Windows Scripting - חלק א': Introduction

מאת ניר נטר

הקדמה

ברוכים הבאים למאמר הראשון בסדרת המאמרים על Windows Scripting! בסדרת המאמרים אציג ואסביר שפות Scripting שונות שמובנות ב-Windows בדגש על PowerShell. במאמר הנוכחי, אציג סקירה כללית על ההסטוריה של שפות Scripting שונות ב-Windows ואף אציג קצת יותר לעומק את חלקן.

כשאני אומר Windows Scripting אני מתכוון לשפות Scripting שמובנות ב-Windows. Windows היא מערכת הפעלה נפוצה, היא נמצאת כמעט בכל בית או עסק. אם נדע לשלוט במערכת ההפעלה על ידי אותן שפות Scripting בצורה טובה, תהיה לנו יכולת מיידית לעבוד ולנצל את המיטב ממערכת ההפעלה בהתראה של רגע, גם אם אין לנו Python או את ה-IDE האהוב עלינו. נוכל בהתראה של רגע לפתוח את ה-CMD, PowerShell או לכתוב VBScript מהיר כדי לעשות מה שאנחנו רוצים - ללא צורך בתוכנות חיצוניות!

אז מה אנחנו יכולים לעשות עם אותן שפות Scripting?! הן ישנות, הן גרועות! תנו לי פייתון עכשיו! אוקיי, אז זה נכון. פייתון זה מגניב וחזק (ואפילו מגיע מראש עם רוב הפצות הלינוקס, אבל אנחנו ב-Windows עכשיו). כבר עברנו על זה, יש המון מערכות Windows ואנחנו רוצים ללמוד להשתמש בהן בצורה הכי

טובה שאפשר. פייתון לא מותקנת מראש ב-Windows, Deal with it. אז בכל זאת, מה אנחנו יכולים לעשות? Well, די הרבה. לאותן שפות Scripting יש התממשקות מאוד טובה עם Windows. הן שפות שמעולות לאוטומציות, ניהול המחשב, הן מעולות ל-System Administrators או כמעט לכל שימוש אחר שמתבצע אל מול מערכת ההפעלה. שלא נדבר על האקרים. אם לתוקף יש גישה למחשב שלך, יש לו מגוון אפשרויות נוחות כדי להריץ לוגיקות קצת יותר מסובכות על המחשב שלך ללא צורך בקימפול, העברת הבינארי למחשב הנתקף והרצתו. ממש נוח, אה?

מיקרוסופט דאגו שיהיה לנו נוח עם מגוון רחב של שפות סקריפטים או שירותים שמנגישים לנו את המחשב ומאפשרים לנו גמישות, כמה נחמד מצדם. הם הביאו לנו את Batch, VBScript ואפילו גרסה שלהם ל-JavaScript, JScript! השיא היה ב-PowerShell, שהוא סוג של CMD על הרבה כוסות קפה, עליו נפרט בהמשך ובסדרת המאמרים לעומק. מיקרוסופט דאגו לנו לשירותים כמו WMI, Windows Management Instrumentation שמנגיש את מערכת ההפעלה בצורה של אובייקטים ככה שנוכל לנהל את מערכת ההפעלה בקלות ולקבל עליה מידע. WMI גם מספקת מנגנון טריגרים עשיר, הרצת Batch/VBScript/Jscript אם קופץ אירוע (קבצי MOF). הם דאגו לנו לעבודה נוחה מול COM בשפות כמו VBScript או PowerShell והם אפילו סיפקו לנו את WSF, Windows Scripting File שמאפשר לנו לכתוב סקריפט אחד בכמה שפות שונות. לדוגמא, נוכל לכתוב פונקציה ב-VBScript ולגשת אליה בפייתון! מגניב, לא?

שרדתם עד כאן? אתם בטח נלהבים כמוני. לפני שנכיר את השפות השונות ואת היכולות השונות, נעבור קצת לשיעור היסטוריה כדי להבין מאיפה הכל התחיל, תחזיקו חזק.

היסטוריה

מערכת ההפעלה DOS הביאה לנו את COMMAND.COM שזו התוכנה הראשונה שרצה כשהמחשב עולה ב-DOS. התוכנה הביאה לנו Shell איתו נוכל לעבוד אל מול המחשב ומערכת ההפעלה. באותו Shell כתבנו פקודות שבעזרתן אנחנו יכולים לסייר במחשב, לקבל עליו מידע ולפקד עליו. רצף פקודות כאלה יוצר לנו סקריפט שכתבנו למערכת ההפעלה כדי לבצע אוטומציות, לממש לוגיקות וכו'... אלה הם קבצי Batch.

