

שם בית הספר: תיכון הרצוג

שם העבודה: IP Grabber

שם התלמיד: עידן רטיק

תעודת זהות: 216239939

שם המנחה: אופיר שביט

שם החלופה:

תאריך ההגשה: 24/05/25

Table of Contents

[מבוא 3](#_Toc198749448)

[ייזום 3](#_Toc198749449)

**[תיאור ראשוני של המערכת](#_Toc198749450)** [3](#_Toc198749450)

[**הגדרת יעדים ומטרות** 3](#_Toc198749451)

[סקירת פתרונות קיימים 3](#_Toc198749452)

[**Grabify** 3](#_Toc198749453)

[**IPLogger** 5](#_Toc198749454)

[**IP Tracker online** 6](#_Toc198749455)

[**סקירת טכנולוגיות בפרויקט** 7](#_Toc198749456)

[**סייגים והגבלות** 7](#_Toc198749457)

[**תיחום הפרוייקט** 7](#_Toc198749458)

[אפיון המערכת 8](#_Toc198749459)

[**יכולות ההורה** 8](#_Toc198749460)

[**פירוט בדיקות** 9](#_Toc198749461)

[**לו״ז** 11](#_Toc198749462)

[פירוט יכולות 11](#_Toc198749463)

[יכולות בצד הלקוח 11](#_Toc198749464)

[יכולות בצד השרת 13](#_Toc198749465)

[רקע תיאורטי 14](#_Toc198749466)

[מונחים מתקדמים ברשתות 14](#_Toc198749467)

[**ARP spoofing** 14](#_Toc198749468)

[**איש באמצע – MITM** 15](#_Toc198749469)

[**הרעלת DNS** 15](#_Toc198749470)

[הטכנולוגיה הרלוונטית 15](#_Toc198749471)

[**שפות תכנות** 15](#_Toc198749472)

[**מערכות הפעלה** 15](#_Toc198749473)

[**תקשורת** 16](#_Toc198749474)

[מבנה וארכיטקטורה של הפרוייקט 16](#_Toc198749475)

[תיאור הארכיטקטורה של המערכת 16](#_Toc198749476)

[**תיאור החומרה** 16](#_Toc198749477)

[**החומרה בשרטוט** 17](#_Toc198749478)

[זרימת מידע 18](#_Toc198749479)

[**כללי** 18](#_Toc198749480)

[**תת תהליך MITM + הרעלת DNS.** 19](#_Toc198749481)

# **מבוא**

## **ייזום**

### **תיאור ראשוני של המערכת**

אני מפתח כלי IP Grabber שמאפשר לאסוף ולאחסן כתובות IP של משתמשים שמבקרים בדף אינטרנט מסוים. הכלי יאפשר גם איסוף מידע נוסף כמו פרטי מערכת הפעלה, דפדפן, ומיקום גיאוגרפי משוער. המוצר המוגמר יהיה פלטפורמה שמאפשרת יצירה ושימוש ב-IP Grabber באמצעות ממשק פשוט ונוח למשתמש.

בחרתי בפרויקט זה כדי להעמיק את הידע שלי באבטחת תקשורת, בטכניקות מתקדמות ברשתות, ופיתוח. הפרויקט משלב היבטים של רשתות, מערכות הפעלה, פיתוח צד-שרת וצד-לקוח, וניתוח נתונים.

רוב הקשיים שאני צופה בהכנה של המערכת יהיו קשורים, לטכניקות מתקדמות ברשתות כגון ARP spoofing, וDNS spoofing, בתקשורת בין פרוססים

המערכת מיועדת להורים, שמעוניינים להשגיח על גישת ילדיהם לאתרים מסוימים. בעזרת שימוש במוצר, יוכלו ההורים לבדוק בצורה מדויקת מתי ומי נכנס לאתרים מסוימים.

### **הגדרת יעדים ומטרות**

* יצירת כלי פשוט וידידותי למשתמש לאיסוף כתובות IP, ללא פעולה אקטיבית של המתחברים לאתר.
* אפשרות לאחסון והצגת הכתובות שנאספו בממשק מאורגן
* יצירת גרפים וניתוחים סטטיסטיים של הנתונים שנאספו
* אפשרות לייצא את הנתונים בפורמטים שונים

חסכונות:

חיסכון בזמן באיתור והתמודדות עם גישות חשודות

ייעול תהליכי ניטור אבטחה

הפחתת הצורך בכלים יקרים יותר לניטור תעבורה

## **סקירת פתרונות קיימים**

### **Grabify**

**סקירה כללית:**

Grabify הוא כלי מבוסס-ווב המאפשר יצירת קישורים מקוצרים לצורכי מעקב אחר מידע טכני של גולשים, בדגש על איתור כתובת ה-IP, מידע על הדפדפן, מערכת ההפעלה, המדינה, ספק האינטרנט ועוד. המשתמש יוצר קישור דרך הפלטפורמה, משתף אותו עם אחרים, וכל גישה לקישור מתועדת.

לאחר שמזינים את הURL שאותו רוצים להסוות, מקבלים קוד שבעזרתו ניגשים אל הנתונים לאחר מכן.

**יתרונות:**

פשטות שימוש - ממשק אינטואיטיבי וידידותי למשתמש, גם ללא ידע טכני.

גישה מיידית למידע - ניתן לצפות במידע שנאסף מיידית לאחר לחיצה על הקישור.

תוספים נלווים - כולל אפשרויות להצגת עמודי הפניה מזויפים (fake redirect) שגורמים למשתמש לחשוב שמדובר באתר רגיל.

מערכת קיצור כתובות - המערכת מקצרת כתובות באופן אוטומטי ומסתירה את מטרת הקישור האמיתית.

