Analysis Report for: DE1B87C13386439460AC2FBC43CD93C6.exe

Overall Functionality

This C code is a highly obfuscated script that appears to be designed to execute malicious actions. It uses a large number of seemingly random variable names and string literals, making it difficult to understand its purpose at first glance. The code heavily relies on environment variables, file manipulation, and command execution. The presence of strings like "Marriage", "Mobile", "Prisoner", and "Tourist" are likely obfuscation techniques, masking the true meaning of the variables. The code appears to download and execute files, potentially making it a downloader or installer for further malicious payloads. The use of `tasklist`, `findstr`, `cmd`, and `copy` suggests attempts to check running processes, search files, and manipulate the system.

Function Summaries

There are no explicitly defined functions in this code. All the operations are performed using built-in commands and system calls directly.

Control Flow

The code's control flow is primarily driven by sequence and conditional statements (`if`). There are no loops (`for`, `while`, `do-while`). The main logic follows this pattern:

1. **Variable Assignments:** Many lines set variables (often obfuscated) with values, integers, or strings. These include assignments involving `Set` command in batch scripting which implies the usage of Windows batch files instead of C code.
2. **System Calls:** The code extensively uses command-line tools:
* `tasklist`: Lists running processes. Used with `findstr` to search for specific process names (potentially checking for the presence or absence of security software).
* `findstr`: Searches text for specific strings. Used in conjunction with `tasklist` and also for file content filtering.
* `copy`: Copies files.
* `set /a`: Performs arithmetic operations.
* `set /p`: Reads input from a file.
* `cmd /c`: Executes commands. This is how external commands are run in the context of the script.
3. **Conditional Execution:** `if` statements check the return codes of commands. Success or failure of a command (like `findstr`) can alter the execution path.
4. **File Manipulation:** The code creates and modifies files (`>`, `>>`, `+`). This is evident in the creation of a file at a path constructed using the obfuscated variable names. Content is appended or created with strings and the output of commands. The `cd` command changes the current directory to the one of the newly created files.
5. **Command Execution:** The script ultimately uses `start` to execute a file whose path is constructed earlier in the script using obfuscated variable names and possibly other dynamically generated file paths.

Data Structures

The code primarily uses simple variables. No complex data structures (arrays, linked lists, etc.) are explicitly defined. The variables are mostly strings or integers. The obfuscation makes it hard to determine the exact data type of each variable because of the way variables are constructed and assigned using command line operations.

Malware Family Suggestion

Given the obfuscation, file manipulation, command execution, and process checking, this code strongly resembles a downloader or installer for a broader malware family. The behavior suggests potential functionality as a:

* **Dropper:** Downloads and executes a secondary payload.
* **Installer:** Installs other malware components.
* **Backdoor:** Could establish persistent access, though this is not explicitly shown.

The lack of self-contained functions and the heavy reliance on external commands make it difficult to categorize it precisely within a specific malware family (e.g., ransomware, trojan, etc.). However, its actions align strongly with the typical initial phases of many malware infections. Further analysis of the payload(s) it downloads and executes would be necessary for definitive classification. The extensive use of obfuscation techniques is also indicative of malware behaviour intended to evade detection.

Conclusion

This code is malicious and should not be executed. Its obfuscation is a significant red flag. Reverse engineering to fully understand the payload it would install would be necessary. The code provided in the problem does not include actual C code. It's actually a batch script that employs many Windows commands to perform its malicious actions.