Analysis Report for: 7728B54CFE1E9CDBE09F9D3A75429F95.exe.c

Overall Functionality

This C code, likely generated from a disassembled malware sample, exhibits characteristics of a sophisticated malware designed for network operations and potentially data exfiltration. It heavily uses Windows API functions for resource loading, file system manipulation, network communication (WinInet), thread creation, and exception handling. The obfuscation through function names (`sub_401000`, `sub_401030`, etc.), extensive use of function pointers, and complex control flow makes analysis challenging but reveals a multi-stage process. The code attempts to fetch and execute commands from a remote server, indicating likely malicious behavior. The date check in `WinMain` suggests the malware may have a limited lifespan or activation period.

Function Summaries

Due to the complexity and obfuscation, providing a complete and precise summary for every function is infeasible. However, here's a summary of some key functions:

* **`WinMain`**: The main entry point. This function orchestrates the malware's operations, including checking system time against a specific date, loading a DLL (`WinInet.dll`), fetching and potentially decoding data from a remote server ("38.47.239.143"), and then executing the data.

* **`sub_401060`**: Loads and parses resources from a given module. It searches for a specific string within the resource data, possibly configuration data or commands.

* **`sub_401110`, `sub_401160`, `sub_4011B0`**: These functions work together to locate resources within multiple loaded modules, likely searching for different configurations based on language or other criteria.

* **`sub_4012F0`**: Retrieves the current executable's path and potentially modifies it, possibly to create a persistent location for the malware or other files.

* **`sub_401510`**: Searches for a specific byte (`a2`) within a string (potentially a path).

* **`sub_401560`**: Copies data to a specified location; this potentially copies configuration data or the fetched code to a designated space.

* **`sub_401BF0`**: Loads and decodes a string from a resource.

* **`sub_402170`**: Performs checks related to the current executable's location and attempts to create a persistent shortcut, implying installation and persistence functionality.

* **`sub_402600`**: Loads functions from "WinInet.dll" (InternetOpenA, InternetConnectA, etc.), crucial for network communication. This indicates direct network capabilities within the malware itself.

* **`StartAddress`**: A simple thread function used in `sub_402170` for potentially timed operations.

* **`sub_40301F` - sub_403350`**: Appear to manage memory allocation and deallocation.

* **`sub_405833` - `sub_40AC7C`**: Functions related to exception handling and potentially error handling related to the network operations or remote code execution.

Control Flow

The control flow is highly obfuscated. Significant functions have nested loops, conditional branches based on various checks (file existence, time checks, string matches), and function pointer calls, making it difficult to give a concise description. However, the general pattern is as follows:

1. **Initialization:** `WinMain` initializes data structures and sets up environments.
2. **Checks:** Various checks are performed (date check, file path checks, existence of specific files, etc.) before major operations are executed.
3. **Resource Loading and Parsing:** Resources are loaded and parsed, likely containing configurations and instructions.
4. **Network Communication:** Functions from `WinInet.dll` are used for network interactions; data is downloaded from a remote server.
5. **Data Processing:** The fetched data is potentially processed (decoded or decrypted).
6. **Code Execution:** The processed data is likely executed; this could involve shell commands or arbitrary code.
7. **Cleanup:** The code attempts to maintain persistence before eventually exiting

Data Structures

The code uses several data structures, some implicitly defined, others apparent through type information from the decompiler:

* **Function pointers:** Used extensively to dynamically call functions, making static analysis very difficult and code more adaptable.

* **Custom data structures:** Internal data structures, often represented as arrays or structures. These structures likely manage network connections, fetched data, and internal malware state. Their exact definitions are unclear due to obfuscation. Structures allocated using the heap

functions suggest dynamic data sizing.

* **`_RTL_CRITICAL_SECTION`**: A Windows structure used for thread synchronization; this hints at concurrency in the malicious actions.

Malware Family Suggestion

Based on the analyzed functionality, this malware appears to be a type of **downloader/dropper**. It downloads and executes additional malicious code from a remote server, acting as an intermediary to install the main payload. The persistence mechanisms, intricate resource loading, and use of obfuscation techniques point towards a fairly advanced and potentially sophisticated malware family. The use of `WinInet.dll` for network operations suggests the malware could also be classified as a **network-based trojan**. More detailed analysis would be needed to precisely determine its family, but the presence of a command-and-control (C&C) server interaction makes this type of classification probable.