**חסרונות:**

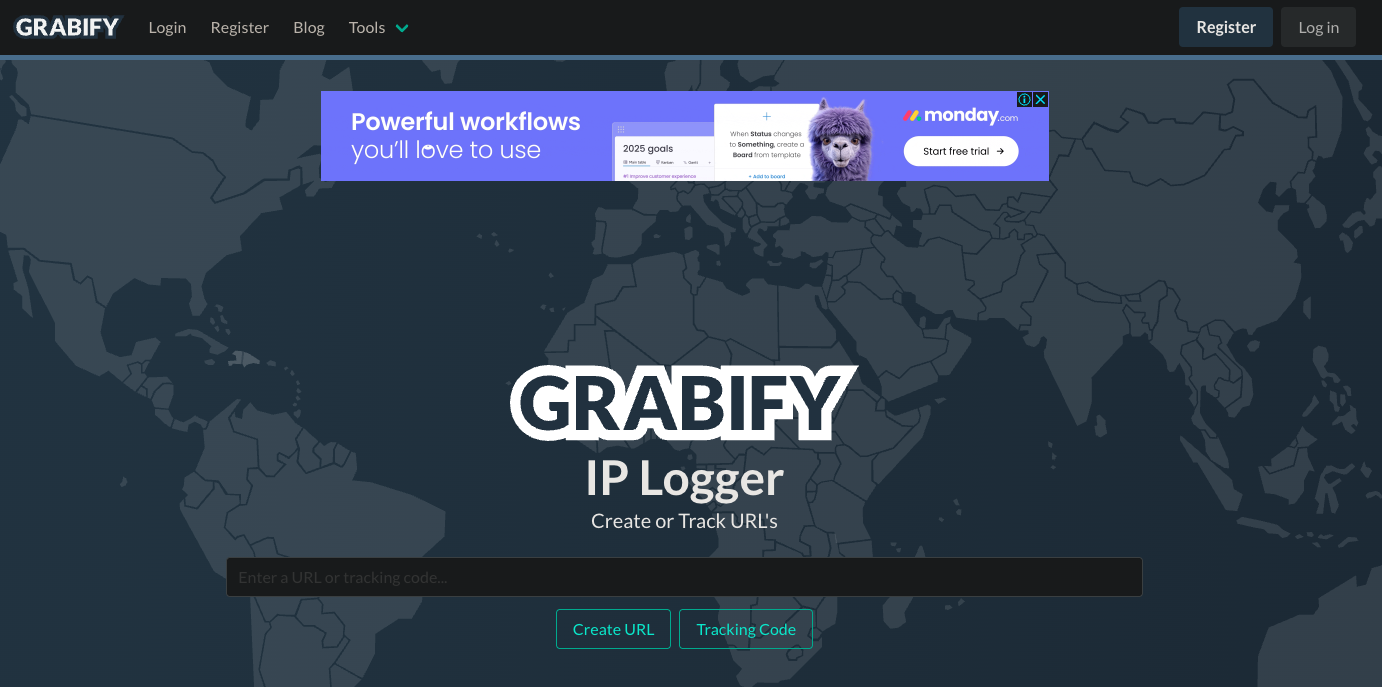
פונקציונליות מוגבלת - הכלי אוסף רק נתונים בסיסיים ואינו מספק כלי ניתוח מתקדמים או תיעוד היסטורי מתוחכם.

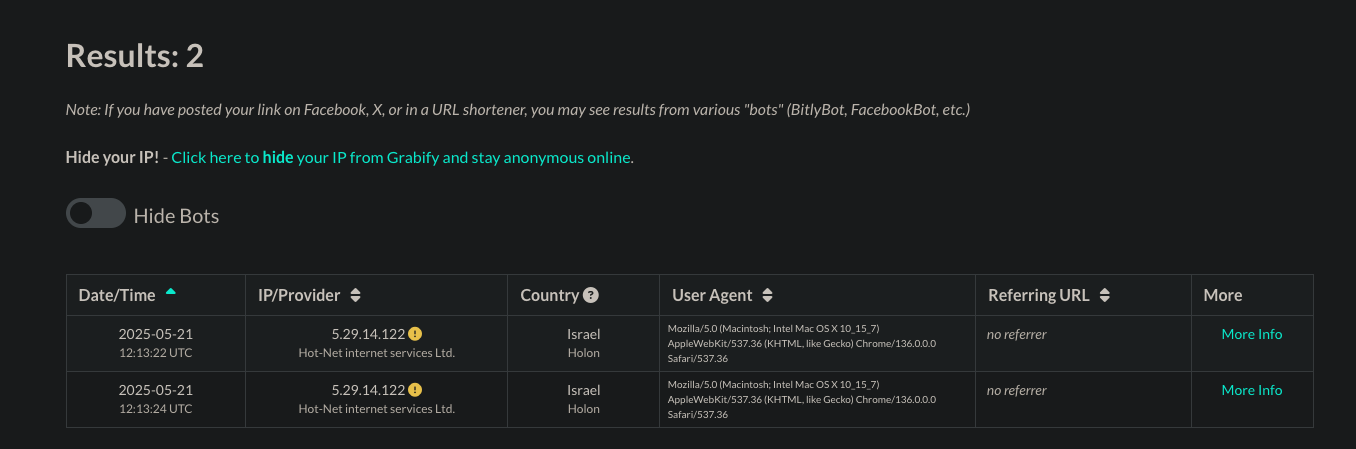
קל לזיהוי - מנגנוני אבטחה (כגון אנטי וירוס או דפדפנים) לעיתים מזהים את הקישורים כחשודים.

חוסר בהתאמה אישית -אין אפשרות ליצור דף נחיתה מותאם אישית או ממשק API מתקדם.

**אופן הפעולה**: כאשר נוצר קישור, Grabify עוקב אחרי כל מי שלוחץ עליו, שומר את כתובת ה-IP ומידע נוסף. ניתן להוסיף הפניות לדפים אחרים כך שהמשתמש לא ישים לב שהוא במעקב.

**ייחודיות:** מיועד בעיקר לשימוש אישי ולא לארגונים. הדגש הוא על קלות ופשטות ולא על עושר בפונקציות.

. 

****

### **IPLogger**

**סקירה כללית:**  
IPLogger הוא שירות ותיק יותר מ-Grabify, והוא מציע קישור למעקב אחרי גולשים עם פיצ'רים מתקדמים יותר. הכלי מציע אפשרות ליצירת כתובות מסוגים שונים, כגון תמונה מוסתרת (invisible image), פיקסל בדוא"ל, או דף הפניה מותאם.

**יתרונות:**

מגוון אפשרויות מעקב - ניתן לבחור בין קישור רגיל, תמונה שקופה, כפתור, או קובץ להורדה.

שירותי API - מאפשר אינטגרציה עם מערכות קיימות או כתיבת סקריפטים מותאמים אישית.

נתוני גיאולוקציה מדויקים - מזהה מדינה, עיר, דפדפן, רזולוציית מסך, ונתוני HTTP נוספים.

מעקב בזמן אמת - תיעוד כל גישה כולל מיקום, שעה, ומידע על המכשיר.

**חסרונות:**

ממשק מיושן - הממשק נראה ישן ומעט עמוס.

יותר מדי פרסומות - השירות החינמי כולל כמות גדולה של פרסומות, מה שפוגע בחוויית השימוש.

ללא מנגנוני הסוואה מתקדמים - קישור שנוצר לא תמיד מוסתר בצורה שמונעת חשד.

