Analysis Report for: 002D3B6AFD4191D6F37C44E36BD8B07C.cs

Overall Functionality

This codebase appears to be a collection of files from a web application and potentially a Dynamics NAV extension. The core functionality revolves around an audit logging system. `_Page_Areas_Audit_Views_AuditLogTest_CreationTest_cshtml.cs` and `_Page_Areas_Audit_Views_AuditLogTest_LoadTest_cshtml.cs` are ASP.NET MVC view files that provide user interfaces for testing the creation and loading of audit logs, respectively. `Codeunit7050260.cs` seems to be a Business Central/Dynamics NAV codeunit that interacts with the audit logging system at a lower level, possibly handling database interactions. The other files are assembly information files and seemingly unrelated components.

Function Summaries

* **`_Page_Areas_Audit_Views_AuditLogTest_CreationTest_cshtml.Execute()`**: This function renders the HTML for a web page used to test the creation of audit log entries. It dynamically generates a form with various input fields for different log types and associated data. It uses dynamic binding (`Microsoft.CSharp.RuntimeBinder`) for setting the page title. It includes a section for JavaScript code to handle the form submission and other client-side logic. It does not return a value (void).

* **`_Page_Areas_Audit_Views_AuditLogTest_LoadTest_cshtml.Execute()`**: This function renders HTML for a web page to test loading (retrieving) audit logs. It's simpler than `CreationTest`, primarily focusing on including a JavaScript file for client-side logic. It does not return a value (void).

* **`FastObjectFactory_app_web_ljwxleeg.Create_ASP__Page_Areas_Audit_Views_AuditLogTest_LoadTest_cshtml()`**: Creates an instance of the `_Page_Areas_Audit_Views_AuditLogTest_LoadTest_cshtml` class. Returns an object (the created instance).

* **`FastObjectFactory_app_web_ljwxleeg.Create_ASP__Page_Areas_Audit_Views_AuditLogTest_CreationTest_cshtml()`**: Creates an instance of the `_Page_Areas_Audit_Views_AuditLogTest_CreationTest_cshtml` class. Returns an object (the created instance).

* **`Codeunit7050260.OnInvoke(int memberId, object[] args)`**: This is a method within a Dynamics NAV codeunit. It acts as a dispatcher, handling different member IDs (presumably function IDs within the codeunit). If `memberId` is 7050000, it calls `OnAfterIsAlwaysLoggedTable`. Otherwise, it throws an error. Returns `null`.

* **`Codeunit7050260.OnAfterIsAlwaysLoggedTable(int tableID, ByRef alwaysLogTable)`**: This event subscriber function is triggered after a table's "always log" property is set. It checks if a specific table ID (2000000073) is being processed and, if so, sets `alwaysLogTable` to `true`. This suggests it's forcing specific tables to always have their changes logged. It does not return a value (void).

* **`Codeunit7050260.OnClear()`**: A method for cleanup; in this case, it's empty. It does not return a value (void).

* **`ConfigRegrasAcesso.CarregarConfiguracaoRegrasAcesso()`**: This function loads configuration for access rules. It contains obfuscated code (likely intentional) and returns a `ConfiguracaoRegrasAcesso` object containing lists of procedures and their associated access levels.

Control Flow

* **`_Page_Areas_Audit_Views_AuditLogTest_CreationTest_cshtml.Execute()`**: The control flow is straightforward: It writes HTML to the response stream using `WriteLiteral`. The primary control flow is determined by the conditional statement checking if a CallSite object is null (for dynamic binding). The `DefineSection` delegate is used to add a script section to the page.

* **`Codeunit7050260.OnAfterIsAlwaysLoggedTable`**: This function uses a conditional statement to check three conditions:
1. A check against a record (`changeLogSetup.Target.ALGetSafe(0, 409565, Array.Empty())`) that determines if a related record is present.
2. A check against a boolean field on that record (`this.changeLogSetup.Target.GetFieldValueSafe(7050000, 34048).ToBoolean()`).
3. A check if the `tableID` matches a specific value (2000000073).
If all three conditions are true, `alwaysLogTable` is set to `true`.

Data Structures

* **`ConfiguracaoRegrasAcesso`**: A class (likely custom) that appears to store configuration data for access rules. It contains a property called `Procs`, which seems to be a collection of `ProcDLL` objects.

* **`ProcDLL`**: A class (likely custom) that seems to represent a stored procedure and its associated access level(s), represented by `EnumAlias` objects.

* **`EnumAlias`**: This data type (likely an enum or similar) represents different access levels or permissions.

* **`NavRecordHandle`, `NavEventScope`, `NavMethodScope`, etc.:** These appear to be types specific to the Microsoft Dynamics NAV/Business Central environment.

Malware Family Suggestion

Based solely on the provided code, there is **no indication of malware**. The code seems to be part of a legitimate application related to audit logging and potentially configuration management. The obfuscation in `ConfigRegrasAcesso.CarregarConfiguracaoRegrasAcesso()` is typical of commercial software aiming to protect intellectual property, not a sign of malicious activity. The Dynamics NAV codeunit suggests integration with an enterprise system, making it highly unlikely to be malicious in origin. However, a complete security analysis would require examining the entire application and its deployment context. The existence of access control logic (as seen in `ConfigRegrasAcesso`) does *not* automatically imply malicious intent.