digital whisper
Technologic Papers

Supply Chain & Open Source Attacks

מאת נועם רשקובסקי

הקדמה

באוגוסט האחרון שמעתי ש-Checkmarx (חברת אבטחת מידע ישראלית) עושים מפגש בנושא Open Source והמשמעויות של השימוש בו, ואחריו מציגים הקרנת בכורה לסרט החדש של Thor. כמובן שמאוד התעניינתי בהקרנה - אה סליחה, בהרצאות, ובאתי. אני חייב להגיד שבהתחלה חשבתי שאין הרבה תוכן בעולם הזה, מסתבר שדווקא יש הרבה.

במאמר זה, אסביר אילו מתקפות יש לקוד פתוח (Open Source), אתן דוגמאות למתקפות, איך נוכל להגן מפניהן ואציג כלי שכתבתי שיכול לסייע בהגנה.

Open Source

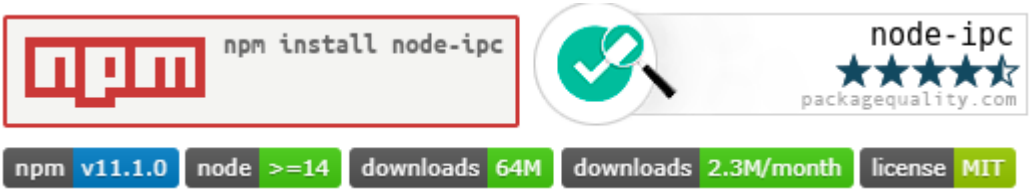
ראשית חשוב להבין בכלל מה זה קוד פתוח. קוד פתוח הוא קוד שנגיש וחופשי לשימוש של הכלל. כלומר, שאני כמפתח רוצה לתכנת משהו, לדוגמה תוכנת מחשבון, אני לא ארצה עכשיו לממש את הקוד לחזקה או ללוגריתם. כמובן שאני גם לא המתכנת הכי טוב שאני מכיר, ויש הרבה אנשים שיותר טובים ממני והאנשים האלה יכתבו את הקוד בצורה יעילה וטובה יותר ממני. מכאן נובעת שאלה איך אני לוקח את המוח של אנשים אחרים ונותן להם לכתוב את הקוד בשבילי? כאן נכנס הקונספט של קוד פתוח. אנשים חכמים מאוד באינטרנט החליטו להפיץ את הקוד שלהם לאנשים שרוצים להשתמש בו.

זאת אומרת, שאם אני רוצה להשתמש בפונקציה של חזקה, אני יכול להשתמש בקוד של מישהו (מ-github או אתרים בסגנון) ולהעתיק אותו לקוד שלי, אפילו להשתמש בספרייה שהם כתבו, או לקחת את המימוש שלהם למחשבון ולהשתמש בו לטובתי.

המתקפות

אז הכול נשמע טוב - אני יכול לקחת קוד של אנשים אחרים, הם יכולים לקחת ממני קוד, הזמן שלי יכול להיות מנוצל על דברים אחרים... אז מה הבעיה בעצם?

הבעיה הראשונה היא שאני לא מכיר את המפתח השני ולא יודע אם הוא אדם ישר שלא מנצל את עמדתו. זאת אומרת, שהקוד שלו עושה את מה שהוא אמור לעשות, והוא לא מוסיף גם קוד זדוני, לדוגמא כמו שקרה לא מזמן עם המתכנת של "node-ipc", ספריה שמטרתה לאפשר תקשורת בין תהליכים ב-java (את הספריה הזאת הורידו 64 מיליון פעמים).



[https://www.npmjs.com/package/node-ipc]

כותב הספריה רצה להביע את עמדתו לגבי המלחמה בין אוקראינה ורוסיה, והוסיף לקוד שלו פונקציה שכאשר הוא מזהה ריצה על מחשב אשר נמצא ברוסיה, הוא מוחק את כל הקבצים שנמצאים עליו.

בכללי הקונספט שעשע אותי, ואז הבנתי שהוא יכל לעשות עוד דברים ולהגיע מאוד רחוק. ככל שהספריה יותר מוכרת, היא תגיע ליותר אנשים, והסכנה למתקפה גדלה עוד יותר. לדוגמה, חברות גדולות בעלות תוכנות נרחבות (פייסבוק, אינסטגרם...) יכולות להשתמש בספריה הזו בתמימות כי היא מוכרת, ופוטנציאלית להפיץ למשתמשים שלהן קוד זדוני.

