

אפיון פרויקט

הגנת סייבר

**מגיש: עומר כפיר (330869017) תיכון הרצוג יב'3**

**שם מנחה: אופיר שביט**

**נושא הפרויקט: פלטפורמת תצפית שקטה על פעילות עובדים בחברה**

תוכן עניינים

[**מבוא** 4](#_Toc183899203)

[**תיאור כללי** 4](#_Toc183899204)

[**מטרות ויעדים** 4](#_Toc183899205)

[**בעיות תועלות וחסכונות** 5](#_Toc183899206)

[**סקר שוק – חקירת פתרונות קיימים** 5](#_Toc183899207)

[**סקירת טכנולוגיית הפרויקט** 7](#_Toc183899208)

[**תיחום פרויקט** 7](#_Toc183899209)

[**פירוט תיאור המערכת (אפיון)** 8](#_Toc183899210)

[**תיאור מפורט של המערכת** 8](#_Toc183899211)

[**פירוט על יכולות שהיא תעניק לכל סוג משתמש** 9](#_Toc183899212)

[**פירוט יכולות המערכת** 10](#_Toc183899213)

[**בדיקות, לו"ז וסיכונים** 15](#_Toc183899214)

[**פירוט בדיקות שמתוכננות לפרויקט** 15](#_Toc183899215)

[**תכנון וניהול לוז** 17](#_Toc183899216)

[**סיכונים** 18](#_Toc183899217)

[**תיאור ארכיטקטורה המערכת** 20](#_Toc183899218)

[**תיאור החומרה** 20](#_Toc183899219)

[**מראה גרפי של קשרי החומרה** 21](#_Toc183899220)

[**תיאור טכנולוגיות רלוונטיות** 22](#_Toc183899221)

[**שפות תכנות** 22](#_Toc183899222)

[**מערכת הפעלה** 23](#_Toc183899223)

[**תחומי עניין** 26](#_Toc183899224)

[**תיאור זרימת המידע במערכת** 27](#_Toc183899225)

[**האזנה שקטה על מחשב המשתמש** 27](#_Toc183899226)

[**איסוף מידע בזמן אמת מן מחשב המשתמש** 28](#_Toc183899227)

[**שליחת מידע המשתמש אל המנהל** 29](#_Toc183899228)

[**איסוף המידע במחשב השרת ל DB** 30](#_Toc183899229)

[**הצגת המידע באופן נגיש בשרת** 31](#_Toc183899230)

[**תיאור האלגוריתמים המרכזיים בתוכנית** 32](#_Toc183899231)

[**ניסוח וניתוח של הבעיה האלגוריתמית** 32](#_Toc183899232)

[**אלגוריתמים קיימים לפתרון הבעיה** 32](#_Toc183899233)

[**סקירת הפיתרון הנבחר** 32](#_Toc183899234)

[**הפנייה למקור רלוונטי** 32](#_Toc183899235)

[**תיאור סביבת הפיתוח** 33](#_Toc183899236)

[**תיאור פרוטוקול התקשורת** 34](#_Toc183899237)

[**דוגמה להודעות:** 34](#_Toc183899238)

[**טבלת הודעות פרוטוקול** 35](#_Toc183899239)

[**תיאור מסכי המערכת** 37](#_Toc183899240)

[**מסך פתיחה (אצל השרת)** 37](#_Toc183899241)

[**מסך הגדרות** 38](#_Toc183899242)

[**מסך ראשי** 39](#_Toc183899243)

[**מסך אישי** 39](#_Toc183899244)

[**דיאגרמת מסכים** 40](#_Toc183899245)

[**תיאור מבני נתונים** 41](#_Toc183899246)

[**סקירת חולשות ואיומים** 42](#_Toc183899247)

# **מבוא**

* **ייזום**

## **תיאור כללי**

* + הפרויקט הינו פלטפורמה שמאפשרת למנהל של חברה להשגיח על פעילות לקוחותיו מרחוק ולוודא שהם מבצעים את העבודה כראוי. הפלטפורמה על המחשבים של העובדים מסתירה את עצמה כך שהעובדים בחברה לא יוכלו להסיר את התוכנית מהמחשב שלהם לבד.
  + בחרתי בפרויקט זה מכיוון שהוא אינו פשוט ויהווה לי אתגר ואף בנוסף הוא עוסק בעיקרו בעולם מערכות ההפעלה, עולם שמרתק אותי. הפרויקט דורש למידה על התנהגות של מערכת ההפעלה והתעסקות עם קוד של מערכת ההפעלה. המטלה הינה לא פשוטה ומביאה עימה מספר אתגרים: הסתרת תהליך במערכת הפעלה והאזנה לפעולות שונות מבלי פגע ויכולות של המשתמש לדעת, העברת מידע אמין למחשב המנהל אשר משמש בתור שרת, תצוגה גרפית בצורה נוחה אצל מחשב המנהל ועוד...

## **מטרות ויעדים**

* + לפרויקט מספר מטרות, חלק מן המטרות עוסקות בכל הנוגע למערכות הפעלה והכוונה להסתיר באופן מוחלט מן העובד בחברה את אותו תהליך בכדי שלא יוכל להסיר אותו/לשנות אותו, בו זמנית המנהל ישלוט באופן מלא על אותם תהליכים ויוכל לעשות בהם כרצונם. על המערכת להיות נגישה כמה שיותר למנהל החברה וביחד עם זאת להיות מוסתרת כמה שיותר מהעובד בחברה שהתוכנה רצה אצלו במחשב.

## **בעיות תועלות וחסכונות**

* + המערכת באה לחסוך זמנים בשביל מנהלים בחברות. חברות אשר ישתמשו במערכת זו יוכלו באופן נגיש להאזין למחשבים של העובדים בחברה שלהם מבלי שהעובדים יוכלו לעשות דבר כנגד לכך. העובדים לא יוכלו למצוא או לשנות את התוכנית המאזינה בכדי "לעבוד" על המנהל שלהם, המנהל יידע על שלל פעילות המחשב שלהם מבלי יוצא מן הכלל (כמובן הפרטים החשובים, לא האזנה מלאה לכל event). המערכת תפתור את הבעיה שנוצרת אצל עובדים בחברות שאינם מנצלים את זמן העבודה באופן מיטבי ואף גם לעיתים לא מבצעים את העבודה הצפויה מהם במהלך העבודה. זוהי הינה בעיה חמורה שמפסידה כסף רב לחברות, וכאן בדיוק מגיעה המערכת שלי. מבלי ידיעת העובדים המנהל יכול בכל רגע נתון להסתכל על נתוני המחשב של העובד אצלו בתוכנה, ומכך להבין אם העובד מבצע את עבודתו כראוי או שצריך להעיר לו ואף אם הדבר מתמשך לפטר את העובד מהחברה. המערכת תנגיש את המידע על הלקוחות השונים למנהל של החברה בכדי שיוכל לתצפת על עובדיו באופן נוח ויעיל, מבלי שיצטרך להתקשות עם תוכנה מסובכת, כמו שידוע כיום הרבה תוכנות בעלות פונקציונליות רבה מתקשות בלשמור על ממשק משתמש נוח וקל לשימוש, ולכן המערכת תדע להציג את המידע באופן נוח מבלי שמנהל החברה יצטרך להתקשות איתה.

## **סקר שוק – חקירת פתרונות קיימים**

* + כידוע כיום עולם אבטחת המידע הוא נושא מבין הנושאים שמעסיקים אותנו ביום יום. קיימים מאות אלפי אם לא מיליוני כלים באינטרנט לאבטחת סייבר. כלים אשר מדמים את פעולותיי של המערכת הם כלים כמו Anti-Virus/Anti-cheat. אותם כלים לרוב יהיו KLM (Kernel Loadable Module) והם יודעים להאזין לפעילויות המערכת, קלטים שונים, תהליכים כאלו ואחרים, ואף להתריע ולשלוח לאדם חיצוני בעת זיהוי סכנה. דרך פעולה זו מדמה את פעולות המערכת בפרויקט שלי, אשר המערכת יודעת לשבת בליבת מערכת ההפעלה, ולשלוח מידע לאדם חיצוני, האומנם המערכת שלי אינה מנתחת את פעילות המחשב ומתריעה רק בעת סכנה, אך היא יודעת להאזין לליבת מערכת ההפעלה ולשלוח את המידע לאדם חיצוני.

**תמונה שמכילה טקסט, צילום מסך, גופן, מספר

התיאור נוצר באופן אוטומטי**





## **סקירת טכנולוגיית הפרויקט**

* + במערכות הפעלה קיימת שונות רבה בין מערכות שונות ואפילו בין גרסאות שונות של אותה מערכת הפעלה. ועל כך, יש חשיבות ותלות רבה בסביבה בה תורץ המערכת. מכיוון שהמערכת עוסקת בלב ליבה של מערכת ההפעלה, קיים קושי בלהפוך אותה דינמית לשלל גרסאות של מערכת הפעלה, ולכן המערכת תהייה מחויבת לרוץ על אותה סביבת עבודה. (בדגש רק צד הלקוח, על צד השרת לא תקפים החוקים הללו). דבר זה יכול להתקיים על ידי הרצה של VMים כאלו ואחרים. הפרויקט יהיה מחוייב לרוץ על הגרסה בה פותח, ללא אפשרות להריץ על גרסאות אחרות של אותה מערכת הפעלה, ולכן חובה על כל מחשב שהמערכת תרוץ עליו להתאים לגרסת מערכת ההפעלה עליה הפרויקט תוכנן לרוץ.

## **תיחום פרויקט**

* + הפרויקט הולך לעסוק בשלל של נושאים שעליהם אפרט:
  + מערכות הפעלה – hook שונים של פונקציות במערכת ההפעלה, הסתרת מידע בפני תהליכי user mode, האזנות לדרייברים, ועוד...
  + רשתות – על מנת להעביר את המידע שנאסף מן מערכת ההפעלה על הפרויקט להשתמש ב socket, בדגש על שימוש בפרוטוקול TCP אשר יעביר את המידע בצורה אמינה למחשב השרת.
  + על הפרויקט להשתמש בסוגי הצפנות שונים בכדי להעביר את המידע בצורה מאובטחת לצד השרת.
  + המערכת איננה עוסקת באבטחה של מערכות הפעלה (כמו שצוין AntiVirus), שמירת סיסמאות בתוך DB ועוד...



