Bezpieczeństwo Systemów i Oprogramowania
Projekt
Antywirus- Część podstawowa oraz rozszerzona

Mateusz Kozieł (311018)

Maj 2022

Spis treści

Link do projektu

1 Część podstawowa

1.1 Tematyka

Tematem projektu jest stworzenie prostego oprogramowania antywirusowego działającego w systemie Linux. Oprogramowanie napisane zostało w języku C/C++ z użyciem systemu budowania *cmake*. Wersja języka, w której domyślnie pisany był program to C++17. Wersja ta została wybrana ze względu na dostarczane przez nią biblioteki standardowe- biblioteka *filesystem*, pozwalająca w prosty sposób zarządzać plikami w systemie Linux.
Dodatkowo wykorzystane zostały dwie zewnętrzne biblioteki:

1. CryptoPP- biblioteka dostarczająca funkcje kryptograficzne oraz ładowanie plików w odpowiednich blokach.

2. CLI11- biblioteka dostarczająca struktur pozwalających stworzyć aplikację Command Line'ową.

Oprogramowanie domyślnie zostało uruchomione oraz testowane na systemie Ubuntu 20.04.4 LTS.

1.2 Linkowalność

Po przeanalizowaniu wszystkich możliwości linkowania statycznego oraz dynamicznego zastosowano statyczne linkowanie biblioteki CLI11 oraz CryptoPP, które nie są na tyle dostępne i powszechne co biblioteki standardowe. Pomimo zwiększenia rozmiarów skompilowanego oprogramowania, ustalono, że dla wygody użytkownika i ze względu na większą przenaszalność oprogramowania należy zastosować statyczne linkowanie mniej powszechnych, wyżej wskazanych bibliotek. Biblioteki standardowe linkowane są natomiast w sposób dynamiczny ze względu na ich powszechność.

1.3 Mechanizm wykrywania niebezpiecznych plików

Zgodnie z założeniami projektu, mechanizm wykrywania niebezpiecznych plików opiera się na porównywaniu skrótów analizowanych plików z bazą skrótów szkodliwych plików. W celu optymalizacji działania programu, baza danych czytana jest do struktury <Key,Value>, jaką jest *unordered_set*. Przy dodawaniu elementu do struktury, liczy on hash elementu i na jego podstawie liczony jest klucz. Takim sposobem podczas przeszukiwania bazy danych, porównywane są tylko skróty, a nie całe stringi, co skutecznie zmniejsza ilość operacji potrzebnych do porównania analizowanych ciągów znaków. Funkcją skrótu stosowaną przy wykrywaniu niebezpiecznych plików jest MD5. Została ona wybrana ze względu na szybkość jej obliczenia oraz fakt, że większość dostępnych baz zawiera skróty właśnie w MD5. Fakt istnienia algorytmów generujących kolizje MD5 w tym przypadku nie jest szkodliwy, ponieważ porównujemy skróty liczonych plików i porównujemy je z bazą, więc potencjalna kolizja nie jest w stanie stworzyć zagrożenia dla oprogramowania.

1.4 Sposób uruchamiania

Stworzone oprogramowanie jest aplikacją konsolową wywoływaną z odpowiednimi subkomendami oraz parametrami. W celu sprawdzenia możliwości programu należy wywołać następujące polecenie *./simple_antivirus*. Oprogramowanie wywoływane jest uruchomieniem pliku wykonywalnego następującą komendą, z poszczególnymi, opisanymi poniżej opcjami.
./simple_antivirus

scan - pozwala na skanowanie zarówno pojedynczego pliku jak i całego drzewa podkatalogów

- path - wskazuje miejsce jakie chcemy przeskanować
- d - wskazuje plik bazy danych, na podstawie którego chcemy analizować pliki.

restore - pozwala na przywrócenie pliku z kwarantanny do poprzedniej lokalizacji

- path - lokalizacja w której znajdował się plik przed przeniesieniem do kwarantanny

show - pozwala na pokazanie plików znajdujących się w kwarantannie wraz z aktualną jego nazwą w kwarantannie oraz datą dodania do kwarantanny

Zaimplementowano bezpieczny sposób wyjścia z programu. Poprzez dostarczenie programowi sygnału siłowego zamknięcia, sygnał zostaje przechwytywany, a następnie, przed wyłączeniem programu zostają wykonane wszystkie niezbędne akcje mające na celu bezpieczne zamknięcie programu.

1.5 System uprawnień dla poprawnego działania aplikacji

W realizacji programu założono, że dany użytkownik jest w stanie przeanalizować tylko te pliki, do których ma dostęp, a katalog kwarantanny jest tworzony w jego katalogu domowym, jako ukryty katalog *.quarantine*. Takim sposobem uruchomienie z uprawnieniami roota pozwoli na przeskanowanie większej ilości miejsc, ale pliki będą przenoszone do kwarantanny w */root/.quarantine*. Również, jeżeli dany plik zostanie dodany do kwarantanny z uprawnieniami roota, to musi zostać przywrócony z kwarantanny z uprawnieniami roota. Pozwala to na separację katalogów kwarantanny w zależności od użytkownika.

