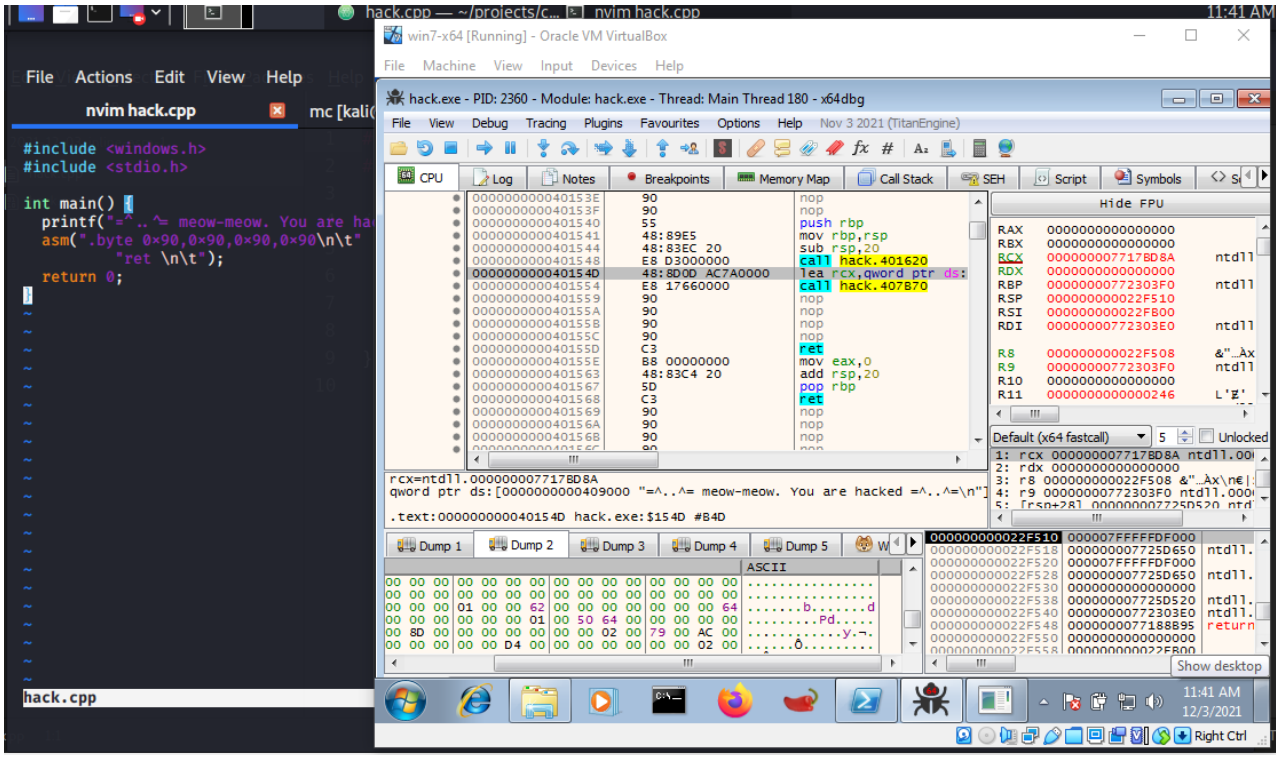
Run shellcode via inline ASM. Simple C++ example.

cocomelonc.github.io/tutorial/2021/12/03/inline-asm-1.html

December 3, 2021

1 minute read

Hello, cybersecurity enthusiasts and white hackers!



```
#include <windows.h>
#include <stdio.h>

int main() {
  printf("=^..^= meow-meow. You are ha
  asm(".byte 0x90,0x90,0x90,0x90\n\t"
      "ret \n\t");
  return 0;
}
```

```
000000000040153E   90              nop
000000000040153F   90              nop
0000000000401540   55              push rbp
0000000000401541   48:89E5         mov rbp,rsp
0000000000401544   48:83EC 20      sub rsp,20
0000000000401548   E8 D3000000     call hack.401620
000000000040154D   48:8D0D AC7A0000 lea rcx,qword ptr ds:
0000000000401554   E8 17660000     call hack.407B70
0000000000401559   90              nop
000000000040155A   90              nop
000000000040155B   90              nop
000000000040155C   90              nop
000000000040155D   C3              ret
000000000040155E   B8 00000000     mov eax,0
0000000000401563   48:83C4 20      add rsp,20
0000000000401567   5D              pop rbp
0000000000401568   C3              ret
0000000000401569   90              nop
000000000040156A   90              nop
000000000040156B   90              nop
```

```
rcx=ntdll.000000007717BD8A
qword ptr ds:[0000000000409000 "=^..^= meow-meow. You are hacked =^..^=\n"]

.text:000000000040154D hack.exe:$154D #B4D
```

```
RAX  0000000000000000
RBX  0000000000000000
RCX  000000007717BD8A  ntdll
RDX  0000000000000000
RBP  00000000772303F0  ntdll
RSP  000000000022F510
RSI  000000000022FB00
RDI  00000000772303E0  ntdll

R8   000000000022F508
R9   00000000772303F0  ntdll
R10  0000000000000000
R11  0000000000000246
```