# **פירוט תיאור המערכת (אפיון)**

## **תיאור מפורט של המערכת**

* + המערכת הינה מערכת האזנה שקטה על מחשבים במערכת הפעלה לינוקס. היא תהייה אחראית על האזנה למתודות שונות במערכת ההפעלה והסתרה שלה בתוך מערכת ההפעלה כך שלאחד העובדים בחברה שאין לו הרשאות גבוהות על אותו מחשב לא יוכל בכלל לדעת שהתוכנית נמצאת לו על המחשב, וגם אם כן מודע לכך לא תהייה באפשרותו כלל למחיקה של התוכנה והפסקת עבודתה. בזכות האזנה של המערכת למחשב העובד היא תוכל לשלוח מידע על פעילות המחשב, וכך המנהל של העובד ידע אם העובד שלו מבצע את עבודתו כראוי. המידע ישלח למחשב של המנהל, שם הוא יוכל להסתכל על המידע של כל עובד בנפרד באופן מסודר, ובכך יוכל באופן יעיל לתצפת על עובדיו מבלי שיצטרך להתאמץ ולבזבז זמן וכסף מיותר על עובדיו, המערכת מייעלת את העבודה של המנהל.

****



## **פירוט על יכולות שהיא תעניק לכל סוג משתמש**

* + היכולות העיקריות שיוסגו מן הפרויקט הינן לצד המנהל. כמו שהוסבר מקודם הפרויקט הוא פרויקט האזנה למחשבים של עובדים בחברה, ומטרת התוכנה הינה להסתתר כמה שיותר במחשבי העובדים. כלומר העובדים לא אמורים כלל להיות מודעים לנוכחות התוכנה על מחשבים, ובמידה וכן, לא יוכלו לעשות דבר בנידון. לכן בשביל צד המשתמש אין לו כלל יכולות שהפרויקט מעניק לו. בניגוד לכך, הפרויקט מעניק שלל יכולות מגוונות למנהל של העובדים. המנהל יוכל בכל רגע נתון לראות את פעילות עובדי החברה במחשביהם, כלומר, אם הוא רוצה לבדוק שעובדיו מבצעים את עבודתם כראוי, יוכל להסתכל בנתוני המערכת אצלו במחשב ולראות נתונים שונים על מחשב העובד, הנתונים יכללו בתוכם – מקשי מקלדת, לחיצות עכבר, צילומי מסך, שמע ועוד... מטרת הפרויקט כפי שהוגדרה הינה לייעל את עבודתו של מנהלים בחברות, והגישה של המנהל לנתונים נגישים בזמן אמת על עובדיו בהחלט מסייעת לכך. אם הנתונים האלו המנהל יכול להבין אילו מעובדיו מבצעים את עבודתם ואילו אינם, ומכיוון שהעובדים לא יכולים לשנות את התוכנה (או שהם בכלל לא מודעים לתוכנה) הם לא יכולים כלל לדעת מתי המנהל מסתכל על הנתונים של מחשביהם, ובכך המנהל יכול לדעת באופן חשאי פרטים על פעילות עובדיו, והאם צריך לשנות נהלים באופן העבודה או בכלל לפטר עובדים שרוב הזמן אינם עובדים לפי הוראות המנהל.

להלן פירוט מפורט יותר של היכולות.

## **פירוט יכולות המערכת**

1.האזנה שקטה על מחשב המשתמש.

2.איסוף מידע בזמן אמת מן מחשב המשתמש.

3.שליחת מידע של מחשב המשתמש אל מחשב השרת.

4.איסוף המידע במחשב השרת אל תוך DB.

5.הצגת המידע באופן נגיש במחשב השרת.

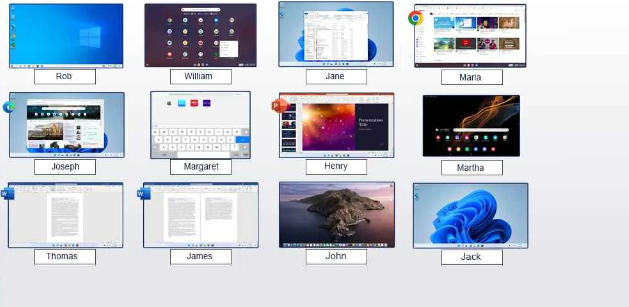
|  |  |
| --- | --- |
| האזנה שקטה על מחשב המשתמש | שם היכולת |
| האזנה שקטה על מחשב המשתמש מאפשרת לכך שאין באפשרות העובדים למחוק/לשבש דברים בתוכנה בזמן פעילותה, כלומר התוכנה תמיד תפעל ואך ורק המנהל הוא בעל היכולת לשנות דברים בתוכנה. בכך המנהל יכול בכל זמן נתון לתצפת על עובדיו מבלי שיוכלו "לעבוד" על התוכנית ולגרום לה לשלוח מידע לא אמין. | מהות היכולת |
| **אצל הלקוח**   * ביצוע hookים שונים על פעולות מבלי לפגוע במהות הפעולות * הסתרת התהליך ב user-mode * האזנה "קלה" כך שאינה משפיע על ביצועי המערכת של מחשב המשתמש | אוסף פעולות/יכולות הנדרשות ליכולת הנ"ל |
| * KLM – Kernel Loadable Module, כלומר תהליך בהרשאות הגבוהות ביותר במחשב – קרנל * מחשב לקוח (עם מערכת הפעלה לינוקס) * הרשאות של המנהל על מחשב הלקוח | אובייקטים נחוצים |

|  |  |
| --- | --- |
| איסוף מידע בזמן אמת מן מחשב המשתמש | שם היכולת |
| איסוף מידע בזמן אמת מן המחשב של המשתמש מאפשרת ליכולות אחרות בפרויקט זה, אסיפת המידע היא יכולת קריטית אצל מחשבי המשתמשים, והיא חלק גדול מפרויקט זה. איסוף המידע בזמן אמת יאפשר למנהל לתצפת על עובדיו בזמן אמת (ולא רק לקבל סיכום), כלומר בכל רגע נתון המנהל בעל יכולת לבדוק את עובדיו ואת עבודותיהם. | מהות היכולת |
| **אצל הלקוח**   * ביצוע hookים שונים על פעולות מבלי לפגוע במהות הפעולות * שמירת המידע בצורת טקסט שניתן לקרוא ולהבין בצורה מונגשת * יצירת סדר במידע, כל פעולה שונה במחשב המשתמש עליה להיות מסודרת לפי סוג הפעולה. | אוסף פעולות/יכולות הנדרשות ליכולת הנ"ל |
| * KLM – Kernel Loadable Module, כלומר תהליך בהרשאות הגבוהות ביותר במחשב – קרנל * מחשב לקוח * קבצים/באפרים לאחסון המידע באופן זמני | אובייקטים נחוצים |

|  |  |
| --- | --- |
| שליחת מידע של מחשב המשתמש אל מחשב השרת | שם היכולת |
| שליחת מידע של מחשב המשתמש אל מחשב השרת מאפשרת לשרת/מנהל לקבל את המידע על עובדיו ובכך מקשר את מחשבי העובדים למחשב השרת. המידע שנאסף על המחשבים נשלח בזמן אמת אל השרת ובכך התוכנית יכולה לבצע יכולות רבות על המידע שמתקבל מן המשתמשים. | מהות היכולת |
| **אצל הלקוח**   * תוכנית שמאזינה לפעולות הלקוח. * שליחת המידע מן מחשב המשתמש לשרת באופן מוצפן   **אצל השרת**   * איסוף המידע. * קבלת המידע אצל מחשב השרת ופענוח המידע. | אוסף פעולות/יכולות הנדרשות ליכולת הנ"ל |
| * מחשב לקוח * Socket * מחשב שרת * קבצים/באפרים לאחסון המידע באופן זמני | אובייקטים נחוצים |

|  |  |
| --- | --- |
| איסוף המידע במחשב השרת אל תוך DB | שם היכולת |
| איסוף המידע במחשב השרת אל תוך DB מאפשר למחשב השרת לשמור את המידע הנשלח מן הלקוחות בזמן אמת ולהפריד אותו לכל לקוח בנפרד. כלומר כל מחשב של עובד בחברה יהיה שמור בתוך ה DB של כלל העובדים, כך שיהיה ניתן להפריד בין העובדים השונים במערכת. | מהות היכולת |
| **אצל השרת**   * איסוף המידע מן הלקוח על ידי פענוח המידע המוצפן. * פענוח המידע לפי הפרוטוקול * הכנסה ל DB את הערכים של הלקוח ממנו נשלחה ההודעה | אוסף פעולות/יכולות הנדרשות ליכולת הנ"ל |
| * מחשב לקוח * Socket * מחשב שרת * טבלת DB אצל השרת | אובייקטים נחוצים |

|  |  |
| --- | --- |
| הצגת המידע באופן נגיש במחשב השרת | שם היכולת |
| הצגת המידע באופן נגיש במחשב השרת מאפשרת למנהל של העובדים באופן יעיל להסתכל על המידע של מחשבי העובדים ובכך לקבוע את יעילות עבודתם ואם הם עובדים כראוי או שאינם מבצעים את עבודתם כלל, ולכן חשוב להציג את המידע באופן מסודר כך שהמנהל יוכל להסתכל בנוחות ובזריזות על המידע. | מהות היכולת |
| **אצל השרת**   * איסוף המידע מן הלקוח על ידי פענוח המידע המוצפן. * פענוח המידע לפי הפרוטוקול * הכנסה ל DB את הערכים של הלקוח ממנו נשלחה ההודעה * הצגה באופן ויזואלי על ידי GUI את נתוני ה DB בזמן אמת | אוסף פעולות/יכולות הנדרשות ליכולת הנ"ל |
| * מחשב לקוח * Socket * מחשב שרת * טבלת DB אצל השרת * GUI במחשב השרת | אובייקטים נחוצים |



# **בדיקות, לו"ז וסיכונים**

## **פירוט בדיקות שמתוכננות לפרויקט**

* + בפרויקט הזה אבצע מספר בדיקות כדי לוודא שהמערכת מתפקדת כפי שתוכננה. הבדיקות יתמקדו בהעברת נתונים וביכולת המעקב אחר פעולות המשתמש, תוך שמירה על יציבות המערכת. ביצוע הבדיקות הינו שלב הכרחי בפיתוח הפרויקט שבלעדיו לא ניתן להכריע אם הפרויקט עובד כראוי

**1.בדיקת העברת נתונים לשרת המרכזי**

|  |  |
| --- | --- |
| מטרה | לבדוק שהנתונים מהמחשב של העובד מועברים לשרת של המנהל באופן אמין |
| איך אבדוק | אגדיר שרת מדומה כ"מנהל" ואעקוב אחרי זרימת הנתונים ממחשבי העובדים. אשתמש בכלים לבדיקת חבילות רשת כדי לוודא שהמידע מגיע בשלמותו ובתזמון הנכון. |