הגיעו שנות ה-90 וחברת Microsoft שחררה סדרת מערכות הפעלה, Windows 9x כשבהן הוסיפו את cmd.exe שלמעשה היה הרחבה ל-COMMAND.COM. הם הוסיפו מגוון רחב של אפשרויות ופקודות חדשות וכיפיות. לאורך שחרורים של גרסאות Windows, מיקרוסופט הוסיפו כלים נוספים כמו wmic.exe, netsh.exe וכו'... שרק האיצו את כוחו של cmd.exe ושל Batch Scripts. בעזרת כלל הכלים שקיימים במערכת ההפעלה ניתן לקבל מידע רב על המחשב, ליצור אוטומציות, לתפעל את המחשב בצורה הרבה יותר מהירה ויעילה! Cmd.exe נמצא במקביל ל-GUI של מיקרוסופט שמאפשר לנהל את המחשב, אבל CLI הרבה יותר שווה מ-GUI.

בשנת 1996 מיקרוסופט שחררה את VBScript, שפה שמבוססת על Visual Basic שמיד היוותה תחליף לקבצי Batch. עבור Administrators זו הייתה קפיצת מדרגה מבחינת כתיבת אוטומציות וניהול הרשת/מחשבים שלהם. השפה הרבה יותר גמישה, בעלת יותר פונקציונאליות והרבה יותר מובנת. היא מאפשרת לנו התממשקות נוחה עם COM, תכונה שמוסיפה לשפה כוח רב. כמו כן, השפה לוקחת חלק בפיתוח אתרים, בצד לקוח ובצד שרת, אך ללא הצלחה רבה. Well, אפשר להבין... רק Internet Explorer תומכת בשפה.

לאורך השנים, VBScript פותחה ומגרסה 1.0 בשנת שחרורה היא הגיעה לגרסה 5.8. אך אל תטעו, כולנו יודעים שמיקרוסופט לא מאוד טובים בספירת הגרסאות למוצרים שלהם. היא קפצה מגרסה 3.0 לגרסה 5.0. לאורך השחרורים של VBScript, מיקרוסופט הוסיפו תמיכה ב-Classes, COM, פונקציות שונות, לולאות מורכבות וכו'...

נתקדם קצת בזמן... החל מ-Windows 98 מיקרוסופט הוסיפו למערכת ההפעלה שלה את WSH, Windows Scripting Host שהיוותה מנוע להרצת סקריפטים. היא מריצה כברירת מחדל רק VBScript ו-Jscript. אך ניתן להריץ דרכה עוד שפות סקריפטינג כמו Perl או Python (רק אם הן מותקנות על המחשב). WSH אפשרה להעיף את קבצי ה-Batch מהעולם והביאה מקום לשפות יותר מורכבות ככה שניתן יהיה לכתוב סקריפטים יותר מוצלחים ועם לוגיקות יותר מורכבות. בנוסף, WSH הכניסה ל-Windows את קבצי WSF, Windows Script File, פורמט קבצי XML שמאפשר הרצת כמה שפות סקריפטינג יחד.

כש-Windows 2000 שוחרר, הוא הביא איתו את WMI שמעניק יכולות רבות לשפות סקריפטינג ב-Windows ובאופן כללי לפיתוח ב-Windows (עד היום). מיקרוסופט לא שכחה את מערכות ההפעלה האחרונות שלה והוסיפה תמיכה במנגנון ל-Windows 98 ו-Windows NT 4.0 החל מ-SP4. במשפט אחד, WMI משקפת את מערכת ההפעלה דרך אובייקטים ומחלקות (נרחיב בהמשך). WMI האיצה שפות כמו VBScript מבחינת פונקציונאליות ואינטרקציה מול Windows.

באוגוסט 2002, מיקרוסופט החלו בפיתוח Shell חדש, פרויקט שנקרא Monad שמטרתו לייצר שפת Scripting חדשה ל-System Administrators. השפה תתבסס על .NET Framework. שפורסמה זמן קצר לפני כן, בפברואר של אותה שנה. ב-יוני 2005 שוחררה גרסת הבטא הראשונה של הפרויקט עד שבאפריל 2006 שונה השם ל-Windows PowerShell. בנובמבר 2006, PowerShell שוחררה למערכות הפעלה החל מ-Windows XP והגיעה מובנת החל מ-Windows 7. כיום PowerShell נמצאת בגרסה 5.0.

שימושים ודוגמאות

אז הלהבתי אתכם בהתחלה (לפחות ניסיתי) ואחר כך שרדתם שיעור היסטוריה קצר. זה הזמן להראות קצת פרקטיקה! עכשיו אראה לכם קצת שימושים ודוגמאות לשפות סקריפטים ב-Windows. נעבור על כמה שפות ומנגנונים של Windows כדי לקבל תחושה על הפוטנציאל שטמון בהם.

