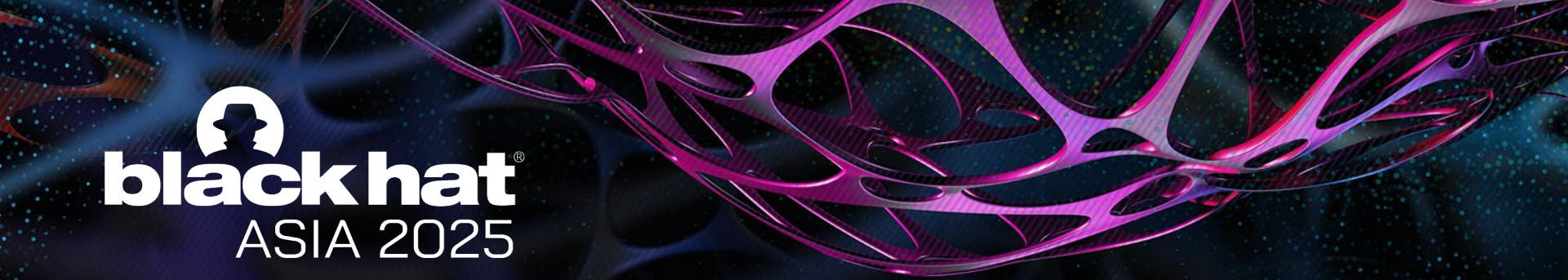


vCenter Lost

How the DCERPC Vulnerabilities Changed the Fate of ESXi

Hao Zheng Zibo Li Yue Liu

TianGong Team of QI-ANXIN Group

Who we are

Hao Zheng
@zhz__6951

Zibo Li
@zblee_

Yue Liu
@Mr_LiuYue

Who we are

TianGong Lab of QI-ANXIN Group

- Focusing on vulnerability discovery and exploitation
- Targeting at Edge Devices/IOT/OS/Virtualization/Browser
- Works published in Black Hat, HITBSecConf, EuroS&P, Usenix, ACM CCS
- Awarded in GeekPwn, Tianfu Cup, Matrix Cup
- Website: https://tiangonglab.github.io/
- X: @TianGongLab

TianGong Lab

Our previous work on VMware

- Long-term Focus on VMware's virtualization security
- Discovered and reported multiple vulnerabilities in both ESXi and Workstation
- Presented our research at DEFCON, HITB

Transition to vCenter Server Research

Noticed VMware vCenter Server Out-of-Bounds Write Vulnerability (CVE-2023-34048)

- memory corruption
- remote code execution
- exploitation in the wild

https://support.broadcom.com/web/ecx/support-content-notification/-/external/content/SecurityAdvisories/0/23677

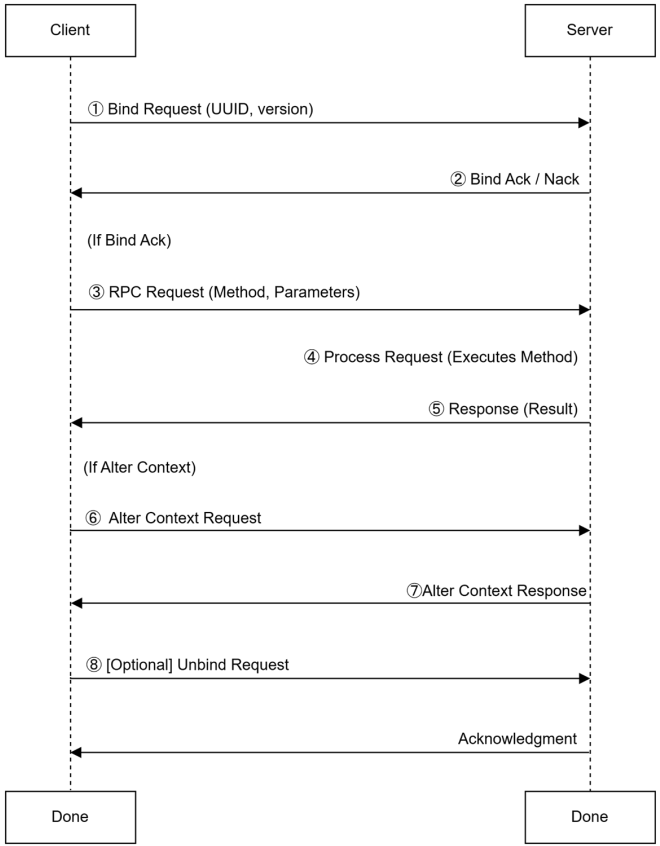
Agenda

1. DCERPC Protocol Overview
2. DCERPC Vulnerabilities Discovery
3. Exploitation Challenges & Techniques
4. Beyond vCenter: Privilege Escalation and Control
5. Conclusion

1. DCERPC Protocol Overview

DCERPC Protocol

- A remote procedure call (RPC) mechanism
- Widely used in Unix and Windows NT systems.
- Uses Interface Definition Language (IDL) to define interfaces.



DCERPC Protocol Structure

- Consists of fixed common header and optional fields
- There are 20 valid packet types

```
typedef struct
{
    unsigned8  rpc_vers;              /* 00:01 RPC version - major */
    unsigned8  rpc_vers_minor;        /* 01:01 RPC version - minor */
    unsigned8  ptype;                 /* 02:01 packet type */
    unsigned8  flags;                 /* 03:01 flags */
    unsigned8  drep[4];               /* 04:04 ndr format */
    unsigned16 frag_len;              /* 08:02 fragment length */
    unsigned16 auth_len;              /* 10:02 authentication length */
    unsigned32 call_id;               /* 12:04 call identifier */
} rpc_cn_common_hdr_t, *rpc_cn_common_hdr_p_t;
```

```
#define RPC_C_CN_PKT_REQUEST              0    /* client -> server */
#define RPC_C_CN_PKT_PING                 1    /* client -> server */
#define RPC_C_CN_PKT_RESPONSE             2    /* server -> client */
#define RPC_C_CN_PKT_FAULT                3    /* server -> client */
#define RPC_C_CN_PKT_WORKING              4    /* server -> client */
#define RPC_C_CN_PKT_NOCALL               5    /* server -> client */
#define RPC_C_CN_PKT_REJECT               6    /* server -> client */
#define RPC_C_CN_PKT_ACK                  7    /* client -> server */
#define RPC_C_CN_PKT_QUIT                 8    /* client -> server */
#define RPC_C_CN_PKT_FACK                 9    /* both directions  */
#define RPC_C_CN_PKT_QUACK               10    /* server -> client */
#define RPC_C_CN_PKT_BIND                11    /* client -> server */
#define RPC_C_CN_PKT_BIND_ACK            12    /* server -> client */
#define RPC_C_CN_PKT_BIND_NAK            13    /* server -> client */
#define RPC_C_CN_PKT_ALTER_CONTEXT       14    /* client -> server */
#define RPC_C_CN_PKT_ALTER_CONTEXT_RESP  15    /* server -> client */
#define RPC_C_CN_PKT_AUTH3               16    /* client -> server */
#define RPC_C_CN_PKT_SHUTDOWN            17    /* server -> client */
#define RPC_C_CN_PKT_REMOTE_ALERT        18    /* client -> server */
#define RPC_C_CN_PKT_ORPHANED            19    /* client -> server */
#define RPC_C_CN_PKT_MAX_TYPE            19
#define RPC_C_CN_PKT_INVALID             0xff
```

