**מטלת מוניטור**

תמונה שמכילה טקסט

התיאור נוצר באופן אוטומטי



הפרויקט מורכב מ-3 חלקים עיקריים:

חלק א- מצב מוניטור שמראה על השירותים הרצים במחשב בכל רגע נתון. המוניטור מראה גם על השירותים הרצים ברגע הנתון וגם מראה את השינויים שחלו בכל נקודת זמן ומייצא נתונים אלו לתוך קבצים שיצרנו.

חלק ב- מצב מוניטור ידני שמאפשר להגדיר 2 נק' זמן שונות ולקבל את כל השינויים שחלו בשירותים בין נק' זמן אלו.

חלק ג- הצפנה של מחלקות הקוד שלנו מפני תוקפים.  
ובנוסף, יצירת מעין 'כלב שמירה' שמתריע בפנינו כאשר נעשו שינויים לא מאומתים בקבצים שיצרנו.

חילקנו את הפרויקט ל-4 מחלקות:

Monitor.py- המחל' אחראית בעיקר ליישם את חלק א(עיין לעיל).

השתמשנו בפונ' מובנות ל- windows ו-linux כדי לקבל מידע על כל ה-services שרצים ברקע והכנסנו את הרשימה לקובץ serviceList, לאחר מכן עשינו בכל פעם השוואה בין נק' זמן סמוכות ע"י dict כדי לראות אילו שינויים נעשו ביניהם ובכל פעם שמצאנו שינוי כזה הוספנו אותו לקובץ status\_Log

manualMonitor.py- המחל' אחראית בעיקר ליישם את חלק ב (עיין לעיל).

פתחנו את הקבצים שיצרנו במחל' monitor ונעזרנו במידע שלהם על מנת ליצור מוניטור ידני. יצרנו קובץ עזר חדש בשם date.txt שבעצם שומר את כל התאריכים שמופיעים בקובץ serviceList ובנוסף מציין באיזו שורה מוזכרים שם כך שאנחנו ניגשים אל התאריכים שהמשתמש בחר ומשווים בין המיקומים שלהם בקובץ serviceList כך שנוכל לדעת על כל שינוי שנעשה ביניהם, בדרך העברנו ל-dict את ה-services בשני הזמנים וכך השוונו ביניהם.

Gui.py- המחל' כשמה כן היא, אחראית על עיצוב הממשק הגרפי מול המשתמש. קישרנו אותה אל כל חלקי הפרויקט כך שהיא מקושרת הן לחלק מצב המוניטור הרגיל, הן למוניטור שמנוהל ידנית והן לחלק ההגנתי ולהתרעות התקיפה שקופצות בחלונית הממשק.

secureSystem.py- אחראי על החלק הקשור בהצפנת קבצי הקוד מול תוקף חיצוני. היחיד שיוכל לגשת אליהם בצורה שאינה מוצפנת הוא זה שיחזיק את המפתח אליהם ובכך נמנעת מהתוקף הגישה.

\*הגשנו את זה כך כדי שלא יהיו תקלות, בעיקרון הקוד אמור להיות מוצפן מההתחלה ועל מנת לקרוא יהיה צריך להפעיל את המחלקה ולהכניס את הסיסמא.  
הוצאנו את המפתח לקובץ ברגע שהמחלקה תופעל והקוד יוצפן ועל מנת לבטל את ההצפנה יש להכניס 1 (או כל מספר אחר שנרצה להגדיר בתכנית) בטרמינל והמחלקה תטען את המפתח מהקובץ ותשחרר את ההצפנה.

ספריות שנעשה בהן שימוש

import platform  
import os  
import time  
import psutil  
from datetime import datetime  
import hashlib

import tkinter

import threading  
from tkcalendar import Calendar, DateEntry  
from cryptography.fernet import Fernet

מטרות לספריות:

Platform- מלמדת אותנו על מערכת ההפעלה דרכה התכנית רצה.

Os- עוזרת לנו לקבל את כל הנתונים על השירותים הרצים במערכת ההפעלה במידה וזה linux

Time- עוזרת לנו להפעיל מנגנון השהייה בין הפעלת מוניטור מפעם לפעם ברווחי זמן שמוגדרים ע"י המשתמש

Psutil- במידה ומערכת ההפעלה היא WINDOWS מאפשרת לעבור ע"י איטרטור על כל השירותים ולראות את המידע עליהם

Datetime- מאפשר לנו לקבל את הזמן והתאריך העכשווים

Hashlib- מאפשר לנו לקבל מעין מס' commit על הקובץ כך שנוכל להשתמש בזה כדי לדעת האם נעשה בו שינוי

Tkinter- מאפשרת לנו את כל הממשק הגרפי ע"י שימוש בפונ' הרבות שיש בספריה זו כדי להציג בצורה גרפית

Threading- מאפשר לנו לרוץ על כמה תהליכים במקביל כך שבכל רגע נתון לדוג' יהיה עדכון שניתן לנו גם על קבלת מידע חדש להכניס לגואי וגם על קבלת התרעות על התקפות

Tkcalendar- מאפשר לנו להשתמש בלוח השנה בשביל לנעוץ בשתי נק' זמן שונות ולקבל את השינויים ביניהם בשירותים שרצים.