Batch

הדוגמא הראשונה היא העתקת ה-System Event Log של מחשב כלשהו לתיקיה יעודית על ה-DC ברשת כלשהי. קבלו סיטואציה, אתם אנשי IT נחמדים שרוצים לתחקר מה קורה במחשבים שלכם ברשת. אבל אתם עצובים כי אתם לא רוצים לצאת מהמשרד שלכם עם דיסק און קי ולגשת פיזית לכל מחשב כדי להעתיק את הקבצים. אתם גם לא רוצים לעשות עבודה שחורה ולהתחבר לכל מחשב מרחוק וכך להעתיק את הקובץ. אז מה אתם עושים? Group Policy! ווהו! ככה תוכלו להריץ קוד על כל המחשבים ברשת ביחד. ומה תריצו? קובץ Batch נחמד ויפה שיעשה לכם את העבודה! כמה נוח:

```
@echo off
net use \\DC\Share\Logs /USER:DOMAIN\USER RandomPassword
for /f %%a in ('wmic nicconfig where "IPEnabled='True'" get MACAddress^|findstr :') do @(
       mkdir \\DC\Share\Logs\%COMPUTERNAME%_%%a
       copy %windir%\System32\winevt\Logs\System.evtx \\DC\Share\Logs\%COMPUTERNAME%_%%a
/Y
)
net use \\DC\Share\Logs /d
```

<u>אז מה עשינו פה?</u>

השתמשנו ב-net use כדי להתחבר ל-Share שיצרנו מבעוד מועד עם User בעל הרשאות מתאימות. לאחר מכן, אנחנו משתמשים ב-for של batch (שהוא עקום במיוחד) כדי לאתחל משתנה %%a עם הפלט של הרצת wmic (שהוא CLI Shell של WMI) שיביא לנו את ה-MACAddress של ה-NIC שפועל כרגע (<u>אני מניח שיש רק אחד</u>).

לאחר מכן, אני משתמש ב-mkdir כדי ליצור תיקיה ששמה הוא "שם המחשב_MAC". לדוגמא: PC_00:11:22:33:44:55. לבסוף, אני משתמש ב-copy כדי להעתיק את הקובץ של System Event Log (עם Flag /Y כדי לבצע overwrite במידה והקובץ כבר קיים).

בסוף, אנחנו מוחקים את החיבור שיצרנו עם net use.

דוגמא נוספת, שהיא די דומה לקודמת לה. הסיפור דומה לסיפור הקודם, אתם אנשי IT נחמדים שרוצים להכיר מקרוב את כל המחשבים שלכם ברשת. לשם כך, אתם רוצים להריץ systeminfo שיספק לכם מידע

כמו hotfixes מותקנים, זיכרון, מערכת הפעלה, תאריך התקנה של המחשב, boot time וכו'... בנוסף, אתם רוצים מידע מלא על רכיבי הרשת שנמצאים במחשב. לשם כך, אתם משתמשים ב-ipconfig /all.

בדוגמא הבאה אנחנו עושים דבר זהה לדוגמא הקודמת, רק שהפעם אנחנו כותבים קובץ ל-Temp Folder אליו אנחנו כותבים מידע על המחשב ומיד לאחר מכן מעתיקים את הקובץ עם המידע שלנו ל-Share המתאים על ה-DC:

```
@echo off
net use \\DC\Share\Info /USER:DOMAIN\USER RandomPassword
for /f %%a in ('wmic nicconfig where "IPEnabled='True'" get MACAddress^|findstr :') do @(
        mkdir \\DC\Share\Info\%COMPUTERNAME%_%%a
        echo. > %temp%\info.txt
        systeminfo >> %temp%\info.txt
        ipconfig /all >> %temp%\info.txt
        copy %temp%\info.txt \\DC\Share\Info\%COMPUTERNAME%_%%a /Y
        del %temp%\info.txt
)
net use \\DC\Share\Info /d
```

VBScript

אז עלינו רמה מסתם פקודות ב-Batch Files. השפה מאפשרת לנו לבצע לוגיקות יותר מורכבות, אנחנו פחות מוגבלים בכתיבת הקוד שלנו. בקיצור - הרבה יותר כיף לנו!

