It rather involved being on the other side of this airtight hatchway: Invalid parameters from one security level crashing code at the same security level (yet again)

devblogs.microsoft.com/oldnewthing/20150422-00

April 22, 2015

Raymond Chen

It's the bogus vulnerability that keeps on giving. This time a security researcher found a horrible security flaw in `SysAllocStringLen`:

> The `SysAllocStringLen` function is vulnerable to a denial-of-service attack. [Long description of reverse-engineering deleted.]
>
> The `SysAllocStringLen` does not check the length parameter properly. If the provided length is larger than the actual length of the buffer, it may encounter an access violation when reading beyond the end of the buffer. Proof of concept:
>
> ```
> SysAllocStringLen(L"Example", 0xFFFFFF);
> ```
>
> Credit for this vulnerability should be given to XYZ Security Labs. Copyright © XYZ Security Labs. All rights reserved.

As with other issues of this type, there is no elevation. The attack code and the code that crashes are on the same side of the airtight hatchway. If your goal was to make the process crash, then instead of passing invalid parameters to the `SysAllocStringLen` function, you can launch the denial of service attack much more easily:

```
int __cdecl main(int, char**)
{
    ExitProcess(0);
}
```

Congratulations, you just launched a denial-of-service attack against yourself.

In order to trigger an access violation in the `SysAllocStringLen` function, you must already have had enough privilege to run code, which means that you already have enough privilege to terminate the application without needing the `SysAllocStringLen` function.

Once again, we have a case of MS07-052: Code execution results in code execution.[1]

Earlier in the series:

- Episode 2.
- Episode 1.

Bonus bogus vulnerability report:

> The `DrawText` function is vulnerability to a denial-of-service attack because it does not validate that the `lpchText` parameter is a valid pointer. If you pass `NULL` as the second parameter, the function crashes. We have found many functions in the system which are vulnerable to the same issue.

[1] Now, of course, if there were some way you could externally induce a program into passing invalid parameters to the `SysAllocStringLength` function, then you'd be onto something. But even then, the vulnerability would be in the program that is passing the invalid parameters, not in the `SysAllocStringLength` function itself.

Raymond Chen

Follow