Windows 10中的DHCP：关键漏洞CVE-2019-0726详细分析

s小胖不吃饭 / 2019-05-26 07:43:00 / 浏览数 4605 安全技术 漏洞分析 顶(0) 踩(0)

当Windows的1月更新发布时，公众均对DHCP客户端中的CVE-2019-0547漏洞的消息所震惊。极高的CVSS评分伴随着微软没有立即发布可利用性指数评估这一事实使得用

MaxPatrol等解决方案可以识别网络上的哪些计算机容易受到某些攻击。其他解决方案若要检测此类攻击是否能起作用，需要描述识产品中的漏洞规则并检测这些产品的攻

■■■■■■■■■■■■■■■■■■■■■■■■

因此，要使用针对DHCP中新发现的漏洞的攻击检测规则来更新我们的产品，我们需要深入了解漏洞所有细节。
对于二进制漏洞，我们通常可以通过使用patch-diff来解决位于其根目录的错误，patch-diff比较并识别由特定补丁或更新对应用程序、库或操作系统内核的二进制代
然而第一步永远是进行初步尝试。

漏洞初探

我们首先需要测试搜索引擎，查询当前已知的有关此漏洞的所有内容。 其中大部分是从MSRC网站上发表的第一手信息。
这种情况是Microsoft漏洞处于内部处理时的典型情况。

从互联网的搜索资料中，我们发现我们正在处理在Windows
10版本1803上运行的客户端和服务器系统中包含内存损坏漏洞，并且当攻击者向DHCP客户端发送特制响应时它会显示。 几天后，该页面出现了指数评级：

Exploitability Assessment

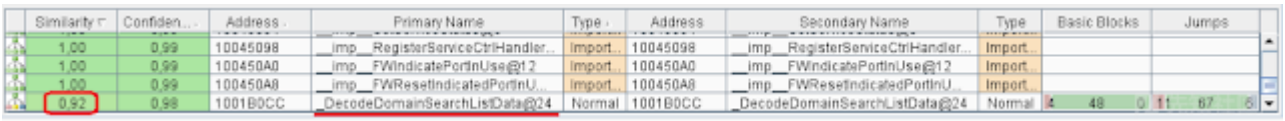
The following table provides an exploitability assessment for this vulnerability at the time of original publication.

Publicly Disclosed	Exploited	Latest Software Release	Older Software Release	Denial of Service
No	No	4 - Not affected	2 - Exploitation Less Likely	Not Applicable

我们可以看到，MSRC的评级为2。这意味着错误很可能是不可利用的，或者利用它是十分困难并且需要花费许多精力。不可否认，微软很少出现如此低分的习惯。因此，我

在同一站点上，我们下载以.msu存档提供的补丁，将其解压缩并查找最有可能与客户端DHCP响应处理相关的文件。现在提供的更新不是作为修复特定错误的单独包，而是

在过多的文件中，我们的搜索出现了几个与过滤器匹配的库，我们将它们与未修补系统上的版本进行比较。
dhcpcore.dll库看起来我们所需要的。同时BinDiff显示出最小的变化：



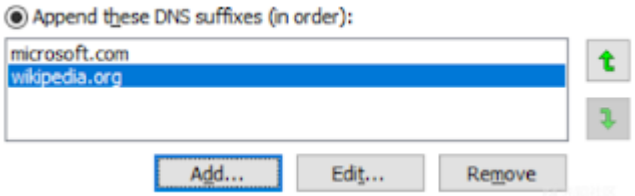
实际上，此处只对一个函数DodedDomainSearchListData进行了更改。 如果用户熟悉DHCP协议及其很少使用的函数，则已经知道该函数处理的列表。

