

פרק 9

RE והצולף האמיתי



אל מה היה לנו?

- ניתוח סטטי

- לוקח זמן רב
- ניתן לכסות חלקים קטנים

- ניתוח דינאמי

- יותר יעיל בזמן
- יותר מסוכן

- Hooking

- נותן נראות יותר גבוהה
- דורש הבנה יותר גבוהה של הקוד



אל מה היה לנו?

Does not
Scale Well

- ניתוח סטטי

- לוקח זמן רב
- ניתן לכסות חלקים קטנים

- ניתוח דינאמי

- יותר יעיל בזמן
- יותר מסוכן

- Hooking

- נותן נראות יותר גבוהה
- דורש הבנה יותר גבוהה של הקוד



אל איך מתמודדים עם כאוויות קוד ונולקות?



לא הכל זה פינארי!



- בדפדפן –

- Java Script
- HTML5

Interpreted Languages and Scripts •

- Python



- Powershell

- שפות קודי ביניים

- Java
- C#



קודי ביניים

- תרגום שפה עילית לקוד ביניים שאינו שפת מכונה

- קוד הביניים הינו שפה פשוטה

- בדרי"כ ממודלת כמכונת מחסנית אבסטרקטית

- שקל לפרש אותה ולהריץ אותה, ללא צורך במהדר נוסף



- בכל מחשב יש תוכנה שמריצה את קוד הביניים

- באינטרפרטציה, ללא צורך בהידור נוסף

- בחלק מהמקרים קוד הביניים מתורגם בזמן ריצה לשפת מכונה

- Just In Time Compiler (JIT)

- שתי דוגמאות עיקריות

שפת קוד ביניים	שפת תכנות	הערות
Java Bytecode 	Java	מתוכננת במיוחד ל-Java
CIL (ידועה גם כ-MSIL)	.NET (C# ועוד)	תומכת במגוון שפות

- נתמקד ב-Java Bytecode – בקיצור JB



מבנה בסיסי של Java Bytecode

- שפה מבוססת מחסנית
 - פקודות JB לוקחות פרמטרים מהמחסנית ומחזירות את תוצאתן בראש המחסנית
 - גם המשתנים המקומיים על המחסנית
- ה-opcode תמיד באורך בית אחד
 - יש מקרה בודד של prefix
 - wide – מאפשר אופרנדים בגודל שני בתים בפקודות מסוימות שבדרי"כ מקבלות אופרנדים בני בית אחד
- פקודות JB יכולות לקבל מספר קטן של אופרנדים
 - כך שאורך פקודת JB משתנה
 - אופרנדים יכולים להיות בגודל בית אחד או שניים
 - לעיתים גם ארבעה



מבנה בסיסי של Java Bytecode

• JB רץ בתוך מכונה וירטואלית

- מתוכננת במיוחד להגנה כנגד חולשות
- למשל, המכונה בודקת חריגה ממערכים
 - מייצרת exception במקרה של חריגה
- המכונה הווירטואלית מוודאת שקפיצות הן תמיד לתחילת פקודות
 - נבדק על ידי ה-verifier בתחילת ההרצה

• כל כניסה על המחסנית היא בת 32 סיביות

- שלם, מצביע, וכו'
- ייצוג של long הוא כשתי כניסות רצופות (64 סיביות)



טכנולוגיית JIT

Just In Time compilation

- הידור קוד בזמן ריצה
- יכולת לבצע אינטרפרטציה לקוד, ולבצע הידור רק לקוד שבשימוש רב
- וללמוד מהאינטרפרטציה איך כדאי לבצע אופטימיזציות
- מאפשר לוודא שהקוד המהודר אינו מתוכנן לגרום נזק
- קוד הביניים פשוט יותר לבדיקה
- הקוד המהודר אינו ניתן לטיפול ע"י תוקף
- כלומר, נוזקה אינה יכולה להשתלט על הקוד...



מבנה קובץ הרצה Java-ק

- קובץ הרצה jar הינו ארכיון ZIP
 - כלומר ניתן לפתוח אותו עם WinZip

```
public class Example
{
    public static int globalnum=5;

    public static void main (String args[])
    {
        int num=3+globalnum;

        if(globalnum<10) { num++; };

        System.out.println("The sum is "+num);
    }
}
```

- קובץ jar כולל
 - קובץ ראשי

○ META-INF/MANIFEST.MF

○ מתאר היכן התכנית הראשית, מיקום ספריות, וכו'

- קבצי מידע כללי

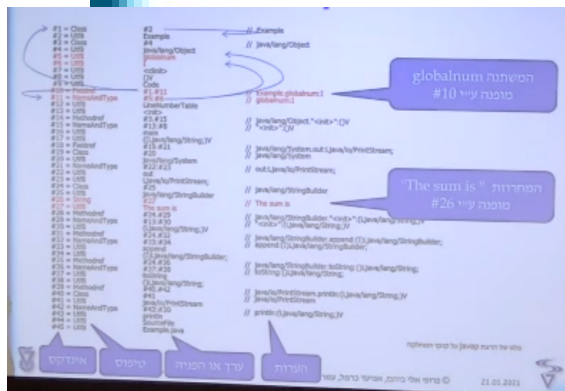
○ author.txt

- וקבצי class

○ הכוללים Bytecode

○ program.class, lib.class, ...

- ייצוגים פנימיים תמיד ב-big-endian



מבנה קובץ Class

• קובץ class כולל

- תיאור התוכן הפומבי של המחלקה
- רשימת קבועים (Constant pool)
 - כוללת את כל הקבועים וסוגם
 - כלומר
 - הגדרות המשתנים (בפרט הפומביים של המחלקה), עם הפניות לטיפוסים
 - ✓ ההפניה היא לאיבר אחר ברשימה
 - מחרוזות
 - ✓ מחרוזות שמופיעות בתכנית
 - ✓ מחרוזות של שמות מחלקות ושמות פונקציות שבשימוש
 - כל הטיפוסים שיש בתכנית
 - מבנה של ענפים של עץ
- הקוד (bytecode) של כל הפונקציות של המחלקה
 - בשפה דמוית אסמבלי



טיפוסים בסיסיים בשפת Java

- להלן רשימת הטיפוסים הבסיסיים בשפת Java וקיצוריהם
 - הקיצורים משמשים בקובץ ה-class

		טיפוס	קיצור
signed	{	integer	I
		long	L
		short	S
		byte	B
Unicode (16 bit)	{	character	C
		float	F
		double	D
		boolean	Z
		reference	A

- למערך יש תוספת "[
- אין טיפוסים unsigned int/long/byte...



חבנה רשימת הקבוצים

- רשימת הקבועים (Constant Pool) היא מערך

- עם הפניות בצורת ענפים של עץ

- כל כניסה במערך מכילה

- אינדקס (מאחד עד מספר הקבועים)

- טיפוס

- Utf8 : טקסט מקודד ב-UTF-8

- משמש לכל הטיפוסים האחרים

- קידוד Utf8 אינו קידוד UTF-8 תקני

- ✓ NUL מקודד בשני בתים, ותווי UTF-8 ארוכים מקודדים שונה

מבתקן

- String : מחרוזת, בצירוף האינדקס של הטקסט UTF-8 שלה

- Class : מחלקה, בצירוף האינדקס של הטקסט UTF-8 של השם שלה



מבנה רשימת הקבוצים

- NameAndType : שם וטיפוס, עם שני אינדקסים ל-UTF-8, האחד לשם המזהה, והשני לקיצור שם הטיפוס (אות אחת) או טיפוס פונקציה
- Methodref : מתודה, עם אינדקס של מחלקה ואינדקס NameAndType
- Fieldref : שם שדה או שם משתנה, עם אינדקס של מחלקה ואינדקס NameAndType
- Integer
- Float
- Long
- ...

