Xcon | XAUTOCAR

XCON XFOCUS INFORMATION SECURITY CONFERENCE

2017

8.23-24

XCON安全焦点
信息安全技术峰会

Fixed, or not fixed, that is the question

Yunhai Zhang

目录

CONTENTS

01

Background

- **Mitigations 5 years ago**

- **Mitigations nowadays**

- **Microsoft launches Bounty Programs at 2013**

Microsoft Bounty Programs

MICROSOFT BOUNTY PROGRAM
MSRC

aka.ms/bugbounty

Calling all Microsoft friends, hackers, and researchers! Do you want to help us protect customers, making some of our most popular products better... and earn money doing so? Step right up!

Microsoft offers direct payments in exchange for reporting certain types of vulnerabilities and exploitation techniques.

Microsoft has championed many initiatives to advance security and to help protect our customers, including the Security Development Lifecycle (SDL) process and Coordinated Vulnerability Disclosure (CVD). We formed industry collaboration programs such as the Microsoft Active Protections Program (MAPP) and Microsoft Vulnerability Research (MSVR),and created the BlueHat Prize to encourage research into defensive technologies. Since June 2013, we've also offered bounties for certain classes of vulnerabilities reported to us. These bounty programs help Microsoft harness the collective intelligence and capabilities of security researchers to help protect customers. As you'll see from the list below, several time-limited programs apply only to preview versions, so we can address the vulnerabilities before the final version is complete.

- Microsoft launches Bounty Programs at 2013
- Many novel bypass techniques are submitted
- Most of them are fixed now
- Sometimes the fix did not really fix the problem

02

Exploit ATL Thunk Pool

2.1 The Original Issue

- **ATL window will allocate Thunk Pool when created**

```
HWND __thiscall ATL::CWindowImplBaseT<ATL::CWindow,ATL::CWinTraits<1442840576,0>>::Create(HMENU this, int a2,
{
  HMENU v9; // esi@1
  struct ATL::_AtlCreateWndData *v10; // edi@1
  ATL::CAtlWinModule *v11; // ecx@1
  HWND result; // eax@2
  HMENU v13; // ecx@5
  void *v14; // eax@8

  v9 = this;
  v10 = (this + 2);
  if ( ATL::CWndProcThunk::Init((this + 2), 0, 0) )
  {
    if ( a8 )
    {
      ATL::CAtlWinModule::AddCreateWndData(v11, v10, v9);
      v13 = hMenu;
      if ( !hMenu && dwStyle & 0x40000000 )
        v13 = v9;
      v14 = a3;
      if ( !a3 )
        v14 = &ATL::CWindow::rcDefault;
      result = CreateWindowExW(
```

2.1 The Original Issue

- **ATL window will allocate Thunk Pool when created**

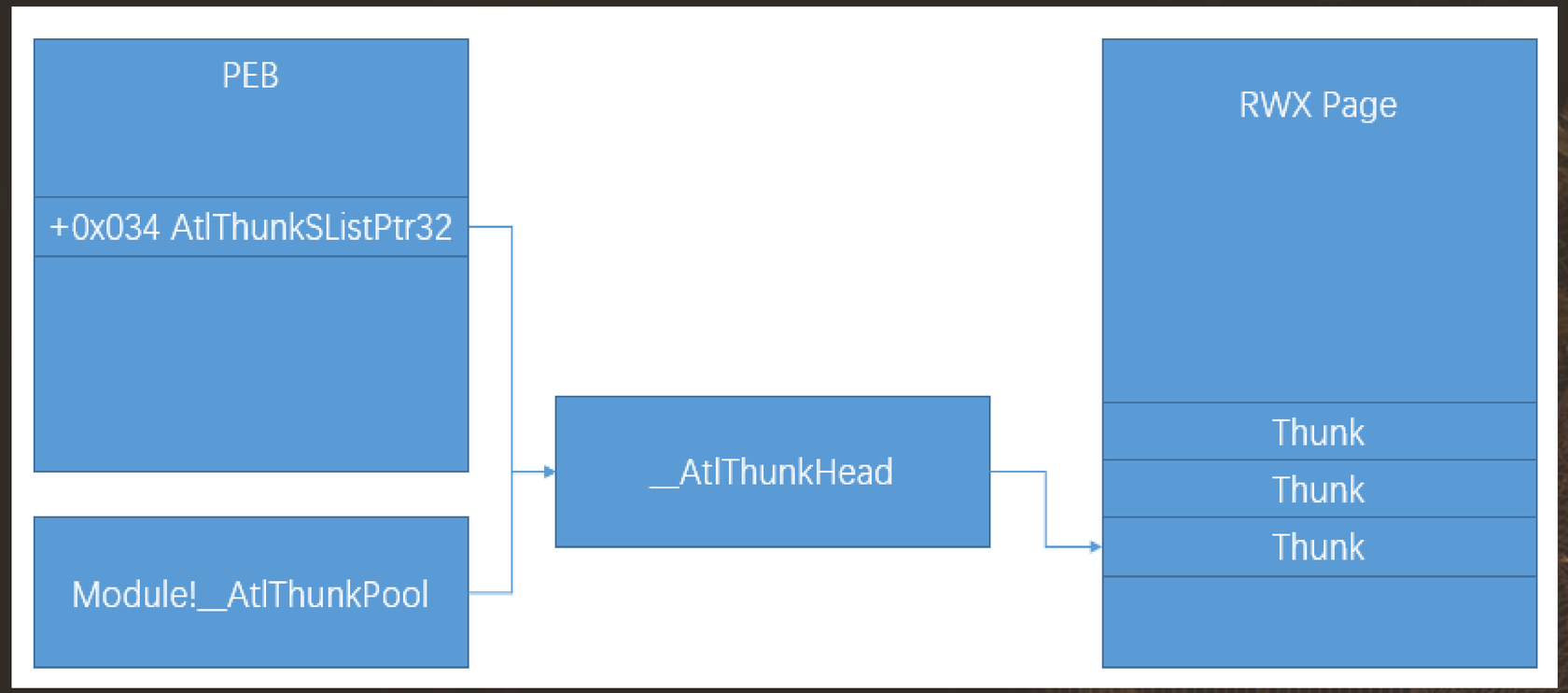
```
int __thiscall ATL::CWndProcThunk::Init(ATL::CWndProcThunk *this, __int32 (__stdcall *a2)(HWND, unsigned int,
{
  signed int status; // esi@1
  void *v5; // eax@2
  int v6; // eax@5
  int v7; // ST04_4@5
  HANDLE v8; // eax@5

  status = 0;
  if ( *(this + 3) || (v5 = __AllocStdCallThunk_cmn(), (*(this + 3) = v5) != 0) )
  {
    if ( a2 || a3 )
    {
      v6 = *(this + 3);
      v7 = *(this + 3);
      *v6 = 0x42444C7;
      *(v6 + 4) = a3;
      *(v6 + 8) = 0xE9u;
      *(v6 + 9) = a2 + -v6 - 0xD;
      v8 = GetCurrentProcess();
      FlushInstructionCache(v8, v7, 0xDu);
    }
    status = 1;
  }
  return status;
}
```

