

Opis przedmiotu zamówienia

I. Przedmiot zamówienia:

Dostawa platformy bezpieczeństwa wspomagającej analizę i testowanie kodu źródłowego.

1. Dostawa platformy bezpieczeństwa aplikacji zintegrowanej z CI/CD oraz IDE, wspomagającej analizę i testowanie kodu na etapie produkcji (shift-left), oferowanej w formie ON-PREMISES dla 330 programistów (**zamówienie podstawowe**), udostępniającej następujące funkcje:
 - a. Statyczna analiza kodu - analizuje kod źródłowy aplikacji podczas produkcji programowania.
 - b. Analiza składu oprogramowania - identyfikuje pośrednie biblioteki open source i wskazuje podatności, warunki licencyjne i ryzyko operacyjne.
 - c. Skanowanie obrazów oraz konfiguracji kontenerów pod kątem bezpieczeństwa.
 - d. Skanowanie konfiguracji Infrastructure as Code pod kątem bezpieczeństwa.
2. Dostawa dodatkowych licencji, o których mowa w pkt 1. dla maksymalnie 330 programistów – (**zamówienie opcjonalne**).

Termin realizacji i zakres

1. Uruchomienie i konfiguracja rozwiązania muszą zostać wykonane w ciągu 120 dni roboczych od zawarcia umowy, w oparciu o uzgodniony harmonogram.
2. Okres gwarancji/ wsparcia: 2 lata

II. Szczegółowy opis wymagań

1. Platforma

1.1. Model wdrożenia

- 1.1.1. Rozwiązanie musi umożliwiać pracę w modelu ON-PREMISES, tj. dane muszą być przetwarzane w ośrodku przetwarzania wskazanym przez Zamawiającego, przynajmniej w zakresie analizowanego kodu źródłowego tj. Zamawiający nie dopuszcza do klonowania repozytorium celem jego skanowania poza jego ośrodkami przetwarzania.
- 1.1.2. Zamawiający dopuszcza model, w którym panel zarządzania rozwiązaniem dostarczany jest w postaci usługi chmurowej, z zastrzeżeniem punktu powyższego.
- 1.1.3. Producent musi posiadać aktualny w dniu złożenia oferty certyfikat zaświadcający, że jego organizacja posiada system zarządzania bezpieczeństwem informacji (ISMS) zgodny z wymaganiami normy ISO/IEC 27001:2013.

1.2. W zakresie zarządzania tożsamością i dostępem, rozwiązanie musi umożliwiać:

- 1.2.1. Uwierzelnianie użytkowników za pośrednictwem SSO.

- 1.2.2. Zarządzanie dostępem z wykorzystaniem protokołu SAML.
- 1.2.3. Zarządzanie użytkownikami za pośrednictwem interfejsu API.
- 1.2.4. Rozliczalność - wszystkie działania użytkownika w środowisku muszą być logowane.
- 1.2.5. Szczegółową kontrolę dostępu, z możliwością kontrolowania, którzy użytkownicy mają prawa odczytu lub odczytu/zapisu, do jakiej grupy projektów.
- 1.3. Interfejs API rozwiązania musi umożliwiać:
 - 1.3.1. Przegląd, dodawanie i usuwanie projektów.
 - 1.3.2. Przegląd wszystkich problemów wybranego projektu.
 - 1.3.3. Uruchamianie skanowania.
 - 1.3.4. Utworzenie zgłoszenia w JiraSD.
 - 1.3.5. Edycję polityki powiadomień o błędach.
 - 1.3.6. Dostęp do listy pośrednio ładowanych bibliotek z informacją o licencjach.
- 1.4. W zakresie raportowania i ustawiania priorytetów rozwiązanie musi:
 - 1.4.1. Raportować wszystkie problemy w kolejności priorytetów, ze wskazaniem ich wagi.
 - 1.4.2. Nadawać priorytet problemom w oparciu o publicznie dostępną informację o dojrzałości exploitów.
 - 1.4.3. Nadawać priorytety problemom w oparciu o dostępność poprawki lub jej brak.
 - 1.4.4. Nadawać priorytet na podstawie tego, czy podatna funkcja jest wywoływana przez kod (osiągalność).
 - 1.4.5. Dostarczać informacje na temat wywołań funkcji podatnych na ataki.
 - 1.4.6. Umożliwiać użytkownikowi ignorowanie podatności do czasu udostępnienia poprawki.
 - 1.4.7. Umożliwiać oznaczanie projektów i filtrowanie na podstawie własnych, niestandardowych oznaczeń.
 - 1.4.8. Umożliwiać grupowanie projektów z różnymi użytkownikami i uprawnieniami użytkowników.
 - 1.4.9. Generować SBOM (Software Bill Of Materials) zawierający wszystkie biblioteki pośrednie we wszystkich projektach.
 - 1.4.10. Wspierać scentralizowane zarządzanie politykami bezpieczeństwa.
 - 1.4.11. Umożliwiać stosowanie różnych polityk bezpieczeństwa do różnych grup projektów.
 - 1.4.12. Umożliwiać zmianę wagi określonych problemów CVE lub CWE
- 1.5. W zakresie alertowania i powiadomienia rozwiązanie musi:
 - 1.5.1. Powiadamiać o nowych podatnościach i problemach licencyjnych przez email.
 - 1.5.2. Udostępniać użytkownikowi tygodniowe podsumowanie jego podatności i problemów licencyjnych.
 - 1.5.3. Wystawiać i przypisywać zgłoszenia w systemie JiraSD.

