MRLAB

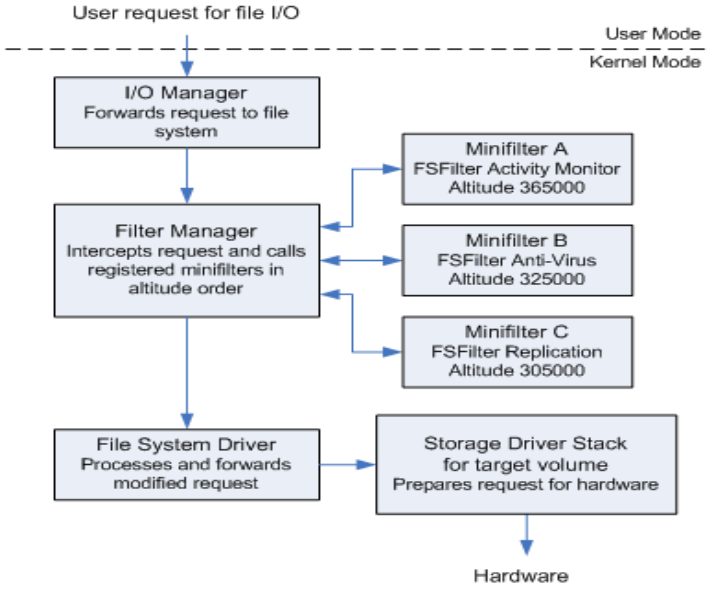
Windows kernel səviyyəsində ümumi ransomware aşkarlanması üçün tələ

Azərbaycan Respublikası Xüsusi Rabitə və İnformasiya Təhlükəsizliyi Dövlət Xidməti - Kompüter İnsidentlərinə qarşı Mübarizə Mərkəzi - Malware Research Lab - S. Abasov - 25 Aprel 2022

Bundan öncə ki, məqaləmizin birində sizlərə ransomware haqqında məlumat vermişdim. Bu məqalədə bu tipli zərərvericiləri aşkarlamaq üçün hansı metodlardan istifadə edə bilərik bu haqda məlumat verəcəm. Məqalədə istifadə ediləcək metod bütün zərərvericilərə qarşı istifadə edilə bilməsədə araşdırma aparılaraq və uyğun əlavələr ilə digər ransomwarelərin aşkarlanmasında effektiv metod olaraq istifadə oluna bilər.

