UNDERSTANDING ETERNALBLUE

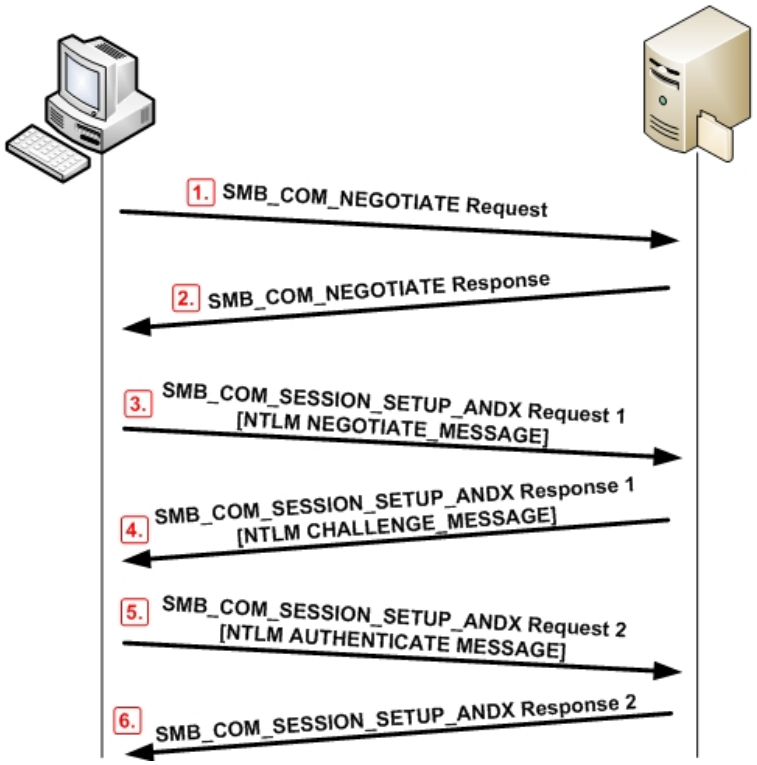
ENPM696 Reverse Software Engineering

Vinaykumar Yennam
Jorge Damian
Dheepshika Raghunathan

LET'S BEGIN!

- We are...
- Eternal blue is an exploitation tool
  - Execute remote shellcode
  - Created by NSA
  - Made public by Shadow Brokers
- Exploits vulnerabilities in SMB protocol
- Used in WannaCry

SMB PROTOCOL CONT'D



- Server Message Block protocol
- Originally open-source and Linux based
  - Modified to operate on Windows
- Used for
  - Shared access to files, printers, and serial ports
  - Miscellaneous communications between nodes on a network
  - Mechanism for authenticated inter-process communication

BUGS IN SMB PROTOCOL

- Bug A - Wrong Casting Bug
  - Occurs when converting File Extended Attributes(FEA) from Os2 structure to NT structure
- Bug B - Wrong Parsing Function Bug
  - Occurs when the wrong parsing function is called when transferring large parameter and data blocks
- Bug C - Non-paged Pool Allocation Bug
  - Allocates large chunk of memory in the kernel non-paged pool, where shellcode could be placed

BUG A

WRONG CASTING BUG

FEA FORMATS

OS2

```
struct Os2Fea{
    UCHAR   ExtendedAttributeFlag; // Flags
    UCHAR   AttributeNameLengthInBytes; // Length of the AttributeName field
    USHORT  AttributeValueLengthInBytes; // Length of the AttributeValue field
    UCHAR   AttributeName[AttributeNameLengthInBytes + 1]; // Extended attribute name
    UCHAR   AttributeValue [AttributeValueLengthInBytes]; // Extended attribute value
}

struct Os2FeaList{
    ULONG   SizeOfListInBytes; // The total size of the FeaRecords + 4 bytes
    UCHAR   Os2FeaRecords[SizeOfListInBytes-4];// A concatenated list of Os2Fea
}
```

NT

```
struct NtFeaList{
  ULONG  NextEntryOffset; // offset to the next NtFea record of NtFeaList type
  UCHAR  Flags;
  UCHAR  NtFeaNameLength;
  USHORT NtFeaValueLength;
  CHAR   NtFeaName[NtFeaNameLength];
  CHAR   NtFeaValue[NtFeaValueLength];
}
```

