Przegląd Projektu Zespołowego Oprogramowanie Antywirusowe Windows 11

Jakub Sztolcman, Bartek Pliński

1. Wprowadzenie

Projekt zakłada stworzenie zaawansowanego oprogramowania antywirusowego dla systemu operacyjnego Windows 11. Architektura oprogramowania będzie oparta na dwóch kluczowych modułach: **aplikacji działającej w trybie użytkownika (User Mode)** oraz **sterownika działającego w trybie jądra (Kernel Mode).** Moduł User Mode zapewni przyjazny interfejs użytkownika (GUI) oraz zarządzanie podstawowymi zadaniami, podczas gdy moduł Kernel Mode umożliwi dostęp do jądra systemu w celu wykrywania zaawansowanych zagrożeń.

2. Architektura i Główne Komponenty

Moduł UserMode (aplikacja):

- **GUI (Graphical User Interface):** Przejrzysty interfejs. Zapewni on możliwość uruchamiania skanowań, przeglądania wyników, konfiguracji ustawień ochrony w czasie rzeczywistym oraz zarządzania kwarantanną.
- **Menedżer Skanowania:** Koordynuje zadania skanowania (pliki, procesy), zarządza kolejką zadań i prezentuje postęp użytkownikowi.
- **Baza Danych Sygnatur:** Przechowuje skróty hash (MD5, SHA-256) znanych złośliwych plików oraz sygnatury kodu malware.

Moduł KernelMode (sterownik):

- **Sterownik (Windows Driver):** Niskopoziomowy sterownik napisany przy użyciu Windows Driver Kit (WDK), który ma uprawnienia do bezpośredniego dostępu do pamięci jądra, struktur systemowych i rejestrów.
- **Monitor Zachowań (Behavioral Monitor):** Część sterownika odpowiedzialna za analizę aktywności procesów i jądra w czasie rzeczywistym.
- **Mechanizm Callback:** Rejestruje funkcje zwrotne (callbacks) w systemie, aby przechwytywać kluczowe zdarzenia (np. uruchomienie procesu, załadowanie sterownika).

3. Opis Funkcjonalności

3.1 Skanowanie systemu plików w oparciu o blacklistę hash-y

- **Cel:** Szybka identyfikacja i neutralizacja znanych, rozpowszechnionych zagrożeń poprzez porównanie z bazą danych skrótów.
- **Działanie:** Moduł skanujący przegląda system plików. Dla każdego napotkanego pliku obliczany jest jego skrót kryptograficzny (np. SHA-256). Obliczony hash jest następnie porównywany z predefiniowaną "czarną listą" (blacklistą). W przypadku dopasowania, plik jest natychmiast blokowany, a użytkownik otrzymuje alert z opcjami usunięcia lub przeniesienia do kwarantanny.

3.2 Skanowanie działających procesów systemowych pod kątem modyfikacji kodu

- **Cel:** Wykrywanie zaawansowanych technik ukrywania złośliwego kodu, takich jak DLL Injection, Process Hollowing lub API Hooking.
- **Działanie:** Dla każdego działającego procesu systemowego, moduł usermode:
    1. **Sprawdza integralność pamięci:** Porównuje kod załadowany w pamięci RAM z oryginalnym obrazem na dysku, szukając podejrzanych niezgodności.
    2. **Skanuje pod kątem sygnatur:** Przegląda regiony pamięci procesu w poszukiwaniu sekwencji bajtów (sygnatur) charakterystycznych dla znanych malware.
    3. **Wykrywa modyfikacje:** Sprawdza, czy główne wątki procesu nie wykonują kodu w niestandardowych, podejrzanych lokalizacjach w pamięci.

3.3 Skanowanie pamięci procesów systemowych pod kątem wykonywalnych alokacji

- **Cel:** Wykrywanie tzw. "shellcode'u" – złośliwego kodu wprowadzanego do pamięci procesu w celu jego wykonania (częste w exploitach).
- **Działanie:** Moduł usermode sprawdza zaalokowaną pamięć we wszystkich procesach systemowych. Szczególną uwagę zwraca na strony pamięci zaalokowane dynamicznie (np. za pomocą `VirtualAllocEx` lub `NtMapViewOfSection`) z ustawionymi uprawnieniami do wykonania (`PAGE_EXECUTE_READWRITE`). Taka kombinacja (dynamiczna alokacja + uprawnienia execute) jest wysoce podejrzana i generuje alert.