**2. בדיקת עקביות הנתונים**

|  |  |
| --- | --- |
| מטרה | לוודא שהנתונים הנאספים ממחשבי העובדים משקפים את הפעולות שבוצעו במדויק. |
| איך אבדוק | אבצע פעולות מוגדרות מראש במחשב העובד, כמו פתיחת תוכנות ועבודה על מסמכים, ואשווה את הנתונים שנשלחו לשרת לאירועים בפועל. כך אוכל לראות שהמערכת עוקבת בצורה מדויקת אחר הפעולות. |

**3.בדיקת השפעה על ביצועי המערכת**

|  |  |
| --- | --- |
| מטרה | לבדוק שהמערכת לא פוגעת בביצועים של מחשבי העובדים |
| איך אבדוק | אבצע מדידת ביצועים (CPU, זיכרון) לפני התקנת המערכת ואחריה, ואשווה את התוצאות. כך אוכל לוודא שהמערכת לא מכבידה על המשאבים של המחשב ומאפשרת עבודה חלקה. |

**4. בדיקת אבטחת הנתונים**

|  |  |
| --- | --- |
| מטרה | לוודא שהנתונים המועברים לשרת מוגנים ולא חשופים לגורמים חיצוניים. |
| איך אבדוק | אשתמש בהצפנה להעברת המידע ואבחן פרוטוקולי תקשורת מאובטחים (AES RSA) על מנת לוודא שהנתונים נשארים חסויים.  אני מתכנן לבצע כל בדיקה עם רישום מפורט של התוצאות, כך אוכל להבין איפה יש צורך בשיפור ולהציג את הממצאים כחלק מהפרויקט החינוכי. |

## **תכנון וניהול לוז**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **מספר** | **פעילות** | **תאריך יעד** | **תאריך בפועל** | **הערות** |
| **1** | הצעת רעיון פרויקט גמר | 17.10.2024 | 14.10.2024 | הצעת הפרויקט הינה פרויקט זה, לא התבצעו שינויים ברעיון הפרויקט המקורי |
| **2** | הצעה לפרויקט גמר | 30.10.2024 | 24.10.2024 | תכנון ראשוני של הפרויקט, הכלים והדרך ביצוע |
| **3** | מסמך אפיון – הגשת ביניים ראשונה | 10.11.2024 | 6.11.2024 | הגשה חלקית של אפיון הפרויקט |
| **4** | מסמך אפיון – הגשת ביניים שנייה | 20.11.2024 | דרוש לשנות | הגשה חלקית של אפיון הפרויקט |
| **5** | מסמך אפיון – הגשה סופית | 30.11.2024 | דרוש לשנות | הגשה סופית של אפיון הפרויקט המלא |
| **6** | פיתוח הפרויקט | 31.3.2025 | דרוש לשנות | פיתוח מלא של הפרויקט |
| **7** | תיק פרויקט | 30.4.2025 | דרוש לשנות |  |
| **8** | הגשת פרויקט גמר | --NULL-- | דרוש לשנות |  |



## **סיכונים**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **הסיכון** | **תיאור הסיכון** | **רמת סיכון** | **תיאור דרכי התמודדות** | **מה בוצע בפועל** |
| אי עמידה בלוח הזמנים | הגשה מאוחרת וחוסר עקביות בלוח זמנים המתוכנן. אי עמידה בלוח הזמנים משפיע על כל תהליך הפרויקט | קל | לדאוג לעקוב אחר הלוח זמנים. | דרוש לשנות |
| המידע שמועבר אינו מוצפן | המידע שמועבר מן הלקוחות לשרת אינו מוצפן ולכן אנשים שאינם חלק מהמערכת יכולים לפענח את ההודעות ולהשתמש בהם לטובתם | קשה | להשתמש בהצפנות שונות, הצפנה אסימפטרית להעברת המפתחות בשביל הצפנה סימטרית בשביל העברת ההודעות | דרוש לשנות |
| ממשק משתמש לא יציב ולא נוח לשימוש | ממשק המשתמש GUI אינו נוח לשימוש מה שמקשה רבות על השימוש בפרויקט מצד השרת. | בינוני | שינוי ממשק המשתמש כך שגם מי שאינו מבין את תוכן הפרויקט יוכל להבין איך להשתמש בממשק המשתמש, בדיקה כנגד אנשים חיצוניים | דרוש לשנות |
| קריסת השרת | השרת נפל בשל סיבה לא ידועה כזו או אחרת | קשה | שימוש באמצעי זהירות בכדי למנוע את קריסתו של השרת (try/except), מזעור של קריסת השרת כך כאשר באמת יקרוס השרת כבר זו תהיינה בעיה חיצונית שהפרויקט לא אחראי עליה | דרוש לשנות |
| עומס יתר על השרת | השרת מוצף בלקוחות ומנהל מול מספר רב של לקוחות session מה שמכביד עליו ויקשה על מחשב השרת לתפקד כראוי | קשה | הגבלתם של מספר הלקוחות שיכולים להתחבר לשרת, מספר קבוע מקסימלי שיקבע מספר גג של לקוחות שיכולים בו זמנית לנהל session מול השרת | דרוש לשנות |
| DB של הלקוחות גדול מדי | ה DB שהשרת שומר בזמן אמת על לקוחותיו יגדל בצורה כה משמעותית מה שיכביד על השרת וביצועיו | קשה | מחיקת מידע אצל כל משתנה כאשר כמות המידע השמורה אצלו בטבלה עברה מכסה מסוימת שנקבעה מראש | דרוש לשנות |
| הלקוח משנה את המערכת אצלו במחשב | הלקוח הצליח לשנות את המערכת ששולחת מידע מהמחשב אל השרת, מה שיפגע בתפקוד הפרויקט | קשה | הסתרת התוכנית כך שתעבוד מאחורי הקלעים, עבודה עם ה kernel בכדי להחביא את התוכנית מ user space. בנוסף תדאג להסתרת כל ביצועיה, שליחת המידע למחשב חיצוני גם כן הוא יוסתר על ידי התהליך. | דרוש לשנות |
| תקשורת לא אמינה | הלקוח והשרת מתקשרים באופן לא אמין, לא כל המידע שהלקוח שולח מגיע בשלמותו אל השרת | קשה | שימוש בפרוטוקול אמין TCP על מנת להבטיח שהמידע שיועבר יגיע בצורה אמינה. | דרוש לשנות |
| מכונה וירטואלית אינה מקבלת מספיק משאבים מהמערכת | המשאבים שניתנו למכונה הוירטואלית עליה רץ הפרויקט לא מספיקים, ולכן מתקשה המכונה לעבוד לפי צרכי הפרויקט | בינוני | נתינה של מספיק משאבים לכל מכונה וירטואלית (RAM, CPU,NETWORK,  STORAGE) | דרוש לשנות |



# **תיאור ארכיטקטורה המערכת**

## **תיאור החומרה**

בפרויקט שלי ישנם רכיבים שונים שפועלים ביחד על מנת להביא את הפרויקט לעבוד בצורה מלאה ולאפשר למנהלים שונים לעקוב אחרי עבודת עובדיהם ועל אמינות עבודתם. ראשית נגדיר את מחשבי הלקוחות, אשר מהווים את אבני היסוד של הפרויקט. הלקוחות ישלחו את המידע אל מחשב השרת, כלומר אל מחשב המנהל. מחשב המנהל ימצא תחת אותה רשת עם מחשבי הלקוחות. המנהל יקבל את המידע מן הלקוחות דרך הרשת, המידע יועבר בצורה מוצפנת, הן בשימוש הצפנה אסימטרית להעברת המפתח והן בשימוש בהצפנה סימטרית בכדי להעביר מידע מוצפן עם המפתחות שהוחלפו בעזרת ההצפנה האסימטרית. מחשבי הלקוחות מתחברים לרשת באמצעות כרטיסי רשת המאפשרים להם לשלוח ולקבל נתונים בצורה מאובטחת. כל מחשב לקוח כולל מערכת לניהול הצפנה, אשר מבטיחה שהמידע המועבר לא ייחשף לגורמים בלתי מורשים. חיבור הרשת בין הלקוח למנהל מתבצע דרך פרוטוקול TCP מאובטח, המבטיח שהנתונים יישלחו בצורה יציבה ומוגנת. מחשב השרת, שממוקם תחת אותה רשת, מקבל את המידע המוצפן מהלקוחות ומעבד אותו בצורה מאובטחת. המחשב מצויד במערכת לניהול הצפנה המפענחת את המידע שהתקבל. לאחר מכן, השרת מציג את המידע בצורה גרפית למנהלים בממשק משתמש נוח וברור, תוך שמירה על הצפנה ואבטחת המידע לאורך כל התהליך. המנהל יכול לעקוב אחרי המידע בצורה ברורה ומסודרת, תוך שמירה על פרטיות הנתונים. לצורך העברת המידע ברשת, ישנו גם תפקיד חשוב של רכיבי הרשת, כגון נתבים ומתגים, המוודאים שהחבילות המוצפנות יגיעו בצורה תקינה ממחשב הלקוח אל מחשב השרת. הנתבים ומתגים אחראים על ניתוב המידע ומוודאים שהמידע יגיע ליעדו הנכון.

## **מראה גרפי של קשרי החומרה**

## **תיאור טכנולוגיות רלוונטיות**

### **שפות תכנות**

שפת תכנות C:

שפת C היא הבחירה המרכזית לפיתוח תוכנות ברמת ליבת מערכת ההפעלה. השפה מאפשרת גישה ישירה למשאבי המערכת ול-API של הליבה, מה שמעניק שליטה מלאה על תהליכים, זיכרון ורכיבי מערכת קריטיים. בעזרת C ניתן לבצע פעולות כמו עבודה עם מצביעים, ניהול זיכרון ישיר ותקשורת עם רכיבי חומרה או חלקים אחרים של המערכת.

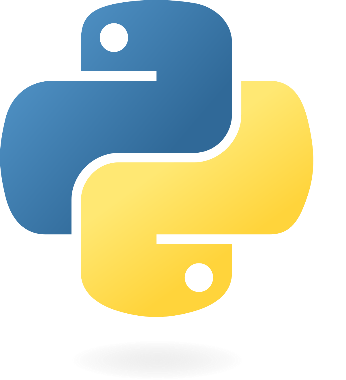
שפת תכנות Python:

Python יכולה לשמש ככלי עזר לפיתוח סביבת העבודה של הפרויקט. היא אידיאלית לאוטומציה של בדיקות, איסוף נתונים ויצירת ממשקים לתקשורת עם רכיבי המערכת. Python מצטיינת בפשטות שלה וביכולות המתקדמות שלה בטיפול ברשתות, ניתוח קבצים ולוגים, ובניהול מהיר של משימות מורכבות.

