

מסמך תוכנה

הגנת סייבר

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

**שם מנחה: אופיר שביט**

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

## **הסבר טכני של הפרויקט**

המערכת היא מערכת האזנה שקטה שמותקנת על מחשבים במערכת הפעלה לינוקס, ומיועדת למנהל החברה כדי לעקוב אחרי פעילות העובדים בצורה דיסקרטית. המערכת מאזינה לפעולות שונות במחשב, כמו מקלדת, עכבר, צילומי מסך ושמע, ומעבירה את המידע למנהל בזמן אמת. המטרה היא לאפשר למנהל לבדוק אם העובדים מבצעים את עבודתם כראוי, מבלי שהעובדים ידעו על נוכחותה של התוכנה או יוכלו לשנותה. בכך, המנהל יכול לקבל תובנות על פעילות העובדים ולבצע שינויים בהתאם.

## **תיאור מעמיק של כל בלוק**

### **שרת**

בפרויקט זה השרת הינו מחשב המנהל של החברה. המנהל של החברה צריך לדעת מידע על פעילות עובדיו במהלך יום העבודה. בכדי לבצע כך המנהל צריך שכל המידע ממחשבי החברה ישלח אליו, וגם בכדי שיצליח לעשות זאת בצורה נוחה, המנהל יצטרך תוכנית שתנגיש לו את המידע בצורה ברורה על כל מחשב ומחשב בנפרד. השרת מורכב ממספר מרכיבים שונים שמחוברים יחד על מנת ליישם פעילות זאת:

GUI

על מנת להציג את המידע אצל השרת בצורה נוחה וברורה עליו להיות בעל ממשק גרפי אשר יהיה נוח לשימוש גם למי שאינו מבין בטכנולוגיה שמתרחשת מאחורי הקלעים. ממשק המשתמש אמור להיות הגיוני מבלי הרבה כפתורים מסובכים עם הוראות ברורות. הממשק הגרפי כולל מספר מסכים.

SOCKET

על מנת לקבל את המידע אצל השרת מן הלקוחות, על השרת לתמוך בתקשורת, ועל כך ישתמש באובייקט של מערכת ההפעלה (SOCKET), אותו אובייקט יסייע לו לקבל את המידע מן הלקוחות בצורה מופשטת ולהתנהל נגדו.

הצפנה

אותו מידע שהולך לישלח בין המחשבים אל מחשב השרת הינו מידע רגיש על מחשבי החברה, ועל כך להיות מוצפן. 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.

תהליכונים

בשביל כל לקוח ששולח את המידע צריך להיות בתוכנה של השרת צד שמקבל את המידע. מכיוון שמספר הלקוחות הינו מוגבל בפרויקט זה, אחת מהדרכים הטובות לבצע זאת הינה לפתוח תהליכון לכל לקוח במחשב השרת, כך שהתקשורת בין הלקוח לשרת תתקבלנה מיד מבלי צורך לחכות.

**לקוח**

בפרויקט זה הלקוח הינו מחשב של הלקוח בחברה. הלקוחות בחברה שולחים את המידע מהמחשב שלהם אל מחשב השרת בכדי ליידע אותו על מידע אצלם. התוכנה ששולחת את המידע אל השרת הינה תוכנה שקטה (חשאית), ועליה להיות מאחורי הקלעים בכדי שהלקוחות לא יוכלו לשבש אותה ולשנות בה דברים. הלקוח מורכב ממספר מרכיבים שונים שמחוברים יחד על מנת ליישם פעילות זאת:

SOCKET

על מנת לשלוח את המידע מהלקוח אל השרת, על הלקוח לתמוך בתקשורת, ועל כך ישתמש באובייקט של מערכת ההפעלה (SOCKET), אותו אובייקט יסייע לו לשלוח את המידע אל השרת בצורה מופשטת ולהתנהל נגדו.