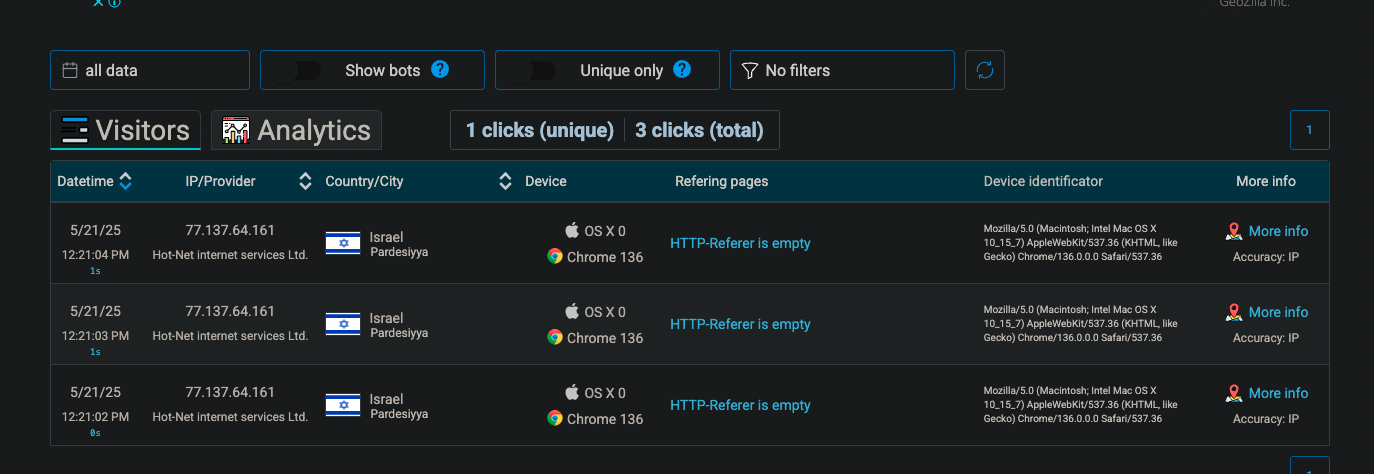
**אופן הפעולה**:

IPLogger מספק כמה דרכים "להחביא" את כלי המעקב – למשל, על ידי החדרת תמונה בלתי נראית באימיילים או דפי אינטרנט. כל כניסה או טעינה של הקובץ גורמת לתיעוד מלא של המשתמש.

**ייחודיות:**

פונה גם למשתמשים מתקדמים – כולל API, פיצ’רים מותאמים אישית, ויכולות מעקב מורחבות. מתאים לשילוב בפרויקטים גדולים יותר.

****

****

### **IP Tracker online**

**סקירה כללית:**  
IP Tracker Online הוא כלי פשוט אך זמין, המיועד לאנשים שרוצים לבדוק כתובת IP ספציפית או לבצע מעקב אחרי קישור אחד, לרוב ללא התחברות לחשבון. הוא אינו כלי קיצור כתובות אלא יותר שירות לבדיקת כתובות IP קיימות..

**יתרונות:**

כלי חינמי לגמרי - ללא צורך ברישום או תשלום – כל הכלים פתוחים.

בדיקת IP קיימת - בניגוד לשני הכלים האחרים, ניתן להזין כתובת IP ידנית ולבדוק עליה פרטים.

**חסרונות:**

**פונקציונליות בסיסית בלבד:** - אין אפשרות ליצור קישורים למעקב – אלא רק לבדוק IP ידוע מראש.

אין API – לא נועד לשימוש אוטומטי, או מקצועי.

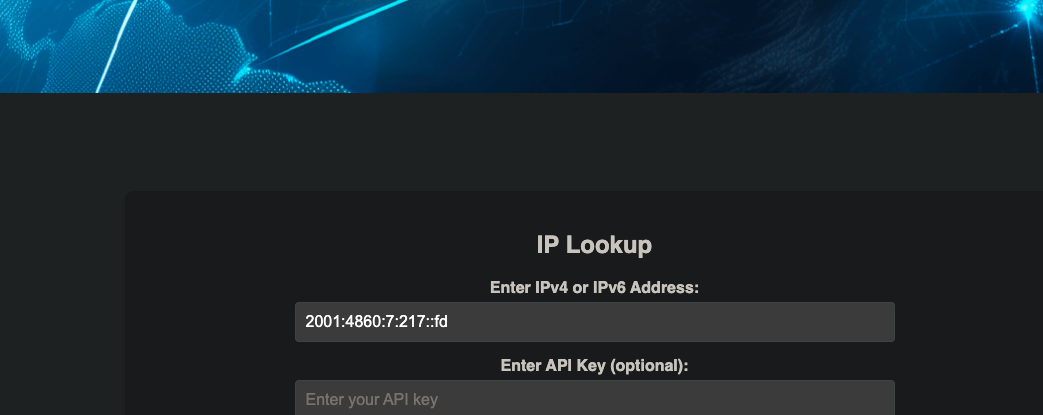
אין תיעוד – כל בדיקה היא חד פעמית, ולא נשמרת.

**אופן הפעולה**:

המשתמש מזין כתובת IP או שם דומיין, והמערכת מפיקה מידע הכולל מיקום גאוגרפי, ספק אינטרנט, ולעתים גם מעלה חשד אם מדובר בproxy או בVPN.

**ייחודיות:**

פועל כמזהה IP ולא ככלי מעקב.



### **סקירת טכנולוגיות בפרויקט**

צד לקוח: HTML, FLASK, CSS, SOCKET/TCP

צד שרת: HTTP,SCAPY, DNS, ARP, Multiprocessing ,Signal ,socket/TCP

נתונים: Json

### **סייגים והגבלות**

על המשתמש להכניס בצורה מפורשת את הURL שדרכו מתבצע המעקב.

חלק מהטכנולוגיות לא יעבדו כראוי אם יש רשת עם הרבה הגנות, או אם משתמשים בדפדפן זהיר.

### **תיחום הפרוייקט**

הפרויקט עוסק במספר תחומים הקשורים לרשתות ומערכות הפעלה -

רשתות:

תקשורת בין לקוח לשרת בעזרת SOCKET

שימוש בפרוטוקול ARP כדי לנתב את בקשות הDNS אל השרת.

שימוש בפרוטוקול DNS כדי לגרום לURL שלא קיימים לעבוד.

שימוש בScapy כדי לשלוח פקטות.

שימוש בFLASK למימוש שרת http בשביל gui בסיסי

שימוש בפרוטוקול http כדי לבצע redirect.

מערכות הפעלה:

שימוש בmultiprocessing בצד השרת.

שימוש בPIPE כדי לדבר בין הפרוססים השונים (IPC)  
שימוש במערכת קבצים לשמירת מידע.

שימוש בsignal כדי לצאת מprocess בצורה נקייה.

הפרויקט יתמקד באיסוף מידע באופן פסיבי בלבד (כלומר, מידע שנשלח כחלק מהבקשות הרגילות) ולא יכלול איסוף אקטיבי או חדירה למערכות.

בנוסף, הפרוייקט יעבוד אך ורק בין מכשירים שנמצאים באותה הרשת. לא קיימת בפרוייקט התעסקות או התייחסות לNAT, port forwarding, וכדו׳.

## **אפיון המערכת**

המערכת משמשת בתור כלי של ההורה, כדי לנטר ולבדוק מתי כל ילד התחבר לאתרים מסוימים.

השרת יחזיק אצלו בצורה לא מוצפנת מאגר של קישורים. המאגר יקשר כל קישור מומצא לקישור אמיתי.