בעיה נוספת שיכולה להיות היא Typosquatting. כלומר, ספרייה שמתחזה עם שם דומה לספרייה מוכרת אחרת. לדוגמה, אני שומע על ספרייה ממש טובה בשביל חישובים מתמטיים בפייתון, לצורך הדוגמה שמעתי שקוראים לה "numpy", וכאשר הסתכלתי על הספרייה באתר הלגיטימי של pypi ראיתי שיש גם "numpy" וגם "numpy2". כשהסתכלתי על הדוקומנטציה של שתיהן, ראיתי של-numpy2 יש שידרוגים של יעילות קוד, ועוד מערכות מתמטיקאיות שאין ל-numpy.

אז כמובן שאני אוריד את numpy2, אשתדרג, למה לא? אני אשתמש בספרייה ואראה שהכול טוב והיא עושה את מה שהבטיחה. רק שלצערי, אחרי כ-4 חודשים, אני אגלה שכל מי שהשתמש ב-numpy2 מתלונן כי הסיסמאות שלו ל-github (שנשמרו בצורה לא מוצפנת על מחשב של משתמש ב-git cli) הודלפו לרשת וכמובן שרק אחרי כ-3 חודשים יישמו פתרון והספרייה תמחק מהאתר. בעיה זו של סיכול בשם הספרייה יכולה לקרות גם כשאני מוריד ספרייה באמצעות פקודה, ומוסיף בטעות עוד אות או מספר. אני יכול להוריד את הדבר הלא נכון, הרי בשביל להוריד משהו אני רק צריך לכתוב "pip install numpy" אז אני יכול לכתוב בטעות: "pip install numpy2".

בעיה אחרת שיכולה להיות היא שהאנשים שעובדים על הפרוייקט\ספרייה יכולים בכול רגע לעזוב או לזנוח אותו, בגלל שלרוב היוצרים אכפת מהפרוייקט שלהם הם יכולים להעביר אותו למישהו אחר שעליו אולי ניתן לסמוך פחות מהיוצר המקורי. אפילו יותר גרוע, בשנת 2018 להרבה מהיוצרים היו סיסמאות מאוד חלשות ואף זהות לשם המשתמש שלהם, שלרוב לא שמו Two Factor Authentication, מה שהוביל לפריצה לחשבונות ופרוייקטים של משתמשים, שיכל להוביל גם לשינוי הפרוייקט ללא ידיעתו של היוצר.

עוד בעיה שיכולה להתרחש, הוא שימוש בספרייה אשר הספריות הנוספות עליה היא מסתמכת זדוניות. נגיד ואתם (הקוראים) האנשים הכי חשדנים, לכן תמיד כשאתם משתמשים בספרייה אתם סורקים את הקוד שלה לפני השימוש ואתם לא רואים משהו יותר מידי חשוד בקוד של המתכנת. אך אתם מגלים מאוחר יותר שהייתה בעיה בספריה הזו, שהתבטאה בשימוש בספרייה נתמכת שבה היה פוגען.

מכאן יכולים לקרות שני דברים: או שהמפתח של אותה ספרייה הכניס את אותה הספרייה הזדונית בכוונה, או שהוא באמת עשה טעות והשתמש בספרייה עם פוגען והפיץ אותו מבלי לדעת. בכל מקרה, קשה לעקוב אחרי כל ייבוא של ספרייה נוספת מתוך ספרייה אחת ראשית, מה שמוביל לנושא גדול יותר.

Supply Chain

שרשרת אספקה הוא קונספט שאומר שלכל פרוייקט\ארגון באשר הוא יש איזושהי שרשרת של ארגונים, משאבים, טכנולוגיות וחברות אחרות שמספקות לו שירותים בשביל שהוא יוכל לעבוד בצורה מיטבית ויעילה. למשל, צבא, התוצר שצבא צריך להביא למדינה שלו הוא בעיקר ביטחון, לשם כך הוא צריך כוח אדם, מזון, חיילים שיודעים להכשיר אחרים (בעלי ידע), תקשורת עם מפקדיו וחמ"ל (חדר מלחמה), רפואה, מודיעין, נשקים וכדומה. מזון ומודיעין, למשל, לאו דווקא מגיע מהצבא עצמו, אלא מסופקים מגורמים חיצוניים.

