Eternal Exploits

Reverse Engineering of FuzzBunch and MS17-010

zerosum0x0

DEFCON

August 1983
Warning!

Presentation **contains** classified information.

Those with active security clearances **must** not disclose.
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
- WORKCONTEXT
  - 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\n  - 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
Primary Transaction Data+Parameter

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

- **Count**
  - How much is in this particular SMB

<table>
<thead>
<tr>
<th>ParameterOffset</th>
<th>DataOffset</th>
</tr>
</thead>
<tbody>
<tr>
<td>ParameterCount</td>
<td>DataCount</td>
</tr>
</tbody>
</table>
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

<table>
<thead>
<tr>
<th>ParameterOffset</th>
<th>DataOffset</th>
</tr>
</thead>
<tbody>
<tr>
<td>ParameterCount</td>
<td>DataCount</td>
</tr>
<tr>
<td>TotalParameterCount</td>
<td>TotalDataCount</td>
</tr>
</tbody>
</table>
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

<table>
<thead>
<tr>
<th>ParameterOffset</th>
<th>DataOffset</th>
</tr>
</thead>
<tbody>
<tr>
<td>ParameterCount</td>
<td>DataCount</td>
</tr>
<tr>
<td>TotalParameterCount</td>
<td>TotalDataCount</td>
</tr>
<tr>
<td>MaxParameterCount</td>
<td>MaxDataCount</td>
</tr>
</tbody>
</table>
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)
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

![Diagram of transaction functions]

- SrvAllocateTransaction → SrvInsertTransaction
- SrvFindTransaction → SrvReferenceTransaction
- SrvDereferenceTransaction → SrvFreeTransaction
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

```c
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) \n    (sizeof(FEA) + (ea)->cbName + 1 + (ea)->cbValue)
```
OS/2 FEALIST

```c
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

```c
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
};
```
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 SrvOs2FeatListSizeToNt(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

```c
ULONG FEALIST.cbList;
SmbPutUshort(&FeaList->cbList, PTR_DIFF_SHORT(fea, FeaList));
```

<table>
<thead>
<tr>
<th></th>
<th>HIDWORD</th>
<th>LODWORD</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Attacker</strong></td>
<td>0001</td>
<td>0000</td>
</tr>
<tr>
<td><strong>Valid Size</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Vuln Size</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>NT Buffer Size</strong></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Bug #1 - Integer Cast Error

ULONG FEALIST.cbList;

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

Win7

<table>
<thead>
<tr>
<th></th>
<th>HIDWORD</th>
<th>LODWORD</th>
</tr>
</thead>
<tbody>
<tr>
<td>Attacker</td>
<td>0001</td>
<td>0000</td>
</tr>
<tr>
<td>Valid Size</td>
<td>0000</td>
<td>ff5d</td>
</tr>
<tr>
<td>Vuln Size</td>
<td></td>
<td></td>
</tr>
<tr>
<td>NT Buffer Size</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Bug #1 - Integer Cast Error

```
ULONG FeeList.cbList;
SmbPutUshort(&FeeList->cbList, PTR_DIFF_SHORT(fea, FeeList));
```

<table>
<thead>
<tr>
<th></th>
<th>HIDWORD</th>
<th>LODWORD</th>
</tr>
</thead>
<tbody>
<tr>
<td>Attacker</td>
<td>0001</td>
<td>0000</td>
</tr>
<tr>
<td>Valid Size</td>
<td>0000</td>
<td>ff5d</td>
</tr>
<tr>
<td>Vuln Size</td>
<td>0001</td>
<td>ff5d</td>
</tr>
<tr>
<td>NT Buffer Size</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Bug #1 - Integer Cast Error

```c
ULONG FeaList.cbList;