שפת תכנות SQL:

בנוסף לשפות התכנות, SQLהיא כלי מרכזי לניהול ואחסון נתונים בצורה מסודרת. היא מאפשרת ליצור מסדי נתונים, לשמור מידע, ולעבוד איתו בקלות דרך שאילתות מובנות. בעזרת SQL ניתן לארגן מידע בטבלאות עם מבנה ברור, להגדיר קשרים בין נתונים שונים, ולשלוף בדיוק את המידע הדרוש בצורה מהירה ויעילה. השפה מתאימה לניהול כמויות גדולות של נתונים ומשמשת בסיס לרוב מערכות אחסון הנתונים המודרניות.

השילוב של C, Python ו-SQL נותן לי גמישות עצומה בעבודה שלי. כל אחת מהשפות מביאה יתרונות ייחודיים, והעבודה ביניהן מאפשרת לי ליצור מערכת חזקה ומותאמת אישית. שפת C מאפשרת לי לעבוד ישירות עם המערכת, עם שליטה מלאה על הזיכרון, התהליכים והמשאבים של המחשב. אני יכול לכתוב קוד יעיל שמבצע את הפעולות הקריטיות באופן מהיר ומדויק. Python, לעומת זאת, נותנת לי דרך פשוטה ואינטואיטיבית ליצור סקריפטים שמנהלים תקשורת עם חלקי המערכת, מעבדים נתונים או מבצעים בדיקות. היכולת שלה לעבוד עם ספריות חזקות לרשתות ולעיבוד נתונים מקלה עליי לפתח חלקים מורכבים בצורה מהירה יותר. SQL

 משתלבת כדי לאפשר לי לשמור נתונים בצורה מסודרת ולשלוף אותם בקלות לפי הצורך. במקום להתעסק עם אחסון נתונים גולמיים, אני יכול להשתמש במבנה של מסדי נתונים כדי לשמור על סדר ולייעל את העבודה שלי. השילוב בין השפות האלה מאפשר לי לעבוד עם שכבת הליבה, ולאחסן את המידע בצורה מאורגנת ונגישה. כל שפה תורמת לתפקיד שונה, וביחד הן יוצרות מערכת מאוזנת תמונה שמכילה גרפיקה, עיגול, צילום מסך, לוגו

התיאור נוצר באופן אוטומטיויעילה.

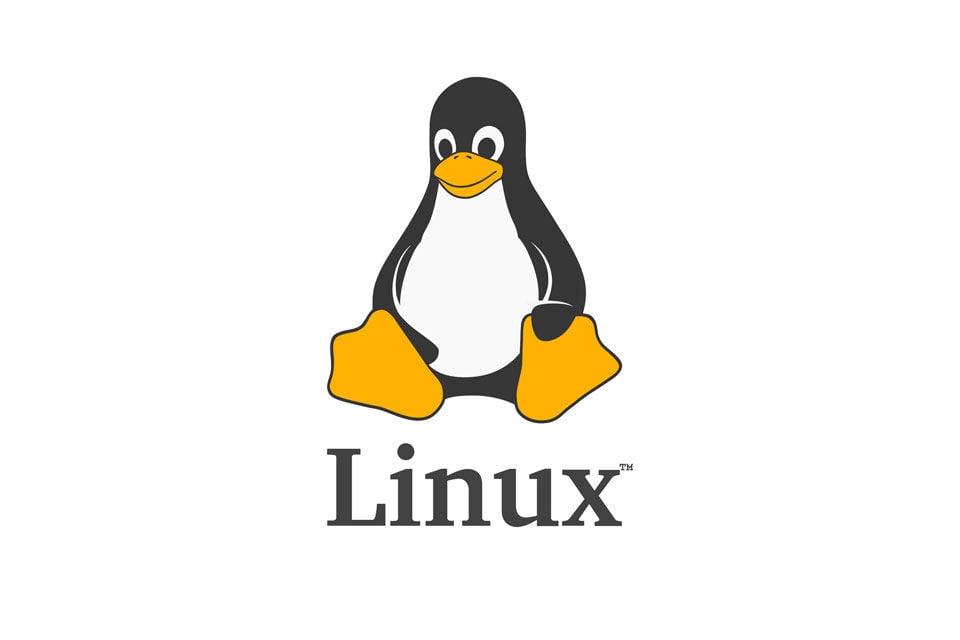
### **מערכת הפעלה**

לינוקס (Linux) היא מערכת הפעלה מבוססת קוד פתוח, כלומר, הקוד שלה זמין לציבור וניתן לשנות אותו, להפיץ אותו ולהשתמש בו ללא מגבלות מסחריות. זו אחת הסיבות לכך שלינוקס נפוצה מאוד בקרב מפתחים, חוקרי אבטחת מידע וסטודנטים שמחפשים ללמוד ולהתנסות במערכת הפעלה ברמה עמוקה.

לינוקס בנויה במודל מודולרי, כך שהרכיבים שלה מחולקים לגרעין ספריות, ותוכנות מערכת. הגרעין אחראי על אינטראקציה עם החומרה, ניהול תהליכים, זיכרון ומערכות קבצים. העובדה שהקוד של הגרעין זמין מאפשרת למפתחים לחקור אותו, להבין את אופן הפעולה של המערכת, ואפילו לכתוב תוספות או שינויים כמו דרייברים.

מכיוון שלינוקס ניתנת להתאמה אישית מלאה, קל יחסית ליצור פרויקטים כמו הפרויקט שלי. ניתן לשנות את קוד הגרעין כדי להוסיף פונקציונליות או להתקין מודול קרנל KLM שמבצע פעולות מתקדמות, כמו גישה לזיכרון, ניהול תהליכים, או עקיפה של בקרות גישה. בנוסף, קהילת המפתחים של לינוקס גדולה, ויש תיעוד רב שיכול לסייע בלמידה ובהתמודדות עם אתגרים טכניים.הגמישות והנגישות של לינוקס הופכות אותה לפלטפורמה אידיאלית לפרויקטים חינוכיים ומחקריים, במיוחד בתחום אבטחת המידע והסייבר.

מכונה וירטואלית היא סביבה שמדמה מחשב עצמאי בתוך מחשב פיזי. בעזרתה ניתן להריץ מערכת הפעלה (כמו לינוקס) בתוך מערכת הפעלה אחרת, כך שיש בידנו סביבה מבודדת לחלוטין, שמנצל את המשאבים של המחשב הפיזי כמו מעבד, זיכרון ואחסון, ומחלק אותם בין מכונות וירטואליות שונות. בפרויקטים שמצריכים עבודה על פיתוח ברמה נמוכה, כמו שינוי גרעין המערכת או עבודה עם מודולים קרנל, מכונה וירטואלית חשובה מאוד. היא מאפשרת ניסוי עם הקוד בסביבה מבודדת, כך שאין חשש לשבש את מערכת ההפעלה הראשית או את המחשב הפיזי. זה גם מאפשר ביצוע בדיקות וחקירות, תוך שמירה על הסביבה האמיתית מפני נזקים אפשריים.

**תקשורת**

התקשורת בפרויקט זה היא חלק מרכזי ומהותי בהצלחתו. המטרה היא לאפשר העברת מידע בין המערכת שבה רץ הקוד לבין מחשב אחר בצורה יעילה, אמינה ובטוחה. התקשורת מהווה את הגשר בין המידע שנאסף לבין היכולת לנתח ולהשתמש בו, ולכן תכנון נכון של מנגנון זה הוא חיוני. בפרויקט מתוכנן שימוש בפרוטוקול TCP (Transmission Control Protocol) לצורך התקשורת.

TCP נבחר בשל היתרונות המשמעותיים שלו באמינות ובניהול מסרים. זהו פרוטוקול שמבטיח שכל המידע הנשלח מגיע ליעדו בדיוק כפי שנשלח, ובסדר הנכון. אמינות זו קריטית במיוחד במערכת שמטרתה להעביר נתונים מדויקים. TCP פועל באמצעות יצירת חיבור מבוקר בין שני מחשבים, תהליך המתחיל ב"לחיצת יד משולשת" (Three-Way Handshake) שמבטיחה ששני הצדדים מוכנים לתקשורת. תהליך זה מאפשר גם התמודדות עם בעיות כמו אובדן נתונים או הפרעות בתקשורת, כיוון שהפרוטוקול כולל מנגנוני תיקון וידוא.

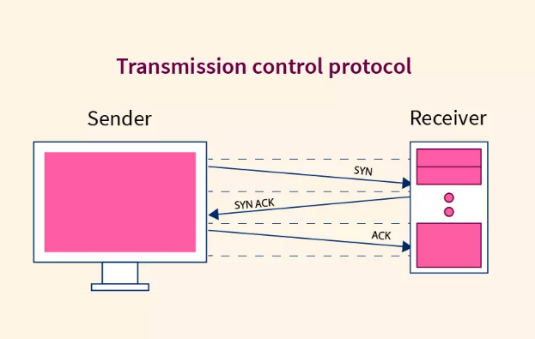
מרכיב חשוב נוסף בתקשורת בפרויקט הוא שימוש בנתונים מוצפנים. הצפנה היא כלי מרכזי להבטחת אבטחת המידע שנשלח, ובמיוחד כאשר מדובר בתקשורת שעשויה לכלול מידע רגיש או קריטי. ההצפנה מגנה על הנתונים מפני גישה לא מורשית, כך שגם אם צד שלישי מצליח ליירט את התקשורת, הוא לא יוכל לפענח את המידע שנשלח.

בעת תכנון הפרויקט, יינתן דגש לשילוב מנגנוני הצפנה חזקים, כמו שימוש באלגוריתמים סטנדרטיים כגון AES או RSA, כדי להבטיח רמת אבטחה גבוהה. הנתונים המוצפנים יישלחו במסגרת חיבור ה-TCP, כך שנוכל ליהנות גם מאמינות הפרוטוקול וגם מהגנה על המידע.

שילוב הצפנה בתקשורת מבטיח שהמערכת לא רק פועלת ביעילות אלא גם משמרת את אבטחת המידע. האיזון בין אמינות, יעילות ובטיחות הוא קריטי להצלחת הפרויקט, והתכנון המוקפד של התקשורת נועד להשיג מטרה זו בצורה מיטבית.

הפרויקט מתוכנן לפעול בתוך רשת מקומית. רשת מקומית היא מערכת תקשורת המחברת מספר מחשבים ומכשירים אחרים בתוך שטח גיאוגרפי מוגבל, כמו בית, משרד או מוסד חינוכי. ה-LAN מתאפיינת במהירות תקשורת גבוהה יחסית וביכולת לשלוט ברשת באופן ישיר, מה שהופך אותה לפתרון אידיאלי לפרויקט זה.