פגיעה בשרשרת אספקה מתבטאת בפגיעה באחד המשאבים של ארגון, בדגש על משאבים מחברות חיצוניות. נמשיך עם הדוגמה של הצבא, בהנחה והצבא שלנו מותקף ואנחנו צריכים להתגונן מאויב שמנסה להיכנס עמוק יותר למדינה שלנו, אנחנו יכולים לנסות לחשוב מה האויב הזה צריך כאשר הוא רוצה לצאת למבצע שכזה. אנחנו יכולים להניח שגם לאויב יש שרשרת אספקה דומה לשלנו.

על מנת לנטרל אותו או להאט אותו, נוכל לפגוע או להכשיל מספר גורמים שהאויב משתמש בהם. כגון, ווידוא שהמודיעין של אותו צבא יהיה שגוי, דבר שיגרום לבילבול וקושי בעת הלחימה. לחלופין נוכל לפגוע בתשתיות, בין אם זה לפגוע לאויב בתקשורת הפנימית עם הפיקוד, או לפגוע במעבר של מזון לפלוגה העויינת. דברים אלו יכולים לשבש פעולות פנימיות ממחוץ לשורות האויב.

המתקפות

כפי שצבא נתקף יכול לחשוב על דרכים לפגיעה באויב מבחוץ ולאו דווקא ישירות בלוחמים, גם תוקפים ברשת יכולים לפגוע בסביבה של חברה בצורה לא ישירה. אם אתם משתמשים באפליקציה כלשהי לפיתוח, למשל כמו Visual Studio (תוכנה מאת מייקרוסופט לעזרה לכתיבת קוד בכל מיני שפות תכנות, אפליקציה זו מוכרת ומשתמשים בה ביום יום כותבי קוד רבים). יום אחד יוצא עדכון לאפליקציה, אתם משדרגים ואתם לא חווים שום דבר חדש או מיוחד, חוץ משימוש גבוה בזיכרון המחשב.

התנהגות זו יכולה לקרות גם בצורה רגילה באפליקציה, לכן אינכם חושדים בדבר. אך זמן לא רב לאחר השדרוג, החברה משחררת הודעה שהיא מתנצלת על הימצאות Cryptominer (פוגען שמטרתו להשתמש במשאבי המחשב בשביל לכרות מטבעות דיגיטליים) שנכנס בלא יודעין בעדכון. מתקפה כזו קרתה גם באמת, ושמה Sunburst. היא התרחשה בחברת SolarWinds, באפליקציה Orion Network (מטרת האפליקציה היא למצוא, לאבחן ולטפל בבעיות תקשורת ברשת כלשהיא). בשנת 2020 חברה זו נפרצה והושתלה דלת אחורית אשר מאפשרת לתוקפים שיצרו אותה הרצת קוד ברשת. מאות חברות הושפעו מהתקיפה, בינהן אפילו המחלקה לביטחון המולדת של ארה"ב.

אתם בתור מתכנתים שמשתמשים בספרייה מוכרת, שאפילו חברות כמו מייקרוסופט, אמזון, ו-VMware משתמשות בה וסומכים עליה ועל התוכן שבה. עדיין, גם בספריות מוכרות ונפוצות, יש סיכוי להמצאות קוד זדוני ודווקא הנרחבות של הספרייה יכולה להוות בעיה. מקרה כזה יכל להתרחש עם ספריית log4j שהיא תוכנה שעוזרת לרשום אירועים שקורים באפליקציה שלכם בצורה מאוד נוחה, קריאה ויפה.

