**Drocsid**

**By:**

Limor Hodory 316306935

Zvika Ravet 316442573

David Raichel 209044833

Idan Shechter 208634071

**Supervisor:**

Yaniv Barkai

**Git:**

https://github.com/Zethu5/Drocsid

תוכן עניינים

[*הגדרת הפרויקט* 4](#_Toc109855075)

[תיאור הפרויקט: 4](#_Toc109855076)

[תוצרים: 4](#_Toc109855077)

[ארכיטקטורה: 5](#_Toc109855078)

[פיצ'רים 6](#_Toc109855079)

[גניבת סיסמאות/כרטיסי אשראי מדפדפנים 7](#_Toc109855080)

[גילוי חיבורי USB והעתקתם 11](#_Toc109855081)

[יצירת יוזר אדמין על המחשב 12](#_Toc109855082)

[הדלקת RDP ושימושו בשביל התחברות מרובת משתמשים 13](#_Toc109855083)

[פגיעה במחשב הנתקף 15](#_Toc109855084)

[ממשק WEB: 17](#_Toc109855085)

[BACKEND 18](#_Toc109855086)

[FRONTEND 22](#_Toc109855087)

[תהליך ההדבקה 26](#_Toc109855088)

[שרת התקיפה 27](#_Toc109855089)

[פונקציות עזר 28](#_Toc109855090)

[Discord API 31](#_Toc109855091)

# הגדרת הפרויקט

## תיאור הפרויקט:

**RAT malware שמתקשרת דרך שרתי דיסקורד.**

**במסגרת פרויקט הגמר פיתחנו תוכנית זדונית הכתובה בפייתון, שמתקשרת עם הc2 שלה באמצעות דיסקורד, למעשה ה-C2 עצמו הוא שרת דיסקורד.  
הצלחנו ליישם את הארכיטקטורה הזו ע"י שימוש ויישום בAPI של דיסקורד הכולל שימוש בbot של דיסקורד שנמצא בהאזנה מתמדת ומחכה לפקודות שלנו.**

**הmalware יספק מגוון של פיצ'רים וישמש להשיג גישה מרחוק על המכונה הנתקפת, בנוסף הmalware ירוץ כל הזמן בצורה מתמשכת ויסתיר את עצמו במהלך הריצה, כך שהמכונה הנתקפת לא תרגיש בדבר.**

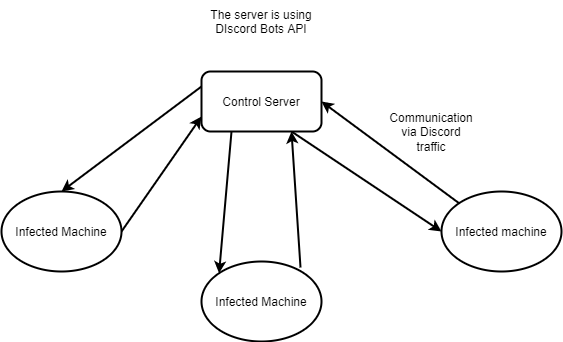
**כל הנתונים בין השרת ללקוח מוצפנים TLS , מה שהופך את תעבורת הרשת של הבוט לקשה לניתוח.**

## תוצרים:

* **וירוס אשר מתקין את עצמו באמצעות dropper..**
* **ממשק ניהול web גרפי אשר מתקשר עם שרת הC2.**
* **שרת C2 אשר מבצע תקשורת שוטפת אל מול כל המכונות המודבקות**

## ארכיטקטורה:

* **שרת C2- שרת דיסקורד ייעודי**
* **Malware- קובץ בינארי שמתחבר לשרת הדיסקורד הייעודי דרך ה-API של דיסקורד**



# פיצ'רים

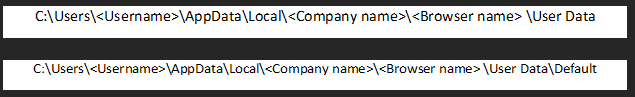
* גניבת סיסמאות/כרטיסי אשראי מארבע דפדנים מרכזיים שרוב האוכלוסיה עושה שימוש בהם כיום והם –Chrome, Firefox, Edge, Opera.
* גילוי חיבור USB למחשב הנתקף והעתקת המידע ממכשיר ה – USB למקום רנדומלי במחשב הנתקף לצורך איסוף מידע.
* יצירת יוזר אדמין על המחשב.
* הדלקת RDP במחשב הנתקף והתלבשות על ה – RDP בצורה כזאת שחיבור למחשב במרוחק לא יתריע לנתקף על החיבור (דבר אשר קורה כאשר יש יותר מבן אחד מחובר למחשב Windows) – גישה בלתי נראית למחשב הנתקף וכל הפונקציונליות הנובעת מכך.
* פגיעה במחשב הנתקף כולל – לקחת בעלות על קבצים/תיקיות, הסרת הרשאות חוץ מהתוקף על הקבצים/תקיות, מחיקת כל המידע מהמחשב כולל קבצי מערכת הפעלה מה שיגרום לקריסת מחשב הנתקף.
* ממשק WEB ל – RAT שמשתמש בשרת ה – 2C כסוג של DB, ה – API נכתב ב – Django וממש המשתמש נכתב ב – Angular.

הממשק נותן לתוקף יכולות תקיפה יותר קלות ונוחות, במקום לכתוב פקודות פשוט אפשר ללחוץ על כפתורים.

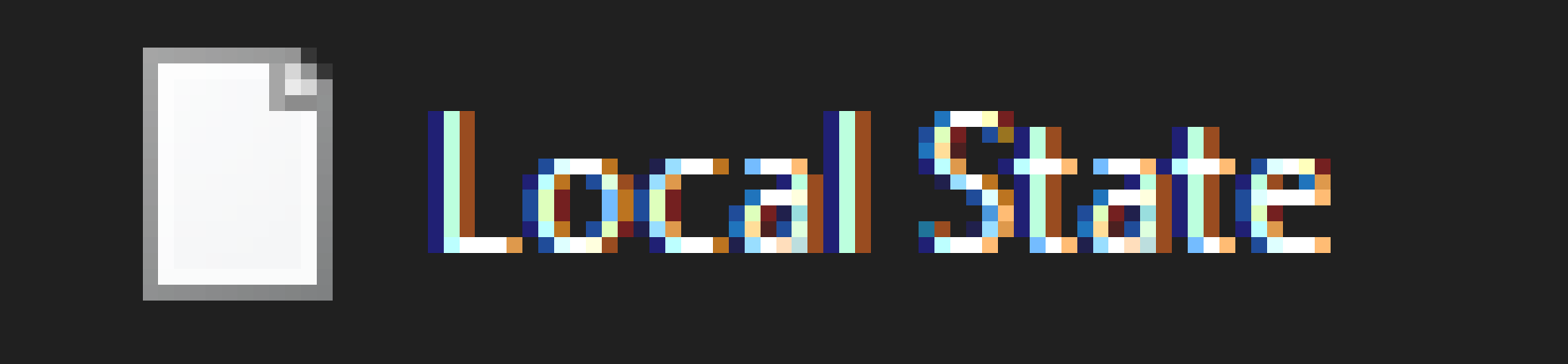
## גניבת סיסמאות/כרטיסי אשראי מדפדפנים

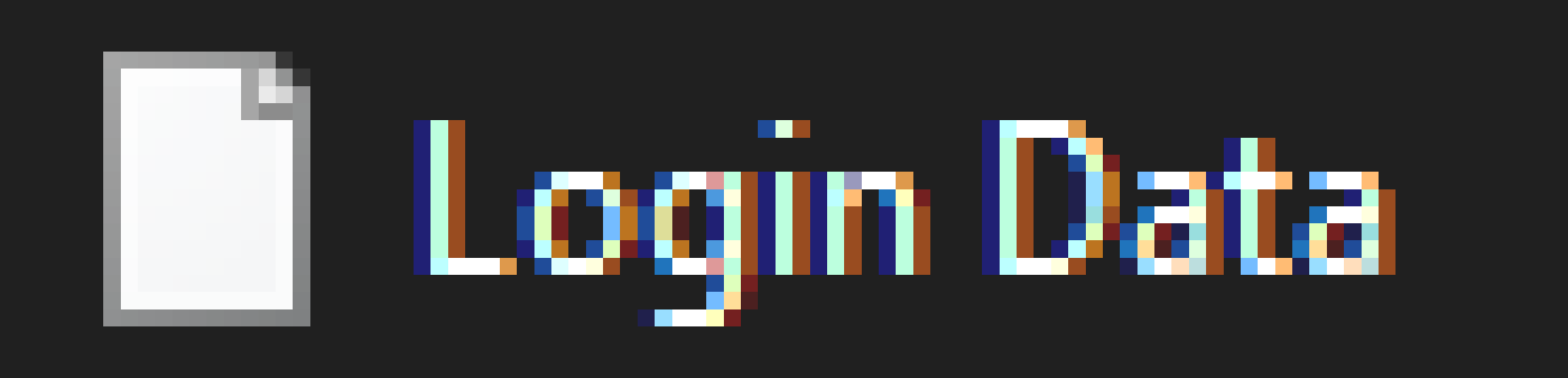
גניבת הסיסמאות ספציפית ל – chrome, opera, edge חוץ מהנתיבים הייתה זהה לגמרי בין כל שלושת הדפדפנים, כנראה בגלל ששלושתם מריצים chromium ומשתמשים באותן שיטות יישום אבטחה.

הקבצים הרלוונטים להתקפה נמצאים תחת שני נתיבים:



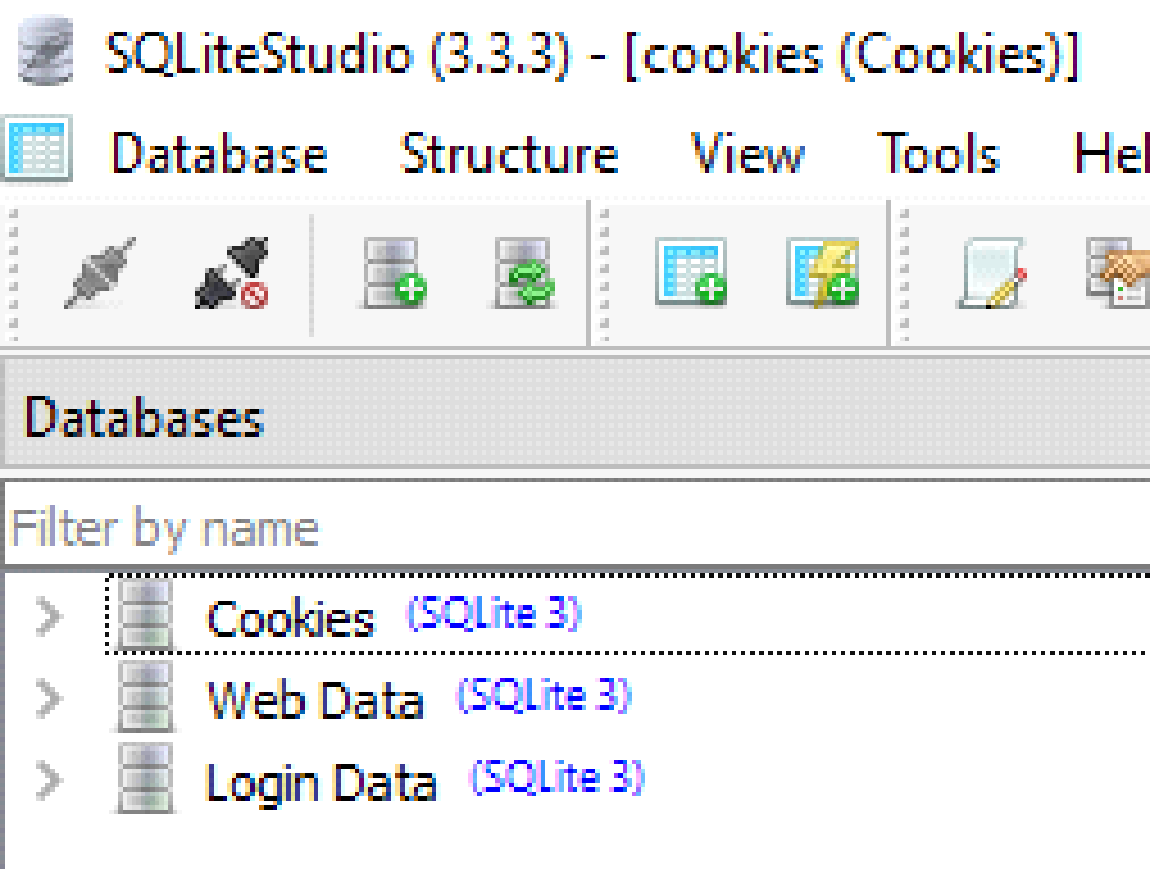
והם:

 - JSON, מכיל את המפתח שבעזרתו נפרוץ את ההצפנה למידע של המשתמש.

