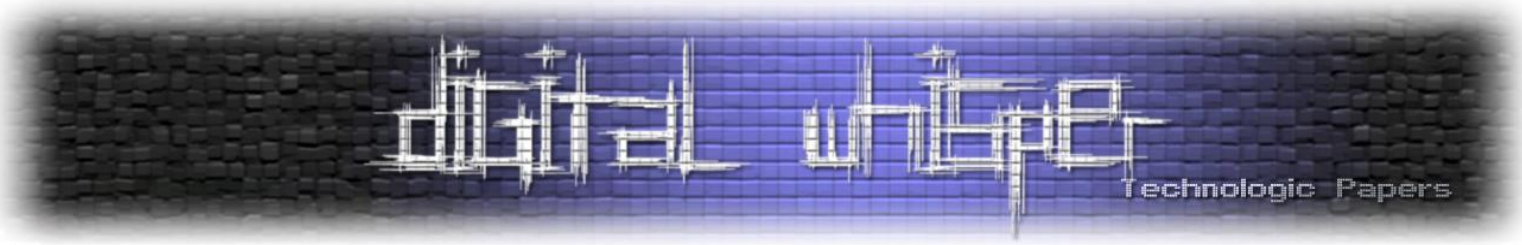


Shortcut Hotkey Exploitation

מאת עידו ולצמן

הקדמה

במאמר זה אציג טכניקת פרסיסטנטיות למערכת ההפעלה Windows המבוססת על מנגנון קיצורי-הדרך של מערכת ההפעלה. אראה כיצד היא עובדת וכיצד ניתן להביא אותה לידי מימוש. כמו כן, בסוף המאמר יינתן הפתרון שלי כיצד ניתן לזהות את השיטה בתור חוקר וכיצד יהיה ניתן לנצל את השיטה כדי להקשות יותר על חיי החוקר בתור תוקף.

המחשב שעליו נעשתה ההדגמה במהלך המאמר הוא Windows 10 version 2004. לצורך המאמר, נדרשת הכרה של הנושאים הבאים:

- Python
- Wininternals
- כלים מ-Sysinternals

המחקר

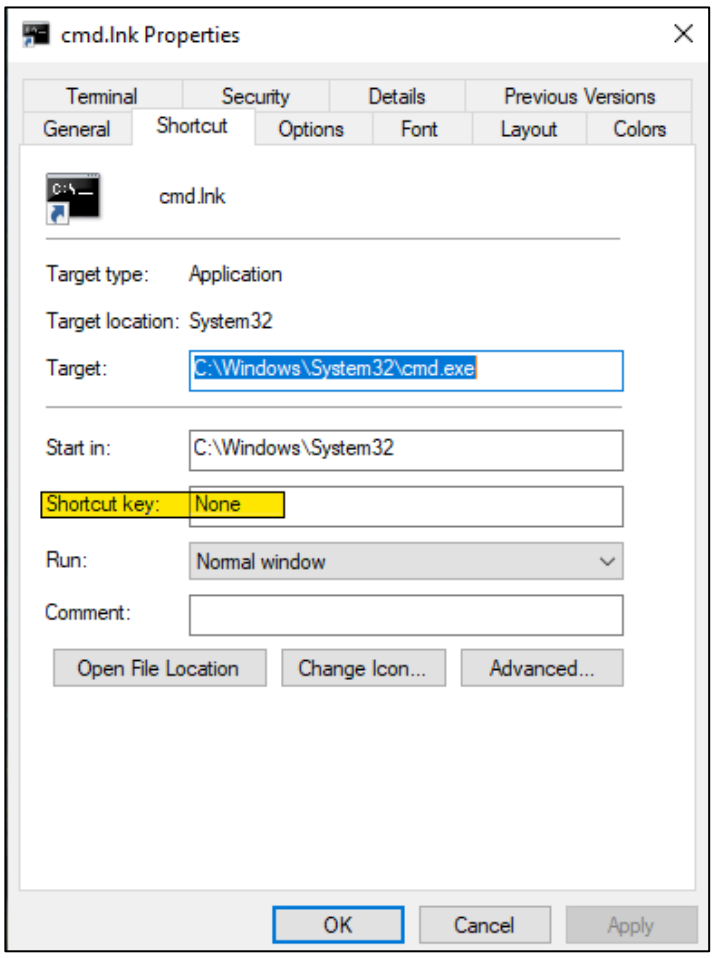
הכל התחיל מזה שקראתי מאמרים על חולשות שהתגלו בקבצי Shortcut שניתן לעשות איתן דברים שונים - בין אם זה הרצת קוד, תקשורת וכדומה. החולשות שהתגלו סקרנו אותי וגרמו לי להעמיק בנושא וללמוד יותר על מהו קובץ Shortcut ומהן תכונותיו.

