Analysis Report for: 43.txt

Overall Functionality

The provided code is VBA (Visual Basic for Applications) code embedded within an Excel (.xls) file. It doesn't contain malicious code in the traditional sense (like shellcode execution or network communication). Instead, it implements event handlers that restrict user interaction within specific cells of an Excel spreadsheet. Specifically, it prevents editing or selection of cells in column A (rows 2 and beyond) and column H (rows 5 and beyond) by displaying a message box indicating that these cells are read-only. When a user attempts to select such cells, the code beeps, moves the selection to the adjacent cell to the right and shows the "read-only" message box.

Function Summaries

The code consists of a single VBA subroutine within the `Sheet1.cls` module:

* **`Worksheet_SelectionChange(ByVal Target As Range)`**: This is an event handler that triggers automatically whenever the user changes the selected cell(s) in the worksheet.
* **Parameters:** `Target As Range` – Represents the range of cells that were newly selected.
* **Return Value:** None (Subroutine).

Control Flow

The `Worksheet_SelectionChange` subroutine's logic is primarily based on nested `If` statements:

1. **Column Check:** It first checks if the selected cell's column (`Target.Column`) is 1 (column A) or 8 (column H).

2. **Row Check (Column A):** If the column is 1 (A), it further checks the row number (`Target.Row`):
* **Row 2:** If the row is 2, it beeps, moves selection one cell to the right (`Offset(0, 1).Select`), and displays a message box indicating that the cell is read-only.
* **Rows > 4:** If the row is greater than 4, it performs the same actions (beep, move selection, display message box).

3. **Row Check (Column H):** If the column is 8 (H), it checks if the row number is greater than 4. If true, it performs the same actions as above (beep, move selection, display message box).

The code's control flow is straightforward, involving simple conditional checks and sequential execution of actions within each branch.

Data Structures

The code primarily utilizes the built-in data structures of VBA and Excel:

* **`Range` object:** This object represents a cell or a selection of cells in the Excel worksheet. The `Target` parameter of the event handler is a `Range` object.
* **Implicit data structures within Excel:** The Excel spreadsheet itself serves as a two-dimensional array of cells, implicitly used by the code through the `Cells` property and the `Offset` method.

Malware Family Suggestion

This code is **not malicious** in the traditional sense. It doesn't exhibit any behaviors characteristic of known malware families. The functionality is consistent with benign macro code intended to protect specific spreadsheet cells from accidental modification. The use of "Kutools for Excel" in the message box suggests it might be part of an add-in or a custom tool designed to restrict access to certain parts of the spreadsheet. While obfuscation techniques are sometimes used in malware, the code presented here is not obfuscated and its behavior is clearly defined. The presence of hex strings, noted by olevba, requires further investigation but is not inherently malicious without context. However, a file with this behavior might be bundled with malicious code in other parts of the file; this analysis only covers the VBA macro.