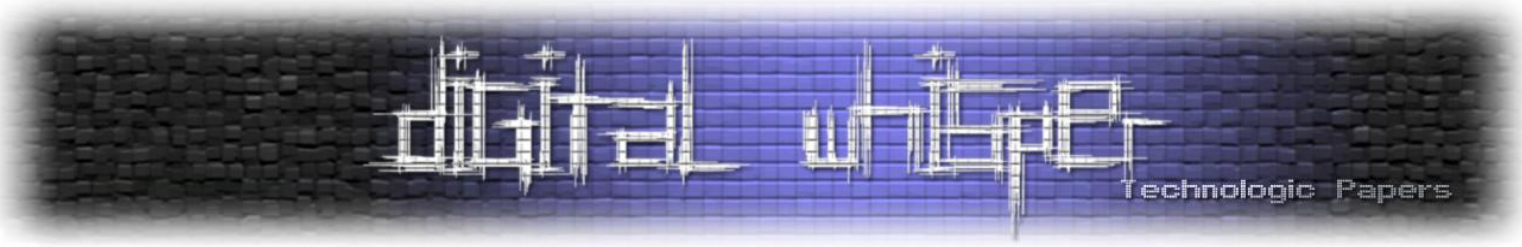


Digital Whisper

גליון 31, אפריל 2012

מערכת המגזין:

מייסדים:	אפיק קסטיאל, ניר אדר
מוביל הפרוייקט:	אפיק קסטיאל
עורכים:	ניר אדר, אפיק קסטיאל, שילה ספרה מלר
כתבים:	ניר גלאון, עו"ד יהונתן קלינגר, אלעד גבאי, לירן בנודיס, עידן פסט ואלכס ליפמן.

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

דבר העורכים

אז אנונימוס (עדיין?) לא הצליחו להפיל את רשת האינטרנט, מה שאומר שבינתיים יש לנו זמן לפרסם גליון Digital Whisper נוסף. מלבד זה שהאינטרנט המשיך לזרום החודש, עברו עלינו לא מעט אירועים לאחרונה, האירוע החשוב לציון (לפחות לדעתי) הוא יום העצמאות.

שלא כמו בחצי מהעיתונים בארץ, אני לא מתכוון לכתוב עכשיו דברים בסיגנון "64 עובדות על הישראלים" / "64 דרכים לדעת שאתה ישראלי אמיתי" וכו', אבל כן הייתי להגיד דבר קטן בהקשר הזה בדברי הפתיחה של הגליון:

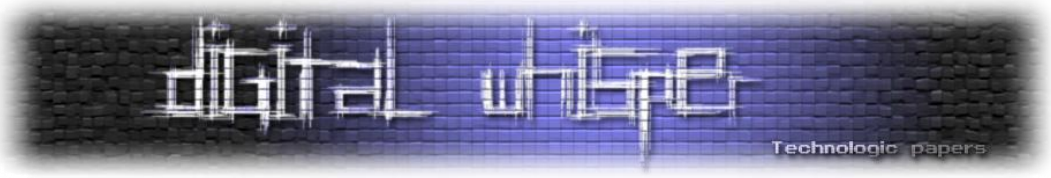
באחד הערבים החודש אכלתי עם ניר ארוחת ערב והוא שאל אותי אם אכפת לי מהמדינה שלנו, בלי לחשוב עניתי לו בחיוב, והוספתי: "נו, בדיוק הייתי בחודש מילואים לפני לא יותר מדי זמן, אתה שואל את הבן אדם הלא נכון...", אחרי המשפט הזה נאמרו עוד מספר בודד של משפטים באותו הקשר ועברנו לדבר על נושאים אחרים. כמה ימים לאחר מכן חשבתי שוב על השאלה הזאת, וממנה התחלתי לחשוב על העניין של חגיגת יום העצמאות ומה הסיבה מאחורי הרעיון של חגיגת עצמאות בתור מדינה שלמה.

ברור לי שאני לא מחדש הרבה, או בכלל, אבל לדעתי שווה לכל אחד לעצור ולחשוב על זה, שכל כך הרבה שנים, כל כך הרבה יהודים נלחמו בכל כך הרבה אויבים שונים על מנת שמדינת ישראל תקום. וגם אחרי שהמדינה קמה, ברצף ההיסטורי שלנו קשה למנות שנים בהן ידענו שקט. מטורף לקרוא את מגילת העצמאות, לעבור על המשפטים המרכיבים אותה, לחשוב על רצף האירועים שהתרחשו מאז ועד היום, ולקום בבוקר לעבודה מבלי לעצור ולחשוב על העובדה שאנחנו עדיין כאן, קמים בבוקר והולכים לעבודה. זה לא ברור מאליו.

אז כן, נכון, יש בתוכנו המון חילוקי דעות, חילוקי דעות בין ימניים לשמאלנים, בין חרדים לחילוניים, בין כל כך הרבה אנשים לכל כך הרבה אנשים אחרים ובלי סוף, אבל, ואולי זה ישמע קצת נדוש, זה שיש לנו את היכולת הזאת של להתווכח אחד עם השני, להביע את הדעה שלנו בציבור מבלי לפחד שלמחרת נקום ונגלה את החנויות שלנו שרופות או שהילדים שלנו חזרו חבולים מבית הספר, זה דבר שאף יהודי לא חלם עליו לפני פחות ממאה שנה.

אז שבוע אחרי, נאחל לכולכם חג עצמאות שמח!

אפיק קסטיאל וניר אדר.



תוכן עניינים

מנגנוני אבטחה באנדרואיד

מאת ניר גלאון

הקדמה

''יש המון וירוסים באנדרואיד" אמר לי חבר לפני כמה ימים, וזה גרם לי לחשוב כי רוב הציבור (הלא מבין) נוהג לחשוב על מערכות פתוחות כמערכות קלות לחדירה וכיוצא בזה כך גם חושבים על אנדרואיד.

המאמר הבא יעסוק בשאלה "האם אנדרואיד היא פלטפורמה מאובטחת או פרוצה", וכדי לענות על השאלה נתקוף את הנושא מכמה זוויות (מודל האבטחה של המערכת, מפתחים והרשאות, ממה המשתמשים צריכים לדאוג וכדו').

למה שיהיה לנו אכפת מאבטחה?

זו שאלת השאלות. לרוב המשתמשים לא אכפת מאבטחת המידע של המכשיר שלהם לפחות עד שהוא נפרץ/נגנב/נאבד, ואז אנו כבר רוצים לדעת שהמכשיר מוגן ולא שכל מי שימצא אותו יוכל לגשת למידע הנמצא עליו.

- אנדרואיד הינה הפלטפורמה הנפוצה ביותר כרגע, היא גדלה מיום ליום וכמו מערכות אחרות בעלות משתמשים רבים, קהילת ההאקרים (הלא טובים) יתמקדו באנדרואיד. הרי כלכלי יותר למצוא פרצה / לכתוב סוס טרויאני למערכת בעלת הרבה משתמשים כדי שנוכל למקסם את הניצול של מה שכתבנו.

- תאגידים גדולים משתמשים היום באנדרואיד. העובדים מחזיקים מכשיר אנדרואידי ומסנכרנים את המייל, מעבירים מצגות וכדו' (בקיצור פותחים חור בהגנה של הארגון).

- כולנו שומעים מידי יום על הרוגלות החדשות שמוצאים ~~בשרקש~~ ב-PlayStore ואני בטוח שאם נכתוב בגוגל Android security נמצא המון כתבות.

יסודות האבטחה של אנדרואיד (או על מה נעבור?):

1. בידוד אפליקציות ומערכת אישורים (בעת התקנת אפליקציות):
   - האם אנחנו יכולים לשלוט על מה האפליקציות שאנו מתקינים יוכלו לעשות?
   - האם אפליקציות שהתגלו כרוגלות יכולות להשפיע על שאר המערכת?
2. "מחבר" האפליקציה:
   - האם אנחנו יכולים לסמוך על כותב האפליקציה?
   - האם אנחנו יכולים לסמוך על האפליקציות שכבר התקנו?
3. קידוד והצפנת המידע:
   - האם המידע שלנו מוגן אם המכשיר שלנו נפרץ / נגנב / נאבד?
4. מרכז הגישה למכשיר

לפני שנתחיל לעבור על הדברים אני ממליץ להכיר באופן בסיסי את הקרנל של המערכת, ניתן לקרוא בקישור הבא:

http://nirgn-startup.blogspot.com/2012/01/android.html

בידוד אפליקציות

כדי להדגים את הנושא נחבר את המכשיר למחשב (בעזרת ה-USB), ניכנס ל-ADB. נרשום PS ונראה את התהליכים הרצים. קיימים הרבה תהליכים רצים (כ-ROOT, דרך אגב), לדוגמה:

- void
- neld
- installd
- service manger
- Zygole
- system_server
- com.foo.app1
- com.bar.app2
- ועוד

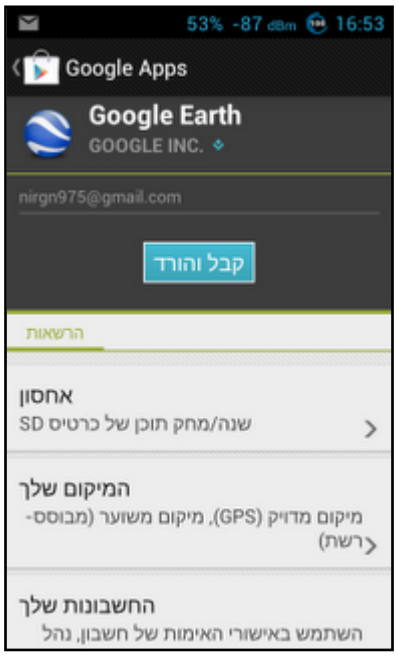
(ה-2 האחרונות הן כבר אפליקציות). ניתן לראות כי הם רצים כתהליכים נפרדים ואינם חולקים (ישירות) זיכרון / קבצי מערכת, אין להם הרשאה לקבצים אחד של השני / לרכיבים אחד של השני וגם אינם יכולים "לדבר" אחד עם השני.

הקרנל של מערכת ההפעלה (ה-"Linux Kernel") מספק את הבידוד (הריצה של כל אפליקציה כתהליך נפרד), פרט נוסף ומעניין (עליו לא נרחיב כעת), הוא שכל אפליקציה מריצה בעצמה Dalvik VM.

מה אנו לומדים מכאן?

- באופן מובנה כל אפליקציה רצה על תהליך נפרד ועם הרשאות מערכת נפרדות. הרעיון הזה אינו חדש ומבוסס על רעיון ישן ומובן שמתבצע במערכות UNIX ככה שאנדרואיד למעשה רוכבת על רעיון שנמצא כבר שנים (ועובד) עליו אנחנו יכולים לסמוך.
- שירותי מערכת (Framework) גם רצים כתהליכים נפרדים (system_server).
- הקרנל של מערכת ההפעלה הוא למעשה הבסיס של בידוד האפליקציות, או בניסוח קצת יותר מקצועי: App Sandbox.
- Dalvik לא מהווה תפקיד (בשונה מ-javaME בו האפליקציות רצו על אותה "מכונה וירטואלית", פה כל אפליקציה רצה על מכונה וירטואלית נפרדת וכולן צריכות לעמוד באותן הרשאות לא משנה אם הקוד נכתב ב-C++ או בג'אווה).
- כל הדברים האלו כוללים גם את אפליקציות המערכת המובנות.

אז איך מפתחים בורחים מ'ארגז החול'?



הרי בפועל אנחנו לא מרגישים שיש ארגז חול, לדוגמה באפליקציית פייסבוק אנו מושכים תמונה מהגלריה, או כשאנו עושים צק'-אין אנו משתמשים באפליקציית ה-GPS. אז רגע, יש ארגז חול או אין ארגז חול?

ארגז החול קיים, אך ניתן להתגבר עליו ע"י בקשת הרשאות. כל פעם שאנו לוחצים על מקש ה"הורד" כדי להוריד ולהתקין אפליקציה נפתח לנו מסך ההרשאות, המסך שעליו אנו רגילים לדלג ולהקיש ישר "קבל והורד" בלי לקרוא איזה הרשאות מיוחדות אנו נותנים לאפליקציה.

ההרשאות שאנו רואים שם אלה קבוצות הרשאות. לדוגמה, רוב האפליקציות מבקשות את ההרשאה "תקשורת רשת" ההרשאה הנ"ל היא קבוצת הרשאות שמכילה בתוכן את כל ההרשאות לאפליקציות

שקשורות לאינטרנט, (או יותר נכון לפתיחת שקעי אינטרנט) לדוגמה האפשרות ליצור שקעי אינטרנט דרך WIFI וגם האפשרות ליצור שקעי אינטרנט דרך האינטרנט הסלולרי.

דוגמה נוספת היא ההרשאה למצלמה (כקוד, זה נראה כך: android.permission.CAMERA), הרשאה נותנת לנו את הגישה למצלמת הסטילס, הוידאו, לאפקטים של המצלמה ולמצב פנורמה. (בקשות שנחשבות "נורמליות" נמצאות למטה תחת "הצג הכל"), דוגמאות נוספות להרשאות שאפליקציות יכולות לקבל:

ACCESS_FINE_LOCATION	לאפליקציה תנתן הגישה לנתוני המיקום של האנדרואיד
ACCESS_NETWORK_STATE	לאפליקציה תנתן הגישה לנתוני קישוריות הרשת
ACCESS_WIFI_STATE	לאפליקציה תנתן הגישה לנתוני קישוריות רשתות ה-WIFI הנמצאות בקרבת האנדרואיד.
CHANGE_WIFI_STATE	לאפליקציה תנתן הגישה לשנות את הקישוריות לרשתות ה-WIFI של האנדרואיד
BATTERY_STATS	לאפליקציה תנתן הגישה לנתוני מצב הסוללה של האנדרואיד
BRICK	לאפליקציה תנתן הגישה לנטרל את האנדרואיד לגמרי
READ_SMS	לאפליקציה תנתן הגישה לקרוא הודעות SMS שנשלחו / התקבלו באנדרואיד

לרשימה נרחבת של כל ההרשאות הקיימות ומה כל אחת מאפשרת:
http://developer.android.com/reference/android/Manifest.permission.html

השאלה המתבקשת היא "האם משתמש רגיל יכול להבחין האם אוסף הבקשות האלו בטוח או לא?"
ההיגיון של גוגל (או יותר נכון צוות האנדרואיד) הוא "בואו ניתן למשתמש הקצה את הכוח להחליט" (דבר שבא גם לידי ביטוי בכל הקשור לפתיחות המערכת), אבל כמו שאנחנו לומדים בשטח משתמשים רגילים פשוט לוחצים "הבא > הבא > הבא" בלי לקרוא.

אז מה החלופות?
החלופה נקראת "הרשאה דינמית", (שזה דרך אגב המצב ב-iOS), אבל לא הכל מושלם:

- זה יציק לנו אם כל פעם תיפתח תיבה קטנה שתבקש אישור.
- זה מקשה על מעבר בין אפליקציות (שזה אחד הדברים העיקריים ששמים עליו דגש בצוות האנדרואיד-שהמעבר יהיה כמה שיותר נקי וחלק) אם כל פעם שניכנס לאפליקציה נצטרך לתת לה הרשאות להכל מחדש.

- זה יוביל למצב של "הבא > הבא > הבא" גם אצל משתמשים שבדרך כלל כן שמים דגש על הנושא.
- אין היגיון בנושא, הרי כבר הורדנו את האפליקציה, "התחייבנו" אליה. אין שום סיבה שלא נרצה לתת לה את האישורים לפעול בצורה תקינה.

אז מה עושים?

צוות האנדרואיד יצר במרקט מערכת תגובות ודירוג. על מנת שנוכל לראות את תגובות המשתמשים ודירוג האפליקציה, דבר שיעזור למשתמשים רגילים לגבש דעה על האפליקציה טרום ההתקנה. אבל שוב, לא הכל מושלם והגענו למצב שבו המשתמשים פשוט מסתכלים על כמות ההורדות של האפליקציה ומניחים שאם הורידו אותה הרבה פעמים אז היא בטח בטוחה.

אז בסוף נגיע לקרן אור (אפשר גם להסתכל עליה אחרת) והיא שגוגל שמרה לעצמה את הזכות "להרוג" מרחוק אפליקציות שהיא מחליטה שהן מזיקות, ז"א שבמידה וכבר הורדנו אפליקציה שהתגלתה כמזיקה וירדה מ-Google Play, גוגל יכולים להרוג אותה מרחוק במכשיר שלנו וככה אנו לא צריכים להיות כל הזמן מעודכנים באפליקציות שהתגלו כמזיקות (אם אפליקציה תוסר מהמכשיר בצורה כזאת המשתמש יקבל מייל מגוגל).

אך בסופו של יום המשתמש אחראי למה שהולך אצלו במכשיר.

מפתח האפליקציה

וזה מוביל אותנו לכמה שאלות בהקשר למקור של היישום:

האם אנו יכולים לסמוך על המפתח של האפליקציה שאנו עומדים להתקין?

התשובה היא כנראה לא. אחת הדרכים שבהם אנדרואיד מתמודדת עם השאלה הזו היא שאפליקציות חייבות להיות חתומות וברגע שמתגלה אפליקציה מזיקה גוגל יכולה להסיר את כל האפליקציות מהמרקט בעלות אותה חותמת (ז"א כל האפליקציות שהגיעו מאותו המפתח, אפילו אם העלה כל אחת מחשבון מפתחים אחר).

החתימה מבוססת על אלגוריתם פרטי-ציבורי מוצפן בו בעיקרון יש לנו מפתח פרטי שמשמש אותנו לחתימה דיגיטלית של האפליקציה והמפתח הציבורי שאחראי על כתיבת התעודה מחוברים ביחד לקובץ ה-APK (החתימה נמצאת בתיקייה Meta). לא נרחיב על איך החותמת בנויה בדיוק אך לקריאה מעמיקה בנושא לחצו כאן. השורה התחתונה היא שכל אחד יכול לחתום את האפליקציה שלו גם אם היא רוגלה זדונית.

האם אנחנו יכולים לסמוך על כך שהאפליקציות שלנו עמידות לחבלה ברגע שהותקנו ואף אחד לא יוכל לשחק בקוד שלהן לאחר שהן הותקנו אצלנו במכשיר?

פה לעומת זאת התשובה היא כנראה כן, התפקיד של החתימה בנושא הזה הינו משמעותי. האפליקציות המתעדכנות (אם באופן ידני או אוטומטי) מסירות באופן אוטומטי את הגרסה הקודמת ומתקינות את הגרסה החדשה (ז"א APK חדש או אם תרצו אפליקציה חדשה לגמרי), אז מאיפה ניתן לדעת שלא החליפו את ה-APK ברוגלה כזאת או אחרת? הסיבה לכך היא החתימה, החתימה של ה-APK החדש חייבת להיות זהה לחתימה של ה-APK הישן ובכך נמנעת כמעט לחלוטין האפשרות להתקנת APK ממקור זר.

קידוד המידע

באופן דיפולטיבי כל הקבצים של האפליקציות מוגדרים כ-"פרטיים" (זאת אומרת שאפליקציות אחרות לא יכולות לתמרן או לערוך את הקבצים ישירות), למעט:

- היוצר של האפליקציה יכול ליצור קבצים שיהיו מוגדרים כ-"MODE_WORLD_READABLE" ו/או "MODE_WORLD_WRITABLE".
- אפליקציות שלהן אותה תעודה + חתימה - ז"א שנוצרו ע"י אותו היוצר (בגלל שהן רצות על אותו UID הן יכולות לחלוק קבצים משותפים).
- קבצים שאנו שמים ב-SD, שלשם כל אפליקציה יכולה לגשת (כמובן בעזרת הרשאה ספציפית).

