NORTHWAVE
CYBER SECURITY

Driver vulnerability research

Getting started with vulnerability research on Windows Kernel drivers

Jan-Jaap Korpershoek

Agenda

Time	Subject
12:00 – 12:03	Introduction
12:03 – 12:06	Our research
12:06 – 12:15	Details about drivers
12:15 – 12:25	Common vulnerabilities
12:25 – 12:28	Automated tools
12:28 – 12:30	Conclusion

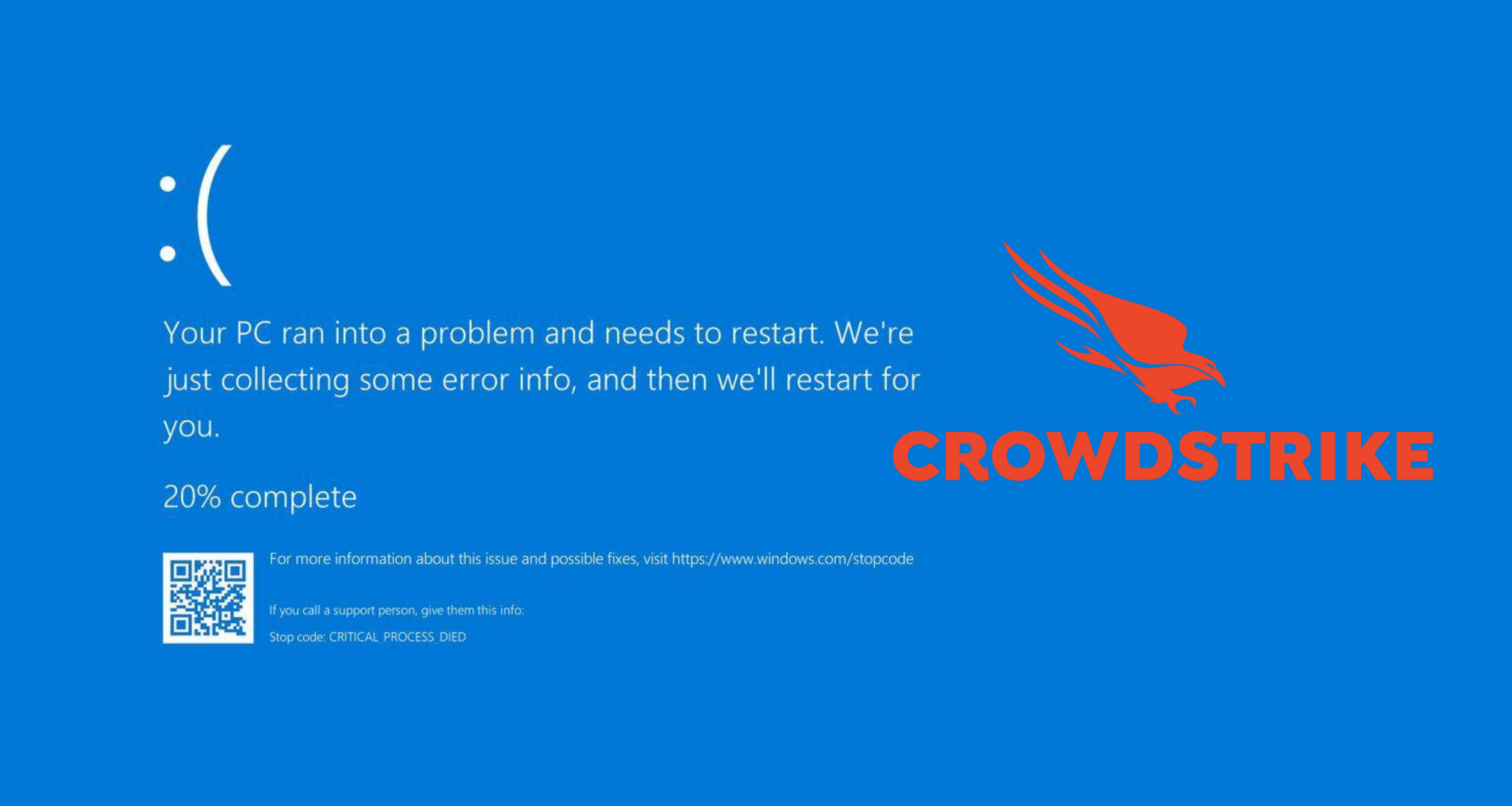
User mode

Kernel mode

User

Admin

Kernel



:(

Your PC ran into a problem and needs to restart. We're just collecting some error info, and then we'll restart for you.

20% complete

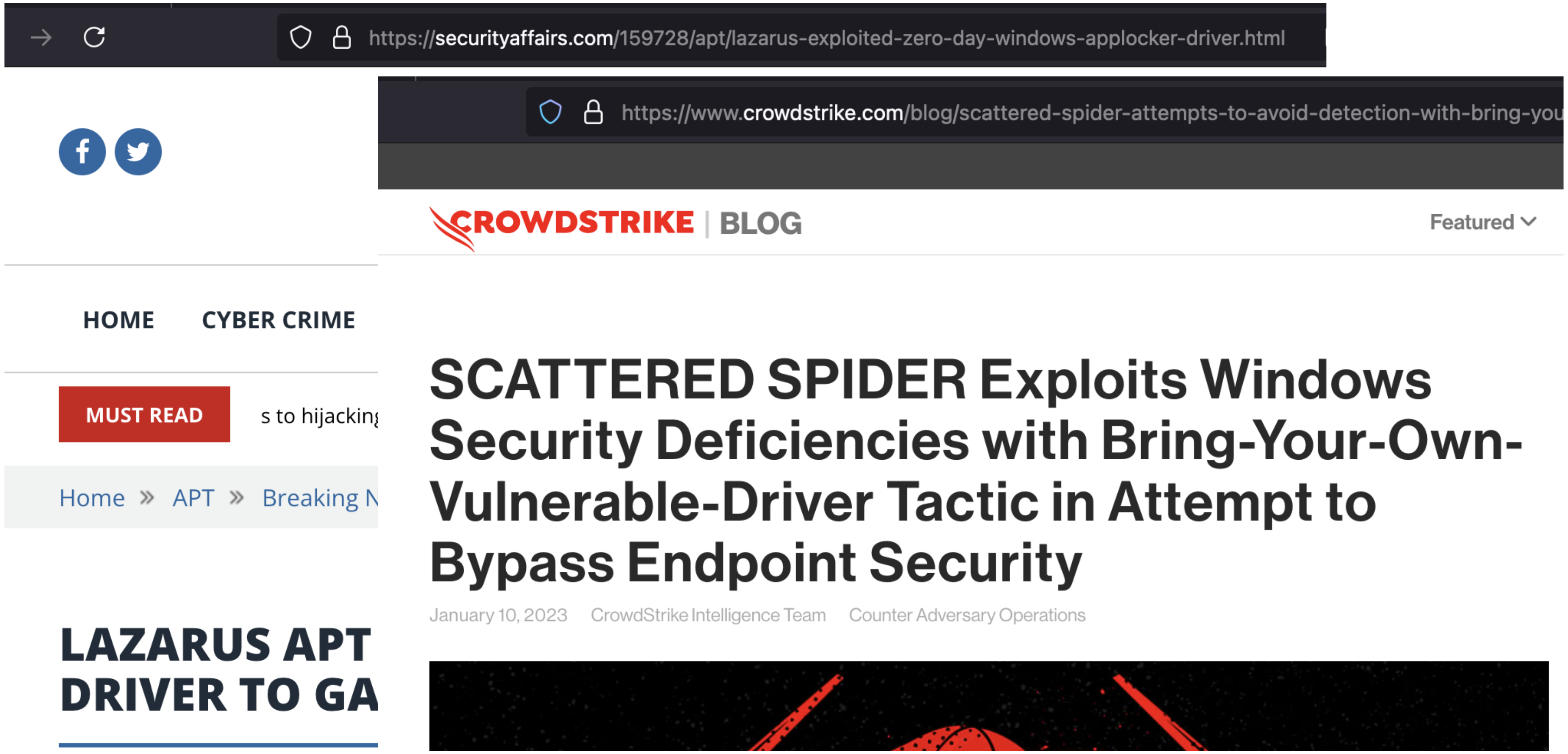
For more information about this issue and possible fixes, visit https://www.windows.com/stopcode

If you call a support person, give them this info:

Stop code: CRITICAL_PROCESS_DIED

CROWDSTRIKE

Threat actors abusing kernel drivers


https://securityaffairs.com/159728/apt/lazarus-exploited-zero-day-windows-applocker-driver.html

HOME CYBER CRIME

MUST READ s to hijacking

Home » APT » Breaking N

LAZARUS APT
DRIVER TO GA

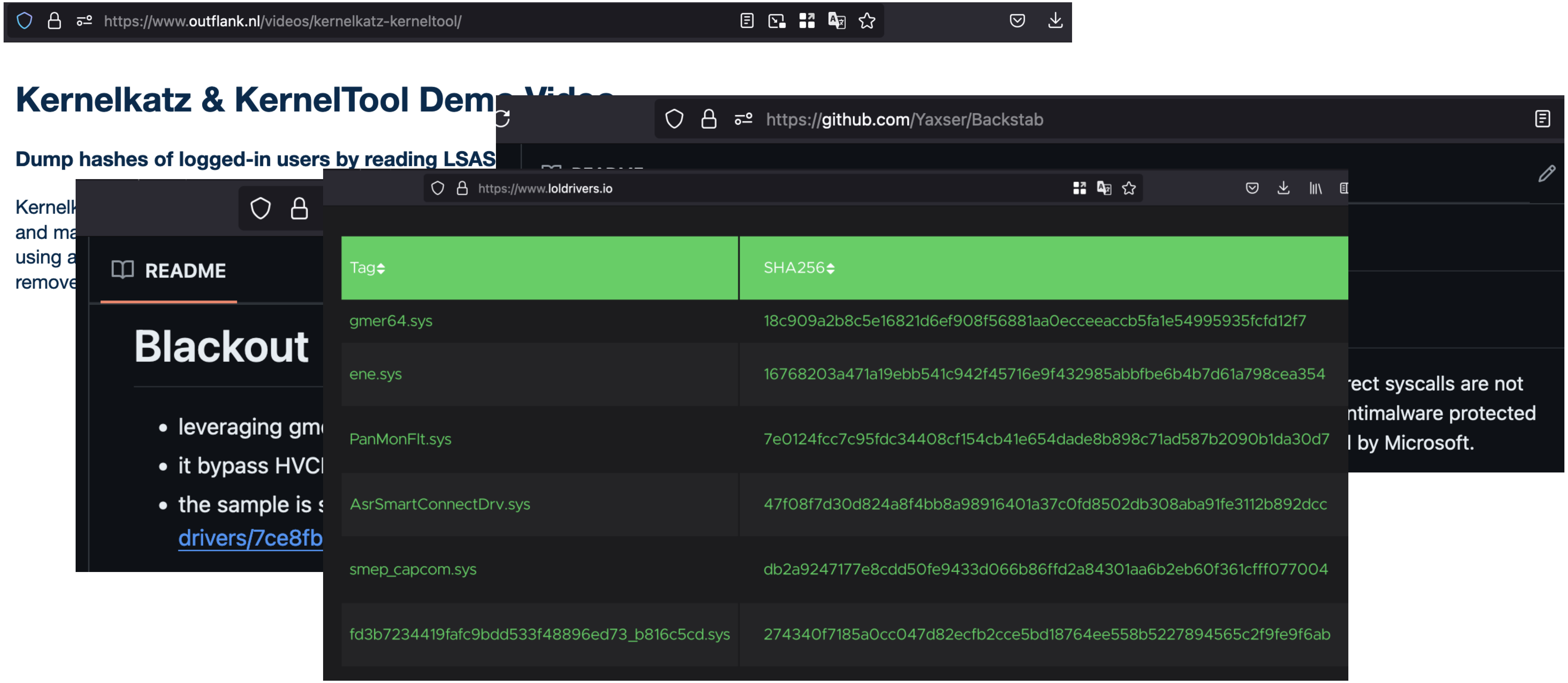
https://www.crowdstrike.com/blog/scattered-spider-attempts-to-avoid-detection-with-bring-you

CROWDSTRIKE | BLOG Featured

SCATTERED SPIDER Exploits Windows Security Deficiencies with Bring-Your-Own-Vulnerable-Driver Tactic in Attempt to Bypass Endpoint Security

January 10, 2023 CrowdStrike Intelligence Team Counter Adversary Operations

Security tooling



https://www.outflank.nl/videos/kernelkatz-kerneltool/

Kernelkatz & KernelTool Dem

Dump hashes of logged-in users by reading LSAS

Kernelk
and ma
using a
remove

https://github.com/Yaxser/Backstab

ect syscalls are not
ntimalware protected
l by Microsoft.

README

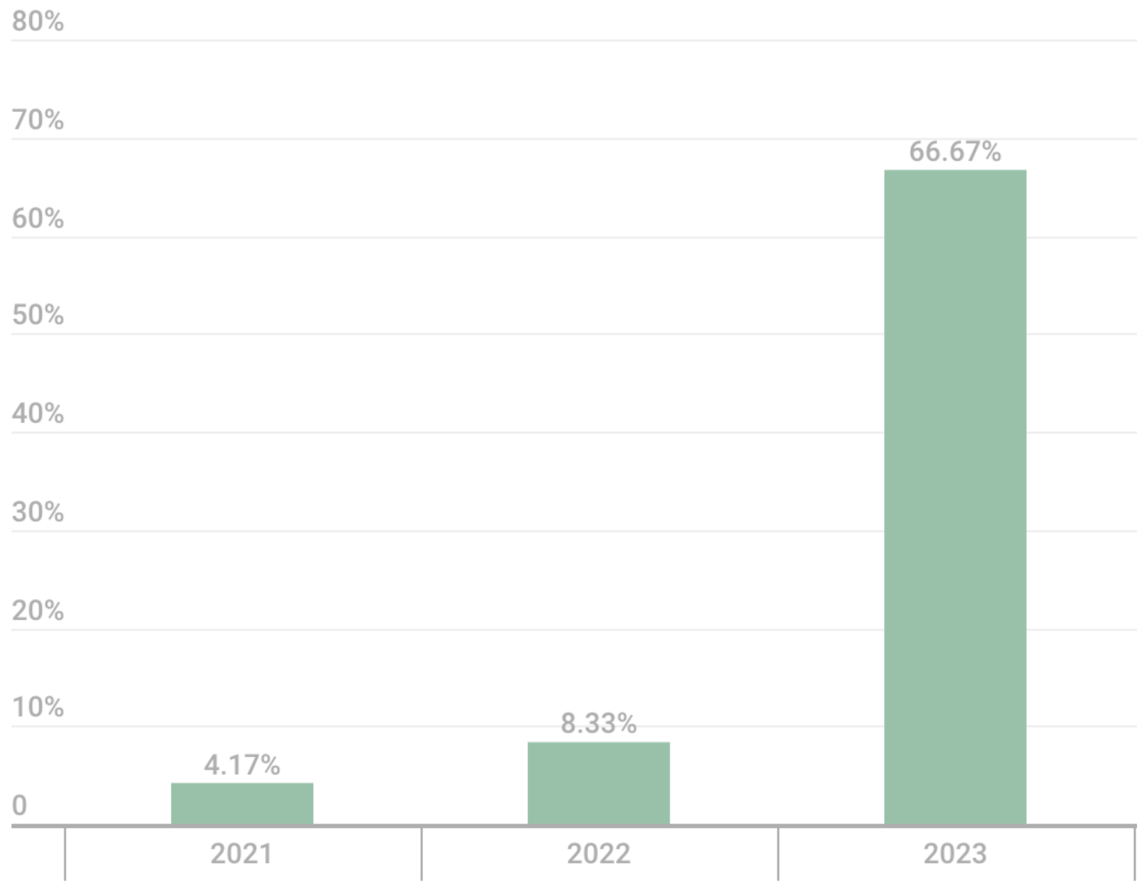
Blackout

- leveraging gm
- it bypass HVC
- the sample is
  drivers/7ce8fb

https://www.loldrivers.io

Tag	SHA256
gmer64.sys	18c909a2b8c5e16821d6ef908f56881aa0ecceeaccb5fa1e54995935fcfd12f7
ene.sys	16768203a471a19ebb541c942f45716e9f432985abbfbe6b4b7d61a798cea354
PanMonFlt.sys	7e0124fcc7c95fdc34408cf154cb41e654dade8b898c71ad587b2090b1da30d7
AsrSmartConnectDrv.sys	47f08f7d30d824a8f4bb8a98916401a37c0fd8502db308aba91fe3112b892dcc
smep_capcom.sys	db2a9247177e8cdd50fe9433d066b86ffd2a84301aa6b2eb60f361cfff077004
fd3b7234419fafc9bdd533f48896ed73_b816c5cd.sys	274340f7185a0cc047d82ecfb2cce5bd18764ee558b5227894565c2f9fe9f6ab

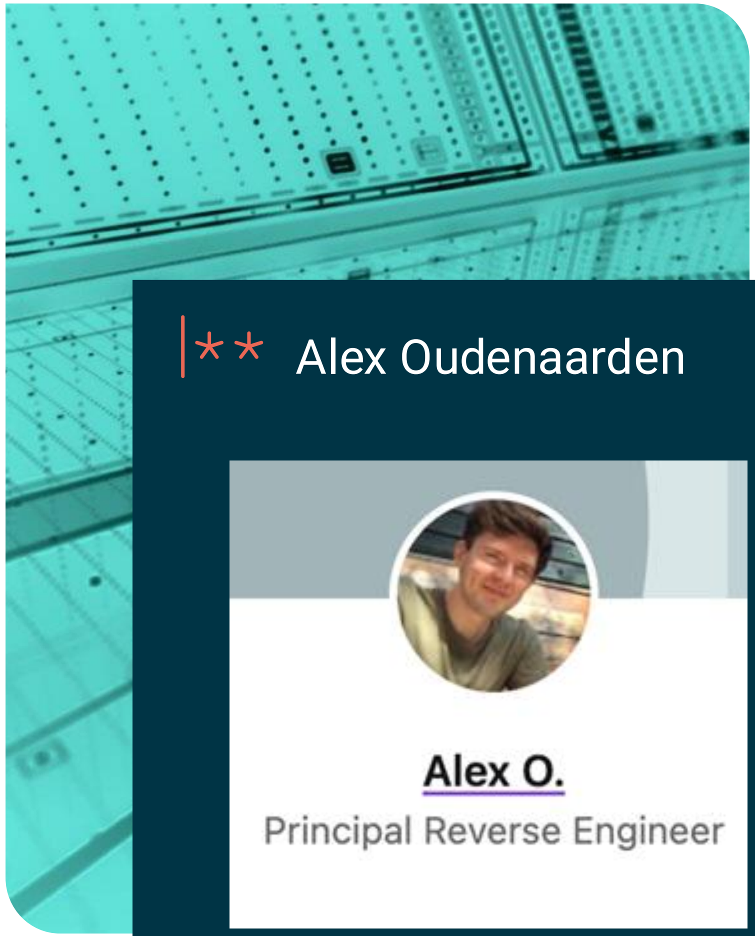
Statistics on BYOVD attack tools by Kaspersky



(data from 2024 was incomplete, and therefore omitted)

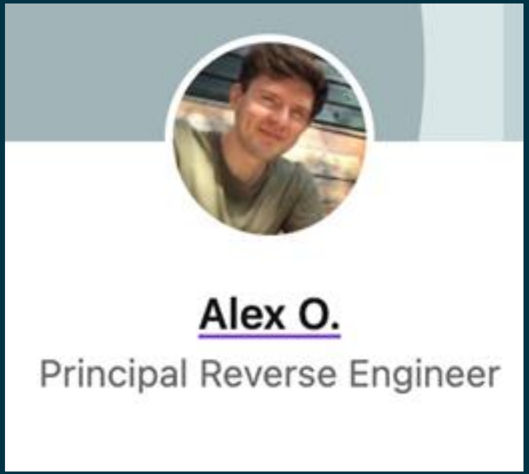
https://securelist.com/vulnerability-exploit-report-q2-2024/113455/

Our Research



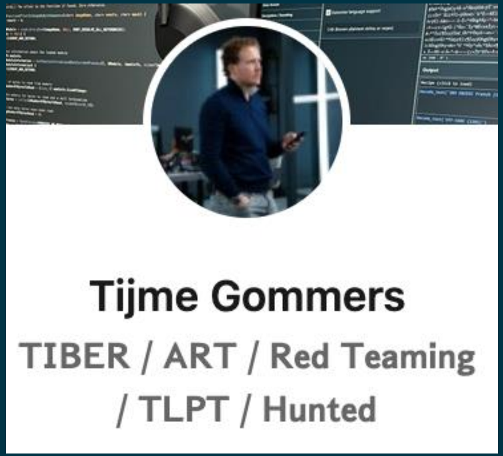
Introducing the team

** Alex Oudenaarden



Alex O.
Principal Reverse Engineer

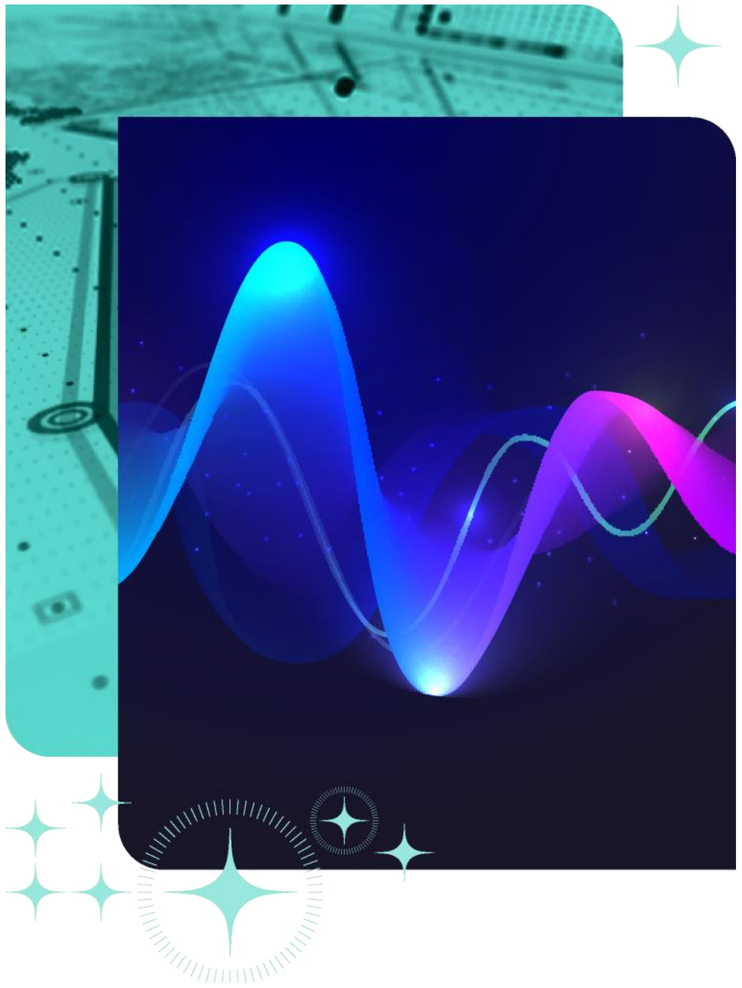
** Tijme Gommers



Tijme Gommers
TIBER / ART / Red Teaming / TLPT / Hunted

** Jan-Jaap Korpershoek

Jan-Jaap Korpershoek
Ethical Hacker at Northwave Cyber Security

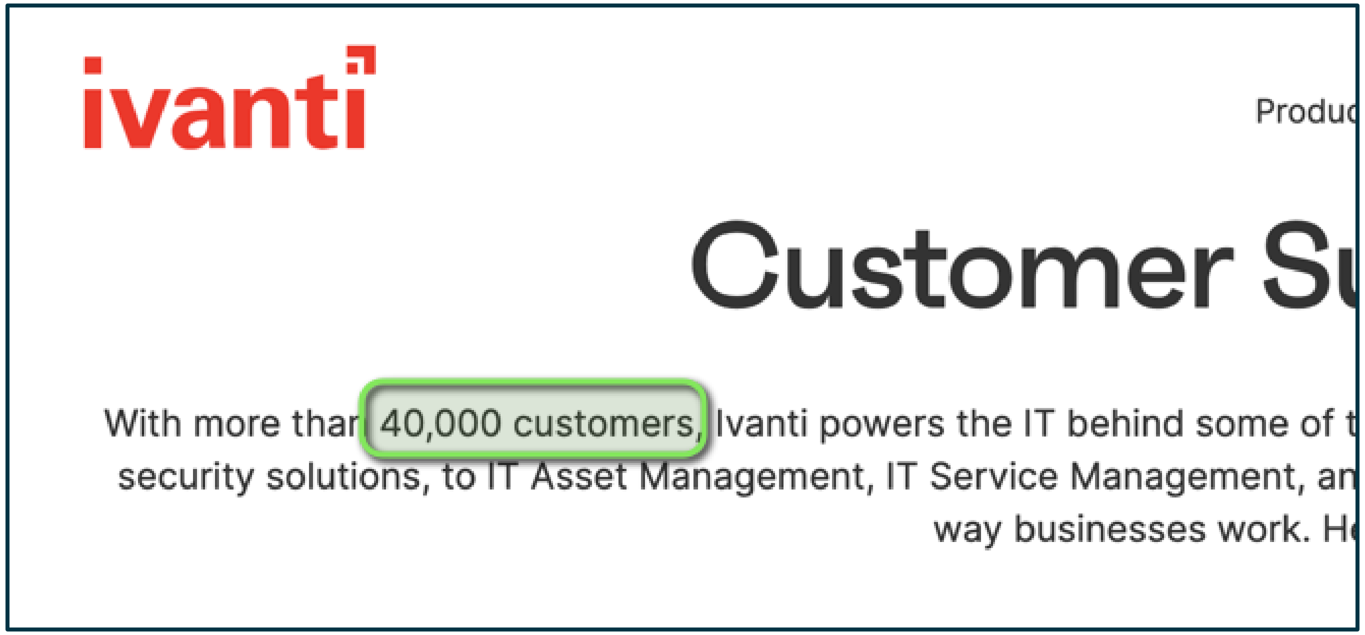


Some statistics

- 4587 drivers
- 79 drivers with good potential manually analysed
- Found 35 vulnerabilities in 24 drivers
- 12 privilege escalation
- 4 render EDR useless

Ivanti/Pulse secure

- Privilege escalation in VPN client
- Proven exploitable



ivanti

Produc

Customer Su

With more than 40,000 customers, Ivanti powers the IT behind some of t
security solutions, to IT Asset Management, IT Service Management, an
way businesses work. He

https://northwave-cybersecurity.com/ivanti-pulse-vpn-privilege-escalation

Macrium

- Privilege escalation in backup software
- Proven exploitable.

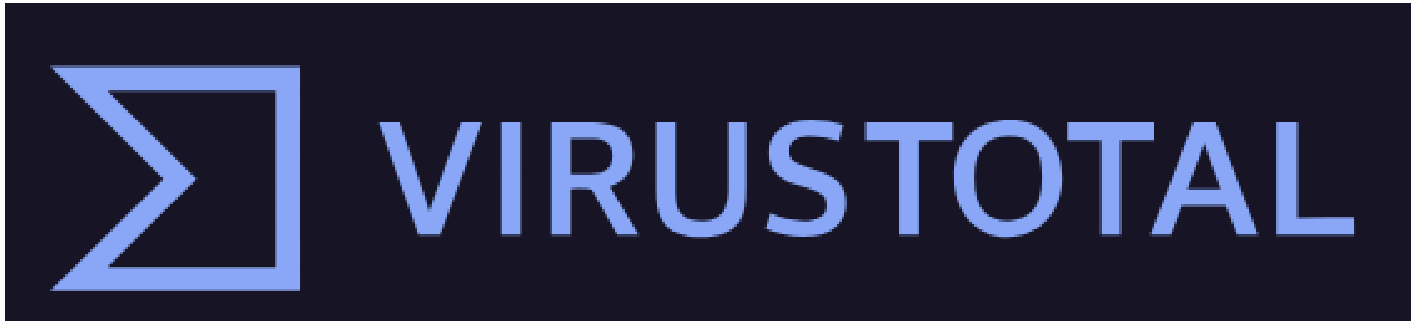
https://northwave-cybersecurity.com/exploiting-enterprise-backup-software-for-privilege-escalation-part-one

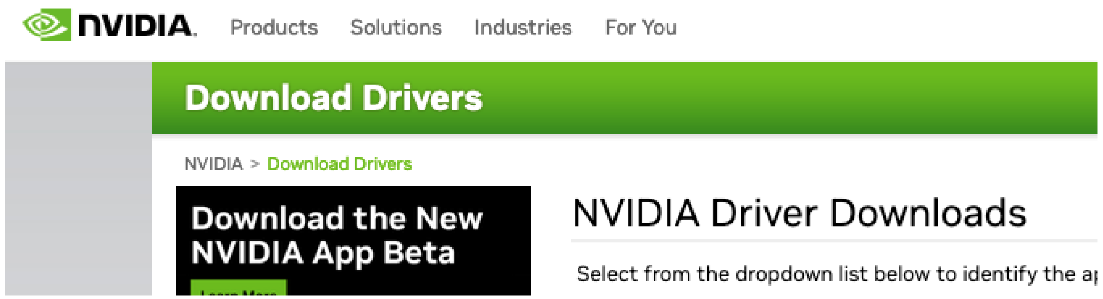
LogMeIn

- Privilege escalation in remote desktop software
- Proven exploitable.

Building a Driver database

Driver sources





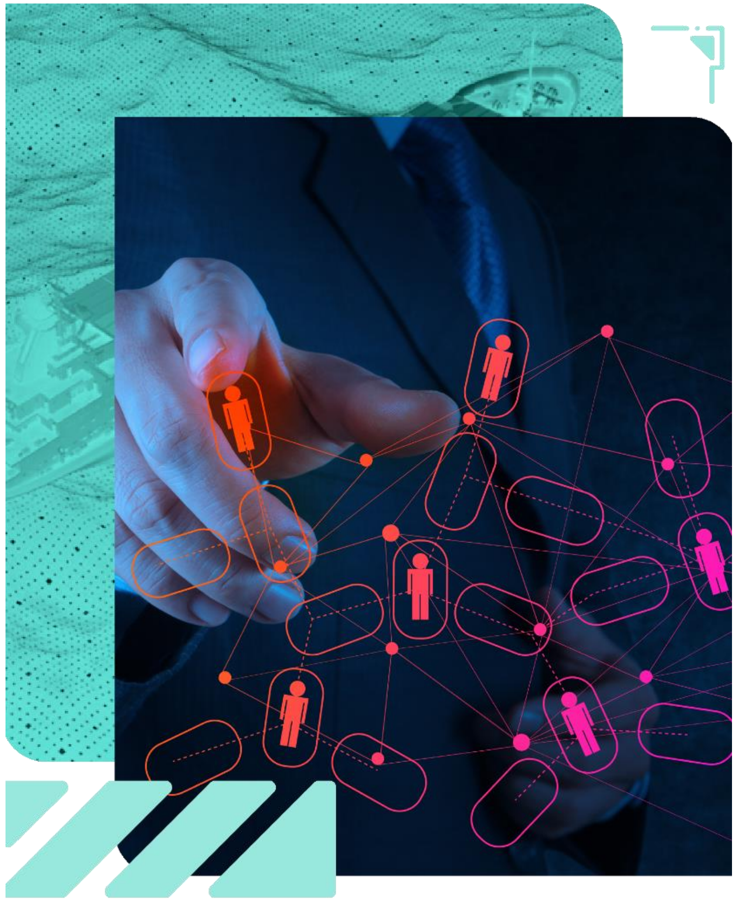
Download Drivers

NVIDIA > Download Drivers

Download the New NVIDIA App Beta

NVIDIA Driver Downloads

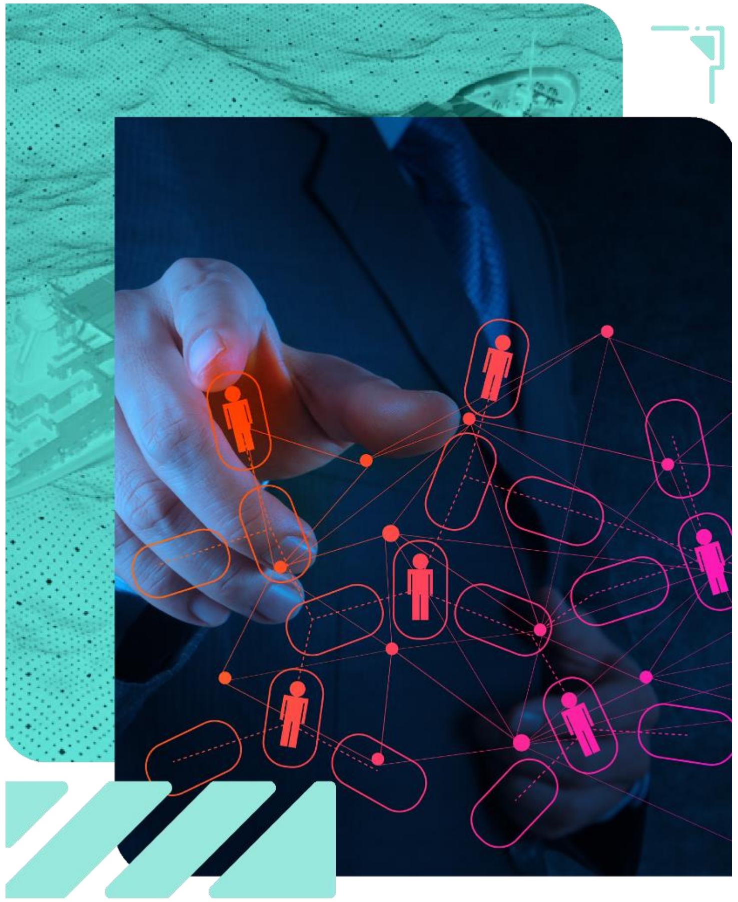
Select from the dropdown list below to identify the a



Postprocessing

- Deduplication
- Keep only most recent version
- Filter by signature
- No special permissions needed
- Automated analysis