2.1 The Original Issue

- **ATL Thunk Pool is PAGE_EXECUTE_READWRITE**

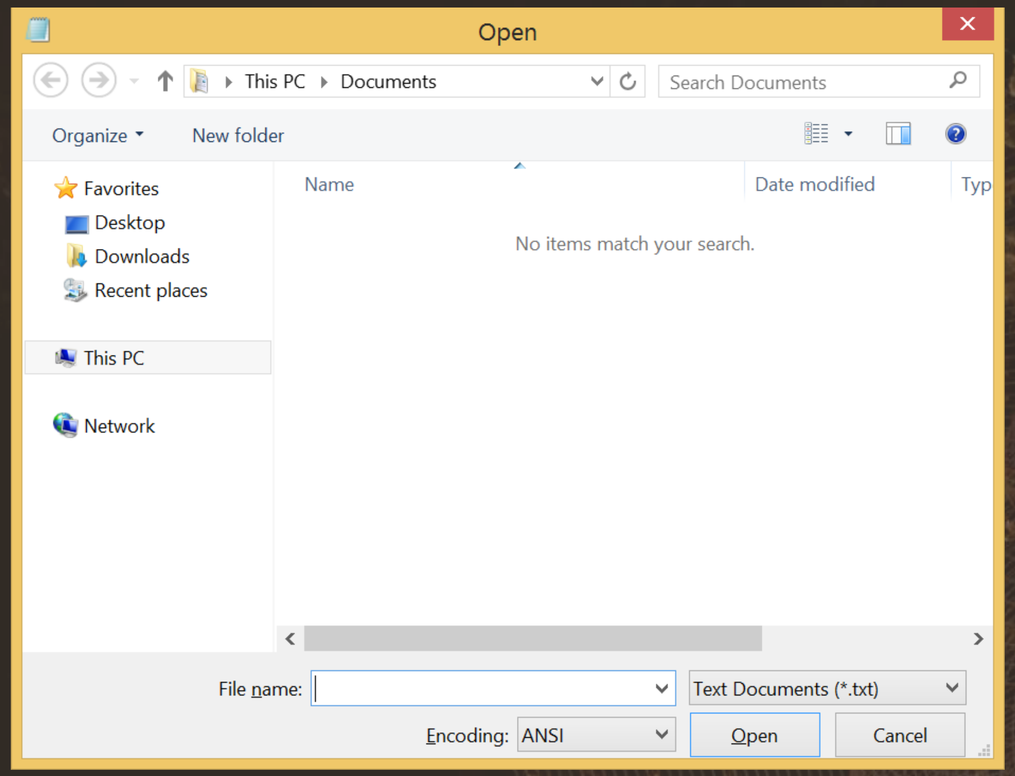
```
void *__stdcall __AllocStdCallThunk_cmn()
{
  union _SLIST_HEADER *v0; // eax@1
  HANDLE v1; // eax@5
  void *result; // eax@5
  LPVOID v3; // eax@8 MAPDST
  int v5; // eax@10
  PSINGLE_LIST_ENTRY v6; // edi@10
  unsigned int v7; // edi@12

  v0 = __AtlThunkPool;
  if ( !__AtlThunkPool )
  {
    if ( !_InitializeThunkPool() )
      return 0;
    v0 = __AtlThunkPool;
  }
  if ( v0 == 1 )
  {
    v1 = GetProcessHeap();
    result = HeapAlloc(v1, 0, 0xDu);
    if ( result )
      return result;
    return 0;
  }
  result = InterlockedPopEntrySList(v0);
  if ( result )
    return result;
  v3 = VirtualAlloc(0, 0x1000u, 0x1000u, 0x40u);
```

- How to locate ATL Thunk Pool



2.1 The Original Issue

- Who use ATL Thunk Pool



2.1 The Original Issue

- **Exploit Plan**
  - **Insert flash to allocate ATL Thunk Pool**
  - **Read the address of ATL Thunk Pool**
  - **Write shellcode to ATL Thunk Pool**
  - **Execute the shellcode**

2.2 The 1st Fix of the Issue

- __AllocStdCallThunk_cmn is replaced by AtlThunk_AllocateData

```
int __thiscall ATL::CWndProcThunk::Init(ATL::CWndProcThunk *this, __int32 (__stdcall *a2)(HWND, unsigned int, u
{
  signed int status; // esi@1
  _DWORD *v5; // eax@2

  status = 0;
  if ( *(this + 3) || (v5 = AtlThunk_AllocateData(), (*(this + 3) = v5) != 0) )
  {
    AtlThunk_InitData(*(this + 3), a2, a3);
    status = 1;
  }
  return status;
}
```

2.2 The 1st Fix of the Issue

- **AtlThunk_AllocateData is implemented in atlthunk.dll**

```
_DWORD *__stdcall AtlThunk_AllocateData()
{
  HANDLE v0; // eax@1
  void *v1; // ecx@1
  _DWORD *v2; // edi@1
  int (*v4)(void); // eax@3 MAPDST
  void *v6; // eax@4
  HANDLE v7; // eax@9
  int v8; // [sp+0h] [bp-10h]@4

  v0 = GetProcessHeap();
  v2 = HeapAlloc(v0, 8u, 8u);
  if ( !v2 )
    return 0;
  v4 = GetProcAddress_AllocateData(v1);
  *v2 = v4 == 0;
  if ( v4 )
  {
    __guard_check_icall_fptr(v4);
    v6 = v4();
    if ( &v8 != &v8 )
      __fastfail(4u);
  }
  else
  {
    v6 = __AllocStdCallThunk_cmn();
  }
```

2.2 The 1st Fix of the Issue

- **AtlThunk_AllocateData is implemented in atlthunk.dll**

```
PVOID __thiscall GetProcAddress_AllocateData(void *this)
{
  PVOID result; // eax@2
  HMODULE v2; // eax@3 MAPDST

  if ( byte_1000D8C0 )
  {
    result = DecodePointer(dword_1000D8C4);
  }
  else
  {
    v2 = LoadLibraryExA("atlthunk.dll", 0, 0x800u);
    if ( v2
      && GetProcAddressSingle(v2, "AtlThunk_AllocateData", &dword_1000D8C4)
      && GetProcAddressSingle(v2, "AtlThunk_InitData", &dword_1000D8BC)
      && GetProcAddressSingle(v2, "AtlThunk_DataToCode", &Ptr)
      && GetProcAddressSingle(v2, "AtlThunk_FreeData", &dword_1000D8B4) )
    {
      _InterlockedOr(&this, 0);
      byte_1000D8C0 = 1;
      result = DecodePointer(dword_1000D8C4);
    }
    else
    {
      result = 0;
    }
  }
  return result;
}
```

2.2 The 1st Fix of the Issue

- atlthunk.dll separate Data and Code

```
.data:10005010 _AtlThunkData   dd offset AtlThunk_0x00(HWND__ *,uint,uint,long), offset _AtlThunkData+10h, 0
.data:10005010                                  ; DATA XREF: AtlThunk_AllocateData()+DF↑o
.data:10005010                                  ; AtlThunk_AllocateData():loc_10004198↑r ...
.data:10005010                 dd offset AtlThunk_0x01(HWND__ *,uint,uint,long), offset _AtlThunkData+1Ch, 0
.data:10005010                 dd offset AtlThunk_0x02(HWND__ *,uint,uint,long), offset _AtlThunkData+28h, 0
.data:10005010                 dd offset AtlThunk_0x03(HWND__ *,uint,uint,long), offset _AtlThunkData+34h, 0
.data:10005010                 dd offset AtlThunk_0x04(HWND__ *,uint,uint,long), offset _AtlThunkData+40h, 0
.data:10005010                 dd offset AtlThunk_0x05(HWND__ *,uint,uint,long), offset _AtlThunkData+4Ch, 0
.data:10005010                 dd offset AtlThunk_0x06(HWND__ *,uint,uint,long), offset _AtlThunkData+58h, 0
.data:10005010                 dd offset AtlThunk_0x07(HWND__ *,uint,uint,long), offset _AtlThunkData+64h, 0
.data:10005010                 dd offset AtlThunk_0x08(HWND__ *,uint,uint,long), offset _AtlThunkData+70h, 0
.data:10005010                 dd offset AtlThunk_0x09(HWND__ *,uint,uint,long), offset _AtlThunkData+7Ch, 0
```