 - מכיל את פרטי ההתחברות של המשתמש.

 - מכיל את פרטי כרטיסי האשראי של המשתמשים.

שני הקבצים הסופיים הם בעצם קבצי 3SQLITE וניתן לראות זאת אם נפתח אותם בעזרת תוכנה מתאימה:



בתור התחלה אנחנו מעתיקים את הקבצים הרלוונטיים לצד ונעזרים בהם כדי למנוע קונפליקטים עם הקבצים הראשיים במידה ומישהו משתמש בדפדפנים האלה באותו הרגע.

לאחר מכן אנחנו מוציאים מ – Local State את המפתח שבעזרתו הצפינו את שאר הקבצים, המפתח בעצמו הוא clear text וניתן למצוא אותו בקובץ ה JSON תחת שתי מפתחות אחד אחרי השני: os\_crypt ואז encrypted\_key.

לאחר שמצאנו את המפתח אנחנו מתחברים לשני קבצי ה – DB ומחלצים משם את המידע המוצפן בהתאמה

המידע על המשתמש אשר נמצא במסד הנתונים Login Data מחולק לשלוש עמודות:origin\_url, username\_value, password\_value ששלושתן נמצאות תחת טבלת logins.

המידע על כרטיסי האשראי של המשתמש אשר נמצא במסד הנתוניםWeb Data מחולק לחמש עמודות: name\_on\_card, expiration\_month, expiration\_year, card\_number\_encrypted, name שחמישתן נמצאות תחת טבלת credit\_cards.

לאחר מכן פיענחנו את המידע שהוצפן ב – AES בעזרת המפתח הסודי שהשגנו ממקודם.

דפדפן Firefox היה שונה משלושת הדפדפנים הקודמים מכמה סיבות:

* צורת השימוש בקבצים של Firefox.
* הדרך שבה היינו צריכים להשתמש בשביל לגנוב את המידע.
* עודף האבטחה של Firefox בצורה של Master Password.

Firefox בנו מנגנון אבטחה חזק יחסית לדפדפנים האחרים מכיוון שבתור התחלה, פרטי האשראי לא נשמרים לוקאלית על מחשב הנתקף אלא ב – CLOUD מה שאומר שהנתונים על כרטיסי האשראי של משתמשי Firefox לא יוכלו להיגנב מהדפדפן עצמו אלא רק בצורה עקיפה.

בנוסף לכך Firefox מאפשרים למשתמשים להשתמש בפיצ'ר שנקרא Master Password שבעצם לפני כל מילוי תיבת טקסט באינטרנט תהיה מה שתהיה – אימייל, משתמש, סיסמה, כרטיס אשראי וכו' המשתמש ייצטרך להכניס סיסמה ראשית ורק אם יכניס אותה הפרטים יתמלאו אוטומטית, מה שבנוסף הדבר הזה עושה הוא שהוא מצפין את קובץ פרטי המשתמש ככה שאנחנו אפילו לא נוכל לפתוח אותו בשביל לנסות לפרוץ אותו אלא אם כן אנחנו יודעים את ה – Master Password, אז אנחנו בודקים לפני האם Master Password קיים ורק אם לא אנחנו ממשיכים.

הקובץ הרלוונטי לפרטי המשתמש נמצא תחת:

ושמו הוא 

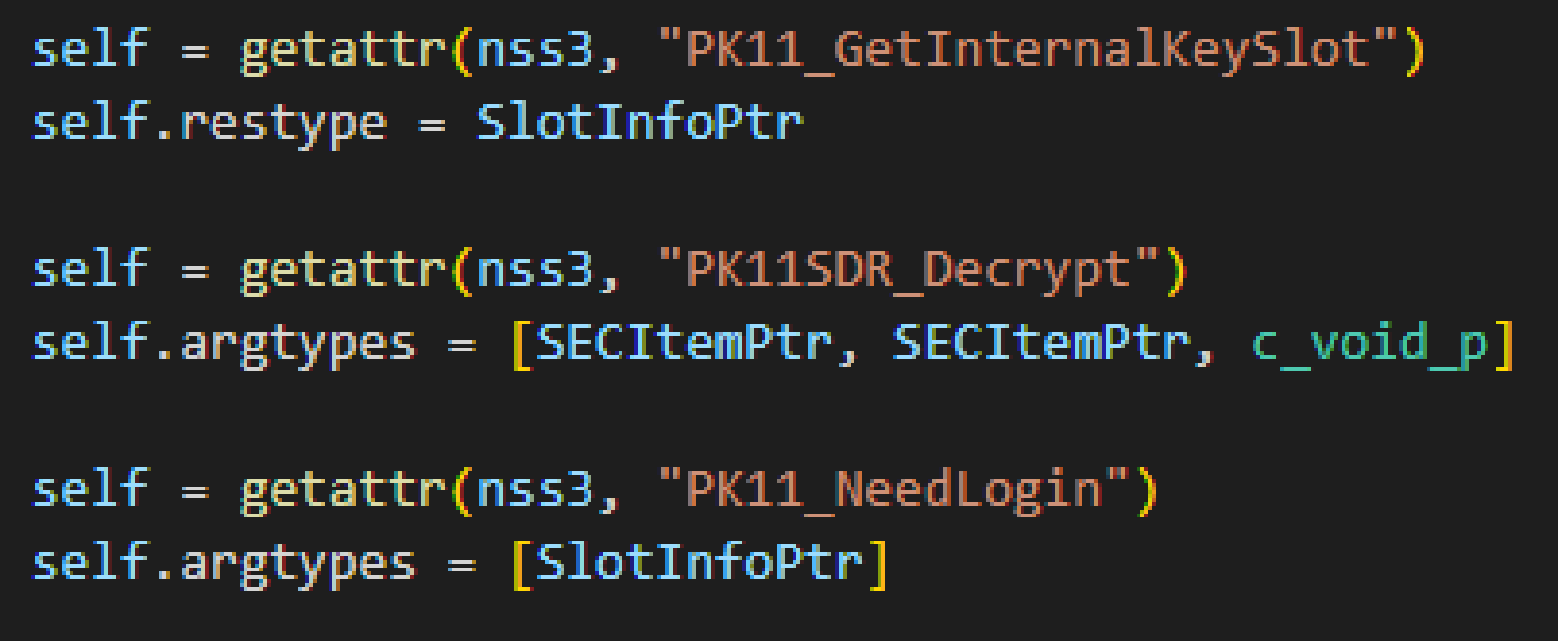
דבר ראשון שעשינו זה לקרוא את הקובץ ולהוציא ממנו שלושה שדות רלוונטיים: hostname, encryptedUsername, encryptedPassword.

לאחר מכן עברנו לשלב ה – decryption.

הדרך שבה תקפנו את Firefox היה דרך שימוש ב – DLL הפנימי של הדפדפן כנגד עצמו:

Firefox בנו לעצמם אוסף פונקציות פנימיות לשימוש ואיסוף פרטי המשתמש ב – DLL שקוראים לו NSS3.dll

טענו את ה – DLL ואת הפונקציות שלו והשתמשנו בהן:



שלושת הפונקציות הבאות היו הרלוונטיות מסך כל הפונקציות.

PK11\_GetInternalKeySlot – נותן גישת חיבור למסד הנתונים הפנימי של Firefox ולמתודות שניעזר בהן לאחר מכן, כל שימוש במתודה דורש שנהיה מחוברים ל – key\_slot.

PK11\_NeedLogin – מוודא שאכן התחברנו כראוי, שאין שימוש ב – Master Password ושניתן להמשיך בשימוש המתודות.

PK11SDR\_Decrypt – המתודה שמפענחת לנו את המידע המוצפן.

לאחר שביקשנו key\_slot בעזרת הפונקציה הראשונה, ווידאנו שהתחברנו כראוי בעזרת הפונקציה השנייה ושאין שמוש ב – Master Password, עברנו על כל הפרטים המוצפנים של המשתמש ופיצחנו אותם עם הפונקציה השלישית.

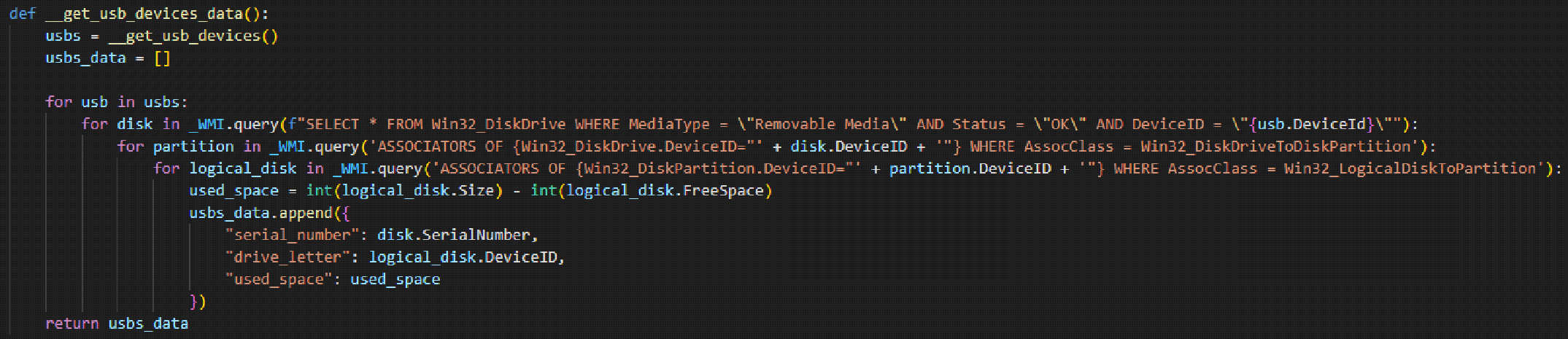
לאחר שפיענחנו את המידע מכל הדפדפנים אנחנו מעתיקים אותו לקובץ JSON ב – clear text אשר יישלח חזרה לשרת ה – C2 של התוקף.

## גילוי חיבורי USB והעתקתם

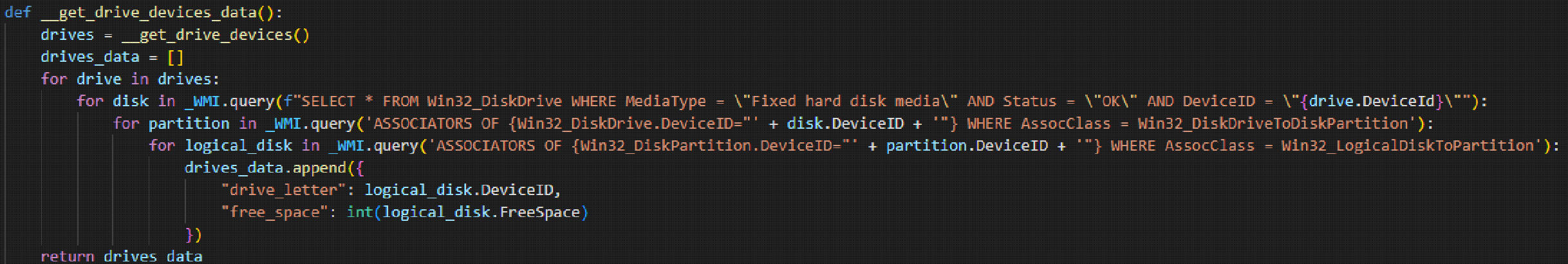
חיבורי USB ניתנים לגילוי בקלות בעזרת מתודות של Win32 אשר Windows נותנת לשימוש בצורה של שאילתות WMI.