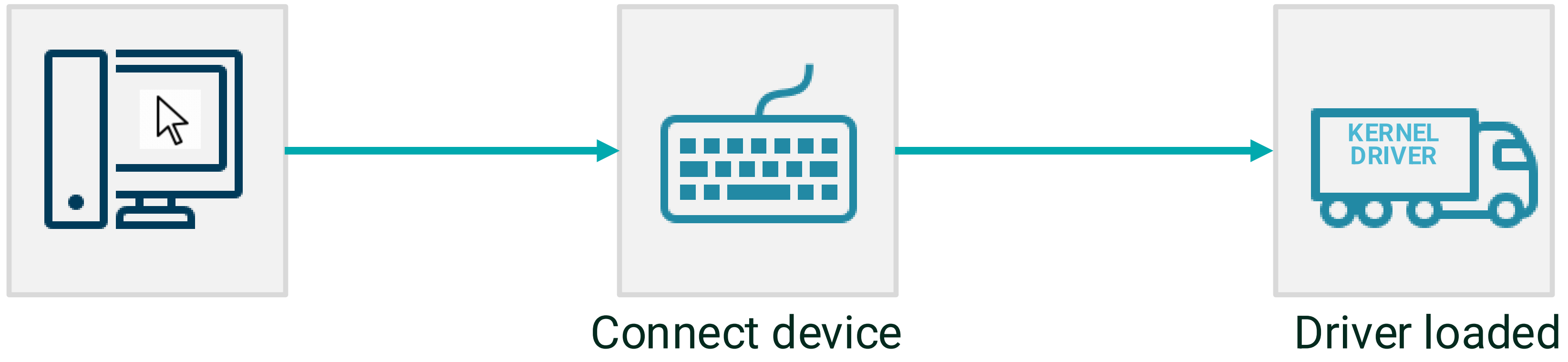
Loading Drivers



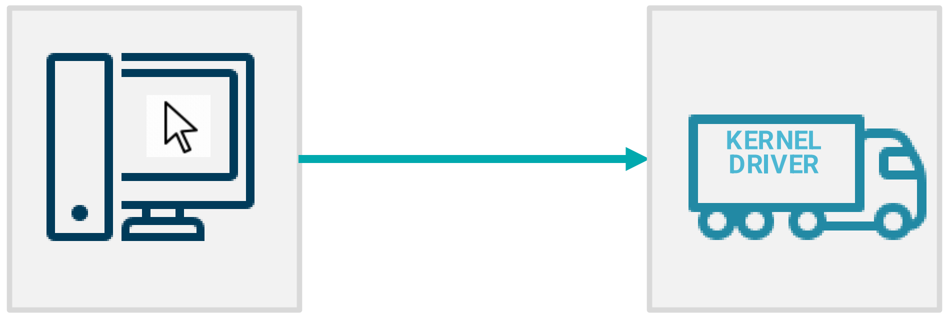
Driver types

- Plug and play (PnP)
- Legacy drivers

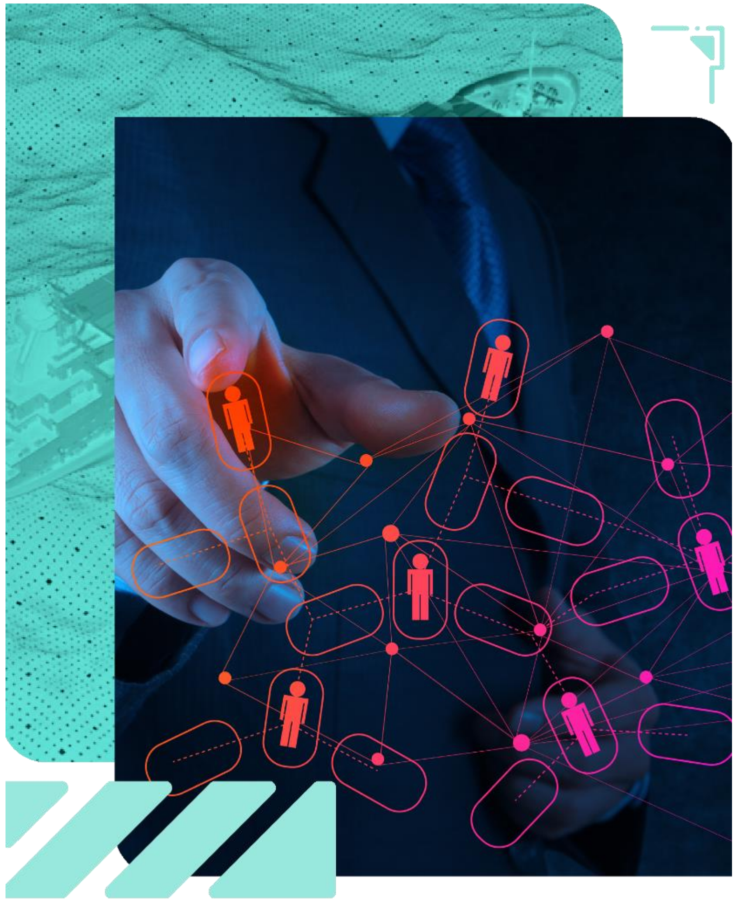
Plug and Play



Legacy drivers

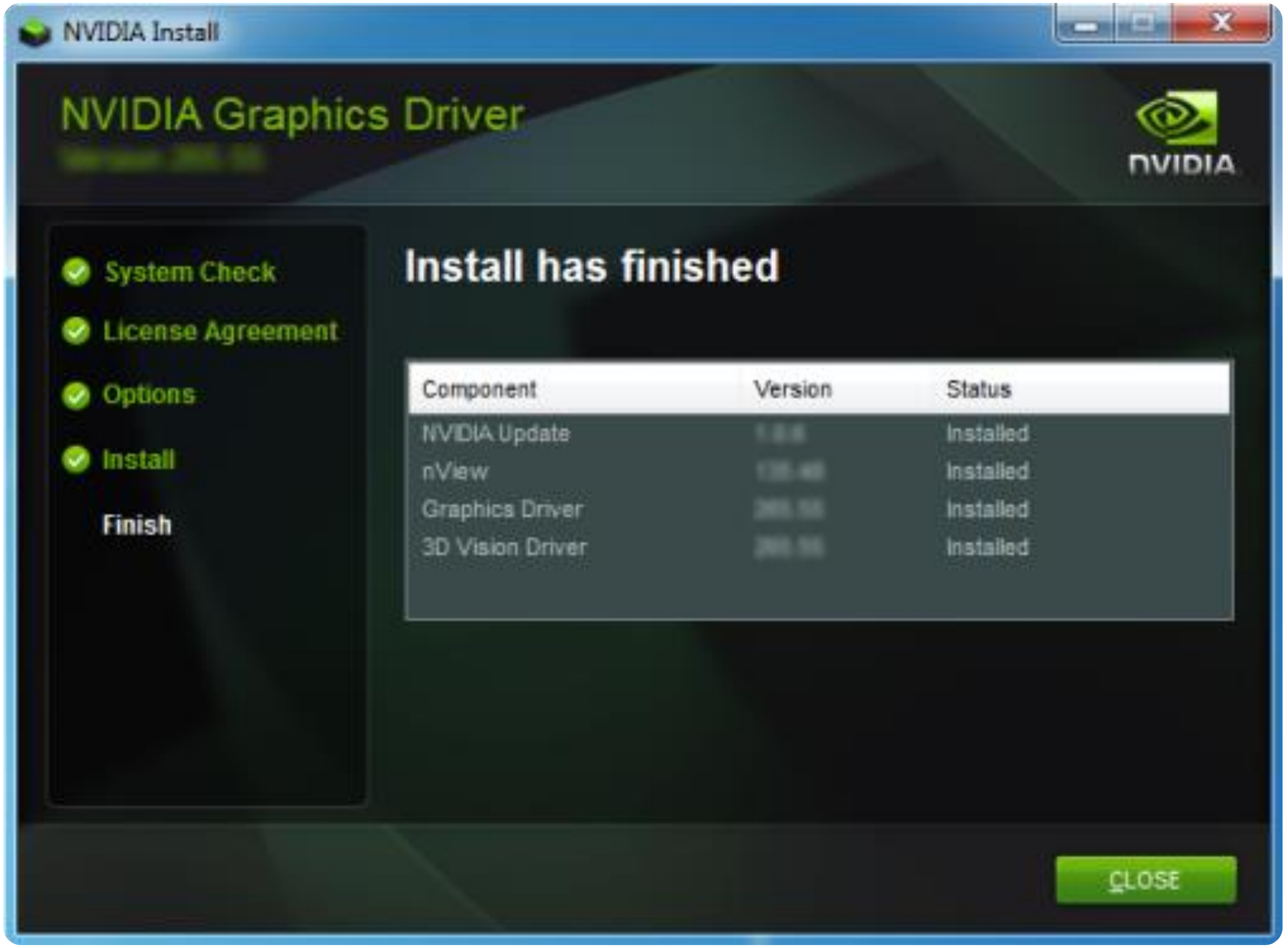


Driver loaded



Device Drivers

- Interact through explicit system call
- Interesting for vulnerability research:
  - Many exploitation examples
  - Easy to trigger vulnerabilities


NVIDIA Install

NVIDIA Graphics Driver

[redacted]

- System Check
- License Agreement
- Options
- Install
- Finish

Install has finished

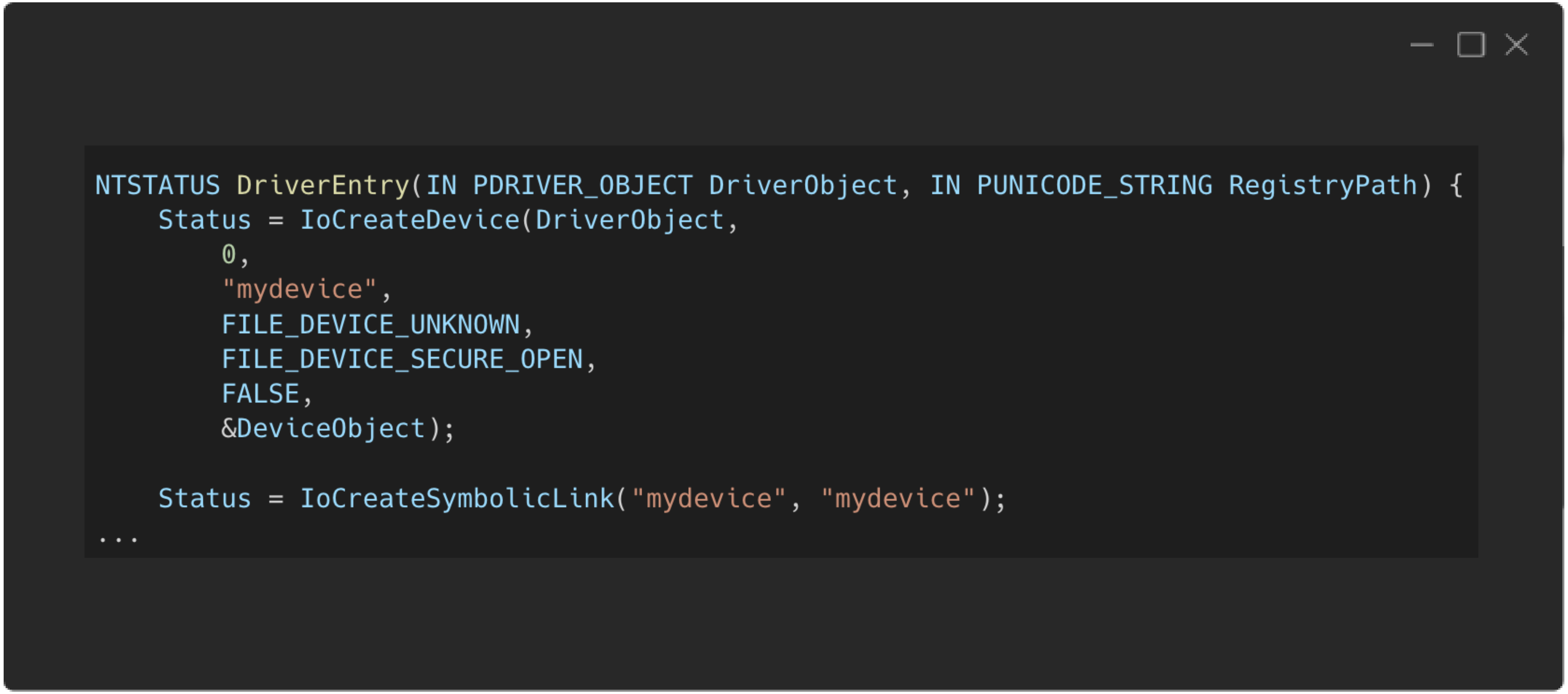
Component	Version	Status
NVIDIA Update	[redacted]	Installed
nView	[redacted]	Installed
Graphics Driver	[redacted]	Installed
3D Vision Driver	[redacted]	Installed

Legacy drivers

```
sc.exe create mydriver binPath= C:\...\mydriver.sys type= kernel
sc.exe start  mydriver
```

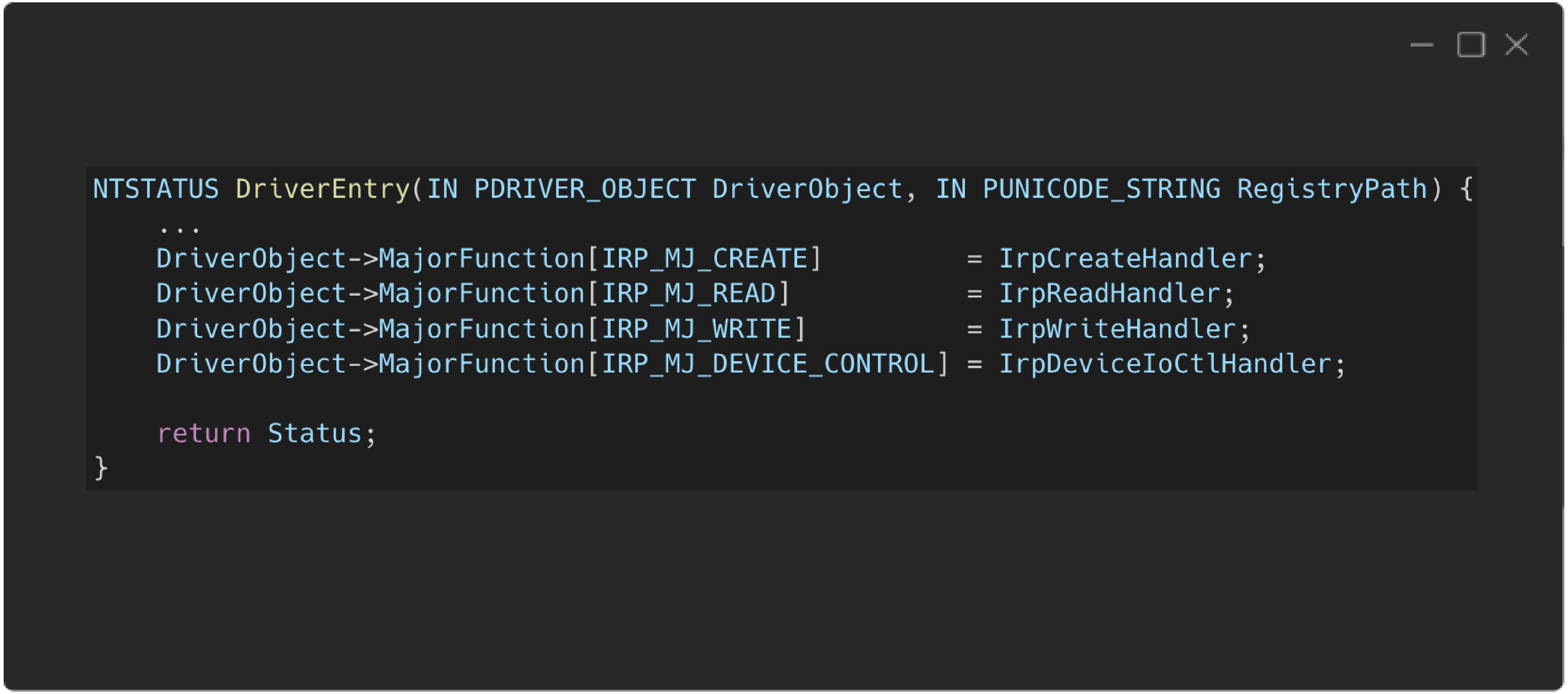
Driver Structure

Structure of WDM IOCTL driver



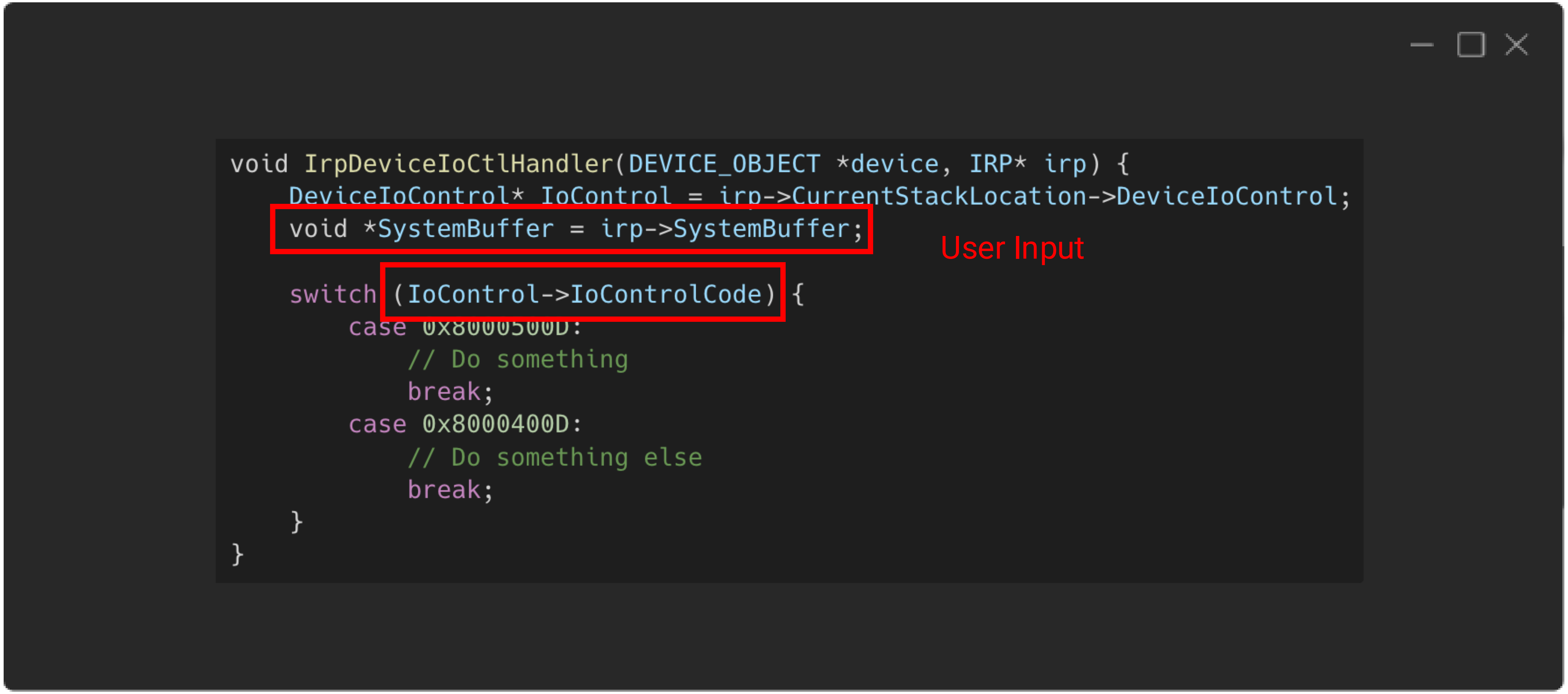
```
NTSTATUS DriverEntry(IN PDRIVER_OBJECT DriverObject, IN PUNICODE_STRING RegistryPath) {
    Status = IoCreateDevice(DriverObject,
        0,
        "mydevice",
        FILE_DEVICE_UNKNOWN,
        FILE_DEVICE_SECURE_OPEN,
        FALSE,
        &DeviceObject);

    Status = IoCreateSymbolicLink("mydevice", "mydevice");
...
```

Major Functions



```
NTSTATUS DriverEntry(IN PDRIVER_OBJECT DriverObject, IN PUNICODE_STRING RegistryPath) {
    ...
    DriverObject->MajorFunction[IRP_MJ_CREATE]         = IrpCreateHandler;
    DriverObject->MajorFunction[IRP_MJ_READ]           = IrpReadHandler;
    DriverObject->MajorFunction[IRP_MJ_WRITE]          = IrpWriteHandler;
    DriverObject->MajorFunction[IRP_MJ_DEVICE_CONTROL] = IrpDeviceIoCtlHandler;

    return Status;
}
```

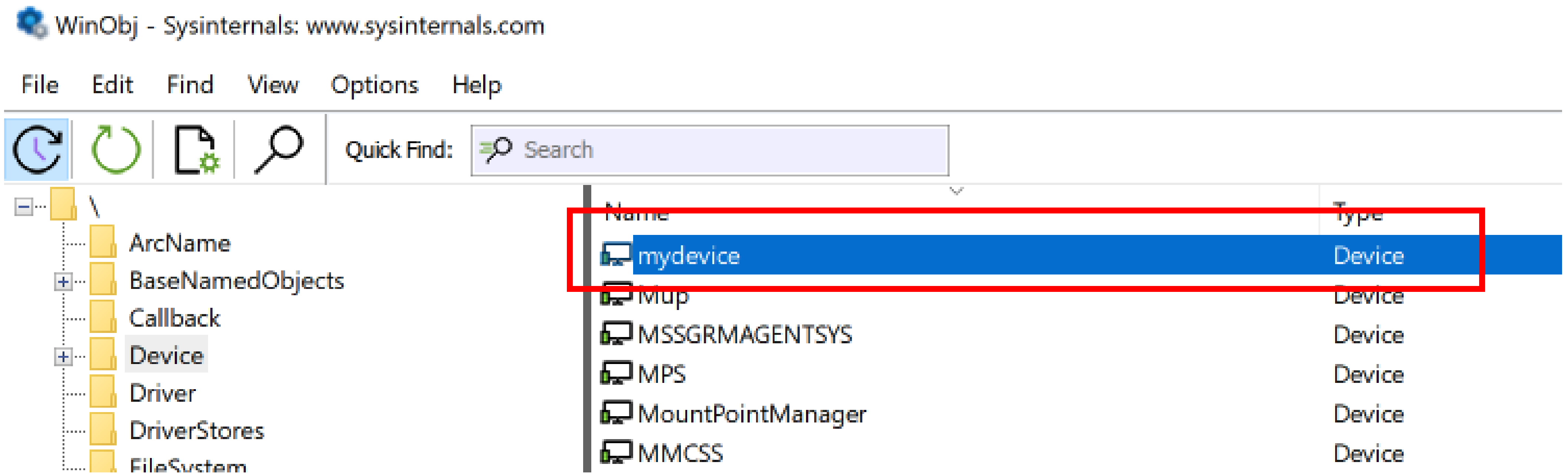
IRP_MJ_DEVICE_CONTROL



```
void IrpDeviceIoCtlHandler(DEVICE_OBJECT *device, IRP* irp) {
    DeviceIoControl* IoControl = irp->CurrentStackLocation->DeviceIoControl;
    void *SystemBuffer = irp->SystemBuffer;

    switch (IoControl->IoControlCode) {
        case 0x8000500D:
            // Do something
            break;
        case 0x8000400D:
            // Do something else
            break;
    }
}
```

Interacting

Device objects



\
ArcName
BaseNamedObjects
Callback
Device
Driver
DriverStores
FileSystem

Name	Type
mydevice	Device
Mup	Device
MSSGRMAGENTSYS	Device
MPS	Device
MountPointManager	Device
MMCSS	Device

DeviceIoControl

```
char * data = ...;
size_t size = 0x1000;

HANDLE handle = CreateFile("\\\\.\\mydevice", ...)

DeviceIoControl(handle, 0x8000500D, data, size, ...)
```

User Input

Access controls

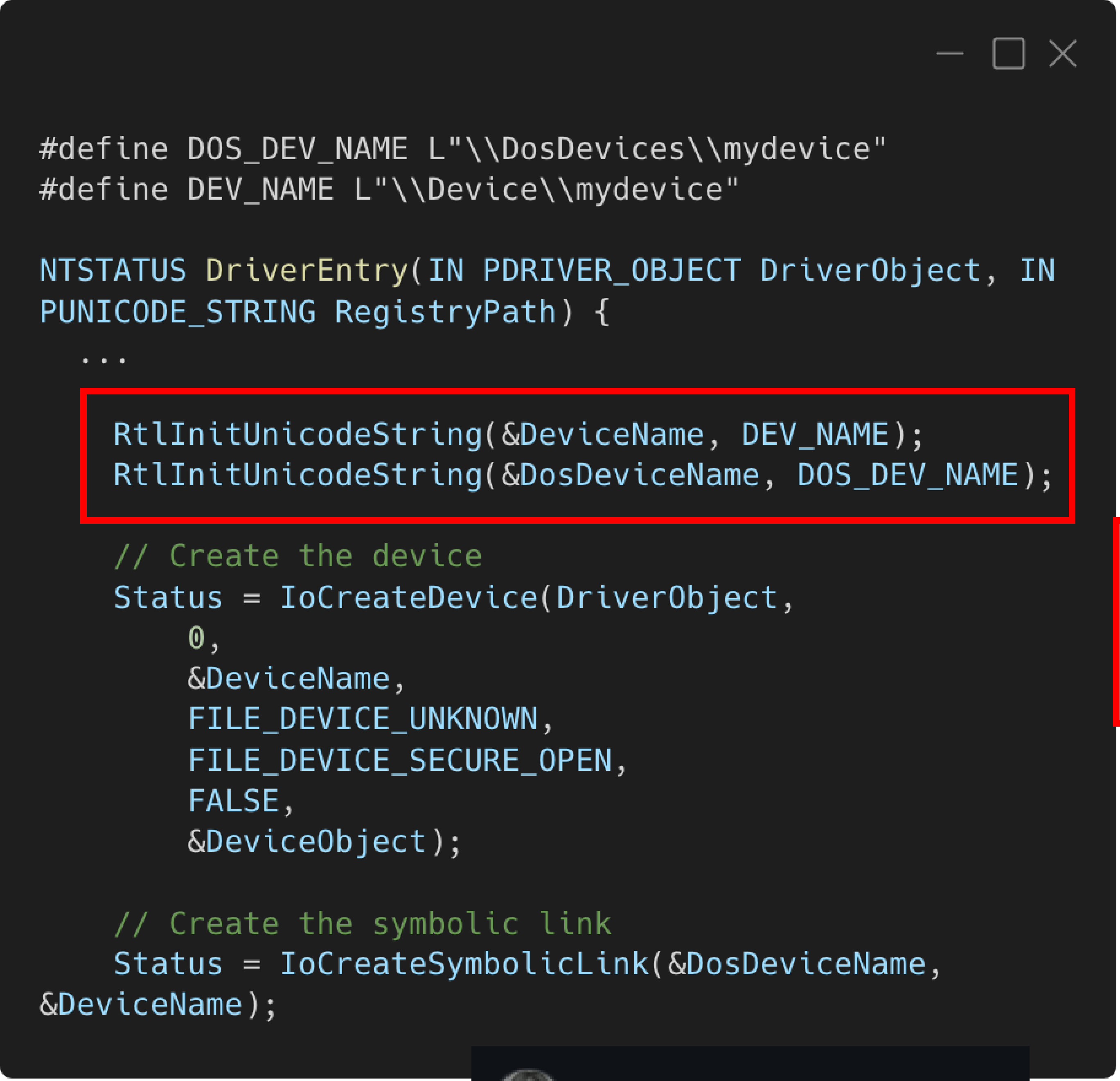
- SDDL -> Access controls on device object
- Stored in INF file or used through IoCreateDeviceSecure

- Find drivers without SDDL or with permissive SDDL

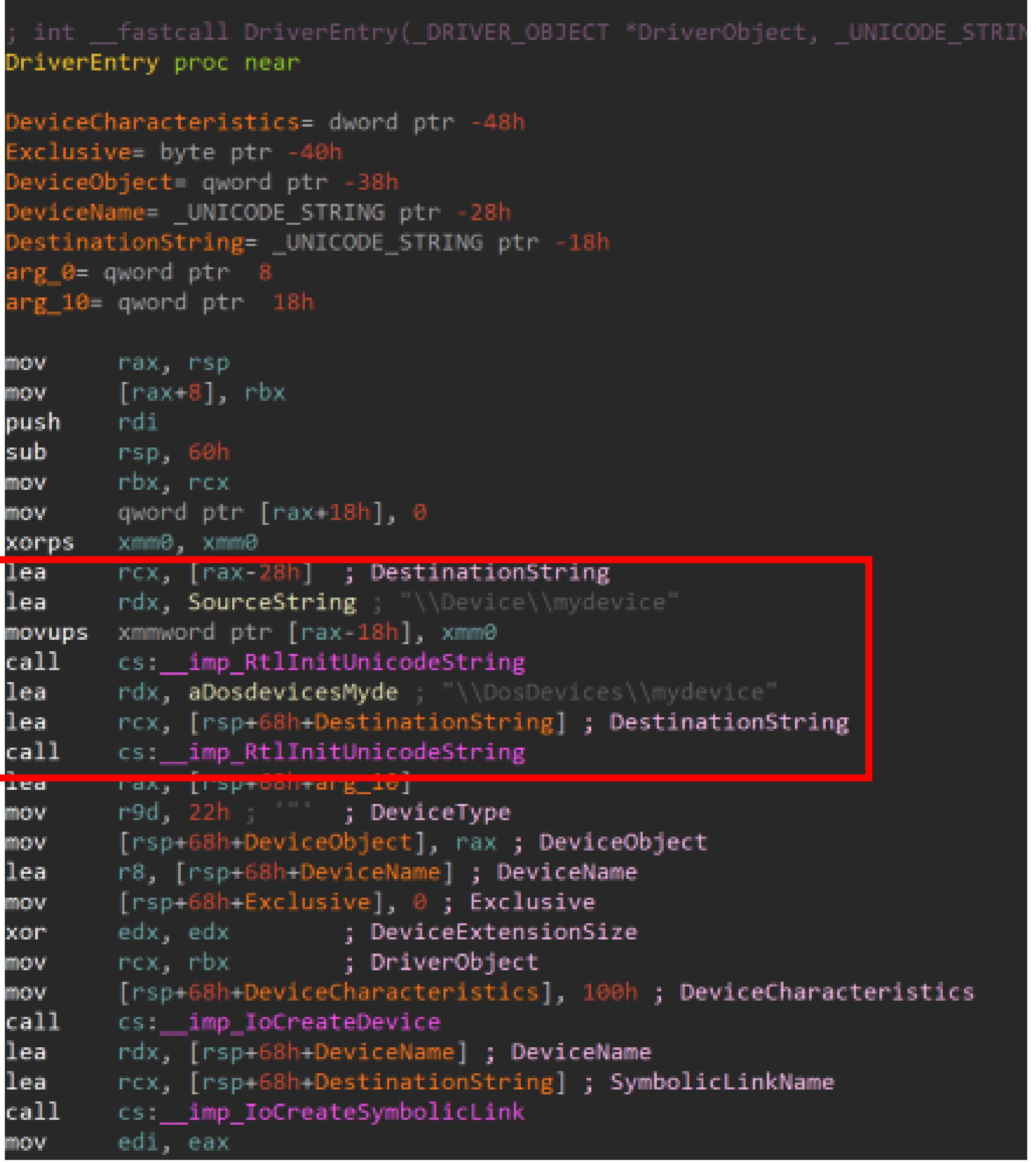
Access controls

- IRP_MJ_CREATE
- Called when user opens device
- Custom access control checks
  - Source process
  - Active user

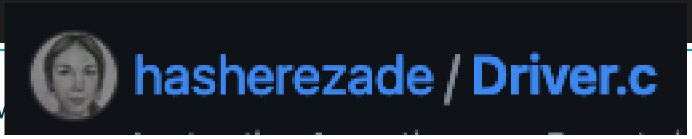
C <-> Assembly

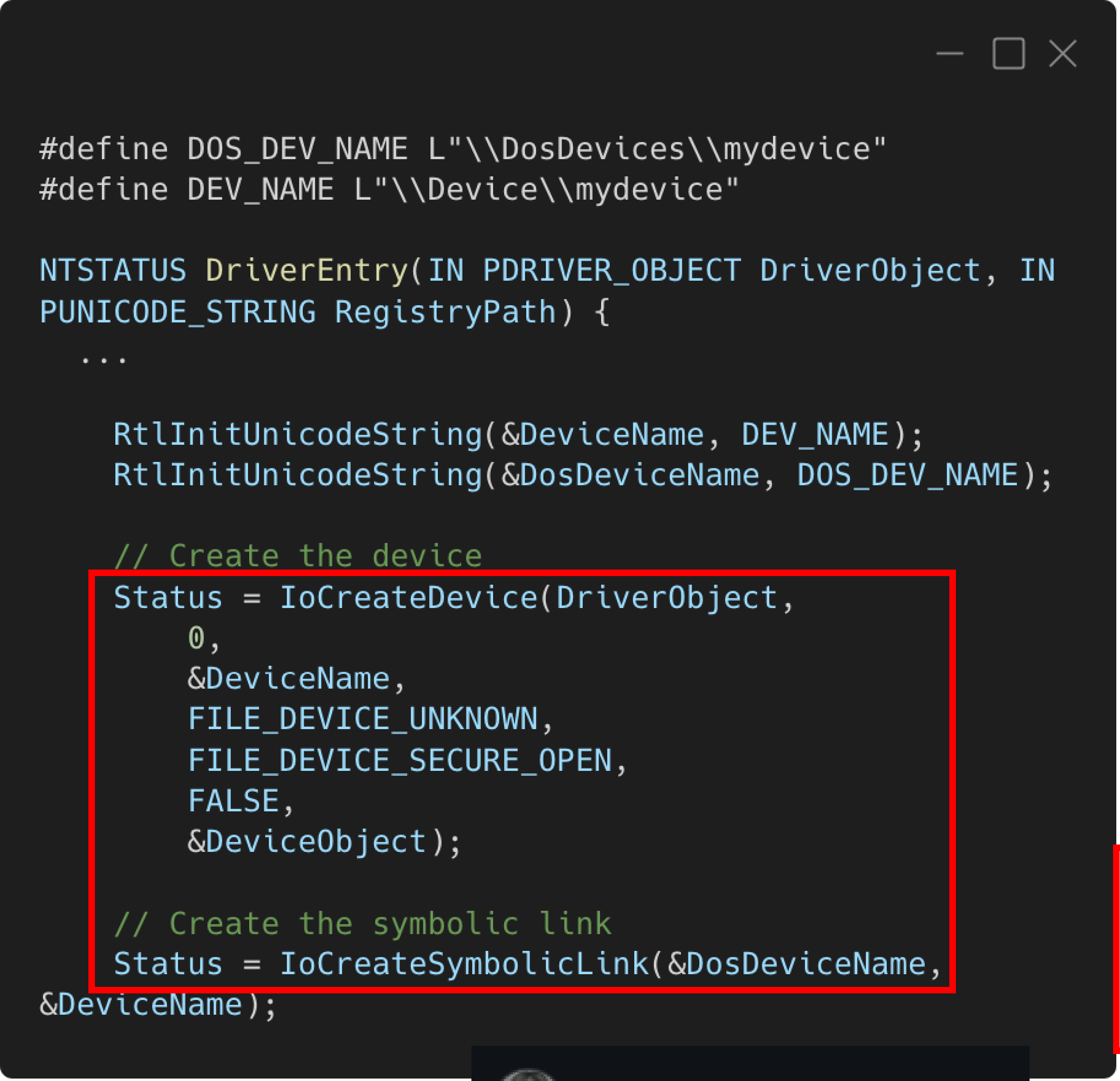


```c
#define DOS_DEV_NAME L"\\DosDevices\\mydevice"
#define DEV_NAME L"\\Device\\mydevice"

NTSTATUS DriverEntry(IN PDRIVER_OBJECT DriverObject, IN
PUNICODE_STRING RegistryPath) {
  ...

    RtlInitUnicodeString(&DeviceName, DEV_NAME);
    RtlInitUnicodeString(&DosDeviceName, DOS_DEV_NAME);

    // Create the device
    Status = IoCreateDevice(DriverObject,
        0,
        &DeviceName,
        FILE_DEVICE_UNKNOWN,
        FILE_DEVICE_SECURE_OPEN,
        FALSE,
        &DeviceObject);

    // Create the symbolic link
    Status = IoCreateSymbolicLink(&DosDeviceName,
&DeviceName);
```

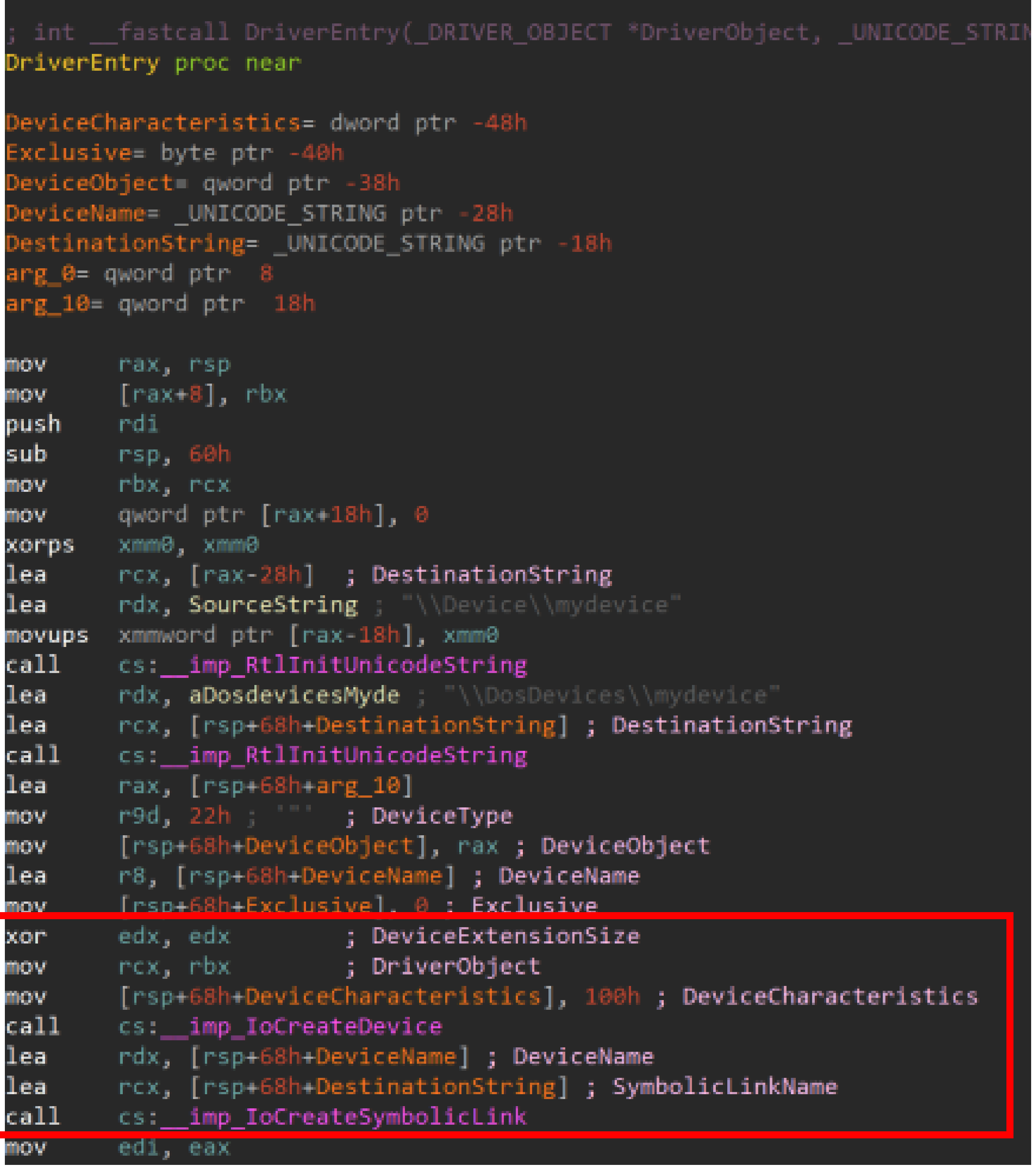


```
; int __fastcall DriverEntry(_DRIVER_OBJECT *DriverObject, _UNICODE_STRIN
DriverEntry proc near

DeviceCharacteristics= dword ptr -48h
Exclusive= byte ptr -40h
DeviceObject= qword ptr -38h
DeviceName= _UNICODE_STRING ptr -28h
DestinationString= _UNICODE_STRING ptr -18h
arg_0= qword ptr  8
arg_10= qword ptr  18h

mov     rax, rsp
mov     [rax+8], rbx
push    rdi
sub     rsp, 60h
mov     rbx, rcx
mov     qword ptr [rax+18h], 0
xorps   xmm0, xmm0
lea     rcx, [rax-28h]  ; DestinationString
lea     rdx, SourceString ; "\\Device\\mydevice"
movups  xmmword ptr [rax-18h], xmm0
call    cs:__imp_RtlInitUnicodeString
lea     rdx, aDosdevicesMyde ; "\\DosDevices\\mydevice"
lea     rcx, [rsp+68h+DestinationString] ; DestinationString
call    cs:__imp_RtlInitUnicodeString
lea     rax, [rsp+68h+arg_10]
mov     r9d, 22h ; '"'  ; DeviceType
mov     [rsp+68h+DeviceObject], rax ; DeviceObject
lea     r8, [rsp+68h+DeviceName] ; DeviceName
mov     [rsp+68h+Exclusive], 0 ; Exclusive
xor     edx, edx        ; DeviceExtensionSize
mov     rcx, rbx        ; DriverObject
mov     [rsp+68h+DeviceCharacteristics], 100h ; DeviceCharacteristics
call    cs:__imp_IoCreateDevice
lea     rdx, [rsp+68h+DeviceName] ; DeviceName
lea     rcx, [rsp+68h+DestinationString] ; SymbolicLinkName
call    cs:__imp_IoCreateSymbolicLink
mov     edi, eax
```

