Raport z audytu aplikacji e-commerce-food-shop-with-qr-codes

Wersja raportu: 1.0
Data wykoania audytu: 11.01.2022
Wykonali: Tomasz Wańczyk, Karol Niekurzak

1. Podsumowanie

Audyt aplikacji realizowany był jako audyt zgodności ze standardem Application Security Verification Standard 4.0.3 jak również pod kątem ogólnego bezpieczeństwa aplikacji.

Ogólny stan zgodnośći aplikacji ze standardem „Application Security Verification Standard 4.0.3" można określić jako Słaby.

Wyróżniono kilka głównych problemów wpływających na niezgodność aplikacji z wymaganiami standardu:

- Nieprawidłowa implementacja mechanizmu haseł
- Problem z Ogólnymi zabezpieczeniami uwierzytelniania
- Credential Recovery
- Konfiguracja ochrony danych
- Brak ograniczeń przepływów logiki biznesowej
- Sposób weryfikacji plików i zasobów
- Niedostateczna Konfiguracja zabezpieczeń
- Braki w implementacji związanej z Session Binding (Implement Digital Identity)
- Braki w zarządzaniu sesją w oparciu o pliki cookie
- Braki w obronie przeciwko lukom w zarządzaniu sesją
- Błędnie zaimplementowany system do obłsugi błędów i wyjątków
- Brak kilku etapowej walidacji przy użyciu panelu administratora:
- Brak ostrzerzeń przy klikaniu w link URL
- Błędy w ochronie przed OS command injection

2. Zakres i cele

2.1 Cel i metodologia audytu

Celem przeprowadzenia audytu była weryfikacja spełnienia standardu bezpieczeństwa ASVS 4.0 na poziomie 1 wobec nowoczesnych aplikacji webowych przez aplikacje e-commerce-food-shop-with-qr-codes.

ASVS 4.0 został wybrany ze względu na najbardziej szeroki zakres bezpiczeństwa wobec aplikacji webowych. Poziom 1 został wybrany dla aplikacji ze względu na jej stan deweloperski.

2.2 Zakres Audytu

Audyt objął następujące obszary :

Audyt został przeprowadzony zgodnie z Level 1 ASVS

V2. Uwierzytelnianie

V3. Zarzadzanie sesjami

V4. Kontrola dostępu

V5. Walidacja, sanityzacja i kodowanie

V7. Obsługa błędów i logowanie

V8. Ochrona danych

V9. Komunikacja

V10. Złośliwy kod

V11. Logika biznesowa

V12. Pliki i zasoby

V13. Api i webserwisy

V14. Konfiguracja

2.3 Wykonane czynności

W ramach audytu zostały wykonane nastepujące czynności :

- Analiza kodu źródłowego aplikacji oraz wybranych bibioltek przez nia wykrozystywanych

