Analysis Report for: ff.txt

Overall Functionality

This C code is a malicious script designed to download and execute a payload. It first downloads an executable file (`SystemEngine.exe`) from a remote server. If the download is successful, it then creates a PowerShell script (`GlobalProfile.ps1`) that downloads a second file (`GlobalData.au3`) containing additional malicious code from the same server. Finally, it executes this downloaded PowerShell script, which in turn executes the downloaded executable with the downloaded data as an argument. The entire process is designed to be stealthy by hiding command windows and using PowerShell for execution. The use of hardcoded URLs and file paths further suggests malicious intent.

Function Summaries

* **`_DOWNLOADFILE($url, $target)`:** Downloads a file from a given URL and saves it to the specified target path.
* Parameters:
* `$url`: The URL of the file to download (string).
* `$target`: The local path where the file should be saved (string).
* Return Value: None. It uses `RunWait` which implicitly returns success or failure.

* **`_CHECKFILE($fpath)`:** Checks if a file exists at the given path.
* Parameters:
* `$fpath`: The path to the file to check (string).
* Return Value: `1` (true) if the file exists, `0` (false) otherwise.

* **`_WRITEPS1($content, $psfile)`:** Writes the given content to a PowerShell script file.
* Parameters:
* `$content`: The content to be written (string).
* `$psfile`: The path to the PowerShell script file (string).
* Return Value: None. Implicitly returns failure if file cannot be opened.

* **`_RUNPS1($psfile)`:** Executes a PowerShell script file.
* Parameters:
* `$psfile`: The path to the PowerShell script file (string).
* Return Value: None. Implicitly returns success or failure via `RunWait`.

Control Flow

* **Main Execution Block:** The code begins by downloading `SystemEngine.exe`. An `If` statement checks if the download was successful using `_CHECKFILE`. If successful, it proceeds to construct the PowerShell script code (`$pscode`), write it to `GlobalProfile.ps1` using `_WRITEPS1`, and then execute it using `_RUNPS1`.

* **`_DOWNLOADFILE`:** This function constructs a PowerShell command string to download a file using `Invoke-WebRequest`. It then uses `RunWait` to execute this command, hiding the command window. Error handling is minimal; failure to download is not explicitly handled.

* **`_CHECKFILE`:** This function simply calls the built-in `FileExists` function and returns its result.

* **`_WRITEPS1`:** This function opens the specified file in write mode (`0x2`). If the file cannot be opened (`$handle = +0xffffffff`), it exits. Otherwise, it writes the content and closes the file. Error handling is basic; it does not report specific file errors.

* **`_RUNPS1`:** This function constructs a PowerShell command string to execute a script with `Bypass` execution policy. It uses `RunWait` to execute this command, hiding the command window. Error handling is also minimal here.

Data Structures

The code primarily uses strings to represent URLs, file paths, and PowerShell commands. No complex data structures are employed. The `$pscode` variable holds the PowerShell script as a single large string.

Malware Family Suggestion

Given its functionality, this code strongly suggests a **downloader/dropper** malware family. It downloads and executes a payload (`SystemEngine.exe`), and then a second file whose purpose is to trigger another action, possibly persistence. The use of PowerShell and obfuscation via a second-stage download increases its stealth and makes analysis more difficult. The hardcoded URLs and file paths are characteristics of many malicious programs. The use of a separate config file via `GlobalData.au3` suggests a modular design that makes future modification easier for the malware authors. The combination of these elements points towards a more sophisticated malware operation rather than a simple single-stage attack.