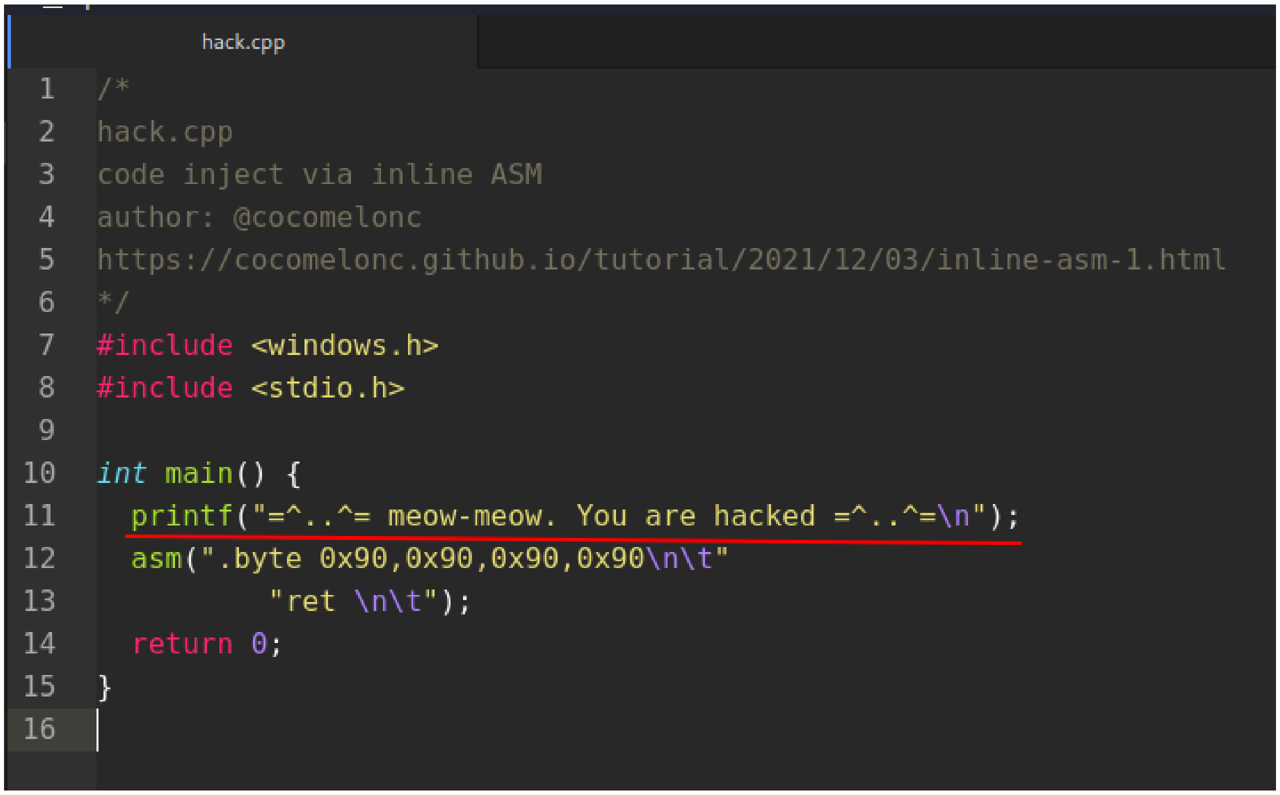
This is a very short post and it describes an example usage inline assembly for running shellcode in malware.