ענף של משתנה globalnum		
#10	Fieldref	
#1 Class	#11 NameAndType	
#2 Utf8	#5 Utf8	#6 Utf8
Example	globalnum	I



שאלות תכנות

```
public class Example
{
    public static int globalnum=5;

    public static void main (String args[])
    {
        int num=3+globalnum;

        if(globalnum<10) { num++; };

        System.out.println("The sum is "+num);
    }
}
```



Constant pool

#1 = Class	#2	// Example
#2 = Utf8	Example	
#3 = Class	#4	// java/lang/Object
#4 = Utf8	java/lang/Object	
#5 = Utf8	globalnum	
#6 = Utf8	I	
#7 = Utf8	<clinit>	
#8 = Utf8	()V	
#9 = Utf8	Code	
#10 = Fieldref	#1. #11	// Example.globalnum:I
#11 = NameAndType	#5: #6	// globalnum:I
#12 = Utf8	LineNumberTable	
#13 = Utf8	<init>	
#14 = Methodref	#3. #15	// java/lang/Object.<init>():()V
#15 = NameAndType	#13: #8	// "<init>":()V
#16 = Utf8	main	
#17 = Utf8	([Ljava/lang/String;)V	
#18 = Fieldref	#19. #21	// java/lang/System.out:Ljava/io/PrintStream;
#19 = Class	#20	// java/lang/System
#20 = Utf8	java/lang/System	
#21 = NameAndType	#22: #23	// out:Ljava/io/PrintStream;
#22 = Utf8	out	
#23 = Utf8	Ljava/io/PrintStream;	
#24 = Class	#25	// java/lang/StringBuilder
#25 = Utf8	java/lang/StringBuilder	
#26 = String	#27	// The sum is
#27 = Utf8	The sum is	
#28 = Methodref	#24. #29	// java/lang/StringBuilder.<init>:(Ljava/lang/String;)V
#29 = NameAndType	#13: #30	// "<init>":(Ljava/lang/String;)V
#30 = Utf8	(Ljava/lang/String;)V	
#31 = Methodref	#24. #32	// java/lang/StringBuilder.append:(I)Ljava/lang/StringBuilder;
#32 = NameAndType	#33: #34	// append:(I)Ljava/lang/StringBuilder;
#33 = Utf8	append	
#34 = Utf8	(I)Ljava/lang/StringBuilder;	
#35 = Methodref	#24. #36	// java/lang/StringBuilder.toString:()Ljava/lang/String;
#36 = NameAndType	#37: #38	// toString:()Ljava/lang/String;
#37 = Utf8	toString	
#38 = Utf8	()Ljava/lang/String;	
#39 = Methodref	#40. #42	// java/io/PrintStream.println:(Ljava/lang/String;)V
#40 = Class	#41	// java/io/PrintStream
#41 = Utf8	java/io/PrintStream	
#42 = NameAndType	#43: #30	// println:(Ljava/lang/String;)V
#43 = Utf8	println	
#44 = Utf8	SourceFile	
#45 = Utf8	Example.java	

המשתנה globalnum
מופנה ע"י #10

"The sum is "
מופנה ע"י #26

פלט של הרצת javap על קובץ המחלקה



אינדקס

טיפוס

ערך או הפניה

הערות

© פרופ' אלי ביהם, אביעד כרמל, עמר

19.01.2021



דוגמת JB fe התכנית

```

static {};                                     // Constructor
    stack=1, locals=0, args_size=0
    0: iconst_5
    1: putstatic      #10                     // Field globalnum:I
    4: return

public static void main(java.lang.String[]);
    stack=4, locals=2, args_size=1
    0: iconst_3         // Short form for   iconst  3 (single byte instead of 3)
    1: getstatic      #10                     // Field globalnum:I
    4: iadd
    5: istore_1        // Short form for   istore  1 (single byte instead of 3)
    6: getstatic      #10                     // Field globalnum:I
    9: bipush         10
    11: if_icmpge      17
    14: iinc           1, 1
    17: getstatic      #18                     // Field java/lang/System.out:Ljava/io/PrintStream;
    20: new             #24                     // class java/lang/StringBuilder
    23: dup
    24: ldc            #26                     // String The sum is
    26: invokespecial  #28                     // Method java/lang/StringBuilder."<init>":(Ljava/lang/String;)V
    29: iload_1
    30: invokevirtual  #31                     // Method java/lang/StringBuilder.append:(I)Ljava/lang/StringBuilder;
    33: invokevirtual  #35                     // Method java/lang/StringBuilder.toString:()Ljava/lang/String;
    36: invokevirtual  #39                     // Method java/io/PrintStream.println:(Ljava/lang/String;)V
    39: return
    
```

כתובת

opcode


אופרנדים

הערות



מזלז האקטחה ק-Java

- תמיכה במגוון מבנים מאובטחים ע"י שפת התכנות

- Strong data typing
- ניהול זיכרון אוטומטי (ושחרור אוטומטי)
- אין cast ללא בדיקה
- אין אריתמטיקה של מצביעים 
- מערכים שמורים בזיכרון הדינמי
 - מוקצים בזמן ריצה (ע"י new), לא על המחסנית

- מכונה וירטואלית שבודקת כל מה שאפשר

- מגוון בדיקות בזמן טעינת מחלקה
 - בפרט, Bytecode verification
 - למשל, כל הקפיצות למיקומים חוקיים (לא לאמצע פקודות JB)
- בדיקות בזמן ריצה
 - למשל בדיקות חריגה ממערך
 - בדיקות שימוש במצביע בעל ערך NULL



מודל האבטחה ב-Java

• sandbox

- מאפשר קביעת הרשאות לקוד שרץ
- כלומר, לא כל ההרשאות של מריץ הקוד מאופשרות
 - למשל, מניעת גישה לקבצים, גישה לרשת או גישה לשרתים מסוימים ברשת, יציאה מהמכונה הווירטואלית, יצירת תהליך חדש, וכו'

• JIT

- תוקף לא יכול לשתול קוד כרצונו
- לסיכום, מניעת בעיות אבטחה מקוד עוין על ידי מניעת היכולת של הקוד לבצע פעולות בעייתיות
 - ברמת שפת התכנות, המכונה הווירטואלית, וקריאות לפעולות מסוימות
 - אולם, ההגנות לא מושלמות
 - בשנים האחרונות התגלו מגוון בעיות אבטחה ב-Java שגרמו להפסקת השימוש ב-Java במספר ארכיטקטורות



מזל האססה ב-Java

- מה לא נעשה?
 - אין בדיקת integer overflow 
- כל שמות המשתנים ברמת המחלקה רשומים ברשימת הקבועים
 - כך שהם נגישים לכל מי שמעוניין לבצע RE
 - בפרט נגישים לדה-קומפילרים 
- ואפילו מספרי השורות כלולים בקובץ...
 - במקור לצרכי דיבוג

דיס-אסמבלרים ודה-קומפילרים

- קיימים מגוון של דיס-אסמבלרים ודה-קומפילרים
 - הלוקחים קוד JB ומתרגמים לקוד קריא



