Analysis Report for: 48B1B314B23036AC3156369BAEB9CD75.exe

Overall Functionality

This VBScript code checks for the presence of a specific version of Adobe Acrobat Reader (version 9.0) on a system using a regular expression pattern match against a registry key. It then uses registry writes to report success or failure via a Lumension Patch Player interface. This suggests it is part of a larger deployment or patching system. If the Acrobat Reader version is found, it exits with a return code of 1, otherwise 0. There's also a set of stub functions (likely placeholders) for a "PLCCAgent".

Function Summaries

* **`ValidateRegExpression(str, expression)`:** This function takes a string (`str`) and a regular expression (`expression`) as input. It uses a regular expression object to test if the string matches the expression and returns `True` if it does, `False` otherwise.

* **`GetRegValue(strKeyPath)`:** This function attempts to read a registry value at the specified key path (`strKeyPath`). It returns the value if successful, and an empty string if an error occurs.

* **`Debugger(message)`:** A simple debugging function that displays a message box if the `isDebugMode` variable is `True`.

* **`PLCCAgent_InitiateSystemShutdown()`:** This function writes "1" to the registry key `HKLM\Software\Lumension\Patch Player\REBOOT`, likely triggering a system reboot as part of a patching process.

* **`PLCCAgent_SetReturnCode(RetCode, RetDesc)`:** This function writes a return code (`RetCode`) and description (`RetDesc`) to the Lumension Patch Player registry keys for `RCODE` and `RDESC`, respectively.

* **`PLCCAgent_PollHost()`:** An empty function – likely a placeholder for future functionality related to communication with a host system.

* **`PLCCAgent_Sleep(Delay)`:** An empty function – likely a placeholder for a function that would pause execution for a specified duration.

* **`WScript_Quit(RetCode)`:** Writes a return code to the Lumension Patch Player registry and then terminates the script.

Control Flow

* **Main Script:** The main script reads configuration settings and then calls `GetRegValue` to retrieve the Adobe Acrobat Reader GUID from the registry. It then passes this value and a regular expression pattern to `ValidateRegExpression` to check if it matches the expected format. Based on the result, it calls `Debugger` (if `isDebugMode` is true) and then `WScript_Quit` with a return code of 1 (success) or 0 (failure).

* **`ValidateRegExpression`:** This function creates a regular expression object, sets its pattern and case-insensitive flag, and uses the `Test` method to check if the input string matches the pattern. The result is directly assigned as the function's return value.

* **`GetRegValue`:** This function uses error handling (`On Error Resume Next`) to gracefully handle cases where the registry key doesn't exist or is inaccessible. It reads the registry value and returns it if successful, or an empty string otherwise.

Data Structures

The code primarily uses simple data types: strings, booleans, and integers. The regular expression object is the most complex data structure used, but it's handled implicitly by the VBScript engine. Registry keys serve as a form of key-value store.

Malware Family Suggestion

While this code snippet itself isn't malicious, its functionality and design raise some concerns. The use of registry manipulation to report to a "Patch Player" system and the way return codes are written to the registry strongly suggest its use within some type of installer or software update mechanism that could be leveraged in a malicious context.

Specifically, it could be part of a dropper or a backdoor installation. The "Patch Player" interaction might be used to report successful installation to a command-and-control (C&C) server, potentially enabling further malicious activity. The ability to trigger a system reboot is another potentially problematic feature often used in malware to ensure persistence or to prevent disruption. The empty functions (`PLCCAgent_PollHost`, `PLCCAgent_Sleep`) are highly suggestive of a larger piece of code capable of other malicious actions.

Therefore, it's difficult to definitively classify this code snippet as belonging to a specific malware family without seeing the rest of the application. However, its behavior aligns with components often found within installers or update mechanisms that could be part of a larger malware deployment strategy and this would be deemed suspicious. The pattern of registry writing would suggest the code might be used in a more advanced persistent threat (APT) or downloader-type malware.