**Learning DLL**

**Dynamic-link library**

מטרה:

# להכיר את הרעיון [הספרייה](http://he.wikipedia.org/wiki/%D7%A1%D7%A4%D7%A8%D7%99%D7%99%D7%94_%28%D7%AA%D7%9B%D7%A0%D7%95%D7%AA%29) המשותפת ב[מערכות ההפעלה](http://he.wikipedia.org/wiki/%D7%9E%D7%A2%D7%A8%D7%9B%D7%AA_%D7%94%D7%A4%D7%A2%D7%9C%D7%94) [חלונות](http://he.wikipedia.org/wiki/%D7%97%D7%9C%D7%95%D7%A0%D7%95%D7%AA) של [מיקרוסופט](http://he.wikipedia.org/wiki/%D7%9E%D7%99%D7%A7%D7%A8%D7%95%D7%A1%D7%95%D7%A4%D7%98)

תאוריה:

**Dynamic-link library**

קובץ DLL עונה למבנה של קובץ PE בחלונות, כאשר ביט אחד בHeader-מבדיל בינו לבין קובץ הרצה רגיל של חלונות )קובץ (EXE כמו כל קובץ,PE קובץ DLL יכול להכיל קוד מקור, נתונים ומשאבים. קובצי DLL רבים של מערכת ההפעלה חלונות ממוקמים בספרייה .system32

שמות קובצי DLL עשויים להסתיים באחת מהסיומות הבאות:

* DLL
* OCX – ספריית ActiveX
* DRV ספריית מנהל התקן -

התורנות:

ספריית קישור-דינמי מהווה רעיון מרכזי במערכות ההפעלה חלונות לדורותיהן. היתרון המרכזי של קובצי DLL הוא המודולריות, שמאפשרת שינוי בקוד המקור במספר תוכניות בו זמנית מבלי לשנות את התוכניות עצמן (על ידי שינוי ה- dll.) יתרון נוסף הוא היכולת להוסיף מודולים לתוכניות קיימות כמו ההרחבות בתוכנות של קרן מוזילה. רעיון זה מנוצל בצורה מקסימלית בטכנולוגיית .COM

שימוש:

תוכנות אשר נוצרות בתהליך של קישור סטטי ,(Static Link) מכלילות את הספריות הרצויות להן בקובץ ההפעלה של התוכנה, ולכן אין להן צורך בקובצי DLL בזמן הריצה. לעומתן, תוכנות אשר נוצרות בתהליך של קישור דינמי ,(Dynamic Link) דורשות הימצאות של קובצי DLL במחשב עליו הן פועלות.

טבלאות ייבוא וטבלאות ייצוא:

בתהליך של קישור דינמי, מרוכזות כל הפונקציות הנמצאות בקובצי DLL חיצוניים לתוך טבלה שכל המצביעים שלה ריקים. במהלך טעינת התוכנית

על ידי המקשר הדינמי של חלונות ,(PE Loader) מתבצעת טעינה של קובצי ה-DLL הרלוונטיים לזיכרון הפיזי (במקרה ואחד מקבצי ה-DLL אשר התוכנה תלויה בהם לא קיים במערכת, התוכנה לא תיטען כלל). לרוב קובצי DLL לא עוברים לזיכרון הווירטואלי, אלא נשארים בזיכרון הפיזי עד שכל התוכניות שזקוקות להם מסיימות את פעולתן.

לאחר שכל קובצי ה־DLL הרלוונטיים נטענו לזיכרון, מתבצע מיפוי כתובות של הפונקציות ועדכון של כל המצביעים בטבלת היבוא. פעולת המיפוי נעזרת בטבלאות ייצוא, שמתאימות מספר הסידורי לכל אחת מהפונקציות בקובץ ה־DLL . בדרך כלל, עדיף ליבא פונקציות באמצעות שמן, שכן בגרסאות השונות של מערכת ההפעלה חלונות הסדר של הפונקציות בקובצי ה־DLL יכול להשתנות. החל ממערכות 32־ביט של חלונות טבלאות היצוא מסודרות לפי שם, ולכן ניתן לאתר את הפונקציות בחיפוש בינארי שהתקורה שלו לא גבוהה במיוחד.

טעינה מושהית(Delay load):

באפשרות זו, טעינה של קובץ DLL תתבצע רק כאשר התוכנית תקרא בפעם הראשונה לפונקציה שנמצאת בו, והקובץ ישאר בזיכרון הפיזי עד לסיום של כל התוכניות שתלויות בו. אופציה זו מאפשרת טעינה מהירה יותר של התוכנה, וכן הפעלה של התוכנה גם ללא הימצאות של כל קובצי ה-DLL הנדרשים. במקרה שהתוכנה תנסה להשתמש בפונקציה שקובץ ה-DLL שלה לא קיים, יזרק חריג מובנה (Structured Exception) בתהליכון (Thread) אשר קרא לפונקציה.