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

<table>
<thead>
<tr>
<th></th>
<th>HIDWORD</th>
<th>LODWORD</th>
</tr>
</thead>
<tbody>
<tr>
<td>Attacker</td>
<td>0001</td>
<td>0000</td>
</tr>
<tr>
<td>Valid Size</td>
<td>0000</td>
<td>ff5d</td>
</tr>
<tr>
<td>Vuln Size</td>
<td>0001</td>
<td>ff5d</td>
</tr>
<tr>
<td>NT Buffer Size</td>
<td>0001</td>
<td>0fe8</td>
</tr>
</tbody>
</table>

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

x86/x64

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

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
stl     [r32] = r21;;
st[20]  = r19
nop.i   0;;
```

DEC ALPHA

```
loc_5EBC0:
andnot  $11, 2, $17
subl    $13, $9, $9
ldl     $19, 0($17)
and     $11, 2, $18
insw1   $9, $18, $20
mskwl   $19, $18, $19
```
Attacker Supplied 0x10000

cbList
NULL FEA
NULL FEA
...
NULL FEA
NULL FEA

Filler-to-Offset FEA

Buffalo Overflow FEA
Invalid FEA

Unallocated Space

605 FEA * 0x5 each
"x00\x00\x00\x00\x00x00"
Attacker Supplied 0x10000

cbList
NULL FEA
NULL FEA
...
NULL FEA
NULL FEA

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
Attacker Supplied 0x10000

Vulnerable "Corrected" OS/2 Size

Unallocated Space

cbList
NULL FEA
NULL FEA
...
NULL FEA
NULL FEA

Filler-to-Offset FEA

Buffalo Overflow FEA

Invalid FEA

605 FEA * 0x5 each
"x00\x00\x00\x00\x00"

NT Buffer Size

\[ \text{start}(\text{FEA}) + \text{len}(\text{FEA}) > 0\times10000 \]
Attacker Supplied 0x10000

Vulnerable "Corrected" OS/2 Size

cbList
NULL FEA
NULL FEA
...
NULL FEA
NULL FEA

Filler-to-Offset FEA

Buffalo Overflow FEA

Invalid FEA

NT Buffer Size

605 FEA * 0x5 each
"x00\x00\x00\x01\x00\x00"

start(FEA) + len(FEA)
>0x10000

Sending last fragment of exploit packet!
Receiving response from exploit packet
ETERNALBLUE overwrite completed successfully (0xC080000D)!
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

<table>
<thead>
<tr>
<th>Scenario</th>
<th>WordCount</th>
<th>Capabilities*</th>
<th>Flags2†</th>
<th>Deduced Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>12</td>
<td>✓</td>
<td>✓</td>
<td>Extended Security</td>
</tr>
<tr>
<td>2</td>
<td>12</td>
<td>✓</td>
<td>x</td>
<td>NT Security</td>
</tr>
<tr>
<td>3</td>
<td>12</td>
<td>x</td>
<td>✓</td>
<td>Invalid</td>
</tr>
<tr>
<td>4</td>
<td>12</td>
<td>x</td>
<td>x</td>
<td>Invalid</td>
</tr>
<tr>
<td>5</td>
<td>13</td>
<td>✓</td>
<td>✓</td>
<td>Extended Security</td>
</tr>
<tr>
<td>6</td>
<td>13</td>
<td>✓</td>
<td>x</td>
<td>NT Security</td>
</tr>
<tr>
<td>7</td>
<td>13</td>
<td>✓</td>
<td>✓</td>
<td>NT Security</td>
</tr>
<tr>
<td>8</td>
<td>13</td>
<td>x</td>
<td>x</td>
<td>NT Security</td>
</tr>
</tbody>
</table>

*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

- Free pool memory
- Random pool memory
- SrvNet look-aside buffers
- SrvNet "groom" buffer
- Session setup "allocation" buffer
- Session setup "hole" buffer
- Exploit OS/2 to NT FEA overflow
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

![Diagram showing EternalBlue Grooming with various buffer types and their possible activities.](image-url)
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

![Diagram showing memory regions and exploit types]

- Free pool memory
- Random pool memory
- SrvNet look-aside buffers
- SrvNet "groom" buffer
- Session setup "allocation" buffer
- Session setup "hole" buffer
- Exploit OS/2 to NT FEA overflow
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

HAL Heap

_SRVNET_WSK_STRUCT

Stage 0 Shellcode

Stage 1 Shellcode

DOUBLEPULSAR

NtOpenFile()

IA32_LSTAR MSR (CPU Syscall Register)

KiSystemCall()
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));
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

DataDisplacement = offset