DHCP及其选项

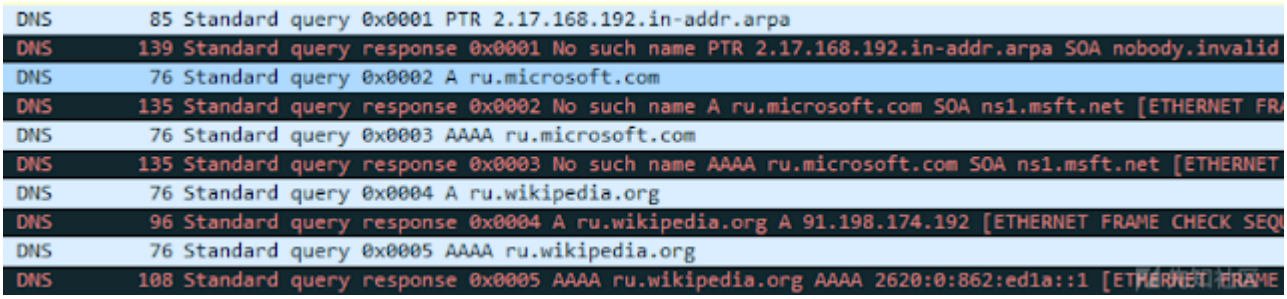
DHCP（RFC 2131|wiki）是一种可扩展的协议，其可扩展性通过options字段实现。
每个选项由唯一标记（编号，标识符），选项中包含了数据大小以及数据本身描述。 这是典型的网络协议，并且在协议中"插入"域搜索选项，其在RFC 3397中进行描述。它允许DHCP服务器在客户端上设置标准域名结尾。 这些将用作以这种方式设置的连接的DNS后缀。

例如，假设在我们的客户端上我们设置了以下名称结尾：

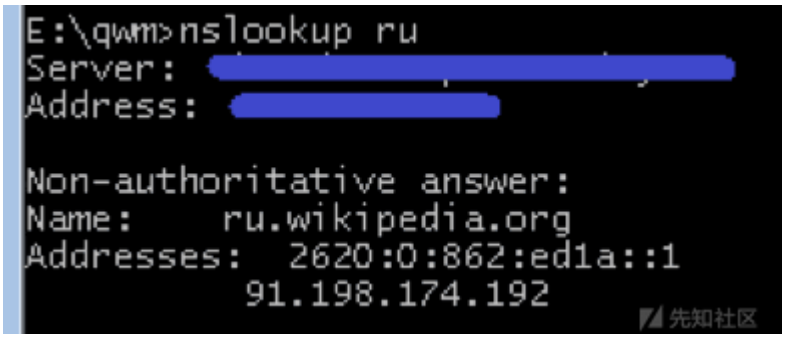
```
.microsoft.com
.wikipedia.org
```



然后，在按域名确定地址的尝试中，这些结尾将逐个插入DNS■■，直到找到匹配为止。
例如，如果用户在浏览器地址栏中键入ru，则首先为ru.microsoft.com形成DNS请求，然后为ru.wikipedia.org形成DNS请求：