הבחירה לפעול בתוך רשת LAN נובעת ממספר סיבות: ראשית, העבודה ברשת מקומית מפחיתה את התלות באינטרנט החיצוני, מה שמאפשר גישה מהירה ואמינה יותר בין המחשבים ברשת. שנית, LAN מספקת רמה מסוימת של פרטיות, שכן התקשורת נשארת בתוך הרשת הפנימית ואינה נחשפת לרשתות חיצוניות שעלולות להיות פגיעות יותר להתקפות. בפרויקט זה, התקשורת תתבצע רק בין מחשבים הנמצאים באותה רשת מקומית, מה שמבטיח יעילות וביצועים מיטביים. הגבלה זו גם מפשטת את תהליך ההגדרה והניהול של התקשורת, כיוון שאין צורך להתמודד עם מורכבויות כמו ניתוב דרך כתובות IP חיצוניות או פתיחת פורטים בנתב. חשוב לציין כי בעוד שה-LAN מספקת יתרונות רבים מבחינת מהירות ואמינות, היא גם מציבה מגבלות – הפרויקט לא יוכל לתקשר עם מחשבים מחוץ לרשת המקומית. עם זאת, מגבלה זו היא חלק מהתכנון ומטרתה להבטיח פשטות, אבטחה ושליטה טובה יותר על המערכת.



### **תחומי עניין**

הפרויקט עוסק במגוון תחומי עניין טכנולוגיים המשלבים ידע תאורטי עם יישום מעשי. להלן תחומי העניין המרכזיים בפרויקט:

רשתות מחשבים ותקשורת נתונים

תכנון מנגנוני התקשורת בפרויקט נעשה תוך שימוש בפרוטוקול TCP, שמספק אמינות וניהול מסרים יעיל. נושא זה כולל הבנה מעמיקה של ניהול חיבורים, טיפול בהפרעות ושמירה על תקשורת תקינה בין רכיבי המערכת.

אבטחת מידע והצפנה

אחד ההיבטים המרכזיים בפרויקט הוא הגנה על הנתונים המועברים בזמן אמת באמצעות הצפנה. שילוב אלגוריתמים חזקים כמו AES או RSA מבטיח הגנה מפני גישה לא מורשית ושמירה על פרטיות המידע, גם אם הוא מיירט על ידי צד שלישי.

מערכות הפעלה וניהול משאבים

פיתוח מודול ליבה בלינוקס מחייב ידע מתקדם במבנה מערכת ההפעלה וניהול משאבים. התחום עוסק בשימוש יעיל במנגנונים כמו קריאות מערכת, ניהול זיכרון ותיאום בין תהליכים בזמן אמת.

תכנות ושפות מחשב

הפרויקט עושה שימוש בשפות כמו C, המאפשרות גישה ישירה למשאבים ושיפור ביצועים. תחום זה כולל כתיבת קוד ברמת ליבה תוך תשומת לב לאופטימיזציה ופתרון בעיות בזמן ריצה.

תכנון מערכות מורכבות

הפרויקט משלב בין מספר תחומי טכנולוגיה, מה שמחייב תכנון מקיף וחשיבה אסטרטגית. המערכת מתוכננת כך שתהיה יעילה, מאובטחת וקלה לתפעול, תוך שילוב פתרונות יצירתיים לאתגרים הנדסיים. תצוגת נתונים בזמן אמת

אחד ההיבטים החשובים בפרויקט הוא הצגת הנתונים הנאספים באופן גרפי ברור ואינטואיטיבי בזמן אמת, מבלי לשמור את המידע לאחר הריצה. המידע מוצג בממשק משתמש ידידותי, שמאפשר ניטור והבנה מיידיים של פעולת המערכת. תחום זה כולל שימוש בטכנולוגיות גרפיות ותכנון חוויית משתמש, תוך התחשבות על גמישות והתאמה למשתמשים.

תחומי העניין הללו מאפשרים ליצור פרויקט עשיר ומאתגר, שמחבר בין ידע טכנולוגי מתקדם לבין יישומים מעשיים בתחום מערכות ההפעלה, אבטחת המידע והנדסת התוכנה.

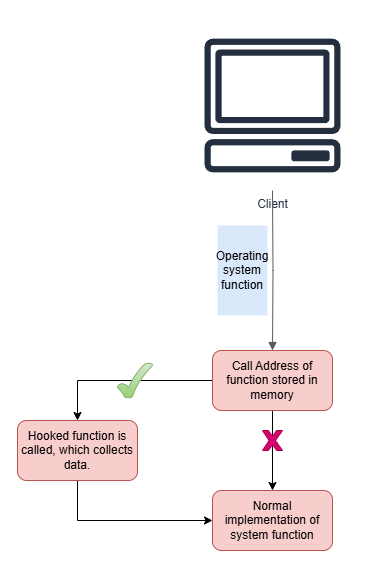
## **תיאור זרימת המידע במערכת**

### **האזנה שקטה על מחשב המשתמש**

תיאור הגרף:

גרף זה מציג את היכולת "האזנה שקטה על מחשב המשתמש". יכולת זו יכולה אך ורק לבוא לידי ביטוי על ידי שיתוף פעולה עם מערכת ההפעלה. ההאזנה השקטה מתבצעת באפשרות החבאת מידע מן המשתמש, שדבר זה אפשרי אך ורק בעזרת קוד מערכת הפעלה. דרך נגישה לעשות זאת מבלי לשנות את קוד מערכת ההפעלה עצמה (פעולה אפשרית, אך דורשת מכל מחשב שרוצה להריץ את פרויקט זה לשנות קוד במערכת ההפעלה שלו. כלומר פעולה ארוכה ולא יעילה, וגם מסוכנת אשר מדובר פה במערכת הפעלה, שהיא התוכנית הכי רגישה על המחשב, וכל שינוי מיותר בה יכול לגרור לסיכונים לכלל התנהלות המערכת) הינה על ידי השתמשות ב Kernel loadable module במערכת הפעלה לינוקס. לינוקס כמערכת הפעלה מאפשרת להוסיף אליה קוד אף תוך כדי זמן ריצה (מפה מגיעים המילים loadable module), הוספת אותו קוד הינו כלי חזק אשר התוכנית רצה בזיכרון של מערכת ההפעלה ובעלת גישה לנתונים רבים שאינם נגישים מהמשתמש. כפי שאפשר לראות בגרף, אפליקציות המשתמש יכולות רק לתקשר עם הקרנל, אך לא בעלות שינוי שלו. הוספת מודול משלנו למערכת הפעלה מאפשר לנו לשנות את הדרך ואת המידע שאליו אפליקציות המשתמש חשופות אליו.

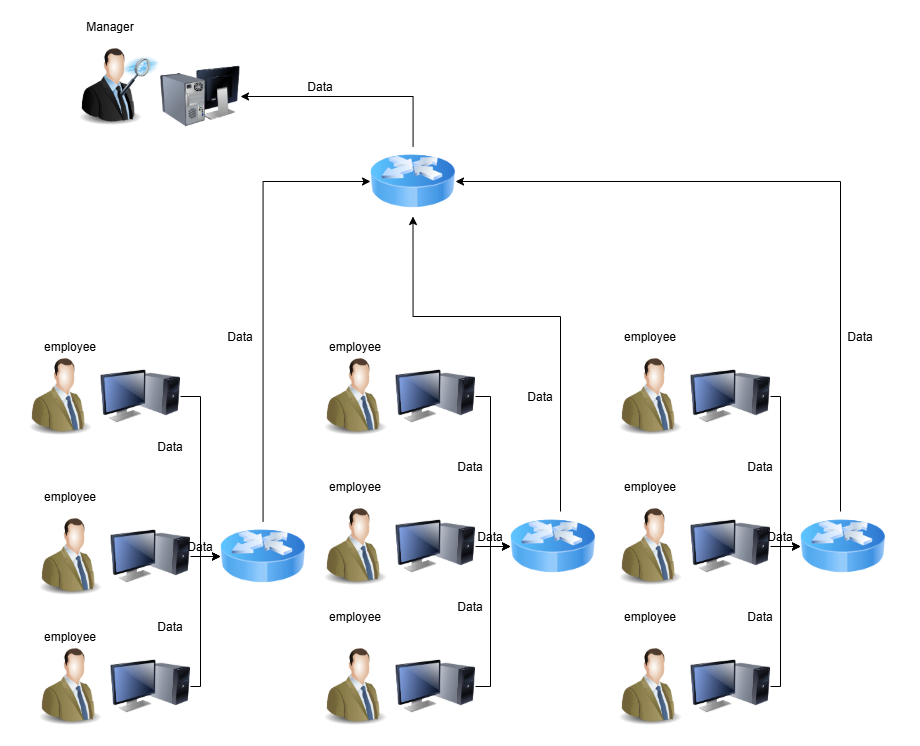
### **איסוף מידע בזמן אמת מן מחשב המשתמש**



תיאור הגרף:

גרף זה מציג את היכולת "איסוף מידע" במחשבי המשתמשים. איסוף המידע מתבצע באמצעות מונח הנקרא Hooking, מונח זה הינו תהליך שבו אנו יוצרים ניתוב מחדש של תהליך זרימת הקוד ולמעשה מנתבים אותו דרך פונקציות שלנו שבעזרתן נוכל להחליט לבצע האזנה בסתר על המידע המועבר במערכת הפעלה כחלק מקריאת הפונקציות. כמו שניתן לראות גם לאחר שינוי הפעולה בכדי שנוכל להאזין, הפעולה המרכזית עדיין נקראת, כלומר לא התבצעה פגיעה בפעולת מערכת, אלא רק האזנה מאחורי הקלעים.

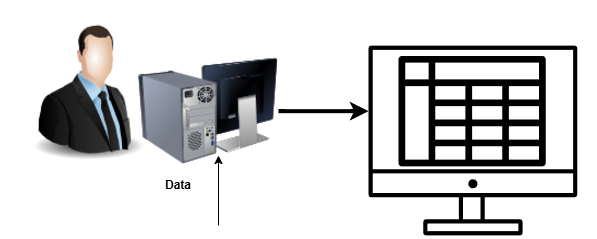
### **שליחת מידע המשתמש אל המנהל**



תיאור הגרף:

גרף זה מציג את היכולת "שליחת מידע מהמשתמשים אל המנהל". השליחה מתבצעת על ידי איסוף המידע מהמחשבים של העובדים והעברתו דרך רכיבי אינטרנט. כל עובד מחובר דרך הרשת לרכיב אינטרנט, אשר מרכז את המידע מכל תחנות העבודה. לאחר מכן, המידע נשלח ישירות למחשב של המנהל, שם הנתונים נאספים ועוברים עיבוד לצורך ניתוח או הפקת דוחות. פעולה זו מאפשרת לארגון לרכז את המידע במקום אחד ולהבטיח שהמנהל מקבל גישה למידע חיוני בזמן אמת.

