Analysis Report for: BackdoorMalware.c

Overall Functionality

This C code appears to be a backdoor or malware component designed to establish a connection to a remote server, execute commands received from the server, and self-destruct. It uses network sockets to communicate, employs obfuscation techniques (bitwise operations, custom math functions), and interacts with Windows API functions for process creation, thread management, and file manipulation. The code's main purpose is to provide a remote attacker with control of the infected machine.

Function Summaries

* **`sub_401000`, `sub_401040`, `sub_401080`**: These functions implement custom, obfuscated mathematical operations. They take an integer and an unsigned integer as input and return a modified unsigned integer. The specific calculations are non-trivial but seem to involve bitwise manipulations and conditional logic, making reverse engineering more difficult.

* **`sub_4010C0`**: This function takes an integer pointer and a character array as input. It appears to perform a custom encoding or decoding operation, transforming the input character array into three integers stored at the address pointed to by `a1`. The return value is the last byte of the input character array.

* **`sub_401130`**: This function manipulates an array of three unsigned integers using the obfuscated mathematical functions (`sub_401000`, `sub_401040`, `sub_401080`) and the boolean function `sub_401190`. It likely implements a custom encryption or obfuscation algorithm. The return value is a single bit (0 or 1).

* **`sub_401190`**: A relatively simple boolean function that checks bit flags in its three input parameters. Returns `TRUE` or `FALSE` based on the number of bits set.

* **`sub_4011E0`, `sub_401220`**: These functions perform the core encryption/decryption routines using the `sub_401130` function. They iterate through a buffer, modifying its bytes based on the result of the custom algorithm.

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

* **`sub_401270`**: Sends data over a socket. It handles partial sends and returns the total number of bytes sent or -1 on error.

* **`sub_4012C0`**: Creates events (`hObject` and `hEvent`) and initializes some data. It waits on `hObject` and then closes it. The purpose is unclear without further analysis but likely related to synchronization or signaling within the malware.

* **`sub_401320`**: The main network communication function. It parses a command string, connects to a remote host specified in the parsed string, and exchanges data with it. It creates a new thread (`StartAddress`) based on the received data and terminates itself. This function represents a key component of the backdoor's functionality.

* **`sub_401600`**: Creates a pipe and associated process. It returns a pointer to a structure containing handles to the created pipe and process.

* **`StartAddress`**: Creates two threads (`sub_401940` and `sub_401A70`), waits for their completion, cleans up resources, and terminates associated processes.

* **`sub_401860`**: Creates a new process (`cmd.exe`) using the provided pipe handles for stdin, stdout and stderr redirection. It appears to be used to execute commands.

* **`sub_401940`**: Reads data from a named pipe, encrypts it using the custom routines, and sends it over a network socket. This is the thread responsible for sending data to the remote server.

* **`sub_401A70`**: Receives data from a network socket, decrypts it using the custom routines, and writes it to a named pipe. This thread receives commands from the server.

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

* **`UserMathErrorFunction`**: A simple function that always returns 0. Likely a placeholder or leftover from development.

Control Flow

The main control flow is driven by the `WinMain` function, which triggers the network communication and command execution via `sub_401320`. The functions `sub_401940` and `sub_401A70` represent infinite loops until the connection is closed or a specific command is received. Conditional logic within `sub_401320` determines the success or failure of connection establishment and command execution. Error handling in many functions is minimal (often just returning -1 or 0), and cleanup is handled primarily in `StartAddress` and `sub_401B50`.

Data Structures

The code uses several important data structures:

* **Sockets:** Used for network communication with the remote server.
* **Pipes:** Used for inter-process communication between the main process and the spawned `cmd.exe` process.
* **Handles:** Used extensively for managing processes, threads, and other Windows objects.
* **Arrays/Structures:** Several custom structures are used for data passing and storage, though their definitions are not explicitly present (due to the decompiler). The `byte_4030A1` array may play a role in encryption.

Malware Family Suggestion

Based on its functionality, the code strongly resembles a **remote access trojan (RAT)**. It establishes a persistent connection, receives and executes commands from a remote server, uses obfuscation techniques to hinder analysis, and attempts to self-delete. The use of pipes and processes suggests a sophisticated RAT designed for stealthy operation. It also shares features with downloader malware, as it may download and execute additional malicious code from its C2 server.