DCERPC in vCenter

- Used in ports 2012, 2014, and 2020

```
tcp        0      0 0.0.0.0:636             0.0.0.0:*               LISTEN      2706/vmdird
tcp        0      0 0.0.0.0:2012            0.0.0.0:*               LISTEN      2706/vmdird
```

```
tcp        0      0 0.0.0.0:2014            0.0.0.0:*               LISTEN      3274/vmcad
```

```
tcp        0      0 0.0.0.0:2020            0.0.0.0:*               LISTEN      2511/vmafdd
```

```
root@localhost [ ~ ]# ldd /usr/lib/vmware-vmdir/sbin/vmdird | grep "dce"
        libdcerpc.so.1 => /opt/likewise/lib64/libdcerpc.so.1 (0x00007f86d206b000)
root@localhost [ ~ ]# ldd /usr/lib/vmware-vmca/sbin/vmcad | grep "dcerpc"
        libdcerpc.so.1 => /opt/likewise/lib64/libdcerpc.so.1 (0x00007fdc52bd0000)
root@localhost [ ~ ]# ldd /usr/lib/vmware-vmafd/sbin/vmafdd | grep "dcerpc"
        libdcerpc.so.1 => /opt/likewise/lib64/libdcerpc.so.1 (0x00007f6a113c6000)
root@localhost [ ~ ]#
```

2. DCERPC Vulnerabilities Discovery

CVE-2024-37079/37080

3a. VMware vCenter Server multiple heap-overflow vulnerabilities (CVE-2024-37079, CVE-2024-37080)

Description:
The vCenter Server contains multiple heap-overflow vulnerabilities in the implementation of the DCERPC protocol. VMware has evaluated the severity of these issues to be in the Critical severity range with a maximum CVSSv3 base score of 9.8.

Known Attack Vectors:
A malicious actor with network access to vCenter Server may trigger these vulnerabilities by sending a specially crafted network packet potentially leading to remote code execution.

Resolution:
To remediate CVE-2024-37079, and CVE-2024-37080 apply the updates listed in the 'Fixed Version' column of the 'Response Matrix' below to affected deployments.

Workarounds:
In-product workarounds were investigated, but were determined to not be viable.

Additional Documentation:
A supplemental FAQ was created for additional clarification. Please see: https://core.vmware.com/resource/vmsa-2024-0012-questions-answers

CVE-2024-37079

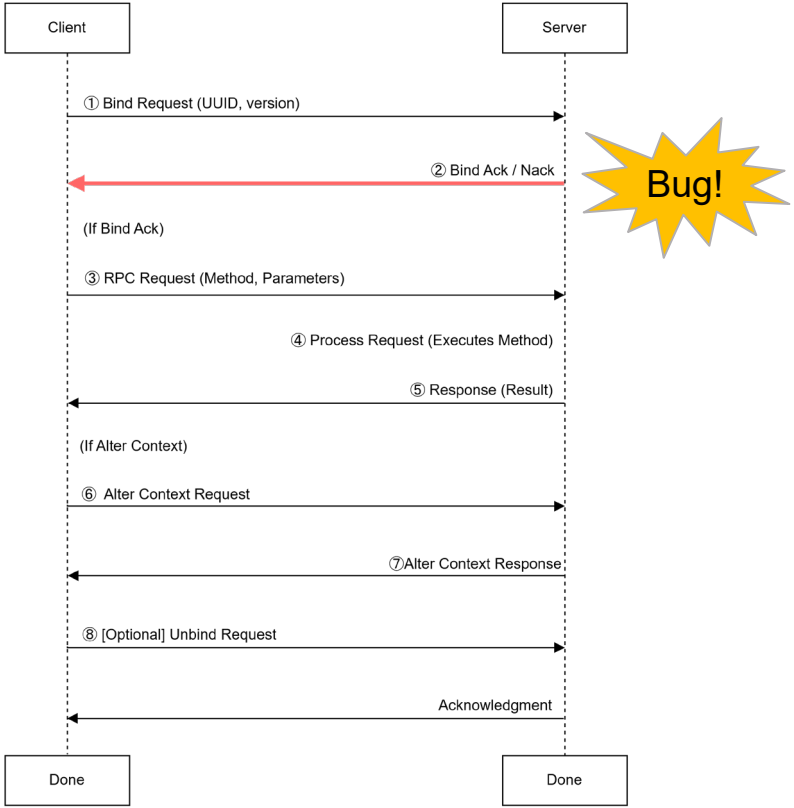
Request → Parsing → ☑ (Well-researched)

Response → Generation → ⚠ (Overlooked vulnerability found)

CVE-2024-37079

response of bind authentication packets

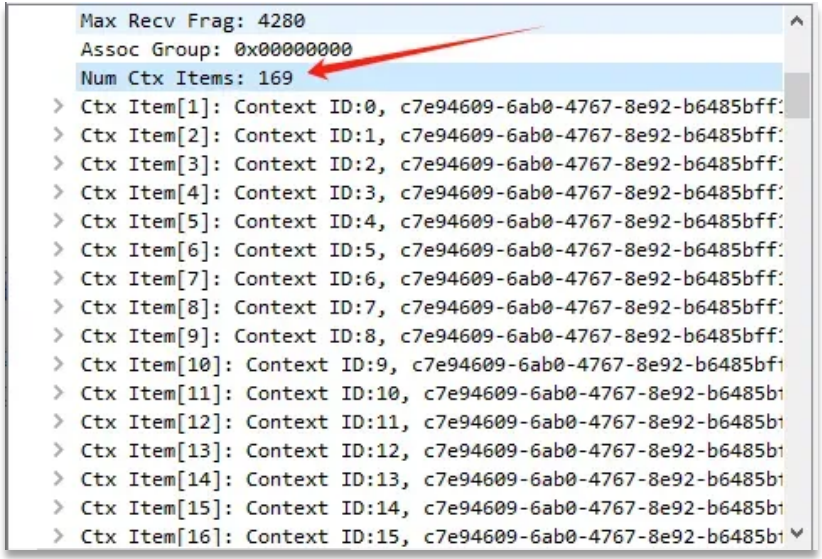
```
INTERNAL void rpc__cn_assoc_process_auth_tlr
(
  rpc_cn_assoc_p_t        assoc,
  rpc_cn_packet_p_t       req_header,
  unsigned32              req_header_size,
  rpc_cn_packet_p_t       resp_header,
  unsigned32              *header_size,
  unsigned32              *auth_len,
  rpc_cn_sec_context_p_t  *sec_context,
  boolean     old_client,
  unsigned32              *st
)
```



CVE-2024-37079

header_size = ((pres_cont_list->n_context_elem - 1) * 0x18) + 0x1c + 0x20

The value of n_context_elem comes from bind request packet



CVE-2024-37079

do_alter_cont_req_action_rtn function checks the number of Ctx Items

1. 0x1C + 0x18 * (pres_cont_list->n_context_elem - 1) ≤ 0xFE4
2. 0x18 * (pres_cont_list->n_context_elem - 1) ≤ 0xFC8
3. (pres_cont_list->n_context_elem - 1) ≤ 0xA8
4. pres_cont_list->n_context_elem ≤ 0xA9

