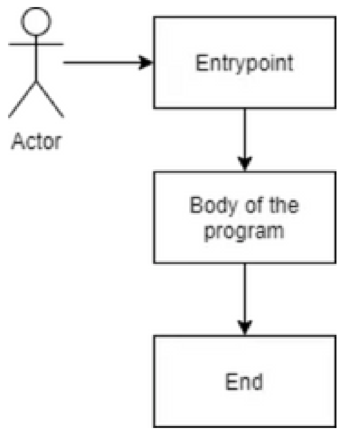
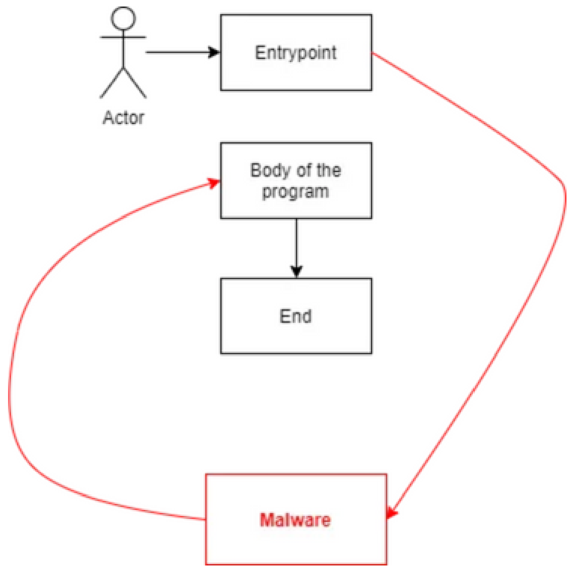
PE (Portable Executable) code injection is a technique used to modify an executable file in a way that introduces new behavior or functionality without the original developer's consent. It's a common method in software exploitation, reverse engineering, and malware creation. Understanding the process in the context of PE code injection involves recognizing how executables are structured and how these structures can be manipulated to execute arbitrary code.

Basically, when you run a trusted & normal executable file. It will be running like this:



However, when you run another file that has a backdoor on it. Then the flow will be like this:



Portable Executable Format

The PE format is a file format for executables, object code, and DLLs, used in Windows operating systems. It includes headers and sections that describe the data and code needed to execute a program.

Now I will be talking on how to do that (second flow) using Lord pE Editor, XVI32 Editor and ollydbg in windows xp machine.

Step 1: Selecting the Target Executable

First, we select the .exe file. In this instance, we are using putty1.exe.

Step 2: Adding a New Section (.kawal) Using LordPE Editor:

Open it in the LordPE Editor and execute the following steps:

Navigate to Lord PE Editor -> PE EDITOR -> open your file: putty1.exe
Click the 'sections' button -> scroll down to the last section & right-click -> choose 'add new section header' -> edit the .NEWSEC section header, rename it to .kawal, and set both the virtual and raw sizes to 6000 -> click 'OK' and save.

Step 3:

Upon opening putty1.exe again, you will encounter an error stating "putty1.exe is not a valid Win32 application."

Step 4: Making Space with XVI32

This issue can be resolved by adding new "00" entries in the file to create space for the newly added section header. To do this, open the file in XVI32, scroll to the bottom of the file, and from the menu below the title bar, select 'edit' -> 'edit string'. Now, in the dialogue box, enter Hex String as "00" & "Insert <<n>> times" as hexadecimal to $6000, then click 'OK' and save the file.

Step 5:

Now, opening putty.exe again should work as expected.

Step 7: Redirecting Execution Flow with OllyDbg

Next, open the file in OllyDbg to modify the entry point of the file.
Step 8:

Copy the first few lines of the file and save them in a notepad or clipboard:

```
0049FBD6 > $ E8 56020000    CALL putty_tr.0049FE31
0049FBDB   .^E9 7AFEFFFF    JMP putty_tr.0049FA5A
0049FBE0  /$ 55             PUSH EBP
0049FBE1  |. 8BEC           MOV EBP,ESP
0049FBE3  |. FF75 08        PUSH DWORD PTR SS:[EBP+8]
0049FBE6  |. E8 0A000000    CALL putty_tr.0049FBF5
0049FBEB  |. F7D8           NEG EAX
```

Step 9:

Press 'M' to view the memory map and locate the codecave address for the section header added in Step 2 using LordPE Editor. In this case, it is .kawal. After finding it, right-click and select 'copy to clipboard' -> 'address'.

The code address of .kawal is 0056F000.

Step 10:

With the .kawal codecave address visible, click on 'Debug' -> 'Restart' -> Confirm with 'Yes'.

Step 11:

Click on the displayed EIP address command, press 'Spacebar' to edit the command, and type "jmp <Code cave Address>". In this case, type:

```
jmp 0056F000
```
Ensure to enable NOPs to allow room for adding malicious code.