אלו הצפנות קיימות לנו במערכת האנדרואיד (את חלקם אני מכיר יותר וחלקם פחות אך לא אסביר על כל אחת מחוסר זמן, מקום וכי לא זה הנושא שלנו. אם מישהו מתעניין בפרוטוקול / תקן ספציפי הוא יותר ממוזמן לגשת לגוגל / ויקיפדיה):

- VPN טלפוני ואינטרנטי (לאירגונים) (עם IPsec והצפנת 3DES ו-AES ומערכת אישורים-CA).
- wifi עם תקן 802.11 ופרוטוקלי אבטחה WPA/2.
- פרוטוקול OpenSSL.
- JCE (מבוסס על - Bouncy Castle) מספק לנו תמיכה בכל האלגוריתמים הנפוצים (הצפנות ציבוריות וסימטריות
- Apache HTTP (תומך ב-SSL).
- Keychain API - אפליקציות יכולות להתקין ולאחסן תעודות אבטחה של המשתמש.
- הצפנת כל הזיכרון (זמין מגרסה 3.0 ומעלה).

הצפנת כל הזיכרון:

בכדי לבצע פעולה זו ניכנס ל-"הגדרות" > "אבטחה" ב-ICS (או "מיקום ואבטחה" בגרסאות נמוכות יותר) > "הצפנה".

כמה חידודים:

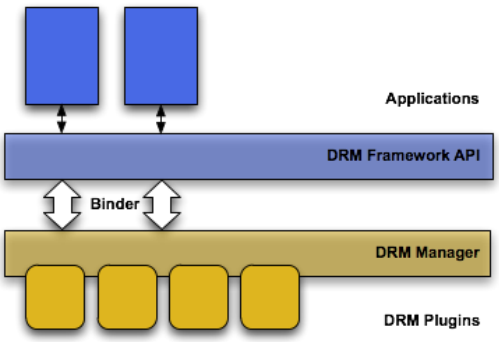
- הפעולה הנ"ל דורשת סיסמת מסך (לא סיסמת "צורה").
- הפעולה מצפינה את כל המידע שנמצא בזיכרון עם AES 128bi ו-CBC ועם SHA 256bit: ESSIV (הסיסמה משולבת בהצפנת SHA).
- ביטול ההצפנה דורשת איפוס מלא של המכשיר!
- ההצפנה מבטלת את אופציית ה-"האצת-חומרה" בקריאה וכתיבה של נתונים (ההערכות הן שפעולה זו מפחיתה בכ-54% את הביצועים של המכשיר בקריאה וכתיבה של קבצים גדולים).

החסרונות הם:

- פעולה זו אינה מונעת התקפות פיזיות, ז"א שאם גנבו לכם את המכשיר פיזית, התוקף יכול להתקין key logger על מנת לשחזר/לפענח את הסיסמה שלכם.
- פעולה זו אינה מונעת התקפות שנקראות Cold-Boot Attacks, אם מישהו יצליח להחזיק את המכשיר שלכם כשהוא דולק וכשהסיסמה כבר פועלת (אפילו אם יש נעילת מסך) הם יוכלו (בעיקרון) להוציא את הזיכרון RAM (לא אומר שזה קל אבל בהחלט אפשרי) לשים אותו במכשיר אחר, להדליק ולחלץ את המידע מה-RAM למחשב וכד'. בעיקרון כל עוד הכנסתם את הסיסמה והיא בזיכרון, ניתן לשחזר אותה.

DRM זכויות ניהול דיגיטליות.

ראשי תיבות: Digital Right Management. ב-API 11 אנדרואיד הוסיפה את האפשרות ל'זכויות ניהול דיגיטליות' - זהו ממשק המספק למכשירי OEM אישור לתוכן מסוים. הרעיון הוא להגן על התוכן של המשתמש באמצעות שימוש ברישיונות דיגיטליים לשימוש בתכנים. מה שמספק את ההגנה/החוצץ הזה הוא ה-Binder.



ה-Binder מספק תקשורת בין תהליכים, באמצעות מודל מוקטן של הקרנל. ה-Binder (כמו הפעילות הסטנדרטית של ה-IPC בלינוקס קרנל) הוא כמו שליח בין תהליכים,

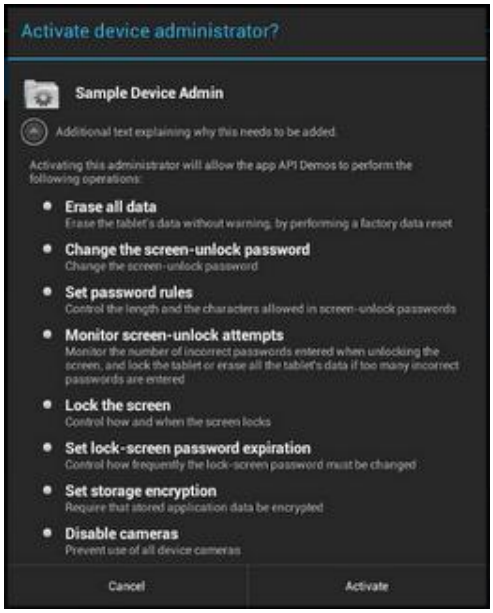
הוא יודע לזהות האם מדובר באיום ו'רץ' בין התהליכים כדי להעביר מידע (על מנת שלא תיהיה תקשורת ישירה). לקריאה מעמיקה יותר על ה-Binder ניתן לקרוא בקישור הבא:

http://www.nds.rub.de/media/attachments/files/2012/03/binder.pdf

DRM framework API: ה-API חשוף לאפליקציות (באמצעות ה-Framework של אנדרואיד) ורץ באמצעות ה-Dalvik בשביל אפליקציות סטנדרטיות.

DRM manager: חושף את ממשק הניהול בשביל פלאגינים (נקראים גם: "סוכנים") ומבצע ניהול רישיונות ופענוח עבור התוכניות.

מרכז הגישה של המכשיר



מרכז הגישה של המכשיר כולל נעילת המסך על ידי סיסמה / קוד Pin / צורה תבנית. בנוסף בגרסה 2.2 (פרויו) התווספה לה אפשרות ה-"אדמין".

למה זה טוב? למשתמש הפשוט זה לא עוזר, לאירגונים זה הכרחי. האירגונים יכולים לשלוח לעובדים אפליקציה שמבקשת להיות מנהל המכשיר (Device Administrator) ונותנים לה שליטה לבצע הכל, החל מלהכריח את המשתמש להשתמש בסיסמה למסך הנעילה ועד נעילת המצלמה כדי שלא יוכלו להשתמש בה (אפילו נעילה של אפליקציית המצלמה בשעות מסויימת, לדוגמה בשעות העבודה, 17-8).

האפשרות הזאת נועדה להגן על המידע של הארגון הנגיש או הנמצא על המכשיר. עוד דוגמה: במידה והמכשיר נגנב הארגון יכול לשלוח הוראה למכשיר למחוק את עצמו.

סינון רוגלות

צוות אנדרואיד חשף בתחילת פברואר שירות חדש בשם Bouncer, שסורק בצורה אוטומטית את המרקט בחיפוש אחר רוגלות ותוכנות זדוניות. השירות מבצע את התהליך ברקע בלי שהמשתמשים או המפתחים מרגישים.

איך השירות עובד:

1. כשמעלים אפליקציה השירות מתחיל לנתח אותה בהשוואה לתוכנות זדוניות וסוסים טרויאנים מוכרים.
2. לאחר מכן השירות מנתח את "התנהגות" האפליקציה ומשווה אותה להתנהגות אפליקציות שנותחו בעבר והתגלו כזדוניות/סוסים טרויאנים.
3. השירות מבצע ריצה של האפליקציה בענן מיוחד של גוגל ומדמה כיצד האפליקציה תעבוד על מכשיר אנדרואיד.
4. בנוסף, השירות מבצע ניתוח של חשבונות מפתחים חדשים כדי למנוע ממפתחי אפליקציות זדוניות לחזור ולהעלות את אותם האפליקציות שוב שוב או אפליקציות חדשות.

יש לציין כי השירות קיים כבר משנת 2011 אך לא נחשף לציבור ובצוות של אנדרואיד מדווחים על ירידה של 40% בהופעת תוכנות זדוניות.

בנוסף, באותו הזמן חברות אבטחה דוגמת Lookout, AVG, Symantec וכד' דיווחו כי ישנה עליה בכמות הרוגלות המופיעות במרקט והמליצו להתקין את תוכנות האבטחה שלהם.

סיכום

צוות אנדרואיד מבצע עידכונים למנגנונים הנוכחיים כל הזמן ומכניס חדשים (דוגמת המסנן רוגלות שנכנס לאחרונה). אפשר לשים לב שנושא האבטחה נמצא על סדר היום אצל גוגל אך בסופו של יום אין מערכת החסינה לחלוטין בנושא אבטחה. כמובן שעם הייתרונות של המערכת מגיעים גם החסרונות, אנדרואיד היא מערכת הפעלה פתוחה (סוג של...) ובשל היותה כזאת, בסופו של דבר המשתמש אחראי על מכשירו ולא חשוב כמה מנגנונים יהיו, הוא עצמו החוליה החלשה במערכת.פרופורציה. שאלת הפרופורציה היא משמעותית; הפרופורציה דורשת שהפגיעה תהיה במידה שאינה עולה על הנדרש. השאלות שיעלו כאן הן האם מעבר לפריצה ומחיקה של החומר המקורי יבוצעו פעולות נוספות כמו "תג מחיר" לאותו פורץ, או האם יבוצע ניקוי ומחיקה רק של המידע של אליס; האם הפריצה תבוצע תוך פגיעה גם באחרים חפים מפשע (נניח, מחשבי זומבי בשימוש אותו פורץ) או רק במחשבים של הפורץ.

מידע דליף

מאת עו"ד יהונתן קלינגר

הקדמה

המאגר הביומטרי אינו אלא עוד אחד ממאגרי המידע שמוקמים לאחרונה במדינתנו. מדינת ישראל סובלת מקדחת מאגרי מידע, שכפי שמגדירים זאת אבנר פינצ'וק (הארץ, 07.06.2011) ומיכאל בירנהק (גלובס, 16.07.2008). המאגר הביומטרי כנראה אינו האחרון מבין המאגרים הפוגעניים שיוקמו וברור שאינו הראשון.

הצורך לנהל מאגרי מידע על אזרחי ישראל, ושאותם מאגרים יהיו בשימוש מתמד, הוא אחד מהמאפיינים של חברת מעקב, בו הפרט אינו אלא נתון. לצורך כך צריך לזכור כמה ישראל חריגה בנוף העולמי; ישראל היא אחת המדינות היחידות בהן יש את המושג "תעודת זהות" ומספר זהות. כאשר תלכו בניכר לפתוח חשבון בנק, יבקשו מכם דרכון, ויחד עם הדרכון יבקשו מכם חשבונות של שירותים על שמכם (נניח, חשמל או ארנונה) על מנת להוכיח כי הנכם מוכרים על ידי רשויות אחרות.

אבל תעודות הזיהוי ומרשם האוכלוסין (שהוא נגע רע משרידי המנדט הבריטי הכובש) אינם המאגרים היחידים שסובלים מקדחת מאגרי המידע; בכדי לנהל חיים מודרניים, אנו מותירים מאחורינו שביל של פירורי מידע שנותרים בכל פעולה שאנו מבצעים: קמנו בבוקר, התלבשנו, צחצחנו שיניים; הפעולה הבאה? יצאנו ורכשנו בבית הקפה מתחת לבית קפה ומאפה ושילמנו בכרטיס אשראי? הפעולה נרשמה במאגר כלשהוא. לא השתמשנו בכרטיס אשראי, אבל העברנו את כרטיס המועדון? גם כן; גם אם איננו חברי מועדון, אך הודענו ברשת החברתית האהובה עלינו שרכשנו שם, הפעולה נרשמה.

לאחר מכן, בדרך לעבודה, פתחנו את תוכנת ה-GPS האהובה עלינו, וזו דיווחה לנו על הפקקים בדרך על סמך נתונים פרטיים של משתמשים אחרים. התקשרנו מהטלפון הסלולרי לבוס להגיד שאנו מאחרים, והשיחה, כמו גם נתוני המיקום שלנו באותה שיחה, נרשמה במאגר המידע של חברת הסלולר שלנו על פי חוק נתוני תקשורת. הגענו לבסוף לעבודה, והחתמנו כרטיס; חלק מאיתנו אף מחתימים כרטיס עם טביעת האצבע שלהם. ההחתמה נרשמה גם כן. לאחר מכן, שתינו את הקפה יחד עם עדכונים מאתר החדשות האהוב עלינו, וזה שמר עלינו פרטים מזהים והבין את הרגלי הקריאה שלנו.

במהלך יום העבודה המעביד גם עוקב אחרינו, בין אם על ידי התקנת מצלמות נסתרות או על ידי טכנולוגיות שנועדו לבדוק את תפוקת העובדים. גם המידע הזה נשמר במאגר.

בדרך חזרה הביתה נסענו דרך כביש שש, שם גם נשמר עלינו מידע מפורט: מתי נסענו ובאיזו מהירות וגם אנו נדרשים להסכים לקבל ממפעילת הכביש דיווחים פרסומיים. ואם בערב נבחר להיות אקולוגיים ולנסוע למקום הבילוי בתחבורה ציבורית, הרי שגם אז כרטיס הרב-קו עוקב אחרינו ורושם במאגר מידע מרכזי את נתוני הנסיעה שלנו (לפחות עד שהנחיה חדשה של משרד המשפטים תכנס לתוקף).

טוב, תמיד יש את האפשרות לשבת בבית ולצפות בטלויזיה, אבל גם אז הפרטיות שלנו עוד בסכנה: לא רחוק היום עד שממירים חכמים ידווחו לחברת התקשורת מהם הערוצים בהם צפיתם, נכון?

אז קריאה פסימית של חמש מאות המילים האחרונות מעידה על המצב העגום שלנו ומעלה תהיה אחת: האם יש באמת סיבה להלחם למען הפרטיות שלנו כאשר בכל יום המידע עלינו נשמר בעשרות מאגרי מידע? וזה עוד לפני שמדברים על מאגרי המידע של הבנקים או קופות החולים. אז מה יש לנו לעשות? האם אנחנו צריכים להלחם בטכנולוגיה ולנסות לשנות את החברה שלנו או להכיר במציאות העגומה שלפיה כל פירורי העוגיות שאנו מותירים במהלך היום עשויים לפגוע בנו? המטרה של הקדמה קצרה זו היא להבהיר מהי כמות המידע שנשמר, ומהן הסכנות, כמובן, באגירה המאסיבית של המידע.

עכשיו, כדי להסביר מהי הסכנה בדליפה של המידע, אפשר לפרק את העניין למספר סכנות: (1) שימוש במידע למטרות כלכליות [לגיטימיות ולא לגיטימיות], כלומר לגרום לכם לרכוש יותר או להוציא מכם כספים במרמה; (2) שימוש במידע למטרות פרסונאליות [לגיטימיות ולא לגיטימיות], כלומר לבצע פעולות בשמכם או להתחזות לכם; (3) שימוש במידע למטרות בטחוניות [לגיטימיות ולא לגיטימיות] כלומר להלחם בפשע או טרור או לבצע מעשי פשע או טרור.

חשוב להבין שהמטרה לשמה נאגר המידע לסיבה הלגיטימית הוא בדרך כלל גם למנוע את הסיכון הלא לגיטימי. לדוגמא, מידע על תנועה של בני אדם בתחבורה ציבורית ומהם האוטובוסים הצפופים ביותר יכול לסייע בתכנון של קווי אוטובוסים, אבל יכול גם לסייע לארגוני טרור לדעת מהם קווי האוטובוס הצפופים ביותר על מנת לתכנן פיגוע. מערכת שמאפשרת זיהוי מול חשבון בנק מסוים, על ידי שמירה של ססמא או טביעת אצבע, מאפשרת גם לזייף את הכניסה לאותו חשבון בצורה לגיטימית. כלומר, ככל שמאגר יותר רגיש, כך יש לאבטח אותו יותר.

לכן, קדחת מאגרי המידע, כפי שנראה במאמר זה, היא בעייתית במיוחד: היא לא רק מיותרת ונובעת מרצון לשמור מידע מיותר, היא גם מאפשרת בדיוק את הנזקים מהם היא מנסה להמנע.

מה היא דליפה?

הדבר הראשון שאני רוצה להגדיר, לצורך המכנה המשותף, הוא את ההגדרה של 'דליפה'; דליפה היא כל גישה שאינה מורשית למאגר, בין אם על ידי האקר איראני שגונב את כל המאגר או חוקר פרטי שמשלם חמישים שקלים לפקיד שיבדוק לו מידע מתוך המאגר. גישה, לצורך העניין היא היכולת לקרוא או לכתוב מידע למאגר. כלומר, דליפה היא היכולת לקרוא או לכתוב מידע למאגר שלא תוכננה כחלק ממטרות המאגר.

דליפה עסקית

דליפה יכולה להיות מצב בו מאגר מידע שמוחזק כחוק ומנוהל בצורה מאוד מאובטחת, עם כל ההגדרות, פשוט נמכר במסגרת עסקים לגורמים עסקיים אחרים שנראים לגיטימיים. דוגמא טובה לכך נחשפה בפברואר 2010: עיריית רמת-גן מכרה מידע לחברת 'קידום' לצורך שיווק (פגיעה מהסוג הראשון). הסיפור הוא כזה: לעירייה, כחוק, יש את מאגר המידע של כל תלמידי בתי הספר. המאגר מאובטח, מוגן ומנוהל כמו שצריך. חברת קידום הגיעה לעירייה והציעה לרכוש ממנה את שמות כל התלמידים; התלמידים, שמעולם לא נתנו את הסכמתם לקבלת מידע שיווקי, גילו שהמידע דלף על אודותיהם והגיע לחברה מסחרית, שניסתה למכור להם מוצרים. אכן, לא מדובר בגניבה של המאגר על ידי האקר איראני או גנב מתוחכם, אבל זוהי דליפה.

הדליפה מהסוג הזה כל כך נפוצה, עד שבאוקטובר 2010 נחשפה פרשה מאוד דומה בנוגע לחברת סלקום; זו רכשה מאגר מידע של מתגייסים והצליבה אותו מול מאגר המידע שלה, כדי לגלות מי מלקוחותיה הוא הורה של ילד שעומד להתגייס ולהציע לו הצעות שיווקיות. גם מקרה זה לא נקשר למשפחות פשע או לגורמים עוינים לישראל, אבל הוא בעל השלכות לא מבוטלות על הכיס שלנו.

הדליפה מהסוג הזה היא דליפה מערכתית; לא מדובר על ארגון פשע שסוחר עם ארגון פשע או על ארגון פשע שסוחר עם גוף מסחרי, אלא על שני גורמים שנתפסים על ידי הציבור כלגיטימיים, אשר מבצעים סחר במידע פרטי בלי ידיעת או הסכמת בעלי המידע, ולא למטרות של גורמי המידע.

כלומר, צריך להבין ש'דליפות' מתרחשות כל יום, וחלק מאיתנו בונה חלק מהחיים שלו על דליפות מהסוג הזה. הסיבה לכך היא העדר אבטחה מתאימה, והרבה פעמים הדבר נובע מרשלנות גרידא. לדוגמא,

כאשר אני מקבל דואר רשום, הדוור מגיע לביתי יחד עם רשימת המכתבים הרשומים, כאשר אני חותם על קבלת המסמך, אני רואה רשימה שלמה של אנשים אחרים שמקבלים דואר ונחשף למידע הפרטי על אודותיהם. זו גם דליפה של מידע, ואיננו יכולים להמנע ממנה במצב הקיים.

דליפת פריטי מידע בודדים

סוג נוסף של דליפות, שאולי מתקרב יותר לדליפה הפלילית שאנחנו מכירים ואוהבים, הוא הדליפה של מכירת מידע; השיטה היא פשוטה: חוקר פרטי או גורם עוין אחר רוצה לברר מידע על אדם מסוים (לדוגמא, לדעת מהי ההכנסה שלו) והוא מוצא פקיד, שבדרך כלל משתכר שכר נמוך ומציע לו, תמורת כסף מסוים (או תוך שהוא סוחט אותו), להעביר לו מידע שהוא שולט בו.

