Analysis Report for: D65FDD9014571C40FA407C7CEC2A9EC4.bat

Overall Functionality

This batch script performs a series of actions that seem designed to obfuscate the creation and deployment of a file named `decode.bat`. The script uses Base64 encoding, temporary directories, looping, and file comparisons to achieve this. It downloads a base64 encoded file, decodes it into a binary file (`source.data`), extracts this binary file (likely containing `decode.bat`), and then manages multiple copies of `decode.bat` across different directories (`%~dp0` and `C:\Temp`), keeping only the smaller version after each iteration of a loop. The final step is to move the resulting `decode.bat` to the script's directory. The presence of a PowerShell script (`encode_folder.ps1`) suggests further actions might be performed within each loop iteration, likely related to data encoding and/or file manipulation.

Function Summaries

This code is a batch script, not a collection of C functions. Therefore, there are no functions in the traditional C sense. The script uses batch commands to achieve its functionality.

Control Flow

1. **Initialization:** Sets variables like `loop_number`, `wait`, and `file_count`. It checks if the script's directory contains files. If not, it exits.
2. **Base64 Decoding:** Creates a temporary directory, downloads a Base64-encoded string, and uses PowerShell to convert and save it as a binary file (`source.data`).
3. **Extraction:** Extracts the contents of `source.data` (likely `decode.bat`) into the script's directory. This assumes `source.data` is a compressed archive, most likely a tar file.
4. **File Management Loop:** A loop iterates a number of times (`loop_number`). Inside the loop:
* It executes a PowerShell script (`encode_folder.ps1` if present). This part is not defined within the provided script.
* It creates `C:\Temp` if it doesn't exist.
* It compares sizes of `decode.bat` in `%~dp0` and `C:\Temp`, keeping the smaller one in `C:\Temp` and deleting the larger one. This introduces unnecessary complexity and may indicate obfuscation.
5. **Post-Loop Cleanup:** Moves the final (presumably smaller) `decode.bat` from `C:\Temp` to the script's directory.
6. **Final Cleanup:** Deletes temporary files and directories.

Data Structures

The script primarily uses simple variables to store file paths, counts, and sizes. No complex data structures are employed.

Malware Family Suggestion

Given the obfuscation techniques used—Base64 encoding, temporary file usage, looping to manage multiple copies of a file, and the likely existence of an external PowerShell script with unknown functionality—this script is highly suspicious and likely part of a malware family. The behavior strongly suggests a dropper or installer. The script downloads and installs `decode.bat`, which could then perform malicious activities. The fact that the script is trying to minimize the size of `decode.bat` indicates an effort to evade detection. The exact nature of the malware would depend on the contents and behavior of the `decode.bat` and `encode_folder.ps1` files. It could be a downloader, RAT (Remote Access Trojan), ransomware, or any other type of malware. Further investigation is needed.

Important Note: Do *not* execute this code. It is very likely malicious. The analysis is based solely on the provided code. The behavior is highly indicative of malicious intent. The external script `encode_folder.ps1` is a major unknown that could further obfuscate the malware's actions.