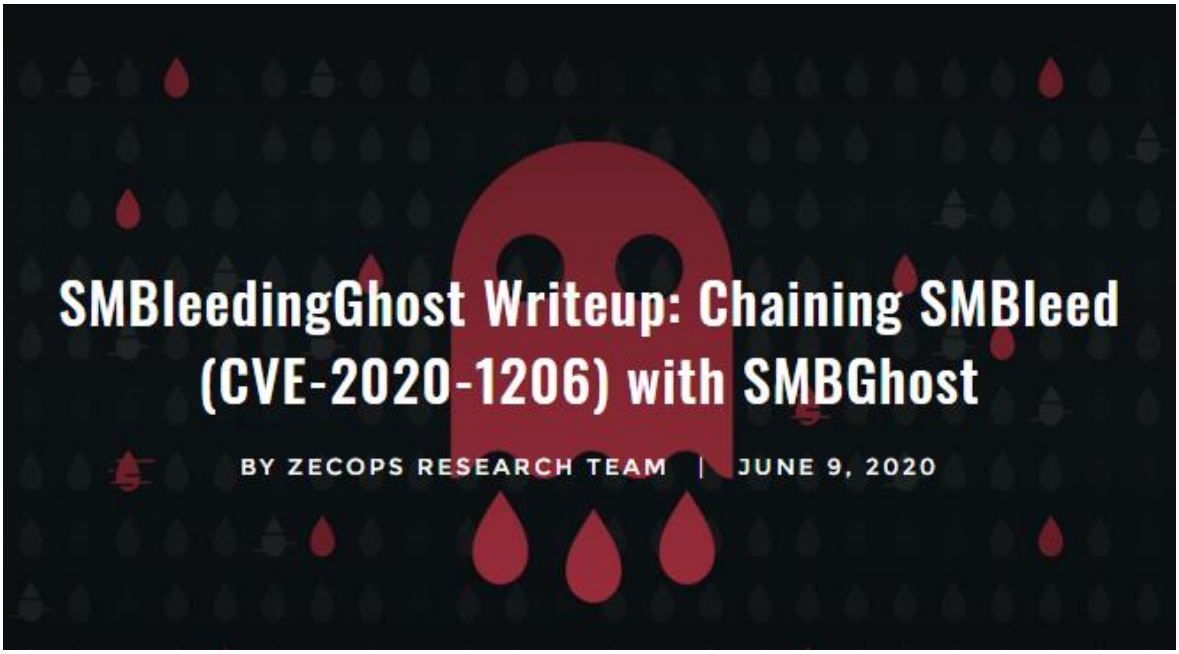
بسمه تعالی

مرکز مدیریت امداد و هماهنگی
عملیات رخدادهای رایانه ای

شرح SMBleedingGhost: زنجیره کردن SMBleed(CVE-2020-1206) با SMBGhost

آسیب پذیری

فهرست مطالب



شکل ۱. شرح SMBleedingGhost: زنجیره کردن SMBleed(CVE-2020-1206) با SMBGhost

۱ مقدمه

آسیب‌پذیری SMBGhost با شناسه CVE-2020-0796 در مکانیزم فشرده‌سازی SMBv3.1.1 حدود سه ماه پیش وصله و برطرف شد. در مطالب مختلفی این آسیب‌پذیری و نحوه بهره‌برداری از آن برای افزایش سطح دسترسی محلی نشان داده شدند. در طول بررسی‌ها درخصوص این آسیب‌پذیری مشخص شد که این تنها نقص در عملکرد از حالت فشرده‌سازی خارج کردن SMB نیست. با بررسی دقیق‌تر مشخص شد که SMBleed در همان تابع SMBGhost رخ می‌دهد. این نقص به مهاجم اجازه می‌دهد تا حافظه‌ی کرنل غیرمجاز را بخواند که در این مطلب، توضیحاتی با جزئیات بیشتر در خصوص این آسیب‌پذیری ارائه خواهد شد.

