

תיק פרויקט - Cyber Chaperone



מגיש: אופק טל

ת.ז: 216295873

מורה מנחה: לינה שמידט

בית ספר: תיכון בן גוריון, פתח תקווה

מקצוע: תכנון ותכנות מערכות במסלול הגנת סייבר

מועד הגשה: 02.06.2024

תוכן עניינים

2.....	תוכן עניינים
4.....	מבוא
4.....	ייזום המערכת
4.....	תיאור ראשוני של המערכת
4.....	הגדרת הלקוח
4.....	הגדרת יעדים ומטרות
4.....	בעיות תועלות וחסכונות
4.....	סקירת פתרונות קיימים
5.....	סקירת טכנולוגיית הפרויקט
5.....	תיחום הפרויקט
6.....	אפיון - פירוט תיאור המערכת
6.....	תיאור מפורט של המערכת:
6.....	פירוט היכולות של כל משתמש במערכת:
7.....	לוח זמנים
8.....	ניהול סיכונים
9.....	תיאור תחום הידע
9.....	יכולות צד שרת
10.....	יכולות צד קליינט
12.....	מבנה
12.....	תיאור ארכיטקטורת חומרה
12.....	תיאור הטכנולוגיה הרלוונטית
13.....	תרשים זרימה : server
14.....	תרשים זרימה: Client
15.....	תיאור האלגוריתמים מרכזיים בפרויקט
16.....	תיאור סביבת הפיתוח
17.....	תיאור פרוטוקול התקשורת
18.....	תיאור מסכי המערכת
18.....	תיאור מסכים - ממשק משתמש שרת
21.....	תיאור מסכים - צד לקוח מחשב קצה
22.....	תרשים מסכים
22.....	תיאור מבני הנתונים:
22.....	סקירת חולשות ואיומים:
23.....	מימוש הפרויקט
23.....	סקירת המודלים והמחלקות המרכיבות את המערכת
23.....	סקירת מודלים של פייטון
25.....	מודולים ומחלקות שאני פיתחתי לצורך הפרויקט
33.....	סקירת בעיות אלגוריתמיות וקטעי קוד מיוחדים
33.....	הבעיה האלגוריתמית: שליחת מידע המיוצר בשלוש thread-ים שונים
37.....	הבעיה האלגוריתמית: הצפנה

38.....	מסמך בדיקות מלא
40.....	מדריך למשתמש
40.....	תוכן :
41.....	התקנת תוכנת שרת :
42.....	התקנה במחשב קצה :
43.....	שימוש בתוכנת CyberChaperone
44.....	רשימת מחשבי קצה וניטור :
45.....	הסרת מחשב קצה מניטור
45.....	תקלות ופתרונות
46.....	עץ קבצים
47.....	רפלקציה
47.....	תיאור תהליך העבודה
47.....	הצלחות :
47.....	אתגרים :
47.....	קשיים ופתרונות :
48.....	תהליך הלמידה
48.....	כלים לעתיד
48.....	תובנות מהתהליך
48.....	במבט לאחור: מה הייתי עושה אחרת?
49.....	שיפורים פוטנציאליים עם משאבים נוספים
49.....	תודות
50.....	ביבליוגרפיה
51.....	נספחים
51.....	קוד צד שרת
51.....	UserInterface.py
72.....	serverToFile.py
74.....	Main.py
74.....	FileToGui.py
76.....	DataService.py
79.....	קוד צד קליינט
79.....	clientUserInterface.py
82.....	client.py

מבוא

ייזום המערכת

תיאור ראשוני של המערכת

- הפרויקט שלי הוא תוכנה המאפשרת למשתמש לצפות בפעילות של עובדים במחשבים שלהם על מנת לפקח על פעולותיהם ולוודא שהם עובדים כראוי בזמן אמת.
- בחרתי בפרויקט בחרתי בפרויקט כהמשך ושיפור לפרויקט קטן יותר, אך דומה שעשיתי בשנה שעברה. הפעם עם יותר ניסיון וידע. ידעתי שביכולתי לשפר את התוצר המוגמר בכמה רמות. אני צופה אתגרים רבים בתכנות ובעיצוב, שכן רמת התכנות תהייה גבוהה יותר והתוכנה תהייה בעלת אמצעים שאיני מכיר מפרויקט העבר.

הגדרת הלקוח

- התוכנה מיועדת להימכר לחברות גדולות בעלות עובדים רבים, התוכנה תשמש דרך לצפות בעבודת העובדים ובכך תמנע התבטלות סמויה ותוכל לעזור למצוא עובדים שאינם עובדים. התוכנה תצטרך להיות נוחה ומובנת לשימוש, ותאפשר למשתמש הסף להבין ולקבל את המידע באופן נוח ומובן.

הגדרת יעדים ומטרות

- לסיים עם פרויקט שאני מרוצה מהתוצר הסופי ובזמן.
- ליצור תוכנה שתאפשר להבין מה מתרחש בצד השני בקלות ובנוחות.
- לייצר תוכנה אשר הפעלתה הראשונית פשוטה ודורשת מעט ידע קודם על מחשבים.
- לייצר תוכנה אשר פועלת ברציפות ובאופן יציב.

בעיות תועלות וחסכונות

- הבעיה המרכזית שמטרת התוכנה לפתור היא את הקושי לפיקוח ומעקב אחרי עובדים בשעות העבודה בחברות גדולות בעידן המודרני- כיום על מנת לוודא שכולם עובדים בשעות העבודה יש להעסיק אחראים רבים שתפקידם לוודא שמתקיימת העבודה. התוכנה שלי תאפשר לאחראי אחד לראות את העיסוק של העובדים בקלות ובמהירות ובכך תאפשר לאחראי אחד לפקח על עובדים רבים.

סקירת פתרונות קיימים

- ישנם פתרונות דומים רבים לבעיה זו, וישנם חברות אשר מציעות שירותים דומים במרשתת, לדוגמת deskTime המאפשר למעסיק לראות את התקדמות העובדים בפרויקטים ואת פריון העובדים. בשונה מdeskTime, התוכנה שלי מאפשרת לראות עבור כל אדם בנפרד במה הוא עוסק בזמן אמת.

סקירת טכנולוגיית הפרויקט

- הפרויקט שלי משתמש במספר רב של טכנולוגיות קיימות ונכתב בפיטון:
 - התקשורת מנוהלת על ידי sockets ו Threading .
 - הוצאת המידע ממחשב המשתמש מנוהלת על ידי מודלי פייטון בשם: pyautogui להוצאת שם החלון הפעיל, scapy למציאת שאילות DNS ו- keyboard לשליפת מידע מהמקלדת.
 - ממשק המשתמש מנוהל על ידי מודלי פייטון: בשם customtkinter, התמונות בממשק המשתמש דרך PIL ואזהרות מערכת דרך CTkMessageBox, הממשק עצמו מורץ ב-Threads.
 - מאגר המידע מנוהל על ידי מודל פייטון בשם sqlite .

תיחום הפרויקט

- חלק ניכר מן הפרויקט שלי הינו העיצוב וממשק המשתמש.
- העברת המידע ותקשורת הינו חלק עיקרי נוסף בפרויקט.
- ולבסוף, שליפת מידע מן המשתמש (keylogging) הינו גם הוא חלק ניכר בפרויקט.

אפיון - פירוט תיאור המערכת

תיאור מפורט של המערכת:

הפרויקט שבחרתי יכולת תוכנה שרת וקליינט, מטרתה של התוכנה היא להתחבר למחשבים רבים בו זמנית ולאפשר לאדם המשתמש בתוכנה לראות את השימוש של במחשבים בזמן אמת. התוכנה פועלת כמעין keylogger ברשות משתמש הסף. למערכת עוד מספר פונקציות שאפרט בהמשך.

פירוט היכולות של כל משתמש במערכת:

צד השרת

- תצוגה גרפית ראשית עבור הלקוח המנהל ובה מספר פעולות:
 - מסך התחברות: התחברות על פי שם וסיסמה.
 - מסך רישום : רישום לפי שם וסיסמה.
- לאחר התחברות:
 - מסך משתמש המציג שם וכתובת IP של המחשב בו המשתמש מחובר.
 - מסך הוספת משתמשים לפיקוח.
 - סרגל צד גלילי בעל כפתורים עם שמות המשתמשים לפיקוח..
- בעת לחיצה על כפתור משתמש
 - חלון משתמש המציג את פעולותיו של המשתמש שנבחר .
- הפעולות כוללות:
 - לחיצות המקשים במקלדת של המשתמש.
 - החלון שבו משתמש המשתמש.
 - בקשות הDNS של מחשב המשתמש.
- שרת אשר פועל באופן עצמאי ובלי תלות בתצוגה הגרפית ששומר את המידע לקבצי txt

צד הקליינט

- תצוגה גרפית פשוטה המאפשרת מספר דברים:
 - לשנות את כתובת ה-IP של השרת אשר אליה הקליינט ינסה להתחבר.
 - לראות את כתובת ה-IP של הקליינט שאותה צריך להוסיף כאשר יוצרים משתמש ב"מסך הוספת משתמשים לפיקוח" בצד השרת..
- קליינט שאוסף ושולח מידע באופן רציף ועצמאי מהתצוגה הגרפית.

הבדיקות (קופסה שחורה):

- בדיקה של התחברות של כמה לקוחות במקביל.
- בדיקה שהגרפיקה עובדת.
- בדיקה של אסיפת המידע מן המשתמש.
- בדיקה שהלקוחות לא קורסים כשהשרת נסגר .
- בדיקה שהתקשורת עובדת בין כמה מחשבים ולא רק כמה לקוחות באותו המחשב.
- בדיקה תקינות של הצפנה ופענוח המידע.

לוח זמנים

פעילות	זמן התחלה מתוכנן	זמן סיום מתוכנן	זמן התחלה בפועל	זמן סיום בפועל	הערות
בדיקת המודלים	20.12.2023	1.1.2024	20.12.2023	10.1.2024	התנסות כוזבת עם pyqt5 גגרה את הזמן
תיק פרויקט	2.2.2024	20.5.2024	5.4.2024	3.6.2024	
עיצוב בסיס ב figma וקוד GUI בסיסי	20.12.2023	1.2.2024	20.12.2023	1.3.2024	
קוד קליינט ושרת בסיסי	1.2.2024	1.3.2024	15.1.2024	10.3.2024	
בניית database משתמשים	1.4.2024	10.5.2024	1.4.2024	12.5.2024	
גימור קוד GUI ועיצוב	10.5.2024	20.5.2024	5.5.2024	3.6.2024	לא סיימתי גימורי קוד GUI והעיצוב לא מוגמר עדיין
גימור קוד שרת וקליינט	1.5.2024	20.5.2024	1.5.2024	1.6.2024	בעיה קטנה מאוד שלקחה המון זמן למצוא :)
סיום פרויקט, גימורים אחרונים	20.5.2024	30.5.2024	27.5.2024	3.6.2024	עדיין לא סיימתי, חסר מעבר pep8, docstrings, יפוי הקוד ועוד. השאר יעשה לאחר הגשת תיק הפרויקט

ניהול סיכונים

סיכון	פתרון
אי עמידה בזמנים	התחלת עבודה מוקדמת ומציאת פתרונות לבעיות אלגוריתמיות מרכזיות תוך שילוב של פיתוח האפליקציה
מודל המרכזי לממשק המשתמש (customtkinter) אינו מתאים לשימוש עם Threading ולכן יש למצוא פתרון ובמהרה, על מנת למנוע את הצורך בהחלפת מודל ממשק	התנסות עם מודלים אחרים, התנסות עם הממשק וחיפוש פתרון.
שימוש מרובה ב Threading עלול לגרום להאטת המחשב	כתיבת קוד מהיר שלא דורש משאבים רבים מהמחשב

תיאור תחום הידע

יכולות צד שרת

- שם היכולת: טיפול בבקשות הירשמות וכניסה למערכת
 - מהות היכולת: קבלת בקשות login | register-מלקוחות, בדיקה האם ניתן לבצע את הבקשה, החזרת תשובה ללקוח (האם ההרשמה/התחברות פעלה בהצלחה או שלא) וטיפול המשך בלקוח. המערכת תבדוק בבסיס הנתונים שלה כדי לבדוק האם יש שם משתמש וסיסמה התואמים לסיסמה והשם משתמש שהוזנו בעבור התחברות ואם קיים הירשמות למערכת אז ישמור במס"ד הנתונים את הנתונים של המשתמש החדש.
 - אוסף יכולות נדרשות:
 - התאמה של משתמשים מול מאגר נתונים קיים
 - הוספת מידע למאגר נתונים
 - החזרת תשובות ללקוחות
 - אובייקטים נחוצים: בסיס נתונים
- שם היכולת: הוספה ומחיקה של משתמש לפיקוח
 - מהות היכולת: הוספת משתמש חדש אשר ממנו מקבלים מידע, שמירת המידע, הצגת המידע ועדכון דינמי של הממשק משתמש ושל מאגר הנתונים
 - אוסף יכולות נדרשות:
 - ממשק גרפי
 - התחברות
 - thread-ים
 - הוספה לבסיס נתונים
 - sockets
 - אובייקטים נחוצים: בסיס נתונים, ממשק משתמש, שרת, threading.
- שם היכולת: שמירת מידע לקבצים נפרדים
 - מהות היכולת: שמירת המידע המתקבל מהקליינט-לשרת לקבצים נפרדים, על מנת לאפשר עדכון ממשקה משתמש לאחר קבלת המידע, ושמירת היסטוריה של המידע המתקבל
 - אוסף יכולות נדרשות:
 - thread-ים
 - השוואה עם בסיס נתונים
 - יצירת קובץ txt
 - כתיבה לקובץ txt
 - אובייקטים נחוצים: בסיס נתונים, threading, כתיבה לקבצים

- שם היכולת: הצגה של מידע של משתמשים לפיקוח בזמן אמת
 - מהות היכולת: הצגה נוחה ומיידית של המשתמשים והפעולות שהם מבצעים במחשב בזמן אמת
 - אוסף יכולות נדרשות:
 - thread-ים
 - עדכון ממשק גרפי מ-thread חיצוני
 - קריאה של קבצי txt
 - אובייקטים נחוצים: threading, ממשק משתמש, קריאת קבצים

יכולות צד קליינט

- שם היכולת: איסוף מידע מהלקוח
 - מהות היכולת: איסוף מידע מהלקוח על מנת לשלוח אותו לשרת, המידע צריך לספק תמונה כללית של מה הלקוח עושה במחשב, המידע כולל את הקלדות המקלדת של הלקוח, את שם האפליקציה עליו הלקוח מסתכל/משתמש, ואת בקשות ה DNS של מחשב הלקוח.
 - אוסף יכולות נדרשות:
 - thread-ים
 - האזנה ללחיצות על המקלדת
 - האזנה לפורט 53 udp
 - האזנה לאפליקציה הממוקמת
 - אובייקטים נחוצים: sniff לתקשורת, האזנה למקלדת, האזנה למערכת.
- שם היכולת: הצפנת המידע לפני השליחה
 - מהות היכולת: שמירת על מידע חשוב בזמן מעברו ברשת על מנת למנוע הדלפות ופריצות
 - אוסף יכולות נדרשות:
 - הצפנה

■ אובייקטים נחוצים: cryptography.fernet מודל

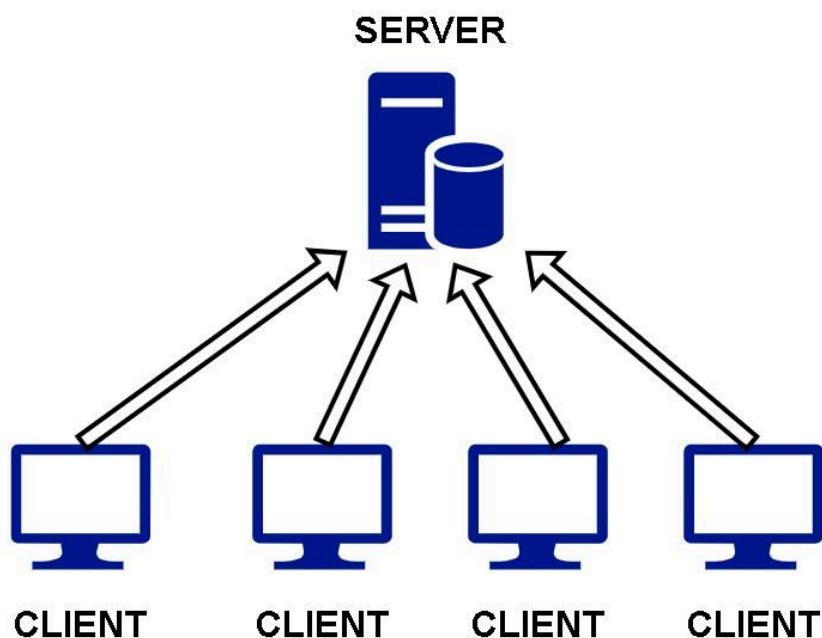
- שם היכולת: שליחת מידע לשרת
 - מהות היכולת: שליחת המידע היא המהות המרכזית של הקליינט
 - אוסף יכולות נדרשות:
 - תקשורת
 - thread-ים
 - אובייקטים נחוצים: threading, sockets

