Akademia Górniczo-Hutnicza
Informatyka Techniczna 2021

Bezpieczeństwo systemów informatycznych

Raport z audytu aplikacji RetireEasy

Realizacja: Sebastian Łabuz

Spis treści

1. Podsumowanie

Audyt aplikacji RetireEasy realizowany był w celu ustalenia zgodności ze standardem Application Security Verification Standard (ASVS) 4.0 oraz pod kątem ogólnego bezpieczeństwa aplikacji w wersji testowej.
Ogólny stan zgodności aplikacji RetireEasy ze standardem „Application Security Verification Standard 4.0" można określić jako zły.
Wyróżniono kilka głównych problemów wpływających na niezgodność aplikacji z wymaganiami standardu:

1. Zaimplementowany w niebezpieczny sposób mechanizmy aplikacji (mechanizmy wejścia / wyjścia, haseł, logowania).
2. Brak zaimplementowanych praktyk dotyczących bezpieczeństwa aplikacji oraz informacji sugerowanych przez wymagania standardu.

Informacje o standardzie można znaleźć w dokumencie OWASP Application Security Verification Standard 4.0, wersja 4.0.3, październik 2021.

2. Zakres i cele

2.1. Cel i metodologia audytu

Celem audytu była wstępna weryfikacja spełniania przez aplikację RetireEasy standardu ASVS 4.0 jako standardu wymagań bezpieczeństwa wobec nowoczesnych aplikacji webowych. ASVS 4.0 został wybrany jako najbardziej szeroki i dokładny standard kierunkowany na aplikacje webowe. Wybrana metoda obejmuje zarówno aspekty ściśle implementacyjne i wdrożeniowe aplikacji jak i analizę architektury oraz etapu projektowania.

2.2. Zakres audytu

Audyt objął następujące obszary:

1. Architektura aplikacji.
2. Modelowanie zagrożeń.
3. Metody uwierzytelniania.
4. Metody kontroli dostępu.
5. Zarządzanie sesją.
6. Metody walidacji i oczyszczania danych wejściowych.
7. Obsługa funkcji kryptograficznych.
8. Obsługa błędów.
9. Ochrona danych.
10. Ochrona przed złośliwym kodem.
11. Bezpieczeństwo logiki biznesowej.
12. Bezpieczeństwo plików.
13. Bezpieczeństwo API.
14. Konfigurację.

2.3. Wykonane czynności

W ramach audytu wykonane zostały następujące czynności:

- Analiza błędów występujących w aplikacji,
- Analiza zgodności aplikacji w wersji testowej z wymaganiami ASVS,
- Rozmowy z deweloperami projektu.

2.4. Napotkane ograniczenia

Analizowana aplikacja jest jeszcze na etapie tworzenia. Z tego też powodu wiele wymagań jest niespełnionych i niezaimplementowanych w aktualnej wersji aplikacji. Dodatkowo w aplikacji występuje wiele nieprawidłowości w działaniu mechanizmów aplikacji wymagających przeprowadzenia testów aplikacji przez developerów.

2.5. Aplikacja RetireEasy

Aplikacja RetireEasy jest aplikacją webową, opartą o architekturę klient-serwer wykorzystującą m.in. takie technologie, jak: serwer HTTP , Mongo DB, Node JS.

2.6. Zleceniodawca

Zleceniodawcą audytu była firma RetirementManagers.

3. Testowane komponenty i obszary

Kategoria ASVS	Wymagania		
	Wszystkie	Wyłączone	Niezgodne
V1: Architektura, projektowanie i modelowanie zagrożeń	42	42	0
V2: Uwierzytelnianie	59	13	26
V3: Zarządzanie sesją	20	5	7
V4: Kontrola dostępu	10	4	3
V5: Walidacja, sanityzacja i kodowanie	30	6	12
V6: Kryptografia (dane w spoczynku)	16	2	7
V7: Obsługa błędów i logowanie	14	5	3
V8: Ochrona danych	17	5	4
V9: Komunikacja	8	7	0
V10: Złośliwy kod	10	7	0
V11: Logika biznesowa	8	3	2
V12: Pliki i zasoby	15	15	0
V13: API i webserwisy	15	6	4
V14: Konfiguracja	27	14	5

Wyłączenia wynikają najczęściej z dwóch przyczyn:

1. Braki związane z aktualną fazą implementacji aplikacji (uzgodnione z klientem)
2. Brak potrzeby implementacji pewnych mechanizmów w związku z celem istnienia aplikacji
3. W związku z charakterem testów, nieudostępniona dokumentacja dotycząca procesów firmy, infrastruktur i implementacji aplikacji

Wiele niezgodności wynika z aktualnej wersji aplikacji i zostanie zaimplementowane w finalnej wersji aplikacji. Aktualnie testowane komponenty zawierają błędy implementacyjne, które wymagają przeprowadzenia wewnętrznych testów w celu wykrycia i naprawy. Przed finalnym wdrożeniem aplikacji rekomendowane jest przeprowadzenie jest pełnego audytu aplikacji.