• למשל

- דיס-אסמבלרים
 - javap
 - גם גרסאות מסוימות של IDA
- דה-קומפילרים
 - JD
 -  JAD
 - Krakatau

• בדרי"כ מייצרים קוד די טוב



דג-קומפילציה ע"י JD מקובץ ה-Class

```
import java.io.PrintStream;
```



```
public class Example
{
    public static int globalnum = 5;

    public static void main(String[] paramArrayOfString)
    {
        int i = 3 + globalnum;

        if (globalnum < 10) i++;

        System.out.println("The sum is " + i);
    }
}
```



Java bytecode *fe* Obfuscation



- העלמת מחרוזות מרשימת הקבועים
 - למשל יצורן בקוד
 - שינוי שמות משתנים ומתודות
 - ניקוי נתוני דיבוג
 - וכמובן, סיבוך הקוד
-
- יש מגוון תוכנות obfuscation ייעודיות ל-Java



קצת על האמנה fe קובץ Class



• רשימת הקבועים

- הרשומות מסודרות לפי סדר
 - ללא אינדקס מפורש בפנים
- כל רשומה מתחילה בבית המתאר טיפוס
 - ומידע באורך קבוע
 - במקרה של UTF-8 – אורך משתנה
 - לאחר הטיפוס, אורך בשני בתים ואז המחרוזת
- כלומר קל לשנות תוכן של רשומה
 - כולל לשנות אורך
 - בלי צורך לשנות דבר ברשומות אחרות
 - כל עוד לא מוחקים ולא מוסיפים רשומות באמצע

• JB



- קפיצות נעשות באופן יחסי למיקום תחילת הפקודה
- שינוי קוד אינו דורש תיקון בקפיצות שאינן קופצות מעל השינוי
- עדיין עדיף לשנות בלי לשנות אורך



הוק'נ'ג fe קבצי Class

- דוגמאות לשינויים ב-JB
 - דוגמא לשינוי מחרוזת ישירות ב-class
 - כולל שינוי אורך
 - דוגמא לשינוי opcode ישירות ב-class
 - למשל שינוי חיבור לכפל
 - שינויים יותר גדולים
- ה-bytecode verifier מונע מאיתנו לשנות בדרכים לא קבילות
 - למשל קפיצה לאמצע פקודה
- לא ניתן לבצע הוקינג לקוד בזיכרון בזמן ריצה
 - אין לתכנית גישה לקוד...



הבדלים ציקריים בין JB ל-CIL

- CIL תוכננה על סמך העקרונות של JB

- JB תוכננה במיוחד עבור Java

- CIL תוכננה לתמוך במגוון שפות



- לכן כוללת מגוון גדול יותר של טיפוסים ומבנים

- המבנה העקרוני של השפות דומה

- שתיהן מבוססות מחסנית

- ואפילו עם פקודות דומות רבות

- אבל CIL למדה מהניסיון של JB

- עם שיפורים

- קוד הביניים תוכנן ל-JIT

- JB תוכננה לתמוך באינטרפרטציה וב-JIT



Java כאנדרואיד

- באנדרואיד הפעלת Java מורכבת יותר
 - כדי לאפשר את יתרונות Java מצד אחד ופעולה יעילה מצד שני
- עד אנדרואיד 4.4 – שימוש ב-Dalvik
 - אפליקציה נשמרת בקובץ APK שבתוכו נשמר קובץ DEX
 - Dalvik executable
 - קוד ביניים ייעודי שיותר קל לבצע לו JIT
 - נוצר ע"י המפתח מקבצי ה-class וה-JAR בזמן יצירת האפליקציה
 - האפליקציה מורצת כ-JIT ע"י הידור ה-DEX בזמן ריצה
- אנדרואיד 5 – שימוש ב-ART
 - Android run time
 - המרת ה-APK לקובץ הרצה בזמן התקנת האפליקציה ובשדרוג מ"ה
 - אין שינוי במבנה קובץ ה-APK
 - האפליקציה מופעלת מקובץ ההרצה, ללא צורך בשום הידור נוסף
- אנדרואיד 6 – ART משולב JIT
 - כדי לחסוך בזמן התקנה ובזמן שדרוג מ"ה
 - ההמרה לקובץ הרצה נעשית בזמן פנוי של המערכת
 - עד אז מבוצע JIT



עכירות?



ומה אם יש לנו נולקה בין אריות?



וירוסים מפורסמים



Stuxnet

- וירוס לתקיפת מערכות חלונות המחברים לבקרים תעשייתיים של סימנס



- כנראה יועד כנגד אירן
 - אבל הדביק גם תחנות גרעיניות ברוסיה ובעולם
 - אפילו הדביק את תחנת החלל הבינלאומית
 - כנראה דרך דיסק USB
- חיפש מחשבים עם תוכנת Step7 וחיבור ל-PLC של סימנס
 - מותנה בחיבור לציודים מסוימים, ובסיבוב בתדירים מסוימים
 - כשמצא כאלה – ניסה להשיג גרסה מעודכנת שלו
 - לא הפעיל את עצמו על מחשבים אחרים

life מטרות הדבקה

• הדבקת מחשבי חלונות

▪ במגוון דרכים

○ מתוכן שתיים דרך הדבקה מדיסק USB

▪ מטרה: לאפשר את שתי התקיפות הנוספות

• הדבקת מערכות ה-SCADA (תוכנה של סימנס)

▪ כדי להציג כאילו הציוד פועל תקין, ולשלוט על הבקרים

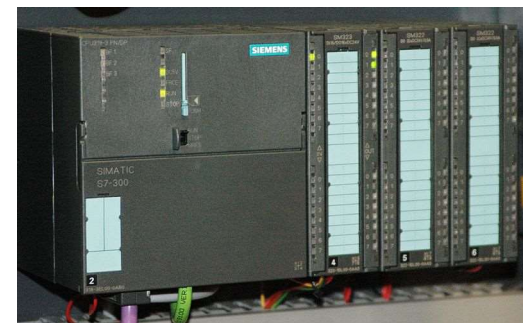
▪ על ידי התקנת DLL, ושימוש בחולשת 0-day של תוכנת סימנס

• שכתוב תוכנת בקרי ה-PLC

▪ כדי לשלוט ישירות על הציוד, ולגרום נזק לציוד



PLC של סימנס – S7-300



Stuxnet

- מחולק על פני מספר רב של קבצים במחשב!!!
- משתלט על מערכת הקבצים
 - ומסתיר קבצי LNK בגודל 4171 בתים
 - ע"י שימוש בחולשה שמאפשרת להעלים אותם מסייר חלונות
- נחתם על ידי סרטיפיקט שנגנב מ-Verisign
- מפעיל בקרים תעשייתיים של סימנס
 - ושולט על צנטריפוגות
 - מסובב אותן מהר מידי לזמנים קצרים
 - באופן שלא יזוהה מה גורם לבעייה הזו
 - עד שהן מתקלקלות