Let’s take a look at example C++ source code of our malware:

```
/*
hack.cpp
code inject via inline ASM
author: @cocomelonc
https://cocomelonc.github.io/tutorial/2021/12/03/inline-asm-1.html
*/
#include <windows.h>
#include <stdio.h>

int main() {
  printf("=^..^= meow-meow. You are hacked =^..^=\n");
  asm(".byte 0x90,0x90,0x90,0x90\n\t"
          "ret \n\t");
  return 0;
}
```

As you can see, the logic is simplest, I'm just adding 4 NOP instructions and printing `meow-meow` string before. I can easily find the shellcode in the debugger based on this `meow` string:

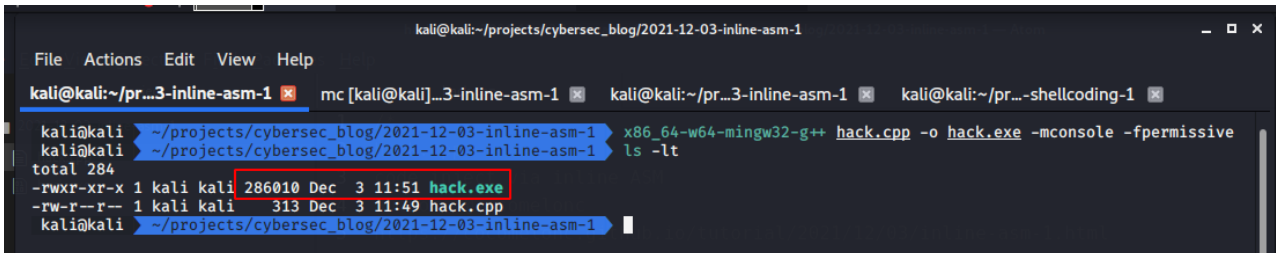


```
/*
hack.cpp
code inject via inline ASM
author: @cocomelonc
https://cocomelonc.github.io/tutorial/2021/12/03/inline-asm-1.html
*/
#include <windows.h>
#include <stdio.h>

int main() {
  printf("=^..^= meow-meow. You are hacked =^..^=\n");
  asm(".byte 0x90,0x90,0x90,0x90\n\t"
          "ret \n\t");
  return 0;
}
```

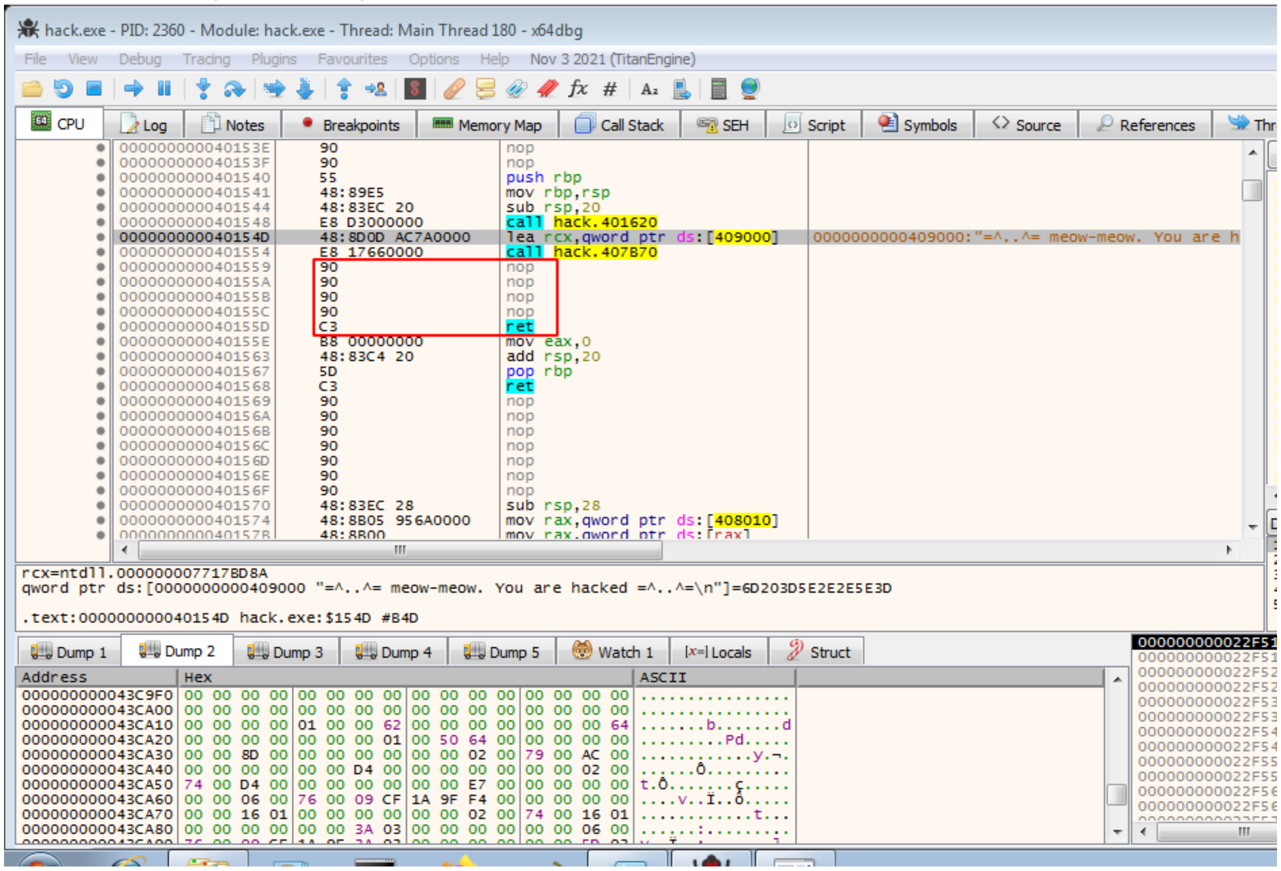
Let's go to compile:

```
x86_64-w64-mingw32-g++ hack.cpp -o hack.exe -mconsole -fpermissive
```



```
kali@kali ~/projects/cybersec_blog/2021-12-03-inline-asm-1 x86_64-w64-mingw32-g++ hack.cpp -o hack.exe -mconsole -fpermissive
kali@kali ~/projects/cybersec_blog/2021-12-03-inline-asm-1 ls -lt
total 284
-rwxr-xr-x 1 kali kali 286010 Dec  3 11:51 hack.exe
-rw-r--r-- 1 kali kali    313 Dec  3 11:49 hack.cpp
kali@kali ~/projects/cybersec_blog/2021-12-03-inline-asm-1
```