דבר ראשון שאנחנו עושים זה בעצם לתשאל את ה – WMI האם באמת קיימים כונני USB מסוג של Removable Media ושהמצב שלהם תקין.

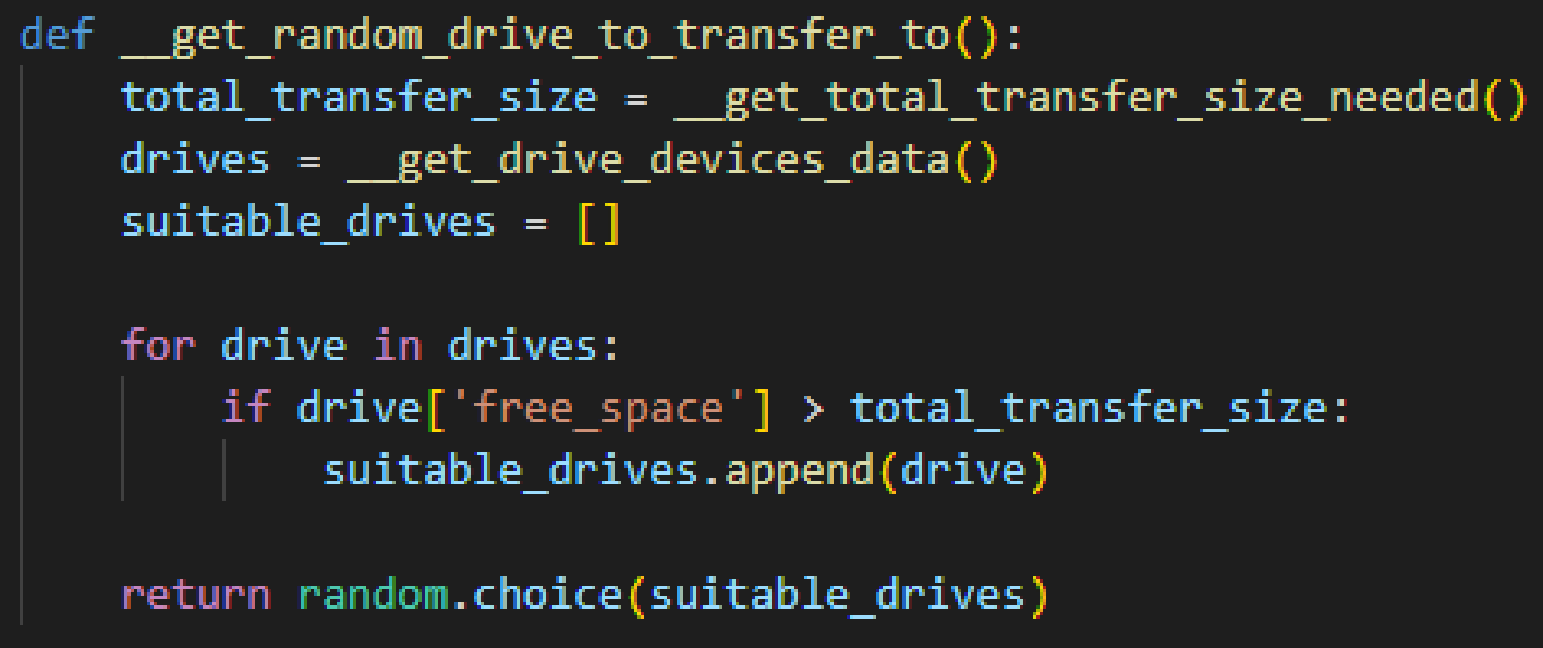
אם אכן קיימים כוננים כאלה אנחנו משיגים את המספר הסיריאלי שלהם, את האות שלהם כפי שהם מוצגים במערכת ההפעלה כגון C:\, D:\ וכו', ואת נפח המידע המשומש ב – USB.



לאחר מכן אנחנו בוחרים נתיב רנדומלי במחשב הנתקף שיש לו מספיק מקום לאכסן את המידע



ומעתיקים לשם את הכל ובנוסף שולחים לתוקף את הנתיב הרנדומלי לשימוש מאוחר יותר.



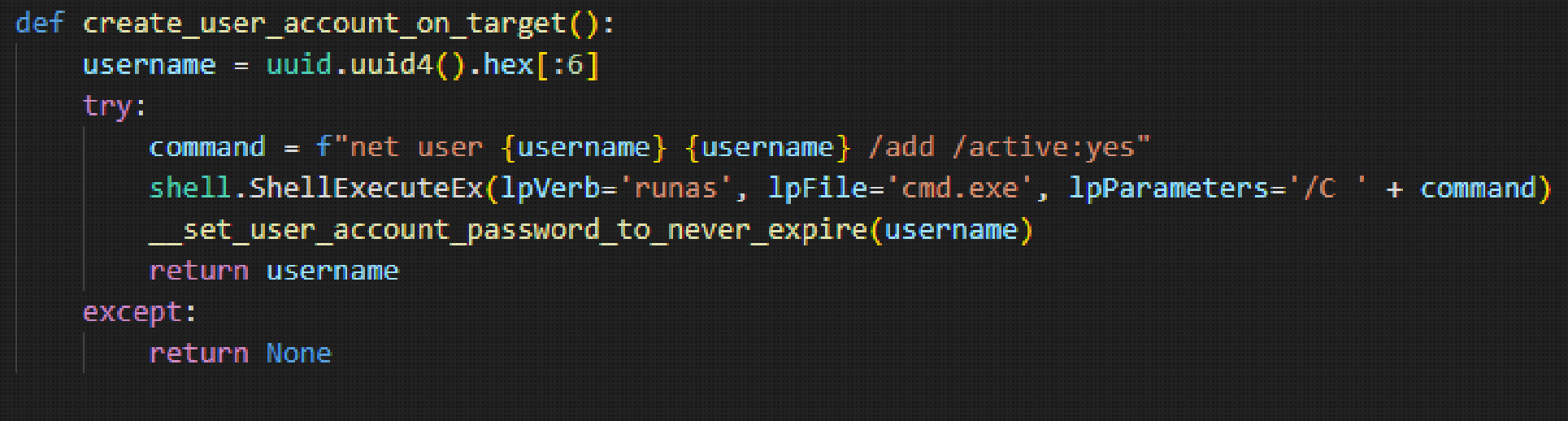
לאחר ההעתקה אנחנו בקלות יכולים לגשת לאותו נתיב עם חיבור מרחוק/העלת הקבצים לשרת C2.

## יצירת יוזר אדמין על המחשב

אם הנתקף כבר אדמין על המחשב אנחנו יכולים להשתמש בזה כדי ליצור עוד אדמינים נוספים בשביל עוד הזדמנויות תקיפה.

כל שלב מוגן ככה שבמידה והמשתמש לא אדמין על המחשב התהליך פשוט ייפסק.

שלב ראשון אנחנו יוצרים משתמש על המחשב בעזרת פקודות CMD רגילות:



שם המשתמש הנוצר יהיה 6 אותיות hex רנדומליות.

בנוסף לכך המשתמש גם לא ייצטרך להחליף סיסמה בשביל להקל על תהליך התקיפה.

שלב שני נוסיף את המשתמש לקבוצת האדמינים הקיימת במחשב:

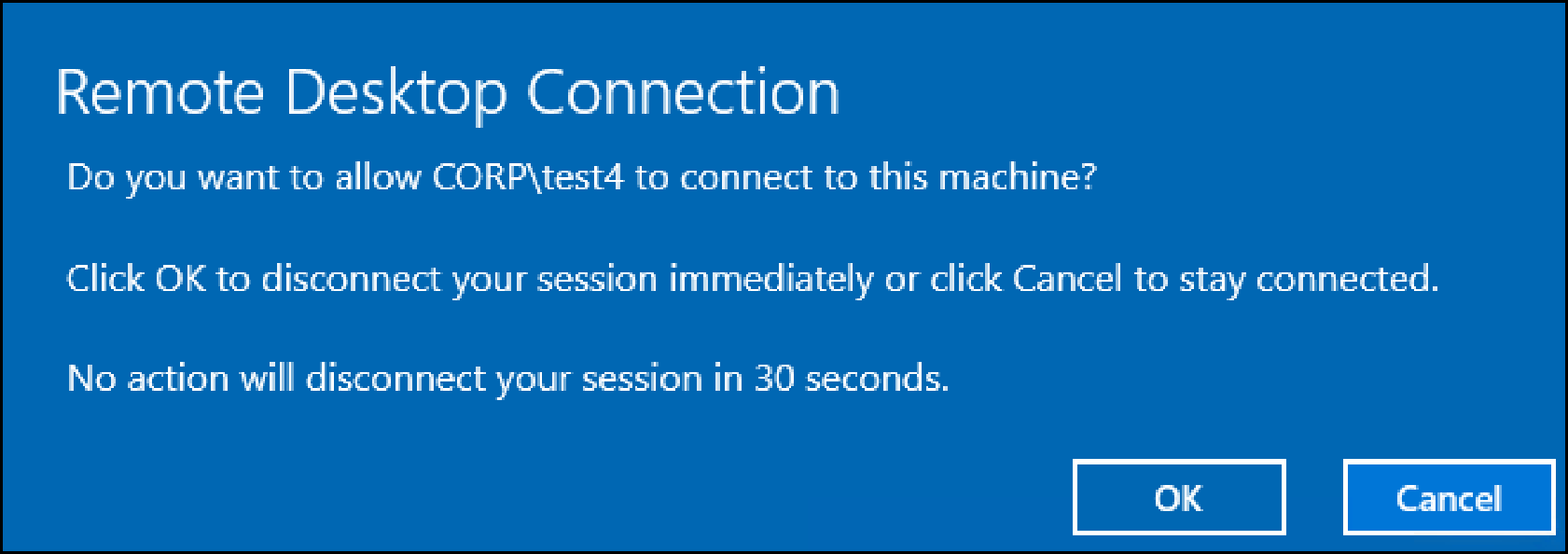


וככה הפכנו אותו מיוזר רגיל לאדמין.

## הדלקת RDP ושימושו בשביל התחברות מרובת משתמשים

ה – RAT שלנו נותן פונקציונליות בסיסית מבחינת ביצוע פעולות במחשב הנתקף, יכול להיות שישנן פעולות אשר נרצה לעשות שה – RAT לא יאפשר לנו ובשביל זה החלטנו ליצור פי'צר שישפעל RDP למחשב וייתן לנו להתחבר אליו בצורה מרוחקת כמו הנתקף.

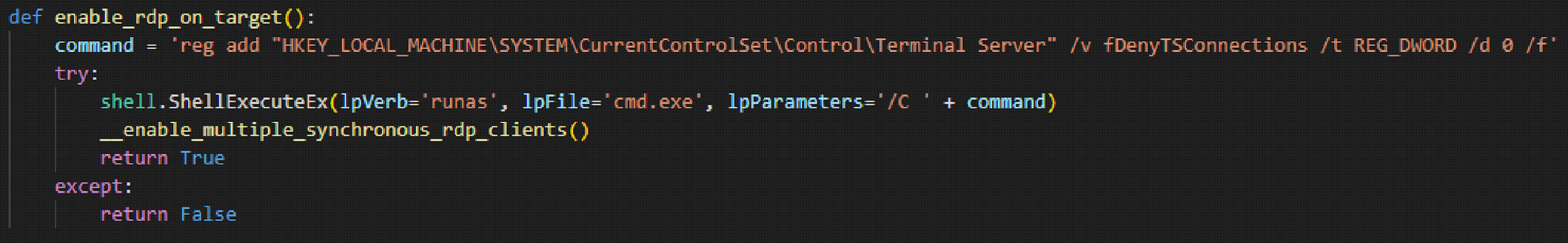
הבעיה בגישה הזאת היא ש – Microsoft ידעו שזה כשל אבטחתי ולכן הגבילו את ההתחברות לכל מחשב בעל מערכת הפעלה של משתמש רגיל כגון Windows vista, 7, 8, 10, 11 וכו' לפעול בצורה כזאת שרק בן אדם יכול להתחבר למחשב אחד בכל פעם ולא תינתן גישה לכמה משתמשים שמנסים להתחבר לאותו המחשב באותו הזמן, תופיעה הודעה של בקשה לנתינת גישה למשתמש השני שמנסה לגשת:



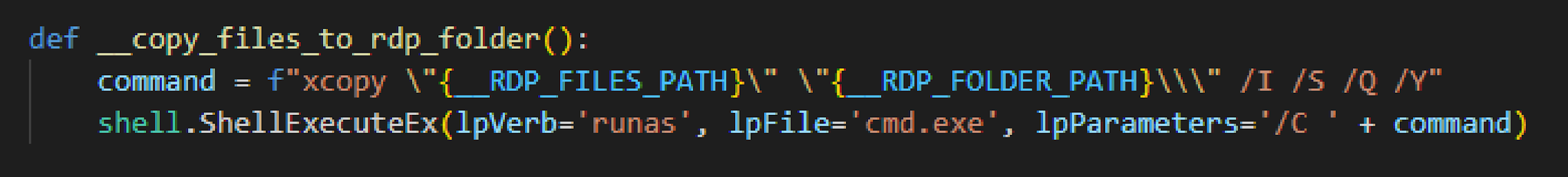
כמובן שבשרתי Windows Server ההגנה הזאת לא משופעלת מפעת היותם שרתים.

