עמוד פתיחה

# תוכן עניינים

[תוכן עניינים 1](#_Toc55577062)

[תיאור הבעיה 2](#_Toc55577063)

[הפתרון 3](#_Toc55577064)

[מרכיבי התוכנה 4](#_Toc55577065)

[אפליקציית Updater 4](#_Toc55577066)

[אפליקציית Launcher 4](#_Toc55577067)

[אפליקציית Installer 5](#_Toc55577068)

[מימוש הפתרון 6](#_Toc55577069)

[אפליקציית Updater 6](#_Toc55577070)

[מודול rsa\_signing 6](#_Toc55577071)

[מודול settings 6](#_Toc55577072)

[מודול registry 6](#_Toc55577073)

[מודול messages 6](#_Toc55577074)

[מודול updater 6](#_Toc55577075)

[מודול service 6](#_Toc55577076)

[אפליקציית Launcher 7](#_Toc55577077)

[אפליקציית Installer 8](#_Toc55577078)

[הוראות התקנה ושימוש 9](#_Toc55577079)

[בדיקות וטסטים 10](#_Toc55577080)

[מצגת 11](#_Toc55577081)

# תיאור הבעיה

כשמתכנת יחיד או חברה קטנה מפתחתים תוכנה, הם רוצים להתמקד בפיתוח השירותים והפיצ'רים שהתוכנה מספקת. אבל, הם הרבה פעמים נתקלים בהמון דברים נלווים שצריך לפתח לתוכנה. מנגנונים כמו: ניהול משתמשים, אימות ורכישת רישיון, הפצת עדכוני תוכנה, שליחת מידע בעת קריסה ועוד...

כל המנגנונים האלה יכולים להיות הכרחיים לחלק מהתוכנות, והן יכולות לעכב משמעותית מפתח או חברה בעלת כמות אנשים מוגבלת שאין לה את משאבי העבודה ליצור אותם. הבעיות הללו יכולות להיות מאוד מורכבות לפעמים, בשביל להבטיח שבעתיד, כשהיקף השימוש בתוכנה יגדל, אותן מערכות ימשיכו לעבוד ללא שינויים משמעותיים וללא הפרעות ללקוחות.

הפרויקט שלי יתמקד באחד מהמנגנונים הללו, ואציג כמה דברים מרכזיים שחשוב שיהיה למנגנון זה בשביל להבטיח שימוש נרחב היקף שיוכל להתאים להרבה מאוד תוכנות (ואולי אפילו לכולן עם התאמות קטנות).

המנגנון שאותו אספק בפרויקט הוא מנגנון ל**הפצת עדכוני תוכנה** בפשטות, יעילות ובטיחות, תוך כדי שמירה על אפשרות להרחבה (Scalability). בנוסף, הוא יספק גם תמיכה בעדכוני תוכנה עבור רשתות סגורות (כדוגמאת ארגונים וכו') בעלי DMZ אשר יכול לגשת לרשת. יש לציין כי ברוב מערכות ההפעלה המבוססת Unix יש Repositories שונים שמאגדים בתוכם עדכונים לתכונות ותכנים אחרים, ולכן הפרויקט יתמקד במערכת ההפעלה Windows, ובעיקר בגרסתה החדשה והאחרונה (כנראה), Windows 10.

דבר אחד שחשוב שיהיה באפליקציית הפרויקט הוא אינטגרציה נוחה של בין אפליקציית הפרויקט לתוכנת המפתח, שמעוניין להפיץ עדכונים עבורה. כלומר, שלמפתח יהיה קל ליצור עדכון תוכנה ולהפיץ אותו לשאר הלקוחות.

הדבר השני הוא יעילות הפצת העדכון. כלומר, בעת הפצת עדכון הוא יגיע במהירות לשאר הלקוחות, בעומס מינימלי על שרת העדכונים ובצורה אמינה. כמובן שישנן מגבלות שלא תלויות תוכנה, כמו מהירות העלאה של שרת העדכונים או המרחק הפיזי בין השרת ללקוח, אך המימוש ינסה למצוא פתרון לצמצום המגבלות הללו.

דבר נוסף הוא אבטחת עדכוני התוכנה. כידוע, קוד זדוני אשר מורץ ע"י המשתמש הוא מקור לפרצת אבטחה חמורה. על כן, ישנו חשש שתוקף זדוני יזייף עדכון תוכנה ובכך יגרום למשתמש להריץ תוכנה שלדעתו היא מוכרת ובטוחה, אך בפועל יורץ קוד שתוקף זדוני שלח. על כן, כל עדכון תוכנה וכל מנגון עדכון הגרסאות צריך לספק אבטחה מפני איום כזה או אחר.

