Analysis Report for: encode_folder.ps1

Overall Functionality

This PowerShell script creates a self-extracting archive disguised as a seemingly harmless batch file. The script compresses all files in its directory (excluding itself and a specified batch file) using `tar` with LZMA compression. The resulting compressed file is then Base64 encoded, and this encoded data is embedded within a generated `decode.bat` file. This batch file, when executed, decodes the Base64 data, reconstructs the compressed archive, extracts its contents, and then self-deletes. This technique is commonly used to obfuscate malicious payloads.

Function Summaries

* **`Create-TempDir`**: Creates a temporary directory if it doesn't already exist.
* Parameters: `$dirPath` (string): The path to the directory.
* Return Value: None.

* **`Copy-FilesToTemp`**: Copies files and subdirectories from a source directory to a temporary directory, excluding specified files.
* Parameters: `$sourceDir` (string), `$destinationDir` (string), `$excludeFiles` (string[]): Source and destination directories, and files to exclude.
* Return Value: An array of `System.IO.FileInfo` objects representing the copied items.

* **`Compress-Files`**: Compresses files in a directory using `tar` with LZMA compression.
* Parameters: `$sourceDir` (string), `$outputFile` (string): Source directory and output file path.
* Return Value: None.

* **`Encode-FileToBase64`**: Encodes a file's content into a Base64 string.
* Parameters: `$filePath` (string): Path to the file.
* Return Value: A Base64 encoded string.

* **`Create-DecodeBat`**: Creates a batch file that decodes a Base64 string, extracts a compressed archive, and then cleans up temporary files.
* Parameters: `$batFilePath` (string), `$base64String` (string), `$dataFileName` (string): Path to the output batch file, the Base64 encoded data, and the name of the data file within the archive.
* Return Value: None.

Control Flow

* **`Create-TempDir`**: Simple conditional statement: checks if the directory exists and creates it if not.

* **`Copy-FilesToTemp`**: Iterates through files and directories using `Get-ChildItem` with `-Recurse`. Filters out files to be excluded. Uses a conditional statement to handle directories and files separately. Error handling (`-ErrorAction Stop`) is included.

* **`Compress-Files`**: Executes the `tar` command using `&`. Includes an `--exclude` option.

* **`Encode-FileToBase64`**: Reads the entire file into memory using `[System.IO.File]::ReadAllBytes`. Performs a Base64 encoding.

* **`Create-DecodeBat`**: Splits the Base64 string into smaller lines to avoid exceeding the command-line length limit. Uses a `StringBuilder` for efficient string concatenation. It constructs a batch script with commands to:
1. Create a temporary directory.
2. Write the Base64 data to a temporary file.
3. Decode the Base64 data using PowerShell.
4. Extract the compressed archive using `tar`.
5. Delete temporary files and directories. Includes error handling.

Data Structures

The primary data structures used are:

* **Arrays:** `$excludeFileName` (string array) holds the list of files to exclude during the copy process. `$sourceItems` stores the results of `Get-ChildItem`. `$splitBase64` stores the lines of split Base64 data.
* **Strings:** Various strings are used to store file paths, Base64 encoded data, and the content of the generated batch file.
* **`System.Text.StringBuilder`:** Used for efficient concatenation of the batch file content, improving performance compared to repeatedly using the `+=` operator for string concatenation.

Malware Family Suggestion

Based on its functionality, this script strongly resembles a **dropper** or a **self-extracting archive packer**. It's designed to deliver a payload (the contents of the compressed archive) in an obfuscated manner. The use of Base64 encoding and a self-deleting batch script are typical techniques used to evade detection by antivirus software. The payload itself could be any type of malware, making it impossible to pinpoint a specific malware family without knowing the contents of the original compressed archive. The structure is consistent with advanced techniques used to deliver malware payloads to systems.