בשנת 2021 התגלתה חולשת הרצת קוד בספרייה הזו, לחולשה קוראים log4shell, שהייתה משומשת בהמון אפליקציות ובהמון מקומות בתוך אפליקציות ענק (כמו AWS או ESXi). אך למה החולשה הזו נחשבת למתקפת שרשרת אספקה? למרות שהבעיה היא פגיעות בקוד הפתוח של הספרייה, היא נכנסה לאפליקציות גדולות יותר והיוותה משאב בשרשרת השירותים שמסופקים לחברות ענק. כלומר, בגלל הפגיעה בשירות שמסופק לאפליקציה כלשהי, האפליקציה עצמה נפגעה וכתוצאה מכך המשתמשים.

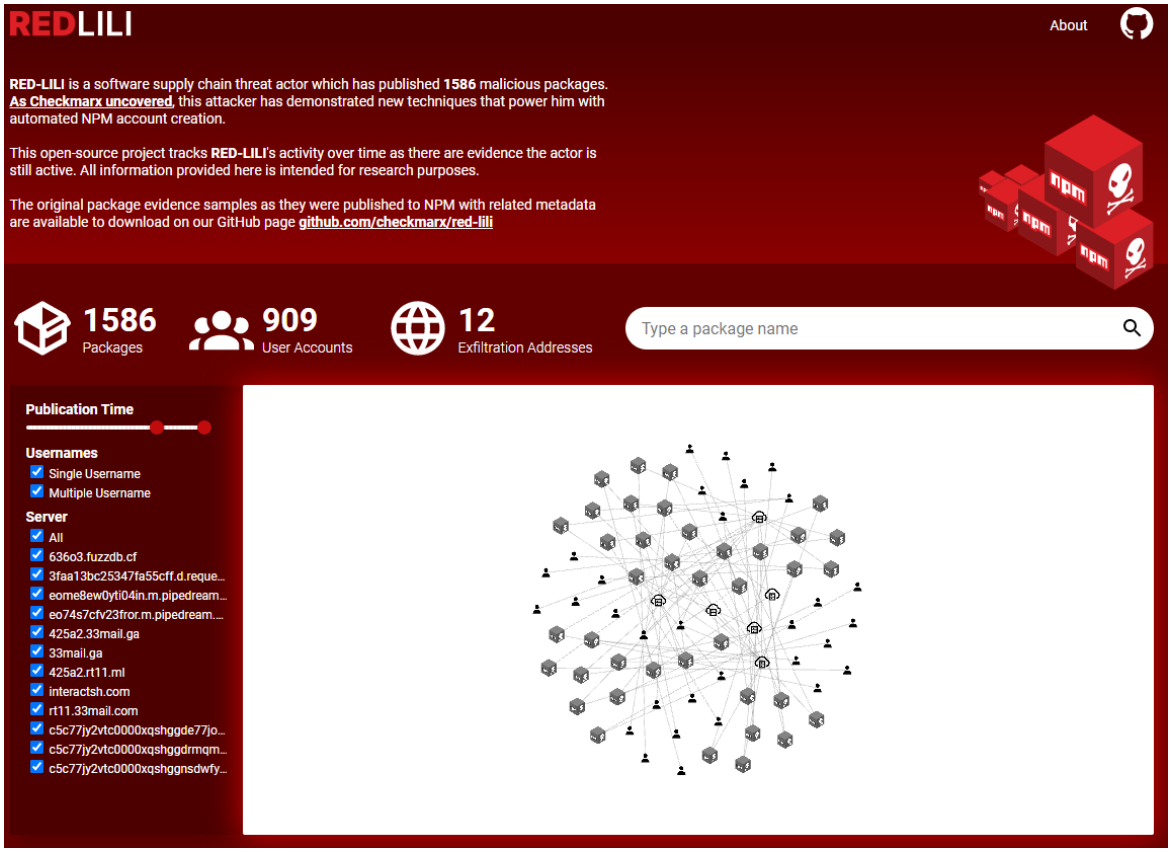
חשוב לציין שבמתקפות של שרשרת אספקה לרוב המשתמש הפשוט לא מודע למה שקורה מאחורי האפליקציות לכן קשה לו לדעת שבכלל מתרחשת מתקפה זדונית עליה ועל כן גם עליו.

הגנות

אחרי שראינו את כול הדוגמאות והבנו מה הסכנות בקוד פתוח ובפגיעת בשרשרת אספקה איך אנחנו מגנים מדבר שכזה?

