Analysis Report for: 1371BD22044C6CE8470D959CDB6232AA.au3

Overall Functionality

This C code, written in AutoIt's AutoItScript language (a dialect of C-like syntax), implements a simple GUI application that displays the current time and indicates whether the current minute falls within a predefined set of allowed time ranges. The GUI shows the current time, a status message ("Comenzar!" or "No comenzar"), and the remaining time until the next allowed range or the end of the current allowed range. It uses extensive hardcoded constants, seemingly representing Windows API calls and system-related data, for managing the GUI and time-related logic.

Function Summaries

The code contains numerous functions, many of which appear to be wrappers around Windows API calls. Here's a summary of some key functions:

* **`_WinAPI_Beep`**: Plays a beep sound. Parameters: `$ifreq` (frequency), `$iduration` (duration). Return value: Boolean indicating success.
* **`_WinAPI_FormatMessage`**: Formats a message based on specified flags, source, message ID, language ID, and arguments. Return value: Number of characters written or error code.
* **`_WinAPI_GetLastError`**: Retrieves the last error code. Return value: Error code.
* **`_WinAPI_GetLastErrorMessage`**: Retrieves the error message string associated with the last error code. Return value: Error message string.
* **`_WinAPI_MessageBeep`**: Plays a system beep sound. Return value: Boolean indicating success.
* **`_WinAPI_MsgBox`**: Displays a message box. Return value: Void (the function doesn't explicitly return a value in the code).
* **`_WinAPI_SetLastError`**: Sets the last error code. Return value: NULL.
* **`_WinAPI_ShowError`**: Displays an error message box, optionally exiting the script. Return value: Void.
* **`_WinAPI_ShowLastError`**: Displays a message box showing the last error. Return value: 1.
* **`_WinAPI_ShowMsg`**: Displays an informational message box. Return value: Void.
* **`_Security__AdjustTokenPrivileges`**: Adjusts token privileges. Return value: Boolean indicating success.
* **`_Security__CreateProcessWithToken`**: Creates a process with a specified token. Return value: Boolean indicating success.
* **`_Security__DuplicateTokenEx`**: Duplicates a token. Return value: Token handle.
* **`_Security__GetAccountSid`**: Retrieves the SID for a given account name. Return value: SID as a string or empty string on error.
* **`_Security__GetLengthSid`**: Retrieves the length of a security identifier (SID). Return value: Length of SID in bytes or 0 on error.
* **`_Security__GetTokenInformation`**: Retrieves information about a token. Return value: The token information as a struct.
* **`_Security__ImpersonateSelf`**: Impersonates the current user. Return value: Boolean indicating success.
* **`_Security__IsValidSid`**: Checks if a SID is valid. Return value: Boolean indicating validity.
* **`_Security__LookupAccountName`**: Looks up an account name. Return value: Array containing account name, SID, and account type.
* **`_Security__LookupAccountSid`**: Looks up a SID. Return value: Array containing account name, account type, domain name.
* **`_Security__LookupPrivilegeValue`**: Looks up a privilege value. Return value: LUID of privilege or 0 on error.
* **`_Security__OpenProcessToken`**: Opens a process token. Return value: Token handle or 0 on error.
* **`_Security__OpenThreadToken`**: Opens a thread token. Return value: Token handle or 0 on error.
* **`_Security__OpenThreadTokenEx`**: Opens a thread token, with retry for impersonation. Return value: Token handle or 0 on error.
* **`_Security__SetPrivilege`**: Sets a token privilege. Return value: Boolean indicating success.
* **`_Security__SetTokenInformation`**: Sets token information. Return value: Boolean indicating success.
* **`_Security__SidToStringSid`**: Converts a SID to a string representation. Return value: SID string or empty string on error.
* **`_Security__SidTypeStr`**: Converts a SID type integer to string. Return value: String representing SID type.
* **`_Security__StringSidToSid`**: Converts a SID string to a SID structure. Return value: Pointer to SID or 0 on error.
* **`_MemFree`**: Frees memory allocated with _MEMINIT. Return value: Boolean indicating success.
* **`_MemGlobalAlloc`**: Allocates global memory. Return value: Handle to allocated memory or 0 on error.
* **`_MemGlobalFree`**: Frees global memory. Return value: Handle to memory or 0 on error.
* **`_MemGlobalLock`**: Locks global memory. Return value: Pointer to memory or 0 on error.
* **`_MEMGLOBALREALLOC`**: Reallocates global memory. Return value: Handle to reallocated memory or 0 on error.
