Analysis Report for: aa.txt

Decoded using latin-1...

Overall Functionality

This VBA macro is designed to process data within an Excel spreadsheet. It appears to be intended for automating report generation or data transformation tasks. The macro reads data from a source sheet ("HojaDatos"), processes and formats it, and then populates a destination sheet ("LOTE"). It also updates the data source of pivot tables on other sheets to reflect the processed data, handles potential errors gracefully, and offers some configurability through user input and environment variables. The presence of file I/O operations (creation of a text file) and the use of `Environ` suggest potential for malicious modification, particularly if the input data or the file name are not properly sanitized.

Function Summaries

* **`Formatear()`**: This is the main subroutine. It orchestrates the entire data processing and formatting process, calling other functions and subroutines as needed. It takes no arguments.
* **`ArrastrarFormula(celda As String, Optional hoja As String)`**: This function autofills formulas down a column in the specified sheet (defaults to "Hoja1"). It extracts the column letter from the input `celda` string.
* **`Marco(rango As String)`**: This subroutine adds a border to a specified range of cells.
* **`FilaTexto(columna As String, Texto As String, Optional hoja As String, Optional iteracion As Integer)`**: This function searches for a specific text within a specified column and sheet (defaults to "Hoja1"), returning the row number of the first occurrence. Allows for searching multiple times.
* **`ConfigurarPagina(hoja As String, Optional ancho As Integer, Optional apaisado As Boolean, Optional filaTitulos As String)`**: This subroutine configures the page setup for printing, allowing specification of page width, landscape orientation, and header rows.
* **`Desbloquear(hoja As String, area As String)`**: This subroutine unlocks specified cells on a worksheet.
* **`ProtegerHoja(hoja As String)`**: This subroutine protects the specified worksheet, preventing accidental modification.
* **`CrearHoja(nomHoja As String)`**: This subroutine creates a new worksheet with the specified name.
* **`EliminarHoja(nomHoja As String)`**: This subroutine deletes a worksheet.
* **`Rellenar(campo As String, longitud As Integer, relleno As String, izquierda As Boolean)`**: This function pads a string to a specified length, either on the left or right.
* **`InsertarFila(fila As Integer)`**: This subroutine inserts a row at the specified location.
* **`EliminarFila(fila As Integer, Optional hoja As String)`**: This subroutine deletes a row.
* **`ColorFondo(rango As String, Color As String)`**: This subroutine sets the background color of a cell range.
* **`CopiarCeldas(hojaOrigen, rangoOrigen, hojaDestino, celdaDestino)`**: This subroutine copies cell values from one location to another.
* **`FechaFinMes(fecha As Date)`**: This function calculates the last day of the month for a given date.
* **`MoverElementoTD(hojaTabla, nomTabla, campo, elemento, Optional final As Boolean)`**: This subroutine moves a PivotTable item to the beginning or end of a field.
* **`CreacionFichero()`**: This subroutine prompts the user for a file name and creates a text file, populating it with data extracted from the Excel sheet. This subroutine contains suspicious operations.
* **`OcultarHoja(nomHoja As String)`**: This subroutine hides a worksheet.
* **`EliminarFilasVacias(filaFin As Long)`**: This subroutine deletes empty rows from a sheet.

Control Flow

The `Formatear()` subroutine is the heart of the macro. Its control flow involves:

1. **Error Handling**: An `On Error GoTo` statement handles potential errors, specifically run-time error 1004 (typically related to object manipulation in Excel).
2. **Debugging Check**: It checks the `DEPURAR_CONSULTAS` environment variable to allow interactive confirmation before proceeding.
3. **Sheet and Data Acquisition**: It dynamically determines the sheet name containing the data based on named ranges ("nFilas").
4. **Data Range Determination**: It calculates the dimensions of the data range based on "nFilas" and hides the column containing "nFilas".
5. **Formula AutoFill**: It iterates through the columns, checks if a cell contains a formula, and calls `ArrastrarFormula()` to autofill the formula down to the calculated last row.
6. **Named Range Creation**: It creates a named range ("rango") encompassing the processed data.
7. **Table Creation or Bordering**: It conditionally creates either an Excel table (for Excel 2007 and later) or adds borders (for older versions).
8. **Autofit Columns**: It automatically adjusts column widths.
9. **Pivot Table Update**: It iterates through sheets, finds pivot tables, and updates their source data using the newly created named range. It also replaces "(blank)" values in PivotTables with spaces.
10. **Final Formatting**: It copies values, clears some rows, creates a table on the "LOTE" sheet, and formats a column as date.
11. **Data Source Deletion**: It deletes the source data sheet.
12. **Alert Reset**: It re-enables alerts.

The other functions generally follow a straightforward control flow, executing specific operations based on their input parameters. Noteworthy is `CreacionFichero()`, which takes user input to determine the output file path and name. This presents a significant security risk as malicious input could be used to overwrite arbitrary files.

Data Structures

The primary data structure used is the Excel spreadsheet itself. Named ranges are used to store intermediate results and data references ("nFilas", "rango"). The macro utilizes variables to store various parameters, such as sheet names, column numbers, row numbers, and cell values.

Malware Family Suggestion

While the provided code doesn't contain obviously malicious payload like shell commands or network connections, the presence of several suspicious features suggests that it *could* be part of a malware distribution or obfuscation scheme:

* **Use of `Environ()`**: Accessing environment variables is frequently used to adapt malware to different systems or to evade detection.
* **`CreateTextFile()`**: Creating a text file in an arbitrary location based on user input is a common tactic to spread a malware component or a secondary stage payload. The path could easily be overwritten with malicious input, leading to arbitrary file creation or overwrite.
* **The absence of direct malicious action**: doesn't exclude this code from being a part of a larger malicious attack. It might be a tool used to manipulate data before a more destructive action takes place. This code acts as a dropper, or sets up the environment for malicious action.
* **Potential for data exfiltration:** It's conceivable that the text file created by `CreacionFichero()` could be designed to exfiltrate data, or that the input data to the macro might itself be malicious.

Therefore, based on its functionalities, it is highly suggestive that the VBA macro is a **Trojan-like downloader**. Further analysis, including inspection of related files and examining the input data used with the macro would be needed to confirm its malicious nature definitively. The IOC `xvars.js` suggests this may be related to a Javascript component, either deployed by the macro, or a relevant external indicator. A static analysis alone is not sufficient to make this determination. Dynamic analysis would be required to definitively assess the level and nature of its maliciousness.