Analysis Report for: a.au3

Overall Functionality

This C code, written using AutoIt's scripting language, implements a Windows service named "e+e.backupservice". The service's primary function is to perform scheduled backups based on configurations read from `backupservice.ini`. It reads job details (including executables to run, schedules, and other parameters) from the INI file and executes these backup jobs at the specified times and days of the week. The service also includes functionality for installation, removal, starting, stopping, and managing the service itself through command-line arguments. Error handling is implemented throughout the code, utilizing AutoIt's error handling mechanism.

Function Summaries

The code contains a vast number of functions. Here's a breakdown of some of the most significant ones categorized for clarity:

* **Windows API Wrapper Functions:** These functions act as wrappers around numerous Windows API calls, providing a layer of abstraction and error handling. Examples include:
* `_WinAPI_GetLastError()`, `_WinAPI_SetLastError()`: Get and set the last error code.
* `_WinAPI_CreateProcess()`, `_WinAPI_OpenProcess()`: Create and open processes.
* `_WinAPI_ReadFile()`, `_WinAPI_WriteFile()`: Read and write files.
* `_WinAPI_CreateWindowEx()`, `_WinAPI_DestroyWindow()`: Create and destroy windows.
* Many others related to GDI, user32, kernel32, etc.

* **Security Functions:** These functions handle privilege adjustments and token manipulation.
* `_Security__AdjustTokenPrivileges()`: Adjusts the privileges of a token.
* `_Security__CreateProcessWithToken()`: Creates a process with a specified token.
* `_Security__DuplicateTokenEx()`: Duplicates a token.
* `_Security__LookupAccountName()`, `_Security__LookupAccountSid()`: Look up account names and SIDs.
* Others related to security access.

* **Service Management Functions:** These functions handle the service lifecycle.
* `_SERVICE_CREATE()`: Creates the service.
* `_SERVICE_DELETE()`: Deletes the service.
* `_SERVICE_START()`, `_SERVICE_STOP()`: Starts and stops the service.
* `_SERVICE_QUERY*()`: Queries various service configurations (status, dependencies, etc.).
* `_SERVICE_SET*()`: Sets various service configurations.
* `_SERVICE_INIT()`, `_SERVICE_CLEANUP()`, `MAIN()`, `_SERVICE_SERVICEMAIN()`, `_SERVICE_CTRL()`: Functions for the service's main loop and control handling.

* **Backup Job Functions:** These functions manage the backup jobs themselves.
* `_RUNJOB()`: Executes a single backup job based on the configuration.
* `_ISRUNNING()`: Checks the status of a running job, killing it if it exceeds the allotted time.
* `_CHECKNEWRUNNING()`: Checks if a new backup job needs to be started at the current time.
* `_KILLJOB()`: Terminates a running backup process.
* `_RELOAD_INI()`: Reloads the service configuration from the INI file.

* **Helper Functions:** Numerous smaller helper functions for tasks such as date/time manipulation, string operations, and logging. Examples include `_DateAdd()`, `_DateDiff()`, `_DateTimeFormat()`, `LOGPRINT()`.

Control Flow

The main control flow is dictated by the presence or absence of command-line arguments and the service's running state. If no command-line arguments are provided, the service initialization (`_SERVICE_INIT()`) occurs, and the service enters its main loop (`MAIN()`). Within `MAIN()`, the service continuously checks for new jobs to run, executes them, and manages the job queue. If command-line arguments are provided, a switch statement determines the action (install, remove, start, stop, restart) and calls the appropriate function.

The `_RUNJOB()` function represents the core logic for executing backup tasks. This function checks for conditions (time, day) before executing the backup command using `Run()`. The `_ISRUNNING()` function provides logic to check if a process is running and to clean up after job completion or timeout.

The service management functions generally follow a pattern of: 1) Open the service control manager and the service itself. 2) Call the appropriate Win32 API function for service creation/deletion/control. 3) Close the handles. Error checking occurs after each API call.

Data Structures

* **INI file (`backupservice.ini`):** This file stores the configuration for the backup jobs. The code reads sections and keys from this file.
* **Arrays:** The code extensively uses arrays to store data, such as job configurations, service information, and lists of windows.
* **DLL Structures:** The code uses several DLL structures (e.g., `$tagsystemtime`, `$tagfiletime`, various service structures) to interact with Windows API functions. These are defined using `DllStructCreate()`.

* **Global Variables:** Numerous global variables store state information, configuration data, and handles.

Malware Family Suggestion

The functionality of this code strongly resembles a **backdoor** or **RAT (Remote Access Trojan)**. While presented as a backup service, several features raise suspicion:

* **Requires Administrator privileges (`#RequireAdmin`):** This is a common characteristic of malware that needs system-level access to perform malicious actions.
* **Scheduled tasks:** The ability to schedule arbitrary executable execution at specific times and days is concerning; malicious code could be easily disguised as a backup job.
* **Obfuscation:** The use of encryption (`_StringEncrypt()`) to protect the password in the configuration file suggests an attempt to hide malicious intent.
* **Process manipulation:** The code interacts with processes (creating, running, stopping). This capability could be used for other malicious activities beyond backups.
* **Extensive API usage:** The code uses numerous Windows API calls that can be used in various malicious ways.

While the code might genuinely perform backups, the potential to abuse its features for malicious purposes is significant. The extensive process manipulation, privilege escalation, and obfuscation techniques are strong indicators of potentially harmful intentions. Further analysis would be needed to determine definitively if this is indeed malware, but the characteristics are very suspicious.