* **`_MemGlobalSize`**: Gets the size of global memory. Return value: Size of memory in bytes or 0 on error.
* **`_MemGlobalUnlock`**: Unlocks global memory. Return value: Boolean indicating success.
* **`_MEMINIT`**: Initializes a memory map for process memory access. Return value: Memory address.
* **`_MemMoveMemory`**: Copies a block of memory. Return Value: void
* **`_MEMREAD`**: Reads memory from a process. Return value: Boolean indicating success.
* **`_MEMWRITE`**: Writes memory to a process. Return value: Boolean indicating success.
* **`_MemVirtualAlloc`**: Allocates virtual memory. Return value: Memory address.
* **`_MemVirtualAllocEx`**: Allocates virtual memory in another process. Return value: Memory address.
* **`_MemVirtualFree`**: Frees virtual memory. Return value: Boolean indicating success.
* **`_MemVirtualFreeEx`**: Frees virtual memory in another process. Return value: Boolean indicating success.
* **`__MEM_OPENPROCESS`**: Opens a process handle, potentially using debug privileges. Return value: Process handle.
* **`_WinAPI_CreateFile`**: Creates or opens a file. Return value: File handle.
* **`_WinAPI_FreeLibrary`**: Frees a loaded library. Return value: Boolean indicating success.
* **`_WinAPI_GetCursorInfo`**: Gets cursor information. Return value: Array with cursor info.
* **`_WinAPI_GetDlgCtrlID`**: Gets the ID of a dialog control. Return value: Control ID.
* **`_WinAPI_GetModuleHandle`**: Gets a module handle. Return value: Module handle.
* **`_WINAPI_GETSTRING`**: Retrieves a string from a pointer. Return Value: The string.
* **`_WINAPI_GETVERSION`**: Gets the OS version. Return value: OS version as a double.
* **`_WINAPI_ISWOW64PROCESS`**: Checks if a process is a 32-bit process running on a 64-bit system. Return value: Boolean.
* **`_WinAPI_LoadImage`**: Loads an image. Return value: Image handle.
* **`_WinAPI_LoadLibrary`**: Loads a library. Return value: Library handle.
* **`_WINAPI_PATHISDIRECTORY`**: Checks if a path is a directory. Return value: Boolean.
* **`_WinAPI_ReadFile`**: Reads from a file. Return value: Boolean indicating success.
* **`_WINAPI_STRLEN`**: Gets the length of a string. Return value: String length.

```
* **`_WINAPI_SWITCHCOLOR`**: Switches the color format. Return value: Modified color.
* **`_WinAPI_WriteFile`**: Writes to a file. Return value: Boolean indicating success.
* **`__CHECKERRORARRAYBOUNDS`**: Checks array bounds, returns an error if invalid. Return value: 0 or an error code.
* **`__CHECKERRORCLOSEHANDLE`**: Closes a handle and sets error if needed. Return value: Error status.
* **`__DLL`**: Loads or checks for a DLL. Return value: 1 on success, 0 on failure.
* **`__ENUMWINDOWSPROC`**: Callback function for EnumWindows. Return value: 1.
* **`__FATALEXIT`**: Exits the script with an error. Return value: Void.
* **`__INC`**: Increases the size of an array. Return value: 1 on success, 0 on failure.
* **`__RGB`**: Converts a color value to RGB format. Return value: RGB color value.
* **`_DateAdd`**: Adds a time/date interval to a date. Return value: Resultant date string.
* **`_DateDayOfWeek`**: Gets the day of the week for a date. Return value: Day of week string.
* **`_DateDaysInMonth`**: Gets the number of days in a month. Return value: Number of days.
* **`_DateDiff`**: Calculates the difference between two dates. Return value: Difference as an integer.
* **`_DateIsLeapYear`**: Checks if a year is a leap year. Return value: 1 if leap year, 0 otherwise.
* **`_DateIsValid`**: Checks if a date is valid. Return value: Boolean.
* **`_DATELASTWEEKDAYNUM`**: Calculates last weekday number of week. Return value: Weekday number.
* **`_DATELASTMONTHNUM`**: Gets last month number of year. Return value: Month number.
* **`_DATELASTMONTHYEAR`**: Gets last month year. Return value: Year.
* **`_DATENEXTWEEKDAYNUM`**: Gets next weekday number of week. Return value: Weekday number.
* **`_DATENEXTMONTHNUM`**: Gets next month number of year. Return value: Month number.
* **`_DATENEXTMONTHYEAR`**: Gets next month year. Return value: Year.
* **`_DateTimeFormat`**: Formats a date and time. Return value: Formatted date and time string.
* **`_DateTimeSplit`**: Splits a date and time string into components. Return value: 1 on success, 0 on error.
* **`_DateToDayOfWeek`**: Calculates the day of the week. Return value: Day of week (1-7).
* **`_DateToDayOfWeekISO`**: Calculates the day of the week using ISO 8601 standard. Return value: Day of week (1-7).
* **`_DateToDayValue`**: Converts a date to a Julian day number. Return value: Julian day number.