לדוגמא, ב-2007 הורשעה עובדת בביטוח לאומי שסחרה במידע פרטי ונחשף כי בכירים בביטוח לאומי ורשות המסים סחרו במידע פרטי. דליפה מהסוג הזה היא דליפה שאינה מערכתית, ואפשר לקרוא לה סחר לא לגיטימי במידע; היא נתפסת כחמורה יותר מהמערכתית, אך הרבה פעמים היא הרבה יותר נקודתית. הסכנה כאן היא פרסונאלית: כלומר, המידע נרכש הרבה פעמים על ידי בעל שחושד שאשתו בוגדת בו, או על ידי גורם שמעוניין לברר את מצבנו הפיננסי. לעיתים רחוקות יותר המידע נרכש על ידי ארגוני פשיעה שמעוניינים למצוא קרבנות יעילים.

זחילת שימושים

הסוג הבא של דליפה הוא זחילת שימושים; מצב זה הוא מצב בו יש אדם שיש לו גישה למאגר למטרה מסוימת, והוא עושה בו שימוש אחר. לדוגמא, ב-2009 הורשע עובד מדינה בכך ששלף מידע מתוך מאגר, ועשה בו שימוש לצורך אכיפה של תשלום מס (עשמ 3275/07 שמואל צ'יילר נ' נציבות שירות המדינה); דליפה מהסוג הזה מוגדרת כזחילת שימושים. זחילת השימושים היא כנראה הדליפה המסוכנת מכל והנפוצה ביותר; לעובדים יש גישה מתמדת למאגרי המידע, והשימוש שלהם הוא בקנה אחד עם מטרות הארגון אשר מחזיק את המידע; השימוש פשוט מנוגד לחוק או לנהלים.

אלא, שלתפוס 'מדליפים' מהסוג הזה בדרך כלל קשה ביותר: הרי הגורם שצריך לבקר על זחילת השימושים הוא הגורם אשר נהנה מההדלפה: רשות המסים, במקרה הזה, אמנם צריכה להתנער מהעובד

הסורר ששלף מידע על נישומים, אבל היא גם מרוויחה כסף מכך שהוא הגדיל את הכנסותיה מגביית מסים.

זחילת שימושים כזו התרחשה גם עם מאגר ה-DNA המשטרתי. במקרה מעניין שנדון לאחרונה על ידי בית המשפט העליון עלתה שאלה הנוגעת להרשעות על סמך ראיות DNA: הסיפור היה פשוט, במסגרת חקירת רצח מסוימת, התבקש אדם לתת דגימת DNA מתוך הסכמה עם המשטרה שהראיה תשמש אך ורק לצורך חקירת הרצח; לאחר שהדגימה הגיעה למעבדה, מצאה אחת השוטרות כי הדגימה תואמת לדגימה אחרת שנמצאה בזירת אונס, על סמך ראיות אלה, הורשע האנס. בית המשפט העליון פסל את הראיה משימוש (עפ 4988/08 איתן פרחי נ' מדינת ישראל) אך הותיר את ההרשעה על כנה.

כאן המקרה היה מעניין ביותר: חוקרת המשטרה אמרה שלא היתה "הצלבה" בין מאגרי המידע, אלא שהיא בעצמה זכרה את אלל ה-DNA וביצעה את ההצלבה בראשה. אכן, המקרה הזה הוא מקרה בו אנס יושב מאחורי סורג ובריח, אך מעניין לדעת כמה 'הצלבות בראש' נערכות עם מידע אחר שנמסר למשטרה, וזו אומרת שלא תעשה בו שימושים לרעה.

דליפה בעקבות פרצת אבטחה

הסוג הרביעי של דליפה הוא פרצת האבטחה; במקרה כזה לא נמכר מידע בזדון, או סתם תוך מניעים כלכליים אלא פשוט המידע לא מוגן טוב מספיק ואדם בעל כישורים מסוימים יכול להשתמש בו. דוגמאות לנושא זה ניתן למצוא ברשת למכביר, אבל דומה שהותרתן של תעודות פטירה ברחוב על ידי בתי חולים, או זריקה של גליונות ציונים ברחוב הם רק קצה הקרחון של הרשלנות בהגנה על מידע פרטי. שני המקרים התרחשו באוקטובר 2009, חודשיים לפני שחוק המאגר הביומטרי התקבל בכנסת, ועדיין הם לא העידו למחוקק על הסכנות בדליפה של מידע או בחוסר היכולת להגן עליו.

מקרה נוסף שהתרחש בחודש יולי 2008 היה פרצת אבטחה באתר בית החולים איכילוב שאפשרה צפיה בפרטים רפואיים של כל מטופל; במקרה אחר, חודש לאחר מכן, פרצת אבטחה באתר האינטרנט של שאול מופז אפשרה לכל אחד לצפות בספר המתפקדים של מפלגת קדימה.

סוג זה של דליפה, שנקרא פרצת אבטחה הוא שונה לגמרי: הוא לא נובע מזדון, אלא מתכנון לקוי. בדרך כלל הוא גם לא ניתן למניעה; גם תכנון טוב אינו חסין לפרצות אבטחה; לאחרונה הציעה גוגל מיליון דולר למי שיצליח למצוא פרצות אבטחה בדפדפן האינטרנט כרום. המטרה כמובן היתה להראות כמה הדפדפן בטוח, אך לא עברו מספר דקות מאז שהחלה התחרות ועד שהצליחו האקרים, עם מיליון דולר של מוטיבציה בכיס, למצוא פרצה.

דליפה בעקבות אבדן מידע

הסוג החמישי של דליפה הוא אבדן מידע; כאן מדובר על מצב בו מידע שאינו מוגן מוצא מתוך מאגר המידע למטרה מסוימת, אך הולך לאיבוד בדרך. כך, לדוגמא, בשנת 2005 הלך לאיבוד כונן USB עם מידע רפואי של 120,000 חולים בהוואי, ב-2006 הלך לאיבוד כונן USB עם מידע צבאי רגיש, ב-2010 כונן נייד עם מידע על מבוטחים רפואיים הלך לאיבוד ובאוגוסט 2011 חברה שנשכרה לצורך בקרה על בדיקת הגנת הפרטיות איבדה מידע רגיש על 4,500 חולים.

אכן, אבדן מידע הוא משהו שמתרחש לעיתים תכופות כאשר מידע מוצא על התקנים ניידים, ובמיוחד כאשר הוא לא מוצפן. כאן אין כוונה זדונית, אלא רשלנות גרידא וחוסר טיפול במידע.

סוגיית המאגר הביומטרי

עכשיו, כשהבנו את חמשת סוגי הדליפות, אני רוצה להכנס לסוגיית המאגר הביומטרי ולשאול האם אפשר באמת לומר שהוא יהיה מאובטח. בתהליך החקיקה ניסו לא אחת לטעון שהמאגר עצמו יהיה מוגן וינקטו בו את כל האמצעים על מנת להגן עליו מדליפה. אלא, שההגדרה של דליפה תמיד הוצגה רק כסוג הקלאסי של דליפה: הגעה של המאגר לאינטרנט בצורה מלאה. בשום מקום לא ניסו להסביר איך יוגן המאגר מפני שימוש לרעה של אלה שקיבלו את הסמכות להשתמש בו (כזכור, כל שוטר יוכל לגשת למאגר).

גם אם נקח בתור הנחה שתערך בקרת גישה ויתועד כל מקרה בו עובד של המאגר יגש למידע, עדיין ישנם מספר כשלים עיקריים: הראשון הוא בכך שגישה, כאמור, היא גם כתיבה וגם קריאה במאגר. נכון להיום מנסים למכור לנו את הטענה שהמאגר הביומטרי יגן עלינו מפני גניבת זהויות וזיוף של תעודות זהות, אלא שכל פקיד במשרד הפנים יוכל להכניס זהות למאגר.

כלומר, המאגר יהיה מאובטח ברמה 11, הוא יוגן על ידי האמצעים הטובים ביותר בשוק, אבל עדיין: יש לו נקודת כשל אחת: כל פקיד במשרד הפנים שרוצה להכניס את בן הדוד שלו ששילם מספר שקלים בזהות בדויה יוכל לעשות זאת, כי על תהליך הכניסה הראשון למאגר אין ביקורת.

כלומר, דליפה מהסוג השני, מתחייבת להתרחש: או על ידי שוטר שמקבל משכורת נמוכה שישוחד על מנת להעביר מידע לגורמי פשע, או על ידי פקיד משרד הפנים שישוחד כדי להכניס זהויות בדויות למאגר. הנקודה החלשה, של הדליפה מהסוג השני, אינה מטופלת בארכיטקטורה של המאגר.

ולסיכום

שלא יהיה ספק, המאגר ידלוף בכל דרך אפשרית: כל אחת מחמש הדרכים תתרחש לאורך זמן ארוך מספיק. הסכנה לכך תהיה משמעותית כיוון שלא יהיה ניתן להשתמש יותר בזיהוי ביומטרי כלל וכלל, כשם שכיום אי אפשר להשתמש במספר הזהות שלנו כאמצעי זיהוי מול גורמים אחרים. אלא, שעד שהמאגר ידלוף במלואו ויגיע לאינטרנט יגרמו לנו הרבה נזקים בדרך כי אנחנו נסתמך על המאגר: אנחנו נאמין שהמידע הביומטרי שלנו מאובטח, ושאף אחד לא עושה בו שימוש, כשבפועל מה שיקרה הוא שימוש לרעה, התחזות וניצול.

המאמר פורסם במקור בבלוג "Intellect or Insanity" של עו"ד יהונתן קלינגר, בפוסט: "חמישה סוגים של דליפה, או על הגדרות של אבטחת מידע".

עיצוב תעבורה

מאת לירן בנוריס ואלעד גבאי

הקדמה

במאמר זה נדון ב-"עיצוב תעבורה" אשר מבצעות ספקיות אינטרנט בארץ ובעולם.

"עיצוב תעבורה" הינה שיטת ניהול תעבורת רשת. בשיטה זו מעכבים חבילות מידע מוגדרות בכדי לקרב את התעבורה לפרופיל רצוי. בעיצוב תעבורה משתמשים בעיקר בכדי להגביל את מהירות הגלישה, אך כמו שנראה במאמר לעיתים משתמשים בטכניקה גם לדברים אחרים.

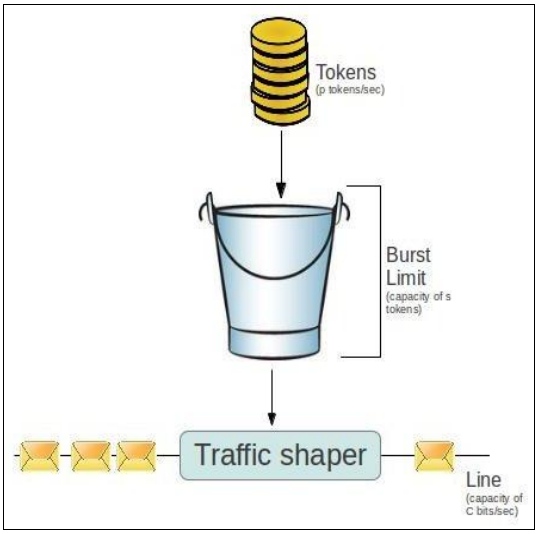
לאחרונה התעורר ויכוח סוער לגבי האם הרשת צריכה להיות ניטראלית, או האם ואם כן, עד איזו רמה מותר לספקיות האינטרנט, שהן מפעילות ובעלות קווי התקשורת, להתערב ולתעדף סוג תעבורה אחד על פני האחר. כמות גדולה של ספקיות אינטרנט רוצות להקטין את התעבורה הנצרכת על ידי אפליקציות שצורכות תעבורה רבה (שיתוף קבצים לדוגמא) בכדי לאפשר מתן תעבורה לאפליקציות שצורכות תעבורה מעטה יחסית (גלישה וכו'). ספקיות האינטרנט יכולות ואולי גם משתמשות בעיצוב תעבורה בכדי להאט שירותים המתחרים בשירותים שהספקית עצמה מפעילה (שיחות VoIP, שירותי ענן למיניהם...).

ספקי התוכן לעומתם נגד התערבות של ספקיות האינטרנט בתעבורה, זאת מכיוון שיכולת זו נותנת לספקיות האינטרנט שליטה גדולה מאוד על חווית המשתמש, לדוגמא, אם אפליקציה מסוימת עובדת לאט לעומת אפליקציה אחרת, המשתמש עלול להפסיק להשתמש בה, ולעבור לזו שעובדת מהר.

ספקיות האינטרנט אינן נוהגות לחשוף כיצד הן מבצעות את עיצוב התעבורה שלהן, דבר זה משאיר את המשתמשים באפלה. מצב זה הביא ליצירת כלים אשר נועדו לאסוף סטטיסטיקות על מעבר הנתונים ולזהות האם ואיך מבצעת ספקית האינטרנט עיצוב תעבורה.

כיצד מעצבים תעבורה?

מעצב תעבורה הינו מודל העברת חבילות בעל כניסה אחת ויציאה אחת (SISO - single-input single-output) שפועל על פי מודל דלי המטבעות:



מודל זה מניח קיום של דלי מטאפורי אשר יכול להכיל s מטבעות. דלי זה מחובר לערוץ בעל קיבולת C bps. בזמן שהדלי לא מלא, נוצרים בו מטבעות בקצב של p מטבעות בשנייה (p<C).

הערוץ יעביר חבילה בגודל L ביטים שמגיעה, רק אם בדלי יש לפחות L מטבעות (ביטים), כשהערוץ מעביר את החבילה ה"מעצב" לוקח, כלומר משלם ב-L מטבעות מהדלי. במצב בו התחלנו כאשר הדלי מלא ב-s מטבעות, ועלינו לשלוח כמות גדולה של חבילות בגודל L כל אחת (לשם הפשטות נניח כי L מתחלק ב s ללא שארית). הערוץ יוכל להעביר k חבילות על הערוץ בעל קיבולת C כאשר:

$$k=\frac{s/L}{1-p/C}$$

ולאחר k החבילות הללו, הערוץ יעביר את החבילות לפי קצב יצירת המטבעות p.

p נקרא גם "קצב העיצוב", מכיוון שבמצב יציב (כאשר אין פרצים - הדלי ריק) המעצב מעביר חבילות בקצב יצירת המטבעות. קיבולת הדלי היא שקובעת את כמות המידע שנוכל להעביר ב"פרץ" המהיר (k החבילות הראשונות). ו-s זהו גודל ה"פרץ" המקסימאלי. נשאלת השאלה מה יקרה כאשר חבילה מגיעה למעצב התעבורה, אך אין מספיק מטבעות בדלי על מנת להעביר אותה הלאה - אין במה לשלם. במצב כזה מעצבי התעבורה מתחלקים לשני סוגים: אלו שזורקים את החבילות, מה שגורם לאיבוד חבילות גדול. ואלו שמחזיקים תור של חבילות. רוב מעצבי התעבורה הם מן הסוג השני, ובהם נתמקד לאורך המאמר.

מדוע שהספקיות ישתמשו במעצבי תעבורה?

אחד השימושים שעושות ספקיות האינטרנט במעצבי תעבורה היא להרשות למשתמש לחרוג מהמהירות עליה שילם (לזמן מוגבל כמובן). כולנו אוהבים לשנוא את הספקיות שלנו ולקטר על איך שהן לא נתנות לנו את השירות עליו שילמנו או מפחיתות מאיכותו, אך כדאי לדעת שספקיות האינטרנט נותנות לנו גם לעבור את המכסה עליה שילמנו, למרות שהיא גם מחבלת לנו בחווית השימוש בדרכים אחרות, אבל נחמד לדעת שיש גם צד שני (גם אם הוא קטן). אז איך בדיוק ספקיות האינטרנט עושות זאת? נחזור למודל הדלי שלנו: ראינו כי כאשר הדלי מלא ואז אנו שולחים\מקבלים פרץ מידע גדול, אנו יכולים לשלוח\לקבל חבילות בקצב גדול יותר מקצב היצירה של המטבעות. זהו בדיוק העניין, כמה שקיבולת הדלי יותר גדולה, ספקית האינטרנט מאפשרת לנו פרץ יותר גדול.

שימוש אחר שספקיות האינטרנט עושות במעצבי תעבורה היא הגבלת קצב התעבורה של משתמש או בכדי הגבלת מקצת התעבורה עבור תוכנה מסויימת (למשל תוכנות P2P). שימוש נוסף בעיצוב תעבורה הוא הגבלת המשתמש לתעבורה עליה שילם, ז"א הכלת חסם עליון על מהירות התעבורה לדוגמה, אם משתמש שילם על 5Mbs, זוהי תהיה המהירות המירבית שיקבל, חסם עליון שכזה נקבע על ידי קצב יצירת המטבעות. יש לציין שספקיות האינטרנט אינן חושפות מידע על הצורה בה הן מעצבות את התעבורה וקשה מאוד לדעת האם התעבורה עוברת עיצוב ואם כן באיזה צורה ומהם הנתונים (קצב יצירת המטבעות, קיבולת הדלי מספר המטבעות שקיים ברגע נתון), האם העיצוב נעשה עבור כל משתמש בנפרד, או שיש עיצובים נפרדים עבור שירותים נפרדים אותם מבקש המשתמש, אולי גם וגם...

תעדוף תעבורה

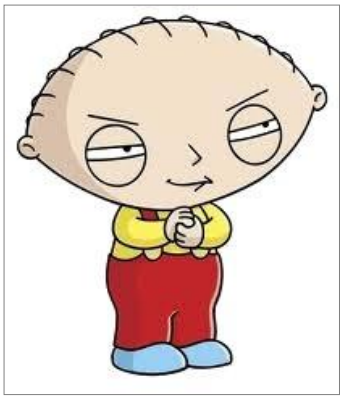
תעדוף תעבורה נעשה על ידי שימוש בעיצוב תעבורה, זהו מצב בוא ספקית האינטרנט נותנת עדיפות לזרם נתונים מסויים על פני זרם אחר. ישנם כמה מימדים של תעדוף תעבורה:

- **תעדוף תעבורה על פי סוגי תעבורה**: בכדי לבצע תעדוף תעבורה על פי סוג התעבורה, ז"א תעבורה של אפליקציות שונות, על הספקית לזהות לאיזו אפליקציה שייכת כל חבילה, ניתן לעשות זאת בשיטות הבאות:

- **על ידי כתובת ה-IP בכותרת החבילה**: כתובות ה-IP של המקור והיעד יכולות להעיד על זרמי נתונים שונים. לדוגמה, חברות שרתים לעיתים יגבילו את מהירות הגלישה של עובדיהן בכדי לאפשר לשרתים מהירות העלאה והורדה טובים יותר, אם נעשה שימוש באותו הקו.
- **על ידי כותרת הפרוטוקול**: ספקיות אינטרנט יכולות להשתמש בפורטים, בסוג הפרוטוקול או מזהים אחרים בכדי לזהות לאיזה זרם שייכת חבילה (לדוגמה: https עובר בפורט 443).
- **על ידי תוכן החבילה**: ספקיות האינטרנט יכולת להשתמש ב-DPI (Deep Packet Inspection) בכדי לזהות את האפליקציה שיצרה את החבילה, כך לדוגמה יכולה הספקית לזהות זרמי נתונים של אפליקציות P2P ולהגביל את תעבורתם.

- **תעדוף תעבורה שלא תלוי בסוג התעבורה**: בנוסף לזרם הנתונים עצמו, הספקית יכולה להשתמש בקריטריונים בלתי תלויים בחבילות. למשל:
  1. **זמן**: ספקית יכולה להגביל את התעבורה בשעות העומס.
  2. **עומס הרשת**: ספקית יכולה להגביל את התעבורה על גבי ערוץ מסוים כאשר הערוץ עמוס.
  3. **המשתמש**: ספקית יכולה להגביל משתמשים שתעבורת הרשת שלהם גדולה.

דלי מלא בתכסיסים



מודל הדלי הינו מודל הגבלת תעבורה בסיסי, ניתן להכליל אותו לצורה כללית בה המודל הוא מודל בעל כניסות מרובות ויציאה אחת (MISO- Multiple Input Single Output). על פי האנלוגיה שלנו, במודל זה קיימים מספר ערוצי כניסה ולכל ערוץ משוייך דלי, ולכל דלי ישנו קצב יצירת מטבעות. עכשיו נצא מהאנלוגיה בחזרה אל המציאות. המודל הזה קיים כמעט בכל נתב היום. הספקית יכולה לשייך כל כניסה (הכניסות אינן פיסיות אלא חלק מן האנלוגיה) לזרם נתונים מסוים ולתעדף אותו על פי הקרטריונים שהצגנו קודם.