2. FUNKCJA STATYCZNEJ ANALIZY KODU MUSI UMOŻLIWIAĆ:

2.1. W zakresie integracji:

- 2.1.1. Integracja lokalna z GitHub Enterprise, Gitlab Enterprise i Bitbucket Server.
- 2.1.2. Dostępność interfejsu linii komend CLI.
- 2.1.3. Automatyzacja skanowania w ramach cyklu kompilacji Continuous Integration.
- 2.1.4. Integracja z narzędziami CI/CD: Jenkins, Bitbucket Pipelines.
- 2.1.5. Przerywanie kompilacji CI/CD przy użyciu konfigurowalnych polityk, w przypadku wykrycia podatności powyżej konfigurowalnego poziomu ryzyka.
- 2.1.6. Integracja ze środowiskiem deweloperskim IDE dla pakietów JetBrains (IntelliJ, PyCharm, PhpStorm, Webstorm) oraz Visual Studio & Visual Studio Code.

2.2. Wsparcie języków programowania przynajmniej dla:

- 2.2.1. Java
- 2.2.2. Javascript
- 2.2.3. Typescript
- 2.2.4. PHP
- 2.2.5. C# / dotnet
- 2.2.6. Python

2.3. Wsparcie procesu remediacji podatności:

- 2.3.1. Czas skanowania testowego repozytorium OWASP <https://github.com/juice-shop/juice-shop> musi być krótszy niż 10 minut.
- 2.3.2. Skanowanie projektów importowanych z Git.
- 2.3.3. Ponowne skanowanie każdego PR Check w ramach testu Git.
- 2.3.4. Blokowanie Pull Request, który wprowadza do kodu nową podatność.
- 2.3.5. Skanowanie dowolnego Pull Request z dowolnej gałęzi do dowolnej gałęzi za pomocą prostej konfiguracji.
- 2.3.6. Skanowanie gałęzi innych niż domyślne oraz wielu gałęzi.
- 2.3.7. Przesyłanie wyników skanowania CLI do interfejsu przeglądarkowego.
- 2.3.8. Ignorowanie wskazanych ścieżek lub folderów (np. foldery testowe).
- 2.3.9. Rekomendacje działań remediacji.
- 2.3.10. Odnajdowanie sekretów i poświadczeń w kodzie.
- 2.3.11. Możliwość ignorowania podatności do czasu dostępności poprawki.
- 2.3.12. Skanowanie zaimportowanych repozytoriów w poszukiwaniu nowo zidentyfikowanych problemów w regularny sposób, zgodnie z polityką.
- 2.3.13. Wsparcie programistów w nauce bezpiecznego kodowania.

