**פרויקט סטנוגרפיה – מבוא לאבטחת המרחב המקוון**

מימוש כללי:

בעת המימוש, הקפדתי להשתמש כמה שיותר בפונקציות של הספרייה NumPy. הסיבה לכך, היא שהספרייה היא למעשה עטיפה של קוד C – זה כמובן רץ יותר מהר מקוד Python (הרי שפת C היא שפה Low-Level שמתקפלת לקוד מכונה, לעומת Python, שנחשבת לאחת השפות הכי High Level שמונעות בעזרת Interpreter).

ספריית Numpy מספקת פעולות מועילות מאוד על מערכים, ובפרט, בשני חלקי המטלה השתמשתי בפונקציות flatten() ו-reshape() – הן מאפשרות לשנות את האופן שבהם מסודרים האיברים:

* Flatten לוקחת מערך ופורשת אותו ״לפי שורות״ – כלומר, מהמימד הראשון לאחרון. ז״א, כאשר היא מופעלת על תמונה, היא פורשת את כל האיברים לפי שורות, לאחר מכן לפי עמודות (פיקסלים), ולבסוף לפי ערוצים (RGB Values). כלומר, תמונה שהופעלה עליה הפונקציה, נותנת לנו גישה לאיברים בסדר שנדרש בשני הסעיפים – מלמעלה למטה ומשמאל לימין. לבסוף מתקבל מערך חד מימדי עם כל האיברים כולם.
* Reshape היא למעשה הכללה של Flatten – היא מאפשרת לשנות את הShape – המימדים של array תוך שמירה על הסדר שלהם ופרישה, כאמור, מהמימד הראשון לאחרון. בעזרתה אפשר להפוך את פעולת Flatten – כדי לקבל שוב array שמתאר תמונה.

מעבר לכך, עוד זוג פונקציות שנעשה בהן שימוש הוא numpy.packbits, numpy.unpackbits. הפונקציות הללו מאפשרות, בהתאמה, לקחת ערך מספרי, ולפרק אותו לסיביות שלו, או להפך, לקחת מערך סיביות, ולאסוף אותן לכדי ערכים.

זאת ועוד, אופרטורים מתמטיים בסיסים ב-NumPy (כמו +,-,\*,/,&,|) פועלים איבר-איבר בצורה מהירה מאוד – וסייעו בעיבוד של התמונה כמערך.

חלק א׳ – הצפנה

חלק זה למעשה היה הפשוט יותר, כמובן. בתור התחלה, בניתי array עם ערכי הסיביות שצריך להכניס בסיבית התחתונה של איברי התמונה – בעזרת unpackbits על מערך שמכיל את ערכי ה-ASCII של מחרוזת הקלט.  
הפכתי את התמונה למערך חד מימדי רציף עם Flatten, לאחר מכן ביצעתי למערך את הפעולה &= 0b11111110 – כלומר, שימרתי מכל ערך בתמונה רק את 7 הסיביות העליונות. כעת, כל שנותר הוא להדליק את הסיביות שצריכות להיות דולקות (כל אלה שצריכות להיות כבויות כבר במצב זה) – ובעזרת האופרטור |= עם כל ערכי הסיביות שצריך להכניס למקומות התחתונים - למעשה קיבלנו את הנדרש.

חלק ב׳ – פענוח

חלק זה, הוא, כמובן, המסובך יותר.

ראשית, היות ולא ידוע האינדקס שבו מתחילה ההודעה, יש להתחיל לחפש ברצף תווים ב״בלוקים״ של 8 בכל אחד מהאינדקסים 0,1,…,7 בלבד – שהרי כבר באינדקס השמיני נחזור על עצמנו (הרי ברור 8%8=0 ואנו מבצעים ״איסוף״ של בתים מתוך רצפים של סיביות – וכל בית מכיל 8 סיביות).

זוהי משמעות הלולאה הראשית של הפונקציה decode. בתוכה, מבוצעות שתי פעולות:

1. איסוף הסיביות והמרה למחרוזות:  
   עבור כל מיקום סיבית שבה יכולים להיות התווים שמכילים את ההודעה (נתון כי ההודעה עשויה להימצא בכל אחת מ-3 הסיביות הנמוכות!), אנו מחלצים את הסיביות במיקום הרלוונטי עבור כל התמונה, ולמעשה לוקחים מתוך הסיביות הללו רצף שלאחר מכן נהפך, בעזרת packbits, reshape ו-decode מ-bytes, למחרוזת ארוכה שכוללת את כל התווים שעשויים להיות תווים שהוחבאו (בכוונה – או תווים שאינם באמת מועילים ונמצאו מקרית), החל מה-offset הרלוונטי (שהוזכר לעיל – בלולאת החיפוש הראשית). את המחרוזות הללו שבונים עבור כל מיקום סיבית מחזירים, והן מועברות לשלב 2.
2. חיפוש ההודעה במחרוזות:

למעשה, החיפוש אינו מסובך מאוד, אך כן כולל מורכבות מסוימת. החיפוש עצמו מורכב מלולאה שרצה על כל האינדקסים במחרוזת הקלט, ומפעילה פונקציה רקורסיבית שבודקת האם ההודעה הנדרשת נמצאת ומתחילה באינדקס הנבדק.  
הפונקציה הרקורסיבית למעשה בונה את ההודעה תו-תו עם ההתקדמות מטה בעץ הקריאות, ותוך כדי נעזרת ביוריסטיקות על מנת להימנע מקריאות מיותרות שיגדילו את זמן הריצה ומסננות את הקלט. הפונקציה מקבלת כקלט 3 פרמטרים:

1. המחרוזות משלב 1 (אחת לכל סיבית – 3 מחרוזות בדיוק במטלה).
2. ההודעה שנבנתה עד כה.
3. האינדקס של המחרוזת, מתוך הפרמטר הראשון, שבה נעשה שימוש בקריאה הקודמת.

בכל קריאה, הפונקציה מנסה להאריך את ההודעה באמצעות התו הבא מכל מחרוזת קלט. היא בודקת שההוספה תקינה בעזרת מספר בדיקות שמונעות קריאות מיותרות:

1. בדיקה שהמחרוזת אותה מנסים לא ריקה – תנאי עצירה, למעשה.
2. בדיקה שהתו שצריך להתווסף הוא תו מותר – [A-Za-z0-9.?!\s] במקרה שלנו.
3. בדיקה שמוודאת שמילים מופרדות עם רווחים. נתון בשאלה, כי כל מילה (שעשויה לכלול או להיות גם רווחים), נמצאת ברצף באותה מחרוזת. אנו מסתמכים על כך – כמותר במטלה.

אם כך, נעשית קריאה רקורסיבית עם ההודעה הנוכחית בתוספת התו הבודד – ואם הקריאה החזירה ערך, ז״א שמצאנו הודעה. אם לא נמצאה כלל הודעה, אנו בודקים את הודעת הקלט – בלי התו הנוסף, ואם היא תקינה – מחזירים אותה. כלומר, הפונקציה שואפת להחזיר הודעה ארוכה ביותר.

בדיקת תקינות ההודעה – על מנת לבדוק את תקינותה של הודעה מסוימת שנמצאה, צריכים להתקיים זוג תנאים:

1. מספר המילים חייב להיות גדול מסף מינימלי. נתון במטלה כי הקלט מכיל משפט עם למעלה מ-20 מילים, ולכן – זה הסף המוגדר.
2. חלק מסוים (שהוגדר כ-50%) מהמילים של ההודעה שנמצאה, חייב להתקיים גם במילון (לאחר נרמול והסרת סמלים כמו ?!.,). אם אין מספיק מילים כאלה, ההודעה נחשבת לא תקינה.

ההנחה היא שנמצאת הודעה אחת ויחידה שעונה לדרישות (מבחינה סטטיסטית זה הגיוני במצב ובו הוחבאה רק הודעה אחת) – אך כמובן ניתן להפוך בקלות את הקוד לקוד שתומך בריבוי הודעות (החזרת רשימות של הודעות תקינות במקום מחרוזת בודדת).

הרצת הפענוח על הדוגמה הניב את הפלט:

“What would you do if you had one hour to solve this task? Brute force? Random search? Well I would choose...”

הרצה של תוכנית ההצפנה עם מחרוזת ארוכה הניבה את תמונת המנעול המצורפת, והפעלה של תוכנית הפענוח על תמונת הפלט – החזירה בדיוק את ההודעה המוצפנת:

“Hello, World! My Name is aviv and I am going to do something very simple today. I am going to write a long message and I hope my code will be able to find it quickly and safely.”

כנדרש.

הרצה על מספר תמונות המצורפות כאן (מלבד המנעול – שעבר הצפנה) בנוסף הניב הדפסה של שגיאה – כולן תמונות אקראיות מהאינטרנט, שלא היה צריך להימצא בהם מסר כלל – ואכן לא נמצא.

A picture containing person

Description automatically generatedA picture containing bubble

Description automatically generated