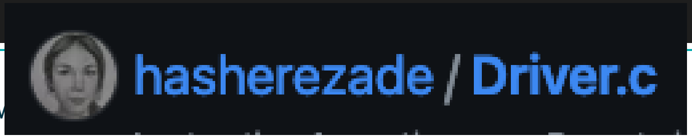




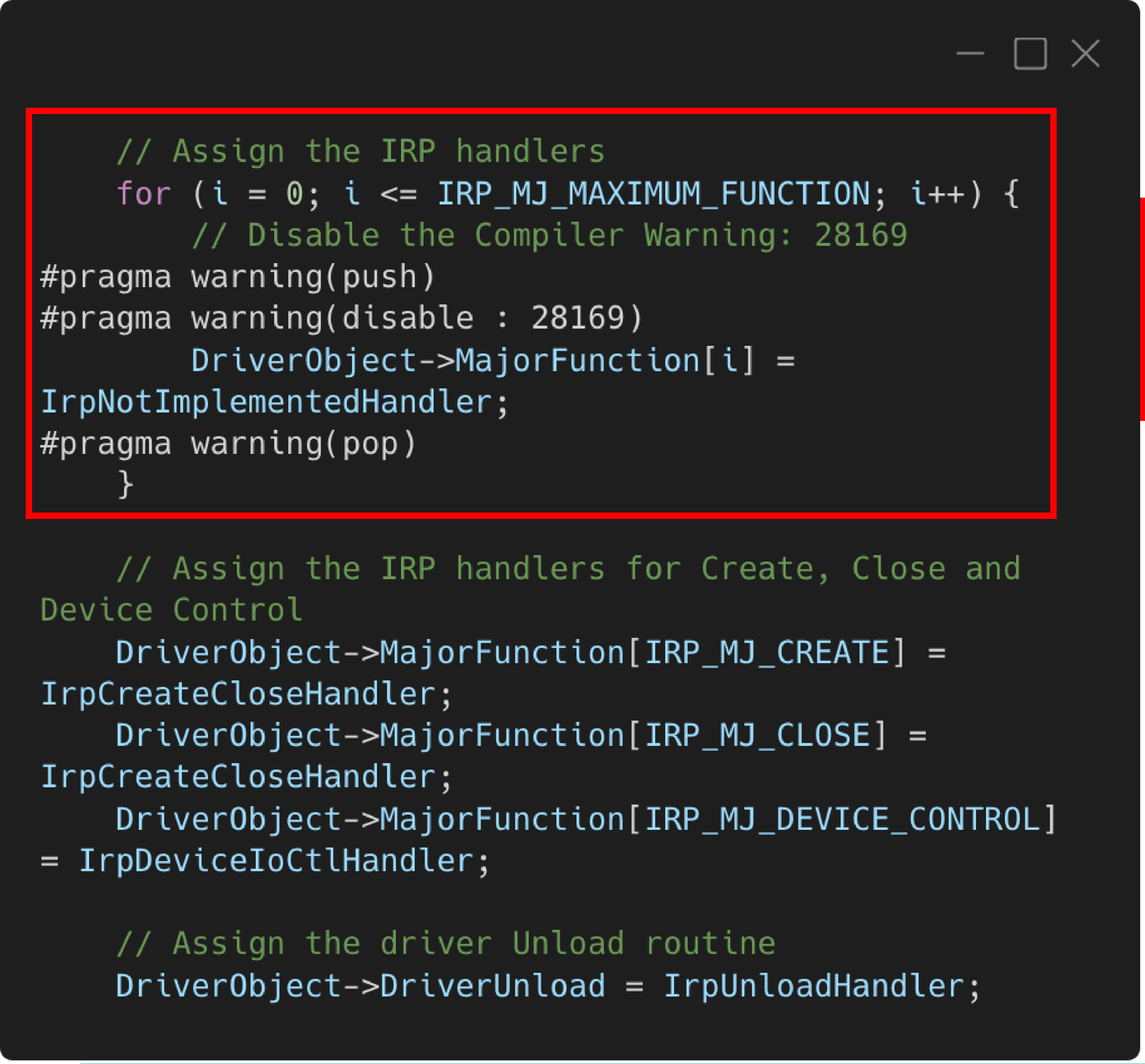
```
#define DOS_DEV_NAME L"\\DosDevices\\mydevice"
#define DEV_NAME L"\\Device\\mydevice"

NTSTATUS DriverEntry(IN PDRIVER_OBJECT DriverObject, IN
PUNICODE_STRING RegistryPath) {
  ...

    RtlInitUnicodeString(&DeviceName, DEV_NAME);
    RtlInitUnicodeString(&DosDeviceName, DOS_DEV_NAME);

    // Create the device
    Status = IoCreateDevice(DriverObject,
        0,
        &DeviceName,
        FILE_DEVICE_UNKNOWN,
        FILE_DEVICE_SECURE_OPEN,
        FALSE,
        &DeviceObject);

    // Create the symbolic link
    Status = IoCreateSymbolicLink(&DosDeviceName,
&DeviceName);
```



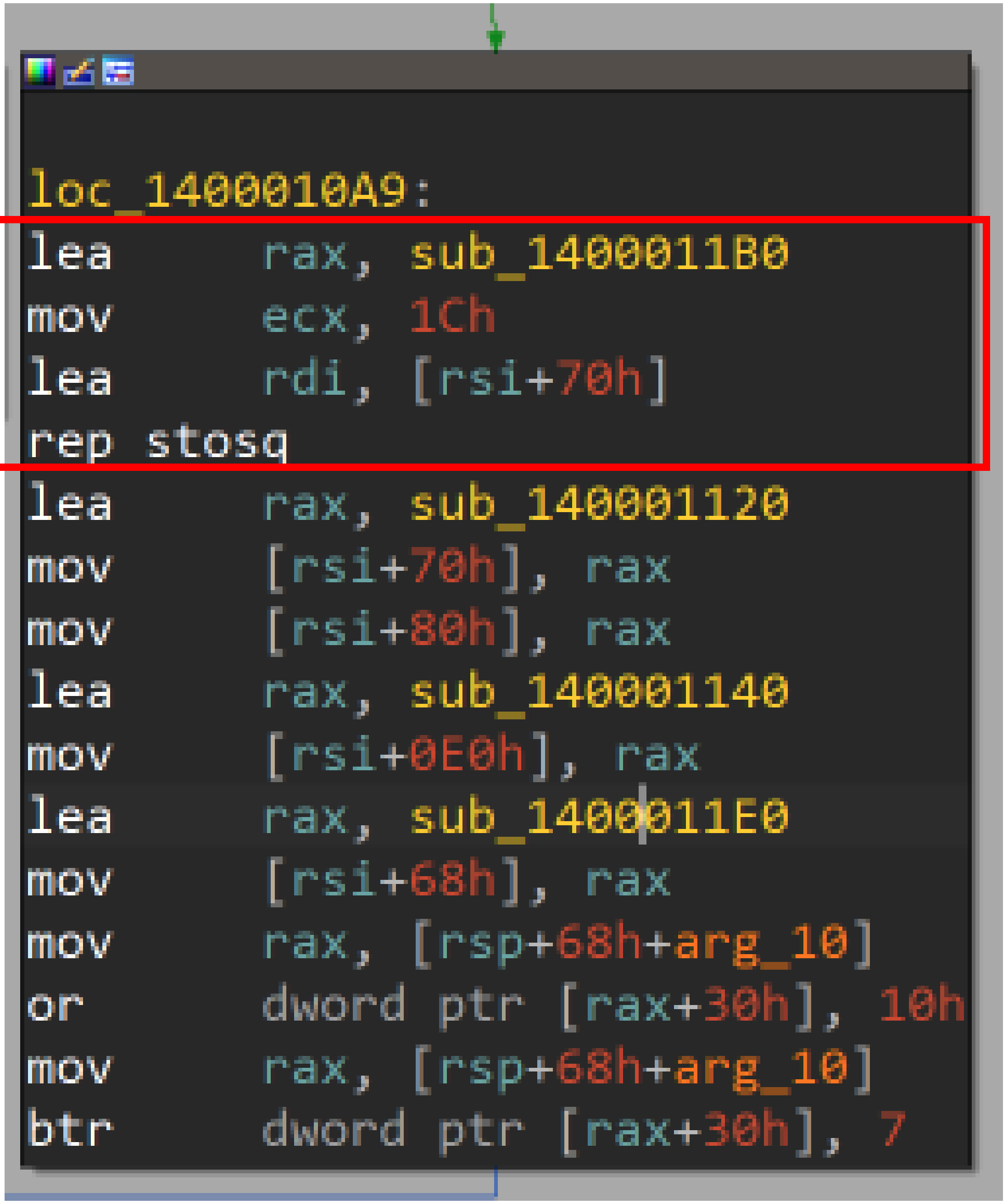
```
; int __fastcall DriverEntry(_DRIVER_OBJECT *DriverObject, _UNICODE_STRIN
DriverEntry proc near

DeviceCharacteristics= dword ptr -48h
Exclusive= byte ptr -40h
DeviceObject= qword ptr -38h
DeviceName= _UNICODE_STRING ptr -28h
DestinationString= _UNICODE_STRING ptr -18h
arg_0= qword ptr  8
arg_10= qword ptr  18h

mov     rax, rsp
mov     [rax+8], rbx
push    rdi
sub     rsp, 60h
mov     rbx, rcx
mov     qword ptr [rax+18h], 0
xorps   xmm0, xmm0
lea     rcx, [rax-28h]  ; DestinationString
lea     rdx, SourceString ; "\\Device\\mydevice"
movups  xmmword ptr [rax-18h], xmm0
call    cs:__imp_RtlInitUnicodeString
lea     rdx, aDosdevicesMyde ; "\\DosDevices\\mydevice"
lea     rcx, [rsp+68h+DestinationString] ; DestinationString
call    cs:__imp_RtlInitUnicodeString
lea     rax, [rsp+68h+arg_10]
mov     r9d, 22h ; '"'  ; DeviceType
mov     [rsp+68h+DeviceObject], rax ; DeviceObject
lea     r8, [rsp+68h+DeviceName] ; DeviceName
mov     [rsp+68h+Exclusive], 0 ; Exclusive
xor     edx, edx        ; DeviceExtensionSize
mov     rcx, rbx        ; DriverObject
mov     [rsp+68h+DeviceCharacteristics], 100h ; DeviceCharacteristics
call    cs:__imp_IoCreateDevice
lea     rdx, [rsp+68h+DeviceName] ; DeviceName
lea     rcx, [rsp+68h+DestinationString] ; SymbolicLinkName
call    cs:__imp_IoCreateSymbolicLink
mov     edi, eax
```



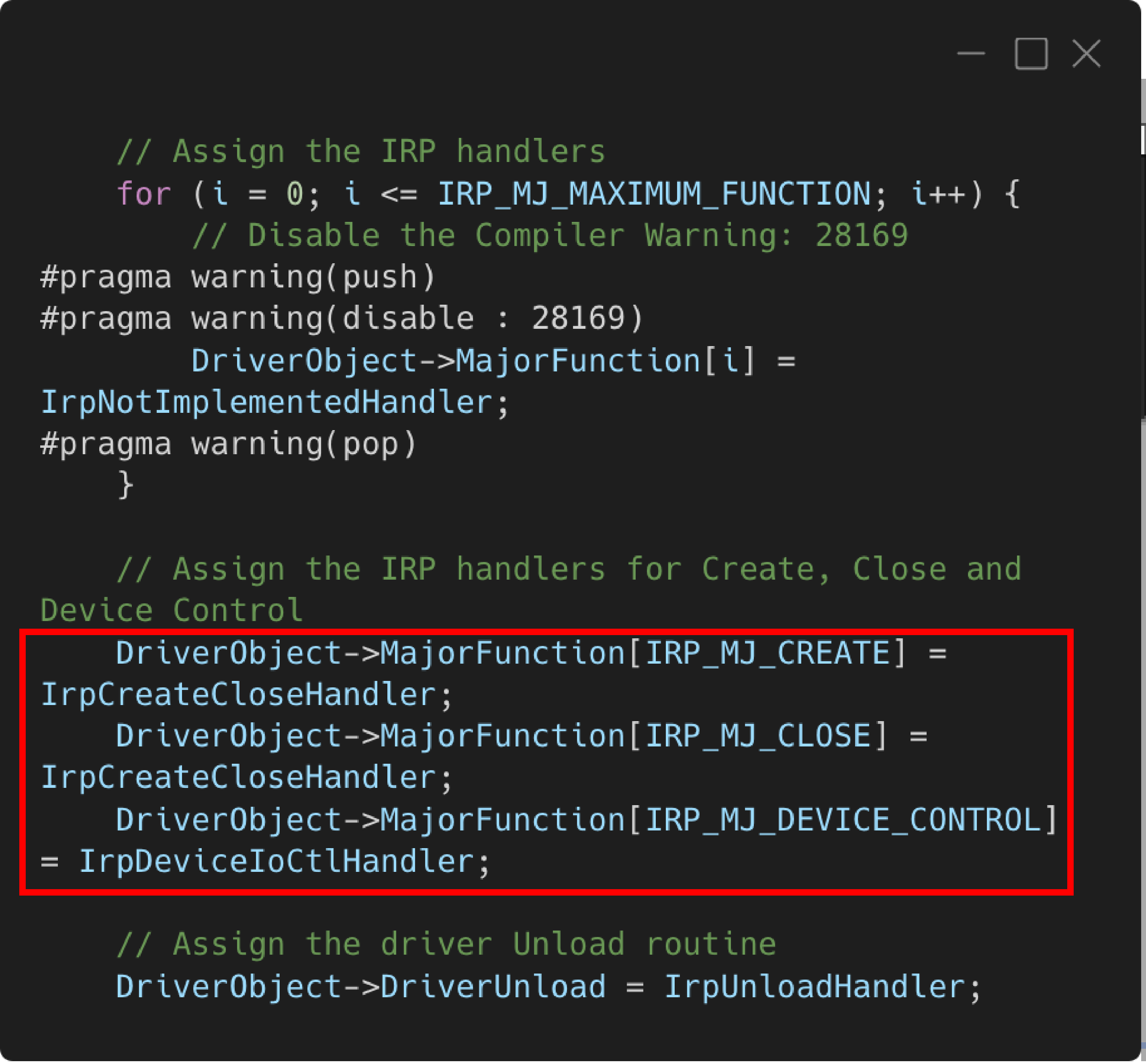
hasherezade / Driver.c



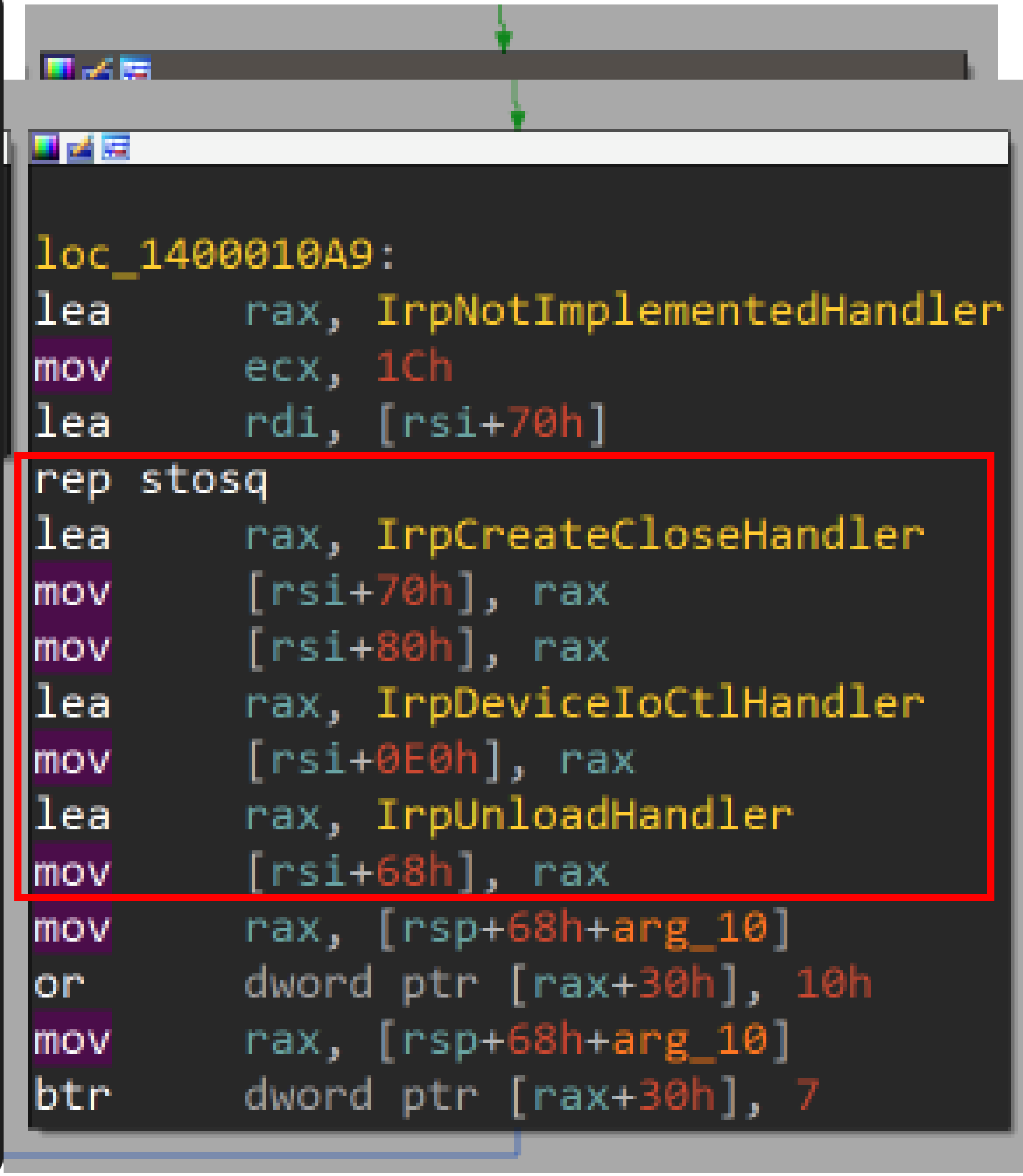
```
    // Assign the IRP handlers
    for (i = 0; i <= IRP_MJ_MAXIMUM_FUNCTION; i++) {
        // Disable the Compiler Warning: 28169
#pragma warning(push)
#pragma warning(disable : 28169)
        DriverObject->MajorFunction[i] =
IrpNotImplementedHandler;
#pragma warning(pop)
    }

    // Assign the IRP handlers for Create, Close and
Device Control
    DriverObject->MajorFunction[IRP_MJ_CREATE] =
IrpCreateCloseHandler;
    DriverObject->MajorFunction[IRP_MJ_CLOSE] =
IrpCreateCloseHandler;
    DriverObject->MajorFunction[IRP_MJ_DEVICE_CONTROL]
= IrpDeviceIoCtlHandler;

    // Assign the driver Unload routine
    DriverObject->DriverUnload = IrpUnloadHandler;
```



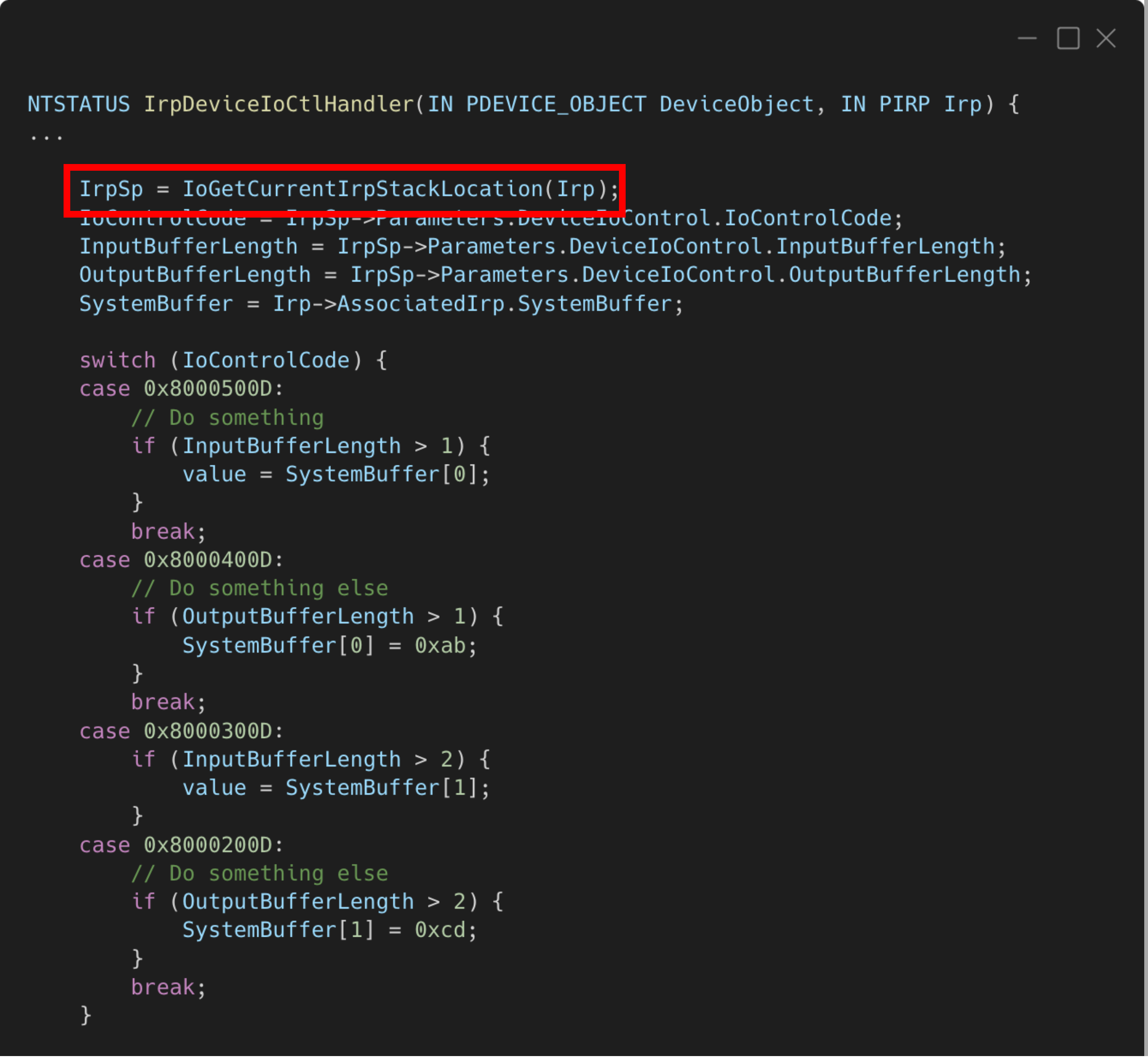
```
loc_1400010A9:
lea     rax, sub_1400011B0
mov     ecx, 1Ch
lea     rdi, [rsi+70h]
rep stosq
lea     rax, sub_14001120
mov     [rsi+70h], rax
mov     [rsi+80h], rax
lea     rax, sub_14001140
mov     [rsi+0E0h], rax
lea     rax, sub_1400011E0
mov     [rsi+68h], rax
mov     rax, [rsp+68h+arg_10]
or      dword ptr [rax+30h], 10h
mov     rax, [rsp+68h+arg_10]
btr     dword ptr [rax+30h], 7
```



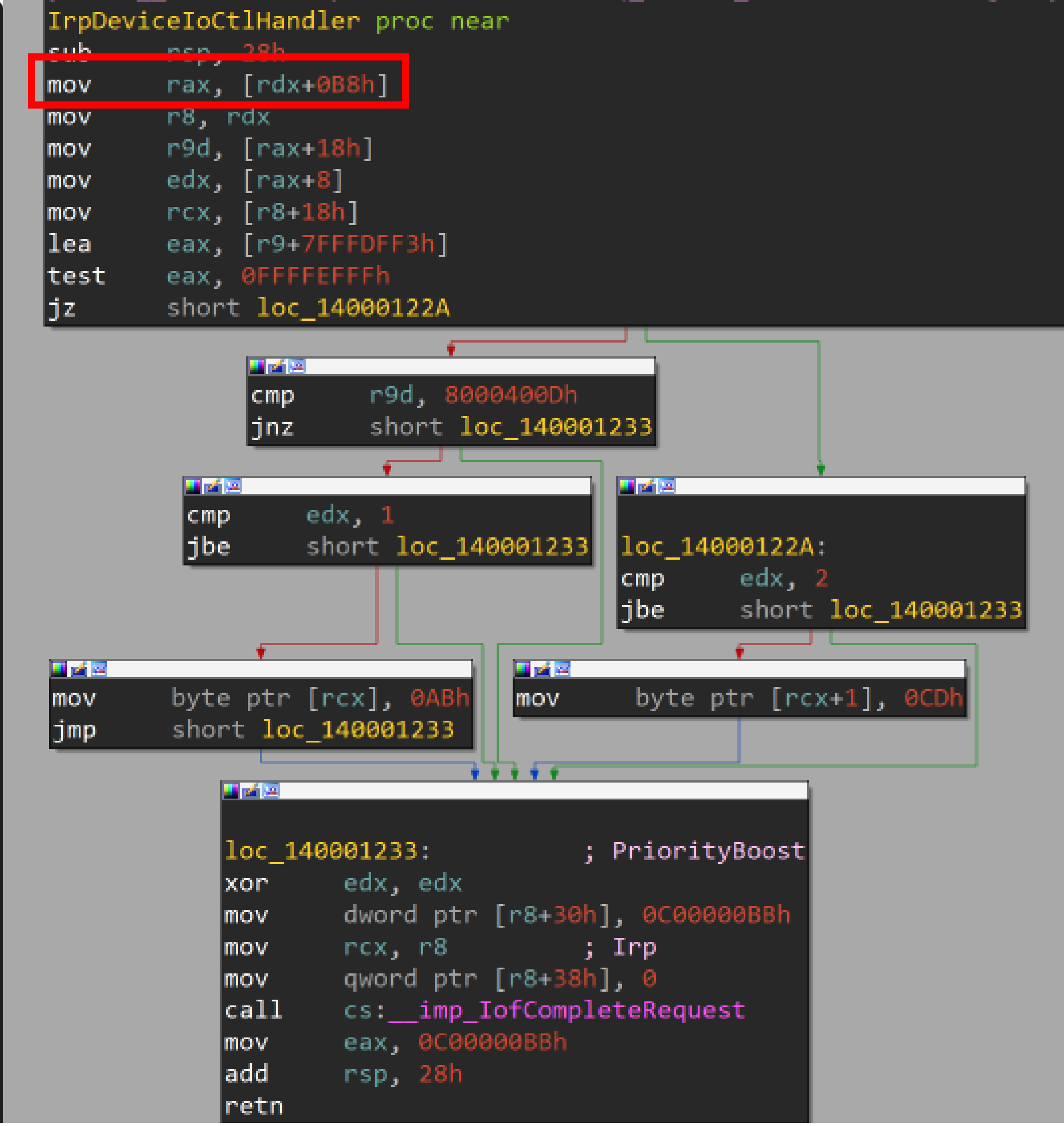
```
    // Assign the IRP handlers
    for (i = 0; i <= IRP_MJ_MAXIMUM_FUNCTION; i++) {
        // Disable the Compiler Warning: 28169
#pragma warning(push)
#pragma warning(disable : 28169)
        DriverObject->MajorFunction[i] =
IrpNotImplementedHandler;
#pragma warning(pop)
    }

    // Assign the IRP handlers for Create, Close and
Device Control
    DriverObject->MajorFunction[IRP_MJ_CREATE] =
IrpCreateCloseHandler;
    DriverObject->MajorFunction[IRP_MJ_CLOSE] =
IrpCreateCloseHandler;
    DriverObject->MajorFunction[IRP_MJ_DEVICE_CONTROL]
= IrpDeviceIoCtlHandler;

    // Assign the driver Unload routine
    DriverObject->DriverUnload = IrpUnloadHandler;
```



```
loc_1400010A9:
lea     rax, IrpNotImplementedHandler
mov     ecx, 1Ch
lea     rdi, [rsi+70h]
rep stosq
lea     rax, IrpCreateCloseHandler
mov     [rsi+70h], rax
mov     [rsi+80h], rax
lea     rax, IrpDeviceIoCtlHandler
mov     [rsi+0E0h], rax
lea     rax, IrpUnloadHandler
mov     [rsi+68h], rax
mov     rax, [rsp+68h+arg_10]
or      dword ptr [rax+30h], 10h
mov     rax, [rsp+68h+arg_10]
btr     dword ptr [rax+30h], 7
```



```
NTSTATUS IrpDeviceIoCtlHandler(IN PDEVICE_OBJECT DeviceObject, IN PIRP Irp) {
...

    IrpSp = IoGetCurrentIrpStackLocation(Irp);
    IoControlCode = IrpSp->Parameters.DeviceIoControl.IoControlCode;
    InputBufferLength = IrpSp->Parameters.DeviceIoControl.InputBufferLength;
    OutputBufferLength = IrpSp->Parameters.DeviceIoControl.OutputBufferLength;
    SystemBuffer = Irp->AssociatedIrp.SystemBuffer;

    switch (IoControlCode) {
    case 0x8000500D:
        // Do something
        if (InputBufferLength > 1) {
            value = SystemBuffer[0];
        }
        break;
    case 0x8000400D:
        // Do something else
        if (OutputBufferLength > 1) {
            SystemBuffer[0] = 0xab;
        }
        break;
    case 0x8000300D:
        if (InputBufferLength > 2) {
            value = SystemBuffer[1];
        }
    case 0x8000200D:
        // Do something else
        if (OutputBufferLength > 2) {
            SystemBuffer[1] = 0xcd;
        }
        break;
    }
```



```
IrpDeviceIoCtlHandler proc near
sub     rsp, 28h
mov     rax, [rdx+0B8h]
mov     r8, rdx
mov     r9d, [rax+18h]
mov     edx, [rax+8]
mov     rcx, [r8+18h]
lea     eax, [r9+7FFFDFF3h]
test    eax, 0FFFFEFFFh
jz      short loc_14000122A
```

```
cmp     r9d, 8000400Dh
jnz     short loc_140001233
```

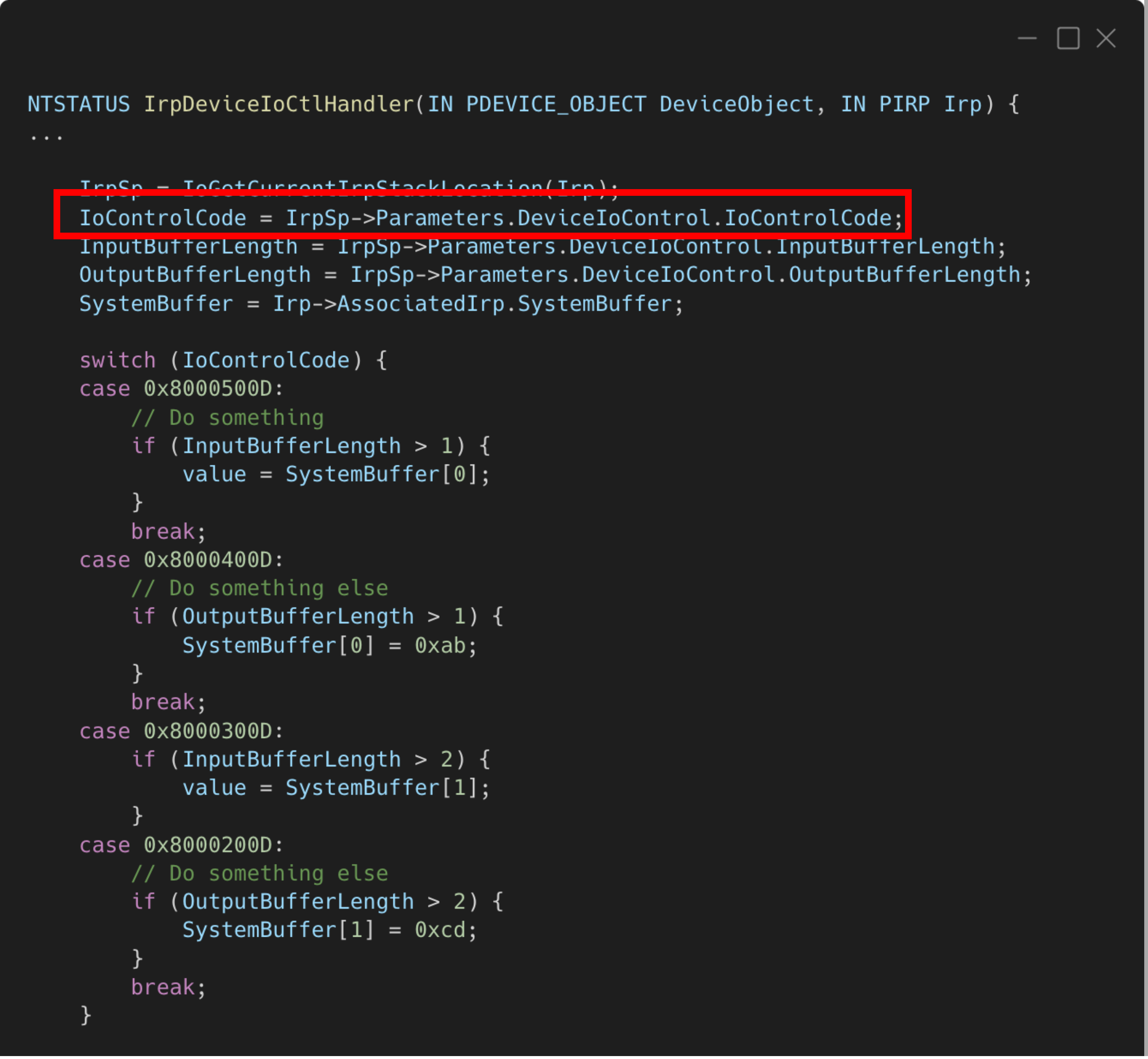
```
cmp     edx, 1
jbe     short loc_140001233
```

```
loc_14000122A:
cmp     edx, 2
jbe     short loc_140001233
```