* **`_DateToMonth`**: Gets the month name. Return value: Month name string.
* **`_DayValueToDate`**: Converts a Julian day number to a date. Return value: Date string.
* **`_DATE_JULIANDAYNO`**: Calculates the Julian day number. Return value: Julian day number.
* **`_JULIANTODATE`**: Converts a Julian day number to a date. Return value: Date string.
* **`_Now`**: Gets the current date and time. Return value: Current date and time string.
* **`_NowCalc`**: Gets the current date and time in a specific format. Return value: Date and time string.
* **`_NowCalcDate`**: Gets the current date in a specific format. Return value: Date string.
* **`_NowDate`**: Gets the current date. Return value: Current date string.
* **`_NowTime`**: Gets the current time in a specific format. Return value: Time string.
* **`_SetDate`**: Sets the system date. Return value: 0 on success, 1 on error.
* **`_SetTime`**: Sets the system time. Return value: 0 on success, 1 on error.
* **`_TicksToTime`**: Converts ticks to hours, minutes, and seconds. Return value: Boolean indicating success.
* **`_TimeToTicks`**: Converts hours, minutes, and seconds to ticks. Return value: Number of ticks.
* **`_WeekNumberISO`**: Calculates the ISO week number. Return value: Week number.
* **`_WEEKNUMBER`**: Calculates the week number. Return value: Week number.
* **`_DAYSINMONTH`**: Returns an array of days in each month for a given year. Return value: Array.
* **`__DATE_TIME_CLONESYSTEMTIME`**: Clones a SYSTEMTIME structure. Return value: Cloned SYSTEMTIME struct.
* **`_Date_Time_CompareFileTime`**: Compares two FILETIME structures. Return value: Comparison result (-1, 0, 1).
* **`_Date_Time_DOSDateTimeToFileTime`**: Converts DOS date and time to FILETIME. Return value: FILETIME structure.
* **`_Date_Time_DOSDateToArray`**: Converts a DOS date to an array of date components. Return value: Array.
* **`_Date_Time_DOSDateTimeToArray`**: Converts DOS date and time to an array. Return value: Array.
* **`_Date_Time_DOSDateTimeToStr`**: Converts DOS date and time to a string. Return value: Date and time string.
* **`_Date_Time_DOSDateToStr`**: Converts DOS date to a string. Return value: Date string.
* **`_Date_Time_DOSTimeToArray`**: Converts DOS time to an array. Return value: Array.
* **`_Date_Time_DOSTimeToStr`**: Converts DOS time to a string. Return value: Time string.
* **`_Date_Time_EncodeFileTime`**: Encodes a date and time into a FILETIME structure. Return value: FILETIME structure.
* **`_Date_Time_EncodeSystemTime`**: Encodes a date and time into a SYSTEMTIME structure. Return value: SYSTEMTIME structure.
* **`_Date_Time_FileTimeToArray`**: Converts FILETIME to an array of date/time components. Return value: Array.
* **`_Date_Time_FileTimeToStr`**: Converts FILETIME to a string. Return value: Date and time string.
* **`_Date_Time_FileTimeToDOSDateTime`**: Converts FILETIME to DOS date and time. Return value: Array [dosDate, dosTime].
* **`_Date_Time_FileTimeToLocalFileTime`**: Converts FILETIME to local FILETIME. Return value: local FILETIME structure.
* **`_Date_Time_FileTimeToSystemTime`**: Converts FILETIME to SYSTEMTIME. Return value: SYSTEMTIME structure.
* **`_Date_Time_GetFileTime`**: Retrieves file times. Return value: Extended [result, array of FILETIME structs].
* **`_Date_Time_GetLocalTime`**: Retrieves local time. Return value: SYSTEMTIME structure.
* **`_Date_Time_GetSystemTime`**: Retrieves system time. Return value: SYSTEMTIME structure.
* **`_Date_Time_GetSystemTimeAdjustment`**: Retrieves system time adjustment information. Return value: Extended return.
* **`_Date_Time_GetSystemTimeAsFileTime`**: Retrieves system time as FILETIME. Return value: FILETIME structure.
* **`_Date_Time_GetSystemTimes`**: Retrieves system times. Return value: Extended return.
* **`_Date_Time_GetTickCount`**: Gets the number of milliseconds elapsed since system startup. Return value: Milliseconds.
* **`_Date_Time_GetTimeZoneInformation`**: Retrieves time zone information. Return value: Array with timezone information.
* **`_Date_Time_LocalFileTimeToFileTime`**: Converts local FILETIME to FILETIME. Return value: FILETIME structure.
* **`_Date_Time_SetFileTime`**: Sets file times. Return value: Boolean.
* **`_Date_Time_SetLocalTime`**: Sets local time. Return value: Boolean.
* **`_Date_Time_SetSystemTime`**: Sets system time. Return value: Boolean.
* **`_Date_Time_SetSystemTimeAdjustment`**: Sets system time adjustment. Return value: Boolean.
* **`_Date_Time_SetTimeZoneInformation`**: Sets time zone information. Return value: Boolean.
* **`_Date_Time_SystemTimeToArray`**: Converts SYSTEMTIME to array. Return value: Array.
```

