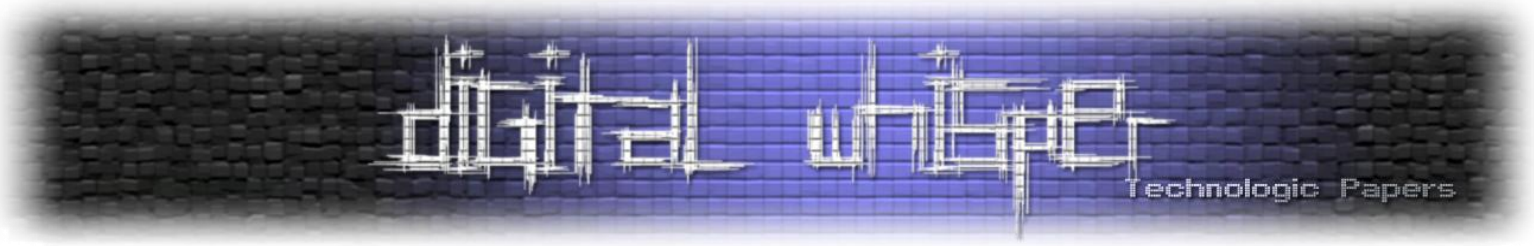



# Userland Rootkits

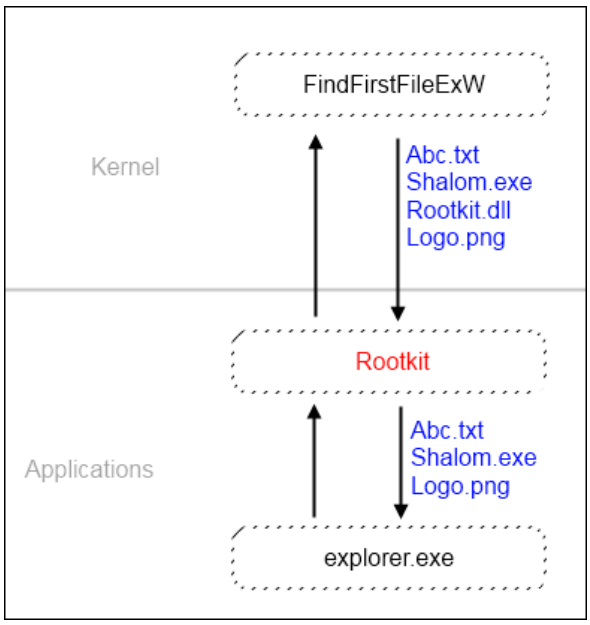
מאת vbCrLf (אורי להב)

## הקדמה

לפני מספר שנים כללה Sony בדיסקי מוזיקה שמכרה תוכנת הגנה מפני העתקה שהותקנה אוטומטית במחשב של הלקוח שקנה את הדיסק. זמן לא רב לאחר מכן מארק רוזינוביץ', יוצר סט הכלים המצוין SysInternals, גילה את התוכנה ופירסם את הממצאים. התוכנה הסתירה את עצמה ואף חשפה את המחשב לבעיות אבטחה. Sony הודיעה על החזרה (recall) של הדיסקים, ובנוסף החברה שפיתחה את ההגנה שחררה כלי להסרת ההגנה. בדיעבד, התברר שכלי התיקון חשף את המחשב לבעיות אבטחה חמורות. עוד יותר באלגן.

חוץ מהעובדה שהיא הותקנה ללא ידיעת המשתמש, הדבר שהיה חמור כל כך בהגנה הזו היה שהיא הייתה **Rootkit**. תוכנה כזו מחבלת בפעולות סטנדרטיות של מערכת ההפעלה על מנת להסתיר את פעילותה. לדוגמה, במקרה של Sony, התוכנה גרמה ל-Windows להתעלם מקבצים ששמם מתחיל ב-"$sys$", וכך הסתירה את עצמה. אגב, אחרי הפרסום הופיעו וירוסים שניצלו את ההזדמנות שתוכנה זו זימנה להם ופשוט קראו לעצמם בשם שמתחיל ב-"$sys$" וכך התחבאו מהמשתמש.