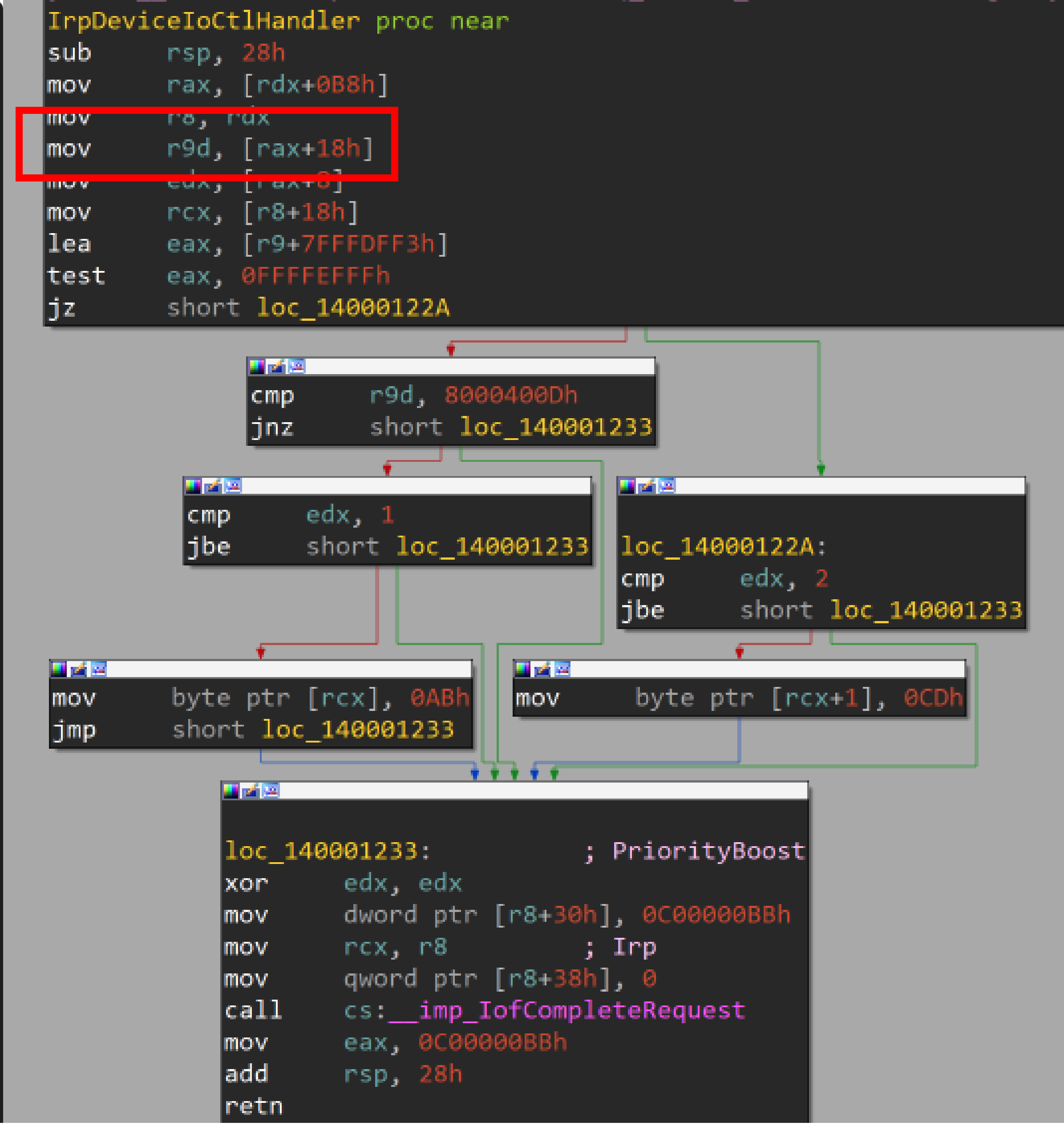
```
mov     byte ptr [rcx], 0ABh
jmp     short loc_140001233
```

```
mov     byte ptr [rcx+1], 0CDh
```

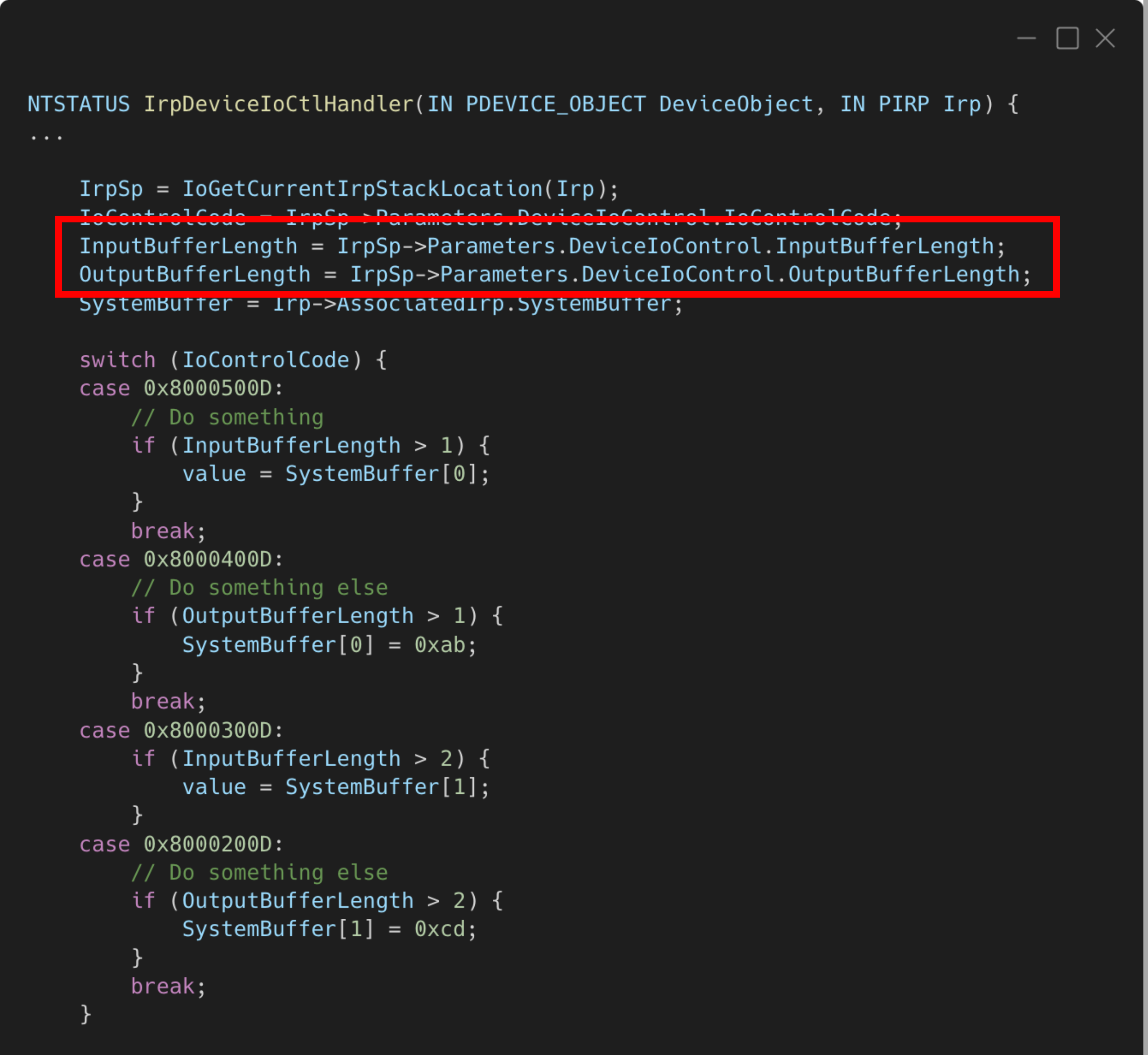
```
loc_140001233:          ; PriorityBoost
xor     edx, edx
mov     dword ptr [r8+30h], 0C00000BBh
mov     rcx, r8         ; Irp
mov     qword ptr [r8+38h], 0
call    cs:__imp_IofCompleteRequest
mov     eax, 0C00000BBh
add     rsp, 28h
retn
```



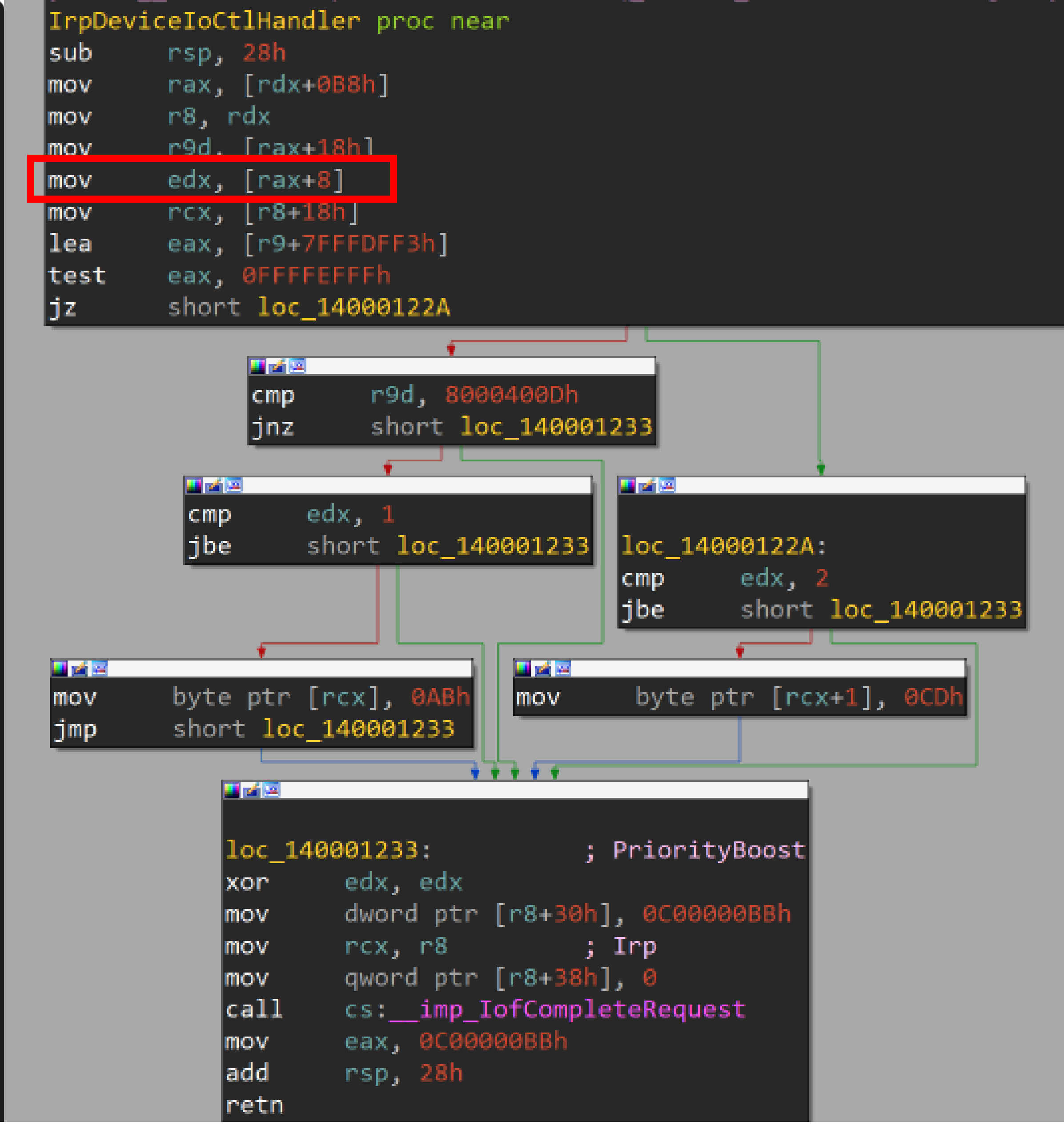
```
NTSTATUS IrpDeviceIoCtlHandler(IN PDEVICE_OBJECT DeviceObject, IN PIRP Irp) {
...

    IrpSp = IoGetCurrentIrpStackLocation(Irp);
    IoControlCode = IrpSp->Parameters.DeviceIoControl.IoControlCode;
    InputBufferLength = IrpSp->Parameters.DeviceIoControl.InputBufferLength;
    OutputBufferLength = IrpSp->Parameters.DeviceIoControl.OutputBufferLength;
    SystemBuffer = Irp->AssociatedIrp.SystemBuffer;

    switch (IoControlCode) {
    case 0x8000500D:
        // Do something
        if (InputBufferLength > 1) {
            value = SystemBuffer[0];
        }
        break;
    case 0x8000400D:
        // Do something else
        if (OutputBufferLength > 1) {
            SystemBuffer[0] = 0xab;
        }
        break;
    case 0x8000300D:
        if (InputBufferLength > 2) {
            value = SystemBuffer[1];
        }
    case 0x8000200D:
        // Do something else
        if (OutputBufferLength > 2) {
            SystemBuffer[1] = 0xcd;
        }
        break;
    }
```



```
IrpDeviceIoCtlHandler proc near
sub     rsp, 28h
mov     rax, [rdx+0B8h]
mov     r8, rdx
mov     r9d, [rax+18h]
mov     edx, [rax+8]
mov     rcx, [r8+18h]
lea     eax, [r9+7FFFDFF3h]
test    eax, 0FFFFEFFFh
jz      short loc_14000122A

cmp     r9d, 8000400Dh
jnz     short loc_140001233

cmp     edx, 1
jbe     short loc_140001233

loc_14000122A:
cmp     edx, 2
jbe     short loc_140001233

mov     byte ptr [rcx], 0ABh
jmp     short loc_140001233

mov     byte ptr [rcx+1], 0CDh

loc_140001233:          ; PriorityBoost
xor     edx, edx
mov     dword ptr [r8+30h], 0C00000BBh
mov     rcx, r8         ; Irp
mov     qword ptr [r8+38h], 0
call    cs:__imp_IofCompleteRequest
mov     eax, 0C00000BBh
add     rsp, 28h
retn
```



```
NTSTATUS IrpDeviceIoCtlHandler(IN PDEVICE_OBJECT DeviceObject, IN PIRP Irp) {
...

    IrpSp = IoGetCurrentIrpStackLocation(Irp);
    IoControlCode = IrpSp->Parameters.DeviceIoControl.IoControlCode;
    InputBufferLength = IrpSp->Parameters.DeviceIoControl.InputBufferLength;
    OutputBufferLength = IrpSp->Parameters.DeviceIoControl.OutputBufferLength;
    SystemBuffer = Irp->AssociatedIrp.SystemBuffer;

    switch (IoControlCode) {
    case 0x8000500D:
        // Do something
        if (InputBufferLength > 1) {
            value = SystemBuffer[0];
        }
        break;
    case 0x8000400D:
        // Do something else
        if (OutputBufferLength > 1) {
            SystemBuffer[0] = 0xab;
        }
        break;
    case 0x8000300D:
        if (InputBufferLength > 2) {
            value = SystemBuffer[1];
        }
    case 0x8000200D:
        // Do something else
        if (OutputBufferLength > 2) {
            SystemBuffer[1] = 0xcd;
        }
        break;
    }
```



```
IrpDeviceIoCtlHandler proc near
sub     rsp, 28h
mov     rax, [rdx+0B8h]
mov     r8, rdx
mov     r9d, [rax+18h]
mov     edx, [rax+8]
mov     rcx, [r8+18h]
lea     eax, [r9+7FFFDFF3h]
test    eax, 0FFFFEFFFh
jz      short loc_14000122A
```

```
cmp     r9d, 8000400Dh
jnz     short loc_140001233
```

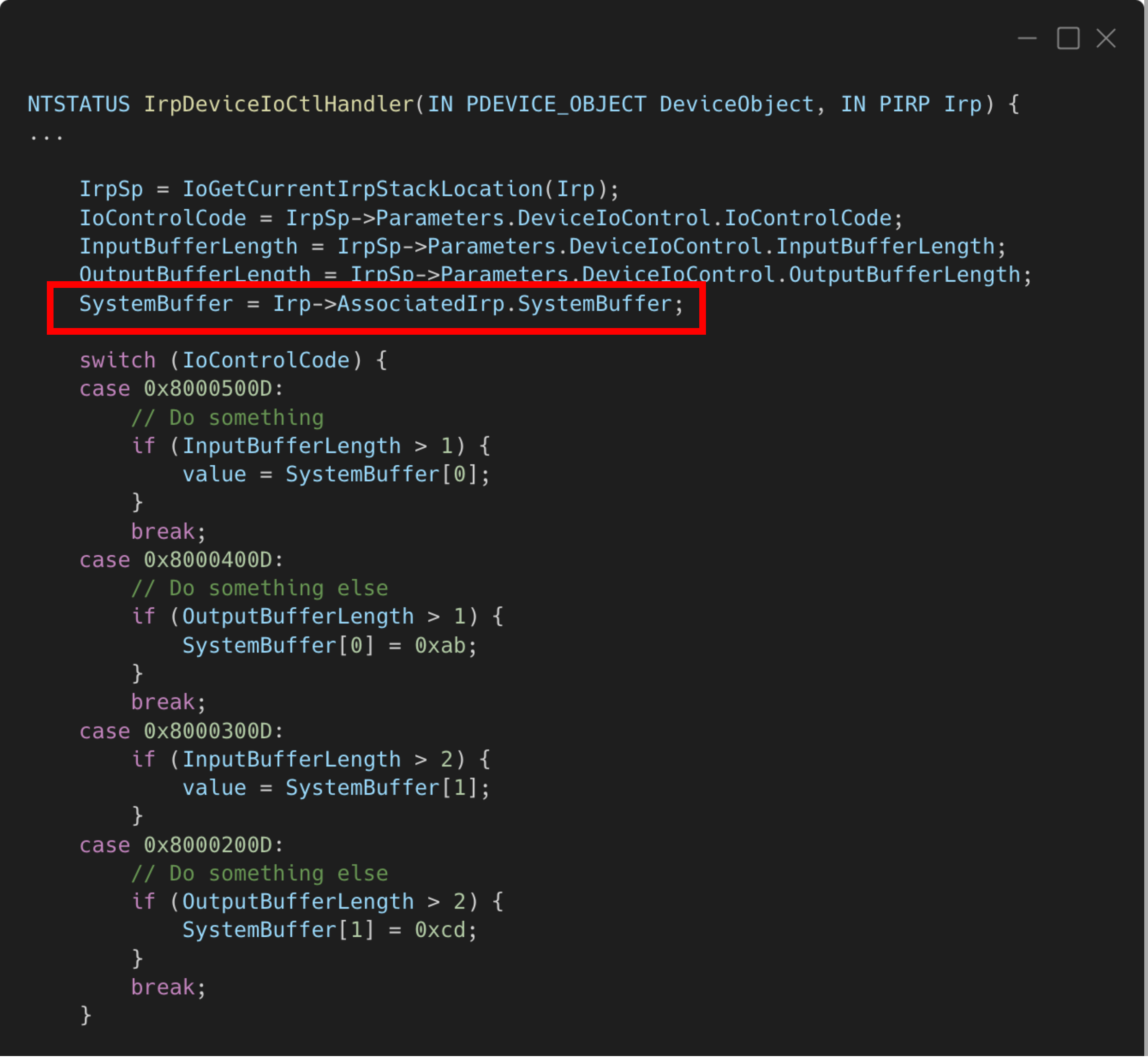
```
cmp     edx, 1
jbe     short loc_140001233
```

```
loc_14000122A:
cmp     edx, 2
jbe     short loc_140001233
```

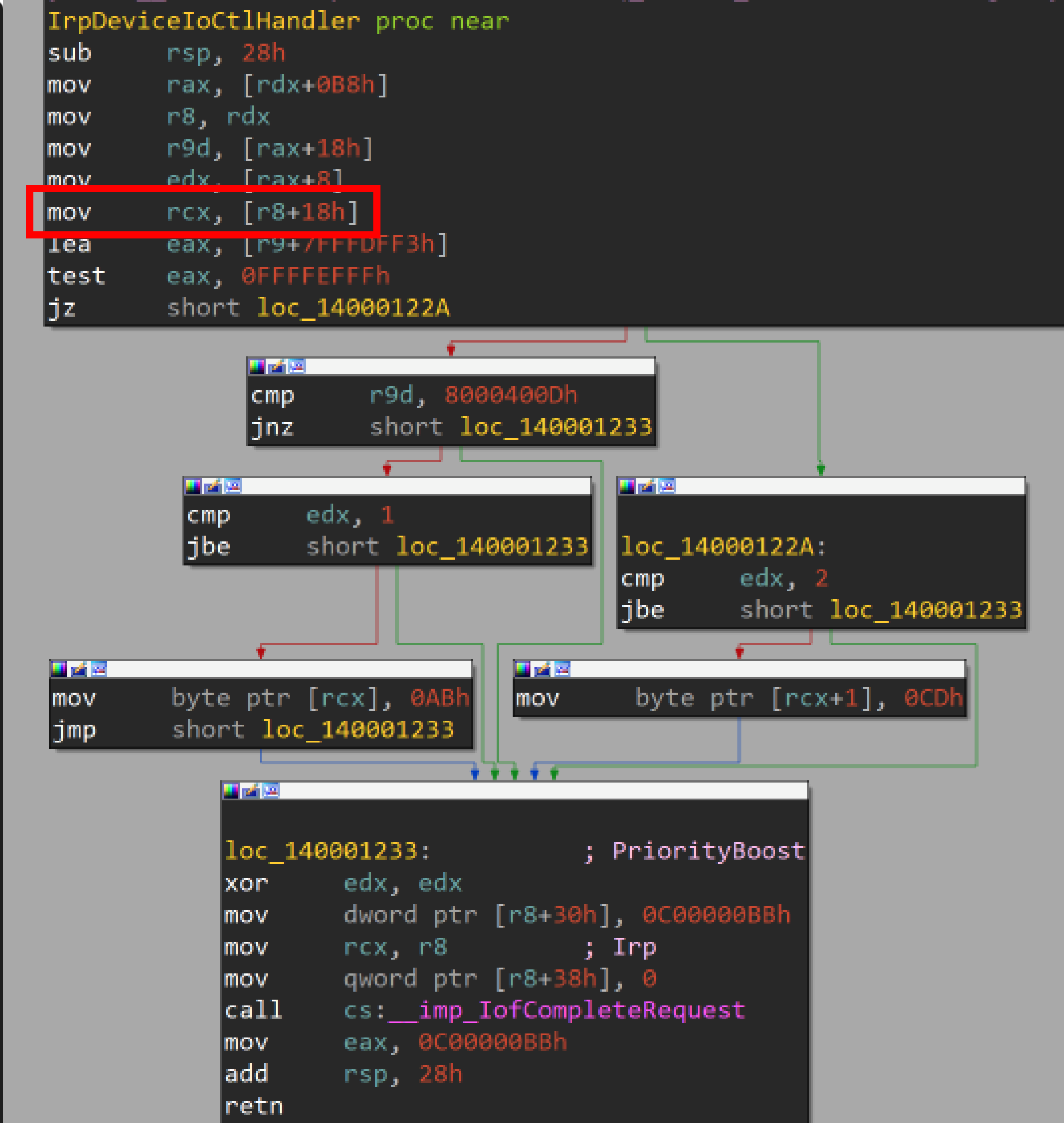
```
mov     byte ptr [rcx], 0ABh
jmp     short loc_140001233
```

```
mov     byte ptr [rcx+1], 0CDh
```

```
loc_140001233:              ; PriorityBoost
xor     edx, edx
mov     dword ptr [r8+30h], 0C00000BBh
mov     rcx, r8             ; Irp
mov     qword ptr [r8+38h], 0
call    cs:__imp_IofCompleteRequest
mov     eax, 0C00000BBh
add     rsp, 28h
retn
```



```
NTSTATUS IrpDeviceIoCtlHandler(IN PDEVICE_OBJECT DeviceObject, IN PIRP Irp) {
...

    IrpSp = IoGetCurrentIrpStackLocation(Irp);
    IoControlCode = IrpSp->Parameters.DeviceIoControl.IoControlCode;
    InputBufferLength = IrpSp->Parameters.DeviceIoControl.InputBufferLength;
    OutputBufferLength = IrpSp->Parameters.DeviceIoControl.OutputBufferLength;
    SystemBuffer = Irp->AssociatedIrp.SystemBuffer;

    switch (IoControlCode) {
    case 0x8000500D:
        // Do something
        if (InputBufferLength > 1) {
            value = SystemBuffer[0];
        }
        break;
    case 0x8000400D:
        // Do something else
        if (OutputBufferLength > 1) {
            SystemBuffer[0] = 0xab;
        }
        break;
    case 0x8000300D:
        if (InputBufferLength > 2) {
            value = SystemBuffer[1];
        }
    case 0x8000200D:
        // Do something else
        if (OutputBufferLength > 2) {
            SystemBuffer[1] = 0xcd;
        }
        break;
    }
```



```
IrpDeviceIoCtlHandler proc near
sub     rsp, 28h
mov     rax, [rdx+0B8h]
mov     r8, rdx
mov     r9d, [rax+18h]
mov     edx, [rax+8]
mov     rcx, [r8+18h]
lea     eax, [r9+7FFFDFF3h]
test    eax, 0FFFFEFFFh
jz      short loc_14000122A
```

```
cmp     r9d, 8000400Dh
jnz     short loc_140001233
```

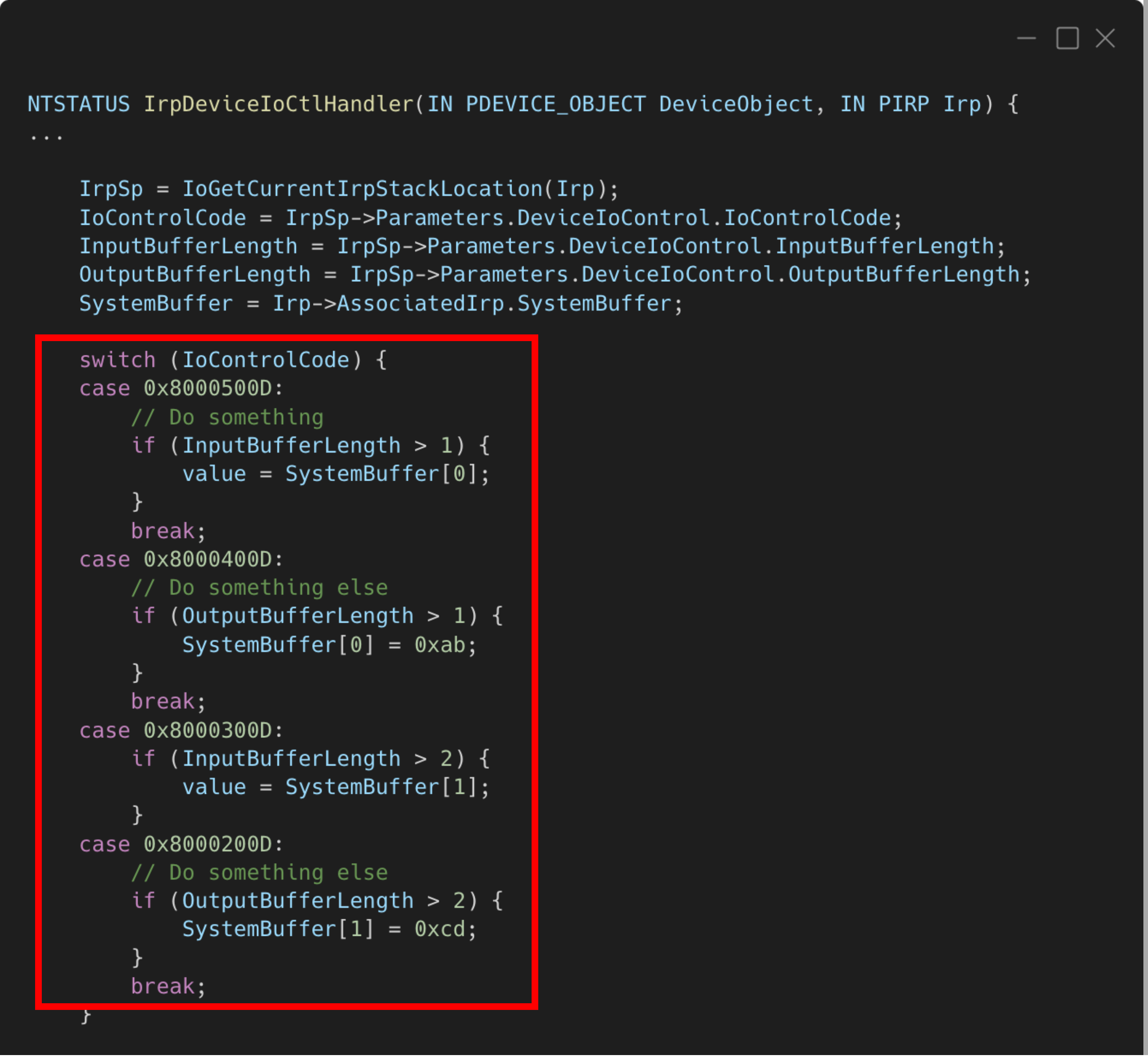
```
cmp     edx, 1
jbe     short loc_140001233
```

```
loc_14000122A:
cmp     edx, 2
jbe     short loc_140001233
```

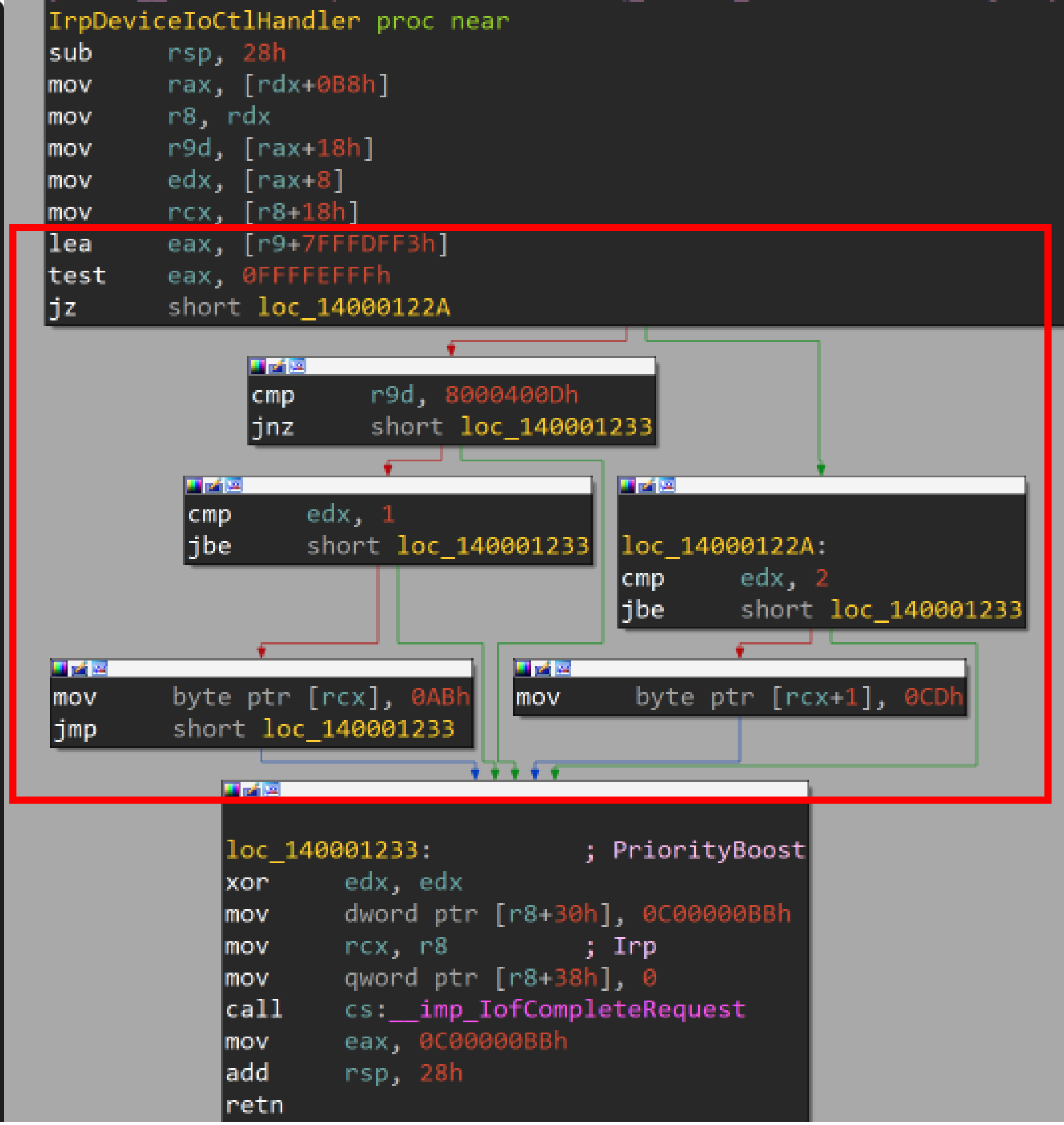
```
mov     byte ptr [rcx], 0ABh
jmp     short loc_140001233
```

```
mov     byte ptr [rcx+1], 0CDh
```

```
loc_140001233:                  ; PriorityBoost
xor     edx, edx
mov     dword ptr [r8+30h], 0C00000BBh
mov     rcx, r8                 ; Irp
mov     qword ptr [r8+38h], 0
call    cs:__imp_IofCompleteRequest
mov     eax, 0C00000BBh
add     rsp, 28h
retn
```



```
NTSTATUS IrpDeviceIoCtlHandler(IN PDEVICE_OBJECT DeviceObject, IN PIRP Irp) {
...

    IrpSp = IoGetCurrentIrpStackLocation(Irp);
    IoControlCode = IrpSp->Parameters.DeviceIoControl.IoControlCode;
    InputBufferLength = IrpSp->Parameters.DeviceIoControl.InputBufferLength;
    OutputBufferLength = IrpSp->Parameters.DeviceIoControl.OutputBufferLength;
    SystemBuffer = Irp->AssociatedIrp.SystemBuffer;

    switch (IoControlCode) {
    case 0x8000500D:
        // Do something
        if (InputBufferLength > 1) {
            value = SystemBuffer[0];
        }
        break;
    case 0x8000400D:
        // Do something else
        if (OutputBufferLength > 1) {
            SystemBuffer[0] = 0xab;
        }
        break;
    case 0x8000300D:
        if (InputBufferLength > 2) {
            value = SystemBuffer[1];
        }
    case 0x8000200D:
        // Do something else
        if (OutputBufferLength > 2) {
            SystemBuffer[1] = 0xcd;
        }
        break;
    }
```



```
IrpDeviceIoCtlHandler proc near
sub     rsp, 28h
mov     rax, [rdx+0B8h]
mov     r8, rdx
mov     r9d, [rax+18h]
mov     edx, [rax+8]
mov     rcx, [r8+18h]
lea     eax, [r9+7FFFDFF3h]
test    eax, 0FFFFEFFFh
jz      short loc_14000122A
```

```
cmp     r9d, 8000400Dh
jnz     short loc_140001233
```

```
cmp     edx, 1
jbe     short loc_140001233
```

```
loc_14000122A:
cmp     edx, 2
jbe     short loc_140001233
```

```
mov     byte ptr [rcx], 0ABh
jmp     short loc_140001233
```

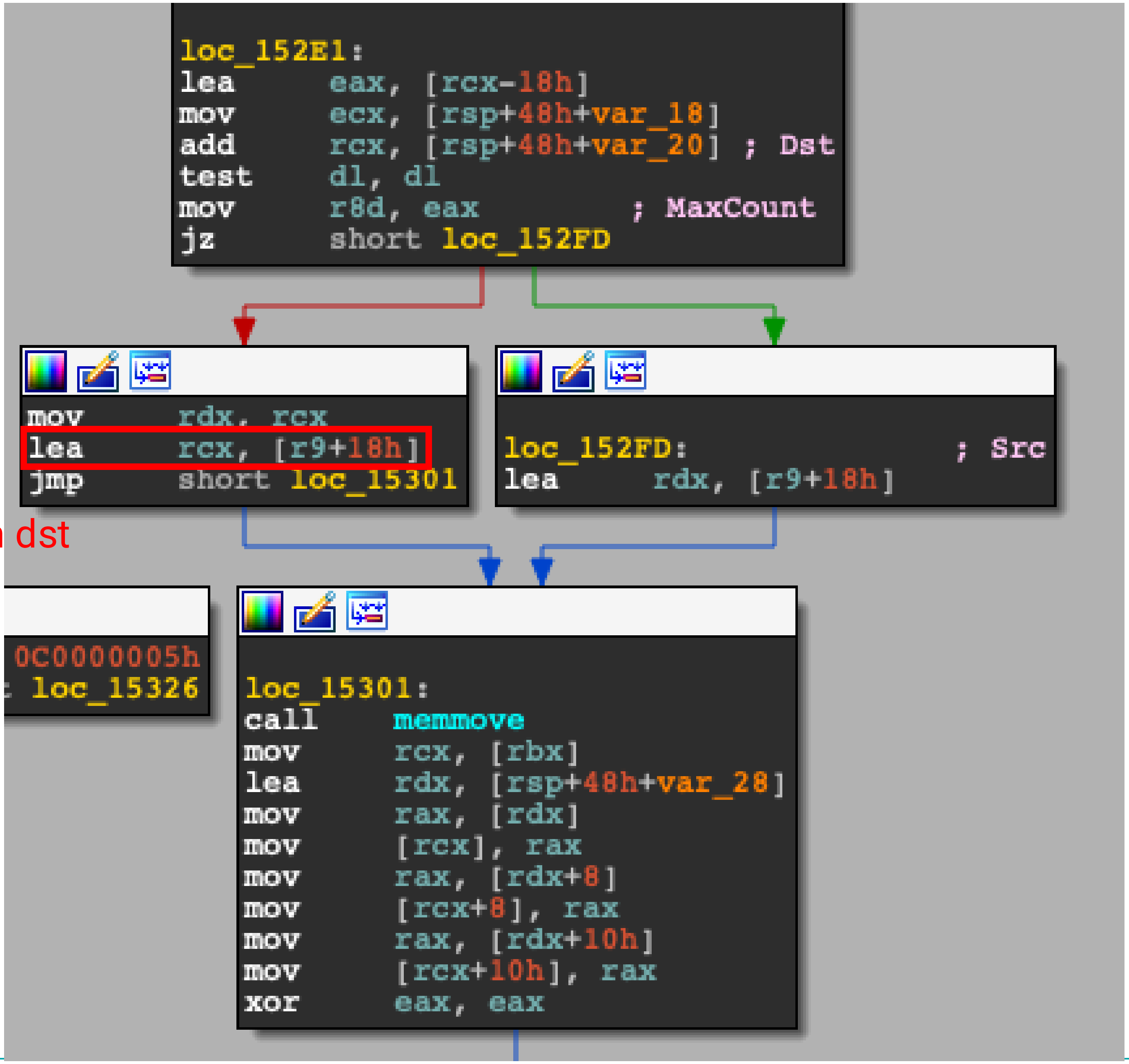
```
mov     byte ptr [rcx+1], 0CDh
```

```
loc_140001233:              ; PriorityBoost
xor     edx, edx
mov     dword ptr [r8+30h], 0C00000BBh
mov     rcx, r8             ; Irp
mov     qword ptr [r8+38h], 0
call    cs:__imp_IofCompleteRequest
mov     eax, 0C00000BBh
add     rsp, 28h
retn
```