```
trans->InData = &stack  
Local Variables  COOKIE  RET ExecuteTransaction()  ...  RET WorkerThread()
```
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
• 8 SMB per TCP packet
• 8 packets per attempt
EternalChampion RCE Trigger

- 8 SMB per TCP packet
- 8 packets per attempt
- 42 attempts
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
}
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_NAMEDPIPE)
    {
        RtlCopyMemory(transaction->InData, ...);

        transaction->InData += writeLength;
        transaction->DataCount += writeLength;
    }
}
Type Confusion Sequence
Type Confusion Sequence

NTCreateAndX

Named Pipe

FID

Mid = FID, PID = X

Primary Transaction

...
Type Confusion Sequence

- NTCreateAndX
- Named Pipe
- FID
- Primary Transaction
  - MID = FID, PID = X
- WriteAndX
  - OtherInfo = FID
- SrvAllocateTransaction() → _TRANSACTION
- SrvFindTransaction()
- InData += DataCount
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

<table>
<thead>
<tr>
<th>Free Heap memory</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fish (victim)</td>
</tr>
<tr>
<td>Dynamite (exploit)</td>
</tr>
</tbody>
</table>
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

<table>
<thead>
<tr>
<th></th>
<th>Free Heap memory</th>
<th>Fish (victim)</th>
<th>Dynamite (exploit)</th>
</tr>
</thead>
</table>
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

<table>
<thead>
<tr>
<th></th>
<th>Free Heap memory</th>
<th>Fish (victim)</th>
<th>Dynamite (exploit)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
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

<table>
<thead>
<tr>
<th></th>
<th>Free Heap memory</th>
<th>Fish (victim)</th>
<th>Dynamite (exploit)</th>
</tr>
</thead>
</table>
Fish-In-A-Barrel

<table>
<thead>
<tr>
<th>Free Heap memory</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fish (victim)</td>
</tr>
<tr>
<td>Dynamite (exploit)</td>
</tr>
</tbody>
</table>
### Fish-In-A-Barrel

<table>
<thead>
<tr>
<th></th>
<th>Free Heap memory</th>
<th>Fish (victim)</th>
<th>Dynamite (exploit)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Blue</td>
<td>Red</td>
<td></td>
</tr>
</tbody>
</table>
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

<table>
<thead>
<tr>
<th>Free Heap memory</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fish (victim)</td>
</tr>
<tr>
<td>Dynamite (exploit)</td>
</tr>
</tbody>
</table>
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

<table>
<thead>
<tr>
<th>Free Heap memory</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fish (victim)</td>
</tr>
<tr>
<td>Dynamite (exploit)</td>
</tr>
</tbody>
</table>
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”

<table>
<thead>
<tr>
<th>Free Heap memory</th>
<th>Grooms</th>
<th>Brides (victim)</th>
<th>Exploit (pointer shift)</th>
</tr>
</thead>
</table>

Diagram showing a lattice structure with categories for free heap memory, grooms, brides (victim), and exploit (pointer shift).
Matched Pairs “Lattice”

<table>
<thead>
<tr>
<th>Free Heap memory</th>
</tr>
</thead>
<tbody>
<tr>
<td>Grooms</td>
</tr>
<tr>
<td>Brides (victim)</td>
</tr>
<tr>
<td>Exploit (pointer shift)</td>
</tr>
</tbody>
</table>
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"

<table>
<thead>
<tr>
<th>Free Heap memory</th>
</tr>
</thead>
<tbody>
<tr>
<td>Grooms</td>
</tr>
<tr>
<td>Brides (victim)</td>
</tr>
<tr>
<td>Exploit (pointer shift)</td>
</tr>
</tbody>
</table>
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"

<table>
<thead>
<tr>
<th></th>
<th>Free Heap memory</th>
<th>Grooms</th>
<th>Brides (victim)</th>
<th>Exploit (pointer shift)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Blue</td>
<td></td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>White</td>
<td></td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Orange</td>
<td></td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>Black</td>
<td></td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
</tbody>
</table>
Matched Pairs “Lattice”

<table>
<thead>
<tr>
<th></th>
<th>Free Heap memory</th>
<th>Grooms</th>
<th>Brides (victim)</th>
<th>Exploit (pointer shift)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Matched Pairs “Lattice”

<table>
<thead>
<tr>
<th>Free Heap memory</th>
<th>Grooms</th>
<th>Brides (victim)</th>
<th>Exploit (pointer shift)</th>
</tr>
</thead>
</table>

[Diagram showing matched pairs with different colored sections representing Free Heap memory, Grooms, Brides (victim), and 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

![Diagram showing the Read-Where Primitive flow]

The diagram illustrates the flow of data and transaction parameters between an exploit and victim transaction, with specific data buffers involved in the process.

- **Exploit Transaction**: Includes parameters and data buffers.
- **Victim Transaction**: Similar structure to exploit with data and buffer components.
- **Leak Transaction**: Also includes parameters and data buffers.

Arrows indicate the direction of data flow, starting from the exploit transaction's input data and displacement, leading to the victim and leak transactions, showcasing the propagation of data through these transactions.
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张EEK_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

