

תיכון
אלון

Net Debugger



תיכון: מקיף ע"ש יגאל אלון

שם פרוייקט: NetDebugger

מגיש: יאיר פוזנסקי, 209599802

מנחים: עידן פלדברג, עודי רז, עומר רודנאי, גיל ליברזון, יעקוב שוצקמן

24.5.2024



תוכן עניינים

1.....	תוכן עניינים
4.....	מבוא
4.....	ייזום
4.....	תיאור ראשוני של המערכת
4.....	בחירת הפרויקט
4.....	אתגרים צפויים
4.....	הגדרת הלקוח
5.....	הגדרת יעדים
5.....	בעיות תועלת וחשכונות
6.....	תחומי הפרויקט
7.....	אפיון
7.....	תיאור מפורט של המערכת
7.....	הסנפת הרשת Sniffer
7.....	הצגת הפקטות
8.....	אפשרויות סינון הפקטות:
8.....	אבחון הרשת Diagnostics
9.....	ניהול סיכונים והתמודדות איתם
9.....	תיאור תחום הידע-יכולות
9.....	הסנפת תעבורת רשת
10.....	בדיקת רשת
10.....	תקשורת צד שרת
11.....	ניהול מסד נתונים
12.....	ארכיטקטורת הפרוייקט
12.....	ארכיטקטורת מערכת
12.....	החומרה בשימוש
12.....	הטכנולוגיות בשימוש
12.....	שפות תכנות:
12.....	ספריות ופקטות עיקריות:
13.....	פרוטוקולי רשת:
13.....	ארכיטקטורת מערכת:
13.....	אבטחה:
13.....	SHA256
14.....	איך SHA 256 עובד
15.....	תרשימי זרימה
15.....	תרשים מחלקות בפרוייקט
16.....	תרשים כללי של הפרוייקט



17.....	תרשים זרימה של הסנפה.....
19.....	האלגוריתמים המרכזיים בפרויקט.....
19.....	אלגוריתם הסנפת פקטות:.....
19.....	אלגוריתם קריאת פקטות:.....
19.....	אלגוריתם סינון הפקטות:.....
20.....	אלגוריתם ביצוע בדיקות אבחון רשת:.....
20.....	סביבת הפיתוח.....
20.....	פרוטוקולי התקשורת.....
20.....	HTTPS (Hypertext Transfer Protocol Secure):.....
20.....	ARP (Address Resolution Protocol):.....
21.....	ICMP (Internet Control Message Protocol):.....
21.....	UDP (User Datagram Protocol):.....
21.....	TCP (Transmission Control Protocol):.....
22.....	מסכי המערכת.....
22.....	מסך הסנפה (Sniffer) - תצוגה כהה.....
23.....	מסך אבחון (Diagnostics) - תצוגה כהה.....
24.....	לא נמצאו פקטות:.....
25.....	תרשים זרימה UI.....
26.....	מבני נתונים.....
26.....	קבצי מערכת:.....
27.....	מימוש הפרויקט.....
27.....	מודלים.....
27.....	צד השרת (Backend):.....
27.....	צד הלקוח (Frontend):.....
28.....	#אלגוריתם הסנפה.....
...	app.py: קובץ Python שמגדיר את אפליקציית Flask, מטפל בנתיבים (routes) ויוצר את השרת. ...
30	
	ysniffer.py: קובץ Python שמכיל את הלוגיקה העיקרית של כלי הסניפר תוך שימוש בספריית Scapy.
33.....	from packet_parser import packet_parser.....
	packet_parser.py: קובץ Python שמכיל פונקציות עזר לניתוח סוגי חבילות שונים באמצעות ביטויים רגולריים.....
35.....	sniffer_db.py: קובץ Python שמנהל את האינטראקציה עם מסד הנתונים SQLite עבור אחסון היסטוריית הסניפר.....
39.....	diagnostics.py: קובץ Python שמכיל פונקציות לביצוע פעולות אבחון רשת כמו השגת טבלת ARP ו-traceroute.....
42.....	main.tsx: נקודת הכניסה לאפליקציית ה-React, שם נעשה האתחול של הקומפוננטה הראשית ומעטפת ה-ChakraProvider.....
43.....	App.tsx: קומפוננטת React הראשית של האפליקציה, שמשתמשת בקומפוננטת TabsNavBar לניווט.....
44.....	DarkModeButton.tsx: קומפוננטת React שמספקת כפתור להחלפה בין מצב בהיר וכהה של הממשק.....
45.....	PacketTable.tsx: קומפוננטת React שיוצרת ומציגה טבלה של נתוני חבילות רשת.....
46.....	iframe.tsx: קומפוננטת React גנרית שמשתמשת כמעטפת מעוצבת לתוכן ילדים (Children).....
	ExpandableText.tsx: קומפוננטת React שמאפשרת למשתמש להציג או להסתיר טקסט באמצעות לחיצה על כפתור.....
47.....	



48.....	Sniffer.tsx: קומפוננטת React שמספקת ממשק משתמש עבור כלי סניפר פשוט, עם אפשרויות סינון, התחלה/עצירה של הסניפר והורדת היסטוריה.
56.....	Diagnostics.tsx: קומפוננטת React שמכילה כלים לאבחון רשת כמו טבלת ARP וכלי traceroute.
61.....	מדריך למשתמש
61.....	התקנה
61.....	הרצה
62.....	גישה (התחברות ל-WebApp)
62.....	שימוש
62.....	בדף Sniffer, תוכל להשתמש בכלי ה-Sniffer:
62.....	בדף האבחון Diagnostics, תוכל להשתמש בכלי האבחון של הרשת:
63.....	רפלקציה
64.....	ביבליאוגרפיה



מבוא

ייזום

תיאור ראשוני של המערכת

הפרויקט "NetDebugger" הוא סוויטה המערבת מכשיר פיזי וממשק משתמש מבוסס WebApp, שתפקידם לפקח על רשתות מחשבים. בדומה לתוכנות כמו Wireshark ו-NetTools, NetDebugger מאפשרת לנתח את התעבורה ברשת, אך היא מציעה ממשק נוח וידידותי יותר, וכן תכונות נוספות שמקלות על ניהול ופיקוח על רשתות.

בחירת הפרויקט

בחירתי בפרויקט זה נובעת מצורך אישי. כמשתמש תכופים ברשתות מחשבים, חיפשתי כלי יעיל וקל לשימוש שיאפשר לי לנתח את פעילות הרשת ולזהות בעיות. לא מצאתי פתרון מתאים בין התוכנות הקיימות, וחשתי שפיתוח NetDebugger יענה על הצורך בצורה מיטבית.

אתגרים צפויים

אני צופה מספר אתגרים במהלך פיתוח הפרויקט: מורכבות טכנית: פיתוח סוויטה הכוללת הן חומרה והן תוכנה מחייב ידע נרחב בתחומים שונים, כגון אלקטרוניקה, תכנות רשת, ופיתוח WebApps. עיצוב ממשק משתמש: יצירת ממשק נוח ואינטואיטיבי, המאפשר למשתמשים מכל הרמות להבין ולנתח את נתוני הרשת, מהווה אתגר משמעותי. התמודדות עם תקלות: פיתוח תוכנה ותכנון חומרה תמיד כרוכים בתקלות בלתי צפויות. ניהול יעיל של תהליך פיתוח יאפשר התמודדות נכונה עם אתגרים אלו.

הגדרת הלקוח

NetDebugger מיועד למשתמשים מכל הרמות, החל ממנהלי רשתות מקצועיים ועד למשתמשים ביתיים. המערכת תספק להם כלי יעיל לניתוח פעילות הרשת, זיהוי בעיות, ושיפור ביצועיה.



הגדרת יעדים

היעדים המרכזיים בפרויקט הם:

פיתוח סוויטה הכוללת הן חומרה והן תוכנה: המכשיר הפיזי יתפוס את נתוני הרשת, בעוד ממשק ה-WebApp יציג אותם למשתמש בצורה ברורה ונוחה.
יצירת ממשק משתמש ידידותי: הממשק צריך להיות אינטואיטיבי וקל לשימוש, תוך מתן אפשרות למשתמשים להגדיר את רמת הפרטים המוצגת.
אספקת תכונות ניתוח נתונים מתקדמות: המערכת תאפשר זיהוי בעיות נפוצות ברשת.
אבטחת מידע: נתוני הרשת יהיו מאובטחים הן במכשיר הפיזי והן בממשק ה-WebApp, ההגנה על ה-WebApp מגיע מהגנת RSA על התקשורת בין השרת ללקוח.

בעיות תועלת וחסכוניות

NetDebugger נועדה לפתור את הבעיות הבאות:

קושי בניתוח פעילות הרשת: משתמשים רבים מתקשים להבין את נתוני הרשת, מה שמקשה עליהם לזהות בעיות ולשפר את ביצועיה.
חוסר יעילות בפתרון בעיות: ללא כלי ניתוח יעיל, פתרון בעיות ברשת יכול להיות תהליך ארוך ומייגע.
צורך במומחיות טכנית: ניהול רשתות מחשבים יעיל דורש ידע טכני נרחב, שאינו זמין לכל משתמש.
NetDebugger תספק פתרון יעיל וקל לשימוש לבעיות אלו, ותאפשר למשתמשים: לנתח את פעילות הרשת בקלות: הממשק הידידותי יאפשר לכל משתמש להבין את נתוני הרשת ולזהות בעיות.
לפתור בעיות במהירות: זיהוי בעיות ופתרון יהיו פשוטים ומהירים יותר, תוך חיסכון בזמן ובמאמץ.

השוואה לפתרונות קיימים

NETTOOLS ו-WIRESHARK הן תוכנות ניתוח רשת המספקות למשתמשים מידע מפורט על פעילות הרשת. עם זאת, תוכנות אלו קשות לשימוש עבור משתמשים שאינם מומחים ברשתות מחשבים. NetDebugger, לעומת זאת, מספק ממשק ידידותי יותר המאפשר למשתמשים מכל הרמות לאבחן את הרשת שלהם.



סקירת טכנולוגיית הפרויקט

NetDebugger משלב טכנולוגיות מוכרות בצורה חדשנית לצורך פיקוח על רשתות מחשבים LAN. המערכת אינה מבוססת על טכנולוגיות חדשות לחלוטין, אלא על שילוב יעיל של טכנולוגיות קיימות עם יכולת גמישות והסתגלות. גישה זו מאפשרת ל-NetDebugger להציע פתרון חדשני שלא קיים בשוק בצורה זמינה.

מגבלות טכנולוגיות

למרות יתרונותיה, ישנן גם מגבלות טכנולוגיות מסוימות ב-NetDebugger. מגבלות אלו נובעות בעיקר מהמכשיר הפיזי הקולט מידע מהרשת. מערכת ההפעלה Windows, לעומת מערכות הפעלה אחרות כמו Linux או MacOS, מגבילה את היקף פעולתה של המערכת ללא שימוש בציוד חיצוני מיוחד.

תחומי הפרויקט

NetDebugger עוסקת בתחומים הבאים:

- **רשתות:** המערכת פועלת באופן נרחב עם פרוטוקולים כמו TCP, ARP, UDP ו-ICMP.
- **מערכות הפעלה:** המערכת עובדת הן במערכות הפעלה Windows והן במערכות Linux.
- **אבטחת מידע:** המערכת מבטיחה אבטחת מידע ברמה גבוהה על ידי שימוש בטכנולוגיות הצפנה מתקדמות (RSA).



אפיון

תיאור מפורט של המערכת

הפרוייקט מציג ממשק רשת למשתמש הקצה שנותן לו שתי דפים מרכזיים, עמוד ההסנפות (Network Sniffer) ועמוד האבחון (Diagnostics). עמוד ה-Sniffer אחראי על התחלת הסנפת רשת, הצגת הפקטות בצורה קומפקטית, מסודרת, עניינית וקריאה וסינון הפקטות.

הסנפת הרשת Sniffer

מוצגות מול המשתמש שתי אלמנטים עבור התחלת ההסנפה, בחירת ממשק רשת (כרטיס רשת, ממשק רשת וירטואלי) וכפתור התחלת הסנפה לאחר דוגמה של ממשק רשת, לדוגמה Wi-Fi, בלחיצה על כפתור ה-START התוכנה תתחיל להסניף את הרשת המקושרת לממשק הרשת הנבחר.

הצגת הפקטות

האפליקציה מספקת למשתמש שמירה של היסטורית ההסנפות בתוך מסד נתונים מקומי ומספקת למשתמש אפשרות בחירה על פי תאריך הסנפה, כאשר נבחר קובץ הסנפה ממשק המשתמש יגיש את טבלאת ההסנפות הרלוונטית. כברירת מחדל, הממשק יגיש את הטבלאה העדכנית ביותר.

הפקטות מוצגות בטבלאה בפורמט הבא

- Index - מספר סידורי
- Time - זמן הכנסה למסד הנתונים
- IP Source - כתובת IP מקור
- IP Destination - כתובת IP יעד
- MAC Source - כתובת MAC מקור
- MAC Destination - כתובת MAC יעד
- Protocol - פרוטוקול תקשורת
- Additional - מידע נוסף שנלקח מהפקטה

*במקרה בוא מספר התווים בקטגוריה אחת עולה על 20 הטקסט יוצג בתוך רכיב טקסט קורס קח שהוא מוחבא עד שהמשתמש לוחץ על כפתור ה-SHOW.

*המשתמש יכול הרענן את טבלאת הפקטות בכדי לקבל את המידע העדכני ביותר גם בעת ריצת ההסנפה.

*מוצג בפני המשתמש כפתור DOWNLOAD להורדת מסד הנתונים.



אפשרויות סינון הפקטות:

- בין פרוטוקולי TCP,UDP,ARP,ICMP
- כתובת IP מקור
- כתובת IP יעד
- כתובת MAC מקור
- כתובת MAC יעד

לאחר הגדרת הסינון ולחיצה על כפתור ה FILTER יוצגו רק הפקטות המתאימות להגדרת הסינון.
*הפקטות מסוננות בחזית עבור מהירות.

אבחון הרשת Diagnostics

עמוד האבחון (Diagnostics) מספק כלים חשובים לאבחון ופתרון בעיות ברשת. הוא כולל את המרכיבים הבאים:

• טבלת ARP:

1. מציגה את טבלת ה-ARP של המחשב המקומי והנתב.
2. מכילה את כתובות ה-IP וכתובות ה-MAC המשויות.
3. ניתן לרענן את הטבלה בלחיצה על כפתור ה-Refresh כדי לקבל את הנתונים העדכניים ביותר.
4. בעת ביצוע ה-Refresh, מוצג כפתור עם סמן טעינה המציין שהפעולה מתבצעת.

• Traceroute

1. מאפשר למשתמש להזין כתובת IP ולבצע פקודת traceroute אליה.
2. מציג את רשימת קפיצות ה-hops (traceroute), הכוללת את כתובות ה-IP ואת זמני התגובה בכל קפיצה.
3. בעת ביצוע ה-traceroute, מוצג כפתור עם סמן טעינה המציין שהפעולה מתבצעת.

הרכיבים מסודרים בצורה ברורה ונוחה לשימוש, ומאפשרים ביצוע פעולות אבחון חשובות ברשת ישירות מהממשק. הנתונים מתעדכנים דינמית מהשרת באמצעות בקשות API, והממשק מספק משוב ויזואלי למשתמש במהלך ביצוע הפעולות.