Step 12:

To save the modified line to the executable, select the changed lines (highlighted in red), right-click, copy to 'Executable' -> 'Selection', right-click on any row, select 'Save File', enter a filename, and save.

After saving, click 'File Menu' -> Open the newly saved file (putty_JMP).

Step 13:

Upon opening the putty_JMP.exe file again, it will display the following:

```
0049FBD6 > $-E9 25F40C00    JMP putty_tr.0056F000
```

Press F7 on this address to be redirected to the new section (.kawal in this case), where message dialogue commands will be placed.

Step 14: Inserting Custom Code or Payload for Message Box

To create a string for the caption, select a line of codes (approximately 200-300 lines), right-click, choose 'Edit' -> 'Fill With zeros'. Copy to 'executable' -> 'selection' -> save as putty_zeros.exe.

Step 15:

1. Open putty_zeros.exe in OllyDbg, and press F7 on the very first line to navigate to the location for adding the code.

2. Leave 10-11 lines from the top and select 4-5 lines to add the dialogue box's caption.
3. Use 'Edit' -> 'Binary' -> 'Edit' to type the string in ASCII, such as "Welcome Kawalpreet".
4. Repeat the process for the header, labeled as "You are hacked!!!".

Step 16:

Note down the addresses of the caption and header, for example:

Caption's address: 0056F018
Header's address: 0056F02F

Step 17:

Press F7 to assemble.

Step 18:

At the beginning of the section, add the following codes:

```
PUSH 0 # For Handler
PUSH <Address of caption>
PUSH <Address of Header Text>
PUSH 0 # For the type of the box
CALL MessageBoxA
```

This will look like:

```
PUSH 0
PUSH 0056F018
PUSH 0056F02F
PUSH 0
CALL MessageBoxA
```

To edit any command, press the Spacebar, type the command, and press 'Assemble'. After this, select the lines highlighted in red, copy to 'executable' -> 'selection', and save as Putty_messagebox.exe.

Step 19:

Open Putty_messagebox.exe and press Shift+F7.

# Redirecting the command again to source Putty.

The original code copied earlier:

```
0049FBD6 > $ E8 56020000 CALL putty_tr.0049FE31
0049FBDB .^E9 7AFEFFFF JMP putty_tr.0049FA5A
```

Change "CALL putty_tr.0049FE31" to "CALL 0049FE31". "putty_tr" denotes the file name; at the time we copied, the file name was different.

So, just use the address, and the file name will be fetched automatically. In the next line, make it jump to the address of the next command, like this:

```
call 0049FE31
jmp 0049FBDB
```

Step 20: Finalizing and Testing the Injected Executable

Select both lines highlighted in red. Copy to 'executable' -> 'selection'. Right-click, select 'Save file as', and name it Putty_assembled.exe. Opening Putty_assembled will display the Message Dialogue Box, and thereafter, Putty will operate normally.

Troubleshooting and Error Handling Note: If there is an error with the header or caption, open putty_assembled.exe and re-insert the header and caption strings at the positions initially filled with zeros. Then, redirect the commands to these new addresses.

—---

Another simple way to understand it:

PE code injection is a bit like customizing a car. Imagine you have a car (the executable file) that works just fine, driving from point A to point B (its intended functionality).

Now, suppose you want this car to do something extra, like play a specific song whenever you start it (injecting new behavior).

Here's how you do it in simpler terms, using the **car analogy**:

Choose Your Car (Selecting putty1.exe):
You pick a car you want to modify. In our case, it's the program called putty1.exe.

Add a New Speaker System (Adding a New Section with LordPE Editor):
You decide to add a new speaker system to your car to play the song. This is like adding a new section to the executable file, a place where your new code (the song) will live.

Make Space for It (Making Space with XVI32):
The new speaker system needs space, so you adjust the car's interior (add "00" bytes with XVI32) to fit it in without causing any issues when driving (running the program).

Install a Button to Play the Song (Redirecting Execution Flow with OllyDbg):
You wire a button into the car's dashboard that, when pressed, will play your song. In the executable, this is like setting up a command (a jump instruction) that tells

the program, "Hey, go to this new section and do what's there before continuing as normal."

Choose Your Song (Inserting Custom Code or Payload):
Now, you decide which song to play and install it into your new speaker system. This step is where you add your own code or behavior into the newly added section of the program.

Test Drive (Finalizing and Testing):
You take the car out for a spin to see if the song plays correctly when you press the button. Similarly, you run your modified program (putty_assembled.exe) to see if the new code executes as expected.

Adjustments (Troubleshooting and Error Handling):
If the song doesn’t play right, or if it causes any issues with how the car runs, you might need to go back and tweak your setup—maybe adjust the wiring or the placement of the speaker system.

In this simplified version, PE code injection is about taking a program and adding your own custom features to it, much like modifying a car with a new gadget or feature. It requires careful planning and execution to ensure the program continues to run smoothly with the new additions.