לבסוף, אולי בהתחלה לא יהיה צורך בכך, אך כנראה שלאחר שימוש זמן באפליקציית הפרויקט עם תוכנה שמפתח בנה, הוא יצטרך להרחיב את היקף השירות שלו. אם מדובר בגדילה של כמות המשתמשים, או התפשטות של החברה למקומות שונים בעולם. הפתרון צריך לספק אופציה פשוטה, שתהיה שקופה למשתמשי הקצה (המשתמשים הפשוטים לפחות, אם מדובר בחברה אז עובדי IT אולי יצטרכו לעשות התאמות במקרים בהם הרשת היא סגורה) על מנת לתמוך בהקמת עוד שרתי הפצה וניהול הפצת עדכוני הגרסאות כך שכולם מסונכרנים בצורה תקינה.

# הפתרון

תוכנת ה-Updater הינה מעטפת לתוכנה קיימת אשר מאפשרת למפתח התוכנה ליצור ולהפיץ עדכונים לתוכנה שלו עבור מערכת ההפעלה Windows. ה-Updater בנוי בצורה גנרית כך שכל מפתח יוכל להתאים אותו עבור התוכנה אותה הוא פיתח, ובכך להיות פטור מהצורך של בניית מנגנון עדכונים מהיר ומאובטח, שאותו מספקת לו תוכנת ה-Updater.

כאשר המפתח סיים לפתח את התוכנה שלו והוא מעוניין להפיץ אותה למשתמשים חדשים, הוא משתמש ב-Updater על מנת ליצור קובץ התקנה לתוכנה. ביצירת קובץ ההתקנה, המפתח יבחר כתובת לשרת עדכונים, ועוד קונפיגורציות אחרות, שיאפשרו ללקוחות שמתקינים את התוכנה לדגום את שרת העדכונים הרשמי ולהוריד ממנו עדכוני תוכנה.

בנוסף לכך, כאשר לקוח מסוים מוריד עדכון תוכנה, הוא מעדכן על כך את שאר המחשבים ברשת שלו. כל מחשב באותה הרשת שמקבל את ההודעה על עדכון התוכנה יוכל להוריד אותה מהלקוח שכבר הוריד את העדכון, ובהמשך לבחור האם להתקין את העדכון או להשאר בגרסא הישנה. דבר זה יוצר יתרון משמעותי מאוד בעת הפצת עדכון, ומאפשר שיפור ביצועים בהפצת עדכון התוכנה והורדה משמעותית מהעומס על שרת העדכון הראשי.

בנוסף, הפתרון הזה מאפשר גם Scalability. במידה והמפתח ירצה ליצור מספר שרתי עדכון שונים במקומות שונים בעולם, כל שעליו לעשות הוא להתקין את התוכנה על השרתים ולהגדיר את אותם שרתים באותו ה-WAN, כך ששרת העדכון הראשי יוכל לעדכן אותם באמצעות broadcast על עדכון תוכנה. בעת עדכון תוכנה, המפתח יוכל בלחיצת כפתור להפיץ את העדכון ובנוסף ליידע את כל שאר שרתי העדכון להוריד את העדכון אליהם, והם כבר יפיצו אותו באותה הדרך שכל לקוח אחר יפיץ את העדכון ברשת שלו.

כיוון ששיטת ההפצה אינה מסתמכת על "מקור מוסמך", אלא שכל לקוח יכול לשלוח קובץ עדכון לתוכנה, נדרש בפתרון גם מנגנון אבטחה המונע מתוקף זדוני לשלוח קובץ עדכון מזויף ובכך ליצור backdoor למחשבו של לקוח אחר. לכן, כל הודעה, שמקורה משרת העדכונים הרשמי, נחתמת בחתימה דיגיטלית בעזרת מפתח הצפנה אסימטרי המבוסס על שיטת ההצפנה RSA. בנוסף, כל קובץ עדכון נחתם גם הוא בחתימה דיגיטלית באותה הצורה. כאשר מפתח התוכנה יוצר את קובץ ההתקנה בפעם הראשונה, מפתחות ההצפנה מוגרלים בשרת. המפתח הפרטי נשמר לוקאלית על שרת העדכון, והמפתח הציבורי מצורף לקובץ ההתקנה כך שכל לקוח יוכל לאמת באצמעותו הודעות המתקבלות משרת העדכון כבר מהרגע הראשון להתקנה (מפתח ההצפנה לא נשלח ברשת ולכן לא יכול להישלח ע"י תוקף זדוני שהגריל מפתח הצפנה משלו).

