

Inherit handles from the parent process

Inherit handles from the parent process :

```
BOOL CreateProcess(  
    lpApplicationName,    // Path to the executable  
    lpCommandLine,        // Command-line arguments  
    lpProcessAttributes,  // Security attributes for the process  
    lpThreadAttributes,   // Security attributes for the primary thread  
    bInheritHandles,      // Inherit handles from parent process  
    dwCreationFlags,       // Process creation flags  
    lpEnvironment,         // Pointer to environment block  
    lpCurrentDirectory,    // Working directory of the new process  
    lpStartupInfo,         // Pointer to STARTUPINFO structure  
    lpProcessInformation   // Pointer to PROCESS_INFORMATION structure  
);
```



output.txt

```
HANDLE hFile = CreateFile ("output.txt",  
    FILE_APPEND_DATA,  
    FILE_SHARE_WRITE,  
    NULL, //Security attributes  
    CREATE_ALWAYS,  
    FILE_ATTRIBUTE_NORMAL,  
    NULL);
```

Parent Process



output.txt

```
HANDLE hFile = CreateFile ("output.txt",  
FILE_APPEND_DATA,  
FILE_SHARE_WRITE,  
NULL, //Security attributes  
CREATE_ALWAYS,  
FILE_ATTRIBUTE_NORMAL,  
NULL);
```

Parent Process

```
CreateProcess(NULL,  
cmdLine,  
NULL, //Security attributes proc  
NULL, //Security attributes thrd  
FALSE, //inherit handle  
0,  
NULL,  
NULL,  
&si, &pi)
```

Child process





output.txt



Not allowed

Want to use it

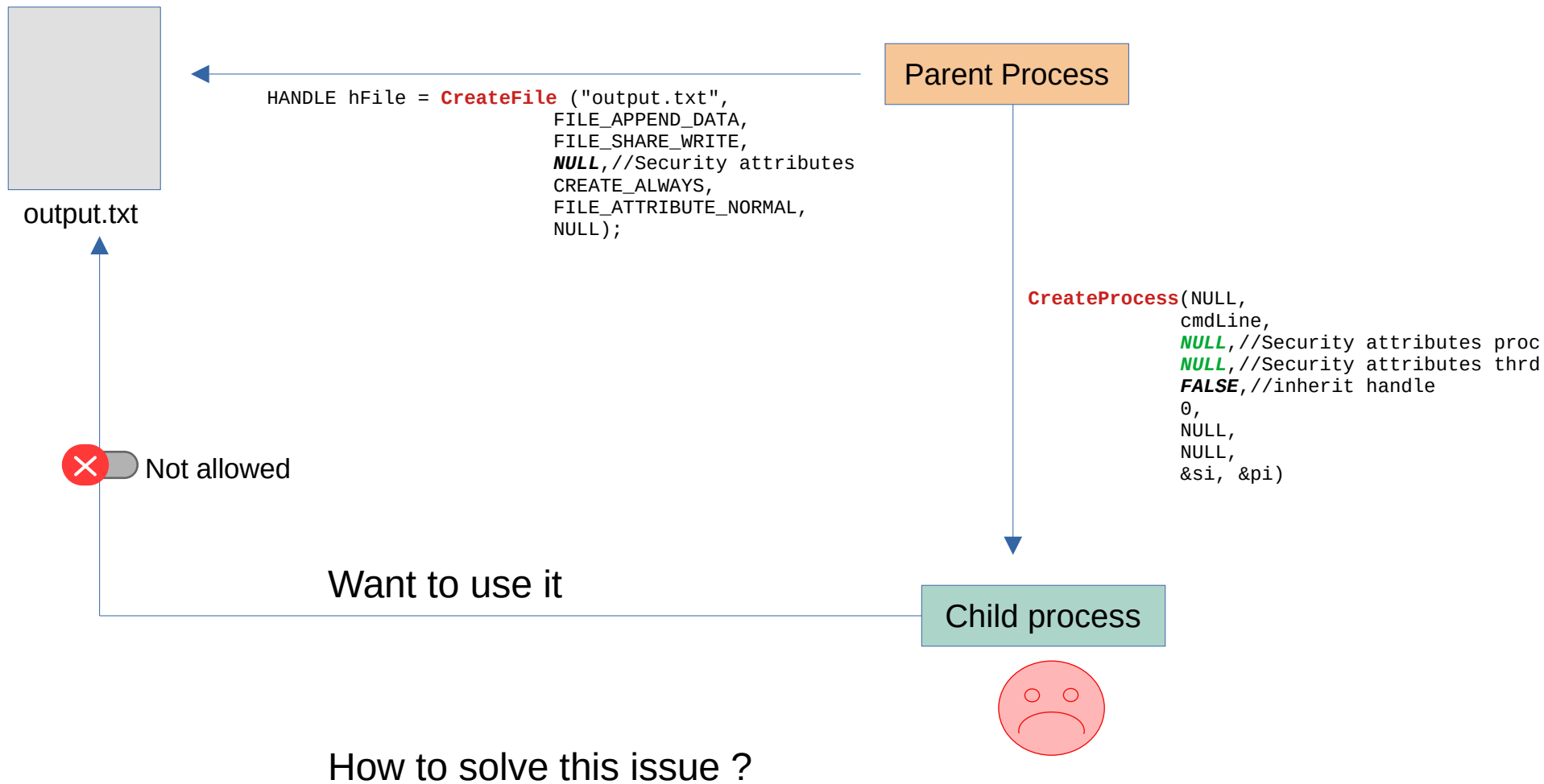
Parent Process

```
HANDLE hFile = CreateFile ("output.txt",  
FILE_APPEND_DATA,  
FILE_SHARE_WRITE,  
NULL, //Security attributes  
CREATE_ALWAYS,  
FILE_ATTRIBUTE_NORMAL,  
NULL);
```

```
CreateProcess(NULL,  
cmdLine,  
NULL, //Security attributes proc  
NULL, //Security attributes thrd  
FALSE, //inherit handle  
0,  
NULL,  
NULL,  
&si, &pi)
```

