Manual Packing

מאת הלל חימוביץ' (HLL)

הקדמה

המונח Packing שנתייחס אליו לאורך המאמר לצורך העניין בתור 'אריזה' או 'קידוד' נועד במקור לדחיסה של קובצי הרצה ללא צורך ברכיב חיצוני. תהליך האריזה, בעקרון דומה למה שקורה כאשר מבצעים דחיסה של קובץ ל- Self-Extracted Archive, תהליך שבו גם מנגנון הדחיסה וגם הנתונים הדחוסים מצורפים יחדיו לאריזה אחת. תהליך האריזה שנתייחס אליו במאמר זה אינו נועד למזעור גודלו של הקובץ המוגמר אלא לביצוע Obfuscation (טשטוש) של קוד המכונה הנמצא בקובץ.

ישנן עשרות אם לא מאות כלים אוטומטים שמבצעים את התהליך הזה בצורה אוטומטית, אך יחד איתם קיימים גם כלים היודעים לפתוח את האריזה הזו באופן אוטומטי, ובנוסף חלק מכלי האריזה האוטומטים מטמיעים חתימה על הקובץ שניתנת לזיהוי בקלות ע"י אנטי וירוס.

במאמר זה נביא דוגמא עקרונית לביצוע של תהליך אריזה ידנית. התהליך כלל אינו מורכב ובסיסי לגמרי. אומנם, התהליך לא יביס את כל כלי האנטיוירוס אבל בעיקר כן יעזור לכם במידה ואתם רוצים לעקוף מנגנון סינון תוכן, אנטיוירוסים קצת פחות מתוחכמים וכדו'.

דרישות קדם

- Ollydbg גרסא הנמוכה מ-2.0. נכנה את התוכנה בקיצור בשם Olly בהמשך.
- LordPE – כלי שימושי לעריכת כותרות ה-PE Header של קבצי Object של מייקרוסופט.
- Notepad
- ידע בסיסי בשפת אסמבלי .
- קובץ אותו היינו מעוניינים לקודד – למאמר זה, אני משתמש ב- nc.exe – חלק יכנו אותו "NetCat – A Hacker Swiss Army Knife"

אבסטרקט

במאמר זה אנחנו נמקם פיסת קוד באסמבלי שתיועד להרצה לפני התחלת התוכנית המקורית, פיסת הקוד הנ"ל הינה בעצם הקוד שמהווה את ה-Unpacker שלנו **וגם** את ה-Packer שלנו.

התהליך שיקרה הוא, שבעת הפעלת האפליקציה , ירוץ קטע הקוד שלנו שתפקידו **לפתוח בזמן ריצה – על הזיכרון את הקוד שאכן אמור להיות מורץ**. הכוונה היא, שבעת טעינת התוכנית הקובץ לא ניתן להרצה, וע"י כך גם אינו ניתן לזיהוי, הקובץ יקודד/יוצפן ב**מפתח** לבחירתנו, ובכמה שורות הראשונות של חיי התוכנית, קטע הקוד שלנו (שהוא אכן קריא למעבד) יבצע שימוש במפתח זה ע"מ לפתוח את שאר קוד האפליקציה ולהעביר את השליטה אליה.