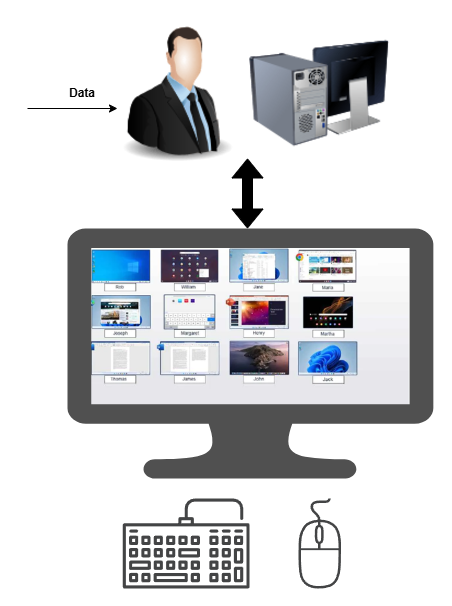
### **איסוף המידע במחשב השרת ל DB**



תיאור הגרף:

גרף זה מציג את היכולת "איסוף המידע במחשב השרת ל DB". איסוף המידע מתבצע על ידי שליחת נתונים מהמחשבים המקומיים של העובדים אל השרת (המנהל), שם הוא שומר את המידע בצורה מסודרת בתוך DB, ה DB יפריד בין המידע שהוא מקבל מהמשתמשים, וכך יוכל המנהל להבחין בין סוגי המידע שנשלחים אליו. תהליך זה מתחיל באיסוף הנתונים, מהתוכנות או מהפעילות של המשתמשים במחשבים שלהם. הנתונים מועברים דרך רכיב אינטרנט באופן מאובטח, ולאחר מכן מגיעים לשרת המרכזי, שם הם עוברים עיבוד ראשוני ונשמרים בטבלאות מסודרות בבסיס הנתונים. תהליך זה מאפשר לארגון לנהל כמויות גדולות של מידע בצורה יעילה, להבטיח אחסון מאובטח של הנתונים, ולספק גישה מהירה לצורך ניתוח, קבלת החלטות או שיפור תהליכים עסקיים.

### **הצגת המידע באופן נגיש בשרת**



תיאור הגרף:

גרף זה מציג את היכולת "הצגת המידע באופן נגיש במחשב השרת". תהליך זה מתאר את הדרך שבה הנתונים שנשמרו בשרת מוצגים בצורה נוחה וקלילה למשתמשים. המידע שנשמר בבסיס הנתונים במחשב השרת מוצג בצורה מאורגנת על מסך, למשל כטבלה או גרף, כך שהמשתמשים יכולים לגשת אליו בקלות, להבין אותו בצורה מהירה ולבצע עיבודים נוספים אם נדרש. הנתונים המוצגים יכולים להיות תוצאות של חיפושים, דוחות או מידע אקטואלי הנוגע לפעילות המשתמש. הגישה למידע נעשית באמצעות מחשבים או מכשירים המחוברים לרשת, והמחשב המרכזי בשרת אחראי על הצגת המידע בצורה מסודרת, נגישה ובצורת המאפשרת למשתמשים להבין את הנתונים במהירות וביעילות.

## **תיאור האלגוריתמים המרכזיים בתוכנית**

### **ניסוח וניתוח של הבעיה האלגוריתמית**

הבעיה האלגוריתמית המרכזית בפרויקט שלי היא מירוץ תהליכים (Race Condition) ,שעלולה להתרחש כאשר מספר תהליכונים מנסים לגשת בו-זמנית לאותו משאב משותף, כמו socket או זיכרון משותף, מבלי שהגישה תסונכרן ביניהם כראוי. הבעיה מתרחשת כאשר תהליכונים מקבילים מבצעים פעולות קריאה או כתיבה על המשאב בו זמנית, דבר שעלול להוביל לשגיאות בתוצאה הסופית, אובדן נתונים או התנהגות לא צפויה של המערכת. דוגמה לכך היא כאשר שני תהליכונים כותבים בו-זמנית לsocket וההודעות עלולות להתערבב זו בזו, דבר שיגרום למידע שגוי בצד השני של החיבור. דוגמה נוספת היא כאשר תהליכונים שונים קוראים בו-זמנית למשאב בזיכרון משותף, דבר שעלול להוביל לתוצאות לא תקינות אם הנתונים עדיין לא הושלמו.

### **אלגוריתמים קיימים לפתרון הבעיה**

הפתרון האופטימלי לבעיה זו הוא להשתמש במנגנונים המאפשרים סנכרון בין תהליכונים. הדרך הפשוטה והיעילה ביותר היא באמצעות נעילות (Locks), ובמיוחד Mutexes, המאפשרים לתהליך אחד בלבד לגשת למשאב בזמן נתון, וכך מונעים מצב שבו שני תהליכונים מבצעים פעולה בו-זמנית על אותו משאב. נעילה כזו מבטיחה שגישה למשאבים תהיה מסודרת וסינכרונית, כך שתהליך אחד מחזיק בנעילה בזמן שהאחרים ממתינים. יש לקחת בחשבון את בעיית ה-Deadlock, שבה שני תהליכונים מחכים לנעילה של זה את זה ולא משחררים אותה. ניתן להימנע מבעיית Deadlock על ידי שימוש בנעילה בסדר מסוים בין תהליכונים, ובכך להבטיח ששני תהליכונים לא ידרשו את אותן הנעילות בו-זמנית.

פתרון נוסף הוא פעולות אטומיות שמאפשרות לבצע פעולות פשוטות כמו עדכון מונה או קריאה וכתיבה על נתונים מבלי שהן יופרעו במהלך הביצוע. פעולות אלה מבטיחות שכל שינוי על משאב יושלם ללא הפרעה מצד תהליכונים אחרים, מה שמונע את בעיית מירוץ התהליכים. אולם, פעולות אטומיות מתאימות בעיקר למקרים שבהם מדובר בשינויים קטנים או פשוטים, כמו עדכון ערכים בודדים בזיכרון, ואינן מתאימות לפתרון בעיות מורכבות כמו עיבוד נתונים או כתיבה לקריאות בו-זמנית. לאחר שבחנתי את כל הפתרונות האפשריים, החלטתי להשתמש ב-Mutexs כפתרון העיקרי.

### **סקירת הפיתרון הנבחר**

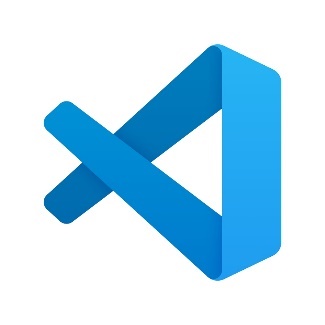
ה-Mutexs נותנים מענה טוב לכל בעיה של גישה לא משוייכת למשאבים, ומבטיחים שאין שני תהליכונים יכולים לגשת לאותו משאב בו זמנית. אני לא אשתמש בתורים, מכיוון שבפרויקט שלי לא מדובר בניהול הודעות בין תהליכונים, אלא בגישה למשאבים משותפים. בנוסף, השימוש בתורים מצריך ניהול נוסף של סדר פעולות ומורכבות שלא מתאימה לדרישות שלי. יתרה מכך, השימוש ב-Mutex מאפשר לי שמירה על קוד פשוט ויעיל.

### **הפנייה למקור רלוונטי**

[מהוMutex](https://www.techtarget.com/searchnetworking/definition/mutex)

## **תיאור סביבת הפיתוח**

תמונה שמכילה גרפיקה, עיצוב גרפי, לוגו, צילום מסך

התיאור נוצר באופן אוטומטיהסביבה בה אני עובד דורשת שימוש במכונה וירטואלית, משום שעלי להפעיל גרסה מסוימת של מערכת לינוקס, שהיא חשובה להרצת הקוד שלי. באמצעות מכונה וירטואלית, אני יכול ליצור סביבה מבודדת שמתפקדת בצורה עצמאית ומסייעת לי לבצע ניסויים ובדיקות בצורה בטוחה ומבוקרת. במכונה הווירטואלית אני מריץ גרסה מסוימת של לינוקס שיכולה להשתנות לפי הצורך, מבלי להשפיע על מערכת ההפעלה הראשית שלי. אני משתמש בכלים כמו VirtualBox או VMware כדי להקים ולהפעיל את הסביבה הווירטואלית הזו. לצורך פיתוח הקוד, אני משתמש בשני עורכי טקסט עיקריים: VIM ו-VSCode.VIM הוא עורך טקסט מאוד פופולרי ונוח למי שמכיר את הפקודות וזקוק לכלים בסיסיים, ומאפשר לי לכתוב קוד בצורה מהירה ויעילה, במיוחד בסביבת לינוקס. הוא קל, גמיש ומאפשר עבודה מהירה עם קבצים ללא צורך בממשק גרפי. מצד שני, אני משתמש גם ב-VSCode, שהוא עורך מודרני ומתקדם יותר, המציע אינטגרציה עם כלים נוספים כמו ניהול גרסאות, דיבאגינג, ותוספים רבים שמקליקים את העבודה עם קוד. בנוסף, אני משתמש ב-Git ו-GitHub לניהול גרסאות הקוד שלי. Git הוא כלי לניהול גרסאות שמאפשר לי לעקוב אחרי השינויים בקוד, לשמור על גרסאות קוד שונות ולשלוט בצורה יעילה בהתפתחויות הקוד לאורך הזמן. GitHub הוא שירות מבית Git שמאפשר לי לשתף את הקוד, לעבוד בצוות ולגבות את הקוד שלי. כלים אלו חשובים במיוחד כשיש צורך לעקוב אחרי שינויים רבים בקוד ולשתף את העבודה עם אחרים. כחלק מהתהליך, אני גם עושה שימוש ב-Wireshark כדי לנתח את התקשורת בין רכיבי המערכת. Wireshark הוא כלי מצוין לתפיסת וניתוח תעבורת רשת, אשר מאפשר לי לראות את זרימת המידע בין השרתים והלקוחות, ולוודא שאין בעיות בתקשורת או בשימוש בכתובת IP. זה חשוב במיוחד לבדיקת האינטראקציות וההעברות נתונים, כדי לוודא שהקוד עובד בצורה תקינה. לסיכום, אני עושה שימוש בסביבה מאוד עשירה שמספקת לי את כל הכלים הנדרשים לפיתוח, בדיקות, וניהול הקוד שלי. מכונה וירטואלית מספקת לי את הגמישות, עורכי הטקסט מאפשרים עבודה נוחה ומהירה, Git ו-GitHub חשובים לניהול הגרסאות, ו-Wireshark מאפשר ניתוח ופתרון בעיות בתקשורת. כלים אלו יחד יוצרים סביבה עוצמתית ומסודרת לפיתוח מתקדם.