2.2 The 1st Fix of the Issue

- atlthunk.dll separate Data and Code

```
; Attributes: bp-based frame

; __int32 __stdcall AtlThunk_0x00(HWND, unsigned int, unsigned int, __int32)
long __stdcall AtlThunk_0x00(struct HWND__ *, unsigned int, unsigned int, long) proc near

arg_4= dword ptr  0Ch
arg_8= dword ptr  10h
arg_C= dword ptr  14h

mov     edi, edi
push    ebp             ; unsigned int
mov     ebp, esp
push    [ebp+arg_C]     ; unsigned int
mov     edx, [ebp+arg_4]
xor     ecx, ecx
push    [ebp+arg_8]     ; unsigned int
call    AtlThunk_Call(uint,uint,uint,long)
pop     ebp
retn    10h
long __stdcall AtlThunk_0x00(struct HWND__ *, unsigned int, unsigned int, long) endp
```

2.2 The 1st Fix of the Issue

- atlthunk.dll separate Data and Code
  - Data is PAGE_READWRITE
  - Code is always PAGE_EXECUTE
  - No PAGE_EXECUTE_READWRITE page now

2.3 Bypass the 1st Fix

- There's a fallback in the fix for compatibility reason

```
_DWORD *__stdcall AtlThunk_AllocateData()
{
  HANDLE v0; // eax@1
  void *v1; // ecx@1
  _DWORD *v2; // edi@1
  int (*v4)(void); // eax@3 MAPDST
  void *v6; // eax@4
  HANDLE v7; // eax@9
  int v8; // [sp+0h] [bp-10h]@4

  v0 = GetProcessHeap();
  v2 = HeapAlloc(v0, 8u, 8u);
  if ( !v2 )
    return 0;
  v4 = GetProcAddress_AllocateData(v1);
  *v2 = v4 == 0;
  if ( v4 )
  {
    __guard_check_icall_fptr(v4);
    v6 = v4();
    if ( &v8 != &v8 )
      __fastfail(4u);
  }
  else
  {
    v6 = __AllocStdCallThunk_cmn();
  }
```

2.3 Bypass the 1st Fix

- When will GetProcAddress_AllocateData failed?

```
PVOID __thiscall GetProcAddress_AllocateData(void *this)
{
  PVOID result; // eax@2
  HMODULE v2; // eax@3 MAPDST

  if ( byte_1000D8C0 )
  {
    result = DecodePointer(dword_1000D8C4);
  }
  else
  {
    v2 = LoadLibraryExA("atlthunk.dll", 0, 0x800u);
    if ( v2
      && GetProcAddressSingle(v2, "AtlThunk_AllocateData", &dword_1000D8C4)
      && GetProcAddressSingle(v2, "AtlThunk_InitData", &dword_1000D8BC)
      && GetProcAddressSingle(v2, "AtlThunk_DataToCode", &Ptr)
      && GetProcAddressSingle(v2, "AtlThunk_FreeData", &dword_1000D8B4) )
    {
      _InterlockedOr(&this, 0);
      byte_1000D8C0 = 1;
      result = DecodePointer(dword_1000D8C4);
    }
    else
    {
      result = 0;
    }
  }
  return result;
}
```

2.3 Bypass the 1st Fix

- **When will GetProcAddress_AllocateData failed?**
  - **LoadLibraryExA failed**
    - There's no such library atlthunk.dll
    - Happens in system without the fix
  - **GetProcAddress failed**
    - Unlikely to happen

2.3 Bypass the 1st Fix

- **LoadLibrary has a fastpath for reload**

LoadLibrary(“atlthunk.dll”)

LoadLibrary(“atlthunk.dll”)

C:\Windows\System32\atlthunk.dll

2.3 Bypass the 1st Fix

- **LoadLibrary has a fastpath for reload**

LoadLibrary(“C:\Users\user\Downloads\atlthunk.dll”)

LoadLibrary(“atlthunk.dll”)

C:\Users\user\Downloads\atlthunk.dll

2.3 Bypass the 1st Fix

- Microsoft Edge used to have an auto-download feature
  - Visit a page that contain <iframe src="path/to/atlthunk.dll"/>
  - atlthunk.dll will be automatically downloaded to %userprofile%\Downloads

2.3 Bypass the 1st Fix

- **Exploit Plan**
  - Trigger auto-download to deliver a fake atlthunk.dll
  - Call LoadLibrary to load the fake atlthunk.dll
  - Insert flash to allocate ATL Thunk Pool
  - Read the address of ATL Thunk Pool
  - Write shellcode to ATL Thunk Pool
  - Execute the shellcode

2.4 The Final Fix of the Issue

- **Microsoft Edge will ask before auto-download**

2.4 The Final Fix of the Issue

- **ACG will prevent allocating PAGE_EXECUTE_READWRITE page**

ACG enables two kernel-enforced W^X policies

- ✓ Code is immutable
- ✓ Data cannot become code

The following will fail with ERROR_DYNAMIC_CODE_BLOCKED

```
VirtualProtect(codePage, …, PAGE_EXECUTE_READWRITE)
VirtualProtect(codePage, …, PAGE_READWRITE)
VirtualAlloc(…, PAGE_EXECUTE*)
VirtualProtect(dataPage, …, PAGE_EXECUTE*)
MapViewOfFile(hPagefileSection, FILE_MAP_EXECUTE, …)
WriteProcessMemory(codePage, …)
…
```

03

Exploit Chakra JIT Engine

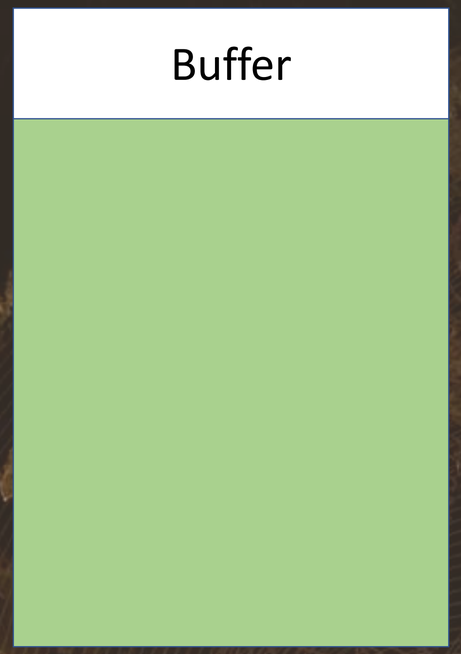
- **How Chakra JIT Engine use memory**

```
Encoder::Encode
    CodeGenWorkItem::RecordNativeCodeSize
        EmitBufferManager::AllocateBuffer
            EmitBufferManager::NewAllocation
                CustomHeap::Heap::Alloc
CodeGenWorkItem::RecordNativeCode
    EmitBufferManager::CommitBuffer
        CustomHeap::Heap::ProtectAllocationWithExecuteReadWrite
        memcpy_s
        CustomHeap::Heap::ProtectAllocationWithExecuteReadOnly
```

