Analysis Report for: 67.vbs

Overall Functionality

This VBA macro code interacts with a Microsoft Access database to populate a Word document table with information based on database query results. Specifically, it has two main functions: `DocResponse` and `DrwResponse`. Both functions check if the Word document is opened from a specific path ("G:\Site Databases\Projects task database" or "G:\NewTasks\Tasks2003.mdb"). If so, they query an Access database (`Tasks2003.mdb` at different paths), retrieve data based on the query, and then populate the first table in the active Word document with that data. If no data is found, a "NOTHING TO REPORT!" message is displayed in the table. The `Document_Open` subroutine calls `DocResponse` when the document is opened. The database interaction involves using ADO (ActiveX Data Objects) to connect to and query the database.

Function Summaries

* **`DocResponse()`**: This function queries the `[DocResponse]` table in the Access database located at "G:\Site Databases\Projects task database\Tasks2003.mdb". It retrieves data and populates a Word document table with project information, document title, reference, and revision. It returns no value (implicitly `void`).

* **`DrwResponse()`**: Similar to `DocResponse()`, but queries the `[DrwResponse]` table in the database located at "G:\NewTasks\Tasks2003.mdb". This function populates the Word document table with drawing-specific information. It also returns no value.

* **`Document_New()`**: This subroutine is empty and does nothing. It's an event handler that triggers when a new Word document is created.

* **`Document_Open()`**: This subroutine is called when the Word document opens. It calls the `DocResponse()` function.

Control Flow

* **`DocResponse()` and `DrwResponse()`:** Both functions follow a similar control flow:
1. **Path Check:** They first check if the Word document is opened from the expected path. If not, they exit.
2. **Database Connection:** They open a connection to the specified Access database.
3. **Query Execution:** They execute a SQL query to retrieve data.
4. **Data Check:** They check if the recordset is empty (`mSet.EOF`). If empty, they display a "NOTHING TO REPORT!" message in the Word document table and exit.
5. **Data Population:** If data is found, they iterate through the recordset (in this case, only processing the first record) and populate the Word document's table with the retrieved data, cell by cell.
6. **Database Closure:** They close the recordset and database connection.

* **`Document_Open()`:** This subroutine simply calls the `DocResponse()` function.

Data Structures

* **`Range`, `Table`, `Cell`:** These are Word objects used to manipulate the content of the Word document's table.
* **`Database`, `Recordset`:** These are ADO objects used to interact with the Access database. The `Recordset` holds the data retrieved from the database query.
* **Strings:** Several string variables (`Swap`, `SQL`, `sDataPath`, etc.) are used to store data, SQL queries, and file paths.
* **Integers:** Integers (`nTable`, `c`) are used for indexing and looping.

Malware Family Suggestion

While this code isn't inherently malicious, its functionality exhibits characteristics commonly seen in malicious macros. The key risks are:

* **Data Exfiltration:** The macro retrieves data from a database and potentially inserts it into a document. A malicious variant could replace this database with a remote source, subtly exfiltrating data from the victim's system.

* **Arbitrary Code Execution:** Although not present here, the framework could be adapted to execute malicious code instead of simply populating a Word document. The use of `DoEvents` might be leveraged for timing attacks or hiding actions from the user.

* **Hidden Execution:** The fact that this macro runs automatically upon opening a document makes it stealthy; the user might not even notice its activity.

* **Path-Based Conditional Logic:** The conditional check of the document's path (`ActiveDocument.Path`) suggests potential targeting of specific users or systems. A sophisticated attacker might modify this check to make it very specific, and then trigger actions only when certain conditions are met, for example if it's running on specific machine.

Overall, this code demonstrates behavior consistent with information-stealing malware or a potential delivery mechanism for a more complex malicious payload. While harmless in its current form, it highlights the risk of untrusted VBA macros. The use of absolute paths in the database connections is also a significant security risk. A more robust solution would avoid hardcoded paths and use registry keys or other more flexible mechanism to locate the database.