Eternal Exploits

Reverse Engineering of FuzzBunch and MS17-010

zerosum0x0

DEFCON

August 1983

RISKSENSE

TRANSFORMING CYBER RISK MANAGEMENT

TOP SECRET//SI/TK//NOFORN//ICATION//4201337//REL TO DEF CON

Warning!

Presentation [redacted] classified information.
Those with active security clearances [redacted].

Spot The Fed

Champ 2018

Agenda

- ~~Recap (~2 mins)~~
  - ~~Equation Group (NSA)~~
  - ~~Shadow Brokers~~
- SMBv1 Internals (~5 mins)
  - ~~Network packets~~
  - Driver structures
- Exploits (~40 mins)
  - Blue
  - Champion
  - Romance
  - Synergy
- ~~Payloads (~10 mins)~~
  - ~~DoublePulsar~~
  - ~~DarkPulsar~~
  - ~~DanderSpritz~~

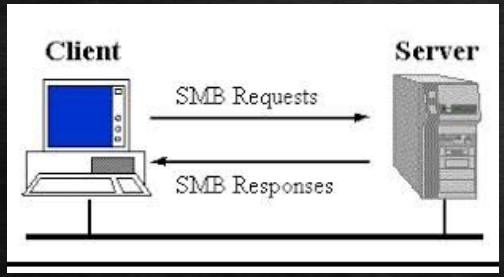
SMBv1 Internals

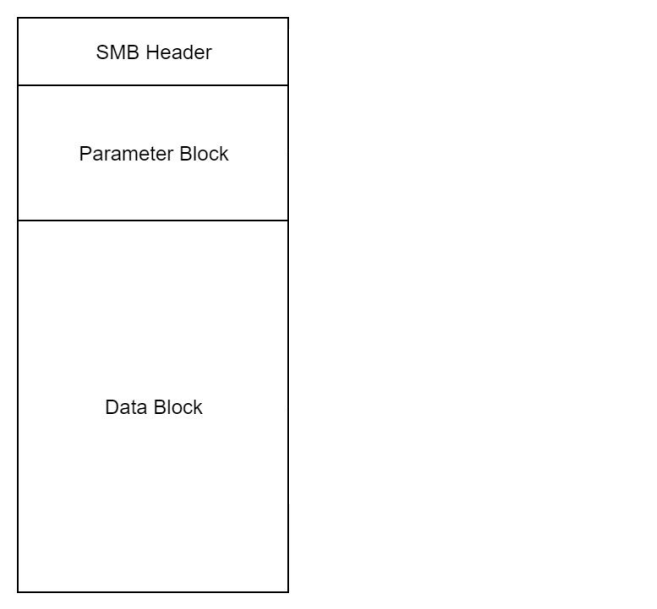
SMB Background

- Server Message Block
- 1983 - Invented by Barry Feigenbaum (IBM)
  - Also, NetBIOS
- Used EXTENSIVELY by Windows
  - "LanMan"
  - File Shares
- Extensible protocol
  - Transport for DCE/RPC
    - psexec

Server Message Block (v1)

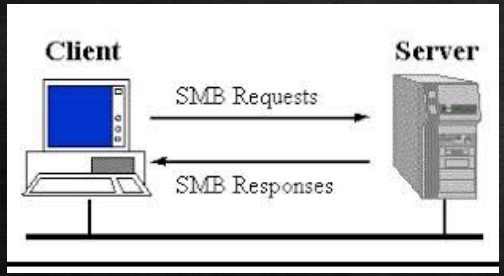
- Header Block
  - Command
  - Flags (request/reply, unicode)
  - Errno
  - Signature
  - UID/TID/PID/MID

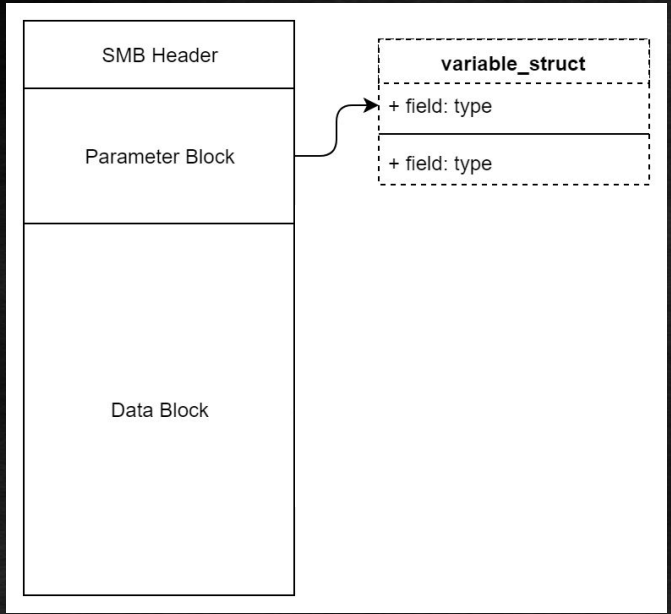




Server Message Block (v1)

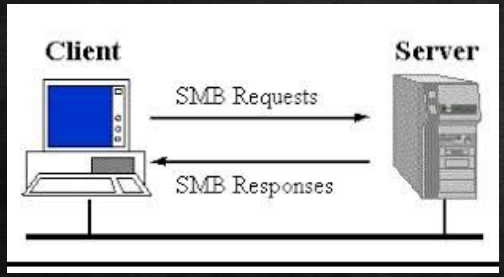
- Header Block
  - Command
  - Flags (request/reply, unicode)
  - Errno
  - Signature
  - UID/TID/PID/MID
- Parameter Block
  - Contains a **struct** specific to the command
    - Fixed size WORD count

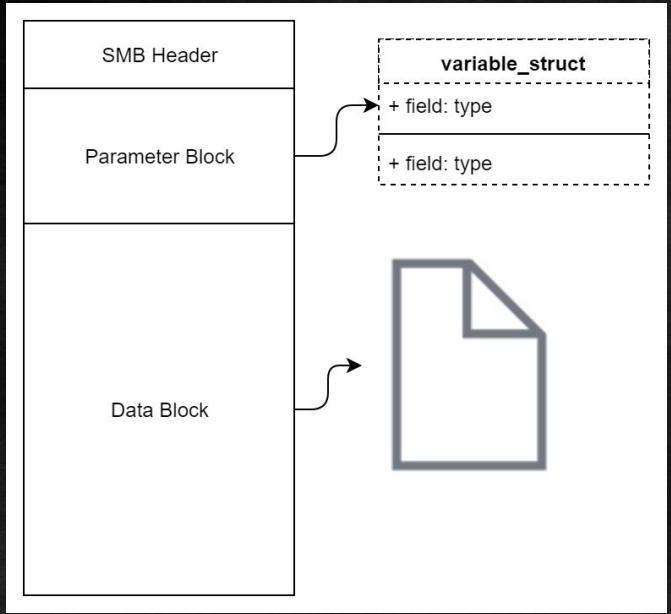




Server Message Block (v1)

- Header Block
  - Command
  - Flags (request/reply, unicode)
  - Errno
  - Signature
  - UID/TID/PID/MID
- Parameter Block
  - Contains a **struct** specific to the command
    - Fixed size WORD count
- Data Block
  - Misc. arbitrary info for the command
    - Variable size BYTE count



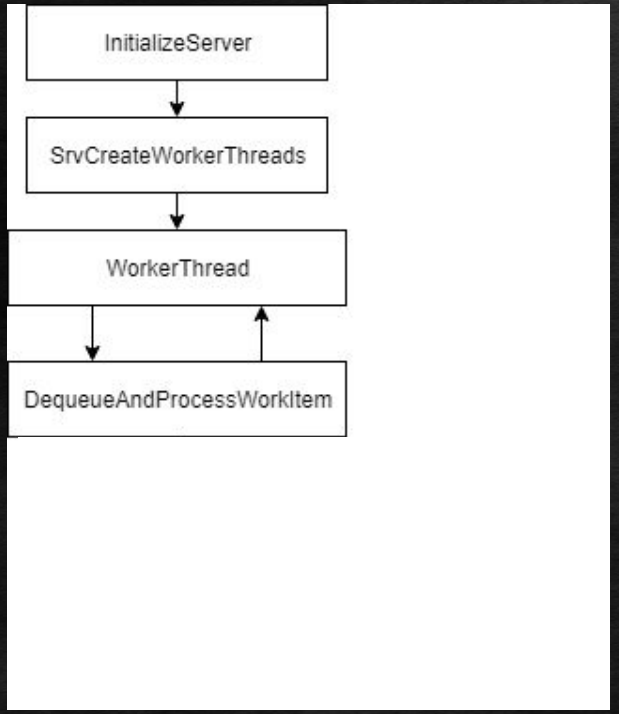


SMBv1 Dialects

- PC NETWORK PROGRAM 1.0
- MICROSOFT NETWORKS 1.03
- MICROSOFT NETWORKS 3.0
- LANMAN1.0
- Windows for Workgroups 3.1a
- LM1.2X002
- LANMAN2.1
- NT LM 0.12
- Cairo

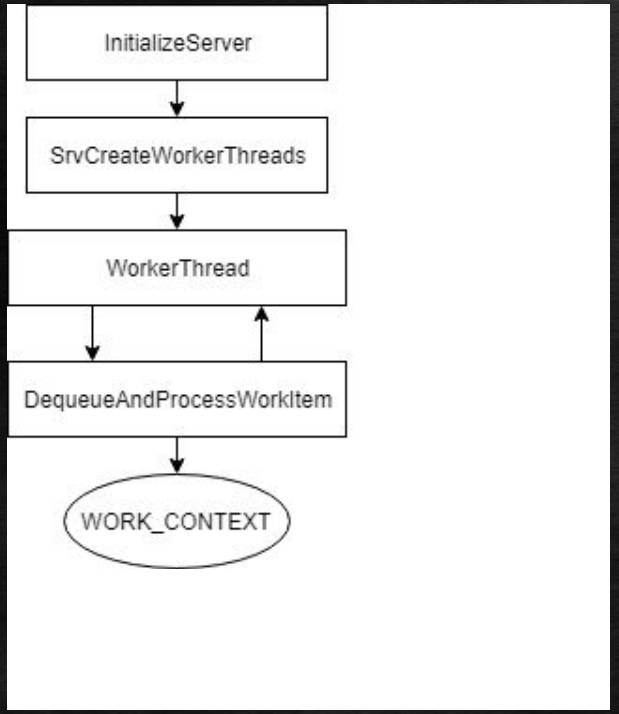
Srv.sys - SMBv1

- SrvWorkQueues
- SrvBlockingWorkQueues
  - Any operation that may take awhile
    - SMB is designed for speed



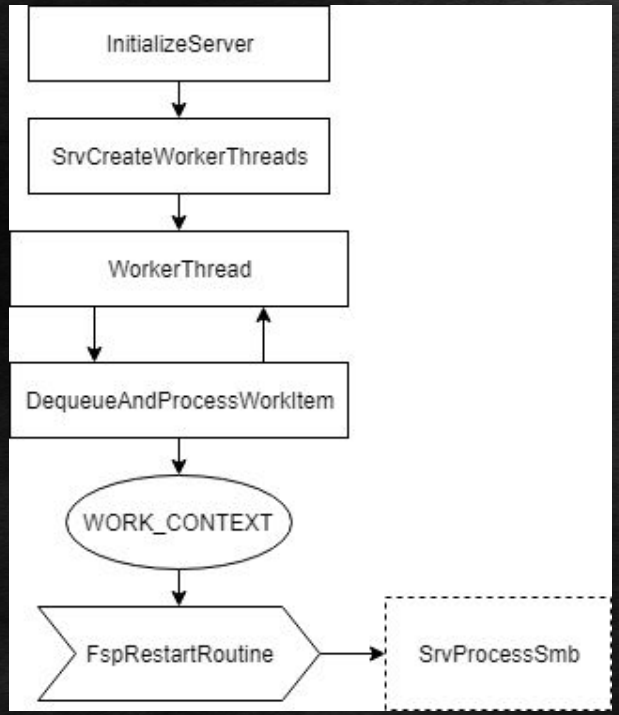
Srv.sys - SMBv1

- SrvWorkQueues
- SrvBlockingWorkQueues
  - Any operation that may take awhile
    - SMB is designed for speed
- WORK_CONTEXT
  - C union mega-struct SMB info



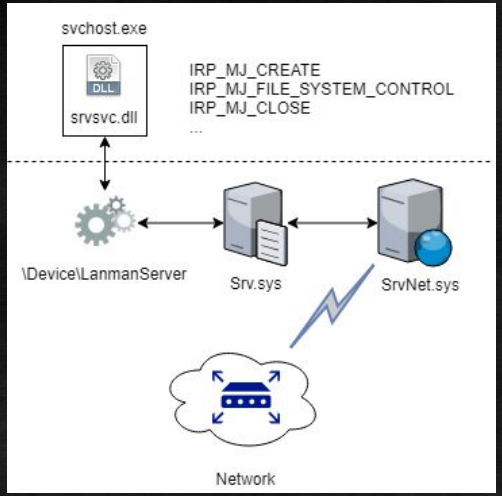
Srv.sys - SMBv1

- SrvWorkQueues
- SrvBlockingWorkQueues
  - Any operation that may take awhile
    - SMB is designed for speed
- WORK_CONTEXT
  - C union mega-struct SMB info
- SMB may be “restarted” multiple times
  - Send to a blocking thread
  - Wait for more data
  - Change FspStartRoutine, re-queue
    - Back of the line...



SrvNet.sys - SMBv1/2/3 Networking

- Added in Vista+
- Handles the networking (WSK)
  - 139 - NetBIOS
  - 445 - SMB Direct
- Registered handlers (undocumented, but trivial)
  - Srv.sys
  - Srv2.sys
- Library exports
  - Memory look-aside lists
  - Auth checks



SMB Messages (of Interest)

- Negotiate
- Session Setup
- Tree Connect
- NT Create
- Transactions

```
struct CONNECTION
{
    // ...

    SMB_DIALECT       SmbDialect;

    // ...

    UNICODE_STRING    ClientOSType;
    UNICODE_STRING    ClientLanManType;

    // ...
};
```

```
struct SESSION
{
    // ...
    PCONNECTION         Connection;
    // ...
    UNICODE_STRING      UserName;
    UNICODE_STRING      UserDomain;
    // ...
    USHORT              MaxBufferSize;
    USHORT              Uid;
    // ...
    BOOLEAN             IsNullSession;
    BOOLEAN             IsAdmin;
    // ...
};
```

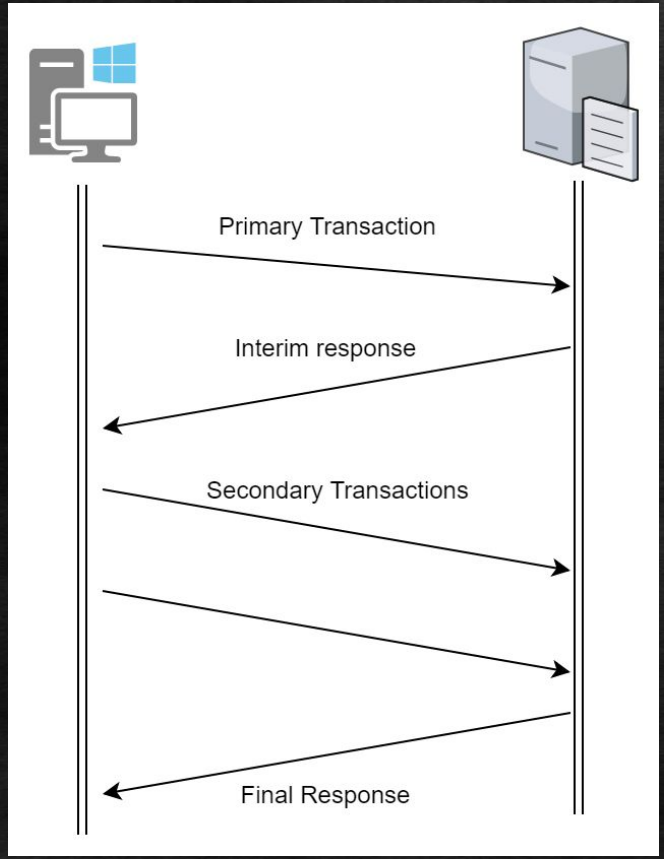
Administrative Trees (Shares)

- $ = generally hidden from UI
- C$
- D$
- ADMIN$
  - C:\Windows\
  - Administrator login required
- IPC$
  - Interprocess Communication Share
    - i.e. also, sometimes access to certain named pipes
  - Often, anonymous login allowed

```
struct TREECONNECT
{
    // ...
    USHORT     Tid;
    // ...
};
```

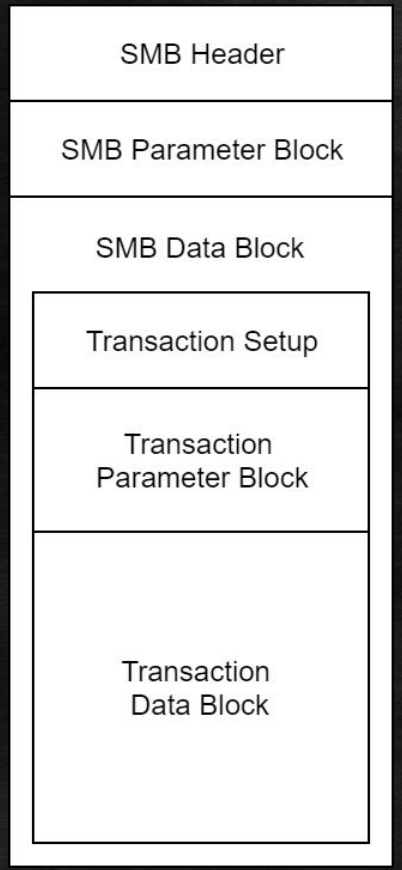
Transaction Life Cycle

- “IOCTL”
  - Perform variety of functions
    - Mostly file-system related
- Can be too large for one SMB
  - Primary
    - Intermediary response
  - Secondary(s)
- "Executed" once all parts are received
  - Like db transactions
  - Final response



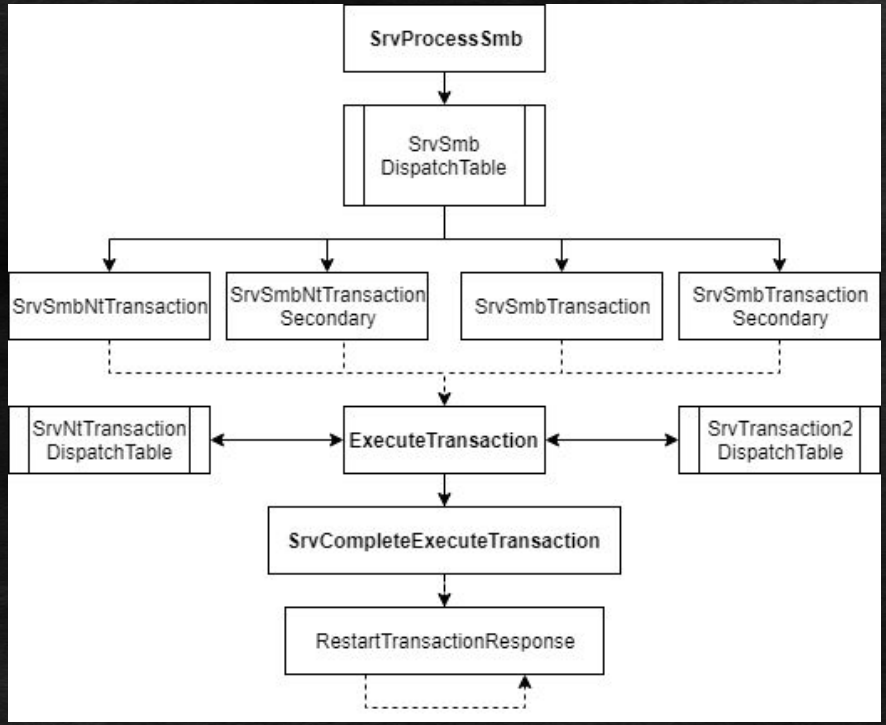
Transaction Packet Layout

- An SMB inside an SMB
  - In addition to SMB Parameter/Data Blocks:
    - Transaction Setup
      - For Primary trans
    - Transaction Parameter
    - Transaction Data



Transaction Type Processing

- Trans *(Trans1)*
  - Mailslots
  - MS-RAP
- Trans2
  - >8.3 shortnames
  - OS/2 to NT file stuff
  - Processed similar to Trans1
- NT Trans
  - Transaction Parameter/Data sizes
    - USHORT -> ULONG
- WriteAndX



Primary Transaction Data+Parameter

- Offset
  - How far into this SMB the TRANS data/parameter blocks begin

ParameterOffset	DataOffset

Primary Transaction Data+Parameter

- Offset
  - How far into this SMB the TRANS data/parameter blocks begin
- Count
  - How much is in this particular SMB

ParameterOffset	DataOffset
ParameterCount	DataCount

Primary Transaction Data+Parameter

- Offset
  - How far into this SMB the TRANS data/parameter blocks begin
- Count
  - How much is in this particular SMB
- TotalCount
  - How much will be sent over all Primary/Secondary SMB

ParameterOffset	DataOffset
ParameterCount	DataCount
TotalParameterCount	TotalDataCount

Primary Transaction Data+Parameter

- Offset
  - How far into this SMB the TRANS data/parameter blocks begin
- Count
  - How much is in this particular SMB
- TotalCount
  - How much will be sent over all Primary/Secondary SMB
- MaxCount
  - Maximum client buffer size to reserve for TRANS response

ParameterOffset	DataOffset
ParameterCount	DataCount
TotalParameterCount	TotalDataCount
MaxParameterCount	MaxDataCount

Secondary Transaction Data+Parameter

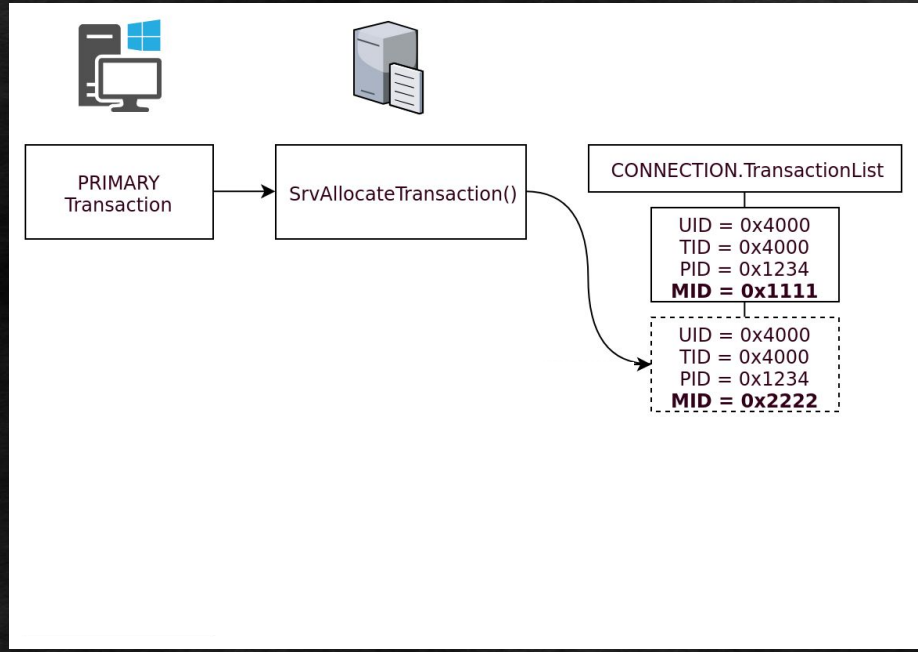
- Offset
  - How far into this SMB the TRANS data/parameter blocks begin
- Count
  - How much is in this particular SMB
- TotalCount
  - "MAY" be less than or equal to Primary SMB
- Displacement
  - An offset where to begin write operation into the server buffer
    - Generally, the cumulative total of preceding Primary+Secondary Count(s)

ParameterDisplacement	DataDisplacement

```
struct TRANSACTION
{
    // ...
    PCONNECTION         Connection;
    PSESSION            Session;
    PTREECONNECT        TreeConnect;
    // ...
    PCHAR       InParameters;
    PCHAR       OutParameters;      // often: = InParameters
    PCHAR       InData;
    PCHAR       OutData;            // often: = InData
    // ...
    USHORT      Tid;
    USHORT      Pid;
    USHORT      Uid;
    USHORT      OtherInfo;          // MID (...or, FID)
    // ...
};
```

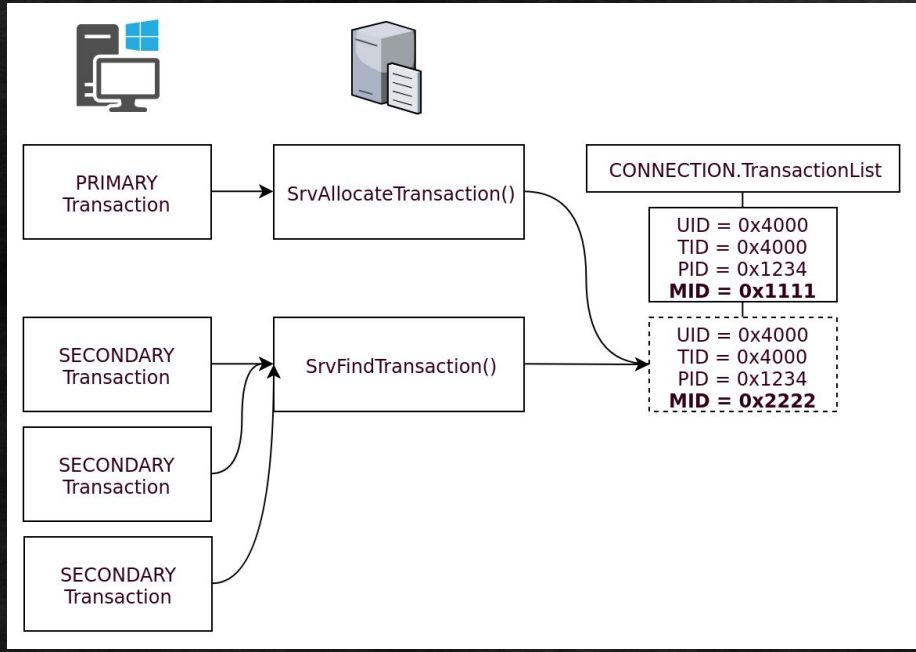
_TRANSACTION Memory

- SrvAllocateTransaction()
  - MIN alloc size = 0x5000
    - Except, Trans1.Setup == 0
  - MAX alloc size = 0x10400
    - STATUS_INSUFF_SERVER_RESOURCES