2.3.14. Wsparcie analizy błędów pokazujące przepływ sterowania linia po linii.

3. FUNKCJA ANALIZY SKŁADU OPROGRAMOWANIA MUSI UMOŻLIWIAĆ:

3.1. W zakresie integracji:

- 3.1.1. Integracja lokalna z GitHub Enterprise, Gitlab Enterprise i Bitbucket Server.
- 3.1.2. Dostępność interfejsu linii komend CLI.
- 3.1.3. Automatyzacja skanowania w ramach cyklu kompilacji Continuous Integration.
- 3.1.4. Integracja z narzędziami CI/CD: Jenkins, Bitbucket Pipelines.
- 3.1.5. Przerywanie kompilacji CI/CD przy użyciu konfigurowalnych polityk, w przypadku wykrycia podatności powyżej konfigurowalnego poziomu ryzyka.
- 3.1.6. Integracja z środowiskami IDE: IntelliJ, Visual Studio & Visual Studio Code, PHPstorm.
- 3.1.7. Integracja z rejestrami Artifactory i Sonatype, zapewnienie, że wszystkie artefakty są skanowane przed dodaniem, okresowo i przed pobraniem, z zastosowaniem konfigurowanej polityki blokowania.

3.2. Wsparcie języków programowania:

- 3.2.1. Bezpośrednie zależności - wykrywanie podatności w bibliotekach ładowanych bezpośrednio, w następujących językach i menedżerach pakietów:
 - Javascript/Node/Typescript z npm i Yarn
 - Java z Mavenem
 - Java z Gradle
 - PHP z Composerem
 - .Net, .Net Core, C# z Nuggetem
 - Python z PIP Installs Packages - Python Package Index (PyPI)
- 3.2.2. Pośrednie zależności - wykrywanie podatności w bibliotekach ładowanych pośrednio w następujących językach i menedżerach pakietów:
 - Javascript/Node/Typescript z npm i Yarn
 - Java z Mavenem
 - Java z Gradle
 - PHP z Composerem
 - .Net, .Net Core, C# z Nuggetem
 - Python z PIP Installs Packages - Python Package Index (PyPI)
- 3.2.3. Rekomendowanie najniższej dostępnej wersji biblioteki nadrzędnej, która mityguje podatność przechodnią, aby zminimalizować ryzyko 'uszkodzenia kodu', dla następujących języków:
 - Javascript/Node/Typescript z npm i Yarn
 - Java z Mavenem

- Java z Gradle
- PHP z Composerem
- .Net, .Net Core, C# z Nuggetem
- Python z PIP Installs Packages - Python Package Index (PyPI)

3.3. W zakresie wsparcia procesu remediacji podatności:

- 3.3.1. Automatyczne generowanie fix pull/merge requestów dla bezpośrednich zależności.
- 3.3.2. Automatyczne generowanie fix pull/merge requestów dla zależności nadrzędnej w celu rozwiązania problemów w bibliotekach podrzędnych.
- 3.3.3. Mechanizm tworzenia fix pull requestów dla najpilniejszych problemów i komunikowania nowych dopiero po ich naprawieniu.
- 3.3.4. Możliwość sprawdzenia pull/merge requestów pod kątem nowych podatności wprowadzanych do zaimportowanego projektu.
- 3.3.5. Możliwość blokowania pull/merge requestów, gdy wprowadzają one nowe podatności, w oparciu o konfigurowalną politykę.
- 3.3.6. Skanowanie gałęzi innych niż domyślne oraz wielu gałęzi.
- 3.3.7. Dostępność interfejsu CLI.
- 3.3.8. Możliwość automatycznego zgłaszania pull requestów w celu aktualizacji zależności.
- 3.3.9. Wizualizacja drzewa zależności dla każdego projektu, z podatnościami, zależnościami i zagnieżdżonymi problemami.
- 3.3.10. Możliwość ignorowania podatności do czasu udostępnienia poprawki.
- 3.3.11. Jeśli nie istnieje aktualizacja rozwiązująca problem bezpieczeństwa, dostawca musi sugerować poprawkę usuwającą podatność tam gdzie to możliwe, w menedżerach pakietów, które na to pozwalają (npm, Yarn).
- 3.3.12. Stałe skanowanie zaimportowanych repozytoriów w poszukiwaniu ostatnio ujawnionych podatności.
- 3.3.13. Ocena ryzyka uszkodzenia kodu przez konkretną poprawkę lub aktualizację w oparciu o dostępne statystyki.