שם היכולת: שינוי כתובת השרת

- מהות היכולת:הכנה ראשונית קלה והתאמה לשינויים עתידיים בקלות.
- אוסף יכולות נדרשות:
 - ממשק משתמש
 - שמירת מידע
 - כתיבה לקבצי txt
- אובייקטים נחוצים: ממשק משתמש, קריאת וכתיבת קבצים

מבנה

תיאור ארכיטקטורת חומרה

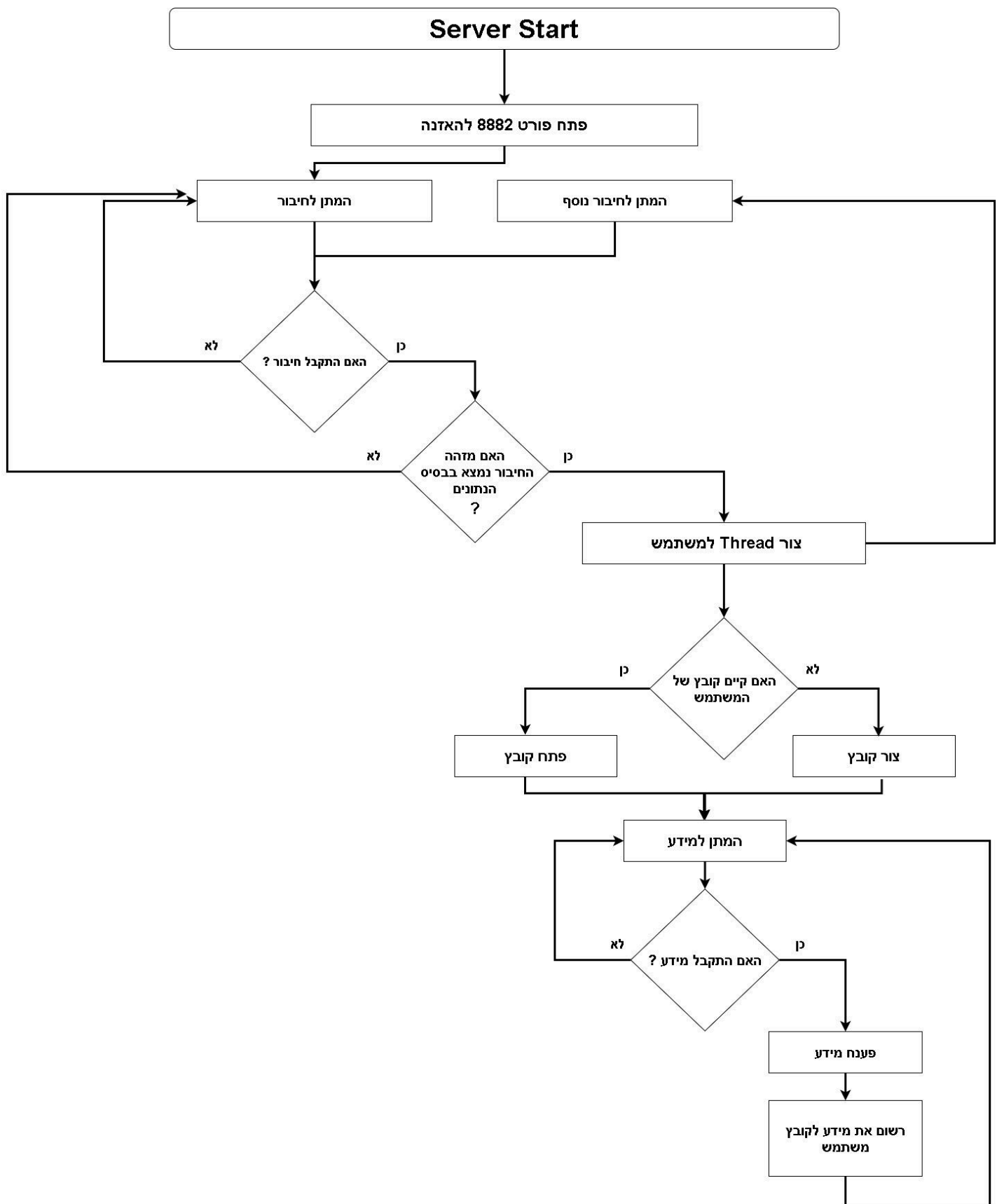


*אל השרת מתחברים לקוחות בכמות בלתי מוגבלת, התרשים הנ"ל מהווה דוגמא למבנה בעל 4 חיבורים.

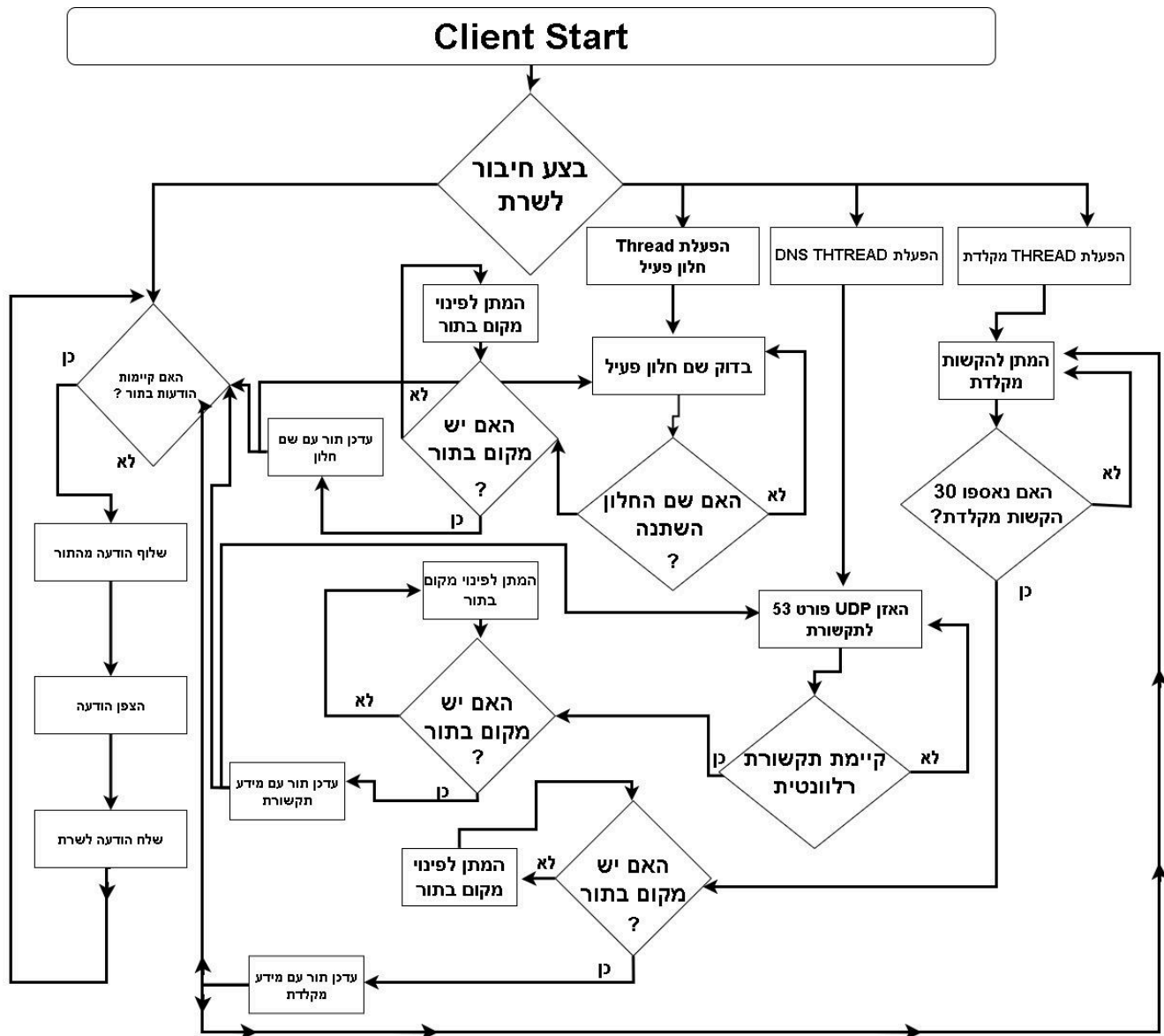
תיאור הטכנולוגיה הרלוונטית

- שפת תכנות: שפת התכנות בה הפרויקט פועל היא python (גרסה 3.10 ומעלה בלבד)
- מערכת הפעלה: windows 10
- תקשורת: UDP באמצעות מודל socket
- תחומי עניין: הצפנה, תקשורת, בסיס נתונים ו sql , ממשק משתמש, עיצוב וגרפיקה, sniff , keylogging , שמירת מידע.

הקוד נבדק ונכתב על מחשבי windows 10 ולכן שימוש במערכת הפעלה אחרת יכולה לגרום לתקלות שלא נצפו ולכן לא מומלצת.
הרצת הקוד דורשת python בגרסה 3.10 ומעלה בלבד ומודולים נוספים שיפורטו בהמשך.

תרשים זרימה : server

תרשים זרימה: Client



תיאור האלגוריתמים מרכזיים בפרויקט

- הבעיה האלגוריתמית: הצפנה
 - אפשריות לפתרון:
 - הצפנה א-סימטרית
 - הצפנה סימטרית
 - הפתרון הנבחר: בחרתי בהצפנה סימטרית בעזרת מודל cryptography.fernet. אני משתמש כרגע במפתח הצפנה קבוע.
 - הבעיה האלגוריתמית: תקשורת
 - אפשריות לפתרון:
 - שימוש ב scapy ליצירת פקטות ושליחתן
 - שימוש ב socket להעברת מידע
 - הפתרון הנבחר: השימוש ב sockets נוח ומאפשר יצירת תקשורת בצורה פשוטה ובטוחה ללא צורך ביצירת הפקטות מ0.
 - הבעיה האלגוריתמית: הקשבה להקשות המקלדת של הקליינט
 - אפשריות לפתרון:
 - שימוש במודל keyboard תוך יצירת מערך האזנה
 - שימוש במודל pynput ולהשתמש ב keyboard.Listener
 - הפתרון הנבחר: בחרתי ליצור מערך האזנה משלי תוך שימוש במודל keyboard בעזרת שימוש ב keyboard.read_event ובדיקה שה read_event שווה ללחיצה על המקלדת בעזרת keyboard.KEY_DOWN
 - הבעיה האלגוריתמית: מציאת האפליקציה הממוקדת
 - אפשריות לפתרון:
 - שימוש ב pyautogui
 - שימוש ב win32gui
 - הפתרון הנבחר: בחרתי להשתמש ב pyautogui תוך שימוש בפונקציה pyautogui.getActiveWindowTitle פשוט מהיר ואפקטיבי ולכן בחרתי להשתמש בו.
 - הבעיה האלגוריתמית: הסנפה של שאילות DNS
 - אפשריות לפתרון:
 - שימוש ב scapy
 - הפתרון הנבחר: למרות ש scapy הוא האופציה היחידה המוכרת לי, היא אינה אופציה רעה כלל-השתמשתי בפונקציות המובנות של scapy על מנת להסניף תוך שימוש בפילטר שמגביל אות לתחום של שאילות DNS (תקשורת מסוג udp בפורט 53)
- sniff(filter="udp and port 53", count=5)

- הבעיה האלגוריתמית: שליחת מידע המיוצר בשלוש thread-ים שונים
 - אפשריות לפתרון:
 - שליחת המידע לבאפר חיצוני
 - שליחת המידע מתוך כל thread בנפרד
 - שימוש ב producer consumer threads
 - הפתרון הנבחר: בחרתי ליישם producer consumer threads מכיון שבתחילת השנה למדנו והתנסנו ביצירתם במהלך שיעורי סייבר, כאשר חיפשתי פתרון לבעיה מזכרתי בכך והצלחתי ליישם אותם בקלות ומהירות יחסית בעקבות הניסיון שכבר היה לי.

תיאור סביבת הפיתוח

- שפת תכנות: שפת התכנות בה הפרויקט פועל היא python (גרסה 3.10 ומעלה בלבד)
- מערכת הפעלה: windows 10
- תקשורת: UDP באמצעות מודל socket
- תחומי עניין: הצפנה, תקשורת, בסיס נתונים ו sql , ממשק משתמש, עיצוב וגרפיקה, sniff , keylogging , שמירת מידע.

הקוד נבדק ונכתב על מחשבי windows 10 בלבד ולכן שימוש במערכת הפעלה אחרת יכולה לגרום לתקלות שלא נצפו ולכן לא מומלצת.

הרצת הקוד דורשת python בגרסה 3.10 ומעלה בלבד והמודלים הבאים:

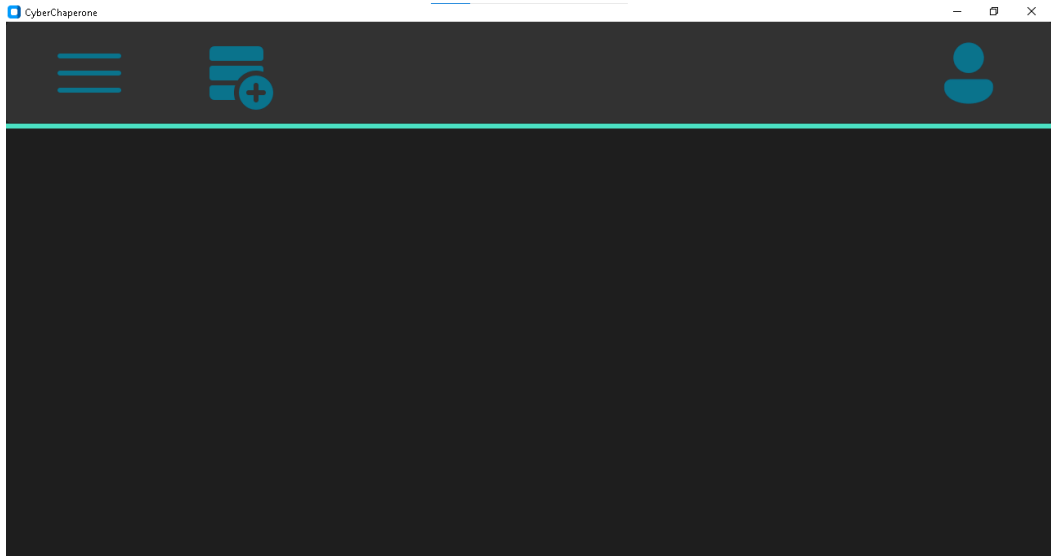
- customtkinter
- CTkMessageBox
- PILLOW
- sqlite3
- socket
- scapy
- pyautogui
- keyboard
- cryptography

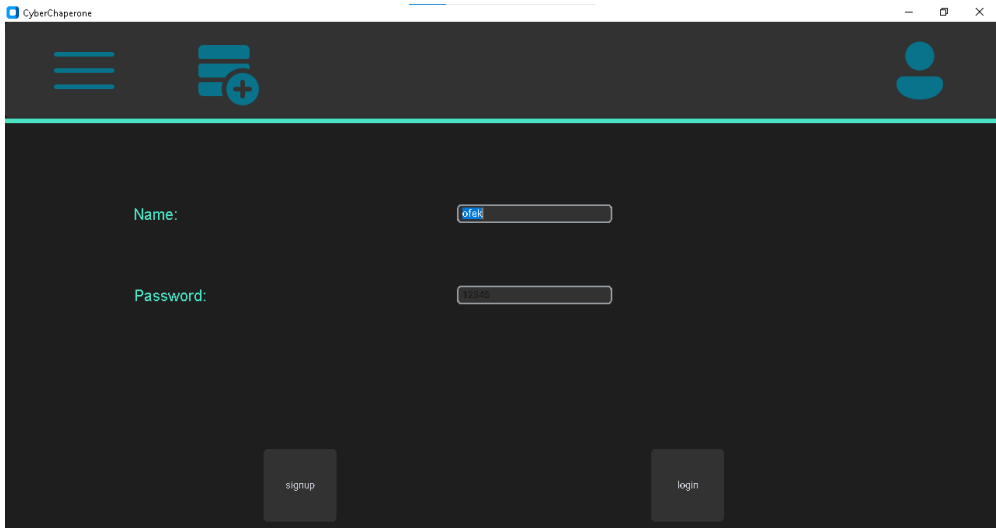
תיאור פרוטוקול התקשורת

תוכנת CyberChaperone מבוססת תקשורת בין הלקוחות לשרת. לשם כך יצרתי פרוטוקול תקשורת מבוסס חבילת פרוטוקולי התקשורת הנפוצה ביותר TCP/IP המהווה את הבסיס לתקשורת ברשת האינטרנט וגם ברשתות פרטיות. אני בחרתי ליישם את הפרוטוקול המאפשר העברת מידע בין לקוח לשרת באמצעות חבילות נתונים. העברת המידע מתבצעת באמצעות פרוטוקול UDP - ראשי תיבות של USER DATAGRAM PROTOCOL. השימוש בחבילות מסוג זה נבחר בגלל מהירות השליחה של המידע - המהירות גבוהה מזו של פרוטוקול TCP. המהירות בצד השרת הופכת חשובה כאשר השרת מטפל במספר רב של תחנות קצה. במוסף ל-UDP יש מבנה חבילה קטן יחסית ומכיל רק פורט מקור ופורט יעד, את הנתונים עצמם וסיביות סיכום ביקורת. את הנתונים בתוך החבילות אשר עוברות מהלקוח לשרת בלבד אני מסדר באופן שיהיה קל לפענח. כך למשל בשליחת נתונים על שאליות DNS הסדר הנתונים הוא תאריך ושעה של השאלית ואחריו את כתובת השם המלא. בדיקות תקינות ועומסים של השרת מאשרת את היכולת של השרת לקבל כמות גדולה של חבילות מידע ולעבד אותם לפי הצורך.