הפעם אביא לכם דוגמא של סקריפט קצרצר שידפיס לכם את כתובת ה-IP החיצונית של המחשב שלכם, ועל הדרך ידפיס לכם האם החיבור שלכם תקין.

```
On Error Resume Next

Sub Main()
    On Error Resume Next

    Dim httpConnection : Set httpConnection = CreateObject("Microsoft.XMLHTTP")
    httpConnection.open "GET", "https://wtfismyip.com/text", False
    httpConnection.send ""

    If httpConnection.status = 200 Then
        WScript.Echo httpConnection.responseText
    Else
        WScript.Echo "Dude... Something is wrong."
    End If
End Sub

Main()
```

אז מה עשינו פה? השתמשנו ב-COM Object שנקרא XMLHTTP כדי לשלוח HTTP GET ל-wtfismyip.com/text. לשם כך, השתמשנו בפונקציה CreateObject. האתר אמור להדפיס את כתובת ה-IP שממנו אנחנו יוצאים. לאחר מכן אנחנו בודקים את ה-HTTP Code שחזר ובודקים האם הוא 200. אם כן, אנחנו לוקחים את ה-Response שאמור להיות רק כתובת ה-IP ואנחנו מדפיסים אותה. אם הייתה בעיה, אנחנו מדפיסים הודעה שמשהו לא בסדר. ממש לא מסובך ☺

האפשרות לגשת לאובייקטי COM שונים על ידי VBScript ממש מחזקת את השפה. הרבה תוכנות שאתם מתקינים על המחשבים מייצאים COM Interfaces כך שניתן להשתמש בהם. לדוגמא:

```
skype.TREGraphicObject
Skype4COM.Application
Skype4COM.ApplicationStream
Skype4COM.ApplicationStreamCollection
Skype4COM.Call
Skype4COM.CallChannel
Skype4COM.CallChannelCollection
Skype4COM.CallChannelManager
Skype4COM.CallChannelMessage
Skype4COM.CallCollection
Skype4COM.Chat
Skype4COM.ChatCollection
Skype4COM.ChatMessage
Skype4COM.ChatMessageCollection
Skype4COM.Client
Skype4COM.Command
Skype4COM.Conference
Skype4COM.ConferenceCollection
Skype4COM.Conversion
Skype4COM.Group
Skype4COM.GroupCollection
Skype4COM.IEProtocolHandler
Skype4COM.Participant
Skype4COM.ParticipantCollection
Skype4COM.PluginEvent
Skype4COM.PluginMenuItem
Skype4COM.Profile
Skype4COM.Settings
Skype4COM.Skype
Skype4COM.SmsChunk
Skype4COM.SmsChunkCollection
Skype4COM.SmsMessage
Skype4COM.SmsMessageCollection
Skype4COM.SmsTarget
Skype4COM.SmsTargetCollection
Skype4COM.User
Skype4COM.UserCollection
Skype4COM.Voicemail
Skype4COM.VoicemailCollection
```

VBScript? לעבוד מול Skype? מי היה מאמין... ניתן למצוא ולעבוד אל מול מגוון רחב של COM Interfaces, חלקם מגיעים מובנים במערכת ההפעלה וחלקם מותקנים עם תוכנות.

WSF

טוב אז כאן אני פחות אפרט, כי אין כל כך על מה לפרט. אבל צריך לציין שזה קיים כי זה פשוט מגניב. Windows Script File מאפשר לך להריץ כמה שפות במקביל כאילו הן שפה אחת. כמו שכתבתי בתחילת המאמר, ניתן לכתוב פונקציה ב-VBScript ולגשת אליה ב-Python.

ויקיפדיה כבר הכינו דוגמא, אז מי אני שלא אשתמש בה?

```
<?xml version="1.0" ?>
<!-- Mixing JScript and VBScript -->
 <job id="SORT-VBScriptWithJScript">
   <script language="JScript">
     function SortVBArray(arrVBArray) {return arrVBArray.toArray().sort();}
   </script>
   <script language="VBScript">
     <![CDATA[
     '** Fastest sort: call the Jscript sort from VBScript
     myData = "a,b,c,1,2,3,X,Y,Z,p,d,q"
     wscript.echo "Original List of values: " & vbTab & myData
     starttime = timer()
     sortedArray = SortVBArray(split(myData,","))
     endtime=timer()
     jscriptTime = round(endtime-starttime,2)
     wscript.echo "JScript sorted in " & jscriptTime & " seconds: " & vbTab & sortedArray
     ]]>
   </script>
 </job>
```

וכמובן, הם לא שכחו לצרף פלט כדי שלא תחשדו שמותחים אתכם:

```
Original List of values:        a,b,c,1,2,3,X,Y,Z,p,d,q
JScript sorted in 0 seconds:    1,2,3,X,Y,Z,a,b,c,d,p,q
```

[מקור: https://en.wikipedia.org/wiki/Windows_Script_File]