3.4. W zakresie polityki licencyjnej:

- 3.4.1. Możliwość przetwarzania SBOM w formatach CycloneDX i SPDX.
- 3.4.2. Konfigurowalna polityka licencyjna z konfigurowalnymi parametrami ryzyka i komentarzami.
- 3.4.3. Możliwość stosowania różnych polityk licencyjnych do grup projektów.
- 3.4.4. Dodawanie niestandardowych notatek, informujących o typie licencji.
- 3.4.5. Przeglądanie i eksportowanie praw autorskich wszystkich licencji bibliotek projektu do pliku CSV.

3.5. W zakresie bazy danych podatności Open Source

- 3.5.1. Wskazywanie konkretnej podatnej funkcji w podatnej bibliotece.
 - 3.5.2. Informacja o punktacji CVSS każdej podatności.
 - 3.5.3. Podatności w bazie danych muszą być udokumentowane w języku angielskim.
 - 3.5.4. Baza danych podatności musi zawierać problemy nieujęte w rejestrze CVE.
 - 3.5.5. Każdy zidentyfikowany problem musi zawierać link do CVE lub innego źródła.
4. FUNKCJA SKANOWANIE OBRAZÓW ORAZ KONFIGURACJI KONTENERÓW MUSI UMOŻLIWIAĆ:
- 4.1. W zakresie integracji:
 - 4.1.1. Integracja z rejestrami: Rozwiązanie posiada integracje z AWS ECR, Azure ACR, rejestrem Artifactory, Docker Hub, Google GCR.
 - 4.1.2. Integracja z Kubernetes w dowolnym środowisku.
 - 4.1.3. Dostępność interfejsu CLI.
 - 4.1.4. Automatyzacja w ramach cyklu kompilacji Continuous Integration.
 - 4.1.5. Integracja z następującymi narzędziami CI/CD: Jenkins, Bitbucket Pipelines.
 - 4.1.6. Przerywanie kompilacji CI/CD w przypadku wykrycia podatności powyżej konfigurowalnego poziomu.
 - 4.1.7. Integracja Git do skanowania plików Docker z GitHub, Gitlab, Bitbucket, Azure repos.
 - 4.2. Skanowanie systemu operacyjnego i obrazu bazowego:
 - 4.2.1. Możliwość skanowania podatności w następujących systemach operacyjnych:
 - Debian,
 - Ubuntu,
 - RHEL,
 - Centos.
 - 4.2.2. Identyfikowanie problemów pochodzących z obrazu bazowego.
 - 4.2.3. Identyfikowanie problemów wynikających z poleceń użytkownika instalujących pakiety z repozytoriów open source.
 - 4.3. W zakresie wsparcia procesu remediacji podatności:
 - 4.3.1. Skanowanie obrazów z rejestrów i prezentacja podatności.
 - 4.3.2. Skanowanie obrazów z CLI w celu testowania obrazów lokalnie i eksport wyników do interfejsu przeglądarkowego.
 - 4.3.3. Integracja z potokami CI/CD i testowanie podczas tworzenia obrazów.
 - 4.3.4. Wizualizacja drzewa zależności dla każdego projektu, z podatnościami, zależnościami i zagnieżdżonymi problemami.
 - 4.3.5. Rekomendowanie alternatywnego obrazu z mniejszą liczbą problemów.

- 4.3.6. Zapewnienie minimalnych zaleceń dotyczących aktualizacji w oparciu o aktualnie używany obraz bazowy.
- 4.3.7. Aktualizacja obrazu bazowego lub alternatywne zalecenia oparte na skanowaniu Dockerfile.
- 4.3.8. Identyfikacja plików manifestów w obrazach i skanowanie ich w poszukiwaniu zależności open source.
- 4.3.9. Skanowanie obrazów uruchomionych w klastrze Kubernetes i monitorowanie ich bezpieczeństwa, znajdowanie podatnych procesów w klastrach Kubernetes.
- 4.3.10. Regularne monitorowanie zaimportowanych obrazów pod kątem nowo wykrytych podatności, w trybie planowym.
- 4.3.11. Wskazywanie źródła błędu: obraz bazowy, instrukcje użytkownika i ładowane biblioteki.
- 4.3.12. Możliwość poprawiania Dockerfile za pomocą Pull Requestów.
- 4.4. W zakresie bazy danych podatności kontenerów:
 - 4.4.1. Pełna punktacja CVSS dla każdej podatności.
 - 4.4.2. Dokumentowane podatności w bazie danych, w języku angielskim.
 - 4.4.3. Baza danych podatności musi obejmować problemy nieujęte w rejestrze CVE.
 - 4.4.4. Każdy zidentyfikowany problem musi zawierać link do CVE lub innego źródła.