תיאור מסכי המערכת

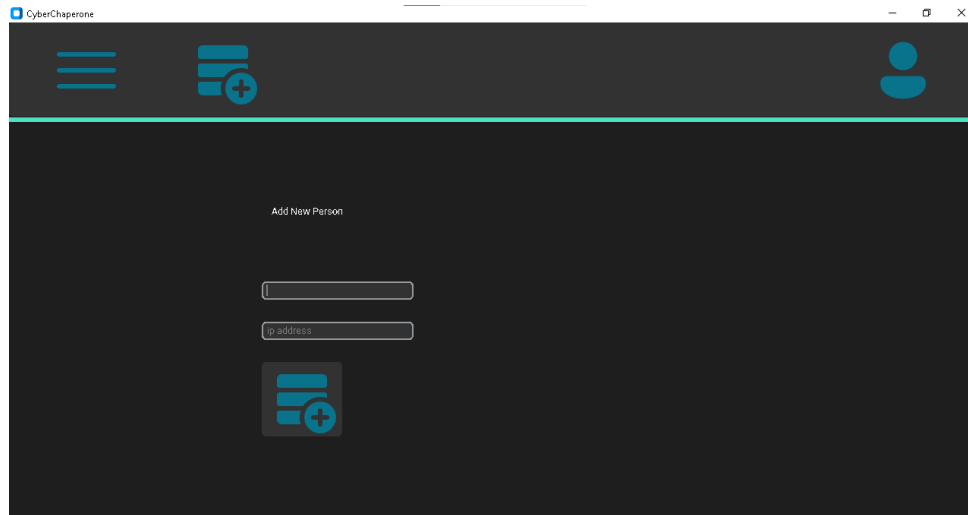
תיאור מסכים - ממשק משתמש שרת

שם מסך : מסך כניסה
תיאור מסך : מסך המשמש כניסה לתוכנת CyberChaperone


שם מסך : מסך כניסה
תיאור מסך : משמש שער אבטחה לכניסה לתוכנה, מאמת את שם המשתמש והסיסמה לפני פתיחת ממשק המשתמש


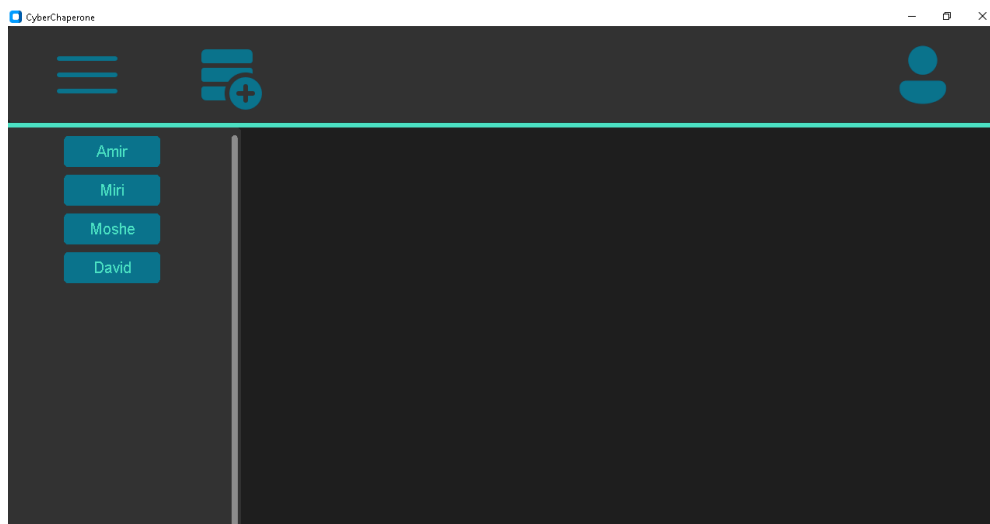
שם מסך : הוספת מחשב קצה לניטור

תיאור מסך : ניתן להוסיף נתוני מחשב קצה לניטור. במסך יקבל נתוני שם המחשב וכתובת IP של המחשב.



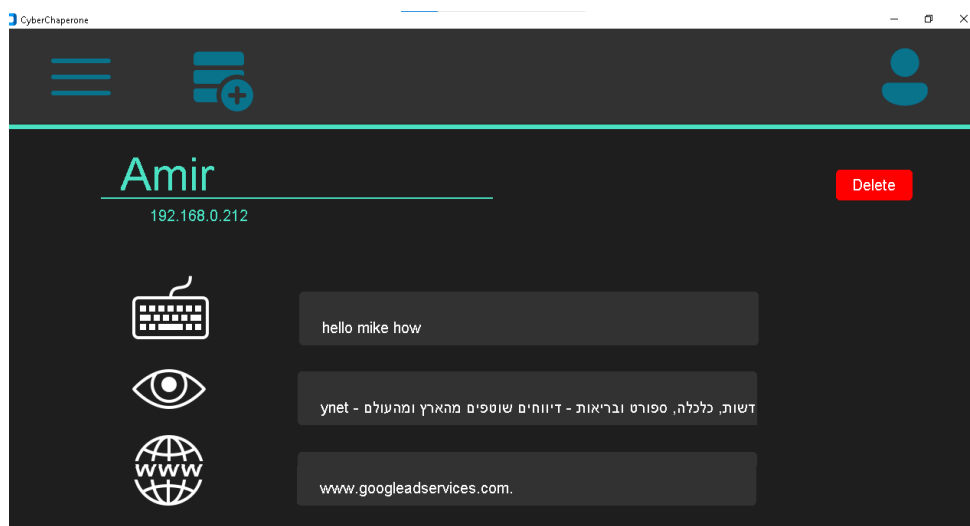
שם מסך : רשימת מחשבי קצה מנוטרים

תיאור מסך : מסך זה מציג את רשימת המחשבים אשר שולחים מידע לניטור. ניתן לגלול ברשימה ולבחור מחשב שאנו רוצים לנטר בזמן אמת.



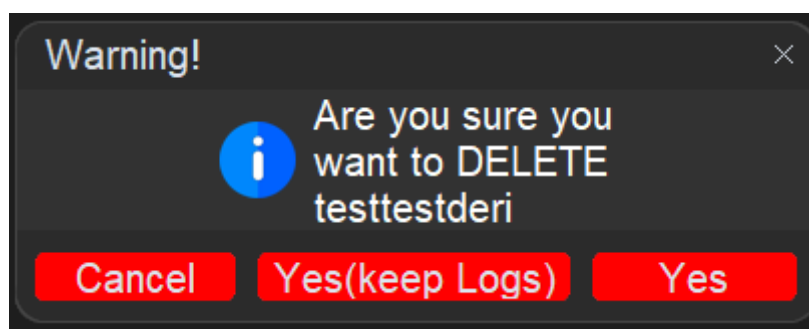
שם מסך : ניטור מחשב קצה

תיאור מסך : מסך זה מציג ניטור של מחשב קצה כולל החלון הפעיל במחשב, הקלדות מקשים ופנייה של המחשב לשרתים באינטרנט באמצעות DNS. במסך זה ניתן למחוק את המחשב מרשימת הניטור.



שם מסך : הודעת אזהרה על מחיקת משתמש

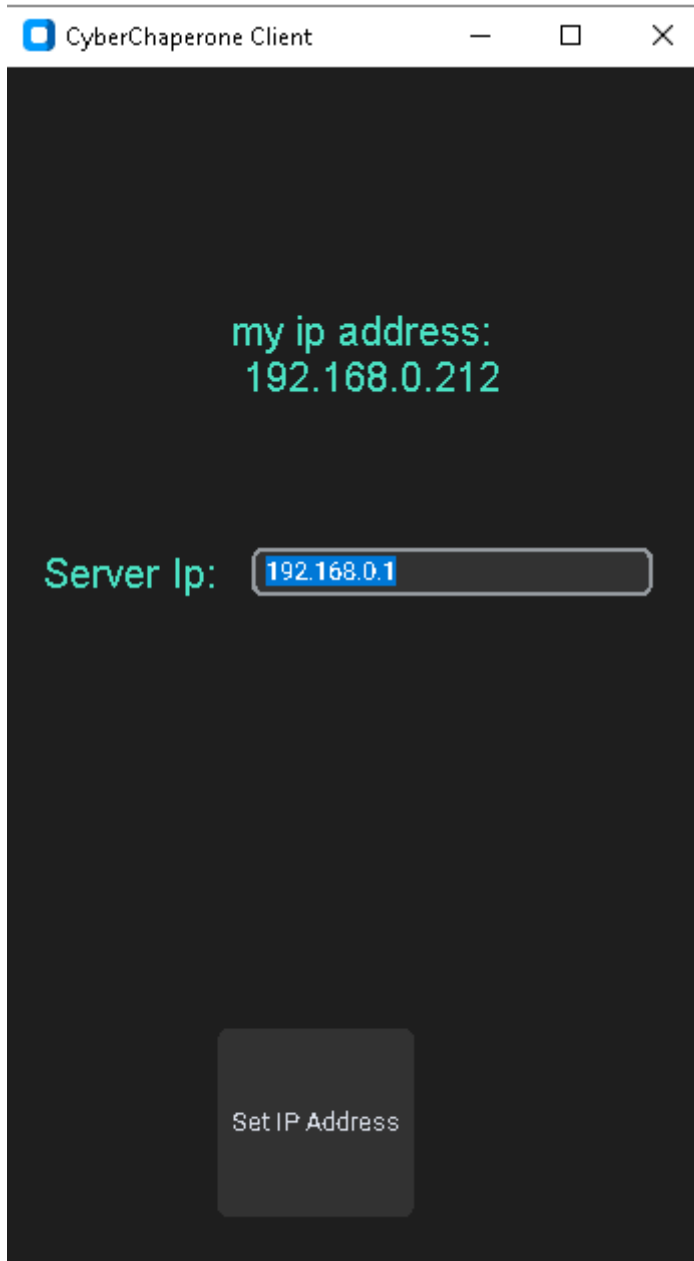
תיאור מסך : מסך אזהרה לפני מחיקת משתמש, בעל 3 אופציות- ביטול המחיקה, מחיקת המשתמש אך שמירת קובץ log של המשתמש ומחיקה מלאה של המשתמש ושל קבצי ה .log



תיאור מסכים - צד לקוח מחשב קצה

שם מסך : הזנה / שינוי כתובת שרת

תיאור מסך : מסך זה משמש להזנת או שינוי כתובת ה-IP של השרת. המסך יציג את כתובת ה-IP מחשב הקצה על מנת לעזור למתקין בתהליך הוספת מחשבי הקצה בשרת.



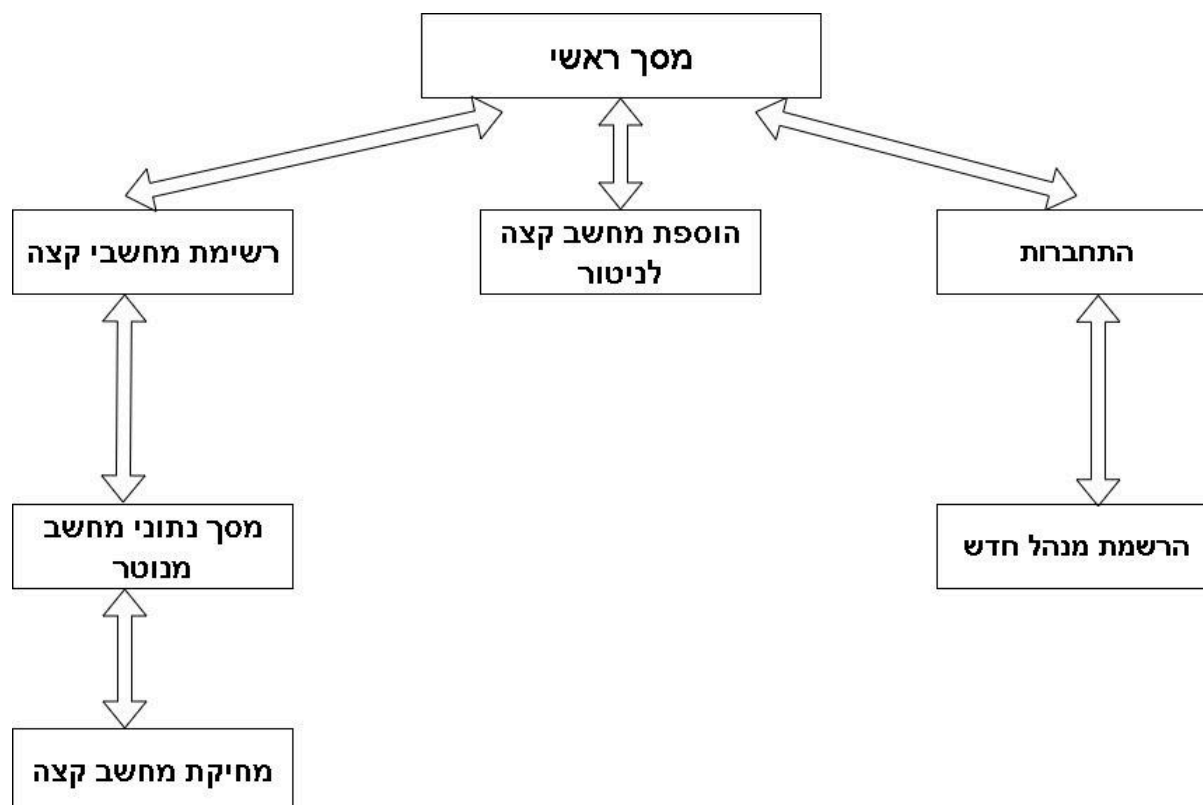
The screenshot shows a window titled "CyberChaperone Client" with a dark background. It displays the following text:

my ip address:
192.168.0.212

Server Ip:

At the bottom, there is a button labeled "Set IP Address".

תרשים מסכים



תיאור מבני הנתונים:

מאגרי נתונים מקומיים שרת:

- DataBase.db מאגר נתונים sql מקומי המכיל את השמות והסיסמאות של המשתמשים, בנוסף הוא מכיל את כתובות ה-IP והשם של הקליינטים שאליהם צריך להתחבר.
- תיקייה בשם "files" שמכילה קבצי txt המכילים את ההודעות שנשלחו מקליינט. לכל קליינט (IP נפרד) קובץ txt נפרד.

מאגרי נתונים מקומיים לקוח:

- קובץ txt בשם "settings" שמכיל את ה-IP של השרת אליו הקליינט מנסה להתחבר.

