OCHRONA OPROGRAMOWANIA

Twórcy oprogramowania jako cel hakerów

Rosnąca ilość ataków supply chain wskazuje na pewien negatywny trend. Wykorzystanie producentów oprogramowania w roli trampoliny do przeprowadzenia ataku jest coraz bardziej popularne i wysoce skuteczne. Tak bardzo, że wg raportu ENISA liczba naruszeń danych pochodzących z ataków na łańcuch dostaw przewyższyła w 2022 r. liczbę naruszeń spowodowanych przez złośliwe oprogramowanie o około 40%.

Producenci oprogramowania i sprzętu zawierającego oprogramowanie, niezależni twórcy, software house'y, integratorzy – wszyscy staliśmy się atrakcyjnym „nośnikiem zaufania".

W atakach supply chain celem atakujących jest wprowadzenie złośliwego kodu do oprogramowania producenta, który następnie rozprzestrzeni się na komputery użytkowników lub inne urządzenia (np. IoT). Odbywa się to poprzez manipulację oprogramowaniem (np. ingerencję w kod wynikowy) i ma na celu wprowadzenie niepożądanych funkcji. Brak lub słaba ochrona kodu wynikowego stwarza więc ryzyko jego infekcji.

W jaki sposób podnieść poziom bezpieczeństwa? Jedną z metod jest szyfrowanie. Mówi o tym chociażby mająca już niedługo obowiązywać dyrektywa unijna o cyberodporności (CRA). Skuteczna ochrona oprogramowania może być wykonana przy użyciu metod kryptograficznych. Możemy pójść jednak o krok dalej i rozszerzyć je o pewne, dedykowane dla ochrony kodu wynikowego, mechanizmy.

DOBÓR ALGORYTMU SZYFRUJĄCEGO I INNE „DETALE"

W kwestii wyboru algorytmu zaleca się korzystanie z gotowych rozwiązań. Szyfry personalizowane mają nieznaną odporność, a dowiedzenie ich skuteczności jest niezwykle trudne i stwarza dodatkowe ryzyko.

Stosowanym powszechnie algorytmem do szyfrowania danych jest symetryczny AES z kluczem o długości 256 bitów. Algorytm ma udowodnioną odporność, co oznacza, że prawidłowe użycie gwarantuje ochronę na poziomie, który zniechęca do jego łamania.

Drugą kwestią jest zaszyfrowanie samego klucza, używanego przez AES. Można tu wykorzystać szyfr asymetryczny ECC (224 bity), posługujący się kluczem publicznym i prywatnym.

Zabezpieczenie kodu wynikowego można rozszerzyć o takie właściwości jak:

Tworzenie wariantów, które zwiększają złożoność kodu wynikowego. Zaimplementowane funkcje są powielane w wielu wariantach. W zależności od podanych parametrów wejściowych funkcja opakowująca (wrapper) wybiera, który wariant jest wykonywany.

Modyfikacja wariantów w taki sposób, aby działały tylko w zakresie wartości prawidłowych dla każdego wariantu. Uniemożliwia to atakującym modyfikacje funkcji wrappera, których celem jest to, aby za każdym razem wykonywała tylko ten sam wariant.

Szyfrowanie wariantów, aby uniemożliwić atakującym inżynierię wsteczną kodu bez jego wcześniejszego odszyfrowania.

Wstawianie pułapek. Oprócz już wygenerowanych wariantów tworzone są dodatkowe warianty – pułapki, które są szyfrowane. Pułapka zawiera kod blokady, co oznacza, że jeśli pułapka zostanie odszyfrowana za pomocą klucza sprzętowego, klucz sprzętowy sam się zablokuje i nie będzie już mógł być używany do odszyfrowywania. Uniemożliwia to atakującemu odszyfrowanie wszystkich metod bez inżynierii wstecznej.

Wybór wariantu klucza sprzętowego. Funkcja opakowująca podczas wyboru wariantu korzysta z klucza sprzętowego. Wysyła do niego parametry wejściowe, a zwracany jest wariant, który ma być użyty. Sprawia to, że atakujący nie jest w stanie rozpoznać za pomocą inżynierii wstecznej funkcji wrappera, który wariant jest potrzebny. Musiałby uruchomić kod dla wszystkich możliwych parametrów wejściowych, aby móc go wskazać z absolutną pewnością.

Zapisywanie statusów w kluczu sprzętowym. Funkcje wykonywane są w takiej kolejności, jaką przewidział programista. Ostatnia metoda służąca do deszyfrowania przechowywana jest w kluczu sprzętowym jako status. W trakcie kolejnego odszyfrowywania następuje weryfikacja oczekiwanego statusu. Jeśli wynik jest negatywny, można uruchomić pułapkę. Uniemożliwia to atakującemu wypróbowanie wszystkich wariantów w dowolnym momencie wykonania programu. Musiałby on w tym celu zawsze wracać do punktu wyjścia. W ten sposób rośnie złożoność przeprowadzanego ataku.

JAK TO DZIAŁA W PRAKTYCE?

Samodzielna implementacja szyfrowania oraz zaawansowanych metod ochrony to złożony proces i dla większości projektów nieopłacalny. Dobór odpowiedniego sprzętu bądź innej metody do bezpiecznego przechowywania kluczy stanowi dodatkowe wyzwanie. Można natomiast skorzystać z jednego z gotowych rozwiązań, jakim jest Code Meter firmy Wibu-Systems.