בשביל לעקוף את ההגנה הזאת הלבשנו RDP WRAPPER על פרוטוקול ה – RDP הקיים של Windows ושפעלנו אותו ככה שכל פעם שמישהו מתחבר למחשב בצורה מרוחקת אז ההגנה הזאת לא משופעלת והחיבור שלו הופך להיות בעצם סוג של תהליך פנימי שמנוהל ע"י המשתמש הראשי של המחשב או במקרה שלנו הנתקף.

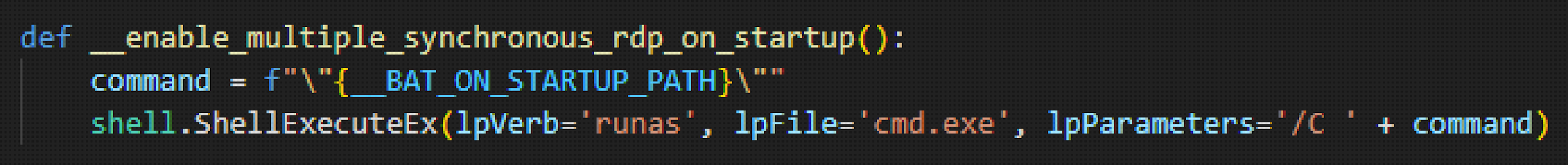
דבר ראשון שאנחנו עושים הוא להפעיל את ה – RDP על מחשב הנתקף:



לאחר מכן אנחנו מעתיקים את ה – WRAPPER לנתיב מסוים:



מפעילים אותו:

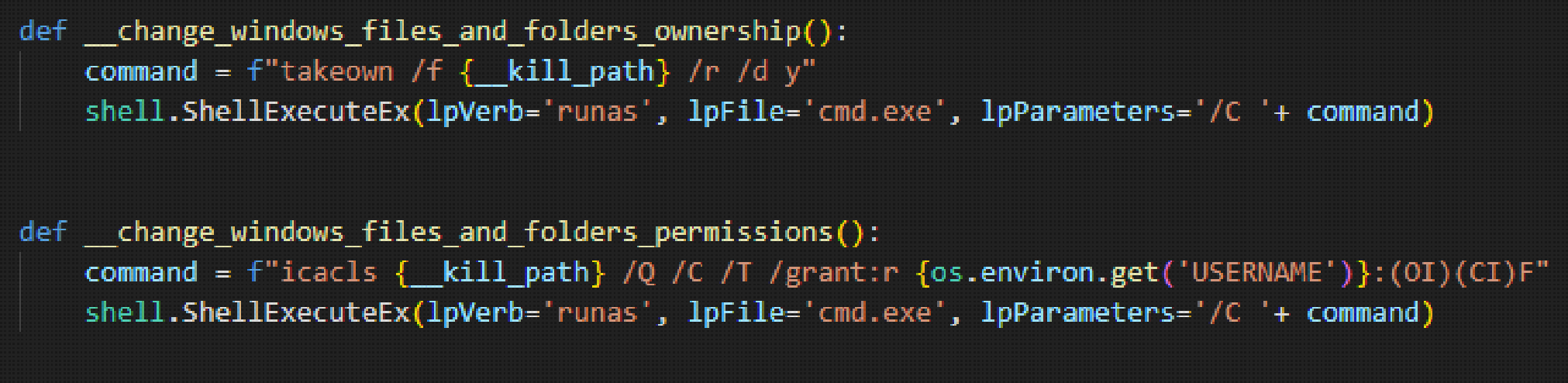


לבסוף כמה משתמשים יוכלו להתחבר למחשב ביחד מבלי חשש שיהיה צריך להתחבר לאישור של הבן שכרגע משתמש במחשב.

## פגיעה במחשב הנתקף

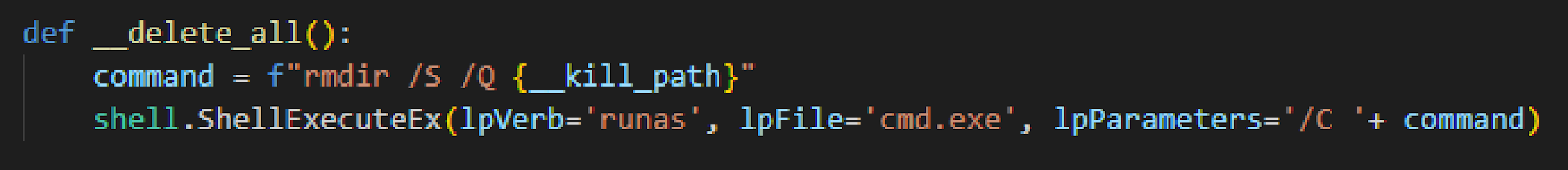
פגיעה במחשב הנתקף זה פיצ'ר שפיתחנו למקרה ונצטרך מאיזושהי סיבה להבטיח שלא ימצאו עקבות שלנו במחשב.

מה שאנחנו עושים זה בעצם לקיחת בעלות על כל הקבצים/התיקיות,להוריד לכולם חוץ מאיתנו את ההרשאות על כל הקבצים/התיקיות ולבסוף למחוק את כל הקבצים/התיקיות הללו.

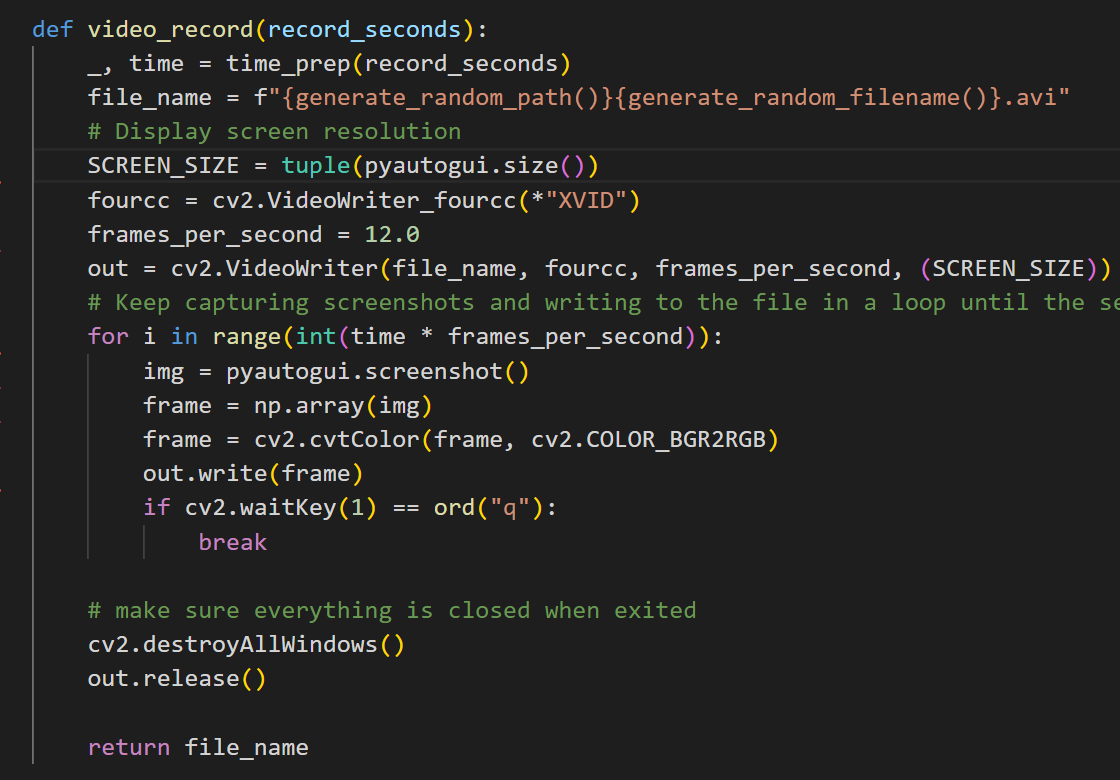


הסיבה שאנחנו קודם כל לוקחים בעלות והרשאות לקבצים/תיקיות היא בשביל שלא ייווצרו קונפליקטים בשלב המחיקה שמערכת ההפעלה פתאום תקפיץ לנו הודעה של חוסר הרשאות וכו'.

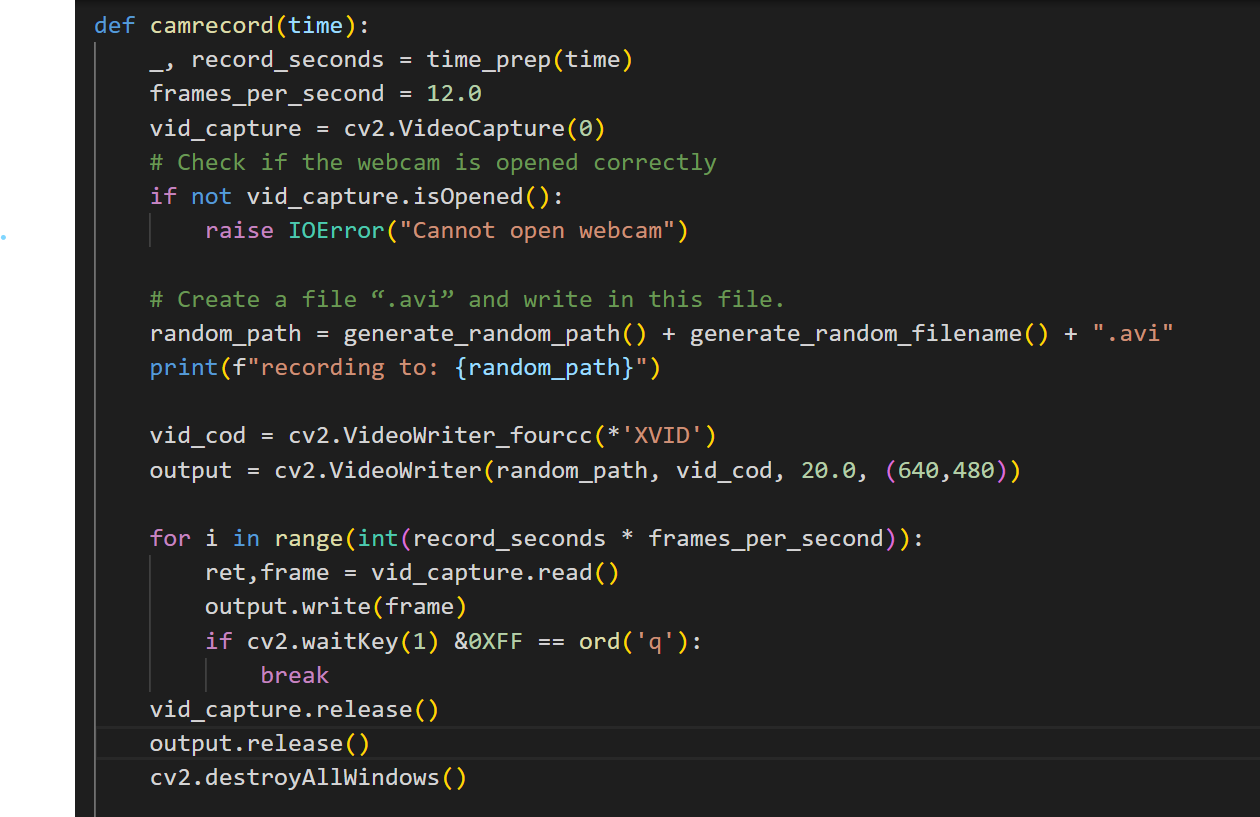
המידע אשר נמחק הוא לא רק מידע פרטי של המשתמש, אלא גם קבצים קריטיים למערכת ההפעלה.



