Analysis Report for: BackdoorMalware (2).c

Overall Functionality

This C code is a malicious program designed to connect to a remote server, receive commands, and execute them on the infected system. It uses several obfuscation techniques to hinder analysis. The program establishes a network connection, likely using a hardcoded IP address and port derived from the decoded `Str` variable, and then acts as a backdoor, receiving and executing commands from the attacker. Self-deletion is attempted at the end through `sub_401B50`. The core functionality is hidden behind several layers of seemingly unrelated mathematical operations and function calls.

Function Summaries

* **`sub_401000`, `sub_401040`, `sub_401080`**: These functions perform bitwise operations on input integers, likely serving as custom obfuscation routines. They modify the input based on a conditional check and bit shifts/XORs. The purpose is to obscure the true mathematical relationship

* **`sub_4010C0`**: This function takes a character array (`a2`) and an integer pointer (`a1`) as input. It interprets the bytes of the input character array and stores transformed values into the integer array pointed to by `a1`. This likely involves a custom encryption or encoding scheme.

* **`sub_401130`**: This function uses the three preceding functions (`sub_401000`, `sub_401040`, `sub_401080`) and `sub_401190` to manipulate an array of unsigned integers. The result is a single bit, which appears to act as some kind of randomized output.

* **`sub_401190`**: This function checks if at least two of three input values have their high-order bits set. The boolean result controls the branchings in other functions. This is likely a simple integrity check or a way to inject randomness.

* **`sub_4011E0`, `sub_401220`**: These functions iterate through a buffer, XORing bytes with a value derived from `sub_401130`. These functions are likely used for some form of encryption or modification of data. `sub_401220` appears to be a simple wrapper for `sub_4011E0`.

* **`WinMain`**: The entry point of the program. It loads a string (likely an encoded IP address and port), calls `sub_4012C0`, and exits.

* **`sub_401270`**: This function sends data over a socket, handling potential partial sends.

* **`sub_4012C0`**: This function initializes events (`hObject`, `hEvent`) and calls `sub_401320` before waiting on an event and closing a handle. This is likely related to the main thread synchronization.

* **`sub_401320`**: The core network communication and command execution function. It decodes the command and control (C&C) server address from `Str`, establishes a socket connection, sends system information to the C&C server, and receives and executes commands. It uses `beginthread` to spawn a separate thread, indicating multi-threaded functionality.

* **`sub_401600`**: Creates two pipes and returns a handle structure to manage inter-process communication (IPC) for the spawned threads.

* **`StartAddress`**: This function sets up and manages the threads created by `sub_401320`. It creates two threads (`sub_401940` and `sub_401A70`) and waits for their completion, handling various termination scenarios. It also handles cleanup of resources, including pipes, threads, and sockets.

* **`sub_401860`**: Creates a `cmd.exe` process using the pipes setup by `sub_401600` for input/output redirection.

* **`sub_401940`**: A thread function that reads data from a named pipe, decrypts it (using `sub_4010C0` and `sub_4011E0`), and sends it to the C&C server.

* **`sub_401A70`**: A thread function that receives data from the C&C server, decrypts it, and writes it to a named pipe. It also checks for 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 and involves several conditional checks during network communication and command processing. `StartAddress` manages the multi-threaded operation, waiting for thread completion and handling various exit conditions. The use of `beginthread` creates parallel execution paths. Error handling is minimal, and failures often result in simply exiting or calling `sub_401B50`.

Data Structures

* The most important data structure is the structure of handles returned by `sub_401600`. It manages handles for pipes and threads, crucial for coordinating the processes.
* Other structures used are fairly standard Windows API structures such as `sockaddr`, `WSAData`, `STARTUPINFOA`, `PROCESS_INFORMATION`, and `SHELLEXECUTEINFOA`.

Malware Family Suggestion

Based on its functionality, this code strongly resembles a **remote access trojan (RAT)**. Key characteristics supporting this classification include:

* **Network Connection:** Establishes a network connection to a remote server.
* **Command Execution:** Receives and executes commands from the remote server.
* **Persistence:** Attempts self-deletion and uses multithreading for resilience.
* **Obfuscation:** Uses multiple layers of obfuscation to hinder reverse engineering.
* **Process Creation:** Uses CreateProcessA to execute cmd.exe.
* **Pipe Creation:** uses pipes for interprocess communication with threads

The specific type of RAT may be difficult to determine without more information (e.g., the C&C server's behavior), but this analysis shows strong RAT attributes. The self-deletion attempt is also a common feature in many malware families.