Advisories

AIDA64 Elevation of Privileges via Arbitrary Physical Memory Mapping

Summary

The AIDA Kernel Driver does not validate data received from the IRP->SystemBuffer.

Some IOCTL codes implement a logic where parameters arriving from the SystemBuffer are directly passed without validation to a routine that:

- Execute the `MmMapIoSpace` function which takes physical address, length, and cache type (hardcoded to 0) and maps specified physical memory into system memory.
- Execute `IoAllocateMdl` that uses the virtual address from `MmMapIoSpace` and the same Length value to associate the created MDL (Memory Descriptor List) with an IRP (I/O Request Packet).
- Then calls `MmBuildMdlForNonPagedPool`, that takes the newly created MDL and initialize the memory for the buffer.
- Finally, a call to `MmMapLockedPagesSpecifyCache`, which takes the allocated physical memory and maps it to a user-mode buffer.

Any user on the system can fully control the Physical Address and the size passed to the `MmMapIoSpace`, and gets back the associated mapped user-mode address.

This technique can be successfully exploited using a physical memory “scanning” approach to find an elevated process token (Ruben Boonen - IBM X-Force), egregiously explained by h0mbre

Affected by the vulnerability was the 0x80112104 IOCTL code.

The weakness was discovered during February 2024 and was assigned CVE-2024-26507.

Vulnerable code

IOCTL Dispatch

```
...
if (IOCTL_Code == 0x80112104) {
  if (SystemBufferSize == 0x48) {
    (IRP->IoStatus).field0_0x0.Status = -0x3ffffff3;
  }
  else {
    BOOL res = AllocatePageMdl(*(DWORD*)Irp->AssociatedIrp.SystemBuffer,
                               *(DWORD*)(Irp->AssociatedIrp.SystemBuffer + 0x4),
                               *(UINT*)(Irp->AssociatedIrp.SystemBuffer + 0x8),
                               (UINT64)&(Irp->AssociatedIrp.SystemBuffer + 0x0C),
                               (PVOID)&(Irp->AssociatedIrp.SystemBuffer + 0x14),
                               (PVOID)&(Irp->AssociatedIrp.SystemBuffer + 0x1C));
    if (res == FALSE) {
      (IRP->IoStatus).field0_0x0.Status = -0x3fffffc2;
    }
    else {
      (IRP->IoStatus).Information = 0xFFFF;
      (IRP->IoStatus).field0_0x0.Status = 0;
    }
  }
  goto LAB_0001ba56;
}
...
```

Vulnerable Function

```
BOOL AllocatePageMdl(DWORD phyAddressLow,DWORD phyAddressHigh,DWORD dwSize,LPVOID *lpPhyAddress
                     LPVOID *lpUserBuffer,PMDLX *pPmdl){
  PHYSICAL_ADDRESS PhysicalAddress;
  PVOID pvVar1;
  PMDL MemoryDescriptorList;

  PhysicalAddress.field0.HighPart = phyAddressHigh;
  PhysicalAddress.field0.LowPart = phyAddressLow;
  pvVar1 = MmMapIoSpace(PhysicalAddress,(ulonglong)dwSize,MmNonCached);
```

```
  *lpPhyAddress = pvVar1;
  if (pvVar1 != (PVOID)0x0) {
    MemoryDescriptorList = IoAllocateMdl(pvVar1,dwSize,'\0','\0',(PIRP)0x0);
    *pPmdl = MemoryDescriptorList;
    if (MemoryDescriptorList != (PMDL)0x0) {
      MmBuildMdlForNonPagedPool(MemoryDescriptorList);
      pvVar1 = MmMapLockedPagesSpecifyCache
                         (*pPmdl,'\x01',MmNonCached,(PVOID)0x0,0,NormalPagePriority);
      *lpUserBuffer = pvVar1;
      if (pvVar1 != (PVOID)0x0) {
        return 1;
      }
    }
  }
  return 0;
}
```

Proof-of-Concept

During the review, it was possible for a low-privileged user in low integrity to elevate his privileges to NT System.

Proof of concept:

- GitHub Source Code: CVE-2024-26507
- YouTube video: CVE-2024-26507

Remediation

No fix has been released or planned to be released for this bug.

References

- Token Abuse for Privilege Escalation in Kernel
- Kernel Exploitation - _EPROCESS Hunt
- Kernel Exploitation - Kernel-2-User Mode Mapping Vulnerability

Copyright © 2019-2020 Alessandro Magnosi. Distributed by an MIT license.