INTERESTING FUNCTIONS

- Available in the srv.sys driver

1. SrvOs2FeaListToNt

   Converts Os2 FEA List to NT FEA List

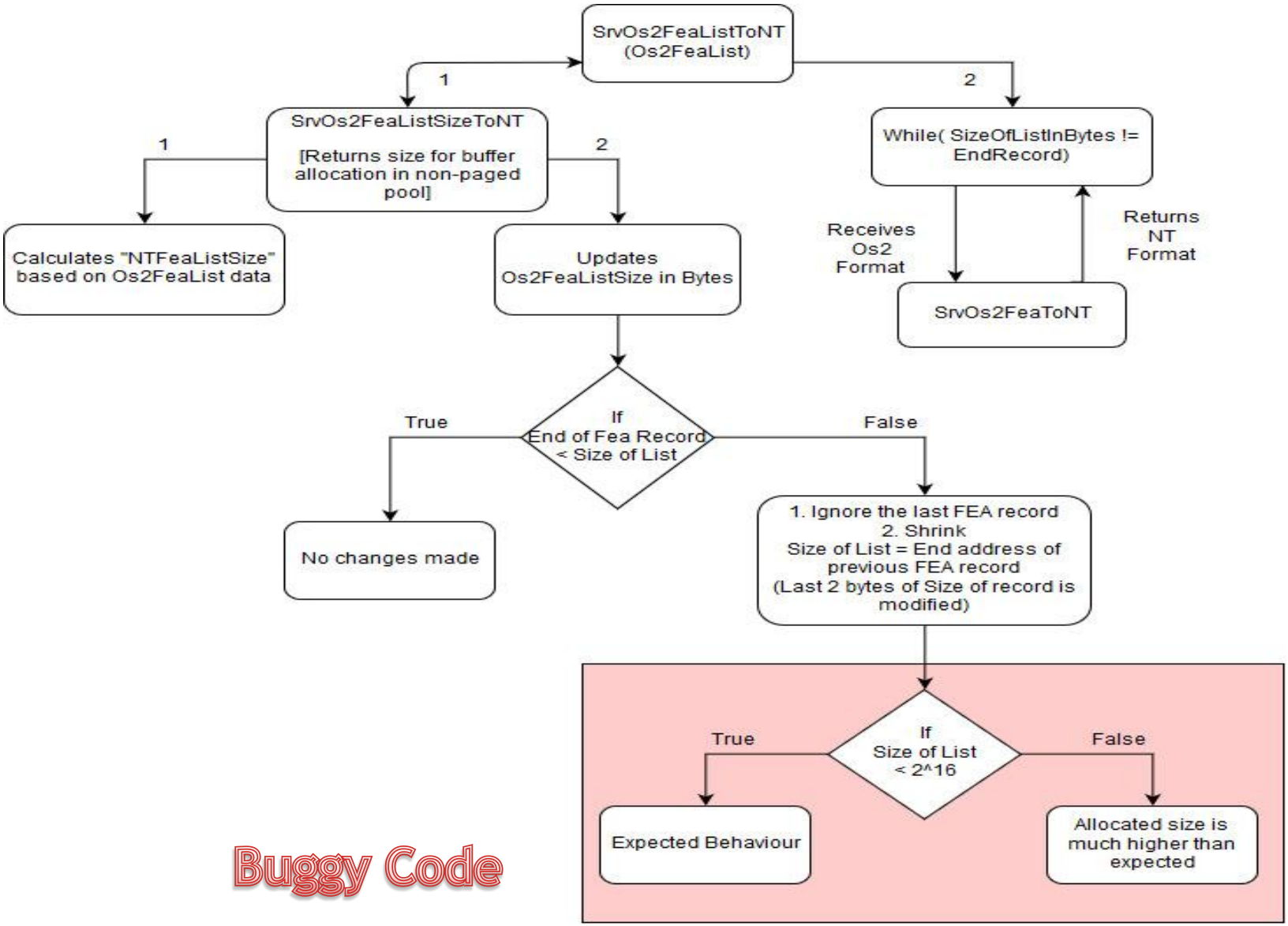
2. SrvOs2FeaListSizeToNT

   Calculates the size needed to convert Os2FeaList structures into the appropriate NtFeaList structures