בנוסף למודל הדלי, ישנם שיטות נוספות לביצוע תיעדוף תעבורה, השיטות הללו הן קצת יותר מלוכלכות אך עושת את העבודה:

- **חסימה**: שיטה אחת לבצע תיעדוף תעבורה היא לסיים את זרם הנתונים בו העדיפות נמוכה יותר (על פי הסטנדרטים של ספקיות האינטרנט), ניתן לעשות זאת באמצעות חסימת החבילות ששייכות לזרם זה, או על ידי הזרקה של חבילת סיום קשר (כמו RST, FIN השייכות ל-TCP) וכך לחסום ולסגור את זרם הנתונים.
- **זריקת חבילות**: הספקית יכולה לזרוק חבילות בקצב קבוע (כל חבילה עשירית) או משתנה.

- **שינוי ה-TCP Window Size**: בפרוטוקול TCP כל חבילה מכילה בשכבת ה-TCP שדה הנקרא window size (למידע נוסף על שדה זה: http://en.wikipedia.org/wiki/Transmission_Control_Protocol). ספקית האינטרנט יכולה לערוך את שדה זה ולהקטינו וכך לגרום לשולח להאט את קצב השליחה.
- **מנגנונים בשכבת האפליקציה**: ספקית האינטרנט יכולה לשלוט בהתנהגות של אפליקציה על ידי שינוי המידע בחבילות שמועברות בפרוטוקול שלה (והיא מזהה כאלו באמצעות מנגנוני DPI). לדוגמה, סביר להניח שחברת בזק לא מסמפטת את אפליקצית Skype כי "למה לא התקשרנו מהבזק?!?". נניח שבאפליקצית סקייפ, שהיא אפליקצית RealTime VoIP מוגדר כי אם חבילה הגיעה באיחור של זמן מסויים, האפליקציה מתעלמת ממנה, כי הרי אין טעם להשמיע משהו שנאמר בעבר. בזק יכולה לגרום לשיבושים רבים באיכות השיחה על ידי שינויים במידע שעובר בחבילות הללו.

אז איך בודקים?

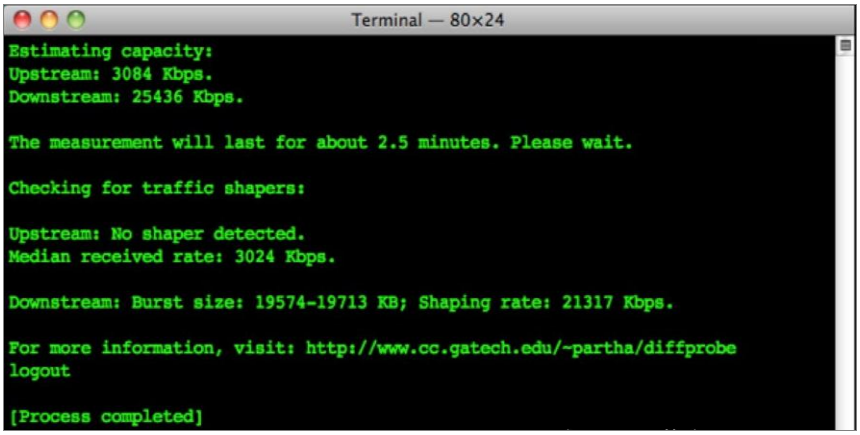
קיימים מספר כלים אשר בודקים אם ספקית האינטרנט משתמשת בשיטות עיצוב או תעדוף תעבורה. כלים יכולים להיות קריאות API, שבהם ניתן להשתמש כאשר כותבים תוכנה, או כשירות, המאפשר למנהלי רשתות לבדוק האם הרשת שלהם נתונה לעיצוב תעבורה.

Shaperprobe

כלי זה הוא כלי קוד פתוח, ניתן להוריד ולהריץ אותו, קיימים גרסאות 64/32bit שלו עבור: Windows, Linux ו OS X ואפילו פלאגין שמתחבר ל-Vuze (קליינט ביטורנט). הוא בנוי מ-6,000 שורות קוד עבור הגרסה נטולת ממשק המשתמש ומתוכנן להיות קל לשימוש, מדוייק ולא חודרני. Shaperprobe עובד בתהליך הנעשה בקצוות (end-to-end), התהליך מזהה את רמת עיצוב התעבורה על המסלול בין השולח למקבל.

כאשר קיבולת צוואר הבקבוק בערוץ היא $C(bps)$, צד אחד שולח חבילות בקצב קבוע של $R_s = C$ ביטים לשנייה, והצד המקבל נותן חותמת זמן לכל חבילה שהתקבלה וסופר כמה ביטים התקבלו בכל מקטע זמן.

יש היתכנות לשגיאה בספירה בגלל שאנו משייכים כל חבילה למקטע זמן מסויים וחבילה מכילה מספר ביטים יחסית גדול, השגיאה היא $\epsilon = \pm S/\Delta$ כאשר S היא ערך ה-MTU ו- Δ היא גודל המקטע, לכן, אם נבחר גודל מקטע גדול מספיק, נוכל להקטין משמעותית את השגיאה.

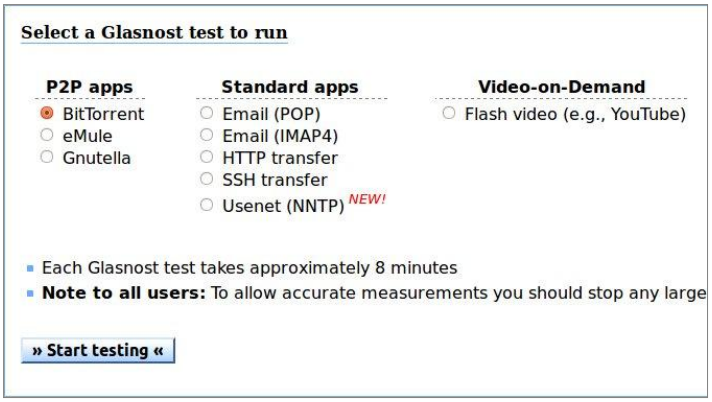


התוכנה מזהה שינוי במספר הביטים שהתקבלו בכל מקטע, אם במקטע מסויים פתאום יש ירידה מספיק גדולה (מעל לרף מסוים) ומדדנו מספיק זמן לפני הירידה בכדי שהירידה הזאת היא לא פשוט סוף הפרץ הראשוני, ומספיק זמן אחרי הירידה בכדי לראות שזה לא משהו זמני שנגרם עקב עומס, אז הירידה נגרמת על ידי מעצב התעבורה.

על פניו, הפיתרון נראה טרוואילי (לפחות בתיאוריה, מעשית יש לזכור שיש הרבה גורמים המחבלים ביכולת לשלוח חבילות בקצב קבוע), חלק עיקרי בתהליך הוא למצוא את C, אנו לא נסביר במאמר זה איך מוצאים קיבולת של ערוץ מכוון שגם תהליך זה אינו פשוט ויש המון מחקרים ומאמרים המנסים למצוא את הדרך הטובה ביותר.

בנוסף לבדיקת הימצאות של מעצב תעבורה, במקרה ו-Shaperprobe מזהה מעצב תעבורה הוא גם ינסה להעריך את המאפיינים שלו (קיבולת הדלי וקצב יצירת המטבעות).

Glasnost

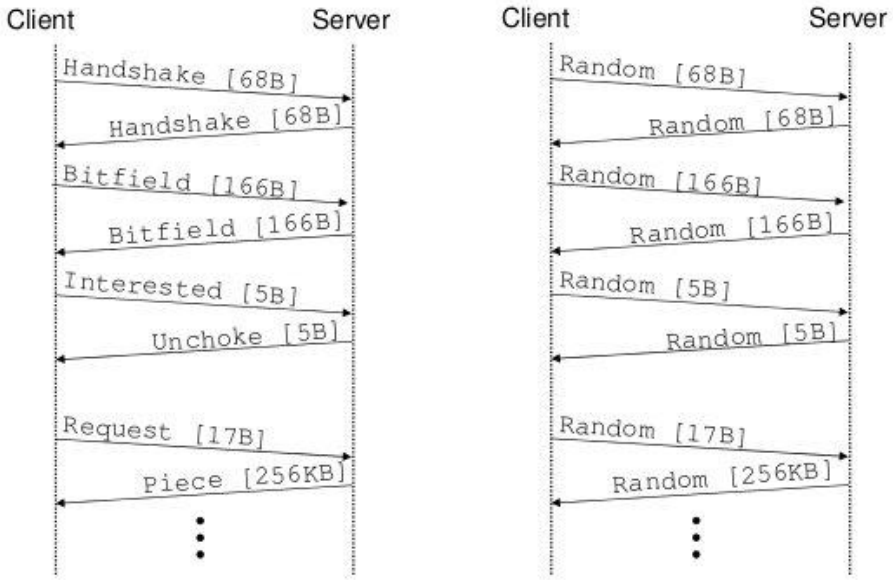


כלי זה משמש לבדיקה של תעדוף תעבורה, הוא מאוד קל לשימוש, אפילו עבור משתמשים ללא כל נסיון וידע, הוא לא דורש הורדה ופועל היישר מן הדפדפן. הכלי נועד להיות כלי שמתפתח בכדי להישאר רלוונטי במידה וספקיות האינטרנט מבצעות שינויים כלשהן או משתמשות במנגנונים חדשים בכדי לבצע תעדוף תעבורה. כל משתמש יכול להעלות לאתר הכלי בדיקות משלו ולהריצן. בנוסף, משתמשים יכולים להוריד את קוד המקור של Glasnost ולהריץ שרת על המחשב שלהם (חשוב לציין שזה עוזר למנוע white-listing על ה-IPים של מעבדות הבדיקה על ידי ספקיות האינטרנט).

בעוד ש-Shaperprobe מזהה רק עיצוב תעבורה הנעשה על פי מודל הדלי, Glasnost מסוגל לזהות את כל סוגי עיצוב התעבורה המתבססים על פרוטוקול TCP (אלו הסוגים היותר נפוצים של תעדוף ועיצוב תעבורה).

Glasnost בניגוד ל-Shaperprobe, לא תוכנן לזהות הגבלות על כל התעבורה של המשתמש, כלומר, שאם ספקית האינטרנט מבצעת הגבלות על פי זמן מסוים, יום או התנהגות המשתמש, Glasnost לא יזהה אותן בבדיקה אחת, אך ניתן לבצע כמה בדיקות בעזרת Glasnost ולהשוות בעצמנו בין התוצאות.
במקום לבדוק האם קיים עיצוב תעבורה על פי בדיקות על מנגנוני עיצוב שונים, Glasnost בודק קיום של עיצוב כלשהו על פי השפעתו על ביצועי התוכנה. הבדיקה נעשית על ידי יצירה של שתי זרמי מידע זהים לאחד השרתים הקיימים עם הבדל קטן שיוצר הבדיקה חושב שהוא שיגרום לספקית לתעדף את הזרמים באופן שונה.

לדוגמה: בדיקה האם הספקית מבצעת תעדוף שלילי לחבילות ביטורנט יכולה להתבצע בשיטה הבאה: שולחים לשרת חבילות עם פרוטוקול ביטורנט ומידע של ביטורנט כזרם ראשון, ובזרם השני שולחים לשרת חבילות עם מידע רנדומאלי אך בגודל זהה לגודל של החבילות בזרם השני, באותו הפורט ולאותו היעד. במשך התהליך השרת והקליינט שומרים מידע על החבילות המגיעות כולל מידע על שגיאות הנגרמות מבעיות רשת (אי הגעת חבילות, ניתוק, וכו'). כאשר הבדיקה נגמרת, הקליינט מעביר את המידע לשרת, השרת מנתח את הנתונים שהוא אסף ביחד עם הנתונים שנוצרו בצד קליינט ומציג את התוצאות. ניתן לחזור על הבדיקות עם פורט שונה, פרוטוקול שונה, תוכן שונה וכו'.

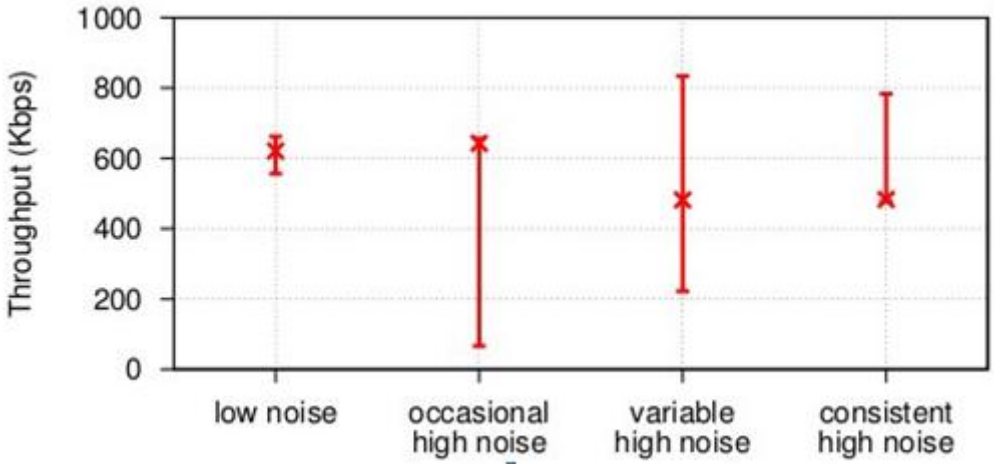


(a) BitTorrent flow (b) Reference flow

היות והתוכנה מתחשבת בסבלנות הקצרה של המשתמשים היא לא יכולה לבצע בדיקות ארוכות, אך בבדיקות קצרות יש להתמודד עם הרעש הנובע מהחלפת מסלולים, עומס ברשת וכו'. Glasnost מניחה שהחלפת מסלולים אינה קוראת בתדירות מספיק גבוהה בשביל להשפיע על הבדיקות, ומתמודדת עם שאר הרעשים המשמעותיים (עומס על הרשת) על ידי בדיקה מקדימה של הערוץ, התוכנה שולחת שני זרמי נתונים זהים לחלוטין ומזהה מה ההבדל ביניהם.

על פי מחקרים שביצעו מפתחי התוכנה, ישנם ארבע סוגי רעש:

- **רמות רעש נמוכות בצורה רציפה**: כל הבדיקות על קצבי העברה שנעשו בין זרמים זהים נפלו בטווח קרוב.
- **בדרך כלל רמת רעש נמוכה אך לפעמים רעש גבוה**: רוב הבדיקות על קצבי העברה שנעשו היו בטווח קרוב אחת לשנייה אך חלק קטן היו רחוקות (ניתן לראות שהמינימום והמקסימום רחוקים, אך החציון קרוב מאוד למקסימום).
- **רעש עם שונות גבוהה מאוד**: הבדיקות נתנו תוצאות שונות מאוד אחת מן השנייה (ניתן לראות שהמינימום והמקסימום רחוקים מאוד אחד מן השני, והחציון רחוק משניהם)
- **בדרך כלל רמת רעש גבוהה אך לפעמים רמת רעש נמוכה**: רוב הנקודות מקובצות ביחד סביב המינימום אך חלק קטן רחוקות (ניתן לראות שהמינימום והמקסימום רחוקים, והחציון קרוב מאוד למינימום).



(הפס מייצג את טווח הערכים שהתקבלו, והאיקס מייצג את החציון)

הסיווג של מפתחי התוכנה בכל מקרה נעשו על ידי שתי הבחנות לגבי המאפיינים של הרעש, הראשון הוא שרעש (עומס) ברשת תמיד מפחית את המהירות בה חבילות עוברות ואף פעם לא משפר אותה. ז"א שכאשר רוב הנקודות מקבצות סביב המינימום אך יש כמה שנמצאות רחוק יותר (כמו במקרה 4), סביר יותר שהנקודות שקרובות למקסימום הן נקודות שלא הופרעו מן העומס. השני הוא שעומס ברשת לרוב אינו קבוע ורציף על פני משך זמן ארוך. בתיאוריה אפשר להסביר את המדידות של מקרה 1 בתור רעש גבוהה ורציף, אך זה ידרוש מן הרעש להיות גבוהה ורציף במשך כל זמן הבדיקה (שהיה 10 דקות) ולכן זה לא סביר. במקרים 2 ו-3 אין אפשרות לדעת האם השינוי בקצבי העברה נעשה על ידי תעדוף של הספקית או על ידי רעש על הערוץ ולכן התוכנה מתמודדת רק עם מקרים 1 ו-4.

מה קורה בארץ?

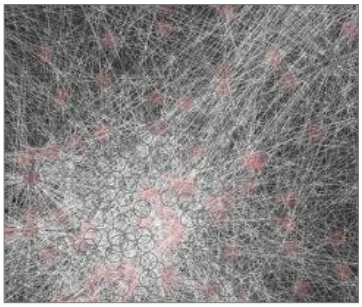
בארצנו הקטנטונת לא נערכו הרבה בדיקות בעניין ואף לא בדיקה יסודית אחת.
ב-2009 נעשתה בדיקה של אתר Ynet ופורסמה כתבה על ידי יהונתן קלינגר וניב ליליאן (קישור לכתבה נמצא בסוף המאמר). בכתבה נעשה שימוש בתוכנת Glasnost שהצגנו קודם בכדי לבדוק האם ספקיות האינטרנט הישראליות מבצעות תעדוף נתונים והאם הן מתערבות בצורה כלשהי במהירות התעבורה כאשר מדובר בתעבורה של תוכנות שיתוף קבצים.

מסקנות התחקיר של Ynet העלו חשד כי החברות בישראל מבצעות תעדוף תעבורה, אך על פי האמור בכתבה הבדיקה לא הייתה רחבה מספיק בכדי להעיד באופן משמעי על כך. הנה כמה נתונים מהאתר של Glasnost:

ספקית	מספר הבדיקות	אחוזים של התעבורה שעוצבה (הסיכוי ל-false positive הינו 1%)			
		העלאות ביטורנט	העלאות שנעשו בפורט של ביטורנט	הורדות ביטורנט	הורדות שנעשו בפורט של ביטורנט
נטוויז'ן 013	160	19	20	18	22
בזק בין-לאומי	474	18	22	17	29
סמייל	185	0	9	6	6

(יש לקחת בחשבון שיכול להיות שהנתונים ישנים ושלא נעשו מספיק ריצות)

סיכום



עיצוב תעבורה נמצא בשימוש על ידי ספקיות האינטרנט וחוסר השקיפות של הספקיות פוגע ישירות בצרכן.

היום כאשר קיימים אלגורתמים ופתרונות חומרה לבדיקת DPI (קיצור של Deep Packet Inspection) והם עובדים במהירות מסחררת, ספקיות האינטרנט יכולות להשתמש באלו בכדי לסווג חבילות לא רק על בסיס פורטים או סוג פרוטוקול, מה שיעשה חיים קשים בהרבה לאלה המנסים לעקוף (בעזרת מנהור, שינוי פורט וכו') את המגבלות והעדיפויות שמפעילה ספקית האינטרנט.

אם ספקיות האינטרנט יהיו שקופות יותר בנוגע לכיצד הן מנתבים חבילות או איזה סוגים של מידע הן מתעדפות למשתמש הפשוט יהיו חיים הרבה יותר קלים, דבר זה גם יוכל לתת לכל ספקית את הייחוד שהיא צריכה, במקום לריב על אותן הלקוחות, ספקיות יוכלו ליצור מסלולים מתועדפים לפי צרכי הלקוח.

אתם מוזמנים לבדוק האם אתם נתונים למנגנוני עיצוב תעבורה שונים, ותוך כדי לעזור להפוך את הרשת למקום שקוף קצת יותר, תוכלו להריץ בדיקות שונות בלינק הבא:

http://measurementlab.net/measurement-lab-tools

לקריאה נוספת

[1] http://www.cc.gatech.edu/%7Edovrolis/Papers/shaperprobe-IMC11.pdf

[2] http://broadband.mpi-sws.org/transparency/results/10_nsdi_glasnost.pdf
[3] http://broadband.mpi-sws.org/transparency
[4] http://measurementlab.net
[5] http://www.ynet.co.il/articles/0,7340,L-3785439,00.html

פיתוח מערכות הפעלה - חלק ב'

נכתב ע"י עידן פסט

הקדמה

מה עשינו בחלק הקודם?