בעזרת השתלטות על פעולתם של מרכיבים בסיסיים במערכת ה-Rootkit משמשת כעין גשר המסנן את הנתונים העוברים דרכו. דוגמה לדבר היא Rootkit המנתבת קריאה לפוקנציות API המבקשות רשימת קבצים (FindFirstFile, FindNextFile) כך שיבוצע קוד שבונה ה-Rootkit כתב. התוכנה הלגיטימית תקרא לפונקצית ה-API, ובמקום להגיע למערכת ההפעלה היא תגיע לקוד של ה-Rootkit, קוד זה יקרא לפונקציה המקורית, יסנן ויוודא שהקבצים או התהליך שהוא רוצה להסתיר לא נמצאים בערך המוחזר מהפונקציה ואז יחזיר את התשובה המסוננת אל התוכנה שקראה לפונקציה. באותה השיטה אפשר לבצע עוד פעולות רבות כמו הסתרת תהליך, גודל תיקיה, וכו'. לעיתים, גם תוכנות לגיטימיות משתמשות בטכניקה זו. הדוגמה הבולטת ביותר היא אנטי-וירוס שמונע מווירוסים לפגוע בו על ידי ביצוע מניפולציות שונות.




**שני סוגי Rootkits: User-Mode ו-Kernel-Mode**

כחלק ממערכת ההגנה והאבטחה של מערכת ההפעלה כיום המעבדים תומכים ב-CPU Modes - מצבי ריצה שונים, אחד ללא הגבלות, ומספר אחרים עם הגבלות מסוימות. קוד מערכת ההפעלה או דרייברים רצים במצב ללא הגבלה, אשר ב-Windows הדבר נקרא "לרוץ ב-Ring0" או ב-"Kernel Mode", לעומת זאת (תוכנות, ואפילו explorer.exe) רצות במצב מוגבל העונה לשם "Ring-3" ב-Windows, או בשם המוכר יותר - User-Mode. מכיוון שזו הגנה ברמת החומרה, עקיפה שלה היא לא קלה אם בכלל אפשרית, אך אפשר לגרום למערכת ההפעלה לתת לנו להפעיל קוד ב-Ring0.

ב-Ring3 אין אפשרות לגשת לאיזורי זיכרון מסוימים השייכים למערכת ההפעלה, אין אפשרות להריץ פקודות מסוימות ובעצם כל דבר שעלול להשפיע ישירות על מצב המחשב כמו תקשורת עם חומרה. רק לקוד שרץ ב-Ring0 יש אפשרות לבצע פעולות אלו. מכיוון שתוכנות מבודדות ברמה כזו, גם אם התוכנה תנסה (אפילו בגלל תקלה) לקרוא או לכתוב בזיכרון המערכת היא לא תוכל, היא תוגבל על ידי המעבד. הגבלה זו, חוץ מאבטחה והגנה מפני תוכנות זדוניות גם תורמת ליציבות המערכת כך שכאשר תוכנה קורסת אין על כך שום השפעה על מערכת ההפעלה, בשונה מדרייבר שכאשר הוא קורס הוא מקריס ביחד איתו את מערכת ההפעלה עם מסך ה-Blue Screen of Death הידוע לשמצה.

מה הקשר ל-Rootkit בכלל? אפשר לבנות Rootkit שירוץ ב-Ring0, ואפשר שירוץ ב-Ring3. Rootkit שרץ ב-Ring0 (או Kernel-Mode Rootkit) רץ בדרך כלל בצורה של דרייבר. מכיוון שיש לו הרשאות מלאות על המערכת הוא יכול להשתלט על פעולתה הבסיסית "הנמוכה" ביותר של מערכת ההפעלה, ולשמש כגשר בין התוכנה שביקשה את המידע (כדוגמת רשימת קבצים) לבין מערכת ההפעלה. הוא נשאר שקוף עבור כל תוכנה שרצה ב-Ring3 מכיוון שאין לה אפשרות לדעת מה קורה מאחורי הקלעים, ומבחינתה מי שענה לבקשה שלה היא מערכת ההפעלה. מכיוון שסוג זה הוא בעצם דרייבר יש צורך בהרשאות מנהל (או פירצת Privilege Escalation) כדי להפעיל אותו. סוג זה קשה יותר לתכנות, וכל תקלה בו עשויה לגרום לקריסת המערכת.

