Bypass Import Address Table

https://github.com/JustasMasiulis/lazy_importer/blob/master/include/lazy_importer.hpp

```
C++ Code
#include <windows.h>
#include "lazy_importer.hpp"

int main(){
    PROCESS_INFORMATION pi;
    _STARTUPINFOW si = { sizeof(si) };
    LI_FN(CreateProcessW)(L"C:\\Windows\\System32\\notepad.exe", nullptr, nullptr, nullptr, FALSE, 0, nullptr, nullptr, &si, &pi);
    WaitForSingleObject(pi.hProcess, INFINITE);

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

This code is a C++ program that demonstrates how to use the "lazy_importer" library to dynamically load and call the Windows API functions without explicitly linking against them. It creates a new process for the Notepad application and waits for it to exit. Here's a step-by-step explanation:

1. **Header Includes**:
   - `<windows.h>`: Provides access to Windows API functions and data types.
   - `"lazy_importer.hpp"`: This appears to be a custom header file that provides a mechanism for lazy loading and calling Windows API functions.
2. **`main` Function**:
   - `PROCESS_INFORMATION pi;`: Declares a structure to hold information about the new process.
   - `_STARTUPINFOW si = { sizeof(si) };`: Declares a `_STARTUPINFOW` structure and initializes its size. This structure is used to specify the startup information for the new process.
   - `LI_FN(CreateProcessW)(L"C:\\Windows\\System32\\notepad.exe", nullptr, nullptr, nullptr, FALSE, 0, nullptr, nullptr, &si, &pi);`:
     - This line demonstrates the use of the lazy_importer library. It calls the `CreateProcessW` function, which is used to create a new process.
     - `LI_FN(CreateProcessW)` is a macro or function provided by the "lazy_importer" library that resolves and calls the `CreateProcessW` function dynamically.
     - The function is called with the following parameters:
       - `"C:\\Windows\\System32\\notepad.exe"`: The path to the executable file for the new process (Notepad in this case).
       - `nullptr, nullptr, nullptr`: These parameters are for specifying security attributes and inheritance of handles, but in this case, they are set to `nullptr` to use default values.
       - `FALSE`: Specifies that the new process does not inherit the environment of the calling process.
       - `0`: Creation flags (no special flags).
       - `nullptr, nullptr`: Specifies the current directory and environment block, which are both set to `nullptr`.
       - `&si`: A pointer to the `_STARTUPINFOW` structure, which specifies the startup information.
       - `&pi`: A pointer to the `PROCESS_INFORMATION` structure, which will receive information about the new process.
   - `WaitForSingleObject(pi.hProcess, INFINITE);`: Waits for the new process to finish by waiting for its handle. `pi.hProcess` contains the handle to the new process.
   - `CloseHandle(pi.hProcess);` and `CloseHandle(pi.hThread);`: Closes the process and thread handles to release system resources after the process has exited.

In summary, this code demonstrates the use of the "lazy_importer" library to dynamically load and call the `CreateProcessW` function, which is used to create a new process. It then waits for the new process to finish and closes the associated handles. This technique is often used when you want to avoid statically linking against Windows API functions and load them at runtime for flexibility or compatibility reasons.