Child process







output.txt



Not allowed

```
HANDLE hFile = CreateFile ("output.txt",  
FILE_APPEND_DATA,  
FILE_SHARE_WRITE,  
NULL, //Security attributes  
CREATE_ALWAYS,  
FILE_ATTRIBUTE_NORMAL,  
NULL);
```

```
SECURITY_ATTRIBUTES sa = { sizeof(sa), NULL, TRUE };
```

Create a security attributes

Want to use it

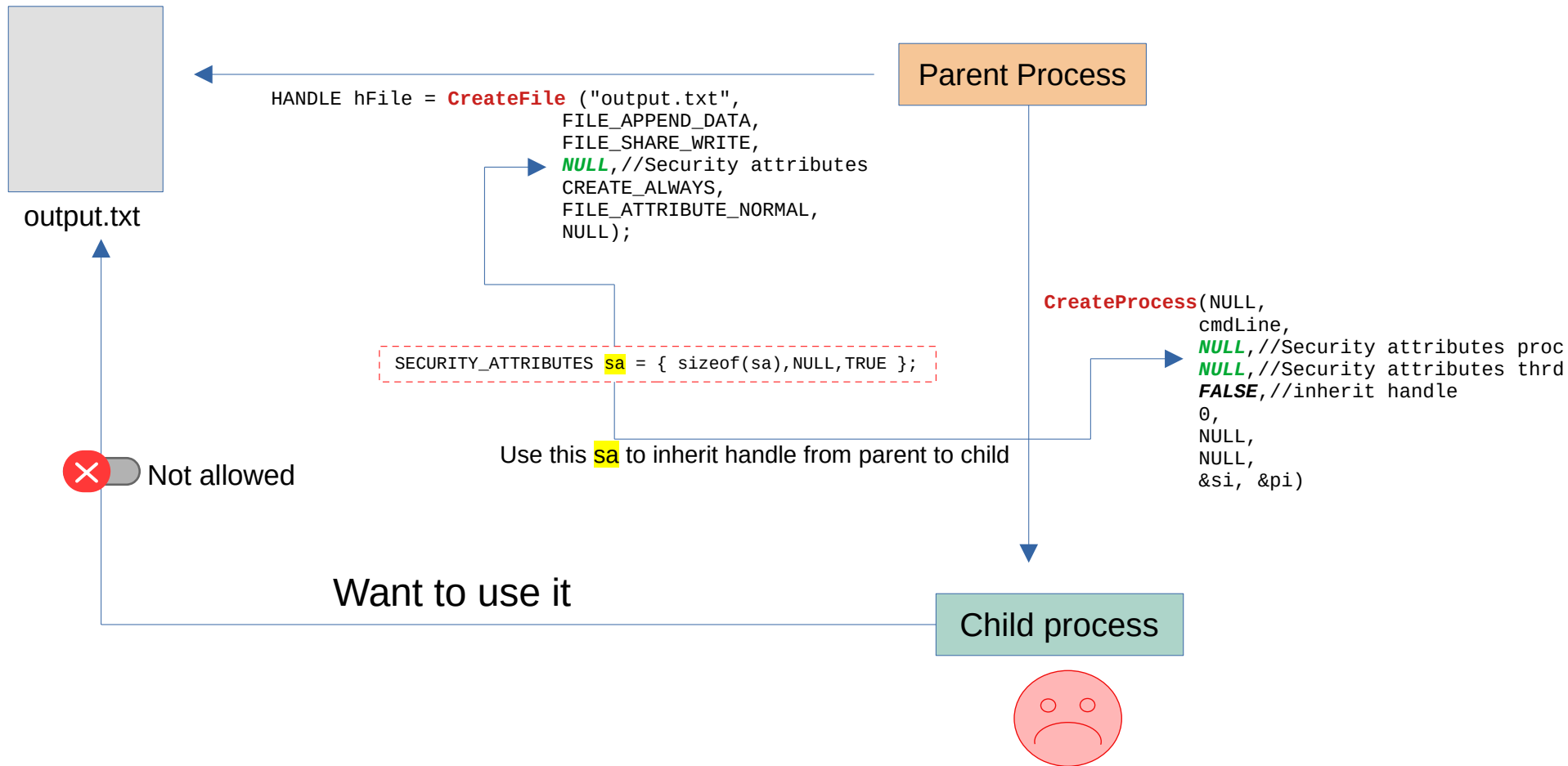
```
typedef struct _SECURITY_ATTRIBUTES {  
    DWORD nLength; //structure length  
    LPVOID lpSecurityDescriptor;  
    BOOL bInheritHandle;  
} SECURITY_ATTRIBUTES;
```

