# פרויקט גמר – סייבר ב'

* Implement Secure Code Scanner.
* Run Popular Static Coed Scanning tools ()
* Create a report on a server per project
* Each project will have a report
* Each report will show for each case what to fix
* Use LLM to recommend code fixes (such as. ChatGPT)

## GUI

* Use Streamlit (or NodeJS or CSS)

General views:

Project 1

Project 2

Project 3

…

|  |  |  |
| --- | --- | --- |
| fix | Bug | Tool |
| Fix suggested by LLM | Detected bug + Code | Detecting tool |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

מבנה העבודה:

1. מימוש כלים לניתוח סטטי, מתחילים מ-2. (Bandit & Pylint)

כל כלי לניתוח סטטי יממש את הממשק interface.StaticCodeAnylazer כך שיהיה ניתן לאגד את הכלים השונים.

\*הממשק נמחק מהעבודה שכן לא היה בו כלל שימוש, אבל העיקרון עומד:

לכל מחלקה של כלי לניתוח סטטי יהיו אותם 3 שיטות:

Scan\_file, get\_name, split\_output

1. מימוש UI בסיסי
2. לוודא שהפלט מכל כלי לניתוח קוד סטטי תקין ועובד.
3. יצירת קישור ל-LLM (chat.gpt) ע"י יצירת ./LLM/chat.py אשר מקבל כקלט שם כלי לניתוח קוד סטטי ומחזיר תגובה לתיקון.

יש לשמור את התגובה לתיקון בקבצים ./LLM/Project\_name/Tool\_name

בשביל לשמור על שמות הקבצים ויכולת לזהותם. חשבנו על 2 אפשרויות:

1. שימוש במסד נתונים אשר מקדיש לכל פרויקט X וקובץ Y תעודת זהות ID כך ששם הקובץ המכיל את תוצאות הסריקה יהיה ID.txt
2. הפתרון השני ששם הקובץ המכיל את תוצאות הסריקה יהיה X-Y.txt. (לאחר החלפת כל התווים הלא חוקיים בשמות קבצים כגון '/')

הפתרון השני יותר פשוט ולכן נשתמש בו לבינתיים.

1. ניצור מערכת הלוקחת את כל הקבצים בתיקייה הראשית של הפרויקט ומוסיפה את כל הקבצים לפרויקט.

(יש לשים לב, כרגע המערכת סורקת רק קבצים עם סיומת .py)

1. הוספת פונקציונליות סריקת הקבצים דרך האתר.
2. הוספת דרך להצגת תוצאות הסריקה לכל קובץ.
3. לממש ignore

תהליך העבודה

* שלב 1:

מימשנו 2 כלים לניתוח קוד סטטי, לבדיקה כך שהדרך שבה אנו מממשים אותם יכולה להתרחב.

* שלב 2:

מימוש UI בסיסי המקבל שם פרויקט, ותיקיית פרויקט ומוסיף את כל קבצי הפייתון תחת הפרויקט

בשלב זה הוחלט כי נתעסק בסריקה של קבצי פייתון בלבד, לפחות עד מועד ההגשה, שכן הוספת כלים לסריקת שפות אחרות (במידה וקיימים, לא הוקדש זמן לחיפוש) יגרור דרישה לבחירת כלים לניתוח קוד עבור כל קובץ, מה שאנו ראינו כמיותר לזמן זה ולא מקדם את מטרת הפרויקט.

* שלב 3:

נעשה את שמירת הפרויקטים באמצעות Json פשוט, שכן כל מה שאנו צריכים זה שמירת שם הפרויקט (שמשמש גם כ ת.ז. הפרויקט), מספר מאפיינים ורשימת הקבצים. הסיבה לשימוש Json היא רשימת הקבצים הארוכה, שכן ייווצר לנו מסד נתונים מנופח.

ראינו שימוש בsql כמיותר **לשלב זה**.

* שלב 4:

הוספת קבצים ידנית לפרויקט תעשה על ידי הכנסת Path אבסולוטי או Path רלטיבי לתיקיית המקור של הפרויקט.