3. SrvOs2FeaToNT

   Converts Os2 record to NT record

CONTROL FLOW LOGIC



Buggy Code

THE BUGGY FUNCTION

```
; Attributes: bp-based frame

; __stdcall SrvOs2FeaListSizeToNt(x)
_SrvOs2FeaListSizeToNt@4 proc near

NTFeaListSize= dword ptr -4
pOs2FeaList= dword ptr  8

mov     edi, edi
push    ebp
mov     ebp, esp
push    ecx
and     [ebp+NTFeaListSize], 0 ; Initializing local variable to 0
push    edi
mov     edi, [ebp+pOs2FeaList] ; edi = pOs2FeaList
mov     edx, [edi]       ; edx=SizeOfListInBytes
add     edx, edi         ; edx=AddrOfEndOfRecords
lea     ecx, [edi+4]     ; ecx=StartOfFirstRecord
cmp     ecx, edx         ; AddrOfFirstRecord > AddrOfEndRecord -->True
                         ; True says no valid records
jnb     short returnFeaListSize ; ja to no valid records
```

```
push    ebx              ; Store register state
push    esi              ; Store register state
```

```
while (*CurrentOs2FeaRecord.AttributeName < EndOfAddressOfOs2FeaList)

whileCondition:          ; Move next record to ebx
lea     ebx, [ecx+4]
cmp     ebx, edx         ; Compare current record with last record address
jnb     short BugA_loc   ; Jump to loop end when ebx(current record address) > edx(last record address)
```

THE BUGGY FUNCTION CON'T

```
while (*CurrentOs2FeaRecord.AttributeName < EndOfAddressOfOs2FeaList)

whileCondition:          ; Move next record to ebx
lea     ebx, [ecx+1]
cmp     ebx, edx         ; Compare current record with last record address
jnb     short BugA_loc   ; Jump to loop end when ebx(current record address) > edx(last record address)
```

```
if(&CurrentOs2FeaRecord.AttributeName[currentOs2FeRecordSize +1] >
EndAddressofOs2FeaList)
movzx   eax, word ptr [ecx+2] ; eax=CurrentOs2FeaRecord.AttributeValueLengthInBytes
movzx   esi, byte ptr [ecx+1] ; esi=CurrentOs2FeaRecord.AttributeNameLengthInBytes
add     esi, eax        ; It calculates length of current record.
lea     eax, [esi+ebx+1] ; eax = EndofCurrentOs2FeaRecordAddress
cmp     eax, edx        ; compare CurrentRecordAddress with EndOfRecordAddress
ja      short BugA_loc  ; If True: Remove current record and shrink it
```

