Error due to buffer overflow and how to fix it

Basically, buffer overflow is caused by the user sending too much data to a program and part of this data is forced to store out of memory that the programmer has allocated.

Network administration **- Basically, buffer overflow often comes from a single cause. This is because the user sends too much data to a program and part of this data is forced to be stored out of the memory that the programmer provided for that program. The buffer overflow process can cause many problems, including one that we need to consider is that when the cache stores data to a certain extent, hackers can run Program code on the system.**

In this article we will explore a buffer overflow situation that hackers can exploit to run code on the system. Then we will look at Data Execution Preventions (DEP), a feature built into the Windows operating system that has a buffer overflow function.

Recognize the buffer overflow phenomenon

In order to understand the buffer overflow phenomenon, we must master high-level programming languages ?? such as C or C ++, as well as have a deep knowledge about the operation of memory stacks.

When writing a program, one of the things that programmers need to carefully consider is that the buffer space size is allocated to specific functions. Buffer memory is an adjoining free area of ??memory that a program can use to store data that other functions can use. Let's consider the following code example:

```
void main()
{
    char bufferA[50];
    char bufferB[16];

    printf("What is your name?\n");

    gets(bufferA);

    strcpy(bufferB, bufferA);

    return;
}
```

Figure 1: AC function is very vulnerable to buffer overflow.

This function is very clear, starting with the declaration of the two variables *bufferA* and *bufferB of* size *50* and *16 respectively* . This program displays a question to the user who asks for the name and uses the get function to get the input. Then the data that the user provides is copied from *bufferA* to the cache parameter and this function is completed.

With such a simple program it may be a bit too much to assume that it can be affected by every attack. However, the problem here lies in the *gets* function. Because the function *gets* no self-checking limits, it is difficult to confirm that the information entered into *bufferA* does not exceed *50* characters. If the user enters more than *50* characters, the program will crash.

The *strcopy* function will copy the data in *bufferA* to *bufferB* . However, *bufferB* is smaller in size than *bufferA* , which means that even if the user enters less than *50* characters, it can still be more than *16* characters in *bufferA* , and when copying to *bufferB* will cause overflow. Buffer and program will also collapse. This small program not only has one but two buffer overflow vulnerabilities.

Exploiting memory overflow

Next we will explore the conditions that cause memory overflow and problems arising from memory overflows? In cases where the cache is overflowed, the data that spills out of the specified cache will have to be stored somewhere else. This data will flow into neighboring memory areas, and usually the program will fail because it cannot handle additional data. On the other hand, when this error is understood by someone with Assembly language (a low-level programming language used in computer programming, microprocessors, microcontrollers and integrated circuits) and stack sets Remember to exploit, everything is worse. In this situation, hackers can cause buffer overflows in a way that they can create their own system commands, then convert these commands into low-level byte code then send them to this program in the format accordingly, these commands will be run.

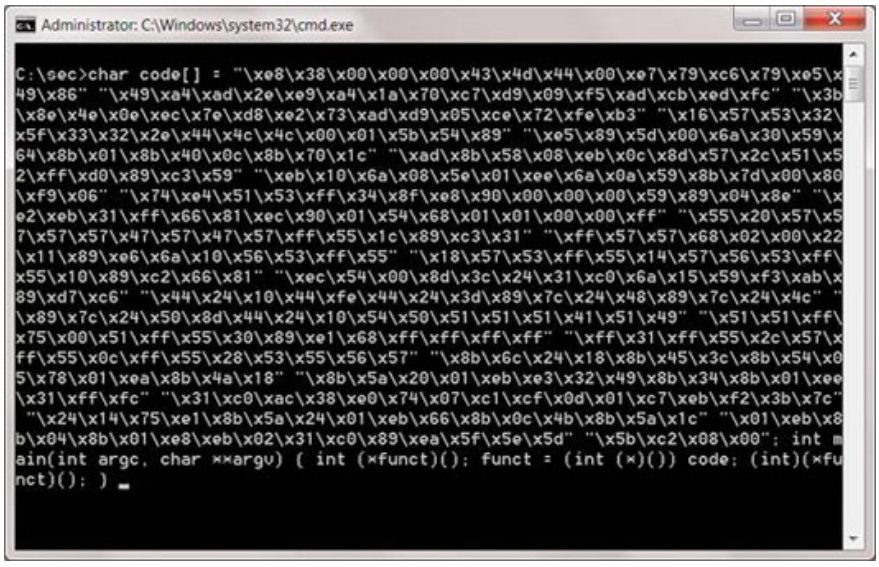

Figure 2: A written Assembly and C Shellcode sample is returned a C: prompt of Windows.