Common Vulnerabilities

Dangerous legitimate functionality

- Arbitrary read+write



```
loc_152E1:
lea     eax, [rcx-18h]
mov     ecx, [rsp+48h+var_18]
add     rcx, [rsp+48h+var_20] ; Dst
test    dl, dl
mov     r8d, eax          ; MaxCount
jz      short loc_152FD
```

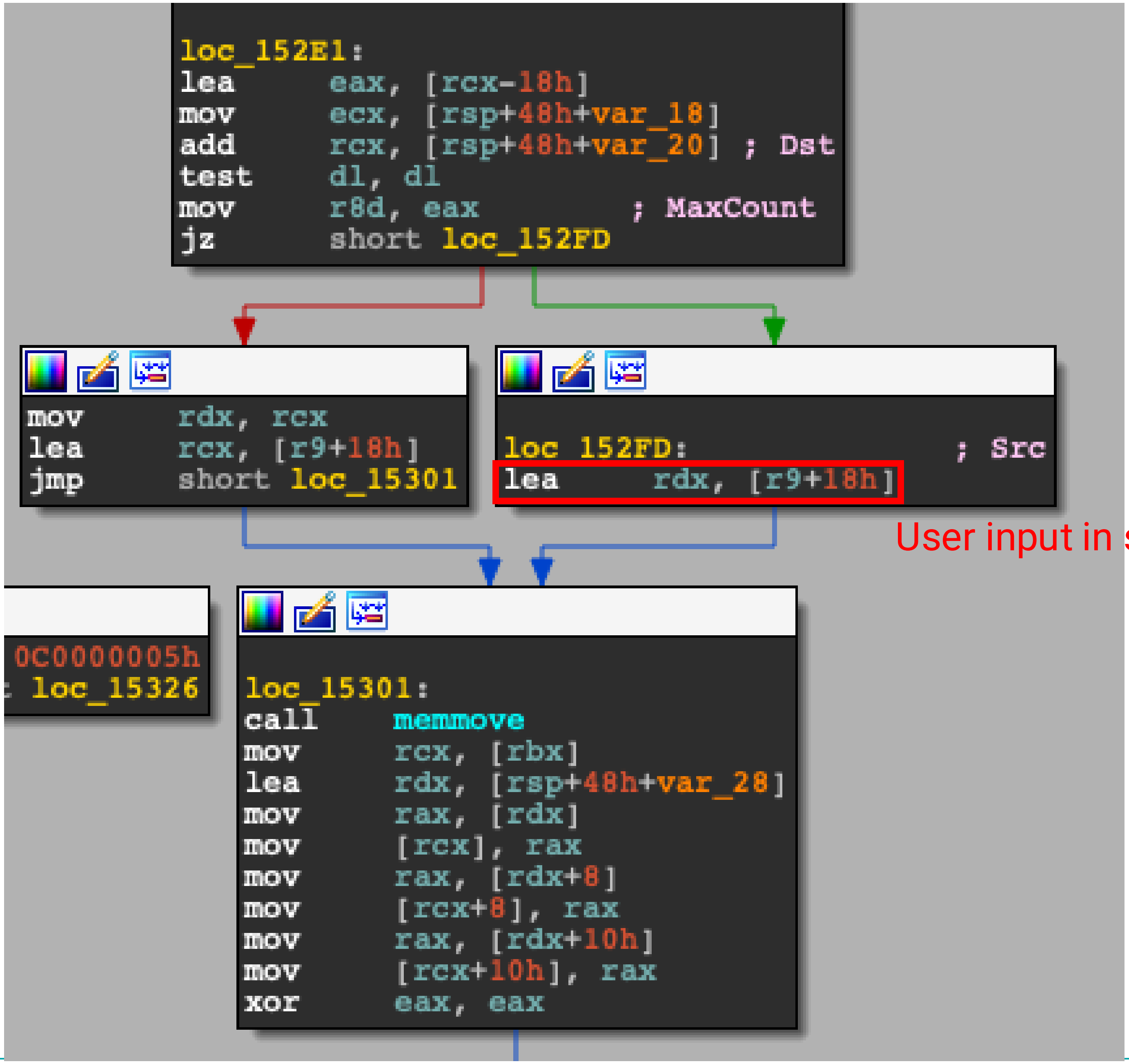
```
mov     rdx, rcx
lea     rcx, [r9+18h]
jmp     short loc_15301
```

```
loc_152FD:                ; Src
lea     rdx, [r9+18h]
```

```
0C0000005h
t loc_15326
```

```
loc_15301:
call    memmove
mov     rcx, [rbx]
lea     rdx, [rsp+48h+var_28]
mov     rax, [rdx]
mov     [rcx], rax
mov     rax, [rdx+8]
mov     [rcx+8], rax
mov     rax, [rdx+10h]
mov     [rcx+10h], rax
xor     eax, eax
```

User input in dst



```
loc_152E1:
lea     eax, [rcx-18h]
mov     ecx, [rsp+48h+var_18]
add     rcx, [rsp+48h+var_20] ; Dst
test    dl, dl
mov     r8d, eax              ; MaxCount
jz      short loc_152FD
```

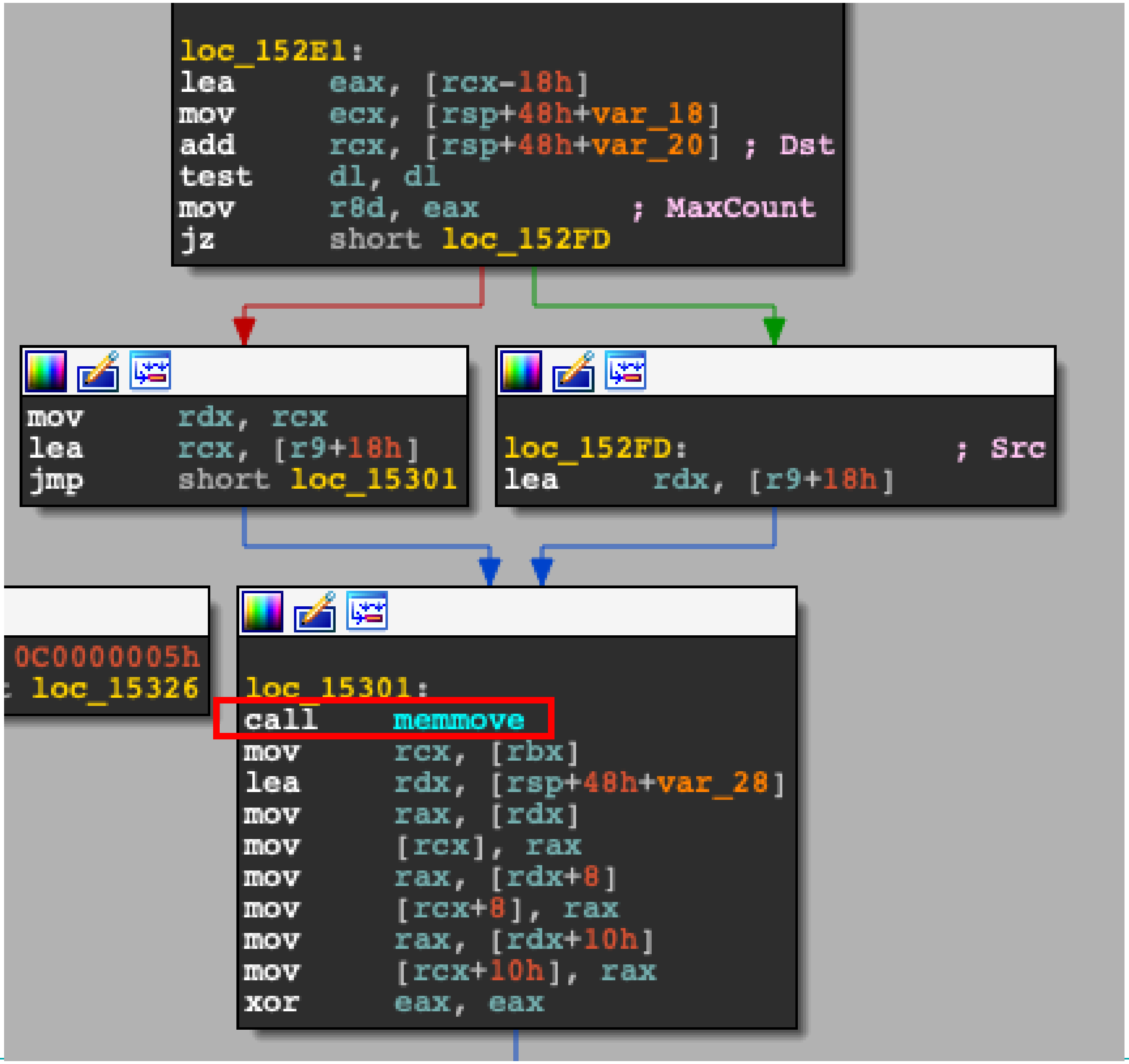
```
mov     rdx, rcx
lea     rcx, [r9+18h]
jmp     short loc_15301
```

```
loc_152FD:                    ; Src
lea     rdx, [r9+18h]
```

```
0C0000005h
loc_15326
```

```
loc_15301:
call    memmove
mov     rcx, [rbx]
lea     rdx, [rsp+48h+var_28]
mov     rax, [rdx]
mov     [rcx], rax
mov     rax, [rdx+8]
mov     [rcx+8], rax
mov     rax, [rdx+10h]
mov     [rcx+10h], rax
xor     eax, eax
```

User input in src



```
loc_152E1:
lea     eax, [rcx-18h]
mov     ecx, [rsp+48h+var_18]
add     rcx, [rsp+48h+var_20] ; Dst
test    dl, dl
mov     r8d, eax              ; MaxCount
jz      short loc_152FD
```

```
mov     rdx, rcx
lea     rcx, [r9+18h]
jmp     short loc_15301
```

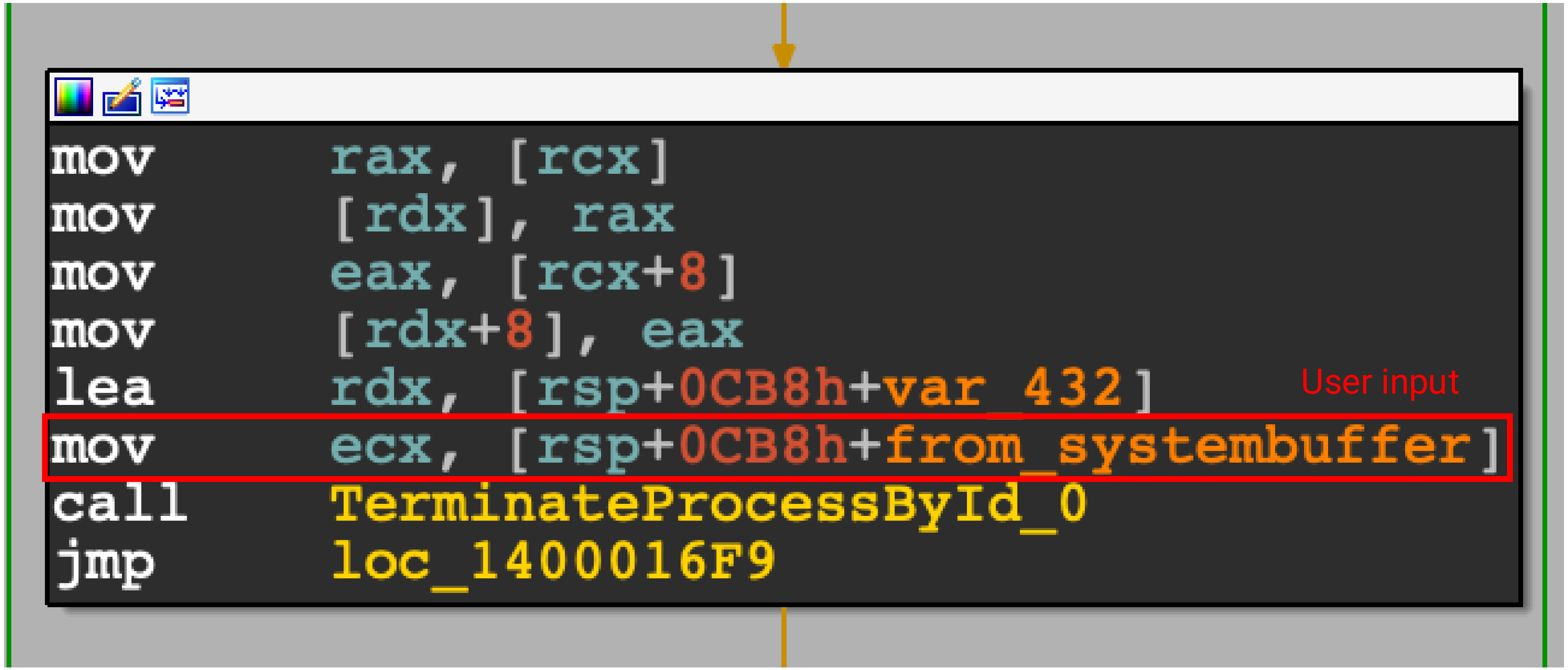
```
loc_152FD:                    ; Src
lea     rdx, [r9+18h]
```

```
0C0000005h
loc_15326
```

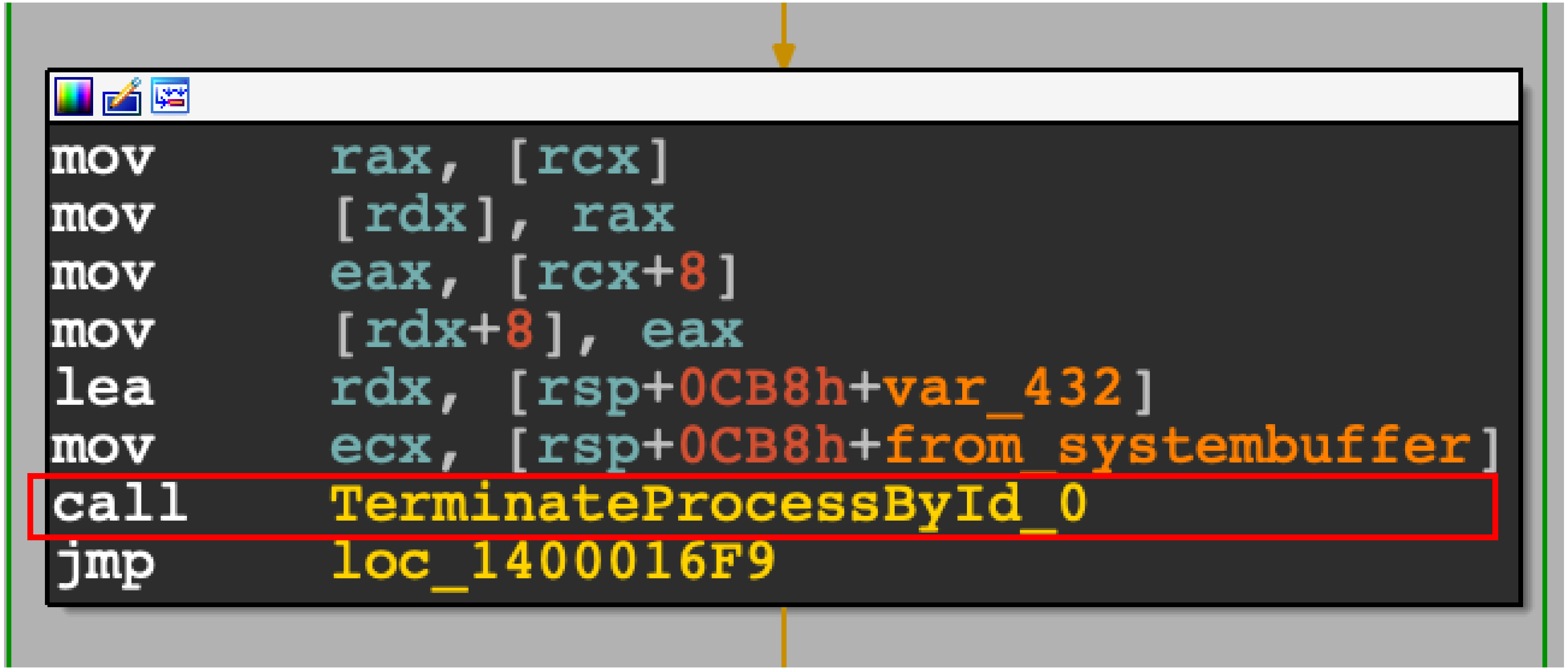
```
loc_15301:
call    memmove
mov     rcx, [rbx]
lea     rdx, [rsp+48h+var_28]
mov     rax, [rdx]
mov     [rcx], rax
mov     rax, [rdx+8]
mov     [rcx+8], rax
mov     rax, [rdx+10h]
mov     [rcx+10h], rax
xor     eax, eax
```

Dangerous legitimate functionality

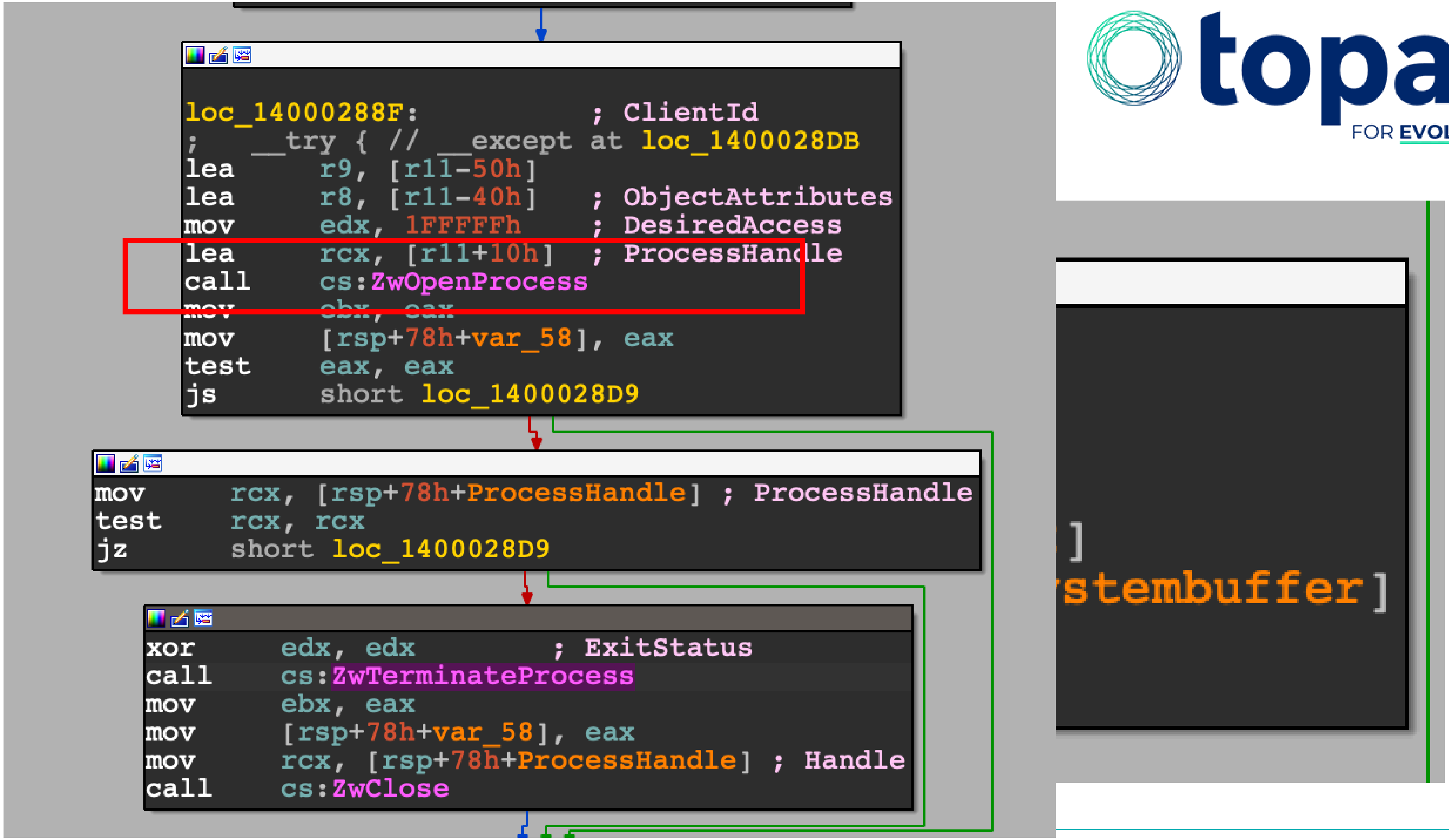
- Arbitrary read+write
- Terminate process



```
mov     rax, [rcx]
mov     [rdx], rax
mov     eax, [rcx+8]
mov     [rdx+8], eax
lea     rdx, [rsp+0CB8h+var_432]
mov     ecx, [rsp+0CB8h+from_systembuffer]
call    TerminateProcessById_0
jmp     loc_1400016F9
```



```
mov     rax, [rcx]
mov     [rdx], rax
mov     eax, [rcx+8]
mov     [rdx+8], eax
lea     rdx, [rsp+0CB8h+var_432]
mov     ecx, [rsp+0CB8h+from_systembuffer]
call    TerminateProcessById_0
jmp     loc_1400016F9
```



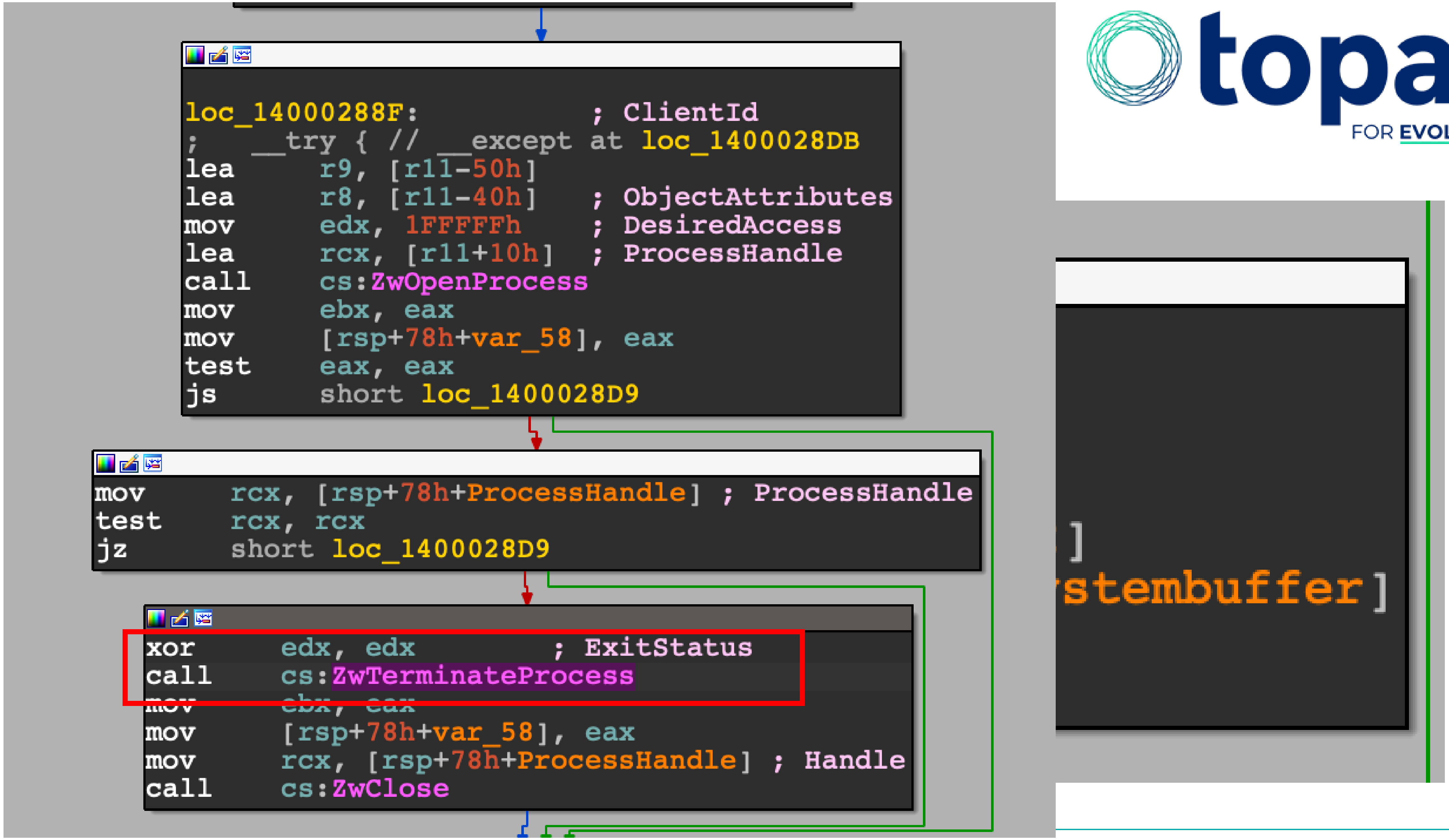
```
loc_14000288F:                    ; ClientId
;   __try { // __except at loc_1400028DB
lea     r9, [r11-50h]
lea     r8, [r11-40h]         ; ObjectAttributes
mov     edx, 1FFFFFh          ; DesiredAccess
lea     rcx, [r11+10h]        ; ProcessHandle
call    cs:ZwOpenProcess
mov     ebx, eax
mov     [rsp+78h+var_58], eax
test    eax, eax
js      short loc_1400028D9
```

```
mov     rcx, [rsp+78h+ProcessHandle] ; ProcessHandle
test    rcx, rcx
jz      short loc_1400028D9
```

```
xor     edx, edx              ; ExitStatus
call    cs:ZwTerminateProcess
mov     ebx, eax
mov     [rsp+78h+var_58], eax
mov     rcx, [rsp+78h+ProcessHandle] ; Handle
call    cs:ZwClose
```

```
]
stembuffer]
```

topaz® FOR EVOLUTION



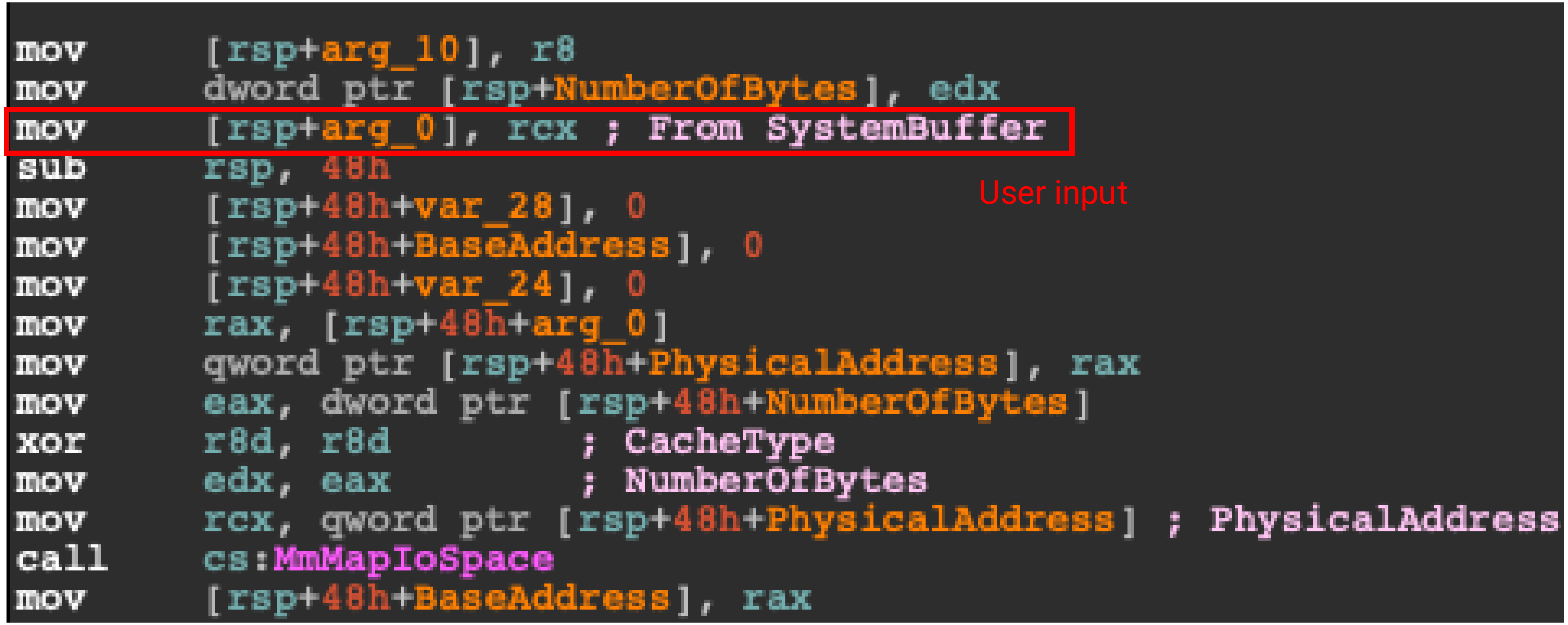
```
loc_14000288F:                  ; ClientId
;   __try { // __except at loc_1400028DB
lea     r9, [r11-50h]
lea     r8, [r11-40h]   ; ObjectAttributes
mov     edx, 1FFFFFh    ; DesiredAccess
lea     rcx, [r11+10h]  ; ProcessHandle
call    cs:ZwOpenProcess
mov     ebx, eax
mov     [rsp+78h+var_58], eax
test    eax, eax
js      short loc_1400028D9
```

```
mov     rcx, [rsp+78h+ProcessHandle] ; ProcessHandle
test    rcx, rcx
jz      short loc_1400028D9
```

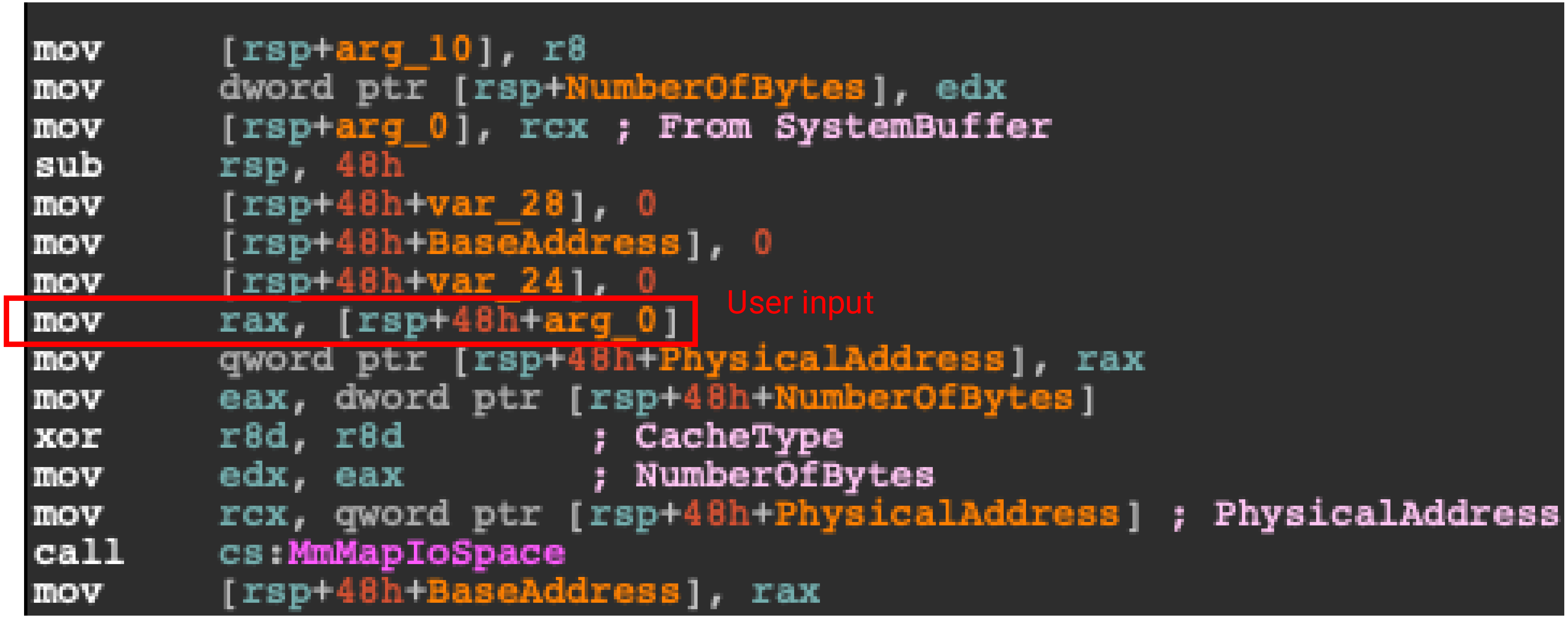
```
xor     edx, edx            ; ExitStatus
call    cs:ZwTerminateProcess
mov     ebx, eax
mov     [rsp+78h+var_58], eax
mov     rcx, [rsp+78h+ProcessHandle] ; Handle
call    cs:ZwClose
```

Dangerous legitimate functionality

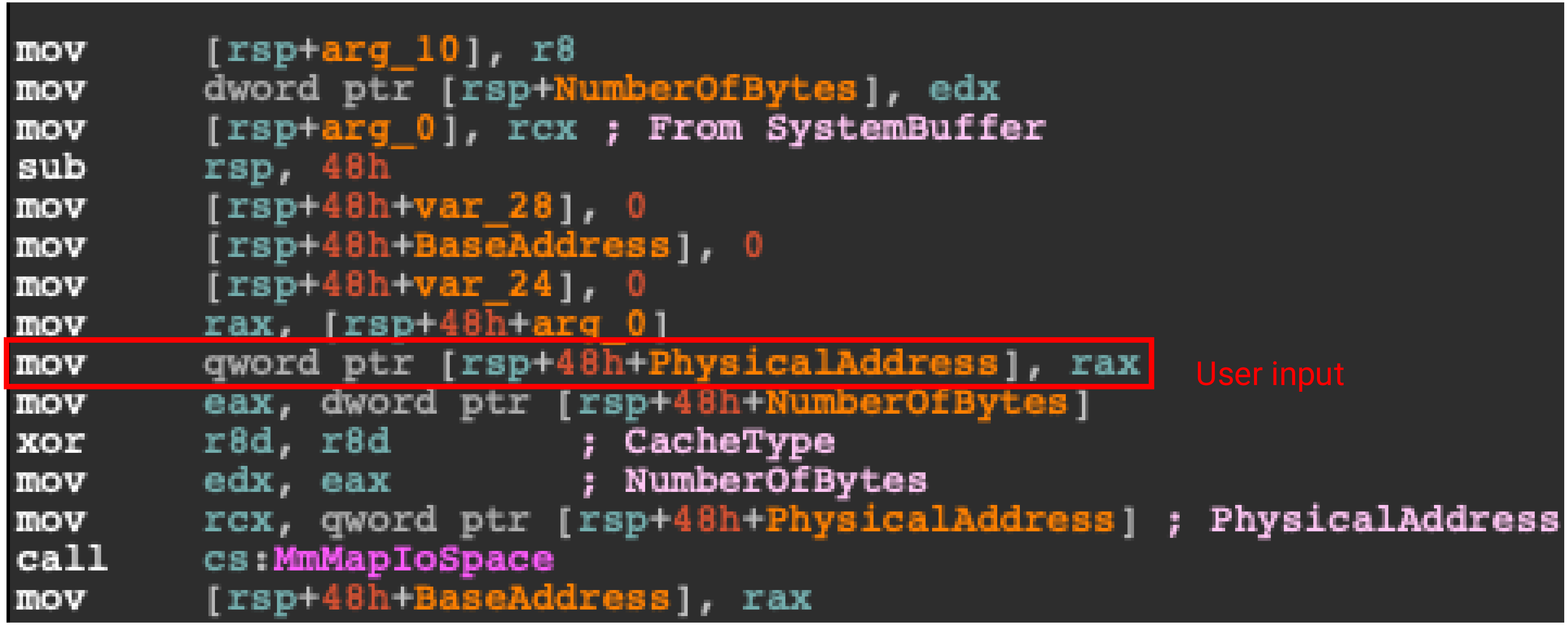
- Arbitrary read+write
- Terminate process
- Physical memory mapping



```
mov     [rsp+arg_10], r8
mov     dword ptr [rsp+NumberOfBytes], edx
mov     [rsp+arg_0], rcx ; From SystemBuffer
sub     rsp, 48h
mov     [rsp+48h+var_28], 0
mov     [rsp+48h+BaseAddress], 0
mov     [rsp+48h+var_24], 0
mov     rax, [rsp+48h+arg_0]
mov     qword ptr [rsp+48h+PhysicalAddress], rax
mov     eax, dword ptr [rsp+48h+NumberOfBytes]
xor     r8d, r8d          ; CacheType
mov     edx, eax          ; NumberOfBytes
mov     rcx, qword ptr [rsp+48h+PhysicalAddress] ; PhysicalAddress
call    cs:MmMapIoSpace
mov     [rsp+48h+BaseAddress], rax
```