WMI

אז... WMI! קיצור של Windows Management Instrumentation. זהו ממשק שמאפשר לך לנהל את המחשב, לקבל עליו מידע, לעדכן מידע ואף להריץ קוד. לא רק על המחשב שלך, אלא גם על מחשבים אחרים ברשת! המנגנון למעשה מנגיש לך את המחשב בצורה של אובייקטים ומחלקות שמחולקים לפי Namespaces שונים. לדוגמא, תוכלו למצוא תחת ה-root\cimv2 Namespace את המחלקה Win32_Process שאם תריצו שאילתה על המחלקה (WMI Query Language - WQL) תקבלו את כל ה-Instances של המחלקה עם הרבה פרמטרים על כל Process. דוגמא נוספת, תוכלו למצוא ב-Namespace root\SecurityCenter2 מחלקות בשם FirewallProduct, AntiVirusProduct, AntiSpywareProduct שה-Instances שלהן יכיל מידע על מוצרי האבטחה שמותקנים על המחשב.

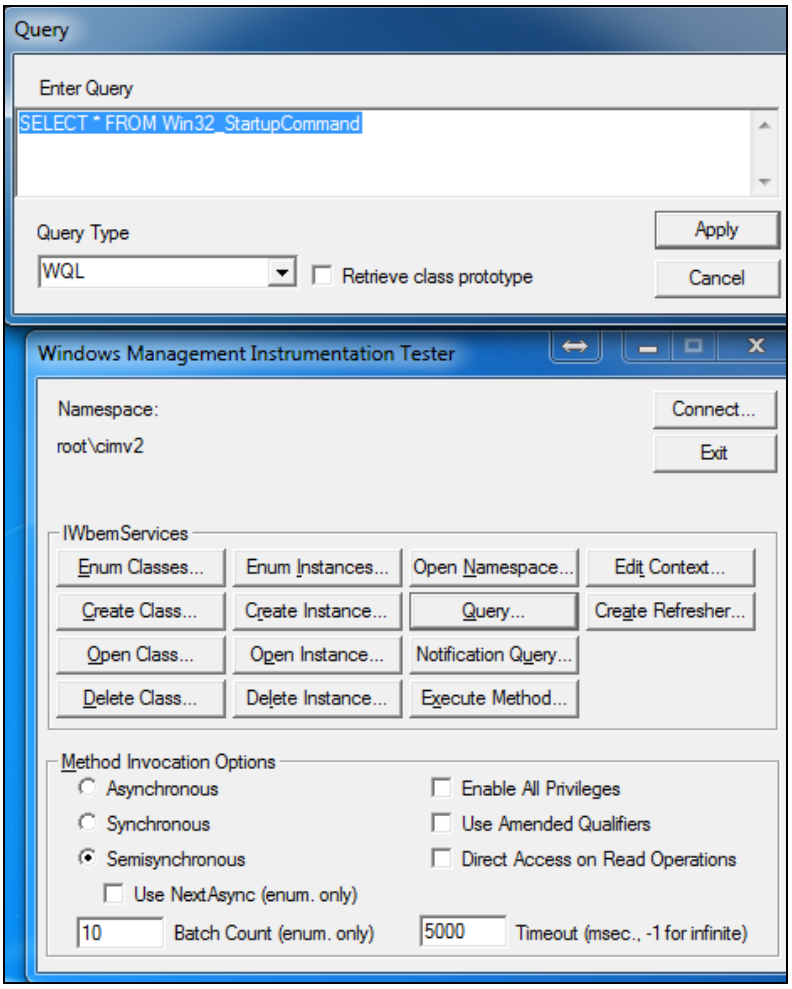
אז יש לנו די הרבה מחלקות שנגישות לנו ויכולות לספק לנו המון מידע! לא אציג פה את כולן, בשביל זה יש לכם תוכנות כמו WMI Explorer. אנו נתרכז במחלקות שנמצאות ב-Namespace שנקרא root\cimv2, זהו ה-Namespace הדיפולטי שם יש מחלקות רבות שמנגישות מידע רב על מערכת ההפעלה, חומרה וכו'... מחלקות שימושיות:

- **Win32_Process** - מחלקה המכילה מידע על תהליכים במחשב.
- **Win32_LogicalDisk** - מחלקה המכילה מידע על כוננים.
- **Win32_ComputerSystem** - מחלקה המכילה מידע על המחשב כמו שם מחשב, דומיין, דגם וכו'...

- **Win32_Service** - מכיל מידע על כל ה-services הקיימים במחשב.
- **Win32_StartupCommand** - מכיל מידע על תוכנות שעולות כשהמחשב עולה.

