# תיק פרויקט - עבודת גמר בהנדסת תוכנה



**מוסד לימודים: תיכון ע"ש חיים הרצוג כפר סבא**

**שם העבודה: Custom VPN System**

**תלמיד: איתמר שייה**

**ת.ז: 330864489**

**שם המנחה: אופיר שביט**

**שם החלופה: יוסי זהבי**

**תאריך הגשה: מאי 2025**



# תוכן עניינים

Contents

[תיק פרויקט - עבודת גמר בהנדסת תוכנה 1](#_Toc197901696)

[תוכן עניינים 2](#_Toc197901697)

[מבוא 4](#_Toc197901698)

[ייזום 4](#_Toc197901699)

[פירוט תיאור המערכת 8](#_Toc197901700)

[תיאור תחום הידע 12](#_Toc197901701)

[פירוט היכולות בצד שרת 12](#_Toc197901702)

[פירוט היכולות בצד לקוח 14](#_Toc197901703)

[מבנה / ארכיטקטורה של הפרויקט 16](#_Toc197901704)

[תיאור הארכיטקטורה של המערכת המוצעת 16](#_Toc197901705)

[תיאור הטכנולוגיה הרלוונטית 17](#_Toc197901706)

[תיאור זרימת המידע במערכת 19](#_Toc197901707)

[תיאור האלגוריתמים המרכזיים 24](#_Toc197901708)

[תיאור סביבת הפיתוח 29](#_Toc197901709)

[תיאור פרוטוקול התקשורת 30](#_Toc197901710)

[תיאור מסכי המערכת 32](#_Toc197901711)

[תיאור מבני הנתונים 35](#_Toc197901712)

[סקירת חולשות ואיומים 38](#_Toc197901713)

[מימוש הפרויקט 40](#_Toc197901714)

[פירוט מימוש המערכת 40](#_Toc197901715)

[ניתוח קטעי קוד מרכזיים 45](#_Toc197901716)

[מסמך בדיקות 52](#_Toc197901717)

[מדריך למשתמש 57](#_Toc197901718)

[עץ קבצים 57](#_Toc197901719)

[התקנת המערכת 58](#_Toc197901720)

[סיכום אישי/רפלקציה 60](#_Toc197901721)

[ביבליוגרפיה 63](#_Toc197901722)

[נספחים – קוד הפרויקט 64](#_Toc197901723)

# מבוא

## ייזום

**VPN** ((Virtual Private Network הינו למעשה שירות דיגיטלי בתוך הרשת להעברת מידע פרטי בצורה אמינה, בטוחה ומאובטחת. מטרתו של ה – VPN הינה לספק למשתמש מעין "ערוץ" מאובטח על גבי רשת שבה אין תשתית מאובטחת, לדוגמה ברשת ציבורית. השירות המוצע על ידי שרת ה – VPN דואג ליצירת תשתית מאובטחת עבור המשתמש באמצעות יצירת התחברות מאובטחת ולאחר מכן שימוש בתור פרוקסי של המשתמש להעברת המידע עם אנונימיות מוחלטת מצד המשתמש. אפרט את האמור לעיל בהמשך.

ישנה בעייתיות מובהקת בעת גלישה ברשת ציבורית ללא אבטחה והינה שגורמים ברשת מסוגלים להבין ולנתח פרטים אודות השימוש שלנו באינטרנט ואף "להסניף" מידע פרטי מהתעבורה ברשת. אי לכך החלטתי לממש את המענה לבעיה הזאת באמצעות יצירת מערכת Custom VPN.  
  
כמו כן, היה לי עניין רב ביצירת פרויקט המתעמק ברשתות, ובד בבד מערב גם קריפטוגרפיה. שיקול נוסף לפרויקט עבורי היה מימוש הכלי OpenSSH, אך לבסוף בחרתי במימוש מערכת VPN היות והרעיון לפרויקט הזה עניין אותי יותר. אני מאמין כי יצירת שרת VPN הינה דרך נהדרת לחקור ולהעמיק את הידע שלי בתחומים הללו. מהותו של שרת VPN הינה הקמת תשתית רשת וירטואלית, ללא צורך בממשק רשת פיזי, אשר תספק שירות בצורה של "מנהרה" מוצפנת על גבי הרשת הפיזית, אשר תהפוך את הגלישה והשימוש באינטרנט למאובטחת, ואף תאפשר לגשר בין ישויות שונות הנמצאות ברשתות מקומיות אחרות (LAN שונה). המערכת תעשה זאת באמצעות האספקט של הרשת הוירטואלית, המנוהלת ע"י השרת.

בפרויקט זה אני צופה מספר אתגרים עיקריים אשר אני עלול להיתקל בהם:

* אימפלמנטציה של אנקפסולציה נכונה ואפקטיבית של פאקטות
* ניהול מצבי רשת
* קביעת פרוטוקול VPN מתאים עבור צורכי השרת שלי ומימושו
* לוודא שהמידע אשר מועבר חסוי ומאובטח בצורה טובה
* ניהול מספר חיבורים בו זמנית

המערכת מיועדת למשתמשים הבאים:

* משתמשים אשר זקוקים לגישה בטוחה לרשתות מרוחקות
* רוצים לשמור על הפרטיות שלהם ברשת
* זקוקים לאבטחה מוגברת עבור התעבורה שלהם
* זקוקים לפתרון VPN נגיש וחינמי

הלקוחותהמיועדים עבור המערכת שהינני מפתח, הינם רבים ונעים מהאדם הממוצע ללא ידע טכנולוגי רב, אך מעוניין להיות בטוח במרחב הדיגיטלי ולהשתמש באמצעי היסייע לו בכך, ועד חברות טכנולוגיות המעוניינות להשתמש באמצעי מאובטח המגשר על בין רשתות שונות.   
על כן, קהל היעד של הפרויקט שלי נרחב ומספק מענה לביקוש הגובר לאבטחת הזהות של המשתמשים ברשת. בנוסף לכך, הפרויקט יספק אבטחה מוגברת עבור המידע המועבר ברשת ולתקשורת המתרחשת, על מנת לענות לצורכיהם של המשתמשים שלא יהיה ניתן לפענח את המידע האישי שלהם שמעבירים על גבי הרשת בעת שימוש באינטרנט.  
כפישפורט לעיל, המטרות המרכזיות של המערכת הינן:



* ניהול הרשת הווירטואלית בצורה נכונה כך שתאפשר לאנשים מרשתות נפרדות לתקשר אחד עם השני.

.

* יצירת מנהרה מאובטחת – השרת יוצר tunnel מאובטחת בה ניתן להעביר מידע בין השרת ללקוח.
* להציע הצפנות חזקות אשר יוודאו כי המידע מאובטח כראוי.

ניתן לצפות שהמערכת תשיג את מטרותיה המרכזיות ותקנה למשתמשיה פרטיות ואבטחה בעת שימוש בכל רשת שהיא.  
היא תשים דגש על פרוטוקול הצפנה נכון ומתקדם, שיאפשר העברת מידע ללא סכנה שגורמים עוינים ישתמשו במידע  
המועבר באמצעות מתקפת MITM לדוגמה.  
רוב שירותי ה – VPN הקיימים כיום דורשים תשלום מסוים עבור שימוש בשרתים שלהם. בהשוואה אליהם, השימוש בשרת ה – VPN שלי יהיה חינמי ונגיש לכולם.  
למרות זאת, בניגוד לספקי שירות VPN אשר יש להם מספר שרתים באזורים גאוגרפיים שונים ובכך נותנים מענה לבעיה חסימת תוכן עבור אזור גיאוגרפי מסוים וכדו', לא אענה על צורך זה.  
הפרויקט שלי לא יהווה כתחליף ישיר לשירותי ה – VPN המוצעים היום, אך ישמש כאפשרות נוספת, נגישה ונוחה לשימוש, שיהיה אפשר להשתמש בה לצורך הבסיסי של אבטחה והסתרת הנוכחות הדיגיטלית.   
בעתסקירת טכנולוגיות הפרויקט, הטכנולוגיה איננה חדשה ובלתי מוכרת. קיימים כיום גופים רבים המציעים שירותים של שרתי VPN. הפרויקט כשלעצמו איננו עוסק בחומרה, אלא בתוכנה המרכיבה את המודל. על כן אין מגבלה משמעותית החלה עליי בעת ביצוע הפרויקט. למרות זאת, ישנה מגבלה הקיימת והינה כפי שצוין לעיל, החוסר בשרתים נוספים באיזורים גיאוגרפיים שונים, ובכך אי מתן פתרון להגבלת תוכן על סמך מיקום גיאוגרפי. אף על פי כן, מגבלה זו איננה הכרחית בפן אבטחת המידע ויצירת ה"מנהרה" בה יועבר המידע.  
**סקירת טכנולוגיית הפרויקט:**

הפרויקט משתמש במספר טכנולוגיות וכלים מתקדמים:

* **Python** כשפת תכנות ראשית בצד השרת והלקוח
* **ממשק TUN** ליצירת ממשקי רשת וירטואליים בלינוקס
* **פרוטוקול UDP** כבסיס לתקשורת בין הלקוח לשרת
* **הצפנה אסימטרית (ECDH)** לחילופי מפתחות
* **הצפנה סימטרית (AES-GCM)** להצפנת התעבורה
* **SQLite** לניהול מסד הנתונים של המשתמשים

**תיחום הפרויקט**

**התחומים בהם הפרויקט עוסק:**

* מערכות הפעלה לינוקס
* רשתות מחשבים ופרוטוקולי תקשורת
* חיבור ממשקי רשת וירטואליים
* הצפנת תעבורת רשת
* ניתוב חבילות מידע
* אבטחת מידע והצפנה
* ניהול משתמשים ואימות
* פיתוח ממשק משתמש גרפי

**התחומים בהם המערכת אינה מטפלת:**

* תמיכה במערכות הפעלה אחרות (Windows, macOS)
* תמיכה בפרוטוקולים אחרים (PPTP, L2TP, SSTP)
* אופטימיזציה לביצועים בתנאי רשת קיצוניים
* דחיסת תעבורה
* ניטור תעבורה וניתוח מתקדם
* שירותי DNS ייעודיים
* הפעלה כשירות ענן

הפרויקט אינו עוסק בתחומים נוספים כגון Mobile, פיתוח חומרה, למידת מכונה ובינה מלאכותית.

## פירוט תיאור המערכת

**תיאור מפורט יותר של המערכת**

מערכת ה – VPN מורכבת משני רכיבים עיקריים: שרת ולקוח.

**שרת ה – VPN:**

* אחראי על יצירת ממשק רשת וירטואלי (TUN)
* מקבל בקשות התחברות מלקוחות
* מבצע אימות משתמשים מול מסד נתונים
* מקצה כתובות IP וירטואליות ללקוחות
* מנתב תעבורה בין הלקוחות והאינטרנט
* אחראי על הצפנת ופענוח התעבורה
* מנהל שכבת אמינות לפרוטוקול התקשורת

**לקוח ה – VPN:**

* מספק ממשק משתמש גרפי נוח למשתמש
* יוצר ממשק רשת וירטואלי בצד המשתמש
* מתקשר עם השרת דרך פרוטוקול UDP מאובטח
* מצפין ומפענח את תעבורת הרשת
* מנתב את כל תעבורת המשתמש דרך המנהרה המוצפנת
* מטפל בהתאוששות מנפילות תקשורת
* מספק מידע סטטוס על החיבור למשתמש

**פירוט היכולות לכל סוג משתמש**

**משתמשי קצה (דרך ממשק המשתמש הגרפי):**

* רישום משתמש חדש (הרשמה)
* התחברות למערכת עם שם משתמש וסיסמה
* התחברות לשרת ה - VPN
* ניתוק מהשרת
* צפייה בסטטוס החיבור
* צפייה בכתובת ה – IP הווירטואלית

**מנהל מערכת (בצד השרת):**

* הגדרת ותצורת השרת
* ניהול רשת ה-VPN הווירטואלית
* ניהול טווח כתובות IP
* ניהול משתמשים (הוספה, הסרה, שינוי הרשאות)
* ניטור חיבורים פעילים
* ניהול הגדרות אבטחה
* עדכון ותחזוקת המערכת

**פירוט הבדיקות המתוכננות:**

|  |  |  |
| --- | --- | --- |
| **סוג בדיקה** | **מטרה** | **שיטת ביצוע** |
| בדיקת התקנה | וידוא שהמערכת מותקנת כראוי | התקנה על מערכת נקייה ובדיקת הפונקציונליות הבסיסית |
| בדיקת תקשורת בסיסית | וידוא תקשורת תקינה בין הלקוח לשרת | בדיקת הקמת חיבור בין הלקוח לשרת |
| בדיקת הצפנה | וידוא שהתעבורה מוצפנת כראוי | ניטור חבילות הרשת ווידוא שהתוכן מוצפן |
| בדיקת ניתוב | וידוא שכל התעבורה עוברת דרך ה - VPN | בדיקה שכתובת ה IP החיצונית הנראית היא של השרת |
| בדיקת ממשק המשתמש | וידוא שממשק המשתמש עובד כמצופה | בדיקת כל הפונקציות דרך הממשק הגרפי |
| בדיקת אמינות | וידוא שהמערכת מתאוששת מנפילות | סימולציה של בעיות רשת ובדיקת התאוששות |
| בדיקת עומסים | וידוא תפקוד תקין תחת עומס | העמסת המערכת במספר רב של חיבורים |