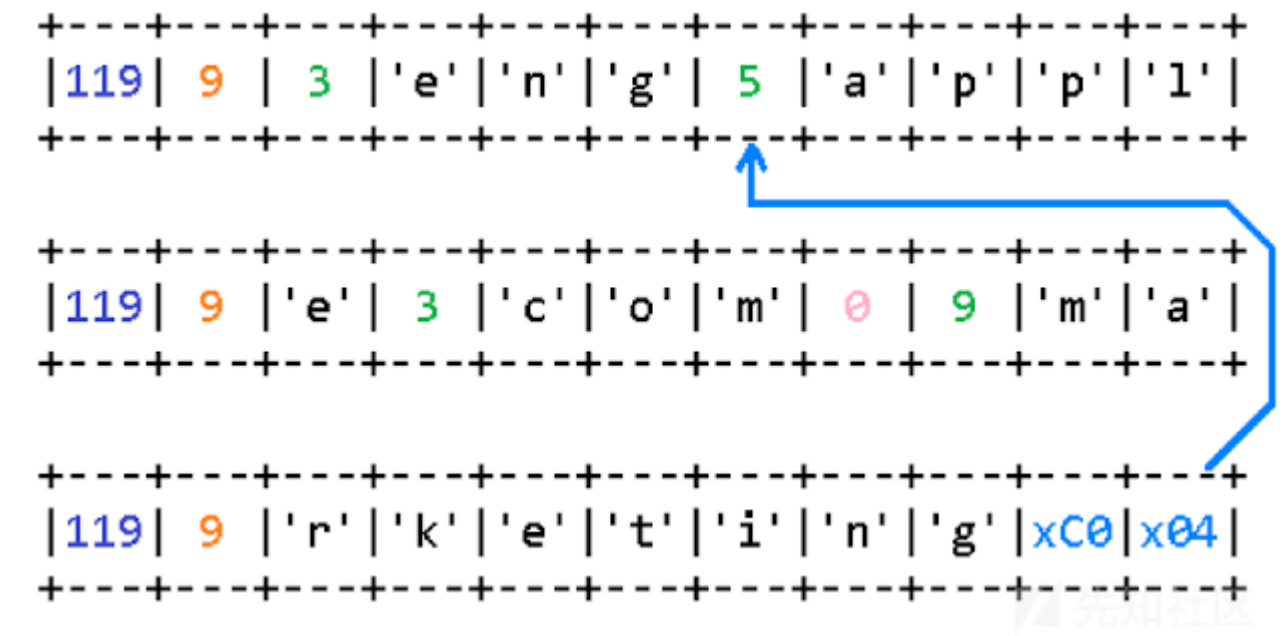
实际上，现代浏览器是非常智能的，因此它们通过重定向到搜索引擎来对类似于FQDN的名称作出反应。 因此，我们稍后将提供较少实用程序的输出：



读者可能会认为这是漏洞的本质。
就其本身而言，当网络上的任何设备都可以识别时，使用DHCP服务器更改DNS■■的能力对使用DHCP请求任何网络参数的客户端构成威胁。 但那还不是全部。
从RFC中可以明显看出，这被认为是非常合法且记录在案的行为。 实际上，DHCP服务器是一个可信组件，能够影响连接到它的设备。

域搜索选项

域搜索选项号为0x77。 与所有选项一样，它带有选项号的单字节标记编码。 和大多数选项一样，标签后面跟着大小的单字节大小的数据。
DHCP消息可以包含多个选项副本。 在这种情况下，来自所有这些部分的数据以与消息中相同的顺序连接。



在取自RFC 3397的示例中，数据被分成三个部分，每个部分包含9个字节。 从图中可以看出，完整域名中的子域名使用单字节名称长度编码，后跟名称本身。
完整的域名代码以空字节结尾（子域名的空值大小）。

此外，该选项使用最简单的数据压缩方法：重新分析点。 该字段可能包含0xc0，而不是域名大小。
然后，下一个字节将建立相对于用于搜索域名结尾的选项的数据开头的偏移量。
因此，在我们的示例中，我们有一个包含两个域后缀的编码列表：

```
.eng.apple.com
.marketing.apple.com
```

DecodeDomainSearchListData函数

数字为0x77的DHCP选项允许服务器在客户端上设置DNS后缀。但不适用于装有Windows操作系统的计算机。传统上，Microsoft系统忽略了此选项，因此在必要时，使用
10版本1803的新版本引入域搜索选项处理时，情况发生了变化。如下所示，dhcpcore.dll中的函数名称已更改，它是包含错误的添加的处理程序本身。

现在让我们开始工作吧。梳理一下代码，这就是我们找到的。正如人们可能猜到的那样，DecodeDomainSearchListData过程解码来自服务器的消息的域搜索选项中的数

```
eng.apple.com,marketing.apple.com
```

DecodeDomainSearchListData是通过UpdateDomainSearchOption函数调用的，此函数将返回的列表写入注册表项的"DhcpDomainSearchList"参数中：

```
HKLM\SYSTEM\CurrentControlSet\Services\Tcpip\Parameters\Interfaces\{INTERFACE_GUID}\
```