- Analiza dokumentacji wykorzystywanych frameworków

- Testy na uruchomionej lokalnie aplikacji

- Konsultacje z deweloperami aplikacji

2.4 Napotkane ograniczenia

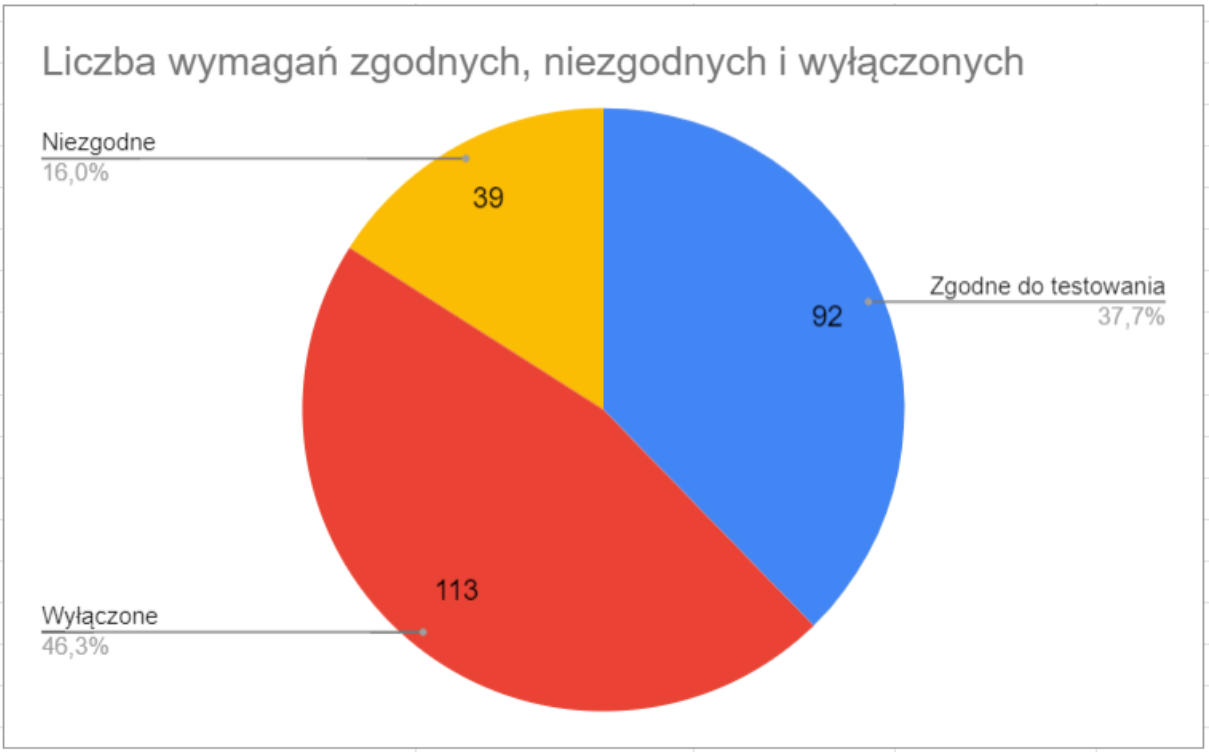
Analizowana aplikacja jest dostępna tylko w formie developerskiej. Nie została zahostowana na żadnym z portali co uniemożliwa przetestowanie niektórych punktów kontrolnych dostępnych tylko dla aplikacji w fazie produkcyjnej, gdyż są zależne od sposobu wdrożenia aplikacji. Dodatkowo aplikację przetestowano aby sprawdzić czy jest zgodna z Level 1 ASVS. Tym samym wiele z wymagań zostało wyłączonych (w tym cała kategoria V1: Architektura, projektowanie i modelowanie zagrożeń)