הסוג השני הוא User-Mode Rootkit. מכיוון שאין לו גישה למערכת ההפעלה עצמה, הוא משתלט על תהליך או תהליכים מסוימים ומנתב קריאות לפונקציות API אליו כך שהערך המוחזר (כמו רשימת הקבצים) יהיה בשליטתו. במקום להגיד ל-Windows "אני אטפל ב-API הזה" הוא אומר לתוכנה "אני ה-Windows ואני אמור לטפל ב-API הזה". מכיוון שה-Rootkit רץ ב-Ring3 קל הרבה יותר לגלות אותו, ובמיוחד אם האנטי-וירוס רץ ב-Ring0 שאז אין ל-Rootkit שום דרך להסתיר את עצמו. כתיבת דרייבר זו היא קלה יותר והרבה פחות 'מסוכנת', זאת אומרת, קריסה של Rootkit כזה לא תגרום לקריסת מערכת ההפעלה אלא רק לאותו תהליך שאליו הוא נדבק.

וכרגיל, לא נשאיר הכל בתאוריה :) ננסה לבנות Rootkit בסיסי עבור Windows XP שיסתיר את עצמו. מכיוון שכתיבת Rootkit שרץ ב-Ring0 זה מסובך יותר (על כך אולי במאמר הבא) החלטתי לכתוב User-Mode Rootkit. התכנון הראשוני היה "להתעלק" בעזרת DLL Injection ו-IAT Hooking על explorer.exe ו-cmd.exe ולהפנות את הפונקציות FindFirstFile ו-FileNextFile אל ה-Rootkit.

רצוי מאוד לקרוא את שני המאמרים שקישרתי אליהם ("שולים מוקשים - על שליטה בזמן ריצה" ו-"IAT Hooking") שבהם יש הסבר מפורט על שתי הטכניקות האלו. אבל למי שאין כח לקרוא, נסביר בקצרה מה הם.

**DLL Injection** או הזרקת DLL- כדי להריץ קוד שלנו בתוך תוכנה רצה השיטה הפשוטה ביותר היא שימוש בהזרקת DLL. בשיטה זו אנו כותבים את הקוד שאנו רוצים להריץ בתוכנה השנייה בתוך DLL, ומבקשים ממערכת ההפעלה לטעון את ה-DLL הזה לתוך התוכנה הרצה. ברגע שה-DLL נטען, הקוד שלנו רץ בתוך מרחב הזיכרון של התוכנה השנייה בדיוק כמו הקוד שלה, מה שמאפשר לנו שליטה מלאה על פעולתה.

**IAT Hooking** - כל קובץ תוכנה מכיל רשימה של כל קבצי ה-DLL (ספרית פונקציות) שהתוכנה תרצה להשתמש בהם, ובאילו פונקציות מתוכם תרצה להשתמש ("Imports"). כאשר מפעילים את התוכנה, מערכת ההפעלה טוענת את כל קבצי ה-DLL שהיו ברשימה ומסדרת טבלה עם רשימת הפונקציות

שהתוכנה בקשה ביחד עם הכתובת שבה הפונקציה יושבת. ואז, בכל קריאה לפונקצייה חיצונית, כולל קריאה ל-API של Windows, התוכנה ניגשת לטבלה ולוקחת משם את הכתובת של הפונקצייה. לטבלה זו קוראים Import Address Table או בקיצור, IAT. בזכות השיטה הזו כל מה שאנו צריכים לעשות כדי לגרום לקריאה ל-API להגיע אלינו הוא לשנות את הכתובת בטבלה של פונקציית ה-API אל כתובת הפונקצייה שאנו כתבנו ב-Rootkit.

חזרה לענייננו. התכנון היה לטעון DLL בעזרת הזרקת DLL לתוך explorer.exe ו-cmd.exe שיעשה IAT Hooking ל-FindFirstFile ו-FindNextFile, וכך להשיג שליטה על רשימת הקבצים. אבל הדברים קצת הסתבכו. הבאתי בקצרה את התהליך שעברתי עד ל-Rootkit פועל, כדי ללמד מזה כמה דברים :)

**טעויות ובעיות**

התחלתי על ידי העתקת קוד הזרקת ה-DLL מהמאמר על שליטה בזמן ריצה (גיליון 18), לא היה ממש מה לשנות בקוד. השלב השני כתיבת ה-DLL שהוא ה-Rootkit עצמו. כמו שאמרתי המטרה היא לעשות Hook לפונקציות API, ולכן העתקתי את הקוד ל-IAT Hooking מהמאמר בנושא (גם מגיליון 18) כמעט בשלמותו.