סקירת חולשות ואיומים:

- קיימת אפשרות פריצה ל-database של שמות המשתמשים והסיסמאות שלהם, אך נדרשת גישה ישירה למחשב שכן ה-databases מקומי ולא מוציא מידע לשרת ולא מקבל מידע מהשרת.
- קיימת אפשרות של פריצה לקבצי ה-txt אשר מכילים את ה-logs אך נדרשת גישה מלאה למחשב.
- קיימת אפשרות של הסנפה של ההודעות מהלקוחות לשרת, השתמשתי בהצפנה סימטרית על מנת למנוע מידע מלדלוף.

מימוש הפרויקט

סקירת המודלים והמחלקות המרכיבות את המערכת

סקירת מודלים של פייטון

שם המודל	איפה הוא בא לידי ביטוי	תפקיד
CTkMessagebox	UserInterface, clientUserInterface	הוספה של חלון אזהרת מערכת שתואם לממשק המשתמש customtkinter
from PIL import Image	UserInterface,	פתיחת תמונת והוספתם לממשק המשתמש
sqlite3	DataService	גישה ושינוי של מאגר נתונים תוך שימוש בקוד sql
socket	UserInterface,serverTo File, client	משמש ליצירת תקשורת UDP בין השרת ללקוחות המחוברים אליו.
scapy	client	שומש על מנת להסניף את בקשות הDNS
pyautogui	client	גישה למערכת על מנת לקבל את שם המסך הממוקד
keyboard	client	דרך להשיג את הקשות המקלדת
cryptography	client, serverToFile	העברה של הודעות דרך socket מוצפנות בצורה סימטרית(מפתח זהה להצפנה ופענוח)
os	UserInterface, serverToFile, FileToGui, DataService, clientUserInterface, client	משמש כדרך לגשת לקבצים במערכת.
customtkinter	UserInterface, clientUserInterface	יצירת ממשק גרפי

משמש למימוש כמה פעולות במקביל, לדוגמא קבלת מידע מכמה קליינטים בו זמנית	serverToFile, FileToGui, client	Threading
השגה של הזמן העכשווי על מנת להוסיף אותו להודעה.	FileToGui, client	time
דרך פשוטה לבדוק האם קלט מסוים הוא IP תקף	clientUserInterface	ipaddress

מודולים ומחלקות שאני פיתחתי לצורך הפרויקט

מחלקה: App

קובץ: UserInterface.py

המחלקה הראשית של האפליקציה, אחראית על פעולת ממשק המשתמש ועל שאר המחלקות המפעילות את ממשק המשתמש

משתנים במחלקה		
שם משתנה	תפקיד המשתנה	סוג הערכים שהמשתנה מקבל
self.db	מכיל את האובייקט שתפקידו לתקשר עם מאגר הנתונים	מחלקה בשם DataService
self.admin_user	מכיל array של מידע על המשתמש שהתחבר	array\None
self.monitor_list	מערך של כל משתמשי הקצה שיכולים להתחבר, משתנה על פי מי שמחובר	array
self.person_page_List	מערך של דפי משתמשי הקצה	array
self.thread_list	שמירה של מערך של Threads שפועלים ברקע.	array
self.back_panel	אובייקט של ממשק גרפי שמצורף לאפליקציה	מחלקה בשם BackPanel
self.signup_page	אובייקט של ממשק גרפי שמצורף לאפליקציה	מחלקה בשם SignUpPage
self.login_page	אובייקט של ממשק גרפי שמצורף לאפליקציה	מחלקה בשם LoginPage
self.add_person_panel	אובייקט של ממשק גרפי שמצורף לאפליקציה	מחלקה בשם AddPersonPage
self.side_panel	אובייקט של ממשק גרפי שמצורף לאפליקציה	מחלקה בשם SlidePanel
self.top_panel	אובייקט של ממשק גרפי שמצורף לאפליקציה	מחלקה בשם TopPanel
self.add_person_page_visibility	משתנה בוליאני שבודק האם מסך ה"add person page" ויזואלי כרגע	True\False

פעולות במחלקה		
שם פעולה	טענת כניסה	טענת יציאה
__init__	הפעולה לא מקבלת כלום	המחלקה יוצרת את ממשק המשתמש ויוצאת מבלי להחזיר.
login_func	הפעולה מקבלת שם משתמש וסיסמא	הפעולה בודקת את שם המשתמש והסיסמא בהשוואה למאגר הנתונים ומחברת את המשתמש
add_person	הפעולה מקבלת שם וIP	הפעולה מכניסה את השם והIP למאגר הנתונים יוצרת דף משתמש וכפתור משתמש ומתחילה Thread לעדכון דף המשתמש
destroy_monitor	הפעולה מקבלת משתמש סף בצורת array	לאחר אזהרה מוחקת את המשתמש ועוצרת את הthread המשוך למשתמש
place_person_page	אין	הפעולה מסתירה או מראה את חלון הוספת המשתמש
person_page	הפעולה מקבלת משתמש סף בצורת array	הפעולה מסתירה או מראה את חלון משתמש הסף
logout	אין	הפעולה מנתקת את המשתמש מהאפליקציה ועוצרת את כל הthreads הפתוחים
signup_func	הפעולה מקבלת שם, סיסמא, וסיסמא נוספת	הפעולה בודקת האם הסיסמא והסיסמא הנוספת שוות, אם לא מחזירה False אחרת היא מוסיפה את השם משתמש והסיסמא למאגר הנתונים ומחזירה True אם ההוספה הצליחה אחרת False

מחלקה: serverThread

קובץ: serverToFile

מחלקת Thread שמנהלת את התקשורת מ client אחד וכותבת את המידע לקובץ התואם

משתנים במחלקה		
שם משתנה	תפקיד המשתנה	סוג הערכים שהמשתנה מקבל
self.conn	שמירת התחברות של socket המשתמש סף	אובייקט socket
self.addr	שמירת ה IP של משתמש הסף	string
self.path	שמירת הנתיב לקובץ שאליו נשמר המידע.	string

פעולות במחלקה		
שם פעולה	טענת כניסה	טענת יציאה
__init__	מקבל חיבור וכותבת	אין
run	אין	הפעולה מקבלת את התקשורת מהקליינט ומכניסה את התקשורת לקובץ המתאים

מחלקה: ClientFileThread

קובץ: fileToGui

מטרת המחלקה היא לקרוא את קבצי משתמש הסף ולעדכן את ממשק המשתמש

משתנים במחלקה		
שם משתנה	תפקיד המשתנה	סוג הערכים שהמשתנה מקבל
self.monitor	שמירת המידע של משתמש הסף	Array
self.person_page	שמירת החלון של משתמש הסף	אובייקט של ממשק גרפי
self.filename	שם הקובץ שהמחלקה צריכה לקרוא	string
self._stop_event	מאורע שמטרתו לקבוע האם לעצור את פעולת threadn	אובייקט של threading.Event

פעולות במחלקה		
שם פעולה	טענת כניסה	טענת יציאה
__init__	מקבלת array ו אובייקט של ממשק גרפי	מממשת את ה thread
stop	אין	מפעילה את מאורע העצירה של threadn
stopped	אין	מחזירה True אם מאורע העצירה הופעל אחרת מחזירה False
read_last_line	אין	קוראת ומחזירה את השורה האחרונה בקובץ txt
run	אין	מפעילה את ה thread שקורא את המידע מקובץ טקסט ומעדכנת את חלון משתמש הסף בהתאם

מחלקה: DataService

קובץ: DataService

המחלקה המתקשרת עם מאגר הנתונים תוך שימוש ב sqlite3

משתנים במחלקה		
שם משתנה	תפקיד המשתנה	סוג הערכים שהמשתנה מקבל
self.db_path	שמירת מיקום מאגר הנתונים	string

פעולות במחלקה		
שם פעולה	טענת כניסה	טענת יציאה
__init__	אין	מממשת את המחלקה ושומרת את מיקום מאגר הנתונים
create_tables	אין	יוצר את מאגר הנתונים ואת הטבלאות במאגר הנתונים אם הם לא קיימות
addAdminUser	מקבלת שם משתמש וסיסמא	אם לא קיים במאגר הנתונים משתמש בעל שם זהה, מוסיף משתמש בעל שם וסיסמא
add_monitor_user	מקבל שם סיסמא ומפתח שתואם למפתח של משתמש הפיקוח	מוסיף משתמש סף למאגר הנתונים
GetAllAdminUsers	אין	מחזיר את כל משתמשי הפיקוח
GetAllMonitorUsers	אין	מחזיר את כל משתמשי הסף
get_monitor_list_by_admin_key	מקבל מפתח של משתמש פיקוח	מחזיר את כל משתמשי הסף בעלי מפתח משתמש פיקוח זהה
check_login	מקבל שם וסיסמא	בודק האם קיים משתמש

בעל שם וסיסמא זהים אם כן מחזיר את המשתמש אם לא מחזיר None		
מוחק את משתמש הסף לפי המפתח	מקבל מפתח של משתמש סוף	delete_monitor_by_key

מחלקה: ClientApp

קובץ: clientUserInterface

המחלקה הראשית של האפליקציה של משתמש הסף, אחראית על פעולת ממשק המשתמש

משתנים במחלקה		
שם משתנה	תפקיד המשתנה	סוג הערכים שהמשתנה מקבל
my_ip_label	הצגת הIP של המשתמש	אובייקט של ממשק משתמש
ip_label	הצגת המיקום לרישום IP השרת	אובייקט של ממשק משתמש
success_label	הצגת האם שינוי IP השרת הצליח	אובייקט של ממשק משתמש
ip_entry	מקום לרשום את כתובת השרת הרצוי	אובייקט של ממשק משתמש
set_button	כפתור לביצוע השינוי של כתובת השרת הרצוי	אובייקט של ממשק משתמש

פעולות במחלקה		
שם פעולה	טענת כניסה	טענת יציאה
SetIp	מקבלת IP	אם הIP תקפה שינוי הIP בקובץ ההגדרות
checkIp	מקבלת IP	בודקת האם הIP תקפה
get_ip	אין	מציאת הIP של המחשב

מחלקה: get_focused_window_thread

קובץ: client

thread אסיפת החלון הממוקד של משתמש הסף והוספתו לבאפר גלובלי

פעולות במחלקה		
שם פעולה	טענת כניסה	טענת יציאה
__init__	אין	מימוש הthread
run	אין	אסיפת החלון הממוקד של משתמש הסף והוספתו לבאפר גלובלי

מחלקה: Port53Scan

קובץ: client

thread איסוף בקשות הDNS של משתמש הסף והוספתן לבאפר גלובלי

פעולות במחלקה		
שם פעולה	טענת כניסה	טענת יציאה
__init__	אין	מימוש הthread
run	אין	thread אסיפת בקשות הDNS של משתמש הסף והוספתן לבאפר גלובלי

מחלקה: detect_key_press

קובץ: client

thread אסיפת הקשות המקלדת של משתמש הסף והוספתו לבאפר גלובלי

פעולות במחלקה		
שם פעולה	טענת כניסה	טענת יציאה
__init__	אין	מימוש threadn
run	אין	thread אסיפת הקשות המקלדת של משתמש הסף והוספתו לבאפר גלובלי
key_decode	מקבל string שמתאר מקש	מחזיר את המקש אם המקש רגיל אחרת מחזיר תיאור של המקש

סקירת בעיות אלגוריתמיות וקטעי קוד מיוחדים

הבעיה האלגוריתמית: שליחת מידע המיוצר בשלוש thread-ים שונים

כאמור נתקלתי בבעיה שהמידע שאני מייצר בקליינט מפוצל ל-3 threads נפרדים. על מנת לשלוח את המידע לשרת נאלצתי להשתמש בפתרון מיוחד: producer consumer אלגוריתם שבו ה threads היוצרים (producer) מייצרים מידע ומכניסים אותו לבאפר גלובלי. מהבאפר הגלובלי הזה thread נוסף (consumer) שולף את המידע ומשתמש בו: במקרה שלי, מצפין אותו ושולח אותו לשרת.

threads היוצרים:

```
class get_focused_window_thread(Thread):
    def __init__(self):
        Thread.__init__(self)

    def run(self):
        global queue
        global stop_event
        temp = None

        while not stop_event.is_set():
            window_title = pyautogui.getActiveWindowTitle()
            if temp != window_title:
                time_string = time.strftime("%m/%d/%Y, %H-%M-%S", time.localtime())
                msg = f"{time_string}:[focused_window]:{str(window_title)}"
                condition.acquire()

                if len(queue) >= QUEUEU_SIZE:
                    print("queue is full, waiting")
                    condition.wait()

                queue.append(msg)
                print(msg)
                condition.notify()
                condition.release()
                temp = window_title
            print("out of loop")

class Port53Scan(Thread):
    def __init__(self):
        Thread.__init__(self)

    def run(self):
```

```
global queue
global stop_event
msg = ""
while not stop_event.is_set():
    try:
        pkts = sniff(filter="udp and port 53", count=5)
        print(pkts)
        for pkt in pkts:
            time_string = time.strftime("%m/%d/%Y, %H-%M-%S", time.localtime())
            msg = f"{time_string}: [PORT 53]: {str(pkt[DNSQR].qname.decode())}"
    except:
        msg = ""
        pass

    if msg != "":
        condition.acquire()

        if len(queue) >= QUEUEU_SIZE:
            print("queue is full, waiting")

            condition.wait()
            queue.append(msg)

        condition.notify()
        condition.release()
        msg = ""

print("out of loop")
```

```
class detect_key_press(Thread):
    def __init__(self):
        Thread.__init__(self)

    def key_decode(self, key):
```

```

try:
    key = key.replace("'", "")
    return chr(key)
except:
    key = key.replace("'", "")
    if len(key) > 1:
        return f"[{key}]"
    else:
        try:
            (ord(key))
        except:
            return key
return key

def run(self):
    global queue
    global stop_event
    msg = ""
    while not stop_event.is_set():
        event = keyboard.read_event()
        if event.event_type == keyboard.KEY_DOWN:
            msg = msg + self.key_decode(event.name)
        if len(msg) >= 30:
            condition.acquire()
            if len(queue) >= QUEUEU_SIZE:
                print("queue is full, waiting")
                condition.wait()
            time_string = time.strftime("%m/%d/%Y, %H-%M-%S", time.localtime())
            finalmsg = f"{time_string}:{keypress}:{msg}"
            queue.append(finalmsg)
            print(finalmsg)
            msg = ""
            condition.notify()
            condition.release()
    print("out of loop")

```

:consomern thread

```

class ConsumerThread(Thread):
    def __init__(self, ip, port):
        Thread.__init__(self)
        self.ip = ip

```

```
self.port = port

def run(self):

    global queue
    global stop_event
    f = Fernet(KEY)

    while not stop_event.is_set():
        try:
            print("attempting to connect to " + self.ip)
            s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
            s.settimeout(5)
            result = s.connect((self.ip, self.port))
            while not stop_event.is_set() and result == None:

                condition.acquire()

                if not queue:
                    print("queue is empty, waiting")
                    condition.wait()
                m = f.encrypt(queue.pop().encode("utf-8"))

                s.send(m)

                condition.notify()
                condition.release()
            print("out of while")
        except KeyboardInterrupt:
            s.close()
            break
        except Exception as e:
            print(e)
            s.close()
```

הבעיה האלגוריתמית: הצפנה

השתמשתי בהצפנה סימטרית תוך שימוש ב Fernet

הצפנה:

```
m = f.encrypt(queue.pop().encode("utf-8"))
```

פיענוח:

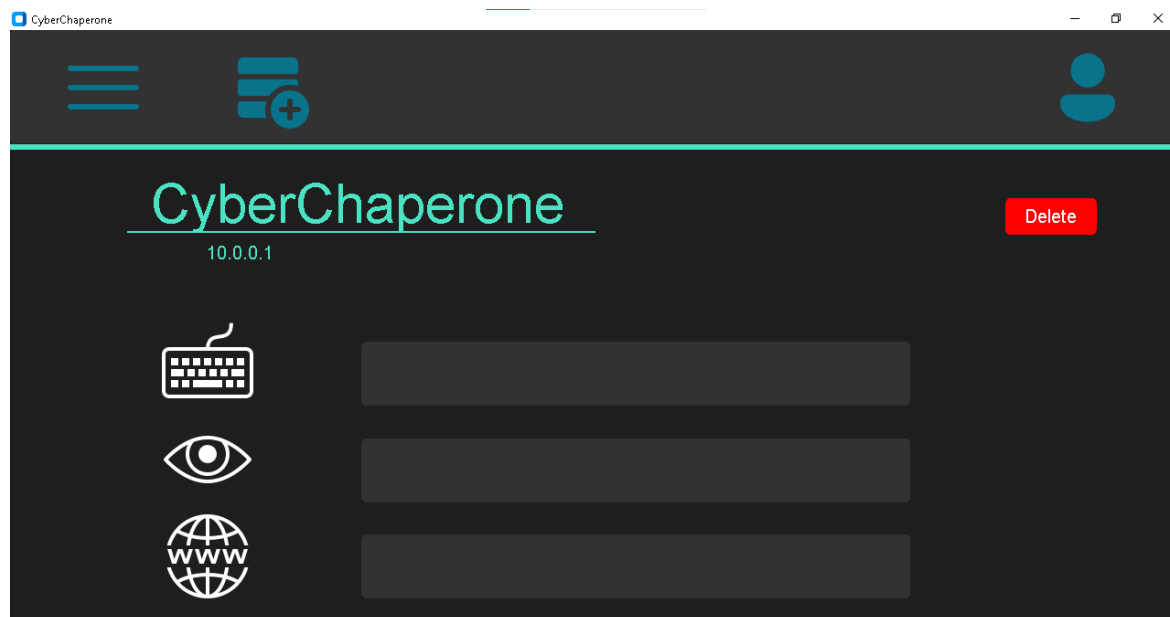
```
data = f.decrypt(self.conn.recv(4096).decode("utf-8"))
```