**תכנון וניהול הלו"ז:**

|  |  |  |  |
| --- | --- | --- | --- |
| **שלב** | **תיאור** | **לו"ז מתוכנן** | **לו"ז בפועל** |
| מחקר ואפיון | חקר טכנולוגיות, הגדרת דרישות ותכנון ראשוני | שבועיים | 3 שבועות |
| תכנון מפורט | תכנון ארכיטקטורה, פרוטוקולים, ממשקים | שבועיים | 3 שבועות |
| פיתוח ליבה | מימוש פרוטוקול תקשורת וממשקי רשת ווירטואלים TUN | 4 שבועות | 4 שבועות |
| פיתוח שרת | מימוש צד שרת, ניהול משתמשים וניתוב חלקי | 3 שבועות | 3 שבועות |
| פיתוח לקוח | מימוש צד לקוח וממשק משתמש גרפי | 3 שבועות | 3 שבועות |
| אינטגרציה | שילוב כלל הרכיבים ועבודה על ניתוב מלא בהתאם לפרוטוקול ולארכיטקטורה | שבועיים | 3 שבועות |
| בדיקות | ביצוע מערך בדיקות מקיף | שבועיים | שבוע |
| תיעוד | הכנת תיעוד טכני ומדריך למשתמש | שבועיים | שבועיים |

**ניהול הסיכונים בפרויקט:**

|  |  |  |  |
| --- | --- | --- | --- |
| סיכון | דרגת חומרה | דרכי התמודדות מתוכננות | התמודדות בפועל |
| בעיות תאימות עם גרסאות לינוקס שונות | בינונית | בדיקת תאימות עם מספר הפצות לינוקס פופולריות | הוגדרו הדרישות המינימליות והותאם הקוד לתמיכה בהפצות העיקריות |
| קשיים בהגדרת ממשקי TUN | גבוהה | לימוד מעמיק של ממשקי TUN וביצוע PoC מוקדם | יישום הדרגתי והתמודדות עם בעיות בזמן אמת |
| בעיות אבטחה בפרוטוקול התקשורת | גבוהה | שימוש בספריות הצפנה מוכחות וסקירת קוד | שימוש בספריית cryptography ובחינה של כל שלבי הצפנה |
| ביצועים לא מספקים | בינונית | תכנון לאופטימיזציה ובדיקות ביצועים | בוצעו מספר שיפורי ביצועים וכיוונון של מנגנוני התקשורת |
| קשיים בניתוב התעבורה | גבוהה | תכנון מנגנון ניתוב מפורט ומבוסס על דוגמאות קיימות | יישום הדרגתי ופתרון בעיות ספציפיות |
| תלות בהרשאות root | נמוכה | תיעוד ברור של הדרישות | הוגדרו במדויק ההרשאות הנדרשות |

# תיאור תחום הידע

## פירוט היכולות בצד שרת

**ניהול ומימוש ממשק TUN:**

* **מהות היכולת:** יצירה וניהול של ממשק רשת וירטואלי בשרת
* **אוסף פעולות:** הקמת ממשק TUN, קביעת כתובת IP לממשק, הגדרת,MTU טיפול בחבילות
* **אובייקטים נחוצים:** ממשק מערכת הפעלה, מודול fcntl, struct

**ניהול חיבורי לקוחות:**

* **מהות היכולת:** ניהול מספר חיבורי לקוחות במקביל
* **אוסף פעולות:** קבלת בקשות התחברות, אימות, הקצאת כתובות IP, מעקב אחר חיבורים פעילים
* **אובייקטים נחוצים:** מחלקת VPNServer, מבני נתונים לניהול חיבורים, מנגנון הקצאת IP

**ניהול משתמשים:**

* **מהות היכולת:** ניהול מסד נתוני משתמשים ואימות
* **אוסף פעולות:** רישום משתמש חדש, אימות משתמש, ניהול מפתחות אבטחה, ניהול סשנים
* **אובייקטים נחוצים:** מחלקת ServerUserManager, מסד נתונים SQLite

**העברת נתונים מוצפנת:**

* **מהות היכולת:** הצפנה ופענוח של תעבורת הרשת
* **אוסף פעולות:** חילופי מפתחות, הצפנה, פענוח, אימות הודעות
* **אובייקטים נחוצים:** מחלקת VPNCrypto, ספריית cryptography

**ניתוב תעבורה:**

* **מהות היכולת:** ניתוב תעבורת רשת בין לקוחות והאינטרנט
* **אוסף פעולות:** קריאת חבילות מממשק TUN, ניתוח כתובות, ניתוב למקום המתאים
* **אובייקטים נחוצים:** טבלאות ניתוב iptables / NAT

**ניהול פרוטוקול אמין:**

* **מהות היכולת:** הבטחת אמינות התקשורת מעל UDP
* **אוסף פעולות:** מספור חבילות, אישורי קבלה, שליחה חוזרת, טיפול בחבילות שלא בסדר
* **אובייקטים נחוצים:** מחלקת ReliableMessageHandler

## פירוט היכולות בצד לקוח

**ניהול ומימוש ממשק TUN:**

* **מהות היכולת:** יצירה וניהול של ממשק רשת וירטואלי בלקוח
* **אוסף פעולות:** הקמת ממשק TUN, קביעת כתובת IP לממשק, הגדרת MTU, טיפול בחבילות
* **אובייקטים נחוצים:** ממשק מערכת הפעלה, מודול fcntl מודול struct

**התחברות לשרת:**

* **מהות היכולת:** יצירת חיבור מאובטח לשרת ה-VPN
* **אוסף פעולות:** יצירת חיבור UDP, הקמת מנהרה מוצפנת, אימות מול השרת
* **אובייקטים נחוצים:** מחלקת VPNClient, מחלקת ReliableMessageHandler

**העברת נתונים מוצפנת:**

* **מהות היכולת:** הצפנה ופענוח של תעבורת הרשת
* **אוסף פעולות:** חילופי מפתחות, הצפנה, פענוח, אימות הודעות
* **אובייקטים נחוצים:** מחלקת VPNCrypto, ספריית cryptography

**ניהול ממשק משתמש גרפי:**

* **מהות היכולת:** הצגת ממשק גרפי למשתמש
* **אוסף פעולות:** הצגת חלונות וטפסים, קבלת קלט מהמשתמש, הצגת מידע סטטוס
* **אובייקטים נחוצים:** מחלקות PyQt5

**ניהול חיבור VPN:**

* **מהות היכולת:** ניהול מצב החיבור
* **אוסף פעולות:** התחברות, ניתוק, חידוש חיבור, טיפול בשגיאות
* **אובייקטים נחוצים:** מחלקת VPNClient, מחלקת ConnectionState

**ניהול הגדרות ותצורה:**

* **מהות היכולת:** שמירה וטעינה של הגדרות המשתמש
* **אוסף פעולות:** שמירת פרטי התחברות, טעינה של הגדרות שמורות
* **אובייקטים נחוצים:** קבצי הגדרות, מחלקות לטיפול בהגדרות

להלן דוגמה מפורטת יותר ליכולת אחת:

**שם היכולת: חילופי מפתחות אבטחה**

**מהות:** ביצוע חילופי מפתחות מאובטחים בין הלקוח לשרת ליצירת מפתח סודי משותף

**אוסף יכולות נדרשות:**

* יצירת זוג מפתחות פרטי/ציבורי בצד הלקוח (ECDH)
* שליחת המפתח הציבורי לשרת
* קבלת המפתח הציבורי של השרת
* יצירת מפתח סודי משותף מהמפתח הפרטי והמפתח הציבורי של השרת
* גזירת מפתח הצפנה מהסוד המשותף
* יצירת מזהה סשן להצפנה

**אובייקטים נחוצים:**

* מחלקת VPNCrypto: לביצוע הפעולות הקריפטוגרפיות
* מחלקתReliableMessageHandler: לשליחת וקבלת הודעות באופן אמין
* ספריית cryptography.hazmat: לשימוש בפונקציות הצפנה מתקדמות
* מחלקות שרת ולקוח: לביצוע חילופי המפתחות וגזירת מפתחות ההצפנה

# מבנה / ארכיטקטורה של הפרויקט

## תיאור הארכיטקטורה של המערכת המוצעת

**תיאור החומרה**

המערכת מבוססת על ארכיטקטורת לקוח-שרת, כאשר שני הצדדים מבוססים על מערכת ההפעלה לינוקס.

ארכיטקטורת החומרה מורכבת מהרכיבים הבאים:

**בצד השרת:**

* שרת לינוקס עם חיבור אינטרנט מהיר ויציב
* כרטיס רשת פיזי המחובר לאינטרנט
* כרטיס רשת וירטואלי (TUN) לניהול התעבורה הפנימית

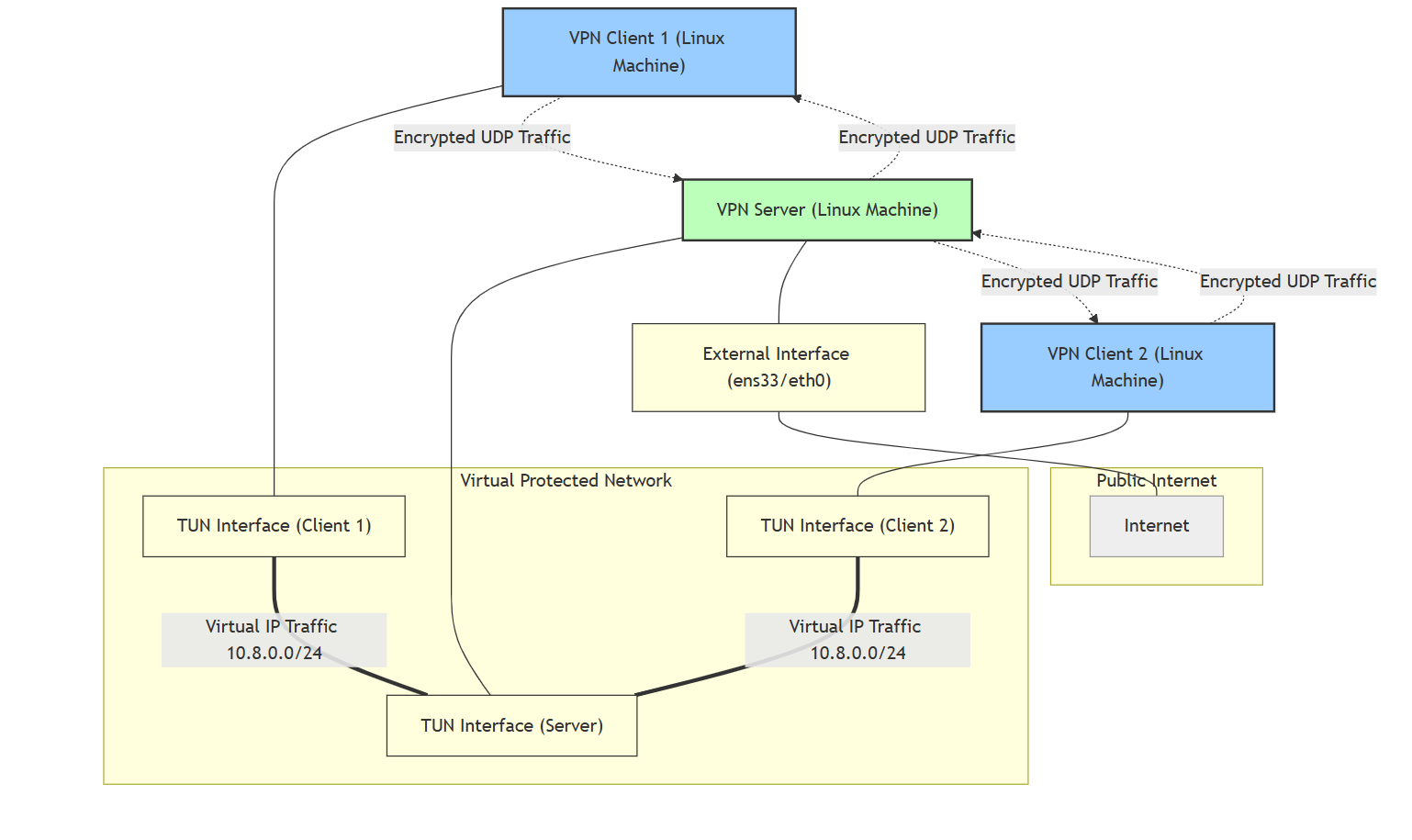
**בצד הלקוח:**

* מחשב לינוקס עם חיבור אינטרנט כלשהו
* כרטיס רשת פיזי המחובר לאינטרנט
* כרטיס רשת וירטואלי (TUN) לניתוב התעבורה דרך המנהרה המוצפנת

הקשרים בין הרכיבים השונים מבוססים על שני סוגי תקשורת:

1. **תקשורת פיזית :** חיבור UDP בין כרטיסי הרשת הפיזיים של הלקוח והשרת.
2. **תקשורת לוגית :** העברת חבילות IP בין ממשקי ה TUN - הווירטואליים, מנותבות דרך המנהרה המוצפנת.