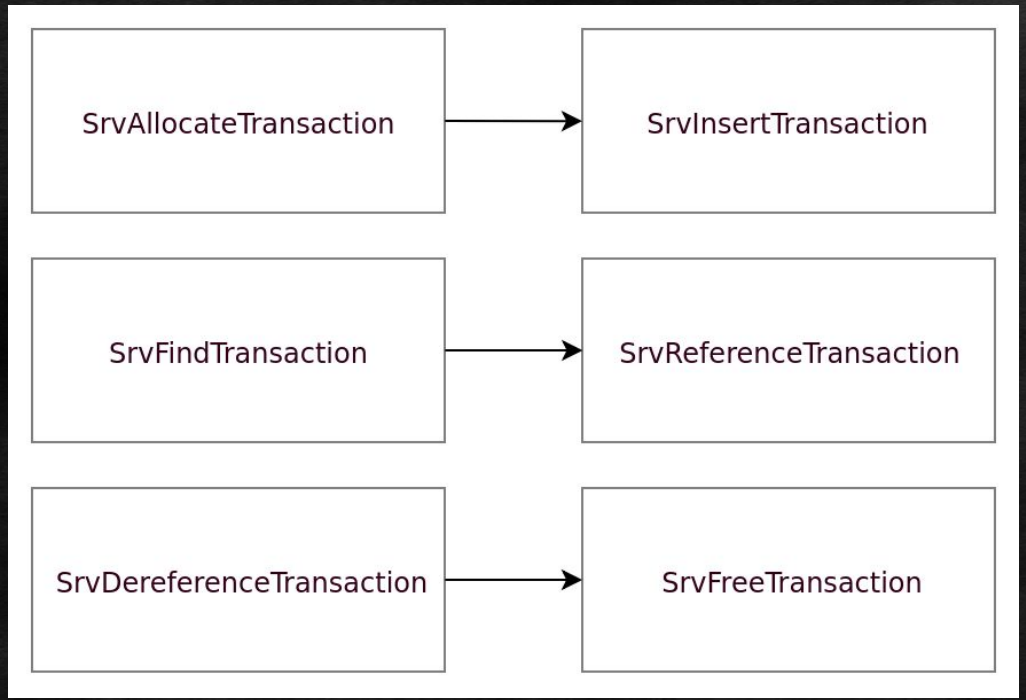
_TRANSACTION Memory

- SrvAllocateTransaction()
  - MIN alloc size = 0x5000
    - Except, Trans1.Setup == 0
  - MAX alloc size = 0x10400
    - STATUS_INSUFF_SERVER_RESOURCES
- SrvFindTransaction()
  - UID - server, const
  - TID - server, const
  - PID - client, const
  - OtherInfo
    - MID - client, arbitrary
    - FID - server, const



Reference Counted Memory Blocks

- WORK_CONTEXT
- CONNECTION
- SESSION
- TREECONNECT
- TRANSACTION



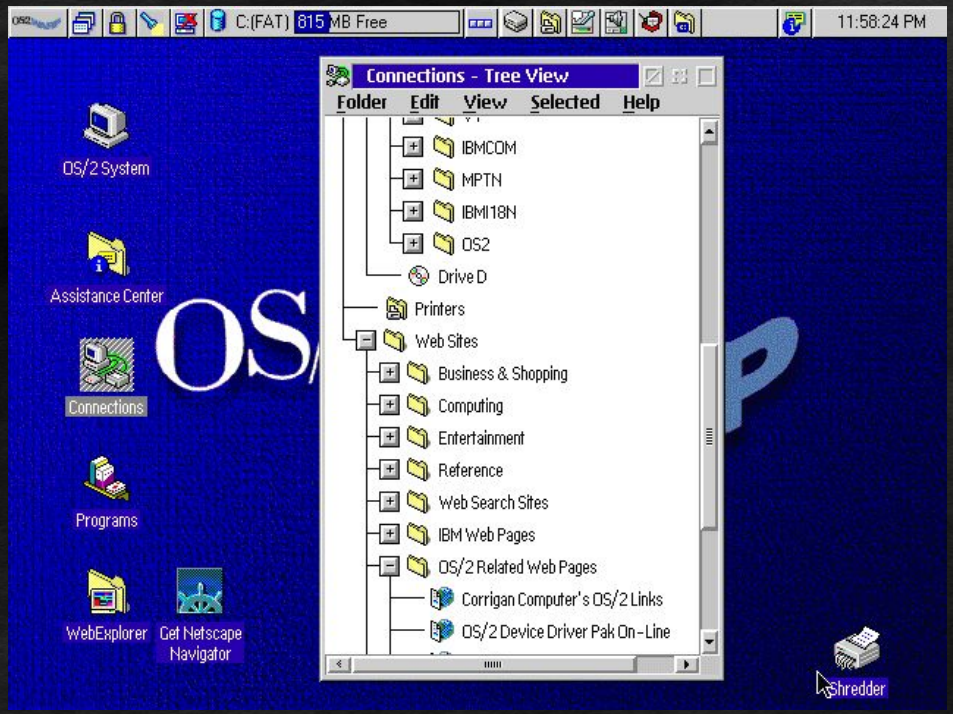
EternalBlue

Extended Attributes (EA)

- Name/Value key-pair
  - Metadata attached to files

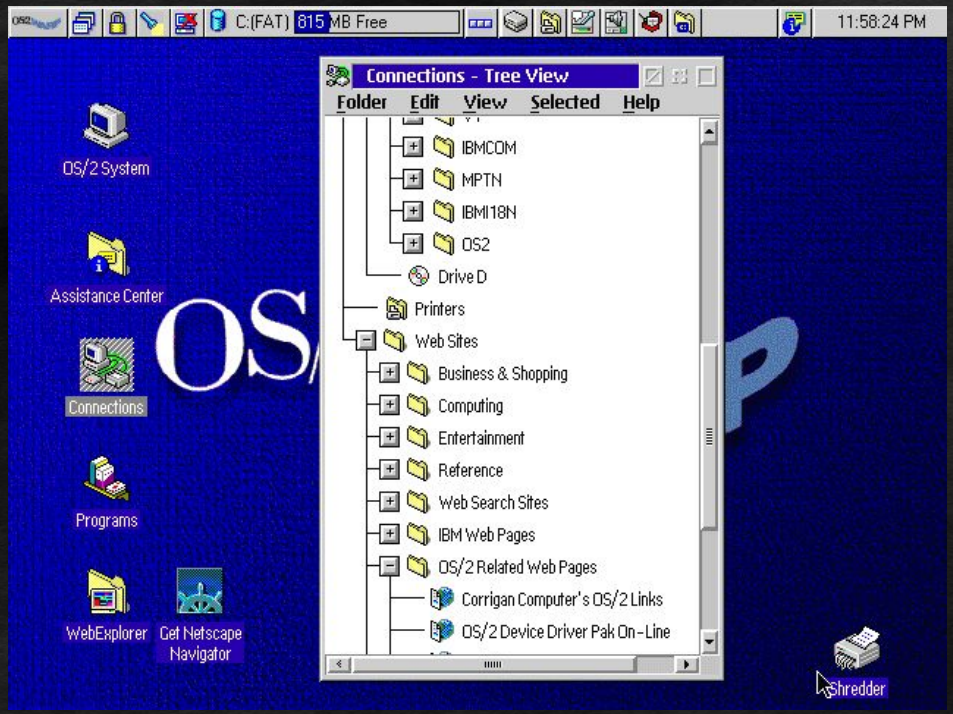
Extended Attributes (EA)

- Name/Value key-pair
  - Metadata attached to files
- OS/2 v1.2
  - Joint Microsoft/IBM OS
  - HPFS



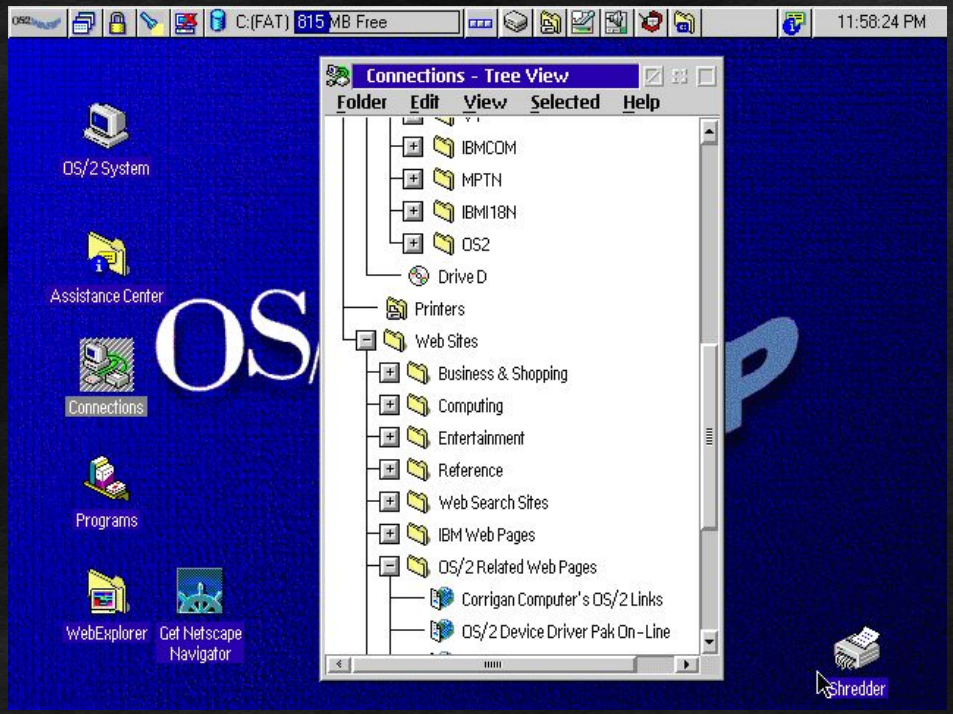
Extended Attributes (EA)

- Name/Value key-pair
  - Metadata attached to files
- OS/2 v1.2
  - Joint Microsoft/IBM OS
  - HPFS
- Windows NT
  - NTFS
    - Alternate Data Streams
  - WSL
    - Linux filesystem emulation
      - permissions, i.e. 0777
      - Case-sensitivity



Extended Attributes (EA)

- Name/Value key-pair
  - Metadata attached to files
- OS/2 v1.2
  - Joint Microsoft/IBM OS
  - HPFS
- Windows NT
  - NTFS
    - Alternate Data Streams
  - WSL
    - Linux filesystem emulation
      - permissions, i.e. 0777
      - Case-sensitivity
- FEA vs. GEA
  - FEA = name+value
  - GEA = name



OS/2 FEA

```
struct FEA
{
    BYTE    fEA;                        // "Flags" = 0x0 or 0x80
    BYTE    cbName;
    WORD    cbValue;

    // CHAR    szName[cbName];      // null-terminator
    // BYTE    chValue[cbValue];    // no null-terminator
};


#define FEA_SIZE(ea) \
    (sizeof(FEA) + (ea)->cbName + 1 + (ea)->cbValue)
```

OS/2 FEALIST

```
struct FEALIST
{
    ULONG  cbList;        // 32-bit size
    FEA    list[];        // Loop over all using NEXT_FEA()
};
```

```
#define NEXT_FEA(ea) \
    (char*)ea + FEA_SIZE(ea)
```

NT FEA

```
struct FILE_FULL_EA_INFORMATION
{
    ULONG       NextEntryOffset;
    UCHAR       Flags;                          // 0x0 or 0x80
    UCHAR       EaNameLength;
    USHORT      EaValueLength;

    // CHAR     EaName[EaNameLength];           // null-terminated
    // BYTE     EaValue[EaValueLength];         // not
    // BYTE     Alignment[+3 & ~3];             // align DWORD
};
```

NT FEA(LIST)

```
struct FILE_FULL_EA_INFORMATION
{
    ULONG       NextEntryOffset;        // Parse list until == 0
    UCHAR       Flags;                  // 0x0 or 0x80
    UCHAR       EaNameLength;
    USHORT      EaValueLength;

    // CHAR     EaName[EaNameLength];       // null-terminated
    // BYTE     EaValue[EaValueLength];     // not
    // BYTE     Alignment[+3 & ~3];         // align DWORD
};
```

```
ULONG SrvOs2FeaListSizeToNt(FEALIST *FeaList)
{
    lastValidLocation = FeaList + FeaList->cbList;
    fea = FeaList->list;
    ntBufferSize = 0;

    while (fea < lastValidLocation) {
        feaSize = fea->cbName + 1 + fea->cbValue;
        if (fea + feaSize > lastValidLocation) {
            SmbPutUshort(&FeaList->cbList, PTR_DIFF_SHORT(fea, FeaList));
            break;
        }
        ntBufferSize += FEA_SIZE(fea);
        fea = NEXT_FEA(fea);
    }

    return ntBufferSize;
}
```

Bug #1 - Integer Cast Error

```
ULONG FEALIST.cbList;

SmbPutUshort(&FeaList->cbList, PTR_DIFF_SHORT(fea, FeaList));
```

Bug #1 - Integer Cast Error

```
ULONG FEALIST.cbList;

SmbPutUshort(&FeaList->cbList, PTR_DIFF_SHORT(fea, FeaList));
```

Win7

	HIDWORD	LODWORD
Attacker	0001	0000
Valid Size		
Vuln Size		
NT Buffer Size		

Bug #1 - Integer Cast Error

```
ULONG FEALIST.cbList;

SmbPutUshort(&FeaList->cbList, PTR_DIFF_SHORT(fea, FeaList));
```

Win7

	HIDWORD	LODWORD
Attacker	0001	0000
Valid Size	0000	ff5d
Vuln Size		
NT Buffer Size		

Bug #1 - Integer Cast Error

```
ULONG FEALIST.cbList;

SmbPutUshort(&FeaList->cbList, PTR_DIFF_SHORT(fea, FeaList));
```

Win7

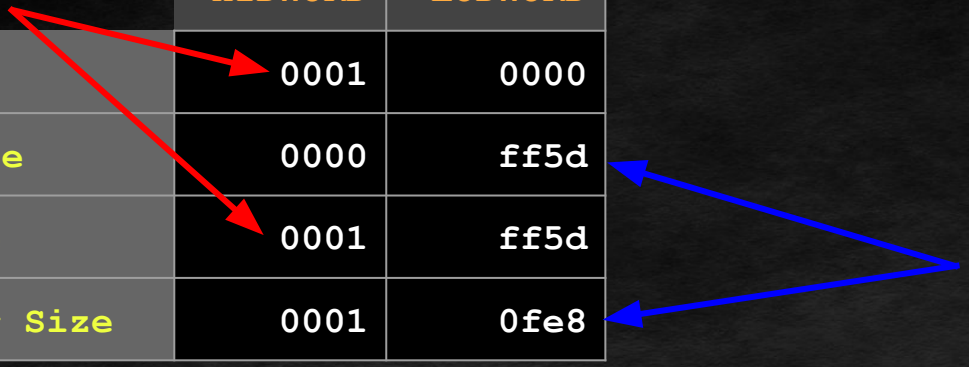
	HIDWORD	LODWORD
Attacker	0001	0000
Valid Size	0000	ff5d
Vuln Size	0001	ff5d
NT Buffer Size		

Bug #1 - Integer Cast Error

```
ULONG FEALIST.cbList;

SmbPutUshort(&FeaList->cbList, PTR_DIFF_SHORT(fea, FeaList));
```

Win7

	HIDWORD	LODWORD
Attacker	0001	0000
Valid Size	0000	ff5d
Vuln Size	0001	ff5d
NT Buffer Size	0001	0fe8



0x1ff5d (OS/2) > 0x10fe8 (NT)

Assembly Analysis

x86/x64

```
srv!SrvOs2FeaListSizeToNt+0x47:
b6bb3c53 2bcf            sub     ecx,edi
b6bb3c55 66890f          mov     word ptr [edi],cx
```

```
srv!SrvOs2FeaListSizeToNt+0x71:
fffff880`039d5931 662bde          sub     bx,si
fffff880`039d5934 66891e          mov     word ptr [rsi],bx
```

ARM

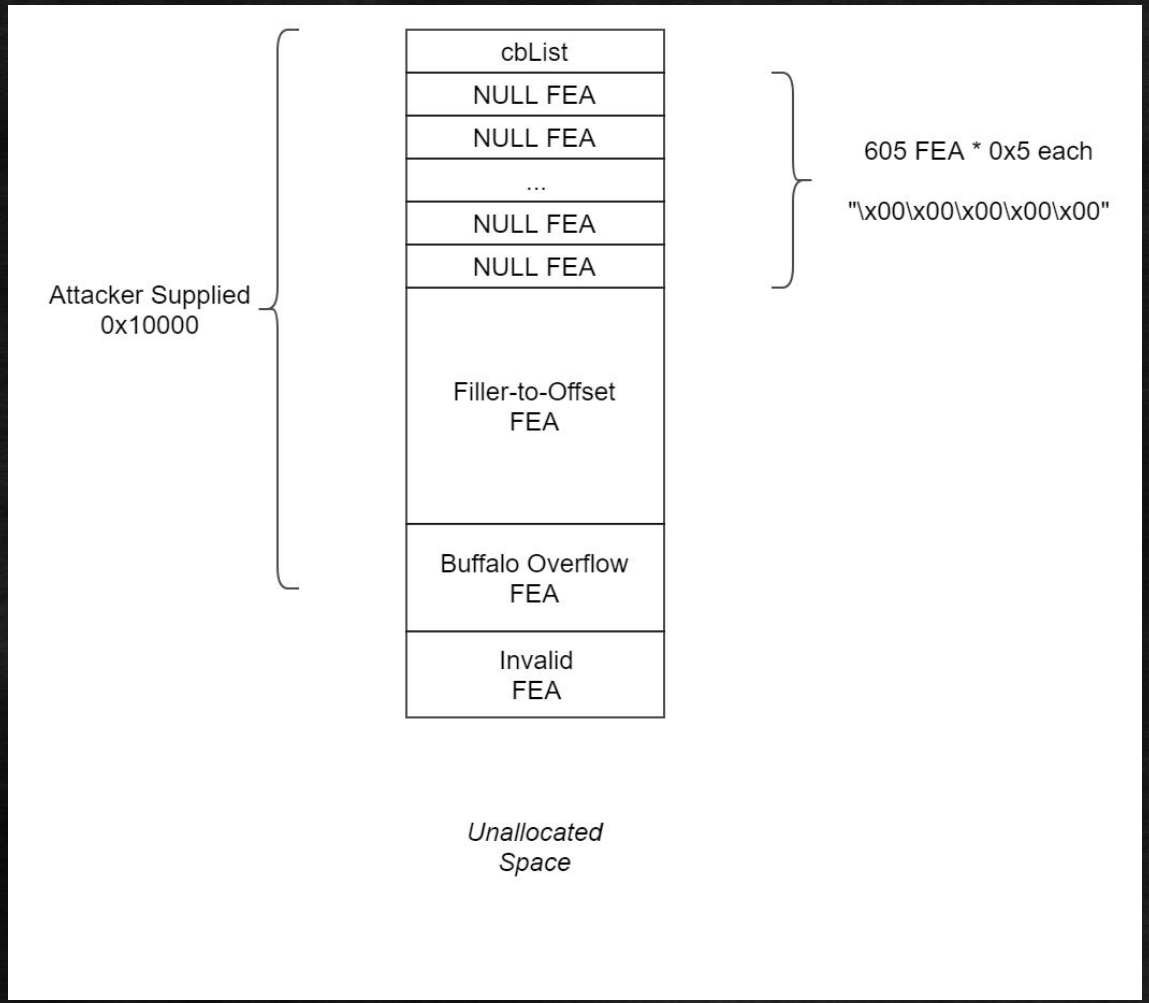
```
loc_2BC9E
SUBS            R3, R6, R5
STRH            R3, [R5]
```

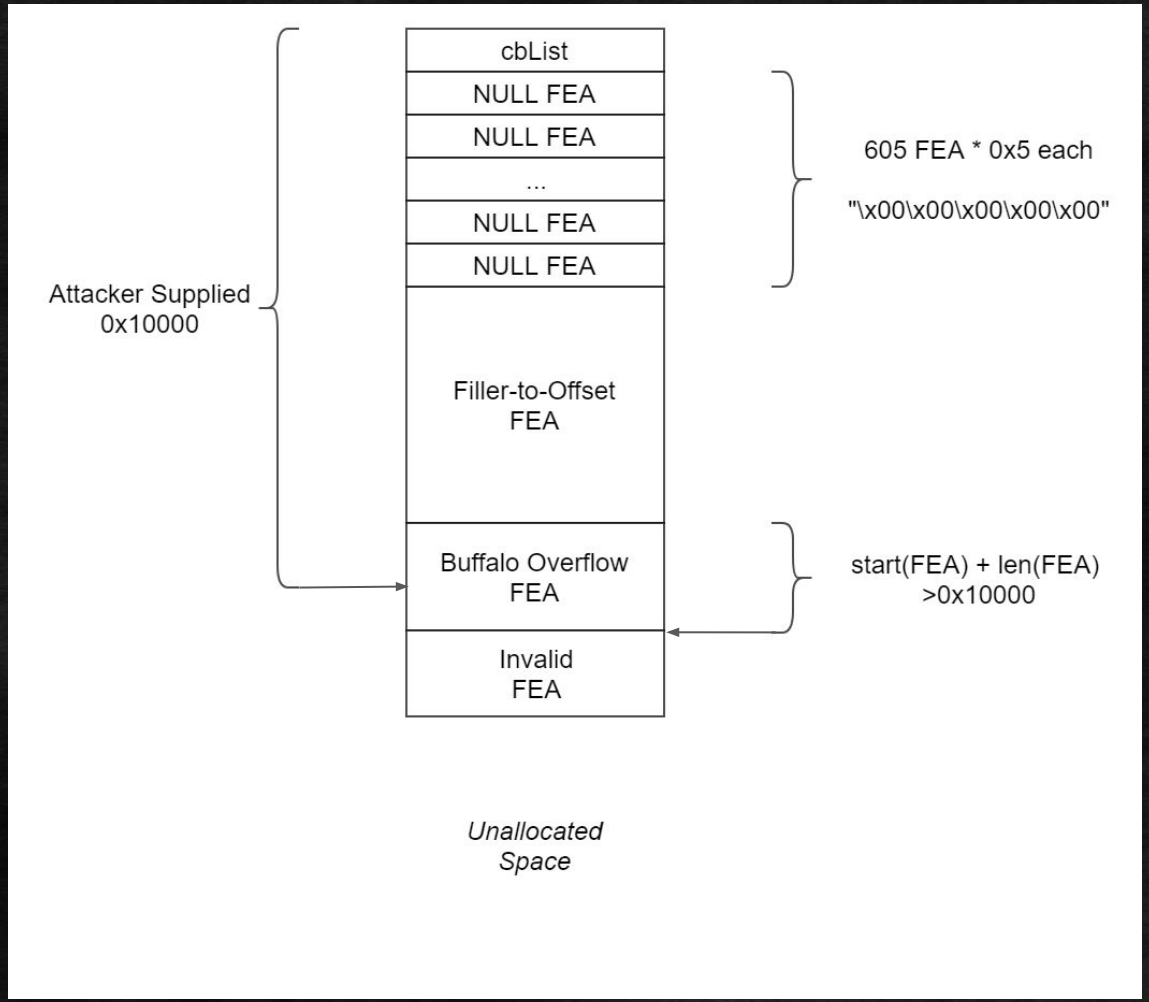
Itanium

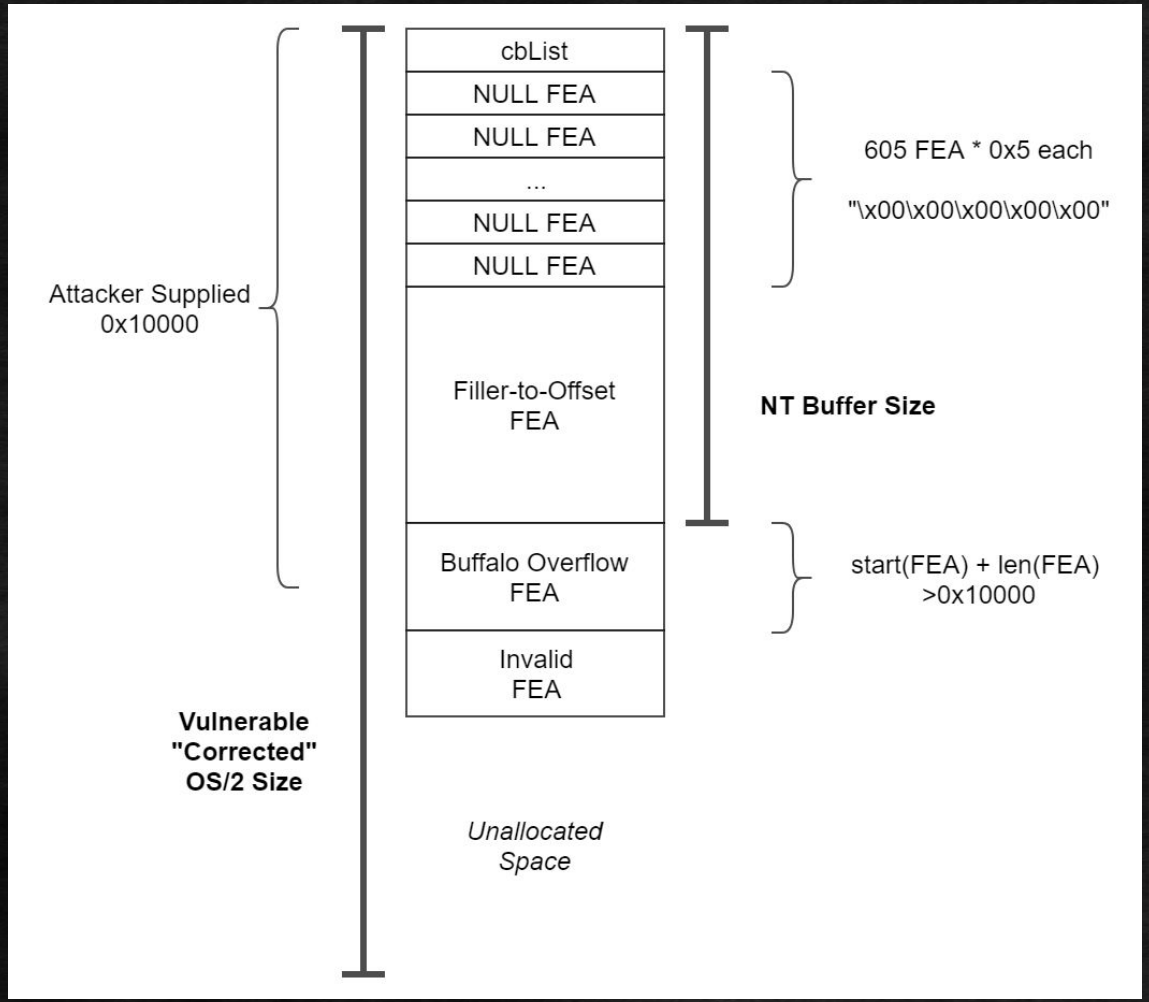
```
loc_A4100:
sub r21 = r35, r32
adds r20 = 1, r32;;
shr.u r19 = r21, 8
st1 [r32] = r21;;
st1 [r20] = r19
nop.i 0;;
```

