ZDI年度五大漏洞第四弹——让指针指向你想要的对象

Agostop / 2018-12-27 10:28:00 / 浏览数 2380 技术文章 翻译文章 顶(0) 踩(0)

原文链接：https://www.zerodayinitiative.com/blog/2018/12/20/really-check-errors-pointing-to-the-object-of-your-desire

这是ZDI评选的2018年五大漏洞的第四个案例，这些评选出来的bug具有一些独特的元素，使得其与今年发布的大约1400条其他报告不同。今天我们来看另一个Pwn2Own_

在2018年Pwn2Own大赛上，MWR实验室成功地攻破了苹果Safari浏览器。漏洞利用首先利用了SVG对象处理中的堆溢出漏洞，然后利用Dock中的未初始化指针达到沙箱逃

在mac操作系统中，可以通过名为"com.apple.dock.server"的Mach服务来访问Dock，这些函数大量使用HIServices框架提供的序列化功能。该漏洞是在DSSetDesktopFo

我们先来看看DSSetDesktopForDisplayAndSpace函数是什么样子的:

```
push    rbp
mov     rbp, rsp
push    r15
push    r14
push    r13
push    r12
push    rbx
sub     rsp, 418h
mov     [rbp+var_24C], r8d
mov     r12, rdx
mov     rax, cs:___stack_chk_guard_ptr
mov     rax, [rax]
mov     [rbp+var_30], rax
mov     [rbp+var_260], rsi
mov     [rbp+var_231], 1
mov     [rbp+var_274], ecx
mov     esi, ecx
lea     r14, [rbp+var_2E0]
mov     rdi, r12
mov     [rbp+var_298], rsi
mov     rdx, r14
call    _UnserializeCFType
mov     ebx, eax
mov     rdi, [r14]      ; id
call    _objc_autorelease
mov     rdi, rax        ; id
call    cs:__imp__objc_retain_ptr
```

这只是函数的开始部分，但是我们可以看到在调用 _UnserializeCFType 之前，堆栈上已经初始化了一些变量。这个函数存在于HIServices中，我们也来看看它:

```
__int64 __fastcall UnserializeCFType(void *a1, __int64 a2, void *a3) {
  return AXUnserializeCFType(0LL, a2, a1, a2, a3);
}
```

这看起来并不是很有趣，但我们还是来深入探讨一下，这是AXUnserializeCFType函数：

```
__int64 __fastcall AXUnserializeCFType(__int64 a1, __int64 a2, _DWORD *a3, unsigned __int64 a4, _QWORD *a5) {
  __int64 result; // rax
  int v6; // esi
  unsigned __int64 v7; // rax
  __int64 (__fastcall *v8)(__int64, _DWORD **, unsigned __int64, _QWORD *, bool); // r9
  _DWORD *v9; // [rsp+8h] [rbp-8h]

  v9 = a3;
  result = 0xFFFF9D8FLL;
  if ( a4 >= 8 ) {
    *a5 = 0LL;
    v6 = *a3;
    if ( *a3 == 'owen' || v6 == 'aela' ) {
      v9 = a3 + 1;
      v7 = (unsigned int)a3[1];
      if ( v7 <= 0xF )
        v8 = sUnserializeFunctions[v7];
      else
        v8 = bogusUnserialize;
      result = v8(a1, &v9, a4 - 4, a5, v6 == 'owen');
    }
  }
  return result;
}
```


该函数首先检查了第四个参数是否大于等于8，如果是，它将初始化第五个参数，使其指向NULL，然后尝试基于数据的类型进行反序列化操作。如果反序列化过程没有出现

你发现什么了吗？我们换种方式来看看DSSetDesktopForDisplayAndSpace函数的开头，可能看起来能更加明显：

```
v109 = 1;
v96 = a4;
v6 = UnserializeCFType(a3, a4, &v87);
v7 = objc_autorelease(*(id *)&v87);
v8 = _objc_retain(v7);
```

这和我们刚开始看的汇编代码是一样的，包括所有变量的初始化。现在你能看到问题了吗？作为UnserializeCFType函数参数的栈上变量从未被初始化，只有当UnserializeC

MWR团队通过使用了一个包含push
rbx指令的函数利用了这一点，该指令可以和未初始化的栈变量的偏移量对齐，当该指令执行时，rbx寄存器会指向Mach消息中的40个字节，这非常适合我们用来控制obj

总共有8个函数包括了易受攻击的代码逻辑，它们也在未初始化第4个参数或未检查返回值的情况下调用了_UnserializeCFType函数。不过也有几个函数中使用了正确的代码

在2016年，lokihardt公布了ZDI-16-282作为他的攻击微软Edge浏览器利用链的一部分，其中利用了在处理'Array.concat'函数时的一个未初始化堆栈变量。2017年，360安
Workstation中从客户机逃逸到宿主机操作系统漏洞的一部分。有关该漏洞的详细信息，可以查看Abdul-Aziz Hariri的博客。

希望你对这个在Pwn2Own中常出现的漏洞类型感兴趣，由未初始化数据造成的漏洞是我最喜欢的漏洞种类之一，看到其他人利用它们来实现代码执行是十分有趣的。

你可以关注我的Twitter@WanderingGlitch,或者关注我们的团队以了解最新的漏洞利用技术和安全补丁。请继续关注将于明天发布的最后一篇年度五大漏洞相关博客。