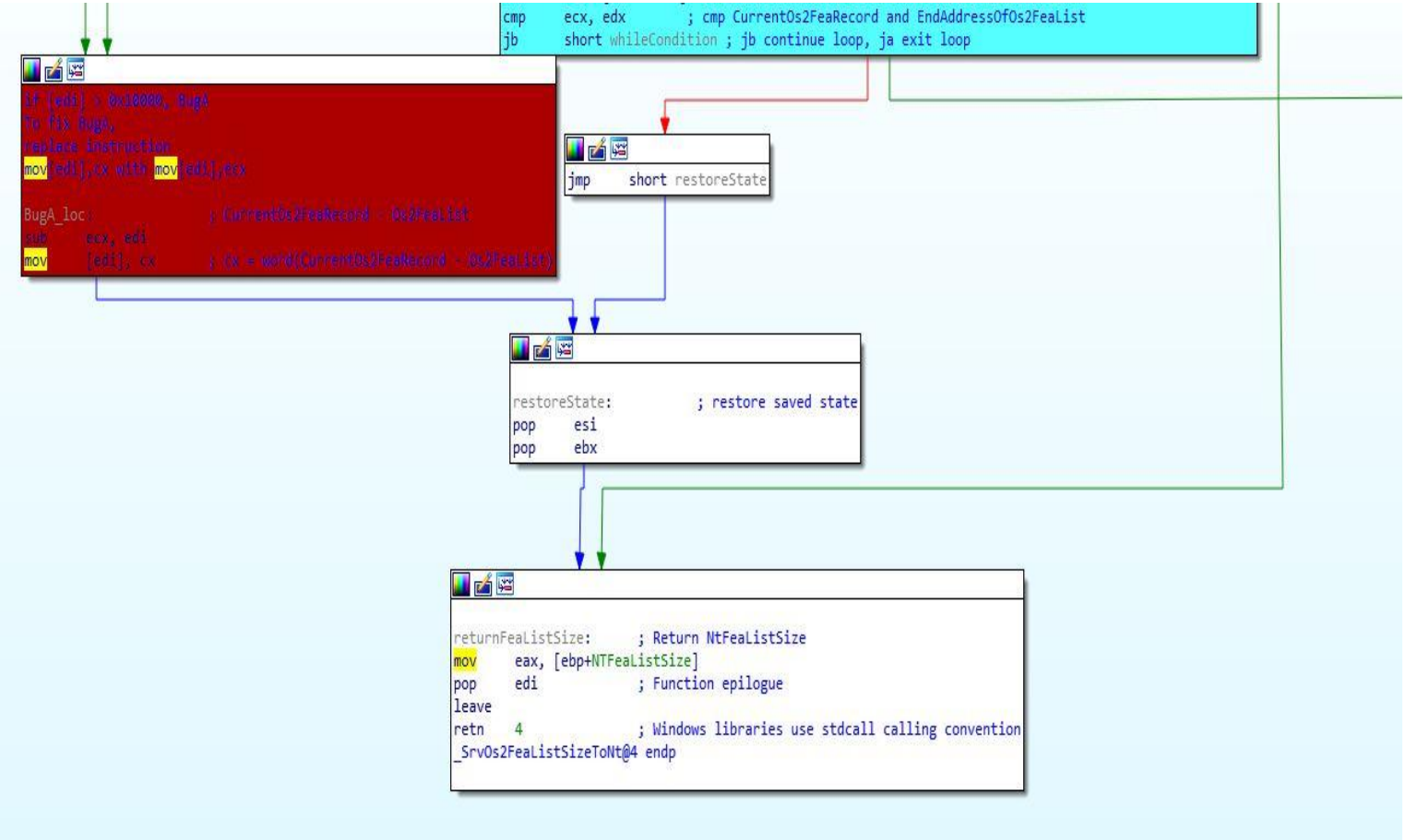
```
if ((RtlSizeTAdd(NTFeaListSize,(CurrentOs2FeaRecordSize + 12) &
0xfffffffc, &NtFeaListSize)&0x80000000)!=0)
lea     eax, [esi+0Ch]
and     eax, 0FFFFFFFCh ; 4-byte Alignment
add     [ebp+NTFeaListSize], eax ; mov esi value to NtFeaListSize variable
lea     ecx, [ecx+esi+5] ; CurrentOs2FeaRecord = CurrentOs2FeaRecord + CurrentOs2FeaRecordSize + 5
cmp     ecx, edx        ; cmp CurrentOs2FeaRecord and EndAddressOfOs2FeaList
jb      short whileCondition ; jb continue loop, ja exit loop
```

```
if [edi] > 0x10000, BugA
To fix BugA,
replace instruction
mov[edi],cx with mov[edi],ecx

BugA_loc:                ; CurrentOs2FeaRecord - Os2FeaList
sub     ecx, edi
mov     [edi], cx        ; cx = word(CurrentOs2FeaRecord - Os2FeaList)
```

```
jmp     short restoreState
```

THE BUGGY FUNCTION CON'T



THE BUGGY CODE

```
if [edi] > 0x10000, BugA
To fix BugA,
replace instruction
mov[edi],cx with mov[edi],ecx

BugA_loc:                ; CurrentOs2FeaRecord - Os2FeaList
sub     ecx, edi
mov     [edi], cx        ; cx = word(CurrentOs2FeaRecord - Os2FeaList)
```

PSEUDOCODE

```
...

NtFeaListSize = 0
EndAddressOfOs2FeaList = (char *)Os2FeaList + Os2FeaList->SizeOfListInBytes
CurrentOs2FeaRecord = &Os2FeaList->FeaRecords;


if(CurrentOs2FeaRecord < EndAddressOfOs2FeaList){

    while(CurrentOs2FeaRecord->AttributeName < EndAddressOfOs2FeaList)

    CurrentOs2FeaRecordSize = CurrentOs2FeaRecord->AttributeValueLengthInByters + CurrentOs2FeaRecord->AttributeNameLengthInBytes;

    //The conditions to this break are:
    //1. The start address of the AttributeName variable in CurrentFeaRecord is smaller than the EndAddressOfFeaList
    //2. The end address of CurrentFeaRecord is bigger than the EndAddressOfFeaList

    if(&CurrentOs2FeaRecord->AttributeName[CurrentOs2FeaRecordSize + 1] > EndAddressOfOs2FeaList)
        break;

    ...


}

...
```