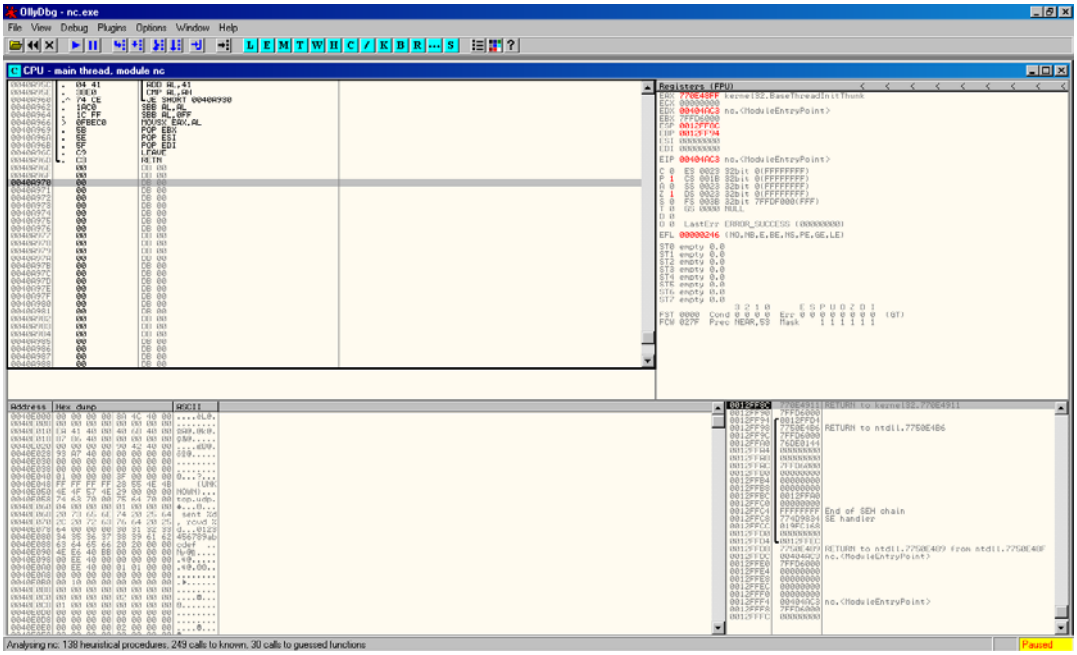
לצורך זה נצטרך לבצע מספר פעולות:

1. לאתר אזור קוד שאינו בשימוש עם די מקום (נצטרך כ-30 בתים).
2. לערוך את ה-PE Header של התוכנית כך ש:
   - תהיה הרשאת כתיבה למקטע הקוד (מכיוון שאנחנו משנים אותו בזמן ריצה)
   - נשנה את ה-Entry Point של התוכנית, כך שיפנה לקוד שלנו.
3. לכתוב את הלולאה המפענחת את הקובץ (במקרה שלנו זו אותה הלולאה גם לפיענוח וגם לקידוד) ובסיומה, מעבירה את השליטה חזרה לתוכנית הראשית.
4. לכתוב את כלל הקוד בצורתו המקודדת לקובץ EXE חדש, יחד עם הלולאה המפענחת.

מציאת אזור קוד פנוי ותכנון טווח ה- Packer

נתחיל את העבודה בטעינת הקובץ ב-Olly. מיד לאחר שטענתם את הקובץ עם Olly, Olly מסמן את נקודת הכניסה של התוכנית. זכרו אותה להמשך, אתם תזדקקו לה.

באופן מפתיע, קבצי EXE מכילים לרוב לא מעט אזורים מתים, אזורים שהם מוקצים, תופסים מקום בקובץ, על ה-HD, וגם בזיכרון לאחר טעינה - אך לעולם אינם בשימוש. אנחנו ננצל את התופעה הנ"ל ונאתר מקום שבו נוכל למקם את הקוד שלנו. איך נעשה זו? פשוט מאוד, נעלה את Olly ונתחיל לרפרף במקטע הקוד. אנחנו מחפשים אזור רציף של אפסים. אזור כזה לרוב מאוד נפוץ בתחילתו או בסופו של הקובץ. לעתים ניתן למצוא אזור כזה גם בחלקים אחרים בקובץ.



כפי שניתן לראות בתמונה (בחלק השמאלי העליון), אותר בסוף הקובץ חלק גדול שאינו בשימוש. למעשה, חלק זה משתרע עד סופו של מקטע הקוד.