לאחר שכל המידע נמחק אנחנו מכבים לנתקף את המחשב, בפעם הבאה שהוא ידליק אותו מערכת ההפעלה לא תצליח לעלות והוא יחוייב לפרמט את המחשב בשביל לגרום לו לעבוד מחדש.

צילום ווידיאו של הנתקף לפי מספר שניות שנחליט 

צילום מסך של הנתקף



# ממשק WEB:

ממשק ה – WEB אשר בנינו בנוי בסופו של דבר משלושה חלקים, ה – WEB, ה – APIים ואחסון המידע של הנתקף בשרת ה – C2 שלנו.

**תצורה כללית:**

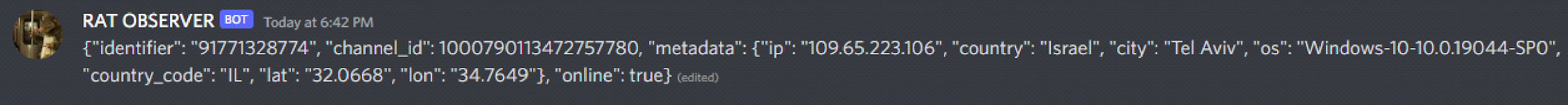
ישנם 2 BOTים שעובדים בסופו של דבר בשביל שכל המערכת תרוץ:



RAT – הנוזקה עצמה שרצה על המחשבים שנדבקו, אחראי לביצוע פקודות שאנחנו שולחים אליו מהשרת בעזרת ה – API של דיסקורד והחזרת התשובה למקום הנכון.

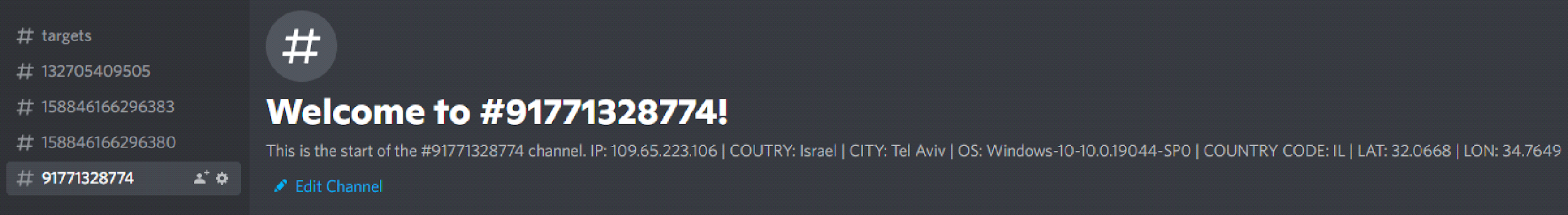
RAT OBSERVER – אחראי לניהול המידע של המחשבים הנדבקים, בנוסף אחראי על בדיקת סטטוס המחשבים – האם למעלה או למטה.

כל פעם שמחשב חדש נדבק ב – RAT שלנו, ה – RAT שולח קריאה ל – RAT OBSERVER שיעדכן בשרת התקיפה שלנו את הפרטים של המחשב בצורה של JSON אבל כהודעה רגילה בשרת הדיסקורד בערוץ המיועד לכך שקראנו לו targets:



בעצם ערוץ הטקסט שפתחנו הופך להיות סוג של מסד נתונים לניהול כל אותם מחשבים נדבקים.

לאחר מכן ה – RAT OBSERVER ייצור למחשב החדש ערוץ טקסט ייעודי בשבילו ששם יישלחו כל הפקודות הנוגעות לאותו מחשב ובנוסף לכך גם כל המידע שהמחשב יחזיר חזרה אלינו:



בשלב הזה למחשב החדש יש ערוץ משלו והפרטים מעודכנים, כעת הוא יישלח לערוץ שנוצר בשבילו הודעת ping! כל דקה בערך כדי לתת משוב לשרת התקיפה שהוא למעלה ושניתן לשלוח אליו פקודות:



ה – RAT OBSERVER ידגום את ההודעות האלה כל שתי דקות ובהתאם יסמן לאותו מחשב האם הוא למעלה או למטה.

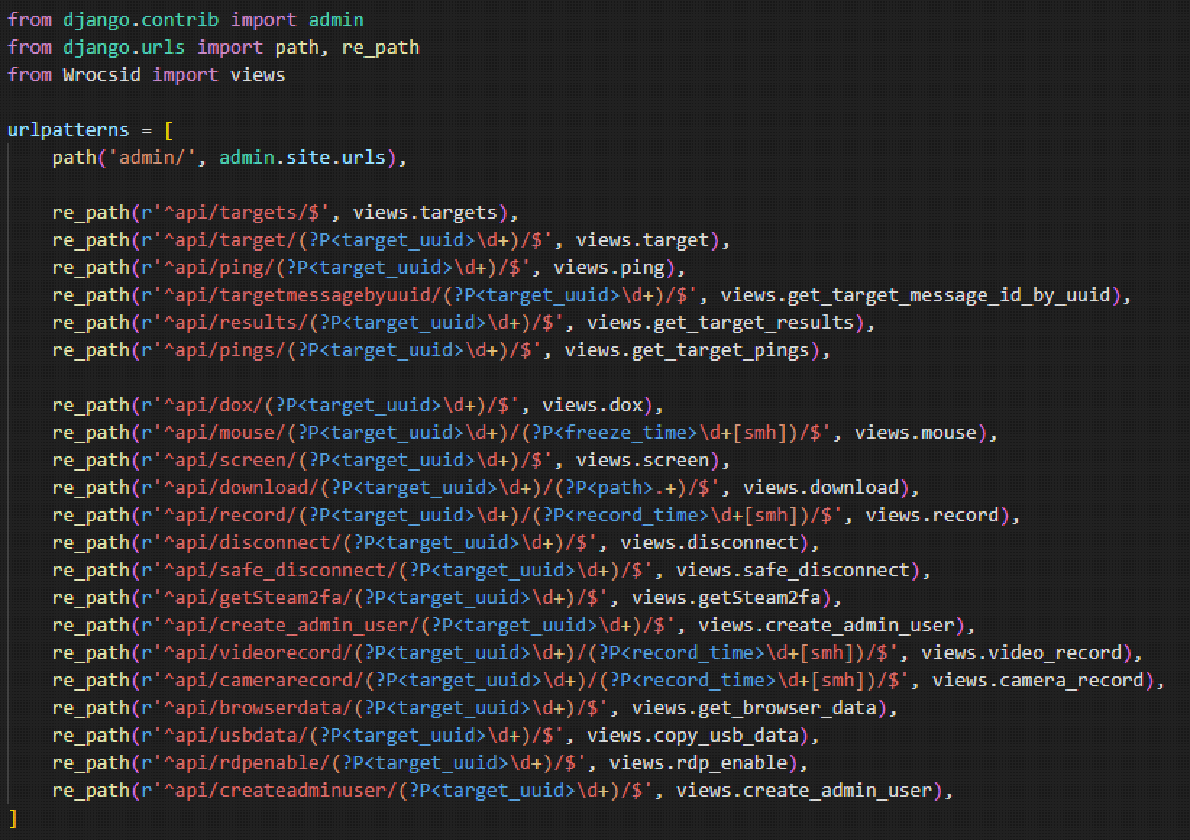
## BACKEND

בגלל שהנוזקה עצמה נכתבה ב – Python, בחרנו להשתמש בתשתית BACKEND מתאימה שהיא בעצם Django.

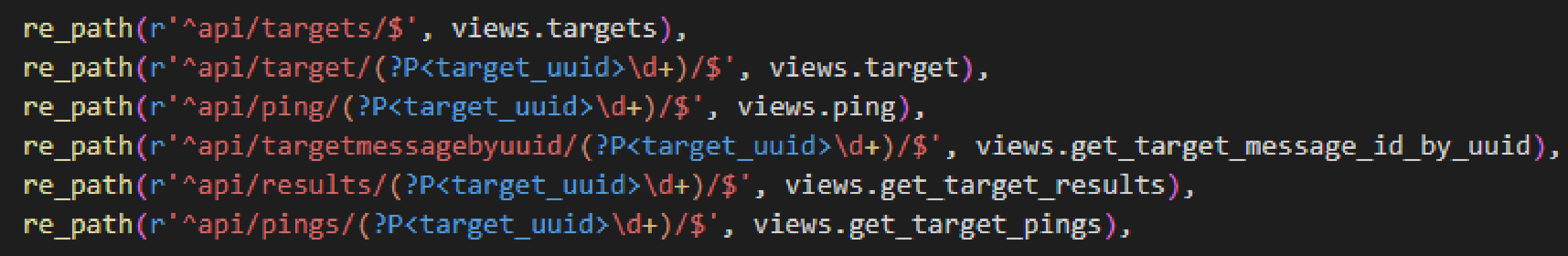
Django נותנת לנו לכתוב APIים ב – python בצורה נוחה וככה שלא נצטרך לשלב כמה שפות שונות אלא להישאר באותה אחת שהתחלנו איתה.

עיקר ה – BACKEND עצמו הוא שני קבצים שהראשון בהם הוא ה – urls.py:

בעצם כל נתיב API נותן לנו גישה לפעולה אחרת אשר רצה על שרת התקיפה או על מחשב הנתקף:



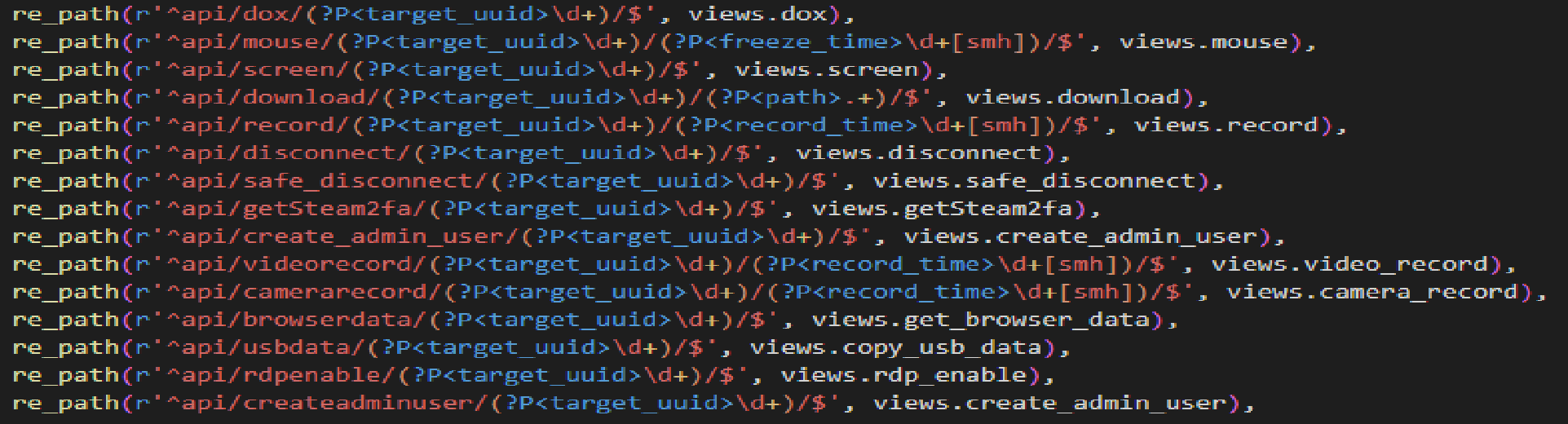
החלק הראשון של ה – APIים הם בעצם פעולות עזר, ולא הפקודות עצמן שהנוזקה מריצה:



לפי הסדר:

* קבלת מידע על כל המטרות שהדבקנו.
* קבלת מידע על מטרה ספציפית.
* ביצוע פינג למחשב.
* קבלת כל ההודעות שהמחשב הנתקף שלח חזרה לשרת התקיפה.
* קבלת כל התוצאות של הפקודות שהרצנו על הנתקף.
* קבלת כל הפינגים שהמחשב הנתקף ביצע.