Now the code is run in the context of the user's initial vulnerable application. That means that if the program is run by the system administrator, the association code also runs in the context of a system administrator. Depending on the size of the cache, hackers can combine different types of code. These types of commonly used code are those called Shellcode. This code will return a Shell (eg a Windows C: prompt) to the person who ran the code. In a suitable context, the person running the code will have full control of the workstation. Buffer overflows can take many forms and scales, a person proficient in controlling the stack will gain full control over all vulnerable systems.

Data Execution Prevention

The simplest method to block the possibility of exploiting the vulnerability caused by the buffer overflow that programmers often use is to ensure that the programming code is secure. In fact, this is not an automated process because it requires a lot of time and effort for re-checking the code to ensure that the integrity of the program code is maintained, so the amount The command line is proportional to the time and effort required. From that requirement, Microsoft developed a feature called Data Execution Prevention (DEP).

DEP, a security feature introduced in Windows XP SP2, is designed to block applications from running code in the inaccessible area of ??memory. DEP appears in both Hardware-based DEP and Software-based DEP configurations.

Hardware-based DEP

DEP is arguably the most secure when using Hardware-based DEP. In this case the processor will mark every memory location as 'unenforceable' if the location does not contain executable code. The purpose of this is that DEP will block any code that runs in unenforceable areas.

The main problem with using Hardware-based DEP is that it is only supported by a few processes. This is possible thanks to the NX feature of Intel's AMD and XD processors.

Software-based DEP

When non-existent hardware-based DEPs, Software-based DEP must be used. This DEP type is built into the Windows operating system. Software-based DEP works by detecting when exceptions are entered by programs and ensuring that these exceptions are a valid part of this program before allowing them to process.

Configure Data Execution Prevention

In Windows 7, DEP can be configured in the System **Control Panel** . Go to **Control Panel | Advanced System Settings** . Select the **Data Execution Prevention** tab.