בפרק הקודם בנינו מערכת הפעלה בסיסית ביותר - הקוד רץ ב-Real Mode ומשתמש בפסיקות של ה-BIOS בכדי להדפיס מחרוזות למסך. כמו כן, הקוד נכתב בשפת סף טהורה. מערכת הפעלה זו הייתה מוגבלת מאוד - היא יכלה לנצל רק Address Space קטן, גודל האוגרים היה 16 ביט, ולא יושמה שום הגנה על גישה לזיכרון.

מה נעשה בפרק זה?

בפרק זה נלמד על המעבר ל-Protected Mode, מצב בו הגישה זיכרון ולחומרה מבוקרת ומוגנת, וניממש אותו. נלמד לכתוב למסך בעזרת כתיבה לזיכרון, בניגוד לשימוש בפסיקות BIOS, ונכתוב את תחילת הגרעין (Kernel) שלנו בשפת C. גם בפרק זה מערכת ההפעלה עדיין תרוץ Live מדיסק, ולא תהיה כתיבה לדיסק הקשיח.

מערכת הפעלה סימן IV

לאור האורך של הקוד שאנו נשתמש בו בפרק זה, אנו נציג ונסביר רק קטעים נבחרים במאמר עצמו. את הקוד המלא ניתן למצוא בכתובת ה-URL הבא:

HTTP://www.digitalwhisper.co.il/files/Zines/0x1F/osdev.zip

boot.asm

ראשית נביט בקובץ boot.asm שנמצא בתיקייה src/boot. קובץ זה משמש כ-Bootloader הדואג לסביבה מתאימה להרצת מערכת ההפעלה שלנו, ומרגע שארגן סביבה זו הוא מעביר את השליטה אל הקוד. יש

לציין שקוד זה נלקח מדוגמה ב-wiki.osdev.org, וקיימים בו מספר רב של שורות קוד שאינן נחוצות במקרה שלנו, אך השארתי אותם למען אלו שירצו להעמיק ולקרוא ב-wiki ויצטרכו את השורות הנוספות בכדי להריץ את קטעי הקוד שמופיעים שם.

```
…
MODULEALIGN equ  1<<0
MEMINFO     equ  1<<1
FLAGS       equ  MODULEALIGN | MEMINFO
MAGIC       equ    0x1BADB002
CHECKSUM    equ -(MAGIC + FLAGS)
 ..
align 4
dd MAGIC
dd FLAGS
dd CHECKSUM

…
push eax
push ebx
call kmain
…
```

נביט על תחילת הקוד:

```
MODULEALIGN equ  1<<0
MEMINFO     equ  1<<1
FLAGS       equ  MODULEALIGN | MEMINFO
MAGIC       equ    0x1BADB002
CHECKSUM    equ -(MAGIC + FLAGS)
…
align 4
dd MAGIC
dd FLAGS
dd CHECKSUM
```

קוד זה נועד לדאוג שמערכת ההפעלה שלנו תהיה Multiboot Complaint, כך ש-Bootloader שתומך בסטנדרט זה יוכל לטעון אותה.

Mutliboot Specifications

באופן היסטורי כל מערכת הפעלה השתמשה ב-Bootloader משלה. רכיב זה ידע לארגן סביבה נכונה ולטעון אך ורק את מערכת ההפעלה לשמה הוא נכתב. דבר זה יצר שתי בעיות, האחת, נעלמה האפשרויות למודולריות, ואיתה רעיון ה-DRY (Don't Repeat Yourself). השניה, התקנת מספר מערכות הפעלה על מחשב אחד הייתה סיוט. כדי לפתור בעיה זו הומצא סטנדרט בשם Multiboot שמאפשר שני

דברים עיקריים: הסטנדרט מאפשר למשתמש להתקין כל Bootloader שתומך בסטנדרט, ו-Bootloader זה יוכל להפעיל כל מערכת הפעלה שתומכת בסטנדרט, כמו כן, הוא מאפשר להתקין מספר מערכות הפעלה על אותו מחשב ולבחור באיזו להשתמש לאחר ההפעלה. ה-Bootloader שתומך ה-Multiboot הנפוץ ביותר שקיים כיום נקרא GRUB, והוא חלק מפרויקט GNU.

באופן טכני, כדי שמערכת ההפעלה שלנו תהיה Multiboot Complaint אנו חייבים לשים ב-8192 הבתים הראשונים של קובץ שלנו כותר מיוחד שדרכו ה-Bootloader יכול לדעת כיצד לטעון את מערכת ההפעלה שלנו. ניתן לקרוא את ההגדרות המלאות של הסטנדרט בכתובת:
http://www.gnu.org/software/grub/manual/multiboot/multiboot.html

מה שרלוונטי למערכת ההפעלה שלנו הוא הפורמט שהכותר צריך להיות בו והסביבה בה מערכת ההפעלה הולכת להיטען אליה. הפורמט של הכותר אמור להיראות כך (יש לציין שכל הכותר צריך להיות מיושר ל-32 ביט):

בתים	שדה	שימוש
0-3	MAGIC	בבתים אלו יש לשים את מספר ה-MAGIC (סדרת בתים מזהה) של הכותר - 0x1BADB002
4-7	FLAGS	סדרה של דגלים שקובעים אפשרויות שונות של טעינת מערכת ההפעלה.
8-11	CHECKSUM	מספר, שכשאר נבצע סכום שלו עם ה-MAGIC וה-FLAGS נקבל 0. שדה זה נועד להצביע על טעויות בקריאת הכותר.
12-47	שדות נוספים	אם FLAGS מסוימים מודלקים ויש לספק מידע נוסף בכדי שהם יוכלו לפעול הוא יופיע בטווח בתים זה.

מעבר לכותר אנו יודעים מספר דברים על מצבי האוגרים אחרי טעינת מערכת ההפעלה. ראשית, אנו נהייה ב-Protected Mode (יש לציין שיש עוד כמה צעדים שיש לבצע בכדי שנוכל להיות ב-Protected Mode במובן המהותי של המילה). שנית, אוגר ה-EBX ימולא במצביע לאזור הזיכרון שבו יושב מידע אודות המערכת (Multiboot Information). שלישית, אוגר ה-EAX ימולא במספר MAGIC שיזהה שמערכת ההפעלה נטענה על ידי Bootloader שהינו תואם את סטנדרט ה-Multiboot - 0x2BADB002.

נביט על חלקו הראשון של קטע קוד זה:

```
MODULEALIGN equ  1<<0
MEMINFO     equ  1<<1
FLAGS       equ  MODULEALIGN | MEMINFO
MAGIC       equ    0x1BADB002
CHECKSUM    equ -(MAGIC + FLAGS)
```

הפקודה equ אחראית על הכרזת אזור בזיכרון שיכיל קבוע מסוים. ניתן להקביל אותה בשימושה להנחיית ה-define# ב-C. כל חמשת הקבועים שאנו מכריזים עליהם פה הולכים להיכתב לשדות שצריך בכותר ה-Multiboot. כפי שניתן לראות ה-MAGIC מכיל את מה שהוגדר בסטנדרט (0x1BADB002), וה-CHECKSUM מכיל את המשלים לסכום ה-MAGIC וה-FLAGS.

מה שמעניין בקטע קוד זה הוא שני הדגלים שנקבעים. אם נשים לב לשלושת שורות הקוד הראשונות נוכל לראות שכל יעוד שלהן הוא "להדליק" את שתי הסיביות הראשונות ב-FLAGS. הסיבית הראשונה אומרת ל-Bootloader לטעון את המודולים הנוספים למערכת ההפעלה שלנו על גבולות של עמודים (על עמודים נלמד בהמשך). הסיבית השנייה אומרת ל-Bootloader לכלול ב-Multiboot Information מידע אודות הזיכרון של המחשב. כפי שציינתי בהתחלה, אין צורך אמיתי להדליק את הדגלים האלה, אך בשביל תאימות עם דוגמאות הקוד ב-Wiki, ובשביל ערכם הלימודי, הפקודות הושארו.

```
align 4
```

שורת קוד זו דואגת ליישור לפי 4 בתים (32 סיביות), שזהו היישור שסטנדרט ה-Multiboot קובע שהכותר צריך להיות בו.

```
dd MAGIC
dd FLAGS
dd CHECKSUM
```

שלושת שורות קוד אלו דואגות לכתוב את כותר הMultiboot לתוך התוכנית. כך, כאשר ה-Bootloader ינסה לטעון את מערכת ההפעלה שלנו, הוא יחפש את הכותר ב-8192 הבתים הראשונים עד שהוא ימצא אותו. זו גם הסיבה שאנחנו כותבים את הכותר כמה שיותר קרוב לתחילת הקובץ (יקח פחות זמן עד שה-Bootloader ימצא אותו).

כעת, נביט בשורות הבאות:

```
push eax
push ebx
call kmain
```

השורה האחרונה די ברורה - היא קוראת לתווית kmain (שבתחילת הקוד של boot.asm הוגדרה כתווית חיצונית), שהיא הפונקציה הראשונה שנקראת במערכת ההפעלה שלנו. שתי השורות הראשונות דוחפות למחסנית את הערך שנמצא באוגר EAX (שכפי שהוגדר ב-Multiboot Specifications, אמור להכיל את הערך 0x2BADB002) ואת הערך שנמצא באוגר EBX (שאמור להכיל את המצביע לטבלת ה-Multiboot Information). הסיבה שאנו דוחפים ערכים אלה למחסנית היא מוסכמת הקריאה (Calling Convention) - cdecl, שהיא מוסכמת הקריאה הסטנדרטית ב-C.

כך, הפונקציה kmain תוכל לקבל את הערכים האלו כפרמטרים. כאשר אנו כותבים ב-C את הקוד, אנו לא שמים לב בכלל ל-cdecl, אך בגלל שאנו כותבים את boot.asm בשפת סף ורוצים שהקוד יתממש עם קוד C, יש לממש את cdecl בעצמנו.

cdecl

כאשר אנו קוראים לפונקציה בקוד שלנו, קורה הרבה יותר בשפת המכונה מאשר קפיצה לאזור זיכרון אחר. בכדי לממש דבר זה יש צורך במוסכמת קריאה - שיטה שבה הקוד יוכל לקרוא לפונקציה, להעביר פרמטרים, לקבל תוצאה, ולהצליח לאחזר את מצב האוגרים לקדמותו (לפני הקריאה לפונקציה). המוסכמה בשפת ה-C היא שיטה בשם cdecl. המוסכמה (באופן פשוט. קיימים מקרי קצה שלא יתנהגו כך) היא כלהלן:

- הפרמטרים הנשלחים לפונקציה נדחפים למחסנית מימין לשמאל.
- הפונקציה נקראת.
- הפונקציה מחזירה את התוצאה באוגר EAX.
- הקורא לפונקציה מפנה את הפרמטרים מהמחסנית.

מוסכמת קריאה זו מאפשרת פונקציות עם מספר פרמטרים משתנה, כדוגמת printf. כדוגמה, אם נקח את קטע הקוד הבא:

```
int bla=foo(bar,foobar);
```

נראה כי פלט שפת הסף יראה בערך כך:

- דחוף את פרמטר foobar.
- דחוף את פרמטר bar.
- קפוץ אל התווית foo.
- קרא את התוצאה מ-EAX לתוך bla.
- הוצא מהמחסנית את הפרמטרים שדחפנו.

kernel.c

עתה נביט בקובץ kernel.c שנמצא בתיקייה src/kernel. קובץ זה מכיל את הגרעין שלנו - החלק החשוב ביותר במערכת ההפעלה. חלק זה מהווה, כפי שמצופה משמו, את הגרעין של פעולות מערכת ההפעלה - הוא מממש את כל שכבת ההפרדה בין החומרה לתוכנה. בקובץ זה נמצא את פונקציית kmain שהיא הפונקציה הראשית שלנו - אליה הקוד קופץ מ-boot.asm.

```
int kmain(void *mbd,unsigned int magic){
      if (magic!=0x2BADB002){
            return -1;
      }
      return 0;
}
```

ניתן לשים לב שהפונקציה kmain מקבלת שתי פרמטרים, כפי שהעברנו לה ב-boot.asm. הפונקציה בודקת שמספר ה-MAGIC הוא אכן כמצופה, ואם לא מחזירה סטטוס שגיאה.

הרצה

בניגוד לפרק הקודם, בפרק זה נממש שיטה יותר יעילה להפוך את הקוד שלנו למערכת הפעלה שניתן להריץ. אנו נכתוב קובץ MAKEFILE שיפשט לנו את תהליך ההידור, ההרכבה והקישור, ו-script ב-bash שיפעיל לנו אוטומטית את כל תהליך ההידור וההפעלה.

MAKEFILE

קובץ MAKEFILE הוא קובץ המנחה את תוכנת make. תוכנה זו היא כלי לביצוע אוטומציה של תהליך בניית התוכנה - ההידור והקישור. כאשר אנו קוראים לתוכנת make היא מחפשת קובץ בשם MAKEFILE (ללא שום סיומת) בתיקייה הנוכחית. קובץ MAKEFILE מורכב מאוסף של מטרות (targets), שלכל אחת

מהן הנחייה לגביה אופן הטיפול באותה מטרה. לכל מטרה יכולה להיות גם רשימה של מטרות שהמטרה הנוכחית תלויה בהן. תחביר הקובץ נראה כך:

```
target: dependencies
      commands
```

בנוסף, קובץ MAKEFILE יכול להכיל הכרזה על משתנים באמצעות אופרטור השוואה (= או :=, ההבל בין השניים מחוץ לנושא המאמר). בכדי לגשת לערך המשתנים האלה יש לעטוף אותם ב-$ וסוגריים, בקובץ MAKEFILE ניתן גם לבצע וקריאות לפקודות שירוצו ב-shell באמצעות התחביר:

```
$(shell CMD)|
```

לדוגמה, אם יש לנו פרוייקט עם שני הקבצים a.c ו-b.c, כאשר a.c תלוי ב-b.c, אנו יכולים לכתוב את ה-MAKEFILE שלו באופן הבא:

```
CC:=gcc
LD:=ld
all: a.c
      $(LD) a.c b.c
a.c: b.c
      $(CC) a.c
b.c:
      $(CC) b.c
```

כאשר אנו נקרא לתוכנת make ולא נציין שום מטרה, make תבצע את מטרת ברירת המחדל שלה - all. בנוסף נהוג להוסיף מטרה נוספת בשם clean שדואגת למחוק את כל הקבצים ש-make יצר ואינם נחוצים.

נביט על קובץ ה-MAKEFILE שלנו:

```
CC:=i586-elf-gcc
CFLAGS:=-c
…
all: kernel.o boot.o
      $(LD) $(LDFLAGS) kernel.o boot.o

kernel.o:
      $(CC) $(CFLAGS) kernel/kernel.c

boot.o: kernel.o
      $(ASM) $(ASMFLAGS) -o boot.o boot/boot.asm
clean:
      $(shell rm -rf *.o kernel.bin)
```

בוא נעבור על השורות:

```
CC:=i586-elf-gcc
CFLAGS:=-c
…
```

כפי שניתן לראות, בשורות הראשונות אנו מכריזים על משתנים. משתנים אלה קובעים איזה מהדר נשתמש, איזה דגלים יועברו ועוד. שימוש במשתנים אלה מקל עלינו בעת הרצון להשתמש במהדרים או בדגלים אחרים (למשל, נרצה להוסיף כמה דגלים שיתריעו לנו על כל חריגה מ-C סטנדרטי לחלוטין). כפי שניתן לראות בקובץ קיים מהדר ומקשר בעלי שמות מוזרים (i586-elf-gcc) - זאת מכיוון שאני משתמש ב-Cross-Compiler בכדי למנוע תלות בספריות שקיימות במערכת ההפעלה שבה אני מהדר את הפרויקט. ניתן למצוא מדריך לבניית Cross-Compiler בכתובת:

http://wiki.osdev.org/GCC_Cross-Compiler

```
all: kernel.o boot.o
	$(LD) $(LDFLAGS) kernel.o boot.o
```

לאחר מכן, אנו מכריזים על מטרה בשם all (היא המטרה שתבוצע כברירת מחדל). כפי שניתן לראות, קבענו שמטרה זו תלויה ב-2 מטרות אחרות. משמע, כאשר אנו מבצעים את מטרה זו, המטרות שהיא תלויה בהן תבוצענה קודם. בפקודות שמפורטות במטרה עצמה אנו מבצעים את הקישור של הקבצים שנוצרו. שימו לב שאנו יודעים שקבצים אלו קיימים כי הגדרנו שהמטרה תלויה במטרות שמייצרות אותם.

```
clean:
	$(shell rm -rf *.o kernel.bin)
```

במטרה זו אנו פשוט "מנקים אחרי עצמנו". אנו מוחקים את כל הקבצים הנגמרים ב-o. (קבצי אובייקט, התוצאה של הידור) ואת קובץ ה-bin שיצרנו מכל התהליך.

מעבר לקובץ ה-MAKEFILE, אנו נשתמש גם ב-script שנכתוב ב-bash, שיבצע את כל התהליך בפקודה אחת:

```
cd src
make
cp kernel.bin ../iso/kernel/kernel.bin
make clean
cd ..
mkisofs -R -input-charset utf8 -b boot/grub/stage2_eltorito -boot-info-
table -no-emul-boot -boot-load-size 4 -o os.iso iso
qemu -cdrom os.iso
```

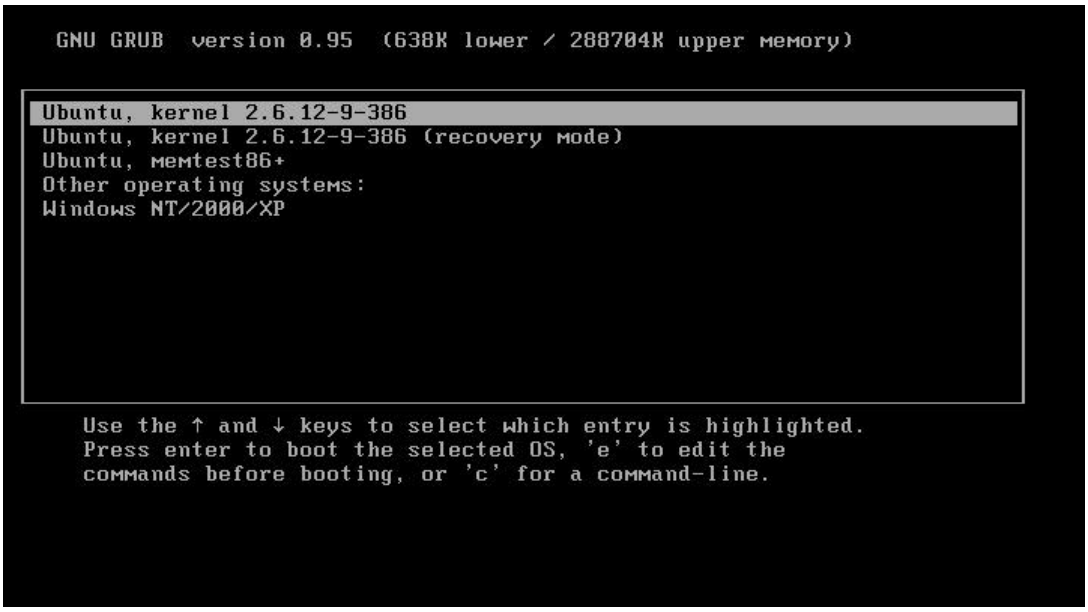
ניתן לראות שאנו מבצעים את כל תהליך ההידור, חיבור, קישור, העתקה והרצה ב-script זה, כך שלא נצטרך לכתוב את כל סדרת פקודות אלה כל פעם מחדש.

אם נביט טוב על הפקודה היוצרת את תמונת הדיסק (mkisofs), נראה שהיא שונה מהפרק הקודם. בניגוד לפרק הקודם, בו הגדרנו את הקובץ שיצרנו כתמונת האתחול, בפרק הזה אנו מגדירים קובץ בשם

stage2_eltorito כתמונה ממנה נבצע אתחול. זאת מכיוון שמפרק זה נשתמש ב-Bootloader חיצוני בשם GRUB.

