<u>שאלה 1 - חלק א' - הסבר על חור האבטחה</u>

נסביר את התהליך אותו מבצע הservice המותקף ולאחר מכן נסביר היכן נמצא חור האבטחה:

- בT1, הservice מייצר קובץ tmp בעל שם ייחודי, בגודל 0 בתים.
- בT2, הservice מייצר handle לקובץ חדש עם שם זהה לקובץ שנוצר בT1.
- בT3, הservice מבצע כתיבה לקובץ בעזרת הhandle, ובתום הכתיבה מבצע סגירה של הhandle.
- בT4, הservice יוצר process חדש המריץ את הקובץ שיצר.

<u>חור האבטחה:</u>

בטווח הזמן שבין סגירת הhandle של הקובץ (סוף T3) לבין יצירת התהליך החדש (T4) שיריץ קובץ זה נוכל להחליף את הקובץ שנוצר בתוכנה זדונית, ובכך לגרום לservice שרץ בהרשאות גבוהות להריץ את התוכנה הזדונית בהרשאות גבוהות גם כן.

<u>הרחבה:</u>

נדגיש כי בחרנו לתקוף (להחליף את תוכן קובץ ההרצה) בשלב שבין T3 לT4 משום ש:

- אילו היינו תוקפים בין T1 לT2 אזי הservice היה דורס את הקובץ אותו שינינו, בגלל ה flag - create always, המבצע דריסה של קובץ במידה והוא כבר קיים.
- לא ניתן לתקוף בין T2 לT3 שכן בשלב זה הhandle פתוח תחת share mode =0, כלומר אינו מאפשר לתהליכים אחרים לפתוח, למחוק, לקרוא או לכתוב לקובץ.

```
uRetVal = GetTempFileName(TEXT(TEMP_DIR_NAME), // directory for tmp files
    TEXT(TEMP_FILE_TEMPLATE),     // temp file name prefix
    0,                  // create unique name
    szTempFileName);  // buffer for name
if (uRetVal == 0){ ... }
```

T1

```
/* Creates the new file to write to for the upper-case version. */
hTempFile = CreateFile((LPTSTR)szTempFileName, // file name
    GENERIC_WRITE,        // open for write
    0,                    // do not share
    NULL,                 // default security
    CREATE_ALWAYS,        // overwrite existing
    FILE_ATTRIBUTE_NORMAL,// normal file
    NULL);                // no template
if (hTempFile == INVALID_HANDLE_VALUE){ ... }
```

T2

```
/* Writes the resource */
fSuccess = WriteFile(hTempFile, pResourceData, cbResourceSize, &dwBytesWritten, NULL);
if ((!fSuccess) || (dwBytesWritten != cbResourceSize)){ ... }

FlushFileBuffers(hTempFile);
CloseHandle(hTempFile);
```

T3

```
//Now we'll execute the resource file
fSuccess = CreateProcess(NULL, szTempFileName,
    NULL, NULL, NULL,
    0, NULL, NULL, &si, &pi);
if (!fSuccess){ ... }
```