ניהול סיכונים והתמודדות איתם

המערכת נמצאת תחת מספר סיכונים שעליה להתגבר.
בתור התחלה קיים סיכון של MAN IN THE MIDDLE, התוכנה מתמודדת עם סיכון זה בעזרת הצפנת RSA.
מתקפת Man in the Middle היא מתקפה שבה גורם שלישי (האיש באמצע \ התוקף) יכול ליירט ולקרוא תקשורת בין שני צדדים מבלי ידיעתם.
הצפנת RSA מגנה מפני מתקפות כאלה על ידי:

- הסתרת תוכן ההודעות: התוקף לא יכול לקרוא את ההודעות המוצפנות מכיוון שהוא אינו יודע את המפתח הפרטי של המקבל.
- אימות זהות השולח: האיש באמצע לא יכול להתחזות לשולח מכיוון שהוא אינו יודע את המפתח הפרטי של השולח.

תיאור תחום הידע – יכולות

הסנפת תעבורת רשת

מהות – לכידה והצגה של תעבורת רשת בזמן אמת
פעולות:

- ❖ התחלה ועצירה של תהליך הסנפת הרשת
 - ❖ בחירת ממשק הרשת להסנפה
 - ❖ הסנפת פקטות מתעבורת הרשת
 - ❖ ניתוח ועיבוד הפקטות שנתפסו
 - ❖ שמירת נתוני הפקטות במסד נתונים
 - ❖ הצגת הפקטות שנתפסו בממשק משתמש
 - ❖ סינון הפקטות המוצגות לפי פרמטרים שונים
 - ❖ הורדת היסטוריית ההסנפה כקובץ מסד נתונים
- אובייקטים: ממשק רשת, פקטה, מסניף רשת (Sniffer), מנתח פקטות (Packet Parser), מסד נתונים, ממשק משתמש.



בדיקת רשת

מהות - ביצוע בדיקות אבחון לקישוריות ולתשתית הרשת

פעולות:

- ❖ קבלת טבלת ה-ARP של המחשב המקומי והנתב
 - ❖ ביצוע פקודת Traceroute לכתובת IP מסוימת
 - ❖ הצגת תוצאות ה-Traceroute, כולל זמני התגובה בכל קפיצה
 - ❖ רענון טבלת ה-ARP לקבלת הנתונים העדכניים ביותר
 - ❖ מתן משוב חזותי בעת טעינת נתוני האבחון
- אובייקטים: טבלת ARP, כתובת Traceroute, IP, ממשק משתמש.

ממשק משתמש אינטראקטיבי (Interactive User Interface)

מהות - מתן ממשק נוח לשימוש לתפעול הפרויקט והצגת תוצאות

פעולות:

- ❖ ניווט בין עמודים שונים (דף הסנפה, דף אבחון)
 - ❖ הזנת נתונים על-ידי המשתמש (כתובות IP, בחירת ממשק רשת)
 - ❖ הצגת נתונים בפורמטים שונים (טבלאות, רשימות)
 - ❖ עדכון דינאמי של נתונים מהשרת (טבלאות הסנפה, טבלת ARP)
 - ❖ הפעלת פעולות בלחיצת כפתור (סינון, התחלה/עצירה של הסנפה)
 - ❖ הורדת קבצים (קובץ מסד נתונים של היסטוריית ההסנפה)
 - ❖ הצגת הודעות משוב ושגיאה למשתמש
- אובייקטים: דפי אינטרנט, רכיבי ממשק משתמש (כפתורים, טבלאות, שדות קלט), פונקציות TypeScript.

תקשורת צד שרת

מהות - טיפול בבקשות מהצד לקוח ומתן תגובות מתאימות

פעולות:

- ❖ הגדרת נתיבים (routes) לטיפול בבקשות HTTP שונות
 - ❖ אחזור נתונים ממסד הנתונים (טבלאות הסנפה, קובץ מסד נתונים)
 - ❖ הפעלה ועצירה של תהליך הסנפת הרשת
 - ❖ קבלת מידע על ממשקי הרשת הזמינים במערכת
 - ❖ ביצוע בדיקות אבחון רשת (Traceroute, טבלת ARP)
 - ❖ שליחת תגובות HTTP מתאימות, כולל קודי סטטוס ונתונים
- אובייקטים: בקשות HTTP, תגובות HTTP, שרת אינטרנט, מסד נתונים, מערכת הקבצים.



ניהול מסד נתונים

מהות - שמירה ואחזור של נתוני הסנפת הרשת במסד נתונים פעולות:

- ❖ יצירת טבלאות במסד הנתונים לאחסון נתוני הפקטות
 - ❖ הוספת רשומות חדשות לטבלת ההסנפה בזמן אמת
 - ❖ אחזור תוכן הטבלאות לפי דרישה
 - ❖ מתן גישה להורדת קובץ מסד הנתונים השלם
 - ❖ יצירת חיבורים וביצוע שאילתות SQL
- אובייקטים: מסד נתונים SQLite, טבלאות SQL, רשומות, שאילתות SQL, קובץ מסד נתונים.

אבטחה

מהות - הבטחת אבטחת המידע והגנה מפני גישה לא מורשית פעולות:

- ❖ יצירת אישור SSL עצמי-חתום לתקשורת מאובטחת
 - ❖ הצפנת התקשורת בין הצד לקוח לשרת באמצעות HTTPS
 - ❖ הפניה אוטומטית של תעבורת HTTP ל-HTTPS
- אובייקטים: אישור SSL, מפתח פרטי, תעודה דיגיטלית, פרוטוקול HTTPS.



ארכיטקטורת הפרויקט

ארכיטקטורת מערכת

החומרה בשימוש

הפרויקט מיועד לפעול על מחשב אישי או שרת עם מערכת הפעלה Windows. המערכת דורשת חיבור לרשת, כאשר המחשב שעליו פועל הפרויקט מחובר לרשת באמצעות ממשק רשת (כרטיס רשת פיזי או וירטואלי). במצב טיפוסי, המחשב יהיה מחובר לנתב או מתג רשת המאפשר תקשורת עם מכשירים אחרים ברשת.

הטכנולוגיות בשימוש

שפות תכנות:

- ❖ Python: שפת התכנות העיקרית לצד השרת ועיבוד הנתונים.
- ❖ TypeScript (React): שפת התכנות לצד הלקוח ולממשק המשתמש.
- ❖ HTML/CSS: לעיצוב ופריסת ממשק המשתמש.

ספריות ופקטות עיקריות:

- ❖ Scapy: ספרייה בשפת Python לביצוע הסנפת רשת ועיבוד פקטות.
- ❖ SQLite: מסד נתונים קל משקל לאחסון נתוני הפקטות והגדרות.
- ❖ Flask: ספריית Python ליצירת אפליקציות אינטרנט בצד השרת.
- ❖ React: ספרייה בשפת TypeScript לבניית ממשקי משתמש אינטראקטיביים בצד הלקוח.
- ❖ Axios: ספרייה בשפת TypeScript לביצוע בקשות HTTP מצד הלקוח לשרת.
- ❖ Chakra UI: ספריית רכיבי עיצוב מבוססת React ליצירת ממשק משתמש מודרני ונגיש.



פרוטוקולי רשת:

- ❖ **Ethernet**: פרוטוקול השכבה הפיזית והקישור נתונים עבור רשתות תקשורת קווית.
- ❖ **TCP (Transmission Control Protocol)**: פרוטוקול שכבת התעבורה לתקשורת אמינה ומבוססת חיבור מסודר.
- ❖ **UDP (User Datagram Protocol)**: פרוטוקול שכבת התעבורה לתקשורת לא אמינה וללא חיבור מסודר (מהירות).
- ❖ **IP (Internet Protocol)**: פרוטוקול שכבת הרשת לניתוב פקטות נתונים ברשתות.
- ❖ **HTTP/HTTPS**: פרוטוקול שכבת האפליקציה להעברת דפי אינטרנט ונתונים בין לקוח לשרת.
- ❖ **ARP (Address Resolution Protocol)**: פרוטוקול לאיתור כתובת ה-MAC המשויכת לכתובת IP.
- ❖ **ICMP (Internet Control Message Protocol)**: פרוטוקול לשליחת הודעות שגיאה ובקרה בין מכשירי רשת.

ארכיטקטורת מערכת:

- ❖ **ארכיטקטורת לקוח-שרת**: הפרויקט מחולק לשני רכיבים עיקריים - צד הלקוח (ממשק המשתמש) וצד השרת (לוגיקת הבדיקות ועיבוד הנתונים).
- ❖ **צד לקוח**: ממשק המשתמש מבוסס על React.ts ומשתמש בספריות כמו Axios ו-Chakra UI לתקשורת עם השרת ולהצגת הנתונים בצורה ידידותית.
- ❖ **צד שרת**: אפליקציית Python מבוססת Flask המטפלת בבקשות מצד הלקוח, מבצעת את פעולות הסנפת הרשת והאבחון, ומנהלת את מסד הנתונים.
- ❖ **מסד נתונים**: SQLite משמש לאחסון נתוני הפקטות, טבלאות הסנפה והגדרות שונות של המערכת.

אבטחה:

תקשורת מוצפנת: השימוש בפרוטוקול HTTPS עם אישור SSL עצמי-חתום מבטיח תקשורת מוצפנת בין הלקוח לשרת בעזרת SHA256.

SHA256

(Secure Hash Algorithm 256-bit) הוא אחד מהאלגוריתמים במשפחת SHA-2, אשר פותחה על ידי המכון הלאומי לתקנים וטכנולוגיה (NIST) בארצות הברית. אלגוריתם זה משמש ליצירת טביעת אצבע דיגיטלית, או Hash, עבור נתונים מכל סוג וגודל. תכונה מרכזית של SHA-256 היא שהוא מייצר פלט קבוע בגודל 256 סיביות (32 בתים), ללא תלות בגודל הקלט.



SHA-256 הוא אלגוריתם קריפטוגרפי הנחשב לבטוח במיוחד בזכות מספר תכונות. ראשית, הוא עמיד בפני התנגשות (Collision resistance), כלומר קשה מאוד למצוא שני קלטים שונים שמייצרים אותו פלט. שנית, הוא מציג עמידות בפני חיפוש כפול (Pre-image resistance), כך שקשה מאוד לשחזר את הקלט המקורי מתוך הפלט בלבד. שלישית, הוא מבטיח עמידות בפני חיפוש שני-כפול (Second pre-image resistance), כלומר קשה למצוא קלט אחר שמייצר את אותו פלט כמו קלט נתון.

תכונות אלו הופכות את SHA-256 לכלי מרכזי ביישומים רבים, כולל אבטחת מידע, אימות קבצים, והצפנת מידע. למשל, הוא נפוץ במיוחד בשימוש במטבעות דיגיטליים כמו ביטקוין, שבהם הוא משמש כחלק ממנגנון הוכחת העבודה (Proof of Work) לאימות עסקאות ברשת הבלוקצ'יין. SHA-256 גם נמצא בשימוש בפרוטוקולים קריפטוגרפיים שונים, כולל SSL/TLS לאבטחת תעבורת אינטרנט.

אין SHA 256 עובד

SHA-256 עובד בתהליך רב-שלבי, הכולל מספר שלבים עיקריים:

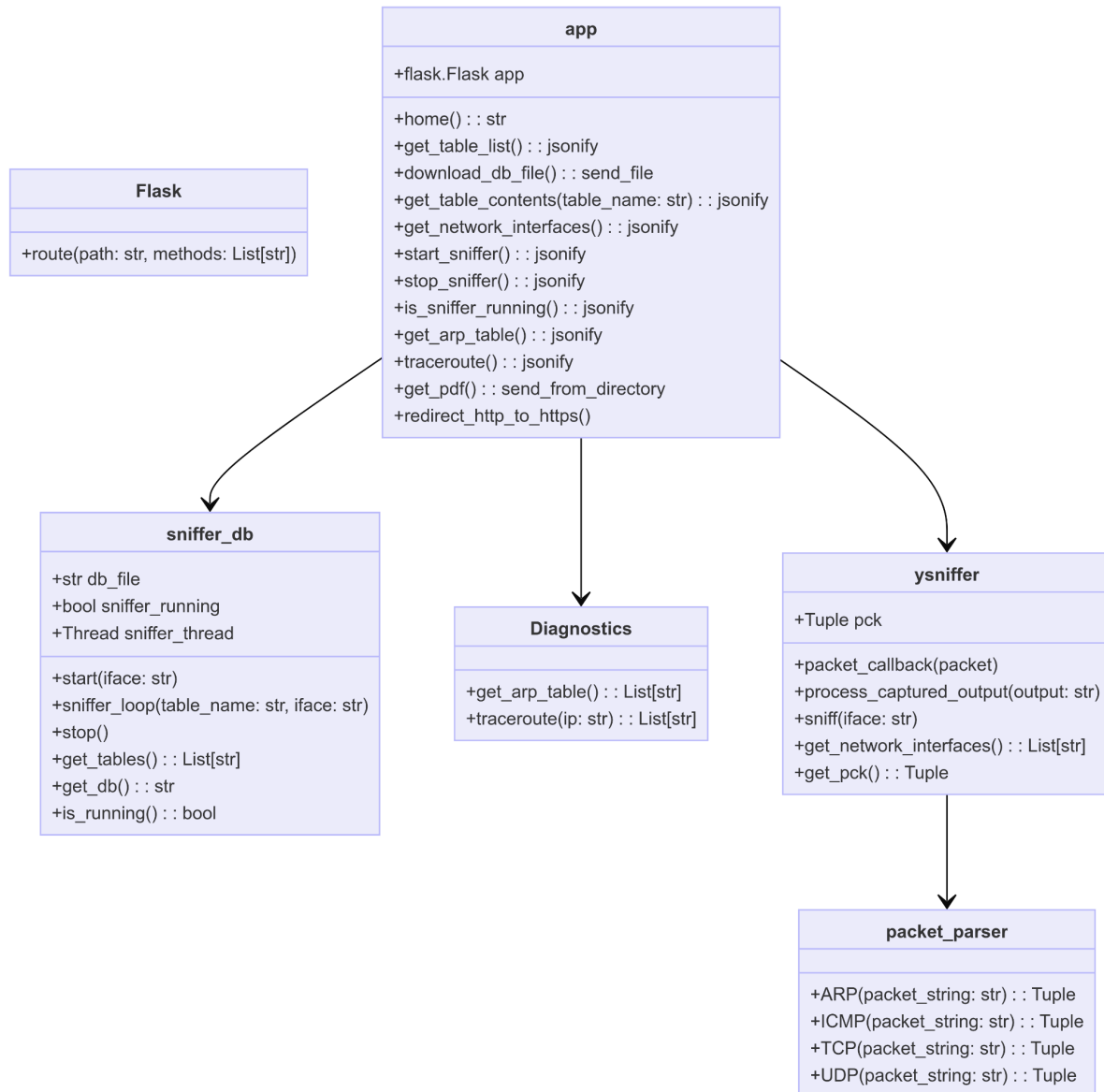
- 1. ריפוד (Padding):** הנתונים המקוריים מרופדים כדי להגיע לגודל שיהיה כפולה של 512 סיביות. תהליך זה כולל הוספת סיבית '1' בסוף הנתונים ולאחריה מספיק סיביות '0' כדי להגיע לגודל הרצוי, ולבסוף הוספת גודל הנתונים המקוריים.
- 2. חילוק לבלוקים (Block Division):** הנתונים המרופדים מחולקים לבלוקים של 512 סיביות כל אחד.
- 3. התחלת ערכים (Initialization):** האלגוריתם מתחיל עם שמונה משתנים (hash values) ראשוניים, כל אחד בגודל 32 סיביות, שנקבעו מראש לפי חישובים מסוימים.
- 4. עיבוד בלוקים (Block Processing):** כל בלוק של 512 סיביות מעובד בלולאה פנימית הכוללת 64 סבבים (rounds). בכל סבב, נעשות פעולות מתמטיות על הקלט ועל המשתנים, כולל חישובי הזזה וסיבוב, ושימוש בקבועים קבועים מראש.
- 5. עדכון משתנים (State Update):** בסיום עיבוד כל בלוק, הערכים של המשתנים מעודכנים בהתאם לתוצאות החישובים.
- 6. חיבור הסופי (Final Hash Value):** לאחר עיבוד כל הבלוקים, הערכים של המשתנים מחוברים יחד ליצירת הפלט הסופי – טביעת האצבע הדיגיטלית בגודל 256 סיביות.