ההורה יתחבר אל המערכת ושם יוצגו לו האופציות, אם להתחבר בתור משתמש רשום או להירשם.

לאחר שיתחבר בתור משתמש, יוכל לנהל בצורה דינמית את מאגר הקישורים שמוחזק בצד הלקוח.

כאשר מישהו ישתמש בקישור המומצא, המערכת תתעד את זמן הכניסה, כתובת ה-IP ממנה בוצעה הגישה, ופרטים נוספים. בנוסף, המערכת תנתב את הילד לקישור האמיתי מבלי שירגיש שהוא מנוטר.

כאשר יתחבר ההורה אל המערכת, תהיה לו אופציה לעבור לעמוד שבו יוכל לראות את כל הכניסות שבוצעו, למיין אותם לפי שעה, לפי מחשב, ועוד. ההורה יוכל לראות גרפים וסטטיסטיקות על תדירות השימוש, שעות הגלישה המועדפות, והתפלגות הגלישה בין המשתמשים השונים, ההורה יכול לייצא את הנתונים הללו לאקסל.

מערכת זו לא תדרוש שום דבר מהילד, חוץ מלהכניס את הקישור המזויף שניתן לו.

### **יכולות ההורה**

* ליצור חשבון.
* להתחבר לחשבון.
* לבקש את הקישור האמיתי, של קישור מסוים, מהמאגר.
* להכניס אל המאגר קישור חדש.
* להוציא מהמאגר קישור קיים.
* לבקש את כל הכניסות לאתר מסוים.
* לעבור בין התפריטים.

### **פירוט בדיקות**

|  |  |  |  |
| --- | --- | --- | --- |
| מס' | יכולת לבדיקה | תיאור הבדיקה | איך בודקים? |
| 1 | יצירת משתמש (הורה) | בדיקת תהליך הרשמה של משתמש חדש למערכת | מריצים את אפליקציית הממשק, נכנסים לדף ההרשמה וממלאים את הטופס עם פרטים תקינים. מוודאים שהמשתמש נוצר במסד הנתונים ושאפשר להתחבר עם הפרטים החדשים. יש לבדוק גם מקרי קצה: הכנסת פרטים לא תקינים, שם משתמש קיים, וסיסמה חלשה. |
| 2 | התחברות משתמש | בדיקת התחברות משתמש קיים | יוצרים משתמש מראש, מתנתקים מהמערכת ומנסים להתחבר מחדש עם שם המשתמש והסיסמה. בודקים שהמערכת מזהה נכון את פרטי ההתחברות ומציגה את ממשק הניהול. יש לבדוק גם התחברות עם פרטים שגויים. |
| 3 | יצירת קישור מזויף | בדיקת יצירת קישור מזויף וקישורו לאתר אמיתי | נכנסים לממשק ניהול הקישורים, יוצרים קישור חדש עם שם ייחודי והאתר האמיתי (למשל google.com). בודקים שהקישור נוצר במסד הנתונים ומוצג ברשימת הקישורים בממשק. |
| 4 | ARP Spoofing | בדיקת יכולת התחזות לנתב | מפעילים את מנגנון ה-ARP Spoofing כאשר Wireshark פועל על מחשב אחר ברשת. בודקים ב-Wireshark שחבילות ה-ARP המזויפות נשלחות לרשת, ושהמחשבים ברשת מעדכנים את טבלאות ה-ARP שלהם להצביע על ה-MAC של השרת שלנו במקום זה של הנתב. |
| 5 | יירוט בקשות DNS | בדיקת זיהוי ויירוט בקשות DNS לדומיינים מזויפים | מפעילים את מנגנון יירוט ה-DNS ואת Wireshark. מהמחשב של משתמש הקצה, מנסים לגשת לדומיין מזויף שיצרנו. בודקים ב-Wireshark שהשרת שלנו מספק תשובת DNS מזויפת המפנה ל-127.0.0.1 או לכתובת IP של השרת שלנו. |
| 6 | ניהול HTTP Server | בדיקת שרת ה-HTTP שמקבל את הפניות | מפעילים את שרת ה-HTTP ומוודאים שהוא עולה ללא שגיאות. מבצעים בקשות HTTP מפוזרות לשרת וצופים בלוגים כדי לוודא שהשרת מקבל אותן כראוי. בודקים את זמני התגובה תחת עומס על ידי יצירת מספר בקשות במקביל. |
| 7 | הפניה לאתר אמיתי | בדיקת הפניית משתמש מהקישור המזויף לאמיתי | לאחר הפעלת כל המנגנונים, ניגשים לקישור מזויף מדפדפן משתמש הקצה. בודקים שהדפדפן מופנה מיד לאתר האמיתי, וצופים ב-Developer Tools של הדפדפן כדי לוודא שמתקבלת הפניית 302 תקינה. |
| 8 | תיעוד כניסות | בדיקת תיעוד מידע על כניסות לקישורים מזויפים | מבצעים גישה לקישור מזויף ממספר מכשירים, דפדפנים ורשתות שונות. לאחר מכן, בודקים במסד הנתונים שכל הכניסות תועדו עם הפרטים המתאימים: כתובת IP, זמן הכניסה, סוג הדפדפן והמכשיר. |
| 9 | הצגת נתונים בממשק הורה | בדיקת הצגת סטטיסטיקות ונתוני כניסה | לאחר צבירת מספר כניסות במערכת, נכנסים לממשק ההורה ובודקים שהנתונים מוצגים נכון בגרפים ובטבלאות. בודקים את העדכון בזמן אמת על ידי פתיחת הממשק במקביל לביצוע כניסות חדשות. |
| 10 | תמיכה בריבוי משתמשים | בדיקת תמיכה במספר הורים ומספר ילדים | יוצרים מספר חשבונות משתמשים (הורים) ולכל אחד מהם מספר פרופילים של ילדים. בודקים שכל משתמש רואה רק את הקישורים והנתונים השייכים אליו, ולא את אלו של משתמשים אחרים. |
| 11 | ביצועי MITM | בדיקת יכולת Man-in-the-Middle | מפעילים את מנגנון ה-MITM ובמקביל מבצעים פעולות גלישה שונות ברשת. בודקים בעזרת Wireshark שהתעבורה עוברת דרך השרת שלנו, ושאין עיכובים משמעותיים או שגיאות תקשורת. |
| 12 | זיהוי DNS המזויף | בדיקת יכולת זיהוי דומיין מזויף | מפעילים את מנגנון יירוט ה-DNS ושולחים בקשות DNS הן לדומיינים מזויפים והן לדומיינים לגיטימיים. בודקים שהמערכת מזהה ומטפלת רק בדומיינים המזויפים ולא מפריעה לתעבורה הרגילה. |
| 13 | ייצוא נתונים | בדיקת ייצוא נתוני שימוש לפורמטים שונים | צוברים כמות מסוימת של נתוני כניסה במערכת, ואז מנסים לייצא אותם בפורמטים שונים (Excel, PDF). פותחים את הקבצים המיוצאים ובודקים שכל הנתונים הרלוונטיים נמצאים בהם ומוצגים נכון. |
| 14 | מולטי-פרוססינג | בדיקת פעולת המערכת במספר תהליכים במקביל | מפעילים את כל רכיבי המערכת במקביל (ARP Spoofing, יירוט DNS, שרת HTTP, ממשק ניהול). בודקים את צריכת המשאבים של המערכת ומוודאים שכל התהליכים מתקשרים ביניהם כראוי. מבצעים פעולות במקביל ובודקים שאין התנגשויות או חסימות. |
| 15 | עמידות בתקלות | בדיקת התנהגות המערכת במקרה של תקלות | מדמים מצבי תקלה שונים: ניתוק רשת פתאומי, כיבוי אחד התהליכים, ניסיון גישה לדומיין לא קיים. בודקים שהמערכת מתאוששת או מטפלת בתקלות בצורה הולמת ולא קורסת. |
| 16 | אבטחת מידע | בדיקת אבטחת המידע הנאסף | מנסים לגשת למידע השמור ללא הרשאות מתאימות. בודקים שתקשורת בין הלקוח לשרת מאובטחת. בודקים חסינות מפני התקפות נפוצות כמו SQL Injection או XSS. |
| 17 | פעולה ברשתות שונות | בדיקת פעולת המערכת ברשתות בעלות תצורות שונות | מפעילים את המערכת ברשתות בעלות תצורות שונות: רשת ביתית רגילה, רשת עם נתב מתקדם, רשת עם הגנות אבטחה. בודקים שהמערכת פועלת כראוי בכל הסביבות. |