* **`_Date_Time_SystemTimeToDateStr`**: Converts SYSTEMTIME to date string. Return value: Date string.
* **`_Date_Time_SystemTimeToDateTimeStr`**: Converts SYSTEMTIME to date and time string. Return value: Date and time string.
* **`_Date_Time_SystemTimeToFileTime`**: Converts SYSTEMTIME to FILETIME. Return value: FILETIME structure.
* **`_Date_Time_SystemTimeToTimeStr`**: Converts SYSTEMTIME to time string. Return value: Time string.
* **`_Date_Time_SystemTimeToTzSpecificLocalTime`**: Converts SYSTEMTIME to timezone specific local time. Return value: Extended return.
* **`_Date_Time_TzSpecificLocalTimeToSystemTime`**: Converts timezone-specific local time to SYSTEMTIME. Return value: Extended return.
* **`_WinAPI_CloseHandle`**: Closes a handle. Return value: Boolean.
* **`_WinAPI_DeleteObject`**: Deletes a GDI object. Return value: Boolean.
* **`_WinAPI_DuplicateHandle`**: Duplicates a handle. Return value: Duplicate handle.
* **`_WINAPI_GETCURRENTOBJECT`**: Gets the current object. Return value: Handle to GDI object.
* **`_WinAPI_GetCurrentProcess`**: Gets the current process handle. Return value: Process handle.
* **`_WinAPI_GetObject`**: Gets information about a GDI object. Return value: Number of bytes copied.
* **`_WINAPI_GETOBJECTINFOBYHANDLE`**: Gets information about an object. Return value: Array of object information.
* **`_WINAPI_GETOBJECTNAMEBYHANDLE`**: Gets the name of an object by handle. Return value: Object name string.
* **`_WINAPI_GETOBJECTTYPE`**: Gets type of GDI object. Return value: Type of GDI object.
* **`_WinAPI_GetStdHandle`**: Gets a standard handle. Return value: Standard handle.
* **`_WinAPI_GetStockObject`**: Gets a stock object. Return value: Stock object handle.
* **`_WinAPI_SelectObject`**: Selects a GDI object. Return value: Handle to previous object.
* **`_WinAPI_SetHandleInformation`**: Sets handle information. Return value: Boolean.
* **`_WINAPI_COMPARESTRING`**: Compares two strings. Return value: Comparison result.
* **`_WINAPI_CREATENUMBERFORMATINFO`**: Creates a number format info structure. Return value: Number format info structure.
* **`_WINAPI_ENUMSYSTEMGEOID`**: Enumerates system geo IDs. Return value: Array of geo IDs.
* **`_WINAPI_ENUMSYSTEMLOCALES`**: Enumerates system locales. Return value: Array of locale IDs.
* **`_WINAPI_ENUMUILANGUAGES`**: Enumerates UI languages. Return value: Array of language IDs or names.
* **`_WINAPI_GETDATEFORMAT`**: Gets the date format. Return value: Formatted date string.
* **`_WINAPI_GETDURATIONFORMAT`**: Gets the duration format. Return value: Formatted duration string.
* **`_WINAPI_GETGEOINFO`**: Gets geographic information. Return value: Geographic information as a string.
* **`_WINAPI_GETLOCALEINFO`**: Gets locale information. Return value: Locale information as a string.
* **`_WINAPI_GETNUMBERFORMAT`**: Gets the number format. Return value: Formatted number string.
* **`_WINAPI_GETSYSTEMDEFAULTLANGID`**: Gets the system default language ID. Return value: Language ID.
* **`_WINAPI_GETSYSTEMDEFAULTLCID`**: Gets the system default locale ID. Return value: Locale ID.
* **`_WINAPI_GETSYSTEMDEFAULTUILANGUAGE`**: Gets the system default UI language ID. Return value: Language ID.
* **`_WINAPI_GETTHREADLOCALE`**: Gets the thread locale ID. Return value: Locale ID.
* **`_WINAPI_GETTHREADUILANGUAGE`**: Gets the thread UI language ID. Return value: Language ID.
* **`_WINAPI_GETTIMEFORMAT`**: Gets the time format. Return value: Formatted time string.
* **`_WINAPI_GETUSERDEFAULTLANGID`**: Gets the user default language ID. Return value: Language ID.
* **`_WINAPI_GETUSERDEFAULTLCID`**: Gets the user default locale ID. Return value: Locale ID.
* **`_WINAPI_GETUSERDEFAULTUILANGUAGE`**: Gets the user default UI language ID. Return value: Language ID.
* **`_WINAPI_GETUSERGEOID`**: Gets the user geo ID. Return value: Geo ID.
* **`_WINAPI_ISVALIDLOCALE`**: Checks if a locale is valid. Return value: Boolean.
* **`_WINAPI_SETLOCALEINFO`**: Sets locale information. Return value: Boolean.
* **`_WINAPI_SETTHREADLOCALE`**: Sets the thread locale ID. Return value: Boolean.
* **`_WINAPI_SETTHREADUILANGUAGE`**: Sets the thread UI language ID. Return value: Boolean.
* **`_WINAPI_SETUSERGEOID`**: Sets the user geo ID. Return value: Boolean.
* **`__ENUMGEOIDPROC`**: Callback function for EnumSystemGeoID. Return value: 1.
* **`__ENUMLOCALESPROC`**: Callback function for EnumSystemLocales. Return value: 1.
* **`__ENUMUILANGUAGESPROC`**: Callback function for EnumUILanguages. Return value: 1.
* **`ESTAENRANGO`**: Checks if a minute is within allowed ranges. Return value: Boolean.
* **`TIEMPOPARAPROXIMOINICIO`**: Calculates time until the next allowed time range. Return value: Seconds.
* **`TIEMPORESTANTEENRANGO`**: Calculates remaining time in the current allowed range. Return value: Seconds.

