Sprawozdanie z projektu Ochrona Danych

Aplikacja została wykonana przez Huberta Dryja.

Funkcjonalności aplikacji

- restrykcyjna weryfikacje danych pochodzących z formularza login-hasło
- przechowywanie hasła chronione funkcją hash i solą
- możliwość umieszczenia na serwerze notatek dostępnych publicznie lub dla określonych użytkowników
- zabezpieczenie transmisji poprzez wykorzystanie protokołu HTTPS
- możliwość zmiany hasła
- sprawdzanie siły hasła i jakości hasła (jego entropii)
- wielokrotne wykorzystanie funkcji hash, żeby wydłużyć ataki brute-force na hash (50 000 razy)
- dodatkowa kontrola spójności sesji (przeciw atakom XSRF)
- weryfikacja liczby nieudanych prób logowania
- dodanie opóźnienia przy weryfikacji hasła w celu wydłużenia ataków zdalnych

Zabezpieczenia

Moja aplikacja została zabezpieczona w wielu miejscach. Wykorzystałem

model wielopoziomowej ochrony w głąb.

Wybrane sposoby zabezpieczeń, których użyłem:

Zabezpieczenie	**Powód/efekt**
Ciasteczka są przesyłane tylko przy połączeniu HTTPS	Znacząco utrudnienie podsłuchania danych, które zawiera ciasteczko
Strona pozwala na połączenie tylko przez HTTPS	Znacząco utrudnienie podsłuchania transmisji
Ustawienie PERMANENT_SESSION_LIFETIME na 60 minut	Utrudnienie ataków, które polegają na przechwyceniu sesji
Przechowywanie informacji o sesji po stronie serwera	Utrudnienie przechwycenia danych i zablokowanie ich podmiany
Do formularzy dodany csrf token	Zablokowanie ataków CSRF
Dodanie nagłówka HTTP Strict-Transport-Security	Wszystkie zapytania HTTPS, zabezpieczenie przed man-in-the-middle
Dodanie nagłówka HTTP Content-Security-Policy	Zdefiniowanie miejsc z których mogą być ładowane zasoby, zabezpieczenie przed XSS
Dodanie nagłówka HTTP X-Content-Type-Options	Blokada detekcji content-type przez przeglądarkę, zabezpieczenie przed XSS
Dodanie nagłówka HTTP X-Frame-Options	Zablokowanie umieszczania strony w elemencie iframe, zabezpieczenie przed ‘clickjacking’

Dodanie nagłówka HTTP X-XSS-Protection	Zabezpieczenie przed XSS, nie pozwala na wyświetlanie zapytania jeśli wygląda na to że w żądaniu pojawił się Javascript
Użycie Jinja	Auto-escape elementów HTML, zablokowanie możliwości ich wyświetlania
Użycie HTML escape przy zapisie do bazy	W przypadku dostępu do danych wiemy że nie zostaną wyświetlone elementy HTML
Użycie ‘questionmarks’ w przypadku zapytań SQL	Zabezpieczenie przed SQL Injection
Opóźnienie logowania (0.5s) w przypadku gdy nieudane	Spowolnienie ataku na hasło
Użycie wyrażenia regex do sprawdzenia poprawności email	Pewność, że to co podaje użytkownik jest adresem email
Generowanie UUID dla każdej notatki	Brak możliwości przewidzenia ile notatek znajduje się w bazie, ani szybkiego znalezienia notatek publicznych bez posiadania linka
Wyświetlanie 404 nawet gdy notatka istnieje ale użytkownik nie ma do niej dostępu	Zaciemnienie informacji o UUID notatek prywatnych
Wymaganie silnego hasła	Zwiększenie czasu do złamania hasła przy użyciu ataku brutal-force

Algorytm hashowania hasła

Do hashowania hasła został użyty algorytm **PBKDF2-HMAC-SHA256**.
Każdy hash zawiera informację jakim algorytem został utworzony. Dzięki temu przy zmianie algorytmu w przyszłości wiem, które hashe dalej korzystają ze starej metody.
Hashe generowane są z losową solą, oraz mają dużą ilość iteracji (w moim przypadku jest to 50 000).
Schemat działania algorytmu jest następujący HMAC(pwd, HMAC(pwd, … HMAC(pwd, salt + number)…).