BUG B

WRONG PARSING FUNCTION BUG

DATA RELATED FUNCTIONS

SMB_COM_TRANSACTION2

```
SMB_Parameters
  {
  UCHAR  WordCount;
  Words
    {
    USHORT TotalParameterCount;
    USHORT TotalDataCount;
    USHORT MaxParameterCount;
    USHORT MaxDataCount;
    UCHAR  MaxSetupCount;
    UCHAR  Reserved1;
    USHORT Flags;
    ULONG  Timeout;
    USHORT Reserved2;
    USHORT ParameterCount;
    USHORT ParameterOffset;
    USHORT DataCount;
    USHORT DataOffset;
    UCHAR  SetupCount;
    UCHAR  Reserved3;
    USHORT Setup[SetupCount];
    }
  }
SMB_Data
  {
  USHORT ByteCount;
  Bytes
    {
    UCHAR Name;
    UCHAR Pad1[];
    UCHAR Trans2_Parameters[ParameterCount];
    UCHAR Pad2[];
    UCHAR Trans2_Data[DataCount];
    }
  }
```

SMB_COM_NT_TRANSACT

```
SMB_Parameters
  {
  UCHAR  WordCount;
  Words
    {
    UCHAR  MaxSetupCount;
    USHORT Reserved1;
    ULONG  TotalParameterCount;
    ULONG  TotalDataCount;
    ULONG  MaxParameterCount;
    ULONG  MaxDataCount;
    ULONG  ParameterCount;
    ULONG  ParameterOffset;
    ULONG  DataCount;
    ULONG  DataOffset;
    UCHAR  SetupCount;
    USHORT Function;
    USHORT Setup[SetupCount];
    }
  }
SMB_Data
  {
  USHORT ByteCount;
  Bytes
    {
    UCHAR  Pad1[];
    UCHAR  NT_Trans_Parameters[ParameterCount];
    UCHAR  Pad2[];
    UCHAR  NT_Trans_Data[DataCount];
    }
  }
```

EXPLAINING THE FUNCTIONS

- SMB_COM_TRANSACTION2
  - Provide support for a richer server-side file system semantics.
  - Trans2 subcommands: allow clients to set and retrieve Extended Attribute key/value pairs, make use of long file names, perform directory searches, etc.

- SMB_COM_NT_TRANSACT
  - Extend the file system feature access offered by SMB_COM_TRANSACTION2.
  - Allow for the transfer of very large parameter and data blocks.

THE WRONG PARSING BUG

- If the data sent via SMB_COM_TRANSACTION2 or by SMB_COM_NT_TRANSACT exceeds the MaxBufferSize established during session setup, or total data to send is bigger than transmitted_data, then the transaction uses the SECONDARY sub-command.
  - Each sub-command has a corresponding sub-command _SECONDARY.
  - The packets that follow the first sub-command have the corresponding _SECONDARY sub-command set as their command.
  - Ex.: SMB_COM_NT_TRANSACT => SMB_COM_NT_TRANSACT_SECONDARY

THE WRONG PARSING BUG, CONT.

- In SMB_COM_TRANSACTION2, the maximum data that can be sent is represented by a parameter in the header of SMB_COM_TRANSACTION2 in the field of a Word size (0xFFFF).
- However, in SMB_COM_NT_TRANSACT, the maximum data that can be sent is represented by a parameter in the header of SMB_COM_NT_TRANSACT in the field of Dword size (0XFFFFFFFF).
- However, there is no validation for which function started the transaction. Thus, it's possible to send SMB_COM_NT_TRANSACT followed by SMB_COM_TRANSACTION2_SECONDARY. This situation can lead to wrong data parsing, and this bug enables Bug A by treating Dword as Word.

BUG C
NON-PAGED POOL ALLOCATION BUG

CONFIGURING THE SMB SESSION

- An SMB_COM_SESSION_SETUP_ANDX request MUST be sent by a client to begin user authentication on an SMB connection and establish an SMB session.
  - At least one SMB_COM_SESSION_SETUP_ANDX MUST be sent to perform a user logon to the server and to establish a valid UID.
  - There are two formats for an SMB_COM_SESSION_SETUP_ANDX request: LM and NTLM authentication, and NTLMv2 (NTLM SSP).