הצפנה

אותו מידע שהולך לישלח בין המחשבים אל מחשב השרת הינו מידע רגיש על מחשבי החברה, ועל כך להיות מוצפן. הלקוח ישלח 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.

KLM

בכדי לשלוח מידע של מחשבי העובדים בחברה בזמן אמת, צריכה להיות תוכנה שמאזינה לפעילות העובדים. ולכן בפרויקט זה בחרתי להשתמש במערכת הפעלה לינוקס בה אפשר להשתמש ב Kernel Loadable Module, כך שאני יכול להוסיף קוד למערכת ההפעלה. אותו קוד יאזין לפונקציות שאני יכול להאזין אליהן ולקבל מהן מידע שלאחר מכן ישלח אל השרת בכדי ליידע אותו על פעילות העובד. מכיוון שזהו KLM, הלקוח יתקשה בלשנות את תוכנה זו אלא אם כן יש לו הרשאות גבוהות על מחשב זה.

### **שאילתות SQL**

SELECT time, action, data FROM <table\_name> WHERE ip=<wanted\_ip> - הצגת כל ההודעות של לקוח אחד.

INSERT INTO <table name> (ip, action, data, time) VALUES (<v1>,<v2>,<v3>, <v4>) – הודעה חדשה

DELETE FROM <table\_name> WHERE ip=<wanted\_ip> - משתמש התנתק

(שאילתת SQL אפשרית היא בכדי להגביל את מספר ההודעות שיהיו במסד נתונים, בכדי להוריד עליו עומסים)

DELETE FROM <table\_name> WHERE time < NOW() - INTERVAL <time\_period> MINUTE – מחיקת כל ההודעות שנשלחו לפני יותר מפרק זמן מסוים בכדי להוריד עומסים.**מודולים ומחלקות**

**מודולים שאני מפתח:**

1. **שם המודול** protocol  
   **תפקיד המודול:** לטפל בתקשורת TCP, כולל שליחה וקבלה של נתונים.
2. **שם המודול**  CryptoMSG  
   **תפקיד המודול:** לטפל בהצפנה ופענוח של הודעות באמצעות מפתחות.
3. **שם המודול** DB  
   **תפקיד המודול:** לנהל בקשות SQL ולבצע פעולות על מסד הנתונים.

**מחלקות שאני מפתח:**

1. **שם המחלקה**clientComm   
   **תפקיד המחלקה:** לאחסן פרטים הקשורים לתקשורת TCP.

**תכונות המחלקה:**

* + **tcp\_socket:**  
    האובייקט שמייצג את ה-TCP Socket.
  + **buffer\_size:**  
    גודל חוצץ לקריאה/כתיבה של נתונים.
  + **timeout:**  
    מגבלת זמן לפעולות תקשורת.

1. **שם המלקה CryptoHandler**  
   **תפקיד המחלקה:** לנהל את פרטי ההצפנה והפענוח.

**תכונות המחלקה:**

* + **public\_key:**  
    המפתח הציבורי המשמש להצפנה.
  + **private\_key:**  
    המפתח הפרטי המשמש לפענוח.

1. **שם המחלקה** DBhandler  
   **תפקיד המחלקה:** לנהל פרטים רלוונטיים לעבודה עם מסד הנתונים.

**תכונות המחלקה:**

* + **connection:**  
    החיבור למסד הנתונים.
  + **Cursor:**

מצביע למסד נתונים.

* + **query\_timeout:**  
    מגבלת זמן לביצוע שאילתות.

1. **שם המחלקה** clientObject
   * client\_comm  
     אובייקט מסוג מחלקת התקשורת
   * crypto\_handler

אובייקט מסוג מחלקת הצפנה ופיענוח.

* + Identification  
    מזהה של אותו לקוח (כתובת ip).

1. **שם המחלקה** WindowManager (מחלקה סטאטית)
   * Windows\_list  
     רשימה של המסכים לפי סדר
   * Current\_screen

אינדקס של המסך המתאים ברשימה.