קובץ Shortcut לפי ההגדרה (fileinfo.com):

“A LNK file is a Shortcut or "link" used by Windows as a reference to an original file, folder"

בתרגום חופשי - קובץ Shortcut הוא משמש כ"מצביע" עבור קובץ / תיקייה, כלומר: בעת הרצתו נריץ את הקובץ שאליו הוא מצביע.

לצורך הלימוד, יצרתי קובץ Shortcut שמצביע ל-CMD:

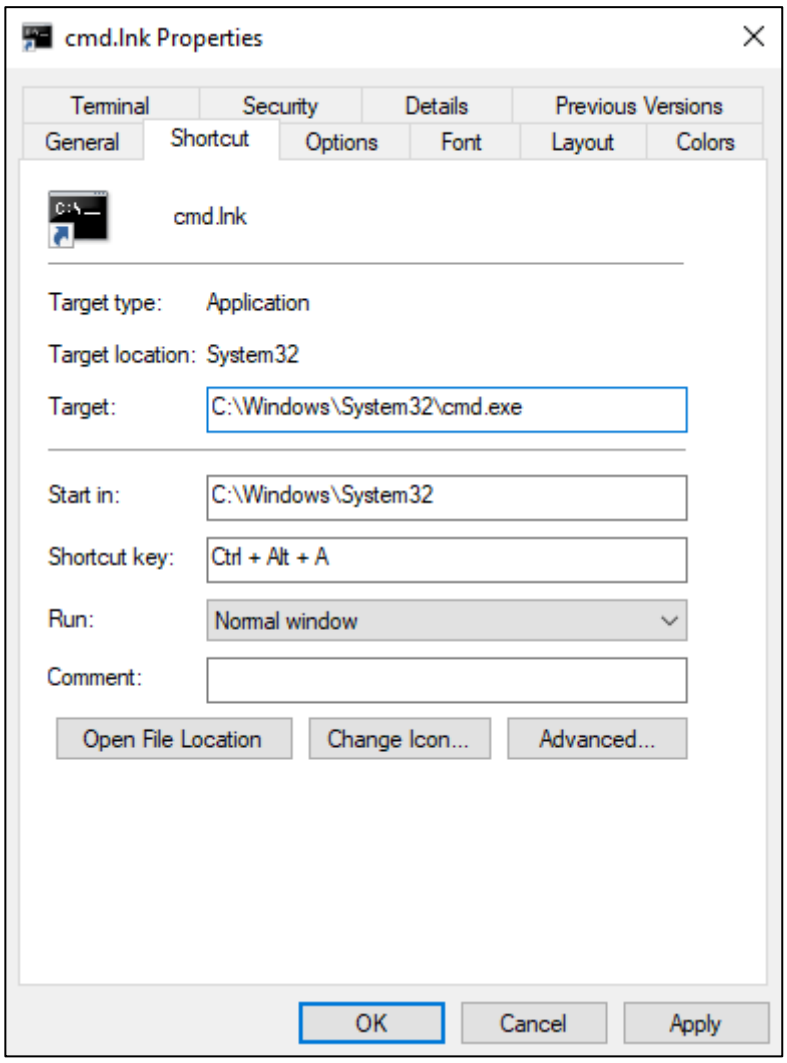


נסתכל על התכונות שנתונות לקובץ:

- שם הקובץ - שם קובץ ה-Shortcut.
- Target - הקובץ / תיקייה אליו הוא מצביע.
- Run - כיצד הקובץ ירוץ: תצוגה מוגדלת / מוקטנת / רגילה.
- Comment - תיאור הקובץ.
- Shortcut key - סדרת מקשים נתונה שבעת לחיצתם ה-Shortcut ירוץ.