GRUB

GRUB הוא Bootloader המהווה חלק מפרוייקט GNU. GRUB נמצא בשימוש כברירת מחדל ברוב ההפצות הנפוצות של לינוקס, וידוע ביכולתו לטעון מספר רב של סוגי קבצים שונים. בין היתר, GRUB גם יודע לטעון מערכות הפעלה בפורמט Multiboot, וזו הסיבה שמימשנו את הסטנדרט במערכת ההפעלה שלנו. ניתן להגדיר את GRUB על ידי קובץ בשם menu.lst, שקובע את רשימת מערכות ההפעלה שקיימות, את השמות שלהן, את הסדר שלהן ואת מיקומן. ניתן להגדיר ל-GRUB לטעון את מערכת ההפעלה מהזיכרון, מדיסק קשיח, מ"דיסק און קי", מפלופי, ואפילו מכתובת ברשת.



(מסך טעינה טיפוסי ל-Grub)

כעת, כל שנותר זה פשוט להריץ את ה-script שכתבנו:

```
./make.sh
```

מערכת ההפעלה, סימן V

כמו בפרק הקודם, עתה נתחיל להדפיס אל המסך. אך בניגוד לפרק הקודם, בו היינו ב-Real Mode, ויכולנו לבצע פסיקות BIOS, בפרק זה עברנו ל-Protected Mode, וכעת אנו לא יכולים לגשת ישירות לחומרה.

screen.c

הבה נביט על הקוד:

```
unsigned char *vidmem=(unsigned char *)0xb8000;

…
void screen_print(char *string){
      int i=0;
      while(string[i]){
            if(y==25){
                  screen_scroll();
                  y=24;
            }
            if(string[i]>0x1f&&string[i]!=0x7f){
                  vidmem[2*(x+y*80)]=string[i];
                  x++;
                  if(x==80){
                        y++;
                        x=0;
                  }
            }
            …
}
```

נביט על השורה הראשונה:

```
unsigned char *vidmem=(unsigned char *)0xb8000;
```

נראה שמוכרז בה מצביע לאזור בזיכרון הנמצא ב-0xb8000. כתובת זו מצביעה לאזור בזיכרון בו נוכל להציג טקסט למסך.

Text Mode Graphics

הכתובת 0xb8000 אינה מצביעה באמת לבייט ה-0xb8000- בזיכרון, אלא ממופה לזיכרון כרטיס המסך בעזרת MMIO. משמע, מבחינת הקוד שלנו, אנו כותבים באופן רגיל לכתובת זו. אך מבחינת המעבד, המידע נכתב לחיץ מיועד על כרטיס המסך. אזור זיכרון זה מאפשר לנו לכתוב תווים בצבעים שונים למסך. מצב ה-Text Mode הוא המצב בו אנו נמצאים לאחר איתחול בעזרת GRUB. במצב זה, המסך מחולק ל-

80 על 25 תאים, ולכל אחד מהתאים מוקצה 2 בתים - הראשון מייצג את ערך ה-ASCII של התו שהתא אמור להציג, והשני מייצג את התכונות של התו. הבית שמייצג את התכונה מחולק לשניים (או שלושה) חלקים. ארבעת הסיביות הראשונות מייצגות את הצבע של התו עצמו, וארבעת (או שלושת) הסיביות האחרונות מייצגות את הצבע של הרקע של אותו תא. במצב בו רק שלושה סיביות משומשת לייצוג צבע הרקע, הסיבית האחרונה קובעת האם התו יהבהב או לא. מצב זה מאפשר שימוש ב-16 צבעים בלבד.

```
void screen_print(char *string){
...
}
```

בפונקציה זו אנו פשוט מדפיסים את המחרוזת למסך. הפונקציה מדפיסה את התו למסך, יורדת שורה במקרה הצורך, גוללת את המסך במקרה הצורך, ואם התו הינו תו שאינו ניתן להדפסה אך מסמל קפיצה מסוימת (שורה חדשה, TAB וכו') היא "מדפיסה" אותו באופן מלאכותי.

הרצה

הנה היופי של ה-MAKEFILE מתגלה: בכדי להריץ את הפרוייקט אנו רק צריכים להוסיף עוד מטרה לקובץ ה-MAKEFILE (screen.o), לגרום ל-kernel.o להיות תלוי בה ולהריץ את ה-script שלנו מחדש!

מערכת ההפעלה, סימן VI

נשים לב לשלושה קבצים חדשים שנוספו:

- gdt.asm
- descriptors.c
- Idescriptors.h

שלושת הקבצים האלה אחראים על כתיבת ערכים ל-GDT.

GDT

GDT (Global Descriptor Table) הוא מבנה בזיכרון שנועד לתאר תכונות של אזורי זיכרון. ה-GDT מהווה את אחד ההבדלים העיקריים בין Protected Mode ל-Real Mode - ניתן להגדיר הגבלות גישה לאזורים מסויימים בזיכרון. ה-GDT מורכז מעד 8192 "שורות" שכל אחת מהן מתארת אזור יחיד שנקרא סגמנט, והיא נראית כך:

סיביות	שדה
0-15	16 הסיביות הראשונות של גבול הסגמנט שאותו השורה מתארת.
16-39	24 הסיביות הראשונות של בסיס הסגמנט שאותו השורה מתארת.
40-47	בית הגישה
48-51	4 הסיביות האחרונות של גבול הסגמנט.
52-55	דגלים
56-63	8 הסיביות האחרונות של בסיס הסגמנט.

בית הגישה נראה כך:

סיבית	שדה
7	Present. האם הסגמנט קיים. אמור להיות דלוק תמיד.
5-6	DPL. הטבעת (Ring) שאותה צריך בכדי לגשת לזיכרון.
4	תמיד 1.
3	Executable. אם הסגמנט דלוק הוא קוד, אם הוא מכובה הוא מידע.
2	Direction/Conforming • אם הסגמנט הוא מידע, הסיבית קובעת אם הסגמנט "גדל" למטה או למעלה. • אם הסגמנט הוא קוד: ערך של 0 קובע שניתן להריץ את הקוד רק מהטבעת שקבועה ב-DPL. ערך של 1 קובע שניתן להריץ את הקוד מטבעת שווה או נמוכה יותר משקבועה ב-DPL.
1	Readable/Writeable. אם סגמנט הוא קוד, הסיבית קובעת האם האזור ניתן לקריאה. אם הסגמנט הוא מידע הסיבית קובעת האם
0	Accessed. אנו לא אמורים לשנות סיבית זו. המעבד משנה אותה כאשר הוא ניגש לסגמנט שאותו השורה מתארת.

הדגלים נראים כך:

סיבית	שדה
3	Granularity. אם הוא מכיל 0 הסגמנט מיושר לגודל של בית, אם הוא מכיל 1 הוא מיושר לגודל של עמוד (4KB).
2	Size. אם הוא מכיל 0, הסגמנט הינו סגמנט בעל ערך 16 ביט, אם הוא מכיל 1 אז הסגמנט הוא סגמנט בעל ערך 32 ביט.
1	תמיד 0.
0	תמיד 0.

את ה-GDT טוענים לזיכרון בעזרת הפקודה lgdt. הפקודה מקבלת מתאר של ה-GDT שמורכבת משני בתים שמגדירים את גודל ה-GDT, ו-4 בתים של מצביע בזיכרון ל-GDT.

Ring

אחת ההגנות העיקריות ב-Protected Mode היא שיטת הטבעות - לכל קוד קיימת רמת הרשאה שונה ולכל רמת הרשאה יש הגבלות משלה. רמת ההרשאה נקבעת על ידי ה-DPL (קיצור של: Descriptor Privilege Level) והיא יכולה להיות בין-0 ל-3, כאשר 0 היא ההרשאה הגבוהה ביותר. למרות שקיימות ארבע אפשרויות, מערכות הפעלה נפוצות משתמשות רק בשתיים מהן - 0 לגרעין, ו-3 לקוד של המשתמש. ככל שהקוד שרץ כרגע נמצא בטבעת גבוהה יותר, כך הוא יכול להריץ פחות פקודות. לדוגמה, קוד שרץ בטבעת 3 לא אמור לגשת לחומרה, אלא דרך קריאה לקוד שנמצא בטבעת 0. דבר זה ממומש על ידי System Calls, והוא מהווה את אחד הנדבכים החשובים באבטחת המידע של המחשב האישי.

נצלול אל תוך הקוד:

```
…
extern void gdt_write(unsigned int);

struct gdt_entry_struct
{
```

```
    unsigned short limit_low;
    unsigned short base_low;
    unsigned char  base_middle;
    unsigned char  access;
    unsigned char  granularity;
    unsigned char  base_high;
} __attribute__((packed));

typedef struct gdt_entry_struct gdt_entry_t;

struct gdt_ptr_struct
{
    unsigned short limit;
    unsigned int base;
} __attribute__((packed));

typedef struct gdt_ptr_struct gdt_ptr_t;

gdt_entry_t gdt_set_gate(unsigned int base,unsigned int limit,unsigned
char access,unsigned char granularity)
{
      gdt_entry_t gdt_entry;
      gdt_entry.base_low=(base&0xFFFF);
      gdt_entry.base_middle=(base>>16)&0xFF;
      gdt_entry.base_high=(base>>24)&0xFF;
      gdt_entry.limit_low=(limit&0xFFFF);
      gdt_entry.granularity=(limit>>16)&0x0F;
      gdt_entry.granularity|=granularity&0xF0;
      gdt_entry.access=access;
      return gdt_entry;
}

void gdt_setup()
{
      gdt_entry_t gdt_entries[5];
      gdt_ptr_t gdt_ptr;
      gdt_ptr.limit = (sizeof(gdt_entry_t) * 5) - 1;
      gdt_ptr.base  = (unsigned int)&gdt_entries;
      gdt_entries[0]=gdt_set_gate(0, 0, 0, 0);
      gdt_entries[1]=gdt_set_gate(0, 0xFFFFFFFF, 0x9A, 0xCF);
      gdt_entries[2]=gdt_set_gate(0, 0xFFFFFFFF, 0x92, 0xCF);
      gdt_entries[3]=gdt_set_gate(0, 0xFFFFFFFF, 0xFA, 0xCF);
      gdt_entries[4]=gdt_set_gate(0, 0xFFFFFFFF, 0xF2, 0xCF);
      gdt_write((unsigned int)&gdt_ptr);
}
```

נביט על השורה הראשונה:

```
extern void gdt_write(unsigned int);
```

בשורה זו מוכרזת הפונקציה gdt_write, והיא מוכרזת כפונקציה שמוגדרת במקור חיצוני. אנו יכולים לממש פונקציה זו רק בשפת סף, ולכן כתבנו אותה בקובץ חיצוני, gdt.asm.

```
struct gdt_entry_struct
{
    unsigned short limit_low;
    unsigned short base_low;
    unsigned char  base_middle;
    unsigned char  access;
    unsigned char  granularity;
    unsigned char  base_high;
} __attribute__((packed));

typedef struct gdt_entry_struct gdt_entry_t;

struct gdt_ptr_struct
{
    unsigned short limit;
    unsigned int base;
} __attribute__((packed));

typedef struct gdt_ptr_struct gdt_ptr_t;
```

שורות קוד אלו מגדירות את הטיפוסים שמתארים ערך אחד ב-GDT, ואת המצביע אותו אנו הולכים לטעון לפקודה lgdt. ההכרזה "__attribute__((packed))" מדריכה את המהדר לא לבצע שום ריפוד בין איברי הטיפוס ולא לשנות את היישור שלהם.

```
void gdt_setup()
{
      gdt_entry_t gdt_entries[5];
      gdt_ptr_t gdt_ptr;
      gdt_ptr.limit = (sizeof(gdt_entry_t) * 5) - 1;
      gdt_ptr.base  = (unsigned int)&gdt_entries;
      gdt_entries[0]=gdt_set_gate(0, 0, 0, 0);
      gdt_entries[1]=gdt_set_gate(0, 0xFFFFFFFF, 0x9A, 0xCF);
      gdt_entries[2]=gdt_set_gate(0, 0xFFFFFFFF, 0x92, 0xCF);
      gdt_entries[3]=gdt_set_gate(0, 0xFFFFFFFF, 0xFA, 0xCF);
      gdt_entries[4]=gdt_set_gate(0, 0xFFFFFFFF, 0xF2, 0xCF);
      gdt_write((unsigned int)&gdt_ptr);
}
```

בפונקציה זאת אנו יוצרים 5 ערכים שלאחר מכן נטען ל-GDT. הערך הראשון ממולא באפסים. ערך זה נקרא Null Descriptor, והוא תמיד חייב להופיע כערך הראשון.

כמו הרבה דברים בתחום פיתוח מערכות הפעלה, לא בדיוק ידוע מדוע יש צורך ב-Null Descriptor, מעבדים עוצבו כך שהם אינם ניגשים לערך ה-0 ב-GDT, ולכן לא יקראו את השורה. מניסיוני האישי מעבדים בד"כ "שורדים" אם שורה זו לא ממולאת באפסים. לכן, רבים משתמשים בה בכדי לטעון לתוכה את הכתובת של ה-GDT, בכדי לפשט את הקריאה ל-LGDT.

לאחר מכן, אנו מגדירים סגמנט קוד וסגמנט מידע עם הרשאות לגרעין, וסגמנט קוד ומידע עם הרשאות למשתמש.

עתה נביט על קוד שפת הסף שטוען את המידע ל-GDT:

```
                                                    global gdt_write

gdt_write:
      mov eax, [esp+4]
      lgdt [eax]
      mov ax, 0x10
      mov ds, ax
      mov es, ax
      mov fs, ax
      mov gs, ax
      mov ss, ax
      jmp 0x08:flush
flush:
            ret
```

בתחילת הקוד אנו מכריזים על התווית gdt_write כתווית שניתן לקרוא לה מבחוץ. כך קוד ה-C שכתבנו ב-descriptors.c יוכל לקרוא לקוד כאילו היה פונקצית C רגילה.

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

בשורות אלו אנו טוענים את הפרמטר שמועבר בקריאה לפונקציה (ראה cdecl), וטוענים את הערך אליו הפרמטר מצביע לתוך האוגר ששומר את הכתובת של GDT.

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

בשורות אלו אנו ממלאים את כל האוגרים שאחראים לסלקטורים (אוגר המציין, בין היתר, את מספר ה"שורה" ב-GDT שמתאר את הסגמנט אליו אנו רוצים לגשת) בערך 0x10.

```
jmp 0x08:flush
```

בשורה זו אנו מבצעים FAR JUMP - קפיצה לאזור קוד "רחוק". במקרה זה הקוד שאנו קופצים אליו לא באמת רחוק, אך אנו מבצעים קפיצה זו בכלל מקרה כי היא דואגת לנקות מספר אוגרים (בין היתר, היא משנה את האוגר CS) ולאתחל אותם מחדש לערכים השאנו רוצים שהם יכילו לאחר המעבר ל-Protected Mode.

השלכות על אבטחת המידע

המעבר ל-Protected Mode בכלל, וה-GDT בפרט, מהווים נכס גדול לאבטחת המידע של מחשבים אישיים. לולא היינו יכולים לנהל את הגישה לזיכרון ברמה חומרתית, לא ניתן היה לאבטח בצורה מהותית את המחשב. כמו כל כלי שנועד להגן, קיימות מספר חולשות שניתן לנצל אותן בכדי לעקוף מנגנון זה. בין היתר, נמצאו כמה חולשות שנוצלו על ידי Rootkits שונים שאפשרו להוסיף שורה נוספת ב-GDT. בשורה זו הוגדר סגמנט שחולש על כל הזיכרון ובעל הרשאות מלאות. כך הקוד הזדוני יכול לגשת לאיזה מקום בזיכרון שירצה.

סיכום

בפרק זה עסקנו במעבר ל-Protected Mode, ב-Bootloader, כתיבה למסך ויצירת GDT. עברנו על המשמעויות של כל רכיב ועל היתרונות שהוא מספק למערכת ההפעלה. בפרק הבא נעסוק בפסיקות וטבלת ה-IDT, ונתחיל לבנות את התשתית להפרדה בין קוד הגרעין לקוד המשתמש.

אתגר

למי שרוצה להרחיב את מערכת ההפעלה שנכתבה עד כה - ניתן לכתוב פונקציה שתדפיס למסך את ה-Multiboot Information שמועבר ל-kmain. בהצלחה!

לקריאה נוספת

מדריכים ב-Wiki שקשורים לפרק זה:

- http://wiki.osdev.org/GDT_Tutorial
- http://wiki.osdev.org/Protected_mode
- http://wiki.osdev.org/Global_Descriptor_Table

הגדרות ה-Multiboot:

- http://www.gnu.org/software/grub/manual/multiboot/multiboot.html

לינק להורדת קוד המקור של Linux גרסה 1.0 - מקור טוב מאוד לקוד מסודר ועובד:

- http://www.kernel.org/pub/linux/kernel/v1.0/
- http://en.wikipedia.org/wiki/Text_mode
- http://wiki.osdev.org/Text_UI

תיקונים והערות לפרק הקודם:

תודה רבה לכל הקוראים שהראו התעניינות במאמר. ארצה להציף ולתקן כמה טעויות, הערות והארות שזוהו על ידי הקוראים.

בטבלה שהציגה את ההבדלים בין מצבי הפעולה השונים הועבר הרושם המוטעה שדפדוף וסגמנטציה הם תכונות שקיימות רק במצב הפעולה שמיישמים אותם, בהתאמה. יש לציין שזה אינו המצב, אלא הנוהג. כראיה, ניתן לראות שכחלק מפרק זה אנו יישמנו סגמנטציה ב-Protected Mode דרך ה-GDT. אך, בניגוד ל-Real Mode, סגמנטציה זו אינה באה לידי שימוש ככלי לניהול אזורי זיכרון - אנו הגדרנו 4 סגמנטים שכולם חולשים על כל הזיכרון הקיים.

הקורא "Salad" ציין שנפלה טעות בהסבר על ה-BIOS. במאמר נכתב שלוח האם טוען את ה-BIOS, אך במציאות המעבד הוא האחראי על פעולה זו. המעבד מתוכנת מראש לטעון בעת ההדלקה את הפקודות שקיימות בכתובת 0xFFFFFFF0-0xFFFFFFFF. דבר זה קורה כשאנו עדיין ב-Real Mode. סדרת כתובות אלו היא היחידה מעל ה-Address Space הרגיל ב-Real Mode שניתן לגשת אליה, וזאת מכיוון שבעת הדלקת המחשב כתובת הבסיס באוגר ה-CS נקבע כברירת מחדל ל-0xFFFF0000. הפקודה בכתובת זו נקראת ה-Reset Vector, והיא בד"כ מכילה קפיצה לאזור אחר בזיכרון (JMP 0xF0000). הכתובת שאלייה אנו קופצים, בדומה לכתובת שלה כתבנו ב-Text Mode, ממופה בעזרת ה-MMIO לקוד ששמור בצ'יפ ה-BIOS, וכך המעבד יכול לקרוא את ה-BIOS.

DLP - Data leakage Prevention

מאת אלכס ליפמן

הקדמה

בעידן שבו המידע נדרש לנוע באופן דינאמי בתוך הארגון ומחוץ לארגון, חברות צריכות להתמודד עם תופעת דלף מידע גם באמצעים אקטיביים. ארגונים נדרשים לשמור על סודיות המידע, בין היתר, כדי לשמור על נכסיהם הרוחניים, תדמיתם ולעמוד בהוראות החוק והרגולציה.

דלף מידע עשוי לנבוע מסיבות שונות ובהן טעויות אנוש, תקלות טכניות ופעולות ריגול עסקי של מתחרים. אובדן מחשבים ניידים המכילים מידע עסקי רגיש, טעויות הפצת דואר אלקטרוני והחדרת תוכנות ריגול וסוסים טרויאניים הפכו לשגרה המחייבת ארגונים לנקות בצעדים לצמצום התופעה.

לאור ריבוי האיומים, יותר ויותר ארגונים מטמיעים מערכות ייעודיות בתחום ניתוח תוכן לצורך מניעת דלף מידע. ביקוש זה הביא בעשור האחרון לבשלות טכנולוגית ולריבוי של מוצרים בתחום ניתוח התוכן, אשר רובם אף נרכשו על ידי ענקי ה-IT ואבטחת המידע הגדולים ם ושולבו בסל מוצריהם.

