בס"ד

Project Structure

בהפעלת המוניטור יוצג למשתמש UI בו יהיה תפריט בחירה בין שני מצבים אפשריים:

1. *הפעלת המוניטור לדגימה אוטומטית – במצב זה יהיה על המשתמש לבחור את פרק הזמן שהמוניטור ימתין בין דגימה לדגימה נוספת. משם המוניטור יחל בדגימותיו ובתיעוד לקבצים כנדרש.*
2. *מצב ידני – במצב זה יהיה על המשתמש להזין שני תאריכים של שתי דגימות שונות (יכניס קודם את התאריך המוקדם יותר ואז את התאריך המאוחר יותר). המוניטור יציג למשתמש את התהליכים החדשים שהחלו לרוץ בין הדגימות ואת התהליכים שהפסיקו לרוץ בין הדגימות.*
3. *יציאה – מסיים את התכנית.*

הסבר על מבנה מחלקות הפרויקט ותפקידיהן

ProcessMonitor

המחלקה הראשית. למחלקה יש שתי פונקציות עיקריות:

- **start_monitor** - תחום אחריותה הוא לתפעל את המוניטור, לתעד לקבצים את מצב המוניטור ולאבטח אותם. ע"מ לבצע משימות אלו, המחלקה משתמשת בספריות ובמחלקות עזר נוספות.
- **samples_diff** – מצב ידני. מאפשרת למשתמש להזין כקלט 2 תאריכים שונים, ותציג למשתמש כפלט את התהליכים החדשים שהתחילו לרוץ בין הדגימות ואת התהליכים שהפסיקו לרוץ בין הדגימות.

המחלקות והספריות שהמחלקה משתמשת בהן:

datetime
time
thread
psutil
ProcessList
StatusLog
FilesHandler
PausingObserver

ProcessList

המחלקה אחראית לטפל בקובץ "processList.csv". למחלקה יש שתי פונקציות עיקריות:

- **write_process_list** - תתעד לקובץ את כל התהליכים שנדגמו.
- **get_sample_by_date** – מקבלת כקלט תאריך מסויים ומחזירה כפלט רשימה של כל התהליכים שנדגמו באותו תאריך.

הספריות שהמחלקה משתמשת בהן:

os
datetime
csv
psutil
stat

בס"ד

StatusLog

המחלקה מיועדת לטיפול בקובץ "Status_Log.txt". למחלקה יש שלוש פונקציות עיקריות:

- **get_new_running_process** – מקבלת כקלט שתי דגימות, ומחזירה כפלט את רשימת התהליכים שהתחילו לרוץ בין הדגימה הראשונה לשנייה.
- **get_killed_process** - מקבלת כקלט שתי דגימות, ומחזירה כפלט את רשימת התהליכים שהפסיקו לרוץ בין הדגימה הראשונה לשנייה.
- **write_status_log** - המחלקה תתעד לקובץ עבור כל דגימה אילו תהליכים חדשים התחילו לרוץ ואילו תהליכים הפסיקו לרוץ מאז הדגימה האחרונה.

הספריות שהמחלקה משתמשת בהן:

os
datetime
psutil
stat

PausingObserver

המחלקה יורשת מהמחלקה watchdog.observers.Observer ומטרתה לאפשר למוניטור לבצע שינויים כ"מורשה". הכוונה היא שכאשר המוניטור יבצע שינויים בקבצי התיעוד שהוא מייצר המשתמש לא יקבל על כך דיווח, מפני ששינויים אלו לגיטימיים.

ע"מ לאפשר פעילות זו הצהרתי על משתנה בוליאני גלובלי PAUSED שיהווה דגל – כאשר ערכו יהיה False אנו לא נתעלם משינויים בקבצים, מפני שאלו יהיו שינויים שהמוניטור עצמו מבצע. כאשר ערכו יהיה True אנו הwatchdog ידווח על שינוי שאינו לגיטימי.

למחלקה שתי פונקציות עיקריות:

pause – משהה את פעילות הwatchdog לצורכי שינויים של המוניטור בקבצי התיעוד.

resume – ממשיך את פעילות הwatchdog לצורכי אבטחת קבצי התיעוד מפעולות לא מורשות.

הספריות שהמחלקה משתמשת בהן:

time
watchdog

FilesHandler

המחלקה יורשת מהמחלקה PatternMatchingEventHandler ששייכת לספריה watchdog ותגדיר את הפעולות שיש לבצע בעת שמתבצעים שינויים לא סבירים באחד מקבצי התיעוד שהמוניטור מייצר.