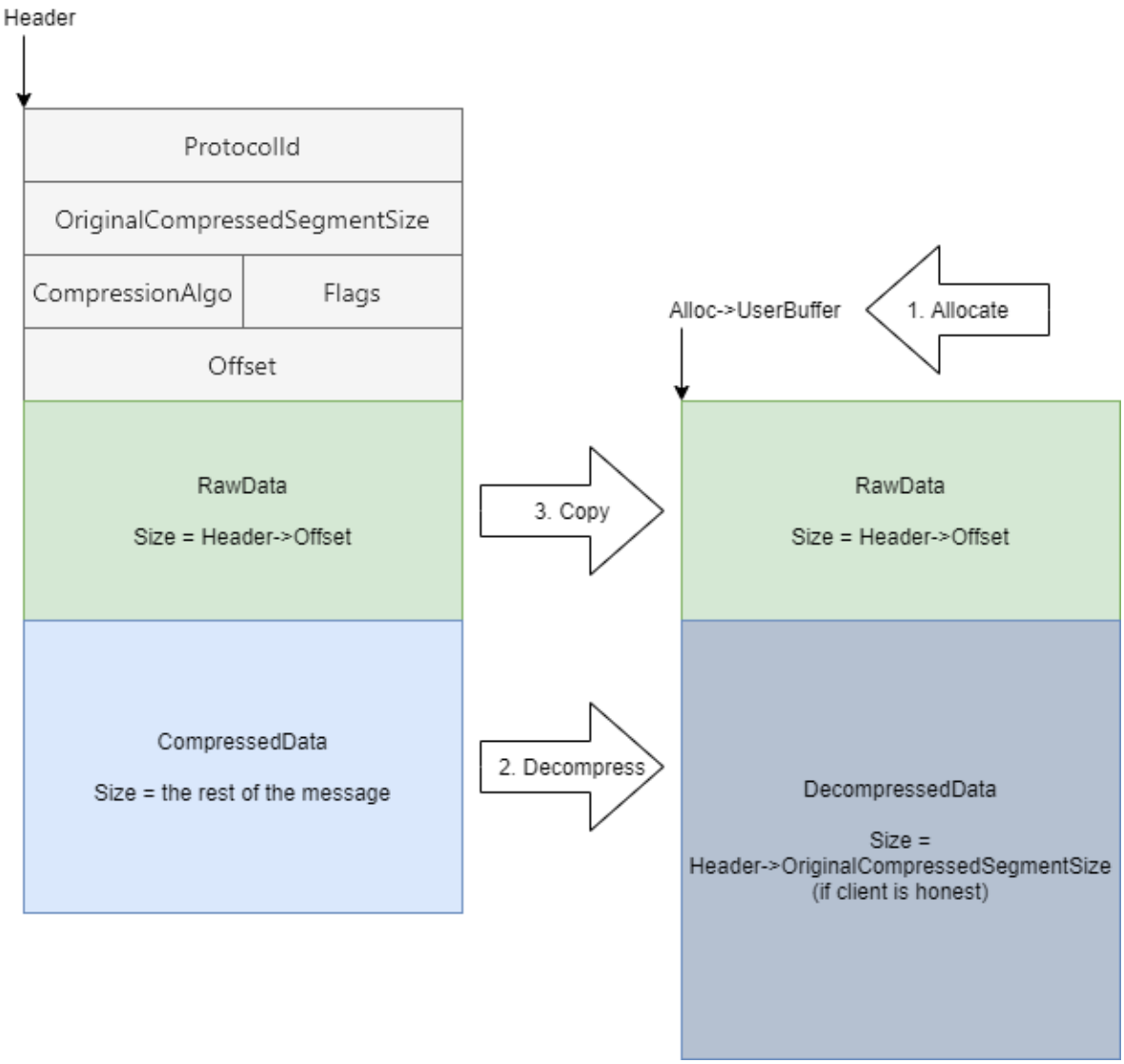
۲ بررسی اولیه

این نقص در همان تابع آسیب‌پذیری SMBGhost، با نام تابع Srv2DecompressData در درایور سرور SMB srv2.sys، رخ می‌دهد. در شکل ۲ نسخه ساده‌ای از تابع، با جزئیات نامربوط حذف شده، آمده است.

```
typedef struct _COMPRESSION_TRANSFORM_HEADER
{
    ULONG ProtocolId;
    ULONG OriginalCompressedSegmentSize;
    USHORT CompressionAlgorithm;
    USHORT Flags;
    ULONG Offset;
} COMPRESSION_TRANSFORM_HEADER, *PCOMPRESSION_TRANSFORM_HEADER;


typedef struct _ALLOCATION_HEADER
{
    // ...
    PVOID UserBuffer;
    // ...
} ALLOCATION_HEADER, *PALLOCATION_HEADER;


NTSTATUS Srv2DecompressData(PCOMPRESSION_TRANSFORM_HEADER Header, SIZE_T TotalSize)
{
    PALLOCATION_HEADER Alloc = SrvNetAllocateBuffer(
        (ULONG)(Header->OriginalCompressedSegmentSize + Header->Offset),
        NULL);
    If (!Alloc) {
        return STATUS_INSUFFICIENT_RESOURCES;
    }


    ULONG FinalCompressedSize = 0;


    NTSTATUS Status = SmbCompressionDecompress(
        Header->CompressionAlgorithm,
        (PUCHAR)Header + sizeof(COMPRESSION_TRANSFORM_HEADER) + Header->Offset,
        (ULONG)(TotalSize - sizeof(COMPRESSION_TRANSFORM_HEADER) - Header->Offset),
        (PUCHAR)Alloc->UserBuffer + Header->Offset,
        Header->OriginalCompressedSegmentSize,
        &FinalCompressedSize);
    if (Status < 0 || FinalCompressedSize != Header->OriginalCompressedSegmentSize) {
        SrvNetFreeBuffer(Alloc);
        return STATUS_BAD_DATA;
    }


    if (Header->Offset > 0) {
        memcpy(
            Alloc->UserBuffer,
            (PUCHAR)Header + sizeof(COMPRESSION_TRANSFORM_HEADER),
            Header->Offset);
    }


    Srv2ReplaceReceiveBuffer(some_session_handle, Alloc);
    return STATUS_SUCCESS;
}
```

شکل ۲. تابع Srv2DecompressData در SMB

تابع Srv2DecompressData پیام فشرده‌شده‌ای را که توسط کاربر ارسال می‌شود، دریافت کرده و مقدار حافظه موردنیاز را اختصاص داده و داده‌ها را فشرده می‌کند. سپس اگر بخش آفست صفر نباشد، داده‌هایی که قبل از داده‌های فشرده شده قرار داده شده‌اند را، همانطور که در ابتدای بافر اختصاص داده شده قرار دارند، کپی می‌کند. در شکل ۳ نحوه کپی کردن نشان داده شده است.