5. FUNKCJA ZABEZPIECZENIA INFRASTRUCTURE as CODE MUSI UMOŻLIWIAĆ:

- 5.1. W zakresie skanowanie plików konfiguracyjnych:
 - 5.1.1. Wykrywanie błędów bezpieczeństwa w plikach konfiguracyjnych Kubernetes.
 - 5.1.2. Wykrywanie błędów bezpieczeństwa w plikach konfiguracyjnych Helm.
 - 5.1.3. Wykrywanie błędów bezpieczeństwa w plikach konfiguracyjnych Docker Composer.
- 5.2. W zakresie wsparcia procesu remediacji podatności:
 - 5.2.1. Wizualne wskazywanie problematycznych linii w kodzie.
 - 5.2.2. Wskazywanie problemów wpływających na zasady wdrażania.
 - 5.2.3. Wykrywanie problemów wpływających na bezpieczeństwo podów.
 - 5.2.4. Zwracanie uwagi na problemy z portami, niezabezpieczone porty, brak ograniczeń portów.
 - 5.2.5. Wskazywanie podatnych ustawień sekretów.
 - 5.2.6. Wykrywanie, czy kontener jest uruchomiony jako root lub z nadmiarowymi uprawnieniami.
 - 5.2.7. Wykrywanie braku ustawionych limitów pamięci, procesora lub innych zasobów.
- 5.3. W zakresie polityk skanowania:
 - 5.3.1. Polityki skanowania muszą być konfigurowalne.
 - 5.3.2. Polityki skanowania muszą pozwalać na skonfigurowanie wagi każdego problemu.

III. Gwarancja i usługi dodatkowe

1. Dostęp do cyfrowych materiałów edukacyjnych (baza wiedzy, filmy szkoleniowe, manuale).
2. Wsparcie wdrożeniowe:
 - 2.1. szkolenia produktowe i administracyjne,
 - 2.2. planowanie i przygotowanie przed wdrożeniem, przegląd warunków wstępnych,
 - 2.3. konfiguracja platformy oraz wszystkich dostarczanych komponentów,
 - 2.4. konfiguracja polityk i priorytetyzacja alertów.
3. Wsparcie edukacji deweloperów w zakresie bezpieczeństwa i adaptacji rozwiązania. Dedykowane warsztaty kickoff oraz dedykowane warsztaty szkoleniowe on-line na wybrany temat z zakresu obsługi platformy (12 warsztatów w ciągu roku).
4. Dostępność dedykowanego inżyniera dostawcy, odpowiedzialnego za realizację umowy.
5. Maksymalny czas rozwiązania zgłoszenia serwisowego z najwyższym priorytetem: 120 minut.
 - 5.1. Zgłoszenia serwisowe z najwyższym priorytetem będą dotyczyły krytycznych awarii, wpływających na bezpieczeństwo wytwarzania oprogramowania CeZ i uniemożliwiających pracę, bez możliwości obejścia.
 - 5.2. Zgłoszenia serwisowe z normalnym priorytetem będą dotyczyły awarii ważnej funkcji usługi ale nie wpływających na bezpieczeństwo wytwarzania oprogramowania CeZ lub bezpieczeństwo wytwarzania oprogramowania będą możliwe dzięki obejściu.
 - 5.3. Zgłoszenia serwisowe z niskim priorytetem będą dotyczyły awarii oprogramowania, mających niewielki lub żaden wpływ na działanie CeZ.
6. Dedykowany kanał w komunikatorze internetowym (komunikacja tekstowa i głosowa)
7. Dostęp online do społeczności klientów (listy dyskusyjne, itp.).
8. Wszystkie usługi mogą być dostarczone zdalnie.