לאחר שהכל מוכן צריך לבחור את פונקציות ה-API שלהם נרצה לעשות Hook כדי שכל קריאה אליהם תעבור קודם בקוד שלנו כדי שנוכל לסנן את התשובה. כאשר רוצים לקבל רשימה של קבצים קוראים בדרך כלל לפונקצייה FindFirstFile עבור הקובץ הראשון ואחר כך FindNextFile כדי לקבל את שאר הקבצים אחד אחד. לכל אחת משתי הפונקציות יש שתי גרסאות - גרסת ANSI וגרסת Unicode. ל-ANSI מוסיפים A בסוף, ול-Unicode מוסיפים W בסוף, לדוגמה - FindFirstFileA או FindFirstFileW. בחרתי לסנן את ארבעת הפונקציות (2 פונקציות כפול 2 גרסאות) לשם השלימות, למרות שרק גרסת ה-Unicode בשימוש אצל cmd.exe ו-explorer.exe. הקוד להוק ב-DLL די חוזר על עצמו, ולכן אראה דוגמה רק הוק לפונקציה אחת:

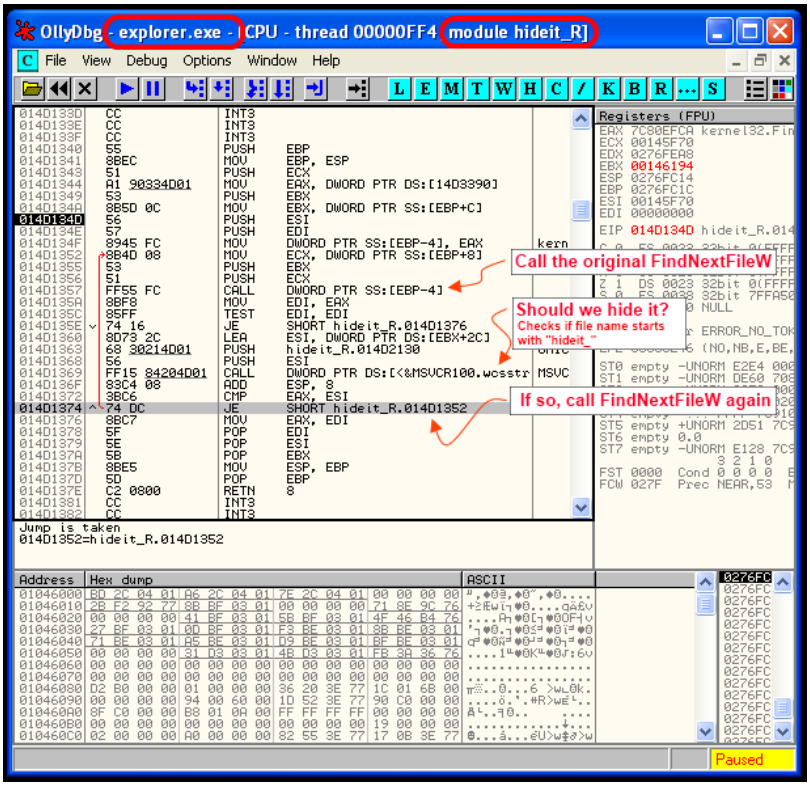
```
HookIAT(GetModuleHandleA(NULL),"FindNextFileW",(DWORD)&newFindNextFileW,
&originalFindNextFileW);
```

זו קריאה לפונקציה שעושה IAT Hooking. **הפרמטר הראשון** אומר לה לעשות Hook "לעצמי" (ה-DLL רץ כבר בתוכנת המטרה מכיוון שהזרקנו אותו לתוכה), **הפרמטר השני** הוא שם הפונקציה שלה אנו רוצים לעשות הוק, **הפרמטר השלישי** הוא כתובת הפונקציה (במקרה שלנו המסננת) שתקרא במקום הפונקצייה FindNextFileW המקורית ולתוך **הפרמטר הרביעי** תכנס הכתובת המקורית של FindNextFileW. מעכשיו כל קריאה ל-FindNextFileW תגיע אל **new**FindNextFileW במקום.

הנה תוכן הפונקציהnewFindNextFileW:

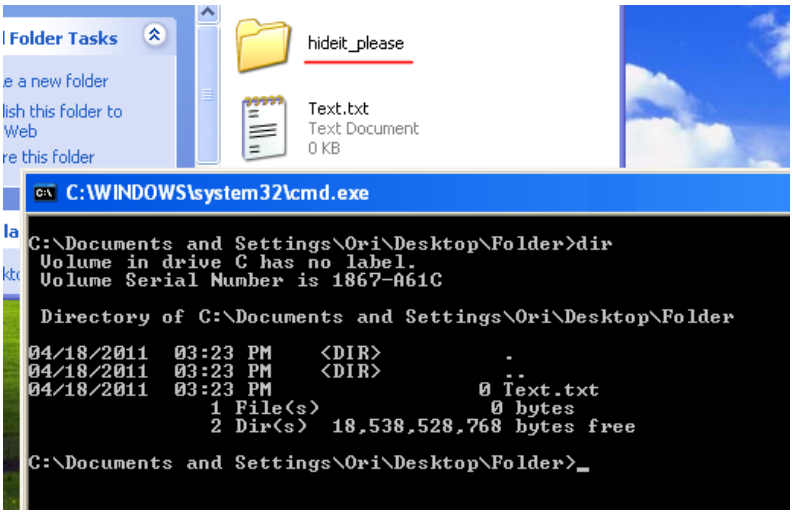
```
BOOL WINAPI newFindNextFileW(__in HANDLE hFindFile, __out LPWIN32_FIND_DATAW
lpFindFileData)
{
    ptrFindNextFileW original = (ptrFindNextFileW)originalFindNextFileW;
    BOOL ret;
    do
    {
        ret = (original)(hFindFile, lpFindFileData);
    } while ((ret != 0) && // While there are more files and ...
      (wcsstr(lpFindFileData->cFileName, HIDEW) == lpFindFileData->cFileName));
    // it's starting with "hideit_"

return ret;
}
```

הקוד די פשוט. הוא מכיל לולאה שממשיכה לבקש את הקובץ הבא כל עוד שם הקובץ מתחיל ב-"hideit_" וברגע שמגיעים לקובץ שלא צריך להחביא מחזירים את התשובה. מה שיקרה זה שהתוכנה סך הכל מבקשת את הקובץ הבא, אבל ה-Rootkit מוודא שלא יחזור אחד מהקבצים שהוא רוצה להסתיר, בעזרת Olly ניתן לראות זאת באופן מובן יותר:




יצרתי תיקיה בשם hideit_please על שולחן עבודה, נכנסתי ל-cmd.exe והפעלתי את התוכנה. כתבתי dir ובאמת התיקיה לא הופיעה, ה-Rootkit עבד. לאחר מכן הפעלתי את התוכנה פעם שנייה, אבל הפעם על

explorer.exe. עשיתי Refresh לשולחן העבודה אבל התיקיה לא נעלמת... ה-Rootkit משום מה לא עובד ב-explorer.exe.




המחשבה הראשונה הייתה שכנראה יש פונקציית API אחרת ש-Explorer משתמש בה. אפיק הציע את NtQueryDirectoryFile, מכיוון שאפילו FindFirstFile ו-FindNextFile הם סך הכל פונקציות נוחות יותר שבעצמן קוראות ל-NtQueryDirectoryFile. רעיון מצוין. עשיתי הוק גם לה, אבל זה עדיין לא עבד.

פתחתי את explorer.exe ב-OllyDbg והרצתי את ה-Rootkit. מצאתי שלוש בעיות:

1. מסתבר שהקוד של ה-IAT Hooking לא תומך ב-Import שהוא על ידי Ordinal (מספר). התוכנה יכולה לבקש "תן לי את FindFirstFileA מ-Kernel32.dll", אבל היא יכולה לבקש גם "תן לי את פונקצייה מספר 12 מ-Kernel32.dll" ובמקרה הזה קוד ההוק לא תמך ולכן ההוקינג נעצר באמצע. תיקנתי את הקוד.
2. ה-explorer.exe באמת לא משתמש באף אחת מהפונקציות האלה! אז איך הוא מראה רשימת קבצים? הוא טוען DLL בשם BrowseUI.dll שמשתמש ב-ShDocVW.dll שמשתמש ב-Shell32.dll שרק הוא קורא לפונקציות ה-API. לכן הוק ל-explorer.exe לא עובד אלא יש צורך בהוק ל-Shell32.dll.
3. אין צורך לעשות הוק ל-NtQueryDirectoryFile, הוא לא משתמש בה ישירות. הוא עושה שימוש בפונקציות FindFirstFileExW (ה-Ex מסמן גרסה מתקדמת יותר עם יותר אפשרויות) ו-FindNextFileW, ולכן אך ורק לשתי הפונקציות האלה יש צורך לעשות הוק.

תיקנתי את קוד ה-IAT Hooking, שיניתי את הקוד שיעשה הוק ל-Shell32.dll והוספתי הוק ל-FindFirstFileExW. הרצתי, לחצתי F5 והתיקיה hideit_please נעלמה משולחן העבודה! ה-Rootkit עבד.