התוצאה הסופית היא hash ייחודי המייצג את הקלט המקורי.

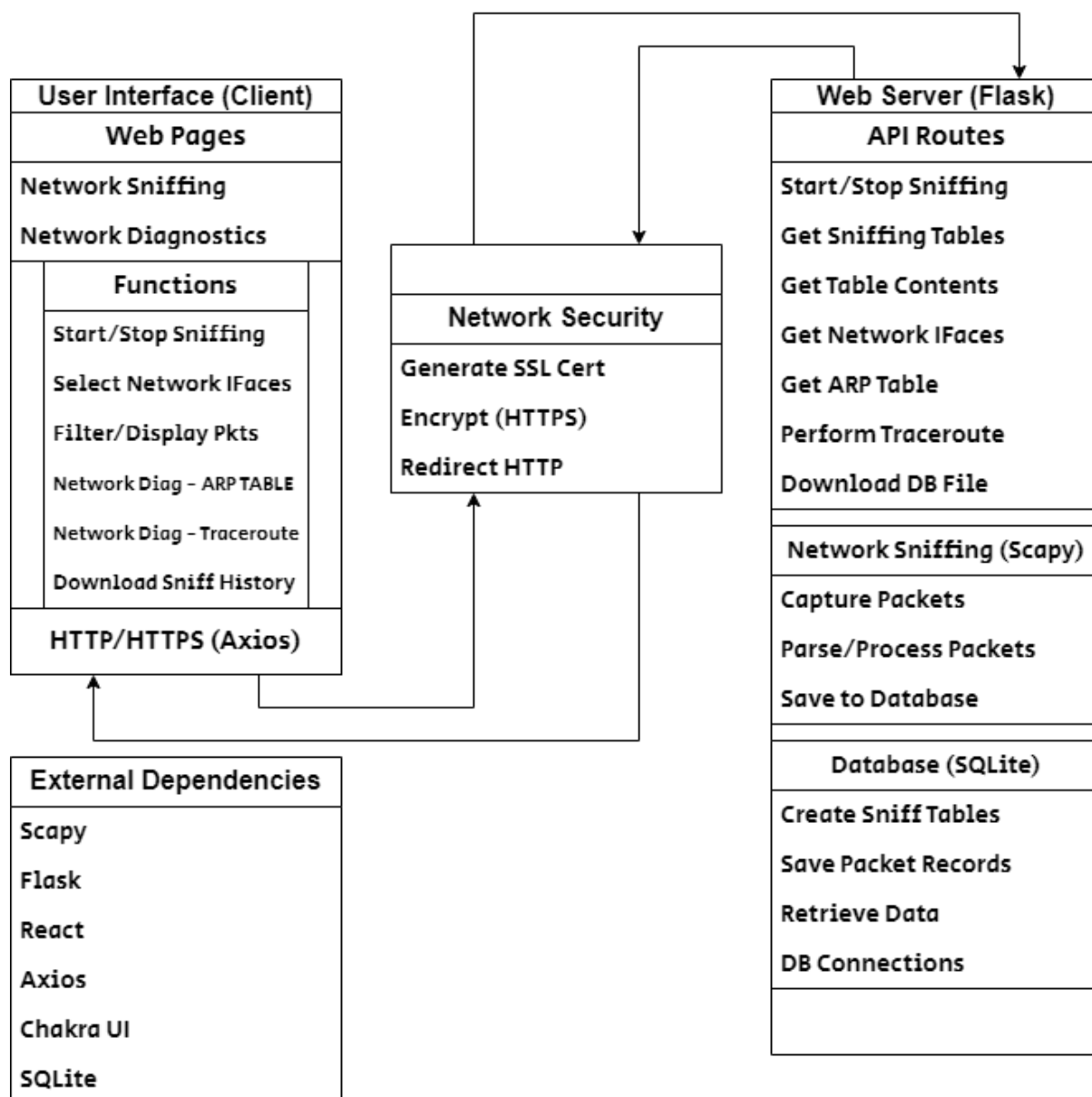


תרשימי זרימה

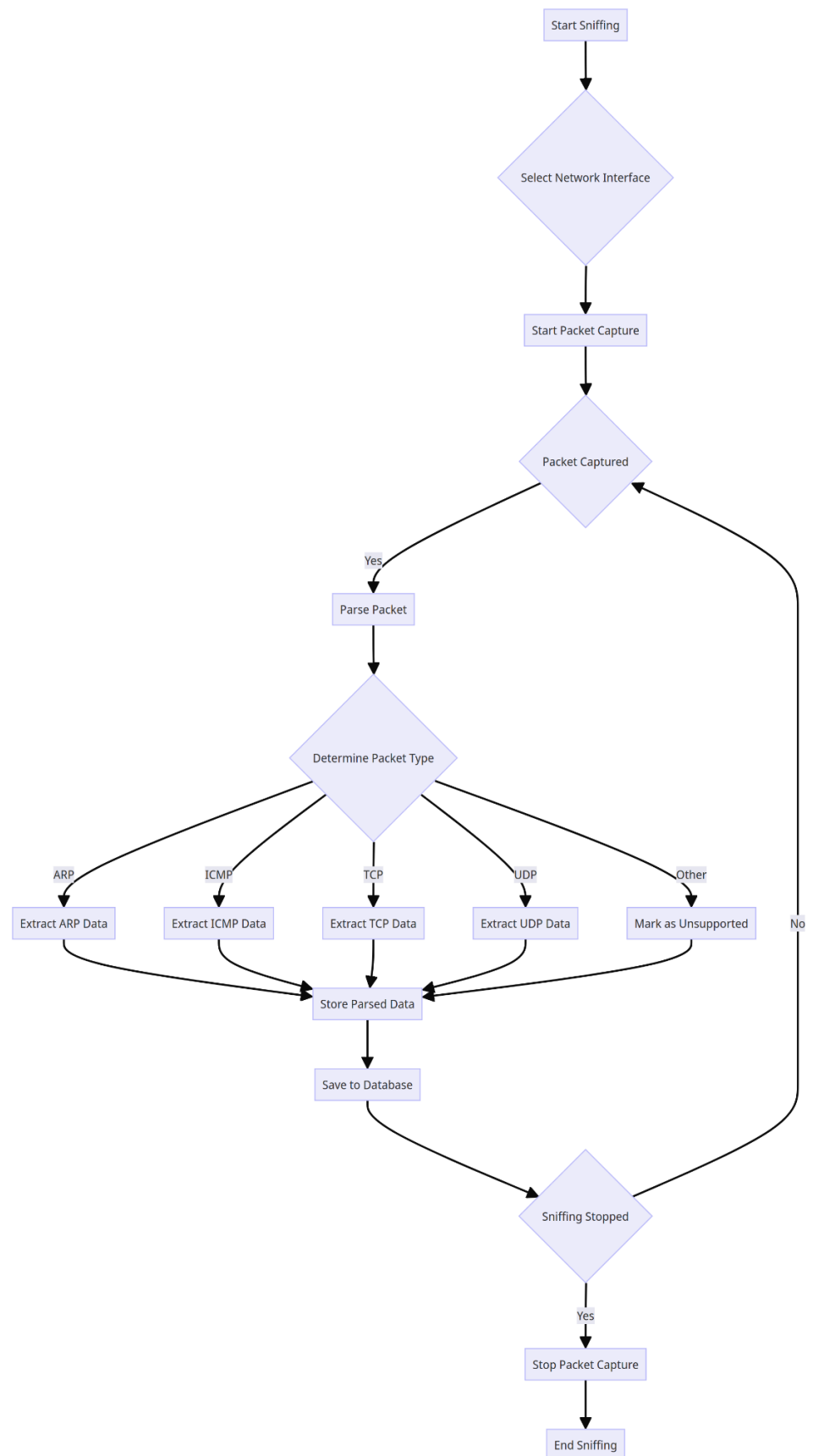
תרשים מחלקות בפרוייקט

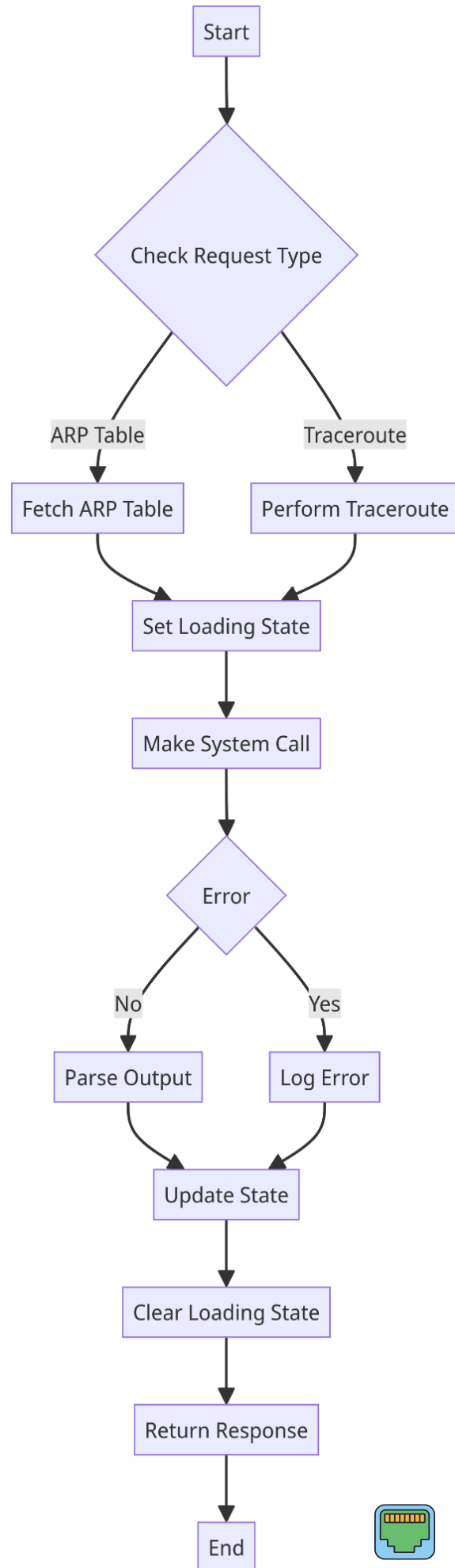


תרשים כללי של הפרויקט



תרשים זרימה של הסנפה





האלגוריתמים המרכזיים בפרויקט

אלגוריתם הסנפת פקטות:

- ❖ האלגוריתם משתמש בספריית Scapy כדי להסניף פקטות רשת בזמן אמת.
- ❖ הוא מאזין לממשק הרשת שנבחר ומחכה לפקטות נכנסות.
- ❖ כאשר מתקבלת פקטה, היא נלכדת ומועברת לפונקציית הקולבק (callback) לעיבוד נוסף.
- ❖ פונקציית הקולבק מנתחת את הפקטה ומחלצת את השדות הרלוונטיים כמו כתובות IP, כתובות MAC, סוג הפרוטוקול ותוכן הפקטה.
- ❖ הנתונים שנאספו נשמרים במסד הנתונים לצורך ניתוח והצגה בהמשך.

אלגוריתם קריאת פקטות:

- ❖ לאחר הסנפת הפקטה, האלגוריתם משתמש בביטויים רגולריים (Regular Expressions) כדי לחלץ את השדות הרלוונטיים מתוך מחרוזת הפקטה.
- ❖ לכל סוג פרוטוקול (כמו UDP, TCP, ICMP, ARP) יש מבנה שונה של מחרוזת הפקטה.
- ❖ האלגוריתם מזהה את סוג הפרוטוקול ומשתמש בביטוי הרגולרי המתאים כדי לחלץ את הנתונים הדרושים.
- ❖ הנתונים שחולצו, כמו כתובות IP וכתובות MAC, מוחזרים כפלט של האלגוריתם.

אלגוריתם סינון הפקטות:

- ❖ בעת הצגת הפקטות שנלכדו למשתמש, מתבצע תהליך סינון כדי להציג רק את הפקטות הרלוונטיות.
- ❖ המשתמש יכול להזין קריטריונים לסינון, כמו כתובת IP מקור, כתובת IP יעד, כתובת MAC מקור, כתובת MAC יעד וסוג פרוטוקול.
- ❖ האלגוריתם עובר על כל הפקטות שנאספו ובודק האם הן תואמות לקריטריוני הסינון שהוגדרו.
- ❖ רק הפקטות שעומדות בקריטריונים מוצגות למשתמש, והשאר מוסתרות.



אלגוריתם ביצוע בדיקות אבחון רשת:

- ❖ האלגוריתם מבצע בדיקות אבחון רשת בסיסיות כמו הצגת טבלת ה-ARP וביצוע Traceroute.
- ❖ להצגת טבלת ה-ARP, האלגוריתם מבצע קריאת מערכת (System Call) כדי לאחזר את הטבלה ממערכת ההפעלה.
- ❖ עבור Traceroute, האלגוריתם מבצע קריאת מערכת עם הפקודה 'traceroute' עבור כתובת ה-IP שסופקה.
- ❖ הפלט של הפקודות מנותח כדי לחלץ את המידע הרלוונטי, כמו כתובות IP וזמני תגובה.
- ❖ המידע שנאסף מועבר בחזרה לממשק המשתמש להצגה.

סביבת הפיתוח

הכלים הדרושים לפיתוח המערכת הם: Node.js עבור צד הלקוח, Python עבור צד השרת (Backend), וסביבה וירטואלית (Virtual Environment) של Python לניהול התלויות והחבילות. סביבת הפיתוח המשולבת (IDE) שנבחרה לפרויקט היא Visual Studio Code, המספקת ממשק משתמש נוח, תמיכה במספר שפות תכנות, ואינטגרציה עם כלי פיתוח שונים. השילוב של Python, Node.js, הסביבה הווירטואלית, ו-VS Code יוצר סביבת פיתוח יעילה ומודולרית.

פרוטוקולי התקשורת

:HTTPS (Hypertext Transfer Protocol Secure)

HTTPS הוא פרוטוקול תקשורת מאובטח המשמש לתקשורת מוצפנת בין דפדפן האינטרנט לשרת האפליקציה. הוא מבוסס על פרוטוקול HTTP הסטנדרטי, אך מוסיף שכבת אבטחה באמצעות הצפנת SSL/TLS. השימוש ב-HTTPS חיוני להגנה על נתונים רגישים, כגון סיסמאות, פרטי כרטיסי אשראי ומידע אישי, מפני יירוט והאזנה בלתי מורשית. בפרויקט, השרת מוגדר להשתמש ב-HTTPS עם אישור SSL בחתימה עצמית. בעת גישה לאפליקציה דרך הדפדפן, מתבצעת תקשורת מוצפנת בין הלקוח לשרת, מה שמבטיח שהנתונים המועברים נשארים חסויים ומוגנים מפני גישה לא מורשית. השימוש ב-HTTPS מספק שכבת אבטחה חשובה לאפליקציה ומגן על פרטיות המשתמשים.

