Analysis Report for: 14FB9224BDA8FDFBB30B608B2067E2A4.au3

Overall Functionality

The C code is highly obfuscated and performs a series of actions that strongly suggest malicious intent. It uses numerous nested loops, conditional statements, and mathematical operations to obscure its true purpose. The code installs files ("acceptancy" and "obtenebrate"), manipulates DLL structures, makes numerous system calls (File operations, Registry operations, Process operations, Network operations), and interacts with the GUI (likely to hide its activities). The core functionality appears to involve loading and executing code from a file ("acceptancy") into a specially created DLL structure, then making numerous calls to a function `$TIVNZNZZ` (likely a wrapper for a Windows API function), passing heavily obfuscated strings and data. This strongly points towards the execution of malicious code.

Function Summaries

* **`LWBAVTOM($bzirdqx)`:** This function takes a space-separated string of numbers (`$bzirdqx`) as input. It splits the string, converts each number to its ASCII character equivalent, adding 0xfffffff9, and concatenates the characters to form a new string. This is a simple form of string decryption or encoding. The return value is the resulting decoded string.

* **`SDVMYZXWJ()`:** This function performs several GUI and system actions, including setting GUI colors and states, checking for process existence and waiting for their closure, making ping requests to suspicious URLs (`htttp://LOz1CK.com`, `htttp://uPbqo1E2.net`), setting window states, and obtaining window class lists. These actions are likely used for reconnaissance or to check for the presence of security software. It doesn't return a value.

* **`DRQCMPIO()`:** This function performs file operations, including getting shortcut information, creating directories, moving and renaming files, reading and writing to files, and registering and unregistering AutoHotkey adlibs. The purpose is likely to manipulate files and system state to further the malware's goals. It doesn't return a value.

* **`SNOFTWP()`:** This function performs several actions, including pinging suspicious URLs (`htttp://qNQy1.net`, `htttp://G6RHdoRn.net`), setting hotkeys, waiting for window closure, getting file attributes, and deleting GUI controls. This function seems to further the malware's execution, potentially creating persistence mechanisms or hiding actions from the user. It doesn't return a value.

* **`XDXJPUKZZ()`:** This function performs various system actions, including setting GUI events, closing internet connections, writing to files, deleting GUI controls, performing trigonometric calculations, setting window tips, setting process priority, reading GUI controls, and closing windows. These actions serve to further the malware's goals, potentially altering system settings or hiding activities. It doesn't return a value.

* **`UQNFTDDID()`:** This function executes a file save dialog, checks for file and process existence, performs process operations, moves and deletes files, sends GUI messages, makes ping requests (`htttp://8vuTgkN.io`), and performs mouse movements. It doesn't return a value.

Control Flow

The control flow is extremely complex due to heavy nesting of `for` loops and `if-else if-else` statements. The majority of the code consists of deeply nested loops that iterate many times, performing various actions within each iteration. The conditional statements often check values of global variables, possibly influencing the execution path dynamically based on system conditions or previous operations. This makes static analysis extremely difficult. The complexity is designed to hinder reverse engineering and analysis. A significant portion of the control flow is dedicated to manipulating global variables using arithmetic operations and conditional checks, ultimately determining file operations, process control, and network interactions.

Data Structures

The primary data structure is the DLL structure created using `DllStructCreate`. The data contained within this structure (`$hrcakgnui`) is the decoded output of `LWBAVTOM` and likely contains the malicious payload. This structure is then used as the target for various `$TIVNZNZZ` function calls, which are likely API calls that manipulate the loaded code. The global variables (`$hdvbgsuc`, `$enwfeyh`, `$habjtjpab`, etc.) seem to serve as counters or flags that affect the execution path within the nested loops.

Malware Family Suggestion

Given the obfuscation techniques, file installations, DLL manipulation, registry access, process management, and network communication to external domains, this code is highly suggestive of a **remote access trojan (RAT)** or a sophisticated **information stealer**. The numerous file operations and process manipulations indicate a possible intention to exfiltrate sensitive data or alter the system's behavior in a way that could allow for further malicious activities. The complexity and obfuscation make definitively classifying the malware family difficult without dynamic analysis in a controlled environment. The use of AutoHotkey-like syntax adds another layer of obfuscation, making it harder to trace its origins and intentions.

Disclaimer: Analyzing potentially malicious code can be risky. This analysis is based solely on static inspection and does not constitute a guarantee of the exact functionality or malware family. Dynamic analysis in a sandboxed environment is necessary for definitive classification and full understanding of the code's behavior. Do not run this code on any system you value.