Analysis Report for: 01AD7A0C1A201750AB870DA22671FC7E.exe.c

Overall Functionality

This C code appears to be the output of a decompiler (Hex-Rays), suggesting it was originally compiled from a higher-level language. The code implements a program that displays a banner, prompts the user for input, and then reveals a flag ("FIAP{b50a2afa53d700fa8e06de2a04c33280}"). The complexity lies in the extensive use of seemingly obfuscated functions, many related to file system operations (directory traversal and globbing), exception handling, and thread management. These functions suggest potential malicious activities, particularly file system enumeration and potentially file modification or execution. The usage of `Sleep` functions hints at timing delays for evasive techniques.

Function Summaries

* **`TopLevelExceptionFilter`**: A Windows exception filter. It handles specific exception codes (e.g., STATUS_STACK_OVERFLOW, STATUS_ACCESS_VIOLATION, STATUS_FLOAT_DENORMAL_OPERAND), attempting to gracefully handle them or possibly hiding errors. Returns `-1` to indicate handled exceptions, `0` otherwise.

* **`sub_4011B0`**: The main entry point after CRT initialization. It sets the unhandled exception filter, initializes CPU features, sets up command-line arguments, and calls `__main` and then `main`. Finally, it exits the process using the return value of `main`.

* **`_mingw32_init_mainargs`**: Initializes the MinGW main arguments (`argc` and `argv`).

* **`mainCRTStartup` and `WinMainCRTStartup`**: Standard MinGW entry points for console and GUI applications, respectively; they both call `sub_4011B0`.

* **`__gcc_register_frame`**: Registers and unregisters frame information for exception handling with libgcc.

* **`__gcc_deregister_frame`**: Unregisters the frame information.

* **`sleep_ms`**: A wrapper for the `Sleep` function, pausing execution for a specified number of milliseconds.

* **`print_flag`**: Prints the flag character by character with delays.

* **`banner`**: Prints a decorative banner to the console.

* **`main`**: The main application logic; calls `__main`, `banner`, prompts the user, and then calls `print_flag`.

* **`_setargv`**: Processes the command line, seemingly performing sophisticated argument parsing and globbing (wildcard expansion). This is heavily obfuscated.

* **`__cpu_features_init`**: Detects CPU features.

* **`__do_global_dtors` and `__do_global_ctors`**: Run global destructors and constructors, respectively.

* **`__main`**: Initializes global variables.

* **`TlsCallback_1` and `__dyn_tls_init`**: Thread-local storage callbacks.

* **`__tlregdtor`**: Thread local storage registration/deregistration.

* **`sub_401BC0`**: Performs cleanup of thread-local storage.

* **`___w64_mingwthr_add_key_dtor` and `___w64_mingwthr_remove_key_dtor`**: Add and remove thread-local storage destructors.

* **`__mingw_TLScallback`**: A thread local storage callback function.

* **`sub_401DF0`**: A function that prints an error message and aborts the program.

* **`sub_401E40`**: A function that performs a memory copy, handling potential memory protection issues using `VirtualProtect`.

* **`_pei386_runtime_relocator`**: Relocates the PE image at runtime.

* **`fesetenv`**: Sets the floating-point environment.

* **`__mingw_glob`**: Implements globbing (wildcard expansion) functionality, similar to the Unix `glob` function. Heavily obfuscated.

* **`__mingw_globfree`**: Frees memory allocated by `__mingw_glob`.

* **`__mingw_dirname`**: Extracts the directory name from a given path. Heavily obfuscated.

* **`__mingw_opendir`, `__mingw_readdir`, `__mingw_closedir`, `__mingw_rewinddir`, `__mingw_telldir`, `__mingw_seekdir`**: These functions appear to implement a custom directory traversal and file reading mechanism. They mimic standard directory functions but are heavily obfuscated.

Control Flow (Significant Functions)

* **`_setargv`**: This function's control flow is extremely complex and obfuscated. It parses the command line arguments, handling quotes, escapes, wildcards (`*`, `?`), and potentially other complex patterns. The numerous loops and conditional statements make static analysis difficult. It uses `__mingw_glob` for wildcard expansion.

* **`__mingw_glob`**: This function's core logic involves recursively traversing directories and matching filenames against patterns. It's heavily obfuscated, using multiple helper functions (`sub_4021A0`, `sub_402240`, `sub_4022B0`, `sub_402570`, `sub_402790`, `sub_4027F0`, `sub_402840`) which themselves are complex and obfuscated.

* **`__mingw_dirname`**: This function's control flow is also quite intricate. It involves converting the input path to wide characters, iterating backward to find the last path separator, and then converting the extracted directory portion back to a multibyte string. It handles both forward slashes and backslashes.

* **`TopLevelExceptionFilter`**: Uses a series of `if` statements to check the exception code and call the appropriate signal handler (if registered).

Data Structures

The code uses several key data structures:

* **`_EXCEPTION_POINTERS`**: Standard Windows structure for exception handling.
* **`fenv_t`**: Standard structure for representing the floating-point environment.
* **`_WIN32_FIND_DATAA`**: Standard Windows structure for file information.
* **`CRITICAL_SECTION`**: Standard Windows structure for thread synchronization.
* **`func_ptr __CTOR_LIST__[]`**: An array of function pointers, used for running global constructors.
* **Custom Structures**: The code utilizes several custom, seemingly self-defined data structures for managing file system globbing results and potentially other internal operations. These structures are not explicitly defined but are inferred from their usage patterns.

Malware Family Suggestion

Based on the observed functionality, this code exhibits characteristics consistent with several malware families:

* **Information Stealer/Recon:** The extensive file system enumeration capabilities (`__mingw_opendir`, `__mingw_readdir`, `__mingw_glob`, etc.), especially coupled with the obfuscation, strongly suggest an information-gathering component. The program could be used to discover files containing sensitive information.

* **Dropper/Downloader:** The sophisticated command-line parsing and globbing hints at the potential ability to process parameters that might define what files to retrieve or process from a network location.

* **Backdoor:** The elaborate exception handling could be used to mask errors and hide from detection.

The use of obfuscation techniques makes it difficult to definitively categorize the malware. However, given the observed features, this code likely represents a piece of malware designed for reconnaissance and potentially malicious actions based on command-line arguments. Further dynamic analysis would be required to fully understand its capabilities and behaviour.