3.4 Monitorowanie wpisów autostartu (Task Scheduler)

- **Cel:** Uniemożliwienie malware'owi utrzymania trwałości w systemie poprzez rejestrację w harmonogramie zadań.
- **Działanie:** Oprogramowanie okresowo skanuje zadania w Task Scheduler. Dla każdego zadania, które uruchamia plik wykonywalny (.exe, .dll) lub skrypt (np. .ps1, .bat), weryfikowany jest jego podpis cyfrowy. Takie pliki bez ważnego podpisu cyfrowego są traktowane jako podejrzane i zgłaszane użytkownikowi.

3.5 Hook funkcji BitBlt w niepodpisanych procesach

- **Cel:** Ochrona prywatności przed złośliwym oprogramowaniem przechwytującym zrzuty ekranu (np. bankowe trojany, szpiegowskie malware).
- **Działanie:** Moduł usermode przechwytuje (hooks) wywołania funkcji `BitBlt` (używanej do kopiowania bitmap) w kontekście każdego niepodpisanego ważnym podpisem cyfrowym procesu. Gdy taki proces próbuje wywołać `BitBlt` wywołanie jest blokowane, a próba zostaje zarejestrowana.

3.6 Integrity check załadowanych sterowników jądra

- **Cel:** Wykrywanie rootkitów, które modyfikują załadowane w pamięci sterowniki systemowe.
- **Działanie:** Sterownik antywirusa okresowo ładuje obrazy wszystkich załadowanych sterowników z dysku i porównuje segmenty wykonywalne bajt po bajcie z ich odpowiednikami załadowanymi w pamięci RAM. Podejrzane rozbieżności są sygnałem obecności zaawansowanego rootkita.

3.7 Wykrywanie ręcznie mapowanych sterowników

- **Cel:** Wykrywanie sterowników załadowanych nielegalnie, z pominięciem menedżera sterowników Windows, co jest standardową techniką rootkitów.
- **Działanie:** Sterownik skanuje pamięć jądra w poszukiwowaniu:
    1. Regionów pamięci oznaczonych jako wykonywalne (EXECUTABLE), które nie są powiązane z żadnym poprawnie załadowanym modułem.
    2. Wątków systemowych, których punkt startowy znajduje się poza granicami jakiegokolwiek poprawnie załadowanego sterownika.
    3. Wszelkie takie wykrycia są traktowane jako krytyczne zagrożenie i są naprawiane – wątki zatrzymywane, kod usuwany z pamięci.

3.8 Wykrywanie hook-ów w funkcjach sterowników Windows

- **Cel:** Wykrywanie rootkitów, które przechwytują (hook) funkcje w jądrze systemu w celu kontroli nad przepływem wykonywania.
- **Działanie:** Sterownik analizuje kluczowe funkcje w często atakowanych sterownikach (np. `ntoskrnl.exe`, `win32k.sys`). Sprawdza, czy początek tych funkcji nie zawiera skoku (`JMP`) lub wywołania (`CALL`) do nieautoryzowanej lokalizacji w pamięci, co jest oznaką modyfikacji przez rootkit.

3.9 Weryfikacja integralności SSDT, SSSDT, MSR

- **Cel:** Ochrona fundamentalnych struktur systemowych przed modyfikacją przez rootkity.
- **Działanie:**
    1. **SSDT/SSSDT (System Service Descriptor Table):** Struktury te to listy wskaźników do funkcji Syscall. Sterownik weryfikuje integralność tej listy poprzez sprawdzenie czy wskaźnik dla danej funkcji wskazuje na miejsce w pamięci jednego ze sterowników systemowych.
    2. **MSR (Model-Specific Registers):** Sterownik sprawdza, czy rejestry MSR (np. `LSTAR` w systemach x64, który przechowuje wskaźnik do procedury obsługi wywołań systemowych `KiSystemCall64`) nie zostały zmodyfikowane, aby przekierowywać wywołania systemowe na złośliwy kod.

3.10 Rejestracja callback'ów dla zdarzeń systemowych

- **Cel:** Aktywna, prewencyjna ochrona w czasie rzeczywistym przed uruchamianiem malware.
- **Działanie:** Sterownik rejestruje w systemie Callbacks aby otrzymywać powiadomienia o kluczowych zdarzeniach, takich jak:
    1. **LoadImage:** Gdy system ma załadować sterownik lub bibliotekę DLL.
    2. **CreateProcess:** Gdy ma zostać uruchomiony nowy proces.

Gdy zdarzenie wystąpi, sterownik przechwytuje żądanie i weryfikuje obiekt na podstawie jego skrótu hash, sygnatury kodu oraz podpisu cyfrowego. Jeśli obiekt pasuje do czarnej listy lub nie posiada ważnego podpisu a jest sterownikiem bądź .DLL który próbuje się załadować do procesu systemowego, jego ładowanie/uruchomienie zostaje zablokowane.