Analysis Report for: 0d

Overall Functionality

This VBA macro code, embedded within an OLE file, appears to be designed to export data from a spreadsheet (likely Excel) to a text file named "c:\orders". The macro is triggered by a CommandButton click (AutoExec). It checks for the presence of data in specific cells (representing company name, code, load date, and delivery date). If any of these cells are empty, it displays a message box prompting the user to fill in the missing information. If all cells are populated, it iterates through a portion of the spreadsheet (cells from 1,1 to 41,17), writing the contents of each cell to the "c:\orders" file.

Function Summaries

The code only contains one subroutine:

* **`CommandButton1_Click()`**: This subroutine is the main function. It's automatically executed when the associated command button is clicked. It doesn't have any parameters and doesn't return a value (it's a `Sub` procedure). Its purpose is to export spreadsheet data to a file based on the presence of data in four key cells.

Control Flow

The `CommandButton1_Click()` subroutine's control flow is primarily driven by nested `If` statements:

1. **Initial Checks:** It checks if cells (5,4), (6,4), (8,4), and (10,4) contain data. Each check leads to a different branch.
2. **Data Export:** If all four cells are non-empty, the code opens "c:\orders" for output (`For Output As #1`), then uses nested loops to iterate through cells (1,1) to (41,17). `Print #1, Cells(i, h);` writes the cell content to the file, followed by `Print #1,` which adds a newline character. Finally, it closes the file (`Close #1`).
3. **Error Handling (Message Boxes):** If any of the four initial cells are empty, a `MsgBox` function displays an error message prompting the user to provide the missing data. The message indicates which field is missing (company name, company code, load date, or delivery date).

Data Structures

The primary data structures used are:

* **Spreadsheet Cells:** The VBA code interacts directly with spreadsheet cells using the `Me.Cells(row, column)` notation. The spreadsheet itself acts as the main data structure.
* **File:** The "c:\orders" file is used as a simple text file for storing the exported data. The `#1` identifier represents the file handle.

Malware Family Suggestion

While this code doesn't directly exhibit malicious behavior like self-replication or network communication, its functionality raises some concerns:

* **Data Exfiltration:** The code exports data to a file on the hard drive. A malicious actor could modify this code to exfiltrate sensitive data from the spreadsheet to a remote location (e.g., by replacing "c:\orders" with a network path).
* **Potential for Further Modification:** The relatively simple structure of the code makes it easily modifiable for malicious purposes. A more sophisticated malware could be easily inserted within its current structure.
* **Steganography:** The file might contain other malicious components not visible in the provided VBA code.

Therefore, the code's functionality strongly suggests a potential for use as a **data exfiltration tool** within a larger malware campaign. It could also be a component in a more complex malware infection that uses it for stealing information later on. A thorough analysis of the entire OLE file (beyond the VBA code) would be necessary to definitively determine its true nature. The presence of "hex strings" (mentioned in the olevba report) further increases suspicion. These hex strings could be used to encode malicious commands or data that are not immediately visible. Decoding them would be crucial.