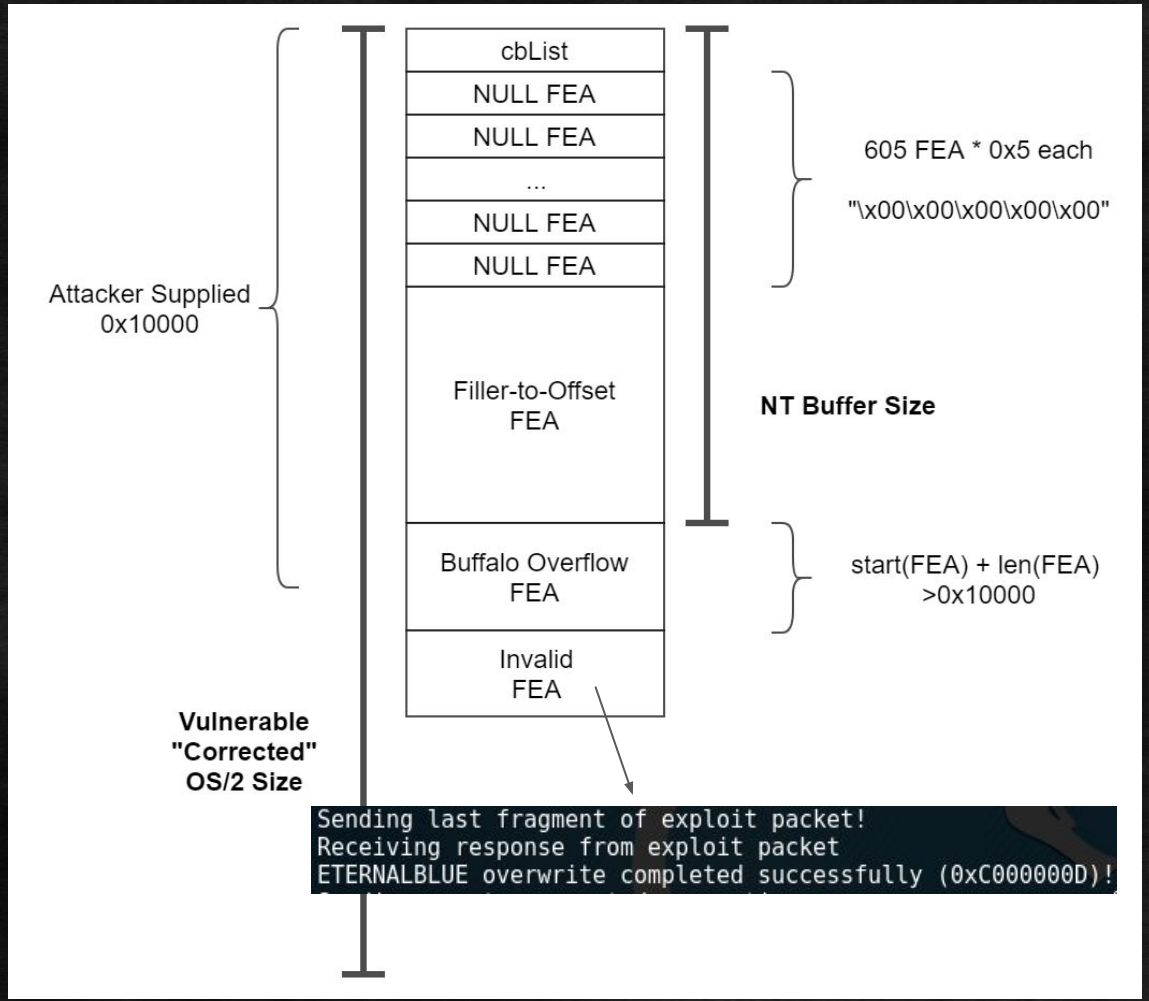
DEC ALPHA

```
loc_5EBC0:
andnot  $11, 2, $17
subl    $13, $9, $9
ldl     $19, 0($17)
and     $11, 2, $18
inswl   $9, $18, $20
mskwl   $19, $18, $19
```

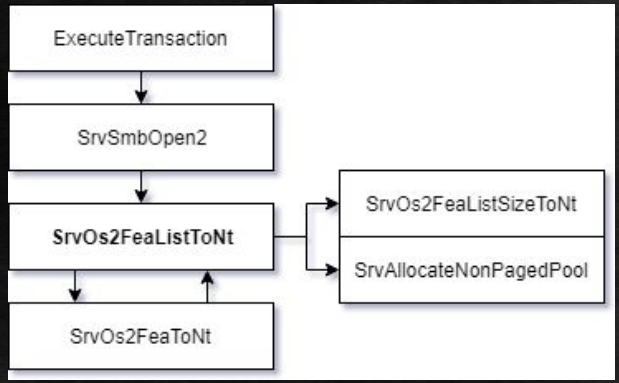

cbList
NULL FEA
NULL FEA
...
NULL FEA
NULL FEA
Filler-to-Offset FEA
Buffalo Overflow FEA
Invalid FEA
Attacker Supplied 0x10000
605 FEA * 0x5 each
"\x00\x00\x00\x00\x00"
Unallocated Space


cbList
NULL FEA
NULL FEA
...
NULL FEA
NULL FEA
Attacker Supplied
0x10000
Filler-to-Offset
FEA
Buffalo Overflow
FEA
Invalid
FEA
605 FEA * 0x5 each
"\x00\x00\x00\x00\x00"
start(FEA) + len(FEA)
>0x10000
Unallocated
Space


cbList
NULL FEA
NULL FEA
...
NULL FEA
NULL FEA
Filler-to-Offset FEA
Buffalo Overflow FEA
Invalid FEA
Attacker Supplied 0x10000
Vulnerable "Corrected" OS/2 Size
Unallocated Space
605 FEA * 0x5 each
"\x00\x00\x00\x00\x00"
NT Buffer Size
start(FEA) + len(FEA) >0x10000

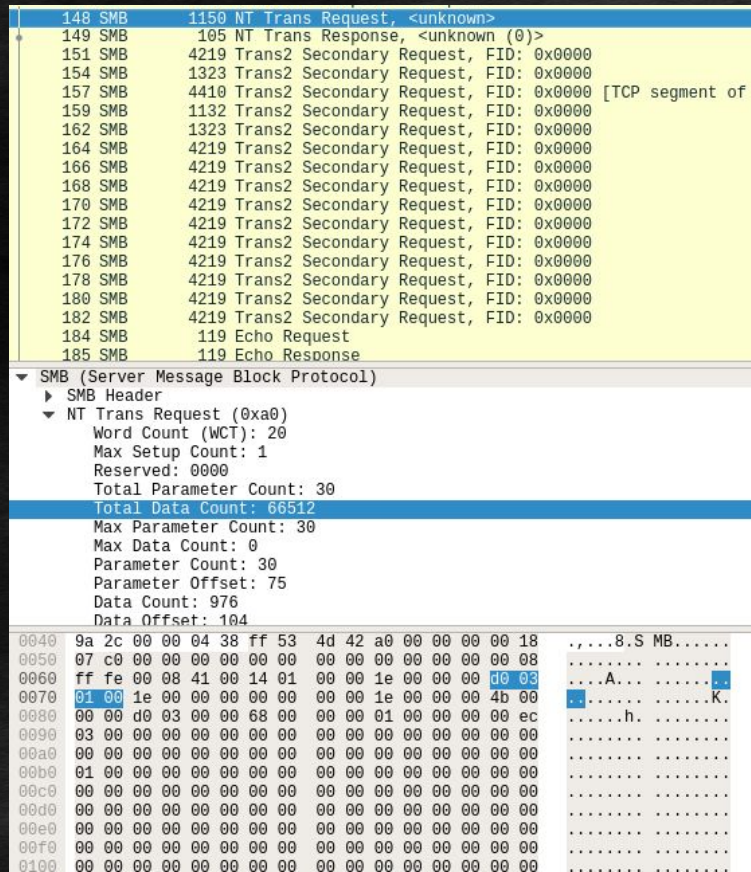

cbList
NULL FEA
NULL FEA
...
NULL FEA
NULL FEA
Filler-to-Offset FEA
Buffalo Overflow FEA
Invalid FEA
Attacker Supplied 0x10000
605 FEA * 0x5 each
"\x00\x00\x00\x00\x00"
NT Buffer Size
start(FEA) + len(FEA) >0x10000
Vulnerable "Corrected" OS/2 Size
Sending last fragment of exploit packet!
Receiving response from exploit packet
ETERNALBLUE overwrite completed successfully (0xC000000D)!

```
packet Trans2_Open2_Parameters
{
    USHORT                  Flags;
    USHORT                  AccessMode;
    USHORT                  Reserved1;
    SMB_FILE_ATTRIBUTES     FileAttributes;
    UTIME                   CreationTime;
    USHORT                  OpenMode;
    ULONG                   AllocationSize;
    USHORT                  Reserved[5];
    SMB_STRING              FileName;
};
packet Trans2_Open2_Data
{
    SMB_FEA_LIST ExtendedAttributeList;
};
```



Bug #2 - Oversized Trans/Trans2 Requests

- Need to send > **WORD** data
  - Bug trigger 0x10000 > 0xffff
- Trans2_Open2 is **WORD**
  - NT Trans allows **DWORD**!
- Can trick transaction dispatch tables
  - They all become generic _TRANSACTION
  - Primary transaction type doesn't matter
    - Final Secondary transaction



Bug #3 - Session Setup Allocation Error

- NT Security vs. Extended Security
  - 13 words vs. 12 words
- Certain flag values can confuse it
  - Reads SMB_DATA_BLOCK size at wrong offset
  - Can reserve large memory
    - Same pool tag as FEA: LSbf
- Free on demand
  - Close client socket
- Not really a "vuln" itself
  - Still in master branch

Scenario	WordCount	Capabilities*	Flags2†	Deduced Type
1	12	✓	✓	Extended Security
2	12	✓	x	NT Security
3	12	x	✓	Invalid
4	12	x	x	Invalid
5	13	✓	✓	Extended Security
6	13	✓	x	NT Security
7	13	x	✓	NT Security
8	13	x	x	NT Security

* CAP_EXTENDED_SECURITY

† FLAGS2_EXTENDED_SECURITY

EternalBlue NonPagedPool Ingredients

- FEALIST overflow
  - Exploit
- Session Setup bug
  - Allocation
  - Hole
- SrvNet.sys network buffers
  - Primary Grooms
  - Secondary Grooms
  - FAKE SMB2
    - IDS bypass?

EternalBlue Grooming

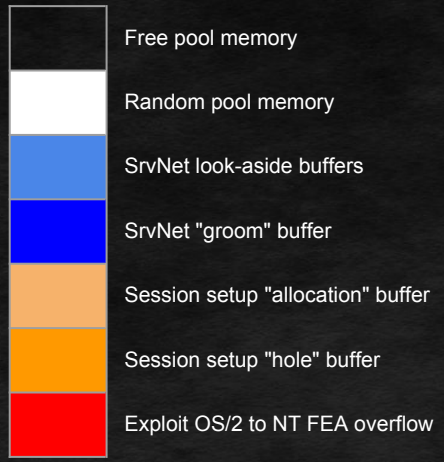
- Step 0. Pre-Exploitation Memory Layout
  - SrvNet has lookaside memory, random stuff is in the pool

EternalBlue Grooming

- Step 1. Send all of FEALIST except last Trans2 secondary
  - The NT FEA Buffer will not be reserved yet

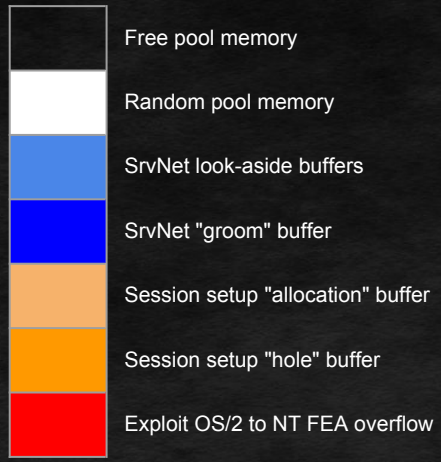
EternalBlue Grooming

- Step 2. Send initial *N* grooms
  - Use up all of SrvNet look-aside, forcing new pool allocations



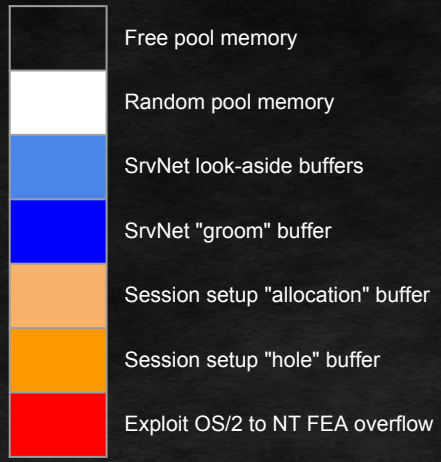
EternalBlue Grooming

- Step 2. Send initial *N* grooms
  - Use up all of SrvNet look-aside, forcing new pool allocations



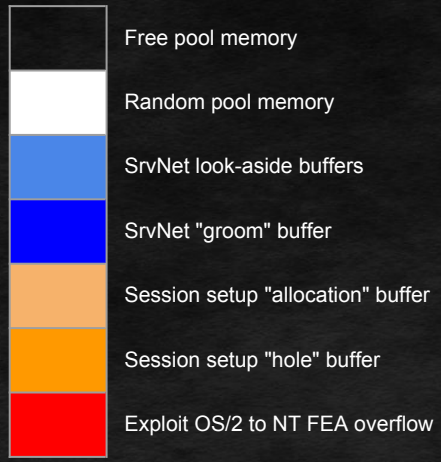
EternalBlue Grooming

- Step 2. Send initial *N* grooms
  - Use up all of SrvNet look-aside, forcing new pool allocations



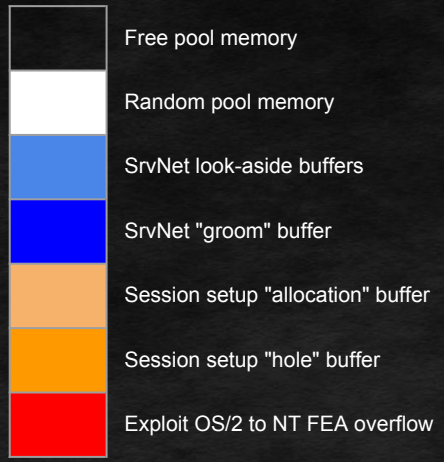
EternalBlue Grooming

- Step 2. Send initial *N* grooms
  - Use up all of SrvNet look-aside, forcing new pool allocations



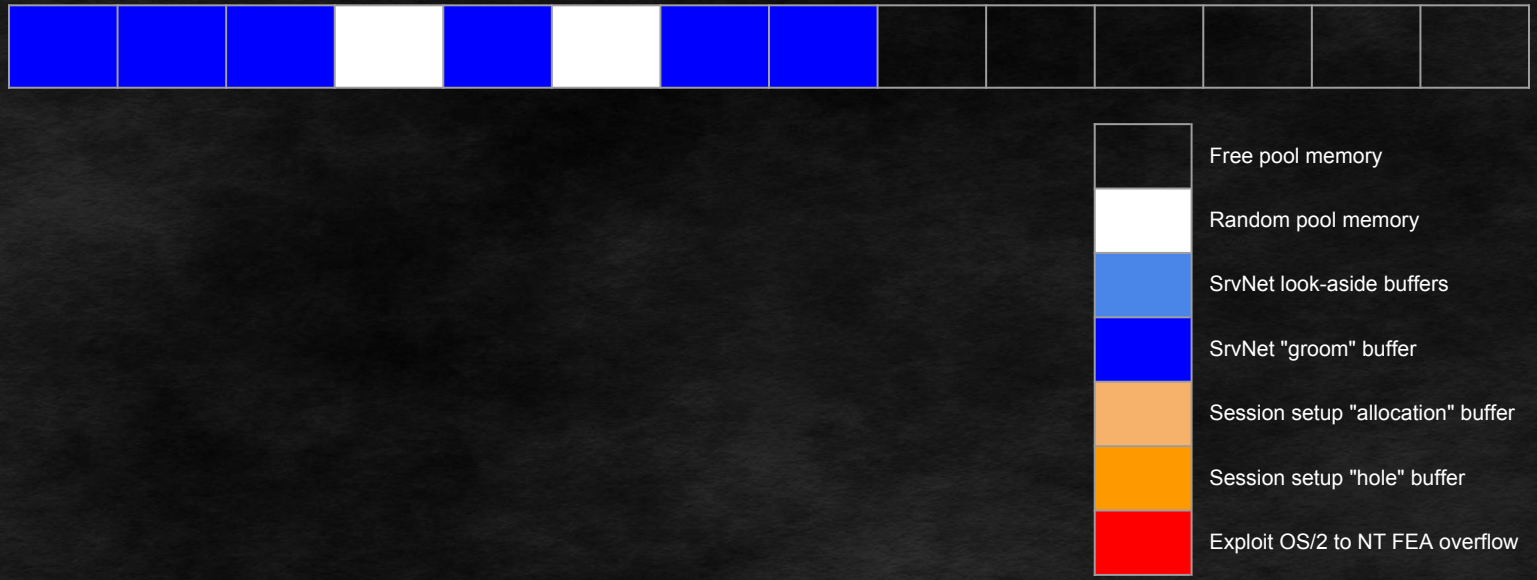
EternalBlue Grooming

- Step 2. Send initial *N* grooms
  - Use up all of SrvNet look-aside, forcing new pool allocations



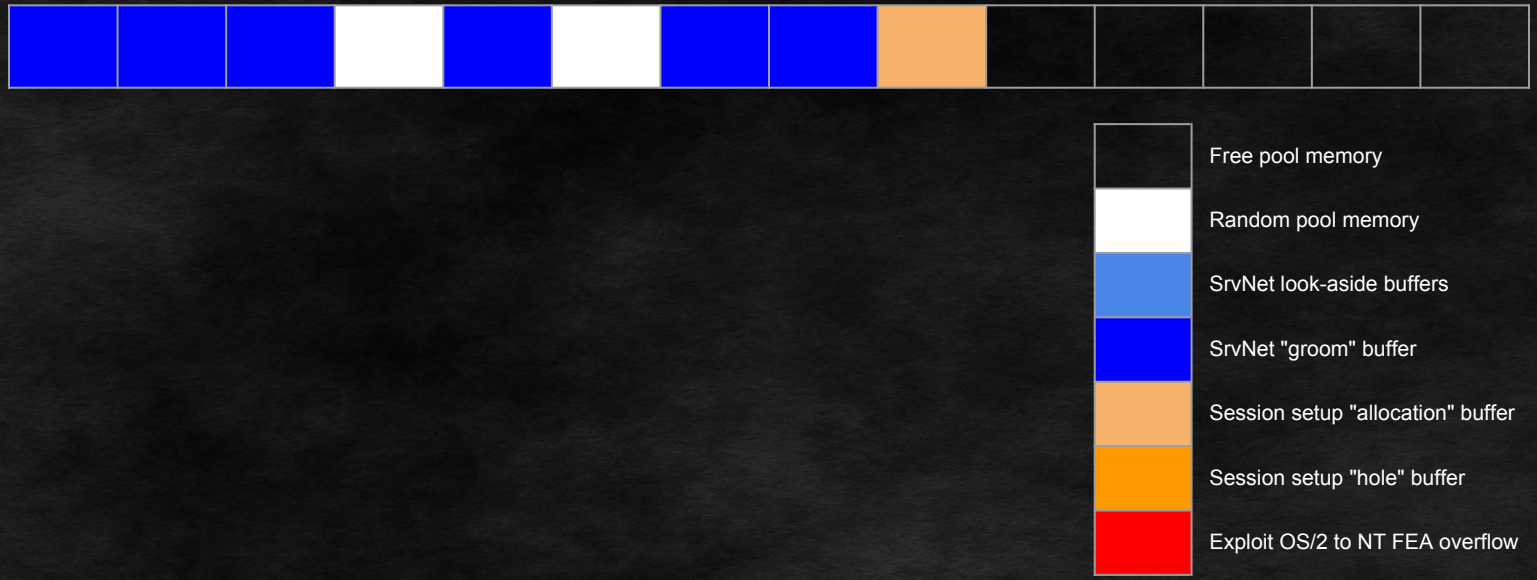
EternalBlue Grooming

- Step 2. Send initial *N* grooms
  - Use up all of SrvNet look-aside, forcing new pool allocations



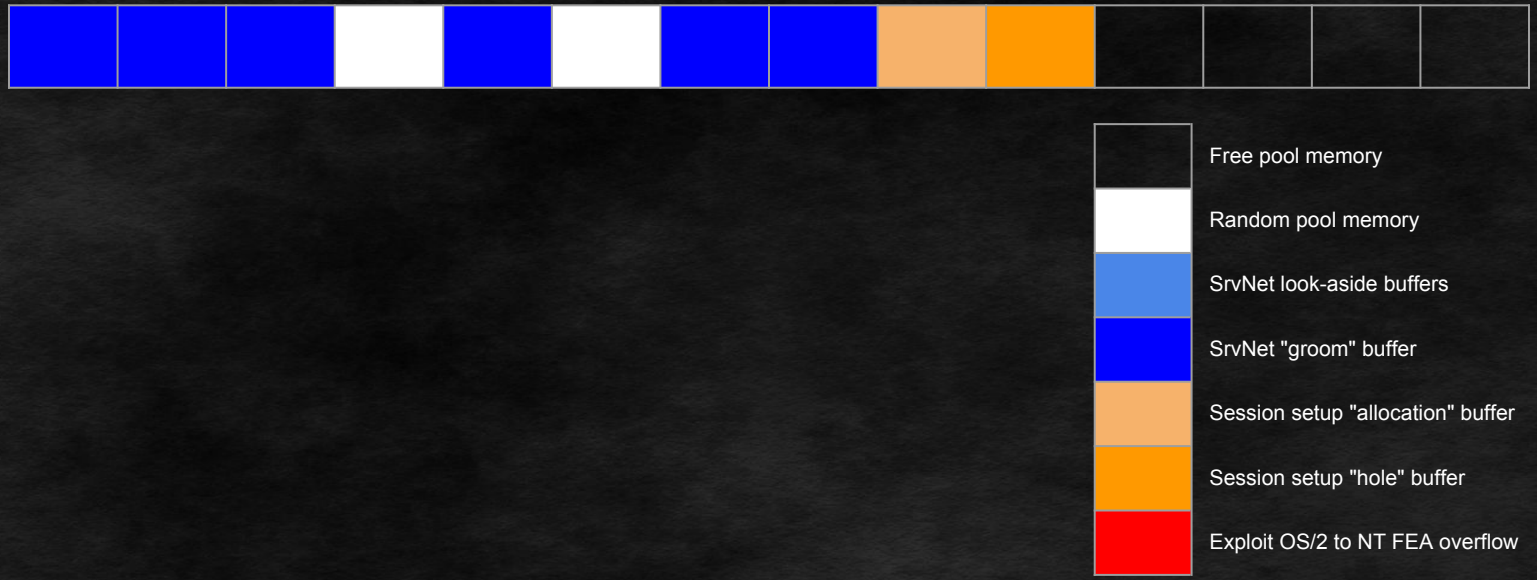
EternalBlue Grooming

- Step 3. Send allocation connection
  - Session Setup bug SMALLER than NT FEA Buffer Size