:ARP (Address Resolution Protocol)

ARP הוא פרוטוקול תקשורת המשמש לגילוי כתובות (MAC (Media Access Control של התקנים ברשת על סמך כתובות ה-IP שלהם. כאשר התקן ברשת צריך לשלוח חבילת נתונים להתקן אחר, הוא צריך לדעת את כתובת ה-MAC של יעד החבילה. פרוטוקול ה-ARP פועל על-ידי



שליחת שאילתת ARP ברשת, המכילה את כתובת ה-IP של היעד. ההתקן בעל כתובת ה-IP המתאימה מגיב עם כתובת ה-MAC שלו. המידע הזה נשמר במטמון ARP כדי ליעל תקשורות עתידיות. בפרויקט, הסניפר לוכד חבילות ARP ומחלץ מהן את כתובות ה-IP וה-MAC של ההתקנים המעורבים. מידע זה מוצג בממשק המשתמש כטבלת ARP, המספקת מיפוי בין כתובות IP לכתובות MAC. ניתוח חבילות ARP מאפשר למשתמשים להבין טוב יותר את המבנה והקישוריות של הרשת.

ICMP (Internet Control Message Protocol):

ICMP הוא פרוטוקול תקשורת המשמש לשליחת הודעות שליטה ושגיאה בין התקנים ברשת. הוא מספק מנגנון עבור התקנים כדי לדווח על בעיות, כגון חבילות שאבדו או שגיאות ניתוב, ולבצע פעולות אבחון רשת בסיסיות. שתי הפקודות הנפוצות ביותר המבוססות על ICMP הן Ping ו-Traceroute. Ping משמש לבדיקת הזמינות והמהירות של חיבור רשת על-ידי שליחת חבילות ICMP Echo Request וקבלת תגובות Traceroute. Echo Reply עוקב אחר המסלול שחבילת נתונים עוברת ברשת על-ידי שליחת חבילות ICMP עם ערכי TTL (Time to Live) הולכים וגדלים. בפרויקט, המשתמשים יכולים לבצע פקודות Ping ו-Traceroute על כתובות IP נתונות באמצעות ממשק האבחון. הסניפר מזהה ומנתח חבילות ICMP כדי לספק מידע על זמינות ההתקנים ועל נתיבי התקשורת ברשת.

UDP (User Datagram Protocol):

UDP הוא פרוטוקול תקשורת ברמת התעבורה חסר חיבור וחסר אמינות. הוא מספק שיטה פשוטה ומהירה להעברת מנות נתונים (datagram) בין התקנים ברשת ללא צורך ביצירת חיבור ייעודי. UDP לא מבטיח שהחבילות יגיעו ליעדן בסדר הנכון או בכלל, והוא לא מספק מנגנוני שליטה בזרימה או בקרת שגיאות. עם זאת, העדר התקורה הזו הופך את UDP למתאים לאפליקציות בזמן אמת, כגון הזרמת מדיה, משחקים מקוונים ושליחת נתוני חיישנים. בפרויקט, הסניפר לוכד חבילות UDP ומפרסר אותן כדי לחלץ את כתובות המקור והיעד, מספרי הפורטים והתוכן שלהן. המשתמשים יכולים להציג את נתוני UDP שנלכדו ולסנן אותם לפי פרמטרים שונים. ניתוח חבילות UDP מאפשר תובנות לגבי יישומים המשתמשים בפרוטוקול זה ומסייע בזיהוי דפוסי תעבורה חשודים או חריגים ברשת.

TCP (Transmission Control Protocol):

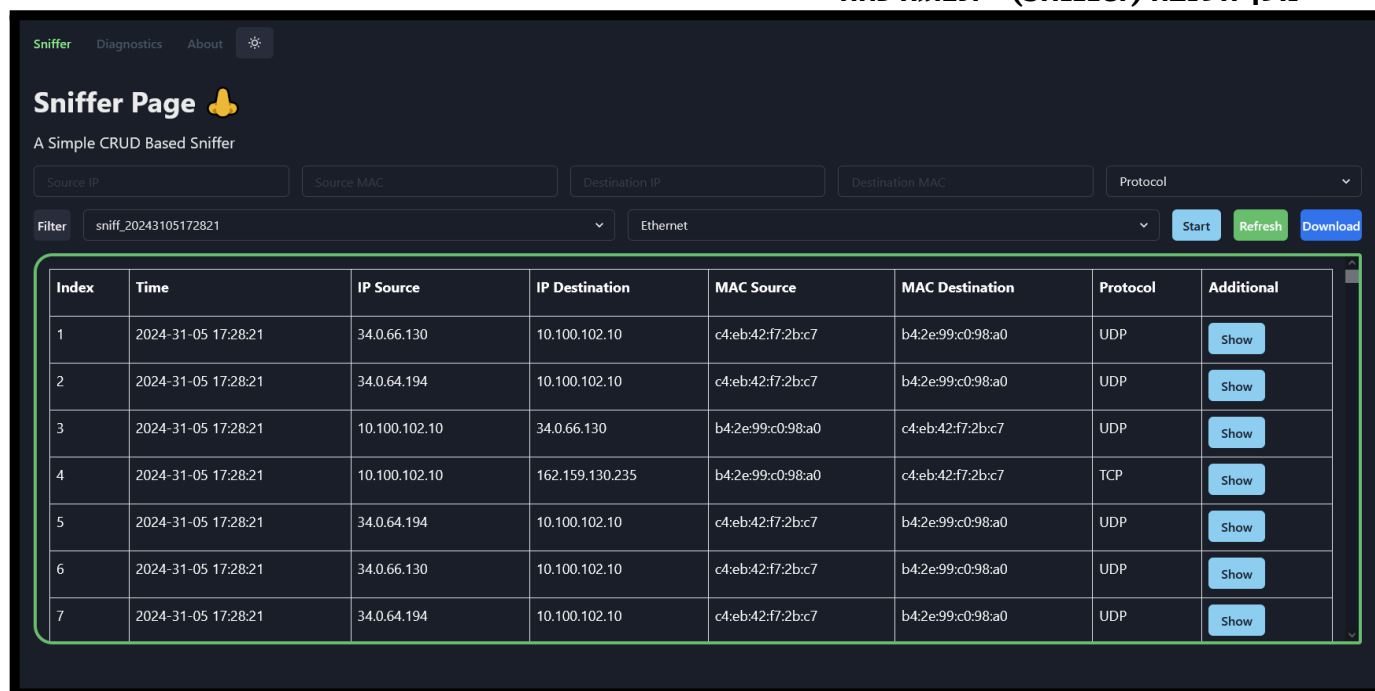
TCP הוא פרוטוקול תקשורת מבוסס חיבור ומהימן ברמת התעבורה. הוא מבטיח שמנות הנתונים יגיעו ליעדן בסדר הנכון וללא שגיאות. לפני שליחת הנתונים, TCP יוצר חיבור ייעודי בין שני ההתקנים באמצעות תהליך "לחיצת יד" בן שלושה שלבים. במהלך ההעברה, TCP מבצע בקרת זרימה ובקרת גודש כדי למנוע הצפת הרשת ולהתאים את קצב השליחה לתנאים המשתנים. אם



חבילות הנתונים אובדות או ניזוקות במהלך השידור, TCP מבקש שליחה מחדש באופן אוטומטי. בשל האמינות שלו, TCP משמש בדרך כלל עבור יישומים הדורשים העברה מדויקת של נתונים, כגון גלישה באינטרנט, דואר אלקטרוני והעברת קבצים. בפרויקט, הסניפר מזהה חבילות TCP ומפרסר אותן לפי השדות שלהן, כולל כתובות המקור והיעד, מספרי הפורטים, מספרי הרצף, מספרי האישור ובקרת הדגלים. ניתוח של נתוני TCP שנלכדו מספק תמונה מקיפה של הקישורים המבוססים על TCP ברשת ומאפשר זיהוי של תקשורות חשודות או בעייתיות.

מסכי המערכת

מסך הסנפה (Sniffer) - תצוגה כהה



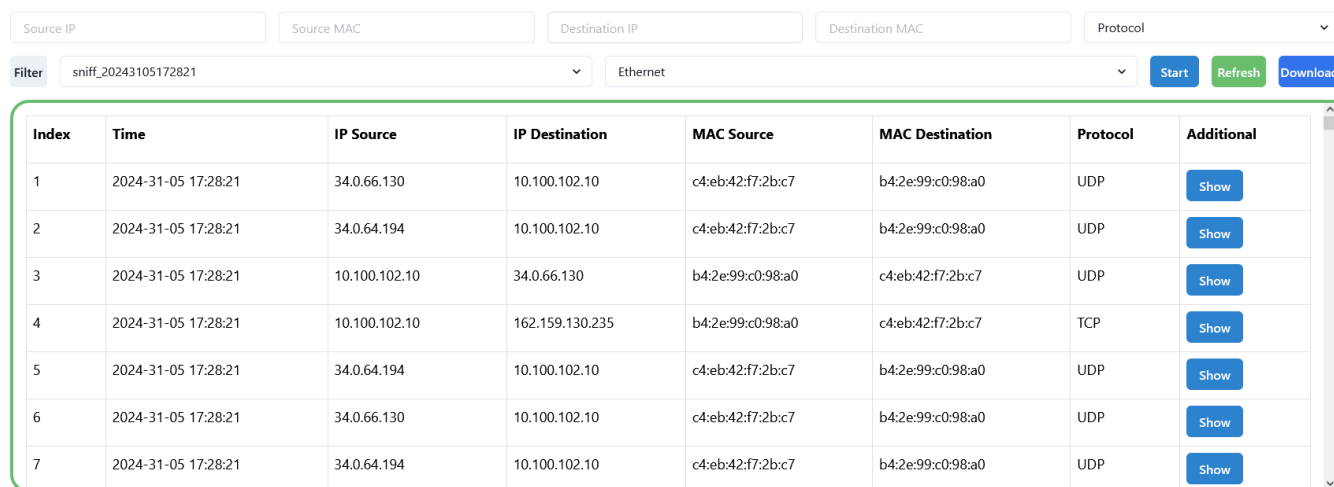
The screenshot shows the 'Sniffer Page' in a dark theme. It features a navigation bar with 'Sniffer', 'Diagnostics', and 'About' links. The main heading is 'Sniffer Page' with a yellow pushpin icon. Below it, the subtitle reads 'A Simple CRUD Based Sniffer'. The interface includes several input fields: 'Source IP', 'Source MAC', 'Destination IP', 'Destination MAC', and a 'Protocol' dropdown menu. A 'Filter' field contains the text 'sniff_20243105172821'. To the right of the filter are three buttons: 'Start' (blue), 'Refresh' (green), and 'Download' (blue). Below these elements is a table with 8 columns: 'Index', 'Time', 'IP Source', 'IP Destination', 'MAC Source', 'MAC Destination', 'Protocol', and 'Additional'. The table contains 7 rows of data. Each row has a 'Show' button in the 'Additional' column.

Index	Time	IP Source	IP Destination	MAC Source	MAC Destination	Protocol	Additional
1	2024-31-05 17:28:21	34.0.66.130	10.100.102.10	c4:eb:42:f7:2b:c7	b4:2e:99:c0:98:a0	UDP	Show
2	2024-31-05 17:28:21	34.0.64.194	10.100.102.10	c4:eb:42:f7:2b:c7	b4:2e:99:c0:98:a0	UDP	Show
3	2024-31-05 17:28:21	10.100.102.10	34.0.66.130	b4:2e:99:c0:98:a0	c4:eb:42:f7:2b:c7	UDP	Show
4	2024-31-05 17:28:21	10.100.102.10	162.159.130.235	b4:2e:99:c0:98:a0	c4:eb:42:f7:2b:c7	TCP	Show
5	2024-31-05 17:28:21	34.0.64.194	10.100.102.10	c4:eb:42:f7:2b:c7	b4:2e:99:c0:98:a0	UDP	Show
6	2024-31-05 17:28:21	34.0.66.130	10.100.102.10	c4:eb:42:f7:2b:c7	b4:2e:99:c0:98:a0	UDP	Show
7	2024-31-05 17:28:21	34.0.64.194	10.100.102.10	c4:eb:42:f7:2b:c7	b4:2e:99:c0:98:a0	UDP	Show

Sniffer Diagnostics About

Sniffer Page

A Simple CRUD Based Sniffer



The screenshot shows the 'Sniffer Page' in a light theme. It features a navigation bar with 'Sniffer', 'Diagnostics', and 'About' links. The main heading is 'Sniffer Page' with a yellow pushpin icon. Below it, the subtitle reads 'A Simple CRUD Based Sniffer'. The interface includes several input fields: 'Source IP', 'Source MAC', 'Destination IP', 'Destination MAC', and a 'Protocol' dropdown menu. A 'Filter' field contains the text 'sniff_20243105172821'. To the right of the filter are three buttons: 'Start' (blue), 'Refresh' (green), and 'Download' (blue). Below these elements is a table with 8 columns: 'Index', 'Time', 'IP Source', 'IP Destination', 'MAC Source', 'MAC Destination', 'Protocol', and 'Additional'. The table contains 7 rows of data. Each row has a 'Show' button in the 'Additional' column.