## **תיאור פרוטוקול התקשורת**

החשיבות של פרוטוקול תקשורת בפרויקט שלי היא מכרעת, שכן הוא מבצע את התיאום וההבנה בין כל הרכיבים במערכת. פרוטוקול תקשורת מספק את הכללים וההנחות שדרכם רכיבי המערכת יכולים להבין אחד את השני ולהעביר מידע בצורה מסודרת וברורה. אם הפרוטוקול מוגדר בצורה טובה, הוא הופך את התקשורת לפשוטה יותר ומובנת לכל הצדדים המעורבים, ללא צורך בהבנה עמוקה של כל רכיב בנפרד. פרוטוקול טוב יכול להגדיר כיצד נשלחים ומתקבלים נתונים, אילו פורמטים נתמכים, איך להתמודד עם טעויות או חיבורים לא יציבים, ואיך ניתן לזהות את הנתונים בצורה אפקטיבית. הוא מספק מנגנונים של סנכרון ותקשורת בין תהליכים או רכיבים שונים במערכת, ומבצע את כל התיאומים הנדרשים כדי להבטיח שהנתונים יגיעו בשלמותם ובזמן. פרוטוקול ברור ומובנה מסייע במניעת טעויות בתקשורת, ומונע תקלות שעלולות להיגרם כאשר אין הבנה משותפת לגבי הציפיות בין הצדדים. יתר על כן, פרוטוקול תקשורת מגדיר את ההתנהגות של המערכת והופך את הפרויקט לנגיש יותר מבחינת פיתוח ותחזוקה.

מבנה הפרוטוקול תקשורת בו אשתמש בפרויקט הינו במבנה הבא:

1. אורך הודעה (כשמונה תווים שמהווים מספר)

2. סוג הפעולה שהתבצעה (המתודה שנקראה/איזה פעולה המשתמש ביצע)

3. המידע הקשור לאותה פעולה (פרמטרים/Return Value)

קיים בכל הודעה רק תו אחד מפריד, אשר השמונה בתים הראשונים יהיו אך ורק לאורך ההודעה, ולכן הבתים לאחר מכן בוודאות קשורים לחלק השני של ההודעה. לכן נשתמש בתו מפריד אך ורק בין החלק השני של ההודעה לחלק השלישי של ההודעה. התו המפריד להודעות יהיה "~". חשוב להתייחס לכך שאם התו המפריד יהיה חלק מהחלק השני של ההודעה, תיווצרנה בעיה בפרוטוקול, לכן נימנע מלשלוח הודעה שכוללת בחלק השני שלה את התו הנ"ל. לא תהיינה בעיה שהתו יופיע כחלק מהחלק השלישי של ההודעה, אשר ברגע שנמצא התו המפריד הראשון, שם נדע לבצע את ההפרדה בין חלקי ההודעה.

### **דוגמה להודעות:**

Key~C00000005 (לחיצת מקלדת)

MUS~100,20000000011 (לחיצת עכבר, מיקום בציר האנכי והאופקי של המסך).

PRC~chrome.exe00000014 (פתיחת תהליך).

ועוד...

### **טבלת הודעות פרוטוקול**

|  |  |  |  |
| --- | --- | --- | --- |
| שם ההודעה | מבנה שדות ההודעה | נשלח מ-/אל- | תיאור |
| KEY | המקש שנלחץ | לקוח אל שרת (עובד אל מנהל) | הודעה המתריעה איזה מקש מקלדת נלחץ אצל המשתמש |
| MUS | מיקום העכבר | לקוח אל שרת (עובד אל מנהל) | בעת לחיצה על העכבר ישלח המיקום הנוכחי של העכבר, הן בציר האנכי והן בציר האופקי |
| PRC | שם תהליך | לקוח אל שרת (עובד אל מנהל) | כאשר תהליך נוצר, השם שלו ישלח אל המנהל כמו בדוגמה להלן. |
| CRC | שם תהליך | לקוח אל שרת (עובד אל מנהל) | כאשר תהליך מסתיים, השם שלו ישלח אל המנהל. |
| CRF | שם הקובץ | לקוח אל שרת (עובד אל מנהל) | כאשר המשתמש מחליט ליצור קובץ חדש, שם הקובץ ישלח אל המנהל |
| DLF | שם הקובץ | לקוח אל שרת (עובד אל מנהל) | כאשר המשתמש מחליט למחוק קובץ קיים במחשב, שם הקובץ ישלח אל המנהל |
| CPY | המידע שמועתק | לקוח אל שרת (עובד אל מנהל) | כאשר המשתמש מעתיק מידע (בעזרת למשל  Ctrl + c), המידע שהועתק יועבר אל המנהל. |
| NDC | שם המכשיר | לקוח אל שרת (עובד אל מנהל) | כאשר המשתמש חיבר מכשיר חדש אל המחשב, שם המכשיר (או מידע אחר שיעזור למנהל להבין מה התחבר למחשב) ישלח אל המנהל. |
| DND | שם המכשיר | לקוח אל שרת (עובד אל מנהל) | כאשר המשתמש מנתק מכשיר שמחובר אל המחשב שלו, המידע על אותו מכשיר ישלח אל המנהל. |
| SHD | אין | לקוח אל שרת (עובד אל מנהל) | המשתמש מתריע למנהל כי הוא בתהליך כיבוי, כלומר המשתמש כיבה את המחשב שלו ומתריע לשרת שהחיבור הולך להסתיים ביניהם. |
| NNC | צד השני של החיבור | לקוח אל שרת (עובד אל מנהל) | כאשר המשתמש יוצר חיבור אינטרנטי חדש עם עוד מחשב, מידע על הצד השני ישלח למנהל |
| CNC | צד השני של החיבור | לקוח אל שרת (עובד אל מנהל) | כאשר המשתמש סוגר חיבור אינטרנט עם עוד מחשב, מידע על הצד השני ישלח למנהל |

## **תיאור מסכי המערכת**

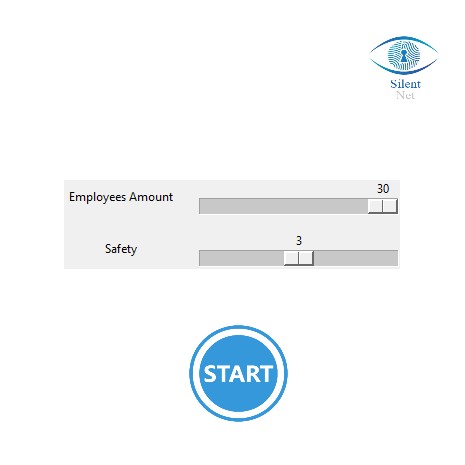
### **מסך פתיחה (אצל השרת)**



מהות מסך:

מסך זה הינו המסך הראשון ביותר שיופיע במחשב השרת (המנהל) כאשר הוא פותח את התוכנית. מסך זה הינו מסך פתיחה, כלומר מחכה לקלט המשתמש על מנת להמשיך הלאה. כפי שאפשר לראות במסך זהו בעצם הלוגו של המערכת "Silent Net".

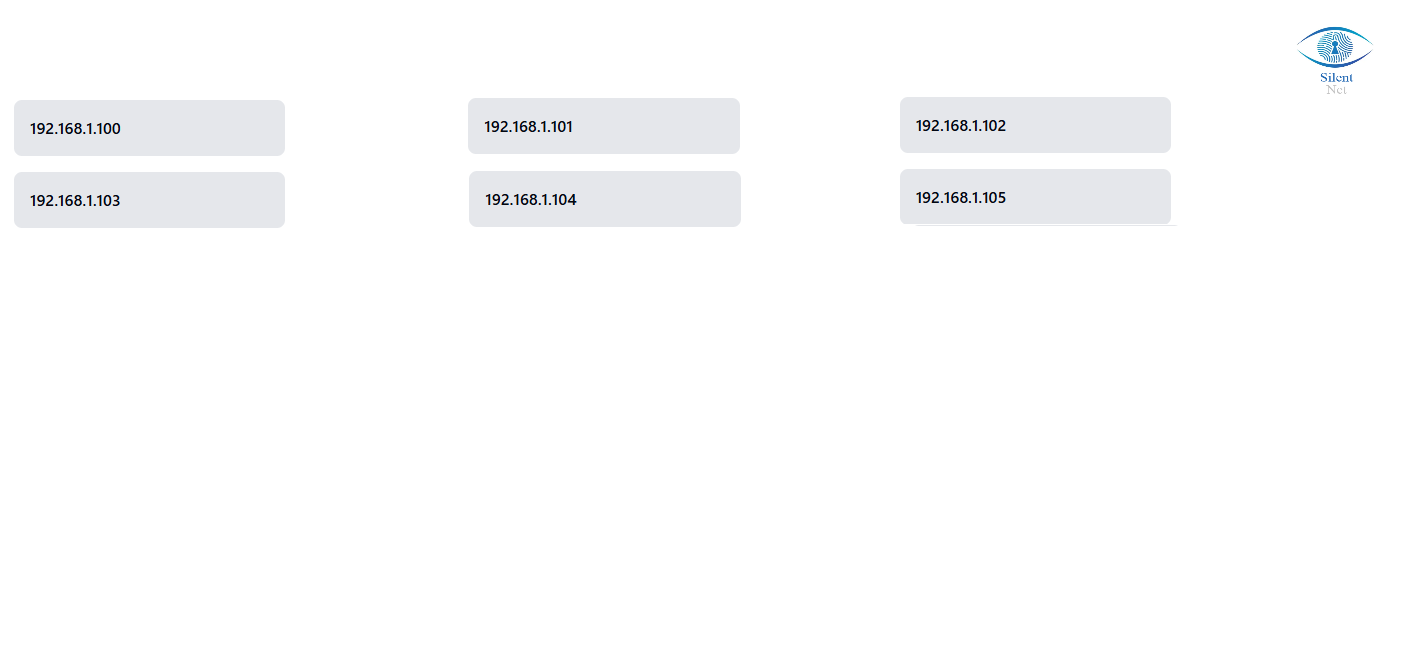
### **מסך הגדרות**



מהות מסך:

מסך זה הינו המסך השני אשר מוצג במחשב השרת, מסך זה מבקש מן המנהל עצמו להכניס באופן ידני את ההגדרות הראשוניות ביותר לתוך הפרויקט. אחת מן ההגדרות (הראשונה) מבקשת את הכמות המקסימלית של לקוחות שיכולים להתחבר למחשב השרת. הגדרה זאתי מונעת מן מחשב השרת לקרוס בעת עומס, בנוסף על כך מונעת התקפה ידועה בשם DDOS/DOS בה מתחברים כמויות גדולות של לקוחות אל מחשב השרת במטרה להפיל אותו בשל עומס. ההגדרה השנייה קובעת את כמות הבטיחות של הפרויקט, כלומר הכוונה היותר מדויקת היא כמה מחשב השרת משאיר קשר עם מחשב מסוים, עד שהוא מנתק אותו בשל היות אותו קשר עם מחשב שמטרותיו זדוניות. רמת הבטיחות היא בסופו של דבר כמה מחשב השרת משאיר קשר עם מחשב לקוח לאחר שמחשב הלקוח שוב ושוב שלח מידע שלא ניתן לפיענוח ולא עומד בפרוטוקול. אם מחשב לקוח שולח באופן עקבי מידע לא אמין, רמת הבטיחות תקבע מתי כבר להפסיק להאמין לאותו לקוח ולנתק את הקשר איתו.