**הסבר נוסף:**

* המחלקה DBHandler תנהל את החיבור למסד הנתונים, ותאפשר הגדרה של פרטים רלוונטיים כמו סוג מסד הנתונים או מגבלת זמן לביצוע שאילתות .חשוב לציין כי המחלקה DBhandler תהיה עם Design pattern בשמו Singleton בכדי למנוע מקרים בהם כמה תהליכונים ניגשים לאותו משאב משותף בו-זמנית race condition)).

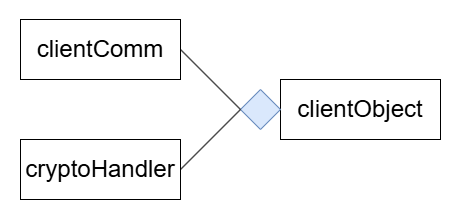
החלוקה הזו שומרת על סדר ומפשטת את הניהול של כל רכיב במערכת.

### **פעולות**

כיוון החלוקה למודולים ומחלקות רבות, רוב הפעולות יהיו בתוך המודולים שאכתוב על מנת לפשט את הקוד ולגרום לו להיות לפי עקרונות תכנות OOP.

מחלקת clientObject

פעולה בונה

def \_\_init\_\_ (self, client\_socket, client\_address):

self.address = client\_address

self.client\_comm = clientComm(client\_socket)

self.crypto\_handler = cryptoHandler()

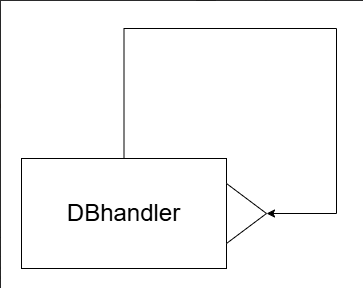
מחלקת cryptoHandler

פעולה בונה

def \_\_init\_\_ (self):

self.public\_key, self.private\_key = rsa. newkeys(1024)

מחלקת clientComm

פעולה בונה

def \_\_init\_\_ (self, client\_socket):

self.client\_socket = client\_socket

מחלקת DBhandler

def \_\_init\_\_(self, db\_name, query\_timeout=30):

self.connection = sqlite3.connect(db\_name)

self.connection.timeout = query\_timeout

self.cursor = self.connection.cursor()

מחלקה WindowManager

def main\_screen():

def personal\_screen():

def settings():

def silent\_net():

פעולות הלקוח

חשוב לציין כי כל הפעולות והמחלקות שאוזכרו עד כה הן נוגעות לחלק השרת של הפרויקט, ואינן נוגעות לחלק הלקוח של הפרויקט. הלקוח בפרויקט זה כמו שכבר אוזכר יהיה קוד שייכתב כחלק מ KLM, ועל כך חייב להיכתב בשפת C אשר אינה תומכת ב OOP. הפעולות באותו KLM הינן פעולות שיופעלו בהתאם לפעולות מערכת הפעלה, ולכן אינן מופעלות על ידי התוכנה עצמה, הן פעולות שמבצעות hook על פעולות אחרות. לצורך הדגמה הינה חתימות של כמה פעולות של האזנה –

האזנה למקלדת

static int handler\_pre\_keyboard(struct kprobe \*p, struct pt\_regs \*regs)

האזנה לעכבר

static int handler\_pre\_mouse(struct kprobe \*p, struct pt\_regs \*regs)

האזנה לפתיחת קבצים

static int handler\_pre\_open(struct kprobe \*p, struct pt\_regs \*regs)

ועוד... (כל הפעולות יהיו בעלות אותם פרמטרים, רק השם שונה. כך נראות פונקציות hook שמשתמשות ב kprobe).

**פעולות חיצוניות**

מודול פרוטוקול

ספריות חיצוניות בשימוש: socket (אין צורך ב pip install, מגיע עם פייתון).

תפקיד המודול: טיפול בתקשורת TCP, כולל שליחה וקבלה של נתונים.