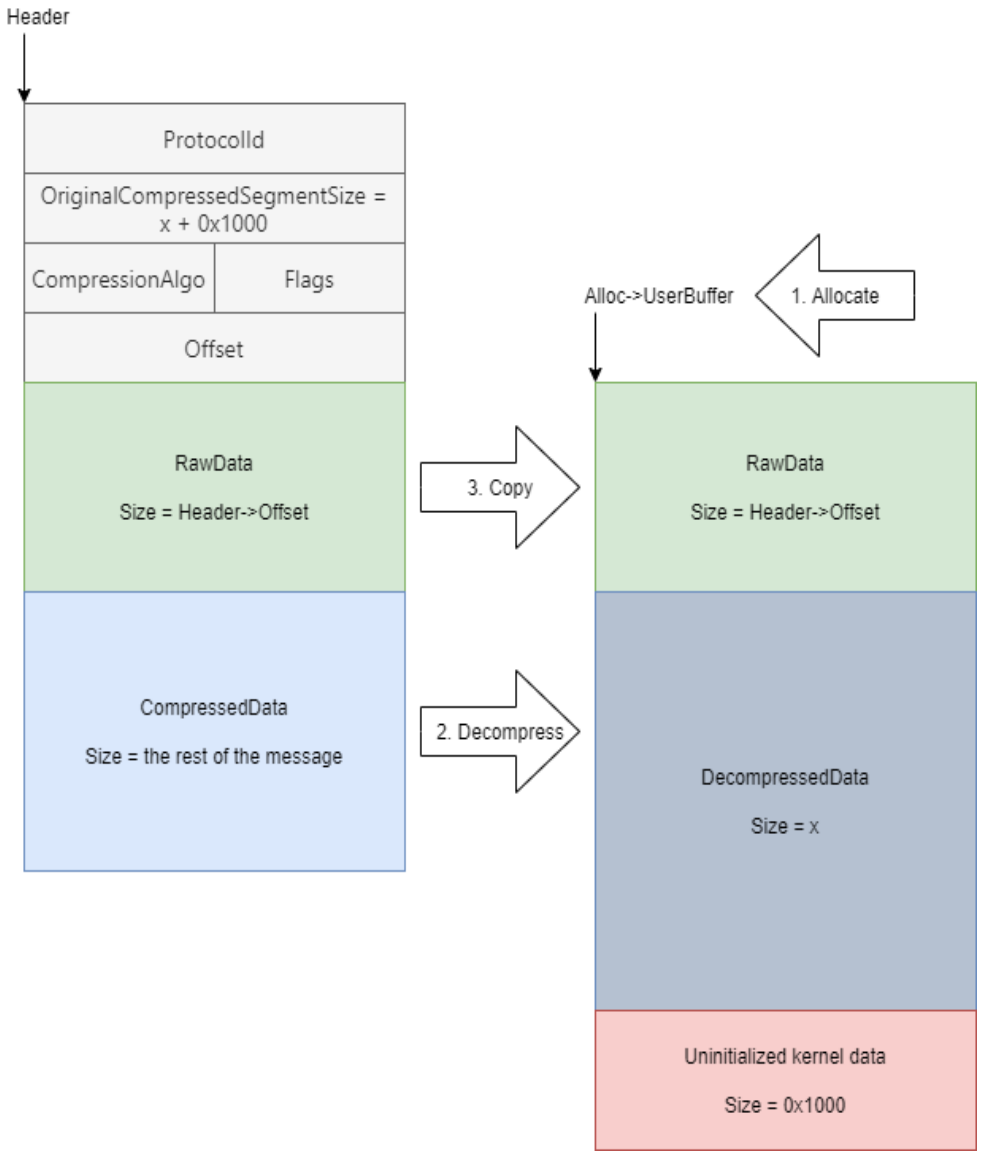


شکل ۳. نحوه‌ی کپی کردن داده‌ها

این آسیب‌پذیری SMBGhost به دلیل عدم بررسی سرریز عدد صحیح رخ داده است. این نقص سه ماه قبل توسط مایکروسافت وصله شد. در شکل ۳ فرض خواهد شد که این تابع برای سرریزهای عدد صحیح بررسی می‌شود و پیام را در این موارد حذف می‌کند اما حتی با وجود این بررسی‌ها، هنوز یک نقص جدی وجود دارد. آیا می‌توان این نقص را تشخیص داد؟ در ادامه این مورد بررسی خواهد شد.

۳ جعل مجدد OriginalCompressedSegmentSize

پیش از این با مقداردهی فیلد OriginalCompressionSegmentSize با یک عدد بسیار بزرگ، از آن بهره‌برداری شده که موجب سرریز عدد صحیح و یک نوشته‌ی خارج از محدوده، شده بود. اگر مقداردهی این فیلد را با عددی که کمی بزرگتر از داده‌های غیرفشرده‌ی ارسالی است تنظیم شود، چه اتفاقی رخ خواهد داد؟ برای مثال اگر اندازه‌ی داده‌ی فشرده‌شده پس از غیرفشرده‌سازی x باشد و اندازه‌ی OriginalCompressedSegmentSize را x+0x1000 تنظیم شود، اتفاق زیر رخ خواهد داد (شکل ۴).



شکل ۴. مثالی برای مقداردهی OriginalCompressedSegmentSize برای بهره‌برداری از آن

همانطور که در شکل ۵ مشخص است، داده‌های مقداردهی نشده کرنل به عنوان بخشی از پیام مورد بررسی قرار می‌گیرند. اگر بررسی‌های قبلی درخصوص این آسیب‌پذیری را مطالعه نکرده باشید ممکن است فکر کنید تابع Srv2DecompressData، بدلیل بررسی‌ای که موجب فراخوانی تابع SmbCompressedDecompresse می شود، فراخوانی نخواهد شد که در شکل ۵ نشان داده شده است.