החלק השני של ה – APIים מתייחס לפקודות שניתן להריץ על המחשב הנתקף:

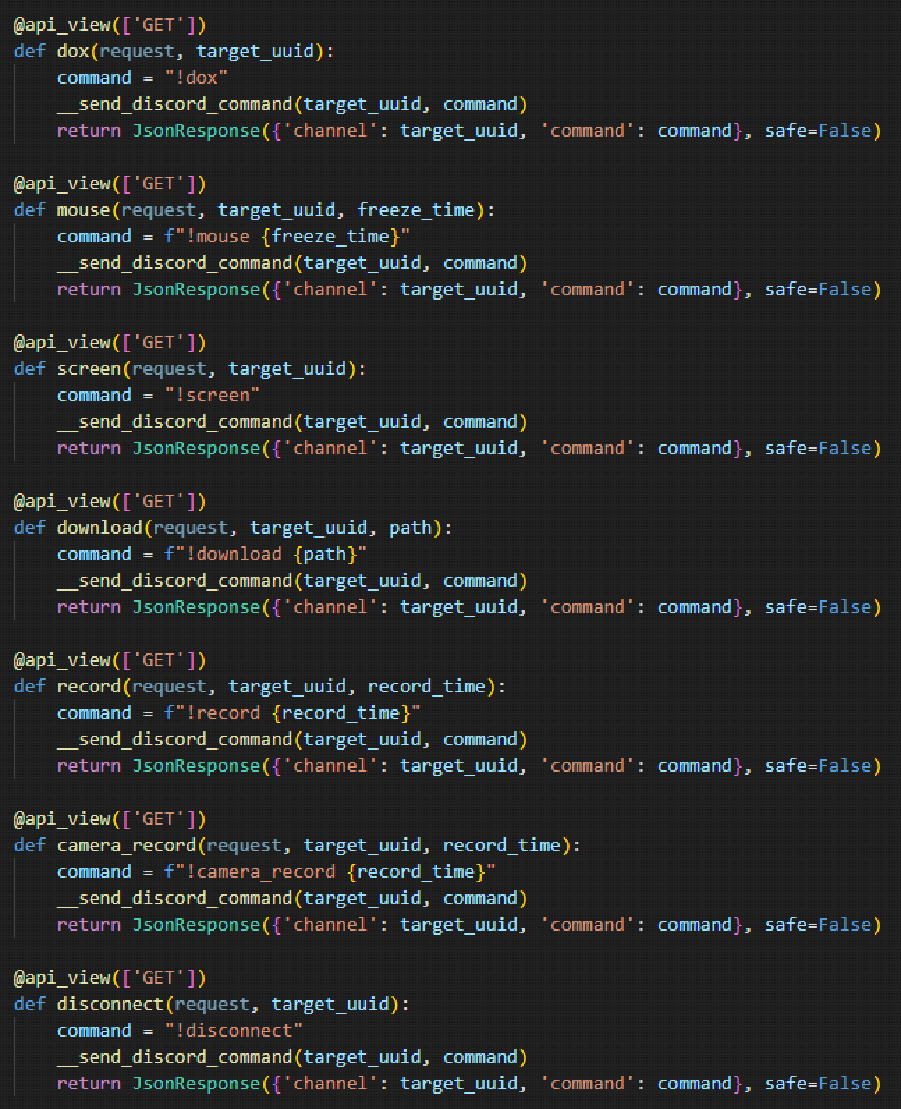


לפי הסדר:

* שליחת IP של הנתקף.
* לעצור תזוזת עכבר.
* צילום תמונת מסך.
* הורדת קובץ ממחשב הקורבן לשרת התקיפה.
* צילום וידאו של המסך של הנתקף.
* התנתקות ממחשב הנתקף.
* התנתקות בטוחה ממחשב הנתקף.
* הורדה של קבצי ה – FA2 של Steam ממחשב הנתקף לשרת התקיפה.
* צילום וידאו של מצלמת הנתקף אם יש לו אחת.
* צילום תמונת של מצלמת הנתקף אם יש לו אחת.
* גניבת פרטי המשתמש/האשראי של המשתמש מהדפדפנים במחשב.
* זיהוי והעתקת מידע מ – USBים שמחוברים למחשב.
* שפעול RDP במחשב הנתקף.
* יצירת אדמין במחשב הנתקף.

כל נתיב API כזה מפעיל פונקציה אחרת אשר אחראית לפעולה שונה.

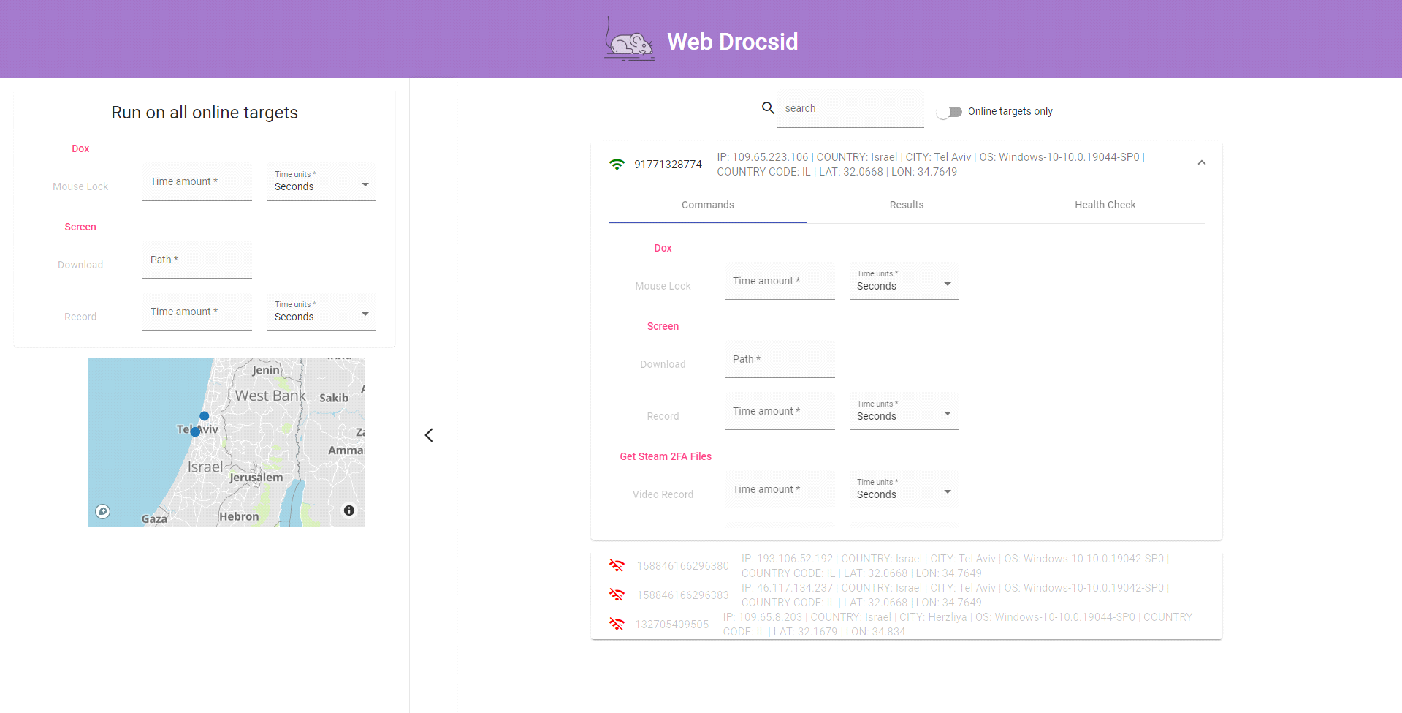
הפונקציות האלה נמצאות בקובץ השני החשוב לנו שנקרא views.py:



לדוגמה כמה פונקציות שנמצאות בו, בעצם מבצע את הפקודה בשרת התקיפה שלנו ומחזיר את המידע בהתאם לפקודה חזרה כ – RESPONSE.

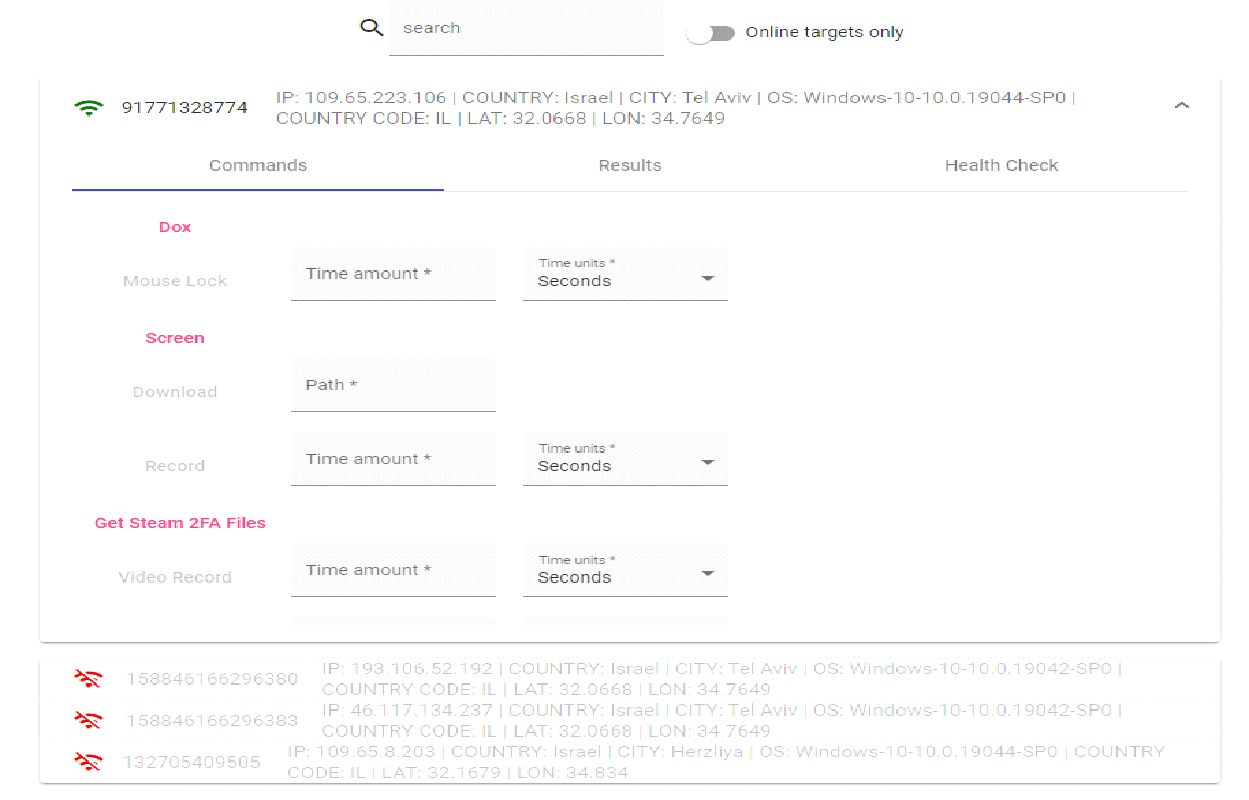
## FRONTEND

ה – FRONTEND בנוי בעזרת תשתית Angular ונראה כך:

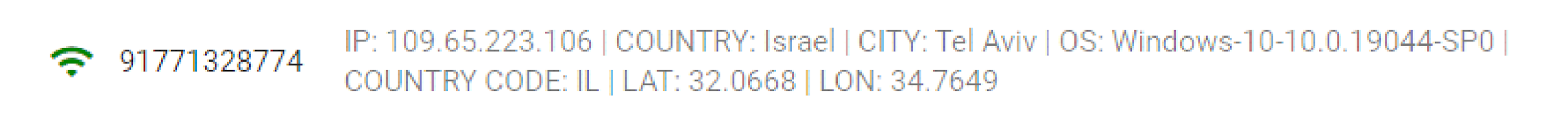


בעצם יש לנו כמה אלמנטים שונים בדף וניגע בכל אחד מהם.