**תרשים המציג את הקשרים בין רכיבי החומרה של המערכת:**



## תיאור הטכנולוגיה הרלוונטית

**שפת תכנות**

Python 3 נבחרה כשפת התכנות העיקרית לפרויקט בשל:

* פשטות ובהירות הקוד
* ריבוי ספריות תומכות לרשתות, הצפנה וממשקי משתמש
* תמיכה טובה בלינוקס
* מהירות פיתוח

**מערכת הפעלה**

לינוקס נבחרה כמערכת ההפעלה בשל:

* תמיכה מעולה במנגנוני רשת מתקדמים
* גישה נוחה לממשקי TUN/TAP
* יכולת לשנות טבלאות ניתוב בקלות יחסית
* פתיחות ופרויקטים דומים שמימשו טכנולוגיות דומות

**תקשורת**

מערכת התקשורת מבוססת על:

* **פרוטוקול UDP** כבסיס לתקשורת בין הלקוח לשרת (בחירה זו נעשתה בשל הגמישות והיעילות של UDP למרות חוסר האמינות שלו)
* **פרוטוקול אמינות מותאם אישית** שנבנה מעל UDP להבטחת העברת המידע באופן מהימן
* **מנגנון הצפנה** המבוסס על ECDH לחילופי מפתחות ו-AES-GCM להצפנת התעבורה

**תחומי עניין מרכזיים**

1. **רשתות וירטואליות**: יצירה וניהול של ממשקי TUN
2. **אבטחת מידע**: הצפנה, אימות, ואבטחת התקשורת
3. **פרוטוקולי רשת**: תכנון מותאם אישית של פרוטוקול תקשורת מעל UDP
4. **ניהול משתמשים**: מערכת רישום, אימות וניהול משתמשים
5. **ממשקי משתמש**: פיתוח ממשק משתמש גרפי ידידותי

## תיאור זרימת המידע במערכת

**זרימת הנתונים בחיבור לשרת**

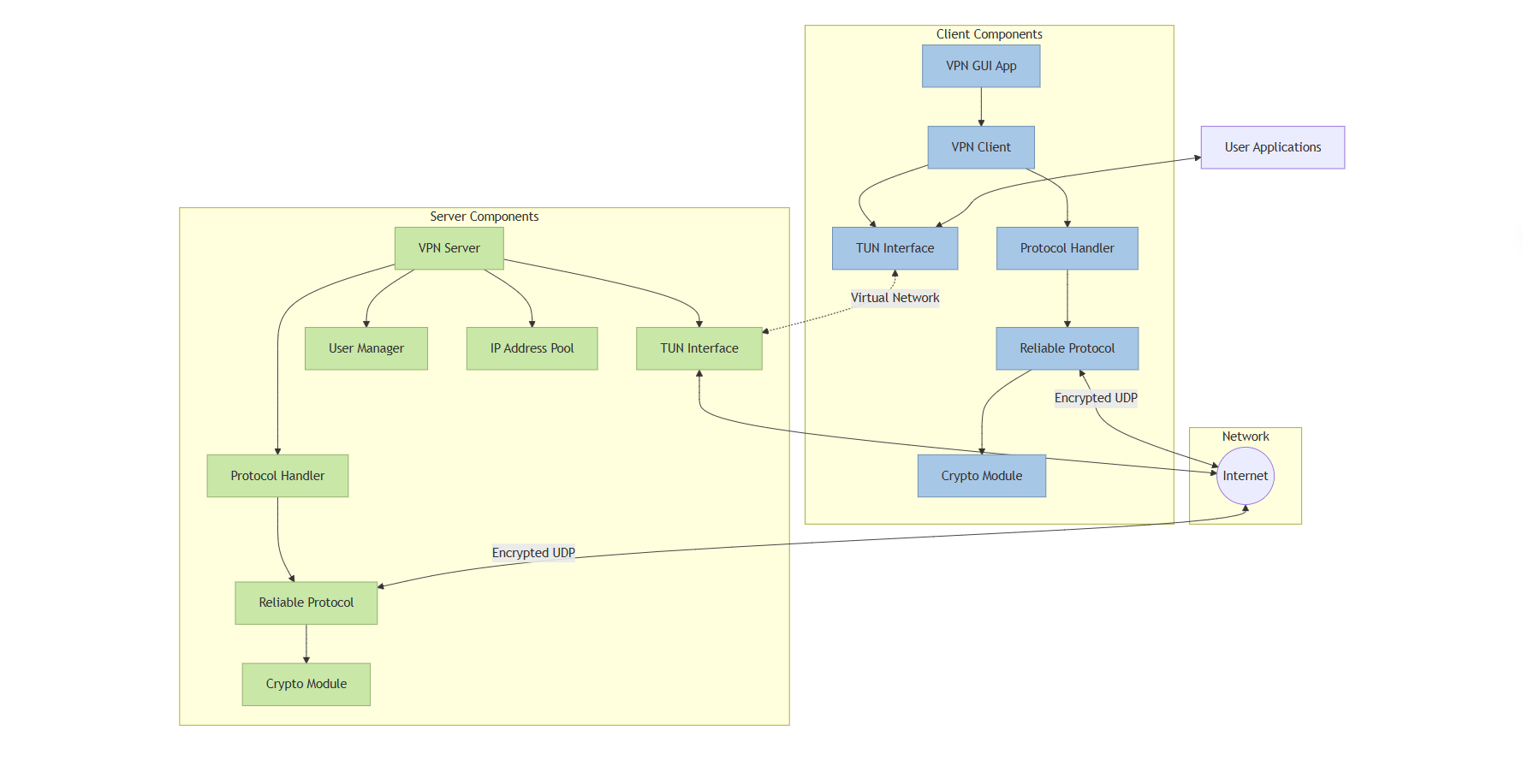
1. לקוח שולח בקשת התחברות לשרת
2. שרת מאמת את זהות הלקוח
3. הלקוח והשרת מבצעים החלפת מפתחות באופן מאובטח
4. השרת מקצה כתובת IP וירטואלית ללקוח
5. השרת שולח נתוני תצורה ללקוח
6. הלקוח מגדיר את ממשק ה – TUN עם הפרמטרים שהתקבלו
7. הלקוח ושרת מעבירים ביניהם אישור התחברות

**זרימת הנתונים בהעברת מידע**

1. יישום בצד הלקוח שולח נתונים דרך ממשק ה - TUN
2. הלקוח מצפין את הנתונים ושולח אותם לשרת דרך UDP
3. השרת מפענח את הנתונים ומעביר אותם לאינטרנט
4. תשובה מגיעה לשרת מהאינטרנט
5. השרת מצפין את התשובה ושולח אותה ללקוח המתאים
6. הלקוח מפענח את הנתונים ומעביר אותם ליישום דרך ממשק ה - TUN

* **תרשימים המתארים את זרימת הנתונים בהמשך**

**זרימת מידע כללית**

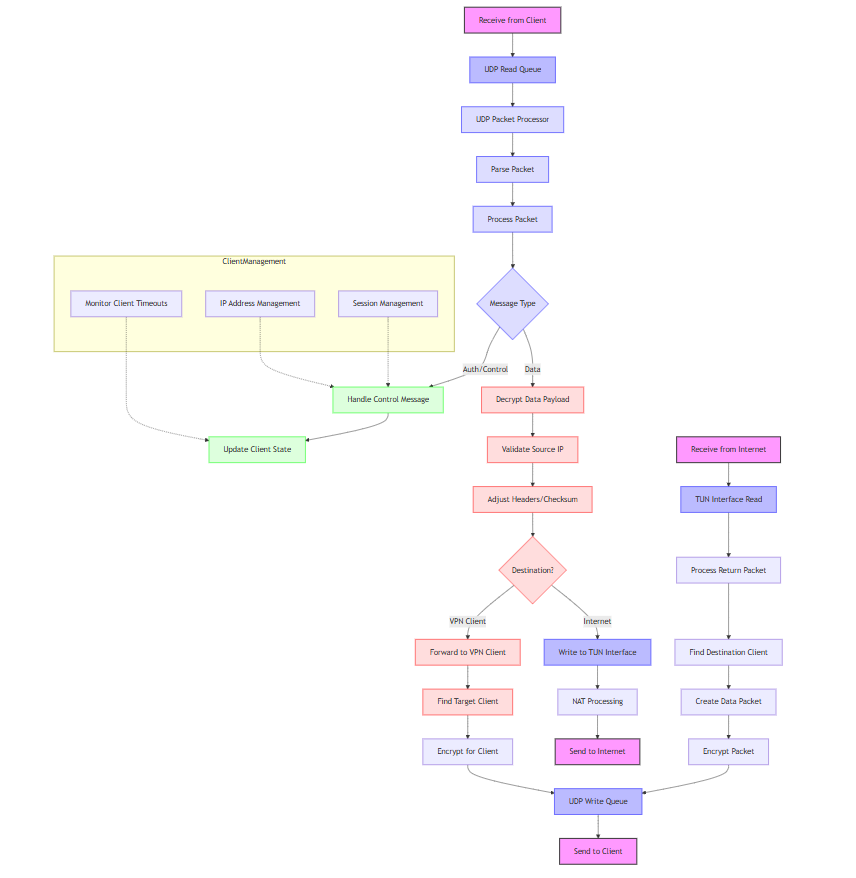
****

**תרשים זרימת מידע בין לקוח ושרת:**

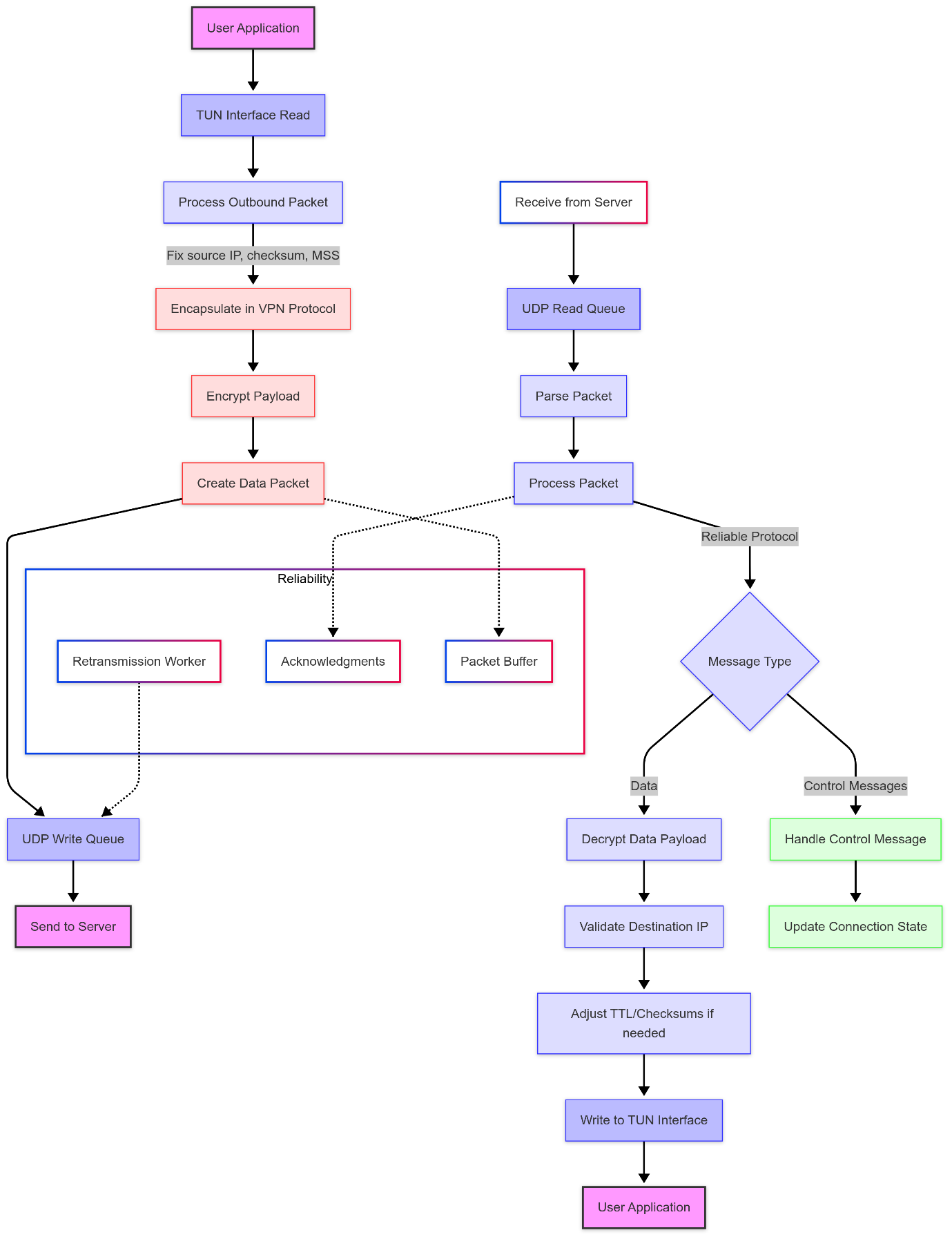
A screenshot of a computer screen

AI-generated content may be incorrect.