T4

<u>שאלה 1 - חלק ב' - פתרון חור האבטחה</u>

<u>אפשרות א':</u>

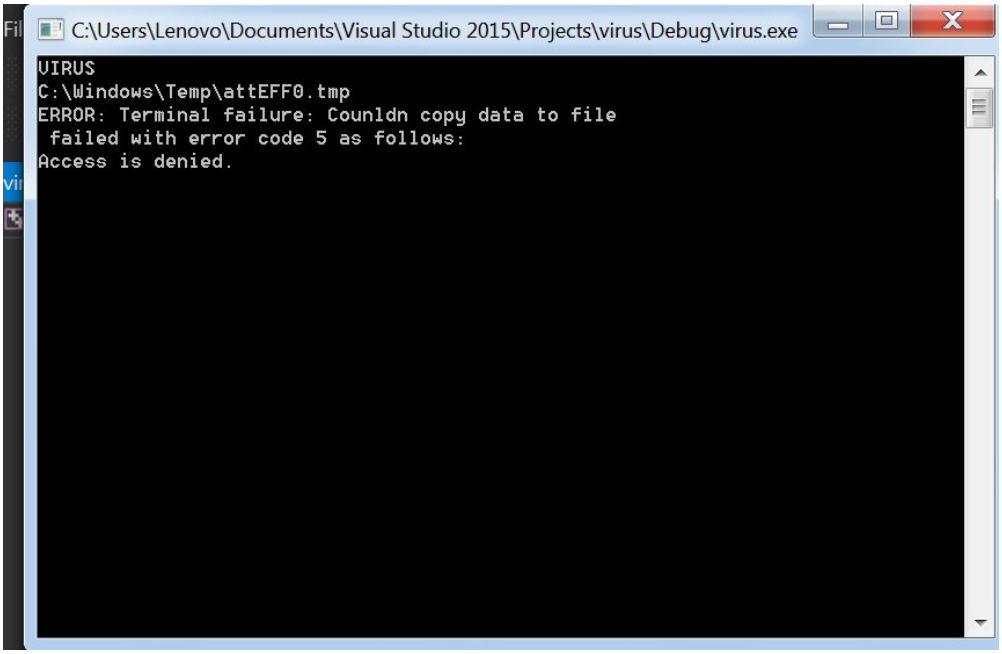
- סדר הפעולות למניעת חולשה זו:
  - קבלת נתיב לתיקיה זמנית תחת הרשאות של התהליך הנוכחי שמריץ את הקוד - בעזרת GetTempPath
  - יצירת הקובץ הזמני שהתהליך צריך בנתיב שהתקבל לעיל - כאשר שם הקובץ מתקבל מ-GetTempFileName
- מאחר והתהליך שלנו רץ בהרשאות גבוהות, שימוש בפונקצייה GetTempPath תייצר נתיב לתיקייה שהיא גם בעלת הרשאות גבוהות - למשל C:\Windows. בשלב השני יצירת הקובץ בתוך התיקייה הנ"ל תייצר קובץ שגם הוא עם הרשאות גבוהות שכן קובץ יורש את ההרשאות של התיקייה בו הוא נמצא ולכן משתמש "תמים" בעל הרשאות נמוכות יותר לא יוכל לכתוב\לדרוס\למחוק קבצים אלה.

להלן שינוי הקוד המתבקש:

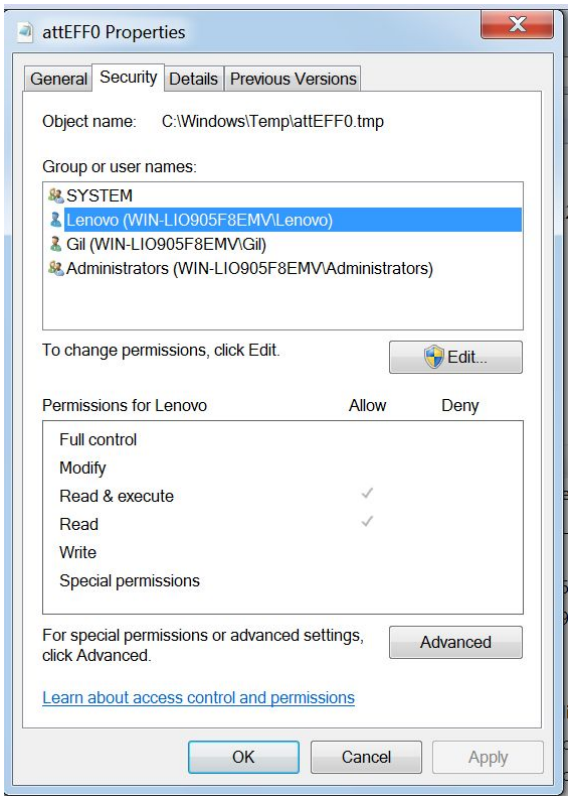
```
//  Gets the temp path env string (no guarantee it's a valid path).
dwRetVal = GetTempPath(MAX_PATH,          // length of the buffer
        lpTempPathBuffer); // buffer for path
if (dwRetVal > MAX_PATH || (dwRetVal == 0)) {
        PrintError(TEXT("GetTempPath failed"));
        if (!CloseHandle(hFile)) {
                PrintError(TEXT("CloseHandle(hFile) failed"));
                return NULL;
        }
        return NULL;
}

//  Generates a temporary file name.
uRetVal = GetTempFileName(lpTempPathBuffer,                     // directory for tmp files
        TEXT(TEMP_FILE_TEMPLATE),     // temp file name prefix
        0,                                  // create unique name
        szTempFileName);            // buffer for name
if (uRetVal == 0) {
        PrintError(TEXT("GetTempFileName failed"));
        if (!CloseHandle(hFile)) {
                PrintError(TEXT("CloseHandle(hFile) failed"));
                return NULL;
        }
        return NULL;
}
```

נדגים את התהליך:

- הפעלנו את הvirus שלנו.
- הפעלנו את הservice המתוקן, להלן הוירוס שלנו מקבל error - access denied (ניתן לראות כי הוירוס מזהה ומנסה לדרוס את הקובץ הנכון).



