Blue Team Labs Online: Injection Series Part 3

The following writeup is for Injection Series Part 3 on Blue Team Labs Online, it's an medium level difficult challenge that involves reverse engineering a binary using IDA among other tools. I really love this injection series, so I highly recommend you complete them all.

How many arguments does the sample take?

If we open up the binary in cutter and go to the main function, we can see that the sample takes one argument:

```
int32_t main (void) {
    /* [00] -r-x section size 4096 named .text */
    eax = *(argv);
    edx = "message";
    ecx = *((eax + 4));
    eax = *((eax + 4));
    do {
        bl = *(eax);
        if (bl != *(edx)) {
            goto label_0;
        }
        if (bl == 0) {
            goto label_1;
        }
        bl = *((eax + 1));
        if (bl != *((edx + 1))) {
            goto label_0;
        }
        eax += 2;
        edx += 2;
    } while (bl != 0);
```

Again, what is the size of the shellcode?

If you up the binary in IDA and scroll down the main function, we can see a call to VirtualAlloc along with all the parameters being pushed to the stack. The dwSzie is what is being allocated for the shellcode:

```
push    40h ; '@'       ; flProtect
push    1000h           ; flAllocationType
push    288             ; dwSize
push    0               ; lpAddress
mov     edi, eax
call    ds:VirtualAlloc
```

Therefore, we can assume that the shellcode is 288 bytes.

In VirtualAlloc what does the flAllocationType value represents?

If you look at the flAllocationType value in the image of the previous question, we can see the hex value 1000. If you search for VirtualAlloc and look at its documentation, we can see that this value represents MEM_COMMIT:

Value	Meaning
MEM_COMMIT 0x00001000	Allocates memory charges (from the overall size of memory and the paging files on disk) for the specified reserved memory pages. The function also guarantees that when the caller later initially accesses the memory, the contents will be zero. Actual physical pages are not allocated unless/until the virtual addresses are actually accessed. To reserve and commit pages in one step, call **VirtualAllocEx** with `MEM_COMMIT | MEM_RESERVE`. Attempting to commit a specific address range by specifying **MEM_COMMIT** without **MEM_RESERVE** and a non-**NULL** *lpAddress* fails unless the entire range has already been reserved. The resulting error code is **ERROR_INVALID_ADDRESS**. An attempt to commit a page that is already committed does not cause the function to fail. This means that you can commit pages without first determining the current commitment state of each page. If *lpAddress* specifies an address within an enclave, *flAllocationType* must be **MEM_COMMIT**.

What is the argument required by the sample to run the shellcode?

In the beginning of the main function, we can see “message” being stored in the edx register:

```
mov     edx, offset aMessage ; "message"
push    ebx
mov     ecx, [eax+4]
mov     eax, ecx
```

This string is then used for a series of comparisons, and from what I can gather, if the string “message” is not supplied, the program will go through another execution route.

What is the payload in Metasploit that would have been used to generate the shellcode?

After looking at different Metasploit payloads for windows, I came across /windows/messagebox which appears to be what we are dealing with in this sample.

What is the API used to create a wait object?

Just after the call to VirtualAlloc, there is a call to CreateThreadpoolWait:

```
push    0               ; pcbe
push    0               ; pv
push    esi             ; pfnwa
call    ds:CreateThreadpoolWait
```