Max(pres_cont_list->n_context_elem) = 0xA9

Max(header_size) = ((0xA9 - 1) * 0x18) + 0x1C + 0x20 = **0xFFC**

CVE-2024-37079

auth_len depends on header_size

*header_size + RPC_CN_PKT_SIZEOF_COM_AUTH_TLR = 0xFFC + 8 = 0x1004

*auth_len = rpc_g_cn_large_frag_size - *header_size = 0x1000 – 0x1004 = **0xFFFFFFFC**


```
*header_size += RPC_CN_PKT_SIZEOF_COM_AUTH_TLR;  // RPC_CN_PKT_SIZEOF_COM_AUTH_TLR = 0x8
*auth_len = rpc_g_cn_large_frag_size - *header_size;
RPC_CN_AUTH_FMT_SRVR_RESP(
      sec->sec_status,
      authn_protocolc,
      sec,
      req_auth_tlr->auth_value,
      req_header->bind.hdr.common_hdr.auth_len,
      resp_auth_tlr->auth_value,
      auth_len);
```

CVE-2024-37079

auth_len indicating how much free space remains

auth_len = 0xFFFFFFFC

Only 4 bytes free space

```
void __fastcall rpc__ntlmauth_cn_fmt_srvr_resp(
        unsigned32 verify_st,
        rpc_cn_assoc_sec_context_p_t assoc_sec,
        rpc_cn_sec_context_p_t sec,
        pointer_t req_auth_value,
        unsigned32 req_auth_value_len,
        pointer_t auth_value,
        unsigned32 *auth_len)
{
  length = assoc_sec->krb_message.length;
  data = assoc_sec->krb_message.data;
  assoc_sec->krb_message.length = 0;
  assoc_sec->krb_message.data = 0LL;
  output_token.value = data;
  v10 = *auth_len;
  output_token.length = length;
  if ( length > v10 )
  {
    gss_release_buffer(&minor_status, &output_token, length, req_auth_value, req_auth_value_len, auth_value);
    *auth_value_len = 0;
  }
  else
  {
    *auth_len = length;
    memcpy(auth_value, data, length);
    gss_release_buffer(&minor_status, &output_token, v11, v12, v13, v14);
  }
}
```

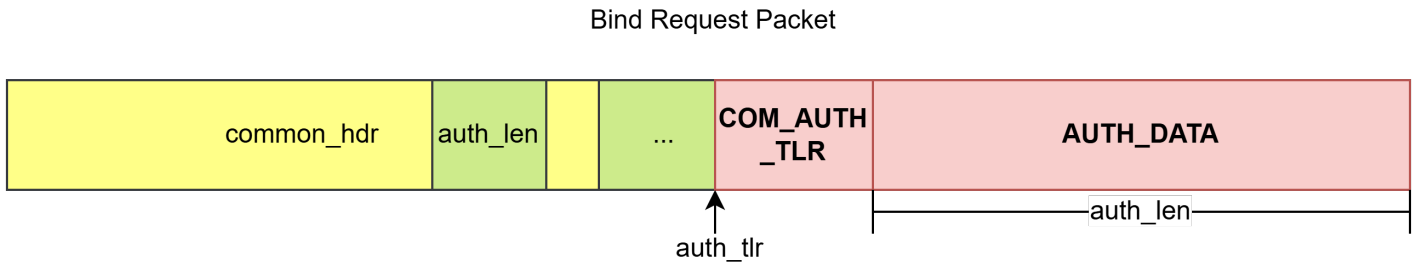
Always false

Overflow!!!

CVE-2024-37080

Authentication Trailer (Auth TLR) is an optional structure appended to a PDU

```
auth_tlr = (rpc_cn_auth_tlr_t *) ((unsigned8 *)(pktp) +
                fragbuf_p->data_size -
                (auth_len + RPC_CN_PKT_SIZEOF_COM_AUTH_TLR));
```



CVE-2024-37080

Is the check for **auth_tlr** sufficient?

Auth TLR validation

```
auth_tlr = (rpc_cn_auth_tlr_t *) ((unsigned8 *)(pktp) +
                fragbuf_p->data_size -
                (auth_len + RPC_CN_PKT_SIZEOF_COM_AUTH_TLR));
            if ( ((unsigned8 *)(auth_tlr) < (unsigned8 *)(pktp)) ||
                ((unsigned8 *)(auth_tlr) > (unsigned8 *)(pktp) + fragbuf_p->data_size) ||
                ((unsigned8 *)(auth_tlr) + auth_len < (unsigned8 *)(pktp)) ||
                ((unsigned8 *)(auth_tlr) + auth_len > (unsigned8 *)(pktp) + fragbuf_p->da
            {
                ......
                st = rpc_s_protocol_error;
                break;
            }
```

CVE-2024-37080

len(**AUTH_DATA**) == **auth_len**?

What If set **auth_len** = 1 without any authentication data? **Validation Pass!**

```
auth_tlr = (rpc_cn_auth_tlr_t *) ((unsigned8 *)(pktp) +
                fragbuf_p->data_size -
                (auth_len + RPC_CN_PKT_SIZEOF_COM_AUTH_TLR));
            if ( ((unsigned8 *)(auth_tlr) < (unsigned8 *)(pktp)) ||
                ((unsigned8 *)(auth_tlr) > (unsigned8 *)(pktp) + fragbuf_p->data_size) ||
                ((unsigned8 *)(auth_tlr) + auth_len < (unsigned8 *)(pktp)) ||
                ((unsigned8 *)(auth_tlr) + auth_len > (unsigned8 *)(pktp) + fragbuf_p->da
            {
                ......
                st = rpc_s_protocol_error;
                break;
            }
```

CVE-2024-37080

auth_len + header_size > pdu_len

leading to an **integer underflow** in input_token.len

```
53  auth_len = __ROL2__(v12, 8);
54  if ( !unpack_ints )
55    auth_len = v15;
56  input_token.base = (void *)(pdu + 24);
57  input_token.len = pdu_len - 32 - (unsigned int)auth_len;
58  tail = *(schn_tail *)(auth_tlr + 8);
59  v17 = schn_unwrap(sec_ctx, sec_level, (__int64)&input_token, (__int64)&output_token, (__int64)&tail);
60  memcpy(input_token.base, output_token.base, output_token.len);
```

Integer Unerflow!!!

CVE-2024-38812

3a. VMware vCenter Server heap-overflow vulnerability (CVE-2024-38812)

Description:
The vCenter Server contains a heap-overflow vulnerability in the implementation of the DCERPC protocol. VMware has evaluated the severity of this issue to be in the Critical severity range with a maximum CVSSv3 base score of 9.8.