הדלקה ע"י Autorun.inf



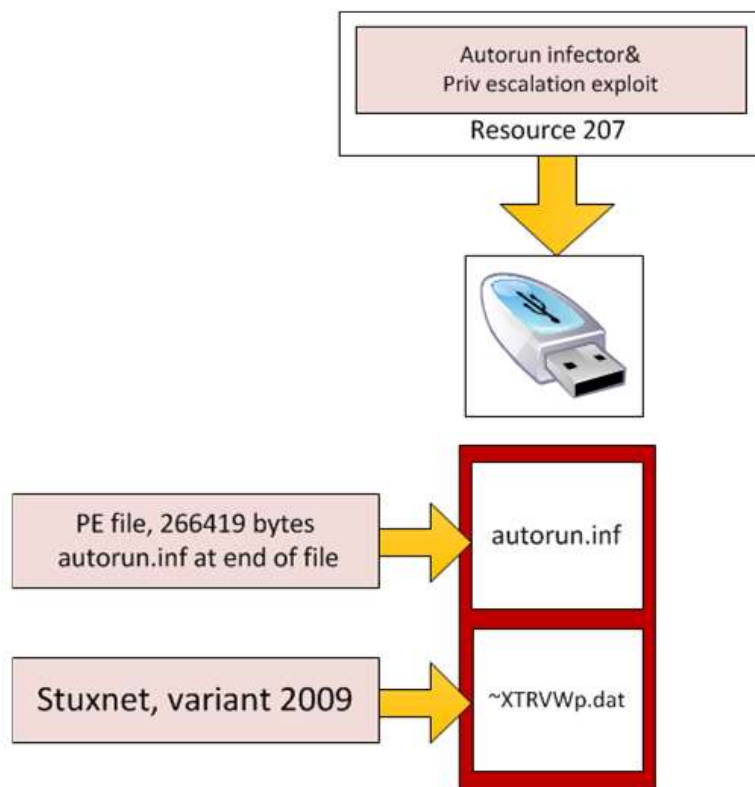
• קובץ Autorun.inf

- מכיל קובץ PE
- שבהמשכו יש נתוני autorun.inf
- באופן שהוא מורץ בעצמו כקובץ הרצה בזמן חיבור ההתקן
- אנטי-וירוס חיפש נתוני בקרה של autorun רק בהתחלה

○ ולכן לא זיהה...

• בנוסף קובץ שנראה קובץ זמני

- שם יש את התוכנה עצמה



סימול P2P

- בנוסף לגישה לשרת C&C, יש ל-Stuxnet יכולת תקשורת בין עותקים של הוירוס
 - ברשת P2P
 - Peer-to-Peer
 - מאפשר לוירוס להתעדכן אם גילה שכן עם גרסה חדשה יותר
- באמצעות רשת כזו, ניתן להתגבר על גילוי שרת ה-C&C
 - שכן ניתן להעביר פקודות, ותוכנה מעודכנת, גם כשהשרת לא פועל
 - וכשהשרת נחטף על ידי גורם זר



Flame

- וירוס שמטרתו לאסוף חומר ממחשבים, ולשלוח אותו לבעל הווירוס
 - דרך שרתי Command & Control
- אוסף חומר מסוגים שונים
 - קבצים
 - צילומי מסך
 - הקלטות אודיו מהמיקרופון
 - ועוד
- התפשט במדינות במזרח התיכון



הסולא

- Flame מחולק על פני כמה קבצי DLL
 - מקשה על זיהוי, ומקשה על דיבוג
- מסתיר את עצמו בדפי זיכרון של תהליכים אחרים
 - עם הזרקת קוד
 - והרשאות דפים שמונעות מאפליקציות ב-user mode לגשת אליהם או להריץ מהם
 - המודולים אינם מופיעים בליווח של המודולים של האפליקציות
 - וגם לא ברשימות התהליכים
- מזהה איזה תוכנות אנטי-וירוס מותקנות
 - ומתאים את עצמו כדי למנוע מהן לזהות אותו
 - למשל על ידי שינוי סיומות שמות הקבצים



מגוון ספריות תוכנה

- גודלו ענק : 20MB

- כולל מגוון ספריות לתמיכה

- בחמישה סוגי הצפנה,
- כמה סוגי דחיסה,
- SQLite,
- מכונה וירטואלית,
- ועוד

- התקנות במ"ה

- משתמש ב-registry
- מתקין דרייבר אודיו מזויף
 - משמש כעוגן נגד מחיקתו מהמחשב



תכונות נוספות

- Flame מסוגל לזהות התקני בלוטות פועלים בסביבתו
 - ולאסוף עליהם נתונים
 - להתחבר אליהם ולקבל מידע
 - כולל גרסאות מעודכנות
 - כולל מכמה קילומטרים, אם משתמשים באנטנה כיוונית מיוחדת
- מסוגל להפוך את הבלוטות שלו למזוהה
 - ולקודד מידע על הסטטוס של עצמו בפרטי הזיהוי של הבלוטות
- צילומי המסך נעשים
 - מידי פעם בפעולה רגילה
 - וכן כאשר פועלות תוכנות "מעניינות"
 - למשל תוכנות מסרים מידיים
- מסוגל למחוק עצמו לבקשת בעליו

