Analysis Report for: script.ps1

This code is not C code; it's a PowerShell script. It's designed to create and set a custom email signature in Microsoft Outlook. Let's analyze it:

Overall Functionality

The PowerShell script automates the creation of an email signature with HTML, text, and RTF versions, and then attempts to set it as the default signature in various versions of Microsoft Outlook. The signature includes contact information, social media links, an image, and a confidentiality notice. Critically, it tries to set and then *immediately* unset the signature as default, a highly suspicious behavior.

Function Summaries (PowerShell cmdlets)

The script doesn't define custom functions; it uses built-in PowerShell cmdlets:

* `Join-Path`: Combines paths to create a new path string. Parameters: `-Path`, `-ChildPath`. Return value: String representing the combined path.
* `Test-Path`: Checks if a path exists. Parameters: `-Path`. Return value: Boolean (True/False).
* `New-Item`: Creates a new item (in this case, a directory). Parameters: `-Path`, `-ItemType`. Return value: Object representing the newly created item.
* `Out-Null`: Suppresses output from a command. No parameters. No return value.
* `Out-File`: Writes content to a file. Parameters: `-FilePath`, `-Encoding`. Return value: Object representing the file.
* `Set-ItemProperty`: Sets a property value in the registry. Parameters: `-Path`, `-Name`, `-Value`. Return value: Object representing the modified registry item.
* `Remove-ItemProperty`: Removes a property from the registry. Parameters: `-Path`, `-Name`, `-ErrorAction`. Return value: Object representing the registry item after modification.
* `Start-Sleep`: Pauses script execution for a specified duration. Parameters: `-Milliseconds`. Return value: None.
* `Write-Host`: Writes output to the console. Parameters: Text to output. Return value: None

Control Flow

1. **Signature Creation:** The script first defines variables for the signature name and path. It creates the signature directory if it doesn't exist. Then, it constructs the HTML content of the signature (a lengthy HTML snippet defining the layout and content of the email signature).

2. **Signature File Writing:** The HTML content is written to an `.htm` file, while empty strings are written to `.txt` and `.rtf` files.

3. **Registry Manipulation:** The core of the script's functionality lies in the loop that iterates through different Microsoft Office versions (`16.0`, `15.0`, `14.0`). For each version, it checks if the corresponding registry key exists. If it does:
* It sets the `NewSignature` and `ReplySignature` registry values to the defined signature name.
* It waits for 500 milliseconds.
* It attempts to *remove* the `NewSignature` and `ReplySignature` registry values. This is unusual and suggests malicious intent.
* `-ErrorAction SilentlyContinue` is used to suppress any error messages related to registry removal, which is another red flag.

4. **Output:** Finally, the script writes a message to the console indicating that the signature has been installed.

Data Structures

* `$firmaNombre`: String variable holding the name of the signature.
* `$signaturesPath`: String variable holding the path to the signatures directory.
* `$htmlPath`, `$txtPath`, `$rtfPath`: String variables holding paths to the signature files.
* `$htmlContenido`: String variable containing the HTML content of the signature.
* `$officeVersions`: Array of strings containing Office version numbers.

Malware Family Suggestion

The unusual behavior of setting and then immediately removing the registry keys for the default email signature strongly suggests **malware**, likely a form of **potentially unwanted program (PUP)** or a **backdoor**. The script's seemingly harmless actions (creating a signature) are a cover for potentially malicious behavior. The immediate removal of the registry keys could be to avoid detection or to create an unstable system state, allowing for later actions. The `-ErrorAction SilentlyContinue` further hides errors related to this registry manipulation. A legitimate script wouldn't need such a destructive and hidden cleanup. A more sophisticated variant could replace the seemingly harmless signature with malware payload. Further analysis of the image sources linked within the script should be done to be certain.