### **לו״ז**

|  |  |
| --- | --- |
| **תאריך** | **משימה** |
| 31.11 | אפיון בסיסי של הפרוייקט |
| 1.1 | הוכחות יכולות בסיסיות (ARP spoofing, dns poisoning) |
| 3.1 | מימוש תקשורת שרת לקוח עם ממשק טקטסואלי |
| 31.3 | חיבור התקשורת עם כל היכולות |
| 10.4 | הכנת ממשק גרפי |
| 31.4 | סיום הכנת קוד כולל בדיקת באגים |

# **פירוט יכולות**

## **יכולות בצד הלקוח**

|  |  |  |  |
| --- | --- | --- | --- |
| **שם** | **מהות** | **יכולות הדרושות למימוש היכולת** | **אובייקטים נחוצים** |
| יצירת משתמש | שליחת לשרת הודעה מוצפנת של יצירת משתמש עם הפרטים הנדרשים. | שליחת הודעה לשרת, בניית הודעה לפי פרוטוקול, קבלת קלט משתמש | תקשורת מוצפנת עם השרת, סביבת משתמש גרפית, מאגר משתמשים, טיפול בשגיאות שהשרת מחזיר. |
| התחברות למשתמש | שליחת הודעה מוצפנת לשרת של התחברות למשתמש. | שליחת הודעה לשרת, בניית הודעה לפי פרוטוקול, קבלת קלט משתמש | תקשורת מוצפנת עם השרת, סביבת משתמש גרפית, מאגר משתמשים, טיפול בשגיאות שהשרת מחזיר. |
| בקשת מידע מהשרת | שליחת לשרת הודעה מוצפנת של לפי הפרוטוקול | שליחת הודעה לשרת, בניית הודעה לפי פרוטוקול, קבלת קלט משתמש | תקשורת מוצפנת עם השרת, סביבת משתמש גרפית, מאגר משתמשים, טיפול בשגיאות שהשרת מחזיר, מאגר קישורים. |
| הוספה של קישור למאגר | שליחת לשרת הודעה מוצפנת של לפי הפרוטוקול | שליחת הודעה לשרת, בניית הודעה לפי פרוטוקול, קבלת קלט משתמש | תקשורת מוצפנת עם השרת, סביבת משתמש גרפית, מאגר משתמשים, טיפול בשגיאות שהשרת מחזיר, מאגר קישורים. |
| מחיקת קישור מהמאגר | שליחת לשרת הודעה מוצפנת של לפי הפרוטוקול | שליחת הודעה לשרת, בניית הודעה לפי פרוטוקול, קבלת קלט משתמש | תקשורת מוצפנת עם השרת, סביבת משתמש גרפית, מאגר משתמשים, טיפול בשגיאות שהשרת מחזיר, מאגר קישורים. |
| בקשה של פירוט התחברויות לאתר מסוים | שליחת לשרת הודעה מוצפנת של לפי הפרוטוקול | שליחת הודעה לשרת, בניית הודעה לפי פרוטוקול, קבלת קלט משתמש | תקשורת מוצפנת עם השרת, סביבת משתמש גרפית, מאגר משתמשים, טיפול בשגיאות שהשרת מחזיר, מאגר קישורים. |
| בקשת הקישור האמיתי התואם לקישור מזויף מסוים. | שליחת לשרת הודעה מוצפנת של לפי הפרוטוקול | שליחת הודעה לשרת, בניית הודעה לפי פרוטוקול, קבלת קלט משתמש | תקשורת מוצפנת עם השרת, סביבת משתמש גרפית, מאגר משתמשים, טיפול בשגיאות שהשרת מחזיר, מאגר קישורים. |
| החלפת מפתחות | החלפת מפתחות AES בעזרת RSA | תקשורת בין שרת ללקוח | עוטף לסוקט, מפתח RSA |
| הצפנה ופענוח מידע | הצפנה ופענוח מידע שנשלח או מגיע מהשרת בעזרת AES | תקשורת בין שרת ללקוח, החלפת מפתחות | עוטף לסוקט, מפתח AES |
| שליחת הודעה מוצפנת לשרת | שילוב בין יכולת הצפנת המידע לבין יכולת שליחת הודעה לשרת. | שליחת הודעה לשרת, הצפנת מידע | עוטף לסוקט, מפתח AES |
| בניית הודעה לפי פרוטוקול | בניית הודעה לפי הפרוטקול, בהתאם לקלט המשתמש. |  | עוטף לסוקט. |
| שליחת הודעה לשרת |  | יצירת תקשורת עם השרת | עוטף לסוקט. |
| יצירת תקשורת עם השרת. | Connect |  | עוטף לסוקט. |
| קבלת קלט משתמש | שימוש בממשק גרפי כדי לקבל ולנתח קלט משתמש. | הצגה ושימוש בממשק גרפי | ממשק גרפי. |
| הצגה ושימוש בממשק גרפי. | הצגת כל התפריטים, מעבר בינהם, והזנת נתונים. |  | Flask |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