כמובן שניתן לערוך את התכונות וישנן עוד תכונות אבל התכונה שמעניינת אותנו היא Shortcut key כיוון שכל פעם שמשתמש ילחץ על מקש מסוים הוא יריץ את התוכנה שלנו, אך כאשר מנסים להגדיר Shortcut key ל-Shortcut, נתקלים בבעיה הבאה: עבור כל תו שניתן יתווספו התווים Ctrl + Alt. הסיכויים שמשתמש ילחץ על התווים Ctrl + Alt + <Letter> היא שואפת לאפס ולכן זה לא יעיל.

במאמר אראה איך לעקוף את המשוחה הזו כך שיהיה ניתן להשתמש בתו בודד כ-hotkey במקום + <Letter> Ctrl + Alt:



[דוגמא לקובץ Shortcut עם Shortcut key]

לצורך מימוש השיטה, נשתמש במודול winshell[2] שמספק לנו Wrapper נוח עבור התממשקות מול ה-Windows Shell.

קצת על המודול עצמו: המודול (winshell) הוא נגזרת של pywin32 (מודול של Python שמאפשר התממשקות מול מ"ה Windows) ונותן כמה פונקציונליות בהן:

- גישה לתיקיות מיוחדות - Desktop, Startup, AppData, StartMenu ועוד.
- ביצוע פעולות על קבצים - מחיקה, העברה, שינוי שם והעתקה.
- קבלת metadata על קובץ.
- פח האשפה - שחזור קבצים, קבלת מידע על קבצים בפח האשפה.
- Shortcuts - יצירת קבצי Shortcut.

לצורך המימוש, נשתמש בעיקר בפונקציונליות של יצירת קבצי Shortcut. כאמור לעיל, התכונות שנצטרך בשביל לממש את החולשה הן:

- בעת יצירת instance, נצטרך לתת את המיקום ל-Shortcut.
- Path - מיקום הקובץ אותו נרצה להריץ.
- Description - ה-Comment (הסבר קצר על ה-Shortcut).
- Shortcut key - Hotkey, ערך ה-ASCII של התו במקלדת. יש לשים לב כי עבור אותיות (Z-A) יש לתת את ערך ה-ASCII של האות ב-capital letter.

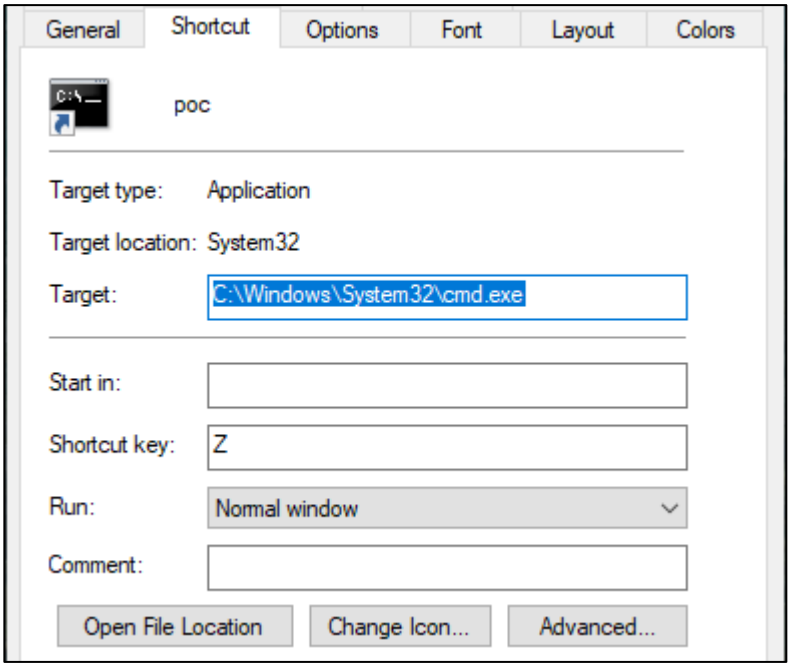
```
# Imports
from os.path import join
import winshell

#Constants
HOTKEY = 90

def main():
    # Creating the Shortcut on my desktop.
    Shortcut = winshell.Shortcut(join(winshell.desktop(), "poc.lnk"))

    # Setting my target path.
    Shortcut.path = r"C:\Windows\System32\cmd.exe"

    # Adding description.
    Shortcut.hotkey = HOTKEY

    # Creating the Shortcut.
    Shortcut.write()

if __name__ == "__main__":
    main()
```

