Analysis Report for: 5c.txt

Decoded using latin-1...

Overall Functionality

This VBA macro, embedded within an Excel file, appears designed to process data, likely from a database or external source, and then generate a formatted report in Excel, including pivot tables, and finally outputting the processed data to a text file. The macro extensively manipulates Excel sheets, ranges, and pivot tables. It also contains functions for formatting, data manipulation, and file I/O. The error handling is rudimentary, simply resuming execution on error 1004, suggesting it may not handle all potential errors gracefully. The suspicious features detected by olevba highlight potential malicious behavior.

Function Summaries

* **`Formatear()`**: This is the main subroutine. It orchestrates the entire data processing and report generation workflow. It handles selecting data sheets, determining data ranges, copying formulas, creating named ranges, generating tables (or adding borders in older Excel versions), refreshing pivot tables, updating pivot table data sources, formatting the output, and finally creating a text file.

* **`Marco(rango As String)`**: Adds borders to a specified cell range (`rango`). It applies both thin and medium weight borders.

* **`FilaTexto(columna As String, Texto As String, Optional hoja As String, Optional iteracion As Integer) As Integer`**: Searches for a specific text (`Texto`) within a given column (`columna`) and optional sheet (`hoja`), returning the row number of the first occurrence (or subsequent occurrences if `iteracion` is greater than 1).

* **`ConfigurarPagina(hoja As String, Optional ancho As Integer, Optional apaisado As Boolean, Optional filaTitulos As String)`**: Configures page setup for printing, including orientation (`apaisado`), number of pages wide (`ancho`), and print titles (`filaTitulos`).

* **`ArrastrarFormula(celda As String, Optional hoja As String)`**: Autofills a formula from a given cell (`celda`) down to the last row (`filaFin`) in the specified sheet (`hoja`).

* **`Desbloquear(hoja As String, area As String)`**: Unlocks specified cells (`area`) on a given sheet (`hoja`).

* **`ProtegerHoja(hoja As String)`**: Protects a sheet (`hoja`) without a password, allowing only unlocked cells to be edited.

* **`CrearHoja(nomHoja As String)`**: Creates a new Excel sheet with a given name (`nomHoja`).

* **`EliminarHoja(nomHoja As String)`**: Deletes an Excel sheet with a given name (`nomHoja`).

* **`Rellenar(campo As String, longitud As Integer, relleno As String, izquierda As Boolean)`**: Pads a string (`campo`) with a specified character (`relleno`) to a given length (`longitud`), either on the left or right depending on (`izquierda`).

* **`InsertarFila(fila As Integer)`**: Inserts a row at a specified position (`fila`).

* **`EliminarFila(fila As Integer, Optional hoja As String)`**: Deletes a row at a specified position (`fila`) on an optional sheet (`hoja`).

* **`ColorFondo(rango As String, Color As String)`**: Sets the background color of a specified range (`rango`) to one of three predefined colors ("verde", "turquesa", "canela").

* **`CopiarCeldas(hojaOrigen, rangoOrigen, hojaDestino, celdaDestino)`**: Copies values from a source range (`rangoOrigen`) on a source sheet (`hojaOrigen`) to a destination cell (`celdaDestino`) on a destination sheet (`hojaDestino`).

* **`FechaFinMes(fecha As Date)`**: Returns the last day of the month for a given date (`fecha`).

* **`MoverElementoTD(hojaTabla, nomTabla, campo, elemento, Optional final As Boolean)`**: Moves a pivot table item (`elemento`) in a specified field (`campo`) of a pivot table (`nomTabla`) on a sheet (`hojaTabla`) to either the beginning or end.

* **`CreacionFichero()`**: Creates a text file containing data extracted from the Excel sheet. The filename and path are obtained from the user via an input box. The function uses `CreateObject("Scripting.FileSystemObject")`, indicating file system access.

* **`OcultarHoja(nomHoja As String)`**: Hides an Excel sheet.

* **`EliminarFilasVacias(filaFin As Long)`**: Deletes rows from the bottom up that are completely empty.

Control Flow

The `Formatear()` subroutine is the main control flow. It follows a sequential structure with several loops:

1. **Initial Setup:** Sets application settings (calculation mode, alerts). Obtains sheet names from named ranges. Error handling for missing ranges

is minimal (Resume Next).

2. **Data Sheet Processing:** Selects the data sheet, detects the number of rows and columns, hides a specific column, and defines the data range.

3. **Formula Copying Loop:** Iterates through columns, checks if a cell contains a formula, and if so, uses the `ArrastrarFormula` function to copy it down.

4. **Table Creation:** Creates a named range for the data and either creates an Excel table (Excel 2007 and later) or adds borders (older Excel versions). Autofits columns.

5. **Pivot Table Processing Loop:** Iterates through sheets, then through pivot tables in each sheet, refreshing them and updating their data sources. Replaces "(blank)" values in pivot tables with spaces. Autofits columns and configures page setup using `ConfigurarPagina` for each sheet containing pivot tables.

6. **Final Formatting:** Selects the "LOTE" sheet, copies and pastes values, clears some rows, creates another table, and autofits columns.

7. **Cleanup:** Deletes the data sheet.

8. **Error Handling:** Very basic error handling (only handles error 1004).

The other functions have relatively straightforward control flows, mostly involving simple conditional statements and loops where necessary.

Data Structures

The primary data structures are those inherent to Excel:

* **Worksheets:** Excel sheets are used to store and manipulate data.
* **Ranges:** Cell ranges are used extensively for data selection, manipulation, and formatting.
* **PivotTables:** Pivot tables are used for data summarization and analysis.
* **Named Ranges:** Used to store references to specific data ranges or cells, simplifying access within the VBA code.

Malware Family Suggestion

The functionality, combined with the suspicious elements flagged by olevba (CreateTextFile, CreateObject, Chr function, hex and base64 strings), suggests that this code could be part of a **macro virus** or a tool used in a **data exfiltration** attack. The generation of a text file, especially with user-specified filename and path through an input box, is highly suspicious, as is the creation of an OLE object. While it performs data processing and report generation tasks that appear benign, the method of data output and the lack of robust error handling raises red flags. The potential to execute arbitrary code through the "Call" and "CreateObject" functions increases the risk significantly. The presence of obfuscation techniques (hex strings, etc.) strengthens the suspicion of malicious intent. The analysis is inconclusive without execution and network traffic analysis. The macro should be analyzed in a sandbox environment to determine its true behavior.