כאשר מתחברים לאתר ניתן לראות את המחשבים אשר נדבקו בנוזקה שלנו מסודרים אחד אחרי השני, בנוסף שניתן לפלטר בין המחשבים אם בצורת חיפוש ואם בצורה של רק מחשבים שלמעלה לשם נוחות:



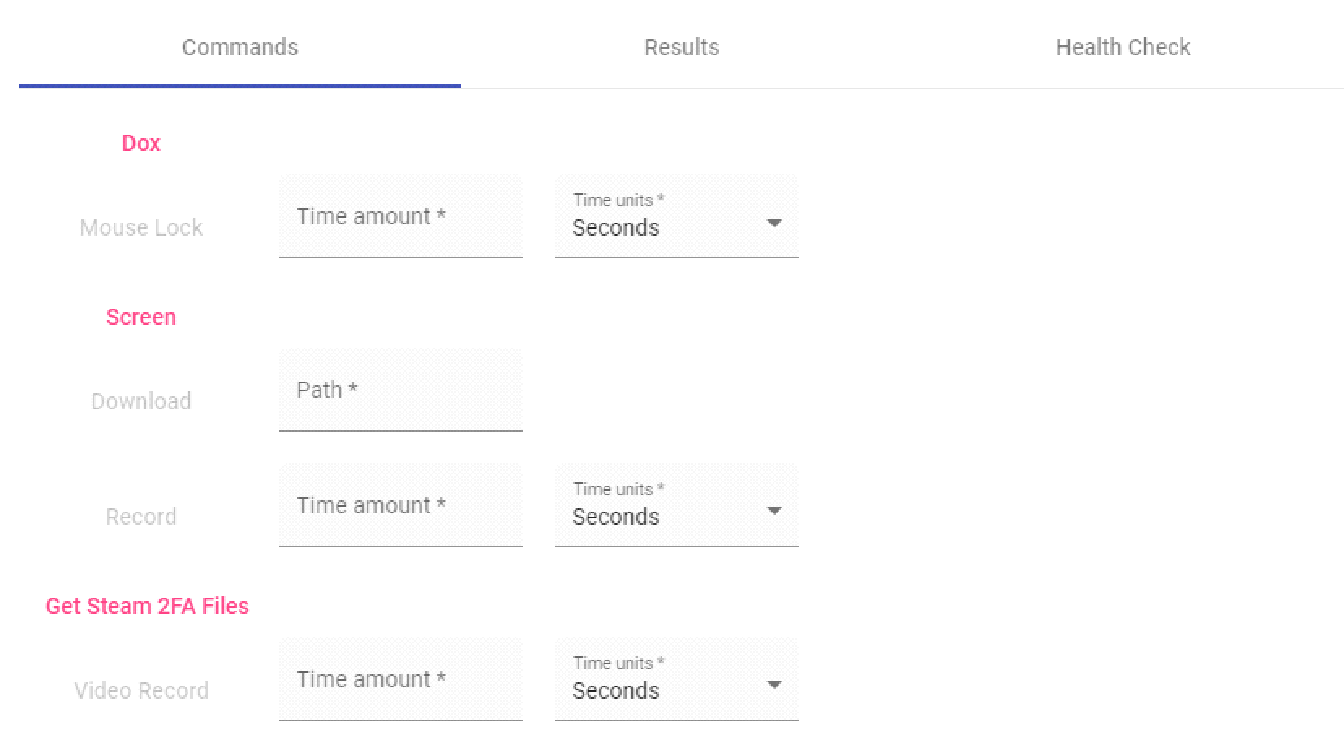
הצורה שבה המחשבים מסודרים היא שלכל מחשב יש TAB משל עצמו עם כמה פרטים:



קודם כל המחשב מראה את ה – Unique ID שהוא קיבל מהנוזקה כדי שנוכל לזהות אותו בהתאם, בנוסף ל – META DATA שאספנו עליו.

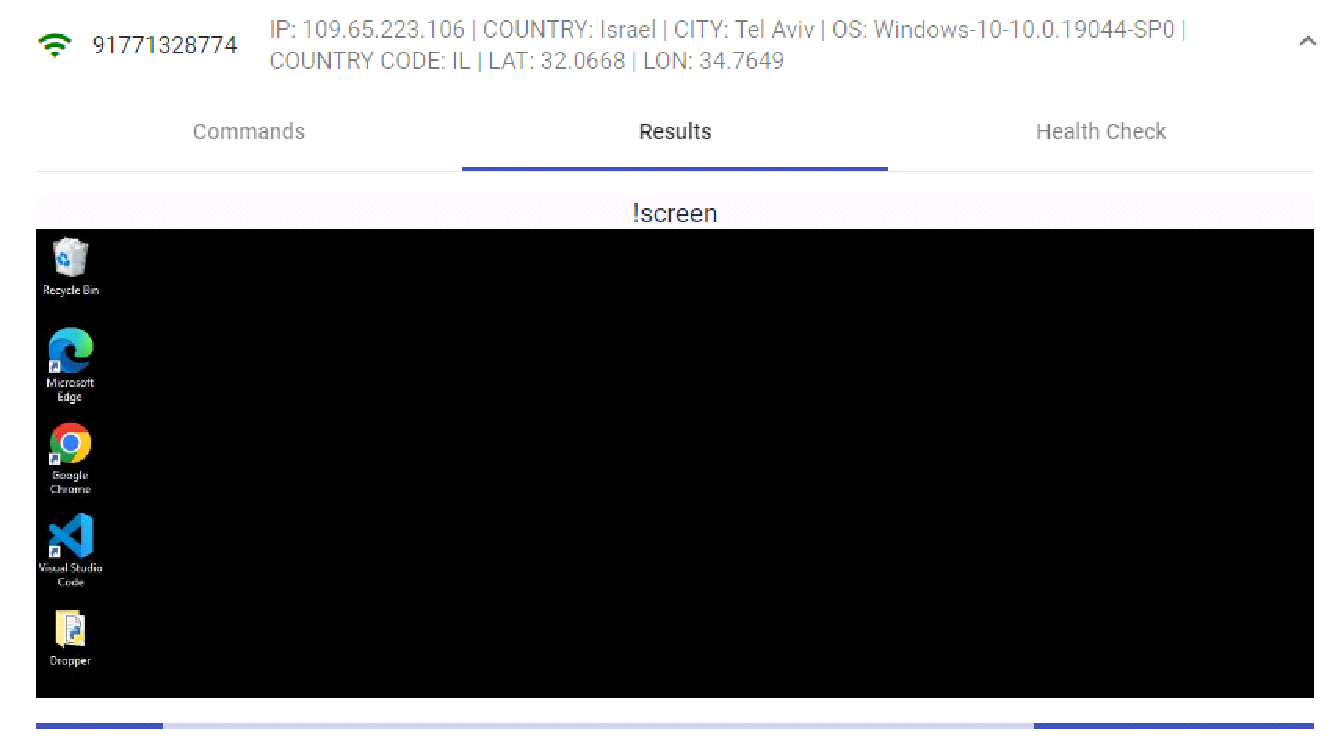
בנוסף לקח בהתאם ל – PINGים שהמחשב הנדבק מחזיר לשרת ה – TABים יראו אם הוא למעלה או למטה וגם ינעלו את ה - TAB בהתאם.

כל מחשב מקבל סט של שלושה חלקים Commands, Results, Health Check:

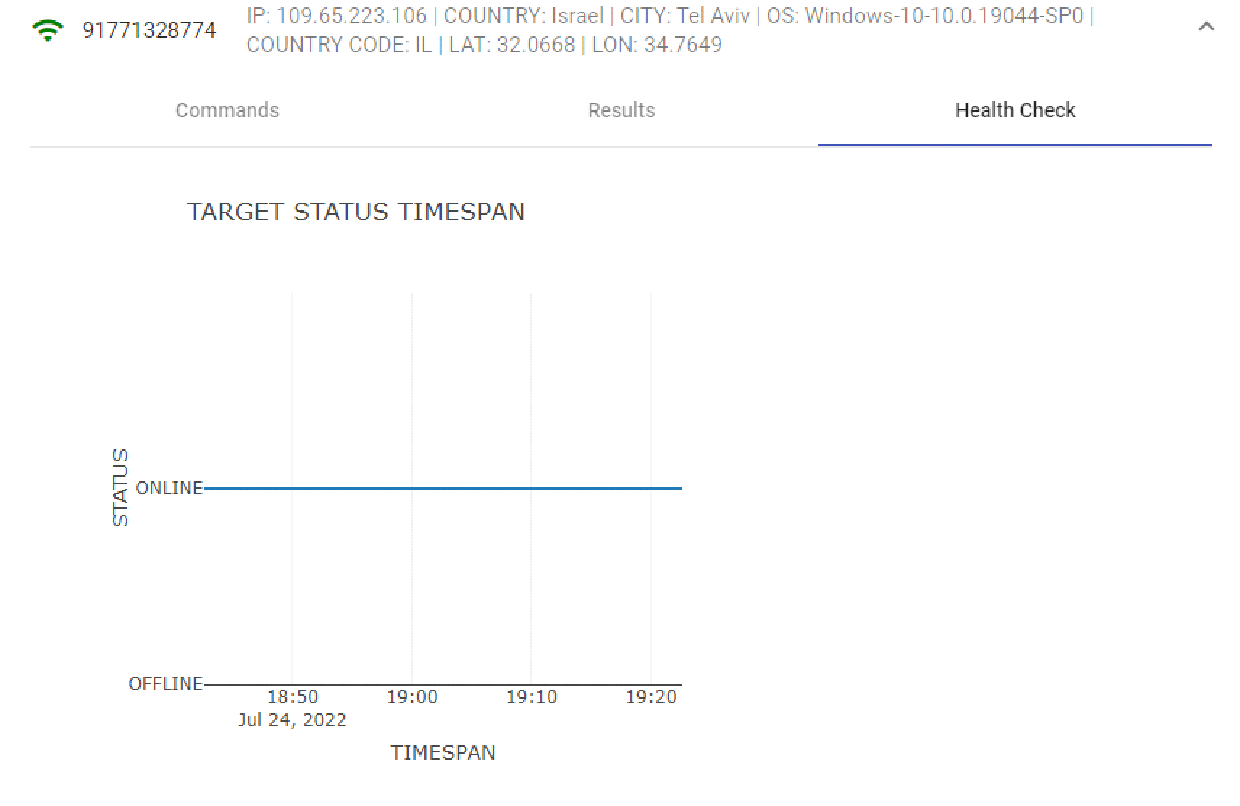


Commands בעצם זה החלק שנותן לתוקף לשלוח למחשב הנתקף פקודות בלחיצת עכבר, רק להכניס ערכים נכונים לתיבות הטקסט בעת הצורך ולחיצה על הכפתור.