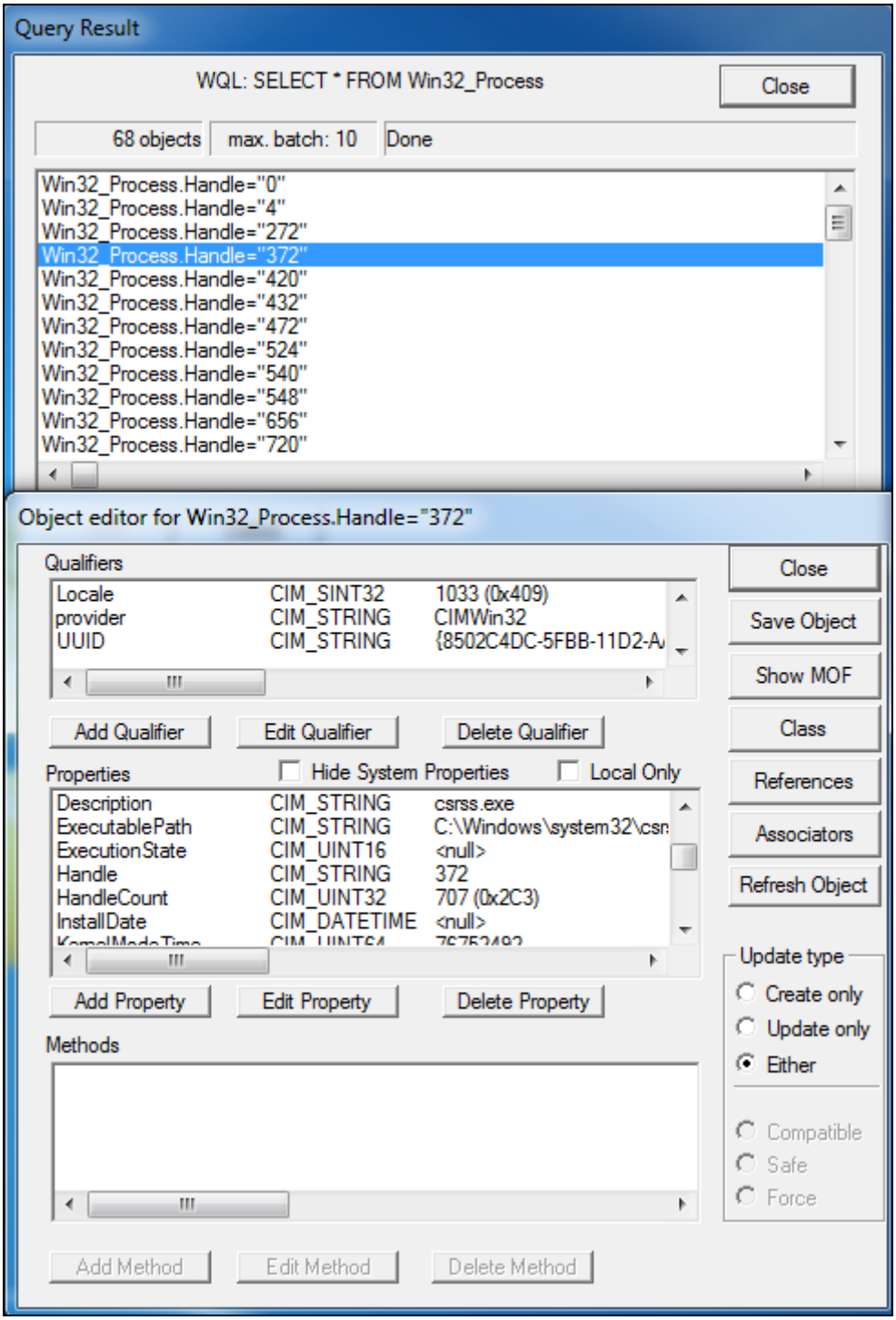
כמובן שהרשימה נמשכת ונמשכת ונמשכת ונמשכת ונמשכת...

ל-Windows יש ממשק מובנה כדי לעבוד עם WMI שנקרא wbemtest והוא נראה ככה (כשחלון הרצת שאילתה פתוח):



כאשר נתחבר ל-Namespace הרלוונטי במחשב כלשהו, נוכל להשתמש בממשק כדי לתשאל אובייקטים (Query), לנהל מחלקות, Instances ועוד...