1.6 Statystyki

Użytkownik ma możliwość podglądu aktualnie analizowanego pliku. Po skończonym skanowaniu, wyświetlane zostają statystyki odnośnie ilości przeskanowanych plików. Dodatkowo, kiedy program napotka na potencjalnie szkodliwy plik, poinformuje nas o pełnej nazwie danego pliku oraz o jego nazwie w katalogu kwarantanny. Dodatkowo istnieje możliwość wyświetlenia statystyk katalogu kwarantanny poprzez wywołanie subkomendy *show*.

1.7 Możliwości uruchomienia

Możliwości uruchomienia są w szczególności dostępne po wywołaniu komendy *./simple_antivirus -h* i zostały opisane w punkcie 4.Sposób uruchomienia. Wyróżnione zostały 3 główne możliwości programu wyróżnione jego subkomendami:

1. scan- opcja skanowania zarówno pojedynczego pliku jak i drzewa podkatalogów
2. restore- opcja przywracania pliku z kwarantanny
3. show- opcja wyświetlenia statystyk katalogu kwarantanny

1.8 Mechanizm kwarantanny

Mechanizm kwarantanny opiera się na przeniesieniu potencjalnie szkodliwego pliku do katalogu kwarantanny odpowiadającego uruchamiającemu użytkownikowi, a następnie odebraniu uprawnień uruchamiania danego pliku oraz zaszyfrowaniu go przy użyciu szyfru blokowego AES. Przy każdym pliku generowany jest odpowiedni klucz szyfrujący oraz odpowiedni wektor inicjalizujący. Dane te zapisywane są do bazy informacji o kwarantannie będącej wygenerowanym plikiem .csv z odebranymi uprawnieniami, które nadawane są w momencie potrzeby odczytu lub zapisu do pliku. Pliki zapisywane są w katalogu kwarantanny w kolejności od najnowszego. W taki sposób zaimplementowane zostały możliwe konflikty nazewnictwa. Dwa pliki, które przed dodaniem do kwarantanny nazywały się tak samo mogą zostać przywrócone wyłącznie w kolejności odwrotnej do dodawania do kwarantanny- od najnowszego do najstarszego.

1.9 Styl kodu

Kod pisany był według wytycznych Google C++ Style Guide. Poddawany był dynamicznej analizie składni zaimplementowanej w zintegrowanym środowisku programistycznym - clang-tidy. Program został przetestowany pod kątem wycieków pamięci przy użyciu narzędzia valgrind. Analiza wykazała zaalokowanie oraz zwolnienie dokładnie takiej samej ilości bajtów.

1.10 Analiza bezpieczeństwa

1.10.1 Funkcja skrótu

Funkcja skrótu MD5 została poddana analizie w punkcie 3. Potencjalne kolizje nie są szkodliwe w tym przypadku użycia funkcji skrótu.

1.10.2 Funkcja szyfru blokowego

Jako szyfr blokowy stosowany w celu uniemożliwienia uruchomienia programu przez roota, użyty został szyfr blokowy AES. Każdy plik szyfrowany jest osobno wygenerowanym kluczem oraz wektorem inicjalizującym. Dane te przechowywane są w pliku informacyjnym katalogu kwarantanny.

1.10.3 Uprawnienia

Największe zagrożenia budzą uprawnienia do konkretnych plików oraz katalogów. Przykładowo, jeżeli użytkownik będzie chciał zmienić uprawnienia do pliku informacyjnego katalogu kwarantanny, to będzie miał taką możliwość, ponieważ jest jego właścicielem. W takiej sytuacji jest w stanie dopisać do pliku informacyjnego linię, definiującą plik, któremu chce zmienić uprawnienia na większe. Musiałby:

1. Zaszyfrować plik odpowiednim kluczem oraz z odpowiednim wektorem i wrzucić do katalogu kwarantanny.
2. Zapisać te dane do pliku informacyjnego
3. Ustawić w pliku informacyjnym odpowiednie poprzednie uprawnienia pliku- takie do których chcemy zmienić uprawnienia
4. Uruchomić program z opcją restore i ustawionymi przez nas parametrami

Opcja ta jest jednak dość mało prawdopodobna, a prawdopodobieństwo jej użycia zostało maksymalnie obniżone poprzez odebranie uprawnień do pliku poza działaniem programu. Zagrożenie mogłoby zostać zupełnie wyeliminowane w przypadku stałego odbioru uprawnień do pliku, a dostęp do odczytu i zapisu realizowany byłby przy użyciu capabilities. W zaawansowanej części projektu planowane jest zastosowanie przedstawionego rozwiązania.

