Analysis Report for: 3F40B27FCBFCF59FF0B7E95E8034A786.exe.c

Overall Functionality

This C code appears to be a highly obfuscated installer or a component of a more complex malware program. The code heavily relies on indirect function calls, complex data structures, and extensive use of arithmetic operations that obscure its true functionality. The code interacts with the Windows API extensively, performing actions related to file system manipulation, registry access, process creation, and potentially other malicious activities. The structure suggests that it unpacks and executes other payloads, likely based on configuration data embedded within the installer file. The decompiled nature indicates it was likely extracted from a compiled binary, adding to the difficulty of analysis.

Function Summaries

Due to the obfuscation, providing precise summaries is difficult. The following provides a general idea based on function names and observed API calls:

* `sub_401000`: Window procedure, likely handling window painting and custom messages.
* `sub_40117D`, `sub_4011EF`, `sub_401299`, `sub_4012E2`: Seem to manipulate a complex data structure (likely a configuration array), possibly performing some form of encryption or data transformation.
* `sub_40136D`, `sub_401389`, `sub_40140B`: Functions involved in file processing and potentially executing embedded actions.
* `sub_401423`, `sub_401434`: String manipulation and file path operations. `sub_401434` is a central function that appears to act as a dispatcher, selecting actions based on a configuration value.
* `sub_402AA9`, `sub_402ACB`: Functions that fetch data from a configuration array.
* `sub_402B0B`, `sub_402B43`, `sub_402B5B`, `sub_402B89`, `sub_402BCD`: Registry access operations (creation, deletion, querying values).
* `DialogFunc`: Dialog box procedure for a custom installer dialog.
* `sub_402CFF`: Manages the creation and destruction of the installer dialog.
* `sub_402D63`: Core function. Appears to be responsible for unpacking and interpreting the configuration data. Crucial for understanding the installer's behavior.
* `sub_402F9C`: File reading and writing operation, probably used for extracting data.
* `sub_403178`, `sub_40318E`, `sub_4031A5`: File I/O helper functions.
* `start`: The main function; initiates the unpacking and execution process.
* `sub_4036BE`, `sub_4036E8`, `sub_403703`, `sub_403738`, `sub_403756`, `sub_403798`: Functions that manage dynamically loaded libraries (DLLs).
* `sub_403A5D`, `sub_403B16`, `sub_403B35`, `sub_403FE2`, `sub_404009`, `sub_40402B`, `sub_40403E`, `sub_404055`, `sub_404070`, `sub_40413F`, `sub_404174`, `sub_4043F4`, `sub_404418`, `sub_404454`, `sub_40449B`, `sub_4047EC`, `sub_404852`, `sub_404917`, `sub_40492F`, `sub_40495C`, `sub_4049DC`, `sub_404A0E`, `sub_405005`, `sub_405091`: A large number of functions related to the installer's user interface and logic. Many functions are helper functions for other operations.
* `sub_4051CF`: Handles installer dialog events.
* `sub_405557`, `sub_4055D4`: Directory creation operations.
* `sub_4055F1`, `sub_405609`, `sub_40564C`, `sub_40566A`, `sub_405686`, `sub_4056EA`, `sub_405732`, `sub_405902`, `sub_40592D`, `sub_405949`, `sub_40596F`, `sub_40599B`, `sub_4059F0`, `sub_405A68`, `sub_405ABE`, `sub_405ADE`, `sub_405B03`, `sub_405B32`, `sub_405B7B`, `sub_405BAA`, `sub_405BD9`, `sub_405D49`, `sub_405D75`, `sub_405DF0`, `sub_405E1E`, `sub_405E51`, `sub_405EC8`, `sub_405EE1`, `sub_405F6A`, `sub_405F8C`, `sub_4061D4`, `sub_40626D`, `sub_406294`, `sub_406302`, `sub_40633E`, `sub_406377`, `sub_4063B9`, `sub_406427`, `sub_406447`: A diverse range of functions for file system, registry, process manipulation, and other tasks.
* Many functions are wrappers or helpers for other functions or Windows API calls. This adds to the obfuscation.

Control Flow

The control flow is highly complex and interwoven. The `sub_401434` function acts as a central dispatcher, using a switch statement to select a specific action based on a code received as input. Each case within the `sub_401434` function may lead to further function calls, loops, and conditional statements.

Data Structures

* **Configuration Array:** A large, complex data structure likely stores configuration information that dictates the installer's behavior. This structure appears to be used by many of the functions mentioned earlier. The exact layout is not immediately clear due to obfuscation.
* **Dynamically Loaded Libraries (DLLs):** The installer appears to dynamically load and use additional functionality from DLLs. This makes reverse engineering difficult.
* **Other Structures:** Numerous other structures are used throughout the code, but their specific purpose is obscured by the obfuscation.

Malware Family Suggestion

Based on the complexity, obfuscation techniques, use of Windows API calls for file system manipulation and registry modification, along with the dynamic loading of DLLs, this code strongly resembles a **dropper** or **installer** associated with a **polymorphic** malware family. It likely unpacks additional malicious payloads onto the system after it runs. The use of anti-analysis techniques hints at a more sophisticated malware author. It cannot definitively be classified into a specific malware family without further analysis of the unpacked payloads. However, the characteristics are consistent with malware that relies on complex unpacking to evade detection and analysis.