Antywirus – dokumentacja

Informacje wstępne na temat kodu:

GUI zostało napisane przy użyciu biblioteki ‘curses’, co przyznaję, nie było najlepszym wyborem. W ostateczności kodu w module interface.py trzeba było napisać znacznie więcej niż ma to sens przy użyciu tej biblioteki. Dołożyłem jednak starań, aby w dużej perspektywie kod był jasno podzielony.

Dostarczone pliki z sygnaturami zostały okrojone do jedynie kilku sygnatur, gdyż wykorzystywanie całej bazy sygnatur znacząco wydłuża proces skanowania, co utrudnia testowanie aplikacji.

Moduł SigBase:

W tym module znajduje się implementacja bazy sygnatur. Realizacja sygnatur typu body wykorzystuje RegExy z modułu re.

Zawiera on klasy:

BodySignature(name, sig):

name: Nazwa wirusa ktrórego wykrywa sygnatura

sig: string zawierający sygnaturę, określoną tak jak w zewnętrznym pliku sygnatur

Reprezentacja sygnatury typu body: określa pewne ustawienie bajtów zawartych w danym pliku. Wykrywanie za jej pomocą wirusa polega na przeszukaniu pliku pod kątem określonego wzorca.

Główne metody:

match(hexFile): szuka sygnatury w podanej reprezentacji heksadecymalnej pliku

HashSignature(name, sig, fileSzie):

name: Nazwa wirusa ktrórego wykrywa sygnatura

sig: string zawierający hash wykrywanego wirusa

fileSize: rozmiar wykrywanego wirusa

Reprezentacja sygnatury typu hash: Sprawdza czy cały plik jest jednym z zarejestrowanych wirusów. Porównuje rozmiary plików, a jeśli te się zgadzają, to także hash

Główne metody:

getSize(): zwraca oczekiwany przez tę sygnaturę rozmiar pliku

match(fileHash): porównuje podany hash z tym zawartym w sygnaturze

SigBase(bodyPath, hashPath):

Argumenty tej klasy są ścieżkami do plików zawierających sygnatury.

Klasa przechowuje obiekty z sygnaturami i implementuje metody służące do wykrywania tych sygnatur w plikach.

Główna metoda tej klasy:

scanFile(fileHash, fileSize, hexFile, callback):

skanuje plik określony parametrami fileHash, fileSize i hexFile w poszukiwaniu wirusów zawartych w bazie. Zwracany rezultat jest w formacie:

((Czy plik jest zainfekowany?, dalsze informacje), Czy da się naprawić plik?)

Dalsze informacje są zależne od tego, jaki typ sygnatury zostanie odnaleziony.

Jeśli parametr callback nie jest None, to oczekiwane jest że jest on funkcją, służącą do przekazywania informacji o zaawansowaniu procesu skanowania, oraz do pobierania informacji na temat potencjalnego przerwania skanu.

Moduł FileManager:

Ten moduł implementuje skaner plików, oraz interfejs komunikacji z nim. Skany realizowane są następująco: najpierw rekurencyjnie znajdowane są wszystkie pliki zawarte w podanym katalogu. Następnie przy pomocy obiektu SigBase sprawdzany jest każdy z plików, a rezultat skanu przchowywany jest w obiekcie ScanReport.

Zawiera on klasy:

Scanner(bodySigPath, hashSigPath, callbacks):

Argumentami są odpowiednie ścieżki do plików z sygnaturami, oraz callbacks, zawierający dwie funkcje służące do raportowania postępu skanu, oraz do pobierania informacji o ewnetualnym przerwaniu skanowania

Instancja tej klasy służy do skanowania plików oraz do przechowywania i zapisywania indeksu plików

Główne metody:

simpleScan(toScan, fast): realizuje skan podanego pliku, szybki jeśli fast jest prawdziwe. Uaktualnia także index plików.

cutOut(fixableInfo): realizuje wycięcie wirusa z pliku. Argumentetem jest element z listy plików naprawialnych udostępnianych przez obiekt ScanReport.

ScanReport():

Metody ‘add’ tego obiektu służą do dodawania do raportu rezultatów skanów pojedyńczych plików. Metoda report() zwraca zawartość raportu.