[דוגמה לקוד פשוט שיוצר Shortcut[3]]

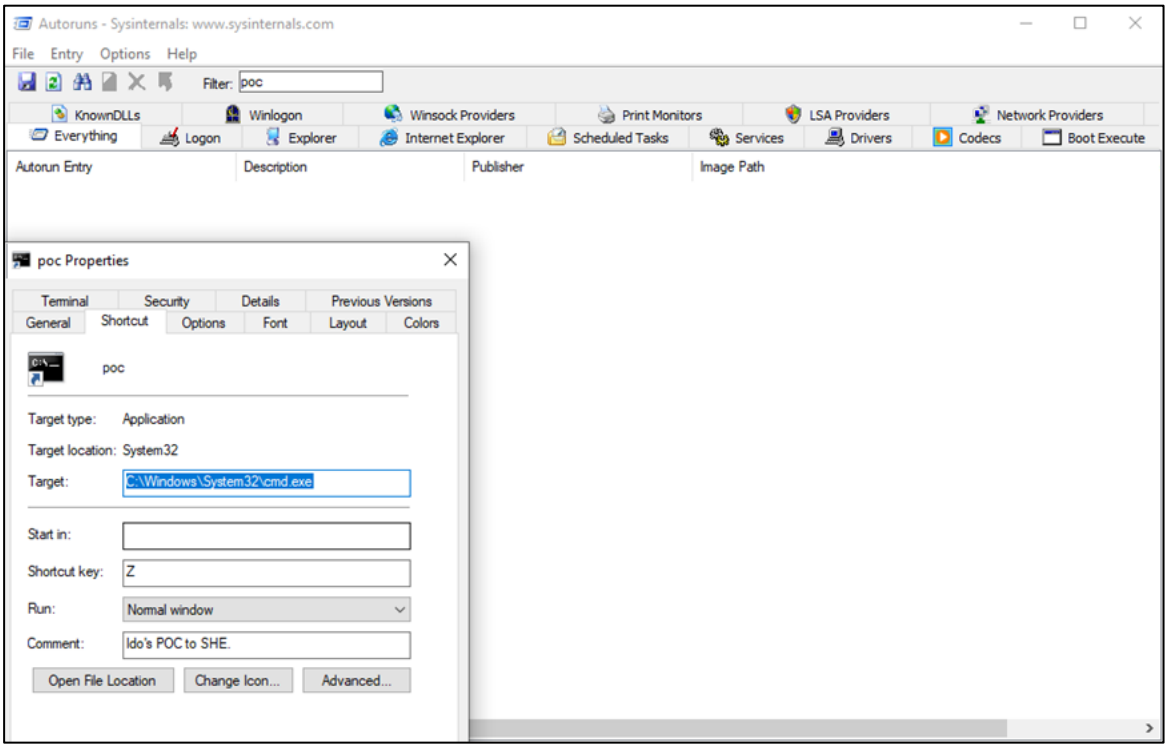
```
def press_letter(letter):
    # Saving the previous clipboard value.
    prev_clipboard = clipboard.paste()
    clipboard.copy(letter)

    # Pressing Ctrl + V.
    control = keyboard.Controller()
    control.press(keyboard.Key.ctrl)
    control.press("v")
    control.release("v")
    control.release(keyboard.Key.ctrl)

    # The print block is there because else python would run first
    # the row below and not the ctrl + v as we want.
    print ""

    # Restoring clipboard state.
    clipboard.copy(prev_clipboard)
```

[דוגמה לקוד שמדמה לחיצה על המקש[6]]

דוגמא לתוצאה

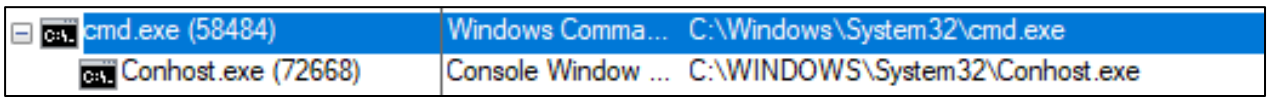


כמובן שנוכל לתת תכונות נוספות כגון פרמטרים, מיקום של icon ועוד - אבל התכונות שצוינו לעיל מספיקות עבור מימוש השיטה. הרצתי autoruns בשביל לראות האם ה-Shortcut מזוהה אך הוא לא זוהה.