2 Część zaawansowana

2.1 Tematyka

Tematyka części zaawansowanej projektu opierała się na usprawnieniu oraz dodaniu dodatkowych funkcji do pierwotnej wersji- części podstawowej. Zagadnieniami rozwijanymi w tym etapie były między innymi:

- Działanie programu w tle
- Implementacja wielowątkowości
- Implementacja dodatkowej funkcji bezpieczeństwa

2.2 Działanie w tle

Działanie w tle zostało zaimplementowane jako równolegle działająca aplikacja, która nie obciąża systemu oraz pozwala na działanie wszystkich innych funkcji. Oparta została ona o mechanizm dostarczony przez moduł *inotify*. Dzięki temu, program jest czuły na wszelkie modyfikacje monitorowanych plików. Zaimplementowano również mechanizm zarządzający dodawanymi lub usuwanymi podkatalogami. Jeżeli zostanie wykryta modyfikacja pliku, liczony jest jego skrót, a następnie porównywany jest z bazą skrótów. Jeżeli zostanie uznany za niebezpieczny, to nałożona zostanie na niego wcześniej zaimplementowana kwarantanna. Program pozwala na bezpieczne zakończenie jego pracy poprzez wykrycie akcji klawiszem Q, lub Esc.

2.3 Wielowątkowość

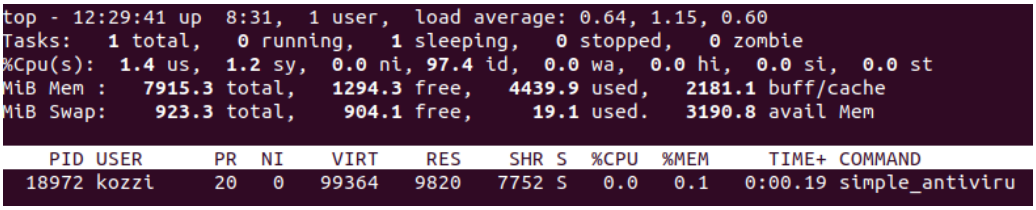
Funkcja wielowątkowości zaimplementowana została jako dodatek do mechanizmu monitorowania. Zastosowano podział na wątek zarządzający monitorowaniem oraz wątki analizujące akcje wykryte przez wątek monitorujący. Takim sposobem, jeżeli wykryta zostanie jakakolwiek akcja na pliku, wątek monitorujący przesyła do kolejnego wątku, dane które ten musi przeanalizować.

2.4 Dodatkowa funkcja

W celu zapewnienia użytkownikowi dostępu do stale aktualizowanych baz danych sygnatur, zaimplementowano w programie komunikację z API VirusTotal, dzięki czemu, użytkownik jest w stanie dokładniej przeanalizować dany plik pod kątem jego bezpieczeństwa. Wymagane jest założone konto oraz uzyskanie API Key. Jeżeli dany plik zostanie określony jako złośliwy, to wyświetlony zostanie wynik przekazujący informacje o dokładnym mechanizmie który go wykrył, oraz jak go klasyfikuje. Wówczas zostaje poddawany kwarantannie.

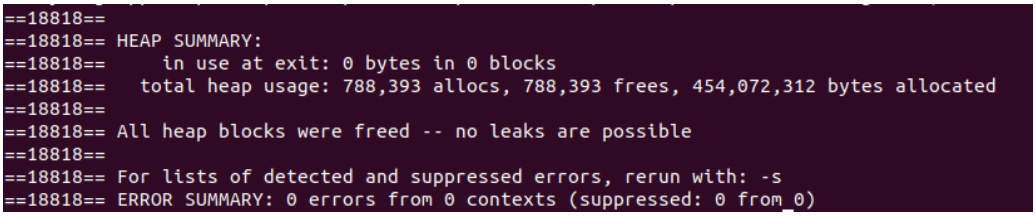
2.5 Obciążenie systemu

Program został poddany analizie podczas pracy w tle jako monitor. Analiza nie wykazała znaczącego użycia podzespołów komputera, co widać na przedstawionym poniżej zrzucie ekranu.



Rysunek 1: Analiza użycia zasobów przy użyciu narzędzia top

Przetestowano również program pod kątem wycieków pamięci. Tutaj również analiza nie wykazała żadnych problemów.



Rysunek 2: Analiza pamięci przy użyciu narzędzia valgrind

2.6 Aktualizacja bazy sygnatur

Aktualizacja bazy sygnatur została zaimplementowana w formie możliwości wskazania konkretnej bazy sygnatur, na której działać ma program. Funkcja ta istniała już w wersji podstawowej oraz nie została zmieniana.

2.7 Aktualizacje części podstawowej

1. Zgodnie z zaleceniami poprawiono bezpieczeństwo bazy danych kwarantanny. Aktualnie baza danych tworzona jest oraz używana z uprawnieniami 000, a program ma do niej dostęp poprzez capabilities, które dodawane są podczas kompilowania programu.

2. Poprawiono czytelność kodu w main.c