3.1 The Original Issue

- **How Chakra JIT Engine use memory**

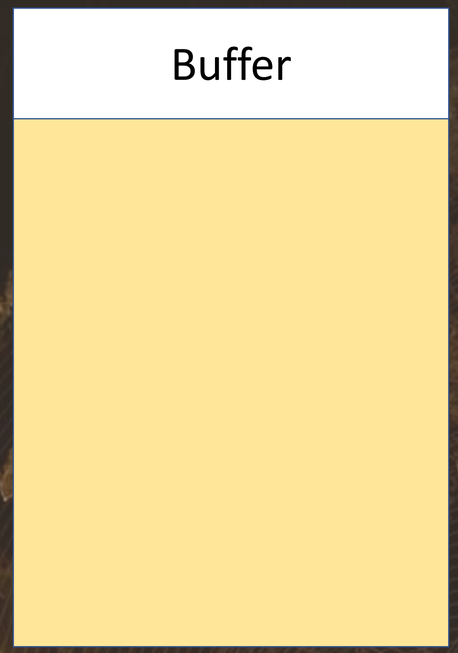
Encoder::Encode
  CodeGenWorkItem::RecordNativeCodeSize
    EmitBufferManager::AllocateBuffer
      EmitBufferManager::NewAllocation
        CustomHeap::Heap::Alloc
CodeGenWorkItem::RecordNativeCode
  EmitBufferManager::CommitBuffer
    CustomHeap::Heap::ProtectAllocationWithExecuteReadWrite
    memcpy_s
    CustomHeap::Heap::ProtectAllocationWithExecuteReadOnly



3.1 The Original Issue

- **How Chakra JIT Engine use memory**

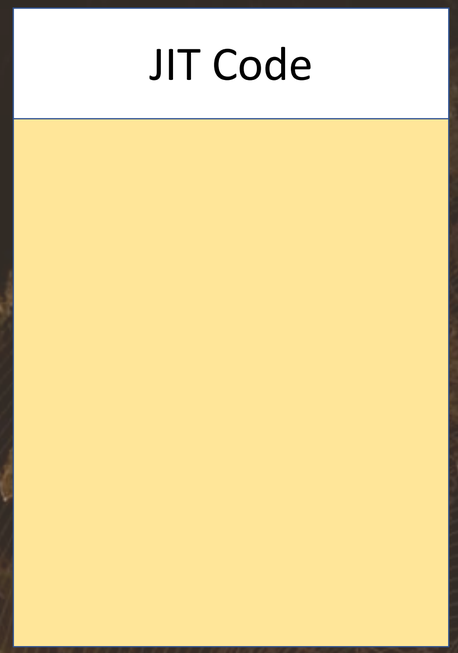
Encoder::Encode
  CodeGenWorkItem::RecordNativeCodeSize
    EmitBufferManager::AllocateBuffer
      EmitBufferManager::NewAllocation
        CustomHeap::Heap::Alloc
CodeGenWorkItem::RecordNativeCode
  EmitBufferManager::CommitBuffer
    CustomHeap::Heap::ProtectAllocationWithExecuteReadWrite
    memcpy_s
    CustomHeap::Heap::ProtectAllocationWithExecuteReadOnly



3.1 The Original Issue

- **How Chakra JIT Engine use memory**

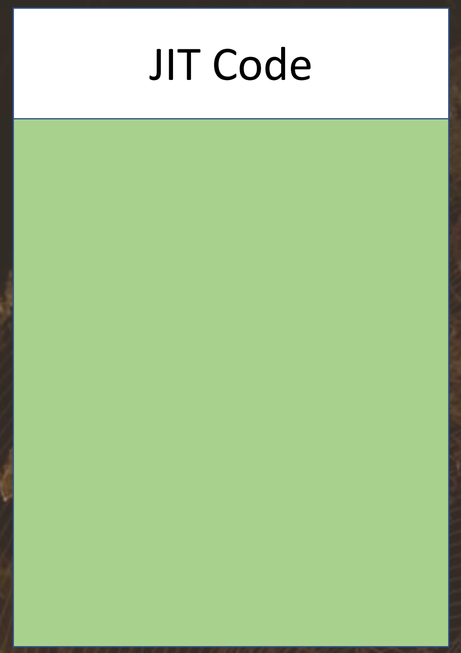
Encoder::Encode
  CodeGenWorkItem::RecordNativeCodeSize
    EmitBufferManager::AllocateBuffer
      EmitBufferManager::NewAllocation
        CustomHeap::Heap::Alloc
CodeGenWorkItem::RecordNativeCode
  EmitBufferManager::CommitBuffer
    CustomHeap::Heap::ProtectAllocationWithExecuteReadWrite
    memcpy_s
    CustomHeap::Heap::ProtectAllocationWithExecuteReadOnly



3.1 The Original Issue

- **How Chakra JIT Engine use memory**

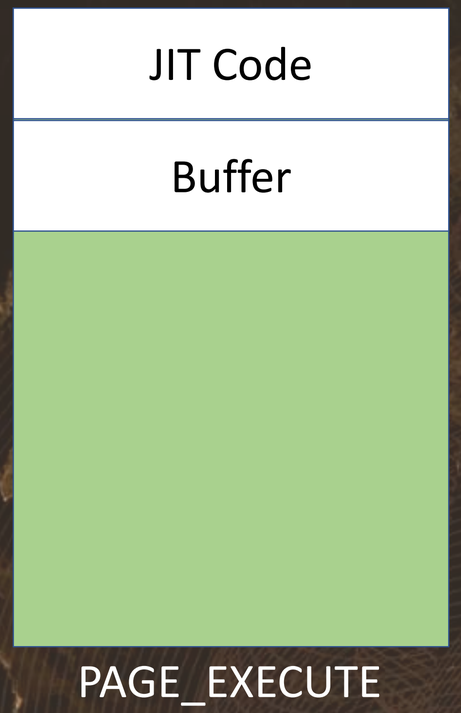
Encoder::Encode
  CodeGenWorkItem::RecordNativeCodeSize
    EmitBufferManager::AllocateBuffer
      EmitBufferManager::NewAllocation
        CustomHeap::Heap::Alloc
CodeGenWorkItem::RecordNativeCode
  EmitBufferManager::CommitBuffer
    CustomHeap::Heap::ProtectAllocationWithExecuteReadWrite
    memcpy_s
    CustomHeap::Heap::ProtectAllocationWithExecuteReadOnly



