Analysis Report for: BackdoorMalware.c

Overall Functionality

This C code exhibits strong characteristics of a malicious program, likely 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 uses obfuscation techniques (complex bitwise operations, unusual function names) to hinder reverse engineering. The use of threads suggests concurrent activities, potentially for increased stealth or to perform multiple tasks simultaneously. The self-deletion attempt at the end further points towards malicious intent.

Function Summaries

* **`sub_401000`, `sub_401040`, `sub_401080`**: These functions appear to be custom encryption/obfuscation routines. They perform bitwise operations on input integers (`a2`) based on a conditional check involving another integer (`a1`). The return value is a modified version of `a2`.

* **`sub_4010C0`**: This function interprets a byte array (`a2`) as a custom data structure and writes it into an integer array (`a1`). It seems to be a data packing/unpacking function.

* **`sub_401130`**: This function combines the results of `sub_401000`, `sub_401040`, and `sub_401080`, possibly as part of a more complex encryption or obfuscation scheme. It uses `sub_401190` for conditional logic.

* **`sub_401190`**: This function performs a check on the high-order bits of its integer parameters, possibly related to a data integrity or error-checking mechanism within the encryption scheme.

* **`sub_4011E0`, `sub_401220`**: These functions appear to perform XOR operations on a buffer, potentially as part of the encryption/decryption process.

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

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

* **`sub_4012C0`**: This function initializes events (`hObject`, `hEvent`) and calls `sub_401320`. It seems responsible for setting up the malware's infrastructure.

* **`sub_401320`**: The core network communication function. It extracts IP address and port from a string, establishes a socket connection, sends the computer name to the remote server, and handles command execution. It uses threads (`beginthread`) to handle communication concurrently.

* **`sub_401600`**: Creates pipes for inter-process communication. Returns a handle structure.

* **`StartAddress`**: This function appears to be the main thread function. It utilizes the pipes established by `sub_401600` to manage a child process and handles thread synchronization and termination.

* **`sub_401860`**: Creates a new process (`cmd.exe`) and redirects its standard input and output using the provided pipes.

* **`sub_401940`, `sub_401A70`**: These are the thread functions, which run concurrently. `sub_401940` reads data from a pipe, decrypts it, and sends it to the remote server. `sub_401A70` receives data from the remote server, decrypts it, and writes it to a pipe.

* **`sub_401B50`**: Attempts to delete the malware's own executable file using `cmd.exe` and `del`.

* **`UserMathErrorFunction`**: A dummy function, likely a placeholder or remnant from the code's development.

Control Flow

The control flow is complex due to obfuscation. However, the main flow is as follows:

1. `WinMain` initializes and calls `sub_4012C0`.
2. `sub_4012C0` sets up events and calls `sub_401320`.
3. `sub_401320` establishes a network connection, sends the computer name to the remote server, and receives commands. The command processing is done within threads launched by `beginthread`.
4. `sub_401940` and `sub_401A70` handle the communication, involving encryption/decryption using custom functions.
5. `StartAddress` creates and manages the child process and threads, handling synchronization and cleanup.
6. Upon command execution completion or an exit command, `sub_401B50` attempts to delete the malware executable.

Data Structures

* **Custom Data Structure (in `sub_4010C0`, `sub_401130`):** A data structure is implied by the byte array processing in `sub_4010C0` and how the integer array is used in related functions. Its exact format isn't explicitly defined but is heavily reliant on bit shifting and masking operations.

* **Socket Handles and Pipe Handles:** The code uses Windows handles for sockets and pipes for inter-process communication.

* **`unk_4030E0`:** This is an unknown data structure, possibly padding or an incomplete structure.

Malware Family Suggestion

Based on its functionality – network communication, command execution, data encryption/decryption, self-deletion – this code strongly suggests a **RAT (Remote Access Trojan)**. The obfuscation and use of threads point towards a more sophisticated and potentially dangerous variant. The detailed analysis points toward a custom-made RAT rather than belonging to a known family, as the encryption and code structure are not easily mapped to existing RATs. Further analysis would be needed to determine if any code similarities to existing malware exist.