Index	Time	IP Source	IP Destination	MAC Source	MAC Destination	Protocol	Additional
1	2024-31-05 17:28:21	34.0.66.130	10.100.102.10	c4:eb:42:f7:2b:c7	b4:2e:99:c0:98:a0	UDP	Show
2	2024-31-05 17:28:21	34.0.64.194	10.100.102.10	c4:eb:42:f7:2b:c7	b4:2e:99:c0:98:a0	UDP	Show
3	2024-31-05 17:28:21	10.100.102.10	34.0.66.130	b4:2e:99:c0:98:a0	c4:eb:42:f7:2b:c7	UDP	Show
4	2024-31-05 17:28:21	10.100.102.10	162.159.130.235	b4:2e:99:c0:98:a0	c4:eb:42:f7:2b:c7	TCP	Show
5	2024-31-05 17:28:21	34.0.64.194	10.100.102.10	c4:eb:42:f7:2b:c7	b4:2e:99:c0:98:a0	UDP	Show
6	2024-31-05 17:28:21	34.0.66.130	10.100.102.10	c4:eb:42:f7:2b:c7	b4:2e:99:c0:98:a0	UDP	Show
7	2024-31-05 17:28:21	34.0.64.194	10.100.102.10	c4:eb:42:f7:2b:c7	b4:2e:99:c0:98:a0	UDP	Show



מסך אבחון (Diagnostics) - תצוגה כהה

Sniffer
Diagnostics
About

Diagnostics Page

Network Diagnostics Tool

ARP Table

IP ADDRESS	MAC ADDRESS
Interface:	10.100.102.10
Internet	Address
10.100.102.1	c4-eb-42-f7-2b-c7
10.100.102.5	26-97-d4-6c-08-3b
10.100.102.6	cc-32-e5-14-3d-e5
10.100.102.7	c0-e7-bf-ea-c2-93
10.100.102.255	ff-ff-ff-ff-ff-ff
224.0.0.22	01-00-5e-00-00-16

Refresh

Traceroute

Enter IP Address

Traceroute

Sniffer
Diagnostics
About

Diagnostics Page

Network Diagnostics Tool

ARP Table

IP ADDRESS	MAC ADDRESS
Interface:	10.100.102.10
Internet	Address
10.100.102.1	c4-eb-42-f7-2b-c7
10.100.102.5	26-97-d4-6c-08-3b
10.100.102.6	cc-32-e5-14-3d-e5
10.100.102.7	c0-e7-bf-ea-c2-93
10.100.102.255	ff-ff-ff-ff-ff-ff
224.0.0.22	01-00-5e-00-00-16

Refresh

Traceroute

Enter IP Address

Traceroute



לא נמצאו פקטות:

Sniffer
Diagnostics
About

Sniffer Page 📌

A Simple CRUD Based Sniffer

Filter

Stop
Refresh
Download

NO PACKETS FOUND No Skibidi Rizz :(

Sniffer
Diagnostics
About

Sniffer Page 📌

A Simple CRUD Based Sniffer

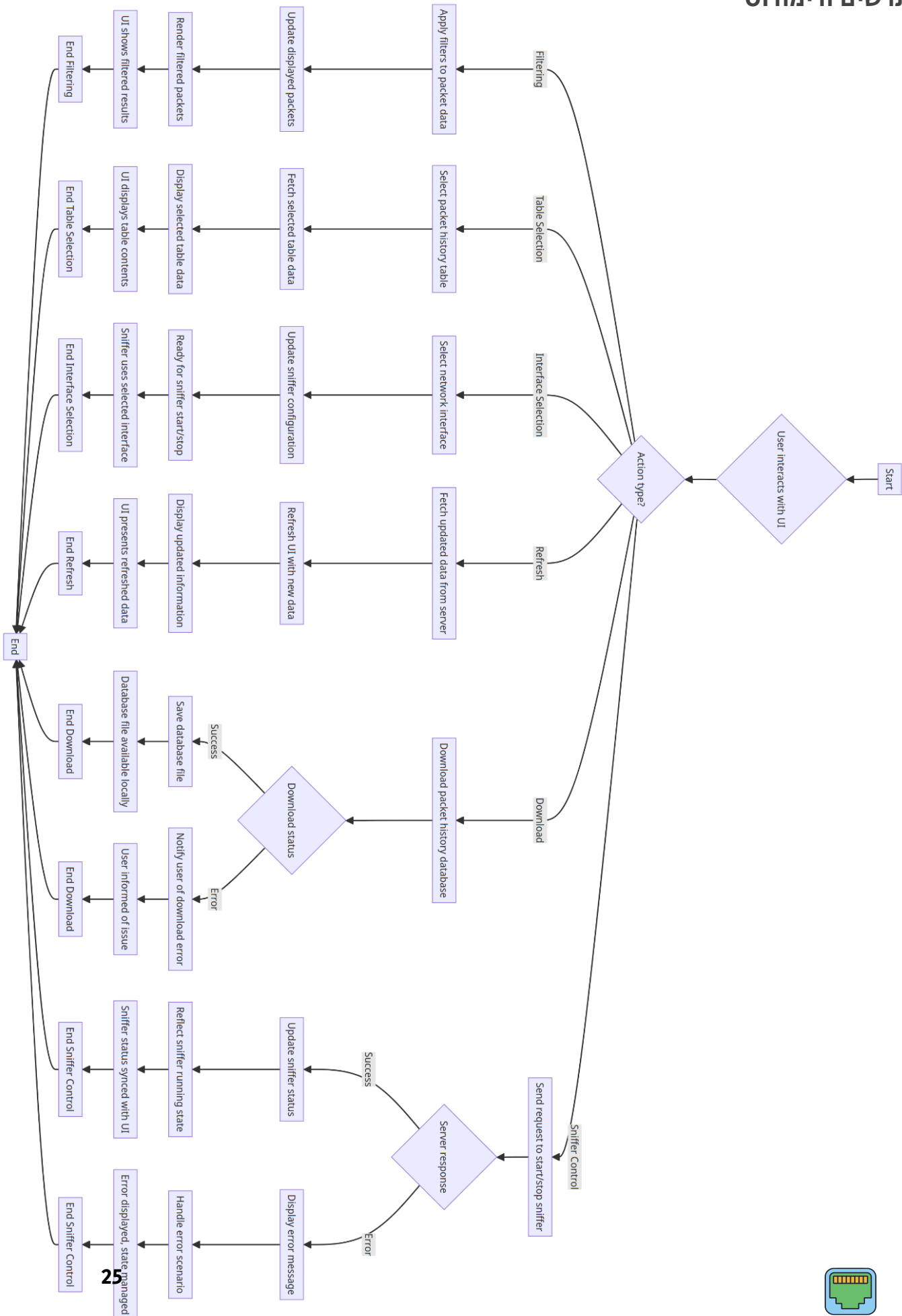
Filter

Stop
Refresh
Download

NO PACKETS FOUND No Skibidi Rizz :(



תרשים זרימה UI



מבני נתונים

מסד הנתונים הראשי (SQLite):

מסד הנתונים SQLite מכיל את כל המידע שנאסף על ידי מערכת ה-Sniffer. המידע מאורגן בטבלאות נפרדות עבור כל הרצה של תהליך הלכידה. בכל טבלה נשמרים הנתונים הבאים:

- ❖ חותמת זמן המציינת את מועד לכידת החבילה
- ❖ כתובות IP ו-MAC של מקור ויעד החבילה
- ❖ סוג הפרוטוקול (TCP, UDP, ARP, ICMP)
- ❖ מידע נוסף כמו תוכן החבילה והדגלים

קבצי מערכת:

האפליקציה בצד הלקוח משתמשת בקבצי מדיה כמו תמונות וסמלילים לצורך ממשק המשתמש. תוכן קבצים אלה הוא סטטי ואינו מושפע מרכיבי הקוד הדינמיים.



מימוש הפרויקט

מודלים

צד השרת (Backend):

1. Flask: מיקרו-פריימוורק ווב בפייתון המשמש לבניית צד השרת של האפליקציה.
2. SQLite3: ספריית מסד נתונים המשמשת לאחסון נתוני ה-sniffing במסד נתונים מקומי.
3. cryptography: ספרייה לביצוע פעולות קריפטוגרפיות כמו יצירת תעודות SSL בחתימה עצמית.
4. scrapy: ספרייה לתפיסה ועיבוד של חבילות רשת.
5. subprocess: מודול המשמש להפעלת תהליכי מערכת כמו פקודות ARP ו-traceroute.
6. datetime: מודול לטיפול בתאריכים וזמנים.
7. os: מודול לאינטראקציה עם מערכת ההפעלה, כמו גישה לקבצים ונתיבים.
8. threading: מודול ליצירת ניהול תהליכים, המשמש להפעלת ה-sniffer ברקע.
9. re: מודול לעיבוד ביטויים רגולריים, המשמש לניתוח פלט החבילות שנתפסו.

מימוש ENDPOINTS

בסדר, הנה השילוב של שני ההסברים:

1. ' / ' (home):
מחזיר את תבנית ה-HTML של דף הבית.
פרוטוקול: HTTP GET
מבנה: ' GET / HTTP/1.1 '
2. 'get_table_list/' (tableList):
מחזיר רשימה של טבלאות מבסיס הנתונים בפורמט JSON.
פרוטוקול: HTTP GET
מבנה: ' GET /tableList HTTP/1.1 '



3. 'download_db_file/(db):

מאפשר הורדה של קובץ מסד הנתונים כקובץ מצורף.

פרוטוקול: HTTP GET

מבנה: 'GET /db HTTP/1.1'

4. 'table/<table_name' (get_table_contents)

מחזיר את התוכן של טבלה ספציפית ממסד הנתונים בפורמט JSON.

פרוטוקול: HTTP GET

מבנה: 'GET /table/<table_name> HTTP/1.1'

5. 'network_interfaces' (get_network_interfaces):

מחזיר רשימה של ממשקי הרשת הזמינים בפורמט JSON.

פרוטוקול: HTTP GET

מבנה: 'GET /network_interfaces HTTP/1.1'

6. 'start_sniffer' (start_sniffer):

מתחיל את פעולת ההסנפה (sniffing) על ממשק הרשת שצוין בבקשה POST.

פרוטוקול: HTTP POST

מבנה: 'POST /start_sniffer HTTP/1.1'

גוף הבקשה: '{<iface>: "<interface_name>"}

7. 'stop_sniffer' (stop_sniffer):

עוצר את פעולת ההסנפה.

פרוטוקול: HTTP POST

מבנה: 'POST /stop_sniffer HTTP/1.1'

8. 'is_sniffer_running' (is_sniffer_running):

מחזיר מידע האם פעולת ההסנפה פועלת כעת או לא בפורמט JSON.

פרוטוקול: HTTP GET

מבנה: 'GET /is_sniffer_running HTTP/1.1'

9. 'arp_table' (get_arp_table):

מחזיר את טבלת ה-ARP של המערכת בפורמט JSON.



פרוטוקול: HTTP GET

מבנה: 'GET /arp_table HTTP/1.1'

10. '/traceroute' (traceroute):

מבצע פקודת traceroute לכתובת IP שצוינה בבקשת POST ומחזיר את התוצאות בפורמט JSON.

פרוטוקול: HTTP POST

מבנה: 'POST /traceroute HTTP/1.1'

גוף הבקשה: '{<ip>: "<ip_address">}'

11. '/pdf' (get_pdf):

מאפשר גישה לקובץ PDF הנמצא בספריית 'static'.

פרוטוקול: HTTP GET

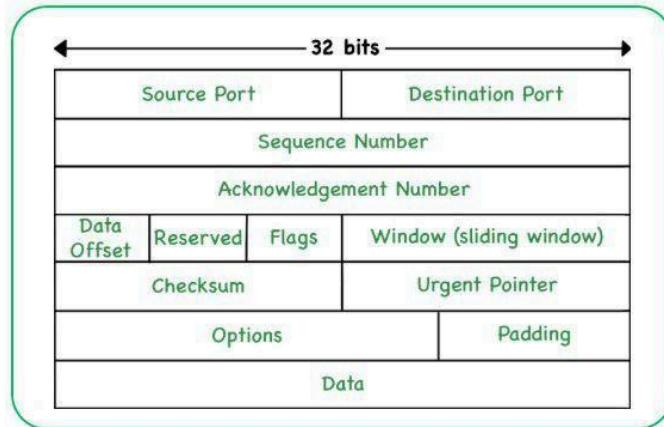
מבנה: 'GET /pdf HTTP/1.1'

כל הבקשות משתמשות בפרוטוקול HTTP, כאשר רוב הבקשות הן מסוג GET, המשמשות לאחזור נתונים מהשרת. הבקשות מסוג POST, כמו 'start_sniffer/' ו-'traceroute/', משמשות לשליחת נתונים לשרת לצורך עיבוד.

המבנה של כל בקשה מורכב מהפרוטוקול (HTTP), סוג הבקשה (GET או POST), הנתוב (URL), וגרסת הפרוטוקול (HTTP/1.1). בקשות POST מכילות גם גוף בקשה בפורמט JSON, המכיל נתונים נוספים הנחוצים לעיבוד הבקשה.

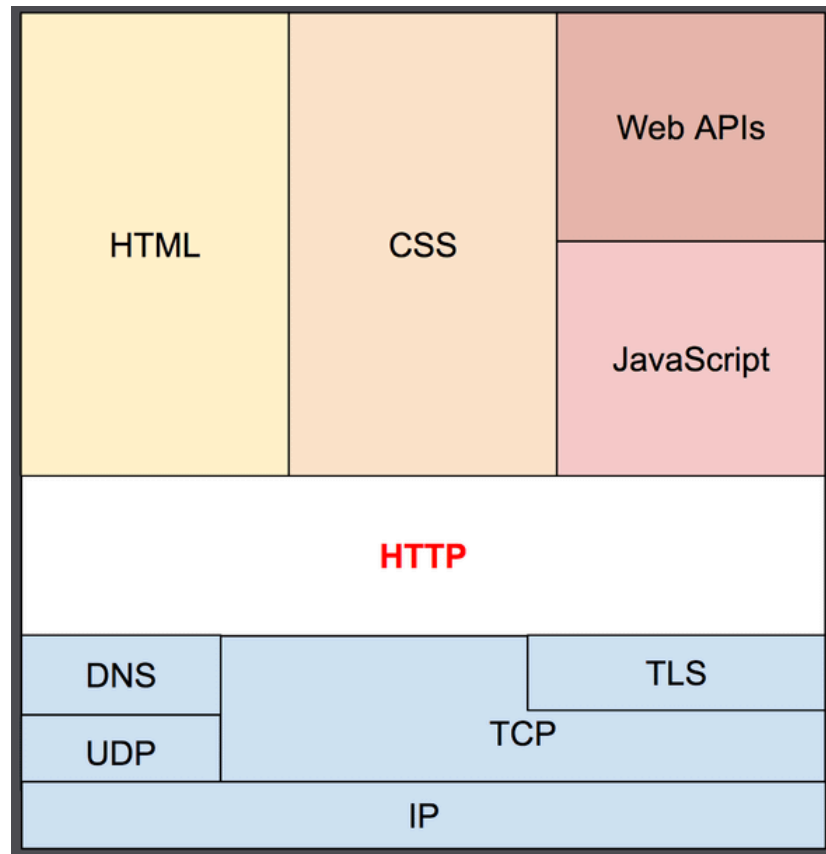
בנוסף, פונקציית 'redirect_http_to_https' פועלת לפני כל בקשה ומבטיחה שכל הבקשות מופנות מ-HTTP ל-HTTPS לצורך תקשורת מאובטחת.





TCP Packet





דוגמה לפרוטוקול האפליקציה HTTP שרוכב על גבי פרוטוקול התעבורה TCP שמבוסס על פרוטוקול הרשת IP.

צד הלקוח (Frontend):

1. **React**: ספריית TypeScript לבניית ממשקי משתמש אינטראקטיביים.
2. **TypeScript**: הרחבה מוקלדת סטטית של TypeScript המוסיפה בדיקת טיפוסים בזמן קומפילציה.
3. **Axios**: ספריית HTTP מבוססת הבטחות לביצוע בקשות API מצד הלקוח.
4. **Chakra UI**: ספריית רכיבי ממשק משתמש ל-React המספקת רכיבים מובנים מראש לבניית ממשקים מודרניים.
5. **React DOM**: חבילה המספקת שיטות ספציפיות ל-DOM לשימוש עם React.
6. **Bootstrap**: ספריית עיצוב פופולרית המספקת מחלקות CSS וקומפוננטות לעיצוב האתר.
7. **styled-components**: ספרייה המאפשרת כתיבת קוד CSS בתוך קבצי הרכיבים של React.