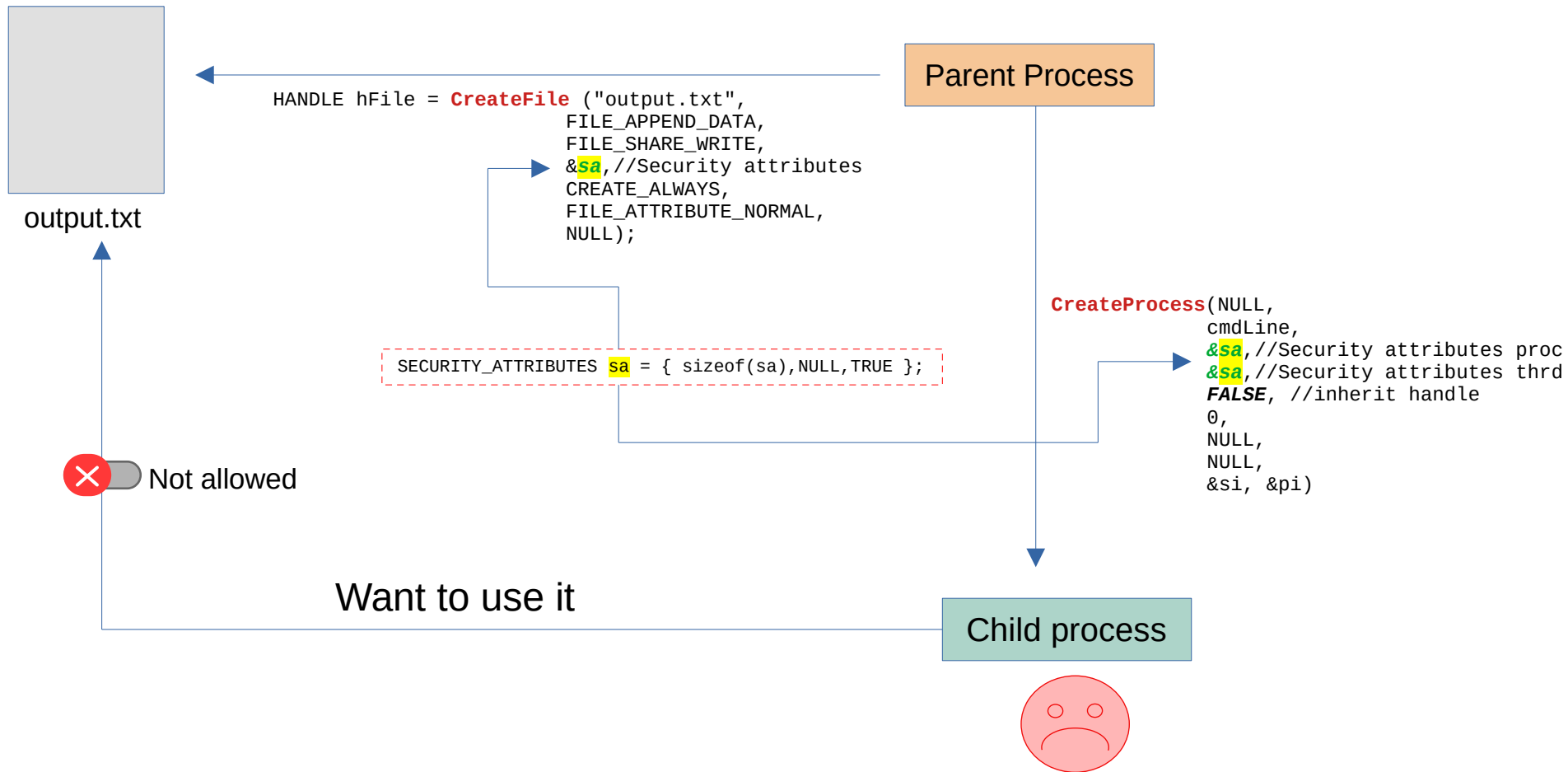
Parent Process

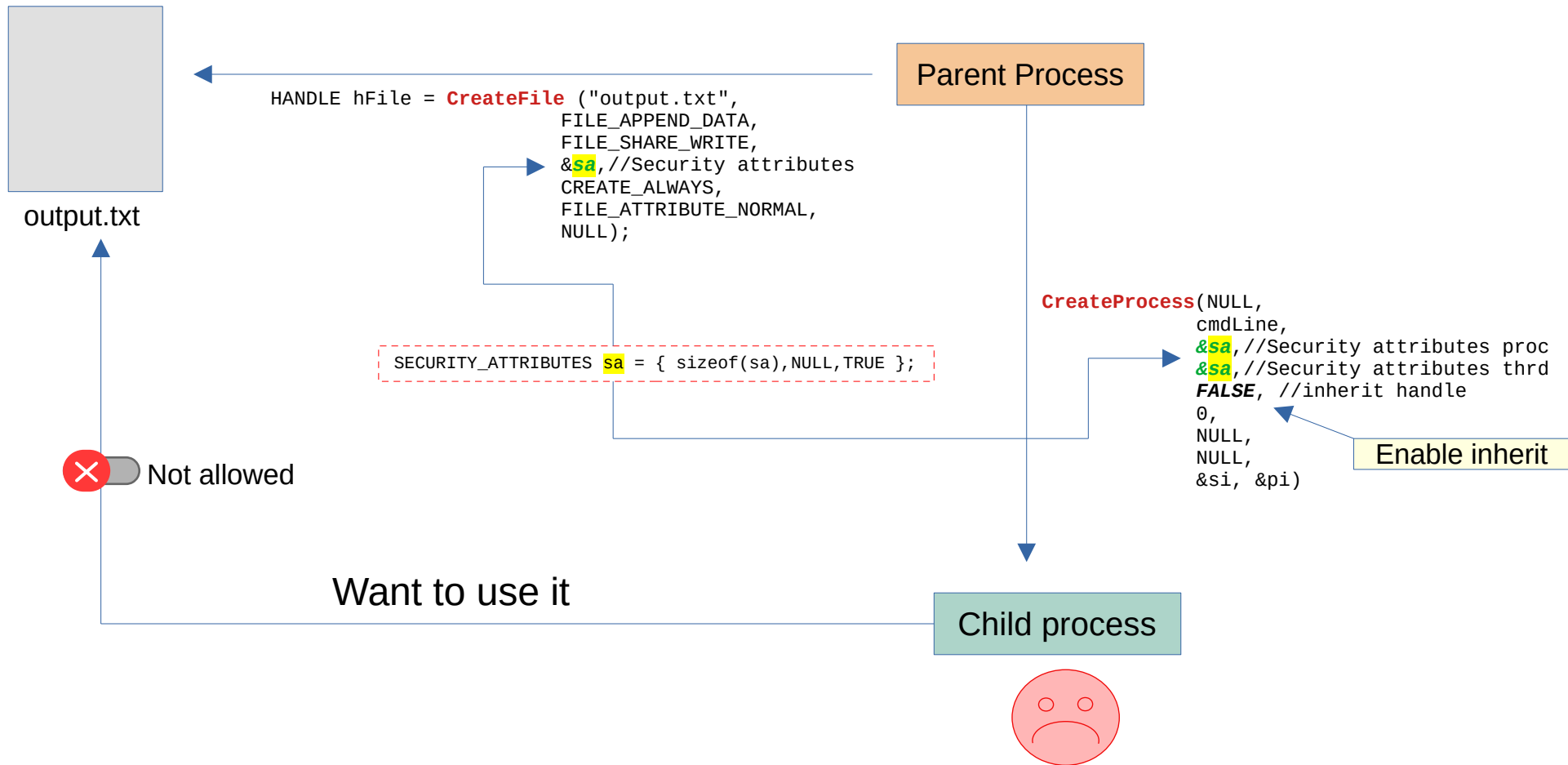
```
CreateProcess(NULL,  
cmdLine,  
NULL, //Security attributes proc  
NULL, //Security attributes thrd  
FALSE, //inherit handle  
0,  
NULL,  
NULL,  
&si, &pi)
```