## **יכולות בצד השרת**

|  |  |  |  |
| --- | --- | --- | --- |
| **שם** | **מהות** | **יכולות הדרושות למימוש היכולת** | **אובייקטים נחוצים** |
| יצירת משתמש חדש | רישום משתמש חדש לצורך שימוש בפלטפורמה | ממשק משתמש, אימות קלט, כתיבה למסד נתונים | טופס רישום, מסד נתונים |
| בדיקת התחברות | אימות זהות משתמש קיים | אימות מול מסד נתונים, השוואת סיסמה, | טופס התחברות, מסד נתונים |
| הוספת קישור למאגר קישורים | יצירת URL שמוביל לסקריפט לכידת IP | יצירת קישור ייחודי, אחסון במסד נתונים | מסד נתונים, ממשק לניהול קישורים |
| מחיקת קישור מהמאגר | הסרת URL שכבר לא בשימוש | שליפה ומחיקה מהמסד | ממשק ניהול, מסד נתונים |
| מציאת קישור תואם | חיפוש קישור לפי מזהה או תבנית | חיפוש טקסטואלי או לפי מזהה ייחודי | מסד נתונים |
| שליפת מידע על ההתחברויות | הצגת היסטוריית כניסות לכל קישור | שליפה מהמסד, סינון לפי מזהה קישור | מסד התחברויות |
| שליחת מידע מוצפן | תקשורת באמצעות AES | אלגוריתם הצפנה (כגון AES), ניהול מפתחות | ספריית קריפטוגרפיה, מפתח הצפנה |
| החלפת מפתחות | החלפת מפתחות AES באמצעות RSA | RSA, שליחת מידע | מודול הצפנה |
| שליחת מידע | שליחת מידע ללקוח. |  | עוטף לסוקט. |
| ניהול שרת HTTP | הפעלת שרת שמאזין לבקשות מהקורבן | פתיחת תת תהליך, סגירת תת תהליך |  |
| ביצוע ARP spoofing | התחזות ל-Gateway כדי ליירט חבילות (מתקפה מקומית בלבד) | פתיחת תת תהליך, סגירת תת תהליך | סקאפי |
| מימוש איש באמצע (MITM) | יירוט תקשורת בין לקוח לשרת | פתיחת תת תהליך, סגירת תת תהליך, הסנפה ושליחה של פקטות. | עוטף לסוקט, סקאפי |
| החזרת תשובת DNS פיקטיבית | שליחת תשובת DNS מזויפת כדי להפנות לשרת HTTP פנימי. | MITM | עוטף לסוקט, סקאפי |
| פתיחת תת-תהליך | הפעלת כלי עזר במקביל (למשל spoofing או sniffer) | ניהול תהליכים במערכת (multiprocessing / subprocess) | גישת מערכת |
| סגירת תת-תהליך | עצירת רכיב פועל במערכת | מעקב אחרי PID, שליחת סיגנל סיום | מזהי תהליך (PID), בקר תהליכים |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

# **רקע תיאורטי**

## **מונחים מתקדמים ברשתות**

***הבהרה: משתמשים בטכנולוגיות אלו גם (ובעיקר) בהקשרים זדוניים. בפרויקט אני משתמש בהם במטרה שאינה מטרה זדונית, והמימוש המדויק שקיים בפרויקט הוא לא מימוש שניתן להשתמש בו בהקשרים זדוניים.***

### **ARP spoofing**

שיטה לשנות בעבור מחשב מסוים את שיוך בין IP לבין MAC, ולמעשה מעניקה היכולת ׳להתחזות׳ למחשב אחר.

באופן טכני, כדי לבצע את השינוי, צריך לשלוח באופן נשנה תשובת ARP שעונה על השאלה ״מי זה המחשב עם הIP הזה״ עם התשובה ״לי״.

בפרויקט אנחנו משתמשים בשיטה זו, כדי שהמחשב של הילד יחשוב שהשרת הוא הrouter.

אנחנו שולחים מספר אקראי של פעמים בשנייה תשובת ARP כמתואר.

### **איש באמצע – MITM**

ופעה שבה צד שלישי מצליח להיכנס לאמצע של תקשורת בין שני צדדים – לדוגמה, בין מחשב של משתמש לבין אתר אינטרנט – מבלי שתידרש פעולה אקטיבית מצידם.

במצב כזה, "האיש שבאמצע" יכול לצפות במידע שעובר, ולשנות אותו אם יבחר.

בפרויקט אני משתמש ב ״איש באמצע״ בשילוב עם ARP spoofing בכדי להסתכל בבקשות בין הילד לבין הראוטר, ובמידת הצורך (בקשת DNS עם דומיין או דומיינים מסוימים) לשלוח תשובה במקום הראוטר.

המשמעות הטכנית של הדבר (והמימוש הספציפי שבו אני משתמש) היא שהמחשב של הילד שולח את התעבורה אל הIP של הראוטר, אך בגלל הARP spoofing, זה מגיע אל השרת. במקרה והבקשה לא רלוונטית השרת מעביר אותה **ללא שינוי** אל הראוטר. **הראוטר יחזיר את הבקשה ישירות אל הילד**, בניגוד לשימושים הזדוניים של איש באמצע.

במידה והפקטה היא בקשת DNS רלוונטית, השרת יחזיר אל המחשב של הילד תשובת DNS, שלכאורה הגיע מהשרת DNS. כתובת הIP שתהיה בתשובה, היא הIP של השרת.

### **הרעלת DNS**

תופעה שבה, בצורה מכוונת איש באמצע מחזיר תשובת DNS לא נכונה.

בפרויקט אני משתמש בזה, כדי לנווט לשרת HTTP שרץ על מכונת השרת במידה והילד הכניס לדפדפן URL שצריך לבצע לו redirect.

## **הטכנולוגיה הרלוונטית**

### **שפות תכנות**

בקאנד באמצעות python.

פרונטאנד באמצעות html וcss.

רנדרינג באמצעות flask בpyhton.

### **מערכות הפעלה**

שימוש בקבצי json לשמירת הנתונים הבאים בצד שרת –

נתוני כניסה לאתרים

משתמשים קיימים

מאגר קישורים

שימוש במודול multiprocessing בכדי להזניק ולכבות תתי תהליכים, בעת הצורך.

תקשורת בין תתי התהליכים (IPC) באמצעות PIPE.

שימוש במודול signal כדי לכבות תת תהליכים בצורה נקייה ולנקות משאבים.

שימוש בLOCK בגישה לקבצים.

### **תקשורת**

תקשורת בסיסית בין שרת ללקוח, באמצעות אובייקט SOCKET מעל TCP בפייתון.

תקשורת עם שרת http כדי לבצע redirect.

שליחת פקטות ARP (שׁכבה 2) בכדי לנתב את בקשות הDNS אל השרת. בעזרת scapy.

הסנפה בעזרת scapy של כל הפקטות שנשלחות, כדי לתפוס פקטות DNS שמבקשות domainים מסוימים.

שליחת פקטות DNS Response בעזרת scapy.

