

CDipper / CVE-2025-67246 Public

<> Code Issues Pull requests Actions Projects Security and quality

1 Branch 0 Tags Go to file Go to file <> Code

CDipper add CVE publication	c9c7de8 · 3 months ago	
README.assets	add CVE publication	3 months ago
ComputerZ_x64.sys	add CVE publication	3 months ago
README.md	add CVE publication	3 months ago
poc.c	add CVE publication	3 months ago

README

CVE-2025-67246

Vulnerability Title:LuDaShi Incorrect Access Control

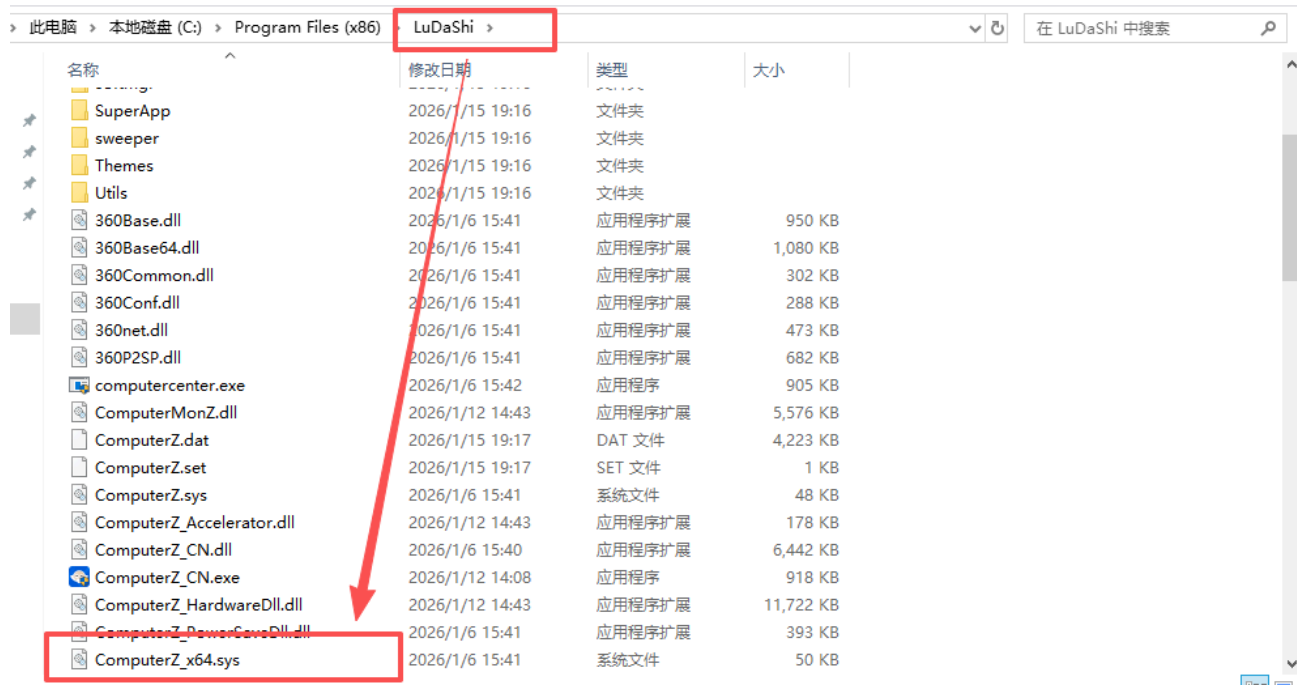
Affected versions:Less than 6.1026.4505.112

Discovery time:December 2, 2025

Discoverer:ZhouRui

Analysis Report:

LuDaShi is a well-known free system utility software that provides free hardware authentication, computer (mobile phone) stability assurance, and system performance enhancement. The **ComputerZ_x64.sys** driver in LuDaShi and its affiliated system products (computer performance optimization, system performance monitoring) contains data that can read the lower 4GB kernel address.