- ניתן לראות בsecurity table כי ל-Lenovo User ישנן רק הרשאות ריצה וקריאה.



<u>אפשרות ב':</u>

ניתן לשנות את הקוד שרץ בservice כך שייצר קובץ בעל הרשאות ספציפיות, ובפרט הרשאות מלאות אך ורק ל system (המשתמש הנוכחי הוא SYSTEM כי אנו מריצים את ה- SERVICE עם PSEXEC).
הגדרת הרשאות כנ"ל מתבצע ע"י בניית ה- STRUCT, SECURITY_ATTRIBUTE המסופק לפונ' CreateFile.

כיצד נבנה את הsecurity attributes הנ"ל?

1. נחלץ את הSID של התהליך הנוכחי מתוך ה - TOKEN_USER ,STRUCT (אותו נחלץ ע"י הפונ' GetCurrentProcess ו- OpenProcessToken).
2. נייצר ACE (access control entry) - המכיל את הSID שקיבלנו קודם לכן (שימוש ב- STRUCT, EXPLICIT_ACCESS).
3. נייצר ACL המכיל את הACE שקיבלנו מ2 (PACL ,STRUCT).
4. נכתוב את הACL מ3 לתוך security descriptor. שימוש ב STRUCT PSECURITY_DESCRIPTOR.
5. נכניס את הsecurity descriptor מ4 לתוך הsecurity attributes.

לאחר שלב זה נקבל security attributes כנדרש וה- SERVICE ייצר את הקובץ עמו.

<u>שאלה 2 - חלק ב' - פתרון חור האבטחה</u>

ננסה למצוא פיתרון שיאפשר לשרת להמשיך ולהריץ shellcode אך יחד עם זאת ינסה לאתר קוד זדוני או להבחין בין גורם מעדכן לגיטימי ושאינו כזה, להלן פתרונות אפשריים ומגבלותיהם:

- **Regex\white listing** של פעולות אפשריות:
  - ע"מ למנוע מהשרת מלהריץ קוד זדוני נאפיין תכונות\מבנה לקוד חוקי. כאשר מתבצעת קריאה לשרת, הקוד ייבחן ע"י מנגנון זה, במידה ויעמוד במבנה\תכונות אלו, הקוד יופעל.
  - לדוגמא, נגדיר מבנה shell code תקין, חייב להתחיל ברצף הפקודות mov .. יכיל את הstrings הבאים: "microsoft corporation" וכו'..
  - פתרון זה טוב אך איננו אידיאלי לבדו, שכן תוקף שיכיר את המבנה אותו אנו מקבלים יוכל בקלות לייצר קוד זדוני בעל תכונות זהות ולהביא את השרת להריצו.

- יצירת socket מאובטח ע"י החלפת **certificates** והצפנת המידע - B2B:
  - Secure socket מגדיר תקשורת מוצפנת כך - כאשר user מתחבר לשרת הוא נדרש לספק certificate מתאים שעליו השרת סומך, במידה והcertificate נמצא בtruststore של השרת, השרת שולח למשתמש certificate עם public key.
  - ה-user מצפין את המידע באמצעות הpublic key, ושולח לשרת. המידע שמתקבל בשרת מפוענח באמצעות הprivate key, וכך למעשה אנו מבטיחים שהמידע שקיבלנו הוא מגורם מאושר ומועבר בצורה מוצפנת.
  - גם לפיתרון זה ישנן חולשות רבות, מתקפות כגון man in the middle, או סתם אם תוקף יצליח לגנוב את ה certificate שעליהם אנו סומכים.

- הרצת **תהליך בן עם הרשאות נמוכות יותר**:
  - בפיתרון זה אנו מעוניינים לקבל shell code ולהמירו לקובץ exe.
  - לאחר מכן באמצעות CreateResrictedToken נייצר token לתהליך בעל הרשאות נמוכות יותר.
  - נריץ את קובץ הexe באמצעות CreateProcessAsUser עם הtoken הנ"ל.
  - חשוב לציין שניתן להריץ process עם הגבלות על ההרשאות באמצעות security attributes מתאימים.
  - הבעייתיות בפתרון זה נובעת מהעובדה שאנו מגבילים את יכולות העדכון של השרת בצורה קיצונית, לדוגמא, לא ניתן יהיה להחליף קבצים בתיקיות מערכת ההפעלה.

אם נרצה להשיג פתרון חזק משמעותית, ניתן יהיה לשלב את 3 הפתרונות הנ"ל יחד ולקבל שרת עם אבטחה טובה יותר.