statements based on network communication and command parsing. Error handling is minimal, often resulting in sleep calls instead of robust error management. The threading model using `WaitForMultipleObjects` allows for concurrent communication and command execution.

Data Structures

The most significant data structures are:

* **`unsigned int` arrays**: Used in obfuscation functions to represent data.
* **`HANDLE` arrays**: Used to manage thread and process handles in `StartAddress`.
* **Pipes**: Used for communication between the main process and the spawned `cmd.exe` process. This is a common technique used in malware to avoid direct process interaction.
* **Sockets**: Used for network communication with the command and control server.

Malware Family Suggestion

Based on its functionality—establishing a network connection, receiving and executing commands, exfiltrating system information, and attempting self-deletion—this code is highly indicative of a **Remote Access Trojan (RAT)**. The sophisticated obfuscation suggests an attempt to evade detection. Specific categorization into a known RAT family would require further reverse engineering and analysis comparing its behavior with known malware signatures. However, the overall structure and techniques strongly point to a RAT.