```
1  if (Status < 0 || FinalCompressedSize != Header->OriginalCompressedSegmentSize) {
2      SrvNetFreeBuffer(Alloc);
3      return STATUS_BAD_DATA;
4  }
```

شکل ۵. فراخوانی Serv2DecompressData

به طور خاص در مثال ما، ممکن است فکر کنید در حالی که مقدار فیلد OriginalCompressedSegmentSize برابر با x + 0x1000 است، مقدار FinalCompressedSize برابر با x خواهد بود. در واقع، مقدار FinalCompressedSize نیز به دلیل شکل پیاده‌سازی تابع SmbCompressedDecompress، x + 0x1000 خواهد شد که در شکل ۶ نشان داده شده است.

```
01  NTSTATUS SmbCompressionDecompress(
02      USHORT CompressionAlgorithm,
03      PUCHAR UncompressedBuffer,
04      ULONG  UncompressedBufferSize,
05      PUCHAR CompressedBuffer,
06      ULONG  CompressedBufferSize,
07      PULONG FinalCompressedSize)
08  {
09      // ...
10
11      NTSTATUS Status = RtlDecompressBufferEx2(
12          ...,
13          FinalUncompressedSize,
14          ...);
15      if (status >= 0) {
16          *FinalCompressedSize = CompressedBufferSize;
17      }
18
19      // ...
20
21      return Status;
22  }
```

