Analysis Report for: 1C9617A26E0F7716E121662EB616D85A.exe.c

Overall Functionality

This C code consists of three functions: `sub_401454`, `sub_401484`, and `sub_404CAC`. The code appears to be obfuscated, possibly by a decompiler, making its exact purpose difficult to determine definitively without more context. However, several suspicious elements suggest malicious intent. The code manipulates memory locations directly (`MEMORY[0x5DE58B0A]`), uses a potentially dangerous function pointer call (`MK_FP`), and employs weak global variables (`byte_4028FC`, `dword_42A480`, `word_42A4C2`) which might be used for configuration or runtime modification.

Function Summaries

* **`sub_401454()`**: This function appears to check the value of the global variable `word_42A4C2`. If it's 65, it returns 0; otherwise, it modifies the global variable `dword_42A480` to point to an offset within `byte_4028FC` and returns the value of `word_42A4C2`. The unusual offset (124) into a 5-byte array suggests potential obfuscation or an attempt to access memory outside the array bounds which could indicate a buffer overflow vulnerability or an attempt to hide malicious code.

* **`sub_401484(_DWORD *a1, int a2)`**: This function takes a pointer to a DWORD (`a1`) and an integer (`a2`) as input. It performs a bitwise XOR operation between the value pointed to by `a1` and `a2`, storing the result at a specific memory address (`MEMORY[0x5DE58B0A]`). It then returns the original pointer `a1`. This strongly suggests encryption or obfuscation of data. The hardcoded memory address further enhances suspicion.

* **`sub_404CAC()`**: This function retrieves a word and a DWORD from the return address on the stack and interprets them as a function pointer. It then calls the function pointed to by this constructed pointer, making it a highly dangerous function call. This technique is often used to execute shellcode or other malicious payloads hidden in memory.

Control Flow

* **`sub_401454()`**: A simple conditional statement checks the value of `word_42A4C2`. If the condition is false, a global variable is modified.

* **`sub_401484()`**: Straightforward. It performs a single XOR operation and returns the input pointer.

* **`sub_404CAC()`**: No loops or conditionals; it directly retrieves data from the stack and calls a function from a dynamically constructed function pointer. This is a very dangerous operation that allows arbitrary code execution.

Data Structures

* **`byte_4028FC`**: A small array of bytes initialized with 5 null bytes ('\x90' – NOP instruction). Its unusual use in `sub_401454` with a large offset is suspicious. Potentially used to hide shellcode or other malicious payload.

* **`dword_42A480`**: An integer variable, possibly used as a pointer or a flag, modified by `sub_401454`.

* **`word_42A4C2`**: A short integer variable, used in a conditional statement in `sub_401454`, potentially used for control flow or configuration.

Malware Family Suggestion

Based on the analysis, this code exhibits strong characteristics of a **polymorphic virus or a downloader**. The use of XOR encryption (`sub_401484`), the dynamically invoked function pointer (`sub_404CAC`), and the obfuscated memory access suggest an attempt to hide malicious functionality and make analysis more difficult. The presence of seemingly innocuous variables (`byte_4028FC`, `dword_42A480`, `word_42A4C2`) that might serve as configuration points or storage for hidden data further strengthens this suspicion. The code is designed to be hard to analyze, which is a hallmark of malicious software. Further investigation would be required to determine the exact nature and payload of the malware. Running this code is extremely dangerous.