Analysis Report for: B77C648FB7E1AFFBE8A4F99BB94B7AB3.exe

Overall Functionality

This batch script downloads, extracts, and executes a malicious payload. It uses Base64 encoding to obfuscate the payload, `certutil.exe` to decode it, and `tar.exe` to extract it. Finally, it executes a batch file named `Show_all_icons.bat` located within the extracted files. The script employs a delay mechanism (`wait`) to potentially avoid immediate detection. The temporary directory is cleaned up after execution.

Function Summaries

The code doesn't contain C functions in the traditional sense; it's a batch script using external commands. The functionality is broken down into stages managed by the sequence of commands.

* **`set "wait=ping localhost -w 1000 -n 2 > nul && ping localhost -w 1000 -n > nul"`**: This line defines a variable `wait` which essentially creates a 5-second delay using the `ping` command. The `-w 1000` option sets the timeout to 1 second, `-n 2` sends two pings, effectively making a delay of 2 seconds per `ping` command which makes total delay of 4 seconds. The `> nul` redirects output to suppress the ping results.

* **`certutil.exe -decode "%TD%\_.b64" "%TD%\source.data" >nul 2>&1`**: This uses the Windows `certutil.exe` utility to decode the Base64 encoded file (`\_.b64`) into a binary file (`source.data`). `>nul 2>&1` redirects both standard output and error output to suppress any output.

* **`tar.exe -xf "%TD%\source.data" -C "%TD%`**: This uses the `tar` utility (likely from a third-party package) to extract the contents of `source.data` into the temporary directory. `-x` extracts, `-f` specifies the archive file, and `-C` specifies the extraction directory.

* **`call "%TD%\Show_all_icons.bat"`**: This line calls the `Show_all_icons.bat` batch file, which is the actual malicious payload. This is the crucial step where the harmful action occurs.

Control Flow

The script follows a linear control flow. There are no loops. Conditional statements are minimal; the `if not exist "%TD%" mkdir "%TD%"` creates the temporary directory if it doesn't already exist. The script's flow is sequential: create temporary directory, decode Base64, extract archive, execute payload, clean up. The `wait` commands introduce artificial delays.

Data Structures

There are no explicit data structures used in the script. The main data structures involved are files: the Base64 encoded file (`\_.b64`), the extracted binary file (`source.data`), and the malicious batch file (`Show_all_icons.bat`). The `%TD%` variable acts as a simple string to store the path to the temporary directory.

Malware Family Suggestion

Based on the code's functionality, this script strongly resembles a **downloader/dropper**. It downloads a malicious payload encoded in Base64, decodes it, extracts it, and then executes the extracted payload. This is a common technique used to deliver malware and evade detection. The payload itself (`Show_all_icons.bat`) is the actual malware, and its nature (what it does) can't be determined from the provided script. The fact that the code uses common system utilities (ping, certutil, tar) to achieve its goal means it's designed to be inconspicuous. It's possible that after extraction more malware would be downloaded or it could have various malicious behaviours ranging from data exfiltration, system compromise to ransomware. Without examining `Show_all_icons.bat` the exact nature of the malware remains unknown but based on the dropper functionality we can categorise it as Malware family (downloader/dropper)