Analysis Report for: da.txt

Decoded using latin-1...

Overall Functionality

This VBA code appears to be designed for managing and processing data within an Excel workbook, likely for financial or accounting purposes. It includes several modules:

* **`EstaPasta_de_trabalho`**: Controls the visibility of scroll bars on specific worksheets ("Resumo") upon activation and deactivation of the workbook.
* **`Plan1`**: Contains a simple subroutine (`mostra`) that copies the string "Haislan" to the clipboard. This function seems unrelated to the main data processing functionality.
* **`Plan2`**: This is the core module, containing a `Worksheet_Change` event handler that performs complex data validation and manipulation based on user input in specific columns (Q, R, S). It updates other cells (Column 17 and 18) based on the values entered and user input through InputBoxes and updates cell formatting (color and number format). It also involves interaction with data from "Plan1" and uses a collection to track changes.
* **`GerarCSV`**: Generates a CSV file from the data in `Plan2`, including a competency field derived from `Plan1`. It offers the user the option to change the file name and open the file's directory. It also deletes a folder after CSV generation.
* **`Funcoes`**: A collection of utility functions:
* `copiaDadosAreaDeTransferencia`: Copies a string to the clipboard.
* `test1`: Attempts to find and activate an Explorer window for a specified folder. If it can't find the window then it open a hyperlink to the folder.
* `CheckFolderOpen`: Checks if a folder is open in Windows Explorer and optionally closes it.
* `CheckPath`: Checks if a given path is a file, directory, or invalid.
* `OpenFolder`: Opens a specified folder in Windows Explorer.
* `changeFileName`: Prompts the user to change a file name before saving and offers option to cancel.
* `VerificaVersaoMacro`: Compares the current macro version with a version stored in a network file (`CONVICTA.txt`). If the network version is newer, it downloads and replaces the current macro, prompting the user for confirmation. It saves a backup of the existing workbook.
* `UserInfo`: Retrieves user information (username, computer name, domain) from environment variables.
* `CopiarArquivoServidor`: Copies files from a local folder to a network share, based on the file name prefix ('I' or 'O').
* `IncluiAcessoRotina`: Logs access information (username, computer name, timestamp, etc.) to a file and copies the files to the network share..
* `OpenExcelDoc`: Opens an Excel file.

Function Summaries

* **`Workbook_SheetActivate(ByVal Sh As Object)`**: Triggered when a sheet is activated. It toggles scroll bars and sets the `ScrollArea` property of the "Resumo" sheet.
* **`Workbook_Deactivate()`**: Restores scroll bars to their default state when the workbook is deactivated.
* **`Workbook_Open()`**: Similar to `Workbook_SheetActivate`, but executes when the workbook is initially opened. It handles the case where "Resumo" is already active on opening.
* **`mostra()`**: Copies the string "Haislan" to the clipboard.
* **`Worksheet_Change(ByVal Target As Range)`**: The main data processing function, triggered by cell changes in `Plan2`. Performs extensive validation and conditional logic on columns Q, R, and S. It handles user input and modifies associated values and cell formatting, particularly for financial input validation and data entry.
* **`gerar_csv()`**: Generates a CSV file from the data in `Plan2`, including data from `Plan1`.
* **`copiaDadosAreaDeTransferencia(valor As String)`**: Copies the input string to the clipboard.
* **`test1()`**: Searches for a specific folder in open Explorer windows; opens the folder in a new window if not found.
* **`CheckFolderOpen(strFolder, Optional closeFolder = True)`**: Checks if a folder is open in Explorer and optionally closes it. Returns `True` if open, `False` otherwise.
* **`CheckPath(path)`**: Checks if a path points to a file ("F"), directory ("D"), or is invalid ("I").
* **`OpenFolder(strFolder)`**: Opens the specified folder in Explorer.
* **`changeFileName(ByRef strPathFileName, Optional strTitleDialog, Optional strFilterFile)`**: Allows the user to change the name of a file before saving, providing file filter and title for the save dialog.
* **`VerificaVersaoMacro(Optional NumeroVersao As String)`**: Checks for a newer version of the macro on a network share and replaces the current macro if a newer version is found.
* **`UserInfo(Optional ByVal Prop As String = "USERNAME")`**: Returns the value of a specified environment variable; defaults to the username.
* **`CopiarArquivoServidor(pathMacroArquivo)`**: Copies files from a local folder to a network share, based on file naming conventions.
* **`IncluiAcessoRotina(Optional valor As String)`**: Logs routine access information to a file and copies log to the network share.
* **`OpenExcelDoc(pathFile As String)`**: Opens an Excel file.

Control Flow

* **`Worksheet_Change`**: This is the most complex function. Its control flow is primarily driven by the `If` statements checking the row number and the first two characters of the `Target.Address` (cell address) to determine which column has been modified. Nested `If` statements further refine the logic based on the cell's value, performing different actions (data validation, cell formatting, input box prompts, and updating other cells). The logic involves numerous checks to validate user input and control behavior based on various conditions. Loops are also used within conditional blocks to iterate through cells.

* **`VerificaVersaoMacro`**: The control flow involves checking the existence of a network file, reading its contents to extract version information,

comparing versions, and prompting the user to update if necessary. If the user chooses to update, it saves a backup of the workbook, copies the new macro file, and opens the new macro file. Error handling is used throughout to manage potential issues with file operations.

* **`GerarCSV`**: This function has a clear sequential flow. First it retrieves data from the workbook, then checks if the target file exists and creates it, and then it iterates through each row and writes the data into the CSV file. Finally, it prompts the user to open the generated file's location.

Data Structures

* **`Collection OldValues`**: Used in `Plan2` (commented out but present) to store the previous values of cells before they are changed, allowing for tracking of modifications.
* **Arrays:** Several functions use arrays (e.g., `Split` function to parse strings in `VerificaVersaoMacro`).
* **`DataObject`**: Used in `copiaDadosAreaDeTransferencia` to handle clipboard operations.

Malware Family Suggestion

While the code itself doesn't directly contain malicious payloads like shell commands or network connections to external resources, its behavior raises significant concerns. The extensive file manipulation capabilities, particularly the automatic download and replacement of the macro itself (`VerificaVersaoMacro`), and the logging of user activity and computer information (`IncluiAcessoRotina`) strongly suggest a potential **Trojan/Backdoor** behavior.

The code could be easily modified to include malicious actions. An attacker could embed malicious code within the update mechanism, replacing the legitimate macro with a backdoor that provides remote access to the affected system. Furthermore, the extensive data collection could be used to exfiltrate sensitive information. The use of obfuscation techniques (hex and base64 strings) also points towards attempts to hide the true nature of the code.

The overall design, with its complex data handling combined with the self-updating capabilities and extensive file I/O and logging to a network share, is highly suspicious and points towards a malicious intent, even without explicit malicious code present. Therefore, a classification as a **potentially malicious VBA macro with backdoor capabilities** is warranted. Further dynamic analysis in a sandbox environment would be needed to confirm the actual malicious behavior.