מסמך בדיקות מלא

שם הבדיקה	מטרת הבדיקה	מה נדרש לבצע	מספר נסיון	תוצאות הבדיקה	כיצד יש לפתור את הבעיה
אי קריסת לקוח	לוודא שהלקוח לא קורס בנפילת שרת	להפעיל את השרת, להפעיל את הלקוח, לכבות את השרת	1	הלקוח קרס	להוסיף try-except ולאפשר נסיון התחברות חוזר
			2	עבר	-
אי קריסת שרת	לוודא שהשרת לא קורס בעקבות קריסת לקוח	להפעיל את השרת, לחבר לקוח, להקריס לכבות לקוח	1	עבר	-
התחברות של מרובה של לקוחות	לוודא שהשרת יכול לנהל כמה לקוחות בו זמנית	להפעיל את השרת, לחבר מספר לקוחות בו זמנית	1	עבר	-
קבלת מידע תקין וזהה	לוודא שהמידע המתקבל זהה למידע היוצא	להפעיל את השרת, לחבר לקוח, להדפיס את המידע הלקוח מוציא, להדפיס את המידע שהשרת מקבל ולהשוות	1	עבר	-
בדיקת עדכון הגרפיקה	לוודא שה thread מעדכן את הגרפיקה	להפעיל שרת, לחבר לקוח להפעיל את הגרפיקה, לשלוח מידע מהלקוח	1	המידע הוצג אך התקבלה error בהרצת הגרפיקה	מכיוון שה error לא פוגעת בהרצת except תספיק
			2	עבר	-
בדיקת עדכון הגרפיקה בעת התחברות של מרובה של לקוחות	לוודא ש thread עדכון גרפיקה נפתח עבור כל משתמש	להפעיל שרת, לחבר מספר לקוחות, להפעיל את ממשק המשתמש, לשלוח מידע מכמה משתמשים בו זמנית	1	עבר	-

-	עבר	1	להפעיל את ממשק המשתמש, ולהתחבר למשתמש	לוודא שההתחברות וגישה למאגר הנתונים פעילה	בדיקת התחברות למשתמש
-	עבר	1	להפעיל את ממשק המשתמש, להתחבר למשתמש, להוסיף משתמש, למחוק משתמש	לוודא שגישה למאגר נתונים והוספת ומחיקת חלקים ממשק משתמש מתאפשרת	בדיקת הוספת מחיקת משתמש סף

מדריך למשתמש



ברוכים הבאים למדריך למשתמש של תוכנת CYBERCHAPERONE - התוכנה המובילה
 ניטור מחשבים.

במדריך זה נסביר כיצד להתקין את התוכנה במחשב הראשי - השרת ובמחשבי הקצה .

בעזרת התוכנה תוכלו לנהל מעקב מדויק אחר הפעולות המתבצעות במחשבי הקצה, לראות את
 חלון הפעולה בו המשתמש צופה במחשב הקצה , מה משתמש הקצה מקליד וגם שמות אילו
 אתרי אינטרנט מחשב הקצה פונה.

כל הנתונים נאספים לתוך קובץ אישי של כל מחשב קצה משמש כארכיב פעולות של המשתמש.

תוכן :

עמוד	נושא	
41	התקנת תוכנת Cyber Chaperone שרת ראשי	1
42	התקנת תוכנת Cyber Chaperone מחשב קצה	2
43	כניסה למסך שרת ראשי	3
43	הוספת מחשבי קצה לניטור	4
44	צפייה במחשב קצה	5.
45	הסרת מחשב קצה מניטור	6.
45	תקלות ופתרונות	7.
46	עץ קבצים	8.

התקנת תוכנת שרת :

נדרש : קבצי התקנה

סדר פעולות :

1. העתק את תוכן ספריית SERVER לספרייה חדשה במחשב שרת
2. הפעל את תוכנת השרת על ידי לחיצה על קובץ ServertoFile. כעת תוכנת השרת המאפשרת איסוף מידע ממחשבי הקצה מופעלת. תוכנה זו צריכה לפעול על עוד רוצים להמשיך לנטר את פעילות מחשבי הקצה ויש צורך להפעיל את התוכנה בכל פעם שהמחשב מופעל מחדש.
3. באפשרותך להפעיל את הקובץ בכל הפעלה של המחשב באופן אוטומטי להפעלה אוטומטית :

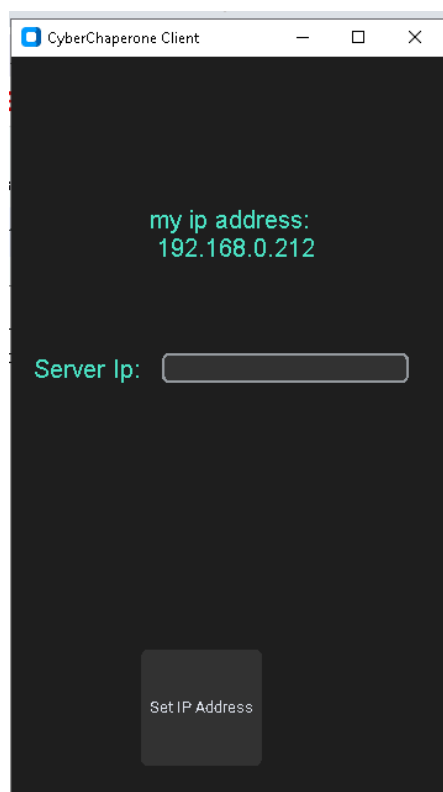
- א. לחץ על מקש חלונות + R
 - ב. העתק קיצור דרך של קובץ ServertoFile לספרייה זו.
 - ג. ניתן לסגור את חלון הקבצים
- כעת כל פעם שהמחשב ידלק - תוכנת ה-ServertoFile תופעל.

התקנה במחשב קצה :

נדרש : קבצי התקנה
כתובת IP של המחשב הראשי - השרת

סדר פעולות :

1. העתק את תוכן תיקיית ההתקנה לתיקיה חדשה במחשב.
2. הפעל את תוכנת CyberChaperone Client על ידי לחיצה על קובץ ClientUserInterface
3. בחלון הבא עליך להזין את כתובת IP של המחשב הראשי : לדוגמה 192.168.0.1



4. לאחר הזנת כתובת השרת לחץ על SET IP ADDRESS.
5. במידה והתקבלה הודעה SUCCESSFULLY CHANGED IP סגור את החלון באמצעות לחיצה על ה-X
6. הפעל את תוכנה מחשב הקצה על ידי לחיצה על קובץ CLIENT. כעת תוכנת מחשב הקצה פועלת ומנטרת את הפעולות המתבצעות על המחשב ושולחת למחשב הראשי - השרת. תוכנה זו צריכה לפעול כל עוד הניטור רצוי ובכל פעם שהמחשב מופעל מחדש.
7. באפשרותך להפעיל את הקובץ בכל הפעלה של המחשב באופן אוטומטי להפעלה אוטומטית :

- א. לחץ על מקש חלונות + R
- ב. העתק קיצור דרך של קובץ CLIENT לספרייה זו.
- ג. ניתן לסגור את חלון הקבצים

כעת כל פעם שהמחשב ידלק - תוכנת ה-CLIENT תופעל.

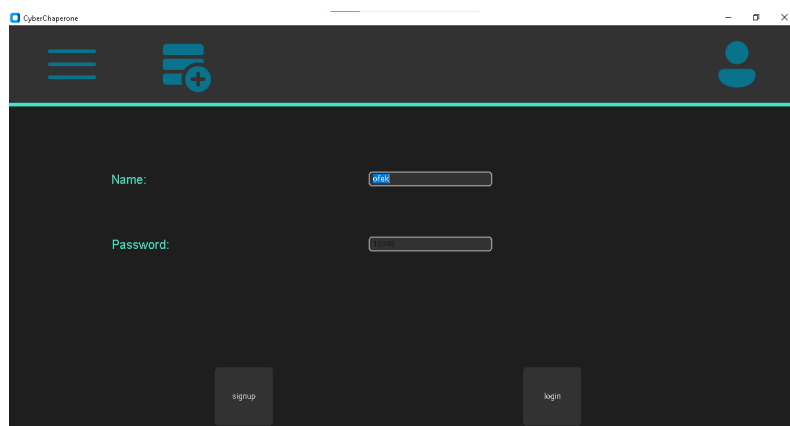
שימוש בתוכנת CyberChaperone :

1. הפעל את תוכנת CyberChaperone על ידי לחיצה על קובץ Main.



2. בצע כניסה מאובטחת על ידי לחיצה על צלמית המשתמש והזן את שם המשתמש והסיסמה.

3. לחץ על כפתור LOGIN

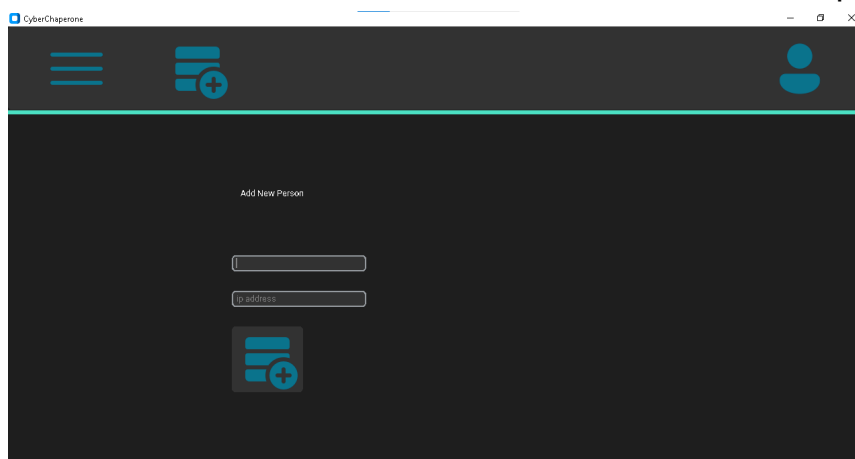


4. ניתן כעת להוסיף רישום למחשבי קצה עליהם מותקנת תוכנת ה-Client. לשם כך לחץ כל



כפתור הוספת מחשב קצה .

5. בחלון זה הזן את השם הרצוי למחשב זה ואת כתובת ה-IP של המחשב.



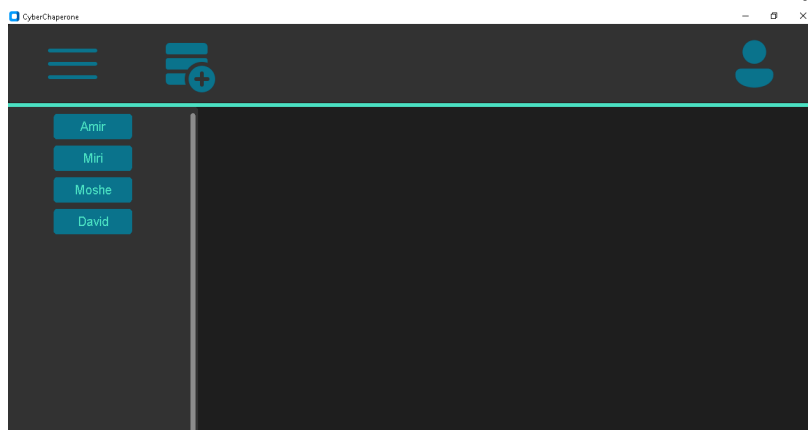

6. לחץ על הוספה .

7. ניתן להוסיף את כל שאר הנתונים של מחשבי הקצה שאנו מנטרים.

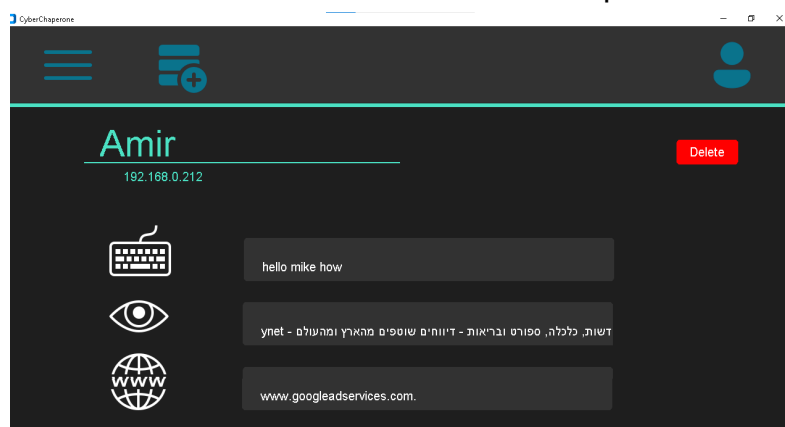
רשימת מחשבי קצה וניטור:



1. לחץ על כפתור רשימת מחשבי קצה
2. בחר מחשב קצה לצפייה בנתונים



3. ניתן לצפות בניטור מחשב הקצה ומה מתבצע במחשב ברגע זה.



לדוגמא במחשב של AMIR ניתן לראות שהוא מקליד : hello mike how
צופה בחלון : Ynet - חדשות כלכלה ספורט....
המחשב שלו פונה לשאילתת DNS
ל: www.googleadservices.com

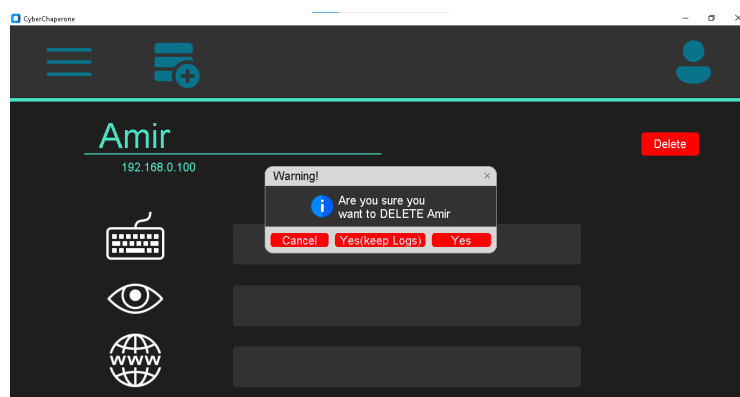


4. ניתן לבחור מחשב אתר לצפייה על ידי לחיצה על כפתור רשימת מחשבי הקצה

הסרת מחשב קצה מניטור



1. לחץ על כפתור רשימת מחשבי קצה
2. בחר את המחשב הקצה להסרה
3. לחץ על DELETE



4. באפשרותך למחוק את מחשב הקצה מניטור יחד עם הקבצים המכילים את ארכיב הניתור או רק למחוק את מחשב הקצה מניטור ללא קבצי הארכיב.

תקלות ופתרונות

תקלה	פתרונות
לא מצליח לבצע כניסה לתוכנת שרת	<ol style="list-style-type: none"> 1. האם כל הקבצים מההתקנה נמצאים בספרייה? 2. האם שם המשתמש הוזן באופן תקין? 3. האם הסיסמא הוזנה באופן תקין?
תוכנת השרת לא מציגה מחשב קצה	<ol style="list-style-type: none"> 1. האם הוספת מחשב קצה לרשימת המחשבים? 2. האם הוספת את כתובת ה-IP של המחשב קצה באופן תקין? 3. נסה להוסיף את המחשב שוב.
תוכנת השרת לא מציגה את נתוני הניטור כלל	<ol style="list-style-type: none"> 1. בדוק חיבור מחשב שרת לרשת המחשבים 2. וודא שתוכנת Server to file פעילה 3. בדוק הרשאה של התוכנה לעבור את חומת האש של התחנה.
תוכנת השרת לא מציגה את נתוני הניטור ממחשב קצה יחיד	<ol style="list-style-type: none"> 1. בדוק חיבור מחשב קצה לרשת המחשבים 2. וודא שתוכנת Client פעילה במחשב קצה 3. בדוק הרשאה של התוכנה לעבור את חומת האש של התחנה.