יחד עם זאת, תהליך הטמעה אפקטיבי של טכנולוגיות אלו מצריך מארגונים היערכות מיוחדת, בדגש על תהליכים ארגוניים שאותם הם נדרשים לבצע בשילוב עם יישום המוצרים הטכנולוגים. במקרים בהם הטמעת טכנולוגיה מבוצעת בניתוק ממחזור החיים של המידע בארגון והתהליכים העסקיים השונים, ומבלי שקיים גורם פרויקטאלי האחראי באופן מתמשך על הנושא, הארגון עלול שלא למצות את הפוטנציאל הגלום בטכנולוגיה זו.

הגדרת מושג DLP

לעתים קיימת אי בהירות לגבי מושג DLP, ואילו פתרונות נכנסים לקטגוריה זו. לעתים מתייחסים אליו כשם כולל למוצרים המיועדים לשמור על סודיות המידע, כך שמוצרים שונים, החל מפתרונות הצפנה ועד למגרסות נייר משווקים תחת כותרת - DLP.

כדי להגדיר באופן מדויק יותר את מושג ה-DLP ניתן להשתמש בהגדרה הבאה:

> מערכות למניעת דלף מידע בפועלות לזיהוי, ניטור ואבטחה של מידע המועבר בתקשורת והמצוי במאגרי מידע וביחידות הקצה, באמצעות ניתוח עמוק של תוכן ותוך שימוש בממשק ניהול מרכזי.

האיום

ניתן לחלק את האיומים על סודיות המידע בארגון לקטגוריות הבאות:

1. **גניבת מידע מהארגון במתכוון**, ממניעים של ריגול עסקי תוך שימוש באמצעים לוגיים כגון וירוסים וסוסים טרויאניים, כדוגמת פרשיית הסוס הטרויאני משנת-2004, של זוג אפרתי, אשר השפיעה, בין היתר, על חברות טלקום מובילות בארץ.

2. **גניבת מידע מהארגון תוך שימוש בגורם פנימי, אנושי, המופעל על-ידי גורם חיצוני**. אירועים מסוג זה יכולים להתבצע על-ידי הוצאת מידע מהארגון דרך הקישור לרשתות חיצוניות, העתקת מידע מתחנות קצה למדיה נתיקה ואף הוצאת מסמכים בתצורה הקשיחה.

3. **דלף מידע הנגרם כתוצאה מכשל טכני, טעויות אנוש או חוסר מודעות עובדים**. לדוגמא, דלף מידע כתוצאה מטעויות הפצה של דוא"ל, Meta-Data בקובצי Office, העדר היכרות של עובדי הארגון עם נהלים והיבטי רגישות המידע בארגון.

4. **דלף מידע מספקים של הארגון המחזיקים במידע רגיש** כתוצאה מאי-יישום של אמצעי הגנה על המידע כפי שמיושם בארגון עצמו, לדוגמא, דלף מידע מספקים כגון בתי דפוס, משרדי פרסום עורכי דין וכו'.

פתרונות טכנולוגיים

ניתן לחלק את מוצרי ה-DLP לשתי משפחות עיקריות:

- **Host Based - זיהוי תוכן ומניעת דלף מתחנות קצה.** טכנולוגיה זו עושה שימוש בסוכן (Agent) המותקן ביחידת הקצה ומנטר ערוצים דרכם המידע יכול להיות מועבר מיחידת הקצה, כגון העתקה למדיה נתיקה, שליחת דוא"ל, שימוש בתוכנות מסרים מיידיים, CD ואף זיהוי תוכן המועתק ל- Clip Board.

- **Network Based - זיהוי ומניעת דלף של תוכן העובר ברשת (בד"כ בקישור החיצוני).** בשיטה זו נעשה שימוש בשרת הממוקם בקישור החיצוני של החברה, המבצע ניתוח פרוטוקולי תקשורת בערוץ היוצא מהארגון, כגון, ניטור נתונים המועברים בדוא"ל, גלישה, FTP, תקשורת של תוכנות מסרים מיידיים ועוד.

יש לציין, כי בעשור האחרון התרחשו מספר מיזוגים ורכישות, במהלכן חברות ענק בתחום מוצרי אבטחת מידע רכשו חברות מוצר באופן שכיום מוצרי DLP של מרבית ענקי אבטחת המידע כוללים הן טכנולוגיית ה-Host Based והן ה-Network Based.

אחד מרכיבי היסוד של טכנולוגיות DLP הינו ה-CMF (קיצור של: Content monitoring and Filtering).

להלן שיטות עיקריות לזיהוי התוכן:

- **זיהוי על-בסיס מילות מפתח:**

בשיטה זו, מערכת DLP תזהה את התוכן ה"דולף" על בסיס מילות מפתח שהוגדרו מראש. בטכנולוגיות המיושמות כיום, בכל מוצרי DLP ניתן לקבוע מילים בודדות, רצף מילים ואף לבנות בסיסי נתונים הכוללים חוקים משוואות המבוססות על מילות מפתח תוך שימוש בכללים בוליאניים כגון AND ו-NOT.

לדוגמא, ניתן לקבוע שאם במסמך הנשלח מחוץ לארגון באמצעות הדואר האלקטרוני קיימת מילה "סודי" או שילוב של מילים "הצעה" וגם "לפטנט", מערכת ה-DLP תייצר הודעה ואף תחסום את מעבר תוכן ההודעה מחוץ לארגון. יתרון בשיטה של קביעת משוואות ניטור תוכן על-בסיס מילות מפתח הינו בכך שניתן לאפיין בדיוק איזה תוכן רגיש רוצים לנטר ובכך לצמצם את כמות התראות השווא.

עם זאת, על מנת לנסח משוואות ולהגדיר מילות מפתח מתאימות נדרשת היכרות מעמיקה עם הסביבה הארגונית ועם המידע הקיים בארגון.

- **זיהוי על בסיס חתימות:**

בשיטה זו, מערכת DLP מבצעת השוואה בין תוכן הזורם מחוץ לארגון לתוכן שנמצא במאגר המידע של המערכת והוגדר מראש כ"מסווג". כיום במרבית מוצרי DLP מיושמת טכנולוגיה העושה שימוש בחתימות סטטיסטיות שמיועדות לזהות מקרים שבהם מסמך מסוים זהה או דומה למסמך שמראש הוגדר כרגיש. בין היתר, שיטה זו נועדה להקל על פעולת ניסוח משוואות ניטור מדויקות. למשל ניתן להגדיר צבר של מסמכים הממוקמים בתיקיית "פטנטים" בשרת קבצים כרגישים וכל מסמך שיועבר מחוץ לארגון בערוצים המנוטרים ע"י ה-DLP ינותח מול המאגר של המסמכים הרגישים. כאמור, כיום מיושמות טכנולוגיות המסוגלות לא רק לזהות מסמכים דומים אלא אף מסמכים שקיים ביניהם דמיון סטטיסטי.

מעבר לשתי שיטור זיהוי תוכן שפורטו, קיימות שיטות נוספות ששילוב ביניהן מאפשר יצירת כללי ניטור אפקטיביים. בין השיטות הנוספות ניתן למנות את הזיהוי על-פי תוכנה שמייצרת את המסמך כגון Office או Paint, יוזר במערכת ההפעלה שייצר את המסמך, תחנת קצה בה נוצר המסמך, נמענים בתפוצת דואר אלקטרוני עוד.

בנוסף קיימות טכנולוגיות לזיהוי תוכן על-בסיס מבנה, לדוגמא, ניטור מספרים הדומים במבנה למספרי כרטיסי אשראי.

ניתן לשלב מספר טכנולוגיות ניטור למשוואה אחת, לדודמא, זיהוי מסמכים הכוללים מילים "סרטוט" וגם (AND) "פרוייקט X" אשר נוצרו בתוכנת AutoCAD ע"י משתמשים השייכים לקבוצת "מחקר" ב- Active Directory.

פרוייקט DLP בארגון

אחד האתגרים המרכזיים הניצבים בעת ההגנה על סודיות המידע בארגונים הינו דינמיות המידע בשילוב מורכבות התהליכים העסקיים. היעדר היכרות עם סוגי המידע הקיימים בארגונים והתהליכים הקשורים לשימוש בו, עשוי להפוך את היכולת לפקח אחר המידע העסקי והרגיש למשימה קשה.

לצורך יישום אפקטיבי של מערכות DLP, ראשית על ארגונים לענות על שאלות מהותיות כגון: מהו המידע הסודי? היכן הוא מאוחסן? מה הם הערוצים דרכם המידע עלול לזלוג? ולבסוף - אילו פעולות יינקטו אם וכאשר יתרחש אירוע של דלף מידע סודי.

להלן תמצית פעולות, אותן מומלץ לבצע תוך כדי יישום מערכת DLP:

שלבי הפרויקט:

1. **ניתוח הסביבה העסקית והאיומים הקיימים (פנימיים/ חיצוניים).** שלב זה נועד למיפוי האיומים על הארגון. אם מדובר הארגון בטחוני, ככל הנראה תהליכי אפיון מידע מסווג יהיו מורכבים לאומת הארגון שכל מה שהוא מנסה לאבטח הם מספרי כרטיסי האשראי הנמצאים במאגרי המידע בארגון. בשלב זה נדרש למפות את האיומים, לרבות מי הם הגורמים היכולים לגרום לדלף מידע, מהם המניעים שלהם, ומהי רמת היכולות של אותם גורמי האיום.

2. **סיווג המידע - הגדרת המידע הסודי/רגיש וסיווגו לפי רמת רגישות** בשלב זה נדרש להגדיר מהו המידע הסודי עליו רוצים לשמור. להלן דוגמאות למידע שעשוי להיות מוגדר בארגון כסודי:

   - קניין רוחני.
   - מידע הקשור לצנעת הפרט כגון מידע רפואי, אישיים ואף צבר של מידע.
   - מידע פיננסי/עסקי.
   - מידע השייך ללקוחות.
   - מידע ביטחוני.
   - מידע טכני אודות מערכות מידע בארגון שעשוי לחשוף חולשות אבטחה.

   שיטות לביצוע תהליך סיווג מידע (DATA CLASSIFICATION):

   - **שיטה ידנית**: קביעת תג סיווג עבור כל מסמך כחלק מתהליך יצירת המסמכים.
   - **שיטה אוטומטית**: שימוש במערכת לומדת/מומחה לקריאת מידע ומתן תגי סיווג למידע באופן אוטומטי.

3. **זיהוי ומיפוי מקומות אחסון של מידע סודי/רגיש**. מטרת פעילות זו הינה לענות על השאלה, היכן ממוקם המידע הרגיש? לצורך זה ניתן לעשות שימוש בכלים E-Discovery שלעתים הנם מודול במערכת ה-DLP. במהלך הפעילות, ימופו מקומות האחסון כגון:

   - זיכרונות ניידים/ מדיה
   - בסיס נתונים
   - שרתי קבצים
   - שרתי אחסון/גיבוי/קלטות
   - אמצעים ניידים - סלולאריים / PDA
   - תחנות קצה

4. **מיפוי וניתוח תהליכים עסקיים ומחזור החיים של המידע בארגון.** לאחר שבוצע מיפוי של המידע הרגיש ומקומות בהם הוא ממוקם, יש למפות את התהליכים העוברים על המידע. מידע יכול להיווצר בתחנת הקצה של המשתמש, לאחר מכן להישלח בדואר האלקטרוני בתוך הארגון, לעלות לשרתי קבצים או למערכות ניהול ידע, להיות מופץ שוב ובכל שלב בזרימת המידע קיים סיכון של דלף.

השלבים המקובלים במחזור החיים של מידע הנם:

- יצירת מידע
- הפצה
- שימוש
- תחזוקה/עדכון/עיבוד
- העברה לארכיון/גיבוי
- השמדה

5. **מיפוי והערכת ערוצי הדלף הפוטנציאלים.** שלב זה נועד לתעדף ולמקד את יישום אמצעי ה-DLP והתהליכים הנלווים לערוצי דלף המסוכנים יותר. לדוגמא, בארגון בו קיימת הקשחה אפקטיבית של תחנות קצה מפני חיבור מדיה נתיקה, רצוי למקד את המאמצים סביב ניטור תוכן היוצא דרך הקישור החיצוני ולהיפך.

ערוצי דלף אפשריים:

- ממשקים וקישורי רשת חיצוניים
- מדיה וזיכרונות ניידים
- מבקרי רשת מזדמנים (ספקים חיצוניים או עובדים זמניים המחברים יחידות קצה לרשת הארגונית באופן זמני).
- פקסים ומדפסות
- אמצעי מחשוב ניידים (עולם ה-Mobile)
- גורמים אנושיים

6. **אפיון דרישות, בחירת מוצרים והטמעה, לרבות התאמה ועיצוב מדיניות, נהלים, תהליכי תגובה ואמצעים משלימים.**

לצורך יישום אפקטיבי של מערכות ה-DLP נדרש לטפל בכלל היבטי אבטחת מידע הארגוני:

- **נהלי עבודה והנחיות למשתמשים** - הכנסת שינויים בנהלים הקיימים לגבי אופן ניהול המידע בארגון, כגון סיווגים ידניים, אופן העברת מידע לגורמים חיצוניים וכו'.
- **תהליכי הטיפול באירועים** - יישום מערכת DLP דורש התאמת תהליכי תגובה וטיפול באירועי הדלף, לרבות זמני תגובה של אנשי אבטחת מידע, שיטות תחקור האירועים, מזעור נזקים וביצוע חקירות. הדבר תלוי באופי הארגון וביכולות של צוות אבטחת המידע.
- **אחריות ותפקידים** - ייתכנו שינויים בהגדרות של בעלי תפקידים מסוימים, כגון מינוי בעלי תפקידים חדשים שיהיו אחראים על שלבים השונים במחזור החיים של מידע).
- **תהליכי מחזור החיים של מידע בארגון** - ייתכנו שינויים במחזור החיים של מידע, לדוגמא, הקפדה על קביעת תגי סיווג לכל מסמך בשלב היצירה. העברת/שיתוף מידע בתוך הארגון באמצעות מערכות ניהול ידע וכו'.
- **נתיב ביקורת, לוגים וחיבור למערכות SIEM**. לצורך שיפור יכולת תחקור אירועי דלף, נדרש לטייב את תקינות הלוגים במערכות נלוות Active Directory/DC.
- **טיפול בערוצים סמויים** - על מנת לוודא כי מירב ערוצי דלף מידע מכוסים ע"י מערכות DLP, נדרש לבצע התאמה של תהליכי עבודה וארכיטקטורת מערכות המידע באופן שלא יאפשר קיום ערוצים סמויים (לא מנוטרים). לדוגמא, הקשחת תחנות קצה מפני מדיה נתיקה, הגבלה של פרוטוקולי תקשורת הניתנים לטיפול ע"י DLP (חסימה של תוכנות Peer to Peer).

לאחר היישום - סקירה והערכה לצורך שיפור וייעול הביצועים של הפתרונות המיושמים. במערכות DLP, בדומה למערכות ניטור אחרות, יש צורך בלימוד מתמיד ושיפור של איכות חוקי ניטור/משוואת בהתאם לאופי האירועים וכמות התראות השווא המיוצרות ע"י המערכת.

סיכום

על מנת למפות את מחזור החיים של המידע - עוד בטרם בחירת המוצר המתאים, נדרשים מנהלי מערכות המידע ומנהלי אבטחת מידע לזהות את המידע הקיים ואת אופן ניהולו וניודו בארגון. על כן, עליהם לענות ראשית על שאלות מהותיות כגון: מהו מידע סודי? היכן הוא מאוחסן? מה הם הערוצים דרכם המידע עלול לזלוג? ולבסוף, אלו פעולות יינקטו אם וכאשר יתרחש אירוע של דלף מידע סודי?

חברות נדרשות להתייחס להיבטים ארגוניים הכרוכים בהקמה ותפעול של מערך ה-DLP. בין היתר, נדרש תיאום בין פונקציות רבות בארגון, כגון: הנהלת החברה, מנהלי אבטחת מידע, היועץ משפטי, גורמי ה-IT, נציגי היחידות העסקיות, ועוד.

כיום, כבר קיימת הכרה בכך שארגונים המעוניינים בהטמעת מוצרים למניעת דלף מידע שתוצאותיהם מורגשות לטווח ארוך, לא יכולים להסתפק בהתקנת כלים טכנולוגיים בלבד אשר רק מפקחים ומסננים תוכן ב"ערוצים יוצאים" ופועלים על בסיס חוקי זיהוי חריגות תוכן כפי שנקבעו ע"י היצרנים, לדוגמא חוקי זיהוי מספרי כרטיסי אשראי.

לשם מיצוי הפוטנציאל טכנולוגי, נדרשת פעילות הכנה תומכת להטמעת המוצרים, המבוססת על מתודולוגיה מובנית ומסודרת שתאפשר בניית חוקי זיהוי בהתאם לסוגי המידע הרגיש והתהליכים העסקיים המשפיעים עליו. פעילות מתודית זו תאפשר התמודדת אפקטיבית עם מקרים של דלף מידע ואף הפחתת דיווחי סרק רבים, שאופייניים לפתרונות הטכנולוגיים העוסקים בזיהוי חריגות.

פרטיות תעסוקתית

מאת עו"ד יהונתן קלינגר

הקדמה

מהו היקף הזכות לפרטיות במקום העבודה וכמה מותר למעסיק לחטט במחשבו של העובד או בחומרים שנמצאים בחזקתו; על שאלה זו אנסה לענות במאמר קצר זה, אשר מהווה הרחבה ועדכון של הרצאה שהעברתי באוגוסט 2010. מאז ועד היום חלו לא מעט התפתחויות, אשר העיקרית בהן היא פסק הדין איסקוב (עע 90/08 טלי איסקוב ענבר נ' הממונה על חוק עבודת נשים היא התחלה מצוינת בנושא פרטיות במקום העבודה) אשר הפך להלכה חלוטה וקבועה שמקבעת את הפרטיות במקום העבודה.

ההחלטה של בית המשפט בנושא איסקוב באה בצורה תמוהה בלי הבנה טכנולוגית; כפי שאראה בהמשך, ועשויה לצור דיסוננס. אלא, שחשוב כבר עכשיו לדעת: אם עיקר עבודתך היא להיות ממונה על אבטחה בארגון, עלייך לעבוד בצמוד לייעוץ המשפטי ולקבל את הפרשנות של אותו ארגון לפסק הדין איסקוב. חשוב לי להבהיר שהמאמר הקצר הזה אינו תחליף לייעוץ משפטי ומהווה רק פרשנות שלי של הדין הקיים, שעשויה להתפס כמחמירה, אך זהירה יותר.

השאלה של פרטיות במקום העבודה היא שאלה משפטית מאוד (וראו גם מעקב בעבודה: טיילור, בנת'האם והזכות לפרטיות, **מיכאל בירנהק**) אבל היא גם שאלה טכנולוגית; כדי להבין כיצד פרטיות במקום העבודה מושגת\נפגעת, יש צורך להבין את המנגנונים הקיימים כיום קודם כל, ומהם האינטרסים והאיזונים. האינטרס של המעביד הוא שליטה מוחלטת בעובד: קבלת מידע מקסימאלי בכל רגע (וראו Workplace Privacy and Employee Monitoring Fact Sheet, Privacy Rights Clearinghouse) והעובד מעוניין להעזב בשקט (כדברי **Louis D. Brandeis** ו**Samuel D. Warren** במאמרם מ-1980: The Right to Privacy). למעביד ישנם אינטרסים מובהקים כדי לעקוב אחר התנהגויות של עובדים במקרים בהם הוא מעוניין להגן על סודותיו המסחריים מפני גזילה (בש"א 10105/07 כלל פיננסים נ' בני דקל), ולהגן על כך שעובדיו לא יעדרו ממקום העבודה (עמר"מ 13028-04-09 בנימין אליהו נ' עיריית טבריה) או יבצעו בו שימושים אישיים.

