Analysis Report for: 3CB0EE7D41DA25DAC3F1A78BB1B54D92.exe.c

Overall Functionality

This C code snippet appears to be the entry point and core logic of a malware program. It parses the command line, extracts arguments, and then calls a function named `NSMClient32` with these arguments. The `NSMClient32` function is likely responsible for the malicious activity of this program. The code carefully handles quoted command-line arguments, suggesting a need to manage spaces or special characters potentially within the arguments passed to it. Ultimately, the program exits using the return value of `NSMClient32`. The use of `GetModuleHandleW` suggests it may need information about its own location in memory. The `weak` declaration for functions suggests possible dynamic linking or runtime resolution of these functions, adding another layer of obfuscation.

Function Summaries

* **`sub_401000(int a1, int a2, int a3, int a4)`:** This function acts as a simple wrapper. It takes four integer arguments but only uses `a3` and `a4`, passing them directly to `NSMClient32`. It returns the result of the `NSMClient32` call.

* **`start()`:** This is the program's entry point. It performs the following actions:
1. Retrieves the command line using `GetCommandLineW`.
2. Parses the command line to extract an argument, handling arguments potentially enclosed in double quotes.
3. Retrieves startup information, using it to determine the `wShowWindow` flag, which defaults to 10 if not specified in the startup flags.
4. Obtains the module handle of the current executable using `GetModuleHandleW`.
5. Calls `sub_401000` (which in turn calls `NSMClient32`) passing it the module handle, 0, the extracted argument, and the `wShowWindow` value.
6. Finally, exits the process using the return value of `sub_401000`.

* **`NSMClient32(_DWORD, _DWORD)`:** This function is not implemented in this snippet; however, its name and context strongly suggest that it is the core malicious function. Based on the name, it may be interacting with a network service or client (possibly a remote server). The parameters may consist of a command and an additional data parameter

Control Flow

* **`start()`:** The `start` function's control flow is largely linear, with the exception of the command-line parsing section. This section contains a loop that iterates through the command line (`while` loop and `for` loop) to extract an argument while handling quoting. The conditional statements (`if`) check for the presence of double quotes and the presence of arguments after the potential quotes. If no argument is found after parsing, the control moves to `LABEL_7`, otherwise the parsed argument will be passed to the subsequent functions.

* **`sub_401000()`:** This function has a trivial control flow; it simply calls `NSMClient32` and returns its result.

Data Structures

* **`_STARTUPINFOW`:** This standard Windows structure holds information about the process's startup, including the `wShowWindow` flag that specifies how the window should be shown (if applicable). This struct is filled by `GetStartupInfoW`.

* **Command Line Argument:** The command line is treated as a string. The code specifically extracts a portion of the command line that is used as an argument to NSMClient32.

Malware Family Suggestion

Based on the code's behavior, this strongly suggests a **downloader/dropper** or a **remote access trojan (RAT)**. The use of a function named `NSMClient32`, its interaction with a command line argument, and its reliance on an external (presumably malicious) function make it highly likely that this program downloads and executes additional malicious code or communicates with a command-and-control (C&C) server for further instructions. The seemingly innocuous command-line parsing and startup information handling are classic obfuscation techniques to make reverse engineering more difficult. Without knowing the functionality of `NSMClient32`, a more precise classification isn't possible. However, its function suggests that this could be an installer for another malware program.