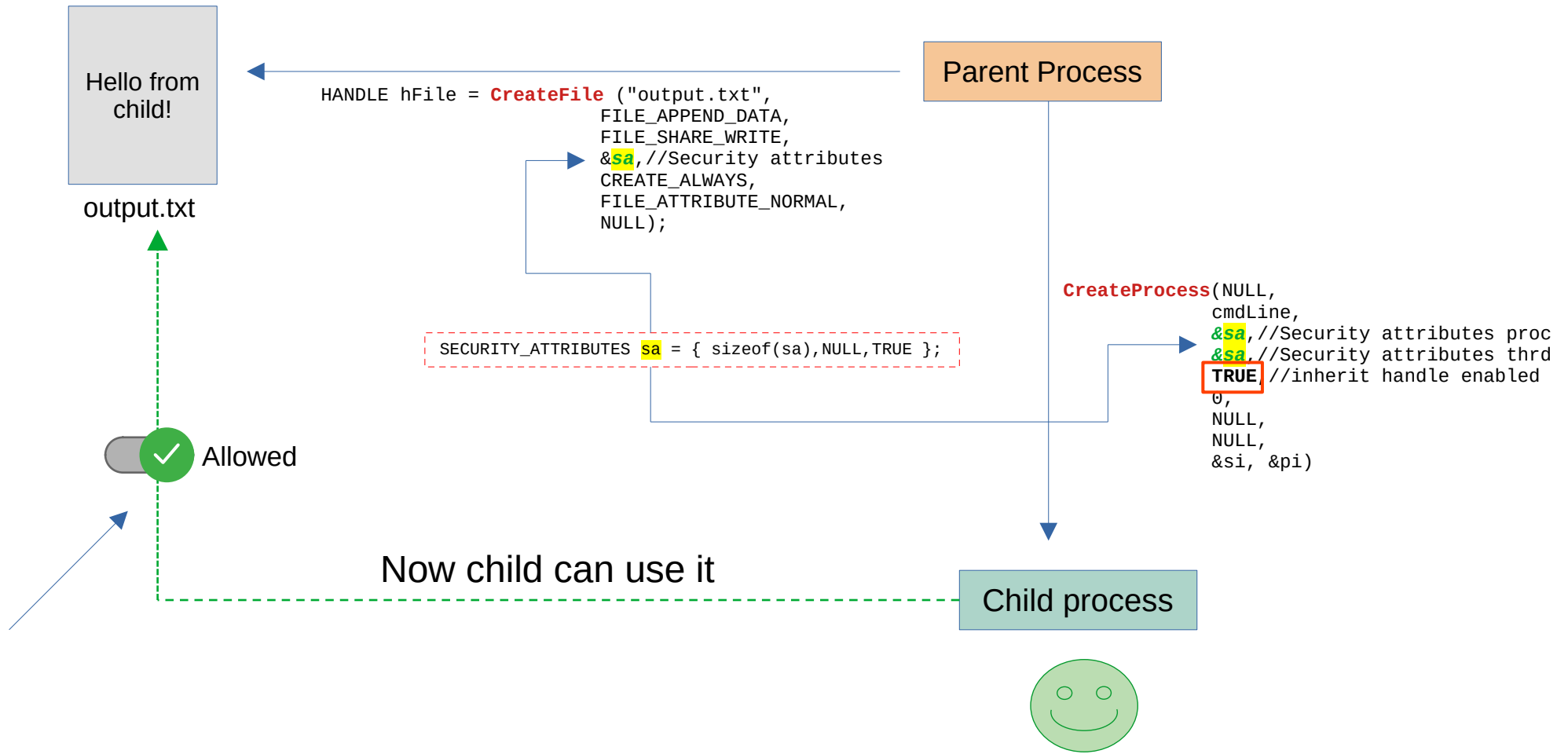
Child process











Parent.c

```
#include <stdio.h>
#include <windows.h>

int main() {

    SECURITY_ATTRIBUTES sa = { sizeof(sa), NULL, TRUE };

    // Create an inheritable file handle

    HANDLE hFile = CreateFile ("output.txt", FILE_APPEND_DATA, FILE_SHARE_WRITE, &sa, CREATE_ALWAYS, FILE_ATTRIBUTE_NORMAL, NULL);
    if (hFile == INVALID_HANDLE_VALUE) {
        printf("Failed to create file. Error: %d\n", GetLastError());
        return 1;
    }

    // Write to the file
    DWORD written;
    WriteFile(hFile, "Hello from Parent!\n", 19, &written, NULL);

    // Convert handle to string (so the child process can receive it)
    char handleStr[20];
    sprintf(handleStr, " %llu", (unsigned long long)hFile); // Add a leading space

    // Prepare process startup information
    STARTUPINFO si = { sizeof(si) };
    PROCESS_INFORMATION pi;

    // Correctly format the command line
    char cmdLine[150];
    sprintf(cmdLine, "\"C:\\\\Users\\\\John\\\\Documents\\\\Temp\\\\child.exe\\\\\"%s\"", handleStr);

    // Create child process
    if (!CreateProcess(NULL, cmdLine, &sa, &sa, TRUE, 0, NULL, NULL, &si, &pi))
    {
        printf("CreateProcess failed. Error: %d\n", GetLastError());
        return 1;
    }
    Sleep(50);
    printf("Parent: Created child process with PID %d\n", pi.dwProcessId);

    // Close handles
    CloseHandle(pi.hProcess);
    CloseHandle(pi.hThread);
    CloseHandle(hFile);
    return 0;
}
```

Parent.c

```
#include <stdio.h>
#include <windows.h>
```

```
int main() {
```

```
    SECURITY_ATTRIBUTES sa = { sizeof(sa), NULL, TRUE };
```

