**Advisory Information**

|  |
| --- |
| Title : Ahnlab V3 Internet security suite Kernel Pool Overflow  Version : Ahnlab V3 Internet security 8.0.7.5 (Build 1373) – MedCoreD.sys 2.2.2.69  Vendor : <http://www.ahnlab.com>  Impact : Moderate  Contact : [moghimi.ahmad@gmail.com](mailto:moghimi.ahmad@gmail.com), [me@mallocat.com](mailto:me@mallocat.com)  Twitter : @mall0cat  CVE : … |

**Vulnerability Information**

|  |
| --- |
| Class  1 – Pool Overflow  Impact  Improper sanitization of certain ioctl requests for MedCoreD.sys driver lead to a pool memory overflow condition that may allow local attacker to execute privileged code in kernel mode or cause denial of service leading to Operating system BSOD.  Remote  No  Local  Yes |

**Vulnerability Detail**

Ahnlab V3 installs *Med engine core* driver *MedCoreD.sys* and the driver is accessible through normal user. So a normal user can send ioctl request to this driver by getting a handle to the device with the path: *"* [*\\\\.\\MeDCoreD\_V3IS80*](file:///\\\\.\\MeDCoreD_V3IS80)*".*

Subroutine sub\_6CB70 in module MedCoreD.sys is responsible for handling some of the ioctrl codes. For ioctl code 0xA3350014 the function reach the following code:

|  |
| --- |
| .text:0006CF69 mov ecx, [ebp+NumberOfBytes]  .text:0006CF6C push ecx ; NumberOfBytes  .text:0006CF6D mov edx, [ebp+inbuffer2]  .text:0006CF70 push edx ; wchar\_t \*  .text:0006CF71 mov eax, [ebp+inpbuffer]  .text:0006CF74 push eax ; wchar\_t \*  .text:0006CF75 call KuConvertSymLinkToDevName |

Subroutine KuConvertSymLinkToDevName takes three arguments as input. The first and second argument are the same and are pointers to the input buffer of ioctl request and will be treated as Unicode data. And the third argument is input size of ioctl request.

The mentioned subroutine reach the following code and allocate a fixed len buffer with size of 0x7D0:

|  |
| --- |
| .text:000A8BD3 push ' kdD' ; Tag  .text:000A8BD8 push 7D0h ; NumberOfBytes  .text:000A8BDD push 0 ; PoolType  .text:000A8BDF call ds:ExAllocatePoolWithTag  .text:000A8BE5 mov [ebp+P], eax  .text:000A8BE8 cmp [ebp+P], 0  .text:000A8BEC jnz short loc\_A8BF8 |

If the allocation success it jump to the following block of code:

|  |
| --- |
| .text:000A8BF8 loc\_A8BF8: ; CODE XREF: KuConvertSymLinkToDevName+3Cj  .text:000A8BF8 mov eax, [ebp+arg\_0]  .text:000A8BFB push eax ; wchar\_t \*  .text:000A8BFC mov ecx, [ebp+P]  .text:000A8BFF push ecx ; wchar\_t \*  .text:000A8C00 call wcscpy |

The subroutine call the insecure wcscpy function without any check on length of the string. So by passing a Unicode string longer than buffer size it would be possible to cause pool overflow.

**Exploitation**

Exploitation of the vulnerability is possible through faking the next pool header and forcing the next pool as a free chunk and then abuse the unlinking algorithm when the subroutine free the previously allocated memory through the following code path:

|  |
| --- |
| .text:000A8CBF mov edx, [ebp+P]  .text:000A8CC2 push edx ; P  .text:000A8CC3 call ds:ExFreePool |

By abusing the unlinking functionality it would be possible to get a write4 condition to arbitrary memory address in kernel. Of course the proof of concept code use another method as Pool Blade that is much better.