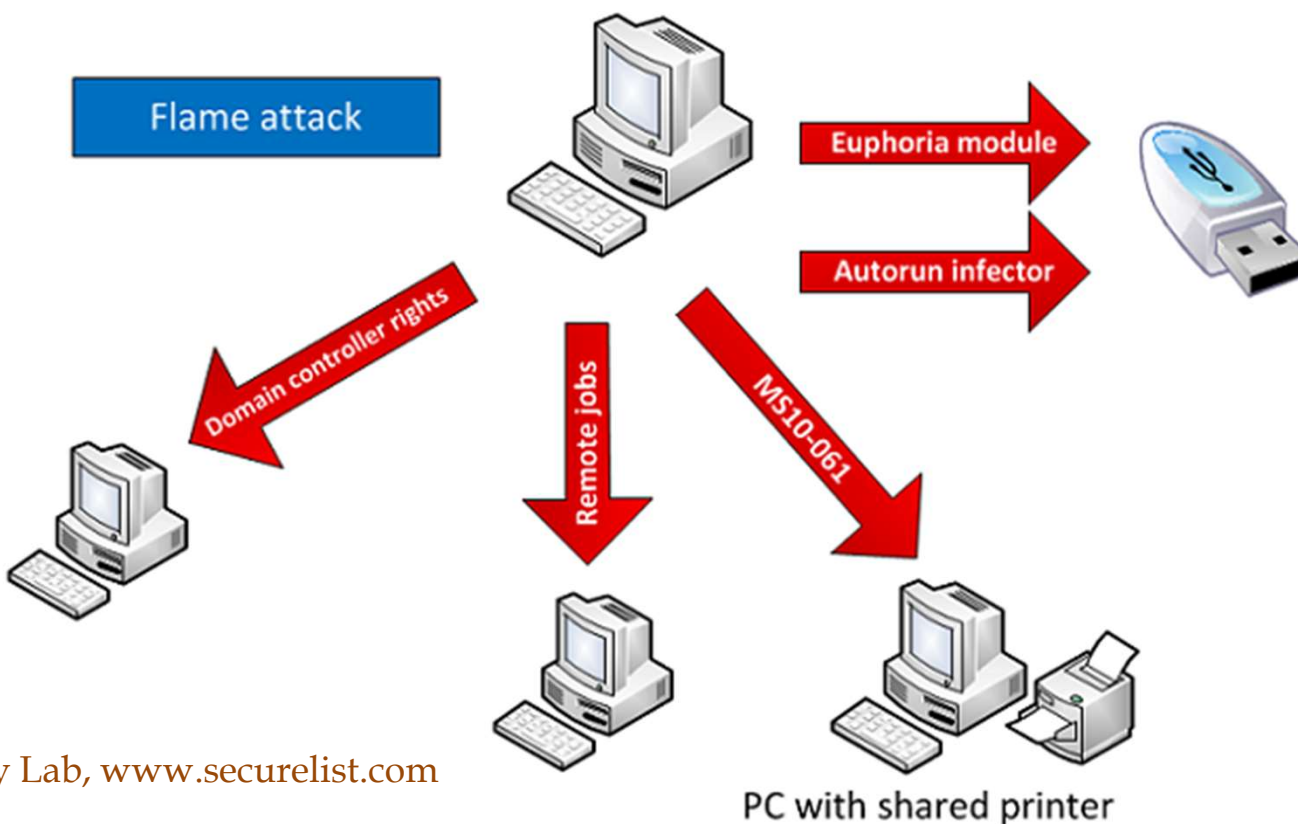


דרכי ההדבקה Flame fe

- Flame כנראה הופץ במקור על ידי התחזות לשרת windows Update
- ואז השתמש בחמש דרכי הדבקה לצורך התפשטות
 - שתיים מהן להדבקה דרך דיסק USB (לא ברור אם היו בשימוש)
 - הדבקה דרך Autorun.inf
 - כמו ב-Stuxnet, עם shell32.dll
 - Euphoria
 - בשימוש בקבצי LNK לתיקיות
 - מאפשר להפעיל קבצי הדבקה מהתיקיות
 - ושלוש מהן לשכפול ברשת המקומית
 - בשימוש בחולשה של פרוטוקול הגישה למדפסות ברשת
 - vulnerability MS10-061
 - גם היא הייתה בשימוש על ידי Stuxnet
 - Remote jobs (כלומר RPC)
 - בשימוש בהרשאות ניהול מרחוק בדומיין (אם יש לו)
 - יוצר חשבון במחשבים אחרים ומעתיק עצמו לשם



דרכי ההדמקה של Flame



אניצת אילוי צ"י זיוף סרטיפיקט

- חלונות מגנה כנגד דרייברים ותוכנות מתחזות ע"י חתימה דיגיטלית

- לכן, כדי לא להתגלות, צריך חתימה עם סרטיפיקט מורשה
 - אבל שאינו מזהה את התוקף

- Flame מתחזה לתוכנה שנחתמה דיגיטלית על ידי מיקרוסופט

- מנצל שירות חתימות לא מעודכן של מיקרוסופט
 - שעדיין השתמש ב-MD5
 - ושמאפשר חתימה על תוכנה

- דרכו הצליח לזייף סרטיפיקט לחתימה על תוכנה
 - בשימוש בהתקפה קריפטוגרפית

- לשם כך, זיוף סרטיפיקט

- בשימוש בפונקציית התמצות MD5 השבורה
- בגרסת שבירה מיוחדת של MD5 לצורך זיוף סרטיפיקטים
- הרעיון פורסם במקור על ידי חוקרים מהולנד
 - וכותבי Flame השתמשו בו לצרכיהם, עם שיפורים משלהם



הסרטיקט המלוף – ע"י שררת MD5

Version	V3
Serial number	3a ab 11 de e5 2f 1b 19 d0 56
Signature algorithm	md5RSA
Signature hash algorithm	md5
Issuer	CN = Microsoft Root Authority,OU = Microsoft Corporation,OU = Copyright (c) 1997 Microsoft Corp.
Valid from	Thursday,10 December 2009 11:55:35 AM
Valid to	Sunday,23 October 2016 6:00:00 PM
Subject	CN = Microsoft Enforced Licensing Intermediate PCA,OU = Copyright (c) 1999 Microsoft Corp.,O = Microsoft Corporation,L = Redmond,S = Washington,C = US
Public key	30 82 01 0a 02 82 01 01 00 fa c9 3f 35 cb b4 42 4c 19 a8 98 e2 f4 e6 ca c5 b2 ff e9 29 25 63 9a b7 eb b9 28 2b a7 58 1f 05 df d8 f8 cf 4a f1 92 47 15 c0 b5 e0 42 32 37 82 99 d6 4b 3a 5a d6 7a 25 2a 9b 13 8f 75 75 cb 9e 52 c6 65 ab 6a 0a b5 7f 7f 20 69 a4 59 04 2c b7 b5 eb 7f 2c 0d 82 a8 3b 10 d1 7f a3 4e 39 e0 28 2c 39 f3 78 d4 84 77 36 ba 68 0f e8 5d e5 52 e1 6c e2 78 d6 d7 c6 b9 dc 7b 08 44 ad 7d 72 ee 4a f4 d6 5a a8 59 63 f4 a0 ee f3 28 55 7d 2b 78 68 2e 79 b6 1d e6 af 69 8a 09 ba 39 88 b4 92 65 0d 12 17 09 ea 2a a4 b8 4a 8e 40 f3 74 de a4 74 e5 08 5a 25 cc 80 7a 76 2e ee ff 21 4e b0 65 6c 64 50 5c ad 8f c6 59 9b 07 3e 05 f8 e5 92 cb d9 56 1d 30 0f 72 f0 ac a8 5d 43 41 ff c9 fd 5e fa 81 cc 3b dc f0 fd 56 4c 21 7c 7f 5e ed 73 30 3a 3f f2 e8 93 8b d5 f3 cd 0e 27 14 49 67 94 ce b9 25 02 03 01 00 01
Enhance key usage	Code Signing (1.3.6.1.5.5.7.3.3) Key Pack Licenses (1.3.6.1.4.1.311.10.6.1) License Server Verification (1.3.6.1.4.1.311.10.6.2)
Authority identifier	Certificate Issuer: CN=Microsoft Root Authority, OU=Microsoft Corporation, OU=Copyright (c) 1997 Microsoft Corp. Certificate SerialNumber=00 c1 00 8b 3c 3c 88 11 d1 3e f6 63 ec df 40
Subject key identifier	6a 97 e0 c8 9f f4 49 b4 89 24 b3 e3 d1 a8 22 86 aa d4 94 43
Key usage	Digital Signature, Certificate Signing, Off-line CRL Signing, CRL Signing (86)
Basic constraints	Subject Type=CA, Path Length Constraint=None
Thumbprint algorithm	sha1
Thumbprint	2a 83 e9 02 05 91 a5 5f c6 dd ad 3f b1 02 79 4c 52 b2 4e 70



אל אי'ק מלה'ס ככל'ס?



מערכות אנטי-וירוס

- מערכות אנטי-וירוס מגנות נגד מגוון נזקות נפוצות
 - וירוסים, תולעים
 - רוגלות, תוכנות פרסום
 - סוסים טרויאניים
 - malicious Browser Helper Objects (BHOs)
 - browser hijackers, key-loggers
 - ransomware (כופרה)
 - backdoors, rootkit
 - malicious LSPs, dialers, fraudtools
- בנוסף, הן מגנות נגד בעיות אבטחה נוספות, כגון
 - הנדסה חברתית
 - עוגיות ריגול
- הן מוגבלות ביכולתן להגן נגד נזקות לא מוכרות
 - ובפרט נגד נזקות ייעודיות שנכתבו במיוחד למטרה מסוימת



אילוי נזקאות צ"י חברות האנטי-וירוס

- זיהוי הוירוסים על ידי חברות האנטי-וירוס
- ויצירת בסיס הנתונים שמשמש לזיהוי במחשבי הלקוחות

- איסוף דוגמאות של וירוסים ונוזקות
- איסוף "מודיעין"

איסוף

- תשתית ניתוח נוזקות
- ויצירת חתימות
- ופיתוח אלגוריתמי זיהוי וניקוי

ניתוח

- וכמובן כתיבת תוכנת אנטי-וירוס
- וקבצי חתימות
- מאפשר זיהוי וניקוי אצל הלקוח

פרסום

■ בסה"כ השקעה ענקית...



