Analysis Report for: BB7F4147C34E2937C6EE7DEF98C8CB52.exe

Overall Functionality

This C# code snippet (disguised as C by using `$` for variables) uses base64 encoding to obfuscate a PowerShell command. The code first defines a base64 encoded string (`$mvhij`). It then decodes this string using `[Convert]::FromBase64String()` and converts the resulting byte array to a string using ASCII encoding. Finally, it executes the decoded PowerShell command using `IEX`. The core functionality is to execute malicious PowerShell code hidden within a base64 string.

Function Summaries

The code doesn't define any functions in the traditional C sense; it utilizes several .NET methods:

* **`[Text.Encoding]::ASCII.GetBytes('...')`**: This method takes a string as input and returns a byte array representing the string encoded in ASCII.
* **`[Text.Encoding]::ASCII.GetString(...)`**: This method takes a byte array as input and returns a string representing the byte array decoded from ASCII.
* **`[Convert]::FromBase64String(...)`**: This method takes a base64 encoded string as input and returns a byte array representing the decoded data.
* **`Sort-Object { Get-Random -SetSeed 1321827412 }`**: This is a PowerShell cmdlet. It takes an array of items and sorts them randomly using a specified seed for reproducibility. This is used to further obfuscate the order of bytes in the base64 string.
* **`IEX`**: This is a PowerShell cmdlet that executes the provided string as a PowerShell command.

Control Flow

The code's control flow is straightforward and linear:

1. A long base64 encoded string is assigned to the variable `$mvhij`.
2. `[Text.Encoding]::ASCII.GetBytes` converts the literal string into a byte array.
3. The bytes are sorted using the `Sort-Object` cmdlet within the PowerShell pipeline, introducing a layer of additional obfuscation. The sorting is deterministic due to the fixed seed.
4. `[Text.Encoding]::ASCII.GetString` converts the sorted byte array back to a string. This string is still the original base64 encoded string, although the byte ordering has been altered by the sorting.
5. `[Convert]::FromBase64String` decodes the base64 string into a byte array.
6. `[Text.Encoding]::ASCII.GetString` converts the decoded byte array into a string which represents the original PowerShell code.
7. `IEX` executes the resulting PowerShell command.

Data Structures

The primary data structures involved are:

* **Strings**: Used to store the base64 encoded string and the decoded PowerShell command.
* **Byte Arrays**: Used as an intermediate representation of the data during the encoding and decoding processes.

Malware Family Suggestion

Given the code's structure and obfuscation techniques, it strongly suggests a **downloader/dropper** type of malware. The base64 encoding and the use of `IEX` to execute arbitrary PowerShell code are common tactics used to hide and execute malicious payloads. The payload itself is not visible in the provided snippet, but the mechanics indicate it will likely download and/or execute additional malicious code from a remote server. The random sorting step doesn't alter the payload itself, it just makes analysis slightly more difficult. The overall goal is to avoid simple detection by antivirus software.