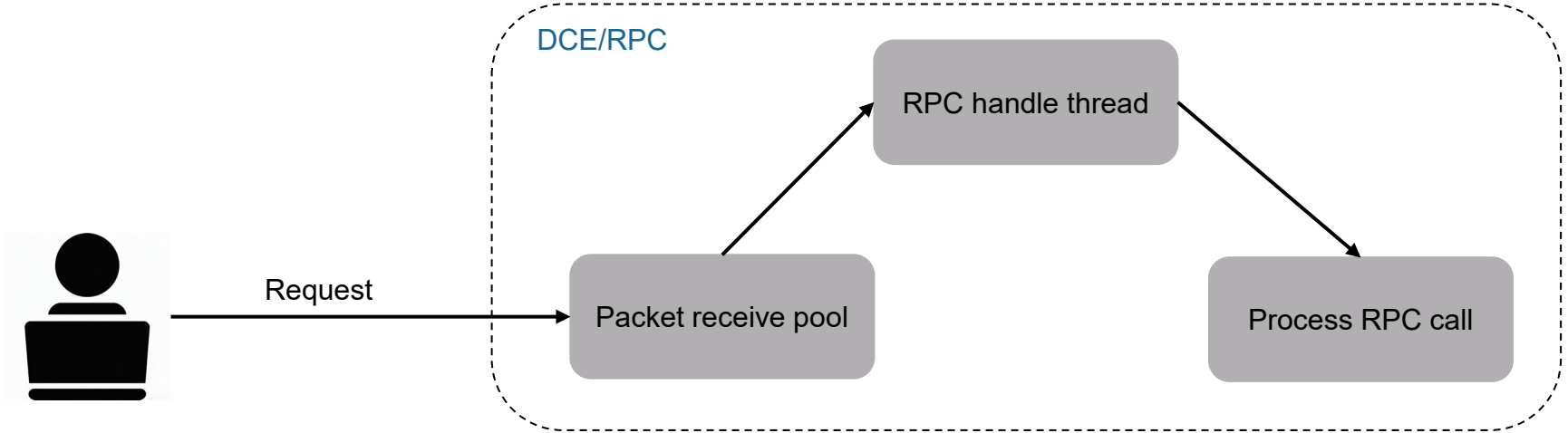
Known Attack Vectors:
A malicious actor with network access to vCenter Server may trigger this vulnerability by sending a specially crafted network packet potentially leading to remote code execution.

Resolution:
To remediate CVE-2024-38812 apply the updates listed in the 'Fixed Version' column of the 'Response Matrix' below to affected deployments.

Workarounds:
In-product workarounds were investigated, but were determined to not be viable.

Additional Documentation:
A supplemental FAQ was created for additional clarification. Please see: https://bit.ly/vcf-vmsa-2024-0019-qna

CVE-2024-38812



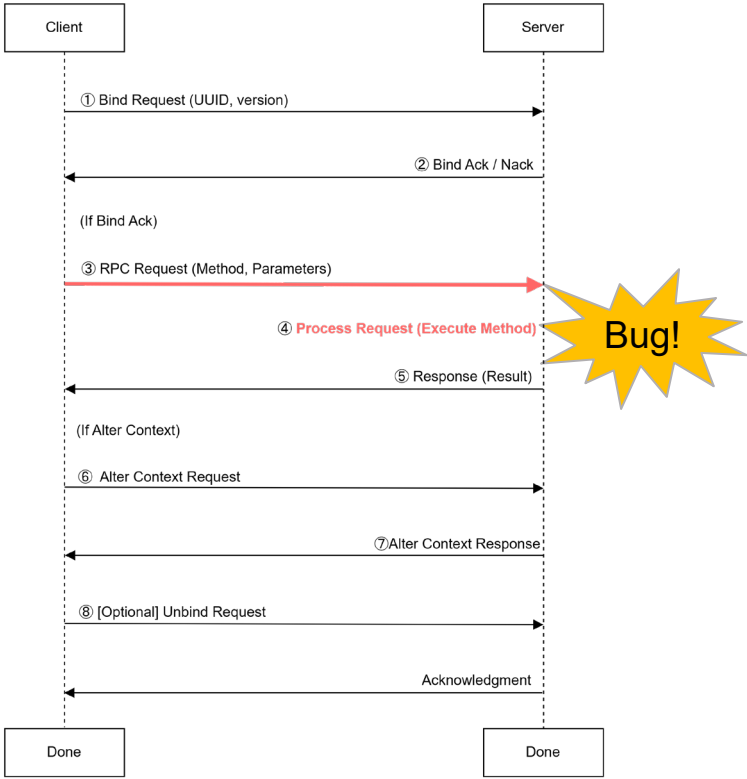
CVE-2024-38812

Call Request Packet

common_hdr	alloc_hint	ctx_id	**op_num**	**stub_data**

op_num: determine the rpc function to invoke

stub_data: parameters encoded using **NDR**



CVE-2024-38812

NDR Array Representation

- Maximum counts
- Offset
- Actual counts
- Elements

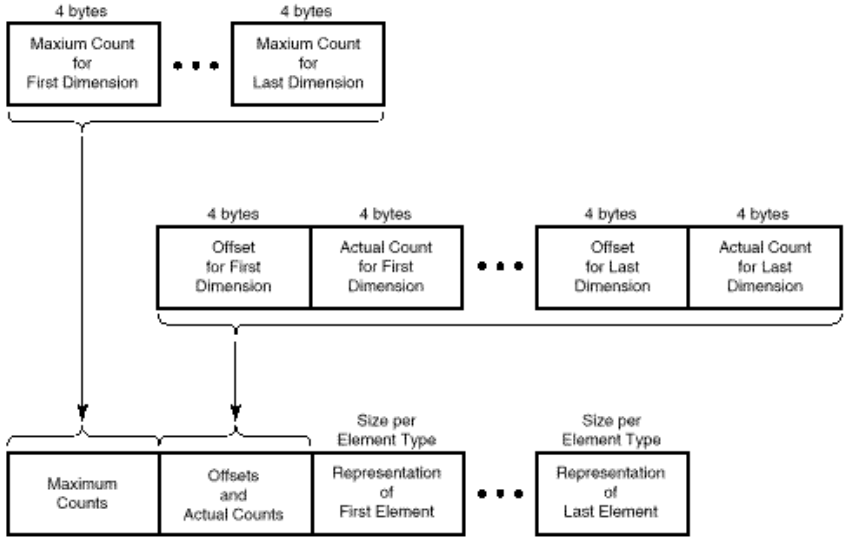


Figure 14-19 Multi-dimensional Conformant and Varying Array Representation

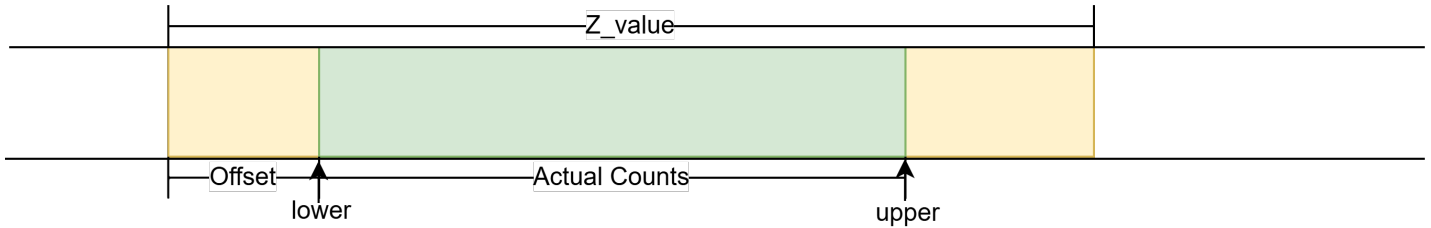
https://pubs.opengroup.org/onlinepubs/9629399/chap14.htm#tagfcjh_31