אילוי נזקאות ע"י חברות האנטי-וירוס איסוף

- חיפוש ברשת
- קבצים חשודים שנשלחים מלקוחות מודאגים
- מלכודות דבש
- מידע שנאסף מתוכנות אנטי-וירוס שהותקנו ע"י לקוחות
 - לקספרסקי יש מיליוני מחשבים מהם הם מקבלים מידע
- איסוף "מודיעין"
- [virustotal.com](https://www.virustotal.com)



אילוי נזקאות

• זיהוי נזקאות נעשה במגוון של אלגוריתמים היוריסטיים



- זיהוי שינויים בקובץ

- בדיקת תוכן קובץ

- זיהוי התנהגות

- בדיקת שינויים במערכת הקבצים (white list)

- וכו'

• שני מקרים עקרוניים

- **זיהוי סטטי – בדיקת קובץ ההרצה**

- למשל אחרי הורדת קובץ מהרשת

- או בהכנסת דיסק USB



- **זיהוי דינמי**

- בזמן ריצה

- וכמובן אפשר לשלב

- למשל זיהוי חשד באופן סטטי, ווידוא דינמי

- ואז מציאת חתימה לוירוס שהתגלה, וזיהויו באופן סטטי

- או קבצי הרצה באמצעות חתימות, וקבצי מערכת באמצעות white list

זיהוי חיובי ושלילי

- זיהוי מקרים נקיים
- ע"י השוואה לתוכן ידוע
- white list

זיהוי חיובי

- זיהוי מקרים חשודים
- חיפוש פעולות לא חוקיות
- מקובל באנטי-וירוס

זיהוי שלילי



מחקר ויכוסים – אנליזה Black Box

• לפעמים מספיק להבין מה תוצרי הפעולה ללא להבין בדיוק

מה נעשה

▪ שינוי קבצים אחרים

▪ שינוי ה-registry

▪ קריאה לפקודות מערכת הפעלה "משונות"

▪ תקשורת החוצה

• הרבה מאוד מניתוח הנוזקות נעשה מ"בחוץ" ולא

"מבפנים"

▪ בדרך כלל מספיק להבין את התוצרים על מנת להבין שהמדובר

בנוזקה.

▪ אז מתי בכל זאת עושים (או מנסים לעשות) ניתוח קוד?



פעילות מרכזית – רצף רקע

- בכל רגע נתון יש אלפי גישות למערכת קבצים וכמו כן לא מעט תקשורת יוצאת/נכנסת.
 - הוירוס מהווה חלק קטן מהגישות האלו.
- יש צורך להכיר את המצב הנורמלי של המערכת על מנת להצליח לזהות את הפעולות החשודות.



ניתוח סינויים במחשב

• SysInternals



▪ סט כלים ותוכנות לניתוח וניהול מחשבי windows :

Process Explorer ○

Process Monitor ○

Autoruns ○

RootkitRevealer ○

ועוד כ-60 כלים.

▪ חברה שנקנתה על ידי MS

• RegShot

▪ הקלטת מצב המערכת registry וגם קבצים והשוואה למול מצב הבא.

• WINDBG Detours או כלי מעקב אחרים על מערכת ההפעלה

▪ בדיקה אלו פונקציות מערכת הפעלה הופעלו..



דוגמא: קובץ

TextOutA	GDI32
GetStockObject	GDI32
VirtualAlloc	KERNEL32
GetCurrentProcess	KERNEL32
GetStartupInfoA	KERNEL32
GlobalAlloc	KERNEL32
GetWindowsDirectoryA	KERNEL32
GetWindowsDirectoryW	KERNEL32
lstrcatW	KERNEL32
GetProcessHeap	KERNEL32
CreateFileW	KERNEL32
CreateWindowExA	USER32
RegisterClassExA	USER32
LoadCursorA	USER32
DefWindowProcA	USER32
ShowWindow	USER32
EndPaint	USER32
BeginPaint	USER32
InvalidateRect	USER32
SendMessageA	USER32
DestroyCaret	USER32
HideCaret	USER32
ShowCaret	USER32
CreateCaret	USER32
SetCaretPos	USER32
GetFocus	USER32
MessageBoxA	USER32
ReleaseDC	USER32
GetDC	USER32
UpdateWindow	USER32
GetMessageA	USER32
TranslateMessage	USER32
DispatchMessageA	USER32
LoadIconA	USER32

■ הפונקציות אליהן
הקובץ עושה Import :

(נראה כי מדובר בתהליך עם ממשק
משתמש. לא אופייני לנוזקה)



Process Explorer

■ נשתמש ב-Process Explorer :

Process Name	Private Bytes	Working Set	Company Name	Architecture
explorer.exe	33,840 K	30,100 K	Microsoft Windows Explorer	Microsoft
vmtoolsd.exe	9,960 K	7,920 K	VMware Tools Core Service	VMware
PWRISOVM.EXE	904 K	544 K	PowerISO Virtual Drive Manager	PowerISO
ctfmon.exe	1,056 K	1,632 K	CTF Loader	Microsoft
cmd.exe	2,280 K	72 K	Windows Command Processor	Microsoft
procexp.exe	10,004 K	16,452 K	Sysinternals Process Explorer	Sysinternals
Procmon.exe	16,092 K	1,372 K	Process Monitor	Sysinternals
7faad9ed4a2e68d77a0d32ca2917e39f.exe	5,964 K	2,000 K	Назначенные задания	Kop
svchost.exe	12,596 K	5,720 K	Generic Host Process for Windows Services	Microsoft
TPAutoConnect.exe	12,392 K	4,240 K	ThinPrint AutoConnect component	Cortado
IEXPLORE.EXE	7,416 K	17,252 K	Internet Explorer	Microsoft
IEXPLORE.EXE	67,924 K	81,116 K	Internet Explorer	Microsoft

■ התהליך מפעיל את svchost.exe ונסגר.

- חלק מהנוזקות מתחזות ל-svchost.exe.
- אבל במקרה הזה הנוזקה מפעילה את ה-svchost המקורי שבא עם הווירוס.



sysinternals תנדד

- נשתמש ב-Process Monitor:

svchost.exe	4788	CreateFile	C:\Documents and Settings\Administrator\Desktop\sql\7faad9ed4a2e68d77a0d32ca2917e39f\7faad9ed4a2e68d77a0d:
svchost.exe	4788	QueryInformation...	C:\Documents and Settings\Administrator\Desktop\sql\7faad9ed4a2e68d77a0d32ca2917e39f\7faad9ed4a2e68d77a0d:
svchost.exe	4788	QueryAllInforma...	C:\Documents and Settings\Administrator\Desktop\sql\7faad9ed4a2e68d77a0d32ca2917e39f\7faad9ed4a2e68d77a0d:
svchost.exe	4788	CreateFile	C:\Documents and Settings\Administrator\Local Settings\Application Data\gapvfbsv.exe
svchost.exe	4788	CreateFile	C:\Documents and Settings\Administrator\Local Settings\Application Data
svchost.exe	4788	WriteFile	C:\Documents and Settings\Administrator\Local Settings\Application Data\gapvfbsv.exe

- **Svchost.exe** העתיק את הוירוס למקום אחר בשם אקראי.
 - שם קובץ .gapvfbsv.exe
- אפשר למצוא עוד דברים.