**תרשים זרימת וניהול מידע בצד שרת:**



**תרשים זרימת וניהול מידע בצד לקוח:**



## תיאור האלגוריתמים המרכזיים

**מעקב אחר פקטות המועדות החוצה וניתוב פקטות תגובה**

בעיה מהותית בעת יצירת מערכת VPN הינה מעקב אחר הפקטות העוברות דרך השרת, היות וישנה תעבורת רשת רבה מצד, ויש צורך בביצוע מעקב מדויק אחר כל פקטה על מנת לנתב את התגובה בחזרה למקור.

**ניסוח הבעיה האלגוריתמית:**

* יש צורך בניתוב תגובות בחזרה ללקוח המתאים
* ישנן פקטות בעלות נתונים ויעדים שונים מכל User Application
* על מנת להשיג ניתוב מדויק יש לבצע מעקב אחר תעבורת הרשת היוצאת/נכנסת

**סקירת פתרונות קיימים:**

1. **IPTables Masquerading**
2. **Custom NAT**

**הפתרון הנבחר:** בחרתי להשתמש בכלי בלינוקס IPTables Masquerading בשל:

* הכלי מציע שימוש בטבלת המעקב של מערכת ההפעלה ברמת הקרנל
* הוא מאפשר מעקב מדויק אחר תעבורת הרשת
* יאפשר לשמור על ביצועים טובים יותר ללא צורך בשימוש בזיכרון רב ומאגרי מידע עבור כל פקטה
* [קישור להסבר פשוט על הכלי IPTables Masquerade](https://billauer.co.il/ipmasq-html.html)
* [קישור להסבר פשוט על ה – Conntrack table של הקרנל בלינוקס](https://blog.cloudflare.com/conntrack-tales-one-thousand-and-one-flows/)

**פרוטוקול אמינות מעל UDP**

אחד האתגרים המרכזיים בפרויקט הוא יצירת מנגנון אמינות מעל פרוטוקול UDP שאינו אמין מטבעו.

**ניסוח הבעיה האלגוריתמית:**

* UDP אינו מספק אישורי קבלה או סדר חבילות
* חבילות עלולות להיאבד, להגיע לא בסדר, או להגיע מספר פעמים
* יש צורך במנגנון שיבטיח העברת כל המידע באופן אמין

**סקירת פתרונות קיימים:**

1. **TCP**  - פרוטוקול אמין, אבל מורכב מדי ופחות מתאים למנהרות VPN
2. **RUDP (Reliable UDP)**  - מספר מימושים קיימים, אך לא תמיד מתאימים לצרכי VPN
3. **DTLS**  - פרוטוקול אמין מעל UDP אך מתמקד באבטחה יותר מאשר באמינות
4. **פרוטוקול מותאם אישית** - פתרון המותאם ספציפית לצרכי ה VPN

**הפתרון הנבחר:** בחרתי לפתח פרוטוקול אמינות מותאם אישית המבוסס על העקרונות הבאים:

* מספור רציף של חבילות
* אישורי קבלה (ACKs)
* חלון גלישה (sliding window)
* שליחה חוזרת של חבילות שלא התקבלו
* טיפול בחבילות שמגיעות לא בסדר

**פסאודו-קוד:**

כאשר שולחים הודעה:

הקצה מספר סידורי להודעה

שמור את ההודעה בתור 'לא מאושרת'

שלח את ההודעה

הפעל טיימר לשליחה חוזרת

כאשר מקבלים הודעה:

אם ההודעה דורשת אמינות:

שלח אישור קבלה (ACK) עם מספר ההודעה

אם ההודעה היא אישור קבלה (ACK):

סמן את ההודעה המתאימה כ'מאושרת'

בטל את הטיימר לשליחה חוזרת

כאשר טיימר לשליחה חוזרת פג:

אם ההודעה עדיין 'לא מאושרת':

אם מספר הניסיונות < מקסימום ניסיונות:

שלח את ההודעה שוב

הגדל את הטיימר (exponential backoff)

הפעל את הטיימר מחדש

אחרת:

דווח על כישלון בשליחה

**חילופי מפתחות מאובטחים**

**ניסוח הבעיה האלגוריתמית:**

* יש צורך להחליף מפתחות בין הלקוח לשרת דרך תווך לא מאובטח
* המפתחות ישמשו להצפנת התקשורת בהמשך
* התהליך צריך להיות בטוח מפני האזנה או תקיפת man-in-the-middle

**סקירת פתרונות קיימים:**

1. **RSA -** אלגוריתם א-סימטרי חזק, אך איטי יחסית ומצריך מפתחות ארוכים
2. **Diffie-Hellman** - פרוטוקול לחילופי מפתחות, אך פגיע לתקיפות man-in-the-middle ללא אימות
3. **ECDH (Elliptic Curve Diffie-Hellman)** - גרסה יעילה יותר של Diffie-Hellman המבוססת על עקומות אליפטיות
4. **SSL/TLS -** פרוטוקולים מלאים לאבטחת תקשורת, אך מורכבים לצרכי המערכת, ששואפת לשמור על ביצועים טובים בהתחשב בסביבת העבודה.

**הפתרון הנבחר -** בחרתי בפרוטוקול ECDH בשל:

* יעילות גבוהה (מפתחות קצרים יותר)
* רמת אבטחה גבוהה
* תמיכה טובה בספריית cryptography של Python

**תיאור האלגוריתם:**

בצד הלקוח:

יצירת זוג מפתחות ECDH פרטי/ציבורי על בסיס עקומה אליפטית

שליחת המפתח הציבורי לשרת

בצד השרת:

קבלת המפתח הציבורי של הלקוח

יצירת זוג מפתחות ECDH פרטי/ציבורי המבוסס על העקומה המשותפת

חישוב המפתח המשותף באמצעות המפתח הפרטי והמפתח הציבורי של הלקוח

שליחת המפתח הציבורי של השרת ללקוח

בצד הלקוח:

קבלת המפתח הציבורי של השרת

חישוב המפתח המשותף באמצעות המפתח הפרטי והמפתח הציבורי של השרת

בשני הצדדים:

גזירת מפתח הצפנה מהמפתח המשותף באמצעות HKDF

שימוש במפתח ההצפנה לאבטחת התקשורת

**קישור לסרטון המסביר בצורה טובה על אלגוריתם ECDH:** [**https://www.youtube.com/watch?v=NF1pwjL9-DE&t=109s&ab\_channel=Computerphile**](https://www.youtube.com/watch?v=NF1pwjL9-DE&t=109s&ab_channel=Computerphile)

## תיאור סביבת הפיתוח

**פרוט כלי הפיתוח הדרושים לפיתוח**

* **שפת תכנות** :Python 3.8 ומעלה
* **מערכת הפעלה:** לינוקס Ubuntu 20.04 או מתקדם יותר (Debian 11 ומעלה)
* **ספריות Python עיקריות:**
  + Cryptography - לאלגוריתמי הצפנה
  + PyQt5 - לממשק משתמש גרפי
  + Scapy - לעבודה עם חבילות רשת
  + sqlite3 - לניהול מסד נתונים
* **כלי פיתוח:**
  + Git - לניהול גרסאות
  + Visual Studio Code - סביבת פיתוח
  + VMWare Workstation Pro – על מנת להריץ מכונה ווירטואלית עם גרסת לינוקס מתאימה לצורך בחינות
* **הרשאות מערכת:**
  + הרשאות root לצורך יצירת ממשקי TUN ושינוי טבלאות ניתוב

**פרוט הסביבה והכלים הנדרשים לבדיקות**

* **כלי בדיקת רשת:**
  + Wireshark - לניטור תעבורת רשת
  + Tcpdump - לתפיסה וניתוח של חבילות רשת
  + ping / traceroute -לבדיקת קישוריות
* **כלי בדיקת מסדי נתונים :**

DB Browser (SQLite)

## תיאור פרוטוקול התקשורת

**תיאור מילולי של פרוטוקול התקשורת**

פרוטוקול התקשורת המותאם לפרויקט זה מבוסס על UDP, עם שכבת אמינות ומעליו הצפנה. הפרוטוקול כולל מספר סוגי הודעות שונים, כשלכל סוג תפקיד מוגדר בתהליך התקשורת.

מבנה הודעה בסיסי:

* **כותרת (header)** - 17 בתים קבועים
* **נתונים (payload)** - אורך משתנה, מוצפן במרבית סוגי ההודעות
* **בדיקת שלמות (checksum)** - 4 בתים

מבנה הכותרת:

* **Magic number (4 בתים)** - לזיהוי הודעות פרוטוקול
* **גרסת פרוטוקול (1 בית)** - מאפשר תאימות לאחור עם גרסאות עתידיות
* **סוג הודעה (1 בית)** - מגדיר את סוג ההודעה
* **מספר רצף (4 בתים)** - לזיהוי ומספור הודעות
* **מספר אישור (4 בתים)** - לאישור קבלת הודעות
* **דגלים (1 בית)** - מכיל אינדיקטורים שונים בנוגע לתעבורת הרשת (SYN, ACK כו')
* **אורך נתונים (2 בתים)** - מציין את אורך חלק הנתונים בהודעה

**פירוט כלל ההודעות הזורמות במערכת:**

|  |  |  |
| --- | --- | --- |
| **שם ההודעה** | **נשלחת מ/אל** | **מבנה השדות בהודעה** |
| Request | לקוח לשרת | כותרת עם דגל SYN + נתונים המכילים פרטי לקוח (מזהה לקוח, גרסת פרוטוקול, יכולות) |
| Authenticate | שרת ללקוח | כותרת + אתגר אימות (16 בתים אקראיים) |
| AuthResponse | לקוח לשרת | כותרת + תשובת אימות HMAC של האתגר עם המפתח המשותף |
| Config | שרת ללקוח | כותרת + הגדרות רשת :כתובת IP וירטואלית, מסכת רשת, כתובת שרת, MTU , שרתי DNS, ניתובים |
| Establish | לקוח לשרת | כותרת + אישור קבלת הגדרות |
| Data | לקוח לשרת/שרת ללקוח | כותרת + פקטות רשת מוצפנות מהשכבה השלישית |
| KeepAlive | לקוח לשרת/שרת ללקוח | כותרת בלבד (לשמירה על חיבור פעיל) |
| Disconnect | לקוח לשרת/שרת ללקוח | כותרת עם דגל FIN + סיבת ניתוק (אופציונלי) |
| Error | לקוח לשרת/שרת ללקוח | כותרת + הודעת שגיאה |
| Ack | לקוח לשרת/שרת ללקוח | כותרת עם דגל ACK + מספר אישור |
| Retransmit | לקוח לשרת/שרת ללקוח | כותרת עם דגל RETRY + מספר רצף של הודעה לשליחה חוזרת |
| KeyExchangeInit | לקוח לשרת | כותרת + מפתח ציבורי של הלקוח |
| KeyExchangeResponse | שרת ללקוח | כותרת + מפתח ציבורי של השרת |

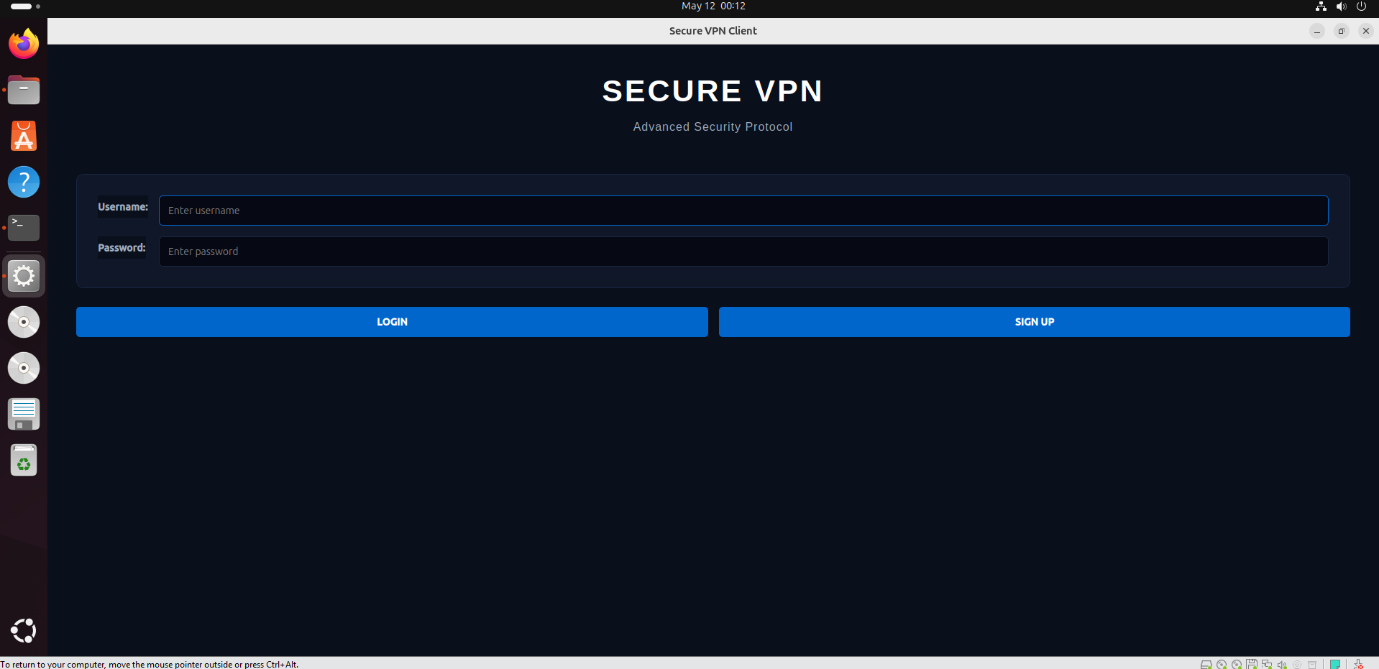
## תיאור מסכי המערכת

**מסך התחברות/הרשמה**

**כותרת/שם/תפקיד המסך:** מסך כניסה והרשמה

**תיאור המסך:** מסך זה מאפשר למשתמשים להתחבר למערכת עם שם משתמש וסיסמה קיימים או להירשם כמשתמשים חדשים. המסך מציג ממשק פשוט וידידותי עם שדות קלט לשם משתמש וסיסמה, ושני כפתורים: התחברות והרשמה.