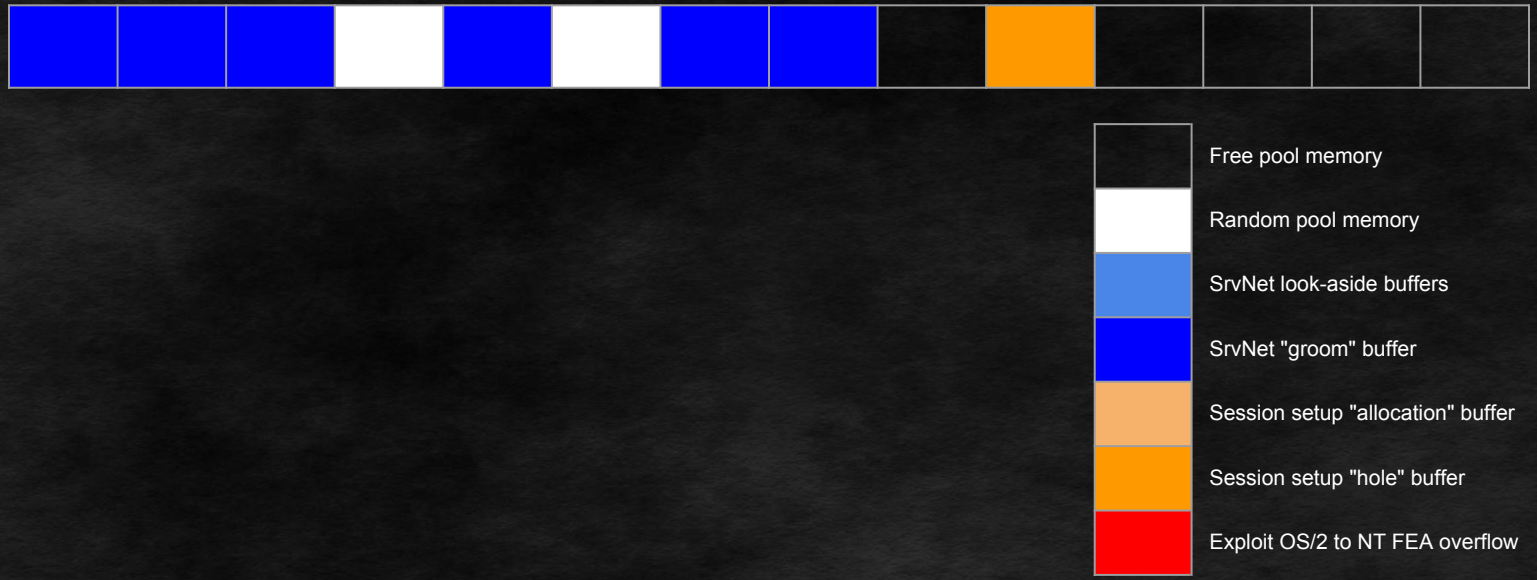
EternalBlue Grooming

- Step 4. Send hole buffer connection
  - Session Setup bug SAME SIZE as NT FEA Buffer Size



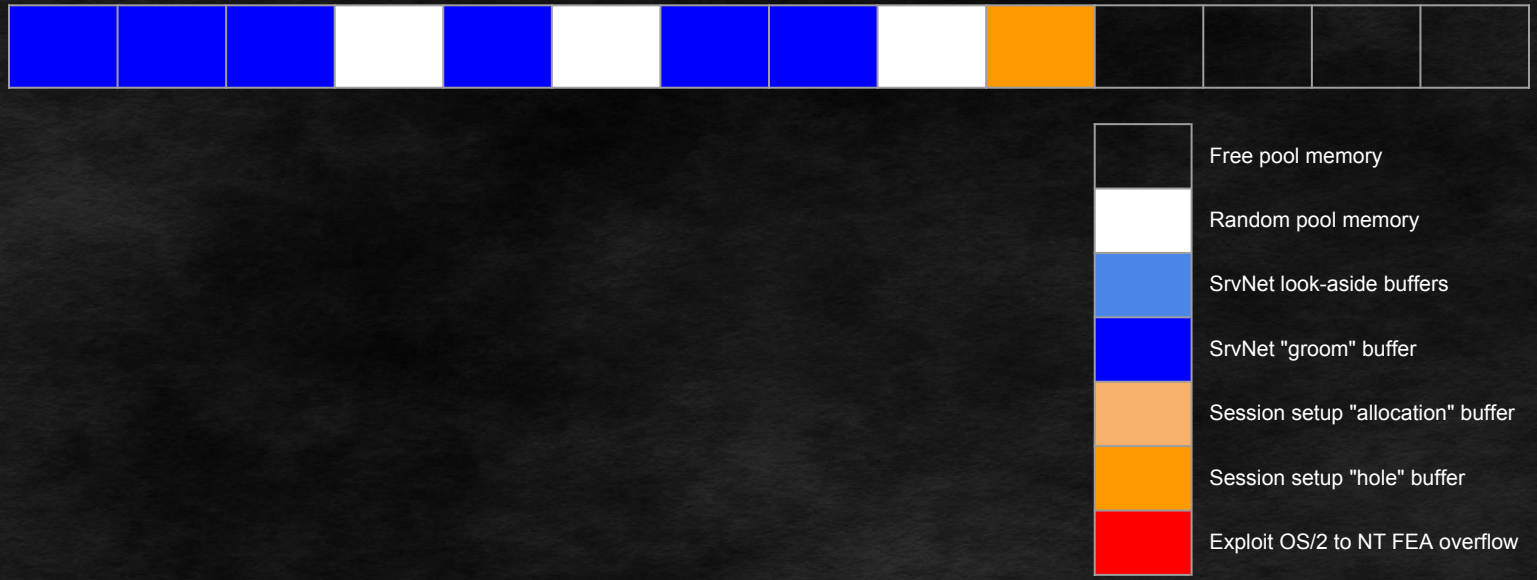
EternalBlue Grooming

- Step 5. Close allocation connection
  - Memory slot can now hold smaller miscellaneous allocations



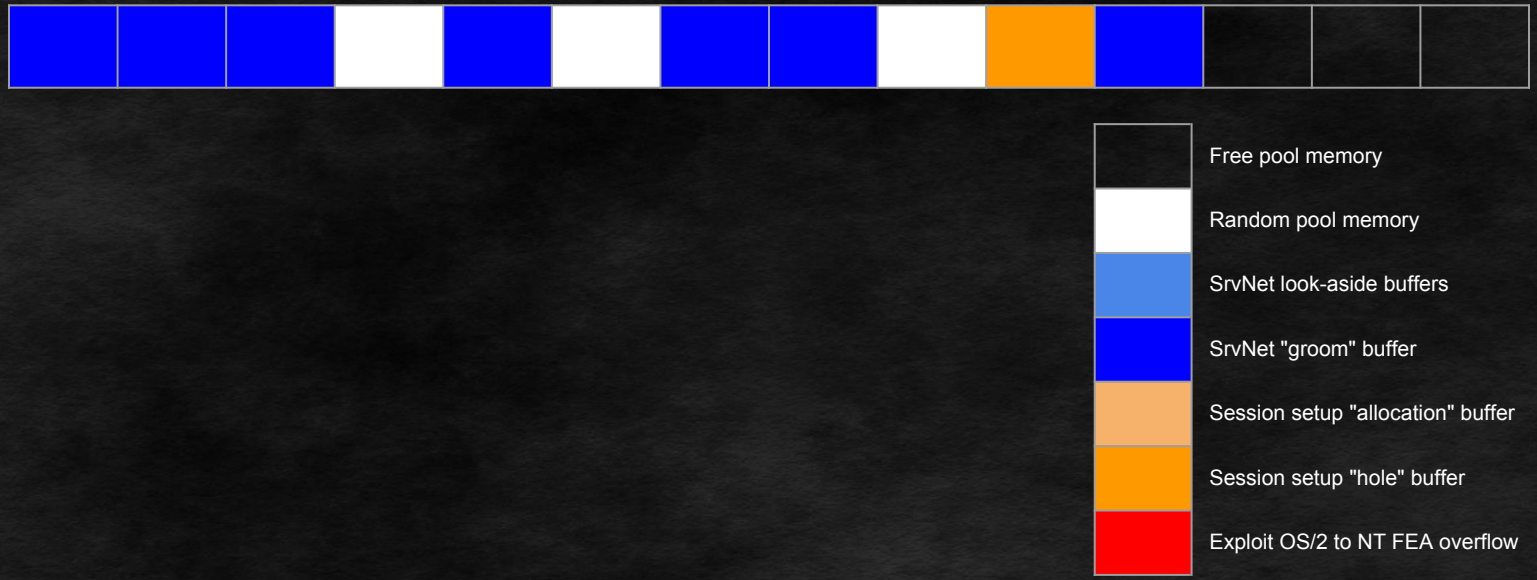
EternalBlue Grooming

- Step 5. Close allocation connection
  - Memory slot can now hold smaller miscellaneous allocations



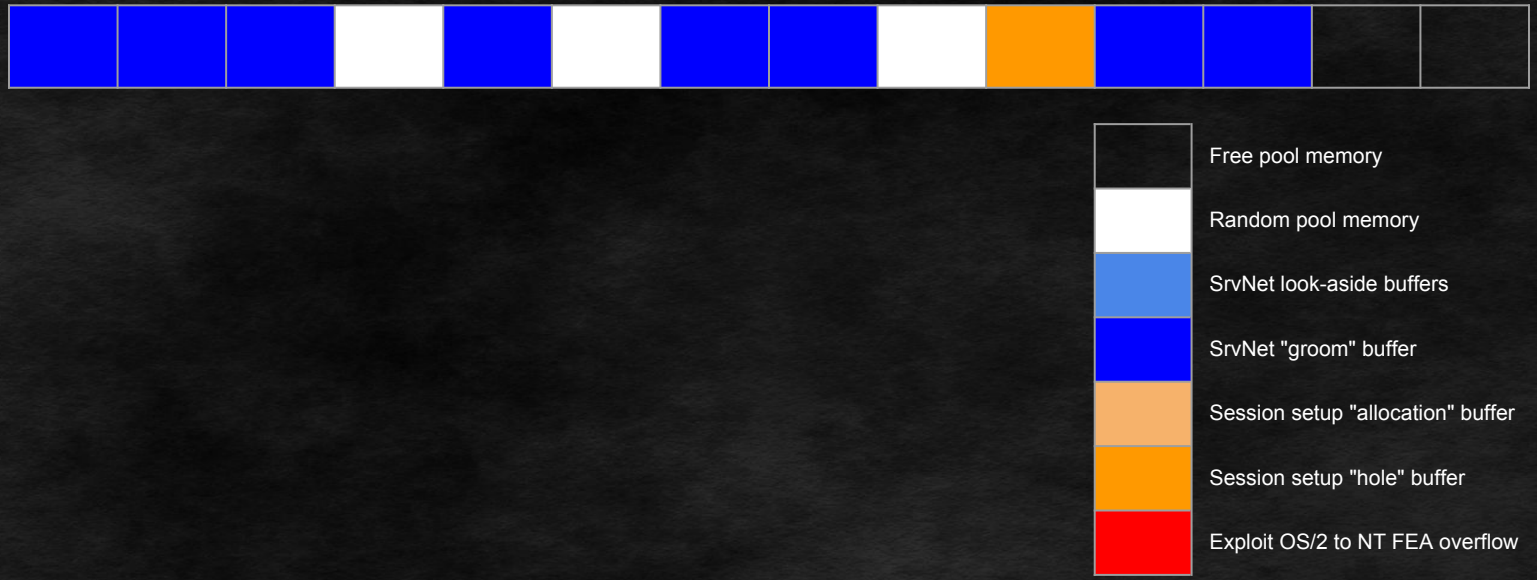
EternalBlue Grooming

- Step 6. Send final groom packets
  - Hopefully a groom is after the Hole buffer



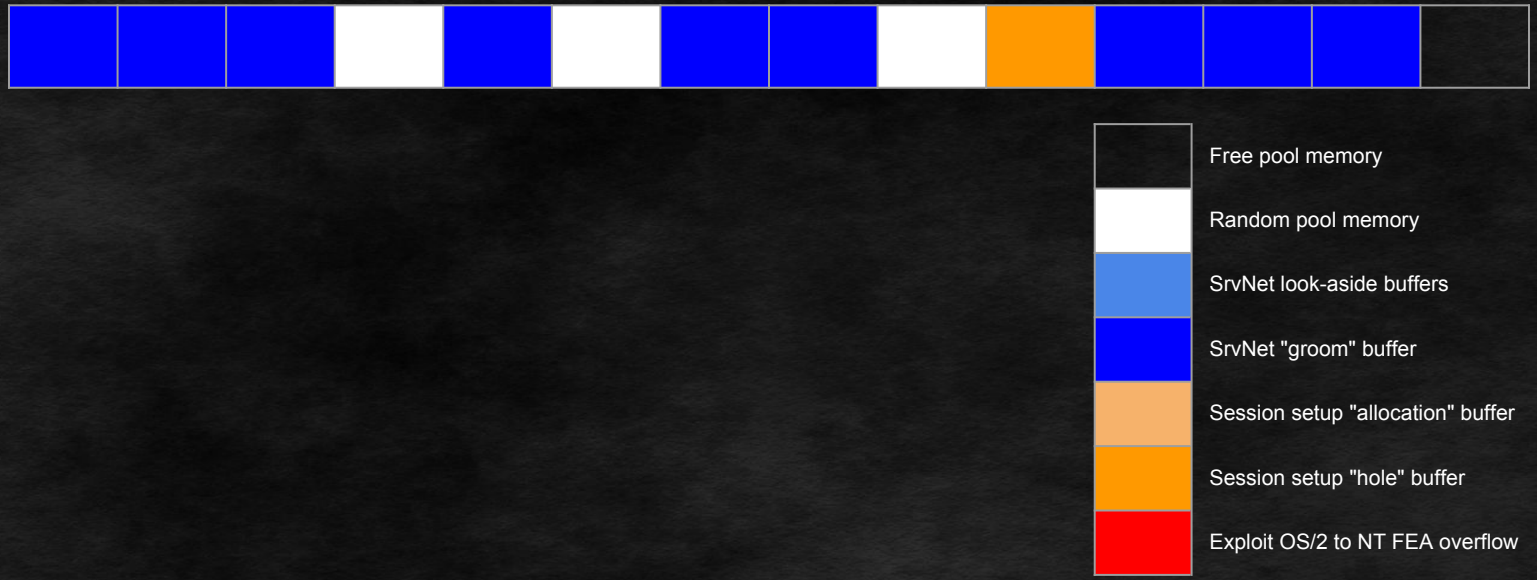
EternalBlue Grooming

- Step 6. Send final groom packets
  - Hopefully a groom is after the Hole buffer



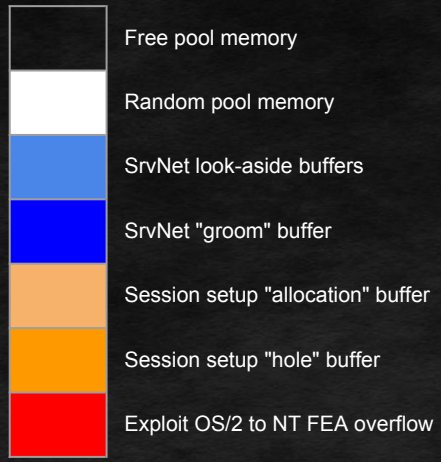
EternalBlue Grooming

- Step 6. Send final groom packets
  - Hopefully a groom is after the Hole buffer



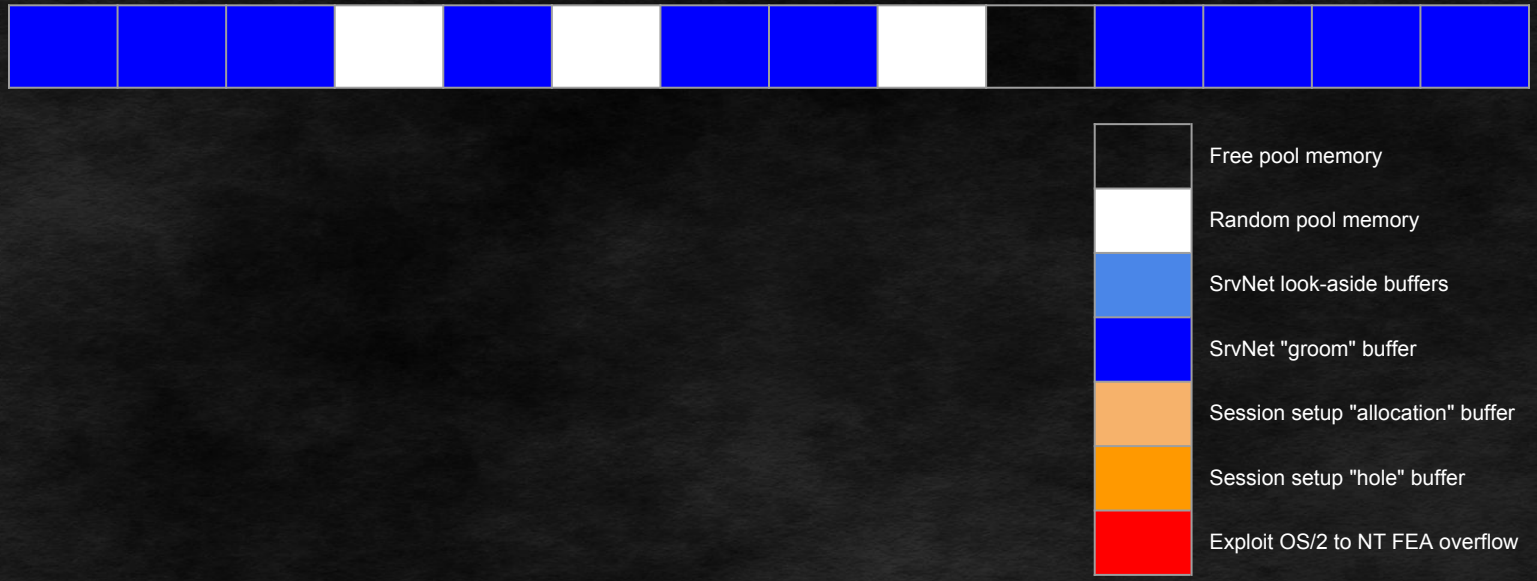
EternalBlue Grooming

- Step 6. Send final groom packets
  - Hopefully a groom is after the Hole buffer



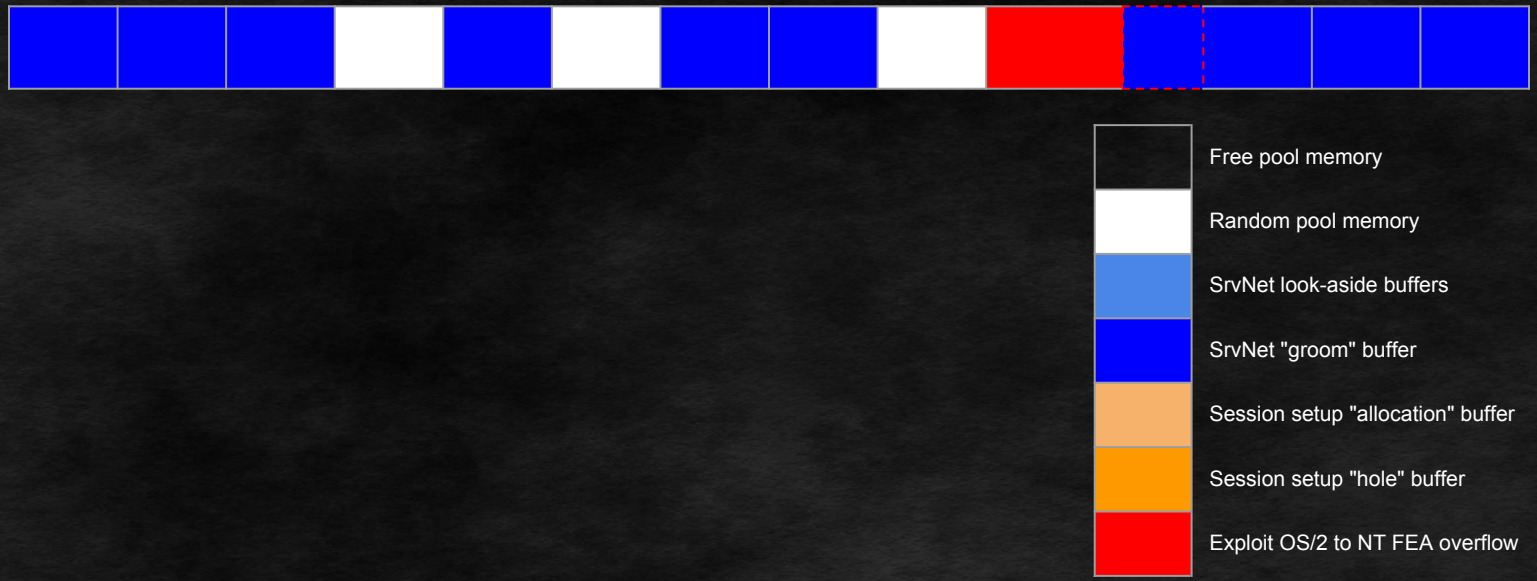
EternalBlue Grooming

- Step 6. Close Hole connection
  - Memory the same size as NT FEA Buffer is now available

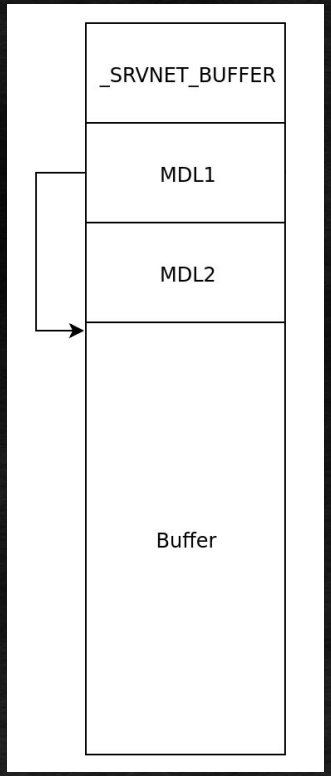


EternalBlue Grooming

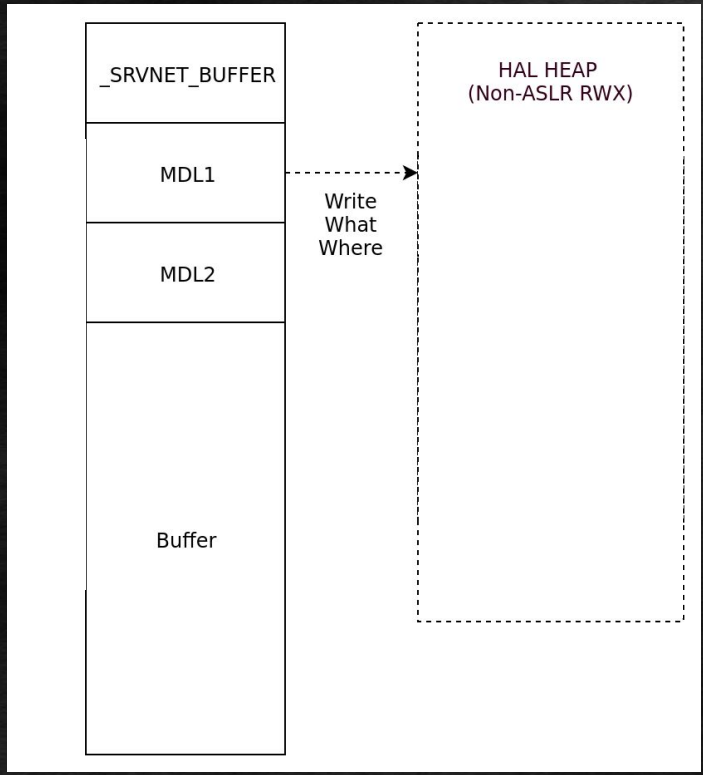
- Step 7. Send final FEALIST exploit fragment
  - Erroneously calculated to fit in the free Hole buffer, overflows into groom



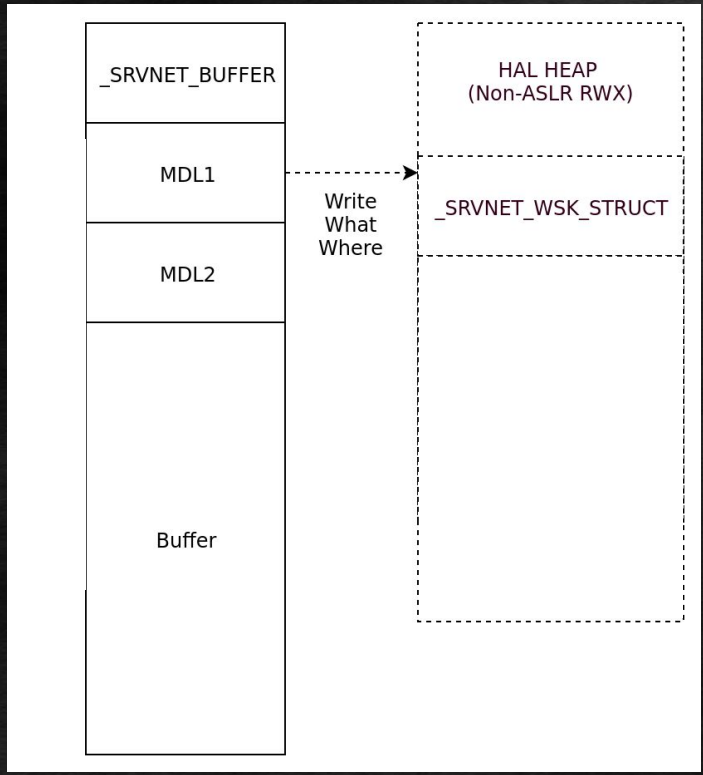
```
struct _SRVNET_BUFFER
{
    // ...

    SRVNET_WSK_STRUCT*   WskContext;

    // ...

    MDL     MDL1; // MapSysVa = &Buffer
    MDL     MDL2;
    CHAR    Buffer[];
};
```