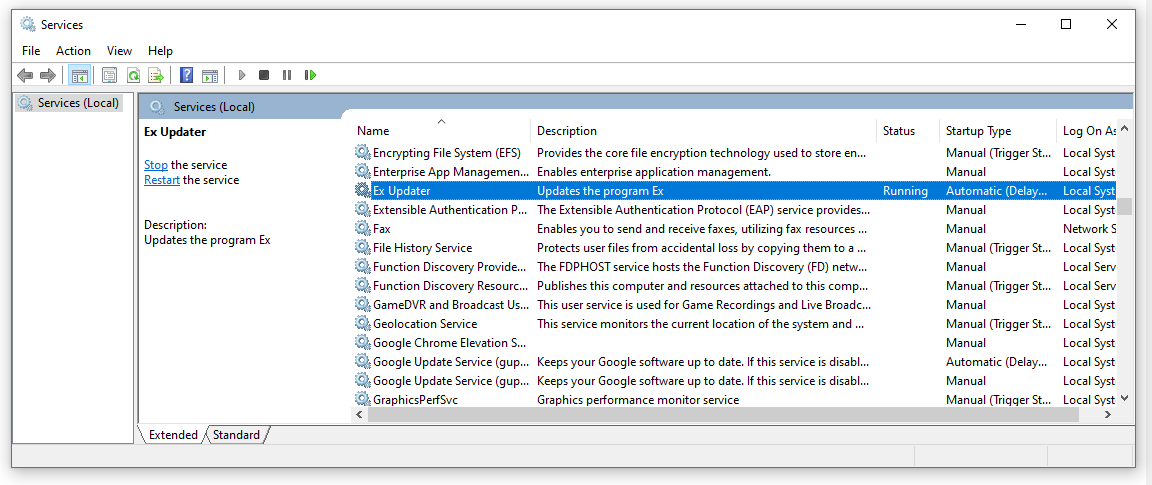
יתר על כן, על מנת שמנגנון ההפצה יהיה מהיר ויעיל, התוכנה לא מחכה שכל לקוח יריץ אותה בנפרד ורק בהרצה שלה העדכון יורד אליו ויופץ ממנו לשאר הלקוחות. במקום זאת, בעת התקנת התוכנה נוצר Service אשר רץ על המחשב תמיד. ה-Service הזה מאזין בקביעות להודעות ברשת ומוריד גרסאות חדשות כשהוא מקבל הודעה על הוצאת גרסא חדשה. לאחר שהגרסא מורדת, הוא מפיץ בתורו הודעה על גרסא חדשה וכן הלאה...

כאשר הלקוח מריץ את התוכנה לאחר שיצאה גרסא חדשה, תופיע לו הודעה שגרסא חדשה הורדה אל המחשב שלו, והיא תשאל אותו האם הוא מעוניין להתקין אותה. במידה ויבחר לעשות זאת, הגרסא החדשה תותקן והגרסא הקודמת תמחק. אחרת, שתי הגרסאות ישארו עד שהמשתמש יבחר להתקין את הגרסא החדשה. בעת הרצת התוכנה, נבדקות כל הגרסאות שהורדו אל המחשב ונמחקות גרסאות שלא נמצאות בשימוש.

# מרכיבי התוכנה

## אפליקציית Updater

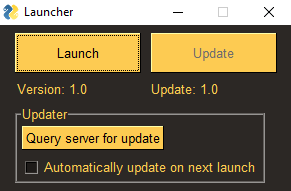
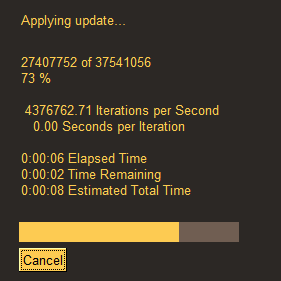
האפליקצייה המרכזית של התוכנה. היא אחראית על קבלת עדכוני גרסא, הורדה שלהם, והפצה של העדכונים לשאר המחשבים ברשת. האפליקצייה היא ליתר דיוק Windows Service שמוגדר לפעול בעליית המחשב והוא תמיד רץ על מנת שיוכל לקבל את עדכוני הגרסא ברגע שהם מופצים, מבלי לפספס אותם, בדומה לאיך ש-Google מעדכנת את התוכנות שלה.