# **מבנה וארכיטקטורה של הפרוייקט**

## **תיאור הארכיטקטורה של המערכת**

### **תיאור החומרה**

**שרת** –

מריץ את השרת הראשי שאחראי על תקשורת עם הלקוחות, ובנוסף מריץ שלושה תתי תהליכים:

* תהליך שמבצע arp spoofing לילד
* תהליך שמממש איש באמצע (mitm), כדי לתפוס ולשנות בקשות dns מסוימות
* תהליך שמריץ שרת http, כדי לטפל בredirectים.

שלוש התהליכים מוזנקים בתחילת ריצת של התוכנית המרכזית, ויוצאים בצורה נקייה כשהתוכנית הראשית מסיימת.

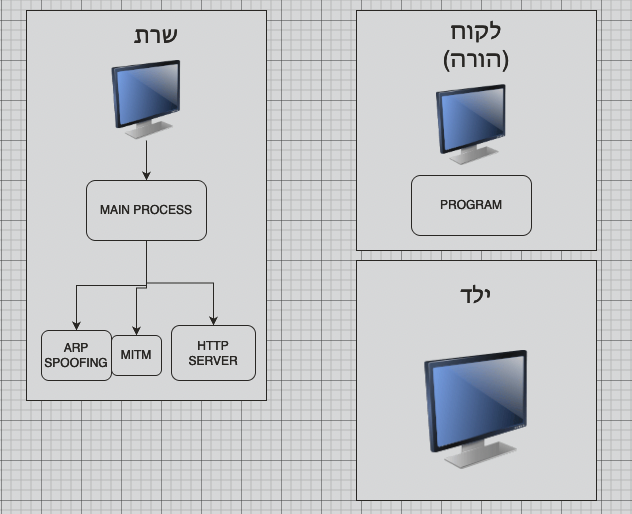
**לקוח** –

הלקוח מריץ את האפליקציה, ואז מתחבר מהדפדפן לhttp://localhost:5000. מאחורי הקלעים ממומש שרת http, שדואג לgui.

**ילד –**

גולש כרגיל בדפדפן. לא צריך לבצע שום פעולה אקטיבית.

### **החומרה בשרטוט**



## **זרימת מידע**

### **כללי**

ישנן שלוש ישויות. הלקוח (ההורה), השרת והילד. הלקוח והשרת מדברים בינהם בפרוטוקול סינכרוני.

מטרת פרוטקול זה (כפי שיורחב בהמשך) היא לתת להורה שליטה דינמית על מסד הנתונים של הקישורים.

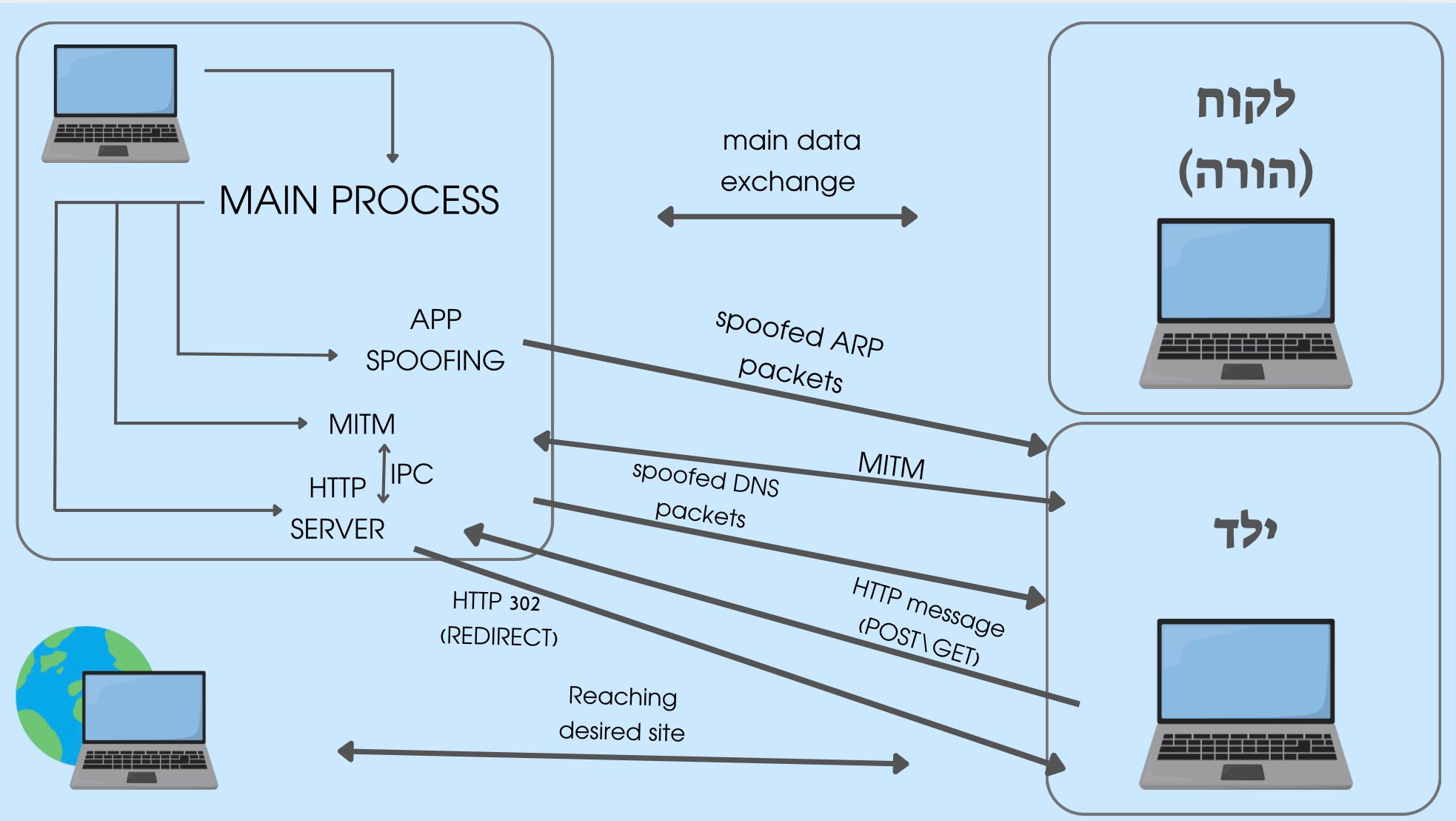
השרת מריץ ברקע תהליכים כפי שפירטתי, במטרה לשנות את התנהגות המחשב של הילד.

השרת גורם למחשב של הילד לחשוב שהשרת הוא הראוטר, ולכן כל התעבורה (ובכלל זה כלל בקשות הDNS), יעברו דרך השרת. בבקשות שאינן בקשות DNS, השרת יעביר כפי שהן אל הראוטר. בגלל המימוש הספציפי של איש באמצע שבו משתמשים (וכפי שהוסבר בחלק הרקע התיאורטי) אין צורך לטפל בהעברת התשובה של הראוטר.