דוגמא למידע מ-Win32_Process עבור Instance של csrss.exe:



WMI מציעה לכם דבר נחמד ויפה שנקרא Notification Query שמאפשר לכם לקבל "התראה", ברגע שמתרחש משהו. ניתן להאזין על אירועים שמתרחשים על סמך מחלקות שונות כשקיימים 4 טריגרים שונים: Creation, Deletion, Modification, Operation. לדוגמא, ניתן להגדיר Notification Query של Creation על Win32_Process כששם התהליך הוא notepad.exe. ככה ניתן לקבל "התראה" כש-notepad.exe נפתח:

```
SELECT * FROM __InstanceCreationEvent WITHIN 1 WHERE TargetInstance ISA
'Win32_Process' and TargetInstance.Name = 'notepad.exe'
```

זה למעשה מנגנון Events מאוד חזק שמערכת ההפעלה מציעה לנו. אז... מה נעשה איתו? הכירו את MOF! Managed Object Format זהו פורמט לניהול אובייקטים של WMI. בעזרת הפורמט תוכלו ליצור Instances של מחלקות קיימות, ליצור מחלקות, ליצור Namespaces וכו'... בעזרת הפורמט תוכלו לכתוב קובץ שיריץ קוד בהינתן טריגר. מגניב, לא? הנה דוגמא:

```
#pragma namespace("\\\\.\\root\\subscription")

instance of __EventFilter as $EventFilter
{
    EventNamespace = "Root\\Cimv2";
    Name  = "New Process Instance Filter";
    Query = "Select * From __InstanceCreationEvent Within 2"
            "Where TargetInstance Isa \"Win32_Process\" "
            "And Targetinstance.Name = \"notepad.exe\" ";
    QueryLanguage = "WQL";
};

instance of ActiveScriptEventConsumer as $Consumer
{
    Name = "TestConsumer";
    ScriptingEngine = "VBScript";
    ScriptText =
    "Set objFSO = CreateObject(\"Scripting.FileSystemObject\")\n"
    "Set objFile = objFSO.OpenTextFile(\"c:\\log.txt\", 8, True)\n"
    "objFile.WriteLine Time & \" \" & \" Notepad started\"\n"
    "objFile.Close\n";
};

instance of __FilterToConsumerBinding
{
    Consumer   = $Consumer;
    Filter = $EventFilter;
}
```

[מקור: http://www.codeproject.com/Articles/28226/Creating-WMI-Permanent-Event-Subscriptions-Using-M#8.ActiveScriptEventConsumerclass7]

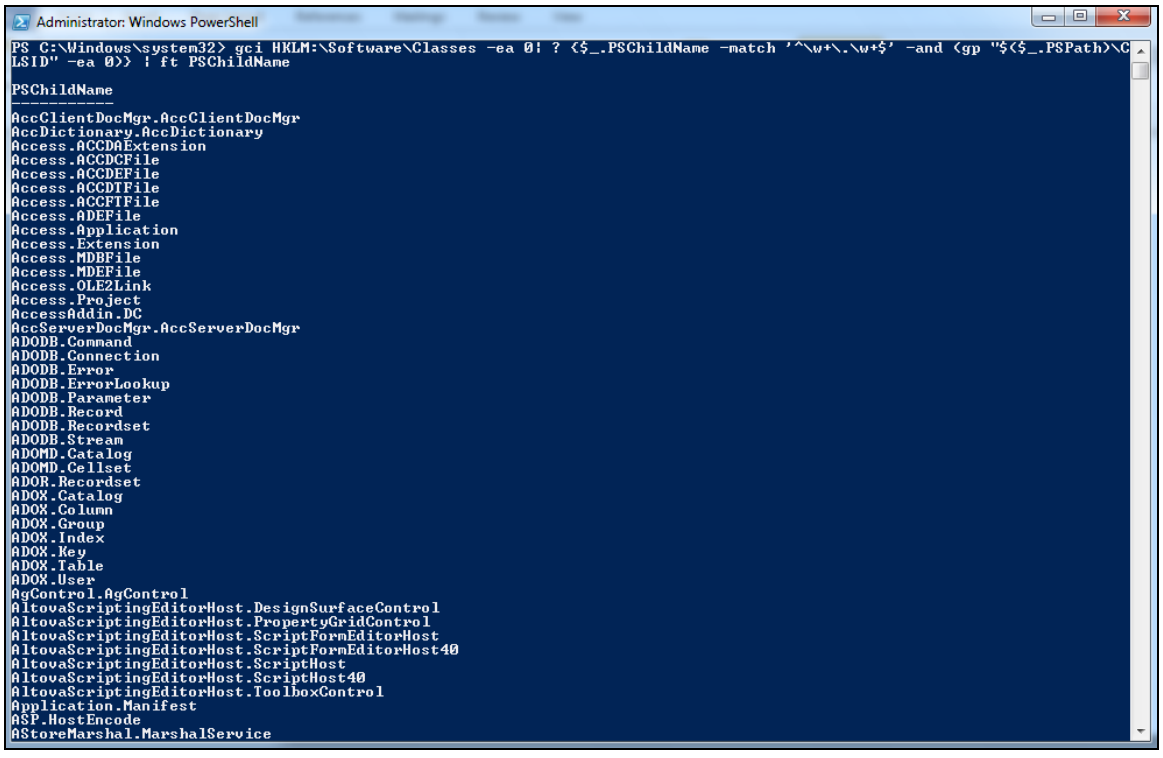
יצרנו מופע של המחלקה __EventFilter שהוא למעשה Notification Query שמאזין על פתיחת Process של notepad.exe. יצרנו מופע של המחלקה ActiveScriptEventConsumer שמאפשר לנו להריץ קוד (ספציפית, VBScript) וקישרנו את ה-Filter ל-Consumer על ידי מופע של המחלקה __FilterToConsumerBinding.