## אפליקציית Launcher

אפליקציית ה-Launcher הינה תוכנה נפרדת בלתי תלויה ב-Updater. היא מותקנת ביחד עם התוכנה ויוצרת קובץ הרצה שאותו המשתמש מריץ כשהוא מעוניין להריץ את התוכנה. כשהמשתמש מריץ את ה-Launcher, נפתח מולו חלון עם פרטים לגבי הגרסא הקיימת, הגרסא המעודכנת ביותר, ואפשרות לבחירה האם הוא מעוניין שהתוכנה תמיד תתעדכן לפני ריצה. בנוסף, הוא יכול לתשאל את שרת העדכונים שלו אם קיימת גרסא חדשה. במידה וכן, ה-Updater כבר ידאג להוריד אותה ולהפיץ אותה, ובהמשך המשתמש יוכל להתקין אותה בפעם הבאה שהתוכנה מורצת.

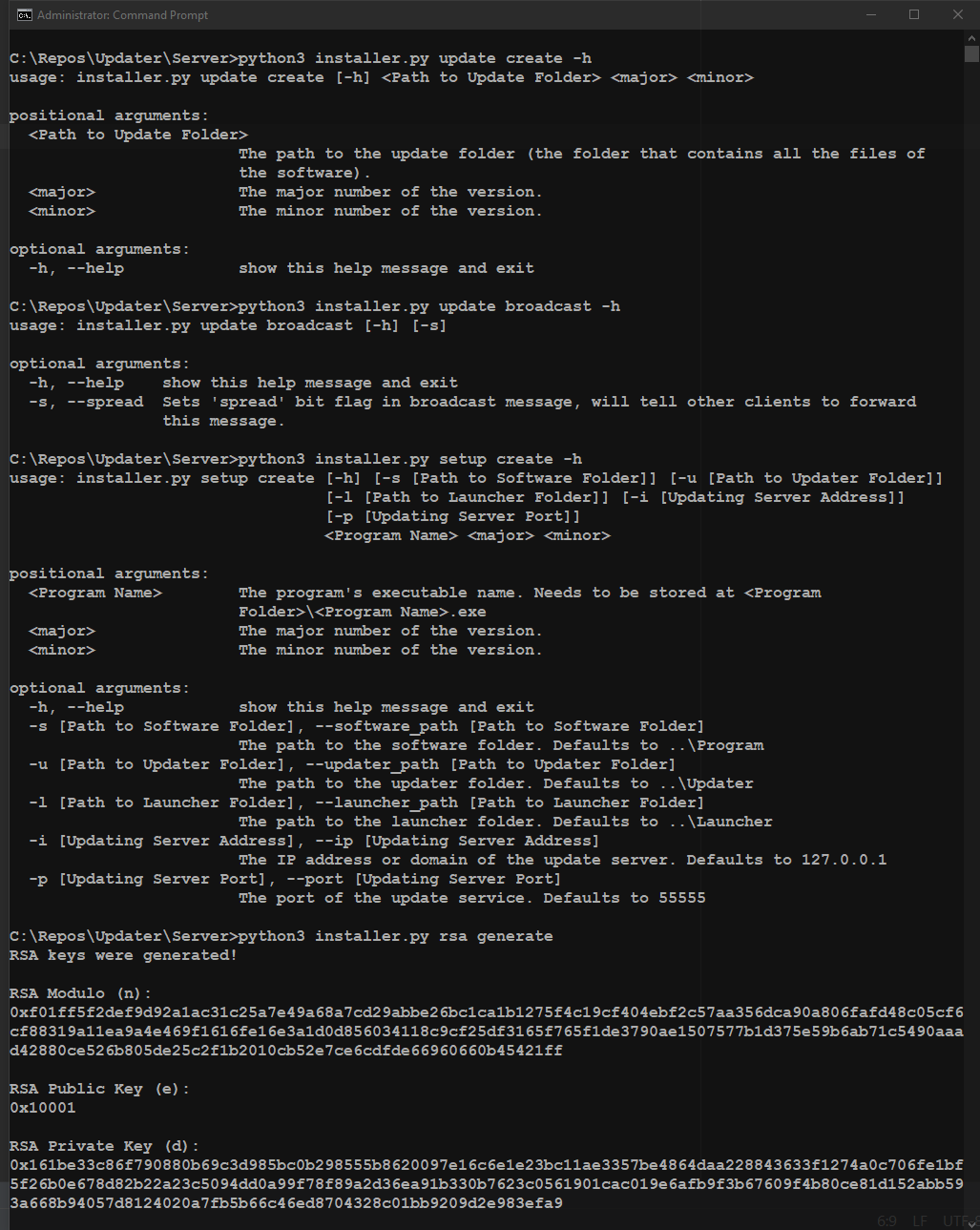
כאשר המשתמש בוחר לעדכן גרסא, ולהתקין את הגרסא המעודכנת, יופיע לו חלון עם progress bar בדומה לחלון הבא:

## אפליקציית Installer

התוכנה הזו היא למעשה מיועדת לשימוש המפתח. בעזרתה הוא יכול ליצור קובץ התקנה עבור התוכנה אותה הוא פיתח, לבחור מפתחות הצפנה עבור תקשורת עם השרת, ליצור עדכוני תוכנה ולהפיץ אותם ללקוחות ברשת. התוכנה מיועדת לרוץ על שרת העדכונים ובעת ריצתה היא מגדירה בו קונפיגורציות מתאימות ע"פ פעולותיו של המפתח. משום כך, התוכנה הזו אינה הופכת לקובץ Executable רגיל של Windows, אלא נותרת קובץ Python שהמפתח יריץ בעזרת ה-Interpreter.

השימוש בתוכנה מתבצע באמצעות ה-CMD והרצת פקודות מסוימות הנתמכות ע"י ה-Installer:



# מימוש הפתרון

## אפליקציית Updater

### מודול rsa\_signing

המודול האחראי על אבטחת המידע שמועבר בין השרת ללקוח ובין הלקוחות. הוא מאפשר לחתום חתימות דיגיטליות באמצעות RSA ולאמת אותן. המודול מייצא את הפונקציות הבאות:

* הפונקציה sign – מאפשרת לחתום על מידע באמצעות מפתח הצפנה פרטי.
* הפונקציה sign\_hash – מאפשרת לחתום על אובייקט hash באמצעות מפתח הצפנה פרטי.
* הפונקציה validate – מאפשרת לוודא חתימה על מידע באמצעות מפתח ציבורי.
* הפונקציה validate\_hash – מאפשרת לוודא חתימה על אובייקט hash באמצעות מפתח ציבורי.

### מודול settings

המודול אחראי על טעינת הגדרות התוכנה בעת תחילת ריצתה ושמירה של הגדרות לאחר שינויים שלהם. בפעם הראשונה חלק מההגדרות יאותחלו לפי פרמטרים דיפולטים, אך לאחר מכן כל ההגדרות ישמרו בקובץ בשם settings.json שישמור את הנתונים בפורמט JSON. חלק מהאפליקציות האחרות גם נעזרות בקובץ הזה בשביל הגדרות שהן צריכות להשתמש בהן. הוא מייצא את הפונקציות הבאות:

* הפונקציה init\_settings - מאתחלת את ההגדרות בתחילת הריצה.
* הפונקציה load\_settings – טוענת את ההגדרות מקובץ ההגדרות settings.json.
* הפונקציה save\_settings – שומרת את ההגדרות בקובץ ההגדרות settings.json.

### מודול registry

המודול אחראי על ביצוע פעולות מול ממשק ה-registry של Windows. בעזרתו מודולים אחראים יכולים לשמור ולאחזר נתונים מה-registry. רוב הנתונים הנשמרים ב-registry תמיד נטענים משם בעת השימוש בהם, כיוון ששם בוודאות ישמרו הנתונים המעודכנים ביותר (במידה ואפליקציה אחרת שינתה את הגדרה מסוימת, ההגדרות באפליקציה אחרת לא יעודכנו אוטומטית, אבל ה-registry כן). הפונקציות שהמודול מייצא הן:

* הפונקציה create\_key – יוצרת registry key חדש.
* הפונקציה key\_exists – בודקת אם registry key בנתיב נתון קיים.
* הפונקציה delete\_key\_recursive – מוחקת registry key ואת כל תכולתו.
* הפונקציה exists – בודקת אם value קיים בנתיב נתון.
* הפונקציה delete – מוחקת value בנתיב הנתון.
* הפונקציה get\_all\_sub\_values – מחזירה רשימה של כל ה-values תחת registry key נתון.
* הפונקציה get\_all\_sub\_keys – מחזירה רשימה של כל ה-keys תחת registry key נתון.
* הפונקציה get\_value – מאחזרת ערך של value מסוים, וממירה אותו למשתנה פייתוני.
* הפונקציה set\_value – שומרת משתנה פייתוני ב-registry תחת registry key נתון.

### מודול messages

המודול

### מודול updater

המודול

### מודול service

המודול

## אפליקציית Launcher

שדגשדג

## אפליקציית Installer

שדג

# הוראות התקנה ושימוש

שדג

# בדיקות וטסטים

שדג

# מצגת

שדג