הדגמת sysinternals

■ נשתמש ב-Autoruns:

Winlogon		Winsock Providers		Print Monitors		LSA Providers		Network Pro	
Everything	Ligon	Explorer	Internet Explorer	Scheduled Tasks	Services	Drivers	Codecs	Boot Execute	Ima
Autorun Entry	Description	Publisher	Image Path	Timestamp					
HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Run					8/4/2013 1:46 PM				
<input checked="" type="checkbox"/>	Command Pro...		c:\windows\system32\cmdprompt.pif	5/30/2014 6:06 PM					
<input checked="" type="checkbox"/>	fpro32		c:\windows\system32\fpro32.pif	5/30/2014 6:06 PM					
<input checked="" type="checkbox"/>	Norton anti virus		c:\windows\rsv32.pif	5/30/2014 6:06 PM					
<input checked="" type="checkbox"/>	PWRISOVM.E...	PowerISO Virtual Drive Man...	c:\program files\poweriso\pwrisovm.exe	7/20/2013 3:18 PM					
<input checked="" type="checkbox"/>	VMware User ...	VMware Tools Core Service	c:\program files\vmware\vmware tools\vmtoolsd.exe	2/26/2013 5:30 AM					
HKLM\SOFTWARE\Microsoft\Active Setup\Installed Components					8/4/2013 1:30 PM				
<input checked="" type="checkbox"/>	Address Book 6	Outlook Express Setup Libr...	c:\program files\outlook express\setup50.exe	4/13/2008 9:30 PM					
<input checked="" type="checkbox"/>	Microsoft Outlo...	Outlook Express Setup Libr...	c:\program files\outlook express\setup50.exe	4/13/2008 9:30 PM					
HKCU\Software\Microsoft\Windows\CurrentVersion\Run					9/22/2013 3:00 PM				
<input checked="" type="checkbox"/>	nvjfrpia		c:\documents and settings\administrator\local settings\application data\gapvfbsv.exe	2/6/2014 4:51 PM					
HKCU\Software\Microsoft\Windows\CurrentVersion\RunOnce					3/22/2013 3:00 PM				
<input checked="" type="checkbox"/>	FlashPlayerUp...	Adobe® Flash® Player Inst...	c:\windows\system32\macromed\flash\flashutil32_11_5_502_110_activex.exe	10/29/2012 4:55 AM					
HKCU\SOFTWARE\Microsoft\Internet Explorer\Desktop\Components					8/4/2013 1:46 PM				
<input checked="" type="checkbox"/>	0		File not found: About:Home						
HKLM\Software\Classes*\ShellEx\ContextMenuHandlers					8/4/2013 4:20 PM				
<input checked="" type="checkbox"/>	ANotepad++	ShellHandler for Notepad++	c:\program files\notepad++\nppshell_05.dll	6/18/2012 6:24 PM					

■ gapvfbsv.exe רץ עם אתחול המערכת.




סיכום הדולמא

- נסיק את המסקנות הבאות:
 - הוירוס הזריק קוד ל-svchost בזמן ריצה.
 - כי רץ ה-svchost המקורי אך הוא התנהג בצורה שונה.
 - הוירוס הסתיר את ה-Import table שלו וייתכן שהיה שימוש ב-Packer.
 - הרי אנחנו יודעים שהיה שימוש ב-CreateProcess וזה לא הופיע ב-Import table.
 - הוירוס ביצע עוד פעולות:
 - העתיק את עצמו למיקום אחר בשם אקראי.
 - הגדיר את עצמו לרוץ עם אתחול המערכת.
 - ועוד פעולות שהיה ניתן לגלות באמצעות Procmon.
 - לקח לנו מספר דקות לגלות את כל זה.



ניתוח תקשורת

- כל Malware חייב לתקשר עם האינטרנט.
 - בשביל לשלוח מידע לתוקף.
 - בשביל להתפשט ברשת.
 - ושלל סיבות אחרות...
- בדרך כלל ההתנהגות של נוזקה שונה מהתנהגות של תוכנה רגילה. דוגמאות:
 - נסיון להתחבר לעשרות מחשבים ברשת ע"י סריקת פורטים.
 - שליחת מידע בשעות לא סטנדרטיות.
 - שליחת מידע בעל אופי חריג.
 - למשל ניצול חולשה.
- בעזרת ניטור חבילות המידע שנכנסות/יוצאות מן המחשב ניתן:
 - לקבל תמונה כללית על הנוזקה.
 - ויותר מזה - לגלות קיום של נוזקות.
- Wireshark היא התוכנה המובילה כיום לניטור תעבורת הרשת שעוברת במחשב מסוים. 
 - פורמט הקובץ - PCAP
- האם ניתן לנטר תעבורה באופן בטוח? מה הסימנים המעידים?



דוגמה לגלישה



- דוגמה לגלישה באתר אינטרנט (עם Wireshark):

128	19.3288070	192.168.2.102	62.219.78.115	HTTP	582 GET / HTTP/1.1
134	19.4334850	192.168.2.102	62.219.78.115	HTTP	627 GET /style.css HTTP/1.1
145	19.4584920	192.168.2.102	62.219.78.115	HTTP	638 GET /images/logo_01.gif HTTP/1.1
148	19.4588960	192.168.2.102	62.219.78.115	HTTP	638 GET /images/logo_02.gif HTTP/1.1
151	19.4606560	192.168.2.102	62.219.78.115	HTTP	638 GET /images/logo_03.gif HTTP/1.1
159	19.4643110	192.168.2.102	62.219.78.115	HTTP	633 GET /images/bg.png HTTP/1.1
181	19.5093640	62.219.78.115	192.168.2.102	HTTP	257 HTTP/1.1 200 OK (text/css)
183	19.5097570	192.168.2.102	62.219.78.115	HTTP	638 GET /images/logo_05.gif HTTP/1.1
233	19.5677730	62.219.78.115	192.168.2.102	HTTP	162 HTTP/1.1 200 OK (GIF89a)
235	19.5687120	192.168.2.102	62.219.78.115	HTTP	638 GET /images/logo_06.gif HTTP/1.1
242	19.5822000	62.219.78.115	192.168.2.102	HTTP	231 HTTP/1.1 200 OK (GIF89a)

+	Frame 128: 582 bytes on wire (4656 bits), 582 bytes captured (4656 bits) on interface 0
+	Ethernet II, Src: Giga-Byt_97:a0:7a (90:2b:34:97:a0:7a), Dst: EdimaxTe_c3:27:ae (00:0e:2e:c3:27:ae)
+	Internet Protocol Version 4, Src: 192.168.2.102 (192.168.2.102), Dst: 62.219.78.115 (62.219.78.115)
+	Transmission Control Protocol, Src Port: 55766 (55766), Dst Port: http (80), Seq: 1, Ack: 1, Len: 52
-	Hypertext Transfer Protocol
+	GET / HTTP/1.1\r\n
	Host: www.trythis0ne.com\r\n
	Connection: keep-alive\r\n
	Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8\r\n
	User-Agent: Mozilla/5.0 (windows NT 6.2; wow64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/28.0.
	Accept-Encoding: gzip,deflate,sdch\r\n
	Accept-Language: he-IL,he;q=0.8,en-US;q=0.6,en;q=0.4\r\n
	Cookie: __utma=187449709.1628208541.1390075373.1397578245.1398548363.8; __utmz=187449709.139673241.
	\r\n