3.1 The Original Issue

- **How Chakra JIT Engine use memory**

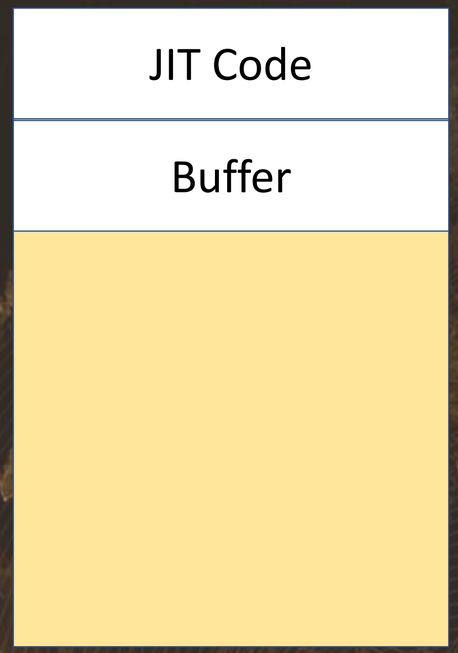
Encoder::Encode
  CodeGenWorkItem::RecordNativeCodeSize
    EmitBufferManager::AllocateBuffer
      EmitBufferManager::NewAllocation
        CustomHeap::Heap::Alloc
CodeGenWorkItem::RecordNativeCode
  EmitBufferManager::CommitBuffer
    CustomHeap::Heap::ProtectAllocationWithExecuteReadWrite
    memcpy_s
    CustomHeap::Heap::ProtectAllocationWithExecuteReadOnly



3.1 The Original Issue

- **How Chakra JIT Engine use memory**

```
Encoder::Encode
    CodeGenWorkItem::RecordNativeCodeSize
        EmitBufferManager::AllocateBuffer
            EmitBufferManager::NewAllocation
                CustomHeap::Heap::Alloc
CodeGenWorkItem::RecordNativeCode
    EmitBufferManager::CommitBuffer
        CustomHeap::Heap::ProtectAllocationWithExecuteReadWrite
        memcpy_s
        CustomHeap::Heap::ProtectAllocationWithExecuteReadOnly
```

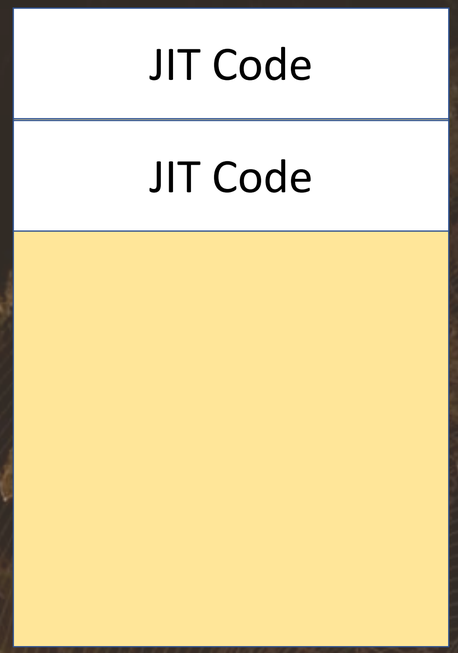


PAGE_EXECUTE_READWRITE

3.1 The Original Issue

- **How Chakra JIT Engine use memory**

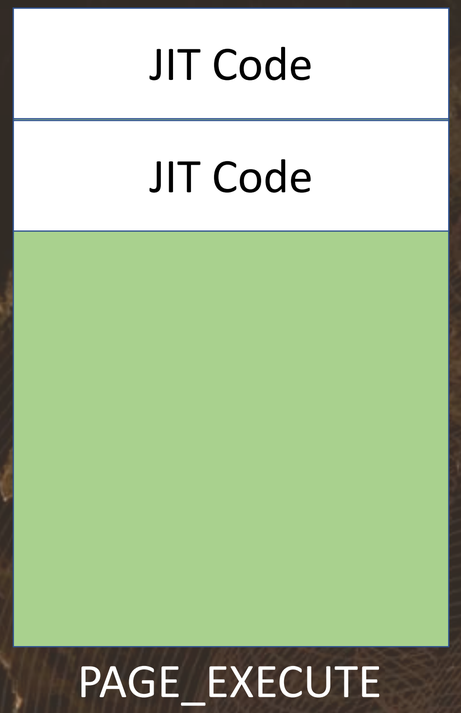
Encoder::Encode
  CodeGenWorkItem::RecordNativeCodeSize
    EmitBufferManager::AllocateBuffer
      EmitBufferManager::NewAllocation
        CustomHeap::Heap::Alloc
CodeGenWorkItem::RecordNativeCode
  EmitBufferManager::CommitBuffer
    CustomHeap::Heap::ProtectAllocationWithExecuteReadWrite
    memcpy_s
    CustomHeap::Heap::ProtectAllocationWithExecuteReadOnly



3.1 The Original Issue

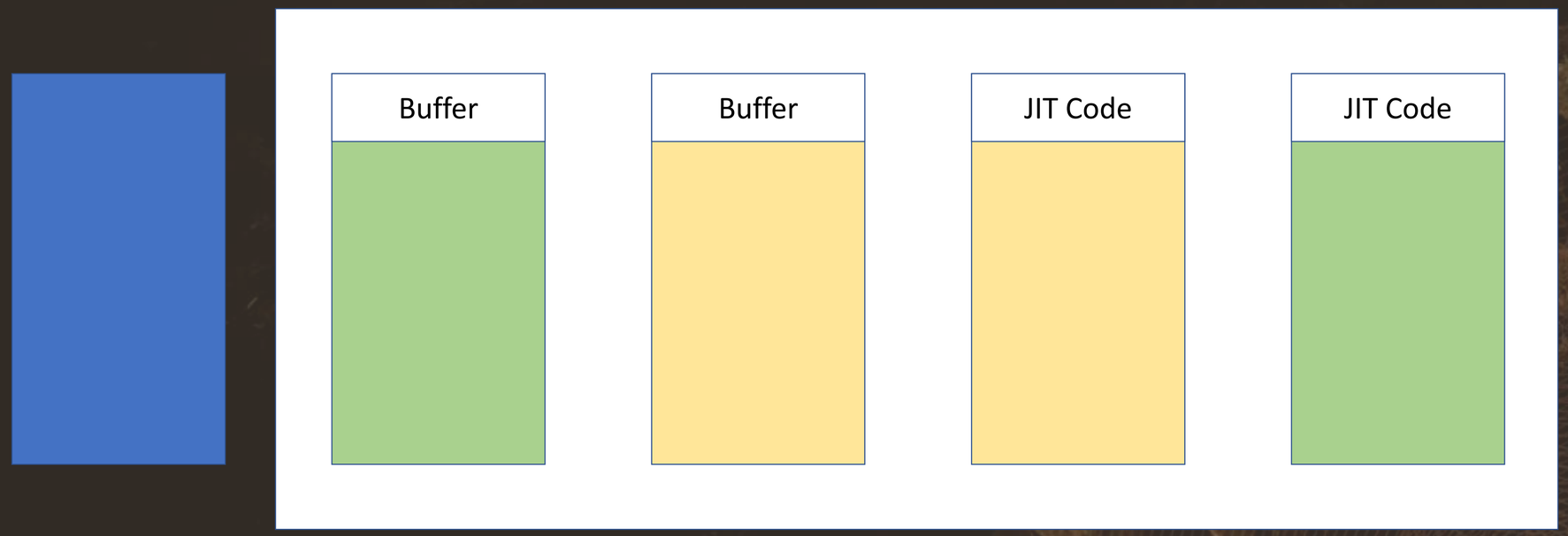
- **How Chakra JIT Engine use memory**

Encoder::Encode
  CodeGenWorkItem::RecordNativeCodeSize
    EmitBufferManager::AllocateBuffer
      EmitBufferManager::NewAllocation
        CustomHeap::Heap::Alloc
CodeGenWorkItem::RecordNativeCode
  EmitBufferManager::CommitBuffer
    CustomHeap::Heap::ProtectAllocationWithExecuteReadWrite
    memcpy_s
    CustomHeap::Heap::ProtectAllocationWithExecuteReadOnly