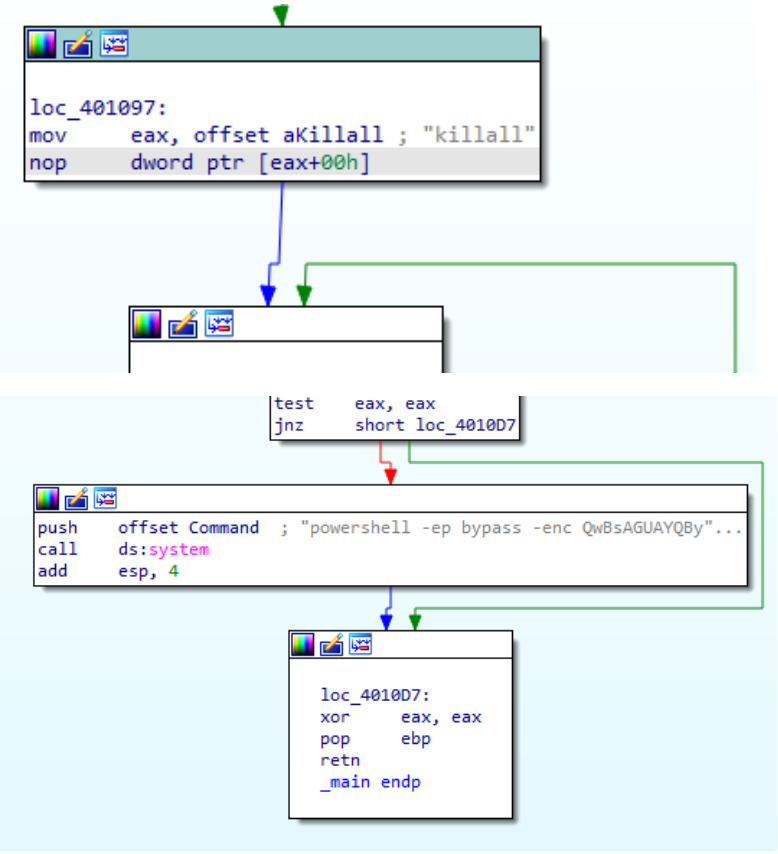
If you look at the Microsoft documentation for this function, you can see that it “Creates a new wait object”, therefore the answer is CreateThreadpoolWait.

What is the library function used to copy shellcode between memory blocks?

After the call to VirtualAlloc, we can see a call to memmove. If you look at C documentation, you can determine that memmove is used to copy a block of memory from one location to another.

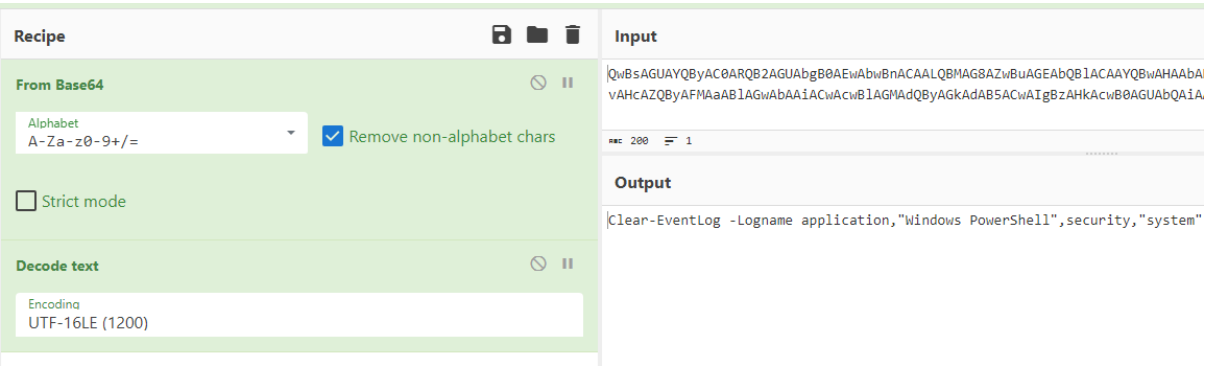
What argument to the sample invokes powershell process?

If the argument "killall" is supplied, this invokes the PowerShell process:



After decoding the powershell, list the log names as in the order in the script

To decode the PowerShell command, all you need to do is copy the string, paste the Base64 encoded text into a tool like Cyberchef, and then use the From Base64 and Decode Text recipes:



You can see that this PowerShell command clears the logs for application, Windows PowerShell, security, and system.