**תמונת מסך:**

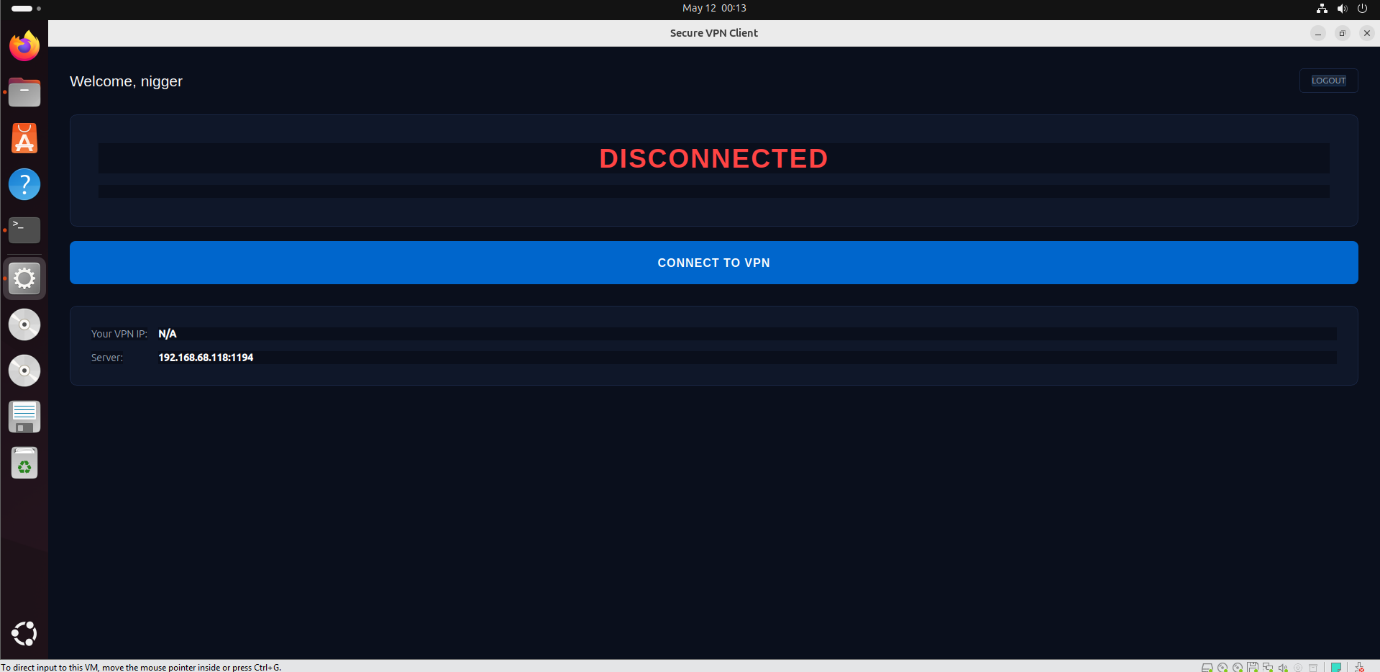
****

**מסך ראשי (Dashboard)**

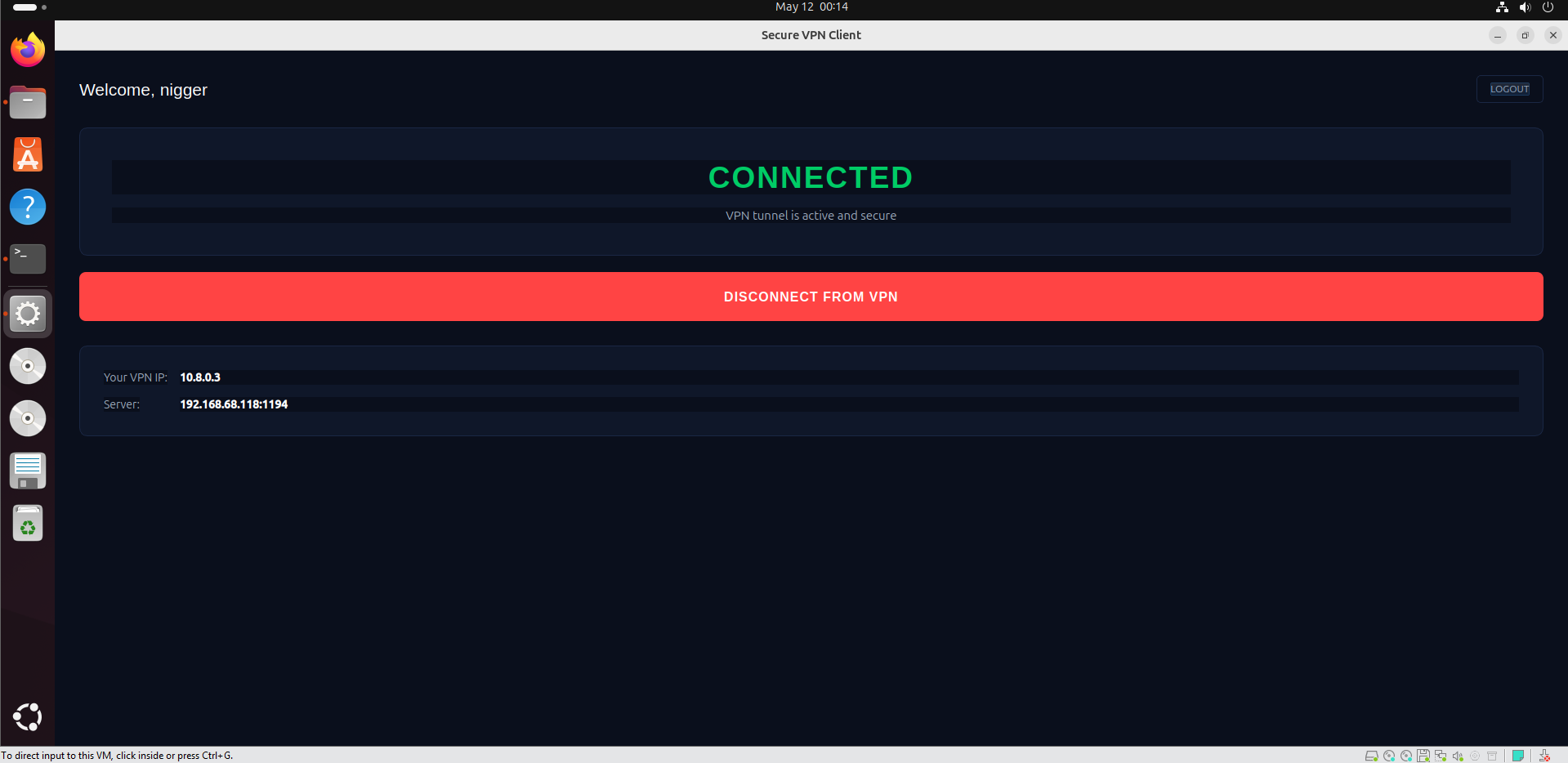
**כותרת/שם/תפקיד המסך:** מסך ראשי של האפליקציה

**תיאור המסך:** המסך הראשי מציג את מצב החיבור הנוכחי, מאפשר למשתמש להתחבר או להתנתק מה – VPN ומציג מידע על החיבור כגון כתובת ה – IP הווירטואלית והשרת אליו מחוברים.

**תמונת מסך (לא מחובר):**

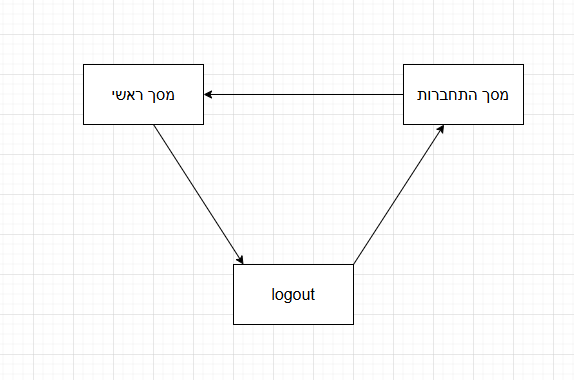
****

**תמונת מסך (מחובר):**

****

**תרשים מסכים**

המעברים בין המסכים פשוטים:



## תיאור מבני הנתונים

**מסד נתונים משתמשים**

**שם המסד:**vpn\_users.db , **סוג**: SQLite

**טבלת Microsoft:**

|  |  |  |  |
| --- | --- | --- | --- |
| **שם שדה** | **טיפוס** | **תיאור** | **דוגמה** |
| id | INTEGER | מזהה ייחודי, מפתח ראשי | 1 |
| username | TEXT | שם משתמש ייחודי | “user1” |
| password\_hash | TEXT | גיבוב של הסיסמה | "a1b2c3d4..." |
| salt | TEXT | ערך אקראי עבור אבטחת הסיסמה | "f7e6d5c4..." |
| created\_at | REAL | זמן יצירת המשתמש | 1620000000.0 |
| last\_login | REAL | זמן התחברות אחרון | 1620100000.0 |

**טבלת Sessions:**

|  |  |  |  |
| --- | --- | --- | --- |
| **שם שדה** | **טיפוס** | **תיאור** | **דוגמה** |
| id | INTEGER | מזהה ייחודי, מפתח ראשי | 1 |
| session\_token | TEXT | מפתח מזהה לסשן | "abc123def456..." |
| user\_id | INTEGER | מזהה המשתמש, מפתח זר | 1 |
| created\_at | REAL | זמן יצירת הסשן | 1620000000.0 |
| expires\_at | REAL | זמן פקיעת הסשן | 1620086400.0 |

**מבני הנתונים בזיכרון**

**ניהול לקוחות בשרת:**

active\_clients = {

"192.168.1.100:12345": {

"client\_id": "abc123",

"state": ConnectionState.CONNECTED,

"last\_activity": 1620000000.0,

"virtual\_ip": "10.8.0.2",

"mtu": 1400

}

}

client\_ips = {

"10.8.0.2": ("192.168.1.100", 12345)

}

client\_by\_id = {

"abc123": ("192.168.1.100", 12345)

}

**ניהול חבילות שטרם אושרו:**

python

unacked\_packets = {

1001: (packet\_data, timestamp, retries)

}

**תכונות לקוח:**

self.connection\_state = ConnectionState.DISCONNECTED

self.virtual\_ip = "10.8.0.2"

self.server\_virtual\_ip = "10.8.0.1"

self.subnet\_mask = "255.255.255.0"

self.dns\_servers = ["8.8.8.8", "8.8.4.4"]

self.routes = ["0.0.0.0/0"]

self.VPN\_MTU = 1400

## סקירת חולשות ואיומים

**שכבת האפליקציה**

**עבודה עם מסד נתונים – SQL Injection**

* **איום :** הזרקת קוד SQL זדוני דרך קלט משתמש
* **פתרון :** שימוש בפרמטרים מוגנים בכל שאילתות SQL, הימנעות מבניית שאילתות באמצעות חיבור מחרוזות

**תהליך ההתחברות - אימות וזיהוי:**

* **איום :** ניחוש סיסמאות, התקפות מילון
* **פתרון :** הגבלת ניסיונות כניסה, שימוש ב-PBKDF2 עם מספר גבוה של איטרציות לאחסון סיסמאות, שימוש ב-salt ייחודי לכל משתמש

**MITM (Man in the Middle):**

* **איום:**  יירוט התקשורת בין הלקוח והשרת
* **פתרון:** שימוש בהצפנה חזקה (AES-GCM) וחילופי מפתחות מאובטחים (ECDH)

