Analysis Report for: a.vbs

Decoded using latin-1...

Overall Functionality

This VBA code appears to be designed for managing and manipulating data within a Microsoft Excel workbook, likely for some kind of project or task tracking system. The code includes several subroutines ("functions" in VBA terminology) categorized into modules. These modules handle different aspects of data management, such as:

* **Data Input/Output:** Functions for saving the workbook with a custom filename (`Speichern`), importing data from other Excel files (`ImportSK`, `ImportCL`), and exporting data to a database sheet (`Datenbank`, `AA_DB`).
* **Data Manipulation:** Functions to delete data from specific ranges in the workbook (`Lösch_K`, `Lösch_C`, `BT01_L` to `BT16_L`, `BT1lösch` to `BT6lösch`, `Tiefbauer_Löschen`), and to insert data (`BT01_P` to `BT15_P`, `Tiefbauer`). Data is also copied between ranges and sheets.
* **Worksheet Management:** Functions to set print areas (`Seite1`, `Seite2`) and restore original cell formatting (`Orginalformat`). The code also manages the visibility and creation of worksheets.
* **User Interaction:** The code makes extensive use of `MsgBox` functions to prompt the user for confirmation before performing actions. An `InputBox` is used in one macro to obtain user input for a Bauteil (component) number.
* **Hidden Functionality and Obfuscation:** The presence of numerous hidden sheets and the complex structure of data manipulation functions suggest an attempt to obfuscate the true purpose of the code. The password-protected sheets ("ddookkuu1234") enhance the obfuscation.

The code's organization suggests a larger application built around managing multiple "Bauteile" (components), possibly within a project context.

Function Summaries

The code contains many functions. Summarizing all of them individually would be excessively long. However, a few key examples illustrate the patterns:

* **`Speichern()`:** Saves the active workbook with a filename constructed from cell values. No parameters; no explicit return value (implicitly returns success or failure based on `Ergebnis`).
* **`Lösch_K()`:** Clears contents of specified cell ranges after prompting user confirmation. No parameters; no return value.
* **`BT01_L()` to `BT16_L()`:** Delete a "Bauteil" (component). Each function deletes a different set of ranges, corresponding to a specific component. No parameters; no return value.
* **`Zeile_17_12()` to `Zeile_87_82()`:** These functions copy data from one set of cells to another. Each function handles a different pair of row ranges. No parameters; no return value.
* **`BT01_P()` to `BT15_P()`:** Insert a "Bauteil" (component). Each function inserts data into different row ranges, corresponding to a specific component. No parameters; no return value.
* **`Datenbank()` and `AA_DB()`:** These seem to be redundant functions that copy data from various cells into a worksheet named "Datenbank". No parameters; no return value.
* **`ImportSK()`:** Imports data from a user-selected Excel file into the "Daten" sheet. No parameters; no explicit return value (error handling present).
* **`ImportCL()`:** Similar to `ImportSK()`, but imports into the "Berechnung" sheet. No parameters; no explicit return value (error handling present).
* **`BilderImport()`:** Imports images from user-selected files and places them into pre-defined ranges on the active sheet. No parameters, no return value. Handles different image insertion methods based on Excel version.

Control Flow

The control flow is largely straightforward in most functions. Many functions follow this pattern:

1. **User Prompt (Optional):** A `MsgBox` asks for confirmation before proceeding.
2. **Data Extraction:** Values are read from various cells using `Range()` and `.Value`.
3. **Data Manipulation:** Data is cleared (`Selection.ClearContents`), copied, pasted, or replaced using `ActiveCell.FormulaR1C1`, `Selection.Copy`, `ActiveSheet.Paste`, and `ActiveCell.Replace`.
4. **Data Insertion (Optional):** Values are written to specified cells.
5. **Worksheet Navigation (Optional):** The code switches between worksheets using `Sheets("SheetName").Select`.
6. **User Feedback (Optional):** A `MsgBox` provides feedback on completion.

The `Speichern()` function has a conditional path based on the result of `Application.GetSaveAsFilename`. The `BT01_L()` to `BT16_L` functions, and `BT01_P()` to `BT15_P()` functions contain conditional user interaction. The `BilderImport()` function branches based on the Excel version. The `Tiefbauer()` and `Tiefbauer_Löschen()` functions have nested `If` statements to handle different scenarios based on cell values. The `BTeinf()` and `BTlös()` functions use a `Do While` loop to repeatedly prompt the user until valid input is received. `FOTO` demonstrates error handling with `On Error GoTo` statements

Data Structures

The primary data structure is the Excel spreadsheet itself. The code utilizes named ranges to refer to specific cells and groups of cells, implicitly creating a structured representation of the data. There are no explicitly defined arrays or other complex data structures within the VBA code, except

for the `arrBereiche` array used in `BilderImport()`.

Malware Family Suggestion

While this code itself is not inherently malicious, its functionality raises strong suspicions of being used as a component in a more extensive malicious application or macro virus. Several factors contribute to this suspicion:

* **Data Exfiltration:** The `Datenbank` functions strongly suggest data exfiltration capabilities. The data being collected could be sensitive information.
* **File System Access:** The `ImportSK` and `ImportCL` functions, combined with the custom filename generation in `Speichern`, indicate the potential for arbitrary file access and manipulation.
* **Obfuscation and Hidden Functionality:** The significant effort to obfuscate the code's intent, the use of numerous hidden sheets, and the password protection ("ddookkuu1234") are classic obfuscation techniques used by malware authors.
* **Self-Modifying Behavior:** The functions dynamically construct file names. This characteristic is common in malware that tries to avoid static detection.
* **Arbitrary Code Execution:** The `Application.Run` calls allow the execution of arbitrary VBA procedures, which could be loaded from external sources (e.g., from a network share), creating an attack vector.

Based on these characteristics, this VBA code would most likely be classified as a component of a **macro virus** or a **malicious document**. It's not a complete standalone malware; it's designed to be part of a larger attack. The specific type of macro virus would depend on what other actions the called functions or external resources perform. The file system access and data exfiltration capabilities make it potentially dangerous.