כמובן שלבדוק עכשיו כל אחד מהקודים של כל אחת מהספריות ולעדכן את הספריות באופן גורף, זו משימה כבדה. לכן יש מערכות כמו Dependabot או Greenkeeper (היום נמצאות ב-Snyk) שמסתכלות על הספריות שאנחנו משתמשים בהן, רואות אם יש עדכון חדש ומעדכנות באופן אוטומטי, בשאיפה תמיד להיות ראשון כשיוצא עדכון לבעיה בקוד (יש עוד יכולות חוץ מהיכולת הזו). מעולה, עכשיו האפליקציה שלנו תהיה מוגנת מכול החולשות שהתגלו בספריה כלשהי. חבל שזה רחוק מהמציאות, גם אם נעדכן כל הזמן את הספריות ו"ננקה" את הקוד מחולשות עבר פוטנציאליות, עדיין לא נוכל להיות מוגנים מהמקרה שמישהו מקבל שליטה לעידכוני ספריה ומעדכן אותה עם הקוד הזדוני.

מה אם נעדכן כל שלושה או אפילו חמישה עדכונים, במטרה לקבל את העדכונים כשהם "איכותיים" יותר, שנבדקו כבר על אנשים אחרים לאורך תקופה כלשהי? גם על זה תוקפים חשבו, ושחררו 5 עדכונים ריקים וזהים על מנת לעקוף את השיטה הזו. קבוצת התקיפה שיישמה את העיקרון הזה נקראת RED-LILI והתגלתה על ידי חברת Checkmarx. כיום, החברה הרימה אתר https://red-lili.info אשר מנסה לעקוב אחרי כל הספריות שמופצות ידי קבוצת תקיפה זו.



[https://red-lili.info]

נשמע שהכל אבוד ואין מה לעשות. אנחנו לא נעבור על כל קוד של כל ספרייה ונראה אם יש שם משהו. זה יגדיל משמעותית את זמן העבודה אז למזלנו דווקא יש מה לעשות. אפשר לסמוך על חברות כמו Checkmarx (not sponsored) ו-Snyk שיעזרו לנו. לשתיהן יש הרבה אפליקציות, כגון Checkmarx SCA, שמטרתה להביא למשתמשים שלהם ספריות וקוד שעבר אנליזות ואבחונים על ידי המערכת שלהם, שבעצם הוא "נקי" ואין בו שום קוד זדוני.

יש גם את Synk SCA שמטרתו דומה, אך אצלם אתם משלמים על פונקציונליות נוספות. יש אופציה חינמית לאפליקציה זו, אך אפשר לעשות רק כ-300 בדיקות לחודש, כאשר האופציה הכי "זולה" זה 500 דולר ל-5 מפתחים לחודש (ל-Checkmarx לעוד מידע תוכלו ליצור קשר עם החברה).

יש לנו גם אפשרות אחרת, שהיא לקחת את הבעיה לידיים שלנו ולקחת שליטה מלאה. זה גם הפתרון הזול לאנשים פרטיים שאין להם כסף לאפליקציות כאלה. פה נכנסים לתמונה הפתרונות החינמיים, כמו Hagana (שנכתב על ידי יעקב [לא מצאתי את השם משפחה])‏, מודול שמטרתו להגן עליכם באמצעות סט חוקים שאתם כותבים בעצמכם לאפליקציה שלכם, שמגדירים בדיוק מה האפליקציה יכולה לעשות על פי הצרכים הספציפיים שלה (בין אם זו גישה תקשורתית, גישה למערכת הקבצים או הרצת קוד). הספרייה שלו תזריק את עצמה לפונקציות הרלוונטיות, בכדי לתפוס את הקריאות לפונקציה ולראות האם הן לגיטימיות או לא. לצערי הרב הספרייה עובדת רק לקוד של nodejs, ורק על חלק מהפונקציות.

לכן החלטתי לכתוב כלי שיעבוד לכל שפת תכנות אשר נכתבה במערכת הפעלה Windows. קוראים לו GuardCode. בדומה ל-Hagana, GuardCode פועל בתצורת "חוקים". ההבדל הוא שבמקום לכתוב את החוקים בתוך הקוד עצמו, כותבים אותם בקובץ נפרד, בתצורה של Regex (שיטה להכיל כמה אפשרויות בשורה אחת או לא להכילן). יש אפשרות לכתוב את החוקים בצורת Whitelist או Blacklist (כאשר Whitelist זה מה שאפשרי לעשות על המערכת, וכל מה שלא מוכל ברשימה יקפוץ, ו-Blacklist זה שהכול אפשרי חוץ ממה שיש ברשימה).