את הקובץ שומרים כקובץ mof ומשתמשים ב-mofcomp.exe כדי להכניס את מה שיצרנו ל- WMI Repository. מגניב! מעכשיו, כל פעם שנפתח notepad, ירוץ הקוד שלנו שירשום את השעה כשנפתח תהליך בשם notepad.exe ☺

PowerShell

טוב אז PowerShell זו ה-שפת סקריפטינג של Windows כרגע. יש לה התממשקות עם .NET., עם WMI, COM ומה לא? השפה נוחה לתפעול, בעלת מגוון רחב של אפשרויות. כמו כן, תוכנות רבות משתמשות בשפה כממשק ניהול. לדוגמא, על שרתי Exchange קיים Console של PowerShell עם cmdlets כדי לנהל את השרת. על השפה עצמה אסביר במאמר הבא, אז בואו נראה כמה דוגמאות כדי לטעום קצת מיכולות השפה ☺

זוכרים את התמונה של ה-COM Interfaces של Skype שהראתי לכם בחלק של VBScript? היא הגיעה מ-Powershell ☺



ה-Oneline הנחמד שמצולם למעלה מתשאל את הרג'יסטרי במקום שנמצאת רשימה של כל ה- COM Interfaces ומבצע סינון לכל מה שרלוונטי להצגה.

בנוסף, יש ל-PowerShell התממשקות ממש טובה עם WMI על ידי Get-WmiObject:

```
PS C:\Windows\system32> Get-WmiObject Win32_Process | ForEach { Write-Host Name: $_.Name `t`t PID: $_.ProcessID }
Name: System Idle Process                PID: 0
Name: System             PID: 4
Name: smss.exe           PID: 272
Name: csrss.exe                  PID: 372
Name: wininit.exe                PID: 420
Name: csrss.exe                  PID: 432
Name: winlogon.exe               PID: 472
Name: services.exe               PID: 524
Name: lsass.exe                  PID: 540
Name: lsm.exe            PID: 548
```

ממש נוח לגשת לפלט של הפקודה בגלל התכונה החזקה של Powershell להנגיש לך את הפלט כרשימה של **אובייקטים**, ככה ניתן לגשת לתכונות שונות בקלות, כמו בדוגמא. את הפלט העברנו ללולאת ForEach וביצענו איטרציה על כל Instance שחזר, שם הדפסנו רק את הפרמטרים Name ו-ProcessID.

כמו כן, תשאול של Event Logs ב-PowerShell הוא די פשוט. ניתן בקלות לתשאל ולפלטר, לדוגמא:

```
PS C:\Windows\system32> Get-EventLog -log System | Where { $_.EventID -eq 1074 } | Measure-Object

Count    : 13
Average  :
Sum      :
Maximum  :
Minimum  :
Property :
```

ככה לדוגמא אנחנו שולפים מה-System Event Logs על EventID 1074 שמייצג כיבוי של המחשב ולאחר מכן משתמשים ב-Measure-Object כדי לקבל מידע סטטיסטי על מה שחזר.

סיכום

אז התחלתי בלנסות להלהיב אתכם קצת על Windows Scripting ועשינו קצת שיעור היסטוריה. לאחר מכן, הראתי לכם שימושים ודוגמאות של שפות ומנגנונים שונים ב-Windows כדי שנקבל קצת מושג על הפוטנציאל ולהבין קצת איך הדברים נראים ומרגישים. אני מקווה שתפסתם את הכוח הגדול שיש בשפות והמנגנונים ש-Windows מציעה לכם. במאמר הבא נחפור קצת יותר ב-PowerShell, נבין את השפה, נלמד איך לעבוד איתה ונכיר את ה-Syntax של השפה. נכיר cmdlets נפוצים וטריקים נחמדים. בקיצור, נראה כמה חזקה השפה.

מקווה שנהנתם מהקריאה!

ליצירת קשר ניתן לפנות ב: neter.nir@gmail.com.