Analysis Report for: da.txt

Decoded using latin-1...

Overall Functionality

This VBA code appears to be designed for managing and manipulating data within an Excel workbook, likely for financial or accounting purposes. It includes features for controlling scroll bars on specific sheets ("Resumo"), data validation and manipulation on a worksheet named "Plan2," generating a CSV file based on "Plan2" data, and functions for interacting with the Windows file system and potentially other applications. The presence of numerous suspicious elements, however, strongly suggests malicious intent. The code exhibits characteristics consistent with a macro virus or a tool designed for data exfiltration or manipulation.

Function Summaries

* **`Workbook_SheetActivate(ByVal Sh As Object)`:** This event handler runs whenever a sheet is activated. It controls the visibility of scroll bars and sets the scroll area based on the activated sheet's name. If the sheet is "Resumo", scroll bars are hidden, and the scroll area is limited. Otherwise, scroll bars are shown.

* **`Workbook_Deactivate()`:** This event handler runs when the workbook is deactivated (e.g., closed or another workbook is selected). It ensures that scroll bars are turned back on for all sheets.

* **`Workbook_Open()`:** This event handler runs when the workbook is opened. It performs the same scroll bar and scroll area logic as `Workbook_SheetActivate` to handle the case where "Resumo" is already active on workbook opening.

* **`mostra()`:** This subroutine copies the string "Haislan" to the clipboard. Potentially a placeholder or a simple test function, but its simplicity raises concern.

* **`Worksheet_Change(ByVal Target As Range)`:** This event handler responds to cell changes in "Plan2." It contains complex logic involving data validation, conditional formatting, and input box prompts. Its functionality seems focused on enforcing data rules and potentially triggering actions based on specific data patterns in columns Q, R, and S. This function is the most suspicious part of the code.

* **`gerar_csv()`:** This subroutine generates a CSV file named "Retorno_*convenente_cod_convenente_competencia*.csv" containing data from "Plan2." It includes features to overwrite existing files and prompt the user to select a different file name. This function shows a strong indication of exfiltrating data to an external file.

* **`copiaDadosAreaDeTransferencia(valor As String)`:** This function copies the input string to the clipboard.

* **`test1()`:** This subroutine attempts to check if a specific folder ("Teste") is open in Windows Explorer. If not, it opens the folder in a new window.

* **`CheckFolderOpen(strFolder, Optional closeFolder = True)`:** This function checks if a specified folder is open in Windows Explorer. It can optionally close the folder.

* **`CheckPath(path)`:** This function checks if a given path is a file ("F"), directory ("D"), or invalid ("I").

* **`OpenFolder(strFolder)`:** This subroutine opens the specified folder in Windows Explorer.

* **`changeFileName(ByRef strPathFileName, Optional strTitleDialog, Optional strFilterFile)`:** This function prompts the user to change the filename before saving.

* **`VerificaVersaoMacro(Optional NumeroVersao As String)`:** This subroutine checks for updates of the macro itself by comparing the version and modification date with those stored in a network file. If a newer version exists, it copies the newer version to the current location.

* **`UserInfo(Optional ByVal Prop As String = "USERNAME")`:** This function retrieves the specified environment variable.

* **`CopiarArquivoServidor(pathMacroArquivo)`:** This function copies files from a local folder to a network folder. It appears to organize files based on naming conventions, separating "Entrada" and "Saida" files.

* **`IncluiAcessoRotina(Optional valor As String)`:** This subroutine logs access information (including the user, machine, and timestamp) to a file.

* **`OpenExcelDoc(pathFile As String)`:** This subroutine opens an Excel file.

Control Flow

The control flow is complex, particularly within `Worksheet_Change`. It heavily relies on `If` statements and nested loops to manipulate data based on the changed cell's location, value, and relationships with other cells. Error handling (`On Error Resume Next`) is used extensively, which could mask malicious actions. The `gerar_csv` function iterates through the rows and columns of Plan2 to build the CSV data, writing each line to a file. The `VerificaVersaoMacro` function uses nested loops to check a text file and then performs conditional actions to update the macro if necessary.

Data Structures

The main data structure is the Excel worksheet itself, acting as a table of data. The code also uses a `Collection` object (`OldValues`) in `Worksheet_Change` to store previous cell values. Arrays are used in a couple of functions for splitting strings.

Malware Family Suggestion

The combination of the following characteristics strongly suggests this code is part of a malicious macro:

* **Data Exfiltration:** The `gerar_csv` function explicitly creates and saves a CSV file containing data from the workbook; this points to potential data theft.
* **File System Manipulation:** The code extensively interacts with the file system (copying files, deleting folders).
* **Environment Variable Access:** The use of `Environ` could be used to gather system information, aiding in identifying the target system.
* **Clipboard Manipulation:** The `copiaDadosAreaDeTransferencia` function suggests an attempt to exfiltrate data using the clipboard.
* **Self-Update Capability:** The `VerificaVersaoMacro` function allows the macro to update itself, making it harder to remove and potentially introducing new malware.
* **Obfuscation:** While not overtly obfuscated, the complexity and extensive use of error handling can make the code harder to analyze.
* **Hidden Functionality:** Commented-out code exists within `Worksheet_Change` indicating possible previously active, now-hidden, malicious routines.
* **Network Access:** Access to network shares points towards an external command-and-control server or data storage location.

Based on the above, it's highly probable this code is part of a **macro virus** or a more sophisticated piece of malware designed for data theft and potentially further system compromise. The self-updating capability elevates the threat level. The data exfiltration aspects suggest it could be related to **financial malware**, aimed at stealing sensitive financial information. Specific categorization (e.g., banking trojan, information stealer) would require further analysis.