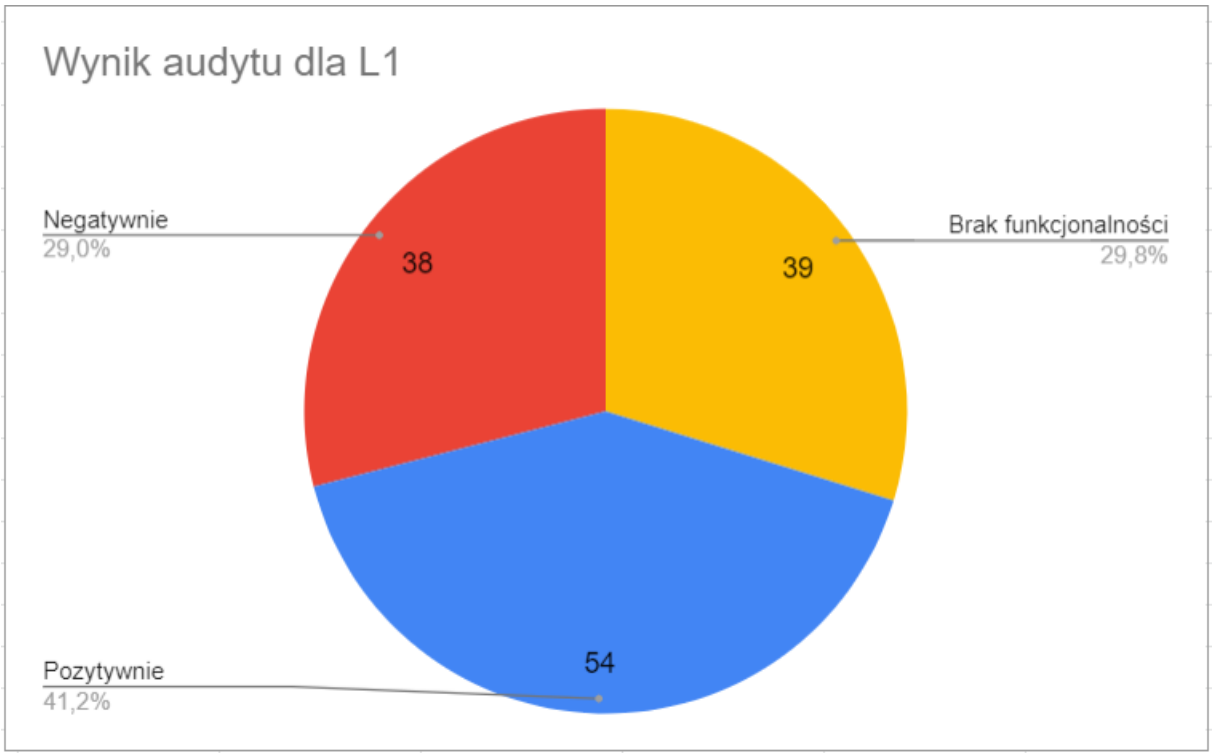
2.5 Aplikacja

Aplikacja e-commerce-food-shop-with-qr-codes jest aplikacją webową opartą o architekturę typu MVT (Model-View-Template). Jako backend aplikacji wykorzystano popularny framework Django (python), frontend zaś został utworzony przy pomocy Bootstrap'a i dodatkowych styli CSS. Użyta baza danych to PostgreSQL.

3. Testowanie komponentów i obszarów

Kategoria ASVS	Wymagania		
	Wszystkie	Wyłączone	Niezgodne
V2: Uwierzytelnianie	57	30	5
V3: Zarządzanie sesją	20	8	1
V4: Kontrola dostępu	10	1	0
V5: Walidacja, sanityzacja i kodowanie	30	3	11
V6: Kryptografia (dane w spoczynku)	16	15	1
V7: Obsługa błędów i logowanie	13	9	1
V8: Ochrona danych	17	11	2
V9: Komunikacja	8	5	3
V10: Złośliwy kod	10	7	2
V11: Logika biznesowa	8	3	0
V12: Pliki i zasoby	15	4	6
V13: API i webserwisy	15	8	4
V14: Konfiguracja	25	9	3




Wynik audytu dla L1
Negatywnie
29,0%
38
39
Brak funkcjonalności
29,8%
54
Pozytywnie
41,2%

4. Przegląd wyników i zaleceń

4.1. Nieprawidłowa implementacja mechanizmu haseł

Poziom ryzyka: Wysoki

Mechanizm haseł ma sporo braków w kontekście zabezpieczeń:

- minimalna długość hasła jest mniejsza niż 12 znaków,
- hasło może składać się z ponad 64 znaków, ale dozwolone są również hasła ponad 128 znakowe,
- +hasła nie są obcinane jeśli zawierają spacje, -kilka spacji nie jest zamienane na jedną,
- użytkownicy nie mogą zmienić hasła po zalogowaniu, mogą je jedynie zresetować przed zalogowaniem, jeśli zapomnieli hasła.
- hasła są częściowo porównywane z pulą najczęściej używanych haseł, jednak nie wykorzystano w tym przypadku zewnętrznego API
- brak max limitu długości hasła, wymogi to min. 8 znaków, nie może się składać z samych cyfr,
- użytkownik nie może wyświetlić wpisywanego hasła w formularz.