→ Creating security attributes

```
    // Create an inheritable file handle
```

```
    HANDLE hFile = CreateFile ("output.txt", FILE_APPEND_DATA, FILE_SHARE_WRITE, &sa, CREATE_ALWAYS, FILE_ATTRIBUTE_NORMAL, NULL);
    if (hFile == INVALID_HANDLE_VALUE) {
        printf("Failed to create file. Error: %d\n", GetLastError());
        return 1;
    }
```

```
    // Write to the file
```

```
    DWORD written;
    WriteFile(hFile, "Hello from Parent!\n", 19, &written, NULL);
```

```
    // Convert handle to string (so the child process can receive it)
```

```
    char handleStr[20];
    sprintf(handleStr, "%llu", (unsigned long long)hFile); // Add a leading space
```

```
    // Prepare process startup information
```

```
    STARTUPINFO si = { sizeof(si) };
    PROCESS_INFORMATION pi;
```

```
    // Correctly format the command line
```

```
    char cmdLine[150];
    sprintf(cmdLine, "\"C:\\Users\\John\\Documents\\Temp\\child.exe\\\"%s\"", handleStr);
```

```
    // Create child process
```

```
    if (!CreateProcess(NULL, cmdLine, &sa, &sa, TRUE, 0, NULL, NULL, &si, &pi))
```

```
    {
        printf("CreateProcess failed. Error: %d\n", GetLastError());
        return 1;
    }
```

```
    Sleep(50);
    printf("Parent: Created child process with PID %d\n", pi.dwProcessId);
```

```
    // Close handles
```

```
    CloseHandle(pi.hProcess);
    CloseHandle(pi.hThread);
    CloseHandle(hFile);
    return 0;
```

```
}
```

Parent.c

```
#include <stdio.h>
#include <windows.h>
```

```
int main() {
```

```
    SECURITY_ATTRIBUTES sa = { sizeof(sa), NULL, TRUE };
```

```
    // Create an inheritable file handle
```

```
    HANDLE hFile = CreateFile ("output.txt", FILE_APPEND_DATA, FILE_SHARE_WRITE, &sa, CREATE_ALWAYS, FILE_ATTRIBUTE_NORMAL, NULL);
    if (hFile == INVALID_HANDLE_VALUE) {
        printf("Failed to create file. Error: %d\n", GetLastError());
        return 1;
    }
```

```
    // Write to the file
```

```
    DWORD written;
```

```
    WriteFile(hFile, "Hello from Parent!\n", 19, &written, NULL);
```

```
    // Convert handle to string (so the child process can receive it)
```

```
    char handleStr[20];
```

```
    sprintf(handleStr, "%llu", (unsigned long long)hFile); // Add a leading space
```

```
    // Prepare process startup information
```

```
    STARTUPINFO si = { sizeof(si) };
```

```
    PROCESS_INFORMATION pi;
```

```
    // Correctly format the command line
```

```
    char cmdLine[150];
```

```
    sprintf(cmdLine, "\"C:\\Users\\John\\Documents\\Temp\\child.exe\\\"%s\"", handleStr);
```

```
    // Create child process
```

```
    if (!CreateProcess(NULL, cmdLine, &sa, &sa, TRUE, 0, NULL, NULL, &si, &pi))
```

```
    {
```

```
        printf("CreateProcess failed. Error: %d\n", GetLastError());
```

```
        return 1;
```

```
    }
```

```
    Sleep(50);
```

```
    printf("Parent: Created child process with PID %d\n", pi.dwProcessId);
```

```
    // Close handles
```

```
    CloseHandle(pi.hProcess);
```

```
    CloseHandle(pi.hThread);
```

```
    CloseHandle(hFile);
```

```
    return 0;
```

```
}
```

passing security attributes

Parent.c

```
#include <stdio.h>
#include <windows.h>
```

```
int main() {
```

```
    SECURITY_ATTRIBUTES sa = { sizeof(sa), NULL, TRUE };
```

```
    // Create an inheritable file handle
```

```
    HANDLE hFile = CreateFile ("output.txt", FILE_APPEND_DATA, FILE_SHARE_WRITE, &sa, CREATE_ALWAYS, FILE_ATTRIBUTE_NORMAL, NULL);
    if (hFile == INVALID_HANDLE_VALUE) {
        printf("Failed to create file. Error: %d\n", GetLastError());
        return 1;
    }
```

```
    // Write to the file
```

```
    DWORD written;
    WriteFile(hFile, "Hello from Parent!\n", 19, &written, NULL);
```

```
    // Convert handle to string (so the child process can receive it)
```

```
    char handleStr[20];
    sprintf(handleStr, " %llu", (unsigned long long)hFile); // Add a leading space
```

```
    // Prepare process startup information
```

```
    STARTUPINFO si = { sizeof(si) };
    PROCESS_INFORMATION pi;
```

```
    // Correctly format the command line
```

```
    char cmdLine[150];
    sprintf(cmdLine, "\"C:\\Users\\John\\Documents\\Temp\\child.exe\\\"%s\"", handleStr);
```

```
    // Create child process
```

```
    if (!CreateProcess(NULL, cmdLine, &sa, &sa, TRUE, 0, NULL, NULL, &si, &pi))
```