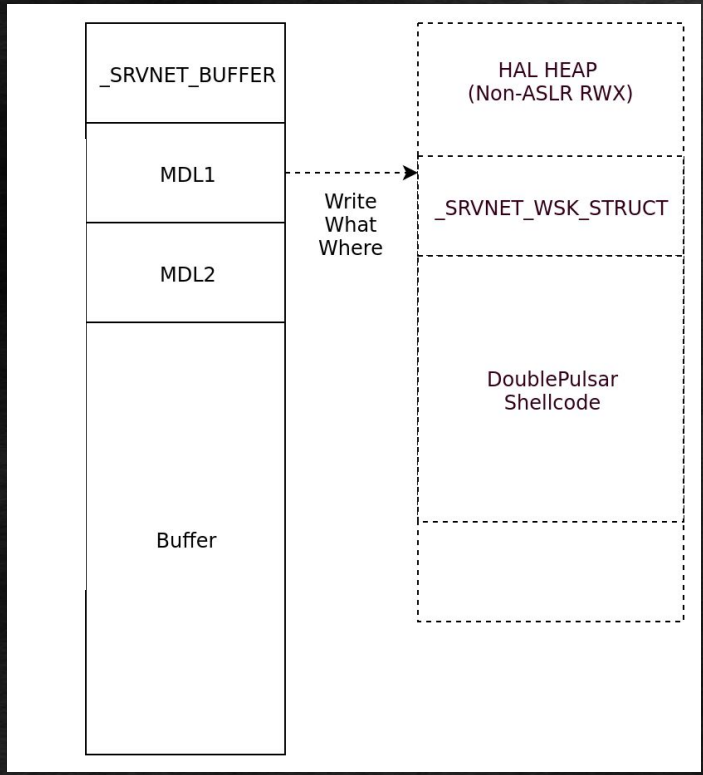
```
struct _SRVNET_BUFFER
{
    // ...

    SRVNET_WSK_STRUCT*   WskContext;

    // ...

    MDL     MDL1; // MapSysVa = &HAL
    MDL     MDL2;
    CHAR    Buffer[];
};
```



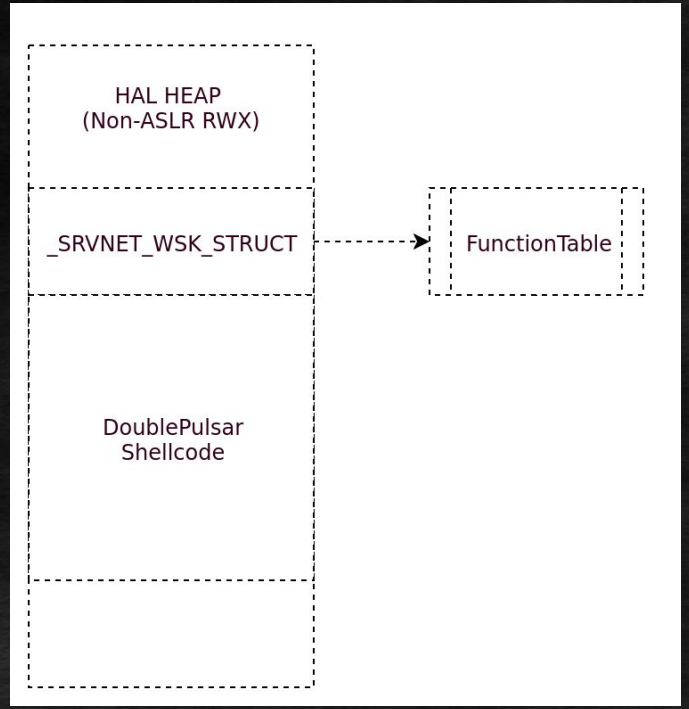
```
struct _SRVNET_BUFFER
{
    // ...

    SRVNET_WSK_STRUCT*   WskContext;

    // ...

    MDL     MDL1; // MapSysVa = &HAL
    MDL     MDL2;
    CHAR    Buffer[];
};
```



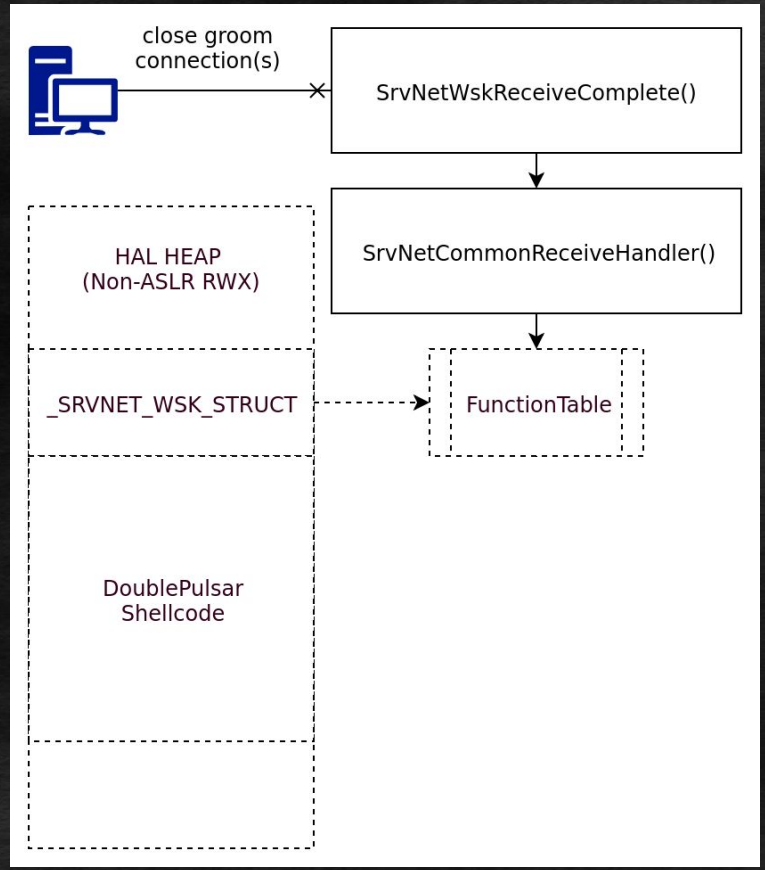
```
struct _SRVNET_BUFFER
{
    // ...

    SRVNET_WSK_STRUCT*   WskContext;

    // ...

    MDL     MDL1; // MapSysVa = &HAL
    MDL     MDL2;
    CHAR    Buffer[];
};
```



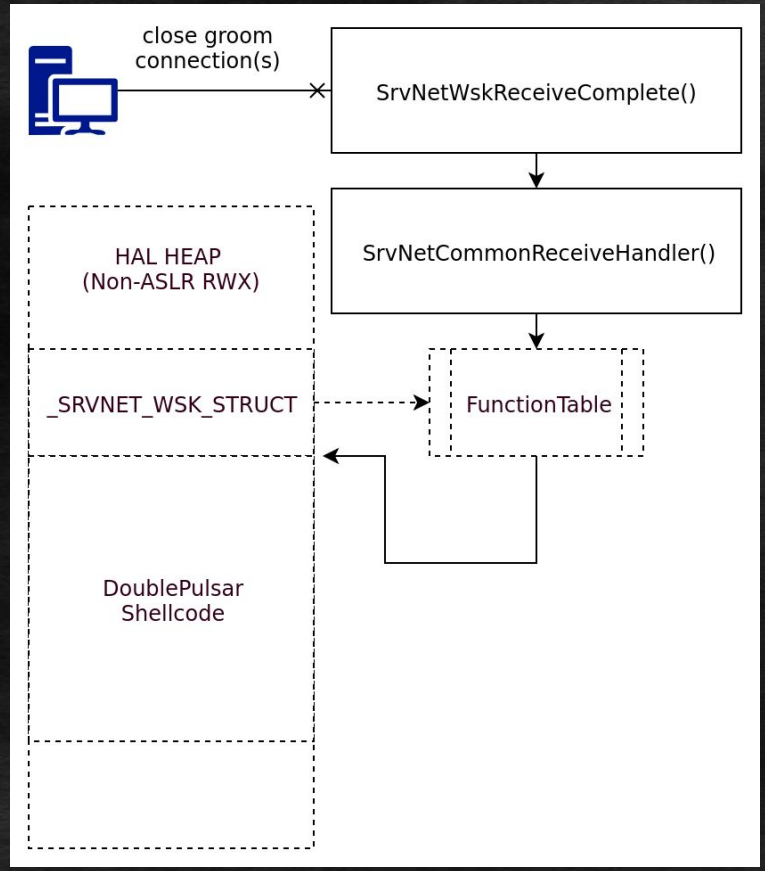
```
struct _SRVNET_WSK_STRUCT
{
    // ...

    PVOID           FunctionTable[];

    // ...
};
```



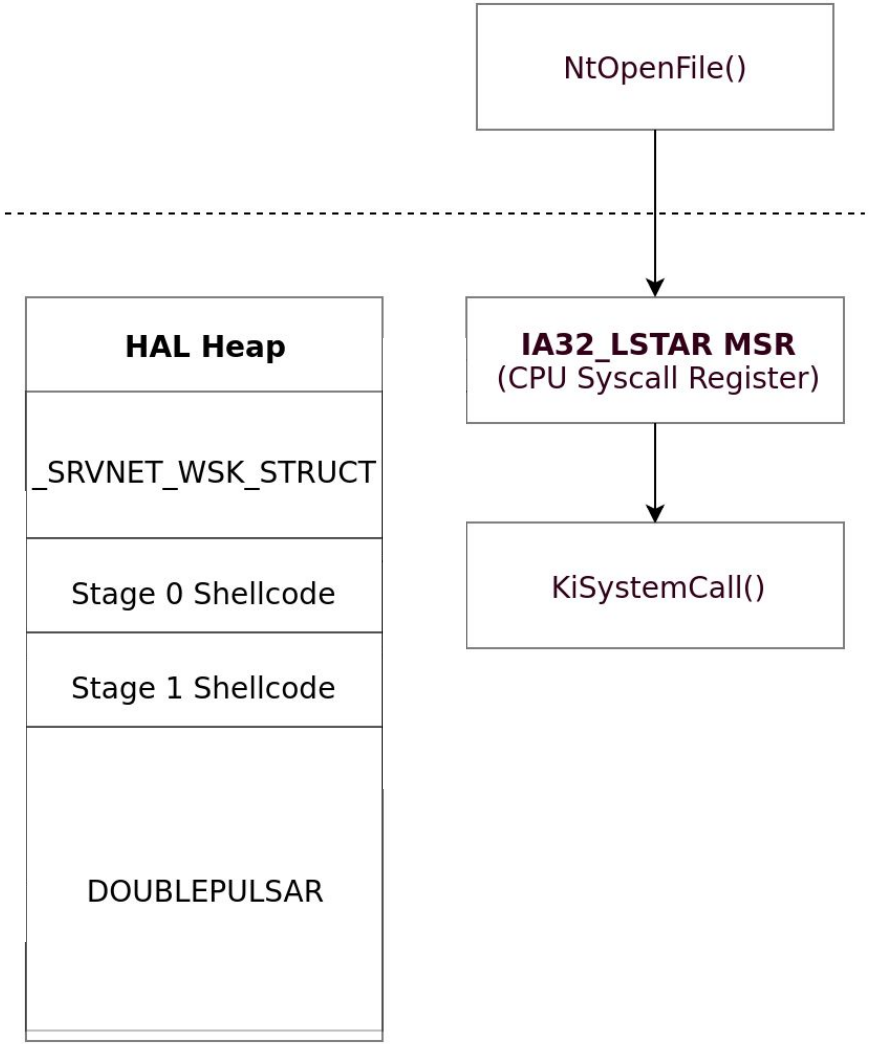
```
struct _SRVNET_WSK_STRUCT
{
    // ...

    PVOID           FunctionTable[];

    // ...
};
```



```
struct _SRVNET_WSK_STRUCT
{
    // ...

    PVOID           FunctionTable[];

    // ...
};
```

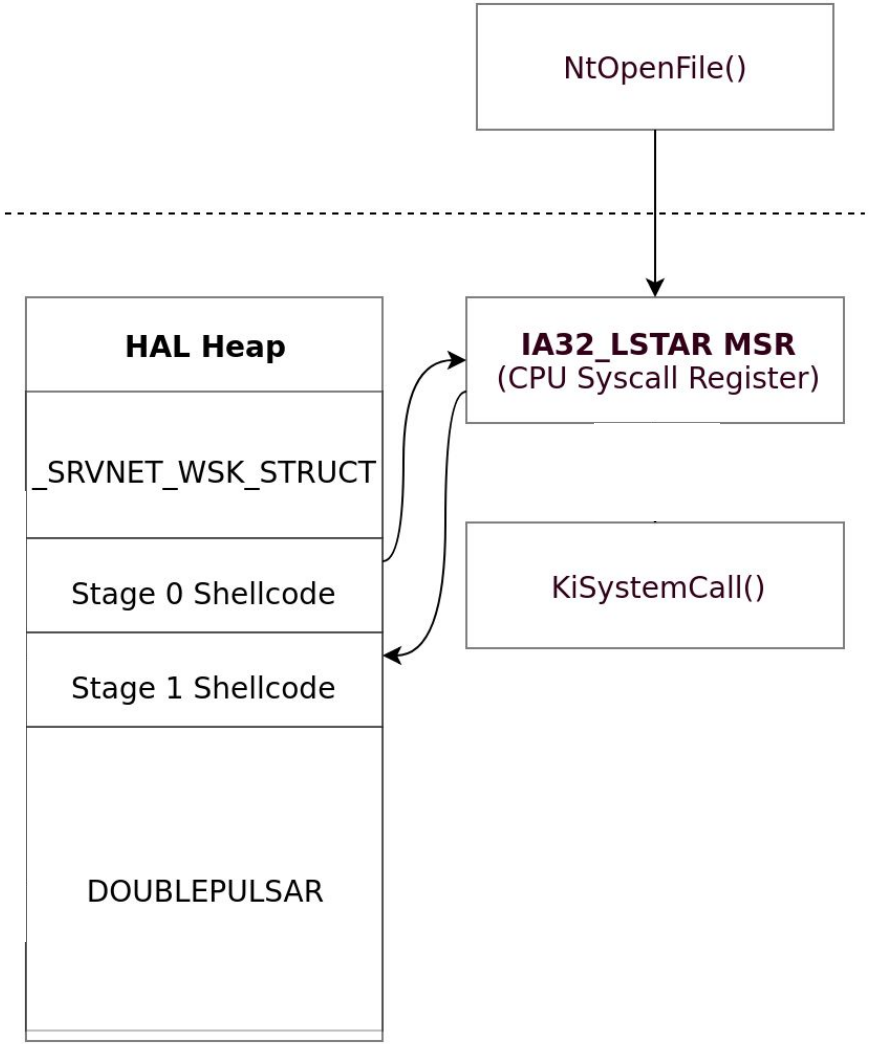


EternalBlue payload



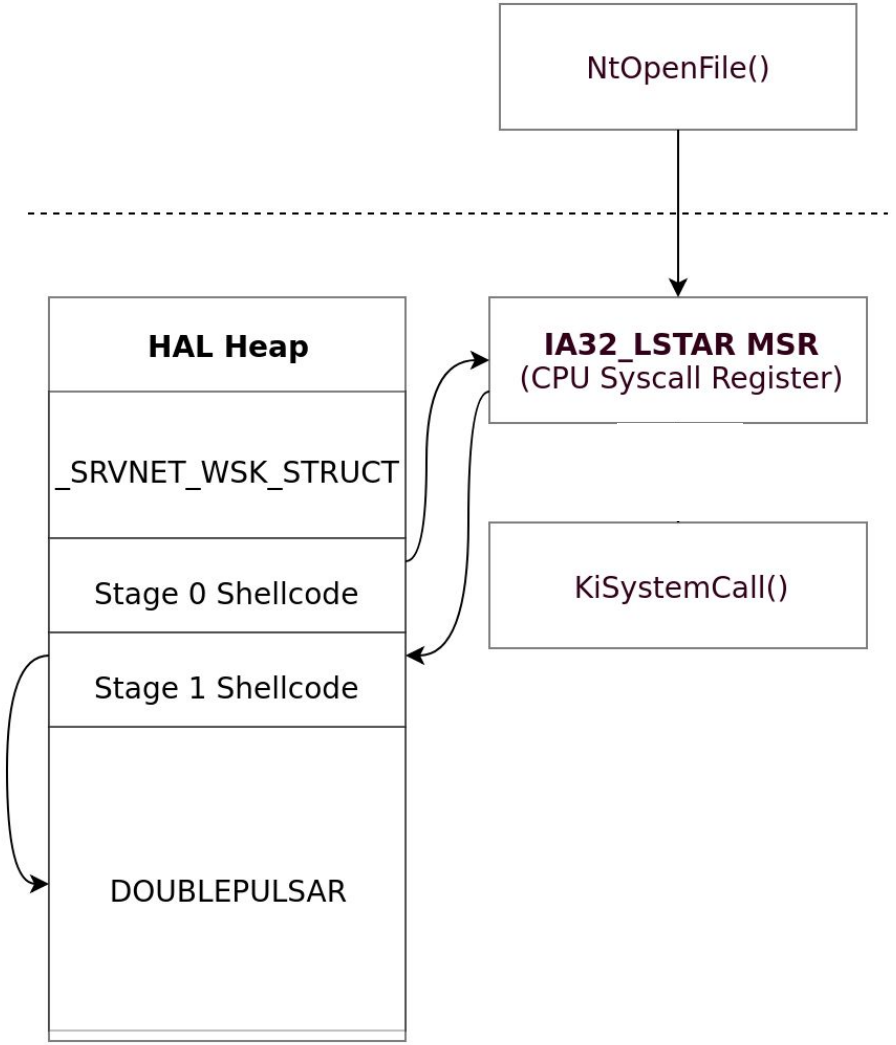
EternalBlue payload

1. Hook syscall handler
   - DISPATCH_LEVEL IRQL
     - Many routines are off limits



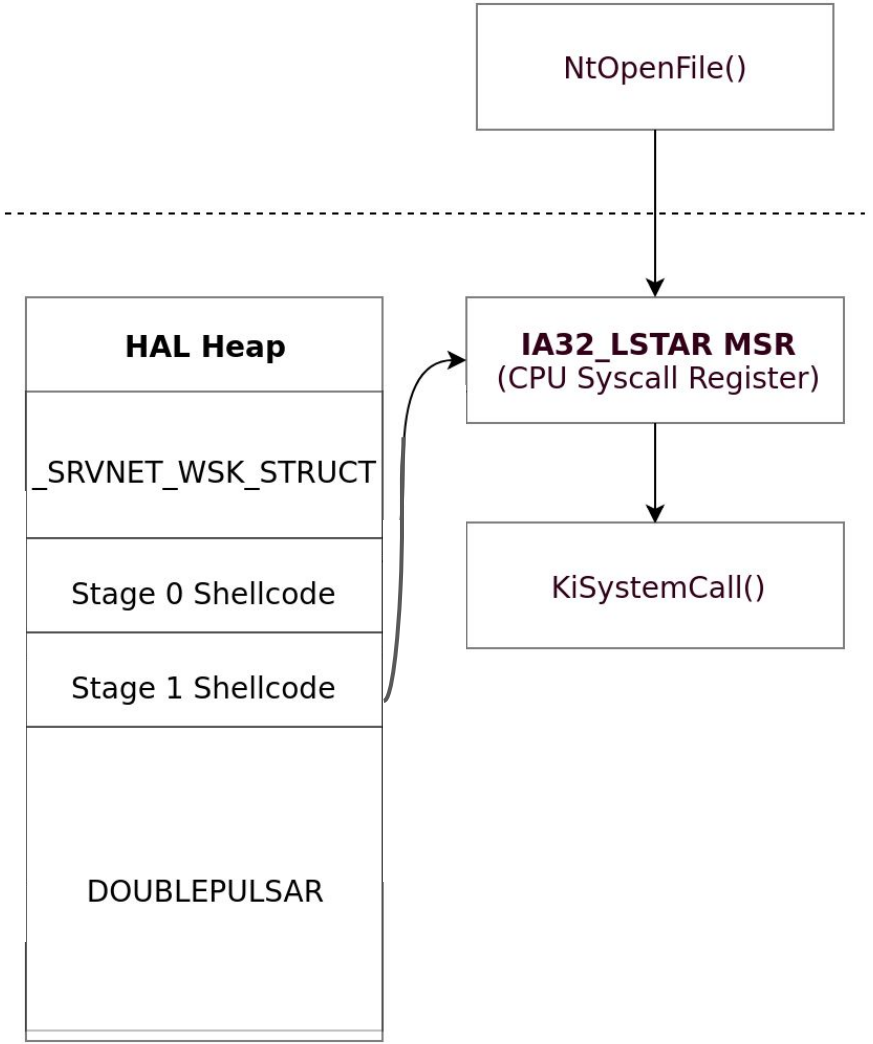
EternalBlue payload

1. Hook syscall handler
   - DISPATCH_LEVEL IRQL
     - Many routines are off limits
2. On next syscall…
   - Transition from user mode
   - Run DOUBLEPULSAR backdoor
     - SrvTransaction2DispatchTable



EternalBlue payload

1. Hook syscall handler
   - DISPATCH_LEVEL IRQL
     - Many routines are off limits
2. On next syscall…
   - Transition from user mode
   - Run DOUBLEPULSAR backdoor
     - SrvTransaction2DispatchTable
3. Restore syscall handler



EternalBlue Patch

SrvOs2FeaListSizeToNt():

```
SmbPutUshort(&FeaList->cbList, PTR_DIFF_SHORT(fea, FeaList));
```

EternalBlue Patch

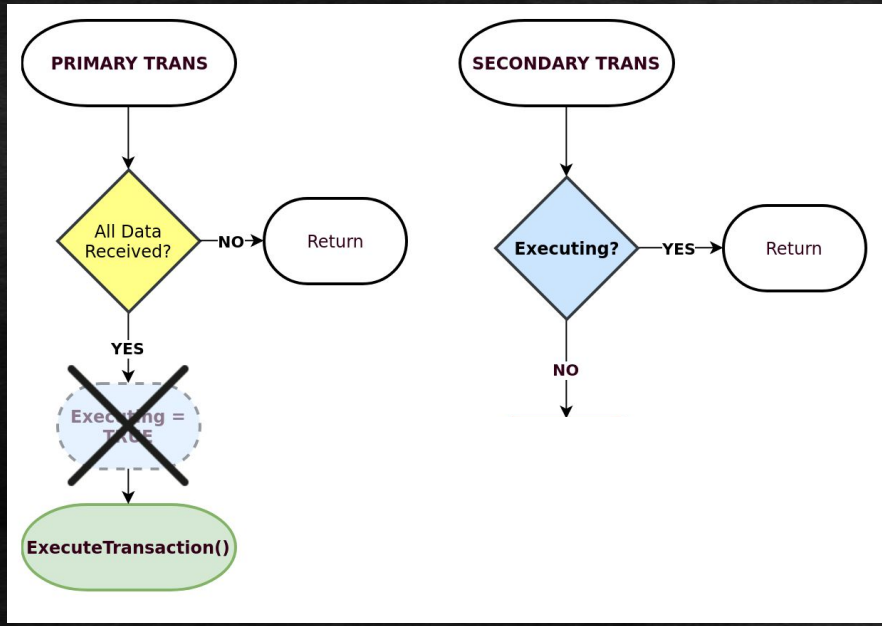
SrvOs2FeaListSizeToNt():

```
    SmbPutUlong (&FeaList->cbList, PTR_DIFF_LONG(fea, FeaList));
```

EternalChampion

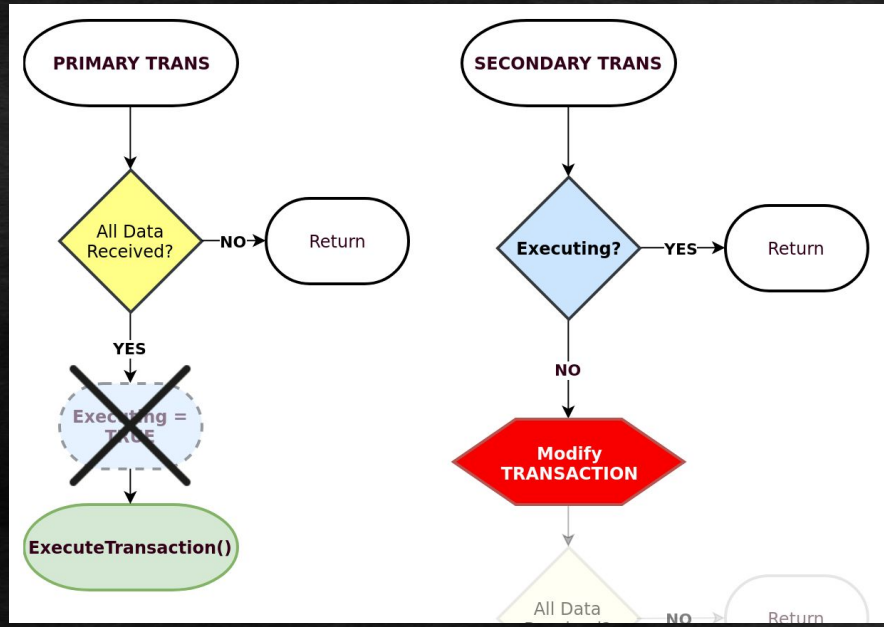
Race Condition

- TRANSACTION.Executing
  - BOOLEAN locking mechanism
  - Checked during Secondary transactions
    - NOT SET if Primary has all data!