Ogromną zaletą Code Meter jest duża elastyczność w zakresie integracji. Mamy do dyspozycji:

- pakiet AxProtector, za którego pomocą z niewielkim nakładem pracy (GUI lub wiersz poleceń) możemy zabezpieczyć kod wynikowy
- bibliotekę SDK i API, które pozwalają na indywidualne użycie funkcji ochrony i szyfrowania w swoim kodzie.

Code Meter wspiera wiele języków programowania: .Net, Java, Python, Java Script, C, C++, systemów operacyjnych: Windows, Linux, macOS i platform (embedded, IoT, Rasbery).

Proces zabezpieczania oprogramowania i/lub danych składa się z trzech kroków:

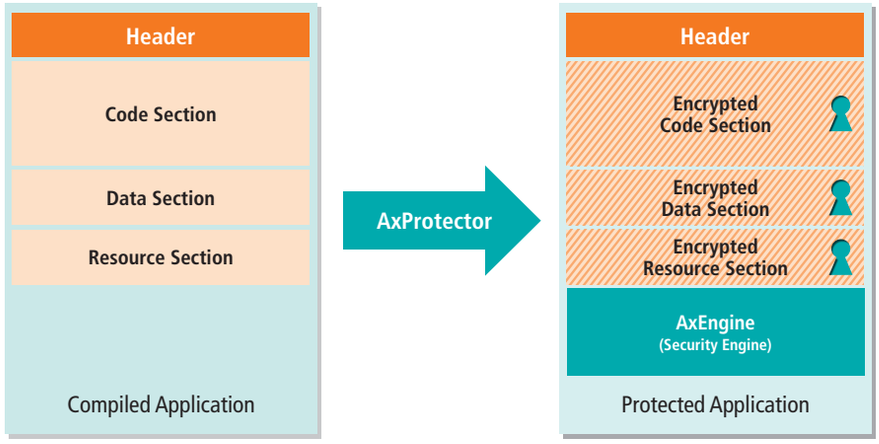
1. Szyfrowanie i integracja metod ochrony oprogramowania, np. przy użyciu AxProtector. Operacja wykonywana jest jednorazowo, dla konkretnego wydania aplikacji. AxProtector monitoruje autentyczność i integralność oprogramowania oraz chroni je przed inżynierią wsteczną.
2. Drugi krok dotyczy zabezpieczenia uruchomienia zaszyfrowanego oprogramowania. W tym celu twórca aplikacji generuje licencję dla konkretnych swoich klientów. Dzięki temu uruchomienie

ich jest możliwe tylko wtedy, gdy dostępny jest odpowiedni klucz licencyjny. Licencje można generować z poziomu wiersza poleceń lub w sposób zautomatyzowany w License Central.

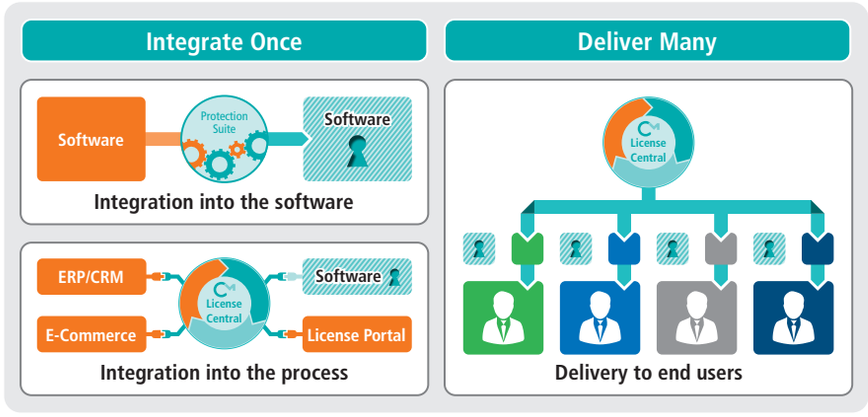
3. Ostatni krok to dystrybucja licencji. Licencje mogą być zapisane na kluczu sprzętowym, w lokalnym – zaszyfrowanym pliku lub w chmurze. Całą logistyką dystrybucji licencji można zarządzać również z poziomu License Central.

Dla twórców dostępna jest do testów darmowa biblioteka SDK, do pobrania po uprzedniej rejestracji na stronie: https://www.wibu.com/pl/formularz-zamowienia-dla-codemeter-software-development-kit-sdk.html

Technologia Code Meter wykorzystuje kryptografię oraz zaawansowane metody ochrony kodu. Odpowiednio zastosowane podnoszą skutecznie poziom bezpieczeństwa. Sposób ich integracji z własnymi rozwiązaniami wymaga odpowiedniego zaplanowania, niemniej wymagany nakład pracy jest o rzędy wielkości niższy od wariantu, w którym wszystkie wymienione mechanizmy mielibyśmy implementować samodzielnie.



Rysunek 1. Elementy skompilowanej aplikacji podlegające ochronie przez AxProtector



Rysunek 2. Sposób integracji zabezpieczeń z oprogramowaniem i dystrybucja licencji

JANUSZ HRYSZKIEWICZ

janusz.hryszkiewicz@wibu.com

Związany z IT od 25 lat, głównie w zakresie rozwiązań dedykowanych (Microsoft, Java, Python, SAP). Autor kilku rozwiązań dla biznesu, które stały się produktami. Obecnie zaangażowany w zwiększenie cyberodporności oprogramowania, AI oraz systemów wbudowanych.