במודול זה אני אטפל בהקמה של חיבורי TCP, שליחה וקבלה של נתונים. הפונקציות העיקריות יהיו:

* **send\_data(client\_socket, data)**:

פונקציה לשליחה של נתונים דרך חיבור TCP. היא תקבל את המידע שצריך לשלוח ותשדר אותו לשרת או ללקוח.

* **receive\_data(client\_socket)**:

פונקציה שתקבל נתונים מחיבור TCP. היא תקרא את המידע שהתקבל מהשרת או הלקוח ותעביר אותו הלאה.

* **close\_connection(client\_socket)**:

פונקציה שתסגור את החיבור הקיים לאחר סיום התקשורת, ותוודא שחרור של המשאבים.

מודול CryptoMSG

ספריות חיצוניות: Crypto (pip install pycryptodome).

תפקיד המודול: טיפול בהצפנה ופענוח של הודעות באמצעות מפתחות.

במודול הזה אני אתמקד בפענוח של הודעות על ידי שימוש במפתחות. הפונקציות העיקריות יהיו:

* **decrypt\_message(key, msg)**:

פונקציה שתפענח את ההודעה המוצפנת ותשיב את המידע המקורי. הפונקציה תקבל את ההודעה המוצפנת ואת המפתח ותפענח אותה.

decrypt\_cipher = AES.new(key, mode,iv)

plain\_text = decrypt\_cipher.decrypt(message)

* dh\_generate\_keys(size)

פונקציה שיוצרת מספרים בשביל החלפת מפתחות לאחר מכן (מפתחות aes), הפונקציה תיצור מספרים בשביל החלפת מפתחות בצורת Diffie Hellman.

מודול DB

ספריות חיצוניות: sqlite3 (אין צורך ב pip install, מגיע עם פייתון).

תפקיד המודול: לנהל בקשות SQL ולבצע פעולות על מסד הנתונים.

המודול הזה יטפל בתקשורת עם מסד הנתונים, יאפשר לי לשלוח שאילתות SQL ולהוציא תוצאות. הפונקציות העיקריות יהיו:

* **connect\_to\_db(db\_name)**:

פונקציה שתיצור חיבור למסד הנתונים.

* **execute\_query(query)**:

פונקציה שתבצע שאילתת SQL(כמו SELECT, INSERT, UPDATE או DELETE)ותשיב את התוצאות או את סטטוס הביצוע של השאילתה.

* **close\_db\_connection()**:

פונקציה שתסגור את החיבור למסד הנתונים לאחר סיום העבודה, ותגייס את המשאבים בצורה נכונה.

## **מבני נתונים ומשתנים גלובליים**

פרויקט זה מחולק לשני צדדים מרכזיים, צד השרת וצד הלקוח. שני הצדדים כתובים שונה ומשתמשים במבני נתונים שונים עקב זאת. לכן נפריד ביניהם.

שרת

הפרויקט שלי אינו מסובך מבחינה חישובית (אינו דורש אלגוריתמים מסובכים מאוד), ולכן השימוש במבני נתונים מסובכים גם כן אינו דרוש. למרות זאת עדיין קיים שימוש מינימלי במבני נתונים שונים כגון:

רשימה – בכדי לשמור את כל התהליכונים הפתוחים (אם זה תהליכון בשביל לקוח, או תהליכון שמקבל לקוחות ועוד...), מכיוון שפרויקט זה משתמש ב TCP שהוא פרוטוקול מבוסס קישור, אין צורך במבנה נתונים גלובלי שמקבל אליו את כל ההודעות וממנו ההודעות ממשיכות הלאה אל היעד המתאים (כמו למשל כאשר משתמשים ב UDP).