עץ קבצים

```
F:.\
├── client
│   ├── client.py
│   ├── clientUserInterface.py
│   └── settings.txt
├── server
│   ├── DataBase.db
│   ├── DataService.py
│   ├── FileToGui.py
│   ├── Main.py
│   ├── serverToFile.py
│   └── UserInterface.py
├── assets
│   ├── account.png
│   ├── addPerson.png
│   ├── eye.png
│   ├── keyboard.png
│   ├── menu.png
│   └── web.png
└── files
    ├── 127.0.0.1.txt
    ├── 172.16.3.181.txt
    └── 172.16.4.72.txt
```

רפלקציה

תיאור תהליך העבודה

הפרויקט בקורס הסייבר התפרס על פני כל שנת הלימודים והיה חוויה מעשירה שסיפקה הבנה מקיפה של עקרונות ופרקטיקות רבות. הפרויקט כלל תכנון, עיצוב, פיתוח, תכנות, integration hell, עליות וירידות רבות. אך לבסוף אני מרוצה מהתוצר המוגמר.

לפי דעתי דבר משמעותי יותר הוא שאני מרוצה מהלמידה וההתפתחות שלי כמתכנת. מעולם לא התקרבת לפרויקט תכנות בגודל הזה, נתקלתי תחומים רבים, הרבה מהם חדשים לי אך למדתי השקעתי והתקדמתי.

הצלחות:

- **תכנון ראשוני:** בתחילת השנה התחלתי עם תכנון כללי, שחילק את הפרויקט למשימות קטנות יותר לפי שלבים. מפת דרכים ברורה זו שמרה אותי ממוקד ובכך עזרה לי לעמוד בלוחות הזמנים.
- **ביצוע הפרויקט:** במהלך השנה יישמתי ולמדתי בהצלחה פרוטוקולים מרכזיים, שעזרו לפתור בעיות בקלות יחסית ומצאתי את עצמי נהנה מכתובת הקוד.
- **למידה והתאמה:** התאמתי את עצמי לאתגרים חדשים והחלתי טכניקות שנלמדו ביעילות תוך כדי התקדמות הפרויקט.

אתגרים:

- **קשיים טכניים:** נתקלתי במספר בעיות טכניות, בעיקר בהוספת הצפנה ובשילוב כלים שונים בתחום הסייבר, נאלצתי להתפשר על התחום הזה בעקבות מגבלות הזמן.
- **מגבלות במודלים מוכנים:** מספר פעמים השתמשתי במודלים מוכנים על מנת לשפר את הפרויקט שלי, לצערי מספר פעמים נתקלתי בבעיות בהם המודלים לא התחברו עם חלק נוסף של התוכנה שלי. נאלצתי לכתוב קטעי קוד "מסורבלים" על מנת ליצור את חלק הפאזל החסר לשילוב בין התוכנה למודל.

קשיים ופתרונות:

- **פתרון בעיות:** נעזרתי בפורומים מקוונים, תיעוד ומחקר עצמאי כדי לפתור בעיות טכניות.
- **בקשת עזרה:** לפעמים לא ידעתי כיצד לפתור בעיה, אך בעזרתם של המורה המנחה, חברי לכיתה ואבי קיבלתי הצעות לתיקון שיפור ופתירת בעיות.
- **ניהול זמן:** לאורך השנה, ניהלתי את הזמן בצורה מדויקת על מנת לאזן בין משימות הפרויקט לבין דרישות אקדמיות אחרות תוך הוספת "זמן פציעות" למקרה ואצטרך זמן נוסף או שאתקל בבעיות לא צפויות.

תהליך הלמידה

במהלך השנתיים בלימודי סייבר, עקומת הלמידה הייתה תלולה אך מתגמלת. למדתי:

- **python:** למרות שהכרתי את שפת התכנות, מעולם לא התנסיתי איתה ובמהלך חצי השנה הראשונה בשיעורי סייבר למדנו את שפת התכנות ומודלים בתוכה.
- **שליטה בכלים:** הפכתי מיומן בשימוש בכלי אבטחת מידע לדוגמת Wireshark.
- **למידה עצמית:** למרות שלמדתי הרבה דרך שיעורי הסייבר, למדתי גם ללמוד לבד - העמקתי והתנסתי בתחומים נוספים בתחום, למדתי בעצמי תוך הכנה לצבא והגעתי לתוצאות.

כלים לעתיד

הכלים והכישורים שרכשתי במהלך הפרויקט הם בעלי ערך רב להמשך הדרך.

בין הכישורים שרכשתי :

- **כישורים תכנון:** שיפור היכולת לנתח ולהעריך סיכוני אבטחת מידע.
- **מיומנות טכנית:** שיפור מיומנות עם תוכנות וכלי אבטחת מידע.
- **כישורי פתרון בעיות:** חיזוק היכולת לפתור בעיות טכניות מורכבות באופן עצמאי.
- **כישורי תכנות:** שיפרתי את יכולותי במעבר מרעיון לתוכנה ליישום בכתיבת קוד.

תובנות מהתהליך

הפרויקט הדגיש את החשיבות של למידה מתמשכת ושיתוף ידע. תובנות הכוללות:

- **סיוע ממומחים:** חיפוש הנחיה ממשאבים מקוונים בתחום היה קריטי. המומחיות שלהם עזרה לי לנווט בעיות מורכבות ולהעמיק את ההבנה שלי בתחום.
- **למידה עצמאית:** פיתוח משמעת ללמוד באופן עצמאי ולהישאר בעל מוטיבציה היה חיוני להצלחה.

במבט לאחור: מה הייתי עושה אחרת?

במבט לאחור, ישנם מספר היבטים שהייתי ניגש אליהם אחרת:

- **ניהול משאבים:** הקדשת זמן ומשאבים נוספים לשלב המחקר הראשוני הייתה מייעלת את תהליך היישום.
- **תיעוד משופר:** שמירה על תיעוד מפורט לאורך כל השנה הייתה מקלה על המעברים בין שלבים ופתרון בעיות.
- **כתיבת קוד נקי:** אני נוהג לכתוב קוד שעובד אך לרוב מתעלם מהדרך שבה אני כותב את הקוד, הרבה פעמים זה גורם לקוד שקשה לקרוא ולהבין ושיש לסדר לאחר מכן. בעתיד אשתדל לכתוב קוד קריא מובן ונקי.

שיפורים פוטנציאליים עם משאבים נוספים

עם משאבים נוספים, ניתן היה לשפר משמעותית את הפרויקט:

- שיפור ההצפנה - מעבר ל private public key, והצפנת סיסמאות ושמות במאגר המידע.
- שיפור הצגת מידע- שימוש בגרפים לסקירת המידע המתקבל.
- הוספת אזהרות AI - ניתוח המידע למציאת התנהגויות שלא תואמות לסביבת העבודה / מציאת פריצות למערכות ועוד.

לדוגמא:

- משתמש נכנס כל יום למחשב מ8:00 בבוקר ל 17:00 אחר הצהריים. יום אחד המשתמש נכנס למחשב בשעה 02:00 בלילה. התנהגות לא רגילה ולכן יכולה לסמן על פריצה.
- אדם נכנס לאתר שאינו מתאים לסביבת העבודה, ה-AI מזהה את האתר וקובע האם הוא לא מתאים או מתאים לעבודה, במקרה ואינו מתאים ידווח למעסיק ו/או יחסום את גישה.

ישנם אופציות רבות נוספות לשימוש בניתוח המידע המתקבל.

תודות

אסיים עם תודה, לאבי ואמי שתמכו בי במהלך בלימודים והכנת הפרויקט בפרט, למורי למדעי המחשב וסייבר שלימדו אותי את רוב מה שאני יודע כיום ועזרו בבניית הפרויקט, ותודה לחברי לכיתה שעזרו לי לעבור את כל התהליך עם חיוך.

ביבליוגרפיה

Biondi, P. (2024). Scapy 2.6.0 documentation.

<https://scapy.readthedocs.io/en/latest/>

BoppreH. (2024). *Keyboard*. PyPI keyboard.

<https://pypi.org/project/keyboard/>

Bora, A. (2024). *Akascap/CTkMessageBox: A fully customizable MessageBox for customtkinter!*.

<https://github.com/Akascap/CTkMessageBox>

Clark, J. A. (2024). *Pillow*. Pillow (PIL Fork).

<https://pillow.readthedocs.io/en/stable/>

A computer science portal for geeks. GeeksforGeeks. (2024).

<https://www.geeksforgeeks.org/>

Gonen, B. (2021). *Python Programming*. Rashi Foundation. 2024,

https://data.cyber.org.il/python/python_book.pdf

Learn to code. W3Schools Online Web Tutorials. (2024).

<https://www.w3schools.com/>

rozenboim, O., gonen, barak, & hod, shlomi. (2020). *Networks*. Rashi foundation. 2024,

<https://data.cyber.org.il/networks/networks.pdf>

Schimansky, T. (2024). *Documentation introduction*. CustomTkinter.

<https://customtkinter.tomschimansky.com/documentation/>

Stack Overflow. (2024).

<https://stackoverflow.com/>

נספחים

קוד צד שרת

UserInterface.py

```
import customtkinter as ctk
from PIL import Image
from DataService import DataService
import os
import socket
from CTkMessagebox import CTkMessagebox

COLOR_TEXT_RED = "#FF0000"
COLOR_BACKGROUND_BLACK = "#212121"
COLOR_BACKGROUND_GRAY = "#323232"
COLOR_ICON_BLUE = "#0D738F"
COLOR_TEXT_GREEN = "#4BE4C5"
COLOR_TEXT_WHITE = "#FFFFFF"

FONT_BUTTON = ("italic", 20)
FONT_TITLE = ("italic", 60)
FONT_NORMAL_TEXT = ("italic", 20)
MONITOR_POS_KEY = 0
MONITOR_POS_NAME = 1
MONITOR_POS_IP = 2
MONITOR_POS_ADMIN_KEY = 3

class SlidePanel(ctk.CTkScrollableFrame):
    def __init__(self, parent, start_pos, end_pos, color):
        """initialize SlidePanel"""
        super().__init__(master=parent, fg_color=color)

        # general attributes
        self.parent = parent
        self.start_pos = start_pos - 0.02
        self.end_pos = end_pos
        self.width = abs(start_pos - end_pos)
        self.button_list = []
```

```
# animation logic
self.pos = self.end_pos
self.in_start_pos = False
self.rely = 0.2
self.relheight = 0.8

# layout
self.place(
    relx=self.pos,
    rely=self.rely,
    relwidth=self.width,
    relheight=self.relheight,
)

## adding buttons in update instead of here for login

def animate(self):
    """choose how to animate the slide panel"""
    if self.in_start_pos:
        self.animate_forward()
    else:
        self.animate_backwards()

def animate_forward(self):
    """animate the slide panel to its forward position"""
    if self.pos > self.end_pos:
        self.pos -= 0.018
        self.place(
            relx=self.pos,
            rely=self.rely,
            relwidth=self.width,
            relheight=self.relheight,
        )
        self.after(10, self.animate_forward)
    else:
        self.in_start_pos = False

def animate_backwards(self):
    """animate the slide panel to its back position"""
    if self.pos < self.start_pos:
        self.pos += 0.018
        self.place(
```

```

        relx=self.pos,
        rely=self.rely,
        relwidth=self.width,
        relheight=self.relheight,
    )
    self.after(10, self.animate_backwards)
else:
    self.in_start_pos = True

def add_button(self, person):
    """add a button to the side panel and connect it to a
    person"""

    button = ctk.CTkButton(
        master=self,
        text=person[MONITOR_POS_NAME], # name
        font=FONT_BUTTON,
        fg_color=COLOR_ICON_BLUE,
        command=lambda person=person:
self.parent.person_page(person),
        width=125,
        height=40,
        text_color=COLOR_TEXT_GREEN,
    )
    button.pack(pady=5)
    self.button_list.append(button)
    return button

def clear(self):
    """destroy all buttons in the panel"""
    for button in self.button_list:
        button.destroy()
    self.button_list = []

class TopPanel(ctk.CTkFrame):
    def __init__(self, parent, sidePanel, color):
        """initialize the top panel"""
        super().__init__(master=parent, fg_color=color)
        self._fg_color = color
        self.master = parent
        menu_button = ctk.CTkButton(

```

```
        self,
        image=ctk.CTkImage(Image.open(r"assets/menu.png"),
size=(90, 90)),
        text="",
        fg_color=COLOR_BACKGROUND_GRAY,
        command=sidePanel.animate,
        width=95,
        height=95,
    )

    add_person_button = ctk.CTkButton(
        self,
        image=ctk.CTkImage(Image.open(r"assets/addPerson.png"),
size=(90, 90)),
        text="",
        fg_color=COLOR_BACKGROUND_GRAY,
        command=self.master.place_person_page,
        width=95,
        height=95,
    )

    my_account_button = ctk.CTkButton(
        self,
        image=ctk.CTkImage(Image.open(r"assets/account.png"),
size=(90, 90)),
        text="",
        fg_color=COLOR_BACKGROUND_GRAY,
        command=lambda: [
            self.master.login_page.place_page(),
            self.master.signup_page.destroy_page(),
        ],
        width=95,
        height=95,
    )

    menu_button.place(relx=0.08, rely=0.5, anchor="center")
    add_person_button.place(relx=0.22, rely=0.5,
anchor="center")
    my_account_button.place(relx=0.92, rely=0.5,
anchor="center")
    colored_box = ctk.CTkFrame(master=parent,
fg_color=COLOR_TEXT_GREEN)
```

```
        colored_box.place(relx=-0.2, rely=0.19, relwidth=1.4,
                           relheight=0.01)

        self.place(relx=0, rely=0, relwidth=1, relheight=0.19)

class AddPersonPage(ctk.CTkFrame):
    def __init__(self, parent):
        """initialize add person page"""
        super().__init__(master=parent,
                           fg_color=COLOR_BACKGROUND_BLACK)
        self.master = parent

        self.add_new_person_label = ctk.CTkLabel(
            master=self,
            text="Add New Person",
            width=120,
            height=25,
            fg_color=COLOR_BACKGROUND_BLACK,
            text_color="white",
            corner_radius=3,
        )
        self.newName = ""

        self.enter_name_entry = ctk.CTkEntry(
            master=self,
            width=200,
            height=25,
            fg_color=COLOR_BACKGROUND_GRAY,
            placeholder_text="Name",
        )
        self.enter_ip_entry = ctk.CTkEntry(
            master=self,
            width=200,
            height=25,
            fg_color=COLOR_BACKGROUND_GRAY,
            placeholder_text="ip address",
        )

        self.add_new_person_button = ctk.CTkButton(
            self,
```

```

        image=ctk.CTkImage(Image.open(r"assets/addPerson.png"),
size=(90, 90)),
        text="",
        fg_color=COLOR_BACKGROUND_GRAY,
        command=lambda: self.master.add_person(
            self.enter_ip_entry.get(),
self.enter_name_entry.get()
        ),
        width=95,
        height=95,
    )

    self.add_new_person_label.place(rely=0.2, relx=0.2)

    self.enter_name_entry.place(rely=0.4, relx=0.2)
    self.enter_ip_entry.place(rely=0.5, relx=0.2)

    self.add_new_person_button.place(rely=0.6, relx=0.2)
    self.place(relx=-1.5, rely=0.2, relwidth=1.3, relheight=0.8)

def place_page(self):
    """place the page"""
    self.place(relx=0, rely=0.2, relwidth=1.3, relheight=0.8)

def destroy_page(self):
    """destroy the page"""
    self.place(relx=-1.5, rely=0.2, relwidth=1.3, relheight=0.8)

class PersonPage(ctk.CTkFrame):
    def __init__(self, parent, person):
        """create a new person page"""
        super().__init__(master=parent,
fg_color=COLOR_BACKGROUND_BLACK)
        self.master = parent
        self.person = person
        self.visible = False

        self.person_label = ctk.CTkLabel(
            master=self,
            text=self.person[MONITOR_POS_NAME],
            width=120,

