Analysis Report for: d3.txt

Overall Functionality

The provided VBA code within an Excel file defines a single macro, `URLPictureInsert`. This macro iterates through cells A4 to A36 of the active worksheet. For each cell, it assumes the cell's value is a filename, inserts a picture from that file into the worksheet, and then resizes and centers the picture within a cell to the right of the filename cell. The macro uses error handling (`On Error Resume Next`) and attempts to gracefully handle cases where a picture insertion fails. The other macros (`List1` and `ThisWorkbook`) are empty.

Function Summaries

* **`URLPictureInsert()`**: This is the only substantive subroutine. It takes no parameters and returns no value (Subroutine). Its purpose is to insert and position pictures from files listed in a worksheet column.

Control Flow

The `URLPictureInsert` subroutine's control flow is as follows:

1. **Initialization:** It initializes variables (`Pshp`, `xRg`, `xCol`). Error handling is enabled using `On Error Resume Next`. Screen updating is turned off for performance.
2. **Iteration:** It loops through each cell in the range A4:A36 using a `For Each` loop.
3. **Picture Insertion:** Inside the loop, it attempts to insert a picture using the cell value as the filename (`ActiveSheet.Pictures.Insert(filenam).Select`).
4. **Error Handling:** If `Pshp` (the Shape object representing the inserted picture) is `Nothing` (meaning insertion failed), it jumps to the `lab:` label, skipping the resizing and positioning steps.
5. **Resizing and Positioning:** If the picture insertion was successful, it calculates the column for placement (`xCol`), gets the target cell (`xRg`), and then resizes the picture to at most 2/3 the width and height of `xRg`, centering it within `xRg`.
6. **Cleanup:** It resets `Pshp` and selects cell A4.
7. **Loop Continuation:** The loop continues to the next cell.
8. **Finalization:** After the loop, screen updating is turned back on.

Data Structures

The code primarily uses simple data structures:

* **Variables:** It uses several variables: `Pshp` (Shape object), `xRg` (Range object), `xCol` (Long integer), `filenam` (String), `Rng` (Range object), and `cell` (Range object).
* **Range Objects:** The code extensively uses Excel's `Range` object to represent cell selections and ranges of cells.
* **Shape Object:** The `Shape` object represents the inserted picture.

Malware Family Suggestion

While the code itself is not inherently malicious, its functionality is suspicious. The macro's behavior of inserting pictures from files listed in a spreadsheet raises concerns. An attacker could leverage this to:

* **Social Engineering:** The filenames could be cleverly disguised to entice a user to open the spreadsheet and potentially execute further malicious code (e.g. through embedded malicious images).
* **Image-Based Malware Delivery:** The picture files themselves could contain malicious code or exploits, which may be triggered upon insertion or viewing.
* **Data Exfiltration:** The spreadsheet could contain a list of files to be accessed and possibly exfiltrated to a remote location.

Therefore, while the code doesn't directly demonstrate classic malware techniques like network communication or file system manipulation, its functionality aligns with the capabilities of a downloader or dropper, used as a first stage in a more complex attack. It is not possible to classify it definitively as belonging to a specific malware family without further context, but its potential for abuse suggests caution. The use of `On Error Resume Next` is also a common obfuscation technique, used to hide errors and make analysis more difficult. A complete analysis would require inspecting the files referenced in the spreadsheet to confirm the presence or absence of actual malicious content.