טופל (Tuple) – בכדי לקבץ נתונים לנתון אחד (קבוצה אחת) בתור אובייקט אשר יהיה נוח יותר לעיסוק והעברה בין פעולות, יהיה צורך באובייקט מסוג טופל, אשר מאפשר זאת. דוגמה אחת לכך יכולה להיות בפרוטוקול התקשורת, אשר לכל הודעה יש גם את הסוג של ההודעה וגם את המידע בתוכה. הרבה יותר הגיוני להפריד את שניהם לתוך טופל במקום להתעסק עם אינדקסים והפרדת סטרינגים בכדי לדעת מה סוג ההודעה ומה המידע בהודעה.

מנעולים – מכיוון שהפרויקט עוסק במספר לקוחות אשר מתקשרים עם מחשב בודד, יהיה צורך בפעילות בו-זמנית בשביל כמה לקוחות ביחד על מחשב השרת. אחת מן הפעולות השכיחות ביותר בפרויקט תהייה גישה אל מסד הנתונים, הבעיה מכך היא שהוא משאב משותף לכל התהליכונים, וכאשר כמה מהם ינסו לגשת אליו ביחד, תוכל להיווצר בעיה (race condition), ועל כך חייב להשתמש במבנה נתונים מסוג מנעול, אשר נותן גישה בודדת לתהליכון בכל רגע נתון, מה שיתגבר על בעיה זו.

לקוח

מנעולים – כמו שהוסבר בשרת, גם כן אצל הלקוח דרושים מנעולים. אחת הדוגמאות לשימוש במנעולים יכולה להיות כאשר המחשב הפעיל כמה פעולות של מערכת ההפעלה ביחד, והמידע צריך לישלח ביחד אל השרת. אם המחשב ינסה לגשת בו-זמנית אל אובייקט socket שמקשר בינו לבין השרת, המידע הולך להיות מקוטע ואף רוב יגרום לבעיות רבות אחרות, ועל כך גם כן אצל הלקוח חובה להשתמש במנעולים.

מכיוון שהלקוח כתוב בשפת C אין מבני נתונים בנויים בתוך השפה (אפשר לממש אבל אין צורך בפרויקט שלי), ולכן אכתוב את ה structים העיקריים בפרויקט שלי.

- struct kprobeמבנה המשמש להגדרת קפיצים (kprobes) במטרה ליירט פונקציות בגרעין.

- struct socketמייצגת סוקט רשת בגרעין.

- struct sockaddr\_inמייצגת כתובת סוקט IPv4.

- struct mutexמבנה לניהול נעילה של משאבים, כך שמאפשר רק חוט אחד לגשת למשאב במקביל.

## **לולאות ופעולות ראשיות**

לקוח

מבנה התוכנית אצל הלקוח אינו מבנה קוד רגיל אשר הוא קוד שמאזין לפעולות המערכת. הפעולות בתוכנית אצל הלקוח מזכירות התנהגות של פעולות אסינכרוניות אשר מחכות לפעולה מסוימת שתתרחש ורק אז פועלות, כך גם פועלות כל הפעולות בתוכנית שלי. לכן אין הגדרה לפעולה ראשית, לכן נגדיר רק פעולות ראשיות. הפעולה הראשית המרכזית בתוכנית היא הגדרת הפעולות, כלומר הפעולה הראשונה שפועלת כאשר טוענים את המודול לזיכרון מערכת ההפעלה. חתימת פעולה זו תיראה כך -

static int \_\_init module\_init(void);

בפעולה זו יוגדרו כל הפעולות שמאזינות וכך למעשה זוהי הפעולה שמביאה את כל הפרויקט לידי ביטוי.