```
<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>SrvSmbWordCount</td>
<td></td>
</tr>
<tr>
<td>-2</td>
<td>SMB_COM_ILLEGAL_COMMAND</td>
</tr>
<tr>
<td>-2</td>
<td>SMB_COM_ILLEGAL_COMMAND</td>
</tr>
<tr>
<td>-1</td>
<td>SMB_COM_NT_TRANSACT</td>
</tr>
<tr>
<td>18</td>
<td>SMB_COM_NT_TRANSACT_SECONDARY</td>
</tr>
</tbody>
</table>
```

```
<table>
<thead>
<tr>
<th>SrvTransaction2DispatchTable</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>SrvSmbOpen2</td>
<td></td>
</tr>
<tr>
<td>SrvSmbFindFirst2</td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
<tr>
<td>SrvTransactionNotImplemented</td>
<td>0x14 (Session Setup)</td>
</tr>
<tr>
<td>SrvTransactionNotImplemented</td>
<td>0x15</td>
</tr>
<tr>
<td>SrvSmbGetDfsReferral</td>
<td></td>
</tr>
<tr>
<td>SrvSmbReportDfsInconsistency</td>
<td></td>
</tr>
</tbody>
</table>
```
EternalRomance Info Leak Patch #1

SrvSmbTransaction() *Before:*

```c
if (subCommand == TRANS_PEEK_NMPIPE)
{
    maxParameterCount = MAX(16, maxParameterCount);
}
SrvAllocateTransaction(&Transaction, ...);
Transaction->MaxParameterCount = maxParameterCount;
```
SrvSmbTransaction() After:

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

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

```c
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:

```cpp
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:

\[ \text{RtlCopyMemory}(\text{transaction->InData} + \text{transaction->DataCount}, \ldots); \]

\[ \text{transaction->InData} += \text{writeLength}; \]
\[ \text{transaction->DataCount} += \text{writeLength}; \]
1. SrvSmbNtTransaction/SrvSmbTransaction() New Code:

SrvAllocateTransaction(&Transaction, ...)

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

SrvInsertTransaction(&Transaction);
1. `SrvSmbNtTransaction/SrvSmbTransaction()` *New Code*:

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

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

SrvInsertTransaction(&Transaction);
```