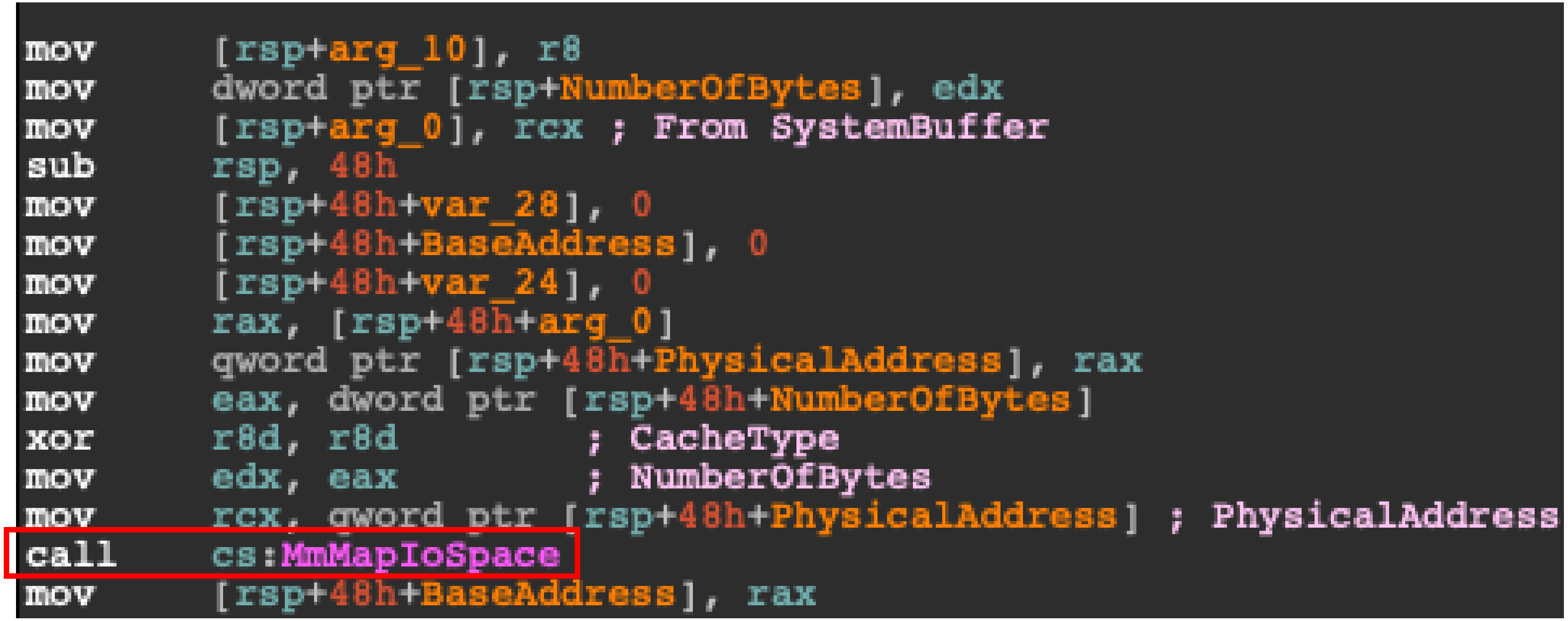


```
mov     [rsp+arg_10], r8
mov     dword ptr [rsp+NumberOfBytes], edx
mov     [rsp+arg_0], rcx ; From SystemBuffer
sub     rsp, 48h
mov     [rsp+48h+var_28], 0
mov     [rsp+48h+BaseAddress], 0
mov     [rsp+48h+var_24], 0
mov     rax, [rsp+48h+arg_0]
mov     qword ptr [rsp+48h+PhysicalAddress], rax
mov     eax, dword ptr [rsp+48h+NumberOfBytes]
xor     r8d, r8d                ; CacheType
mov     edx, eax                ; NumberOfBytes
mov     rcx, qword ptr [rsp+48h+PhysicalAddress] ; PhysicalAddress
call    cs:MmMapIoSpace
mov     [rsp+48h+BaseAddress], rax
```



```
mov     [rsp+arg_10], r8
mov     dword ptr [rsp+NumberOfBytes], edx
mov     [rsp+arg_0], rcx ; From SystemBuffer
sub     rsp, 48h
mov     [rsp+48h+var_28], 0
mov     [rsp+48h+BaseAddress], 0
mov     [rsp+48h+var_24], 0
mov     rax, [rsp+48h+arg_0]
mov     qword ptr [rsp+48h+PhysicalAddress], rax
mov     eax, dword ptr [rsp+48h+NumberOfBytes]
xor     r8d, r8d            ; CacheType
mov     edx, eax            ; NumberOfBytes
mov     rcx, qword ptr [rsp+48h+PhysicalAddress] ; PhysicalAddress
call    cs:MmMapIoSpace
mov     [rsp+48h+BaseAddress], rax
```

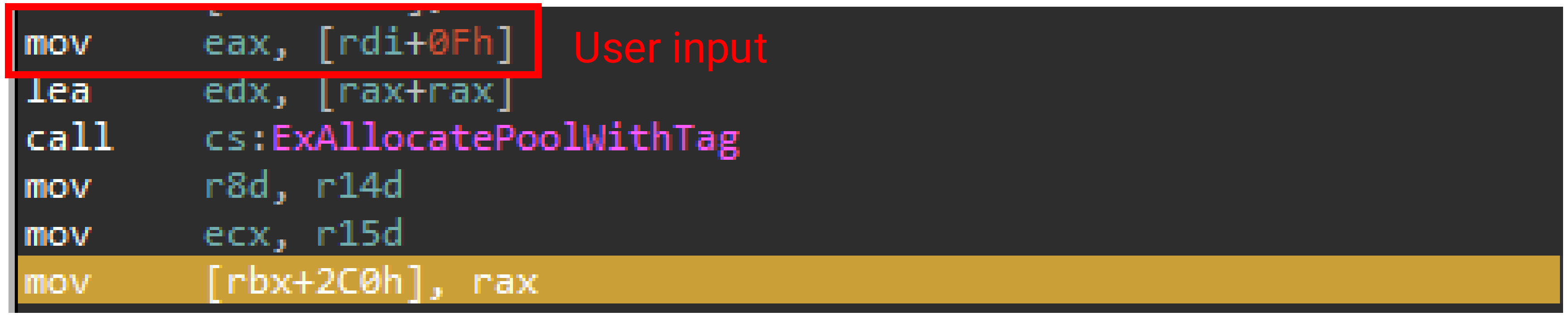
User input



```
mov     [rsp+arg_10], r8
mov     dword ptr [rsp+NumberOfBytes], edx
mov     [rsp+arg_0], rcx ; From SystemBuffer
sub     rsp, 48h
mov     [rsp+48h+var_28], 0
mov     [rsp+48h+BaseAddress], 0
mov     [rsp+48h+var_24], 0
mov     rax, [rsp+48h+arg_0]
mov     qword ptr [rsp+48h+PhysicalAddress], rax
mov     eax, dword ptr [rsp+48h+NumberOfBytes]
xor     r8d, r8d        ; CacheType
mov     edx, eax        ; NumberOfBytes
mov     rcx, qword ptr [rsp+48h+PhysicalAddress] ; PhysicalAddress
call    cs:MmMapIoSpace
mov     [rsp+48h+BaseAddress], rax
```

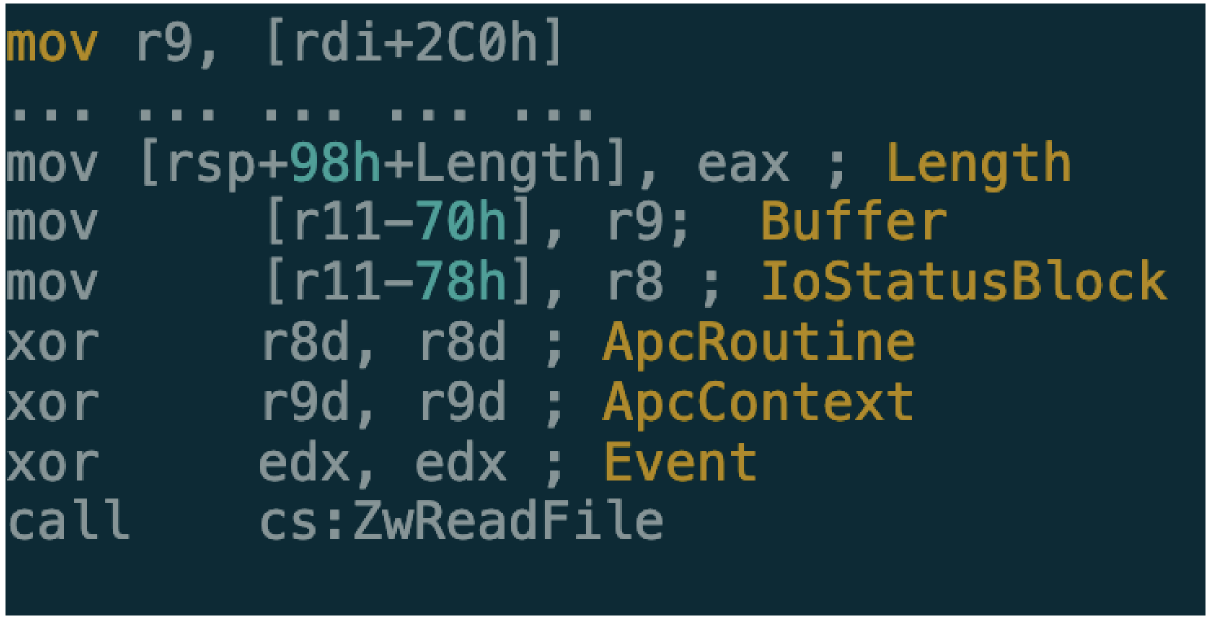
Buffer overflow

- User-controlled size of allocation

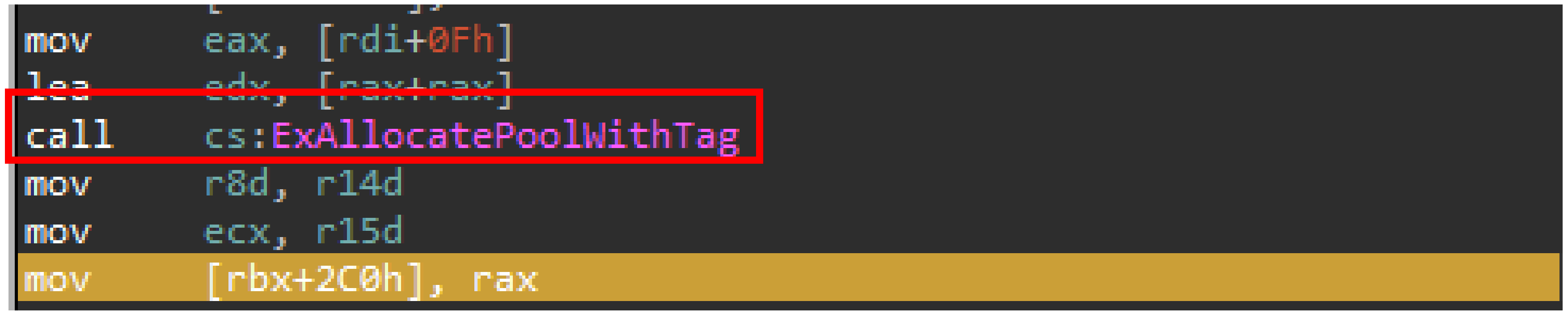


```
mov     eax, [rdi+0Fh]
lea     edx, [rax+rax]
call    cs:ExAllocatePoolWithTag
mov     r8d, r14d
mov     ecx, r15d
mov     [rbx+2C0h], rax
```

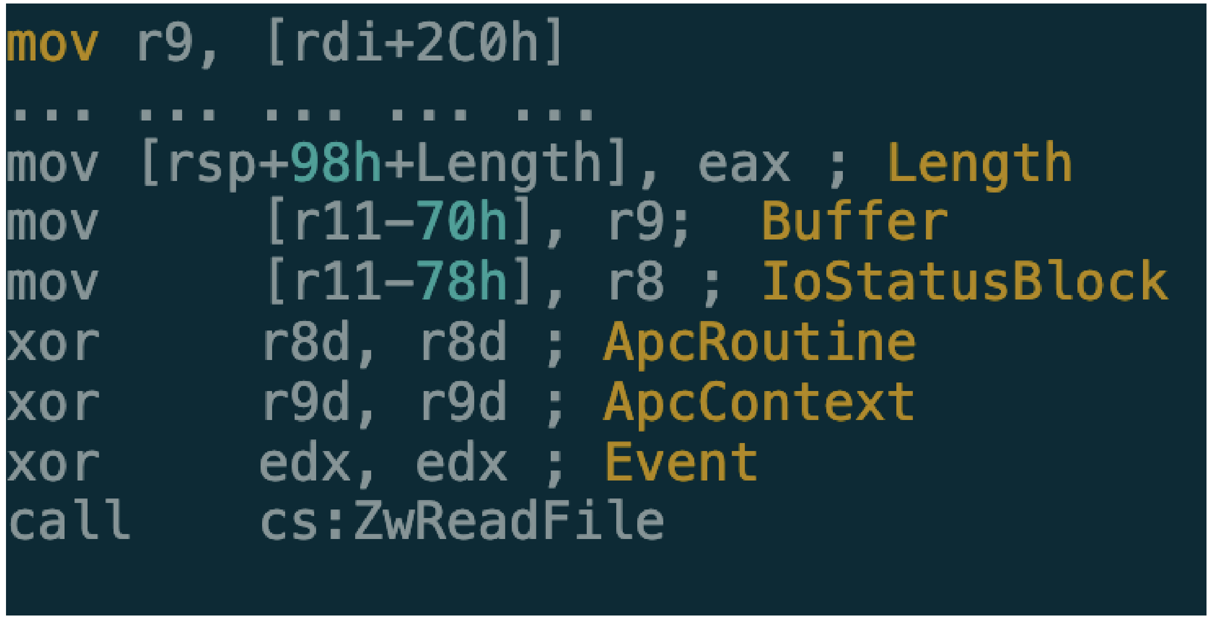
User input



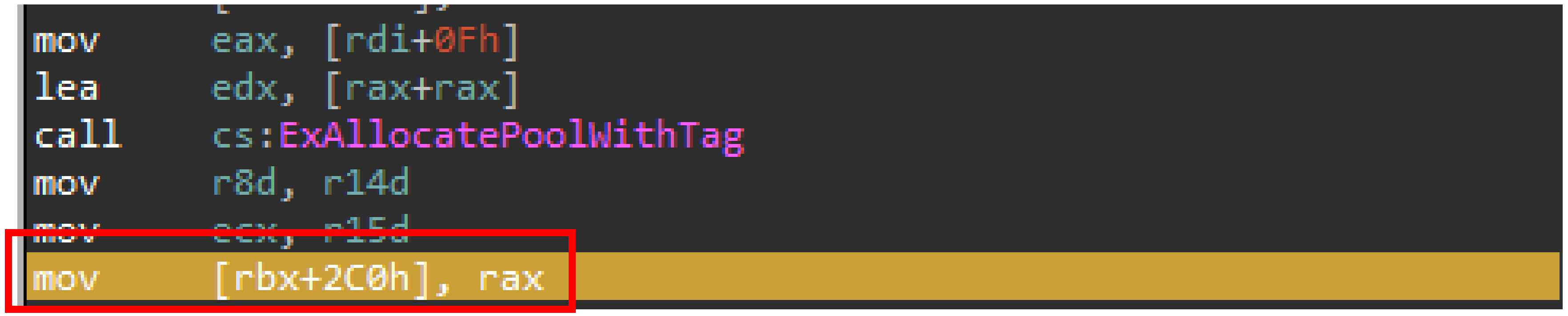
```
mov r9, [rdi+2C0h]
... ... ... ... ...
mov [rsp+98h+Length], eax ; Length
mov     [r11-70h], r9;  Buffer
mov     [r11-78h], r8 ; IoStatusBlock
xor     r8d, r8d ; ApcRoutine
xor     r9d, r9d ; ApcContext
xor     edx, edx ; Event
call    cs:ZwReadFile
```



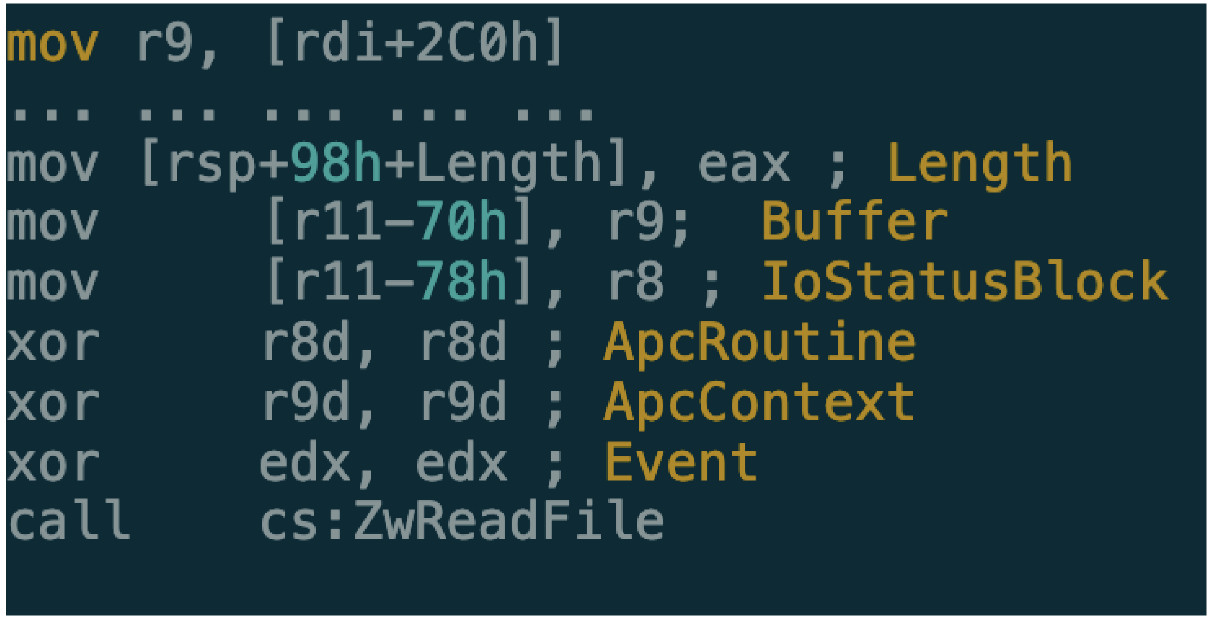
```
mov     eax, [rdi+0Fh]
lea     edx, [rax+rax]
call    cs:ExAllocatePoolWithTag
mov     r8d, r14d
mov     ecx, r15d
mov     [rbx+2C0h], rax
```



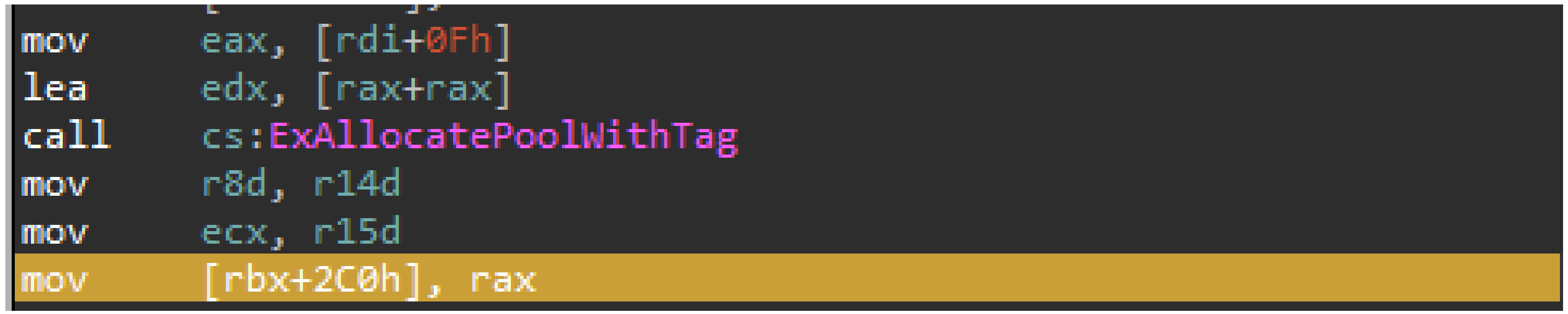
```
mov r9, [rdi+2C0h]
... ... ... ... ...
mov [rsp+98h+Length], eax ; Length
mov      [r11-70h], r9;  Buffer
mov      [r11-78h], r8 ; IoStatusBlock
xor      r8d, r8d ; ApcRoutine
xor      r9d, r9d ; ApcContext
xor      edx, edx ; Event
call     cs:ZwReadFile
```



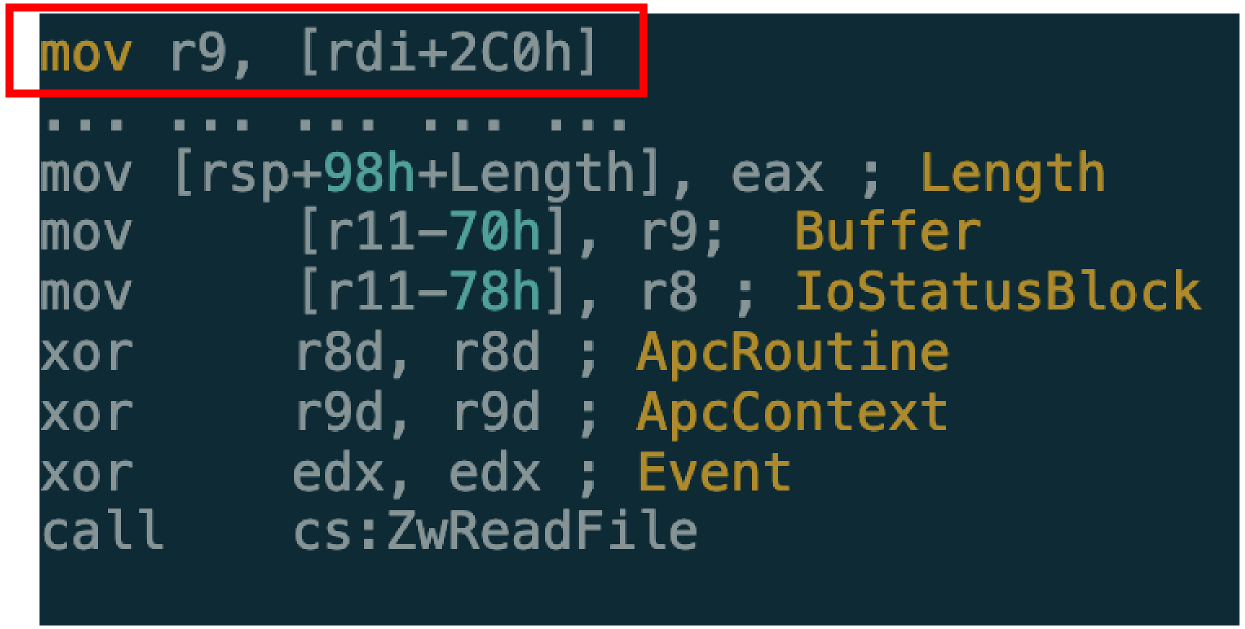
```
mov     eax, [rdi+0Fh]
lea     edx, [rax+rax]
call    cs:ExAllocatePoolWithTag
mov     r8d, r14d
mov     ecx, r15d
mov     [rbx+2C0h], rax
```



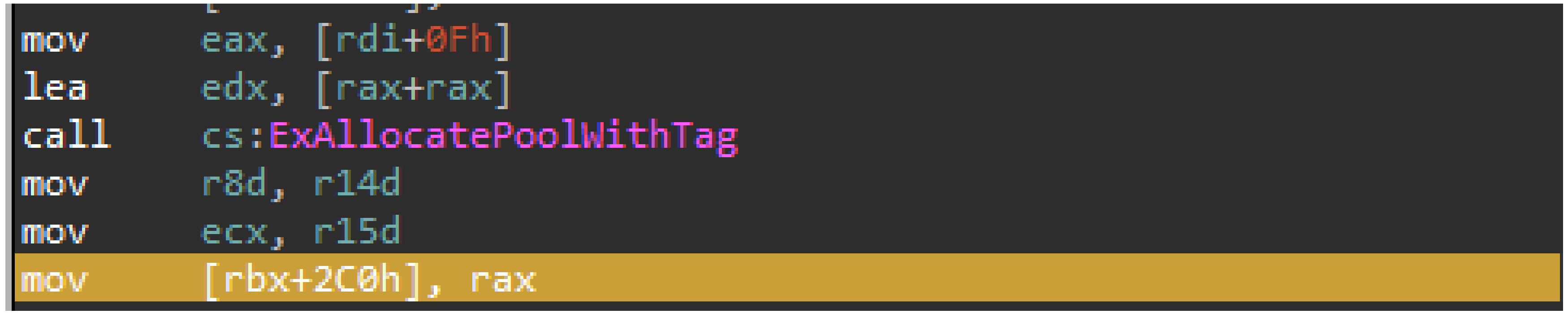
```
mov r9, [rdi+2C0h]
... ... ... ... ...
mov [rsp+98h+Length], eax ; Length
mov      [r11-70h], r9;  Buffer
mov      [r11-78h], r8 ; IoStatusBlock
xor      r8d, r8d ; ApcRoutine
xor      r9d, r9d ; ApcContext
xor      edx, edx ; Event
call     cs:ZwReadFile
```



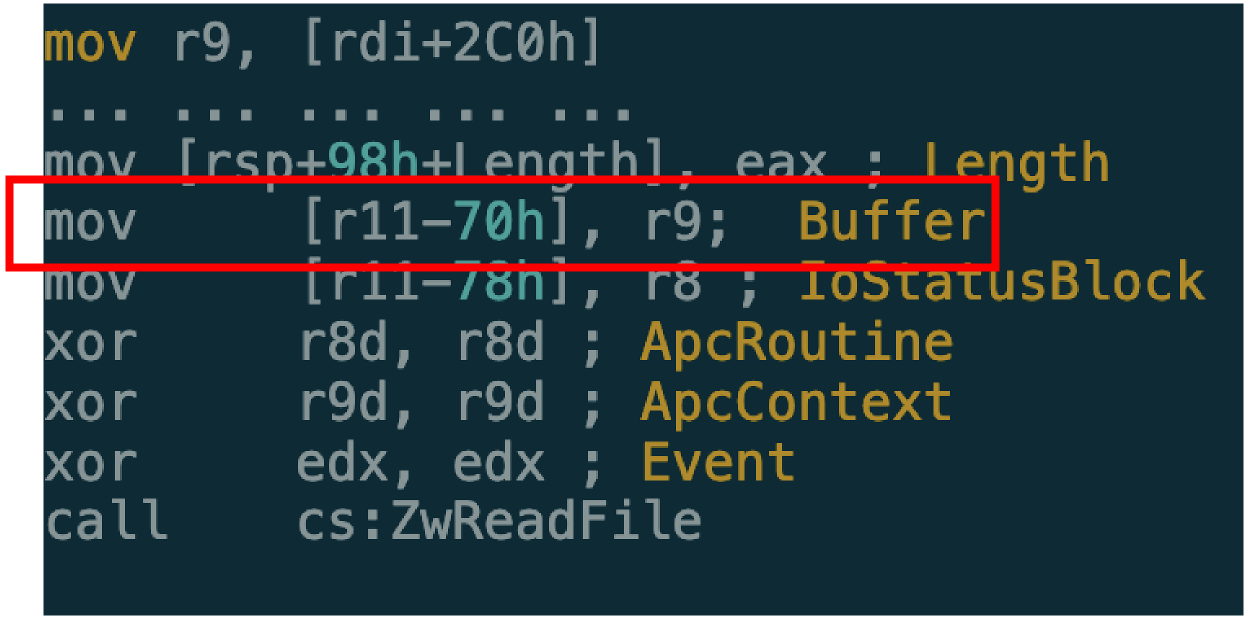
```
mov     eax, [rdi+0Fh]
lea     edx, [rax+rax]
call    cs:ExAllocatePoolWithTag
mov     r8d, r14d
mov     ecx, r15d
mov     [rbx+2C0h], rax
```



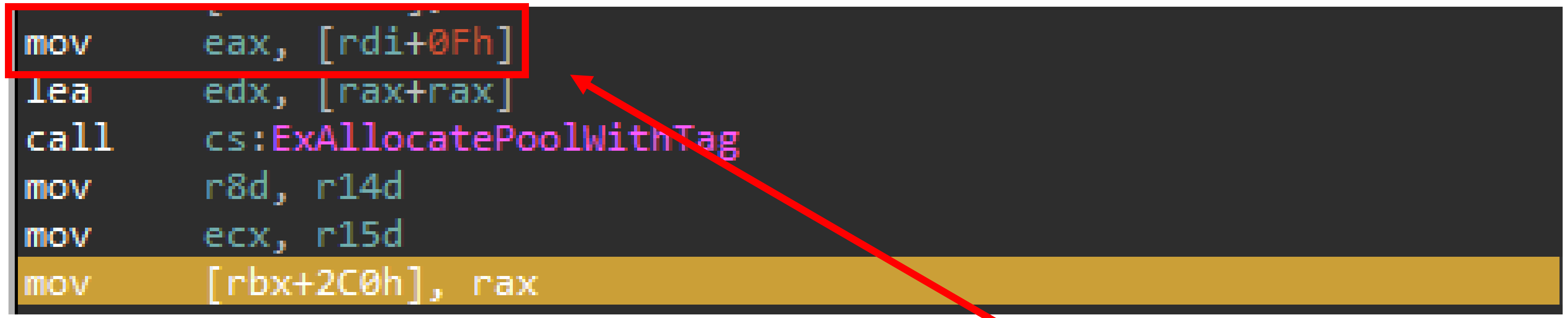
```
mov r9, [rdi+2C0h]
... ... ... ... ...
mov [rsp+98h+Length], eax ; Length
mov     [r11-70h], r9;  Buffer
mov     [r11-78h], r8 ; IoStatusBlock
xor     r8d, r8d ; ApcRoutine
xor     r9d, r9d ; ApcContext
xor     edx, edx ; Event
call    cs:ZwReadFile
```



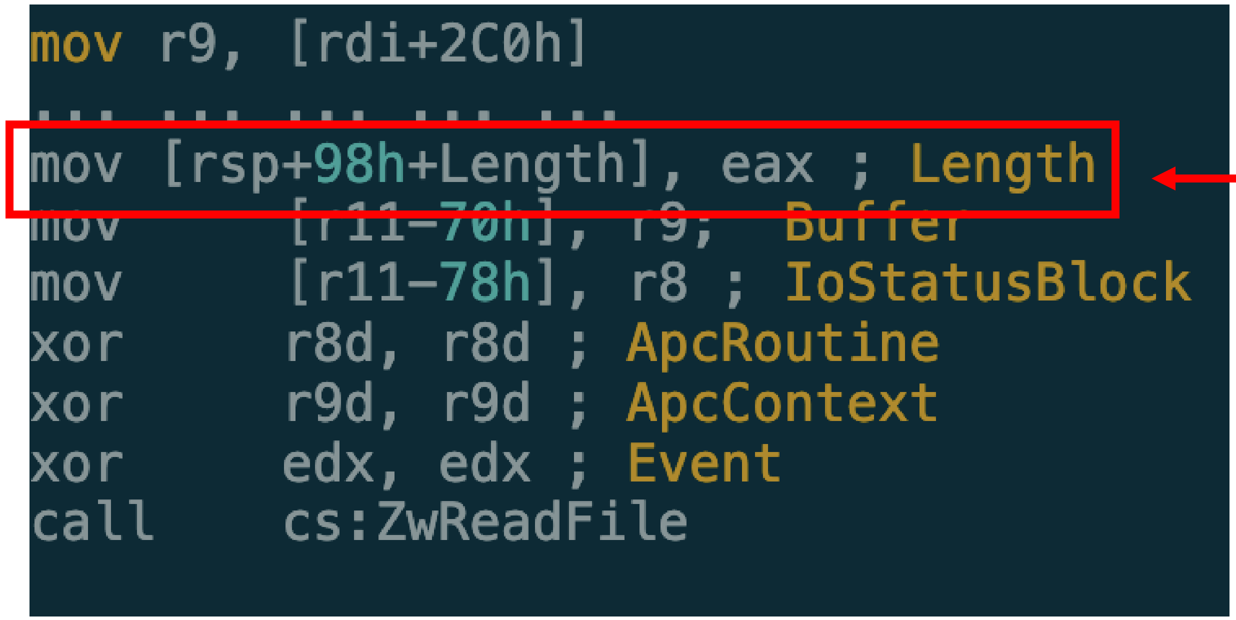
```
mov     eax, [rdi+0Fh]
lea     edx, [rax+rax]
call    cs:ExAllocatePoolWithTag
mov     r8d, r14d
mov     ecx, r15d
mov     [rbx+2C0h], rax
```



```
mov r9, [rdi+2C0h]
... ... ... ... ...
mov [rsp+98h+Length], eax ; Length
mov      [r11-70h], r9;  Buffer
mov      [r11-78h], r8 ; IoStatusBlock
xor      r8d, r8d ; ApcRoutine
xor      r9d, r9d ; ApcContext
xor      edx, edx ; Event
call     cs:ZwReadFile
```



```
mov     eax, [rdi+0Fh]
lea     edx, [rax+rax]
call    cs:ExAllocatePoolWithTag
mov     r8d, r14d
mov     ecx, r15d
mov     [rbx+2C0h], rax
```

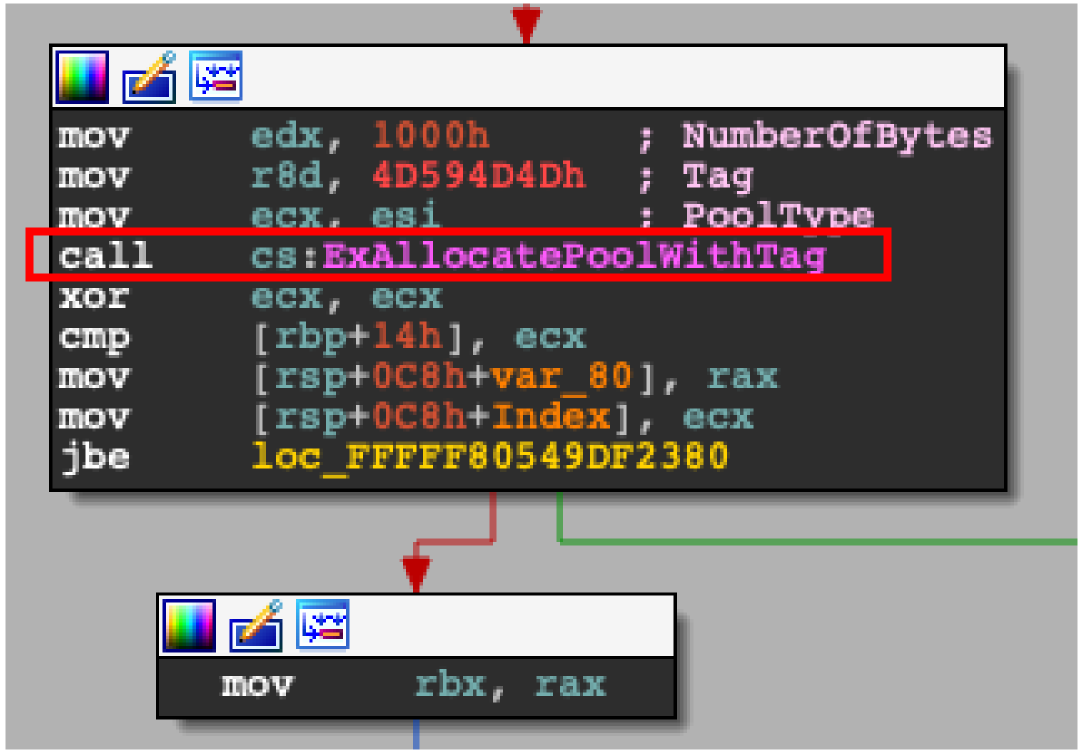


```
mov r9, [rdi+2C0h]

mov [rsp+98h+Length], eax ; Length
mov       [r11-70h], r9;  Buffer
mov       [r11-78h], r8 ; IoStatusBlock
xor       r8d, r8d ; ApcRoutine
xor       r9d, r9d ; ApcContext
xor       edx, edx ; Event
call      cs:ZwReadFile
```