כאשר הילד ירצה לגלוש אל אחד מן האתרים המנוטרים, ויזין בדפדפן את URL שלא קיים, תישלח בקשת DNS, על אותו הדומיין. הבקשה תישלח אל השרת, שישים לב שהדומיין קיים במסד הנתונים שההורה מנהל באופן דינמי, ולכן במקום להעביר לראוטר יחזיר תשובה מזויפת, עם הIP שלו. בנוסף, ינטר את הכניסה, ויכתוב אותה במסד נתונים הרלונטי, וגם יעדכן את תת התהליך של שרת הHTTP, שבוצע ניסיון התחברות לכתובת מסוימת, ולאיזו כתובת לעשות redirect.

הדפדפן ינסה לפנות אל השרת, ויגיע לשרת HTTP, שמורץ כתת תהליך מהתהליך הראשי של השרת. שרת הHTTP, ציפה לפנייה הזאת (כיוון שתת התהליך של הMITM עדכן אותו), וידע לאן המחשב של הילד ניסה להתחבר, וישלח תשובת HTTP עם קוד 302 (redirect) עם הכתובת התואמת.

תוצאת כל התהליך, היא שהילד הכניס לדפדפן איזשהו URL מזויף, והגיע לאתר אמיתי, בלי שירגיש, וכניסתו נותרה ותועדה. ההורה בוחר לאלו אתרים לייצר קישור מזויף, שלאחר מכן ייתן לילד כדי שתיחבר דרכם.

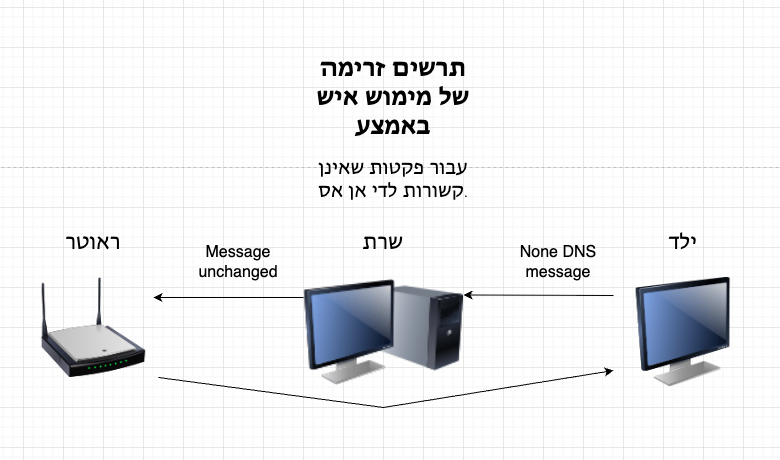


### **תת תהליך MITM + הרעלת DNS.**

פקטות רגילות (איש באמצע) –

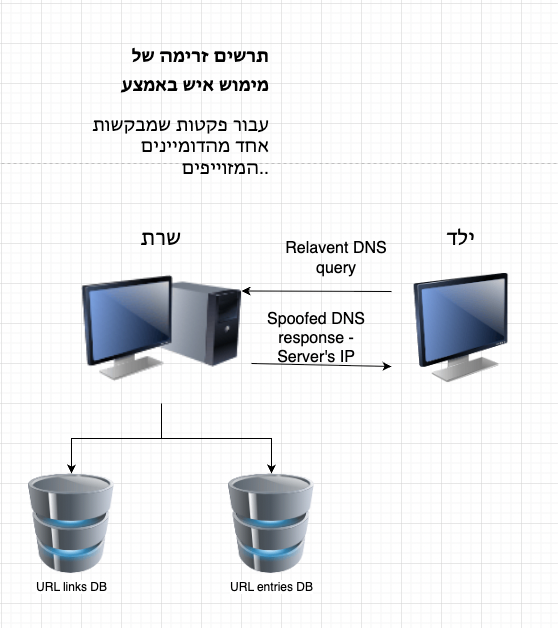
השרת יקבל את ההודעה של הילד, יבין שהיא לא רלוונטית, ויעביר אותה **ללא שינוי** אל הראוטר. הראוטר יחזיר את ההודעה ישירות אל המחשב של הילד.

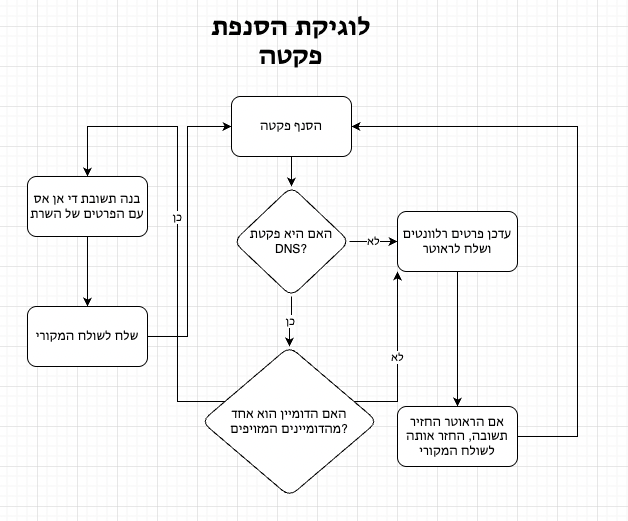
**הבהרה:** מימוש זה שונה מהמימוש הנפוץ של MITM, שבו בדר״כ גם התשובה עוברת דרך האיש באמצע. במקרה שלנו, אין צורך בלהתפרץ להודעות החוזרות מהראוטר.



פקטות DNS, שהדומיין שלהם נמצא במאגר בתור קישור מנוטר (הרעלת DNS) -

השרת יזהה שמדובר בדומיין שרשום בתור דומיין שצריך לנטר, ולכן יבנה ויחזיר תשובת DNS, עם הIP שלו, ישלח לשרת HTTP לאן לעשות redirect, ויתעד את הכניסה. במקרה כזה אין שימוש באיש באמצע.





## **אלגוריתמים מרכזיים בפרויקט**

אין

## **סביבת הפיתוח**

### **כלי הפיתוח הדרושים**

|  |  |
| --- | --- |
| תחום | כלים / טכנולוגיה שתעשה בהם שימוש |
| Frontend | * HTML * CSS 3 * Browser (firefox, chrome) |
| Backend | * Python 3 * Socket * FLASK (python) |
| DB | * SQLite * Json |
| Networking | * Scapy * ARP * DNS * HTTP |
| IDE | * Visual Studio Code |
| ניהול גרסאות | * Git |

### **סביבה וכלים הדרושים לבדיקות**

לבדוק gui – דפדפן.

לבדוק תקשורת בין שרת ללקוח – socket עובד, וterminal.

לבדוק את אפקטיביות תהליכי הרקע של השרת – מספר מחשבים באותה הרשת, וwireshark מותקן. זאת בנוסף לscapy וsocket מותקנים ומעודכנים.