Moduł Interface:

Poza klasami Interface i Window, reszta klas w tym module dzieli się na dwa rodzaje: te dziedziczące po klasie Window, będące poszczególnymi ekranami do jakich wchodzi użytkownik podczas korzystania z programu, oraz te nie dziedziczące po innych obiektach – te reprezentują obiekty wyświetlane w oknach.

Window(stdscr, actions, legend):

Stdscr to ekran główny, na którym ma odbywać się rysowanie. Actions to słownik reakcji programu na konkretne klawisze wciskane przez użytkownika. Legend to instrukcja wyświetlana na dole ekranu.

Główne metody:

getAction(): pobiera klawisz od użytkownika, a następnie realizuje przypisaną mu akcję

setStatus(): ustawia pole ze statusem (prawy dolny róg) na wkazany tekst (ew. Też kolor)

draw(): rysuje zawartość okna (atrybut items)

__exit__() oraz __enter__(): pozwalają na otwieranie okien za pomocą słowa kluczowego with. Wszystkie zmiany które dane okno wprowadza w terminalu (np. Zmiana widoczności kursora lub czasu oczekiwania na wejście) pownny być obecne w metodzie enter, a odwrócenie tych zmian w close. W ten sposób okno po wyjściu z niego nie pozostawia niechcianych zmian.

Klasy dziedziczące po Window:

Struktura wszystkich tych klas jest bardzo podobna: definiują one akcje dla danego okna, oraz legendę, a także umieszczają w atrybucie items wszystkie elementy które mają być w oknie narysowane. Definiują też metodę get(), która obsługuje akcje wynikające z danego okna. Znajduje się tam główna pętla, której częściami jest rysowanie okna, opuszczanie go, bądź pewne szczególne reakcje, nie zawarte w polu actions. Jeśli jakieś okno potrzebuje zdefiniować konkretną akcję dla dużej liczby klawiszy, tak jak pole tekstowe, to może nadpisać metodę getAction().

Klasy wyświetlanych obiektów:

Te klasy również są podobne: każda definiuje metode draw(), która rysuje dany obiekt na ekranie, oraz metody związane z akcjami dostępnymi dla danego obiektu. Te często są przekazywane bezpośrednio do atrybutu actions klasy Window, gdyż przeważnie na wyświetlanym ekranie jest tylko jeden interaktywny element.

Interface(stdscr):

Ta klasa inicjalizuje obiekt interfejsu graficznego. Znajdujące się w niej metody służą do wyświetlania konkrentych okienek i zwracania wynikających z nich akcji, oraz do komunikacji ze skanerem, aby wyświetlany mógł być postęp skanu, bądź aby skan mógł być przerwany.

W tym module zdefiniowanan jest też funkcja printCmdResult(report, fixed, denied), która jest wykorzystywana do wypisywania rezultatu skanu na konsolę, w przypadku korzystania z aplikacji poprzez podanie agumentów w terminalu.

Moduł antivirus:

Moduł ten zawiera dwie główne funkcje: cmdMain() oraz interactiveMain().

cmdMain(arguments): wykonuje skan na podstawie argumentów podanych w konsoli i wypisuje na konsoli rezultat.

interactiveMain(stdscr): otwiera odpowiednie okna w miarę poruszania się użytkownika przez interfejs. Przeprowadza też odpowiednie skany. Korzysta z oddzielnego wątku w tym celu, aby móc odebrać informację od użytkownika o chęci przerwania skanu, oraz wyświetlać brzebieg skanu.

Zawiera też klasę **ScanThread(Thread)**, która służy do otwierania wątku skanowania.

Testy:

SigBase – testowanie tego modułu odbywa się poprzed oddzielne przetestowanie wyszukiwania sygnatur w ciągach heksadecymalnych oraz porównywanie hashów. Następnie testowana jest też główna funkcja scanFile, której format wyniku jak i jego zawartość są weryfikowane.

FileManager – testowanie tego modułu opiera się na przetestowaniu jego podstawowych funkcjonalności – skanowania plików oraz wycinania z nich wirusów. Testowane są różne rodzaje infekcji plików, oraz rezultat prośby o wycięcie wirusa z jednego z plików.

Antivirus - testowanie odbywa się przez wysłanie polecenia do terminala, z prośbą o skan. Wysyłane są dwie różne prośby, w celu przetestowania flagi -denied. Odbywa się na tym etapie więc testowanie funcjonalności programu jako całości, nie licząc interfejsu graficznego.