```



```
        height=25,
        fg_color=COLOR_BACKGROUND_BLACK,
        text_color=COLOR_TEXT_GREEN,
        font=FONT_TITLE,
    )
    self.ip_label = ctk.CTkLabel(
        master=self,
        text=self.person[MONITOR_POS_IP],
        width=120,
        height=25,
        fg_color=COLOR_BACKGROUND_BLACK,
        text_color=COLOR_TEXT_GREEN,
        font=FONT_BUTTON,
    )

    self.delete_person_button = ctk.CTkButton(
        master=self,
        text="Delete",
        height=40,
        width=100,
        fg_color=COLOR_TEXT_RED,
        text_color=COLOR_TEXT_WHITE,
        font=FONT_BUTTON,
        command=lambda person=person:
self.master.get_master().destroy_monitor(
    person
),
    )

    self.bar = ctk.CTkFrame(master=self,
fg_color=COLOR_TEXT_GREEN)

    self.gray_frame1 = ctk.CTkFrame(
        master=self, width=600, height=70,
fg_color=COLOR_BACKGROUND_GRAY
    )

    self.gray_frame2 = ctk.CTkFrame(
        master=self, width=600, height=70,
fg_color=COLOR_BACKGROUND_GRAY
    )

    self.gray_frame3 = ctk.CTkFrame(
```

```
        master=self, width=600, height=70,
        fg_color=COLOR_BACKGROUND_GRAY
    )

    self.text_label = ctk.CTkLabel(
        master=self.gray_frame1,
        text="",
        width=120,
        height=25,
        fg_color=COLOR_BACKGROUND_GRAY,
        text_color=COLOR_TEXT_WHITE,
        font=FONT_BUTTON,
    )

    self.window_label = ctk.CTkLabel(
        master=self.gray_frame2,
        text="",
        width=120,
        height=25,
        fg_color=COLOR_BACKGROUND_GRAY,
        text_color=COLOR_TEXT_WHITE,
        font=FONT_BUTTON,
    )

    self.dns_label = ctk.CTkLabel(
        master=self.gray_frame3,
        text="",
        width=120,
        height=25,
        fg_color=COLOR_BACKGROUND_GRAY,
        text_color=COLOR_TEXT_WHITE,
        font=FONT_BUTTON,
    )

    self.keyboard_picture = ctk.CTkLabel(
        master=self,
        text="",
        image=ctk.CTkImage(Image.open(r"assets/keyboard.png"),
        size=(100, 100)),
        width=95,
        height=95,
    )

    self.eye_picture = ctk.CTkLabel(
        master=self,
```

```

        text="",
        image=ctk.CTkImage(Image.open(r"assets/eye.png"),
size=(100, 100)),
        width=95,
        height=95,
    )
    self.web_picture = ctk.CTkLabel(
        master=self,
        text="",
        image=ctk.CTkImage(Image.open(r"assets/web.png"),
size=(100, 100)),
        width=95,
        height=95,
    )

    self.keyboard_picture.place(rely=0.35, relx=0.13)
    self.eye_picture.place(rely=0.55, relx=0.13)
    self.web_picture.place(rely=0.75, relx=0.13)

    self.gray_frame1.place(rely=0.4, relx=0.3)
    self.gray_frame2.place(rely=0.6, relx=0.3)
    self.gray_frame3.place(rely=0.8, relx=0.3)

    self.text_label.place(rely=0.5, relx=0.05)
    self.window_label.place(rely=0.5, relx=0.05)
    self.dns_label.place(rely=0.5, relx=0.05)

    self.person_label.place(rely=0.05, relx=0.12)
    self.bar.place(rely=0.17, relx=0.1, relwidth=0.4,
relheight=0.005)
    self.ip_label.place(rely=0.19, relx=0.15)

    self.delete_person_button.place(relx=0.85, rely=0.1)

def update_text(self, text):
    """Update the text label"""
    self.text_label.configure(text=text)

def update_ip(self, text):
    """update the ip label"""
    self.ip_label.configure(text=text)

```

```
def update_window_text(self, text):
    """update current window text"""
    self.window_label.configure(text=text)

def update_dns_text(self, text):
    """update current dns request text"""
    self.dns_label.configure(text=text)

def get_master(self):
    """return the master of the object as a fix to a .cget not
    working"""
    return self.master

def place_page(self):
    """determines if you need to place a page or destroy it"""
    if self.visible:
        self.move_destroy_page()
    else:
        self.move_place_page()

def move_place_page(self):
    """places the page so you can see it"""
    self.place(relx=0, rely=0.2, relwidth=1, relheight=0.8)
    self.visible = True

def move_destroy_page(self):
    """destroys the page so you wont see it"""
    self.place(relx=-1, rely=0.2, relwidth=1, relheight=0.8)
    self.visible = False

class LoginPage(ctk.CTkFrame):
    def __init__(self, parent):
        """crates the login page"""
        super().__init__(master=parent,
fg_color=COLOR_BACKGROUND_BLACK)
        self.master = parent
        self.visible = False
        self.name_label = ctk.CTkLabel(
            master=self,
            text="Name:",
            width=120,
```

```

        height=25,
        fg_color=COLOR_BACKGROUND_BLACK,
        text_color=COLOR_TEXT_GREEN,
        font=FONT_BUTTON,
        anchor="w",
        justify="left",
    )
    self.password_label = ctk.CTkLabel(
        master=self,
        text="Password:",
        width=120,
        height=25,
        fg_color=COLOR_BACKGROUND_BLACK,
        text_color=COLOR_TEXT_GREEN,
        font=FONT_BUTTON,
        anchor="w",
        justify="left",
    )
    self.enter_name_entry = ctk.CTkEntry(
        master=self,
        width=200,
        height=25,
        fg_color=COLOR_BACKGROUND_GRAY,
        placeholder_text="Name:",
    )
    self.enter_password_entry = ctk.CTkEntry(
        master=self,
        width=200,
        height=25,
        fg_color=COLOR_BACKGROUND_GRAY,
        placeholder_text="Password:",
    )
    self.login_button = ctk.CTkButton(
        self,
        text="login",
        fg_color=COLOR_BACKGROUND_GRAY,
        command=lambda: [
            self.master.login_func(
                self.enter_name_entry.get(),
                self.enter_password_entry.get()
            ),
            self.login_mode(),
        ]
    )

```

```
        ],
        width=95,
        height=95,
    )
    self.signup_button = ctk.CTkButton(
        self,
        text="signup",
        fg_color=COLOR_BACKGROUND_GRAY,
        command=lambda: [self.master.signup_page.place_page(),
self.place_page()],
        width=95,
        height=95,
    )

    self.user_name_label = ctk.CTkLabel(
        self,
        text="username",
        width=120,
        height=25,
        fg_color=COLOR_BACKGROUND_BLACK,
        text_color=COLOR_TEXT_GREEN,
        font=FONT_BUTTON,
        anchor="w",
        justify="left",
    )

    self.ip_label = ctk.CTkLabel(
        self,
        text="ip",
        width=120,
        height=25,
        fg_color=COLOR_BACKGROUND_BLACK,
        text_color=COLOR_TEXT_GREEN,
        font=FONT_BUTTON,
        anchor="w",
        justify="left",
    )

    self.logout_button = ctk.CTkButton(
        self,
        text="logout",
        fg_color=COLOR_BACKGROUND_GRAY,
        command=lambda: self.master.signup_page.place_page(),
```

```
        width=95,
        height=95,
    )

    self.login_mode()

def place_page(self):
    """determines if you need to place a page or destroy it"""
    if self.visible:
        self.move_destroy_page()
    else:
        self.move_place_page()

def move_place_page(self):
    """places the page so you can see it"""

    self.place(relx=0, rely=0.2, relwidth=1.3, relheight=0.8)
    self.visible = True

def move_destroy_page(self):
    """destroys the page so you wont see it"""
    self.place(relx=-1, rely=0.2, relwidth=1, relheight=0.8)
    self.visible = False

def login_mode(self):
    """if not logged in, show the login page else show the user
    information page and logout option"""
    self.enter_password_entry.delete(0, last_index="end")
    if self.master.admin_user != None:

        self.name_label.place(relx=2, rely=2)
        self.password_label.place(relx=2, rely=2)
        self.enter_name_entry.place(relx=2, rely=2)
        self.enter_password_entry.place(relx=2, rely=2)
        self.login_button.place(relx=2, rely=2)
        self.signup_button.place(relx=2, rely=2)

        # show my login info and allow to logout
        temp_text = str(self.master.admin_user[1])
        self.user_name_label.configure(text=temp_text)
        self.user_name_label.place(rely=0.20, relx=0.1)
        self.ip_label.place(rely=0.5, relx=0.4)
```

```

        self.ip_label.configure(text=self.get_ip())
        self.logout_button.place(relx=0.4, rely=0.7)

        pass
    else:
        self.user_name_label.place(relx=2, rely=2)
        self.ip_label.place(relx=2, rely=2)
        self.logout_button.place(relx=2, rely=2)

        # show login page
        self.name_label.place(rely=0.20, relx=0.1)
        self.password_label.place(rely=0.40, relx=0.1)

        self.enter_name_entry.place(rely=0.2, relx=0.35)
        self.enter_password_entry.place(rely=0.40, relx=0.35)
        self.login_button.place(rely=0.8, relx=0.5)
        self.signup_button.place(relx=0.2, rely=0.8)

def get_ip(self):
    """gets the ip associated with the server"""
    try:
        hostname = socket.gethostname()
        ipv4_address = socket.gethostbyname(hostname)
        return ipv4_address
    except socket.gaierror:
        return "There was an error resolving the hostname."
    except Exception as e:
        return f"An unexpected error occurred: {e}"

class SignUpPage(ctk.CTkFrame):
    def __init__(self, parent):
        """Initializes the SignUp page"""
        super().__init__(master=parent,
fg_color=COLOR_BACKGROUND_BLACK)
        self.master = parent
        self.visible = False
        self.name_label = ctk.CTkLabel(
            master=self,
            text="Name:",
            width=120,
            height=25,

```



```
        fg_color=COLOR_BACKGROUND_BLACK,
        text_color=COLOR_TEXT_GREEN,
        font=FONT_BUTTON,
        anchor="w",
        justify="left",
    )
    self.password_label = ctk.CTkLabel(
        master=self,
        text="Password:",
        width=120,
        height=25,
        fg_color=COLOR_BACKGROUND_BLACK,
        text_color=COLOR_TEXT_GREEN,
        font=FONT_BUTTON,
        anchor="w",
        justify="left",
    )
    self.password_check_label = ctk.CTkLabel(
        master=self,
        text="Password:",
        width=120,
        height=25,
        fg_color=COLOR_BACKGROUND_BLACK,
        text_color=COLOR_TEXT_GREEN,
        font=FONT_BUTTON,
        anchor="w",
        justify="left",
    )
    self.enter_name_entry = ctk.CTkEntry(
        master=self,
        width=200,
        height=25,
        fg_color=COLOR_BACKGROUND_GRAY,
        placeholder_text="Name:",
    )
    self.enter_password_entry = ctk.CTkEntry(
        master=self,
        width=200,
        height=25,
        fg_color=COLOR_BACKGROUND_GRAY,
        placeholder_text="Password:",
    )
```

```

self.enter_password_check_entry = ctk.CTkEntry(
    master=self,
    width=200,
    height=25,
    fg_color=COLOR_BACKGROUND_GRAY,
    placeholder_text="Password:",
)

self.signup_button = ctk.CTkButton(
    self,
    text="signup",
    fg_color=COLOR_BACKGROUND_GRAY,
    command=lambda: [self.signup()],
    width=95,
    height=95,
)

self.warning_label = ctk.CTkLabel(
    master=self,
    text="",
    width=120,
    height=25,
    fg_color=COLOR_BACKGROUND_BLACK,
    text_color=COLOR_TEXT_RED,
    font=FONT_BUTTON,
    anchor="w",
    justify="left",
)

self.name_label.place(rely=0.1, relx=0.1)
self.password_label.place(rely=0.2, relx=0.1)
self.password_check_label.place(rely=0.3, relx=0.1)

self.enter_name_entry.place(rely=0.1, relx=0.3)
self.enter_password_entry.place(rely=0.2, relx=0.3)
self.enter_password_check_entry.place(rely=0.3, relx=0.3)

self.signup_button.place(rely=0.5, relx=0.5)
self.warning_label.place(relx=0.2, rely=0.6)

def place_page(self):
    """determines if you need to place a page or destroy it"""
    if self.visible:
        self.move_destroy_page()
    else:

```

```

        self.move_place_page()

    def move_place_page(self):
        """places the page so you can see it"""
        self.place(relx=0, rely=0.2, relwidth=1.3, relheight=0.8)
        self.visible = True

    def move_destroy_page(self):
        """destroys the page so you wont see it"""
        self.place(relx=-1, rely=0.2, relwidth=1, relheight=0.8)
        self.visible = False

    def signup(self):
        """sings up the user"""
        worked = self.master.signup_func(
            self.enter_name_entry.get(),
            self.enter_password_entry.get(),
            self.enter_password_check_entry.get(),
        )

        if not worked:
            self.warning_label.configure(text="Username already exists")
        else:
            self.warning_label.configure(text="")

from FileToGui import ClientFileThread

class BackPanel(ctk.CTkFrame):
    # all of this just to workaround .cget method not working on ctk objects
    def __init__(self, parent):
        """crates a back panel"""
        super().__init__(master=parent,
            fg_color=COLOR_BACKGROUND_BLACK)

    def get_master(self):
        """returns the master as a fix for .cget not working in ctk"""
        return self.master

```

```

class App(ctk.CTk):
    def __init__(self):
        """starts the application"""
        super().__init__(fg_color=COLOR_BACKGROUND_BLACK)
        self.geometry("900x640")
        self.title("CyberChaperone")
        self.db = DataService()
        self.db.create_tables()
        self.protocol("WM_DELETE_WINDOW", self.on_closing)

        self.admin_user = None
        self.monitor_list = []
        self.person_page_List = []
        self.thread_list = []

        # all of this just to workaround .cget method not working on
        ctk

        self.back_panel = BackPanel(self)
        self.back_panel.place(relx=0, rely=0, relwidth=1,
                               relheight=1)

        self.signup_page = SignUpPage(self)
        self.login_page = LoginPage(self)

        self.add_person_panel = AddPersonPage(self)

        self.side_panel = SlidePanel(self, 0, -0.25,
                                       COLOR_BACKGROUND_GRAY)
        self.top_panel = TopPanel(self, self.side_panel,
                                    COLOR_BACKGROUND_GRAY)

        self.add_person_page_visibility = False
        # button

    def login_func(self, name, password):
        """Login function, that handles the login and starts
        threads"""
        temp_admin_user = self.db.check_login(name, password)
        if self.admin_user == None and temp_admin_user != None:
            self.admin_user = temp_admin_user

```

```

        self.monitor_list =
self.db.get_monitor_list_by_admin_key(
    self.admin_user[0]
)
        self.login_page.place_page()
        for monitor in self.monitor_list:
            self.add_monitor(monitor)

def add_person(self, name, ip):
    """Add a new person to the list and the database, crates the
new needed gui and threads and starts them"""
    if self.admin_user == None:
        return

    monitor = self.db.add_monitor_user(name, ip,
self.admin_user[0])
    # updating list
    self.monitor_list =
self.db.get_monitor_list_by_admin_key(self.admin_user[0])
    self.add_monitor(monitor)

def add_monitor(self, monitor):
    monitor_button = self.side_panel.add_button(monitor)
    monitor_PersonPage = PersonPage(self.back_panel, monitor)
    self.person_page_List.append((monitor, monitor_PersonPage,
monitor_button))
    thread = ClientFileThread(monitor, monitor_PersonPage)
    thread.start()
    self.thread_list.append(thread)

def destroy_monitor(self, monitor):
    """deletes a monitor user after issuing a warning"""
    msg = CTkMessageBox(
        master=self,
        width=400,
        height=40,
        title="Warning!",
        message=f"Are you sure you want to DELETE
{monitor[MONITOR_POS_NAME]}",
        option_1="Yes",
        option_2="Yes (keep Logs)",
        option_3="Cancel",

```

```

        fg_color=COLOR_BACKGROUND_GRAY,
        button_color=COLOR_TEXT_RED,
        text_color=COLOR_TEXT_WHITE,
        font=FONT_NORMAL_TEXT,
        option_focus="Cancel",
    )

    response = msg.get()

    if response == "Cancel" or response == None:
        return
    if response == "Yes":
        this_folder = os.path.dirname(os.path.abspath(__file__))
        path = os.path.join(this_folder,
f"files/{monitor[MONITOR_POS_IP]}.txt")
        if os.path.isfile(path):
            os.remove(path)

    for x in self.person_page_List:
        if x[0][MONITOR_POS_KEY] == monitor[MONITOR_POS_KEY]:
            x[1].destroy()
            x[2].destroy()
            self.person_page_List.remove(x)

    self.db.delete_monitor_by_key(monitor[MONITOR_POS_KEY])
    break

    for thread in self.thread_list:
        if thread.monitor[MONITOR_POS_KEY] ==
monitor[MONITOR_POS_KEY]:
            thread.stop()
            break

def place_person_page(self):
    """
    if not self.add_person_page_visibility:
        self.add_person_panel.place_page()
    else:
        self.add_person_panel.destroy_page()
        self.add_person_page_visibility = not
self.add_person_page_visibility