CVE-2024-38812

```
typedef struct IDL_bound_pair_t {
    idl_long_int lower;
    idl_long_int upper;
} IDL_bound_pair_t;
```

Convert to IDL_bound_pair_t

- lower = rang_list + Offset
- upper = Lower + (Actual Counts) * sizeof(Element)

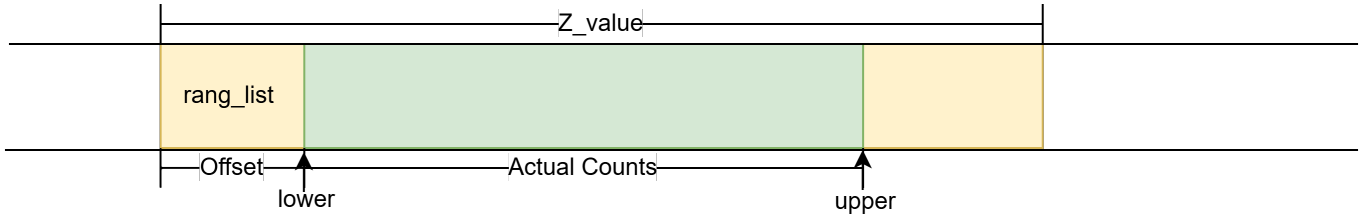


CVE-2024-38812

```
    if ( upper - range_list->lower > *Z_values )
LABEL_52:
      dcethread_exc_raise(&rpc_x_invalid_bound, "../dcerpc/idl_lib/ndrui.c", 0x47Cu);
    v11 = 1LL;
    while ( v7 > (unsigned int)v11 )
    {
      v12 = range_list[v11].upper - range_list[v11].lower;
      if ( v12 > Z_values[v11++] )
        goto LABEL_52;
    }
```

Practical implementation

- Z_value = Max Counts * sizeof(Element)
- range_list = malloc(Z_value)

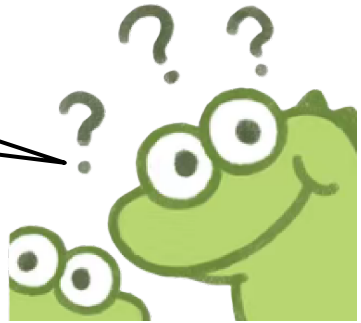


CVE-2024-38812

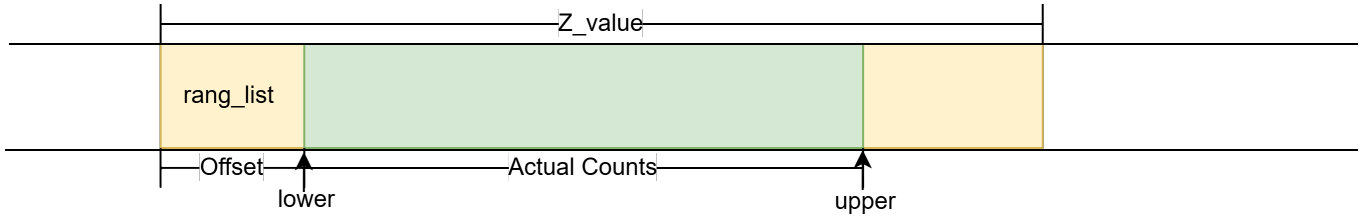
```
    if ( upper - range_list->lower > *Z_values )
LABEL_52:
      dcethread_exc_raise(&rpc_x_invalid_bound, "../dcerpc/idl_lib/ndrui.c", 0x47Cu);
    v11 = 1LL;
    while ( v7 > (unsigned int)v11 )
    {
      v12 = range_list[v11].upper - range_list[v11].lower;
      if ( v12 > Z_values[v11++] )
        goto LABEL_52;
    }
```

Practical implementation

Only check whether upper - lower is less than Z_value?



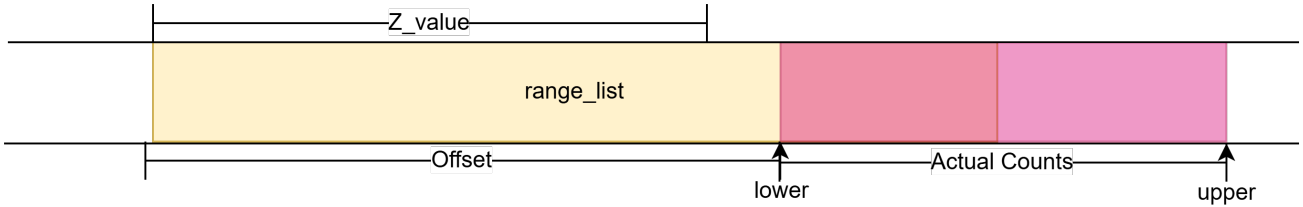
- Z_value = Max Counts * sizeof(Element)
- range_list = malloc(Z_value)



CVE-2024-38812

```
    if ( upper - range_list->lower > *Z_values )
LABEL_52:
      dcethread_exc_raise(&rpc_x_invalid_bound, "../dcerpc/idl_lib/ndrui.c", 0x47Cu);
    v11 = 1LL;
    while ( v7 > (unsigned int)v11 )
    {
      v12 = range_list[v11].upper - range_list[v11].lower;
      if ( v12 > Z_values[v11++] )
        goto LABEL_52;
    }
```

Practical implementation



CVE-2024-38813

3b. VMware vCenter privilege escalation vulnerability (CVE-2024-38813)

Description:
The vCenter Server contains a privilege escalation vulnerability. VMware has evaluated the severity of this issue to be in the Important severity range with a maximum CVSSv3 base score of 7.5.

Known Attack Vectors:
A malicious actor with network access to vCenter Server may trigger this vulnerability to escalate privileges to root by sending a specially crafted network packet.

Resolution:
To remediate CVE-2024-38813 apply the updates listed in the 'Fixed Version' column of the 'Response Matrix' below to affected deployments.

Workarounds:
None.

Additional Documentation:
A supplemental FAQ was created for additional clarification. Please see: https://bit.ly/vcf-vmsa-2024-0019-qna

CVE-2024-38813

Port Binding in the Initialization Phase

```
status = VmDirSyncCounterWaitEvent(gVmdirGlobals.pPortListenSyncCounter, &LDAP_ports_status);
if ( status )
{
  VmDirLog1(
    VMDIR_LOG_DEBUG,
    0xFFFFFFFF,
    "[file: %s][line: %d] [%s,%d]",
    "lotus/vmdir/server/vmdir/init.c",
    500LL,
    "lotus/vmdir/server/vmdir/init.c",
    500LL);
  return status;
}
if ( LDAP_ports_status )
{
  VmDirLog1(VMDIR_LOG_WARNING, 0xFFFFFFFF, "%s: NOT all LDAP ports are ready for accepting services.",
  goto LABEL_210;
}
```

if port occupied,

Stop & Return

```
LABEL_210:
    VmDirLog1(VMDIR_LOG_INFO, 0xFFFFFFFF, "Config MaxLdapOpThrs (%d)", gVmdirGlobals.dwMaxFlowCtrlThr);
    VmDirLogFeatureStateSwitches();
    return Mutex;
}
```