Unrelated

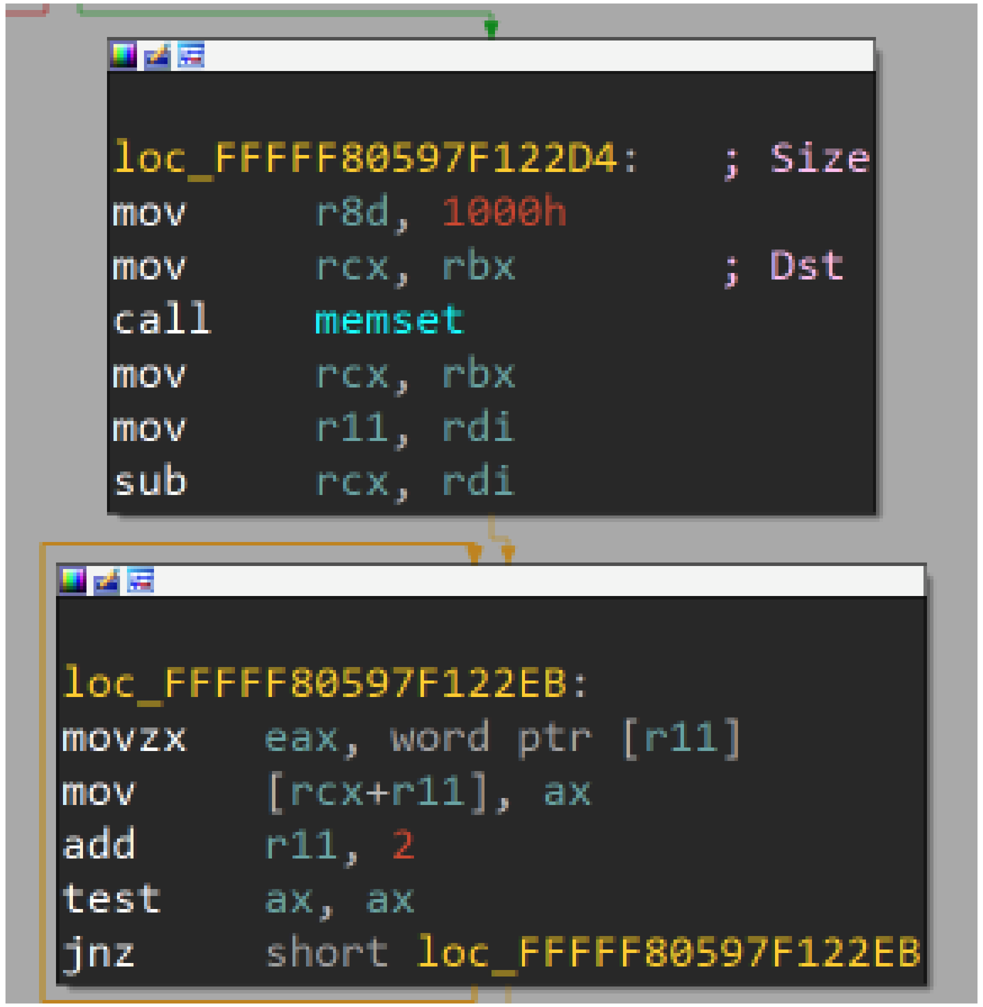
Buffer overflow

- Registry key



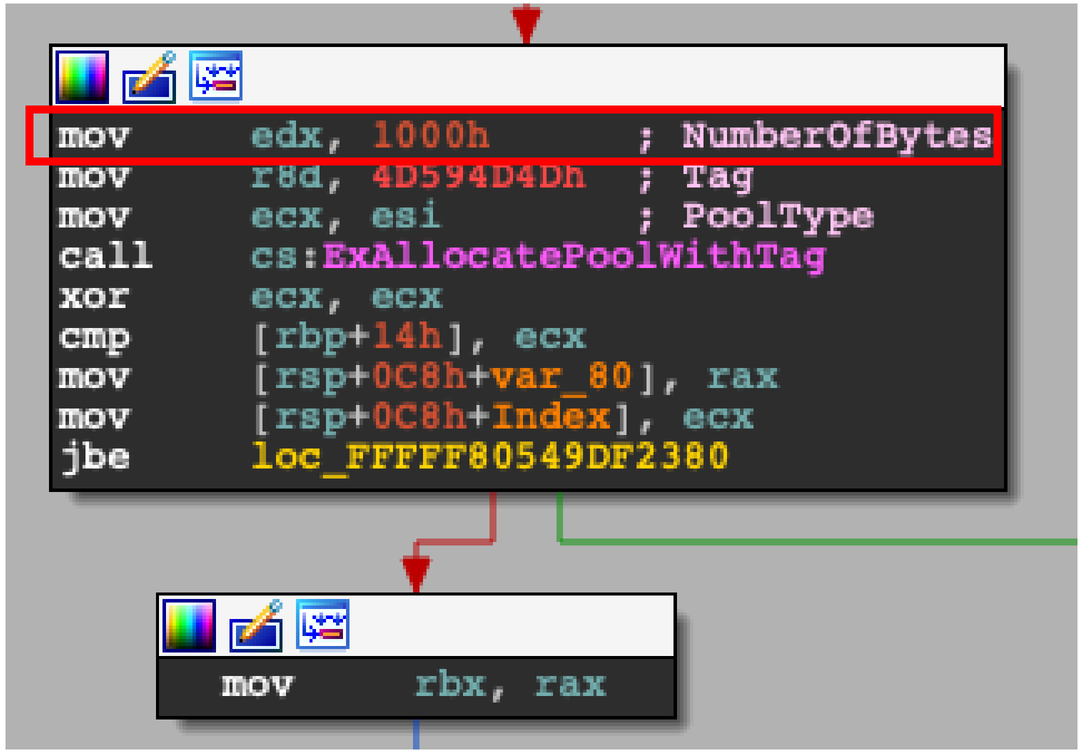
```
mov     edx, 1000h          ; NumberOfBytes
mov     r8d, 4D594D4Dh      ; Tag
mov     ecx, esi            ; PoolType
call    cs:ExAllocatePoolWithTag
xor     ecx, ecx
cmp     [rbp+14h], ecx
mov     [rsp+0C8h+var_80], rax
mov     [rsp+0C8h+Index], ecx
jbe     loc_FFFFF80549DF2380
```

```
mov     rbx, rax
```



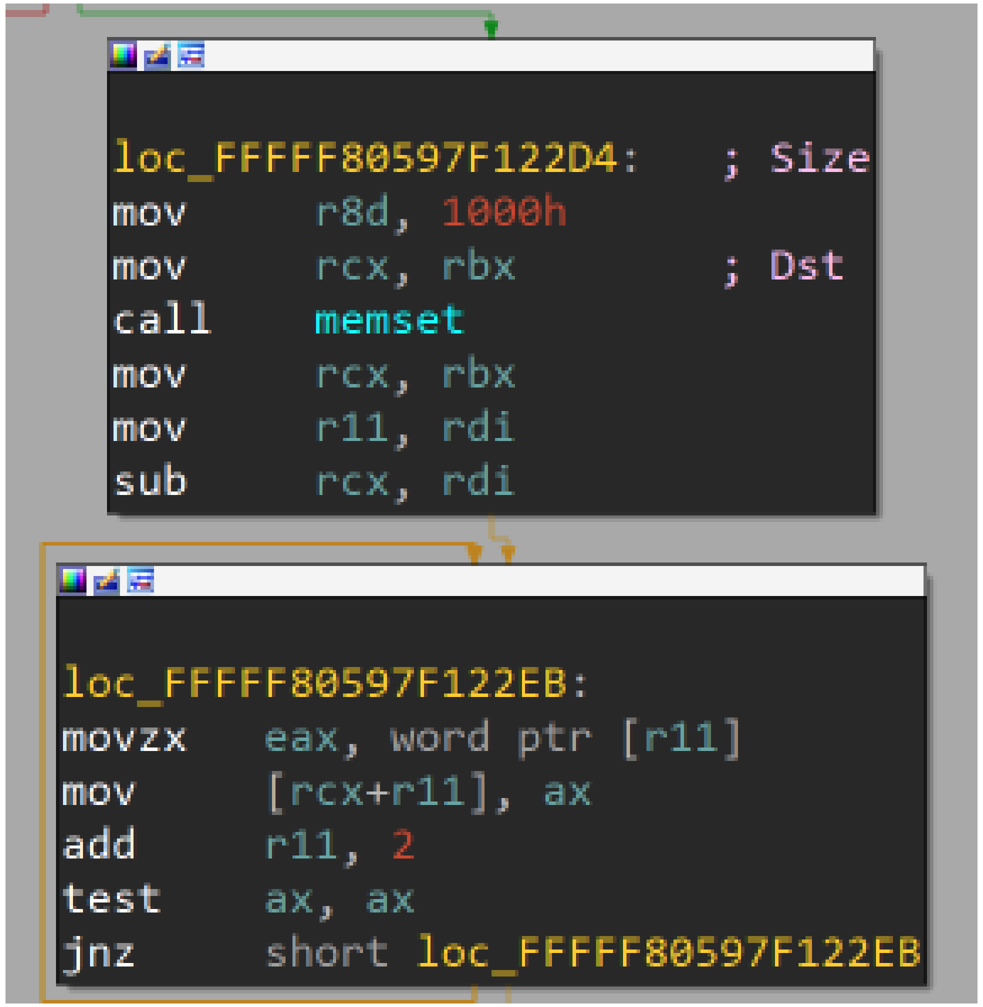
```
loc_FFFFF80597F122D4:       ; Size
mov     r8d, 1000h
mov     rcx, rbx            ; Dst
call    memset
mov     rcx, rbx
mov     r11, rdi
sub     rcx, rdi
```

```
loc_FFFFF80597F122EB:
movzx   eax, word ptr [r11]
mov     [rcx+r11], ax
add     r11, 2
test    ax, ax
jnz     short loc_FFFFF80597F122EB
```



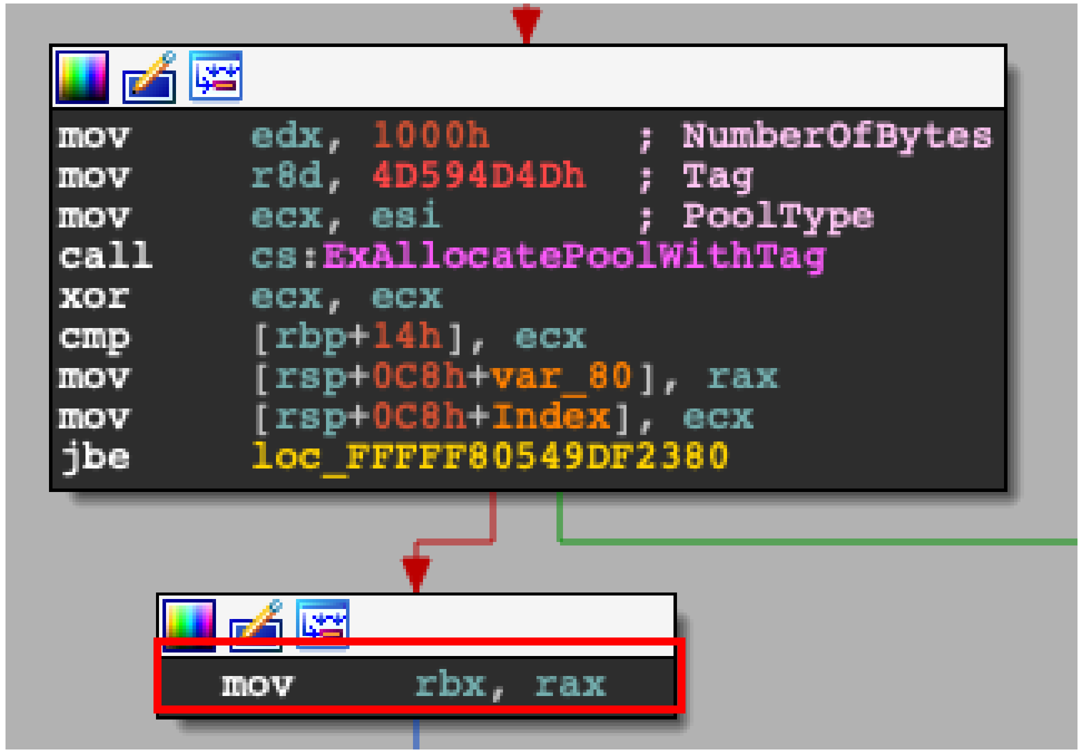
```
mov     edx, 1000h          ; NumberOfBytes
mov     r8d, 4D594D4Dh      ; Tag
mov     ecx, esi            ; PoolType
call    cs:ExAllocatePoolWithTag
xor     ecx, ecx
cmp     [rbp+14h], ecx
mov     [rsp+0C8h+var_80], rax
mov     [rsp+0C8h+Index], ecx
jbe     loc_FFFFF80549DF2380
```

```
mov     rbx, rax
```



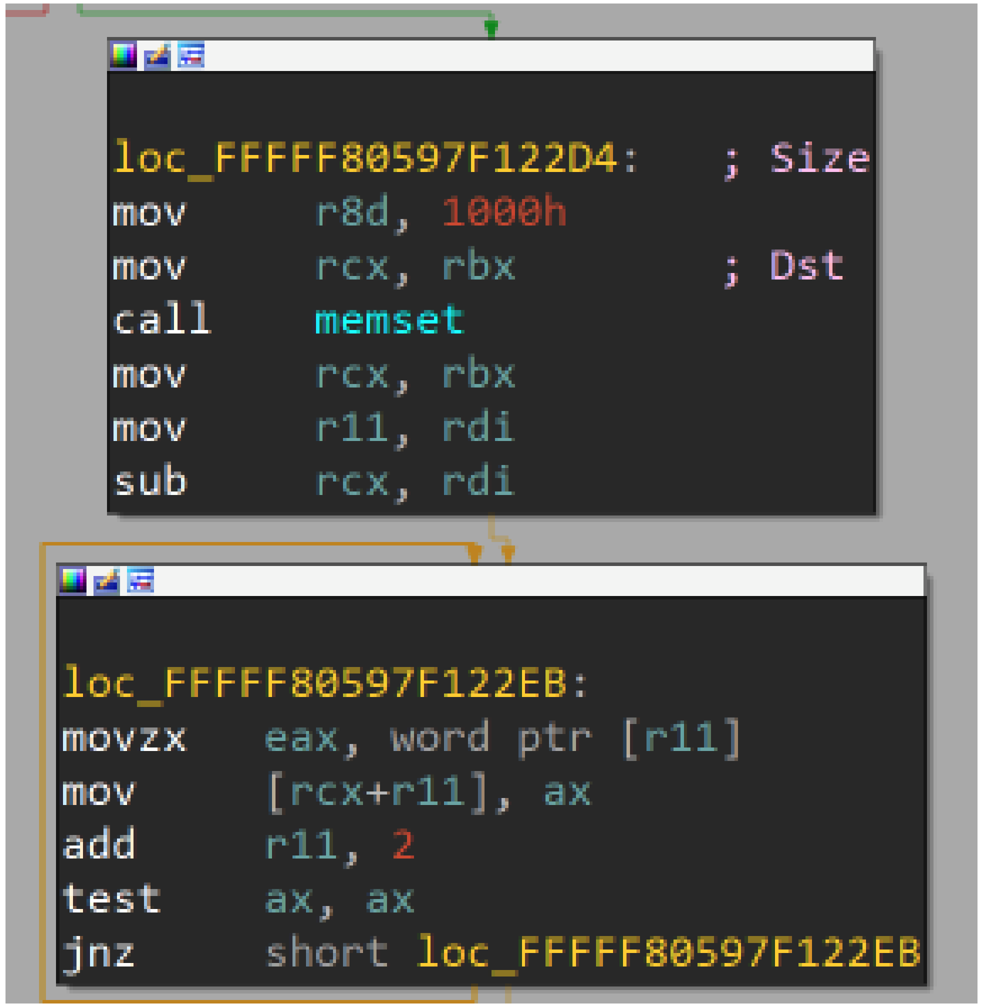
```
loc_FFFFF80597F122D4:    ; Size
mov     r8d, 1000h
mov     rcx, rbx         ; Dst
call    memset
mov     rcx, rbx
mov     r11, rdi
sub     rcx, rdi
```

```
loc_FFFFF80597F122EB:
movzx   eax, word ptr [r11]
mov     [rcx+r11], ax
add     r11, 2
test    ax, ax
jnz     short loc_FFFFF80597F122EB
```



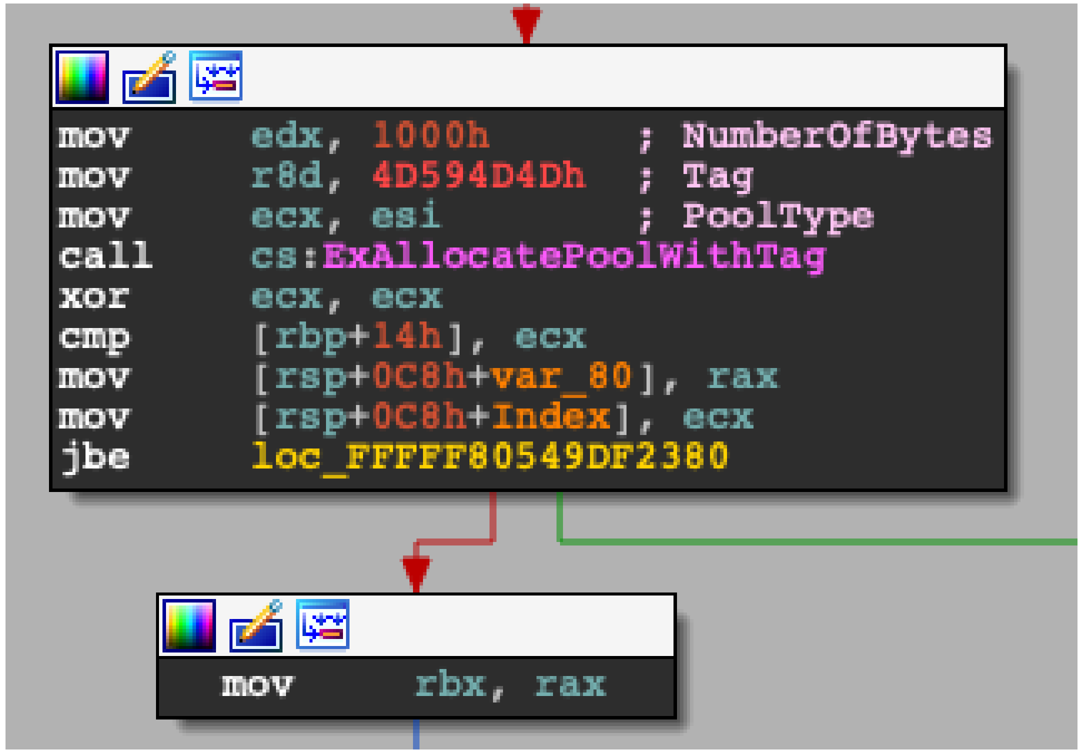
```
mov     edx, 1000h          ; NumberOfBytes
mov     r8d, 4D594D4Dh      ; Tag
mov     ecx, esi            ; PoolType
call    cs:ExAllocatePoolWithTag
xor     ecx, ecx
cmp     [rbp+14h], ecx
mov     [rsp+0C8h+var_80], rax
mov     [rsp+0C8h+Index], ecx
jbe     loc_FFFFF80549DF2380
```

```
mov     rbx, rax
```



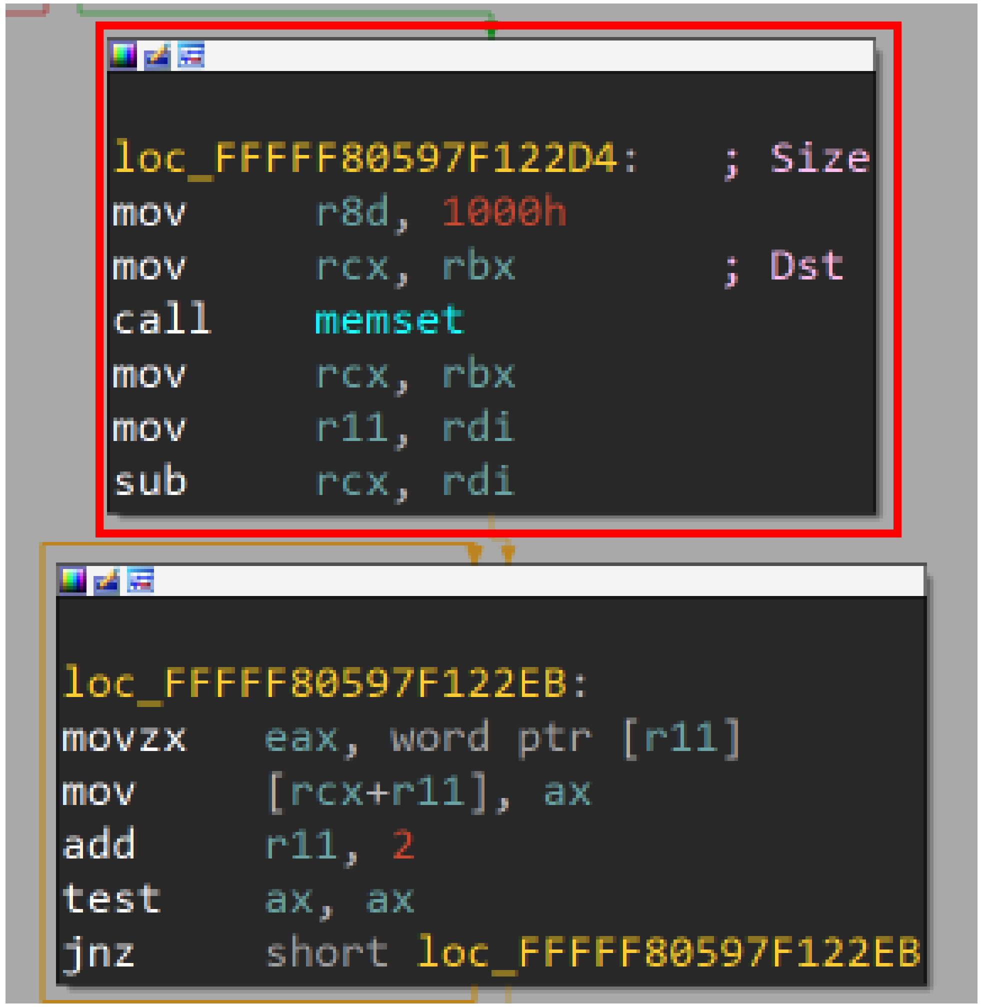
```
loc_FFFFF80597F122D4:   ; Size
mov     r8d, 1000h
mov     rcx, rbx        ; Dst
call    memset
mov     rcx, rbx
mov     r11, rdi
sub     rcx, rdi
```

```
loc_FFFFF80597F122EB:
movzx   eax, word ptr [r11]
mov     [rcx+r11], ax
add     r11, 2
test    ax, ax
jnz     short loc_FFFFF80597F122EB
```



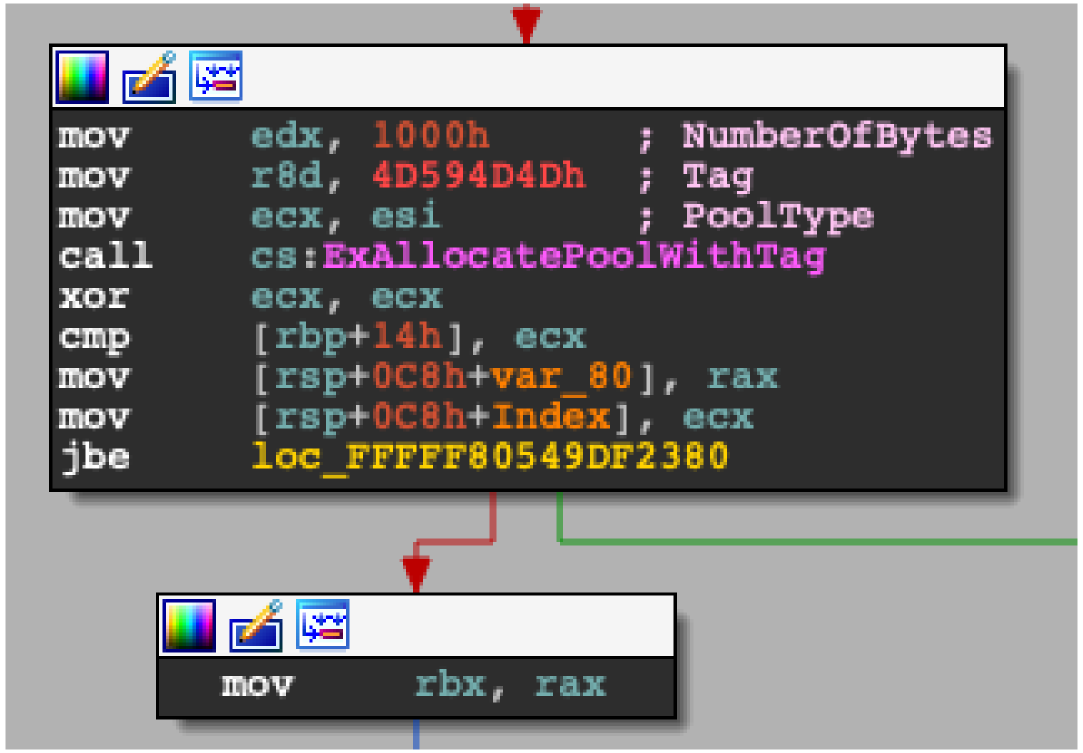
```
mov     edx, 1000h        ; NumberOfBytes
mov     r8d, 4D594D4Dh    ; Tag
mov     ecx, esi          ; PoolType
call    cs:ExAllocatePoolWithTag
xor     ecx, ecx
cmp     [rbp+14h], ecx
mov     [rsp+0C8h+var_80], rax
mov     [rsp+0C8h+Index], ecx
jbe     loc_FFFFF80549DF2380
```

```
mov     rbx, rax
```



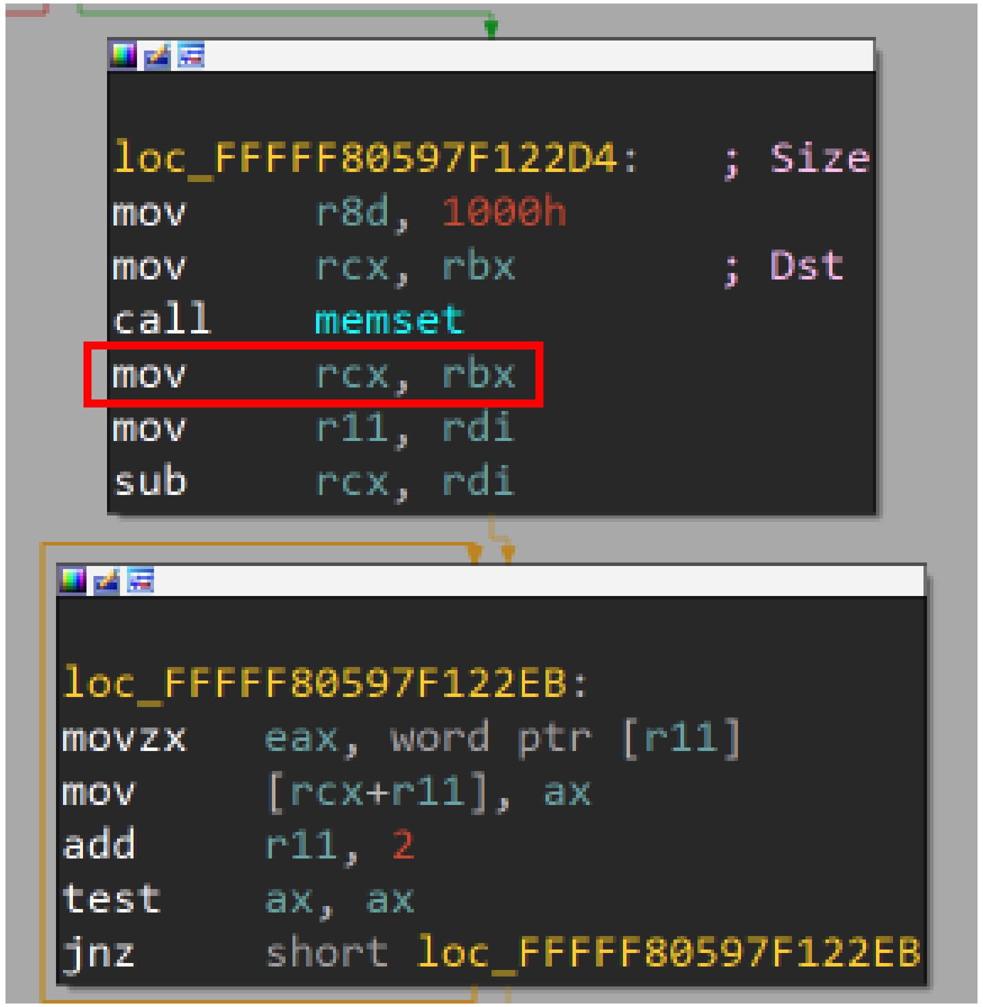
```
loc_FFFFF80597F122D4:    ; Size
mov     r8d, 1000h
mov     rcx, rbx         ; Dst
call    memset
mov     rcx, rbx
mov     r11, rdi
sub     rcx, rdi
```

```
loc_FFFFF80597F122EB:
movzx   eax, word ptr [r11]
mov     [rcx+r11], ax
add     r11, 2
test    ax, ax
jnz     short loc_FFFFF80597F122EB
```



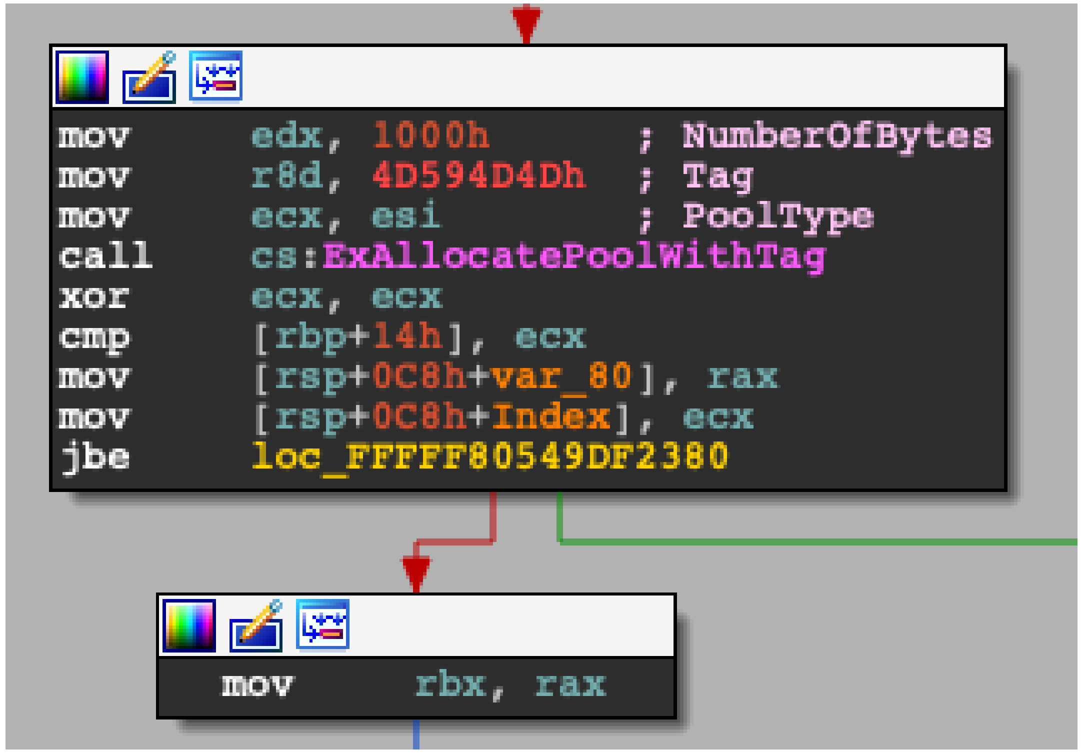
```
mov     edx, 1000h          ; NumberOfBytes
mov     r8d, 4D594D4Dh      ; Tag
mov     ecx, esi            ; PoolType
call    cs:ExAllocatePoolWithTag
xor     ecx, ecx
cmp     [rbp+14h], ecx
mov     [rsp+0C8h+var_80], rax
mov     [rsp+0C8h+Index], ecx
jbe     loc_FFFFF80549DF2380
```

```
    mov     rbx, rax
```



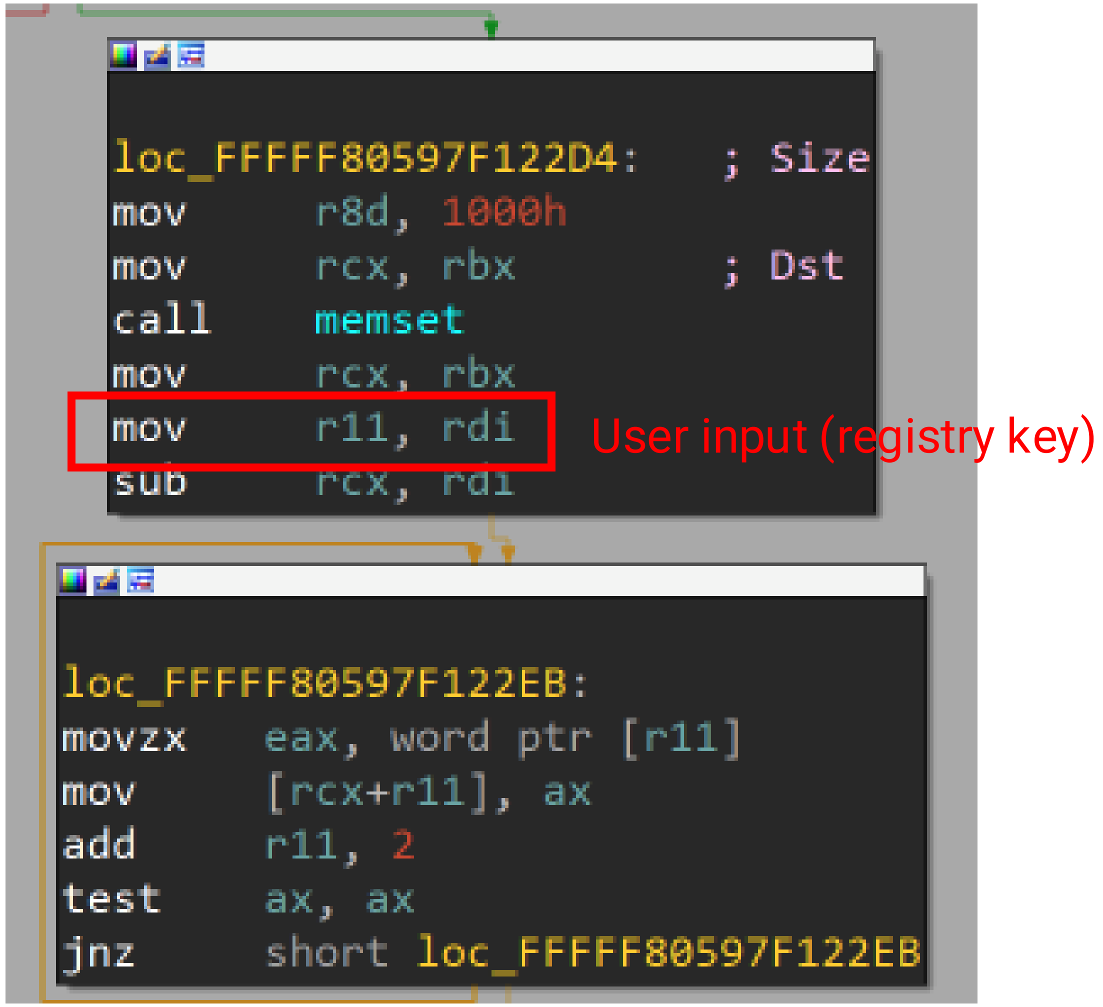
```
loc_FFFFF80597F122D4:     ; Size
mov     r8d, 1000h
mov     rcx, rbx          ; Dst
call    memset
mov     rcx, rbx
mov     r11, rdi
sub     rcx, rdi
```

```
loc_FFFFF80597F122EB:
movzx   eax, word ptr [r11]
mov     [rcx+r11], ax
add     r11, 2
test    ax, ax
jnz     short loc_FFFFF80597F122EB
```



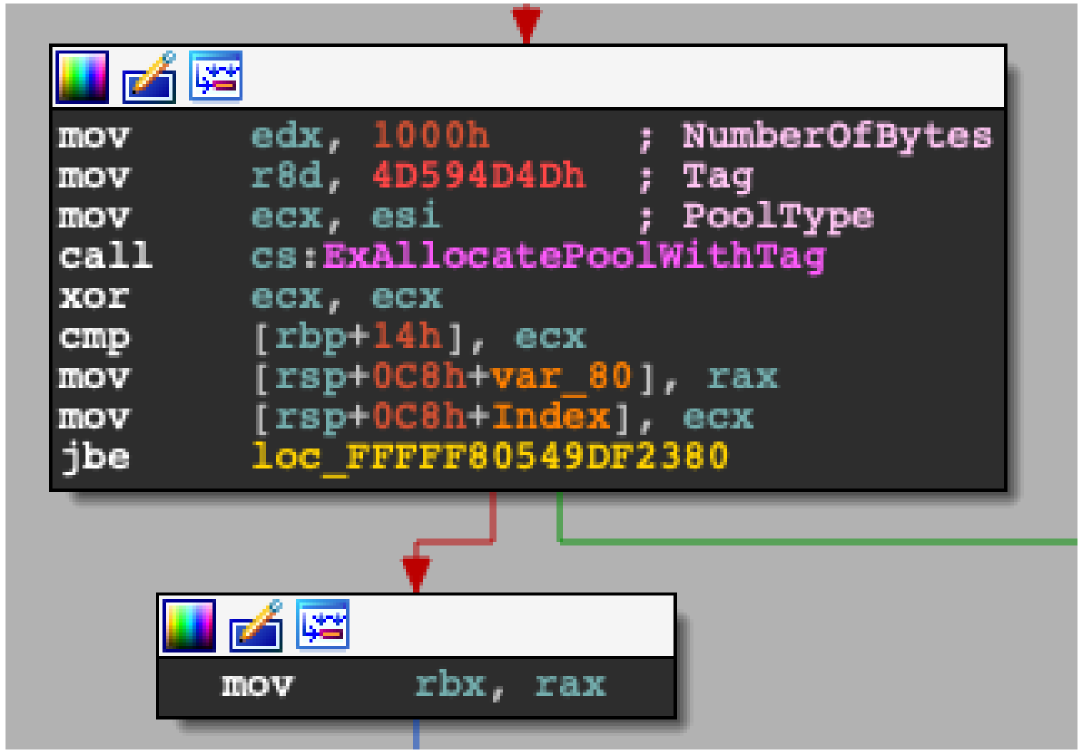
```
mov     edx, 1000h          ; NumberOfBytes
mov     r8d, 4D594D4Dh      ; Tag
mov     ecx, esi            ; PoolType
call    cs:ExAllocatePoolWithTag
xor     ecx, ecx
cmp     [rbp+14h], ecx
mov     [rsp+0C8h+var_80], rax
mov     [rsp+0C8h+Index], ecx
jbe     loc_FFFFF80549DF2380
```

```
mov     rbx, rax
```