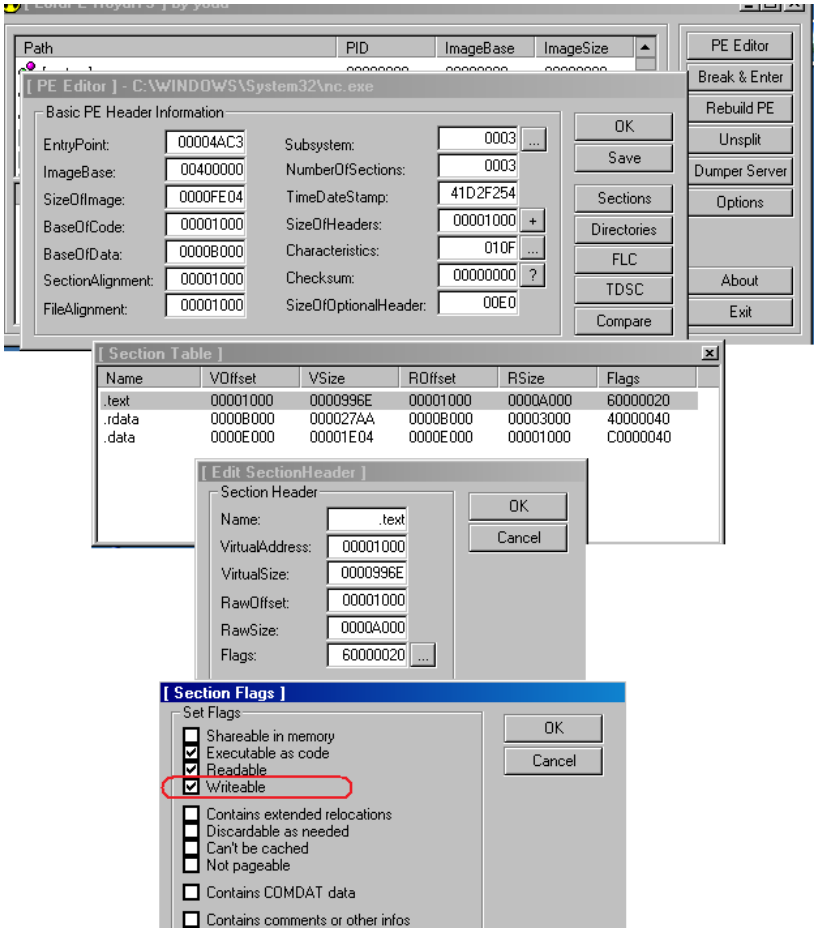
אנו נשתמש במיקום המסומן בכתובת הזיכרון 0040A970, ושם נמקם את הקוד שלנו שיבוצע לפני ה- EP המקורי של התוכנית.

דבר נוסף שתצטרכו לרשום לעצמכם הוא מאיזה בית בתוכנית תרצו להחל את האריזה שלכם על הקובץ ועד לאיזה בית. <u>לא ניתן לארוז הכל!</u> אתם לא יכולים לארוז את קוד ה- Packer עצמו - הוא חייב להיות קריא כבר מתחילת התוכנית. לכן, נרשום לנו לדוגמא ככתובת התחלה את תחילת מקטע הקוד, לדוגמא 0x401000. כתובת הסיום תהיה עד תחילת קטע ה-Unpacker שלנו, לדוגמא 40A96E. במקרה כזה האיזור שנארוז משתרע על פני 996E בתים.

אפשור כתיבה למקטע הקוד ושינוי נק' הכניסה של התוכנית

אנו הולכים לכתוב לתוך קובץ ההרצה קטע הקוד שישנה את תוכן מקטע הקוד של התוכנית בזמן ריצה. ב-Windows קיימת הגנה נגד דבר זה מכיוון שהדבר אינו אמור לקרות בפעולה רגילה של תוכנית תקינה. במצב שקיימת הגנה ההנחה היא שאם היה ניסיון לכתוב על מקטע הקוד הרי שזו שגיאה שלא באמת אליה התכוון המשורר.

ניתן לכבות את ההגנה על מקטע הקוד באמצעות שינוי ה- Attributes במקטע (Section) המתאים בכותר ה-PE, לצורך זה נעלה את LordPE. נפתח באמצעותו את הקובץ שלנו (תחת PE Edit), נלחץ על Sections נבחר את המקטע ששמו .text (זהו שמו לרוב – אך זה לא חייב להיות שמו) ונערוך את כותרתו.