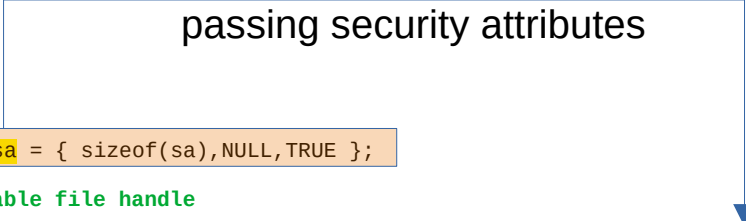
```
    {
        printf("CreateProcess failed. Error: %d\n", GetLastError());
        return 1;
    }
```

```
    Sleep(50);
    printf("Parent: Created child process with PID %d\n", pi.dwProcessId);
```

```
    // Close handles
```

```
    CloseHandle(pi.hProcess);
    CloseHandle(pi.hThread);
    CloseHandle(hFile);
    return 0;
```

passing security attributes



Parent.c

```
#include <stdio.h>
#include <windows.h>
```

```
int main() {
```

```
    SECURITY_ATTRIBUTES sa = { sizeof(sa), NULL, TRUE };
```

```
    // Create an inheritable file handle
```

```
    HANDLE hFile = CreateFile ("output.txt", FILE_APPEND_DATA, FILE_SHARE_WRITE, &sa, CREATE_ALWAYS, FILE_ATTRIBUTE_NORMAL, NULL);
    if (hFile == INVALID_HANDLE_VALUE) {
        printf("Failed to create file. Error: %d\n", GetLastError());
        return 1;
    }
```

```
    // Write to the file
```

```
    DWORD written;
```

```
    WriteFile(hFile, "Hello from Parent!\n", 19, &written, NULL);
```

```
    // Convert handle to string (so the child process can receive it)
```

```
    char handleStr[20];
```

```
    sprintf(handleStr, "%llu", (unsigned long long)hFile); // Add a leading space
```

```
    // Prepare process startup information
```

```
    STARTUPINFO si = { sizeof(si) };
```

```
    PROCESS_INFORMATION pi;
```

```
    // Correctly format the command line
```

```
    char cmdLine[150];
```

```
    sprintf(cmdLine, "\"C:\\Users\\John\\Documents\\Temp\\child.exe\\\"%s\"", handleStr);
```

```
    // Create child process
```

```
    if (!CreateProcess(NULL, cmdLine, &sa, &sa, TRUE, 0, NULL, NULL, &si, &pi))
```

```
    {
        printf("CreateProcess failed. Error: %d\n", GetLastError());
        return 1;
    }
```

```
    Sleep(50);
```

```
    printf("Parent: Created child process with PID %d\n", pi.dwProcessId);
```

```
    // Close handles
```

```
    CloseHandle(pi.hProcess);
```

```
    CloseHandle(pi.hThread);
```

```
    CloseHandle(hFile);
```

```
    return 0;
```

```
}
```

passing security attributes

```
graph TD
    A[passing security attributes] --> B[CreateFile]
    A --> C[CreateProcess]
```


Child.c

```
#include <windows.h>
#include <stdio.h>

int main(int argc, char *argv[]) {

    if (argc != 2) {
        printf("Child: No handle received!\n");
        return 1;
    }

    // Convert string back to HANDLE
    HANDLE hFile = (HANDLE)_strtoi64(argv[1], NULL, 10);

    if (hFile == INVALID_HANDLE_VALUE) {
        printf("Child: Invalid handle received!\n");
        return 1;
    }

    // Write to the inherited file handle

    DWORD bytesWritten;
    const char *message = "Hello from the child process!\n";
    WriteFile(hFile, message, strlen(message), &bytesWritten, NULL);

    printf("Child: Wrote to file using inherited handle.\n");

    // Close the file handle
    CloseHandle(hFile);
    return 0;
}
```