Analysis Report for: 309FB5E60CBB906488773278C27CE897.cs

Overall Functionality

This C# codebase implements a custom PowerShell host that interacts with the user through Windows Forms dialogs. It's designed to run a PowerShell script (onedrive_check_smart_v7.ps1) embedded as a resource within the application. The script's functionality isn't directly revealed in this code, but the host provides input/output mechanisms and handles user credentials securely. The code includes functions for displaying input boxes, choice boxes, and progress bars, all within a custom graphical user interface (GUI). The application can also extract the embedded PowerShell script to a file if a command-line argument is provided.

Function Summaries

* **`Choice_Box.Show(Collection arrChoice, int intDefault, string strTitle, string strPrompt)`**: Displays a dialog box with radio buttons based on the provided `ChoiceDescription` collection. The user selects an option. Returns the index of the selected choice (or -1 if cancelled or an error).

* **`Console_Info.IsInputRedirected()`**: Checks if standard input is redirected (e.g., from a file). Returns `true` if redirected, `false` otherwise.

* **`Console_Info.IsOutputRedirected()`**: Checks if standard output is redirected. Returns `true` if redirected, `false` otherwise.

* **`Console_Info.IsErrorRedirected()`**: Checks if standard error is redirected. Returns `true` if redirected, `false` otherwise.

* **`Credential_Form.PromptForPassword(string caption, string message, string target, string user, PSCredentialTypes credTypes, PSCredentialUIOptions options)`**: Prompts the user for credentials using the Windows Credential UI. Returns a `User_Pwd` object containing the username and password (or `null` if cancelled).

* **`Input_Box.Show(string strTitle, string strPrompt, ref string strVal, bool blSecure)`**: Displays a dialog box for user input. The `blSecure` parameter determines whether the input should be masked (for passwords). The entered value is returned in `strVal`. Returns a `DialogResult` (OK or Cancel).

* **`Input_Box.Show(string strTitle, string strPrompt, ref string strVal)`**: An overload of the above function that defaults to non-secure input.

* **`MainApp.Main(string[] args)`**: The main entry point of the application. Parses command-line arguments, runs the embedded PowerShell script within a runspace, and handles potential exceptions. Returns the exit code.

* **`MainModule.MainModule(MainAppInterface app, MainModuleUI ui)`**: Constructor for the custom PowerShell host.

* **`MainModule.SetShouldExit(int exitCode)`**: Sets the exit code and signals the application to exit.

* **`MainModuleRawUI.FlushInputBuffer()`**: Creates an invisible form presumably to clear the input buffer. (This is a non-standard approach)

* **`MainModuleRawUI.ReadKey(ReadKeyOptions options)`**: Reads a single key press from the user.

* **`MainModuleUI.Prompt(string caption, string message, Collection descriptions)`**: Presents a prompt to the user in form of a message box, handling different input types (string, secure string, array etc).

* **`MainModuleUI.PromptForChoice(string caption, string message, Collection choices, int defaultChoice)`**: Presents a choice using the `Choice_Box` functionality.

* **`MainModuleUI.PromptForCredential(string caption, string message, string userName, string targetName, PSCredentialTypes allowedCredentialTypes, PSCredentialUIOptions options)`**: Prompts for credentials.

* **`MainModuleUI.ReadLine()`**: Reads a line of text from the user via an input box.

* **`MainModuleUI.ReadLineAsSecureString()`**: Reads a secure string from the user via an input box.

* **`MainModuleUI.Write(...)`**: Writes text to the console (implemented via MessageBox).

* **`MainModuleUI.WriteProgress(...)`**: Updates a progress bar.

* **`Progress_Form.Update(ProgressRecord objRecord)`**: Updates the progress bar form with data from the `ProgressRecord`.

* **`ReadKey_Box.Show(string strTitle, string strPrompt, bool blIncludeKeyDown)`**: Displays a dialog to capture a key press event.

Control Flow

* **`MainApp.Main(string[] args)`**: This function has a complex control flow. It parses command-line arguments to determine whether to extract the embedded PowerShell script, run it with parameters, or display help. The bulk of the processing happens within the `Runspace` and `PowerShell` objects which handle script execution and output. Error handling uses try-catch blocks to catch exceptions during execution. Finally, it displays a message box if the `-wait` flag was used.

* **`Choice_Box.Show(...)`**: The function creates a Windows Form with radio buttons, a label (if a prompt is provided), and an OK button. It then iterates through the `arrChoice` collection to add radio buttons for each choice. It handles potential resizing of radio buttons if their text is too long. The control flow concludes by showing the form modally and returning the index of the selected radio button.

* **`Input_Box.Show(...)`**: This function creates a simple form containing a label for the prompt, a textbox for input, and OK/Cancel buttons. It uses `Marshal.PtrToStringUni` which is unusual for this purpose. The logic for handling secure input is straightforward.

* **`Credential_Form.PromptForPassword(...)`**: This function leverages a Windows API call (`CredUIPromptForCredentials`) to obtain credentials securely. Its control flow is simple; it either successfully gets credentials or returns null.

* **`Progress_Form.Update(...)`**: This function is responsible for managing the progress bars and updating the UI based on progress records. It handles the addition and removal of progress bars depending on the state of the progress records. The logic is quite involved to manage nested progress bars and their positions within the form.

Data Structures

* **`ChoiceDescription`**: (Defined elsewhere, not shown in the provided code) Presumably a class containing a label and a help message for a choice in a choice box.

* **`Credential_Form.User_Pwd`**: A simple class to hold username, password, and domain information.

* **`Progress_Form.Progress_Data`**: A struct used in `Progress_Form` to store UI elements associated with a specific progress bar.

* **`PSObject`**: A PowerShell object used to represent data passed between the PowerShell script and the host.

* **`Collection`**: A collection of `ChoiceDescription` objects.

* **`Collection`**: A collection of `FieldDescription` objects.

Malware Family Suggestion

While the code itself isn't inherently malicious, its functionality is suspicious and could be easily weaponized. The ability to run an arbitrary PowerShell script (`onedrive_check_smart_v7.ps1`), coupled with credential harvesting (`Credential_Form.PromptForPassword`) and its obfuscated nature raises strong concerns.

Based on its functionality, this code could be part of a:

* **Information Stealer:** The embedded script could be designed to exfiltrate sensitive data from the victim's system, especially if it's combined with credential harvesting.

* **Remote Access Trojan (RAT):** If the script establishes a remote connection, the program becomes a RAT capable of controlling the system remotely.

* **Backdoor:** The script could act as a backdoor, allowing remote attackers to gain persistent access to the victim's system.

The use of seemingly benign dialog boxes makes it more likely to be successful in its malicious mission. The extraction functionality allows for potential lateral movement, allowing the malware to spread. The lack of any clear purpose in the provided script further enhances suspicion. Proper analysis of `onedrive_check_smart_v7.ps1` is crucial to determining its exact malicious behaviour. The overall design and the use of seemingly innocent names attempt to hide malicious functionality.