它存储特定网络接口的主要参数。

```
11  is_ok = &dhcp_options->domain_search_list_is_ok;
12  DecodeDomainSearchListData(
13    dhcp_datagram->domain_search_list_option,
14    dhcp_datagram->domain_search_list_option_size,
15    &dhcp_options->domain_search_list,
16    &dhcp_options->domain_search_list_size,
17    &dhcp_options->domain_search_list_count,
18    &dhcp_options->domain_search_list_is_ok);
19  dhcp_datagram_->parsing_is_ok = *is_ok;
20  if ( !*is_ok )
21    goto LABEL_15;
22  if...
23  reg_key = 0;
24  res = RegOpenKeyExW(dhcp_options_->domain_search_list_reg_key, 0, 0, 0xFu, &reg_key);
25  if ( res
26    || (res = RegSetValueExA(
27                reg_key,
28                "DhcpDomainSearchList",
29                0,
30                1u,
31                (const BYTE *)dhcp_options_->domain_search_list,
32                dhcp_options_->domain_search_list_size),
33        RegCloseKey(reg_key),
34        res) )
```

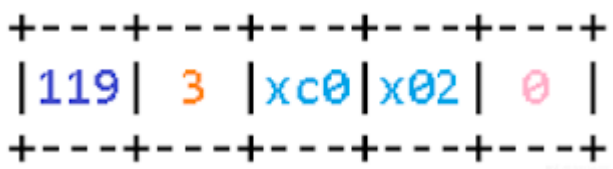
DecodeDomainSearchListData函数进行两次传递。 在第一次传递时，它执行除输入缓冲区之外的所有操作。
所以第一遍用于计算保存返回数据所需的内存大小。 在第二遍中，为该数据分配存储器并填充分配的存储器。 该函数不是太大 - 大约250条指令 -
它的主要工作是处理输入流中字符的三种可能变体中的每一种：1）0x00,2）0xc0或3）所有值。
与DHCP相关的错误的修复归结为在第二遍开始时添加对结果缓冲区大小的检查。 如果大小为零，则不为缓冲区分配内存，并且该函数完成执行并返回错误：

```
37  while ( 1 )
38  {
39    if ( !*dest_is_data_ok )
40      goto LABEL_49;
41    pass_ = ++pass;
42    if ( pass != DECODEDOMAINPASSES_DO_COPY )
43      goto LABEL_8;
44    if ( *data_ptr )
45    {
46      HeapFree(DhcpGlobalHeap, 0, *data_ptr);
47      size_ptr_ = size_ptr;
48      *data_ptr = 0;
49    }
50    if ( !*size_ptr_ )                          // has been added
51      break;
52    buf = HeapAlloc(DhcpGlobalHeap, HEAP_ZERO_MEMORY, *size_ptr_);
53    size_ptr_ = size_ptr;
54    buf_ = buf;
55    source_size_ = source_size__;
56LABEL_8:
57    *size_ptr_ = 0;
58    count_of_0xc0_domains = 0;
```


因此，只有当目标缓冲区的大小为零时，漏洞才会显示出来。 并且在一开始该函数检查其输入，其大小不能小于两个字节。
因此，利用需要找到以输出缓冲区的大小等于零的方式形成的非空域后缀选项。

攻击过程

我们首先想到的是使用重解析来确保非空输入数据生成一个空的输出字符串：



设置为使用具有此类内容的选项进行响应的服务器确实会导致对未更新的客户端的访问冲突。当函数解析部分完整域名时，它会将该部分复制到目标缓冲区并附加句点。
在RFC的此示例中，以下数据将按以下顺序复制到缓冲区：

```
1). eng.
2). eng.apple.
3). eng.apple.com.
```

然后，当在输入数据中遇到零域大小时，该函数将目标缓冲区中的前一个字符从句点更改为逗号：

```
4). eng.apple.com,
```

进行解析操作：

```
5). eng.apple.com,marketing.
6). eng.apple.com,marketing.apple.
7). eng.apple.com,marketing.apple.com
8). eng.apple.com,marketing.apple.com
```