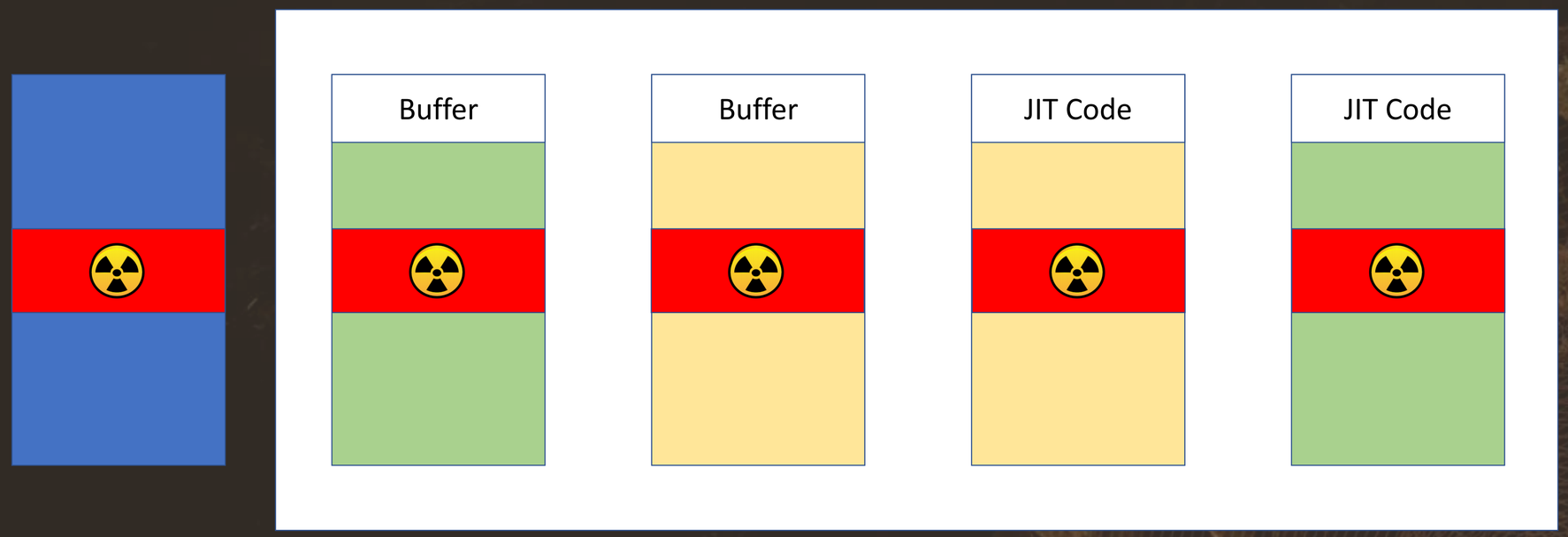
3.1 The Original Issue

- How Chakra JIT Engine use memory



3.1 The Original Issue

- Protect is an attribute of the whole page

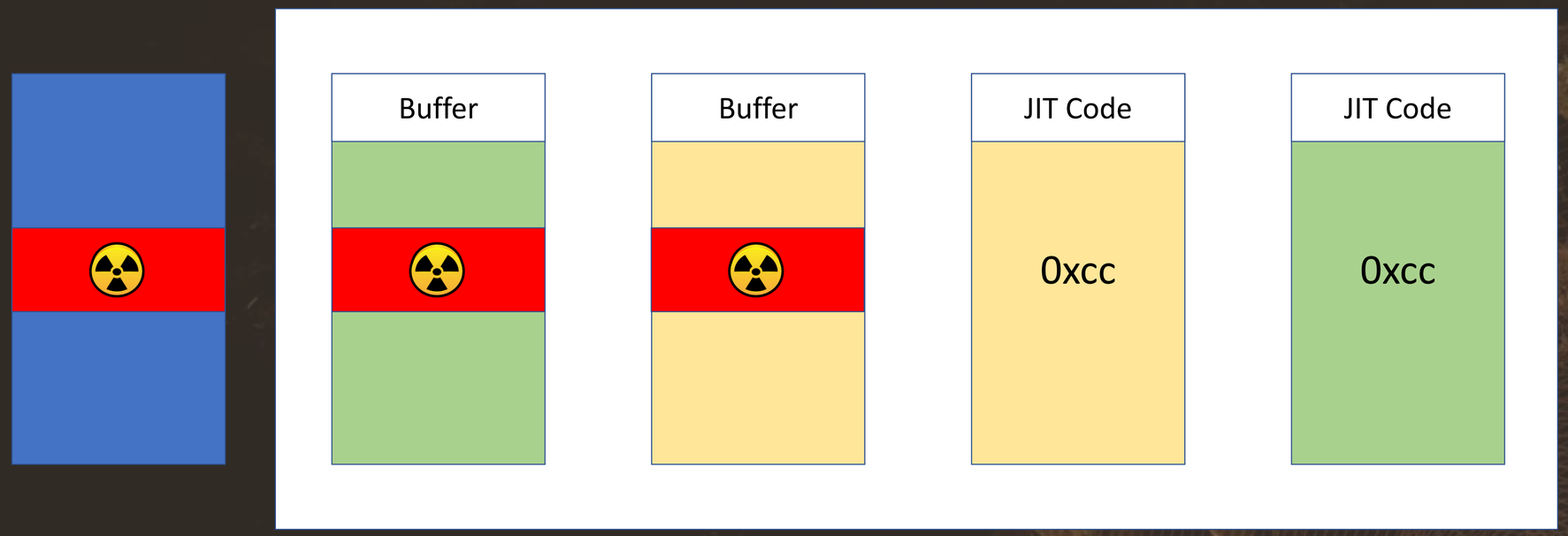


3.1 The Original Issue

- **Exploit Plan**
  - **Find the address of the next page used by JIT Engine**
  - **Write shellcode to the middle of the page**
  - **Trigger JIT compilation**
  - **Wait the JIT compilation to complete**
  - **Execute the shellcode**