בסוף החלטנו, לפישוט הפרויקט, לא לקבל Path אבסולוטי, אלא ליצור רק פרויקטים בעלי תיקיית מקור ולהוסיף רק מסלולים רלטיביים מתיקייה זו.

* שלב 5:

הוספת UI לסריקת הפרויקט ומימוש סריקת הפרויקט. השיטות בLLM/chat.py (שם יתעדכן בהמשך) אחריות על סריקות הקבצים.

החלטנו שסריקת קובץ תישמר לפי הכלי לניתוח הקוד הסטטי, באופן הבא:

עבור סריקת כלי Bandit לקובץ app.py בתיקייה C:/temp, הסריקת תישמר בקובץ:

./LLM/Bandit/C-temp-apppy.txt

\*נשתמש בתרגום מסלול חד-חד ערכי (chat.get\_normalized\_path) כך שסריקת קבצים תהיה מקושרת בפשטות למיקום הקובץ, ובונוס נוסף משמירה זו, היא יכולת שיתוף הסריקות בין פרויקטים, במידה והם חולקים קבצים.

לאחר שמירת הסריקות, נפרמט ונסכם אותם בתיקייה scans ואת התיקונים של chatgpt נשמור בתיקייה fixes.

לאחר תקלות בעניין שמירת הקבצים, בשל escape characters או ", : שמנעו שמירה בJSON, החלטנו לשמור את הפתרונות בבסיס 64.

הסיבה שלא עשינו זאת עבור סיכום הסריקות, היא שהתווים בעברית בבסיס 64 לא מתורגמים בחזרה לעברית. לכן, שם בחרנו לתרגם את התווים הפוגעים כמו \ ל- / . מעבר לכך, לא מצאנו בעיות.

בעתיד, כמובן שנרצה להרחיב את התוכנה.

נעשה זאת ע"י הוספה פשוטה של כלי לניתוח סטטי בקובץ analyzers.py שמממש את כלל השיטות הדרושות: get\_name(), scan\_file(), split\_output().

רישום במערך STA והוספת תיקייה לשמירת הסריקות במערך Out\_paths.

**תקלות שנתקלנו בהם במהלך הדרך:**

* כאשר יוצרים \ עורכים פרויקט, לפעמים ישנן תיקיות כמו /venv או קבצים בשם \_\_pycache\_\_ שאנו לא מעוניינים לסרוק, והם מציפים לנו את הפרויקט.

לכן, הוספנו דרך להכניס התעלמות מקבצים אלו כאשר מוסיפים קבצים אוטומטית לפרויקט.

כניסות אשר מתחילות עם /, למשל /venv או /venv/lib ימנעו מכל הקבצים הנמצאים בתיקייה /<main\_project\_directory>/venv/lib להיכנס לפרויקט

כניסות שלא מתחילות עם /, למשל pycache, ימנעו מכל קובץ אשר המסלול אליו מכיל pycache להיכנס לפרויקט.

למשל, /<main\_project\_directory>/pycache/lib או /<main\_project\_directory>/\_\_pycache\_\_ או /<main\_project\_directory>/dir/\_\_pycache\_\_. לא יכנסו לפרויקט.

* הגבלת מספר הקבצים שניתן להוסיף אוטומטית לפרויקט ל-500, שכן פרויקט בסדר גודל שאנו רוצים לעבוד איתו לא יעבור את מספר זה, והוא מונע הצפה של קובץ השמירות אם במקרה מישהו לא איית נכון את שם התיקייה להתעלם ממנה, וכו'.

כמובן שזו מגבלה שניתן לשנות, אם נחליט להגדיל את הפרויקט

* נוצרה גם בעיה של מחיקת פרויקט ללא מחיקת הסריקות, מה שמוביל לבזבוז מיותר של זיכרון. לכן, הוספנו מספר אפשרויות למחיקת סריקות – אחת, בפתיחת הפרויקט,

השנייה, במחיקת הפרויקט והשלישית, מחיקת כל הסריקות במסך הבית

* בעיה נוספת שעלתה היא פתיחת הקבצים של הסריקות. בגלל שמספר בקשות יכולות להישלח, נוצר לנו מצב בו הקובץ נפתח פעמיים במקביל ותוכנו נמחק. לכן, הוספנו מנעול.