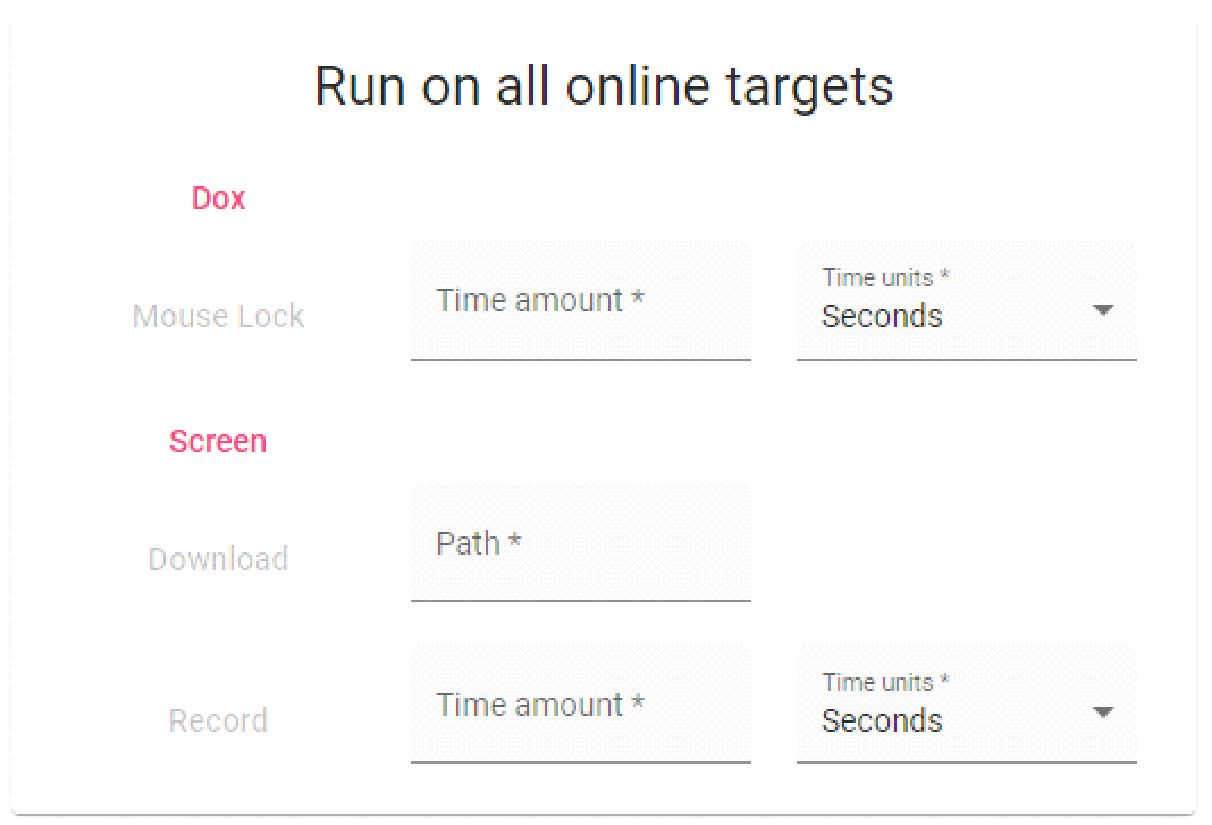
Results יראה לנו את כל התוצאות שחזרנו לנו ממחשב הנתקף לאחר ששלחנו אליו פקודות.



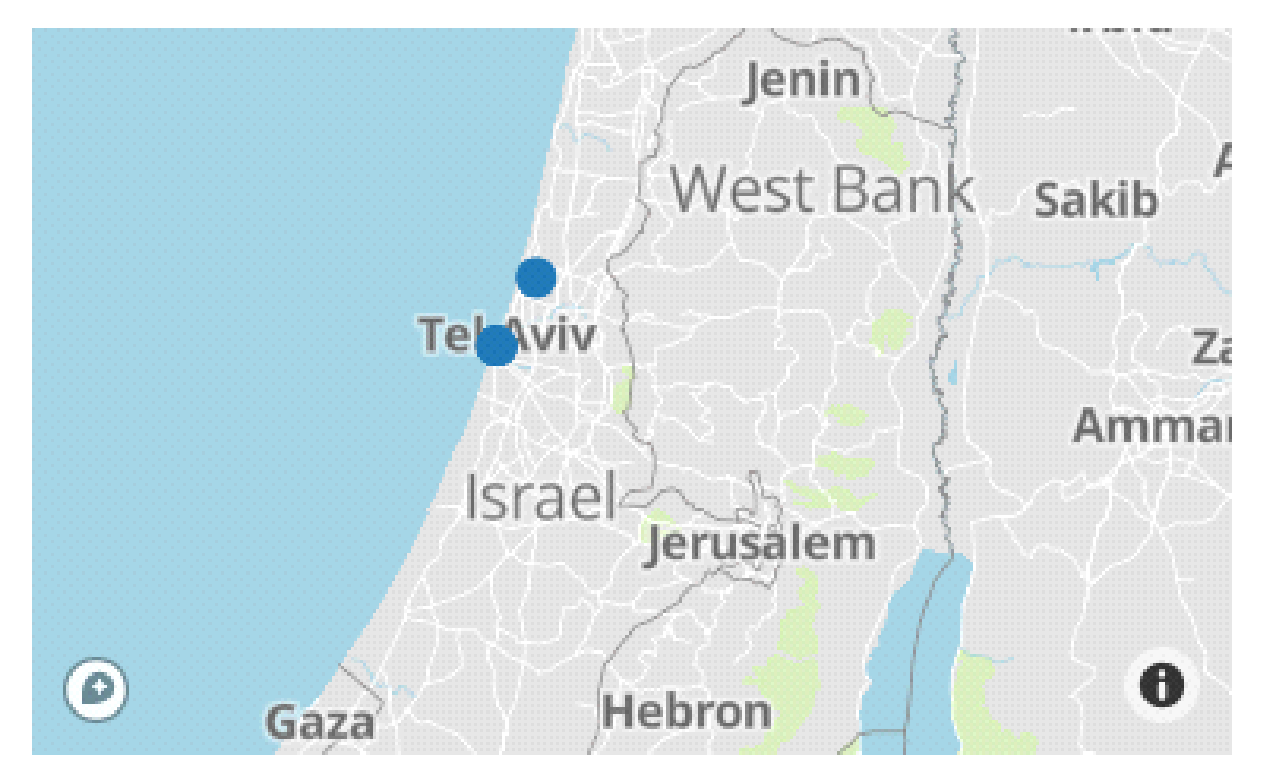
Health Check יראה לנו את מצב המחשב הנתקף בתקופה האחרונה, האם הוא למטה או למעלה.



בנוסף לזה יש לנו TAB צדדי אשר אפשר לפתוח ובו יש אותן פקודות אשר קיימות על כל מחשב בנפרד אלא שפה הפקודות ירוצו על כל המחשבים שכרגע למעלה:



ומתחת לזה נמצאת מפה שמראה לנו את המיקום העדכני של כל מחשב ומחשב, פה נעזרנו בטכנולוגיה של חברה בשם Plotlyjs אשר נותנת לנו שימוש ב – API המפות שלהם בעבור הרשמה לחשבון אישי:



# תהליך ההדבקה

מאחר והנוזקה נכתבה בפייתון, היא מורכבת ממספר סקריפטים, מה שהופך את תהליך ההפצה וההדבקה למסורבל ופחות אמין (אמין מעיני הנתקף)  
לכן, עלה צורך לחלק את תהליך ההדבקה לשלבים על ידי שימוש בdropper ייעודי  
ה-dropper הוא למעשה סקריפט פייתון רזה שמטרתו ברורה - הורדת הנוזקה במלואה אל נתיב אקראי, והרצה שלה.  


ה-dropper:  
1. מוצא את שם היוזר שהריץ את הdropper ויוצר נתיב אקראי תחת התיקיה שלו  
2. ניגש לשרת ייעודי ("שרת תקיפה") ומוריד ממנו את הנוזקה באמצעות בקשת HTTP  
3. מחלץ את קובץ הzip המכיל את קבצי הנוזקה  
4. מריץ את קובץ הmain של הנוזקה בפרוסס נפרד

# שרת התקיפה

שרת התקיפה הינו סקריפט פייתון מבוסס fastapi המאוחסן על גבי heroku  
תכליתו של שרת זה, לאחסן את קבצי הנוזקה באופן אנונימי (ה-domain מקושר למשתמש שנפתח באופן אנונימי) תוך כדי מתן גישה נוחה לקבצים המאחוסנים (נדרשת בקשת HTTP פשוטה בכדי להוריד את הקבצים הנמצאים על השרת)

השרת מגדיר 2 endpointים שאליהם נוכל לפנות על מנת לגשת לresourceים רלוונטים

1. קבצי ה-malware (אותם יוריד הdropper)
2. קובץ ה-dropper עצמו (נוכל לגשת לקובץ הdropper בקלות)

# פונקציות עזר

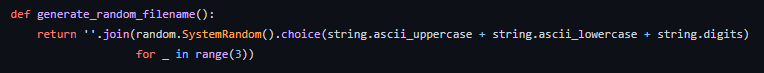
מציאת כתובת ה-ip של הנתקף באמצעות API חיצוני

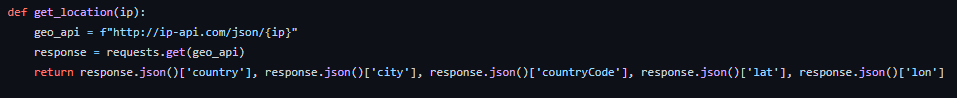


יצירת נתיב אקראי במערכת ההפעלה - רלוונטי בעת התקנת הנוזקה ויצירת עמידות

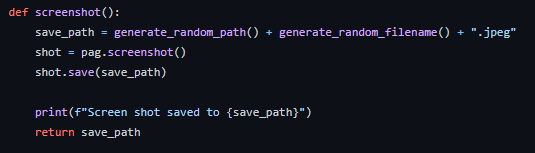


יצירת שם קובץ אקראי

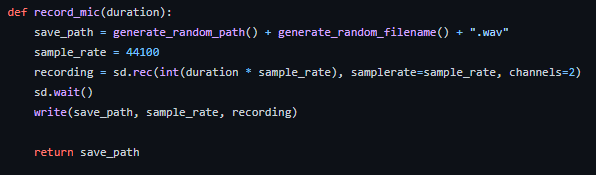


איתור הנתקף על ידי שימוש בAPI חיצוני

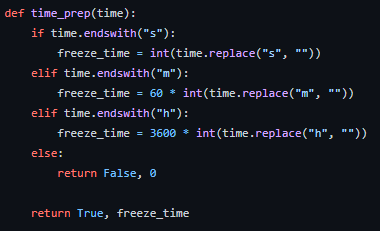
צילום המסך של הנתקף על ידי שימוש בpyautogui



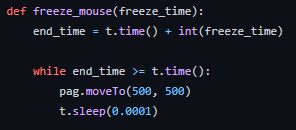
הקלטת המיקרופון של הנתקף



פירסור time input מהמשתמש לטובת פונקציות שמקבלות זמן כפרמטר (שניות, דקות, וכו)



הקפאת העכבר של הנתקף למשך הזמן המבוקש



# Discord API

ה-API של דיסקורד מקנה למפתחים כלים נוחים לפיתוח בוטים לפלטפורמת דיסקורד.  
אותם בוטים הם לכאורה משתמשים רגילים אשר מסוגלים להתממשק עם שרתי דיסקורד ולממש את הפונקציונליות שלהם בהתאם.

בוט מסוגל להאזין לeventים שונים המתרחשים בשרת אליו הוא מחובר, כגון שליחת הודעה חדשה, התחברות / התנתקות של משתמשים ועוד רבים נוספים.  
עובדה זו מאפשרת לנו ליצור מנגנון Command and Control המתבסס על דיסקורד **בלבד** בצורה הבאה:

1. לוגיקת הבוט ממומשת בסקריפט python אשר רץ על מחשבו של הנתקף
2. הבוט מתממשק אל שרת ייעודי (כעת הבוט מאזין להודעות הנשלחות בצ'אט של השרת)
3. בעת שליחת הודעה, הבוט יבדוק את תוכן ההודעה (on\_message)
4. במידה והטקסט שנשלח מסמל פקודה שהוא מכיר (בהתבסס על המימוש שלו) הבוט יפעל בהתאם, ובמקרה הצורך יחזיר את הפלט שיצר אל השרת (לדוגמא - הבוט זיהה הודעה המסמלת פקודה לצילום מסך, הבוט יצלם את המסך וישלח את התמונה לשרת)