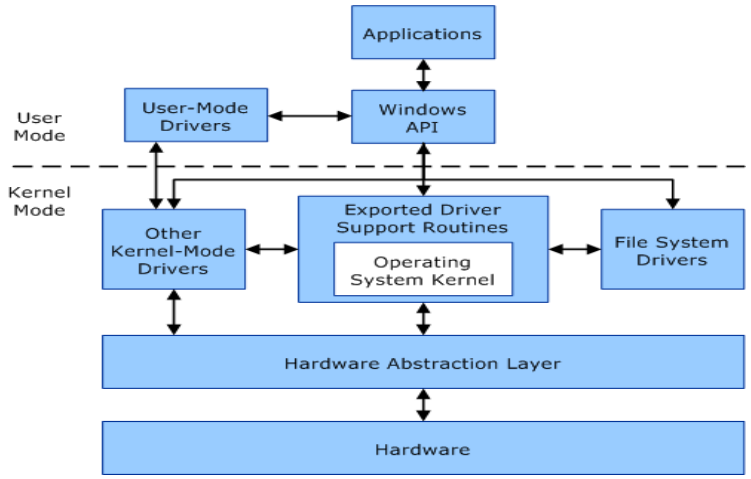
Yanaşma

Bildiyiniz kimi bu tipli zərərvericilərin əsas məqsədi sistemdə mövcüd olan, istifadəçinin kritik fayllarını (sənəd,şəkil,müsiqi,mətn vs) şifrələyərək istifadəçidən fidyə istəməkdir.Əsas məqsədimiz zərərverici bu tip əməliyyatları həyata keçirən zaman davranış patternlərini analiz edərək əməliyyatı həyata keçirən zərərlinin aşkarlanmasıdır. Bunun üçün sistemdə kiçik bir honeypot (bal qabı) hazırlamaqdır. Honeypot mexanizmi uzun zamandır istifadə edilən bir metoddur. Zərərvericilər normal fəaliyyətləri zamanı özləri bilmədən tələyə düşür və digər tərəf isə bundan istifadə edərək zərərli haqqında məlumat toplayırlar.Məndə bu tip zərərvericiləri aşkar etmək üçün sistemdə honeypot quraraq nəticənin nə olacağını öyrənmək istədim. Bunun üçün **Windows Filter Manager** konsepsiyasından istifadə edəcəyəm. Bunun köməkliyi ilə honeypotumuz üzərində hansı əməliyyatlar aparılır bunları aşkar etmək mümkündür.Filter Manager – Filter manager sistem kernel sürücüsüdür. Bu sürücü istifadəçilərə öz file-system filter sürücülərini yazmağa imkan yaradır. File-system stack -də özünə yer ayırsan bu sürücülər fayl əməliyyatları haqqında məlumat toplamaq və lazım gələr isə müdaxilə etmək imkanına sahibdirlər.Köhnə "**legacy filter**"- dən fərqli olaraq minifilter adı verilən bu mexanizm daha rahat və daha sürətli filter əməliyyatlarına imkan yaradır.

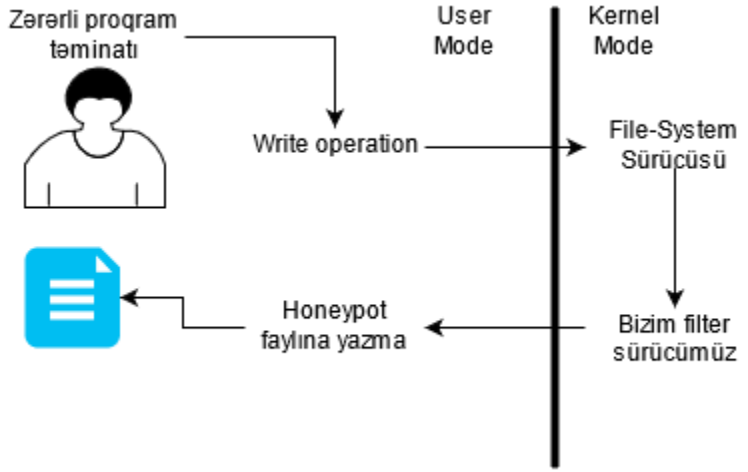


Daha ətraflı: https://docs.microsoft.com/en-us/windows-hardware/drivers/ifs/filter-manager-concepts

Hazırlayacağım honeypot üçün ilk olaraq sistemdə kritik fayl formatlarında saxta fayllar yaradacağam. Minifilter köməkliyi ilə bu fayllar üzərində hansı əməliyyatlar aparılır bunları müşahidə edərək hədəf faylın zərərverici olub olmadığına qərara verəcəyəm. **Metod yalnız yanaşma olaraq qəbul edilməlidir. Dəqiqliyi test edilməmişdir**. Hər hansı bir proqram təminatı sistemdə olan fayllar üzərində əməliyyat apardığı zaman arxa fonda sistem funksiyaları çağrılır. Yəni faylı açmaq üçün istifadə edilən hər hansı bir funksiya **kernel32.dll** kitabxanasında **CreateFile*** funksiyasını , bu funksiya isə öz növbəsində ntdll.dll içərisində Nt-api funksiyalarını çağırır. NT funksiyaları isə gate dediyimiz qapılardan istifadə edərək cari threadı kernel səviyyəsinə salır. Əməliyyat tamamlandıqdan sonra isə kernel səviyyəsindən yenidən istifadəçi səviyyəsinə geri qaytarılır.