2. `SrvFindTransaction()` *New Code*:

```c
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)

<table>
<thead>
<tr>
<th>Type</th>
<th>Pointer Dereference</th>
<th>Offset</th>
</tr>
</thead>
<tbody>
<tr>
<td>WORK_QUEUE</td>
<td>Connection-&gt;PreferredWorkQueue</td>
<td>variadic</td>
</tr>
</tbody>
</table>

- **Given:** Connection
### Quest for RWX Memory (via remote read)

<table>
<thead>
<tr>
<th>Type</th>
<th>Pointer Dereference</th>
<th>Offset</th>
</tr>
</thead>
<tbody>
<tr>
<td>WORK_QUEUE</td>
<td>Connection-&gt;PreferredWorkQueue</td>
<td>variadic</td>
</tr>
<tr>
<td>KTHREAD</td>
<td>PreferredWorkQueue-&gt;IrpThread</td>
<td>0x198</td>
</tr>
</tbody>
</table>

- **Given:** Connection
Quest for RWX Memory (via remote read)

<table>
<thead>
<tr>
<th>Type</th>
<th>Pointer Dereference</th>
<th>Offset</th>
</tr>
</thead>
<tbody>
<tr>
<td>WORK_QUEUE</td>
<td>Connection-&gt;PreferredWorkQueue</td>
<td>variadic</td>
</tr>
<tr>
<td>KTHREAD</td>
<td>PreferredWorkQueue-&gt;IrpThread</td>
<td>0x198</td>
</tr>
<tr>
<td>KPROCESS</td>
<td>IrpThread-&gt;Process</td>
<td>0x220</td>
</tr>
</tbody>
</table>

- Given: Connection
### Quest for RWX Memory (via remote read)

<table>
<thead>
<tr>
<th>Type</th>
<th>Pointer Dereference</th>
<th>Offset</th>
</tr>
</thead>
<tbody>
<tr>
<td>WORK_QUEUE</td>
<td>Connection-&gt;PreferredWorkQueue</td>
<td>variadic</td>
</tr>
<tr>
<td>KTHREAD</td>
<td>PreferredWorkQueue-&gt;IrpThread</td>
<td>0x198</td>
</tr>
<tr>
<td>KPROCESS</td>
<td>IrpThread-&gt;Process</td>
<td>0x220</td>
</tr>
<tr>
<td>PVOID</td>
<td>KProcess-&gt;ProcessListEntry.Blink</td>
<td>0x240</td>
</tr>
</tbody>
</table>

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

* [https://www.geoffchappell.com/studies/windows/km/ntosknl/structs/kprocess/index.htm](https://www.geoffchappell.com/studies/windows/km/ntosknl/structs/kprocess/index.htm)
## Quest for RWX Memory (via remote read)

<table>
<thead>
<tr>
<th>Type</th>
<th>Pointer Dereference</th>
<th>Offset</th>
</tr>
</thead>
<tbody>
<tr>
<td>WORK_QUEUE</td>
<td>Connection-&gt;PreferredWorkQueue</td>
<td>variadic</td>
</tr>
<tr>
<td>KTHREAD</td>
<td>PreferredWorkQueue-&gt;IrpThread</td>
<td>0x198</td>
</tr>
<tr>
<td>KPROCESS</td>
<td>IrpThread-&gt;Process</td>
<td>0x220</td>
</tr>
<tr>
<td>PVOID</td>
<td>KProcess-&gt;ProcessListEntry.Blink</td>
<td>0x240</td>
</tr>
</tbody>
</table>

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

* https://www.geoffchappell.com/studies/windows/km/ntoskrl/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
- @msftsecresponse
- @swithak
  - https://swithak.github.io/SH20TAATSB18/Home/
- @francisckrs
- @msuiche
Thanks!

zerosum0x0