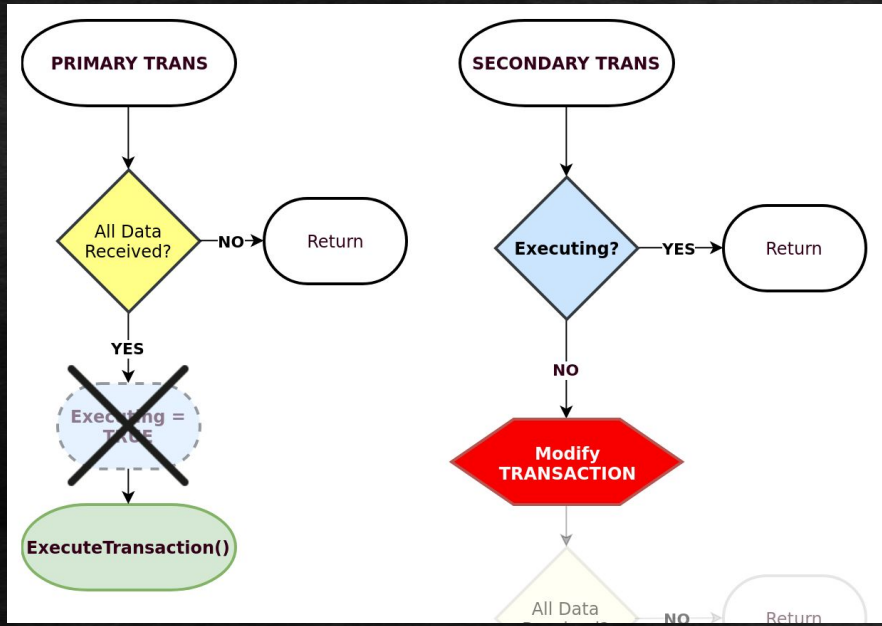
Race Condition

- TRANSACTION.Executing
  - BOOLEAN locking mechanism
  - Checked during Secondary transactions
    - NOT SET if Primary has all data!
- Modify executing TRANSACTION!
  - Info leak on single-core
  - Stack overwrite on multi-core



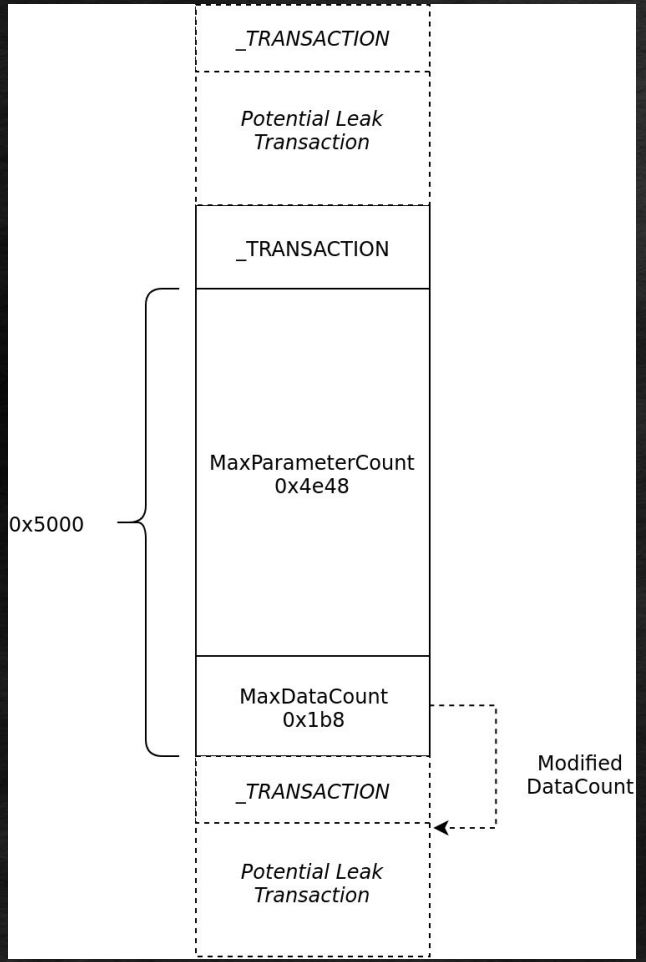
Race Condition

- TRANSACTION.Executing
  - BOOLEAN locking mechanism
  - Checked during Secondary transactions
    - NOT SET if Primary has all data!
- Modify executing TRANSACTION!
  - Info leak on single-core
  - Stack overwrite on multi-core
- CHAMPION
  - CHAMPIONS WIN RACES!



Leak a TRANSACTION

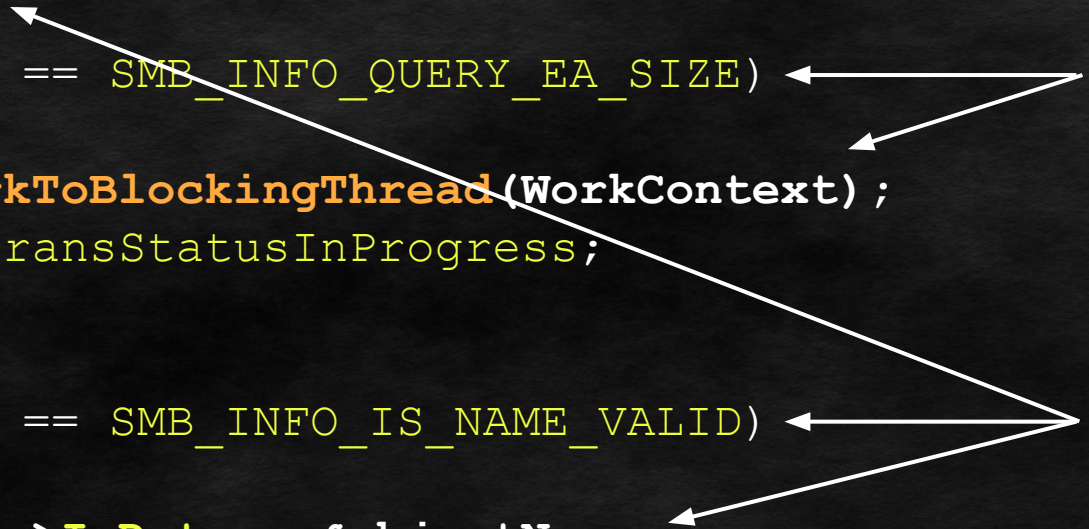
- Need a SMB which echos back Data
  - MS-RAP
    - WNetAccountSync
    - NetServerEnum2
  - NT_RENAME
    - Requires valid FID
- Primary Trans
  - Data > CONNECTION.MaxBufferSize
    - Requires restart (multiple response SMB)
    - Always winrar a Race!
- Secondary Trans sends more data
  - Increases DataCount
  - Use Displacement=0



```
SrvSmbQueryPathInformation(WorkContext)
{
    UNICODE_STRING objectName;

    if (subCommand == SMB_INFO_QUERY_EA_SIZE)
    {
        SrvQueueWorkToBlockingThread(WorkContext);
        return SmbTransStatusInProgress;
    }

    if (subCommand == SMB_INFO_IS_NAME_VALID)
    {
        transaction->InData = &objectName;
    }

    // ...
}
```

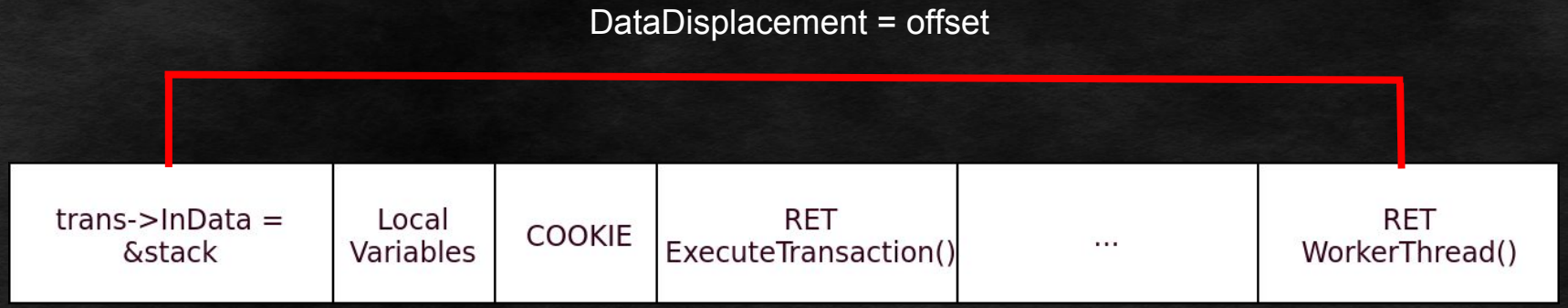
```
SrvSmbQueryPathInformation(WorkContext)
{
    UNICODE_STRING objectName;

    if (subCommand == SMB_INFO_QUERY_EA_SIZE)
    {
        SrvQueueWorkToBlockingThread(WorkContext);
        return SmbTransStatusInProgress;
    }

    if (subCommand == SMB_INFO_IS_NAME_VALID)
    {
        transaction->InData = &objectName;
    }

    // ...
}
```

```
SrvSmbQueryPathInformation(WorkContext)
{
    UNICODE_STRING objectName;

    if (subCommand == SMB_INFO_QUERY_EA_SIZE)
    {
        SrvQueueWorkToBlockingThread(WorkContext);
        return SmbTransStatusInProgress;
    }

    if (subCommand == SMB_INFO_IS_NAME_VALID)
    {
        transaction->InData = &objectName;
    }

    // ...
}
```



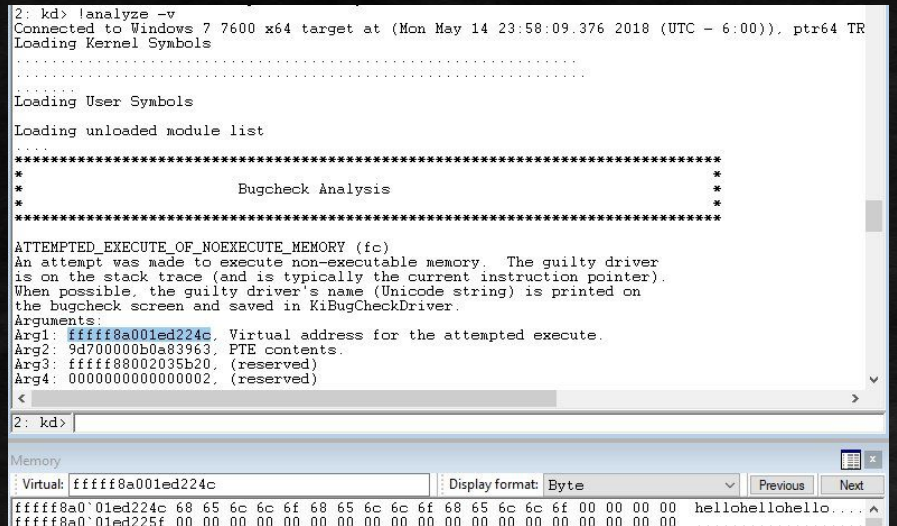
Overwrite RIP/EIP to Shellcode

- After SMB_INFO_IS_NAME_VALID, send another secondary trans
  - Displacement = stack offset
- Overwrite RET WorkerThread to Stage 0 Shellcode



RWX Shellcode Location

- No DEP (x86)
  - Write at LEAKED TRANSACTION->InData
- DEP (x64)
  - Write at LEAKED TRANSACTION->CONNECTION.ClientOSName
    - I.E. same Session Setup bug used in EBlue

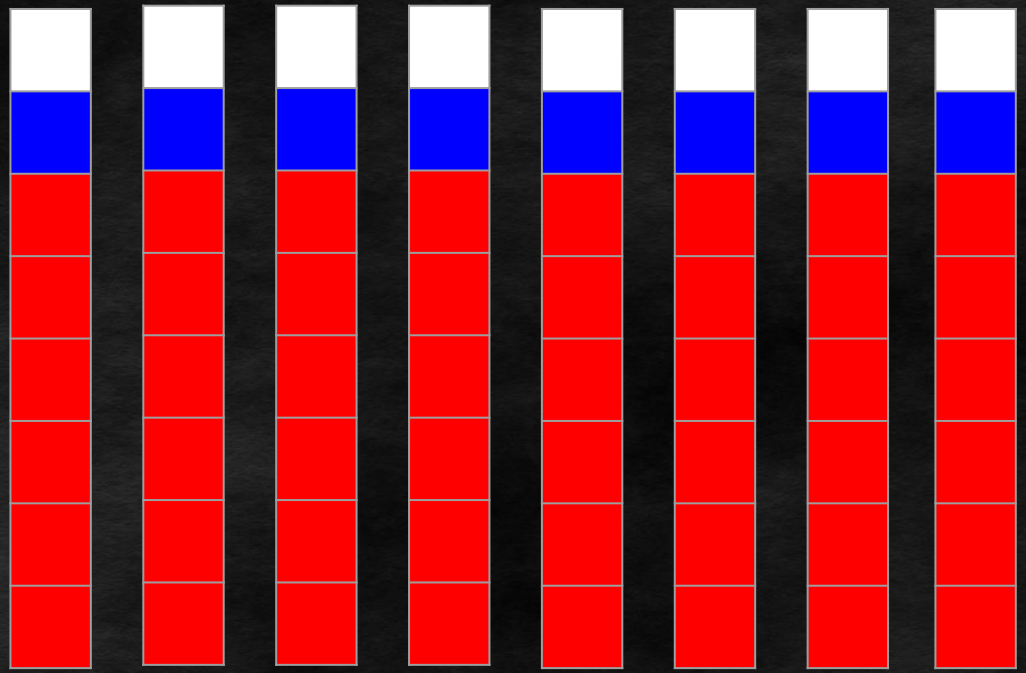


EternalChampion RCE Trigger

- 8 SMB per TCP packet

	Trans2	SMB_INFO_QUERY_EA_SIZE	Restart
	Trans2 Secondary	SMB_INFO_IS_NAME_VALID	InData = &stack
	Trans2 Secondary	DataDisplacement	Overwrite RET

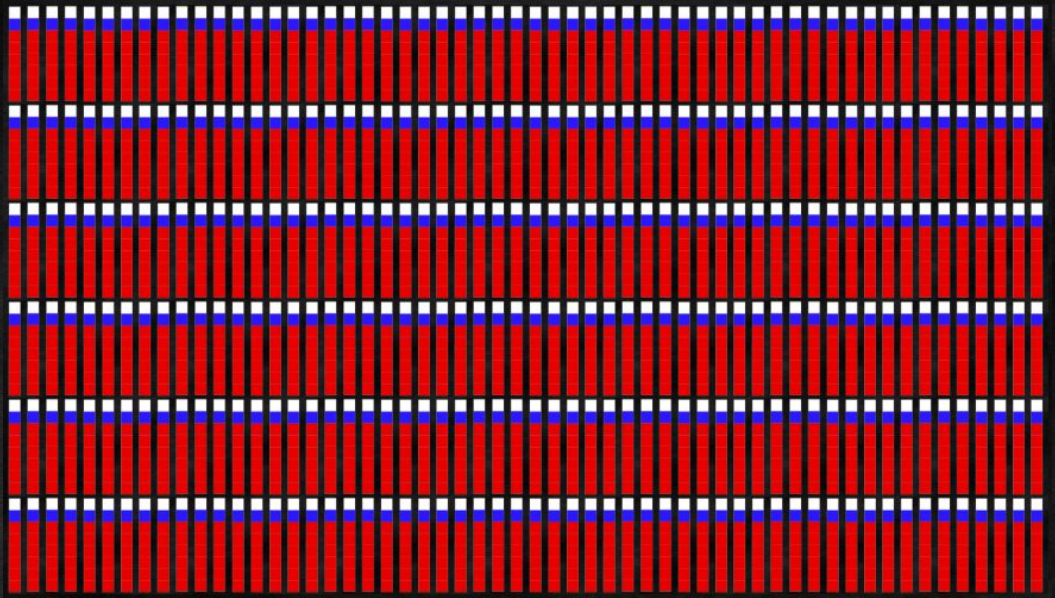
EternalChampion RCE Trigger



- 8 SMB per TCP packet
- 8 packets per attempt

	Trans2	SMB_INFO_QUERY_EA_SIZE	Restart
	Trans2 Secondary	SMB_INFO_IS_NAME_VALID	InData = &stack
	Trans2 Secondary	DataDisplacement	Overwrite RET

EternalChampion RCE Trigger



- 8 SMB per TCP packet
- 8 packets per attempt
- 42 attempts

	Trans2	SMB_INFO_QUERY_EA_SIZE	Restart
	Trans2 Secondary	SMB_INFO_IS_NAME_VALID	InData = &stack
	Trans2 Secondary	DataDisplacement	Overwrite RET

EternalChampion Shellcode

1. Loop CONNECTION.TransactionList
    - Find special identifier at start of Data buffer
        - AKA: egghunter

EternalChampion Shellcode

1. Loop CONNECTION.TransactionList
   - Find special identifier at start of Data buffer
     - AKA: egghunter
2. Copy primary payload from egg (DOUBLEPULSAR)
   - Access to pool functions
     - Can allocate large RWX space
   - Execute main stage

EternalChampion Shellcode

1. Loop CONNECTION.TransactionList
   - Find special identifier at start of Data buffer
     - AKA: egghunter
2. Copy primary payload from egg (DOUBLEPULSAR)
   - Access to pool functions
     - Can allocate large RWX space
   - Execute main stage
3. ++SrvBlockingWorkQueues->AvailableThreads

EternalChampion Shellcode

1. Loop CONNECTION.TransactionList
   - Find special identifier at start of Data buffer
     - AKA: egghunter
2. Copy primary payload from egg (DOUBLEPULSAR)
   - Access to pool functions
     - Can allocate large RWX space
   - Execute main stage
3. ++SrvBlockingWorkQueues->AvailableThreads
4. KPCR->Prcb.CurrentThread->StartAddress
   - Use global kernel data structures
   - Resume execution
     - JMP to srv!WorkerThread() loop

EternalChampion Patch

SrvSmbTransaction/SrvSmbNtTransaction():

```
if (all_data_received)
{
    ExecuteTransaction(transaction);
}
else
{
    // send interim response
}
```

EternalChampion Patch

SrvSmbTransaction/SrvSmbNtTransaction():

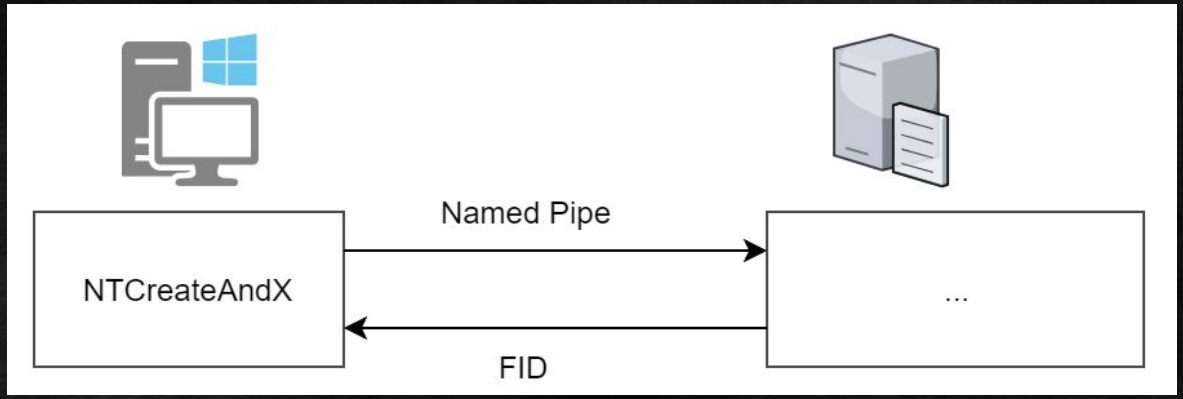
```
if (all_data_received)
{
    transaction->Executing = TRUE;
    ExecuteTransaction(transaction);
}
else
{
    // send interim response
}
```

EternalRomance

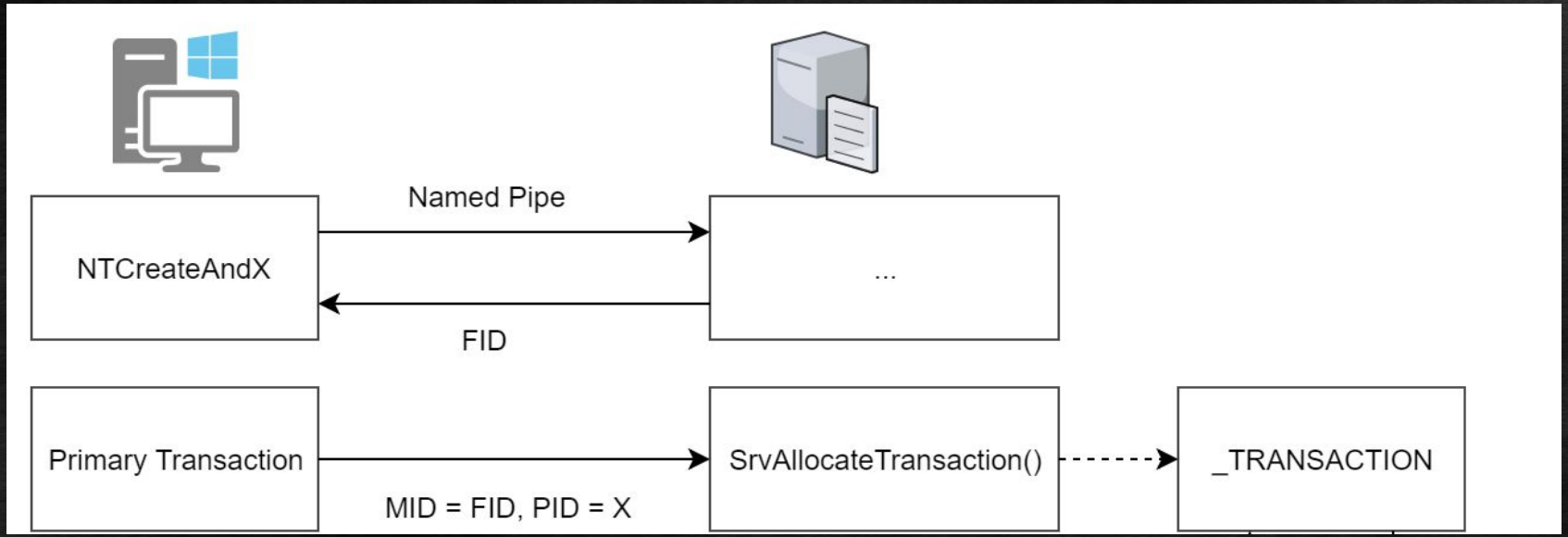
```
PTRANSACTION SrvFindTransaction (
    IN PCONNECTION Connection,
    IN PSMB_HEADER SmbHeader,
    IN USHORT Fid OPTIONAL)
{

    if (SmbHeader->Command == SMB_COM_WRITE_ANDX)
        OtherInfo = Fid;
    else
        OtherInfo = SmbHeader->Mid;


    // search TransactionList by UID/TID/PID/OtherInfo
}
```

```
SrvSmbWriteAndX ( PWORK_CONTEXT )
{
    transaction = SrvFindTransaction(connection, header, fid);

    if (writeMode & SMB_WMODE_WRITE_RAW_NAMED_PIPE)
    {
        RtlCopyMemory(transaction->InData, ...);

        transaction->InData += writeLength;
        transaction->DataCount += writeLength;
    }
}
```