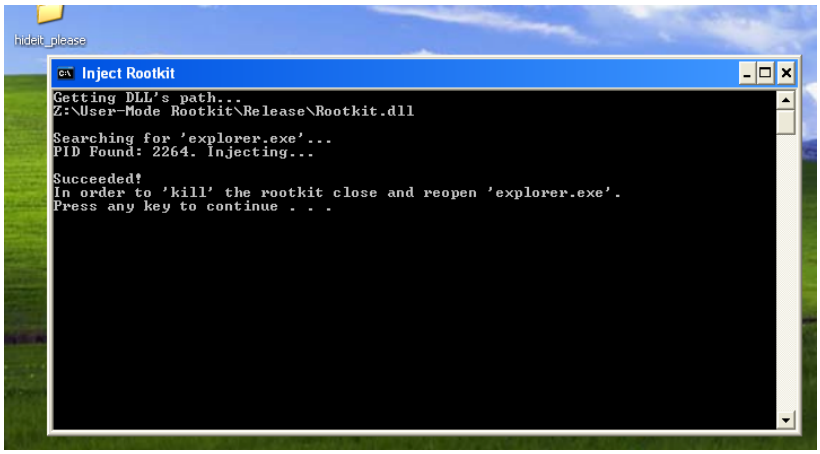
לסיכום: כתבנו פונקציות חלופית ל-FindFirstFileExW ול-FindNextFileW ועשינו הוק אליהם ב-explorer.exe ו-cmd.exe כדי שהקריאה תגיע אליהם. בתוכם קראנו לפונקציה המקורית וסיננו את התשובה. באותה שיטה יכולנו גם לסנן את Process32First ו-Process32Next כדי להסתיר תהליכים. כדאי עכשיו לחזור לתרשים שהצגנו בהתחלה, הוא יהיה ברור יותר.

**הערות אחרונות**

**שרידות:** ברגע שיכבו את המחשב או אפילו יפתחו ויסגרו את explorer.exe ה-Rootkit ייסגר ולא יהיה מה שיחזיר אותו. יש כמה אפשרויות לגרום לו להמשיך לפעול:

- AppInit_DLLs כדי שייטען אוטומטית לתוכנות שרצות במערכת
- אפשר להשתמש בתיקיה Startup או בערכים ברג'יסטרי להפעלה אוטומטית (Run למינהם)
- הזרקת קוד שיטען את ה-DLL לתוך תוכנה לגיטימית (הדבקה בינארית)
- וכל דרך אחרת שתגרום לקוד שלנו לרוץ אוטומטית...

**מתקפת נגד:** אז מה עושים נגד Rootkit? איך נוכל לדעת האם התשובה שאנו מקבלים היא אוטנטית או לא? מארק רוזינוביץ' שהזכרנו בתחילת המאמר בנה כלי מעניין. הכלי נקרא Rootkit Revealer וכדי לגלות Rootkits הוא קורא את תוכן הכונן הקשיח ברמה הנמוכה ביותר (Raw) מנתח ומשווה את התוצאות מול מה שהוא מקבל מה-API. אם יש שוני סימן שמישהו בדרך מחבל בתוצאות. השיטה מאוד אפקטיבית, מכיוון שכדי להסתיר את ה-Rootkit מכלי כזה יש לסנן גם את הקריאות מהכונן הקשיח ולסנן אותם וזה קשה ומסובך בהרבה (מצריך הבנה של מבנה מערכת הקבצים).



**לסיכום**

Rootkit הוא רכיב היושב בין התוכנות שרצות ובין מערכת ההפעלה, ומסנן כל מידע המגיע ממערכת ההפעלה כך שמבחינת התוכנות שרצות במערכת הוא בכלל לא קיים. השימוש בטכניקה זו הוא בדרך כלל זדוני להסתרת ווירוסים ורוגלות.

תודה לאפיק קסטיאל (cp77fk4r) על העזרה בכתיבת המאמר. קובץ מהודר של ה-Rootkit וקוד המקור מצורפים למאמר זה.

את הקוד המלא ואת הבינארי שהוצג במאמר ניתן להוריד מהקישור הבא:

http://www.digitalwhisper.co.il/files/Zines/0x14/User-Mode Rootkit.7z

אתם מוזמנים לבקר בבלוג של אחי ושלי:

http://www.MerkazHaKfar.co.cc/

לתגובות והערות -vbCrLf@GMail.com