CVE-2024-38813

If port binding succeeds, drop privileges(**setgid, setuid**)

```
v32 = setgid(v28->pw_gid);
if ( v32 )
{
  v33 = strerror(v32);
  VmDirLog1(VMDIR_LOG_ERROR, 0xFFFFFFFF, "setgid failed: %s", v33);
  v29 = 1724LL;
  v40 = 1724LL;
}
else
{
  ppLda = v28->pw_uid;
  v34 = getuid();
  VmDirLog1(VMDIR_LOG_INFO, 0xFFFFFFFF, "Modifying uid from %d to %d", v34, ppLda);
  v35 = setuid(v28->pw_uid);
  if ( !v35 )
    goto LABEL_210;
  v36 = strerror(v35);
  VmDirLog1(VMDIR_LOG_ERROR, 0xFFFFFFFF, "setuid failed: %s", v36);
```

CVE-2024-38813

The code looks perfectly fine, so **where is the vulnerability**?

```
v32 = setgid(v28->pw_gid);
if ( v32 )
{
  v33 = strerror(v32);
  VmDirLog1(VMDIR_LOG_ERROR, 0xFFFFFFFF, "setgid failed: %s", v33);
  v29 = 1724LL;
  v40 = 1724LL;
}
else
{
  ppLda = v28->pw_uid;
  v34 = getuid();
  VmDirLog1(VMDIR_LOG_INFO, 0xFFFFFFFF, "Modifying uid from %d to %d", v34, ppLda);
  v35 = setuid(v28->pw_uid);
  if ( !v35 )
    goto LABEL_210;
  v36 = strerror(v35);
  VmDirLog1(VMDIR_LOG_ERROR, 0xFFFFFFFF, "setuid failed: %s", v36);
```

3. Exploitation Challenges & Techniques

Challenges in Exploiting vmdird

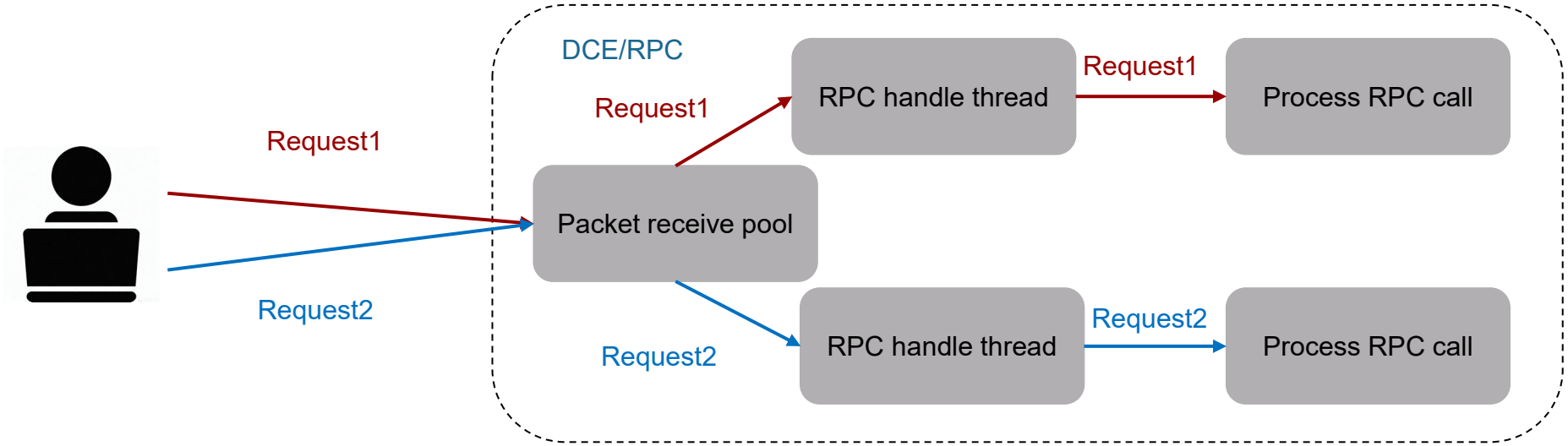
- **Multiple Memory Protection Mechanisms**

  vmdird process with multiple memory protection mechanisms enabled, including RELRO, Stack Canary, NX, PIE, and ASLR.

- **Triggered by network requests**

  uncontrollable memory allocations and releases make it difficult to precisely control memory layout

Multithread



Multithread

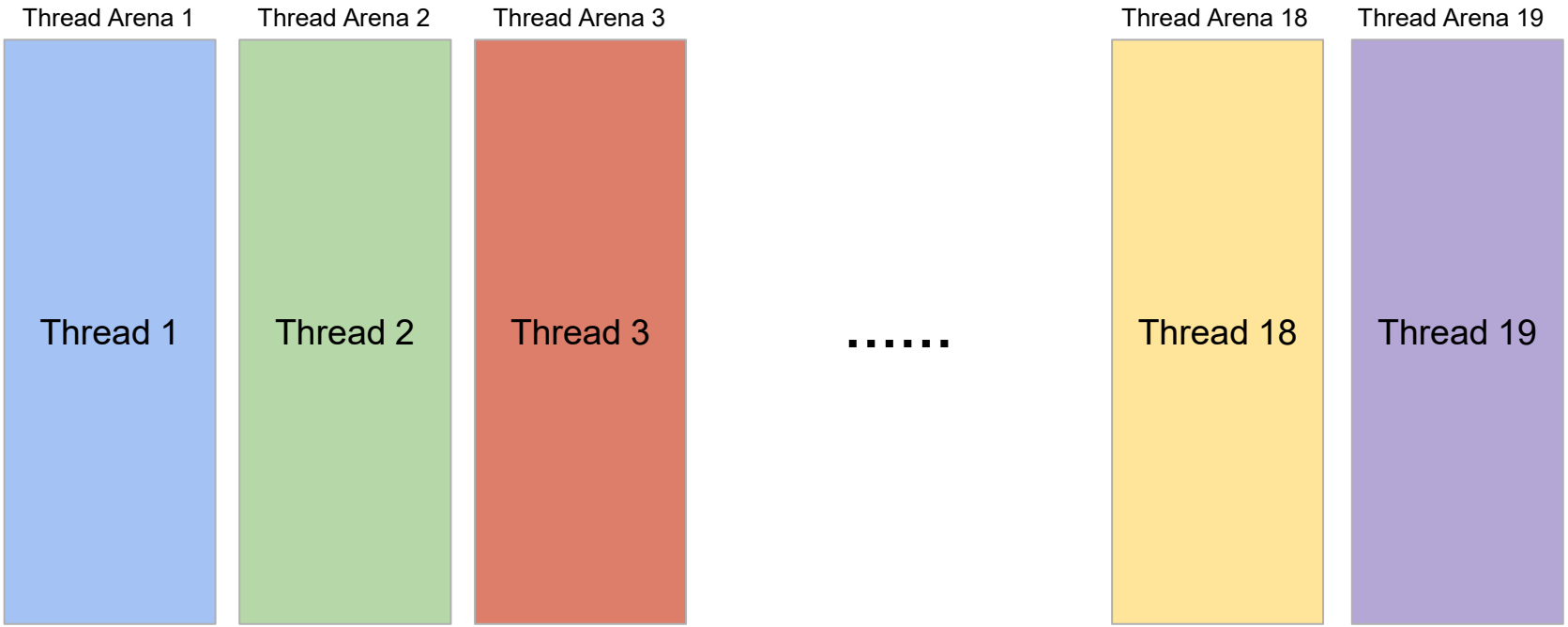
- Multithread arena
- Memory Isolation

```
0x7f79257fb000     0x7f79257fc000     0x1000          0x0   ---p
0x7f79257fc000     0x7f7925ffc000     0x800000        0x0   rw-p
0x7f7928000000     0x7f7928214000     0x214000        0x0   rw-p
0x7f7928214000     0x7f792c000000     0x3dec000 thread1 0x0 ---p
0x7f792c000000     0x7f792c114000     0x114000        0x0   rw-p
0x7f792c114000     0x7f7930000000     0x3eec000 thread2 0x0 ---p
0x7f7930000000     0x7f7930114000     0x114000        0x0   rw-p
0x7f7930114000     0x7f7934000000     0x3eec000       0x0   ---p
0x7f7934000000     0x7f7934114000     0x114000        0x0   rw-p
```