4. Przegląd wyników

4.1. Potencjalny dostęp do serwerów

Poziom ryzyka: KRYTYCZNE

Poprzez błędną implementację mechanizmów modyfikacji danych przez użytkowników możliwa jest ingerencja w działanie aplikacji serwera. Poprzez możliwość zainfekowania serwera skryptami możliwa jest infiltracja danych zawartych w aplikacji, jak i zarówno zasobów infrastruktury serwerów. Może to prowadzić do infekcji wirusowych, co daje dużą gamę możliwości dla potencjalnych intruzów.

Rekomendacje

Poprzez zastosowanie walidacji mechanizmów wejścia można ograniczyć możliwości, np. wprowadzenia złośliwego oprogramowania. Należy również ograniczyć uprawnienia aplikacji (serwera) do minimalnej ilości potrzebnych zasobów (np. bazy danych).

4.2. Możliwość wystąpienia Denial Of Service

Poziom ryzyka: KRYTYCZNE

Poprzez błędną implementację mechanizmów modyfikacji danych przez użytkowników możliwa jest ingerencja w działanie aplikacji serwera. Skutkiem czego może być wgranie szkodliwego skryptu, który spowoduje zablokowanie możliwości serwera do udostępniania swoich usług. Oznaczać to może zablokowanie użytkownikom dostępu do aplikacji.

Rekomendacje

Poprzez zastosowanie walidacji mechanizmów wejścia można ograniczyć możliwości, np. wprowadzenia złośliwego oprogramowania. Dodatkowo można wprowadzić również skrypt, którego zadaniem będzie pilnowanie prawidłowego funkcjonowania serwera.

4.3. Nieprawidłowa implementacja mechanizmu haseł

Poziom ryzyka: WYSOKIE

Mechanizm haseł nie spełnia wymogów bezpieczeństwa stawianych przez standard:

- minimalna długość haseł jest mniejsza niż 12 znaków (1 znak), a maksymalna długość hasła to 20 znaków (aplikacji informuje użytkownika o wymaganym haśle o długości od 6 do 18 znaków),
- użytkownicy nie mogą samodzielnie modyfikować, resetować lub przypomnieć sobie swojego hasła,
- brak weryfikacji wprowadzania różnych znaków (minimum 1 duża litera, mała litera, cyfra i znak specjalny)
- brak jest weryfikacji siły hasła,
- nie istnieje mechanizm informujący użytkownika o zmianach hasła.

4.4. Udostępnianie błędów realizacji kodu

Poziom ryzyka: WYSOKIE

Komunikaty o błędach występujących w aplikacji wyświetlane są użytkownikowi. Komunikaty te mogą służyć intruzom do poznawania implementacji aplikacji.

Rekomendacje
Wystąpienia błędów w aplikacji powinny być zapisywane na serwerach z autoryzowanym dostępem, a w ostateczności w plikach. Logi powinny służyć tylko i wyłącznie developerom. Użytkownika powinno się jedynie poinformować o ewentualnym błędzie (jeśli tego oczekuje użytkownik), jednak bez podawania szczegółów.

4.5. Ryzyko wycieku danych użytkowników / brak szyfrowania istotnych danych

Poziom ryzyka: WYSOKIE

Poprzez błędną implementację mechanizmów modyfikacji danych przez użytkowników możliwa jest ingerencja w działanie aplikacji serwera. Intruz może uzyskać dostęp do danych użytkowników. Brak szyfrowania danych również może spowodować wyciek danych użytkownika. Intruz obserwujący komunikacje może przechwycić wrażliwe dane.

Rekomendacje
Najlepszym rozwiązaniem byłoby zaimplementowanie algorytmów szyfrujących oraz wykorzystywanie tych algorytmów dla danych wrażliwych użytkownika.

4.6. Błędna implementacja logowania użytkowników

Poziom ryzyka: ŚREDNIE

Panel logowania umożliwia poznawanie nazw użytkowników istniejących w systemie. Oprócz tego dostępne jest również konto „admin”. Nie istnieje również żaden mechanizm blokujący już wspomniane „skanowanie” systemu. Aplikacja nie proponuje również weryfikacji wieloetapowej.

Rekomendacje
Konta administratorskie nie powinny być dostępne z poziomu aplikacji udostępnianej publicznie. Nie należy również informować w sposób szczegółowy o wystąpieniu błędu logowania. Wielokrotne błędne wprowadzanie danych powinno skutkować weryfikacją użytkownika oraz tymczasowym zablokowaniem dostępu do usługi dla tego użytkownika. Aplikacja powinna umożliwiać weryfikacje wieloetapową, np. poprzez SMS, e-mail, klucze dostępu.

5. Załączniki

1. Szczegółowy wykaz wymagań OWASP ASVS 4.0 z wynikiem audytu dla każdego wymagania.