**DOS/DDOS:**

* **איום:** הצפת השרת בבקשות
* **פתרון:** הגבלת קצב בקשות מכתובת IP יחידה, זיהוי ומניעת בקשות כפולות

**העלאת קבצים:**

* **איום:** העלאת קבצים זדוניים
* **פתרון:** המערכת אינה מאפשרת העלאת קבצים

**שכבת התעבורה**

**פרוטוקול TCP - לחיצת יד משולשת:**

* **איום:** התקפות SYN Flood
* **פתרון:** הגבלת מספר חיבורים פתוחים בו-זמנית ושימוש ב – UDP

**הצפנה:**

* **איום:** פענוח תוכן התעבורה
* **פתרון:** שימוש ב AES-GCM שהוא אלגוריתם הצפנה חזק עם אימות מובנה

**שכבת הרשת**

**פרטיות המשתמש:**

* **איום:** חשיפת כתובת ה – IP האמיתית של המשתמש
* **פתרון:** ניתוב כל התעבורה דרך השרת, הסתרת כתובת המקור, ושימוש בכתובות ווירטואליות

**DNS Leakage**

* **איום:** דליפת בקשות DNS שעוקפות את המנהרה המוצפנת
* **פתרון:** ניתוב גם בקשות DNS דרך המנהרה, ושימוש בשרתי DNS מוגדרים

**שכבת מערכת ההפעלה**

**הרשאות מערכת:**

* **איום:** גישה לא מורשית להגדרות מערכת
* **פתרון:** הרצת השרת עם הרשאות מינימליות הנדרשות, הגבלת הפעולות שניתן לבצע

**חולשות ממשקTUN :**

* **איום:** ניצול חולשות בממשק ה – TUN לגישה למערכת
* **פתרון:** יצירת ממשק TUN עם הרשאות מינימליות, בדיקות קלט קפדניות

# מימוש הפרויקט

## פירוט מימוש המערכת

**סקירת המודולים והמחלקות**

**מודולים/מחלקות מיובאים:**

|  |  |
| --- | --- |
| **שם הספרייה** | **תיאור** |
| socket | מודול המאפשר תקשורת רשת ברמת TCP/UDP, משמש ליצירת החיבורים בין לקוח לשרת |
| threading | מודול לתמיכה בריבוי משימות מקבילות, חיוני להפעלת רכיבי לקוח ושרת במקביל |
| fcntl | מודול לביצוע קריאות מערכת ו - ioctl בלינוקס, משמש ליצירה וניהול של ממשקי TUN |
| queue | מודול המספק מבני תור עם תמיכה בתיאום בין ת'רדים, משמש להעברת הודעות בין תהליכים |
| cryptography | ספרייה מקיפה לפעולות הצפנה ואבטחה, משמשת לאלגוריתמי ECDH, AES-GCM ועוד |
| hashlib | מודול לחישוב פונקציות גיבוב קריפטוגרפיות כמו SHA-256, MD5, לזיהוי לקוחות |
| logging | מודול לניהול יומן אירועים ותיעוד, מאפשר תיעוד מסודר של פעולות המערכת |
| struct | מודול לאריזה ופריסה של נתונים בינאריים, חיוני למימוש מבנה הפרוטוקול ומבנה החבילות |
| PyQt5 | ספרייה לבניית ממשקי משתמש גרפיים מבוססי Qt, משמשת לבניית ממשק הלקוח הגרפי |
| sqlite3 | מודול לעבודה עם מסדי נתונים SQLite, משמש לניהול פרטי משתמשים וסשנים |
| scapy | ספרייה לניתוח, יצירה ומניפולציה של חבילות רשת, משמשת לניתוח חבילות IP/TCP/UDP |
| json | מודול לעיבוד וניתוח קידודי פורמט JSON, משמש להעברת הגדרות תצורה ופרטי חיבור בין לקוח לשרת |
| time | מודול לגישה לפונקציות הקשורות לזמן, משמש למדידת זמני timeout ומנגנוני keepalive |
| os | מודול המספק ממשק למגוון פונקציות של מערכת ההפעלה, משמש לניהול קבצים ותהליכים |
| subprocess | מודול לניהול תהליכי משנה, משמש להרצת פקודות מערכת כמו הגדרות iptables וחוקי ניתוב |
| sys | מודול המספק גישה למשתנים וכלים של המעבד Python, משמש ליציאה ולטיפול בשגיאות |
| platform | מודול לזיהוי פלטפורמת מערכת ההפעלה, משמש להתאמת פעולות ספציפיות ללינוקס |
| ipaddress | מודול לטיפול וניתוח של כתובות IP ורשתות, משמש לניהול מאגר כתובות ה – IP וחישובי רשת |
| signal | מודול לטיפול באותות מערכת הפעלה, משמש לטיפול בסיגנלים כמו Ctrl+C לסגירה מסודרת |
| traceback | מודול לאחזור ותצוגה של מידע מחריגות, משמש לתיעוד מפורט של בעיות בזמן ריצה |
| zlib | מודול לדחיסה ופתיחת דחיסה, משמש בעיקר לחישובי CRC32 checksum לחבילות הנתונים |
| enum | מודול להגדרת ערכים קבועים, משמש להגדרת קבועים כמו סוגי הודעות ומצבי חיבור |
| typing | מודול להגדרת הנחיות סוגי משתנים, משפר את הקריאות ומאפשר בדיקת טיפוסים סטטית |
| random | מודול ליצירת מספרים אקראיים, משמש ליצירת מזהים, סיסמאות, מפתחות ואתגרי אימות |

**מודולים/מחלקות שפותחו בפרויקט**

**protocol.py - הגדרת פרוטוקול VPN**

**שם המחלקה:** MessageHandler

**תפקיד המחלקה:** טיפול ביצירה, פענוח וניהול של הודעות פרוטוקול ה - VPN

**תכונות המחלקה:**

* **Role :** תפקיד (שרת או לקוח)
* **magic\_number :**  מספר קסם לזיהוי הודעות פרוטוקול
* **version :**  גרסת הפרוטוקול
* **client\_id :**  מזהה ייחודי ללקוח
* **Crypto :**  אובייקט להצפנה ופענוח
* **Encrypted :**  דגל האם התקשורת מוצפנת

**פעולות במחלקה:**

* **create\_packet :**  יצירת חבילת נתונים לשליחה
* **parse\_packet :** פענוח חבילת נתונים שהתקבלה
* **process\_packet :**  עיבוד חבילת נתונים והחזרת המידע הרלוונטי
* **create\_\*\_packet :**  מגוון פונקציות ליצירת סוגי הודעות שונים

**reliable\_protocol.py - פרוטוקול אמין**

**שם המחלקה:** ReliableMessageHandler

**תפקיד המחלקה:** הוספת שכבת אמינות מעל MessageHandler הבסיסי

**תכונות המחלקה:**

* **unacked\_packets :**  מילון של חבילות שטרם אושרו
* **pending\_acks :**  רשימת אישורים ממתינים לשליחה
* **delivery\_callbacks :**  פונקציות לקריאה בעת אישור קבלה

**פעולות במחלקה:**

* **create\_packet :**  עטיפה למתודה המקורית עם הוספת מעקב אחר חבילות
* **process\_packet :**  עטיפה למתודה המקורית עם טיפול באישורי קבלה
* **register\_delivery\_callback :**  רישום פונקציית callback לאישור קבלה
* **wait\_for\_delivery :**  המתנה סינכרונית לאישור קבלת חבילה

**crypto.py - אבטחה והצפנה**

**שם המחלקה:**  VPNCrypto

**תפקיד המחלקה:** טיפול בהצפנה, פענוח וחילופי מפתחות

**תכונות המחלקה:**

* **private\_key :**  מפתח פרטי לחילופי ECDH
* **public\_key :**  מפתח ציבורי לחילופי ECDH
* **peer\_public\_key :**  מפתח ציבורי של הצד השני
* **shared\_key :**  מפתח משותף מחילופי ECDH
* **encryption\_key :**  מפתח להצפנה סימטרית

**פעולות במחלקה:**

* **generate\_keypair :**  יצירת זוג מפתחות ECDH
* **set\_peer\_public\_key :**  קביעת המפתח הציבורי של הצד השני
* **generate\_shared\_key :**  יצירת מפתח משותף מהמפתחות
* **encrypt :**  הצפנת נתונים עם AES-GCM
* **Decrypt :**  פענוח נתונים מוצפנים

**client.py - צד לקוח**

**שם המחלקה :**  VPNClient

**תפקיד המחלקה:** ניהול צד הלקוח של ה - VPN

**תכונות המחלקה:**

* **bind\_address :**  כתובת ופורט לקשירת הסוקט
* **server\_address :**  כתובת ופורט השרת
* **virtual\_interface :**  שם ממשק ה - TUN
* **virtual\_ip :**  כתובת IP וירטואלית שהוקצתה ללקוח
* **connection\_state :**  מצב החיבור הנוכחי

**פעולות במחלקה:**

* **start\_client :**  אתחול והפעלת הלקוח
* **stop\_client :**  עצירת הלקוח וניקוי משאבים
* **connect\_to\_server :**  יצירת חיבור לשרת
* **run\_client :**  הפעלת כל התהליכים הנדרשים

**server.py - צד שרת**

**שם המחלקה :**  VPNServer

**תפקיד המחלקה:** ניהול צד השרת של ה - VPN

**תכונות המחלקה:**

* **bind\_address :**  כתובת ופורט לקשירת הסוקט
* **network :**  רשת ה – VPN הווירטואלית
* **active\_clients :**  מילון של לקוחות מחוברים
* **ip\_pool :**  מאגר כתובות IP פנויות

**פעולות במחלקה:**

* **start\_server :**  אתחול והפעלת השרת
* **stop\_server :**  עצירת השרת וניקוי משאבים
* **run\_server :**  הפעלת כל התהליכים הנדרשים
* ***handle\_client*\* :**  מגוון פונקציות לטיפול בפעולות לקוח

**user\_manager.py - ניהול משתמשים**

**שם המחלקה:**  ServerUserManager

**תפקיד המחלקה:** ניהול משתמשים, הרשמה, אימות וניהול סשנים

**תכונות המחלקה:**

* **db\_file :**  קובץ מסד הנתונים
* **\_lock :**  מנעול לגישה בטוחה למסד הנתונים

**פעולות במחלקה:**

* **register\_user :**  רישום משתמש חדש
* **authenticate\_user :**  אימות משתמש קיים
* **validate\_session :**  בדיקת תוקף של סשן
* **invalidate\_session :**  ביטול סשן קיים
* **cleanup\_expired\_sessions :**  ניקוי סשנים שפג תוקפם

**vpn\_gui\_app.py - ממשק משתמש גרפי**

**שם המחלקה:**  MainWindow

**תפקיד המחלקה:** ניהול ממשק המשתמש הגרפי

**תכונות המחלקה:**

* **vpn\_client :**  אובייקט לקוח VPN
* **stacked\_widget :**  אובייקט לניהול מסכים
* **connection\_status :**  תווית מצב חיבור

**פעולות במחלקה:**

* **setup\_ui :**  הגדרת ממשק המשתמש
* **handle\_login :**  טיפול בהתחברות
* **handle\_signup :**  טיפול בהרשמה
* **toggle\_vpn\_connection :**  הפעלה או ניתוק חיבור ה-VPN
* **update\_connection\_status :**  עדכון תצוגת מצב החיבור

## ניתוח קטעי קוד מרכזיים

**הסבר על יכולת חילופי מפתחות**

מנגנון חילופי המפתחות הוא חיוני ליצירת ערוץ תקשורת מאובטח בין הלקוח לשרת.

def create\_key\_exchange\_init\_packet(self) -> bytes:

"""יצירת חבילת אתחול החלפת מפתחות"""

*# יצירת זוג מפתחות חדש*

public\_key = self.crypto.generate\_keypair()

*# יצירת חבילה עם המפתח הציבורי כמטען*

return self.create\_packet(MsgType.KeyExchangeInit, public\_key)

def create\_key\_exchange\_response\_packet(self, client\_public\_key, client\_addr=None) -> bytes:

"""יצירת חבילת תגובה להחלפת מפתחות """

*# הגדרת המפתח הציבורי של הלקוח*

self.crypto.set\_peer\_public\_key(client\_public\_key)

*# יצירת זוג המפתחות שלנו וקבלת המפתח הציבורי*

public\_key = self.crypto.generate\_keypair()

*# יצירת המפתח המשותף*

self.crypto.generate\_shared\_key()

*# הפעלת הצפנה*

self.encrypted = True

*# יצירת והחזרת חבילת התגובה*

return self.create\_packet(MsgType.KeyExchangeResponse, public\_key, client\_addr=client\_addr)

def handle\_key\_exchange\_response(self, peer\_public\_key) -> bool:

"""עיבוד תגובת החלפת מפתחות (בצד הלקוח)"""

*# הגדרת המפתח הציבורי של השרת*

if not self.crypto.set\_peer\_public\_key(peer\_public\_key):

return False

*# יצירת המפתח המשותף*

if not self.crypto.generate\_shared\_key():

return False

*# הפעלת הצפנה להודעות עתידיות*

self.encrypted = True

return True

הקוד הזה מיישם את פרוטוקול ECDH לחילופי מפתחות:

1. הלקוח מייצר זוג מפתחות ושולח את המפתח הציבורי לשרת
2. השרת מקבל את המפתח הציבורי של הלקוח, מייצר זוג מפתחות משלו
3. השרת מחשב את המפתח המשותף ושולח את המפתח הציבורי שלו ללקוח
4. הלקוח מקבל את המפתח הציבורי של השרת ומחשב את המפתח המשותף
5. שני הצדדים משתמשים במפתח המשותף לגזירת מפתח הצפנה

**הסבר על יכולת יצירת ממשק TUN**

יצירת ומניפולציה של ממשק TUN היא אחת היכולות המרכזיות בפרויקט:

def \_set\_up\_tun(self):

""" TUN""" הקמת ממשק

try:

# וודא שמודול TUN זמין

if not os.path.exists('/dev/net/tun'):

logger.info("טוען מודול TUN/TAP...")

os.system("modprobe tun")

except Exception as e:

logger.error(f"נכשל בטעינת מודול TUN: {e}")

logger.error("אנא ודא שמודול TUN זמין")

sys.exit(1)

try:

# פתיחת קובץ התקן TUN

self.tun\_fd = os.open("/dev/net/tun", os.O\_RDWR)

# יצירת ממשק TUN

ifr = struct.pack("16sH", b"tun%d", IFF\_TUN | IFF\_NO\_PI)

ifr = fcntl.ioctl(self.tun\_fd, TUNSETIFF, ifr)

# קבלת שם הממשק

self.virtual\_interface = ifr[:16].strip(b'\x00').decode()

# הגדרת מצב לא-חוסם

flags = fcntl.fcntl(self.tun\_fd, fcntl.F\_GETFL)

fcntl.fcntl(self.tun\_fd, fcntl.F\_SETFL, flags | os.O\_NONBLOCK)

# ביטול תכונות offloading שעלולות להפריע לתעבורה במנהרה

os.system(f"ethtool -K {self.virtual\_interface} rx off tx off sg off tso off ufo off gso off gro off lro off 2>/dev/null || true")

logger.info(f"ממשק TUN '{self.virtual\_interface}' נוצר")

except Exception as e:

logger.error(f"נכשל בהקמת ממשק TUN: {e}")

raise

הקוד הזה:

1. בודק שמודול TUN זמין במערכת ההפעלה
2. פותח את קובץ ההתקן של TUN
3. מבצע ioctl להגדרת ממשק TUN חדש
4. מגדיר את הממשק למצב לא-חוסם לקריאה אסינכרונית
5. מבטל תכונות offloading שונות שעלולות לגרום לבעיות עם חבילות מוצפנות

**הסבר על יכולת ניתוב תעבורת רשת**

הגדרת הניתוב היא קריטית לתפקוד נכון של ה – VPN :

def \_setup\_client\_nat(self):

"""הגדרת NAT וניתוב עבור לקוח ה-VPN"""

try:

# הפעלת העברת IP

os.system("echo 1 > /proc/sys/net/ipv4/ip\_forward")

# קבלת תחום מכתובת IP וירטואלית ומסכה

cidr\_prefix = self.\_subnet\_mask\_to\_cidr(self.subnet\_mask)

vpn\_subnet = f"{self.server\_virtual\_ip}/{cidr\_prefix}"

# ניקוי חוקים קיימים שעלולים להתנגש

os.system("ip rule flush")

os.system("ip route flush table 200")

# הוספת חוקים ברירת מחדל בחזרה

os.system("ip rule add from all lookup main pref 32766")

os.system("ip rule add from all lookup default pref 32767")

# יצירת טבלת ניתוב VPN

# ניתוב ברירת מחדל דרך IP וירטואלי של השרת

os.system(f"ip route add default via {self.server\_virtual\_ip} dev {self.virtual\_interface} table 200")

# הוספת ניתוב לתחום ה-VPN כולו בטבלאות main ו-VPN

os.system(f"ip route add {vpn\_subnet} dev {self.virtual\_interface}")

os.system(f"ip route add {vpn\_subnet} dev {self.virtual\_interface} table 200")

# וידוא שתעבורה לכתובת האמיתית של השרת עוברת דרך ממשק פיזי

os.system(f"ip rule add to {self.server\_address}/32 table main")

# הגדרת MTU Clamping עבור חיבורי TCP

os.system(f"iptables -t mangle -A POSTROUTING -o {self.virtual\_interface} -p tcp --tcp-flags SYN,RST SYN -j TCPMSS --set-mss {VPN\_MSS}")

os.system(f"iptables -t mangle -A OUTPUT -p tcp --tcp-flags SYN,RST SYN -j TCPMSS --set-mss {VPN\_MSS}")

# הוספת חוקי מעקב חיבורים

os.system(f"iptables -A INPUT -i {self.virtual\_interface} -m conntrack --ctstate ESTABLISHED,RELATED -j ACCEPT")

os.system(f"iptables -A OUTPUT -o {self.virtual\_interface} -m conntrack --ctstate ESTABLISHED,RELATED -j ACCEPT")

# קבלת כל תעבורה בממשק TUN

os.system(f"iptables -A INPUT -i {self.virtual\_interface} -j ACCEPT")

os.system(f"iptables -A OUTPUT -o {self.virtual\_interface} -j ACCEPT")

# הפעלת Path MTU Discovery

os.system("sysctl -w net.ipv4.ip\_no\_pmtu\_disc=0")

# נטרול ICMP redirects

os.system("sysctl -w net.ipv4.conf.all.send\_redirects=0")

os.system("sysctl -w net.ipv4.conf.all.accept\_redirects=0")

# נטרול reverse path filtering בממשק ה-VPN

os.system(f"sysctl -w net.ipv4.conf.{self.virtual\_interface}.rp\_filter=0")

os.system("sysctl -w net.ipv4.conf.all.rp\_filter=0")

logger.info("ניתוב ו-NAT של הלקוח הוגדרו בהצלחה")

except Exception as e:

logger.error(f"נכשל בהגדרת NAT של הלקוח: {e}")

raise

הקוד הזה מבצע:

1. הפעלת העברת IP ברמת הקרנל
2. יצירת טבלת ניתוב נפרדת עבור תעבורת VPN
3. הגדרת חוקים לניתוב תעבורה דרך ממשק ה-TUN
4. הגדרת חוקי iptables לטיפול בחבילות TCP ו-MTU
5. הגדרת מעקב חיבורים ומדיניות אבטחה

## מסמך בדיקות

**הבדיקות שתוכננו בשלב האפיון**

**בדיקת התקנה והפעלה בסיסית:**

**מטרת הבדיקה:** לוודא שהמערכת מותקנת ומופעלת כראוי

**מה בוצע בפועל:**

1. התקנת התלויות הנדרשות (Python 3.8+, PyQt5, cryptography, scapy)
2. הרצת סקריפט השרת עם הרשאות root
3. הרצת אפליקציית הלקוח עם הרשאות root
4. בדיקת יכולת לוגין והרשמה במערכת

**תוצאות הבדיקה:**

* השרת עלה בהצלחה והאזין בפורט המוגדר
* אפליקציית הלקוח עלתה בהצלחה והציגה את מסך הכניסה
* נרשם משתמש חדש בהצלחה
* בוצעה התחברות בהצלחה

**כיצד נפתרו הבעיות (אם היו):**

* התגלתה בעיה בהרשאות לממשק TUN - נוספו הנחיות לריצה עם הרשאות מתאימות

**בדיקת חיבור VPN :**

**מטרת הבדיקה:** לוודא שניתן ליצור חיבור VPN פעיל בין הלקוח לשרת

**מה בוצע בפועל:**

1. הרצת השרת והלקוח על מחשבים שונים ברשת
2. התחברות משתמש וניסיון ליצור חיבור VPN
3. בדיקת יצירת ממשק TUN בשני הצדדים
4. בדיקת הקצאת כתובת IP וירטואלית

**תוצאות הבדיקה:**

* החיבור נוצר בהצלחה
* ממשקי TUN נוצרו בשני הצדדים
* הלקוח קיבל כתובת IP מתחום ה - VPN
* סטטוס החיבור הוצג כראוי בממשק המשתמש

**כיצד נפתרו הבעיות (אם היו):**

* בתחילה היו בעיות בהקמת חיבור ה - VPN - התברר שהיו הגדרות firewall שחסמו את התקשורת UDP, הוספו חוקי firewall מתאימים.
* לעתים הוצגה שגיאת "Connection Timeout" - הוגדלו זמני Timeout לחיבור והתקשורת התייצבה.

**בדיקת הצפנה:**

**מטרת הבדיקה:**  לוודא שהתעבורה בין הלקוח לשרת מוצפנת כראוי

**מה בוצע בפועל:**

1. חיבור לשרת ה - VPN
2. ניטור התעבורה באמצעות Wireshark
3. בדיקת תוכן החבילות ב - Wireshark
4. שליחת נתונים דרך החיבור ובדיקת ההצפנה

**תוצאות הבדיקה:**

* חבילות ה – UDP בין הלקוח לשרת הכילו תוכן שנראה אקראי
* לא ניתן היה לזהות תוכן ה - HTTP/HTTPS בתעבורה
* פרוטוקול חילופי המפתחות עבד כמצופה
* הצפנת AES-GCM תפקדה כראוי

**כיצד נפתרו הבעיות (אם היו):**

* בתחילה התגלתה בעיה שמידע הועבר ללא הצפנה לפני השלמת חילופי המפתחות - תוקן כך שהתקשורת האפליקטיבית תתחיל רק לאחר השלמת חילופי המפתחות.

**בדיקת ניתוב:**

**מטרת הבדיקה:** לוודא שכל התעבורה מהלקוח עוברת דרך ה- VPN

**מה בוצע בפועל:**

1. חיבור לשרת ה - VPN
2. בדיקת טבלאות הניתוב בלקוח באמצעות פקודת ip route show
3. פתיחת הכלי tcpdump עם האזנה על הממשק הווירטואלי בעת הרצה

**תוצאות הבדיקה:**

* טבלאות הניתוב הראו שכל התעבורה מנותבת דרך ממשק ה – TUN
* החבילות היוצאות לרשת עברו דרך הממשק הווירטואלי ‘tun0’ והועברו לשרת לניתוב

**כיצד נפתרו הבעיות (אם היו):**

* התגלתה דליפת DNS כאשר בקשות DNS לא עברו דרך המנהרה - נוספה הגדרה ספציפית לניתוב DNS דרך המנהרה.
* מספר אתרים מסוימים לא היו נגישים - תוקנה בעיה בהגדרות MTU .

**בדיקת אמינות:**

**מטרת הבדיקה:** לוודא שהמערכת מתאוששת מנפילות תקשורת זמניות

**מה בוצע בפועל:**

1. יצירת חיבור VPN פעיל
2. סימולציה של בעיות רשת (ניתוק רשת זמני, חסימת פורטים)
3. בדיקת התאוששות החיבור

**תוצאות הבדיקה:**

* המערכת התאוששה בהצלחה מניתוקים קצרים (עד 30 שניות)
* לאחר ניתוקים ארוכים יותר, החיבור הוקם מחדש אוטומטית
* מנגנון ה- keepalive זיהה כראוי ניתוקים ופעל להקמת חיבור מחדש

**כיצד נפתרו הבעיות (אם היו):**

* לעתים המערכת לא זיהתה נכון ניתוק ממושך - נוסף מנגנון לבדיקה אקטיבית של החיבור
* במקרים מסוימים השרת לא שחרר משאבים של לקוחות מנותקים - תוקן באמצעות מנגנון ניקוי משופר

**פירוט בדיקות נוספות שבוצעו**

**בדיקת ממשק משתמש**

**מטרת הבדיקה:** לוודא שממשק המשתמש הגרפי עובד כמצופה

**מה בוצע בפועל:**

1. בדיקת כל הפונקציות דרך הממשק הגרפי
2. בדיקת תגובות המערכת לפעולות משתמש
3. בדיקת תצוגת סטטוס החיבור
4. בדיקת התנהגות הממשק במצבי שגיאה שונים

**תוצאות הבדיקה:**

* כל הפונקציות עבדו כמצופה
* עדכוני סטטוס הוצגו בזמן אמת
* הודעות השגיאה היו ברורות ומועילות
* הממשק היה ידידותי ונוח לשימוש

**כיצד נפתרו הבעיות (אם היו):**

* מספר שגיאות UI זוהו והודעות שגיאה עודכנו להיות ברורות יותר
* תוקנה בעיה בעדכון סטטוס החיבור בזמן אמת

**בדיקת אבטחה**

**מטרת הבדיקה:** לזהות פרצות אבטחה פוטנציאליות במערכת

**מה בוצע בפועל:**

1. ניסיונות חדירה באמצעות מניפולציה של חבילות
2. ניסיונות SQL Injection במערכת הרישום/התחברות
3. ניסיונות לעקוף את מנגנון האימות
4. בדיקת חוזק ההצפנה