当输入数据结束时，剩下的就是用空字符替换最后一个逗号，这里有一个准备好写入注册表的字符串：

```
9). eng.apple.com,marketing.apple.com
```

当攻击者发送如上所述形成的缓冲区时会发生什么？从示例中我们可以看到它包含的列表由单个元素组成 -
一个空字符串。在第一次传递时，该函数计算输出数据大小。由于数据不包含任何非零域名，因此大小为零。

在第二遍中，为数据分配堆内存块并复制数据。但解析函数会立即遇到指示域名结尾的空字符，因此，如前所述，它会将前一个字符从句点更改为逗号。然后我们遇到了问题

但是，这仅在32位系统上发生。使用unsigned
int存储目标缓冲区迭代器的当前位置会导致处理x64系统时发生更改。让我们更仔细地看一下负责将逗号写入缓冲区的代码片段：

```
180003D60 loc_180003D60:                          ; CODE XREF: DecodeDomainSearchListData+E7↑j
180003D60                 xor     r9d, r9d        ; r9 = 0
180003D63                 lea     ecx, [r9+1]     ; rcx = 1
180003D67                 test    r10b, r10b
180003D6A                 jz      short loc_180003D6F
180003D6C                 add     [r11], ecx
180003D6F
180003D6F loc_180003D6F:                          ; CODE XREF: DecodeDomainSearchListData+1AE↑j
180003D6F                 cmp     ebp, 2          ; state == SECOND_PASS
180003D72                 jnz     short loc_180003D7C
180003D74                 mov     eax, [rsi]      ; rax = pos
180003D76                 sub     eax, ecx        ; --eax
180003D78                 mov     byte ptr [rax+rbx], ',' ; buf[pos-1] = ','
180003D7C
180003D7C loc_180003D7C:                          ; CODE XREF: DecodeDomainSearchListData+1B6↑j
```

使用32位寄存器eax从当前位置减去，但在寻址缓冲区时，代码寻址完整的64位寄存器rax。在AMD64架构上，任何具有32位寄存器的操作都会将寄存器的高半字清零。这意
[0xffffffff]，这是在为缓冲区分配的内存之外。

这些发现与Microsoft的可利用性评分密切相关，因为要利用此漏洞，攻击者必须学习如何在DHCP客户端上执行远程堆攻击，以及对堆内存分配进行充分控制以确保预设值

否则，将数据写入未经检查的地址将导致svchost.exe进程失败，其中包含当前可能托管的所有服务，以及操作系统随后重新启动这些服务。如果情况许可，攻击者也可以

CVE-2019-0726

如果我们仔细查看导致错误的数据类型并将该数据与错误发生进行比较，我们可以看到域名列表已经发生更改，即生成的缓冲区大小不会为零，
但我们仍然会尝试将其写入缓冲区。 为此，列表的第一个元素必须是空字符串，而所有其他元素可能包含名义域名。 例如：

```
+---+---+---+---+---+---+---+
|119| 5 | 0 | 2 |'r'|'u'| 0 |
+---+---+---+---+---+---+---+
```

该选项包括两个元素。 第一个域后缀为空，它立即以空字节结束。 第二个后缀是.ru。
计算出的输出字符串大小为3个字节。同时，数据开头的零将强制函数将逗号写为结果字符串中的前一个字符，但由于迭代器在字符串中的当前位置是零，它将在分配的缓冲