Zalecenia:

Należy zmodyfikować mechanizm haseł tak aby spełniał takie wymagania jak:

- minimalna długość hasła powinna wynosić 12 znaków,
- hasło nie powinno być dłuższe niż 128 znaków,
- jeśli w haśle znajdują się spacje, kilka spacji powinno być sklejane w jedną,
- dodać mechanizm zmiany hasła po zalogowaniu (dostępny jest tylko reset hasła przed logowaniem), taki mechanizm powinien wymagać podania starego hasła.

4.2. Ogólne zabezpieczenia uwierzytelniania

Poziom ryzyka: Średni

System zabezpieczający uwierzytelnianie posiada braki:

- brak implementacji CAPTCHA,
- konto nie jest blokowane przy masowych próbach ataku (brute force)
- brak weryfikacji email przy utworzeniu konta - użytkownik jedynie dostaje wiadomość mailową bez konieczności potwierdzenia założenia konta

Zalecenia:

- dodać do mechanizmu logowania/rejestracji CAPTCHA,
- zaimplementować mechanizm, który będzie blokował konto po przekroczeniu danej ilości prób logowania (np. 15, 100), w celu uchronienia przed metodami takimi jak Brute Force,
- dodać wymóg zweryfikowania konta po jego rejestracji.

4.3. Credential Recovery

Poziom ryzyka: Średni

- domyślne konto administratora jest dostępne, jednak wynika to z faktu, że aplikacja jest tylko w trybie developerskim.
- użytkownik nie dostaje informacji e-mail po pomyślnym zresetowaniu hasła.

Zalecenia:

- wyłączyć domyślne konto administratora i wprowadzić nową grupę użytkowników z podobnymi uprawnieniami,
- dodać wysyłanie wiadomości e-mail po pomyślnym zresetowaniu hasła.

4.4. Konfiguracja ochrony danych

Poziom ryzyka: Niski

- dane uwierzytelniające nie są usuwane po zakończeniu sesji
- brak informacji dla użytkowników o sposobie wykorzystania ich danych
- brak przedstawienia polityki wykorzystywania wrażliwych danych

Zalecenia :

- usuwanie danych uwierzytelniajacych po zakończenu sesji
- przed wypuszczeniem aplikacji na etap produkcyjny należy określić jasną politykę wykorzystywania poszczególnych danych
- poinformowanie użytkownika które z danych są wykorzystywane

4.5. Brak ograniczeń przepływów logiki biznesowej

Poziom ryzyka: Średni

- brak limitu dla działań biznesowych i transkacji.
- brak mechanizmu chroniącego przed nadmiernymi połączeniami.
- brak weryfikacji czy kroki są przetwarzane w realistycznym czasie ludzkim.
- brak dodatkowej weryfikacji przy składaniu zamówienia.

Zalecenia :

- przed wypuszczeniem aplikacji na etap produkcyjny należy dodać limity czasowe na przeprowadzanie zamówień
- ograniczenie maksymalnej liczby zamówień przez jednego użytkownika w określonym czasie do prawdopodobnej wartości
- dodatkowa weryfikacja użytkownika przy składaniu zamówienia np. reCAPTCHA

4.6 Weryfikacja plików i zasobów

Poziom ryzyka: Niski

- brak skanera antywirusowego dla dodawanych zasobów.
- możliwość wykonywania przesłanej zawartości jako HTML/JavaScript ale tylko z poziomu admina.

Zalecenia :

- poprawa weryfikacji zasobów
- zablokowanie możliwości wykonywania żadań z przesłanych plików.

4.7 Niedostateczna Konfiguracja zabezpieczeń

Poziom ryzyka: Wysoki

- składniki aplikacji wymagają uaktualnienia.
- nie wszystkie niepotrzebne funkcje i konfiguracje zostały usunięte.
- nagłówek Strict-Transport-Security nie pojawia się we wszystkich odpowiedziach.
- brak nagłowka Referrer-Policy.
- nagłówek Cross-Origin Resource Sharing nie ma określonej listy zaufanych domen.