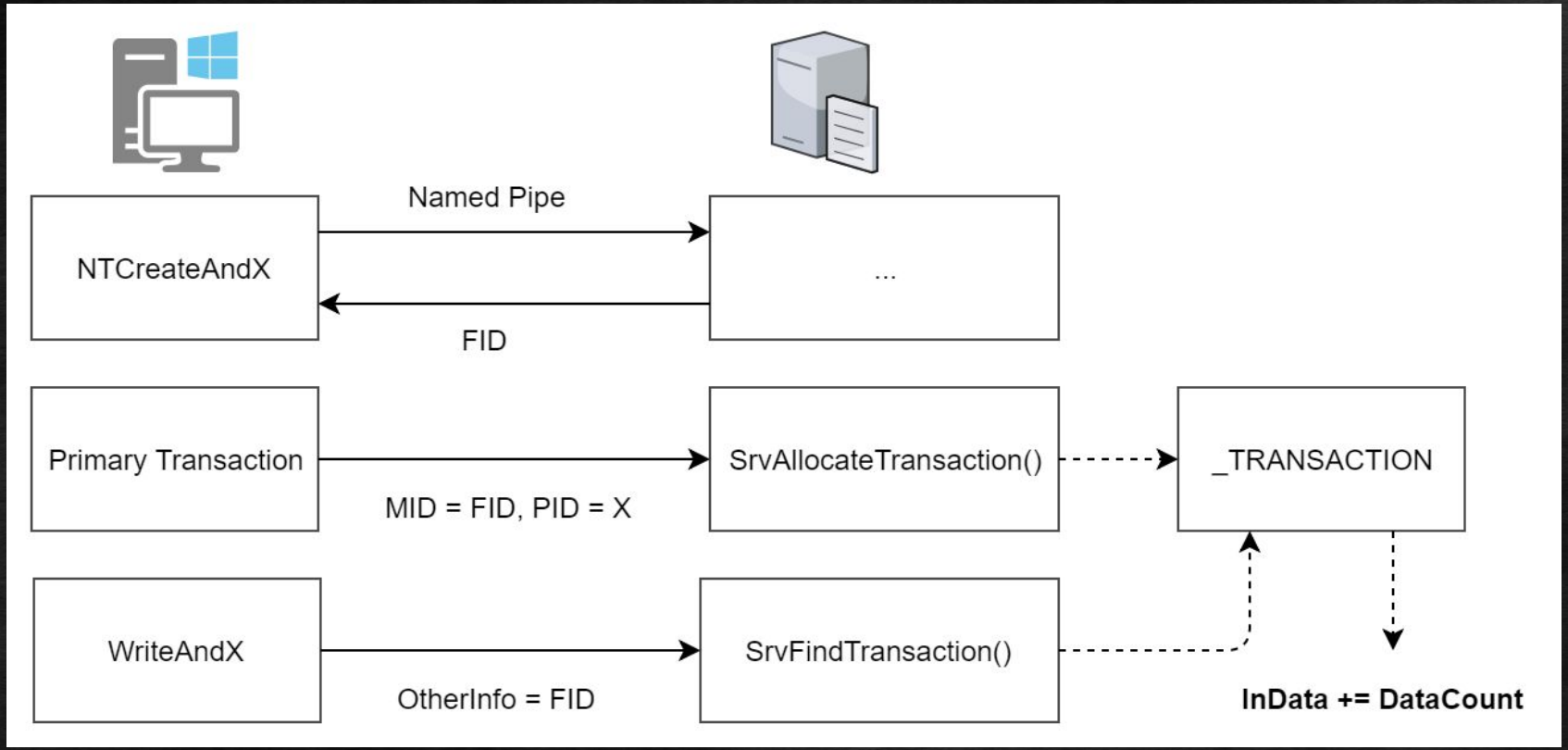
Type Confusion Sequence



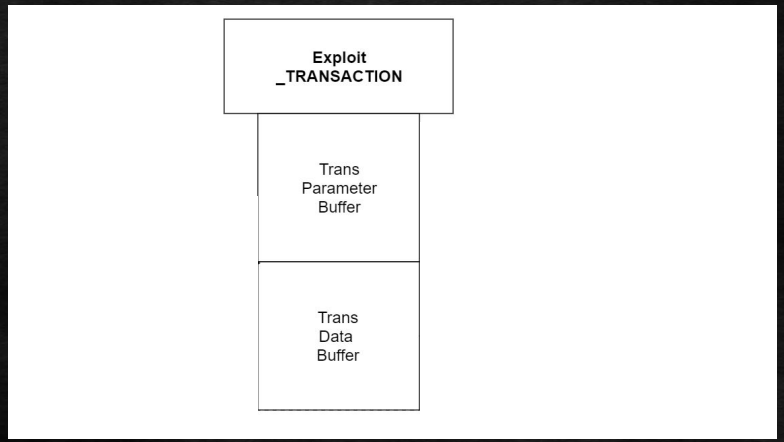
Type Confusion Sequence



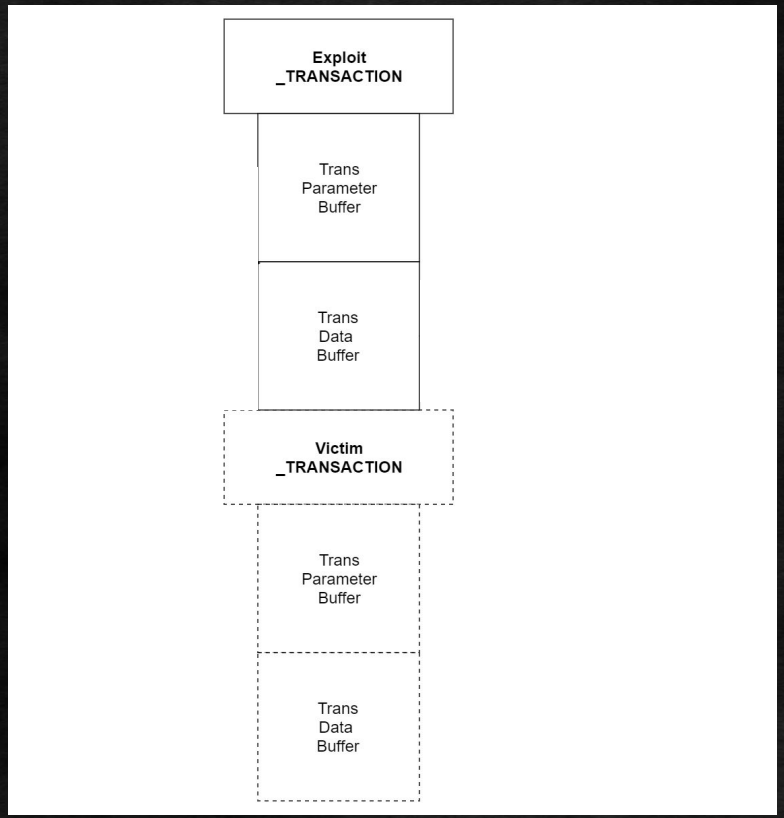
Type Confusion Sequence



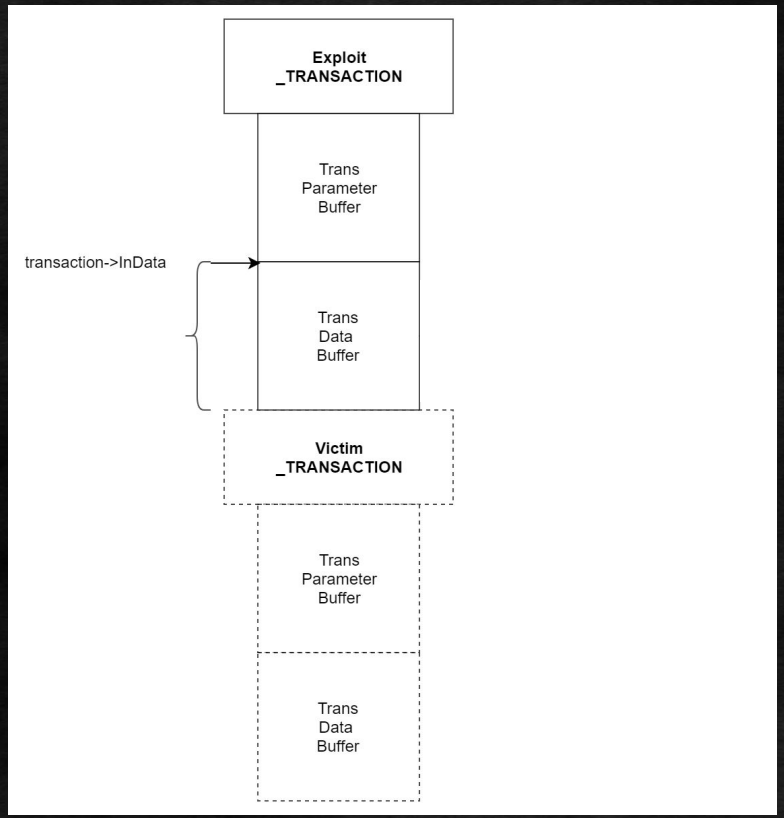
Pointer Shift Sequence



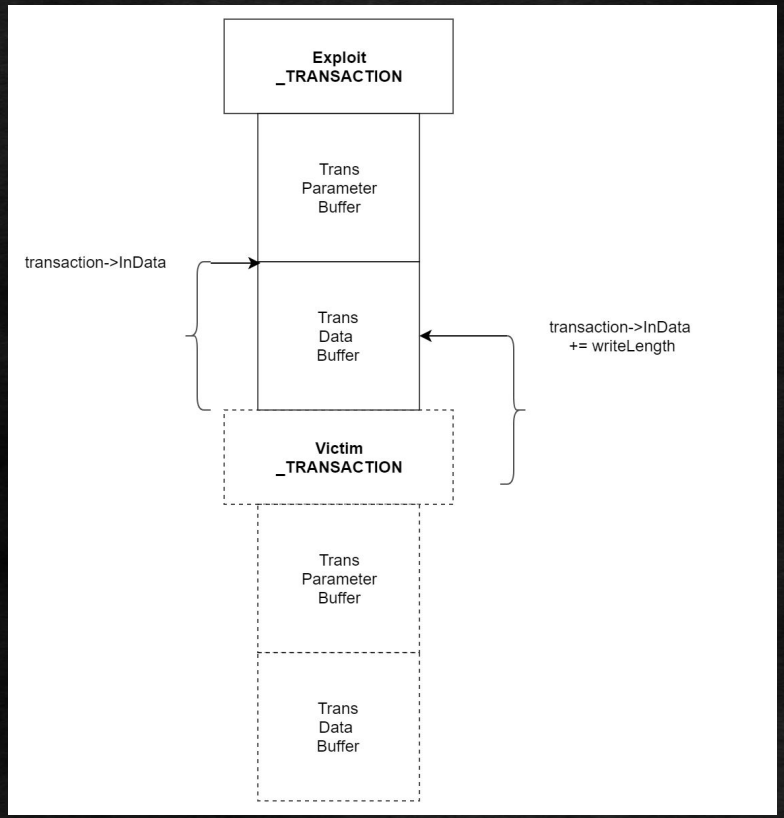
Pointer Shift Sequence



Pointer Shift Sequence

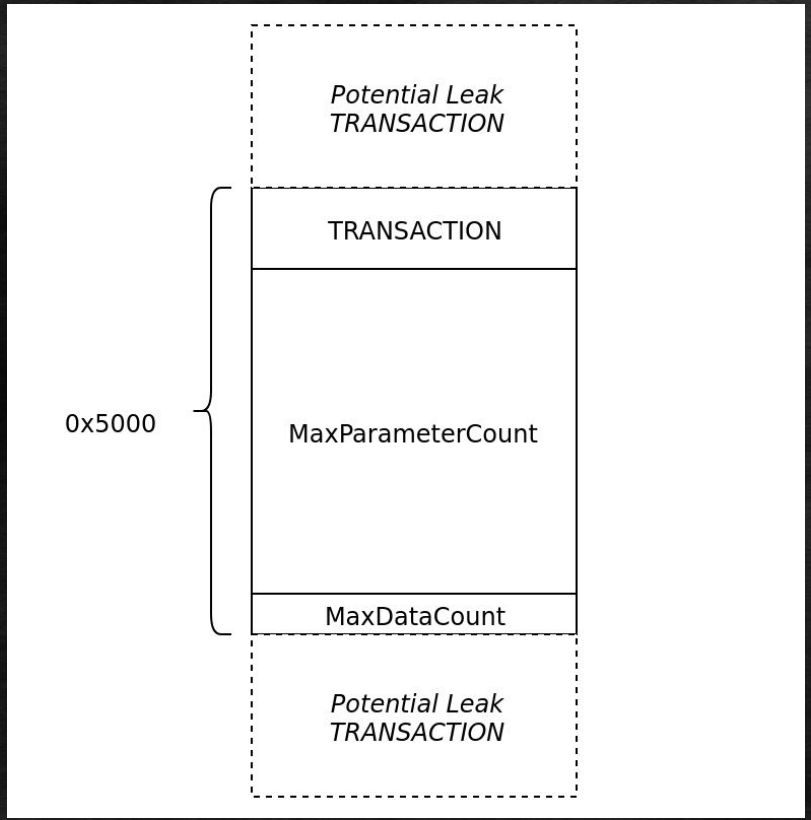


Pointer Shift Sequence



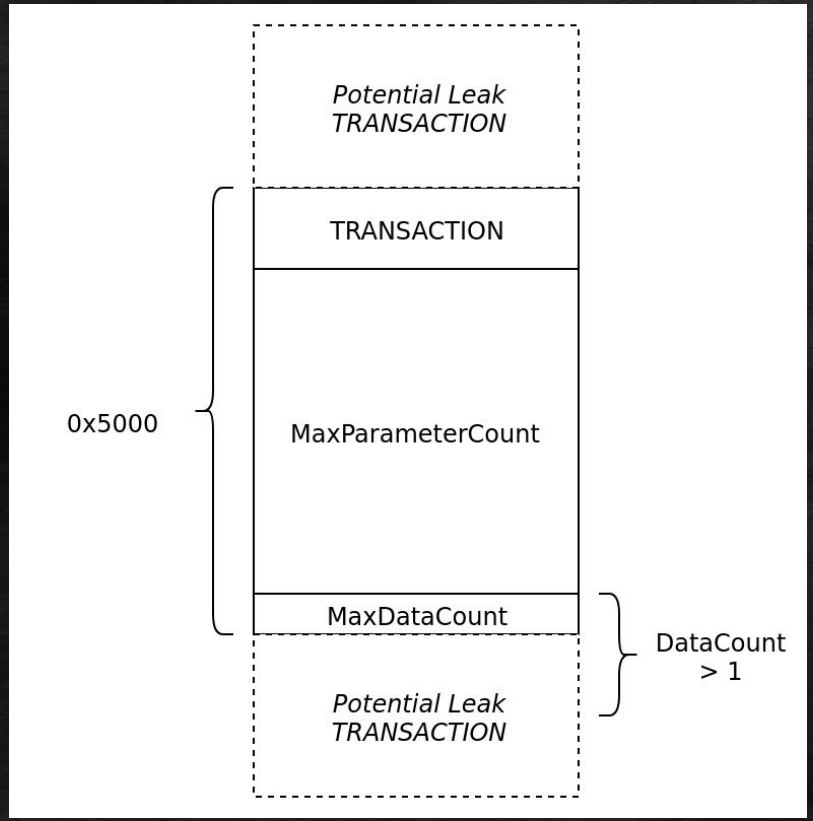
Info Leak

- Bug #1 - TRANS_PEEK_NMPIPE
  - Expects MaxParameterCount=16
    - But takes client value
  - MaxParameterCount to fill min. Space
  - MaxDataCount=1



Info Leak

- Bug #1 - TRANS_PEEK_NMPIPE
  - Expects MaxParameterCount=16
    - But takes client value
  - MaxParameterCount to fill min. Space
  - MaxDataCount=1
- Bug #2 - DataCount > MaxDataCount
  - Put >1 data in pipe
  - Peek



Paged Pool Grooming Methods

1. Fish-in-a-Barrel
   - “Remote API” (MS-RAP)
     - Fish/Dynamite
2. Matched Pairs
   - “Lattice”
     - Brides/Grooms → Romance?
3. ~~Classic~~
   - ~~“Sandwich”~~
     - ~~Frag/Padding~~

- Each: 3 exploit attempts

Fish-In-A-Barrel

- SrvXsPortMemoryHeap - 1MiB
  - Private heap, Pre-allocated
    - No fighting in the paged pool with other kernel allocations
  - MS-RAP transactions only, Rarely used
    - Babby's first heap feng shui
- Removed in 7+
  - SMAP?
  - privesc??

Fish-In-A-Barrel

	Free Heap memory
	Fish (victim)
	Dynamite (exploit)

Fish-In-A-Barrel

	Free Heap memory
	Fish (victim)
	Dynamite (exploit)

Fish-In-A-Barrel

	Free Heap memory
	Fish (victim)
	Dynamite (exploit)

Fish-In-A-Barrel

	Free Heap memory
	Fish (victim)
	Dynamite (exploit)

Fish-In-A-Barrel

	Free Heap memory
	Fish (victim)
	Dynamite (exploit)

Fish-In-A-Barrel

	Free Heap memory
	Fish (victim)
	Dynamite (exploit)

Fish-In-A-Barrel

	Free Heap memory
	Fish (victim)
	Dynamite (exploit)

Fish-In-A-Barrel

	Free Heap memory
	Fish (victim)
	Dynamite (exploit)

Fish-In-A-Barrel

	Free Heap memory
	Fish (victim)
	Dynamite (exploit)

Fish-In-A-Barrel

	Free Heap memory
	Fish (victim)
	Dynamite (exploit)

Fish-In-A-Barrel

	Free Heap memory
	Fish (victim)
	Dynamite (exploit)

Fish-In-A-Barrel

	Free Heap memory
	Fish (victim)
	Dynamite (exploit)

Fish-In-A-Barrel

	Free Heap memory
	Fish (victim)
	Dynamite (exploit)

Fish-In-A-Barrel

	Free Heap memory
	Fish (victim)
	Dynamite (exploit)

Fish-In-A-Barrel

	Free Heap memory
	Fish (victim)
	Dynamite (exploit)

Fish-In-A-Barrel

	Free Heap memory
	Fish (victim)
	Dynamite (exploit)

Fish-In-A-Barrel

	Free Heap memory
	Fish (victim)
	Dynamite (exploit)

Fish-In-A-Barrel

	Free Heap memory
	Fish (victim)
	Dynamite (exploit)

Fish-In-A-Barrel

	Free Heap memory
	Fish (victim)
	Dynamite (exploit)

Fish-In-A-Barrel

	Free Heap memory
	Fish (victim)
	Dynamite (exploit)

Fish-In-A-Barrel

	Free Heap memory
	Fish (victim)
	Dynamite (exploit)

Fish-In-A-Barrel

	Free Heap memory
	Fish (victim)
	Dynamite (exploit)

Fish-In-A-Barrel

	Free Heap memory
	Fish (victim)
	Dynamite (exploit)

Fish-In-A-Barrel

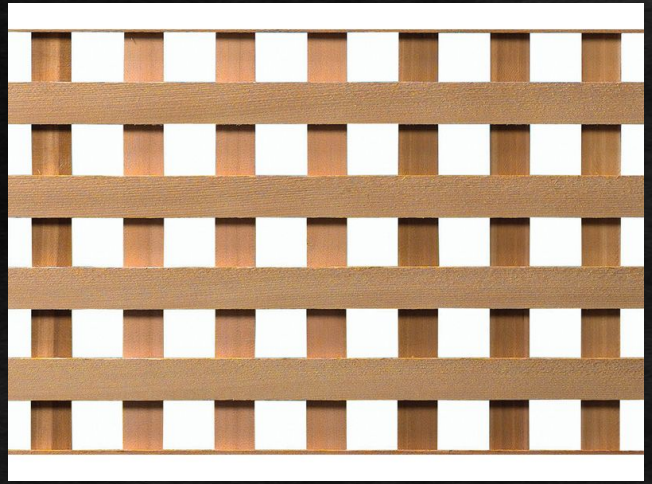
	Free Heap memory
	Fish (victim)
	Dynamite (exploit)

Fish-In-A-Barrel

	Free Heap memory
	Fish (victim)
	Dynamite (exploit)

Matched Pairs “Lattice”

- All versions of Windows
  - Including, 7+
- Must overcome pool contention
  - Not a private heap
    - Normal, Paged Pool
    - PASSIVE_LEVEL



Matched Pairs “Lattice”

	Free Heap memory
	Grooms
	Brides (victim)
	Exploit (pointer shift)

Matched Pairs “Lattice”

	Free Heap memory
	Grooms
	Brides (victim)
	Exploit (pointer shift)

Matched Pairs “Lattice”

	Free Heap memory
	Grooms
	Brides (victim)
	Exploit (pointer shift)

Matched Pairs “Lattice”

	Free Heap memory
	Grooms
	Brides (victim)
	Exploit (pointer shift)

Matched Pairs “Lattice”

	Free Heap memory
	Grooms
	Brides (victim)
	Exploit (pointer shift)

Matched Pairs “Lattice”

	Free Heap memory
	Grooms
	Brides (victim)
	Exploit (pointer shift)

Matched Pairs “Lattice”

	Free Heap memory
	Grooms
	Brides (victim)
	Exploit (pointer shift)

Matched Pairs “Lattice”

	Free Heap memory
	Grooms
	Brides (victim)
	Exploit (pointer shift)

Matched Pairs “Lattice”

	Free Heap memory
	Grooms
	Brides (victim)
	Exploit (pointer shift)

Matched Pairs “Lattice”

	Free Heap memory
	Grooms
	Brides (victim)
	Exploit (pointer shift)

Matched Pairs “Lattice”

	Free Heap memory
	Grooms
	Brides (victim)
	Exploit (pointer shift)

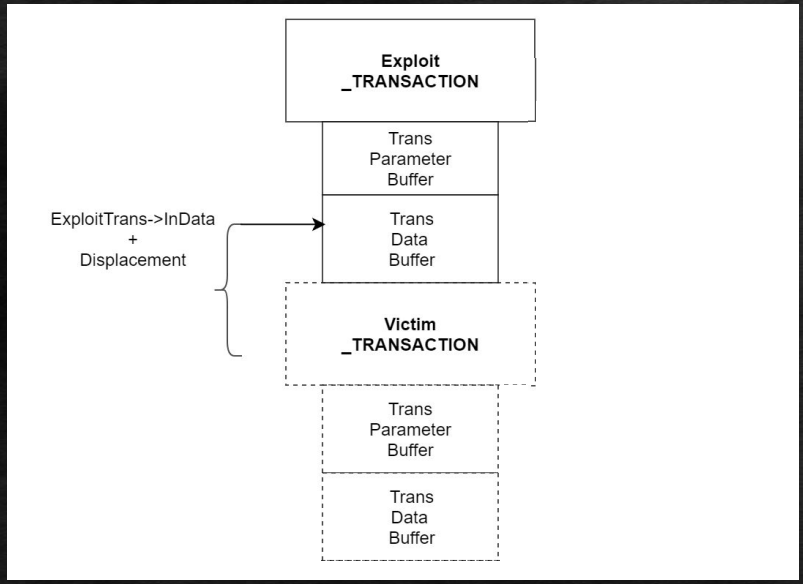
Matched Pairs “Lattice”

	Free Heap memory
	Grooms
	Brides (victim)
	Exploit (pointer shift)

Matched Pairs “Lattice”

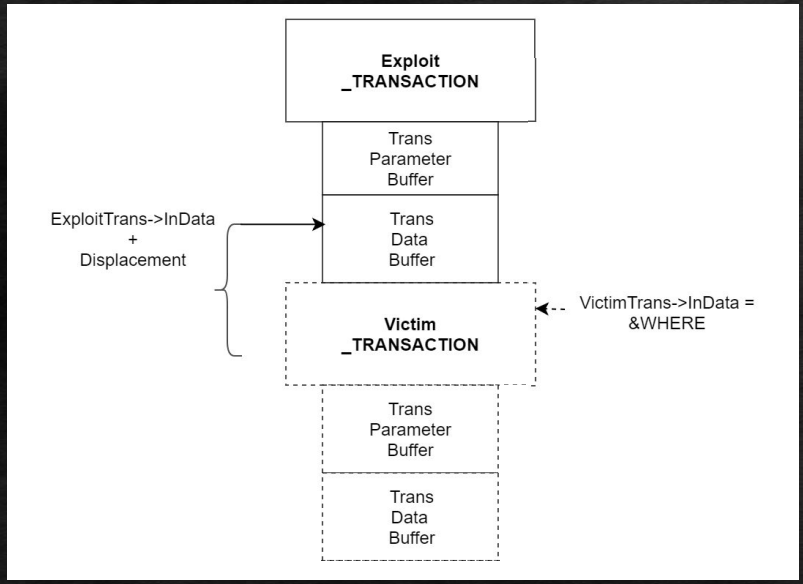
	Free Heap memory
	Grooms
	Brides (victim)
	Exploit (pointer shift)

Write-What-Where Primitive



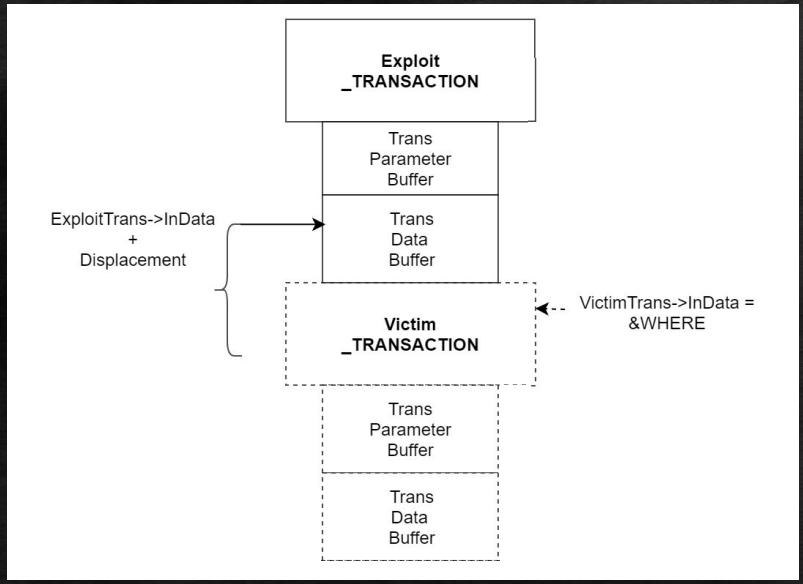
Write-What-Where Primitive

1. Exploit Transaction (PID=X)
   - Set VictimTrans->InData to &WHERE
   - Set VictimTrans->Executing to FALSE
   - Increase reference count!
     - Don’t want it to get freed
   - etc...