בנוסף, אנחנו צריכים לשנות גם את נקודת הכניסה לתוכנית שתפנה לקוד שלנו, שכאמור, בזיכרון יושב בכתובת 0040A970, אך בכותר ה- PE אנחנו צריכים לציין את ה- Address (RVA Relative Virtual) של ה- EP החדש, ולא הכתובת שלו בזיכרון. כדי לעשות זאת נעבוד על פי הנוסחא הבאה:

ImageBase + RVA = Virtual Address (address in memory)

כלומר - ע"מ להגיע מהכתובת הווירטואלית בזיכרון שיש לנו 0040A970, אל ה- RVA של ה- EP שאנחנו צריכים לציין, נחסר מן ה- VA את ה- ImageBase, שניתן לראותו ב- LordPE. במקרה שלי, החישוב מסתכם בכתובת A970 – ואותה אציין ב- LordPE בתור ה- Entry Point של התוכנית.

עכשיו שערכנו את כל מה שצריך בכותר ה-PE, אפשר לגשת ולהתחיל לכתוב את הלולאה שלנו.

כתיבת לולאת הקידוד וקידוד הקובץ

מנגנון האריזה שבחרתי להציג פה הוא הצורה הכי פשוטה שקיימת לבצע את תהליך הזה, ב-MetaSploit Framework קימים כלים דומים למה שיוצג פה. האריזה שבחרתי תתבצע באמצעות פעולת XOR על כל אחד מהבתים של מקטע הקוד. פעולת XOR הינה פעולה דו כיוונית, כלומר אם נריץ פעולת XOR על תא זיכרון, ונבצע אותה בשנית עם הערך החדש באותו תא הזיכרון, הערך יחזור להיות הערך המקורי שלו. מכאן - התהליך שפותח את הקובץ שלנו להרצה, יהיה אותו תהליך בדיוק המקודד אותו לשמירה.

לאחר שערכנו עם Lord PE את כל הנדרש, נעלה את ה-OllyDbg ונגש למיקום שבו החלטנו למקם את קטע הקוד.

להלן קטע הקוד באסמבלי עם הסברים נלווים, המבצע את פעולת ה-Packing וה-Unpacking:

```
mov EDI, 00401000       ;     Start of area to be packed
mov ECX, 0996E          ;     Amount of bytes from that address
packer_loop:
xor byte ptr [EDI], 56  ;     xor each byte. 56 is the key that we will
                        ;     use to xor each byte.
inc edi                 ;     move on to the next address to xor
loop packer_loop        ;     loop this 996E times
jmp 404AC3              ;     jmp to the original program EP
```

אם נריץ את קטע הקוד הזה בלבד (ניתן למקם BreakPoint בפקודה האחרונה), בפעם הראשונה הקוד יקודד את מקטע הקוד של התוכנית. אם נריץ אותו פעם נוספת, הוא יפתח אותה.