شکل ۶. مقداردهی SmbCompressDecompress

نتیجه می‌گیریم که در حالتی که فرآیند خارج کردن از حالت فشرده‌سازی موفقیت‌آمیز باشد، FinalCompressedSize برای نگهداری مقدار CompressedBufferSize که اندازه بافر است، به‌روزرسانی می‌شود.

۴ بهره‌برداری از این آسیب‌پذیری

پیام SMB که جهت نمایش این آسیب‌پذیری از آن استفاده می‌شود(پیام SMR2 WRITE) حاوی فیلدهایی از قبیل میزان بایت برای نوشتن و پرچم‌ها و یک بافر با طول متغیر است. این موارد برای بهره‌برداری از این نقص مناسب است، زیرا می‌توان پیامی را به گونه‌ای طراحی کرد که عنوان را تعیین کرده اما بافر با طول متغیر، حاوی داده مقداردهی نشده، باشد. PoC بر روی مخزن WindowsProtocolTestSuites مایکروسافت قرار داده و این مورد کوچک به تابع فشرده‌سازی اضافه ودر شکل ۷ نشان داده شده است.

```
// HACK: fake size
if (((Smb2SinglePacket)packet).Header.Command == Smb2Command.WRITE)
{
    ((Smb2WriteRequestPacket)packet).PayLoad.Length += 0x1000;
    compressedPacket.Header.OriginalCompressedSegmentSize += 0x1000;
}
```

شکل ۷. اضافه شدن یک بخش کوچک به تابع فشرده‌سازی

البته برای استفاده از این PoC [1]، اعتبارنامه‌ها و یک بخش مشترک با دسترسی نوشتن مورد نیاز است که در بیشتر سناریوها موجود هستند اما این نقص در تمامی پیام‌ها وجود دارد و به احتمال فراوان می‌توان بدون احراز هویت از آن بهره‌برداری کرد. همچنین لازم به ذکر است که نشت حافظه از تخصیص حافظه‌های قبلی در NonPagedPoolNx نشأت می‌گیرد و با توجه به اینکه میزان حافظه تخصیص یافته کنترل می‌شود، ممکن است بتوان داده‌های نشتی را تا حدی کنترل کرد.

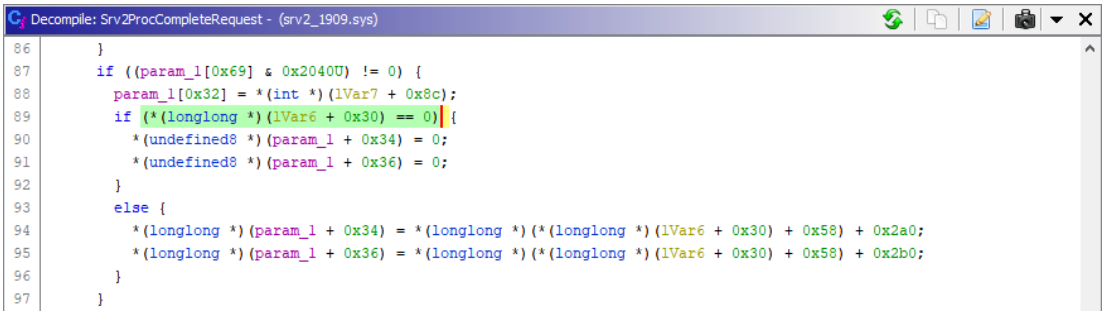
۵ نسخه‌های آلوده‌ی ویندوز

در حال حاضر نسخه‌های ۱۹۰۳، ۱۹۰۹ و ۲۰۰۴ ویندوز ۱۰ آلوده هستند. در حین بررسی‌ها، آزمایش PoC باعث کرش کردن یکی از ماشین‌های ویندوز ۱۰ نسخه ۱۹۰۳ شد. پس از تجزیه و تحلیل کرش انجام شده با Neutrino، مشاهده شد که اولین نسخه‌های وصل‌نشده ویندوز ۱۰ نسخه ۱۹۰۳ در هنگام کار با بسته‌های معتبر و فشرده SMB، دارای یک نقص درخصوص اشاره‌گر تهی ارجاع داده نشده، بودند.