בנוסף על כך, אפשרי "לנטר" על כל פעולה שניתן לבצע על קובץ במערכת הפעלה של Windows (בעתיד יהיה ניטור על מפתחות מערכת של Windows Registry), ועל תקשורת שנפתחת). GuardCode מסתמך על מוצר של Microsoft, ועל SDK של OpenProcmon מ-Github. המוצר שהתבססתי עליו ביצירה של GuardCode הוא Procmon. המטרה של Procmon היא להראות למשתמש כל דבר שקורה במערכת של Windows. כלומר, כאשר תהליך עושה משהו לקובץ, או פותח תקשורת,Procmon יתפוס זאת וכנ"ל לגבי GuardCode. בסופו של דבר GuardCode הוא Antivirus לתהליך ספציפי, עם חוקים שאתם כותבים ובשליטה מלאה שלכם. אם אתם רוצים עוד שליטה על הקוד שלכם אתם יכולים לכתוב בעצמכם את ה"driver" של Procmon. השימוש בשני הכלים מסייע לכם לבדוק ואף לחסום אנומליות בספרייה\מודול שעלולות להיות זדוניות(בהתאם למה שאתם מגדירים). כגון, גישה לקבצי מערכת SAM ו-SYSTEM, שני הקבצים הללו יכולים לספק לתוקף את הסיסמאות לכל המשתמשים במחשב שלכם.

סיכום

בסופו של דבר, פיתוח הוא מרכיב מרכזי בכל ארגון כיום. אפילו הקיוסק השכונתי צריך פיתוח, בין אם זה למטרות פרסום ובין אם זה לשמירת היסטוריית עסקאות במסד נתונים. המתקפות של קוד פתוח רלוונטיות לעבודה של כל מפתח, מאחר וכמעט אין תוכנה שלא משתמשת בספרייה שנכתבה ע"י אדם חיצוני שלא קשור לפיתוח. כמובן שאני לא אומר שנפסיק להשתמש בספריות, כי בכל זאת, הן חלק גדול מהפיתוח ותורמות לחיסכון בזמן, אבל חשוב לפקוח את העיניים.

חשוב לראות שלספרייה שמשתמשים בה יש תדמית טובה, לרוב אם הן בעלות הרבה הורדות וחשוב גם להתעדכן מידי פעם על הספרייה, בין אם זה לראות עדכונים ובין אם התראות של אנשי אבטחה לגבי חולשות (כאלה שהוטמעו בכוונה, וכאלה שנמצאו בטעות). אחרי הכל, כשמשתמשים בספרייה זדונית, ניתן בטעות לפגוע בשרשרת אספקה של חברה אחרת מבלי לדעת.

לפי דעתי, זו האחריות גם של המפתח וגם של איש האבטחה וההגנה לשמור על המוצר שמוציאים ועל תהליך הפיתוח. לכן, איש האבטחה צריך לשים את המגננות המתאימות ולתחזקן וגם המפתח צריך לשים עין על תוספות לא מוכרות בקוד, אפילו להגן על עצמו ולמצוא דרכים לבדוק את אבטחת הקוד שלו, בין אם זה באמצעות פתרונות חינמיים או פתרונות בתשלום.

קישורים

- https://github.com/noam242/GuardCode
- https://github.com/yaakov123/hagana/
- https://red-lili.info/
- https://he.wikipedia.org/wiki/Log4Shell
- https://www.checkpoint.com/cyber-hub/threat-prevention/what-is-a-supply-chain-attack
- https://www.techtarget.com/whatis/feature/SolarWinds-hack-explained-Everything-you-need-to-know
- https://lirantal.medium.com/malicious-modules-what-you-need-to-know-when-installing-npm-packages-12b2f56d3685
- https://www.mandiant.com/solarwinds-break-resource-center
- https://snyk.io/blog/typosquatting-attacks/
- https://www.npmjs.com/package/node-ipc