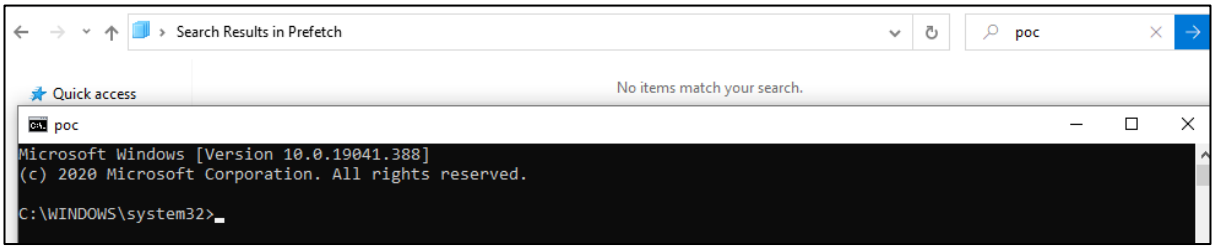
[פילטרתי עבור poc (השם של קובץ ה-Shortcut)]

רציתי לבדוק אם אני מריץ את הקובץ מהו שם הקובץ שנתפס כרץ ואלו התוצאות מהרצה של procexp ו-procmon:



[הרצה של procmon - גם כאן לא מצויין שם / נתיב של קובץ ה-Shortcut]

כמו כן, ניסיתי לראות האם הקובץ מופיע ב-Prefetch-ים וב-Registry:



[בדיקה של prefetch-ים]

כפי שניתן לראות - קובץ ה-Shortcut לא מזוהה בשום שלב באמצעות autorun, procexp, procmon ולא נמצא ב-registry או בתיקיית ה-Prefetch-ים. דבר זה קורה בגלל שכמו שצוין בהתחלה, קובץ Shortcut הוא רק מצביע לקובץ הרצה. זאת אומרת, שאנחנו נוכל להריץ דברים בלי שגם המשתמש הפשוט וגם חוקר בעת חקירה של העמדה יוכלו לראות - אלא אם כן הם יכירו את השיטה לפני כן.

מאחורי הקלעים

השיטה עובדת מצוין ומספקת תוצאות נהדרות, אבל מה קורה מאחורי הקלעים? לשם כך, הסתכלתי על המודול winshell וחיפשתי את המקום בו מוגדר הערך hotkey - הערך שגורם לכך שבהקשה על המקלדת הקובץ יורץ:

```
def _get_hotkey (self):
  return self._shell_link.GetHotkey ()
def _set_hotkey (self, hotkey):
  self._shell_link.SetHotkey (hotkey)
hotkey = property (_get_hotkey, _set_hotkey)
```

['הגדרה של הערך 'hotkey]

לאחר שראיתי מהו הערך בדקתי איפה מתבצע השימוש בו ומהי הפונקציה שיוצרת את קבצי ה-Shortcut:

```
def __init__ (self, lnk_filepath=None, **kwargs):
  self._shell_link = wrapped (
    pythoncom.CoCreateInstance,
    shell.CLSID_ShellLink,
    None,
    pythoncom.CLSCTX_INPROC_SERVER,
    shell.IID_IShellLink
  )
  self.lnk_filepath = lnk_filepath
  if self.lnk_filepath and os.path.exists (self.lnk_filepath):
    wrapped (
      self._shell_link.QueryInterface,
      pythoncom.IID_IPersistFile
    ).Load (
      self.lnk_filepath
    )
  for k, v in kwargs.items ():
    setattr (self, k, v)
```

[תצלום חלקי של הקוד שרץ כאשר יוצרים Shortcut]

כפי שניתן לראות, הפונקציה הנקראת היא CoCreateInstance[4] מהמודול pythoncom. Pythoncom זה מודול אחד מתוך החבילה הנקראת pywin32 (הוזכרה בתחילת המאמר) שתפקידו להתממשק מול פונקציות API.