Cryptograpy- מאפשר לנו להצפין ולפענח בחזרה קבצים.

מחלקות ומבני נתונים שהשתמשנו בהם:

השתמשנו במחל' שציינו למעלה

Monitor.py

manualMonitor.py

Gui.py

secureSystem.py

במחל' המוניטור ובידני נעזרנו במבנ"ת dict כדי לשמור את השירותים והסטטוס העכשוי שלהם במהלך ההעברה והקריאה מהקבצים.

**דרכי התגוננות מפני האקרים:**

1. הצפנו את הקוד ע"י המחלקה secureSystem.py כפי שהוסבר למעלה.

בכך תוקף מבחוץ לא יוכל לקרוא את הקוד ובכך לא יוכל לשנות/להוסיף לקוד, וכן לא יוכל ללמוד על התנהגות הקוד ולהתחקות אחריו.

1. יצרנו התרעות שקופצות למסך כאשר נעשה שינוי לא על ידינו בקבצים שיצרנו, וכך המשתמש יידע שהתבצעה פריצה ויוכל להתנהג בהתאם, ולדעת שהמידע שיש בקבצים לא אמין.

**אז איך בעצם מריצים את הקוד?**

1. המשתמש צריך לשמור את ארבעת קבצי הפייתון שבתכנית.

2. מריצים את הקובץ gui.py.

3. לוחצים על כפתור "monitor" בחלונית שנפתחת ומגדירים שם את הזמן שיעבור בין כל בדיקה של ה-services ולוחצים על "ok".

4. לאחר שהמוניטור מחזיר את המידע על ה-services ועל השינויים שנעשו בהם ניתן גם לעבור למצב ידני ובשביל זה צריך ללחוץ על כפתור "monitor" ולהגדיר זמן התחלה וסוף לבדיקת השינויים ביניהם ולהקיש על "ok" לאישור.

5. בנוגע לשימוש בשכבת ההגנה שלנו עליו להפעיל את המחל' "secureSystem.py" ועיין ערך ההסבר לעיל.

תוספת הגואי והסברים על התכנית דרך צילומי מסך ממנו:

נדגים תחילה את התכנית שלנו דרך צילומי מסך מהריצה שלה על ה-windows:

תמונה שמכילה טקסט, מסך, צילום מסך

התיאור נוצר באופן אוטומטי

קובץ date.txt בו נעזרנו כדי לשמר את המקומות בהם מופיעים תאריכים בקובץ serviceList.log ואת השורה בה הם מופיעים

תמונה שמכילה טקסט

התיאור נוצר באופן אוטומטי

צילום מסך של המוניטור בפייתון

תמונה שמכילה טקסט

התיאור נוצר באופן אוטומטי

צילום של serviceList.log מהשעה 19:15:33

תמונה שמכילה טקסט

התיאור נוצר באופן אוטומטי

צילום של serviceList.log מהשעה 19:15:44

תמונה שמכילה שולחן

התיאור נוצר באופן אוטומטי

צילום של המוניטור הידני שבו מופיעים השינויים ב-WerSvc, wlidsvc services בין השעות שצילמנו למעלה.

נדגים את אותו תהליך בלינוקס:

תמונה שמכילה טקסט, צג, אלקטרוניקה, צילום מסך

התיאור נוצר באופן אוטומטי

צילום של ה-services שרצים בלינוקס

תמונה שמכילה טקסט, צג, צילום מסך, אלקטרוניקה

התיאור נוצר באופן אוטומטי

עצרנו service קיים בשם apparmor דרך הטרמינל ולאחר מכן עצרנו אותו וקיבלנו תיעוד תחילה של כל ה-service שרצו בעת הפעלת התכנית ולאחר מכן של השינויים שנעשו מאז שזה בעצם השינוי שהפעלנו דרך הטרמינל- תחילה stopped apparmor ולאחר מכן running apparmor.

תמונה שמכילה טקסט, צג, אלקטרוניקה, שחור

התיאור נוצר באופן אוטומטי

תיעוד של ה-services שרצו בשעה 09:24:30 וביניהםapparmor .

תמונה שמכילה טקסט, לוחית

התיאור נוצר באופן אוטומטי

תיעוד של ה-services שרצו בשעה 09:24:44 ושם לא מופיע apparmor (כיוון שעצרנו אותו דרך הטרמינל)

תמונה שמכילה טקסט, מסך, אלקטרוניקה

התיאור נוצר באופן אוטומטי

תיעוד של ה-services שרצו בשעה 09:24:58 ושם מופיע apparmor (אחרי שהחזרנו אותו לפעולה דרך הטרמינל)

תמונה שמכילה טקסט, אלקטרוניקה, צילום מסך

התיאור נוצר באופן אוטומטי

המסך של המוניטור הידני שמראה על השינויים שנעשו בין השעות 9:24:30 לבין 9:24:44 ומצביע על עצירת apparmor.

תמונה שמכילה טקסט

התיאור נוצר באופן אוטומטי

המסך של המוניטור הידני שמראה על השינויים שנעשו בין השעות 9:24:44 לבין 9:24:58 ומצביע על חזרת apparmor לפעולה.

תמונה שמכילה טקסט

התיאור נוצר באופן אוטומטי

החלונית שקופצת להזהיר את המשתמש מפני שינויים שנעשו בקבצים.