ملاحظات عامة حول كتابة Shellcode:

- الـ Shellcode يجب أن يكون مُمثّل بالـ Machine code
- يوجد عدة طرق لكتابة الـ Shellcode ومن ثم تحويله الى الـ Machine code
- فيما يلي بعض الطرق (اليدوية ، بدون الإستعانة بأدوات تحاكي العملية) :

❖ **1 – بإستخدام لغة الأسمبلي Assembly**

1) كتابة الـ Shellcode بلغة الـ Assembly
2) عمل Compile للـ Assembly code ، المخرج < file.obj
3) عمل Disassemble للـ file.obj (الهدف من الخطوة (2) و (3) هو الحصول على الـ Machine code المقابل للـ Assembly code)

مثال (هذا المثال فقط لتوضيح جزئية الـ Machine code ، بغض النظر عما يقوم به كود الأسمبلي المقابل له) :

```
[root@linux-server root]#
[root@linux-server root]# objdump -D a.out | grep -A20 main.:
08048460 <main>:
 8048460:       55                      push   %ebp
 8048461:       89 e5                   mov    %esp,%ebp
 8048463:       83 ec 08                sub    $0x8,%esp
 8048466:       90                      nop
 8048467:       c7 45 fc 00 00 00 00    movl   $0x0,0xfffffffc(%ebp)
 804846e:       89 f6                   mov    %esi,%esi
 8048470:       83 7d fc 09             cmpl   $0x9,0xfffffffc(%ebp)
 8048474:       7e 02                   jle    8048478 <main+0x18>
 8048476:       eb 18                   jmp    8048490 <main+0x30>
 8048478:       83 ec 0c                sub    $0xc,%esp
 804847b:       68 08 85 04 08          push   $0x8048508
 8048480:       e8 93 fe ff ff          call   8048318 <_init+0x38>
 8048485:       83 c4 10                add    $0x10,%esp
 8048488:       8d 45 fc                lea    0xfffffffc(%ebp),%eax
 804848b:       ff 00                   incl   (%eax)
 804848d:       eb e1                   jmp    8048470 <main+0x10>
 804848f:       90                      nop
 8048490:       b8 00 00 00 00          mov    $0x0,%eax
 8048495:       c9                      leave
 8048496:       c3                      ret
[root@linux-server root]# _
```

هذه هي الجزئية الخاصة بالـ Machine code بعد عملية الـ Disassemble

4) إضافة الـ Prefix التالي : **\x** قبل قيم الـ HEX في الـ Shellcode

❖ **2- بإستخدام أحد اللغات البرمجية عالية المستوى (high-level language)**

1) كتابة الـ Shellcode بأحدى اللغات عالية المستوى (لغة C مثلًا)
2) عمل Compile للـ file.c ، المخرج > file.exe
3) عمل Decompile للـ file.exe (الهدف من الخطوة (2) و (3) هو الحصول على الـ Machine code)

- في حالة إستخدام إحدى لغات البرمجة عالية المستوى لكتابة الـ Shellcode يجب الإنتباه بأن الـ Compiler قد يقوم بإدراج بعض تعليمات الـ Assembly
- التعليمات أو الإضافات التي يقوم بإدراجها الـ Compiler تختلف بإختلاف لغة البرمجة + الـ Compiler نفسه

ملاحظات خاصة بكتابة Shellcode على نظام Windows :

- البرامج (exe.) الخاصة بنظام التشغيل Windows تحتوي على مكتبات (أو ملفات .dll) مضمنة معها
- ملفات الـ .dll في الأساس ماهي إلا ملف برمجي قد يُكتب بلغة C أو لغات أخرى
- الوظيفة الأساسية لملفات الـ .dll دعم عمل للبرنامج الأساسي (exe.) والغرض منها يختلف بحسب طبيعة البرنامج والوظيفة التي يؤديها، على سبيل المثال برنامج (FTP) مدمج معه ملف .dll معين يُتيح للبرنامج الوصول الى بعض موارد النظام
- ملفات الـ dll. قد يتم مشاركتها (أي إمكانية الوصول لها) بين أكثر من برنامج (أي أكثر من برنامج يستطيع إستخدام نفس ملف الـ dll. في نفس الوقت)
- في مرحلة كتابة الـ Shellcode قد نقوم بكتابة Shellcode الهدف منه هو "فتح جلسة في الـ Command prompt في النظام"، يوجد بعض الأمور التي يجب التأكد منها في حالة أردنا كتابة مثل هذا الـ Shellcode وهي :
  - ❖ التأكد من وجود ملف .dll يحتوي على دالة (function) تقوم بهذا الغرض، حيث أننا سنقوم بإستخدام هذه الدالة الموجودة في ملف الـ .dll ونداءها (في داخل الـ Shellcode الخاص بنا) عن طريق جلب عنوان الذاكرة الخاص بها، وهذا العنوان نجده من خلال البحث في ملف الـ .dll (نقوم بعملية disassemble له)
  - ❖ في حالة عدم وجود أي من أساليب الحماية مثل ASLR فالعناوين الخاصة بالدوال في ملفات الـ .dll تكون ثابته (على سبيل المثال لو قمنا بعمل disassemble لملف الـ .dll في جهاز آخر سنحصل على نفس العنوان)
  - ❖ لكن في حالة وجود أحد أساليب الحماية من "إستغلال ثغرات فيضان ذاكرة التخزين المؤقت" فالعناوين الخاصة بالدوال تختلف (مع كل عملية تشغيل للجهاز يتم إعطاء عناوين جديدة)