Decompile: Srv2ProcCompleteRequest - (srv2_1903.sys)

```
86      }
87      if ((param_1[0x69] & 0x2040U) != 0) {
88        param_1[0x32] = *(int *)(lVar7 + 0x8c);
89        *(longlong *)(param_1 + 0x34) = *(longlong *)(*(longlong *)(lVar6 + 0x30) + 0x58) + 0x2a0;
90        *(longlong *)(param_1 + 0x36) = *(longlong *)(*(longlong *)(lVar6 + 0x30) + 0x58) + 0x2b0;
91      }
92      if (*(char *)(lVar6 + 7) != '\0') {
93        LOCK();
94        piVar2 = (int *)(lVar7 + 0xc4);
95        iVar5 = *piVar2;
96        *piVar2 = *piVar2 + -1;
97        param_1[0x29] = iVar5;
```

شکل ۸. یک سیستم وصل نشده و نقص در اشاره‌گر تهی ارجاع‌داده نشده



شکل ۹. یک سیستم وصله شده و اضافه کردن روال بررسی اشاره‌گر تهی

در ادامه خلاصه‌ای از نسخه‌های آلوده‌ی ویندوز ۱۰ و نصب به‌روزرسانی‌های مربوطه، آورده شده‌است:

Windows 10 Version 2004

Update	SMBGhost	SMBleed
KB4557957	Not Vulnerable	Not Vulnerable
Before KB4557957	Not Vulnerable	Vulnerable

Windows 10 Version 1909

Update	SMBGhost	SMBleed
KB4560960	Not Vulnerable	Not Vulnerable
KB4551762	Not Vulnerable	Vulnerable
Before KB4551762	Vulnerable	Vulnerable

Windows 10 Version 1903

Update	Null Dereference Bug	SMBGhost	SMBleed
KB4560960	Fixed	Not Vulnerable	Not Vulnerable
KB4551762	Fixed	Not Vulnerable	Vulnerable
KB4512941	Fixed	Vulnerable	Vulnerable
None of the above	Not Fixed	Vulnerable	Potentially vulnerable*

در ضمن باید تاکید کرد که تلاشی برای دور زدن نقص اشاره‌گر تهی ارجاع داده نشده، انجام نگرفته است اما ممکن است این کار از طریق متدی دیگر(بطور مثال، استفاده از SMBGhost Write-What-Where primitive) ممکن باشد.

۶ زنجیرکردن SMBleed با SMBGhost برای فراهم‌آوری پیش تایید اجرا از راه دور در SMBleedingGhost

بهره‌برداری از آسیب‌پذیری SMBleed با پیچیدگی بیشتر نسبت به مورد قبلی ممکن است. در این گزارش از این آسیب‌پذیری به همراه نقص موجود در SMBGhost برای فراهم‌آوری شرایط اجرای کد از راه دور (RCE) استفاده شده است.

۷ تشخیص

کاربران با استفاده از ZecOps Neutrino بهره‌برداری از SMBleed و SMBGhost را تشخیص می‌دهند و کار اضافه‌ای توسط کاربر لازم نیست. همچنین می‌توان اشاره کرد که SMBleed و SMBGhost را می‌توان از راه‌های مختلف دیگری از جمله بررسی Crash Dump و ترافیک شبکه تشخیص داد.

۸ توصیه‌ها

کاربران می‌توانند نگرانی‌های خود را درخصوص آسیب‌پذیری‌های SMBleed و SMBGhost با انجام توصیه‌های زیر از بین ببرند:

۱. به‌روزرسانی ویندوز این مشکل را کاملاً برطرف می‌کند.(پیشنهاد می‌شود)
۲. مسدود کردن پورت ۴۴۵، حرکات بعدی با استفاده از این آسیب‌پذیری‌ها را متوقف می‌کند.
۳. اجرای ایزوله‌سازی هاست.
۴. غیرفعال کردن فشرده‌سازی SMB 3.11(پیشنهاد نمی‌شود)

۹ منابع

1. https://blog.zecops.com/vulnerabilities/smbleedingghost-writeup-chaining-smbleed-cve-2020-1206-with-smbghost/
2. https://github.com/ZecOps/CVE-2020-1206-POC
3. https://github.com/ZecOps/CVE-2020-0796-RCE-POC