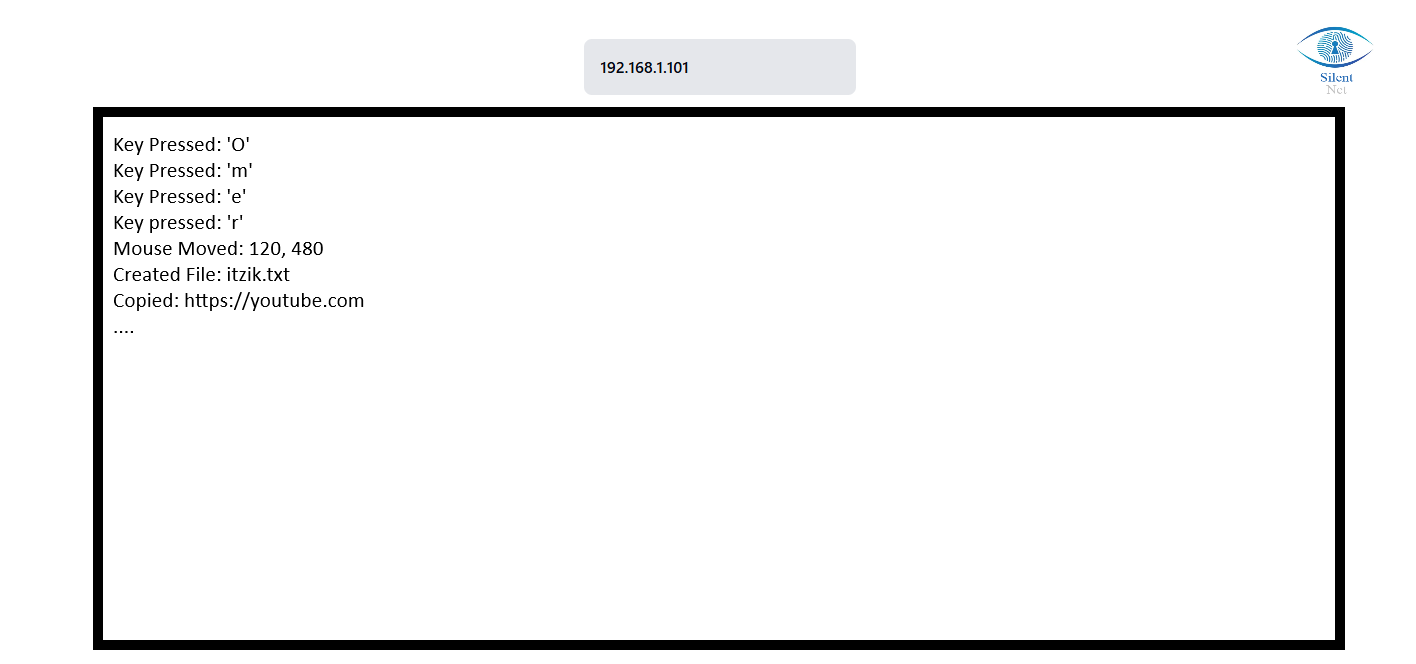
### **מסך ראשי**



מהות מסך:

מסך זה הינו המסך השלישי אשר מוצג במחשב השרת, מסך זה מראה בזמן אמת את המחשבים המחוברים כרגע למחשב השרת ואשר מנהלים איתו קישור. אותם מחשבים שולחים מידע אל מחשב השרת. כל כתובת של מחשב פה היא בעצם גם כפתור שעליו אפשר ללחוץ ולהגיע למידע הישיר על כל מחשב במשרד.

### **מסך אישי**



מהות מסך:

מסך זה הינו המסך הרביעי אשר מוצג במחשב השרת (לא בהכרח חייב להיות הרביעי). מסך זה מראה בזמן אמת את המידע אשר מתקבל מן אותו מחשב בעל כתובת מסוימת. ניתן להגיע למסך זה על ידי לחיצה על אחת הכתובות הקיימות במסך מספר שלוש.

### **דיאגרמת מסכים**

תמונה שמכילה טקסט, צילום מסך, תרשים, מקביל

התיאור נוצר באופן אוטומטי

## **תיאור מבני נתונים**

הפרויקט שלי עוסק בניהול מידע שנשלח ממחשבים שונים, כאשר כל מחשב מזוהה על פי כתובת ה-IP שלו. המידע שנשלח מהמחשב יכול לכלול נתונים שונים, כמו מיקום העכבר, שמות קבצים או תהליכים, ונתונים מועתקים כמו תווים או בייטים. כל מחשב שומר את המידע האחרון שנשלח, והמערכת עדכנתו באופן רציף. במערכת שלי, כל כתובת IP משמשת כמפתח (key), וערך המפתח הוא המידע האחרון שנשלח מהמחשב. המידע יכול להיות טקסטואלי (כמו שם קובץ או תהליך), מספרי (כמו מיקום עכבר בקואורדינטות), או בייטים (כמו נתונים שהועתקו). כל עדכון מידע ממחשב ספציפי מחליף את המידע הקודם שנשלח ממנו. כדי לאחסן את המידע בצורה מסודרת, כל מחשב יש לו שדה ייחודי (כתובת ה-IP), ושדה נוסף שישמור את המידע האחרון שנשלח ממנו, עם תאריך ושעה של עדכון המידע. המידע במערכת יכול להשתנות בכל זמן, וכך כל פעם שמחשב שולח נתון חדש, הוא מעדכן את המידע הקיים.



במקרה זה, כל לקוח (לפי כתובת ה-IP שלו) ישמור את כל ההודעות שנשלחו. לדוגמה, אם מחשב עם כתובת ה-IP 192.168.1.101 שלח מיקום עכבר, ואז שלח קובץ שנפתח ושוב שלח טקסט מועתק, כל אחת מההודעות תשמר בנפרד, כך שיהיה אפשר לראות את כל ההיסטוריה של ההודעות שנשלחו על ידי המחשב. לכל שורה בטבלה יש:

-כתובת IP של המחשב ששולח את ההודעה.

-סוג המידע שנשלח (למשל, מיקום עכבר, קובץ נפתח, טקסט מועתק).

-המידע עצמו, כלומר הערך שנשלח (כמו קואורדינטות של העכבר, שם הקובץ, הטקסט המועתק או קוד ההֶקסָדֶצִימָלִי של הבייטים).

-תאריך ושעה שבהם ההודעה נשלחה.

מבנה כזה יאפשר לשמור את כל ההודעות שנשלחו, כך שתוכל לעקוב אחרי כל היסטוריית ההודעות של כל לקוח.

## **סקירת חולשות ואיומים**

בפרויקט שלי, השימוש בהצפנה כמו RSA, DH ו-AES חשוב מאוד כדי להגן על המידע המועבר בין השרתים למחשבים הלקוחות. הצפנה א-סימטרית כמו RSA ו-DH מאפשרת לנו להבטיח שהמידע שמועבר יהיה מוגן מפני התקפות של האזנה או שינוי מידע על ידי תוקפים. השימוש במפתחות פרטיים וציבוריים מבטיח שהמידע יישאר סודי גם אם התוקף מצליח להשתלט על התקשורת בין השרת ללקוח. בנוסף, הצפנה סימטרית כמו AES מצפינה את המידע באמצעות אותם מפתחות שהועברו בין שני צידי התקשורת, כך שהתקשורת תהייה מוצפנת מקצה לקצה. למרות שהשימוש בהצפנה עוזר להגן על המידע במערכת, ישנם מספר חולשות שיכולות להיגרם בפרויקט הזה: התקפות MITM (Man-in-the-Middle): למרות שהמידע מוצפן, אם לא מבוצע אימות נכון של הזהות בין השרת ללקוח, תוקף יכול להוציא את המידע מהתקשורת ולשלוח מידע כוזב. הצפנה כמו RSA ו-DH עוזרת למנוע את הבעיה הזאת, אך אם המפתחות לא מנוהלים בצורה מאובטחת, התקפות MITM עדיין אפשריות. Spoofing: אם הלקוח לא מבצע אימות של השרת או אם התוקף יכול לשנות את כתובת ה-IP של הלקוח, הוא יכול להציג את עצמו כלקוח אמיתי ולהעביר מידע כוזב. שימוש בהצפנה ואימות זהות כמו RSA יכול למנוע את הבעיה הזאת. התקפות DDOS: המערכת עלולה להיתקל בהתקפות DDOS, בהן תוקף מנסה להציף את השרת עם בקשות במטרה להוריד אותו. אם אין מנגנוני זיהוי ובקרה על בקשות לא חוקיות, המערכת עלולה להתרסק או להיות לא זמינה. בזכות מסך הפתיחה בו המנהל מגדיר את הכמות המקסימלית של לקוחות (ומגדיר את רמת הבטיחות), גם אם תהיינה התקפה היא לא תהייה יעילה ולא תשפיע כלל על הפרויקט.



קשיים:

Timeout- על סוקט קרנלי, היה קושי בלמצוא את הפקודה שמאפשרת לעשות זאת.

ביצוע פעולות blocking עם interrupt context היה תוקע את כל המערכת וכל המערכת מפסיקה לעבוד.

חיפוש פעולות שנקראות על ידי הקרנל.

פתרונות:

-בהתחלה הפיתרון היה להריץ תהליכון שבודק, לאחר מכן אחרי חיפוש מעמיק מצאתי פקודה קרנלית שנותנת לבצע timeout שליחת וקבלת הודעות.

-לאחר קריאת ספר מצאתי מבנה נתונים שפותח תהליכונים מאחרי הקלעים ומתזמן אותם ב process context לפי סדר – workqueue.

-הרבה פעולות של הקרנל שעליהם אני מבצע הוקים דורשות חיפוש ב source code, לכן הורדתי את ה source code על המחשב שלי וכל פעם שאני צריך למצוא פקודה אני מחפש שם.