Zalecenia :

- Zmiana defaultowych wartości dla komponentów middleware dostarczanych wraz z Django pozwalających na uchronienie się na powyższe zagrożenia.
- usunięcie zbędnych/testowych/przykładowych konfiguracji i funkcji przed etapem produkcyjnym

4.8. Braki w implementacji związanej z Session Binding (Implement Digital Identity)

Poziom ryzyka: Niski

Wykryto następujące braki w implementacji:

- tokeny sesji składają się z 32 bitów,

Zalecenia:

- Poprawić implementację i zwiększyć liczbę bitów w tokenach sesji. W tym przypadku ryzyko nie jest tak duże, ponieważ Django dobrze chroni przed atakami sesji.

4.9. Braki w zarządzaniu sesją w oparciu o pliki cookie

Poziom ryzyka: Średni

Wykryto następujące braki:

- brak ustawionych atrybutów HttpOnly w ciasteczkach, co pozwala na dostęp do nich przez JavaScript,
- brak ustawionych atrybutów SameSite,
- brak przedrostka "__Host-". Zalecenia:
- W pliku settings.py dodać ustawienia:
    - SESSION_COOKIE_HTTPONLY = True,
    - SESSION_COOKIE_SAMESITE = True,
    - SESSION_COOKIE_NAME = '__Host-'.

4.10. Braki w obronie przeciwko lukom w zarządzaniu sesją

Poziom ryzyka: Średni

- Aplikacja nie wymaga re-autentykacji lub dwu etapowej weryfikacji przed wprowadzeniem wrażliwych danych (adres dostawy, adres email).

Zalecenia:

- Poprawić implementację po wprowadzeniu adresu email podczas rejestracji poprzez konieczność potwierdzenia założenia konta, oraz dodać wysyłanie e-mail'a po dodaniu adresu dostawy.

4.11 Błędnie zaimplementowany system do obłsugi błędów i wyjątków

Poziom ryzyka: Krytyczny

Wykryto następujące błędy w systemie do obsługi błędów i wyjątków:

- mechanizm wysyłania emaili jest zahardkodowany, co powoduje błędy aplikacji, jeśli ustawienia nie zostaną zmienione. Jeśli skrzynka z której wysyłane są wiadomości do użytkowników, którzy tworzą konto w serwisie wymaga podwójnej walidacji, aplikacja wyrzuci błąd.

Zalecenia:

- Dodać odpowiedni mechanizm obsługi wyjątków.

4.12 Brak kilku etapowej walidacji przy użyciu panelu administratora:

Poziom ryzyka: Średni

- Logowanie oraz modyfikowanie aplikacji w panelu administratora nie wymaga kilku etapowej walidacji.

Zalecenia:

- Zaimplementować system obsługujący kilku-etapową walidację dla systemu administratora, np. potwierdzenie e-mail lub kod SMS.

4.13 Brak ostrzerzeń przy klikaniu w link URL

Poziom ryzyka: Niski

- W panelu administratora widnieje link do serwisu map google w celu nawigacji do klienta, jednak po jego kliknięciu nie widnieje żadne zagrożenie, a adres do którego on przekierowuje nie znajduje się na white-liście.

Zalecenia:

- Dodać serwis map google do white-listy.

4.14 Błędy w ochronie przed OS command injection

Poziom ryzyka: Wysoki

- Przy dodawaniu produktów do koszyka zauważono lukę w bezpieczeństwie, która pozwala na wykonanie ataku OS command injection.

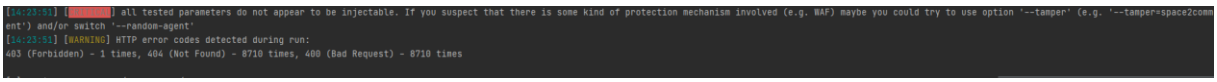
Zalecenia:

- Dodać walidację do formularza, gdzie podawane jest food_id.