SMB_COM_SESSION_SETUP_ANDX

LM and NTLM

```
SMB_Parameters
  {
  UCHAR  WordCount;
  Words
    {
    UCHAR  AndXCommand;
    UCHAR  AndXReserved;
    USHORT AndXOffset;
    USHORT MaxBufferSize;
    USHORT MaxMpxCount;
    USHORT VcNumber;
    ULONG  SessionKey;
    USHORT OEMPasswordLen;
    USHORT UnicodePasswordLen;
    ULONG  Reserved;
    ULONG  Capabilities;
    }
  }
SMB_Data
  {
  USHORT ByteCount;
  Bytes
    {
    UCHAR      OEMPassword[];
    UCHAR      UnicodePassword[];
    UCHAR      Pad[];
    SMB_STRING AccountName[];
    SMB_STRING PrimaryDomain[];
    SMB_STRING NativeOS[];
    SMB_STRING NativeLanMan[];
    }
  }
```

NTLMv2 (NTLM SSP)

```
SMB_Parameters
  {
  UCHAR  WordCount;
  Words
    {
    UCHAR  AndXCommand;
    UCHAR  AndXReserved;
    USHORT AndXOffset;
    USHORT MaxBufferSize;
    USHORT MaxMpxCount;
    USHORT VcNumber;
    ULONG  SessionKey;
    USHORT SecurityBlobLength;
    ULONG  Reserved;
    ULONG  Capabilities;
    }
  }
SMB_Data
  {
  USHORT ByteCount;
  Bytes
    {
    UCHAR      SecurityBlob[SecurityBlobLength];
    SMB_STRING NativeOS[];
    SMB_STRING NativeLanMan[];
    }
  }
```

SMB_COM_SESSION_SETUP_ANDX, CONT.

- In both formats, the request is split into 2 sections:
  - SMB_Parameters – Contains parameters of sizes between 1-4 bytes. The WordCount field represents the total length of SMB_Parameters struct members in a Word size.
  - SMB_Data – Contains data in a variable size. The ByteCount field represents the length of the SMB_Data struct members section in bytes.
- Summing the size of the fields, in the first format, the WordCount equals 13 and in the second format (extended security), the WordCount equals 12.
- The SMB_COM_SESSION_SETUP_ANDX request is handled by the BlockingSessionSetupAndX function. This function wrongly calculates ByteCount, which leads to an allocation of controlled size – bigger than the packet data – in the non-paged pool.

UNDERSTANDING THE BUG

- Sending an SMB_COM_SESSION_SETUP_ANDX request as Extended Security (WordCount 12) with CAP_EXTENDED_SECURITY, but without FLAGS2_EXTENDED_SECURITY, the request will be processed wrongly as an NT Security request (WordCount 13) by the GetNtSecurityParameters function.

UNDERSTANDING THE BUG, CONT.

- As a result, the function reads ByteCount from the wrong offset in the struct, and allocates space in the non-paged kernel pool for NativeOs and NativeLanMan unicode strings.
  - This bug allows you to send a small packet that leads to a big allocation in the non-paged pool, which is used to create a big allocation as a placeholder.
  - This allocation will later be freed (creating a HOLE) and allocated again by an NtFea chunk that will overflow the next chunk.

THE BUGGY CODE

Search

Tag Scope

Idx	Name	Blo...	Size
512	sub_35b98	332	5590
287	sub_3f1ae	293	4250
3...	sub_21420	268	4075
356	BlockingSessionSetupAndX	237	3584
351	sub_23133	171	2778
125	sub_158d1	147	2719
94	sub_55d2d	131	2449
341	sub_274c9	119	2108
419	sub_2a7a9	70	2064
172	sub_26c8f	131	2036
186	sub_1b680	147	1982