#אלגוריתם הסנפה

```

from packet_parser import packet_parser # יבוא המחלקה
packet_parser מהקובץ packet_parser.py

from collections import Counter # יבוא מחלקת Counter ממודול collections (אינו בשימוש בקוד זה)
from scapy.all import * # יבוא כל המודולים מספריית scapy
import io # יבוא מודול io לטיפול בקלט/פלט

class ysniffer: # הגדרת מחלקת ysniffer
    pck = () # כטאפל ריק pck הגדרת משתנה ברירת המחדל

    def packet_callback(self, packet): # פונקציה לטיפול בחבילות שנתפסו
        captured_output = io.StringIO() # לאחסון StringIO יצירת אובייקט
        פלט שנתפס
        print(packet.show(dump=True), file=captured_output) # הדפסת
        פרטי החבילה לתוך אובייקט הפלט
        output = captured_output.getvalue() # קבלת הערך של הפלט שנתפס
        כמחרוזת
        self.process_captured_output(output=output) # קריאה לפונקציה
        עם הפלט שנתפס process_captured_output

    # Create a function to handle the captured output
    def process_captured_output(self, output:str): # פונקציה לעיבוד
        הפלט שנתפס
        if output.find('TCP') != -1: # בדיקה אם הפלט מכיל 'TCP'
            self.pck = packet_parser.TCP(output.replace("\n", "")) #
            packet_parser באמצעות מחלקת TCP פענוח חבילת
        elif output.find('UDP') != -1: # בדיקה אם הפלט מכיל 'UDP'
            self.pck = packet_parser.UDP(output.replace("\n", "")) #
            packet_parser באמצעות מחלקת UDP פענוח חבילת
        elif output.find('ARP') != -1: # בדיקה אם הפלט מכיל 'ARP'
            self.pck = packet_parser.ARP(output.replace("\n", "")) #
            packet_parser באמצעות מחלקת ARP פענוח חבילת
        elif output.find('ICMP') != -1: # בדיקה אם הפלט מכיל 'ICMP'
            self.pck = packet_parser.ICMP(output.replace("\n", "")) #
            packet_parser באמצעות מחלקת ICMP פענוח חבילת
        else: # מקרה אחר, אם הפרוטוקול אינו נתמך
            self.pck =
            ('Unsupported', 'Unsupported', 'Unsupported', 'Unsupported', 'Unsuppor
    
```



```

ted',output) # השמת טאפל של ערכים לא נתמכים
    pass

def sniff(self, iface): # פונקציה להתחלת sniffing
    sniff(count=1, filter=None, iface=iface,
    prn=self.packet_callback) # עם הגדרות מתאימות sniffing התחלת

    @staticmethod # דקורטור המציין שזו פונקציה סטטית
    def get_network_interfaces(): # פונקציה לקבלת ממשקי הרשת הזמינים
        return [iface.name for iface in get_working_ifaces()] # החזרת
        רשימה של שמות ממשקי הרשת הפעילים

    #GETPACKET#
    def get_pck(self): # פונקציה לקבלת החבילה האחרונה שנתפסה
        tmp_pck = self.pck # השמת החבילה הנוכחית למשתנה זמני
        self.pck = None # איפוס משתנה ה
        return tmp_pck # החזרת החבילה הזמנית

if __name__ == '__main__': # בדיקה אם הקובץ הנוכחי מורץ כתוכנית ראשית
    # "Unit Testing" ;)
    for i in range(80): # (לולאה שרצה 80 פעמים (למטרות בדיקה)
        sniffer = ysniffer() # ysniffer יצירת מופע של המחלקה
        sniffer.sniff('Ethernet') # עם הממשק sniff קריאה לפונקציה
        'Ethernet'
        pck = sniffer.get_pck() # קבלת החבילה האחרונה שנתפסה
        if pck != None: # בדיקה אם החבילה אינה None
            print(pck) # הדפסת החבילה
    
```



app.py: קובץ Python שמגדיר את אפליקציית Flask, מטפל בנתיבים (routes) ויוצר את השרת.

```
import datetime
from flask import Flask, jsonify, redirect, request, render_template,
send_file
from sniffer_db import sniffer_db
import sqlite3
from cryptography.hazmat.primitives import serialization, hashes
from cryptography.hazmat.primitives.asymmetric import rsa
from cryptography import x509
from cryptography.x509.oid import NameOID
from cryptography.hazmat.backends import default_backend
from diagnostics import Diagnostics
from ysniffer import ysniffer
import os

current_dir = os.path.dirname(os.path.abspath(__file__))
app = Flask(__name__)

# Flask routes

@app.route('/')
def home():
    return render_template('index.html')

@app.route('/tableList', methods=['GET'])
def get_table_list():
    tables = sniffer_db.get_tables()
    return jsonify(tables)

@app.route('/db', methods=['GET'])
def download_db_file():
    db_file = os.path.abspath(r".venv\sniff_history.db")
    print(db_file)
    return send_file(db_file, as_attachment=True)

@app.route('/table/<table_name>', methods=['GET'])
def get_table_contents(table_name):
    # Connect to the database
    conn = sqlite3.connect(sniffer_db.db_file)
```



```

        cursor = conn.cursor()

        # Query the table
        cursor.execute(f"SELECT * FROM {table_name}")
        table_data = cursor.fetchall()

        # Close the connection
        conn.close()

        return jsonify(table_data)

@app.route('/network_interfaces', methods=['GET'])
def get_network_interfaces():
    interfaces = ysniffer.get_network_interfaces()
    return jsonify(interfaces)

@app.route('/start_sniffer', methods=['POST'])
def start_sniffer():
    iface = request.json['iface']
    sniffer_db.start(iface)
    return jsonify({'status': 'success', 'message': 'Sniffer started'})

@app.route('/stop_sniffer', methods=['POST'])
def stop_sniffer():
    sniffer_db.stop()
    return jsonify({'status': 'success', 'message': 'Sniffer stopped'})

@app.route('/is_sniffer_running', methods=['GET'])
def is_sniffer_running():
    is_running = sniffer_db.is_running()
    return jsonify({'is_running': is_running})

@app.route('/arp_table', methods=['GET'])
def get_arp_table():
    arp_table = Diagnostics.get_arp_table()
    return jsonify(arp_table)

@app.route('/traceroute', methods=['POST'])
def traceroute():
    ip = request.json['ip']
    hops = Diagnostics.traceroute(ip)
    return jsonify(hops)

```



```
# Self-signed certificate generation

def generate_self_signed_cert():
    # Create a key pair
    key = rsa.generate_private_key(
        public_exponent=65537,
        key_size=2048,
        backend=default_backend()
    )

    # Generate a self-signed certificate
    subject = issuer = x509.Name([
        x509.NameAttribute(NameOID.COMMON_NAME, "My Self-Signed Server"),
        x509.NameAttribute(NameOID.ORGANIZATION_NAME, "My Organization"),
        x509.NameAttribute(NameOID.ORGANIZATIONAL_UNIT_NAME, "My Unit"),
        x509.NameAttribute(NameOID.COUNTRY_NAME, "US"),
        x509.NameAttribute(NameOID.STATE_OR_PROVINCE_NAME, "California"),
        x509.NameAttribute(NameOID.LOCALITY_NAME, "San Francisco"),
    ])

    cert = x509.CertificateBuilder().subject_name(
        subject
    ).issuer_name(
        issuer
    ).public_key(
        key.public_key()
    ).serial_number(
        x509.random_serial_number()
    ).not_valid_before(
        datetime.datetime.utcnow()
    ).not_valid_after(
        datetime.datetime.utcnow() + datetime.timedelta(days=365)
    ).sign(key, hashes.SHA256(), default_backend())

    # Save the certificate and private key
    with open("server.crt", "wb") as f:
        f.write(cert.public_bytes(serialization.Encoding.PEM))

    with open("server.key", "wb") as f:
        f.write(key.private_bytes(
            encoding=serialization.Encoding.PEM,
```



```

        format=serialization.PrivateFormat.TraditionalOpenSSL,
        encryption_algorithm=serialization.NoEncryption(),
    ))

@app.before_request
def redirect_http_to_https():
    if request.url.startswith('http://'):
        url = request.url.replace('http://', 'https://', 1)
        code = 301
        return redirect(url, code=code)

if __name__ == '__main__':
    # Generate a self-signed certificate if it doesn't exist
    try:
        open("server.crt", "rb").close()
        open("server.key", "rb").close()
    except FileNotFoundError:
        generate_self_signed_cert()

    # Start the Flask server with SSL
    app.run(host='localhost', port=8000, ssl_context=('server.crt',
'server.key'))

```

שמכיל את הלוגיקה העיקרית של כלי הסניפר תוך Python קובץ **ysniffer.py**:
Scapy שימוש בספריית.

```
from packet_parser import packet_parser
```

```

from collections import Counter
from scapy.all import *
import io

class ysniffer:

    pck = ()

    def packet_callback(self, packet):
        captured_output = io.StringIO()
        print(packet.show(dump=True), file=captured_output)
        output = captured_output.getvalue()
        self.process_captured_output(output=output)

```



```
# Create a function to handle the captured output
def process_captured_output(self,output:str):
    if output.find('TCP') != -1:
        self.pck = packet_parser.TCP(output.replace("\n", ""))
    elif output.find('UDP') != -1:
        self.pck = packet_parser.UDP(output.replace("\n", ""))
    elif output.find('ARP') != -1:
        self.pck = packet_parser.ARP(output.replace("\n", ""))
    elif output.find('ICMP') != -1:
        self.pck = packet_parser.ICMP(output.replace("\n", ""))
    else:
        self.pck =
('Unsupported','Unsupported','Unsupported','Unsupported','Unsupported',output)
        pass

def sniff(self, iface):
    sniff(count=1, filter=None, iface=iface, prn=self.packet_callback)

@staticmethod
def get_network_interfaces():
    return [iface.name for iface in get_working_ifaces()]

#GETPACKET#
def get_pck(self):
    tmp_pck = self.pck
    self.pck = None
    return tmp_pck

if __name__ == '__main__':
    # "Unit Testing" ;)
    for i in range(80):
        sniffer = ysniffer()
        sniffer.sniff('Ethernet')
        pck = sniffer.get_pck()
        if pck != None:
            print(pck)
```



packet_parser.py: קובץ Python שמכיל פונקציות עזר לניתוח סוגי חבילות שונים באמצעות ביטויים רגולריים.

```

import re

class packet_parser:
    @staticmethod
    def ARP(packet_string):
        # Define the regular expressions
        src_regex = r"src\s*=\s*(\w+:\w+:\w+:\w+:\w+:\w+)"
        dst_regex = r"dst\s*=\s*(\w+:\w+:\w+:\w+:\w+:\w+)"
        dst_ipv4_regex = r"pdst\s*=\s*(\d{1,3}\.\d{1,3}\.\d{1,3}\.\d{1,3})"
        src_ipv4_regex = r"psrc\s*=\s*(\d{1,3}\.\d{1,3}\.\d{1,3}\.\d{1,3})"
        load_regex = r"load\s*=\s*'([\w\\+]+'"

        # Extract the values using regex
        try:
            src = re.search(src_regex, packet_string).group(1)
        except AttributeError:
            src = "Unknown"

        try:
            dst = re.search(dst_regex, packet_string).group(1)
        except AttributeError:
            dst = "Unknown"

        try:
            psrc = re.search(src_ipv4_regex, packet_string).group(1)
        except AttributeError:
            psrc = "Unknown"

        try:
            pdst = re.search(dst_ipv4_regex, packet_string).group(1)
        except AttributeError:
            pdst = "Unknown"

        try:
            load = packet_string
        except AttributeError:
            load = "Unknown"

        return (psrc, pdst, src, dst, 'ARP', load)
    
```




```
@staticmethod
def ICMP(packet_string):
    # Define the regular expressions
    src_regex = r"src\s*=\s*(\w+:\w+:\w+:\w+:\w+:\w+)"
    dst_regex = r"dst\s*=\s*(\w+:\w+:\w+:\w+:\w+:\w+)"
    dst_ipv4_regex = r"dst\s*=\s*(\d{1,3}\.\d{1,3}\.\d{1,3}\.\d{1,3})"
    src_ipv4_regex = r"src\s*=\s*(\d{1,3}\.\d{1,3}\.\d{1,3}\.\d{1,3})"
    load_regex = r"load\s*=\s*'([\w\\]+)'"

    # Extract the values using regex
    try:
        src = re.search(src_regex, packet_string).group(1)
    except AttributeError:
        src = "Unknown"

    try:
        dst = re.search(dst_regex, packet_string).group(1)
    except AttributeError:
        dst = "Unknown"

    try:
        psrc = re.search(src_ipv4_regex, packet_string).group(1)
    except AttributeError:
        psrc = "Unknown"

    try:
        pdst = re.search(dst_ipv4_regex, packet_string).group(1)
    except AttributeError:
        pdst = "Unknown"

    try:
        load = packet_string
    except AttributeError:
        load = "Unknown"

    return (psrc, pdst, src, dst, 'ICMP', load)

@staticmethod
def TCP(packet_string):
    # Define the regular expressions
    src_regex = r"src\s*=\s*(\w+:\w+:\w+:\w+:\w+:\w+)"
```



```

dst_regex = r"dst\s*=\s*(\w+:\w+:\w+:\w+:\w+:\w+)"
dst_ipv4_regex = r"dst\s*=\s*(\d{1,3}\.\d{1,3}\.\d{1,3}\.\d{1,3})"
src_ipv4_regex = r"src\s*=\s*(\d{1,3}\.\d{1,3}\.\d{1,3}\.\d{1,3})"
load_regex = r"###\[ IP \]###(.*)"

# Extract the values using regex
try:
    src = re.search(src_regex, packet_string).group(1)
except AttributeError:
    src = "Unknown"

try:
    dst = re.search(dst_regex, packet_string).group(1)
except AttributeError:
    dst = "Unknown"

try:
    psrc = re.search(src_ipv4_regex, packet_string).group(1)
except AttributeError:
    psrc = "Unknown"

try:
    pdst = re.search(dst_ipv4_regex, packet_string).group(1)
except AttributeError:
    pdst = "Unknown"

try:
    load = re.search(load_regex, packet_string).group(1)
except AttributeError:
    load = "Unknown"

return (psrc, pdst, src, dst, 'TCP', load)

@staticmethod
def UDP(packet_string):
    # Define the regular expressions
    src_regex = r"src\s*=\s*(\w+:\w+:\w+:\w+:\w+:\w+)"
    dst_regex = r"dst\s*=\s*(\w+:\w+:\w+:\w+:\w+:\w+)"
    dst_ipv4_regex = r"dst\s*=\s*(\d{1,3}\.\d{1,3}\.\d{1,3}\.\d{1,3})"
    src_ipv4_regex = r"src\s*=\s*(\d{1,3}\.\d{1,3}\.\d{1,3}\.\d{1,3})"
    load_regex = r"###\[ IP \]###(.*)"