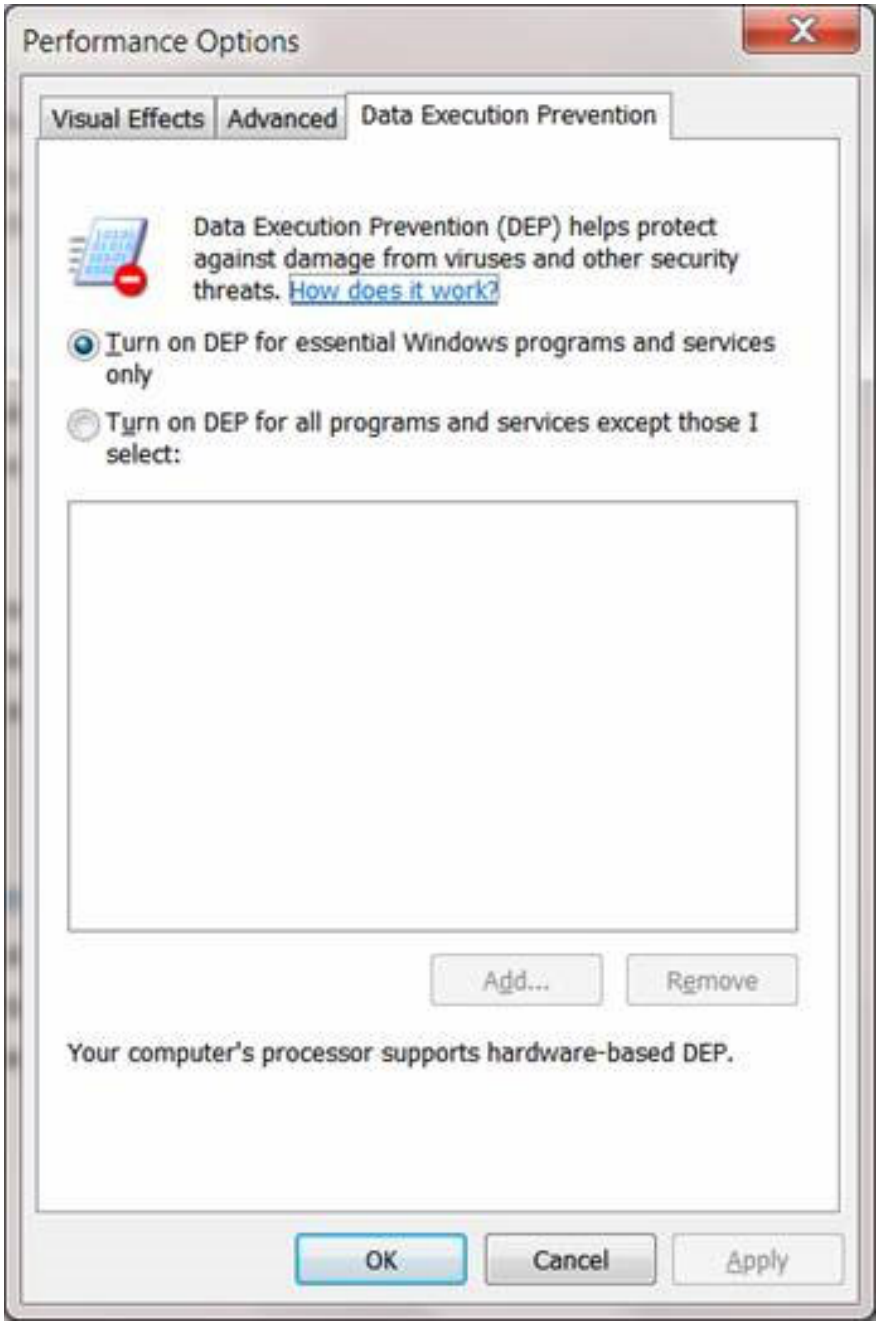

Figure 3: DEP default configuration in Windows 7.

DEP has two configuration options. The default option in Figure 3 is called OptIn configuration. This option only applies DEP for critical system programs and services. This is a low level of security (called an OptOut configuration), but if you want to use a higher level of security you should choose the second option, which applies DEP for all programs and translations service on the system. Note that the bottom of this dialog box indicates whether the CPU you are using supports Hardware-based DEP.

Although the configuration of DEP applies to all programs and services with the highest level of security, it is important to remember that it contains compatibility issues. First, some programs will perform orthodox functions that can be blocked by DEP by the method they operate. In these cases we have to create an exception

for that program. On the DEP configuration dialog, click the Add button and select the appropriate executable files. Microsoft suggests that only registered drivers should be used to prevent possible compatibility issues when selecting this second option.