**תוצאות הבדיקה:**

* לא זוהו פרצות SQL Injection
* מערכת האימות וההצפנה עמדה בניסיונות חדירה
* פרוטוקול התקשורת היה עמיד למניפולציה של חבילות
* אחסון הסיסמאות היה מאובטח (hashing עם salt)

**כיצד נפתרו הבעיות (אם היו):**

* זוהתה אפשרות לביצוע replay attack - נוסף מנגנון למניעת חזרה על הודעות ישנות
* נמצאה חולשה בתהליך אימות דו-כיווני - תוקן על ידי הוספת אימות הכולל אתגר ללקוח בשלב לחיצת היד בעת הקמת חיבור VPN

# מדריך למשתמש

## עץ קבצים

מערכת VPN כוללת את הקבצים הבאים:

VPN/

├── server/

│ ├── server.py - שרת ה-VPN העיקרי

│ ├── protocol.py - הגדרות פרוטוקול התקשורת

│ ├── reliable\_protocol.py - פרוטוקול אמינות מעל UDP

│ ├── crypto.py - פונקציות הצפנה ואבטחה

│ ├── user\_manager.py - ניהול משתמשים ואימות

│ └── vpn\_users.db - מסד נתוני משתמשים (נוצר אוטומטית)

│

├── client/

│ ├── client.py - לקוח ה-VPN העיקרי

│ ├── protocol.py - הגדרות פרוטוקול התקשורת (זהה לשרת)

│ ├── reliable\_protocol.py - פרוטוקול אמינות (זהה לשרת)

│ ├── crypto.py - פונקציות הצפנה ואבטחה (זהה לשרת)

│ └── vpn\_gui\_app.py - אפליקציית ממשק משתמש גרפי

│

└── README.md - תיעוד כללי ומידע על המערכת

## התקנת המערכת

**פירוט הסביבה הנדרשת**

**דרישות חומרה:**

* CPU: מעבד בעל 2 ליבות לפחות
* RAM: 2GB לפחות
* דיסק: 100MB פנויים לפחות
* כרטיס רשת פעיל עם חיבור לאינטרנט

**מערכת הפעלה:**

* Linux (Ubuntu 20.04 או חדש יותר, Debian 11 או חדש יותר)
* גרסת קרנל 5.0 ומעלה (נדרש עבור תמיכה מלאה בממשקי TUN)

**פירוט הכלים הנדרשים**

**תלויות מערכת:**

bash

sudo apt-get update

sudo apt-get install -y python3 python3-pip python3-dev python3-pyqt5 net-tools tcpdump iptables ethtool

**ספריות Python:**

bash

pip3 install cryptography PyQt5 scapy

**מיקומי קבצים**

* **שרת:** יש להתקין את קבצי השרת בתיקייה של בחירתכם, למשל /opt/vpn\_server
* **לקוח:** יש להתקין את קבצי הלקוח בתיקייה של בחירתכם, למשל /opt/vpn\_client

**נתונים התחלתיים**

**חשבון משתמש ברירת מחדל:**

* משתמש: admin
* סיסמה: admin123

**הגדרות ברירת מחדל:**

* כתובת שרת: 192.168.68.118
* פורט שרת: 1194
* רשת VPN: 10.8.0.0/24

**רשת**

**דרישות רשת:**

* פורט UDP 1194 פתוח בשרת (ניתן לשנות בקובץ ההגדרות)
* אפשרות לניתוב חבילות IP (IP Forwarding) פעילה בשרת
* גישה ברמת root לצורך הגדרת ממשקי רשת וירטואליים וטבלאות ניתוב

**ארכיטקטורה מינימלית נדרשת**

רכיבי ליבה מינימליים:

* שרת VPN הפועל על מכונה עם כתובת IP ציבורית קבועה
* לקוח VPN הפועל על מכונה עם חיבור לאינטרנט כלשהו
* חיבור UDP פתוח בין הלקוח לשרת

# סיכום אישי/רפלקציה

בפרויקט זה פיתחתי מערכת VPN מאובטחת על בסיס פרוטוקול UDP עם מנגנוני אבטחה והצפנה מתקדמים. העבודה על הפרויקט הייתה מלאה באתגרים והייתה חוויה מלמדת ומעשירה. אשתף את התובנות העיקריות שלי מהתהליך:

**תהליך העבודה על הפרויקט**

העבודה על פרויקט ה-VPN הייתה מורכבת ומאתגרת. התחלתי במחקר מעמיק של עקרונות ה-VPN והטכנולוגיות המשמשות למימושם. הבנתי שאני צריך לטפל במספר נושאים מורכבים:

1. יצירה וניהול של ממשקי רשת וירטואליים
2. פיתוח פרוטוקול תקשורת מאובטח
3. מימוש מנגנוני הצפנה חזקים
4. פיתוח מנגנון אמינות מעל UDP
5. הגדרת ניתוב וחוקי Firewall מתאימים

ההצלחות העיקריות היו במימוש מנגנון האמינות מעל UDP, שאפשר לי להשתמש ביתרונות של UDP (מהירות, פשטות) תוך פתרון החסרונות (חוסר אמינות, אובדן חבילות). כמו כן, הצלחתי ליצור מערכת הצפנה חזקה שמבטיחה פרטיות המשתמש.

האתגרים הגדולים היו בתחום הגדרת ממשקי ה-TUN וטבלאות הניתוב. לקח לי זמן להבין את הדרך הנכונה לנתב תעבורה דרך ממשק וירטואלי ולטפל בשינויים בטבלאות הניתוב של Linux. בנוסף, היו אתגרים בפיתוח מנגנון ההתאוששות מנפילות תקשורת, שהיה חיוני לחוויית משתמש טובה.

**תהליך למידה**

במהלך העבודה על הפרויקט, נדרשתי ללמוד נושאים רבים שלא היו מוכרים לי קודם:

1. **עבודה עם ממשקי TUN/TAP** - למדתי איך לתקשר עם רכיבי קרנל לינוקס וליצור ממשקי רשת וירטואליים
2. **פרוטוקולי הצפנה מתקדמים** - העמקתי בהבנת ECDH, AES-GCM והשימוש בהם
3. **ניתוב מתקדם בלינוקס** - למדתי על טבלאות ניתוב מרובות, חוקי iptables, ו-NAT
4. **פיתוח פרוטוקול תקשורת אמין** - חקרתי את העקרונות של TCP והתאמתי אותם לצרכי המערכת
5. **פיתוח ממשק משתמש גרפי עם PyQt5** - שיפרתי את הידע שלי בפיתוח ממשקי משתמש גרפיים

הלמידה העצמאית הייתה חלק משמעותי מהפרויקט. נעזרתי בתיעוד רשמי, מאמרים מקצועיים, פורומים טכניים וקוד פתוח דומה כדי להבין את המושגים והטכניקות הנדרשות.

**אילו כלים נלקחים להמשך**

הפרויקט הזה הקנה לי מספר כלים יקרי ערך להמשך הדרך:

1. הבנה מעמיקה של רשתות ופרוטוקולי תקשורת
2. ידע בתחום אבטחת מידע והצפנה
3. יכולת לעבוד עם ממשקי מערכת הפעלה ברמה נמוכה
4. ניסיון בפיתוח מערכות מבוזרות עמידות לכשלים
5. מיומנויות דיבוג מתקדמות

**תובנות מהתהליך**

אחת התובנות המשמעותיות מהפרויקט היא ההבנה שלעתים הפתרון הפשוט יותר הוא הטוב יותר. בתחילה ניסיתי ליישם פתרונות מורכבים לבעיות שונות, אך לבסוף גיליתי שפתרונות פשוטים יותר היו יעילים ואמינים יותר.

קיבלתי עזרה ממספר מקורות במהלך העבודה. בפורומים טכניים כמו Stack Overflow מצאתי תשובות לשאלות ספציפיות, ופרויקטי קוד פתוח כמו OpenVPN ו-WireGuard שימשו כהשראה ומקור ללמידה.

שיתוף המידע עם עמיתים היה גם הוא משמעותי. דיונים עם חברים לספסל הלימודים עזרו לי להתגבר על מכשולים ולשפר את הפתרונות שלי.

**במבט לאחור**

במבט לאחור, ישנם מספר דברים שהייתי מיישם אחרת:

1. הייתי מתחיל עם תכנון ארכיטקטורי מפורט יותר, במקום לפתח בגישת "נראה לאן זה יוביל"
2. הייתי משקיע יותר זמן בתכנון וכתיבת בדיקות אוטומטיות מקיפות
3. הייתי מפצל את הקוד ליותר מודולים קטנים עם אחריות ממוקדת
4. הייתי משתמש במערכת ניהול קונפיגורציה במקום הגדרות קשיחות בקוד

**שיפורים עתידיים**

אם היו ברשותי משאבים נוספים, הייתי משפר את הפרויקט במספר דרכים:

1. **תמיכה במערכות הפעלה נוספות** - הרחבת התמיכה ל - Windows ו - macOS
2. **אופטימיזציה של ביצועים** - שיפור מהירות העברת הנתונים ומזעור עיכובים
3. **הוספת ממשק ניהול משתמשים למנהל המערכת** - יצירת ממשק אדמיניסטרטיבי מלא
4. **שיפור מנגנוני האבטחה** - הוספת אימות דו-שלבי ותמיכה בתעודות דיגיטליות
5. **דחיסת נתונים** - הוספת דחיסת נתונים להגדלת רוחב הפס האפקטיבי

עבודה על פרויקט זה העניקה לי הבנה מעמיקה של רשתות VPN ואבטחת מידע. למדתי לא רק על הטכנולוגיה, אלא גם על ניהול פרויקט מורכב והתמודדות עם אתגרים. אני גאה בתוצאה הסופית ומאמין שהמערכת מספקת פתרון VPN יעיל, מאובטח ופתוח.

# ביבליוגרפיה

* 1. [**Dzone – OpenVPN**](https://dzone.com/articles/architecture-of-openvpn-for-internal-networks)
  2. [**Digital Whisper – Protocol Tunneling**](https://www.digitalwhisper.co.il/files/Zines/0x14/DW20-3-Tunneling.pdf)
  3. [**Techdocs – understanding TCP MSS Clamping**](https://techdocs.broadcom.com/us/en/vmware-cis/nsx/vmware-nsx/4-1/administration-guide/virtual-private-network-vpn/understanding-tcp-mss-clamping.html)
  4. [**Scapy Documentation – TUN/TAP interfaces**](https://scapy.readthedocs.io/en/latest/layers/tuntap.html)
  5. [**Computerphile – Aes GCM**](https://www.youtube.com/watch?v=-fpVv_T4xwA&t=315s&ab_channel=Computerphile)
  6. [**Computerphile – Elliptic Curve Cryptography**](https://www.youtube.com/watch?v=NF1pwjL9-DE&t=2s&ab_channel=Computerphile)
  7. [**Cryptobook - ECDH Key Exchange**](https://cryptobook.nakov.com/asymmetric-key-ciphers/ecdh-key-exchange)
  8. [**Cloudflare - Virtual networking 101: bridging the gap to understanding TAP**](https://blog.cloudflare.com/virtual-networking-101-understanding-tap/)
  9. [**Wikipedia – TUN/TAP**](https://en.wikipedia.org/wiki/TUN/TAP)
  10. [**Kernel.org - Universal TUN/TAP device driver**](https://docs.kernel.org/networking/tuntap.html)
  11. [**Baeldung - What Is the TUN Interface Used For?**](https://www.baeldung.com/linux/tun-interface-purpose)
  12. [**Wikipedia - CRC**](https://en.wikipedia.org/wiki/Cyclic_redundancy_check)
  13. [**Stack Overflow - When and how to use Python's RLock**](https://stackoverflow.com/questions/16567958/when-and-how-to-use-pythons-rlock)
  14. [**Emqx - Linux Conntrack and MQTT Connections**](https://www.emqx.com/en/blog/emqx-performance-tuning-linux-conntrack-and-mqtt-connections)
  15. [**Arthurchiago - Connection Tracking (conntrack): Design and Implementation Inside Linux Kernel**](Connection%20Tracking%20(conntrack):%20Design%20and%20Implementation%20Inside%20Linux%20Kernel)
  16. [**Linux Community – Forwarding IP packets between interfaces**](https://community.unix.com/t/forwarding-the-ip-packet-from-tun0-to-from-eth0/240429)
  17. [**Stack Exchange - How to relay traffic from tun to internet?**](https://unix.stackexchange.com/questions/588938/how-to-relay-traffic-from-tun-to-internet)
  18. [**Surf shark – VPN ports**](https://surfshark.com/blog/vpn-port)
  19. [**Nord VPN – ip forwarding on linux**](https://nordvpn.com/blog/ip-forwarding-linux/)
  20. [**Geeksforgeeks – ip command on linux**](https://www.geeksforgeeks.org/ip-command-in-linux-with-examples/)
  21. [**Geeksforgeeks – Masquesrading for NAT**](https://www.geeksforgeeks.org/using-masquerading-with-iptables-for-network-address-translation-nat/)

# נספחים – קוד הפרויקט