```



```
# Extract the values using regex
try:
    src = re.search(src_regex, packet_string).group(1)
except AttributeError:
    src = "Unknown"

try:
    dst = re.search(dst_regex, packet_string).group(1)
except AttributeError:
    dst = "Unknown"

try:
    psrc = re.search(src_ipv4_regex, packet_string).group(1)
except AttributeError:
    psrc = "Unknown"

try:
    pdst = re.search(dst_ipv4_regex, packet_string).group(1)
except AttributeError:
    pdst = "Unknown"

try:
    load = re.search(load_regex, packet_string).group(1)
except AttributeError:
    load = "Unknown"

return (psrc, pdst, src, dst, 'UDP', load)
```



sniffer_db.py: קובץ Python שמנהל את האינטראקציה עם מסד הנתונים SQLite עבור אחסון היסטוריית הסניפר.

```
import sqlite3

import datetime
from ysniffer import ysniffer
import threading
from time import sleep

class sniffer_db:
    sniffer_running = False
    sniffer_thread = None
    db_file = r'.venv\sniff_history.db'

    @staticmethod
    def start(iface):
        # Create a table
        timestamp = datetime.datetime.now().strftime("%Y%d%m%H%M%S")
        table_name = f"sniff_{timestamp}"
        sniffer_db.sniffer_running = True

        # Start the sniffer in a thread
        sniffer_db.sniffer_thread =
        threading.Thread(target=sniffer_db.sniffer_loop, args=(table_name, iface))
        sniffer_db.sniffer_thread.start()

    @staticmethod
    def sniffer_loop(table_name, iface):
        # Connect to the database
        conn = sqlite3.connect(sniffer_db.db_file)
        cursor = conn.cursor()

        # Create the table
        cursor.execute(f"""
            CREATE TABLE {table_name}
            (id INTEGER PRIMARY KEY AUTOINCREMENT,
             time TEXT,
             psrc TEXT,
             pdst TEXT,
```



```

        src TEXT,
        dst TEXT,
        protocol TEXT,
        load TEXT)
    """)

while sniffer_db.sniffer_running:
    sniffer = ysniffer()
    sniffer.sniff(iface)

    # Parse the packet data
    psrc, pdst, src, dst, protocol, load = sniffer.pck

    # Insert
    timestamp = datetime.datetime.now().strftime("%Y-%d-%m %H:%M:%S")
    cursor.execute(f"""
        INSERT INTO {table_name}
        (time, psrc, pdst, src, dst, protocol, load)
        VALUES (?, ?, ?, ?, ?, ?, ?)
    """, (timestamp, psrc, pdst, src, dst, protocol, load))
    conn.commit()

conn.close()

@staticmethod
def stop():
    sniffer_db.sniffer_running = False
    sniffer_db.sniffer_thread.join()

@staticmethod
def get_tables():
    # Connect to the database
    conn = sqlite3.connect(sniffer_db.db_file)
    cursor = conn.cursor()

    # Query for table names
    cursor.execute("SELECT name FROM sqlite_master WHERE type='table'")
    tables = [table[0] for table in cursor.fetchall()]
    tables.remove('sqlite_sequence')

    # Close the connection
    conn.close()

```



```

        return tables

    @staticmethod
    def get_db():
        return sniffer_db.db_file
        #using self breaks this... no idea why avoid changes
    @staticmethod
    def is_running():
        return sniffer_db.sniffer_running

if __name__ == "__main__":
    tables = sniffer_db.get_tables()
    print(tables)

    # Get the database file name
    db_file = sniffer_db.get_db()
    print(db_file)
    sniffer_db.start('Ethernet')
    if input() == 'c':
        sniffer_db.stop()

```



diagnostics.py: קובץ Python שמכיל פונקציות לביצוע פעולות אבחון רשת כמו השגת טבלת ARP ו-traceroute.

```

import subprocess
import socket

class Diagnostics:
    @staticmethod
    def get_arp_table():
        arp_table = []

        # Get the local ARP table
        local_arp_output = subprocess.check_output(["arp",
            "-a"]).decode("utf-8")
        local_arp_lines = local_arp_output.split("\n")
        for line in local_arp_lines:
            if line.strip():
                arp_table.append(line.split())

        # Get the router ARP table
        try:
            router_ip = socket.gethostbyname("router")
            router_arp_output = subprocess.check_output(["ssh", router_ip,
                "arp", "-a"]).decode("utf-8")
            router_arp_lines = router_arp_output.split("\n")
            for line in router_arp_lines:
                if line.strip():
                    arp_table.append(line.split())
        except (socket.gaierror, subprocess.CalledProcessError):
            pass

        return arp_table

    @staticmethod
    def traceroute(ip):
        traceroute_output = subprocess.check_output(["tracert", "-d",
            ip]).decode("utf-8")
        traceroute_lines = traceroute_output.split("\n")

        hops = []
        for line in traceroute_lines:
            if line.startswith(" "):

```



```

        hop_info = line.split()
        if len(hop_info) >= 8:
            hop = f"{hop_info[1]} {hop_info[2]} {hop_info[3]}
{hop_info[4]} {hop_info[7]}"
            hops.append(hop)

    return hops

def main():
    print("ARP Table:")
    arp_table = Diagnostics.get_arp_table()
    for entry in arp_table:
        print(" ".join(entry))

    print("\nTraceroute:")
    ip = input("Enter the IP address to traceroute: ")
    hops = Diagnostics.traceroute(ip)
    for hop in hops:
        print(hop)

if __name__ == "__main__":
    main()

```

main.tsx: נקודת הכניסה לאפליקציית ה-React, שם נעשה האתחול של הקומפוננטה הראשית ומעטפת ה-ChakraProvider.

```

import React from "react";
import ReactDOM from "react-dom/client";
import App from "./App";
import styled from "styled-components";
import "bootstrap/dist/css/bootstrap.css";
import "./index.css";
import { ChakraProvider } from "@chakra-ui/react";

ReactDOM.createRoot(document.getElementById("root") as HTMLElement).render(
  <React.StrictMode>
    <ChakraProvider>
      <App />
    </ChakraProvider>
  </React.StrictMode>

```




```
);
```

App.tsx: קומפוננטת React הראשית של האפליקציה, שמשתמשת

בקומפוננטת **TabsNavBar** לניווט.

```
import DarkModeButton from "../components/DarkModeButton";
import ExpandableText from "../components/ExpandableText";
import TabsNavBar from "../components/TabsNavBar";
import Sniffer from "../pages/Sniffer";

function App() {
  return (
    <div>
      <TabsNavBar></TabsNavBar>
    </div>
  );
}

export default App;
```

DarkModeButton.tsx: קומפוננטת React שמספקת כפתור להחלפה בין מצב

בהיר וכהה של הממשק.

```
import React from "react";
import { Button, useColorMode } from "@chakra-ui/react";
import { MoonIcon, SunIcon } from "@chakra-ui/icons";

const DarkModeButton = () => {
  const { colorMode, toggleColorMode } = useColorMode();

  return (
    <Button onClick={toggleColorMode}>
      {colorMode === "light" ? <MoonIcon /> : <SunIcon />}
    </Button>
  );
};
```



```
export default DarkModeButton;
```

PacketTable.tsx: קומפוננטת React שיוצרת ומציגה טבלה של נתוני חבילות רשת.

```
import React from "react";
import ExpandableText from "../ExpandableText";
import { Text, useColorModeValue } from "@chakra-ui/react";

interface PacketData {
  [key: string]: any;
}

interface PacketTableProps {
  packets: PacketData[];
}

const columnNames = [
  "Index",
  "Time",
  "IP Source",
  "IP Destination",
  "MAC Source",
  "MAC Destination",
  "Protocol",
  "Additional",
];

const PacketTable: React.FC<PacketTableProps> = ({ packets }) => {
  const textColor = useColorModeValue("black", "white");

  return (
    <table className="table table-bordered">
      <thead>
        <tr>
          {columnNames.map((name) => (
            <th key={name}>
              <Text color={textColor} fontSize="lg">
                {name}
              </Text>
            </th>
          ))}
        </tr>
      </thead>
    </table>
  );
}
```



```

        </th>
      )}
    </tr>
  </thead>
  <tbody>
    {packets.map((packet, index) => (
      <tr key={index}>
        {Object.values(packet).map((value, idx) => (
          <td key={idx}>
            {typeof value === "string" && value.length > 20 ? (
              <ExpandableText>{value}</ExpandableText>
            ) : (
              <Text color={textColor} fontSize="lg">
                {value}
              </Text>
            )}
          </td>
        )}
      </tr>
    )}
  </tbody>
</table>
);
};

export default PacketTable;

```

Iframe.tsx: קומפוננטת React גנרית שמשמשת כמעטפת מעוצבת לתוכן ילדים (Children).

```

import React from "react";
import { Box } from "@chakra-ui/react";

interface IframeProps {
  children: React.ReactNode;
}

const Iframe: React.FC<IframeProps> = ({ children }) => {

```



```
return (
  <Box
    borderWidth="4px"
    borderColor="#6AC26F"
    borderRadius="3x1"
    overflow="auto"
    maxHeight="500px"
    padding="4"
  >
    {children}
  </Box>
);
};

export default Iframe;
```

ExpandableText.tsx: קומפוננטת React שמאפשרת למשתמש להציג או להסתיר טקסט באמצעות לחיצה על כפתור.

```
import React, { useState } from "react";
import { Button, ButtonGroup } from "@chakra-ui/react";

interface ExpandableTextProps {
  children: React.ReactNode;
}

const ExpandableText = ({ children }: ExpandableTextProps) => {
  const [isExpanded, setIsExpanded] = useState(false);

  const toggleExpand = () => {
    setIsExpanded(!isExpanded);
  };

  return (
    <div>
      <Button
        type="button"
        onClick={toggleExpand}>
```



```

        data-bs-toggle="collapse"
        data-bs-target="#collapseExample"
        aria-expanded={isExpanded}
        aria-controls="collapseExample"
        onClick={toggleExpand}
        colorScheme="blue"
    >
        {isExpanded ? "Hide" : "Show"}
    </Button>
    <div
        className={`collapse ${isExpanded ? "show" : ""}`}
        id="collapseExample"
    >
        <div className="card card-body">{children}</div>
    </div>
</div>
);
};

export default ExpandableText;

```

Sniffer.tsx: קומפוננטת React שמספקת ממשק משתמש עבור כלי סניפר פשוט, עם אפשרויות סינון, התחלה/עצירה של הסניפר והורדת היסטוריה.

```

import React, { useState, useEffect } from "react";
import axios, { AxiosError } from "axios";
import Iframe from "../components/Iframe";
import PacketTable from "../components/PacketTable";
import {
    Box,
    Text,
    Heading,
    Input,
    Select,
    Button,
    Grid,
    useColorModeValue,
    Spinner,

```



```

    CircularProgress,
    useToast,
    FormControl,
    FormLabel,
  } from "@chakra-ui/react";

interface PacketData {
  [key: number]: any;
}

const Sniffer = () => {
  const [sourceIP, setSourceIP] = useState("");
  const [sourceMAC, setSourceMAC] = useState("");
  const [destinationIP, setDestinationIP] = useState("");
  const [destinationMAC, setDestinationMAC] = useState("");
  const [protocol, setProtocol] = useState("");
  const [packets, setPackets] = useState<PacketData[]>([]);
  const [filteredPackets, setFilteredPackets] = useState<PacketData[]>([]);
  const [tableList, setTableList] = useState<string[]>([]);
  const [selectedTable, setSelectedTable] = useState("");
  const [isSnifferRunning, setIsSnifferRunning] = useState(false);
  const [refreshKey, setRefreshKey] = useState(0);
  const [isDownloading, setIsDownloading] = useState(false);
  const [networkInterfaces, setNetworkInterfaces] = useState<string[]>([]);
  const [selectedInterface, setSelectedInterface] = useState("");
  const toast = useToast();

  useEffect(() => {
    const fetchTableList = async () => {
      try {
        const response = await axios.get("/tableList");
        const tables = response.data;
        setTableList(tables);
        setSelectedTable(tables[tables.length - 1]); // Set the default table
to the latest one
      } catch (error) {
        console.error("Error fetching table list:", error);
      }
    };

    fetchTableList();
  }, [refreshKey]);

```



```
useEffect(() => {
  const fetchData = async () => {
    try {
      const response = await axios.get(`/table/${selectedTable}`);
      const fetchedPackets = response.data;
      setPackets(fetchedPackets);
      setFilteredPackets(fetchedPackets);
      console.log("Fetched packets:", fetchedPackets);
    } catch (error) {
      console.error("Error fetching packet data:", error);
    }
  };

  if (selectedTable) {
    fetchData();
  }
}, [selectedTable, refreshKey]);

useEffect(() => {
  const fetchNetworkInterfaces = async () => {
    try {
      const response = await axios.get("/network_interfaces");
      const interfaces = response.data;
      setNetworkInterfaces(interfaces);
      setSelectedInterface(interfaces[0]); // Set the default interface to
the first one
    } catch (error) {
      console.error("Error fetching network interfaces:", error);
    }
  };

  fetchNetworkInterfaces();
}, []);

const handleFilter = () => {
  console.log("Filter values:", {
    sourceIP,
    sourceMAC,
    destinationIP,
    destinationMAC,
    protocol,
  });
};
```



```

});
console.log("Packets before filtering:", packets);

const filtered = packets.filter((packet) => {
    console.log("Packet:", packet);

    const matchesSourceIP = sourceIP
        ? packet[2].toLowerCase().includes(sourceIP.toLowerCase())
        : true;
    const matchesSourceMAC = sourceMAC
        ? packet[4].toLowerCase().includes(sourceMAC.toLowerCase())
        : true;
    const matchesDestinationIP = destinationIP
        ? packet[3].toLowerCase().includes(destinationIP.toLowerCase())
        : true;
    const matchesDestinationMAC = destinationMAC
        ? packet[5].toLowerCase().includes(destinationMAC.toLowerCase())
        : true;
    const matchesProtocol = protocol
        ? packet[6].toLowerCase() === protocol.toLowerCase()
        : true;

    console.log("Matches:", {
        matchesSourceIP,
        matchesSourceMAC,
        matchesDestinationIP,
        matchesDestinationMAC,
        matchesProtocol,
    });

    return (
        matchesSourceIP &&
        matchesSourceMAC &&
        matchesDestinationIP &&
        matchesDestinationMAC &&
        matchesProtocol
    );
});

setFilteredPackets(filtered);
console.log("Filtered packets:", filtered);
};
    
```




```
const handleStartStopSniffer = async () => {
  try {
    if (isSnifferRunning) {
      await axios.post("/stop_sniffer");
      setIsSnifferRunning(false);
    } else {
      await axios.post("/start_sniffer", { iface: selectedInterface });
      setIsSnifferRunning(true);
    }
    setRefreshKey((prevKey) => prevKey + 1);
  } catch (error) {
    console.error("Error starting/stopping sniffer:", error);
  }
};

const fetchSnifferStatus = async () => {
  try {
    const response = await axios.get("/is_sniffer_running");
    setIsSnifferRunning(response.data.is_running);
  } catch (error) {
    console.error("Error fetching sniffer status:", error);
  }
};

useEffect(() => {
  fetchSnifferStatus();
}, []);

const handleRefresh = () => {
  setRefreshKey((prevKey) => prevKey + 1);
};

const handleDownloadHistory = async () => {
  setIsDownloading(true);
  try {
    const response = await axios.get("/db", {
      responseType: "blob",
    });
    const url = window.URL.createObjectURL(new Blob([response.data]));
    const link = document.createElement("a");
    link.href = url;
  }
};
```



```

        link.setAttribute("download", "sniffer_history.db");
        document.body.appendChild(link);
        link.click();
    } catch (error) {
        console.error("Error downloading history:", error);
        if (axios.isAxiosError(error)) {
            const axiosError = error as AxiosError;
            if (axiosError.response && axiosError.response.status === 404) {
                toast({
                    title: "File Not Found",
                    description: "The history file was not found on the server.",
                    status: "error",
                    duration: 5000,
                    isClosable: true,
                });
            } else {
                toast({
                    title: "Download Error",
                    description:
                        "An error occurred while downloading the history file.",
                    status: "error",
                    duration: 5000,
                    isClosable: true,
                });
            }
        } else {
            toast({
                title: "Download Error",
                description:
                    "An unknown error occurred while downloading the history file.",
                status: "error",
                duration: 5000,
                isClosable: true,
            });
        }
    }
    setIsDownloading(false);
};

const textColor = useColorModeValue("black", "white");

return (