Yaradacağım sürücü kernel səviyyəsində fayl əməliyyatlarını izləyərək bizim saxta fayllara yazma vəya silmə əməliyyatılarını izləyəcək. Əgər əməliyyyat bir neçə honeypot faylı üzərində aşkarlanar isə əməliyyatı həyata keçirən prosesin təhlükəli olduğuna qərara veriləcək.
Qeyd edilənləri daha yaxşı anlamaq üçün aşağıda ki, şəkilə diqqət yetirin.



Minifilter sürücüsünün hazırlanması - Minifilter sürücümüzü hazırlamaq üçün ilk öncə kiçik bir araşdırma aparmalıyıq. Daha dəqiq desək hansı **IRP** sorğuları izləməliyik bunu öyrənərək sürücümüzü buna uyğun yazmalıyıq.Test əməliyyatını wannacry ransomware üzərində apardım. 2 ədəd fayl (h1.docx, h2.bmp) yaradaraq ransomware-nin bu fayllar üzərində hansı əməliyyatları apardığına baxdım.

```
wanna.exe   IRP_MJ_WRITE   C:\Users\admin\Desktop\honeypot\h1.txt
wanna.exe   IRP_MJ_WRITE   C:\Users\admin\Desktop\honeypot\h2.bmp
```

Yuxarıda ki, şəkildə gördüyünüz kimi zərərli şifrələnəcək fayla **IRP_MJ_WRITE** sorğusu üzərindən məlumat yazır. Mən filter sürücümdə məhz bu sorğuları izləyərək əgər yaratdığım honeypot fayllarına məlumat yazılar isə həmin prosesi şüphəli proses kimi qəbul edəcəyəm. Lazım olan məlumatları topladıqdan sonra keçirəm kodların hazırlanmasına.Callback operation zamanı **IRP_MJ_WRITE** sorğusunu izləmək üçün **FsFilter1PreWrite** funksiyasını istifadə edəcəyəm.

```
CONST FLT_OPERATION_REGISTRATION Callbacks[] = {
    {IRP_MJ_WRITE, 0, FsFilter1PreWrite, NULL},
    {IRP_MJ_OPERATION_END}
};
```

Bu funksiyaya gələn məlumatlar üzərindən ilk öncə hansı fayla məlumat yazıldığını görürük. Yazılan fayl bizim honeypot faylımızdırsa **IRP_MJ_WRITE** sorğusunu göndərən prosesin PID dəyərini bir siyahı içərisində saxlayıram. Əgər eyni proses digər honeypot fayllarınada eyni sorğunu göndərər isə bu proses şüphəli proses olaraq qeydə alınacaq.

P.S Kodların hazırlanmasında məqsəd məlumat vermək xarakterlı olduğu üçün heç bir təhlükəsizlik tədbirləri görülməyib. İlk öncə gələn **PFLT_CALLBACK_DATA** üzərindən fayl məlumatlarını **FltGetFileNameInformation** funksiyasının köməkliyi ilə götürdüm.

```
status = FltGetFileNameInformation(Data,
    FLT_FILE_NAME_NORMALIZED | FLT_FILE_NAME_QUERY_DEFAULT,
    &FileNameInfo);
```

Daha sonra **FltParseFileNameInformation** fayl adı parse edirik. **PreWrite** əməliyyatı zamanı lazım olduğundan çox məlumat gəldiyi üçün parse edilən fayl adı içərisində bizə lazım olan fayl uzantılarını qəbul etmək üçün RtlCompareUnicodeStringfunksiyasını çağırmaq işimizi biraz rahatlacaq. Burada yalınız ***.bmp** və ***.docx** uzantılarına yazma əməliyyatlarını qəbul edirəm.

```
status = FltParseFileNameInformation(FileNameInfo);
if (!NT_SUCCESS(status))
{
    FltReleaseFileNameInformation(FileNameInfo);
    return FLT_PREOP_SUCCESS_NO_CALLBACK;
}

if (RtlCompareUnicodeString(&extension, &FileNameInfo->Extension, TRUE) != 0)
{
    FltReleaseFileNameInformation(FileNameInfo);
    return FLT_PREOP_SUCCESS_NO_CALLBACK;
};
```

