Analysis Report for: 58886729BF31372C8D2AF3244B678CC4.exe

Decoded using latin-1...

This code is written in VBScript, not C. The `On Error Resume Next` statement is a clear indicator of this. Let's analyze it as VBScript code.

Overall Functionality

The VBScript code aims to uninstall an Excel add-in ("Excel 2003[redacted].xlam"). It first prompts the user for confirmation. Then, it attempts to remove the add-in from Excel, deletes the add-in file from the user's AppData folder, and finally, provides a message box indicating success or failure. The add-in's name is hardcoded, suggesting a targeted action rather than a general add-in removal utility. The Japanese characters in the code suggest the target audience and likely the add-in itself is Japanese.

Function Summaries (VBScript implicitly defines functions)

The code doesn't define functions in the traditional sense, but uses built-in VBScript objects and methods. Here's a summary of the key components:

* **`MsgBox()`**: Displays a message box to the user with a specified message and buttons (YesNo, OK, etc.). Returns the button the user clicked.
* **`CreateObject()`**: Creates an instance of a COM object (e.g., Excel Application, FileSystemObject, WScript.Shell).
* **`objExcel.Addins.Count`**: Returns the number of add-ins currently installed in Excel.
* **`objExcel.Addins.item(i)`**: Returns the i-th add-in object.
* **`objAddin.Name`**: Returns the name of an Excel add-in.
* **`objAddin.Installed`**: A property that controls whether an add-in is installed (True/False).
* **`objExcel.Quit()`**: Quits the Excel application.
* **`objWshShell.SpecialFolders("Appdata")`**: Returns the path to the user's AppData folder.
* **`objFileSys.FileExists()`**: Checks if a file exists at a given path.
* **`objFileSys.DeleteFile()`**: Deletes a file.
* **`Err.Number`**: Contains the error code if an error occurred.

Control Flow

1. **Confirmation Dialog:** The script starts with a `MsgBox` asking the user if they want to uninstall the add-in. If the user clicks "No," the script exits using `WScript.Quit`.

2. **Excel Add-in Removal:** A loop iterates through all installed Excel add-ins. If an add-in with the specified name (`addInFileName`) is found, its `Installed` property is set to `False`, effectively uninstalling it.

3. **Excel Quit:** The Excel application is closed using `objExcel.Quit()`.

4. **Add-in File Deletion:** The script attempts to delete the add-in file from the user's AppData folder. It checks if the file exists before attempting deletion. If the file is not found, an error message is displayed.

5. **Error Handling:** The `On Error Resume Next` statement attempts to suppress errors. The script checks `Err.Number` at the end; if it's non-zero, an error message is displayed. This is poor error handling. It doesn't provide specific details about what went wrong.

Data Structures

The script primarily uses simple variables to store strings (paths, names) and a COM object to interact with the file system and Excel. There are no complex data structures.

Malware Family Suggestion

While not strictly malware, the script's functionality and design choices raise concerns. The targeted removal of a specific add-in, combined with the poor error handling (`On Error Resume Next`), suggests a potential for malicious use:

* **Targeted Add-in Removal:** A legitimate software uninstaller would generally offer more options or a more robust interface for selecting add-ins. This script's targeted behavior raises suspicion.

* **Poor Error Handling:** The `On Error Resume Next` statement masks errors, preventing the user from understanding if the uninstallation failed. A malicious actor could use this to hide the failure of an operation critical to their malicious goals.

* **Potential for Abuse:** A malicious actor could modify this script to remove legitimate add-ins or other critical files.

Therefore, while the code itself is not inherently malicious, its functionality and coding practices align with characteristics of a **potentially unwanted program (PUP)** or a component of more significant malware. It's a script that could be easily repurposed for malicious purposes by changing the target add-in. A sophisticated attacker might leverage this type of technique to make an infection more persistent.