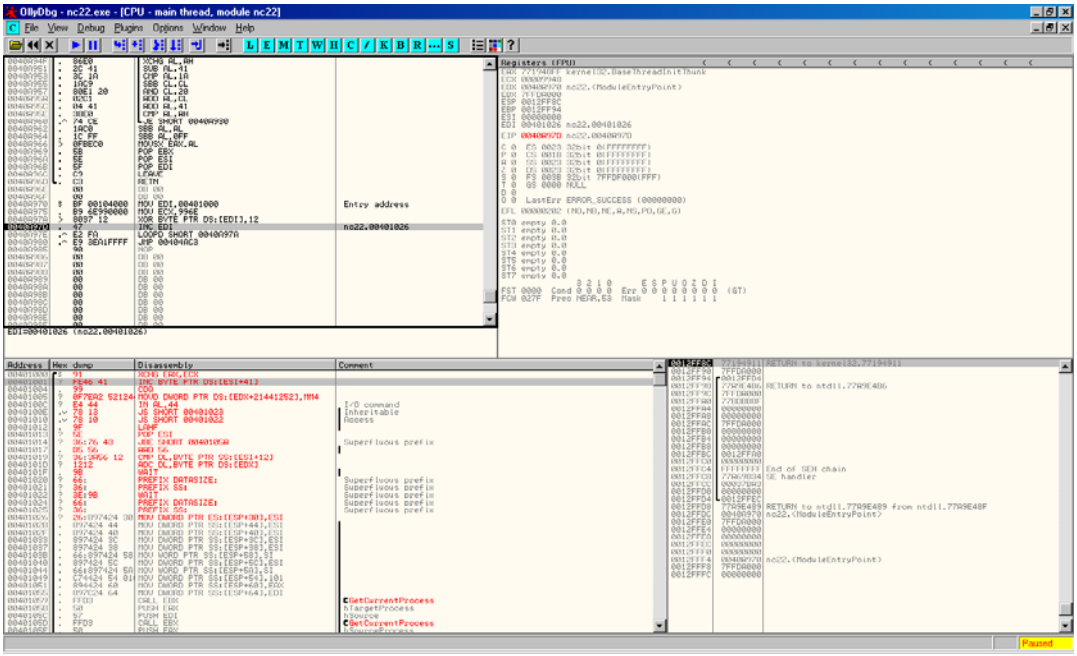
כדי לא להרוס את הקובץ המקורי נשמור את הקובץ כמו שהוא כ- EXE חדש. נלחץ לחצן ימיני Copy to executable -> All changes ונקבל חלון המכיל את ה- EXE החדש שלנו, אם ננסה לסגור את החלון תופיע לנו שאלה האם אנחנו רוצים לשמור אותו, אמרו כמובן כן.

עכשיו פתחו באמצעות Olly את הקובץ החדש והנה הוא נפתח ישירות אל קוד ה-Packer שלנו. עכשיו כל מה שנותר לעשות זה להריץ את הלולאה פעם אחת ללא הרצת הקובץ בשביל שקטע הקוד הזה יבצע XOR לכל החלק שציינו.

מקמו BreakPoint על הפקודה האחרונה (jmp) והריצו את הקובץ.

<u>הערה:</u> אם Olly לא צובע לכם באדום את הפקודות זה אומר שהוא לא שם לב לשינוי ולא ישמור לכם את השינויים לקובץ חדש, כדי להתעלות על (הבאג?) זה פשוט ערכו איזה שהוא ערך בתחילת מקטע הקוד – אל תשנו, רק תעשו Ctrl+E ותאשרו, זה יגרום ל-Olly לשים לב לשינויים שהולכים להתבצע במקטע זה.

<u>הצעה:</u> אם אתם רוצים לראות את התהליך בצורה אלגנטית, כוונו את ה- Dump view להציג קוד אסמבלי, וקפצו לכתובת תחילת אזור הזיכרון, לאחר מכן, לחצו שוב ושוב על כפתור ה- Step over וראו כיצד הקוד שלכם מקודד את קוד התוכנית, מאוד מגניב ☺.



כאשר הגעתם לפקודת jmp הנכם נמצאים במצב שכל מקטע הקוד עבר קידוד עם XOR במפתח שבחרתם (Olly מסמן בתים ששונו באדום). על מצב הקובץ הנוכחי, נבצע שוב שמירה כמו שהזכרתי קודם. זהו – יש לכם קובץ ארוז שלא מזוהה ע"י סורקי חתימות בתור NetCat (או כל דבר אחר שבחרתם לארוז).

מה שיקרה כשתריצו את הקובץ החדש, נקודת הכניסה לתוכנית תהיה לולאת ה-Unpacking. הלולאה תבצע, החל מתחילת מקטע הקוד ועבור כל Byte, XOR עם הפתח שבחרתם. מכיוון ש-XOR היא פעולה הפיכה, ברגע שתעשו XOR בפעם השניה (הפעם הראשונה היתה כשהקובץ קידד את עצמו) ה-Byte יחזור לערכו המקורי. בסוף לולאת ה-Unpacking יקפוץ הקוד חזרה לתוכנית הראשית המפוענחת והיא תתחיל לרוץ.