Bura qədər olan əməliyyatları test etmək üçün sürücümüzü compile edib Dbgview alətinin köməkliyi ilə loglarabaxdım.

```
21   \Device\HarddiskVolume3\Users\admin\Desktop\honeypot\h2.docx
22   90584375000 - STORMINI: StorNVMe - POWER: ACTIVE
23   90584375000 - STORMINI: StorNVMe - POWER: IDLE
```

```
C:\Windows\System32\cmd.exe - python
C:\Users\admin\Desktop\honeypot>python
Python 3.10.0 (tags/v3.10.0:b494f59, Oct  4 2021, 19:00:18) [MSC v
it (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more informat
>>> f = open("h2.docx","wb")
>>> f.write(b'1')
1
>>> f.close()
```

İstədiyim nəticəni aldıqdan sonra fayl pathını MS-DOS formatına convert etdim. Bu fayl yolunu compare zamanı lazımdır.

```
C:\Users\admin\Desktop\honeypot\hl.bmp
```

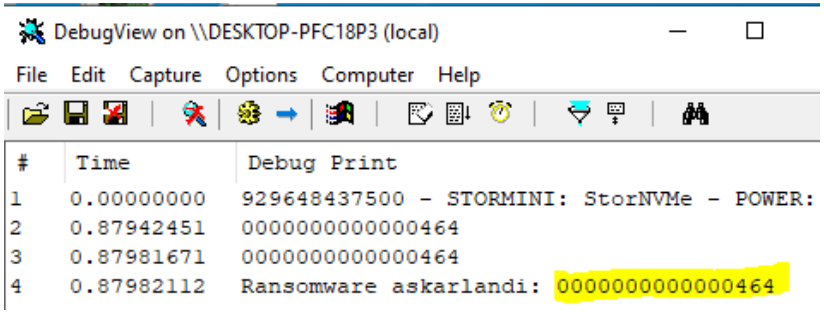
Bundan sonra ilk olaraq compare funksiyası ilə yazılan faylları müqayisə edirəm.

```
BOOLEAN Compare(PUNICODE_STRING str1)
{
    if (RtlCompareUnicodeString(str1, &honeypot_files[0], TRUE) == 0)
    {
        return TRUE;
    }
    if (RtlCompareUnicodeString(str1, &honeypot_files[1], TRUE) == 0)
    {
        return TRUE;
    }
    return FALSE;
}
```

Əgər yazılan fayl bizim array içərisində olan fayllar ilə eynidirsə bu zaman datanı yazan prosesin pid dəyəri başqa bir array içərisinə yazılır. Əgər hər 2 fayla yazan PID eyni olarsa bu zaman sürücü **ransomwarenin** aşkarlandığı haqqında mesaj verir.

```
if (Compare(&DosName->Name))
{
    DbgPrint("%p\n", PsGetCurrentProcessId());
    if (dwCount == 1)
    {
        if (PsGetCurrentProcessId() == (HANDLE)pids[dwCount - 1])
        {
            DbgPrint("Ransomware askarlandi: %p\n", PsGetCurrentProcessId());
        }
    }
    else {
        pids[dwCount] = PsGetCurrentProcessId();
        dwCount += 1;
    }
}
```

Sürücümüzü rebuild edərək test edirik.



Oxşar yanaşma ilə digər zərərliləridə aşkarlamaq üçün mexanizm hazırlana bilər. Burada vacib məsələ bu tipli zərərliləri analiz edərək oxşar tərəflərini təyin etmək və sürücünü buna uyğun yazmaqdır.

İstinadlar

https://docs.microsoft.com/en-us/windowshardware/drivers/ifs/filter-manager-concepts
https://en.wikipedia.org/wiki/Ransomware