Analysis Report for: c.txt

Overall Functionality

This C code exhibits strong characteristics of malware. It's highly obfuscated using a custom encoding scheme (`VKDNHDMYHSQJNHC` and `HOJTTVMGDVIY`) to hide its true functionality. The code appears to deploy a payload (`RUNPE` function) and interacts with system resources (`READRESOURCES`, `ATALMTCGGG`). The heavy use of encoded strings and seemingly arbitrary function names strongly suggests an attempt to evade detection by antivirus software. The core functionality involves decoding several long strings representing data, likely shellcode or configuration information. This decoded data is then passed to functions that seem to handle process execution, resource manipulation, and potentially encryption/decryption. The ultimate goal is likely to execute malicious code within the victim's system.

Function Summaries

* **`EXECUTE`**: This is not explicitly defined, but it's used throughout the code and likely represents a function (possibly a system call or API) that executes commands or code.

* **`DllStructGetData`**: (Obtained through `EXECUTE`) This likely retrieves data from a DLL structure. Its precise purpose is obfuscated.

* **`BinaryToString`**: (Obtained through `EXECUTE`) This likely converts binary data (such as shellcode) into a string representation.

* **`HOJTTVMGDVIY`**: This function acts as a custom decoder. It takes two encoded strings as input, decodes them, and performs a XOR operation based on the length of the second decoded string. The result is a decoded string.

* **`VKDNHDMYHSQJNHC`**: This function performs a substitution cipher. It takes an encoded string of comma-separated numbers, translates these numbers into corresponding characters from a predefined alphabet, and concatenates them to create a decoded string. This decoded string is likely a key or index used by `HOJTTVMGDVIY`.

* **`WOVJHAMFHU`**: This function contains a conditional statement that executes more decoded code based on the result of a call to `REKTJQJGQK` and `DYKACAYBNA`.

* **`BJDWBWCLQG`**: This function contains a conditional statement; if the condition is met, it executes a block of decoded code.

* **`READRESOURCES`**: This function retrieves resources from the system. The parameters `$resname` and `$restype` specify the resource name and type. The function then returns the decoded resource data.

* **`KWEWIXXUPI`**: This function executes several blocks of decoded code.

* **`VTAHSYNEZW`**: This function appears to run an infinite loop (`While (0x1)`), periodically executing decoded code and calling the `QPBAEMCVUP` function if a specific condition is met. This suggests persistence functionality.

* **`ACL`**: This function seems to manipulate Access Control Lists (ACLs) using the provided `$handle`.

* **`ARSVGRRUTH`**: This function executes several blocks of decoded code.

* **`WSJJHPIQQO`**: This function appears to handle some form of encryption or data manipulation. It takes encoded data and keys as input and outputs an encrypted/decrypted key.

* **`JMJAWLOMYX`**: This function contains a loop that executes several blocks of decoded code multiple times.

* **`CWKNFGVTJS`**: This function seems to handle error checking and potentially cleanup based on the provided handle.

* **`KFXBQAQZVG`**: This is an empty function.

* **`ATALMTCGGG`**: This function seems to create or manipulate a file named "rrinstaller.exe". This could be the dropper or the actual payload.

* **`QPBAEMCVUP`**: This function executes the main payload using the `RunPE` function.

* **`RUNPE`**: This function is the likely payload execution mechanism. It takes paths and flags for persistence and protection as parameters and executes decoded shellcode.

* **`BACAPGUPRQ`**: This function contains a conditional statement, if the condition is met, it executes a block of decoded code.

Control Flow

The control flow is complex and deliberately obfuscated but can be generally described as follows for some key functions:

* **`HOJTTVMGDVIY`**: A simple `For` loop iterates through the decoded first input string, performing XOR operations based on the length of the second decoded string and returning the final decoded string.

* **`READRESOURCES`**: The function directly calls `REKTJQJGQK` and `DYKACAYBNA` on decoded strings to retrieve and process data. No

loops or conditionals are present within the function itself.

* **`VTAHSYNEZW`**: An infinite `While` loop executes decoded code repeatedly. A conditional check (`If`) inside the loop triggers the execution of `QPBAEMCVUP`, providing persistence.

* **`ACL`**: The function calls `EXECUTE` and then a decoded function (`$BN`) on a decoded string. The subsequent logic suggests it's setting or modifying ACLs for the specified handle.

* **`WSJJHPIQQO`**: This function uses nested function calls and lacks clear conditional branches, suggesting complex encryption/data manipulation routines.

* **`RUNPE`**: This function concatenates several decoded strings, presumably shellcode, and executes the resulting binary data using `Execute`.

Data Structures

The primary data structure is the array `$__g_acryptinternaldata`. This array's size and usage within the `WSJJHPIQQO` function suggest it's used to store intermediate data during encryption/decryption or other data manipulation processes. Another implied data structure is used to represent file handles and other system objects—the exact structure is not visible due to obfuscation.

Malware Family Suggestion

Based on the functionality of the code (custom encoding, shellcode execution, and persistence mechanisms), this sample most closely resembles a **downloader/dropper** or a **backdoor** type of malware. The `RUNPE` function, suggesting the execution of a separate payload, strongly points toward a downloader/dropper. The persistence mechanism (`VTAHSYNEZW`) and potential file manipulation (`ATALMTCGGG`) further suggest backdoor capabilities, implying the malware might aim to provide remote access to the infected system. The encryption/decryption aspects hint at an attempt to communicate with a command and control (C&C) server to receive further instructions or exfiltrate stolen data. The exact classification would require further analysis of the decoded shellcode and network activity.