narenas_limit

```
(gdb) x/4gx 0x7f7ec13f9000+0x1D3C98
0x7f7ec15ccc98: 0x0000000000000020      0x0000000000000000
0x7f7ec15ccca8: 0x0000000000000000      0x0000000000000000
```

Multithread



Heap grooming

- receive_packet function

```
if ( !fbp )
  fbp = rpc__cn_fragbuf_alloc(1u);
if ( fbp->data_size <= 9 )
{
  frag_length = 0;
  v5 = fbp->max_data_size - fbp->data_size;
  goto LABEL_11;
}
```

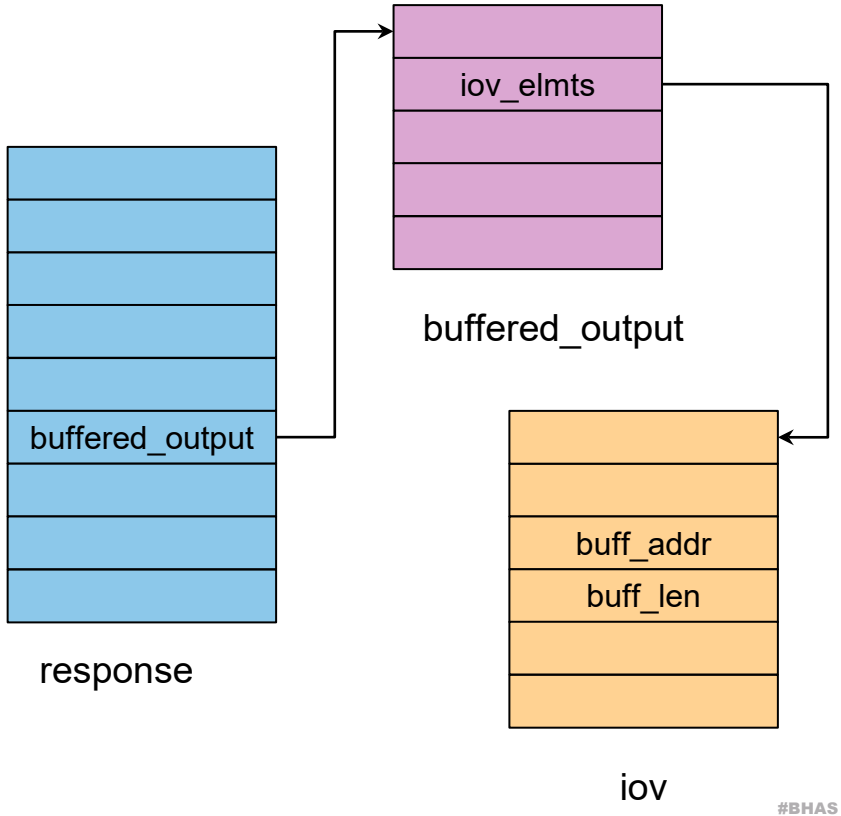
Infoleak Object

- a lot of log output functions in dce/rpc
- **syslog object** has function pointer

```
19    *(_DWORD *)v3 &= ~1u;
20    *(_DWORD *)(v3 + 116) |= 0x80u;
21    *(_QWORD *)(v3 + 240) = a1;
22    *(_QWORD *)(v3 + 248) = a2;
23    *(_QWORD *)(v3 + 224) = malloc;
24    *(_QWORD *)(v3 + 232) = free;
```

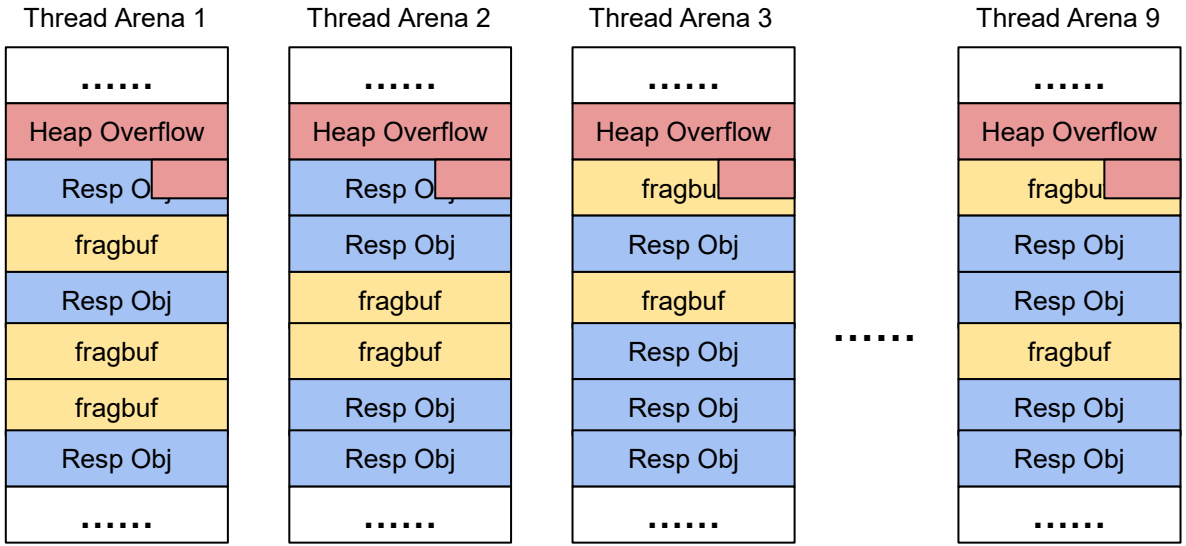
Out of Bound Read

- response packet output buffer structure
- **resonse->buffered_output->iov_elmts->buff_len**



Infoleak Memory layout

- heap spray on each thread heap
- Overwrite **resp_obj.buffered_output.iov_elmts.buff_len**
- Leak memory data from response.