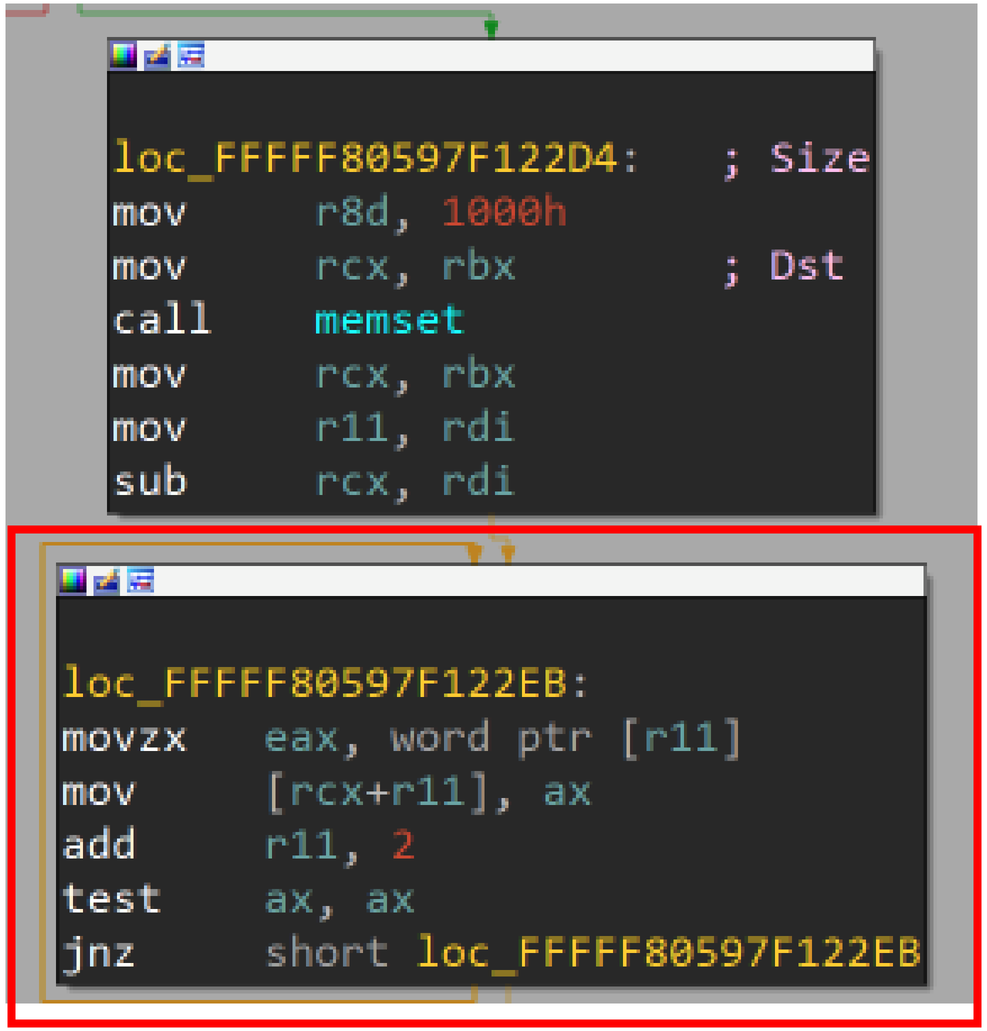


```
loc_FFFFF80597F122D4:    ; Size
mov     r8d, 1000h
mov     rcx, rbx         ; Dst
call    memset
mov     rcx, rbx
mov     r11, rdi
sub     rcx, rdi
```

```
loc_FFFFF80597F122EB:
movzx   eax, word ptr [r11]
mov     [rcx+r11], ax
add     r11, 2
test    ax, ax
jnz     short loc_FFFFF80597F122EB
```


```
mov     edx, 1000h          ; NumberOfBytes
mov     r8d, 4D594D4Dh      ; Tag
mov     ecx, esi            ; PoolType
call    cs:ExAllocatePoolWithTag
xor     ecx, ecx
cmp     [rbp+14h], ecx
mov     [rsp+0C8h+var_80], rax
mov     [rsp+0C8h+Index], ecx
jbe     loc_FFFFF80549DF2380
```

```
mov     rbx, rax
```


```
loc_FFFFF80597F122D4:       ; Size
mov     r8d, 1000h
mov     rcx, rbx            ; Dst
call    memset
mov     rcx, rbx
mov     r11, rdi
sub     rcx, rdi
```

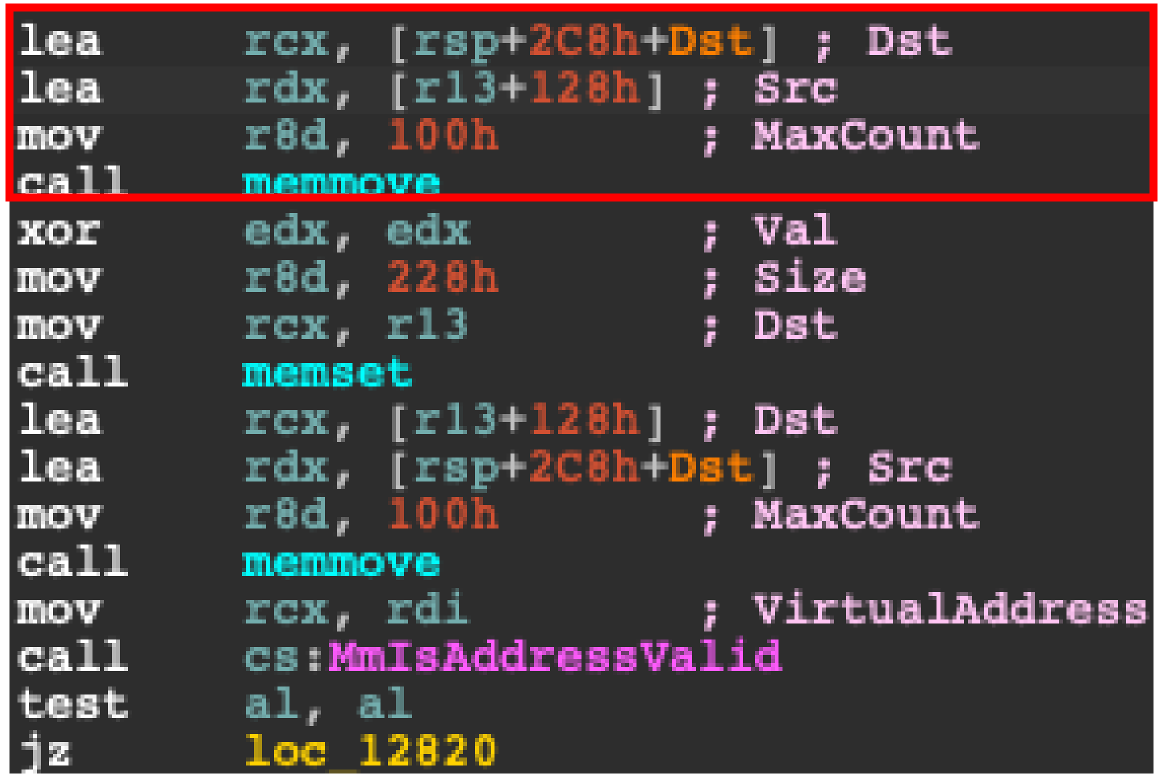
```
loc_FFFFF80597F122EB:
movzx   eax, word ptr [r11]
mov     [rcx+r11], ax
add     r11, 2
test    ax, ax
jnz     short loc_FFFFF80597F122EB
```

Copy registry key into buffer

Data leak

- Missing bounds check on input buffer
- Memory after the input buffer is copied to the output buffer

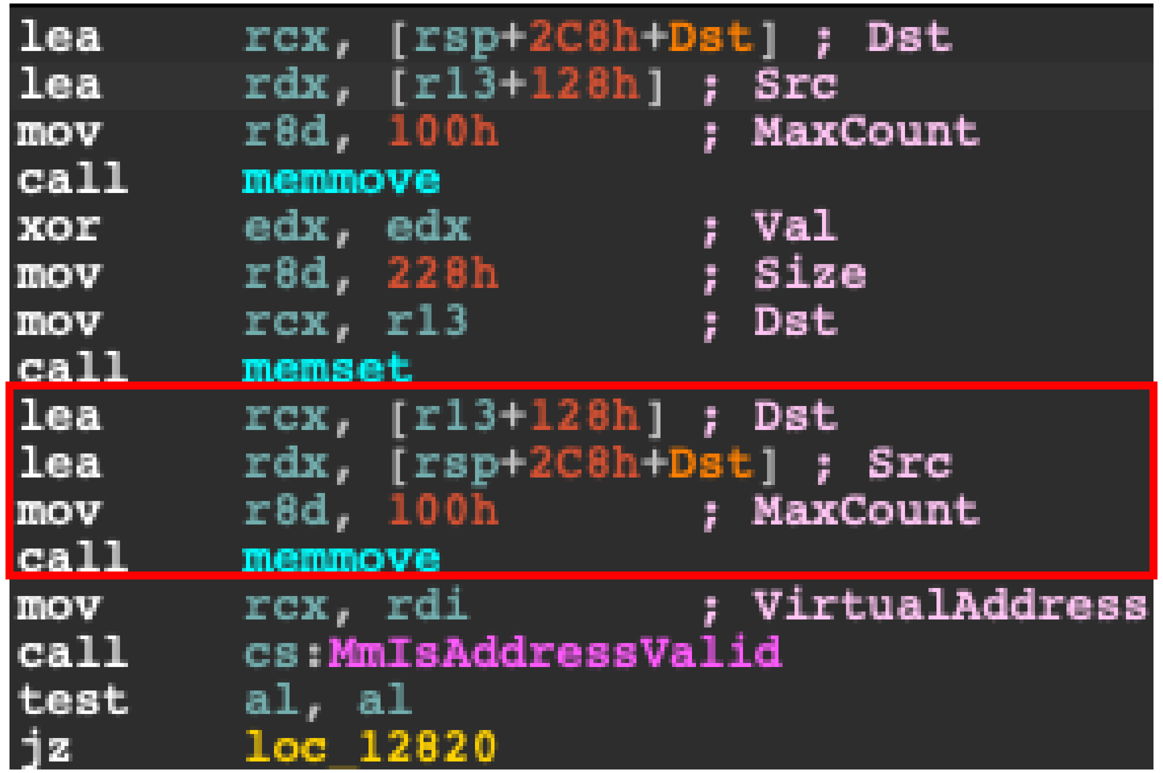
CoreTemp



```
lea     rcx, [rsp+2C8h+Dst] ; Dst
lea     rdx, [r13+128h] ; Src
mov     r8d, 100h       ; MaxCount
call    memmove
xor     edx, edx        ; Val
mov     r8d, 228h       ; Size
mov     rcx, r13        ; Dst
call    memset
lea     rcx, [r13+128h] ; Dst
lea     rdx, [rsp+2C8h+Dst] ; Src
mov     r8d, 100h       ; MaxCount
call    memmove
mov     rcx, rdi        ; VirtualAddress
call    cs:MmIsAddressValid
test    al, al
jz      loc_12820
```

Copy to buffer

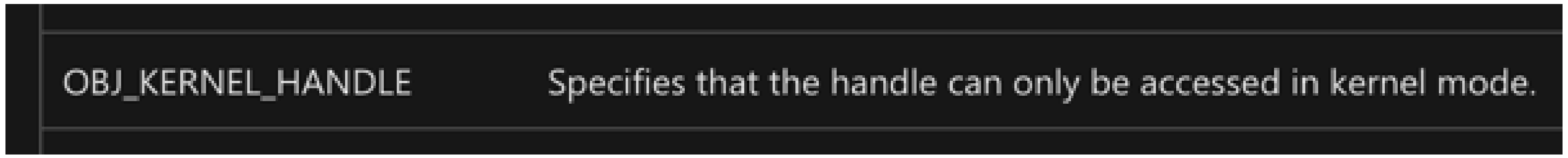
CoreTemp



```
lea     rcx, [rsp+2C8h+Dst] ; Dst
lea     rdx, [r13+128h] ; Src
mov     r8d, 100h       ; MaxCount
call    memmove
xor     edx, edx        ; Val
mov     r8d, 228h       ; Size
mov     rcx, r13        ; Dst
call    memset
lea     rcx, [r13+128h] ; Dst
lea     rdx, [rsp+2C8h+Dst] ; Src
mov     r8d, 100h       ; MaxCount
call    memmove
mov     rcx, rdi        ; VirtualAddress
call    cs:MmIsAddressValid
test    al, al
jz      loc_12820
```

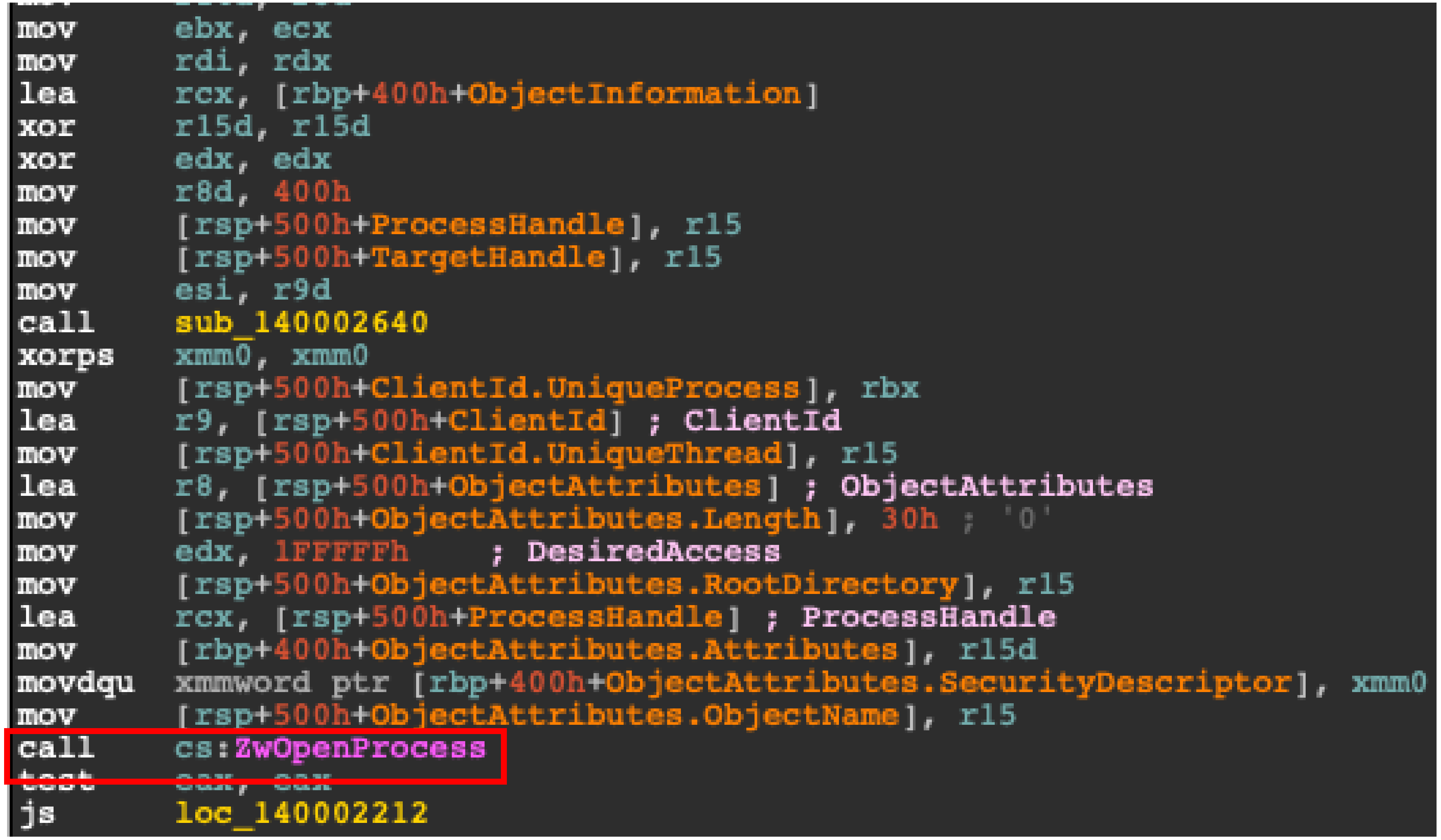
Copy from buffer

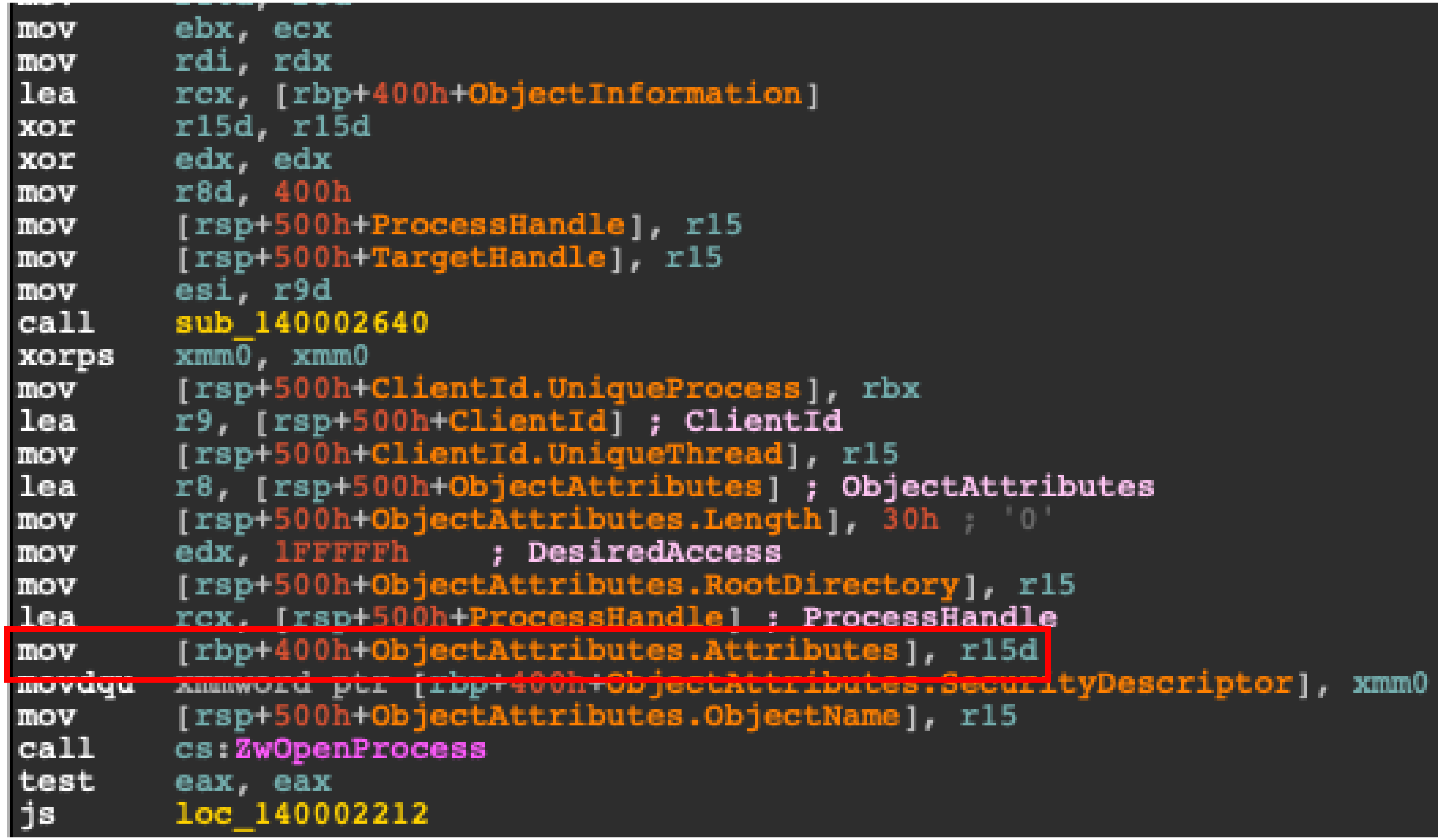
Handle leaks



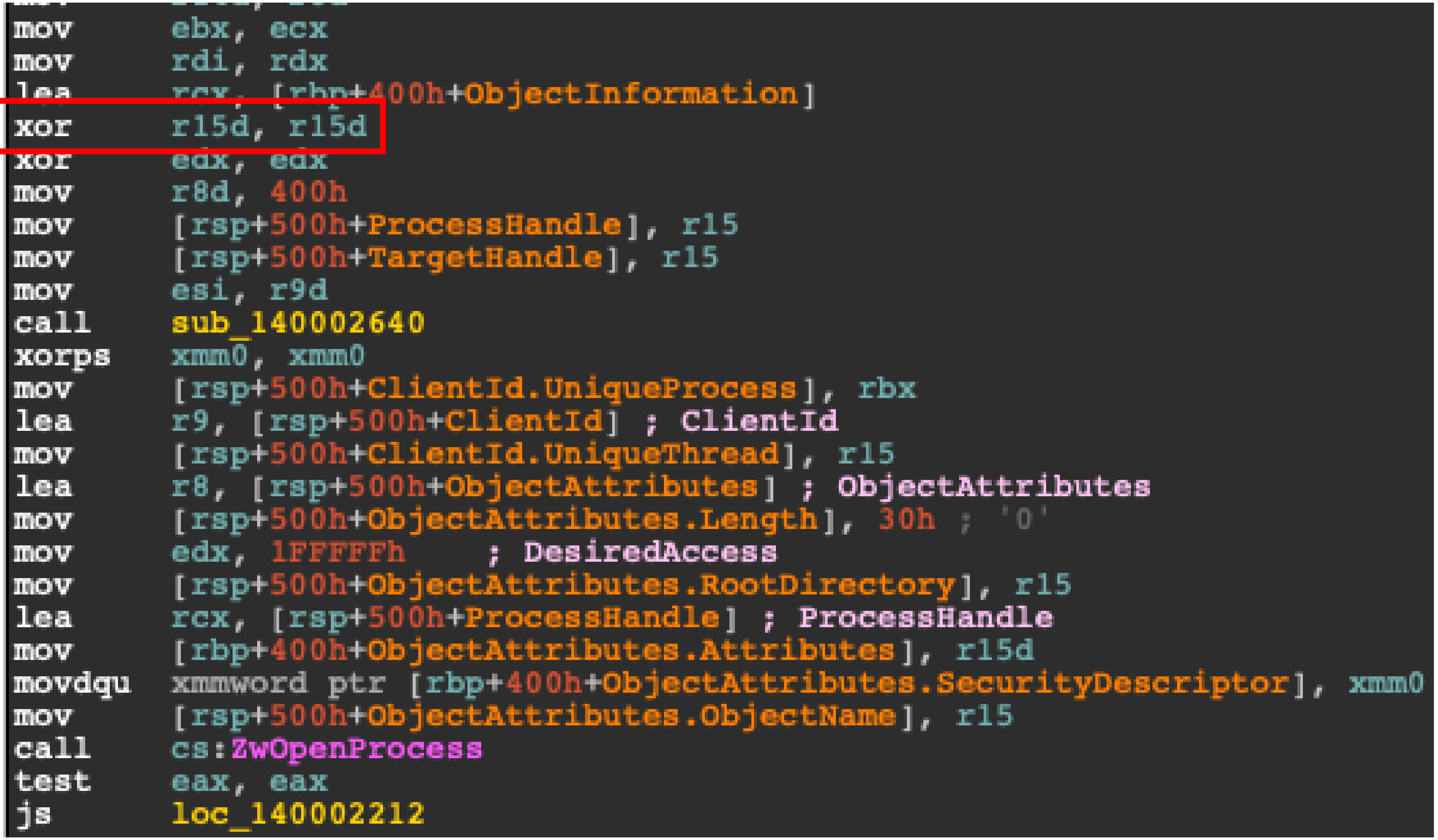
OBJ_KERNEL_HANDLE   Specifies that the handle can only be accessed in kernel mode.

- Handle appears in user-mode handle table
- Can be exploited through a race condition


```
mov     ebx, ecx
mov     rdi, rdx
lea     rcx, [rbp+400h+ObjectInformation]
xor     r15d, r15d
xor     edx, edx
mov     r8d, 400h
mov     [rsp+500h+ProcessHandle], r15
mov     [rsp+500h+TargetHandle], r15
mov     esi, r9d
call    sub_140002640
xorps   xmm0, xmm0
mov     [rsp+500h+ClientId.UniqueProcess], rbx
lea     r9, [rsp+500h+ClientId] ; ClientId
mov     [rsp+500h+ClientId.UniqueThread], r15
lea     r8, [rsp+500h+ObjectAttributes] ; ObjectAttributes
mov     [rsp+500h+ObjectAttributes.Length], 30h ; '0'
mov     edx, 1FFFFFh     ; DesiredAccess
mov     [rsp+500h+ObjectAttributes.RootDirectory], r15
lea     rcx, [rsp+500h+ProcessHandle] ; ProcessHandle
mov     [rbp+400h+ObjectAttributes.Attributes], r15d
movdqu  xmmword ptr [rbp+400h+ObjectAttributes.SecurityDescriptor], xmm0
mov     [rsp+500h+ObjectAttributes.ObjectName], r15
call    cs:ZwOpenProcess
test    eax, eax
js      loc_140002212
```



```
mov     ebx, ecx
mov     rdi, rdx
lea     rcx, [rbp+400h+ObjectInformation]
xor     r15d, r15d
xor     edx, edx
mov     r8d, 400h
mov     [rsp+500h+ProcessHandle], r15
mov     [rsp+500h+TargetHandle], r15
mov     esi, r9d
call    sub_140002640
xorps   xmm0, xmm0
mov     [rsp+500h+ClientId.UniqueProcess], rbx
lea     r9, [rsp+500h+ClientId] ; ClientId
mov     [rsp+500h+ClientId.UniqueThread], r15
lea     r8, [rsp+500h+ObjectAttributes] ; ObjectAttributes
mov     [rsp+500h+ObjectAttributes.Length], 30h ; '0'
mov     edx, 1FFFFFh    ; DesiredAccess
mov     [rsp+500h+ObjectAttributes.RootDirectory], r15
lea     rcx, [rsp+500h+ProcessHandle] ; ProcessHandle
mov     [rbp+400h+ObjectAttributes.Attributes], r15d
movdqu  xmmword ptr [rbp+400h+ObjectAttributes.SecurityDescriptor], xmm0
mov     [rsp+500h+ObjectAttributes.ObjectName], r15
call    cs:ZwOpenProcess
test    eax, eax
js      loc_140002212
```

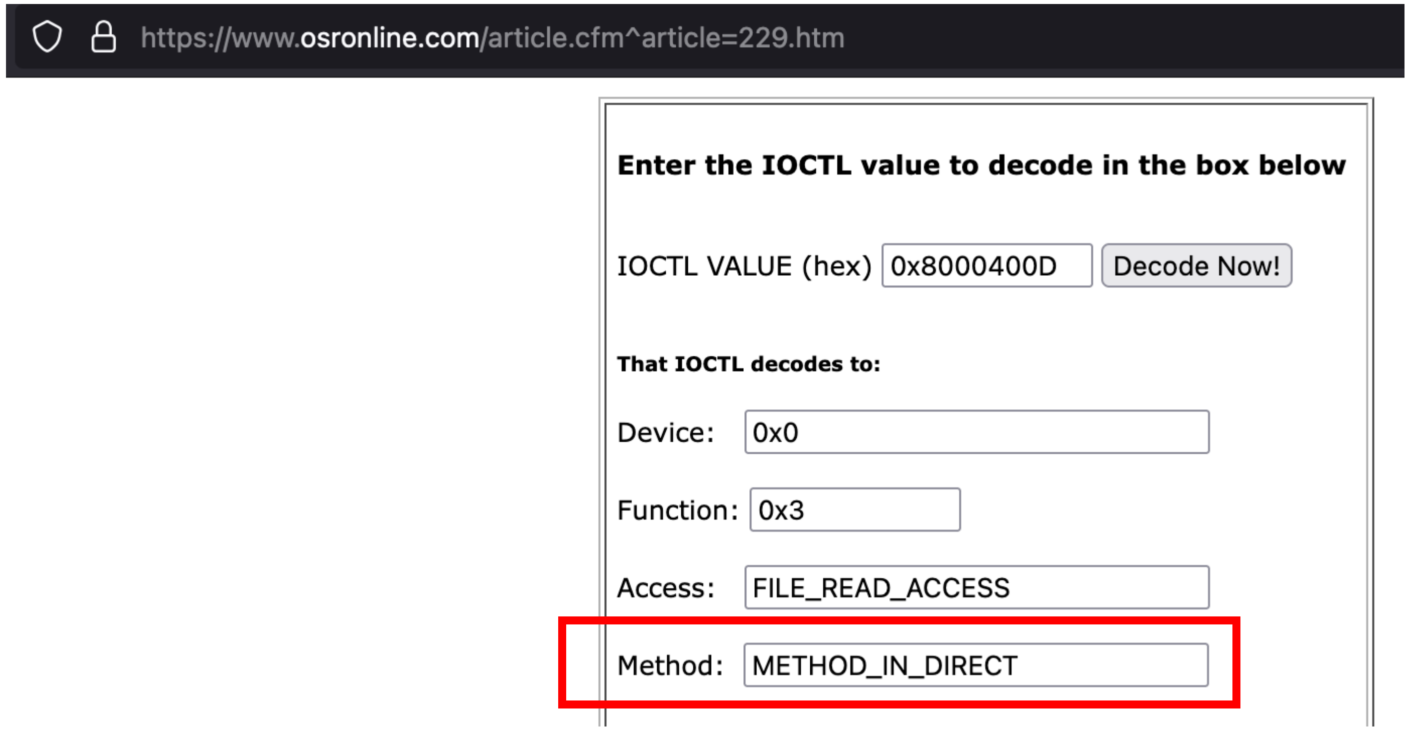


```
mov     ebx, ecx
mov     rdi, rdx
lea     rcx, [rbp+400h+ObjectInformation]
xor     r15d, r15d
xor     edx, edx
mov     r8d, 400h
mov     [rsp+500h+ProcessHandle], r15
mov     [rsp+500h+TargetHandle], r15
mov     esi, r9d
call    sub_140002640
xorps   xmm0, xmm0
mov     [rsp+500h+ClientId.UniqueProcess], rbx
lea     r9, [rsp+500h+ClientId] ; ClientId
mov     [rsp+500h+ClientId.UniqueThread], r15
lea     r8, [rsp+500h+ObjectAttributes] ; ObjectAttributes
mov     [rsp+500h+ObjectAttributes.Length], 30h ; '0'
mov     edx, 1FFFFFh    ; DesiredAccess
mov     [rsp+500h+ObjectAttributes.RootDirectory], r15
lea     rcx, [rsp+500h+ProcessHandle] ; ProcessHandle
mov     [rbp+400h+ObjectAttributes.Attributes], r15d
movdqu  xmmword ptr [rbp+400h+ObjectAttributes.SecurityDescriptor], xmm0
mov     [rsp+500h+ObjectAttributes.ObjectName], r15
call    cs:ZwOpenProcess
test    eax, eax
js      loc_140002212
```

Method direct

- METHOD_IN_DIRECT or METHOD_OUT_DIRECT
- User can write at same time as driver reads
- Value can be changed between load and read
- Check if locking is implemented to prevent this

Method direct



https://www.osronline.com/article.cfm^article=229.htm

Enter the IOCTL value to decode in the box below

IOCTL VALUE (hex) 0x8000400D Decode Now!

That IOCTL decodes to:

Device: 0x0

Function: 0x3

Access: FILE_READ_ACCESS

Method: METHOD_IN_DIRECT

Automated Tools

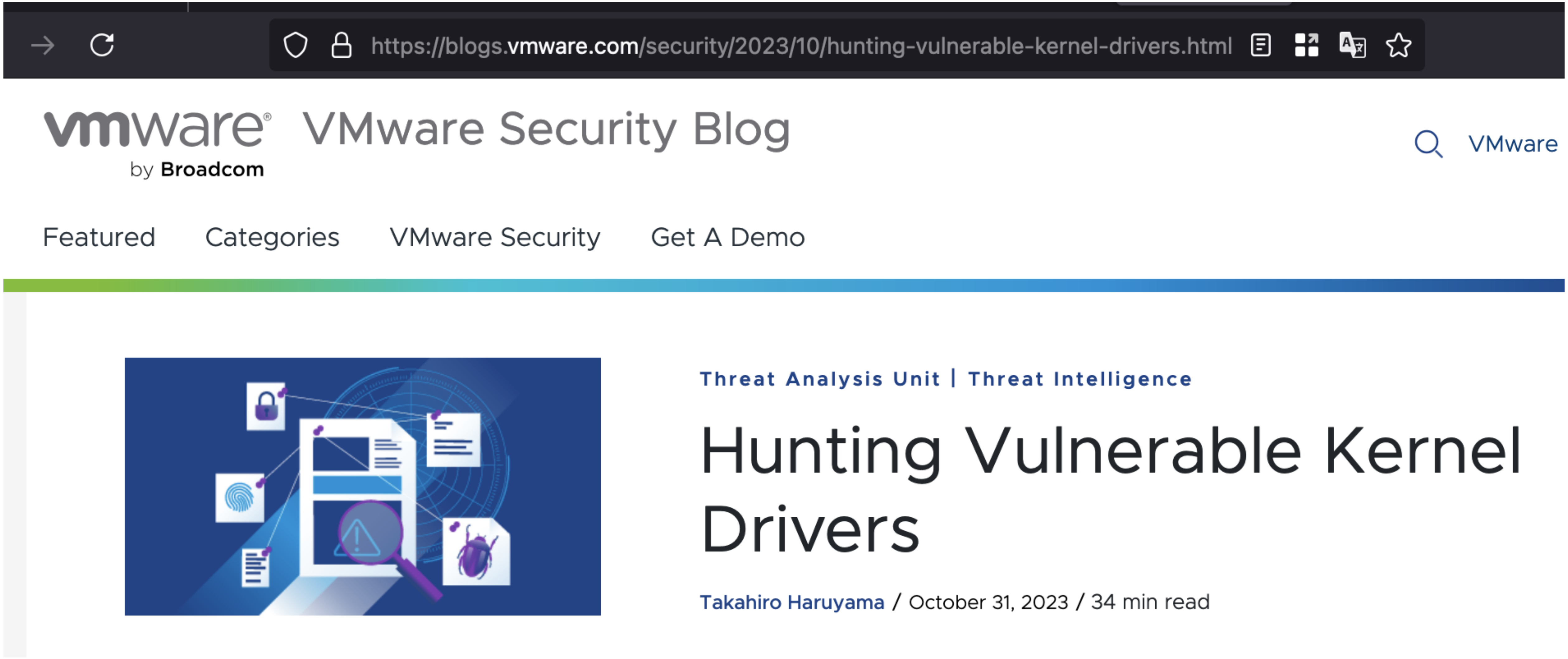
POPKORN: Popping Windows Kernel Drivers At Scale

Anonymous Author(s)

ABSTRACT

External vendors develop a significant percentage of Windows kernel drivers, and Microsoft relies on these vendors to handle all aspects of driver security. Unfortunately, device vendors are not immune to software bugs, which in some cases can be exploited to gain elevated privileges. Testing the security of kernel drivers remains challenging:

called BYOB ("bring your own bug") makes it possible to load an unsigned driver into the kernel by piggybacking on a signed-but-vulnerable driver [56]. For instance, the LoJax and Slingshot malware families [19, 60] ship a signed-but-vulnerable driver with the malware itself, which allows for loading of the malware into the kernel. This problem also negatively affects game vendors, as players can



https://blogs.vmware.com/security/2023/10/hunting-vulnerable-kernel-drivers.html

vmware by Broadcom VMware Security Blog

VMware

Featured Categories VMware Security Get A Demo

Threat Analysis Unit | Threat Intelligence

Hunting Vulnerable Kernel Drivers

Takahiro Haruyama / October 31, 2023 / 34 min read

Conclusion

Conclusion

- Kernel driver vulnerabilities have impact
- Research is important
- There are many vulnerabilities to be found
- You can contribute!

Thank You