הפונקציה CoCreateInstance (כמו שנכתב ב-MSDN) אחראית על יצירה של קובץ ושיוך שלו לסוג מסוים. למה הכוונה? במקרה שלנו, הפונקציה אחראית על אתחול של קובץ וסימונו כקובץ מסוג Shortcut. דוגמה לקריאה לפונקציה CoCreateIntsance בשביל ליצור Shortcut:

```
CoCreateInstance(&CLSID_ShellLink, NULL, CLSCTX_INPROC_SERVER,
&IID_IShellLink, &pShellLink);
```

- CLSID_ShellLink - ערך מוגדר שמייצג את ה-CLSID (מזהה מיוחד עבור כל COM Object).
- CLSCTX_INPROC_SERVER - ערך שמציין כי הקוד שנריץ ירוץ ב-context של ה-process הנוכחי.
- IID_IShellLink - ערך מוגדר שמייצג Interface עבור ה-ShellLink.
- pShellLink - מצביע ל-Shortcut שנוצר.

שימושים אפשריים

- אפשר, כמו שהודגם במאמר, לתת את ה-Target של הקובץ כקובץ הרצה רגיל.
- כמובן שניתן גם לתת DLL ולהריץ אותו עם rundll32.
- אם רוצים להיות מתוחכמים יותר ולקחת את השיטה צעד אחד קדימה אפשר לעשות פקודה ב-PowerShell שאותה ה-Shortcut יריץ כל פעם שתתבצע קריאה אליו. אפשר גם כמובן לתת פקודות ב-WMI, CMD.

אומדן יכולות וחולשות של השיטה

- **הרשאות** - הרשאות של משתמש רגיל, התוכנה רצה תחת המשתמש ולכן השיטה יכולה לשמש גם כ-Privilege Escalation.
- **רעש** - השיטה לא רועשת בכלל, כפי שניתן לראות מן הדוגמאות לא procmon ולא procexp זיהו שמקור ההרצה הוא קובץ הShortcut.
- **תועלת** - השיטה מביאה תועלת רבה ונותנת הרצת קוד ע"י המשתמש בלי ידיעתו ושרידות חזקה מאוד שקשה מאוד לגלות אותה.
- **חולשה** - הדבר היחיד שיש לשים לב הוא שבעת לחיצה על המקש לא נכתב התו המבוקש ולכן התוכנה שמשתמשת בשיטה הזו צריכה באופן עצמאי לכתוב את התו בכדי שהמשתמש לא יחשוד.

הפתרון

נכון למתי שפורסם המאמר, השיטה לא מוכרת אבל מאוד סביר שתוקפים בעתיד ישתמשו בה. בגלל שאין אפשרות לבטל את ה-"feature", אני ממליץ שבעת הגעה לעמדה נגועה לזכור את השיטה ואולי גם להכין סורק[5] שמחפש באופן אקטיבי קבצי Shortcut בעלי hotkey.

סיכום

היה לי חשוב במאמר להציף את השיטה שהיא לא מוכרת ויכולה לתרום רבות ליצירת אחיזות על העמדה ולהקשות את החיים של החוקר כאשר הוא בא לעמדה. מה שאני אוהב בשיטה הזאת שהיא נותנת הרבה אפשרויות וביסוס נהדר מבלי הרשאות מיוחדות או הבנה עמוקה של מ"ה.

אני לא יודע אם Microsoft ישנו את ה-"feature" הזה כך שלא יהיה אפשר לתת רק מקש בודד ומהו הפתרון ההגנתי שיינתן לשיטה, אני יודע שאין ספק שהחל מהרגע שהשיטה הזאת תקבל את הפרסום שמגיע לה היא תשנה את המשחק.

ביבליוגרפיה

[1] קישור לבלוג שהשיטה צוינה לראשונה:

http://www.hexacorn.com/blog/2015/03/13/beyond-good-ol-run-key-part-29/

[2] קישור לדף pip של המודול winshell:

https://pypi.org/project/winshell/

[3] קישור לדף ה-Github שלי ל-POC לשיטה:

https://github.com/Idov31/hotkey_exploitation/blob/master/poc.py

[4] קישור להסבר על הפונקציה CoCreateInstance:

https://docs.microsoft.com/en-us/windows/win32/api/combaseapi/nf-combaseapi-cocreateinstance

[5] קישור לדף ה-Github שלי עם ה-scanner שנועד למצוא את השיטה:

https://github.com/Idov31/hotkey_exploitation/blob/master/scanner.py

[6] קישור לדף ה-Github שלי לקוד שמדמה לחיצה על מקש מסויים:

https://github.com/Idov31/hotkey_exploitation/blob/master/listener.py