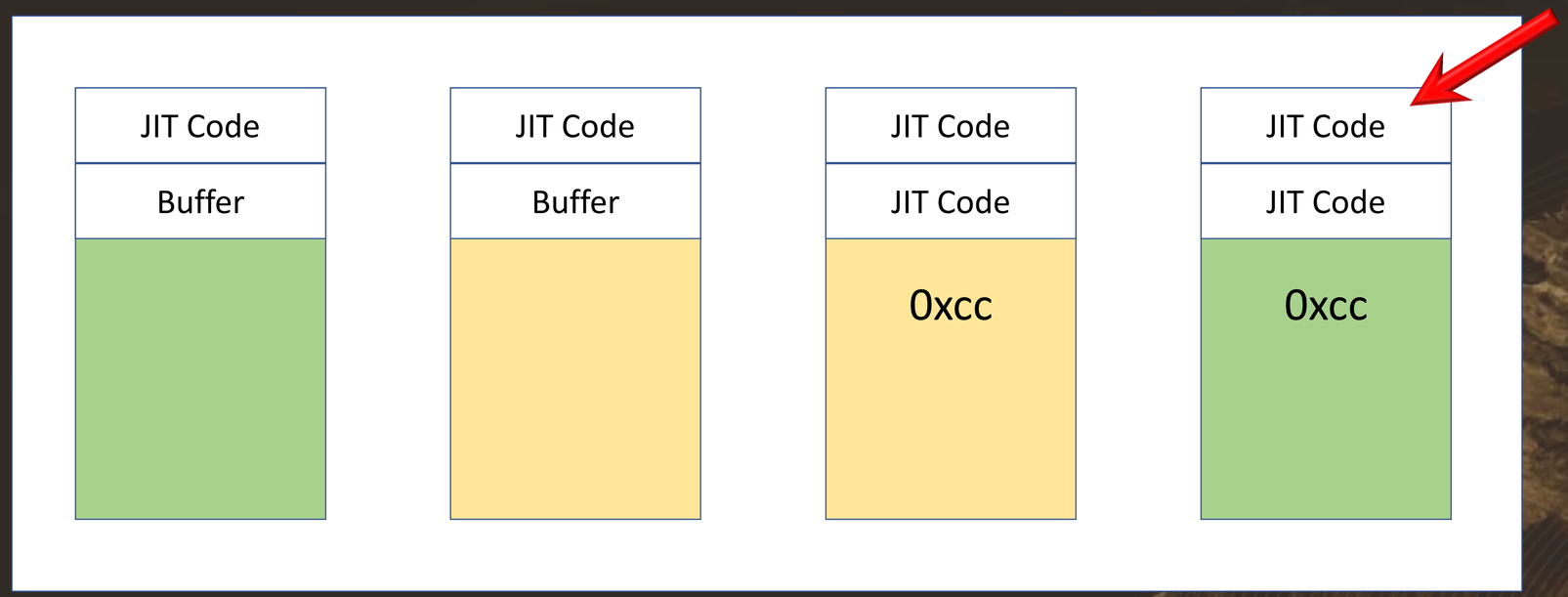
3.2 The 1st Fix of the Issue

- Chakra JIT Engine will fill the remain part with 0xcc after copy



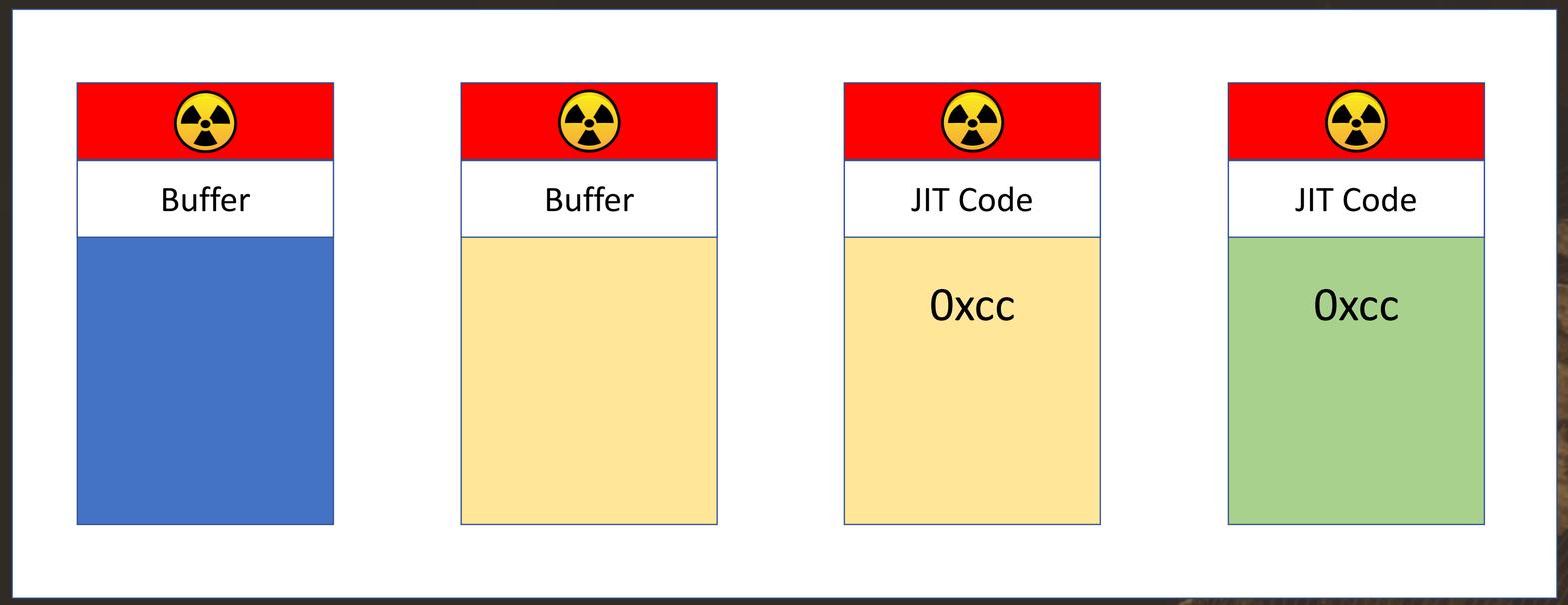
3.3 Bypass the 1st Fix

- Chakra JIT Engine won't fill the precedent memory



3.3 Bypass the 1st Fix

- Shellcode in the used part of a page will survive



3.3 Bypass the 1st Fix

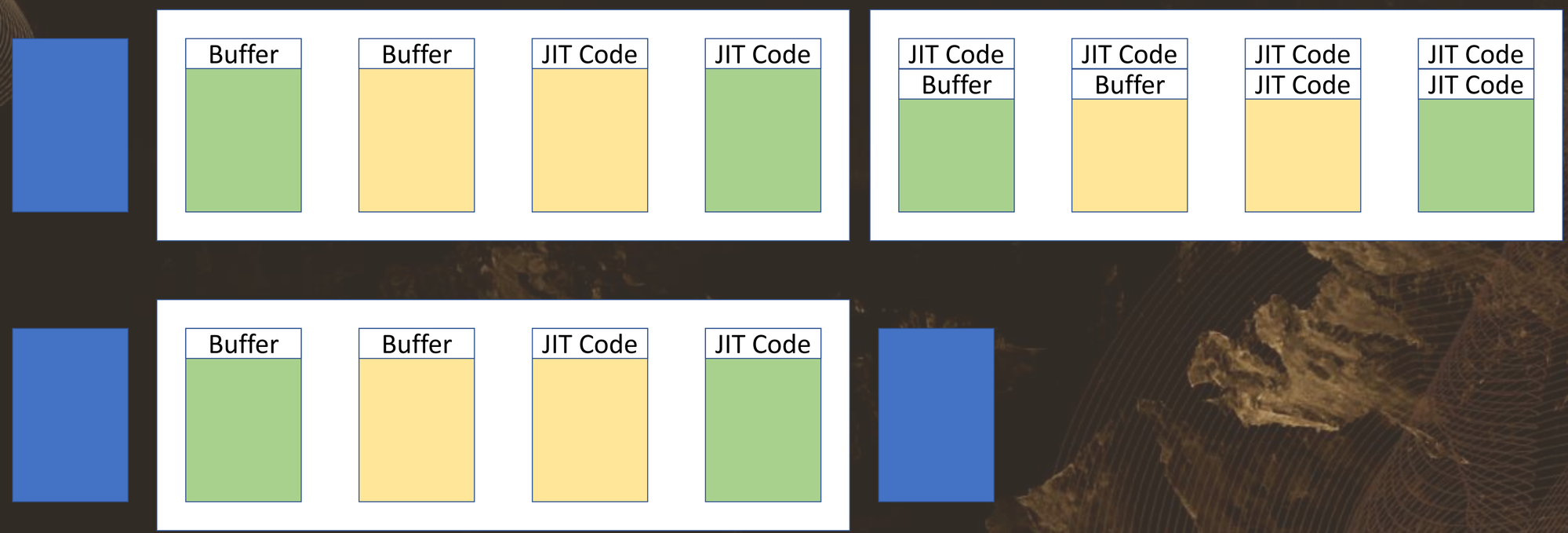
- Exploit Plan
  - Insert a fake page to the proper bucket of CustomHeap
  - Write shellcode to the beginning of the page
  - Trigger JIT compilation
  - Wait the JIT compilation to complete
  - Execute the shellcode