המחלקה מממשת את כל הפעולות שיש לבצע עבור כל אחד מארבעת האירועים שקורים לקבצי התיעוד: שינוי, יצירה, מחיקה והעברה.

המחלקות והספריות שהמחלקה משתמשת בהן:

time
PausingObserver
watchdog

הסבר על ספריות מיוחדות שהיו בשימוש:

Psutil

ספריה זו איפשרה לי לבצע דגימות של תהליכים שרצים. הספריה הייתה הכרחית למימוש של המוניטור. בתוך הספרייה יש מחלקה בשם Process ובתוכה מאפיינים חשובים כמו pid, name ועוד. במימוש המוניטור נעזרתי בשתי פונקציות שהספריה מספקת. האחת היא process_iter() שמחזירה אוסף של אובייקטים מסוג Process שמייצגים את כל התהליכים שרצים כרגע. השנייה היא pids() שמחזירה אוסף של כל ה-pid של התהליכים שרצים כרגע.

ספריה זו תומכת גם ב-windows וגם ב-linux.

Watchdog

ספריה זו סיפקה לי שירות חשוב בהגנה על קבצי התיעוד שהמוניטור כותב אליהם. השירות שהיא מספקת הוא שירות watch service. ספריה זו מאפשרת לי לדווח על מקרים של שינויים לא רצויים בקבצים, מחיקתם, העברתם או יצירתם. דאגתי לתפעל את הספריה כך שלא יהיו למשתמש הודעות שווא על שינויים שהמוניטור עצמו ביצע. כך וידאתי שלא התבצע שינוי ע"י גורם חיצוני....

ספריה זו תומכת גם ב-windows וגם ב-linux.

Stat

בעזרת הקבועים של המחלקה S_IREAD, S_IRGRP, S_IROTH, S_IWUSR יכולתי לשנות את מאפייני הקובצים ולהפוך אותם ל-read only בין דגימה לדגימה. יש לציין ששינוי המאפיינים הוא מחסום קל למי שירצה לחבל בקבצים, אך גם זה עוזר לשמור על הקבצים (לדוגמא, לא יתבצעו בטעות שינויים לא רצויים, או כאשר הפורץ לא הספיק לסיים לערות את הקובץ והוא חזר להיות read only).

ספריה זו תומכת גם ב-windows וגם ב-linux.

Csv

הספרייה מספקת למוניטור את הפונקציונליות הדרושה על מנת לעבוד עם קבצי csv. השתמשתי במחלקה זו מכיוון שכתיבת הקובץ processList.csv תהיה יעילה יותר ופשוטה יותר בשלב שבו המשתמש ירצה להפעיל את המוניטור במצב "ידני" (השוואה בין שתי דגימות מתאריכים שונים).

ספריה זו תומכת גם ב-windows וגם ב-linux.

בס"ד

שאר הספריות הן סטנדרטיות ותומכות גם הן ב-windows וב-linux (ביצעתי נסיון הרצה בשתי מערכות ההפעלה).

הסבר על אופן ההגנה על הקבצים:

הקבצים שהמוניטור מייצר יהיו מוגנים על ידי ההגבלה read only ודיווח למסך ה-console אודות שינוי, מחיקה, יצירה או העברה של הקבצים.

דרכים נוספות שחשבתי עליהן ע"מ להגן על הקבצים, אולם לא מצאתי דרך לממש אותן cross platform (או בכלל) בהתחשב לזמן שעמד ברשותי ביחס למורכבות שבמימוש ההגנות הללו:

- סיסמאות על הקבצים (לא מצאתי דרך פשוטה לעשות זאת בפייתון).
- העלאת הקבצים לאחר כל עידכון לגיבוי בענן עם שם משתמש וסיסמה (גיטהאב זו אפשרות טובה כי ניתן לעיין בגירסאות קודמות).
- הצפנת הקבצים. אך צריך לאפשר למורשים צפייה בקבצים... לכן צריך לממש ממשק משתמש שבהזנת סיסמה יוצגו על המסך הקבצים לאחר פיענוח (או פיענוח הקבצים לזמן מוגבל והצפנתם מחדש).
- להגדיר את הקבצים להיות במצב hidden.

--

דביר ברזילי