Remove Hi/LO macros Remove potentially dead code Remove NOPs


```
int BlockingSessionSetupAndX() {
    edi = edi;
    var_20 = var_20 & 0x0;
    var_30 = var_30 & 0x0;
    esp = esp - 0x60;
    esi = ecx;
    eax = esi + 0x160;
    var_14 = esi;
    if (*(int8_t *)eax == 0x4c) {
            *(int8_t *)eax = 0x24;
    }
    esp = esp - 0x4;
    sub_118a0(esi);
    ebx = *(esi + 0x4c);
    if (*(int8_t *)(ebx + 0x1) != 0x2) goto loc_313b4;

loc_22267:
    ecx = *(esi + 0x6c);
    var_8 = 0x0;
    var_34 = *(esi + 0x74);
    var_2 = *(int8_t *)(ebx + 0x7b);
    eax = *(ebx + 0x60);
    esp = esp - 0x4;
```

THE BUGGY CODE CON'T

```
        }
        edx = *(ebx + 0x12c);
        edx = edx >> 0x1f & 0x1;
        var_1 = edx;
        if (edx != 0x0) {
                edx = *(int16_t *)(*(esi + 0x68) + 0xa);
                edx = edx >> 0xb & 0x1;
                var_1 = edx;
        }
        var_1C = (*(int16_t *)(*(esi + 0x68) + 0xa) & 0xffff) >> 0xf & 0x1;
        if ((*(ebx + 0x60) > 0x1) || (edx == 0x0)) goto loc_31599;
```

```
loc_31599:
    stack[-104] = &var_28;
    stack[-108] = &var_10;
    esp = esp - 0x24;
    eax = GetNtSecurityParameters(esi, &var_3C, &var_40, &var_14, &var_38, &var_54, &var_4C, stack[-108], stack[-104]);
    var_C = eax;
    if (eax >= 0x0) {
            var_8 = 0x0;
            stack[-104] = &var_4C;
            stack[-108] = &var_54;
            esp = esp - 0xc;
            eax = sub_23a3b(&var_8, stack[-108], stack[-104]);
            eax = var_8;
            if (eax != 0x0) {
                    stack[-104] = &var_30;
                    stack[-108] = var_24;
                    esp = esp - 0x28;
                    eax = sub_2b75f(eax + 0x80, eax, *(esi + 0x4c), &var_54, var_14, var_38, var_3C, var_40, stack[-108], stack[-104]);
                    var_C = eax;
            }
            else {
                    var_C = 0xc0000205;
            }
```

THE BUGGY CODE CON'T

```
            }
            else {
                    var_24 = 0x0;
            }
    }
    edx = *(ebx + 0x12c);
    edx = edx >> 0x1f & 0x1;
    var_1 = edx;
    if (edx != 0x0) {
            edx = *(int16_t *)(*(esi + 0x68) + 0xa);
            edx = edx >> 0xb & 0x1;
            var_1 = edx;
    }
    var_1C = (*(int16_t *)(*(esi + 0x68) + 0xa) & 0xffff) >> 0xf & 0x1;
    if ((*(ebx + 0x60) > 0x1) || (edx == 0x0)) goto loc_31599;

loc_22390:
    COND = *(int8_t *)(ecx + 0x1) != 0xff;
    var_38 = *(esi + 0x74);
    if (COND) goto loc_314a8;

loc_223a0:
    if (*(int16_t *)(ecx + 0x9) == 0x0) {
            esp = esp - 0x8;
            sub_22ae2(ebx, 0x0);
    }
    stack[-104] = &var_28;
    stack[-108] = &var_10;
    esp = esp - 0x14;
    eax = GetExtendedSecurityParameters(esi, &var_40, &var_3C, stack[-108], stack[-104]);
    var_C = eax;
    if (eax < 0x0) goto loc_2252e;

loc_223d0:
    eax = *(esi + 0x68);
    eax = *(int16_t *)(eax + 0x1c) & 0xffff;
```

THE BUGGY CODE CON'T

```
loc_223a0:
    if (*(int16_t *)(ecx + 0x9) == 0x0) {
            esp = esp - 0x8;
            sub_22ae2(ebx, 0x0);
    }
    stack[-104] = &var_28;
    stack[-108] = &var_10;
    esp = esp - 0x14;
    eax = GetExtendedSecurityParameters(esi, &var_40, &var_3C, stack[-108], stack[-104]);
    var_C = eax;
    if (eax < 0x0) goto loc_2252e;

loc_223d0:
    eax = *(esi + 0x68);
    eax = *(int16_t *)(eax + 0x1c) & 0xffff;
```

SUMMARY

THANK YOU!