And run in `x96dbg` (on `Windows 7 x64` in my case):



```
hack.exe - PID: 2360 - Module: hack.exe - Thread: Main Thread 180 - x64dbg

000000000040153E  90                 nop
000000000040153F  90                 nop
0000000000401540  55                 push rbp
0000000000401541  48:89E5            mov rbp,rsp
0000000000401544  48:83EC 20         sub rsp,20
0000000000401548  E8 D3000000        call hack.401620
000000000040154D  48:8D0D AC7A0000   lea rcx,qword ptr ds:[409000]   0000000000409000:"=^..^= meow-meow. You are h
0000000000401554  E8 17660000        call hack.407B70
0000000000401559  90                 nop
000000000040155A  90                 nop
000000000040155B  90                 nop
000000000040155C  90                 nop
000000000040155D  C3                 ret
000000000040155E  B8 00000000        mov eax,0
0000000000401563  48:83C4 20         add rsp,20
0000000000401567  5D                 pop rbp
0000000000401568  C3                 ret
0000000000401569  90                 nop
000000000040156A  90                 nop
000000000040156B  90                 nop
000000000040156C  90                 nop
000000000040156D  90                 nop
000000000040156E  90                 nop
000000000040156F  90                 nop
0000000000401570  48:83EC 28         sub rsp,28
0000000000401574  48:8B05 956A0000   mov rax,qword ptr ds:[408010]

rcx=ntdll.000000007717BD8A
qword ptr ds:[0000000000409000 "=^..^= meow-meow. You are hacked =^..^=\n"]=6D203D5E2E2E5E3D

.text:000000000040154D hack.exe:$154D #B4D

Address           Hex                                              ASCII
000000000043C9F0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
000000000043CA00  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
000000000043CA10  00 00 00 00 01 00 00 62 00 00 00 00 00 00 00 64  .......b.......d
000000000043CA20  00 00 00 00 00 00 00 01 00 50 64 00 00 00 00 00  .........Pd.....
000000000043CA30  00 00 8D 00 00 00 00 00 00 00 02 00 79 00 AC 00  ............y.¬.
000000000043CA40  00 00 00 00 00 00 D4 00 00 00 00 00 00 00 02 00  ......Ô.........
000000000043CA50  74 00 D4 00 00 00 00 00 00 00 E7 00 00 00 00 00  t.Ô.......ç.....
000000000043CA60  00 00 06 00 76 00 09 CF 1A 9F F4 00 00 00 00 00  ....v..Ï..ô.....
000000000043CA70  00 00 16 01 00 00 00 00 00 00 02 00 74 00 16 01  ............t...
000000000043CA80  00 00 00 00 00 00 3A 03 00 00 00 00 00 00 06 00  ......:.........
```

As you can see, the highlighted instructions are my NOP instructions, so everything work perfectly as expected.

The reason why it's good to have this technique in your arsenal is because it does not require you to allocate new `RWX` memory to copy your shellcode over to by using `VirtualAlloc` which is more popular and suspicious and which is more closely investigated by the blue teamers.

I hope this post spreads awareness to the blue teamers of this interesting technique, and adds a weapon to the red teamers arsenal.

inline assembly
source code in Github

> This is a practical case for educational purposes only.

Thanks for your time and good bye!

PS. All drawings and screenshots are mine