```



```

<div>
  <Box maxW="32rem">
    <Heading mb={4}>Sniffer Page 🚨</Heading>
    <Text fontSize="xl">A Simple CRUD Based Sniffer</Text>
  </Box>
  <Grid templateColumns="repeat(5, 1fr)" gap={4} mb={4}>
    <Input
      placeholder="Source IP"
      value={sourceIP}
      onChange={(e) => setSourceIP(e.target.value)}
    />
    <Input
      placeholder="Source MAC"
      value={sourceMAC}
      onChange={(e) => setSourceMAC(e.target.value)}
    />
    <Input
      placeholder="Destination IP"
      value={destinationIP}
      onChange={(e) => setDestinationIP(e.target.value)}
    />
    <Input
      placeholder="Destination MAC"
      value={destinationMAC}
      onChange={(e) => setDestinationMAC(e.target.value)}
    />
    <Select
      placeholder="Protocol"
      value={protocol}
      onChange={(e) => setProtocol(e.target.value)}
    >
      <option value="">All</option>
      <option value="TCP">TCP</option>
      <option value="UDP">UDP</option>
      <option value="ARP">ARP</option>
      <option value="ICMP">ICMP</option>
    </Select>
  </Grid>
  <Box display="flex" alignItems="center" mb={4}>
    <Button onClick={handleFilter} mr={4} px={4}>
      Filter
    </Button>
  </Box>

```



```

<Select
    value={selectedTable}
    onChange={(e) => setSelectedTable(e.target.value)}
    mr={4}
    minWidth="200px"
>
    {tableList.map((table) => (
        <option key={table} value={table}>
            {table}
        </option>
    ))}
</Select>
<Select
    value={selectedInterface}
    onChange={(e) => setSelectedInterface(e.target.value)}
    mr={4}
    minWidth="200px"
>
    {networkInterfaces.map((iface) => (
        <option key={iface} value={iface}>
            {iface}
        </option>
    ))}
</Select>
<Button
    onClick={handleStartStopSniffer}
    colorScheme={isSnifferRunning ? "red" : "blue"}
    leftIcon={isSnifferRunning ? <Spinner size="sm" /> : undefined}
    mr={4}
    px={6}
>
    {isSnifferRunning ? "Stop" : "Start"}
</Button>
<Button
    onClick={handleRefresh}
    bg="#6AC26F"
    color="white"
    mr={4}
    px={6}
>
    Refresh
</Button>

```



```

    <Button
      onClick={handleDownloadHistory}
      bg="#3477eb"
      color="white"
      disabled={isDownloading}
      px={6}
    >
      {isDownloading ? (
        <CircularProgress isIndeterminate size="24px" color="white" />
      ) : (
        "Download"
      )}
    </Button>
  </Box>
  <Iframe key={refreshKey}>
    {filteredPackets.length > 0 ? (
      <PacketTable packets={filteredPackets} />
    ) : (
      <Text color="red" fontSize="xl">
        NO PACKETS FOUND {"No Skibidi Rizz :("}
      </Text>
    )}
  </Iframe>
</div>
);
};

export default Sniffer;

```

Diagnostics.tsx: קומפוננטת React שמכילה כלים לאבחון רשת כמו טבלת
ARP וכלי traceroute.

```

import React, { useState, useEffect } from "react";
import axios from "axios";
import Iframe from "../components/Iframe";
import {
  Box,

```



```

Text,
Heading,
Input,
Button,
Grid,
useColorModeValue,
Table,
Thead,
Tbody,
Tr,
Th,
Td,
TableContainer,
List,
ListItem,
Spinner,
} from "@chakra-ui/react";

const Diagnostics = () => {
  const [arpTable, setArpTable] = useState<string[][]>([]);
  const [tracerouteIP, setTracerouteIP] = useState("");
  const [tracerouteHops, setTracerouteHops] = useState<string[]>([]);
  const [refreshKey, setRefreshKey] = useState(0);
  const [isLoadingArpTable, setIsLoadingArpTable] = useState(false);
  const [isLoadingTraceroute, setIsLoadingTraceroute] = useState(false);

  useEffect(() => {
    const fetchArpTable = async () => {
      setIsLoadingArpTable(true);
      try {
        const response = await axios.get("/arp_table");
        setArpTable(response.data);
      } catch (error) {
        console.error("Error fetching ARP table:", error);
      }
      setIsLoadingArpTable(false);
    };

    fetchArpTable();
  }, [refreshKey]);

  const handleRefresh = () => {

```



```

        setRefreshKey((prevKey) => prevKey + 1);
    };

    const handleTraceroute = async () => {
        setIsLoadingTraceroute(true);
        try {
            const response = await axios.post("/traceroute", { ip: tracerouteIP });
            setTracerouteHops(response.data);
        } catch (error) {
            console.error("Error performing traceroute:", error);
        }
        setIsLoadingTraceroute(false);
    };

    const bgColor = useColorModeValue("white", "gray.700");
    const textColor = useColorModeValue("black", "white");

    return (
        <div>
            <Box maxW="32rem">
                <Heading mb={4}>Diagnostics Page 🕒</Heading>
                <Text fontSize="xl">Network Diagnostics Tool</Text>
            </Box>
            <Grid templateColumns="repeat(2, 1fr)" gap={4} mb={4}>
                <Box bg={bgColor} p={4} borderRadius="md" boxShadow="md">
                    <Heading size="md" mb={4}>
                        ARP Table
                    </Heading>
                    <Iframe>
                        <TableContainer>
                            <Table variant="simple">
                                <Thead>
                                    <Tr>
                                        <Th>IP Address</Th>
                                        <Th>MAC Address</Th>
                                    </Tr>
                                </Thead>
                                <Tbody>
                                    {arpTable.map((entry, index) => (
                                        <Tr key={index}>
                                            <Td>{entry[0]}</Td>
                                            <Td>{entry[1]}</Td>
                                        </Tr>
                                    ))}
                                </Tbody>
                            </Table>
                        </TableContainer>
                    </Iframe>
                </Box>
            </Grid>
        </div>
    );

```



```

        </Tr>
    )})
</Tbody>
</Table>
</TableContainer>
</Iframe>
<Button
    onClick={handleRefresh}
    mt={4}
    bg="#6AC26F"
    color="white"
    isLoading={isLoadingArpTable}
    loadingText="Refreshing..."
    spinnerPlacement="end"
    disabled={isLoadingArpTable}
>
    Refresh
</Button>
</Box>
<Box bg={bgColor} p={4} borderRadius="md" boxShadow="md">
    <Heading size="md" mb={4}>
        Traceroute
    </Heading>
    <Input
        placeholder="Enter IP Address"
        value={tracerouteIP}
        onChange={(e) => setTracerouteIP(e.target.value)}
        mb={4}
    />
    <Button
        onClick={handleTraceroute}
        bg="#6AC26F"
        color="white"
        mb={4}
        isLoading={isLoadingTraceroute}
        loadingText="Tracing..."
        spinnerPlacement="end"
        disabled={isLoadingTraceroute}
    >
        Traceroute
    </Button>
</Iframe>

```




```

        <List spacing={3}>
          {tracerouteHops.map((hop, index) => (
            <ListItem key={index}>{hop}</ListItem>
          ))}
        </List>
      </Iframe>
    </Box>
  </Grid>
</div>
);
};

export default Diagnostics;

















```



מדריך למשתמש

התקנה

ודא שכל הקבצים והתיקיות של הפרויקט נמצאים במחשב.
כל החבילות וה Interpreter כלולים בסביבה הוירטואלית.

	_pycache_	27/05/2024 17:20	File folder	
	Include	18/05/2024 15:21	File folder	
	Lib	18/05/2024 15:21	File folder	
	Scripts	20/05/2024 12:51	File folder	
	share	18/05/2024 15:23	File folder	
	static	27/05/2024 17:20	File folder	
	templates	23/05/2024 0:13	File folder	
	app.py	22/05/2024 23:01	Python File	5 KB
	diagnostics.py	22/05/2024 14:21	Python File	2 KB
	packet_parser.py	19/05/2024 22:33	Python File	5 KB
	pyvenv.cfg	18/05/2024 15:21	Configuration Sou...	1 KB
	server.crt	27/05/2024 17:20	Security Certificate	2 KB
	server.key	27/05/2024 17:20	KEY File	2 KB
	sniff_history.db	27/05/2024 17:20	SQLite	936 KB
	sniffer_db.py	27/05/2024 17:20	Python File	3 KB
	ysniffer.py	22/05/2024 23:02	Python File	2 KB

הרצה

-הפעל את השרת על ידי הרצת הפקודה הבאה בטרמינל:

```
python app.py
```

השרת יתחיל לרוץ ויהיה זמין בכתובת 'https://localhost:8000'.



גישה (התחברות ל-WebApp)

פתח את הדפדפן שלך והיכנס לכתובת <https://localhost:8000>.
האפליקציה תיטען ותוצג בדפדפן.

שימוש

- בדף Sniffer, תוכל להשתמש בכלי ה-Sniffer:
- הזן את פרטי הסינון (Source IP, Source MAC, Destination IP, Destination MAC, Protocol) בשדות המתאימים.
- ❖ לחץ על כפתור "Filter" כדי לסנן את החבילות על סמך הפרטים שהוזנו.
 - ❖ בחר טבלה מהרשימה הנפתחת כדי להציג את תוכן הטבלה.
 - ❖ בחר ממשק רשת מהרשימה הנפתחת שבו ברצונך להפעיל את ה-Sniffer.
 - ❖ לחץ על כפתור "Start" כדי להתחיל את תהליך ה-Sniffing או על "Stop" כדי לעצור אותו.
 - ❖ לחץ על כפתור "Refresh" כדי לרענן את הנתונים המוצגים.
 - ❖ לחץ על כפתור "Download" כדי להוריד את היסטוריית ה-Sniffing.
- בדף האבחון Diagnostics, תוכל להשתמש בכלי האבחון של הרשת:
- ❖ הטבלה מציגה את טבלת ה-ARP עם כתובות ה-IP וה-MAC המתאימות.
 - ❖ לחץ על כפתור "Refresh" כדי לרענן את טבלת ה-ARP.
 - ❖ הזן כתובת IP בשדה הקלט ולחץ על כפתור "Traceroute" כדי לבצע traceroute לכתובת שצוינה.



רפלקציה

בתחילת הפרויקט, הרגשתי שאני מוכן ומזומן לאתגר שעומד בפניי, בזכות הידע והניסיון הקודם שצברתי. התקופה בתיכון הייתה מלאה באתגרים שתרמו להתפתחות האישית והמקצועית שלי, וידעתי שהניסיון הזה יסייע לי גם בפרויקט הנוכחי. עם זאת, גם עם כל הניסיון והידע שהיו לי, הפרויקט דרש ממני למידה של תחומים חדשים כמו שפת התכנות TypeScript, שהייתה מאתגרת במיוחד.

העבודה על הפרויקט התחילה באנרגיה גבוהה וברעיונות רבים, והשתמשתי בידע הקודם שלי כדי לבנות את הפיצ'רים השונים של הפרויקט בצורה שיטתית. התחלתי בעבודה על החלקים החשובים ביותר ולאחר מכן חיברתי ביניהם לכדי מוצר סופי. היכולת שלי לארגן ולתכנן את העבודה ביעילות התבררה כקריטית להצלחתי בפרויקט.

כשהבנתי שהפרויקט מורכב וגדול יותר משחשבתי בתחילה, הצלחתי לשמור על קצב עבודה עקבי בזכות הידע הקודם שלי והגישה המתודולוגית שנקטתי. למדתי במהלך הפרויקט כיצד לשלב ידע חדש עם ניסיון קודם, והרגשתי שאני מצליח להוציא מעצמי תוצרים מרשימים שלא ראיתי בעבר.

התהליך כלל גם רגעים מאתגרים, במיוחד במעבר לשימוש ב-TypeScript, אך ההתמודדות עם הקשיים הללו חיזקה את יכולותיי המקצועיות והאישיות. חקרתי לעומק נושאים חדשים, ביצעתי ניתוחים מורכבים ופיתחתי פתרונות יצירתיים. הפרויקט דרבן אותי להמשיך וללמוד, ובכך להתפתח באופן מתמיד.

אני גאה על כך שהצלחתי לעמוד בקריטריונים הגבוהים שהצבתי לעצמי, ושפרויקט זה העניק לי תחושת הישג וגאווה. הפרויקט היה עבורי הזדמנות לחוות את עולם הסייבר בצורה מעמיקה וישירה, ובכך חיזק את תחושת השייכות שלי לתחום ואת הרצון לעסוק בו בעתיד. בסופו של דבר, הפרויקט הוכיח לי שהידע והניסיון הקודם שלי הם נכס יקר, ושבעזרתם אוכל להתמודד עם כל אתגר מקצועי שיבוא בעתיד.



ביבליאוגרפיה

HTML.com Tutorials - <https://html.com/>

TypeScript Handbook -

<https://www.typescriptlang.org/docs/handbook/intro.html>

Bootstrap by Example -

<https://www.tutorialrepublic.com/twitter-bootstrap-tutorial/>

Using Axios with React -

<https://www.digitalocean.com/community/tutorials/react-axios-react>

Using Chakra UI -

<https://www.smashingmagazine.com/2021/06/getting-started-chakra-ui/>

Axios Documentation - <https://axios-http.com/docs/intro>

JSON.org - <https://www.json.org/json-en.html>

MDN Web Docs: CSS - <https://developer.mozilla.org/en-US/docs/Web/CSS>

Flask by Example -

<https://realpython.com/flask-by-example-part-1-project-setup/>

RSA (cryptosystem) - Wikipedia -

[https://en.wikipedia.org/wiki/RSA_\(cryptosystem\)](https://en.wikipedia.org/wiki/RSA_(cryptosystem))

Official TypeScript Documentation - <https://www.typescriptlang.org/docs/>

Official Bootstrap Documentation - <https://getbootstrap.com/docs/>

Learn CSS Layout - <http://learnlayout.com/>

Understanding RSA Encryption -

https://www.tutorialspoint.com/cryptography_with_python/cryptography_with_python_rsa_encryption.htm

MDN Web Docs: JSON -

<https://developer.mozilla.org/en-US/docs/Learn/TypeScript/Objects/JSON>

Real Python Tutorials - <https://realpython.com/>

MDN Web Docs: HTTP - <https://developer.mozilla.org/en-US/docs/Web/HTTP>

Python Documentation - <https://docs.python.org/3/>

Official Chakra UI Documentation - <https://chakra-ui.com/docs>

Flask Mega-Tutorial by Miguel Grinberg -

<https://blog.miguelgrinberg.com/post/the-flask-mega-tutorial-part-i-hello-world>

TypeScript Deep Dive - <https://basarat.gitbook.io/typescript/>



Automate the Boring Stuff with Python -

<https://automatetheboringstuff.com/>

MDN Web Docs: HTML -

<https://developer.mozilla.org/en-US/docs/Web/HTML>

MDN Web Docs: HTTPS -

https://developer.mozilla.org/en-US/docs/Web/Security/HTTP_strict_transport_security

CSS Tricks - <https://css-tricks.com/>

Official Axios GitHub Repository - <https://github.com/axios/axios>

JSON Tutorial by TutorialsPoint -

<https://www.tutorialspoint.com/json/index.htm>

Official Flask Documentation - <https://flask.palletsprojects.com/en/2.0.x/>