Arbitrary Address Write

- Leveraging the **fragbuf** structure
- Keep reading until the packet is complete.
- In each loop iteration, **iov_base** is updated from **fragbuf->data_p**.

```
while (need_bytes > 0)
    {

        iov.iov_base = (byte_p_t)((unsigned8 *)(fbp->data_p) + fbp->data_size);
        iov.iov_len = need_bytes;
        serr = rpc__socket_recvmsg(assoc->cn_ctlblk.cn_sock,&iov,1,addr,&bytes_rcvd);
        ......
        fbp->data_size += bytes_rcvd;

        if ((frag_length == 0) && (fbp->data_size >= RPC_C_CN_FRAGLEN_HEADER_BYTES))
        {
            ......
            break;
        }

        if (frag_length == 0)
        {
            need_bytes = fbp->max_data_size - fbp->data_size;
        }
    }
```

Arbitrary Address Write

- Heap spray on each thread heap
- Overwrite **frag_obj->data_p** to an arbitrary address.
- Subsequent data sent will be written to the specified address.

Control Flow Hijacking

- vCenter uses the glibc heap manager
- hijack control flow by overwriting **__free_hook**

```
.bss:00000000001D3C80 __free_hook     db    ? ;               ; DATA XREF: LOAD:0000000000008B40↑o
.bss:00000000001D3C80                                         ; .got:__free_hook_ptr↑o
.bss:00000000001D3C81                 db    ? ;
.bss:00000000001D3C82                 db    ? ;
.bss:00000000001D3C83                 db    ? ;
```

4. Beyond vCenter:

Privilege Escalation and Control

Privilege Escalation

- Ports with the **FD_CLOEXEC** flag will not be inherited by child processes
- The file descriptor of port 2012 will not be inherited

```
listen(15, 128)                         = 0
socket(AF_INET6, SOCK_STREAM, IPPROTO_IP) = 16
fcntl(16, F_SETFD, FD_CLOEXEC)          = 0
setsockopt(16, SOL_SOCKET, SO_REUSEADDR, [1], 4) = 0
setsockopt(16, SOL_IPV6, IPV6_V6ONLY, [1], 4) = 0
bind(16, {sa_family=AF_INET6, sin6_port=htons(2012), sin6_flowinfo=htonl(0), inet_pton(AF_INET6, "::", &sin6_addr),
sin6_scope_id=0}, 28) = 0
```

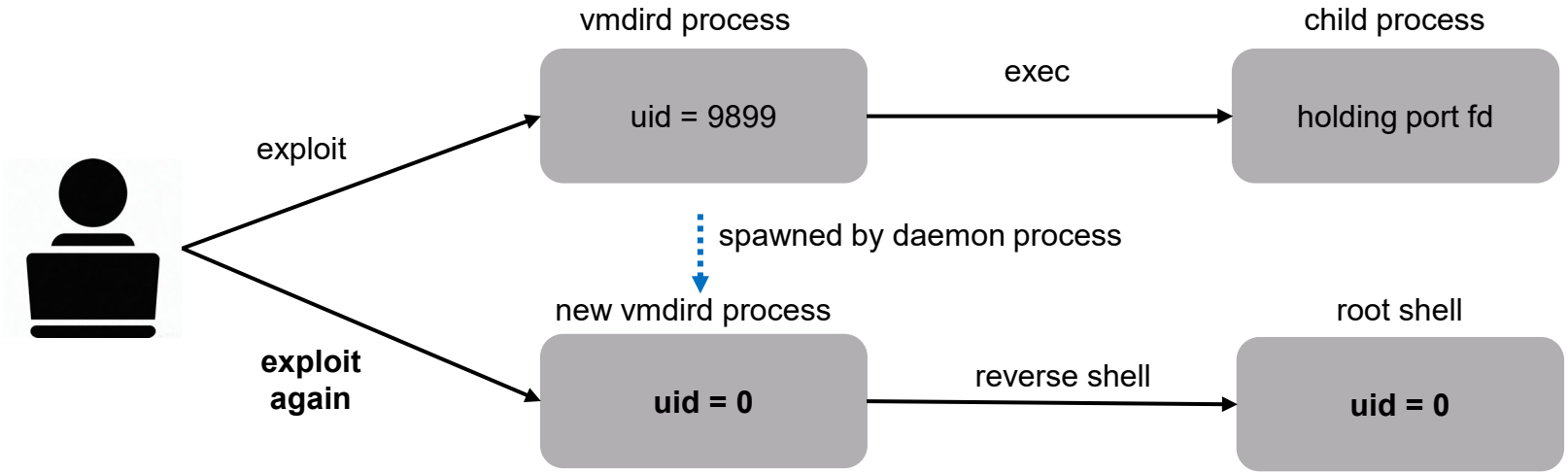
```
socket(AF_INET, SOCK_STREAM, IPPROTO_IP) = 15
fcntl(15, F_SETFD, FD_CLOEXEC)          = 0
setsockopt(15, SOL_SOCKET, SO_REUSEADDR, [1], 4) = 0
bind(15, {sa_family=AF_INET, sin_port=htons(2012), sin_addr=inet_addr("0.0.0.0")}, 16) = 0
```

Privilege Escalation

- 2012、 636 and 389 are all LDAP ports
- However, **FD_CLOEXEC** flag is not set for ports 636 and 389

```
1039    status = VmDirSyncCounterWaitEvent(gVmdirGlobals.pPortListenSyncCounter, &LDAP_ports_status);
1040    if ( status )
1041    {
1042      VmDirLog1(
1043        VMDIR_LOG_DEBUG,
1044        0xFFFFFFFF,
1045        "[file: %s][line: %d] [%s,%d]",
1046        "lotus/vmdir/server/vmdir/init.c",
1047        500LL,
1048        "lotus/vmdir/server/vmdir/init.c",
1049        500LL);
1050      return status;
1051    }
1052    if ( LDAP_ports_status )
1053    {
1054      VmDirLog1(VMDIR_LOG_WARNING, 0xFFFFFFFF, "%s: NOT all LDAP ports are ready for accepting services.",
```

Privilege Escalation



Control ESXi

- When ESXi initially connects to vCenter Server, it creates an account named vpxuser.
- vCenter Server uses vpxuser account to manage virtual machines on ESXi.

```
[root@localhost:~] cat /etc/passwd
root:x:0:0:Administrator:/:/bin/sh
dcui:x:100:100:DCUI User:/:/bin/sh
vpxuser:x:500:100:VMware Workstation administration account:/:/bin/sh
```

Control ESXi

- The PostgreSQL database in vCenter stores the connected esxi information
- The password is encrypted using OpenSSL Symmetric EVP
- The key can be easily obtained in vCenter

```
root@localhost [ ~ ]# psql -U postgres -d VCDB -c "select ip_address,user_name,password from vpx_host;"
  ip_address  | user_name |                                                  password
--------------+-----------+-----------------------------------------------------------------------------------------
              | vpxuser   |
(1 row)
```

```
root@localhost [ ~ ]# cat /etc/vmware-vpx/ssl/symkey.dat
                                                                                59231645a4350565c
```

```
➜  vcenter
```

```
➜  ~ nc -lvvp 1337
Listening on 0.0.0.0 1337
```

5. Conclusion

Conclusion

Bug Research Tips

- Focusing on Boundary Check and Data Content Detection
- Finding the Hidden Gems in Overlooked Areas

Exploitation Tips

- Leveraging Key Context Structures
- Mastering and Exploiting Low-Level Defense Mechanisms

Control Tips

- Dual Exploit Privilege Escalation
- Exploiting internal mechanisms

Thanks!

Q&A