Control Flow

* **Main GUI Loop**: The main loop continuously retrieves messages using `GUIGetMsg()`. It checks for the close event (`$gui_event_close`) or a button click (`$btncerrar`) to exit the loop. Otherwise, it updates the GUI with the current time, status, and remaining time.
* **`ESTAENRANGO`**: Iterates through a 2D array `$rangospermitidos` to check if the input minute falls within any of the defined ranges.
* **`TIEMPOPARAPROXIMOINICIO`**: Finds the next allowed range and calculates the remaining time until its start.
* **`TIEMPORESTANTEENRANGO`**: Finds the current range and calculates the time until the range ends.

Data Structures

* **`$rangospermitidos`**: A 2D array defining allowed time ranges (minutes).
* **Various Structures**: The code extensively uses DllStructCreate to define and work with Windows API structures like `SYSTEMTIME`, `FILETIME`, etc. These are used to manage date and time operations.

Malware Family Suggestion

While the code itself is not malicious, the extensive use of Windows API functions related to security (e.g., token manipulation, process creation) and memory management, combined with the ability to set system time, raises a red flag. A sophisticated attacker could modify this code to:

* **Elevate privileges**: The functions `_Security__AdjustTokenPrivileges`, `_Security__SetPrivilege`, `_Security__OpenProcessToken`, etc., could be misused to gain elevated privileges on a compromised system.
* **Inject malicious code**: The memory management functions (like `_MemVirtualAllocEx`, `_MEMWRITE`) could be used to inject malicious code into running processes.
* **Modify system time**: The ability to set system time (`_Date_Time_SetSystemTime`, `_Date_Time_SetLocalTime`) could be used to cover tracks or to create discrepancies in log files for stealth purposes.

Therefore, while the presented code is innocuous, its underlying functions and capabilities make it easily adaptable for use as a component in a variety of malware, potentially including:

* **Rootkits**: To hide malicious processes or modifications to the system.
* **Privilege-escalation exploits**: To elevate the privileges of the malware to gain administrator or system-level access.
* **Backdoors**: To maintain persistent access to a compromised system.

It's crucial to note that the potential for malicious use stems from the functions used and not necessarily the current implementation. A security review should be conducted on any code utilizing such API calls.