现在我们需要通过实际测试来确认我们的理论结果。
让我们模拟一个DHCP服务器使用带有present选项的消息响应客户端请求的情况，当我尝试在为结果字符串分配的缓冲区的0xffffffff位置写一个逗号时，我们立即发现

```
*** An Access Violation occurred in C:\WINDOWS\System32\svchost.exe -k
LocalServiceNetworkRestricted -p:
The instruction at 00007FFB34413D87 tried to write to an invalid address, 0000025301FFC3DF
 *** enter .exr 000000D5FCDFD3B0 for the exception record
 ***  enter .cxr 000000D5FCDFCEC0 for the context

3: kd> .exr 000000D5FCDFD3B0
ExceptionAddress: 00007ffb34413d87 (dhcpcore!DecodeDomainSearchListData+0x01cb)
   ExceptionCode: c0000005 (Access violation)
   Parameter[0]: 0000000000000001
   Parameter[1]: 0000025301ffc3df
Attempt to write to address 0000025301ffc3df

3: kd> .cxr 000000D5FCDFCEC0
rax=00000000ffffffff rbx=0000025200ad6fd0 rcx=0000000000000001
rdx=0000000000000000 rsi=0000025200ad6fc8 rdi=0000025201ffc3e0
rip=00007ffb34413d87 rsp=000000d5fcdfd5c0 rbp=0000000000000002
r8=0000025200ad7100  r9=0000000000000000 r10=ff3fff3f3fffc300
dhcpcore!DecodeDomainSearchListData+0x1cb:
0033:00007ffb`34413d87 c604382c        mov     byte ptr [rax+rdi],2Ch
ds:002b:00000253`01ffc3df=??

3: kd> k
# Child-SP          RetAddr           Call Site
00 000000d5`fcdfd5c0 00007ffb`34413ea7 dhcpcore!DecodeDomainSearchListData+0x1cb
01 000000d5`fcdfd630 00007ffb`34427090 dhcpcore!UpdateDomainSearchOption+0x5b
02 000000d5`fcdfd680 00007ffb`34418e68 dhcpcore!DhcpExtractFullOptions+0x2a0
03 000000d5`fcdfe1a0 00007ffb`3442c465 dhcpcore!UpdateDhcpContext+0x80
04 000000d5`fcdfe1f0 00007ffb`34428d45 dhcpcore!RenewLease+0x6cd
05 000000d5`fcdfe5b0 00007ffb`3442ba2a dhcpcore!DhcpRenewState+0xe9
06 000000d5`fcdfe720 00007ffb`3443b29b dhcpcore!ReRenewParameters+0x26a
07 000000d5`fcdfe9f0 00007ffb`34421ad8 dhcpcore!AcquireParameters+0x8b
08 000000d5`fcdfea20 00007ffb`34424a34 dhcpcore!DhcpApiProcessAdapterOnlyApi+0x310
09 000000d5`fcdfed90 00007ffb`398243f3 dhcpcore!RpcSrvRenewLease+0x94
0a 000000d5`fcdfedc0 00007ffb`3988dbdd RPCRT4!Invoke+0x73

3: kd> db r8
00000252`00ad7100  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  ................
00000252`00ad7110  00 00 00 00 00 00 00 00-00 00 00 00 63 82 53 63  ............c.Sc
00000252`00ad7120  35 01 05 36 04 c0 a8 11-fe 33 04 00 00 07 08 01  5..6.....3......
00000252`00ad7130  04 ff ff 00 03 04 c0 a8-11 02 06 04 c0 a8 11 01  ................
00000252`00ad7140  02 0f 0b 6c 6f 63 61 6c-64 6f 6d 61 69 6e 2c 04  ...localdomain,.
00000252`00ad7150  c0 a8 11 02 77 05 00 02-72 75 00 ff 0e 01 03 06  ....w...ru......
```

这里寄存器r8包含一个指向传入选项的指针，rdi包含分配的目标缓冲区的地址，而rax包含该缓冲区中必须写入字符的位置。这些是我们在安装了所有更新的系统上获得的

■■■■■■■■■■■■■■[http://blog.ptsecurity.com/2019/05/dhcp-security-in-windows-10-analyzing.html](http://blog.ptsecurity.com/20

上一篇：APT28分析之DDE样本分析 下一篇：CVE-2019-1821：思科P...

1. 0 条回复
   - 动动手指，沙发就是你的了！