```

```
def person_page(self, person):
    """placing the presence page of by the key of the person"""
    for monitor, panel, button in self.person_page_List:
        if monitor[MONITOR_POS_KEY] == person[MONITOR_POS_KEY]:
            panel.place_page()
        else:
            panel.move_destroy_page()

def logout(self):
    """logs out and clears all info from gui, stops unneeded
    threads"""
    self.side_panel.clear()
    for x in self.thread_list:
        x.stop()
    self.thread_list = []
    self.monitor_list = []
    self.admin_user = None

def signup_func(self, name, password, password_check):
    """signup a user to db after checking user input"""
    if password == password_check:
        return False

    return self.db.add_monitor_user(name, password)

def on_closing(self):
    """closes the application"""
    print("-----CLOSING-----")
    for x in self.thread_list:
        x.stop()

    self.destroy()
    exit()
```

serverToFile.py

```
import threading
import os
import socket
from DataService import DataService
from cryptography.fernet import Fernet

KEY = b"afcbXv_0yebyn2iJbbt_DDvQdec3f96ImNXLq-AAGT0="

MONITOR_POS_KEY = 0
MONITOR_POS_NAME = 1
MONITOR_POS_IP = 2
MONITOR_POS_ADMIN_KEY = 3

class serverThread(threading.Thread):
    def __init__(self, conn, addr):
        threading.Thread.__init__(self)
        self.conn = conn
        self.addr = addr[0] # ip

        this_folder = os.path.dirname(os.path.abspath(__file__))
        self.path = os.path.join(this_folder,
f"files/{self.addr}.txt")

    def run(self):
        f = Fernet(KEY)
        if not os.path.isfile(self.path):
            with open(self.path, "x") as file:
                pass

        while True:
            with open(self.path, "a", encoding="utf-8") as file:
                try:
                    data =
f.decrypt(self.conn.recv(4096).decode("utf-8"))
                    data = data.decode("utf-8")
                    file.write(f"{data} \n")
                except Exception as e:
                    print(e)
                    break
```



```
# server IP and PORT
IP = "0.0.0.0"
PORT = 8882

def main():
    """main function to start the server"""
    print("starting server")
    with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
        s.bind((IP, PORT))
        thread_list = []
        data = DataService()
        try:
            while True:
                try:
                    s.listen(1)
                    conn, addr = s.accept()
                    print(f"Connect attpted to server by {addr}")
                    if addr[0] in [
                        str(x[MONITOR_POS_IP]) for x in
data.GetAllMonitorUsers()
                    ]:
                        print(f"Connectd")
                        x = serverThread(conn, addr)
                        x.start()
                        thread_list.append(x)
                    else:
                        print(f"connection from unknown - ignoring:
{addr}")
                except Exception as e:
                    print(e)

            except Exception as e:
                print(e)
                for i in thread_list:
                    i.join()

if __name__ == "__main__":
    main()
```

Main.py

```
from UserInterface import App

if __name__ == "__main__":
    try:
        app = App()

        app.mainloop()
    except Exception as e:
        print(e)
```

FileToGui.py

```
import threading
import time
import os

MONITOR_POS_KEY = 0
MONITOR_POS_NAME = 1
MONITOR_POS_IP = 2
MONITOR_POS_ADMIN_KEY = 3

class ClientFileThread(threading.Thread):
    def __init__(self, monitor, person_page):
        threading.Thread.__init__(self)
        self.monitor = monitor
        self.person_page = person_page
        this_folder = os.path.dirname(os.path.abspath(__file__))
        self.filename = os.path.join(
            this_folder, f"files/{monitor[MONITOR_POS_IP]}.txt"
        )
        self._stop_event = threading.Event()

    def stop(self):
        self._stop_event.set()
```

```

def stopped(self):
    return self._stop_event.is_set()

def read_last_line(self):
    last_line = None
    if not os.path.exists(self.filename):
        return None

    with open(self.filename, "rb") as f:
        try: # catch OSError in case of a one line file
            f.seek(-2, os.SEEK_END)
            while f.read(1) != b"\n":
                f.seek(-2, os.SEEK_CUR)
        except OSError:
            f.seek(0)
        last_line = f.readline().decode()

    return last_line

def run(self):
    while not self.stopped():
        time.sleep(1)
        data = self.read_last_line()

        if data:
            data_split = data.split(":", 3)
            try:
                match data_split[1]:
                    case "[keypress]":
                        data = data_split[2].replace("[space]",
" ")

                        self.person_page.update_text(data)
                    case "[focused_window]":

self.person_page.update_window_text(data_split[2])
                    case "[PORT 53]":

self.person_page.update_dns_text(data_split[2])

# updating Tkinter with an outside thread is not
recommended-

```

```

        # as tkinter is not thread safe. However, I use it
        here as a workaround-
        # is unavailable\not practical for this project
        currently.
        # for now, catching the exception and continuing is
        all thats possible.
        except RuntimeError as e:
            pass

```

DataService.py

```

import sqlite3
import os.path

class DataService:
    def __init__(self):
        base_dir = os.path.dirname(os.path.abspath(__file__))
        self.db_path = os.path.join(base_dir, "DataBase.db")

    def create_tables(self):
        with sqlite3.connect(self.db_path) as db:
            curs = db.cursor()
            curs.execute(
                """ CREATE TABLE IF NOT EXISTS "AdminUsers" (
                    "key" INTEGER PRIMARY KEY AUTOINCREMENT
                ,
                    "username" text NOT NULL UNIQUE,
                    "password" text NOT NULL
                ); """
            )
            db.commit()
            curs.execute(
                """CREATE TABLE IF NOT EXISTS MonitorUsers (
                    "key" INTEGER PRIMARY KEY AUTOINCREMENT
                ,
                    "name" TEXT NOT NULL,
                    "ip" TEXT NOT NULL,
                    "adminKey" INTEGER NOT NULL
                ); """
            )

```

```

    )
    db.commit()

def addAdminUser(self, username, password):
    with sqlite3.connect(self.db_path) as db:
        try:
            curs = db.cursor()
            if username in curs.execute("SELECT username FROM
adminUsers"):
                return False

            curs.execute(
                "INSERT INTO adminUsers VALUES (NULL , ?, ? )",
[username, password]
            )
            db.commit()
            return True
        except Exception as e:
            return False

def add_monitor_user(self, ip, name, admin_key):
    with sqlite3.connect(self.db_path) as db:

        curs = db.cursor()
        curs.execute(
            "INSERT INTO MonitorUsers VALUES (NULL, ?, ?, ?) " "
RETURNING key",
            [name, ip, admin_key],
        )
        row = curs.fetchone()
        (inserted_id,) = row if row else None
        db.commit()
        return [inserted_id, name, ip, admin_key]

def GetAllAdminUsers(self):
    with sqlite3.connect(self.db_path) as db:
        curs = db.cursor()
        curs.execute("SELECT * FROM adminUsers")
        rows = curs.fetchall()

    return rows

```

```
def GetAllMonitorUsers(self):
    with sqlite3.connect(self.db_path) as db:

        curs = db.cursor()

        curs.execute("SELECT * FROM 'MonitorUsers' ")
        rows = curs.fetchall()

    return rows

def get_monitor_list_by_admin_key(self, adminKey):
    with sqlite3.connect(self.db_path) as db:
        curs = db.cursor()

        curs.execute(
            """SELECT * FROM MonitorUsers WHERE
MonitorUsers.adminKey = ? """,
            [adminKey],
        )
        rows = curs.fetchall()
    return rows

def check_login(self, username, password):
    with sqlite3.connect(self.db_path) as db:
        curs = db.cursor()

        curs.execute(
            """SELECT * FROM AdminUsers WHERE
AdminUsers.username = ? AND AdminUsers.password = ? """,
            [username, password],
        )
        rows = curs.fetchall()

    if len(rows) == 0:
        return None
    else:
        return rows[0]

def delete_monitor_by_key(self, key):
    with sqlite3.connect(self.db_path) as db:
        curs = db.cursor()
```

```
curs.execute("delete from MonitorUsers where
MonitorUsers.key = (?)", [key])
db.commit()
```

קוד צד קליינט

clientUserInterface.py

```
import customtkinter as ctk
import os
import ipaddress
from CTkMessagebox import CTkMessagebox
import socket

COLOR_TEXT_RED = "#FF0000"
COLOR_BACKGROUND_BLACK = "#212121"
COLOR_BACKGROUND_GRAY = "#323232"
COLOR_ICON_BLUE = "#0D738F"
COLOR_TEXT_GREEN = "#4BE4C5"
COLOR_TEXT_WHITE = "#FFFFFF"

FONT_BUTTON = ("italic", 20)
FONT_TITLE = ("italic", 60)
FONT_NORMAL_TEXT = ("italic", 20)
MONITOR_POS_KEY = 0
MONITOR_POS_NAME = 1
MONITOR_POS_IP = 2
MONITOR_POS_ADMIN_KEY = 3

class ClientApp(ctk.CTk):
    def __init__(self):
        super().__init__(fg_color=COLOR_BACKGROUND_BLACK)
        self.geometry("350x600")
        self.title("CyberChaperone Client")
        self.my_ip_label = ctk.CTkLabel(
            master=self,
            text=f"my ip address: \n {self.get_ip()}",
```

```
        width=120,
        height=25,
        fg_color=COLOR_BACKGROUND_BLACK,
        text_color=COLOR_TEXT_GREEN,
        font=FONT_BUTTON,
        anchor="w",
        justify="left",
    )
    self.ip_label = self.PersonLabel = ctk.CTkLabel(
        master=self,
        text="Server Ip:",
        width=120,
        height=25,
        fg_color=COLOR_BACKGROUND_BLACK,
        text_color=COLOR_TEXT_GREEN,
        font=FONT_BUTTON,
        anchor="w",
        justify="left",
    )

    self.ip_entry = ctk.CTkEntry(
        master=self,
        width=200,
        height=25,
        fg_color=COLOR_BACKGROUND_GRAY,
        placeholder_text="",
    )

    self.set_button = ctk.CTkButton(
        self,
        text="Set IP Address",
        fg_color=COLOR_BACKGROUND_GRAY,
        command=lambda: self.SetIp(self.ip_entry.get()),
        width=95,
        height=95,
    )

    self.success_label = ctk.CTkLabel(
        master=self,
        text=f"",
        width=120,
        height=25,
        fg_color=COLOR_BACKGROUND_BLACK,
        text_color=COLOR_TEXT_GREEN,
```



```
        font=FONT_BUTTON,
        anchor="w",
        justify="left",
    )
    self.success_label.place(rely=0.5, relx=0.1)
    self.my_ip_label.place(rely=0.20, relx=0.32)
    self.ip_label.place(rely=0.40, relx=0.05)

    self.ip_entry.place(rely=0.40, relx=0.35)
    self.set_button.place(rely=0.8, relx=0.3)

def SetIp(self, ip):
    if not self.checkIp(ip):
        msg = CtkMessageBox(
            master=self,
            width=400,
            height=40,
            title="Error!",
            message=f"Invalid IP address",
            option_1="Ok",
            fg_color=COLOR_BACKGROUND_GRAY,
            button_color=COLOR_ICON_BLUE,
            text_color=COLOR_BACKGROUND_BLACK,
            font=FONT_NORMAL_TEXT,
        )
    return

    this_folder = os.path.dirname(os.path.abspath(__file__))
    filename = os.path.join(this_folder, f"settings.txt")
    with open(filename, "w") as file:
        file.write(ip)
    self.success_label.configure(text="successfully changed ip")

def checkIp(self, ip):
    try:
        ipaddress.ip_address(ip)
        return True
    except ValueError:
        return False

def get_ip(self):
    try:
```

```

        hostname = socket.gethostname()
        ipv4_address = socket.gethostbyname(hostname)
        return ipv4_address
    except socket.gaierror:
        return "There was an error resolving the hostname."
    except Exception as e:
        return f"An unexpected error occurred: {e}"

if __name__ == "__main__":
    app = ClientApp()
    app.mainloop()

```

client.py

```

import time
import os
import pyautogui
import socket
import time
from threading import Event, Thread, Condition
import time
from scapy.all import IP, DNS, DNSQR, sniff, UDP
import keyboard

from cryptography.fernet import Fernet

QUEUE_SIZE = 40
queue = []
condition = Condition()
stop_event = Event()
KEY = b"afcbXv_0yebyn2iJbbt_DDvQdec3f96ImNXLq-AAGT0="

class get_focused_window_thread(Thread):
    def __init__(self):
        Thread.__init__(self)

    def run(self):
        global queue

```

```

global stop_event
temp = None

while not stop_event.is_set():
    window_title = pyautogui.getActiveWindowTitle()
    if temp != window_title:
        time_string = time.strftime("%m/%d/%Y, %H-%M-%S",
time.localtime())
        msg =
f"{time_string}:[focused_window]:{str(window_title)}"
        condition.acquire()

        if len(queue) >= QUEUEU_SIZE:
            condition.wait()

        queue.append(msg)
        condition.notify()
        condition.release()
        temp = window_title

# not working to be fixed
class Port53Scan(Thread):
    def __init__(self):
        Thread.__init__(self)

    def run(self):
        global queue
        global stop_event
        msg = ""
        while not stop_event.is_set():
            try:
                pkts = sniff(filter="udp and port 53", count=5)
                for pkt in pkts:
                    time_string = time.strftime("%m/%d/%Y,
%H-%M-%S", time.localtime())
                    msg = f"{time_string}:[PORT
53]:{str(pkt[DNSQR].qname.decode())}"
            except:
                msg = ""
                pass

```

```
        if msg != "":
            condition.acquire()

            if len(queue) >= QUEUEU_SIZE:
                condition.wait()
            queue.append(msg)

            condition.notify()
            condition.release()
            msg = ""

class detect_key_press(Thread):
    def __init__(self):
        Thread.__init__(self)

    def key_decode(self, key):
        try:
            key = key.replace("'", "")

            return chr(key)
        except:
            key = key.replace("'", "")
            if len(key) > 1:
                return f"[{key}]"
            else:
                try:
                    (ord(key))
                except:
                    return key
        return key

    def run(self):
        global queue
        global stop_event
        msg = ""
        while not stop_event.is_set():

            event = keyboard.read_event()
            if event.event_type == keyboard.KEY_DOWN:
                msg = msg + self.key_decode(event.name)
```

```

        if len(msg) >= 30:
            condition.acquire()

            if len(queue) >= QUEUEU_SIZE:
                condition.wait()

            time_string = time.strftime("%m/%d/%Y, %H-%M-%S",
time.localtime())
            finalmsg = f"{time_string}:[keypress]:{msg}"
            queue.append(finalmsg)
            msg = ""
            condition.notify()
            condition.release()

class ConsumerThread(Thread):
    def __init__(self, ip, port):
        Thread.__init__(self)
        self.ip = ip
        self.port = port

    def run(self):

        global queue
        global stop_event
        f = Fernet(KEY)

        while not stop_event.is_set():
            try:
                print("attempting to connect to " + self.ip)
                s = socket.socket(socket.AF_INET,
socket.SOCK_STREAM)
                s.settimeout(5)
                result = s.connect((self.ip, self.port))
                while not stop_event.is_set() and result == None:

                    condition.acquire()

                    if not queue:
                        condition.wait()
                    m = f.encrypt(queue.pop().encode("utf-8"))

```

```
s.send(m)

        condition.notify()
        condition.release()
    except KeyboardInterrupt:
        s.close()
        break
    except Exception as e:
        print(e)
        s.close()

if __name__ == "__main__":
    thread_list = []

    this_folder = os.path.dirname(os.path.abspath(__file__))
    filename = os.path.join(this_folder, f"settings.txt")
    with open(filename, "r") as file:
        ip = file.read()
    port = 8882
    try:
        x = ConsumerThread(ip, port)
        thread_list.append(x)
        x.start()
        lst = [get_focused_window_thread, detect_key_press,
Port53Scan]

        for x in lst:
            time.sleep(1) # this somehow makes the python
interpreter not crash
            x = x()
            x.start()
            thread_list.append(x)

    except Exception as e:
        print(e)
        stop_event.set()
        for threads in thread_list:
            threads.join()
```