5. Sprawdzone zostało również

1. Sql injection

Jedna z częstrzych i niebezpiecznych podatności w aplikacjach webowych. W aplikacji wykorzystywana jest baza danych postgresql będąca jedną z najbardziej popularnych wyborów. W ramach przeprowadzenia testu wykorzystane zostało narzędzie Sqlmap pozwalające na automatyczne przeprowadzenie testów. W testowanej aplikacji mimo kilku prób również z wykorzystaniem level=5 risk=3 nie wykryto podatności na SQLInjection. Powodem odporności jest wykorzystanie frameworka Django (Python) w którym zapytania są konstruowane przy użyciu parametryzacji zapytań. Kod SQL zapytania jest definiowany niezależnie od parametrów zapytania. Parametry dostarczone przez użytkownia są pomijane przez podstawowy sterownik bazy danych.

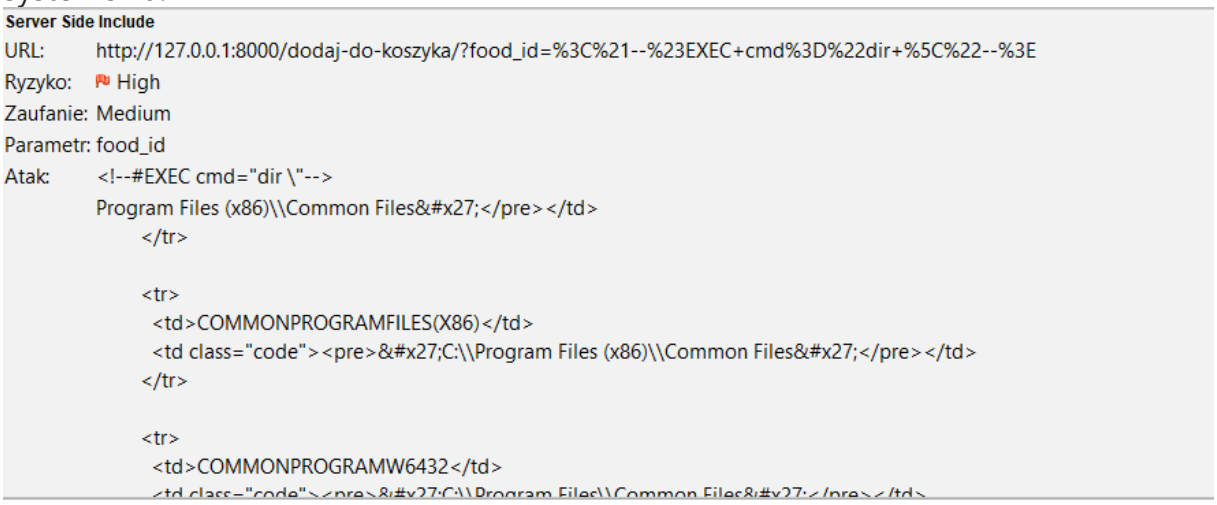


2. OS command injection

Poziom ryzyka: Wysoki

Jest to luka w zabezpieczeniach sieci Web, która umożliwia atakującemu wykonanie dowolnych poleceń systemu operacyjnego (OS) na serwerze, na którym uruchomiona jest aplikacja, i zazwyczaj w pełni naraża aplikację i wszystkie jej dane. Bardzo często atakujący może wykorzystać podatność na wstrzyknięcie polecenia systemu operacyjnego, aby złamać inne części infrastruktury hostingowej, wykorzystując relacje zaufania do skierowania ataku na inne systemy w organizacji.

OS Command Injection przeprowadzono za pomocą OWASP Zap Proxy. Jak się okazało, w jedym z formularzy brakuje walidacji wprowadzanych danych, co pozwala na użycie tej metody. Miejsce w którym można je wykonać to dodawanie do koszyka produktów i w miejsce food_id wstawiana jest komenda systemowa.



Zalecenia:

Dodać walidację do formularza, gdzie podawane jest food_id.

6. Załączniki

Wraz z raportem audytu załączono raport z jednej sesji programu OWASP Zap Proxy, arkusz excel w którym sprawdzano checklistę ASVS level 1 oraz obrazy wykresów i tabeli.
https://github.com/KISiM-AGH/projekt-zaliczeniowy-projekt-zaliczeniowy-wt_nk