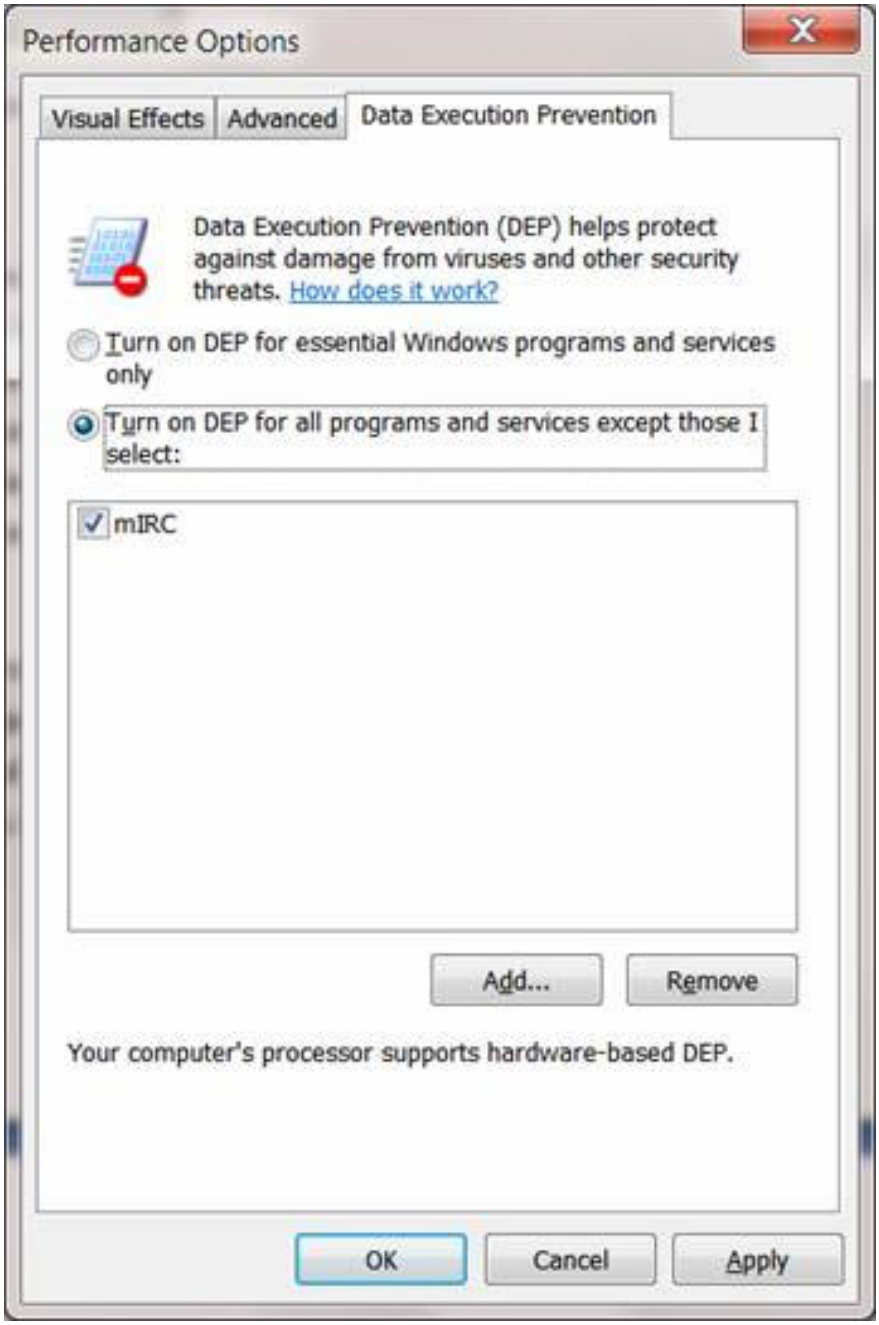

Figure 4: Configure DEP for all services and create an exception program.

After executing and saving the configuration, when the executable code appears in the DEP executable area, the message shown in Figure 5 will be displayed.

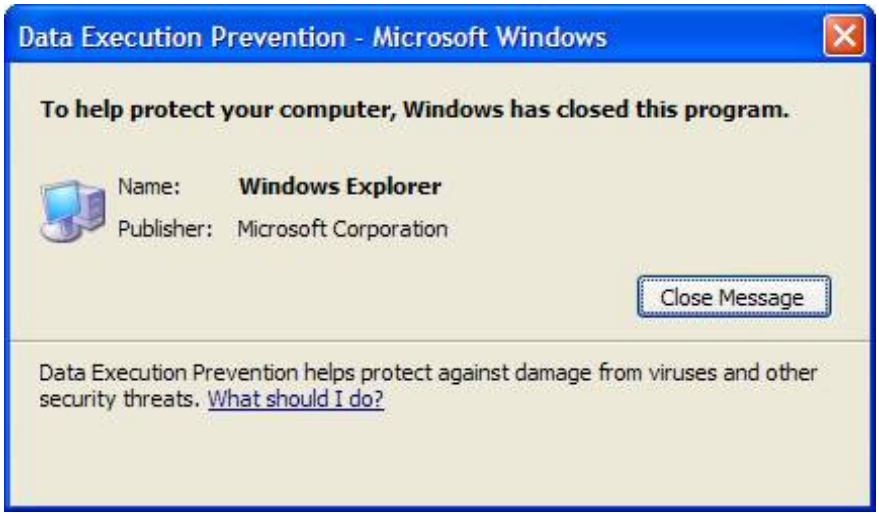

Figure 5: DEP intercepts execution in Windows Explorer.

Conclude

Buffer overflows seem to be nothing but a threat to the most noticeable system security capabilities. Basically, hackers often target this error to exploit the system. If you are a programmer, you should limit this error by ensuring code checking and code security measures. If you are a system administrator, you can use DEP to prevent possible security risks.

You finished reading the article "**Error due to buffer overflow and how to fix it**" edited by the **TipsMake** team. We hope this article has provided you with many useful tech tips and tricks. You can search for similar articles on tips and guides. Thank you for reading and for following us regularly.