אלא שכדי להבין את מהות הזכות לפרטיות במקום העבודה, ובטרם נתחיל לדון בדיני העבודה והייחוד שלהם, יש להבין את הזכות לפרטיות. הזכות לפרטיות אינה זכות מוחשית כמו הזכות לקניין (מיכאל

בירנהק, שליטה והסכמה: הבסיס העיוני של הזכות לפרטיות) אלא היא זכות מופשטת כמו הזכות לכבוד; לא בצורה מקרית מאוגדת הזכות לפרטיות בתוך חוק יסוד: כבוד האדם וחירותו, וחוק הגנת הפרטיות מגדיר את הפגיעה בפרטיות בצורה קאוזאלית, כאשר סעיף 2(1) לחוק מותיר קשת רחבה של מקרים להגדרת הפרטיות על ידי "הטרדה אחרת" (וראו לעניין זה דנ 9/83 בית הדין הצבאי לערעורים נ' ועקנין), אולם דומה ששינויי הטכנולוגיה והמהות מונעים את היכולת להגדיר מהי פגיעה בפרטיות, עד כדי שלעיתים מעשה שלא היה פוגע בפרטיות אם היה נעשה מול אדם אחד או פעם אחת פוגע בפרטיות כאשר הוא מבוצע כלפי ציבור או בצורה מתמשכת (Maynard v. United States 08-0303).

כלומר, את הזכות לפרטיות ניתן להגדיר בצורה חברתית (א 6023/07 אפריאט נ' ידיעות אחרונות) כתלויה בנסיבות העניין ובהתאם לתחושת הבטן. אלא שהעדר היכולת לספק הגדרה בטוחה של מהי פגיעה בפרטיות מונעת גם את היכולת לספק ייעוץ משפטי ודאי לעובדים או מעבידים.

מהות הזכות לפרטיות היא טיפול במידע על אודות הפרט, על רכושו שמהווה את זרועו הארוכה (כגון רכב או טלפון סלולרי), ועל הגלישה שלו באינטרנט (רע"א 4447/07 רמי מור נ' ברק) ופרסומים שהוא ביצע. בניגוד לתפיסות עבר, הזכות לפרטיות אינה קיימת רק על התנהגויות של אדם ברשות הפרט (בג"צ 6650/04 פלונית נ' בית הדין הרבני האזורי בנתניה) או במקומות שהם קואזי-פרטיים, אלא גם על מקומות ציבוריים (רע"א 6902/02 צדיק נ' ליבק, עניין *אפריאט*). אולם, הפסיקה קבעה חד משמעית כי **הזכות לפרטיות היא זכות אנושית** (ע"א 7151-10-09 גרפונט בע"מ ואח' נ' גיא פלד ואח') ולא חלה על תאגידים; מעבר לכך, הפסיקה גם הכירה בכך שהפגיעה בפרטיות לא יכולה להתרחש כאשר אדם מראש מתפרנס מעצם הצגת דמותו (תא (חי') 534-08 חוה קורן ישראלי נ' שי כהן). אבל במקום לומר כיצד הפרטיות אינה מוגדרת או מוגדרת, קיומה של פרטיות תלוי יותר מכל בתחושת הבטן הסובייקטיבית של הפרט.

למעביד ישנן דרכים רבות לשלוט בהתנהגות העובד, וחשוב להבין שפגיעה בפרטיות העובד לא נועדה לאיסוף המידע, אלא לייצר תחושה של פיקוח אשר תשלוט בהתנהגויות נורמטיביות או מותרות. המעביד משתמש במספר כלים כיום כדי לפקח ולשלוט בעובדיו; אלא חשוב לזכור כי טכנולוגיות חדשניות עשויות לערער ולשנות את המסד לשיח ולהציע אתגרים חדשים. לדוגמא, חברת Microsoft רשמה לאחרונה פטנט המאפשר לה לבדוק את היכולת של עובדים לעמוד ביעדים בהתאם לקצב ליבם, זיעה, ועוד. בעצם, את הפגיעה בפרטיות במקום העבודה אפשר לחלק לפי שלוש המתודות של אבטחת המידע: "מה שאתה יודע, מה שיש לך ומה שאתה". כל אחת מהפגיעות בפרטיות עשויה לבוא בצורה אחרת, אך כולן מגדירות את הזכות לפרטיות טוב יותר מההגדרות המקובלות.

את המתודלוגיה של הזכויות לפרטיות במקום העבודה אפשר לחלק כך: מה שהעובד יודע; החל מהמרשם הפלילי של התובע, אופיו המיני, העדפתו הדתית ומידע אישי עליו, דרך ססמאותיו לחשבונותיו הפרטיים,

סודותיו המסחריים וסודותיהם המסחריים של מעבידים קודמים. מה שיש לעובד: ניטור אמצעי התקשורת של העובד, החל מהטלפון הסלולרי שלו ותכתובות הדואר האלקטרוני שלו, בחינת תעבורת האינטרנט ופלט השיחות שלו, ביצוע איכון GPS וסלולרי על מיקומו וגם בחינת אחסנה פיסית שלו: קבצי המחשב במקום העבודה, רשומות ומגירות אישיות. לבסוף, מה שהעובד הוא: ביומטריה, על שלל שלביה: טביעות אצבע, רשתית, קרנית, הזעה ודופק; תחת ההגדרה הזו תכנס גם הביומטריה הקיימת בפוליגרף, כלומר: האם העובד משקר על מה הוא יודע ומה יש לו, וגם מעקב מתמד אחר העובד באמצעות מצלמות והאזנה.

ראשית, נתייחס לשאלת **מה שאתה יודע**: חוק שוויון הזדמנויות בעבודה אוסר מהמעסיק לבקש מעובדים מידע אודות עברם הצבאי, אבל גם חוק המרשם הפלילי ותקנות השבים שקובע בסעיף 19(ב) כי "מי שיש עליו מידע אינו חייב לגלות הרשעה שהתיישנה למי שאינו רשאי להביאה בחשבון; שאלה בדבר עברו הפלילי תיחשב כאילו אינה מתייחסת להרשעה שהתיישנה והנשאל לא ישא באחריות פלילית או אחרת מחמת שלא גילה אותה למי שאינו רשאי עוד להביאה בחשבון". מעבר לכך, החוק מטיל איסור פלילי (סעיף 22(א)) על מי שמבקש מידע שאין לו סמכות על פי דין לקבל מהמרשם: כלומר, שאלה האם לעובד יש עבר פלילי.

אולם, מה שהעובד יודע אינו רק המרשם הפלילי והשירות הצבאי, אלא גם אמונותיו, דתו ועוד; במקרה אחד, חייב כונס נכסים נתבעים בתביעה של סוד מסחרי למסור את סמאת ההצפנה (בש"א 9455/09 תום קפלן נ' קבוצת PCIC, וכך גם היה במקרה אחר, תפ (ת"א) 40071/04 מדינת ישראל נ' חברת בוריס פקר הנדסה בע"מ, בו לא ברור כיצד קיבלו החוקרים גישה למחשב הנאשמים) אולם, במקרים אחרים בבתי משפט זרים, נפסק כי אין חובה להעביר את סמאות המחשב של אדם (Boucher v. State, 2007 WL 4246473 (United States District Court for the District of Vermont).

סביר להניח שבהתחשב בהלכת **בנימין אליהו** והמגמות הגוברות בתחום הפרטיות, יש להתייחס ביתר משקל להלכות זרות ולהלכות אלו על חשבון הלכות אחרות כמו עב (ת"א) 5524/03 אנט קירש נ' זכוכית בידודית 18 ז.ב. בע"מ, בה קבע בית המשפט כי אין נטילת הסמא מהעובד מנוגדת להוראות הדין ובניגוד לעב (ת"א 7615/02 דוד בנימין נ' קוריג'ין בע"מ בה בית המשפט אישר את ניתוקו של עובד מהדואר האלקטרוני הארגוני והחלפת הסמא. כלומר, השאלה האם למעסיק מותר לבקש את סמתך או מידע הנמצא בראש שלך כפופה בין היתר לדין הכללי, אך דיני העבודה נוטים להגן על העובד מלחשוף מידע פרטי.

לעומת ההגנה על מה שאתה יודע, ההגנה על **מה שיש לך** היתה פחותה בבתי המשפט. אכן, בהלכת **בנימין אליהו** עיגן בית המשפט המחוזי את הזכות לפרטיות בדואר אלקטרוני של עובד (באותו המקרה, דובר על פסילה חוקתית של ראיות שהושגו בפגיעה בפרטיות לפי סעיף 32 לחוק הגנת

הפרטיות), אבל גם במקרים אחרים עוגנה הזכות בצורה יחסית בהתאם לאינטרס של העובד, אך נקבע כי לעובד עצמו אין כח מיקוח, ולכן כל ויתור על פרטיות חייב להעשות במפורש (עניין איסקוב, עע 06/010121 טלי איסקוב נ' הממונה על חוק עבודת נשים ועב' 1158/06 רני פישר נ' אפיקי מים) במקרה אחר בית המשפט המחוזי החליט, בהליך של המרצת פתיחה כי "**מתחם הפרטיות אינו נקבע עפ"י דיני הקניין והזכות לפרטיות היא זכותו של האדם ולא זכותו של המקום**, וכלל זה חל גם כאשר תוכן הודעת הדוא"ל הוא בעניין עסקי. גם העובד במישור של יחסי עובד-מעביד זכאי למתחם פרטיות במקום העבודה כאשר הוא משתמש בדוא"ל בשרת החברה, וכל שכן בעל מניות בחברה במישור מערכת היחסים בין בעלי המניות בחברה" (ה"פ 1529/09 חן בת שבע ואח' נ' יוני בן זאב).

אולם, קריאתו של דואר אלקטרוני היא רק פן אחד של הפגיעה ב**מה שיש לך**, כאשר לעובד יש מחשב, אשר הוא גם רכוש של המעביד, או כל ציוד אלקטרוני אחר, ניתן להשתמש במידע עליו על פי התורה הקניינית, כפי שההלכה בעניין **טלי איסקוב** אימצה. אולם, בעניין בשא (ת"א) 5298/09 עודד רובין נ' חברת תשתיות נפט ואנרגיה איפשר בית המשפט להגיש דוחות איכון של איתוראן אך לא להציג את הדוחות שנאגרו מחוץ לשעות העבודה, כאיזון בין הזכות לפרטיות לזכות המעביד לקניין.

עד שהגיעה הלכת איסקוב, בתי הדין לעבודה ניסו לאזן בין זכותו של המעביד לזכות לפרטיות; כך, במקרה אחד פסק בית הדין כי: "*לא מצאנו ממש בטענות המבקשת לפיהן השימוש שעשתה המשיבה בדו"חות איתוראן מהווה פגיעה בפרטיות. מדובר בעובדי המשיבה, הנוהגים לצורך עבודתם בכלי רכב השייכים למשיבה. בכלי הרכב מותקנים מכשירי איתוראן. הדו"חות של איתוראן המלמדים היכן היו העובדים, מתייחסים לשעות העבודה של העובדים ולא מעבר לכך. איתור מקום הימצאם של העובדים בזמן העבודה עת מדובר בעבודה שמטיבה הינה ניידת, אינו פוגע בפרטיות, והוא בא לסייע בידי המעבידים לפקח על העובדים. זאת במיוחד כאשר העובד מוסר מידע בדו"חות הנוכחות היכן הוא נמצא והבדיקה של המעביד נעשית בדיעבד*" (סק (ב"ש) 1026/06 הסתדרות העובדים הכללית החדשה נ' תש"ן תשתיות נפט ואנרגיה בע"מ).

מנגד, כאשר לא היה אינטרס חזק למעביד, היו מקרים שקבעו כי אין אפשרות לקבל קבצים, תוכנות או מידע הנמצאים בחזקת העובד, אם אינם מהווים סוד מסחרי (עב (י-ם) 1808/03 אופטיק דורון בע"מ נ' זכאי מיכל, עב (ת"א) 1733/01 דיימקס מערכות 1988 בע"מ נ' נטקוד בע"מ). כלומר, דרישת המידתיות לצורך הפגיעה בפרטיות התקיימה בצורה רפויה יותר כאשר מדובר ב**מה שיש לך** לעומת **מה שאתה יודע**, כאשר הרציונאל לכך היה הרציונאל הקנייני, שניתן להבין ממנו שמידע גבוה יותר (פלט שיחות ואיכונים) נחשבים פחות פוגעים בפרטיות מאשר תוכן דואר אלקטרוני (והשוו פ 40206/05 מדינת ישראל

נגד פילוסוף ואחרים). התפישה של הבנת החפץ (חומר המחשב) כמקרה פרטי של שיחה, כזה שמבין שהקריאה של הדואר האלקטרוני, כמו האזנה לקו הטלפון (אשר במקרה הזה יש חוק ספציפי נגד, חוק האזנת סתר), היא בעלת פוטנציאל פוגעני יותר מאשר איכון או פלט שיחות היא תפישה ארכאית שאינה מתחשבת בכך שבעידן הדיגיטלי זהות הנמען או מיקומו של עובד מהווים פגיעה בפרטיות לא פחות מאשר תוכן השיחה (ראה, לעניין זה את עניין Maynard). לעניין זה, אגב, בתי המשפט פסקו כי התקנת מצלמות אשר נועדו לפקח על העובדים היא כהרעה בתנאי העבודה ופגיעה בפרטיותם (דמ 2734/00 אייזנר אוסנת נ' ריצ'מונד מפעלי סריגה בע"מ), אך עניין המצלמות עוד רחוק מלהתחיל כדיון נפרד.

משהגיע פסק הדין בעניין איסקוב, בית הדין נקט בגישה אחרת, ופסק כך: הדואר האלקטרוני יתחלק לשלושה סוגים:

- דואר "מקצועי" (לדוגמא: office@example.com)
- דואר אישי (jonathan@example.com)
- דואר פרטי (jonklinger@gmail.com)

פסק הדין קובע כי המעביד יכול לנטר את הדואר המקצועי בכפוף לידיעה מוקדמת של העובדים ולאבחנות נוספות; את הדואר האישי המעביד לא יכול לקרוא אלא בהסכמה ספציפית של העובד לכל קריאה, וגם זו חייבת להעשות רק על מנת להגן על אינטרס ראוי של המעביד, תוך שהוא מקבל את ההסכמה בצורה מודעת ומושכלת; **את הדואר הפרטי המעסיק יוכל לקרוא רק בכפוף לצו** אנטון פילר **שינתן על ידי בית המשפט במקרים חריגים במיוחד**.

אלא, שהנקודה החשובה ביותר כאן היא לאו דווקא הדואר האלקטרוני, אלא גישת בית המשפט שמאמצת את המודל המניעתי: **בית המשפט קובע לא אחת כי מעביד רשאי לחסום את הגישה של עובדיו לרשתות חברתיות וכי הוא רשאי לקבוע מדיניות שימוש במחשב, עד כדי העדפה לעיתים של המתחם הקנייני של המעביד על זכויות עובדים**:

"על המעסיק לבחון אמצעים חלופיים למעקב ובהם טכנולוגיות שמטרתן לחסום שימוש בלתי ראוי (blocking) שעושים העובדים במחשב. זאת, תוך שיעדיף שימוש בטכנולוגיות מעקב שיש בהן כדי להוות תחליף ראוי לחדירה לנתוני תוכן של תכתובת אישית ולקריאת תוכן הדואר האלקטרוני".

כלומר, על המעביד לחסום גישה למקומות בלתי רצויים, ולא להתערב בתעבורה. הדבר מעלה שאלות רבות כאשר מדובר בהתקנת Firewall או אנטי-וירוס ארגוני. האם אותו אנטי-וירוס "קורא" את הדואר של העובד או לאו? האם ה-Firewall שחוסם גישה לאתרים מסוימים צריך גם לדווח על נסיונות הגישה והאם

האנטי וירוס צריך להעביר למנהל הרשת העתק של הקבצים הנגועים, גם אם הדבר פוגע בפרטיות העובדים?

אולם, הנושא המעניין יותר הוא הפרטיות של **מי שאתה**. הפגיעה בפרטיות של מי שאתה היתה באה לידי ביטוי לאחרונה בפרשיה שהייתי מעורב בה, בה עובדת של צומת ספרים סרבה לתת טביעת אצבע עבור שעון הנוכחות ופוטרה מעבודתה (בקשה דחופה לבית המשפט להחזרה לעבודה: עב 17342-07-10 דק נ' צומת ספרים, החלטת בית המשפט במעמד צד אחד: עב 17342-07-10 דק נ' צומת ספרים). לאחר התערבות של בית המשפט הוחזרה העובדת למשרתה, אולם השאלה עוד נמצאת בחיתולים וראוי לשאול מהי המידה בה מעסיק יכול לשמור מידע על מיהו העובד שלו. מדובר על מעקב מסוגים שונים של ביומטריה. קודם כל, השאלה היא מהי היכולת של מעביד להעביר את עובדיו בחינת פוליגרף.

במקרה אחד, תא (אשד') 2016/03 פרחצ'קוב יוליה נ' חב' מיל סטון עיבודי שיש בע"מ, עובדים שפוטרו לאחר כשלונם בבדיקת פוליגרף תבעו את מעבידם בגין פגיעה בפרטיות; אלא שבמקרה הנ"ל **התביעה היתה בעקבות פרסום תוצאות הפוליגרף, ולא בטענה כי הפוליגרף עצמו פוגע בפרטיות**. לכן, בית המשפט נמנע מלדון בשאלה האם איומים בפיטורין לאחר אי הסכמת עובד לבדיקת פוליגרף היא פגיעה בפרטיותו בהקשר של דיני העבודה, כך היה גם במקרה אחר, בו בית הדין הארצי לעבודה הותיר את השאלה האם בדיקת פוליגרף לעובד היא חוקתית בצריך עיון (דב"ע 4-07/97 אוניברסיטת תל-אביב - ההסתדרות הכללית החדשה, האגף לאיגוד מקצועי ואח'). לעומת זאת, **אהרן ברק** רואה (פרשנות במשפט, כרך שלישי, פרשנות חוקתית (ירושלים, תשנ"ד), עמ' 431-433) את הצורך בבדיקת פוליגרף כפוגעת בכבוד האדם ולא כפגיעה בפרטיות: "**היפנוזה בחקירה פוגעת בכבוד האדם. שימוש בפוליגרף פוגע בכבוד האדם. 'שוק חשמלי' פוגע בכבוד האדם. כפייה לתת 'טביעות אצבעות' פוגעת בכבוד האדם**".

נראה כי השאלה האם אי הסכמת העובד להבדק בפוליגרף, עם מלוא ההסתייגויות החוקתיות שיש מאמינות המכשיר צריכה להבחן לפי מבחנים חוקתיים של מידתיות וסבירות; בעניין אוניברסיטת תל-אביב נאמר כי "*לגבי כל מבחן ומבחן יש לבדוק באיזו מידה המבחן אכן משקף נכונה את הנתונים שבדיקתם נתבקשה. יש לבדוק את דייקנות כל מבחן על-פי תוקפו (VALIDITY) ומהימנותו (RELIABILITY)*". אולם לא די בכך; הפוליגרף הוא רק ביומטריה מסוג אחד של "**מה שאתה**", אלא "מה שאתה" במובן הרחב יותר. למרות התפיסה הרווחת כי ביומטריה היא טכנולוגיה מאובטחת, הרי שבעקבות הצורה הרעועה שביומטריה זולגת (אנו משאירים טביעות אצבע בכל מקום, וניתן לצלם אותנו מרחוק), ביומטריה היא אחת הצורות הבטוחות פחות לאבטחה של מידע אצל המעביד (וזה עוד לפני ששאלת המאגרים הביומטריים נכנסה לפועל).

לסיכום

ניתן לחלק את המידע הפרטי שנאגר על העובד בצורה של מידע שהעובד יודע, עליו הדין מגן יותר באמצעות איסורים על אפליה או שאילת שאלות מסוימות, מידע שהעובד מחזיק, בו בתי המשפט דורשים הסכמה מפורשת על מנת לגשת אליו, וזאת בכל פעם ופעם, שבהם בתי המשפט עדיין לא הצליחו להחליט מהי הגישה המועדפת.

דברי סיום

בזאת אנחנו סוגרים את הגליון ה-31 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 מאי.

אפיק קסטיאל,

ניר אדר,

30.04.2012