3.4 The 2nd Fix of the Issue

- Chakra enforce CFG in JIT Engine
  - Page is allocated with PAGE_TARGETS_NO_UPDATE
  - JIT function entry is enable by calling SetProcessValidCallTargets

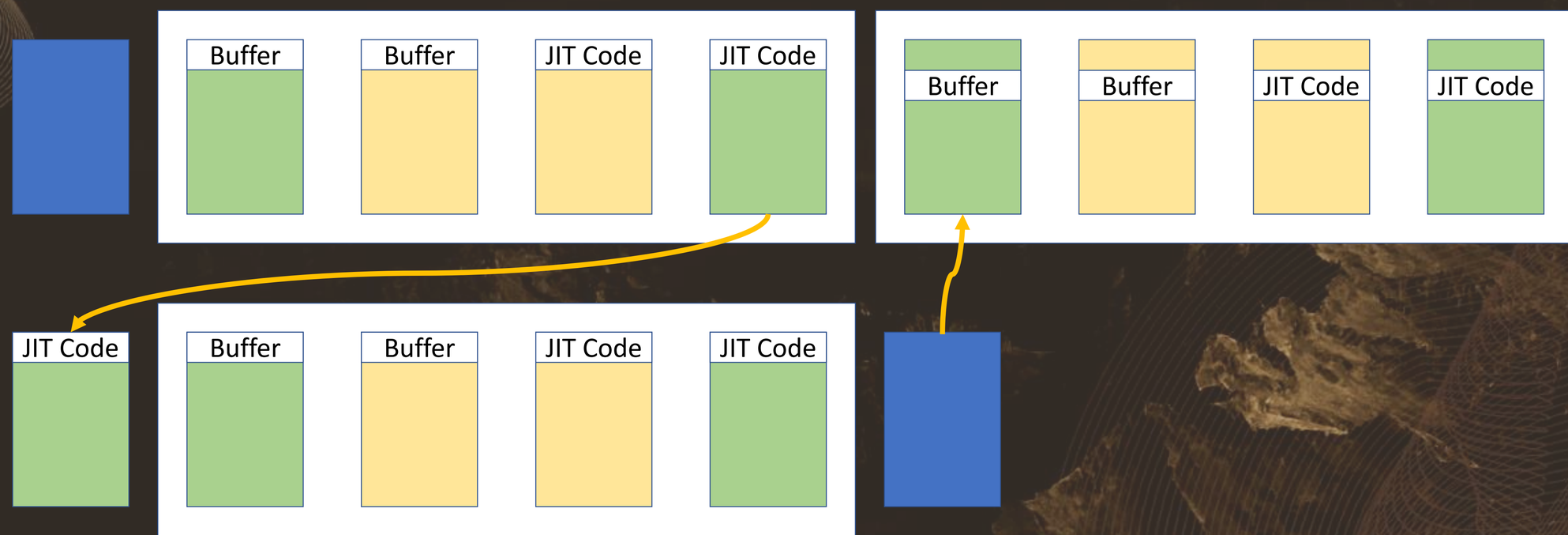
3.5 Bypass the 2nd Fix

- Launch another instance of JIT Engine



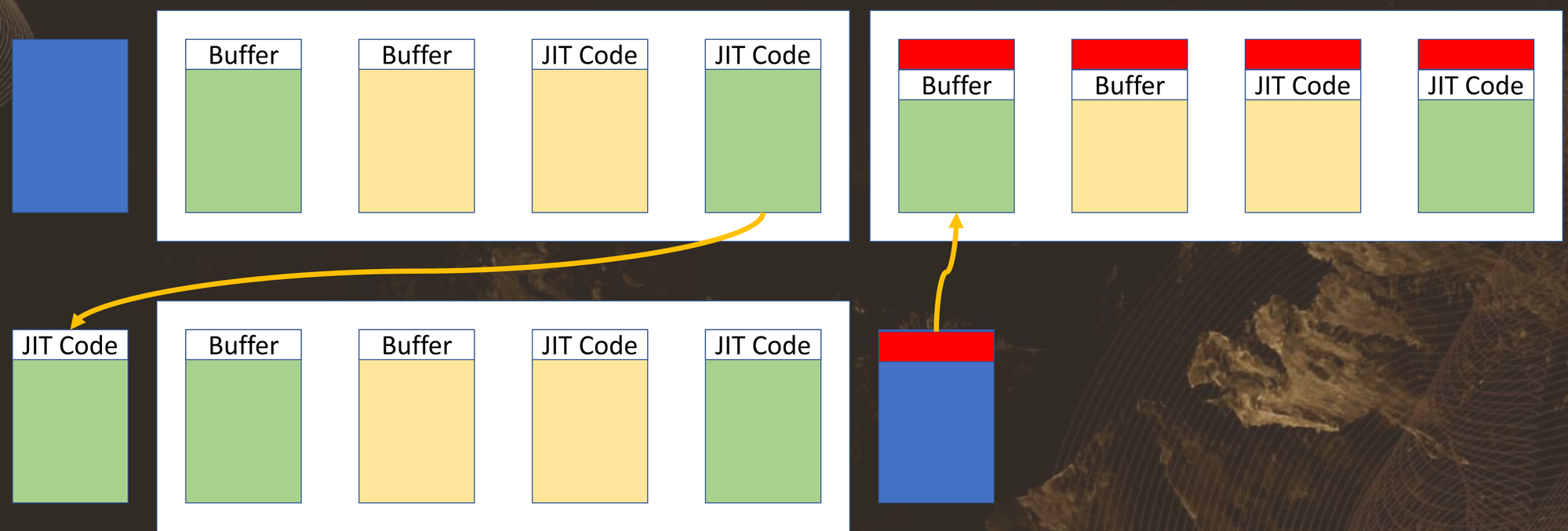
3.5 Bypass the 2nd Fix

- Let the 2 instances use the same page



3.5 Bypass the 2nd Fix

- **Modify the generated JIT Code**



3.5 Bypass the 2nd Fix

- **Exploit Plan**
  - Trigger JIT compilation for function FuncA
  - Read the JIT code for FuncA's address AddrA
  - Launch the second JIT Engine
  - Modify the CustomHeap of the second JIT Engine to use AddrA
  - Trigger JIT compilation in the second JIT Engine
  - Release the second JIT Engine
  - Write shellcode to AddrA
  - Trigger JIT compilation in the first JIT Engine
  - Call FuncA to execute shellcode

3.6 The Final Fix of the Issue

- Out-of-process (OOP) JIT
  - The whole work of JIT compilation is move to a dedicated process
  - The renderer process do not manage memory used in JIT


Q&A