נזקה ברשת

- נזקה כאמור, מתנהגת שונה מתוכנה רגילה.
- דוגמאות לפעולות חשודות של נזקה:
 - סריקת מחשבים – אם הסריקה היא בתוך הרשת אנחנו נראה המון בקשות ARP יוצאות מהמחשב (הדגמה בהמשך).
 - סריקת פורטים – נראה התחברות לאותו IP אך עם פורטים שונים.
 - בדרך כלל חיבורים שנכשלים מרמזים על פעולה חשודה. שכן תוכנה רגילה לא מנחשת לאילו פורטים להתחבר.
 - בקשות HTTP מוזרות – למשל שליחת כמויות מסוימות של מידע לשרת C&C.
 - שימוש בפרוטוקול מסוים בצורה חריגה.
 - יש צורך להכיר מאות פרוטוקולים בשביל להגדיר מה זה חריג, אבל ההיכרות הכרחית.
 - ועוד



זיהוי התנהגות Sandbox-ק

- מערכת זיהוי התנהגות (Sandbox) אוספת מגוון נתונים ואירועים ממהלך ריצה
 - ובודקת אם הם צפויים או לא
 - כלומר אם התוכנה שרצה מבצעת סדרת אירועים כזו
 - או שתוכנה שאינה נוזקה צפויה גם כן לבצע סדרה כזו
- אירועים שנאספים יכולים לכלול
 - קריאה לקריאות מ"ה חשודות
 - כמו כתיבה לזיכרון של תהליך אחר
 - פתיחת קבצים
 - למשל כתיבה לקובץ הרצה אחר
 - יצירת תהליכים חדשים
 - שינוי בווקטור הפסיקות
 - ועוד – דוגמאות?
- דוגמא - Cuckoo



עכירות?



ונניח ומצאתי ויכוס – מה עכשיו?

או מה יש באנטי ויכוס *efi*?



חתימות fe וירוסים

• חתימה של וירוס: **מחרוזת ייחודית המופיעה בו, ולא**



מופיעה בתוכנות אחרות

- השיטה הקלאסית לזיהוי וירוסים ונוזקות היא על ידי זיהוי ה"חתימות" שלהם
- יצור החתימות דורש השוואה לכל התוכנות המוכרות!
 - היו מקרים שיצרני אנטי-וירוס בחרו בטעות חתימות שכן מופיעות בתוכנות אחרות, למשל WORD או מ"ה Windows עצמה
- שיפור: המחרוזת נבחרת כביטוי רגולרי
 - למה זה משפר?

• סריקת כל הקבצים במערכת

- וגם כל התכניות בזיכרון
 - כולל בזמן טעינת תוכנית חדשה
 - וכשהיא כבר רצה (למשל, אחרי שוירוס פענח את הקוד שלו)



חתימות *fe* וירוסים

- אנטי וירוסים מקבלים מידי פעם קובץ חתימות חדש
 - שישמש מאותו רגע להשוואה עם כל הקבצים במערכת
- יצרן האנטי-וירוס מחשב אותו על סמך
 - כל הוירוסים המוכרים
 - וכל התוכנות החוקיות המוכרות
- הבדיקה נעשית מול התוכן הנוכחי בלבד
 - אין השוואה לתוכן היסטורי
 - למשל לגרסה מקורית של הקבצים
 - בדיקה מאד יעילה (מעבר אחד מהיר על כל קובץ)



אזכוריתאים היוריסטיים להאנה

- מסוגלים לזהות וירוסים לא מוכרים
 - אבל בלי לדעת מה הם מבצעים
- טכנולוגיה – חיפוש היוריסטי בקוד
 - חיפוש פעולות מחשירות בקוד
 - חיפוש מחרוזות קוד מחשירות
 - מחרוזות שוירוסים נוטים להפעיל
 - ביצוע RE של קוד באופן אוטומטי
 - כדי לזהות פעולות חשודות
- עלולים לטעות...
 - תוכנה חוקית עשויה להפעיל קוד מחשיר



חסימת התנהלות אסורה

- תוכנות אנטי-וירוס יכולות לחסום קריאה לקריאות מ"ה
 - או פונקציות ספריה מסוימות
 - כשצפוי שתוכנה חוקית לא תקרא להן
- בזמן קריאה לקריאות מ"ה כאלה
 - או אם קריאות מסוימות מופיעות ב-Import Table
- האנטי-וירוס יעצור את התוכנית
 - או יפתח חלון עם שאלה למשתמש
 - או ישלח דוגמא "הביתה"



White List

- במערכות מסוימות התוכנות להן מותר לרוץ ידועות מראש
 - למשל מערכות ייעודיות, בהן המשתמש לא מתקין תוכנה נוספת
 - או שהתקנות הן נדירות
- במקרים כאלה, במקום לחפש וירוסים ונוזקות
 - נבדוק אם הקבצים במערכת זהים למקוריים שהיו בהתקנה
 - כלומר ששום קובץ לא השתנה
 - ולא נוסף שום קובץ חדש
 - **השוואה לתוכן היסטורי**
 - **השתנה או לא השתנה?**
- ניתן לעשות זאת ביעילות בעזרת פונקציות תמצות
 - מספיק להשוות תמציות
 - ואין צורך להשוות לעותק מקורי מלא
- תוכנות: Tripwire, McAfee ePO ואחרות



IPS/IDS

- Intrusion prevention/detection systems
- רכיבים שמטרתם לנטר את התעבורה ולחפש פעולות שנחשבות זדוניות.
- יכולים לתפקד כרכיבים נפרדים או בתוך רכיבים קיימים (למשל ראוטרים).
- IPS מנסה לחסום את הפעולות האלו בעוד IDS תפקידו רק לדווח.
- איך זה עובד?
- זיהוי סריקות או מתקפות רשת (מתקפות על פרוטוקלים).
- חתימות על חבילות שמוגדרות כחשודות.
- למשל חתימה על אקספלויט מסוים.
- ניתוח פעולות חריגות (אנומליות).
- כלומר ללמוד איך הרשת צריכה להתנהג במצב נורמלי ולזהות פעולות שחורגות מן המצב הנורמלי. כל רשת מתנהגת בצורה שונה.
- ועוד שיטות



אנטי-וירוס - סיכום

ניתוח: בדיקה ואילו

- ניתוח : בדיקה וגילוי
 - על ידי ניתוח אנושי
 - מערכות אוטומטיות הבודקות פרמטרים חשודים
 - שינוי קבצים, הדבקת קבצים אחרים, פעילויות לא מקובלות

• טכנולוגיות בשימוש

- ניתוח סטטי ודינמי
- הרצה במכונות וירטואליות
 - מונע גרימת נזק, קל לבטל הנזקים
- אמולציה
 - תוך חיפוש פעולות שתוכנות רגילות לא נוהגות לבצע

• מטרה

- מציאת מאפיינים ייחודיים על פיהם תוכנות ה-AV יפעלו
- העדפה מובנית למניעת false-positives מלמציאת יותר נזקות