קישור יזום:

קיימות שלוש פונקציות בסיסיות ב-API של ,Windows שמאפשרות לטעון ספריות DLL באופן יזום ומבוקר. למעשה, גם שתי האפשרויות הקודמות משתמשות בפונקציות אלה באופן שקוף למפתח התוכנה. הפונקציות ממוקמות בקובץ ,**kernel32.dll** אשר נטען למרחב הזיכרון בתהליך האתחול של חלונות ונשאר שם עד לכיבוי המחשב או ליציאה ממערכת ההפעלה:

הפונקציה **LoadLibrary** מקבלת את המיקום המוחלט, המיקום היחסי או המיקום הסתמי שם קובץ PE (ראה בסעיף סדר טעינה), טוענת אותו למרחב הזיכרון ומחזירה את הכתובת אליה הוא נטען .(Base Address)

הפונקציה מנהלת מנגנון ,**Reference count** שמונע טעינה כפולה של אותו קובץ , כלומר: כאשר תוכנית קוראת לפונקציה בקובץ DLL כלשהו, הפונקציה טוענת את הקובץ רק אם הוא לא נמצא כבר בזיכרון הפיזי (דינמי). הפונקציה יכולה לקבל שם קובץ בפורמט .Unicode

הפונקציה **GetProcAddress** מחפשת כתובות של פונקציות בקובצי DLL שנמצאים במרחב הזיכרון. הפונקציה מקבלת את כתובת הבסיס של ה־DLL ואת שמה של הפונקציה שאותה רוצים לייבא. שם הפונקציה לא יכול להיות בפורמט .Unicode

הפונקציה **FreeLibrary** משחררת את קובץ ה־DLL ממרחב הזיכרון.

כישלון באחת מהפונקציות הנ"ל מתבטא בהחזרת ערך שגיאה, איתו ניתן להתמודד באופן מבוקר באמצעות הודעה למשתמש או ניסיון לטעינת קובץ אחר.

היתרון בקישור יזום הוא האפשרות שהוא נותן למפתח להתמודד עם שגיאות שיכולות להתרחש בזמן ריצה, וללא קישור יזום היו גורמות להודעת שגיאה של מערכת ההפעלה או לסיום של התוכנית.

סדר טעינה:

כאשר נטען קובץ DLL בעל שם סתמי, ללא ציון מיקום מלא, המערכת תחפש אותו במיקומים הבאים, לפי סדר:

1. הספרייה ממנה נטענה התוכנה
2. ספריית system32
3. ספריית system
4. ספריית windows
5. הספרייה הנוכחית של התוכנה (Current working directory)
6. הספריות המופיעות במשתנה הסביבה .PATH

סדר החיפוש עשוי להשתנות, בעיקר משיקולי אבטחה. לדוגמה, במידה ונבחרת רמת אבטחה נמוכה יותר, יתבצע חיפוש בספרייה הנוכחית לפני ספריות system32 ו.system-

קובצי DLL נפוצים במערכת ההפעלה חלונות:

להלן מספר דוגמאות של קובצי DLL שנטענים עם האתחול של מערכת ההפעלה חלונות.

|  |  |
| --- | --- |
| **שם הקובץ** | **תיאור** |
| **kernel32.dll** | Windows NT BASE API Client DLL |
| **user32.dll** | Windows XP USER API Client DLL |
| **advapi32.dll** | Advanced Windows 32 Base API |
| **ntdll.dll** | NT Layer DLL |
| **msvcrt.dll** | Windows NT CRT DLL |

**שלב 1:**

באמצעות תוכנה PEBrowsePro (או אחרת) לפתוח קובץ mspaint.exe (צייר)

לעתיק את כל הDLL שמשתמשת צייר

**שלב 2:**

Graphics Device Interface (GDI) is a Microsoft Windows application programming interface.

GDI ליבת רכיב מערכת הפעלה אחראי לייצוג אובייקטים גרפיים ומשדר אותם להתקני פלט כגון צגים ומדפסות.

מ Windows XPהיה משופר ל GDI+

להעתיק שלוש פונקציות גרפיות מGDI32.DLL ושלוש פונקציות מGDIPLUS.DLL לציור בסיסי ולפרש שימוש בהם.

**שלב 3:**

באמצעות תוכנה PEBrowsePro לפתוח קובץcmd.exe

מה סיבה שהקובץ לא משתמש בGDI?

**שלב 4:**

באמצעות תוכנה PEBrowsePro לפתוח קובץpython.exe

האם python משתמש רק בDLL של מערכת הפעלה או לא?