Analysis Report for: 7728B54CFE1E9CDBE09F9D3A75429F95.exe.c

Overall Functionality

This C code exhibits characteristics strongly suggestive of malware. It appears to be a backdoor or downloader, designed to establish a connection to a remote server, download and execute additional malicious code. The code heavily obfuscates its functionality through extensive use of function pointers, indirect calls, and custom data structures. The date check and the network communication functions point to an attempt at evasion and persistence.

Function Summaries

Due to the complexity and obfuscation, precise descriptions for all 324 functions are impractical. Below are summaries of some notable functions:

* `WinMain`: The program's entry point. It performs several actions including a date check, loading external libraries ("WinInet.dll", "USER32.DLL"), and attempting network communication. It appears to be the main orchestrator of the malicious activity.
* `sub_401030`: Throws a C++ exception, likely used for error handling or to disrupt analysis.
* `sub_401060`, `sub_401110`, `sub_401160`, `sub_4011B0`: These functions appear to handle resource loading from a module, likely used to retrieve embedded strings or code. The use of `FindResourceA` and `FindResourceExA` hints at embedded resources within the malware itself.
* `sub_4012F0`: This function retrieves the malware's own file path and manipulates it, possibly to create a persistent installation or hide its location.
* `sub_402170`: This function performs checks related to file paths and existence, likely determining if the malware is in a suitable location or attempting to move to a new one. It uses `SHGetFolderPathA` to get a system folder path (potentially to create a persistent installation) and checks for specific file paths ("\\AppData\\Local\\Temp\\").
* `sub_402600`: Loads functions from "WinInet.dll", which is frequently used for network operations (HTTP, FTP). This suggests network communication capabilities.
* `StartAddress`: A thread function that waits for a period and then sets an event, possibly signaling the completion of a task or synchronization.
* Functions starting with `sub_4030`: These appear to manage some sort of custom data structure, likely for memory allocation and management. Use of `HeapAlloc`, `HeapFree`, etc. indicates custom heap manipulation.
* Many functions with names like `sub_40xxxx` appear to be smaller helper functions, possibly for string manipulation, memory management, or other low-level operations. Their exact purpose is obscured by the obfuscation.

Control Flow

* `WinMain`: The control flow in `WinMain` is complex and conditional. It checks a date, attempts to load functions from "WinInet.dll" and "USER32.DLL". The main sequence involves establishing a connection (using `dword_40F838`, `dword_40F844`, `dword_40F824`), downloading data (`dword_40F83C`), modifying it, and potentially executing it (`VirtualAlloc_0` suggests memory allocation for code execution). Error handling appears to be implemented using exception handling and function returns.
* `sub_401060`: This function iterates through a resource until it finds a null-terminated string.
* `sub_4011B0`: This function iterates through a list of modules until it finds one containing a specific resource.
* Many functions have nested conditionals and loops, making detailed analysis extremely difficult.

Data Structures

The code uses several data structures, most notably:

* Custom structures used by functions with `sub_4030` prefixes. These likely involve manual memory management and could be used to mimic standard C++ structures such as vectors or lists, aiding in obfuscation.
* The `_RTL_CRITICAL_SECTION` struct from Windows is used for thread synchronization, suggesting multi-threaded operations within the malware.

Malware Family Suggestion

Based on the analysis, this code likely belongs to a **backdoor or downloader** family of malware. Its primary functionality seems to be to establish a connection to a remote Command and Control (C2) server, download additional malicious code, and execute it on the victim's machine. The use of obfuscation techniques and multiple checks make it an advanced sample. Specific sub-classification requires further investigation of the network communication details and the downloaded payload. The extensive use of custom heap management adds another layer of sophistication which is commonly seen in more advanced malware. The date check is indicative of possible time-based activation or expiry.