שרת

מבנה התוכנית אצל השרת בנוי בצורה כך שעליו לקבל לקוחות ובו זמנית גם לקבל מידע מן אותם לקוחות שכבר מחוברים אליו (אשר פרויקט זה משתמש בTCP שהינו מבוסס קישור), ועל כך חייבות להיות כמה לולאות ופעולות ראשיות בפרויקט זה (עם פסאודו קוד – לא קוד אמיתי ולא סינטקס מדויק, אלא רק הרעיון של הלולאה):

פעולה שמקבלת לקוחות

def recv\_clients():

while True:

client,addr = get\_client()

client\_thread = thread(recv\_data, args=(client, addr))

client\_thread.start()

פעולה שמקבלת מידע מן הלקוחות

def recv\_data(client\_socket, address):

while True:

data = get\_data(client\_socket)

update\_data(data, address)

פעולה שעוקבת אחר פעולת המנהל, כלומר מראה לו את המסך המבוקש

def screen\_display():

# Calling all screen functions, each function has its own loop and checks

## **מסכים**

### **מסך פתיחה (אצל השרת)**

קשר למחלקות:

כל המסכים קשורים למחלקה WindowManager, מסך זה הינו קשור אך ורק למחלקה זו.

### **מסך הגדרות**

קשר למחלקות:

מסך זה קשור למחלקה WindowManager, בנוסף ישפיע על משתנים גלובאליים במודול protocol, כגון כמות מקסימלית של לקוחות ורמת בטיחות של המערכת.

**מסך ראשי**

קשר למחלקות:

מסך זה קשור למחלקה WindowManager, המסך בנוסף צריך להיות בעל גישה למשתנה גלובלי שמכיל את כל הלקוחות המחוברים כעת למערכת (כל כתובות ה IP).

### **מסך אישי**

קשר למחלקות:

מסך זה קשור למחלקה WindowManager, בנוסף מסך זה קשור למחלקה DBhandler – בכדי לקבל את המידע הספציפי על לקוח מסוים על המערכת לבצע שאילתת SQL.

**פרוטוקול (חמשת צעדים):**

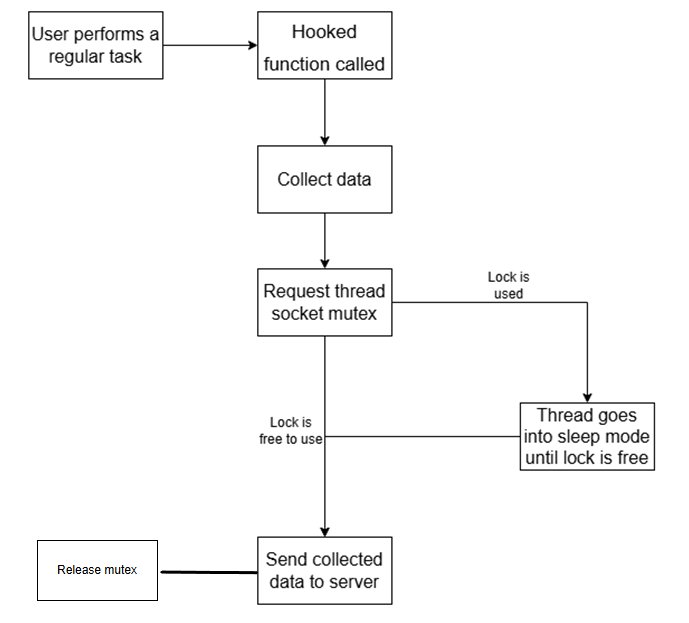
1. מטרה: העברת מידע על הלקוחות אל השרת בזמן אמת.

2. הפרוטוקול הינו סינכרוני, וכל ההודעות כולל כולן הינן מצד הלקוח (לא כולל התחברות שכולל בתוכו הודעה מהשרת אל הלקוח three way handshake).

3. + 4. נכתב במסמך אפיון.

5. אין שגיאות, הפרוטוקול בפרויקט הינו רק בשביל שהלקוח יעדכן את השרת על מצביו השונים, אין צורך בשגיאות אשר אף צד לא תלוי בשני, וכאשר קיימת תקלה (כמו למשל הודעה לא לפי פרוטוקול) מתעלמים ממנה (אם השגיאה חוזרת השרת מעיף את הלקוח).

לוגיקת פעולות



תמונה שמכילה טקסט, תרשים, קו, מקביל

התיאור נוצר באופן אוטומטי