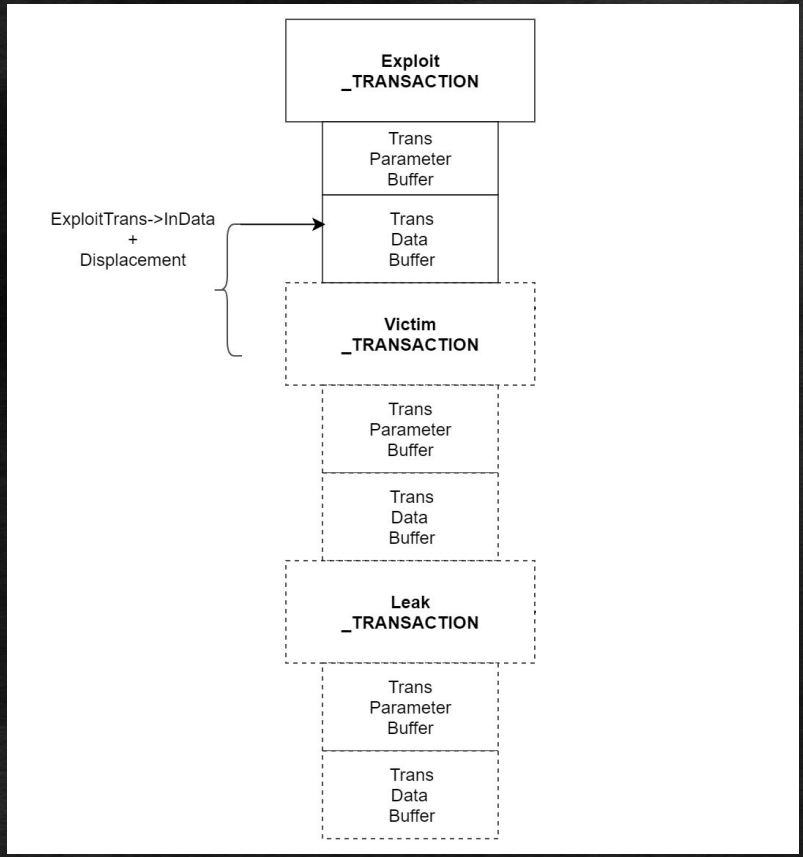


Write-What-Where Primitive

1. Exploit Transaction (PID=X)
   - Set VictimTrans->InData to &WHERE
   - Set VictimTrans->Executing to FALSE
   - Increase reference count!
     - Don’t want it to get freed
   - etc...
2. VictimTrans Secondary (MID=0)
   - Trans Data Block = WHAT[]

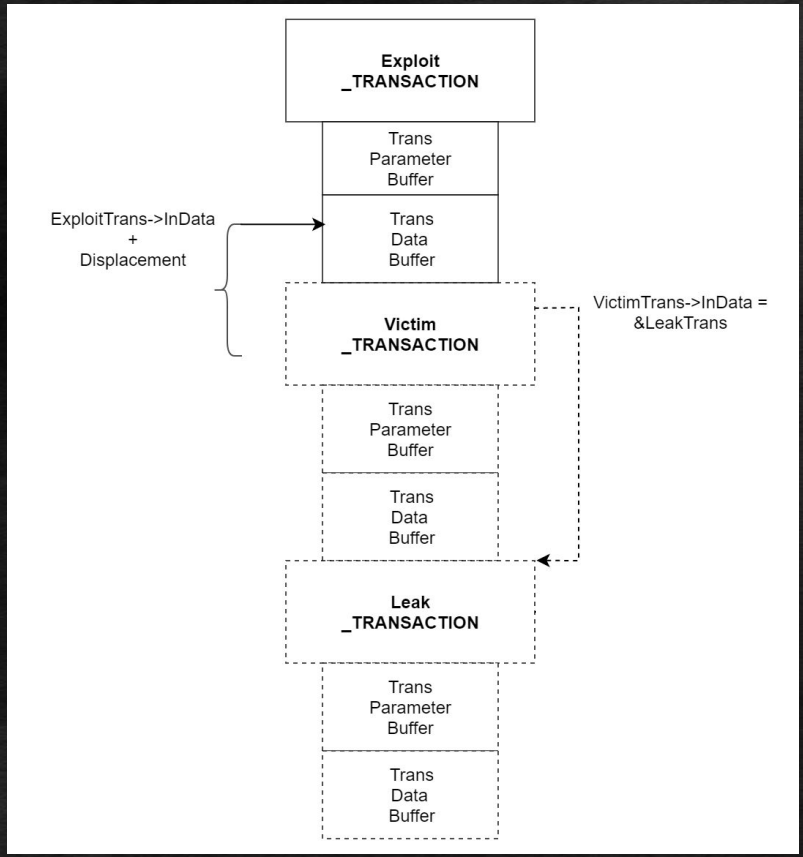


Read-Where Primitive



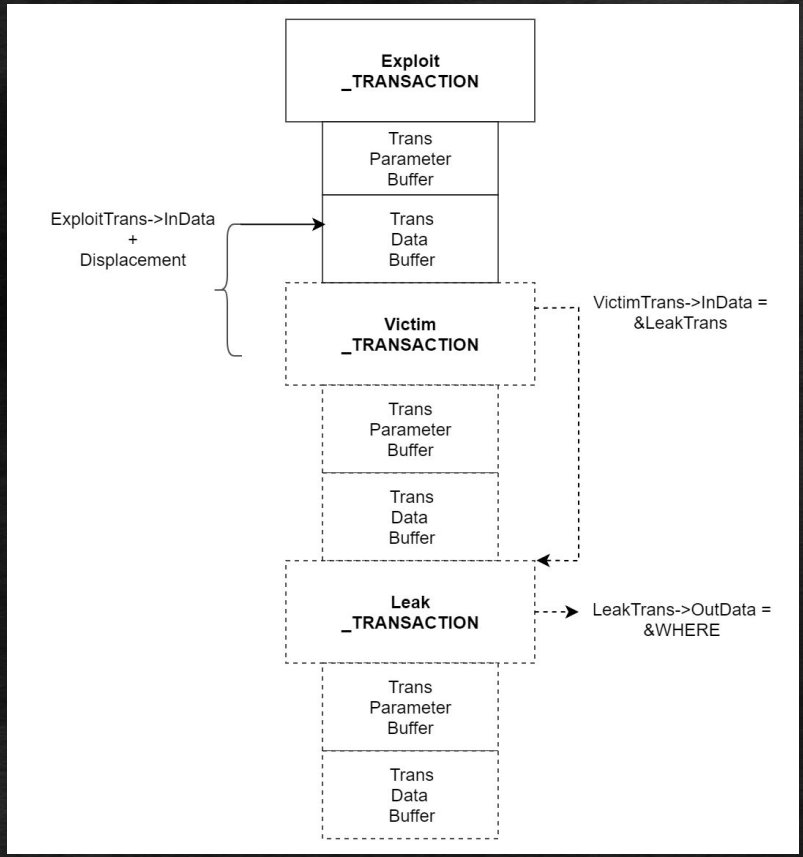
Read-Where Primitive

1. Exploit Transaction (PID=X)
   - Modify VictimTrans to point at LeakTrans
     - Address inferred by its contents
     - VictimTrans now modifies LeakTrans



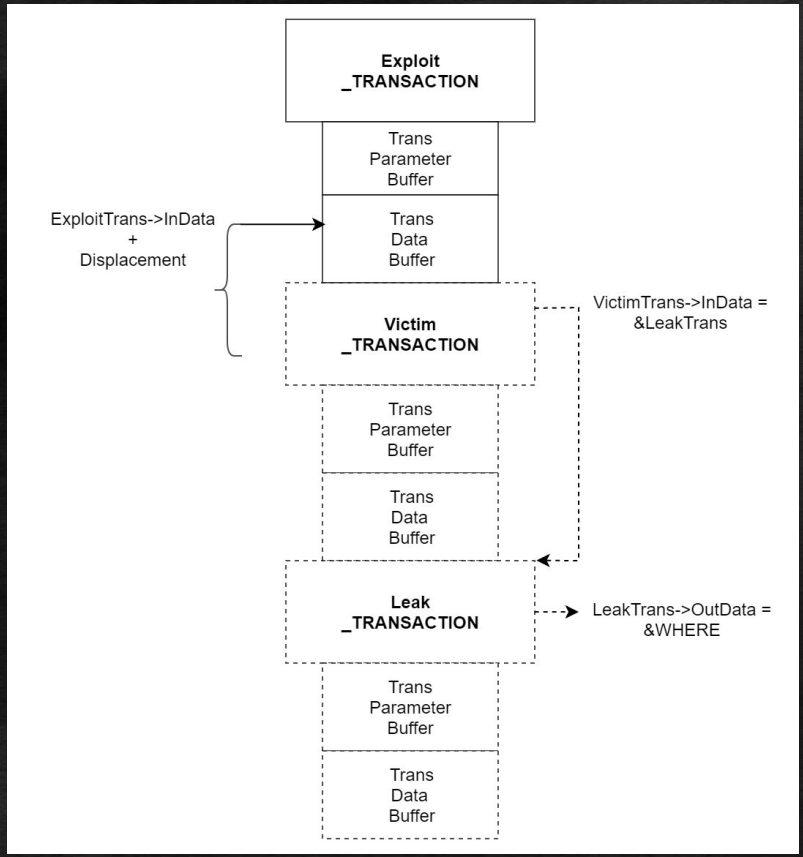
Read-Where Primitive

1. Exploit Transaction (PID=X)
   - Modify VictimTrans to point at LeakTrans
     - Address inferred by its contents
     - VictimTrans now modifies LeakTrans
2. VictimTrans Trans_Secondary (MID=0)
   - LeakTrans->OutData = &WHERE
   - LeakTrans->Setup = TRANS_PEEK_NMPIPE
   - LeakTrans->MaxDataCount = size_t



Read-Where Primitive

1. Exploit Transaction (PID=X)
   - Modify VictimTrans to point at LeakTrans
     - Address inferred by its contents
     - VictimTrans now modifies LeakTrans
2. VictimTrans Trans_Secondary (MID=0)
   - LeakTrans->OutData = &WHERE
   - LeakTrans->Setup = TRANS_PEEK_NMPIPE
   - LeakTrans->MaxDataCount = size_t
3. LeakTrans Trans_Secondary
   - Echos back the LeakTrans->OutData

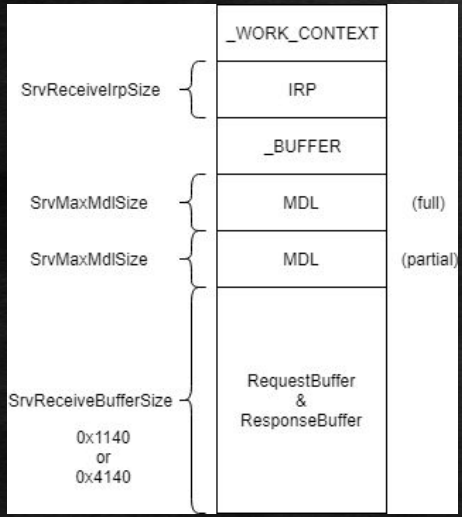


Quest for RWX NonPagedPool

1. Exploit Trans
   - Set VictimTrans->OutParameters = NULL
2. Send Secondary Victim Transaction

```
if (VictimTrans->OutParameters == NULL)
    VictimTrans->OutParameters = WorkContext->ResponseBuffer;
```

3. Read Primitive
   - Read address just set
4. Write Primitive
   - Send shellcode



Quest to Execute the Shellcode

1. Locate Transaction2DispatchTable
   - FIND in srv.sys .data section (read primitive)
2. Hook a Trans2 subcommand
   - REPLACE a pointer in table (write primitive)
3. Fake Trans2 executes the hook
   - Subcommand = hooked index
   - Similar methodology as DOUBLEPULSAR

- Given:
  - Read/write primitives
  - Leaked TRANSACTION has CONNECTION pointer

Locate Transaction2DispatchTable

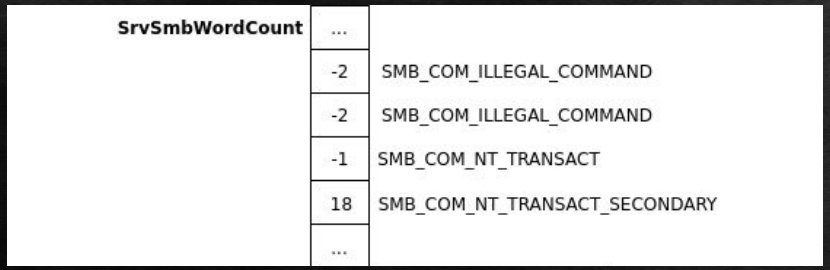
1. Read in LeakTrans->CONNECTION

Locate Transaction2DispatchTable

1. Read in LeakTrans->CONNECTION
2. CONNECTION->EndpointSpinLock
    - SrvGlobalSpinLocks
        - Inside PE .data section

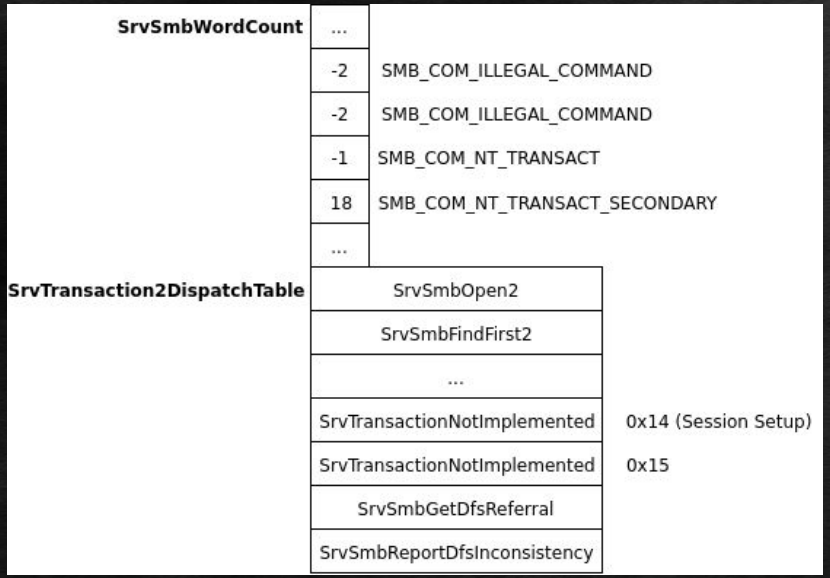
Locate Transaction2DispatchTable

1. Read in LeakTrans->CONNECTION
2. CONNECTION->EndpointSpinLock
   - SrvGlobalSpinLocks
     - Inside PE .data section
3. Read backwards, SrvSmbWordCount
   - Illegal commands = -2 (0xfe)
   - If we see a bunch of fefe, we're close



Locate Transaction2DispatchTable

1. Read in LeakTrans->CONNECTION
2. CONNECTION->EndpointSpinLock
   - SrvGlobalSpinLocks
     - Inside PE .data section
3. Read backwards, SrvSmbWordCount
   - Illegal commands = -2 (0xfe)
   - If we see a bunch of fefe, we're close
4. Transaction2DispatchTable
   - Function pointers #0x14 == #0x15
     - SrvTransactionNotImplemented



EternalRomance Info Leak Patch #1

SrvSmbTransaction() *Before*:

```
if (subCommand == TRANS_PEEK_NMPIPE)
{
    maxParameterCount = MAX(16, maxParameterCount);
}

SrvAllocateTransaction(&Transaction, ...);

Transaction->MaxParameterCount = maxParameterCount;
```

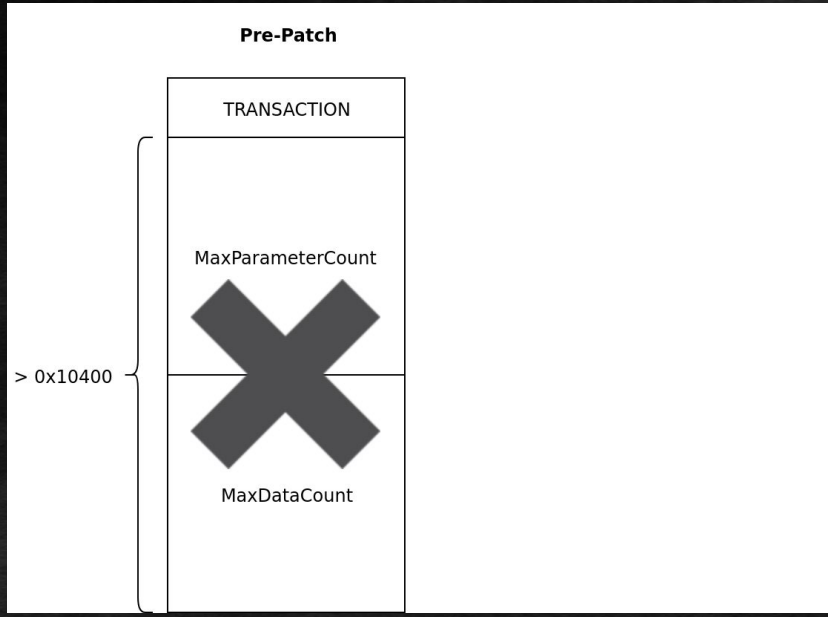
EternalRomance Info Leak Patch #1

SrvSmbTransaction() *After*:

```
if (subCommand == TRANS_PEEK_NMPIPE)
{
    maxParameterCount = 16;
}

SrvAllocateTransaction(&Transaction, ...);

Transaction->MaxParameterCount = maxParameterCount;
```

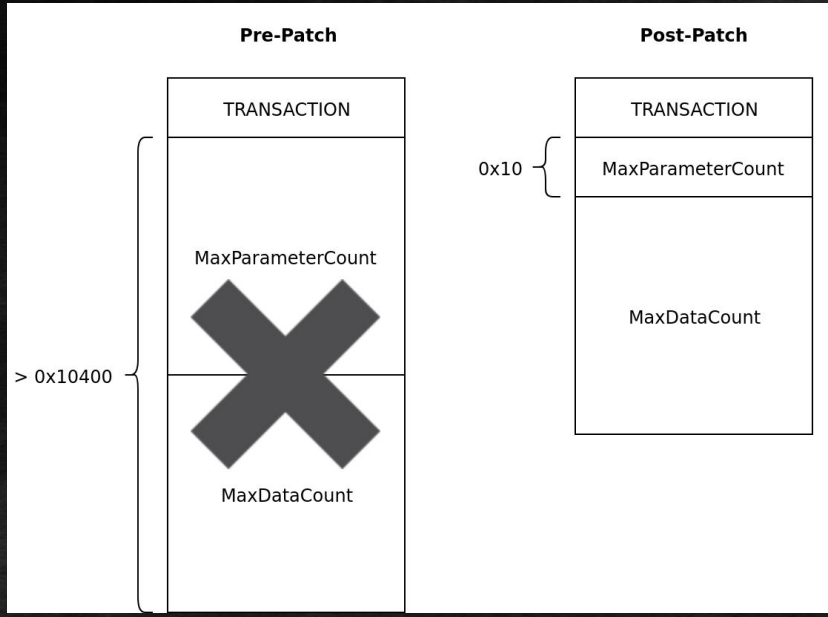
MS17-010 Scanners

- Max TRANSACTION allocation size=0x10400
  - 0xC0000205 - STATUS_INSUFF_SERVER_RESOURCES
- Send MaxParameterCount+MaxDataCount > 0x10400



MS17-010 Scanners

- Max TRANSACTION allocation size=0x10400
  - 0xC0000205 - STATUS_INSUFF_SERVER_RESOURCES
- Send MaxParameterCount+MaxDataCount > 0x10400
  - Patch fixes MaxParameterCount to 16
    - Passes allocation routine!
  - Different NT error (i.e. invalid FID)



EternalRomance Info Leak Patch #2

SrvCompleteExecuteTransaction() *New Code*:

```
if (transaction->DataCount > transaction->MaxDataCount)
    transaction->DataCount = transaction->MaxDataCount;


if (transaction->ParameterCount > transaction->MaxParameterCount)
    transaction->ParameterCount = transaction->MaxParameterCount;
```

EternalRomance RCE Patch #1

SrvSmbWriteAndX() *Before*:

```
RtlCopyMemory(transaction->InData, ...);

transaction->InData += writeLength;
transaction->DataCount += writeLength;
```

EternalRomance RCE Patch #1

SrvSmbWriteAndX() *After*:

```
RtlCopyMemory(transaction->InData + transaction->DataCount, ...);

transaction->InData += writeLength;
transaction->DataCount += writeLength;
```

EternalRomance RCE Patch #2

1. SrvSmbNtTransaction/SrvSmbTransaction() *New Code*:

```
SrvAllocateTransaction(&Transaction, ...)

Transaction->SecondaryCommand =                    /* 0x38 */
            SMB_COM_NT_TRANS_SECONDARY;

SrvInsertTransaction(&Transaction);
```

EternalRomance RCE Patch #2

1. SrvSmbNtTransaction/SrvSmbTransaction() *New Code*:

```
SrvAllocateTransaction(&Transaction, ...)

Transaction->SecondaryCommand =                    /* 0x38 */
            SMB_COM_NT_TRANS_SECONDARY;

SrvInsertTransaction(&Transaction);
```

2. SrvFindTransaction() *New Code*:

```
if (FoundTrans->SecondaryCommand != NewSmb->Command)
    return NULL;
```

EternalSynergy

EternalSynergy 1.0.1

- Same buffalo overflow, read/writes, as EternalRomance
  - Matched pairs
  - "Classic"
- Same info leak as EternalChampion
  - NT_Rename Race Condition
    - TRANS_PEEK_NAMED_PIPE is fixed…
- Srv.sys is using NonPagedPoolNx for Work Items!
  - Needs DEP bypass

Quest for RWX Memory (via remote read)

Type	Pointer Dereference	Offset
WORK_QUEUE	Connection->PreferredWorkQueue	variadic

- Given: Connection

Quest for RWX Memory (via remote read)

Type	Pointer Dereference	Offset
WORK_QUEUE	Connection->PreferredWorkQueue	variadic
KTHREAD	PreferredWorkQueue->IrpThread	0x198

- Given: Connection

Quest for RWX Memory (via remote read)

Type	Pointer Dereference	Offset
WORK_QUEUE	Connection->PreferredWorkQueue	variadic
KTHREAD	PreferredWorkQueue->IrpThread	0x198
KPROCESS	IrpThread->Process	0x220

- Given: Connection

Quest for RWX Memory (via remote read)

Type	Pointer Dereference	Offset
WORK_QUEUE	Connection->PreferredWorkQueue	variadic
KTHREAD	PreferredWorkQueue->IrpThread	0x198
KPROCESS	IrpThread->Process	0x220
PVOID	KProcess->ProcessListEntry.Blink	0x240

- Given: Connection
- Obtain: ProcessListEntry.Blink
  - nt!KiProcessListHead*

* https://www.geoffchappell.com/studies/windows/km/ntoskrnl/structs/kprocess/index.htm

Quest for RWX Memory (via remote read)

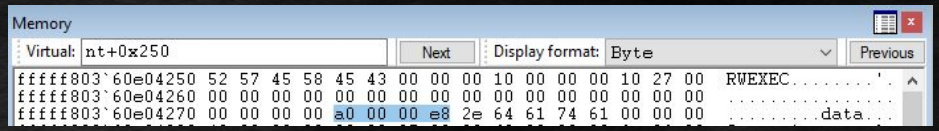
Type	Pointer Dereference	Offset
WORK_QUEUE	Connection->PreferredWorkQueue	variadic
KTHREAD	PreferredWorkQueue->IrpThread	0x198
KPROCESS	IrpThread->Process	0x220
PVOID	KProcess->ProcessListEntry.Blink	0x240

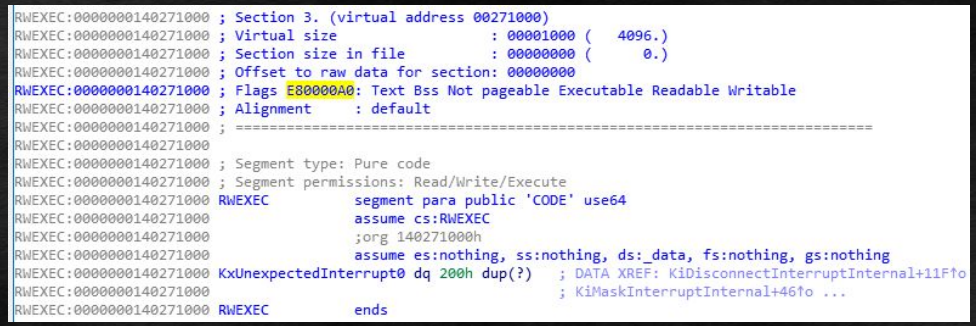
- Given: Connection
- Obtain: ProcessListEntry.Blink
  - nt!KiProcessListHead*
- Search backwards by page size for 'MZ'
  - ntoskrnl.exe PE header

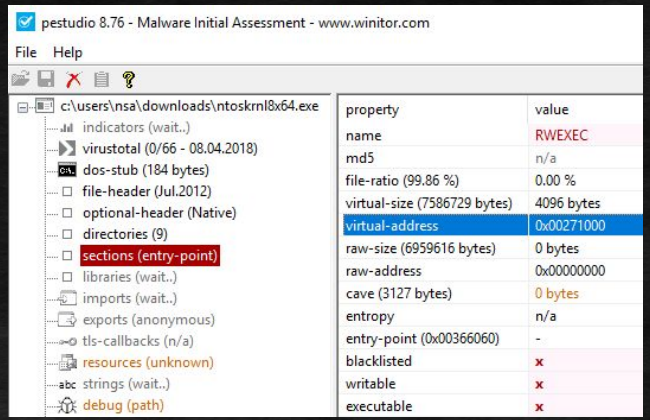
* https://www.geoffchappell.com/studies/windows/km/ntoskrnl/structs/kprocess/index.htm

ntoskrnl.exe RWEXEC Section

- Remote read offset 0x250 into &ntoskrnl.exe
- Check section headers:
  - +0x08 == 0x1000 (Virtual Size: 4096)
  - +0x0C <= 0x800000 (Virtual Addr: 0x271000 &KxUnexpectedInterrupt)
  - +0x24 == 0xE80000A0 (Segment permissions: RWX)







Additional Research

- @sleepya_
  - https://github.com/worawit/MS17-010
- @n_joly
  - https://hitcon.org/2017/CMT/slide-files/d2_s2_r0.pdf
- @jennamagius and @zerosum0x0
  - https://keybase.pub/jennamagius/EternalBlue_RiskSense-Exploit-Analysis-and-Port-to-Microsoft-Windows-10.pdf
- @msftsecresponse
  - https://blogs.technet.microsoft.com/srd/2017/06/29/eternal-champion-exploit-analysis/
  - https://blogs.technet.microsoft.com/srd/2017/07/13/eternal-synergy-exploit-analysis/

- @swithak
  - https://swithak.github.io/SH20TAATSB18/Home/
- @franciscKrs
  - https://speakerdeck.com/franciscK/danderspritz-how-the-equation-groups-2013-tools-still-pwn-in-2017
- @msuiche
  - https://www.comae.io/reports/us-17-Suiche-TheShadowBrokers-Cyber-Fear-Game-Changers.pdf

Thanks!

zerosum0x0