This driver exposes an IOCTL interface(0xF1002508) to user space. This interface does not adequately validate the passed memory address when processing user-provided input parameters. User-space processes can pass an arbitrary lower 4GB address value to the driver through this interface. The driver then uses this address to read physical memory content and returns the result to the user-space caller. Because this address parameter lacks effective access control checks, attackers can use it to read system memory regions that are normally inaccessible through the kernel virtual address space, resulting in the leakage of sensitive information. This issue allows local administrators to obtain system memory data without additional privileges by leveraging the loaded vulnerable driver, thus compromising the operating system's memory access isolation mechanism.

This driver provides the application layer with a usable symbolic link to the device object(ComputerZ). Further down, in the dispatch function sub_11008 for handling IOCTL.

```

1 NTSTATUS __stdcall DriverEntry(_DRIVER_OBJECT *DriverObject, PUNICODE_STRING RegistryPath)
2 {
3     NTSTATUS result; // eax
4     int v4; // edi
5     PDEVICE_OBJECT DeviceObject; // [rsp+40h] [rbp-98h] BYREF
6     struct _UNICODE_STRING DestinationString; // [rsp+48h] [rbp-90h] BYREF
7     struct _UNICODE_STRING SymbolicLinkName; // [rsp+58h] [rbp-80h] BYREF
8     WCHAR SourceString[20]; // [rsp+68h] [rbp-70h] BYREF
9     WCHAR Dst[24]; // [rsp+90h] [rbp-48h] BYREF
10
11     memmove(SourceString, L"\\Device\\ComputerZ", 0x24ui64);
12     memmove(Dst, L"\\DosDevices\\ComputerZ", 0x2Cui64);
13     RtlInitUnicodeString(&DestinationString, SourceString);
14     RtlInitUnicodeString(&SymbolicLinkName, Dst);
15     result = IoCreateDevice(DriverObject, 0, &DestinationString, 0xF100u, 0, 0, &DeviceObject);
16     if ( result >= 0 )
17     {
18         v4 = IoCreateSymbolicLink(&SymbolicLinkName, &DestinationString);
19         if ( v4 >= 0 )
20         {
21             DriverObject->MajorFunction[6] = (PDRIVER_DISPATCH)sub_11000;
22             DriverObject->MajorFunction[14] = (PDRIVER_DISPATCH)sub_11008;
23             DriverObject->DriverUnload = (PDRIVER_UNLOAD)sub_11264;
24             return 0;
25         }
26     }
27     else
28     {
29         IoDeleteDevice(DeviceObject);
30         return v4;
31     }
32     return result;
33 }

```

When IoControlCode is **0xF102508**, the following code will be executed. It maps arbitrary kernel address data in the lower 4GB to the application layer via MmMapIoSpace. An attacker can construct a Proof-of-Concept (POC) to read arbitrary size data from any physical address in the lower 4GB and return it to the application layer.

```

923     v15 = *(unsigned int *)&MasterIrp->Size + 1;
924     if ( Length < (unsigned int)v15 )
925     {
926 LABEL_25:
927         v4 = 0xC0000206;
928         goto LABEL_53;
929     }
930     PhysicalAddress.QuadPart = *(unsigned int *)&MasterIrp->Type;
931     v16 = MmMapIoSpace(PhysicalAddress, v15, MmNonCached);
932     if ( v16 )
933     {
934         v17 = MasterIrp;
935         v18 = v16;
936         for ( i = v15 >> 2; i; --i )
937         {
938             *(_DWORD *)&v17->Type = *v18++;
939             v17 = (struct _IRP *)((char *)v17 + 4);
940         }
941         MmUnmapIoSpace(v16, v15);
942         v4 = 0;
943     }
944     else
945     {
946         v4 = 0xE0000000;
947     }
948     v5 = v15;
949 LABEL_53:
950     v21 = Irp;
951     Irp->IoStatus.Information = v5;
952     v21->IoStatus.Status = v4;
953     IoCompleteRequest(v21, 0);
954     return v4;
955 }

```

Load this driver, run the POC code in this repository, and the attack effect is as follows. In this example, I tried to read 8 bytes of data from physical address 0xF0000, but the amount of data read is arbitrary.

Releases

No releases published

Packages

No packages published

Contributors 1



CDipper CDipper

Languages

● C 100.0%