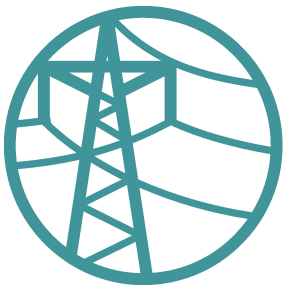

WYDZIAŁ
ELEKTROTECHNIKI
I INFORMATYKI
POLITECHNIKI RZESZOWSKIEJ

Dawid Pnidara

Bezpieczeństwo danych w aplikacjach webowych

Praca dyplomowa magisterska

Opiekun pracy:
dr inż. Bogdan Kwiatkowski

Rzeszów, 2024

Spis treści

1. Wstęp **5**
1.1. Cel i zakres pracy 5
2. Zagrożenia w aplikacjach webowych **5**
2.1. Ataki i incydenty bezpieczeństwa 5
2.2. Przykłady znanych ataków 6
3. Mechanizmy zabezpieczeń w aplikacjach webowych **7**
3.1. Autoryzacja i Uwierzytelnianie 8
3.1.1. Rola autoryzacji i uwierzytelniania 8
3.1.2. Metody uwierzytelniania 9
3.2. Przetwarzanie danych wejściowych 9
3.2.1. Zagrożenia związane z danymi wejściowymi 10
3.2.2. Walidacja danych 11
3.3. Protokół HTTPS 12
3.4. Nawiązywanie połączenia przez protokół HTTPS 13
4. Metody Testowania Bezpieczeństwa w Aplikacjach Webowych . . . **14**
4.1. Rodzaje testów bezpieczeństwa 14
4.2. Narzędzia do przeprowadzania testów bezpieczeństwa 15
4.3. Penetrowanie Aplikacji Webowych 17
4.3.1. Cel i zakres testów penetracyjnych 17
4.3.2. Praktyki bezpiecznego penetrowania 17
5. Technologie wykorzystane do utworzenia aplikacji **17**
5.1. JavaScript 17
5.2. TypeScript 17
5.3. React 18
5.4. Node.JS 18
6. Efekty pracy **19**
7. Podsumowanie **19**
Literatura **19**

1. Wstęp

W dzisiejszym świecie, aplikacje webowe stały się fundamentalnym narzędziem wykorzystywanym przez praktycznie każde przedsiębiorstwo, instytucję ale również i zwykłe osoby. Służące przy rozmaitych usługach online takich jak komunikacja czy handel, aplikacje webowe stały się bardzo popularne co niestety wiąże się również ze wzrostem zagrożeń związanych z bezpieczeństwem danych. Bardzo często można przeczytać lub usłyszeć o incydentach związanych z naruszeniami bezpieczeństwa, wyciekami danych jak i wielu innych atakach. Przykładem zagrożeń takich jak ataki SQL Injection, Cross-Site Scripting (XSS), Cross-Site Request Forgery (CSRF) czy ataki typu zero-day są stale obecne i stanowią poważne ryzyko dla organizacji, instytucji oraz użytkowników końcowych.

1.1. Cel i zakres pracy

Celem pracy jest zgłębienie problematyki bezpieczeństwa danych w aplikacjach webowych poprzez utworzenie aplikacji webowej na której zostaną przeprowadzone testy penetracyjne, odporności na ataki oraz obsługi błędów. Wykazane błedy wykryte podczas testów zostaną następnie użyte jako podstawa do poprawienia działania aplikacji by ponownie przeprowadzić testy i porównać wyniki.

2. Zagrożenia w aplikacjach webowych

Aplikacje webowe są narażone na różnorodne ataki, takie jak SQL Injection, gdzie intruz wstrzykuje złośliwy kod SQL, czy Cross-Site Scripting (XSS), umożliwiający wstrzykiwanie skryptów do przeglądarki użytkownika. Dodatkowo, ataki typu Cross-Site Request Forgery (CSRF) mogą prowadzić do nieautoryzowanych akcji w imieniu zalogowanego użytkownika. Zagrożenia te podkreślają potrzebę skutecznych mechanizmów zabezpieczeń. Wzrost ataków zero-day dodaje do tego wyzwanie, wymagając bieżącej analizy i reakcji na nowe, jeszcze nieznane zagrożenia.

2.1. Ataki i incydenty bezpieczeństwa

Zagrożenia w aplikacjach webowych to różnego rodzaju ataki i incydenty, które mogą naruszyć bezpieczeństwo danych, prywatność użytkowników i integralności aplikacji. Jako powszechne metody ataków można wymienić:

- **SQL Injection (SQLi):** Atak polegający na wstrzykiwaniu złośliwego SQL do danych wejściowych, co może prowadzić do dostępu do bazy danych osób nie autoryzowanych i wycieku lub modyfikacji danych.

- **Cross-Site Scripting (XSS):** Atak, w którym atakujący wstrzykuje złośliwy kod JavaScript do treści strony, który jest wykonywany przez przeglądarkę użytkownika. To może prowadzić do kradzieży sesji czy przechwycenia danych użytkownika.

- **Cross-Site Request Forgery (CSRF):** Atak polegający na wymuszeniu wykonania nieautoryzowanych działań w imieniu zalogowanego użytkownika poprzez wykorzystanie podrobionego żądania HTTP.

- **Naruszenie tożsamości:** Atak na procesy uwierzytelniania i zarządzania tożsamością użytkowników, co może prowadzić do nieautoryzowanego dostępu do kont użytkowników.

- **Ataki na aplikacje mobilne:** Ataki na aplikacje na urządzenia mobilne, w tym próby manipulacji kodem źródłowym, reverse engineering i nieautoryzowane dostępy.

- **Phishing:** Ataki polegające na podszywaniu się pod zaufane źródło, aby wyłudzić dane użytkowników, takie jak hasła czy dane finansowe.

- **Ataki typu Zero-Day:** Wykorzystanie luk w oprogramowaniu lub systemach operacyjnych, których dostawca jeszcze nie zdążył naprawić.

2.2. Przykłady znanych ataków

- **Incydent Equifax (2017):** Equifax, jedna z trzech głównych agencji oceniających zdolność kredytową, doświadczyła ataku hakerskiego, w wyniku którego wyciekły dane osobowe około 147 milionów osób. Wyciek obejmował informacje takie jak numery ubezpieczenia społecznego, daty urodzenia i informacje kredytowe.

- **Atak na Yahoo (2013-2016):** W latach 2013-2016, Yahoo doświadczyło dwóch masowych ataków hakerskich, w wyniku których skradziono dane ponad mi-

liarda kont użytkowników. Wyciekły informacje takie jak nazwy, adresy e-mail, hasła i pytania zabezpieczające.

- **Atak na Target (2013):** Atak hakerski na sieć sklepów Target spowodował wyciek danych ponad 40 milionów kart kredytowych. Atakujący uzyskali dostęp do systemu punktów sprzedaży, gdzie przechowywane były dane płatnicze klientów.

- **Wyciek danych Facebook-Cambridge Analytica (2018):** Firma Cambridge Analytica pozyskała dane od około 87 milionów użytkowników Facebooka bez ich zgody. Dane te zostały następnie wykorzystane do personalizacji treści politycznych w czasie kampanii wyborczych.

- **Atak na Sony Pictures Entertainment (2014):** Atak hakerski na Sony Pictures spowodował wyciek dużych ilości poufnych informacji, w tym e-maili, dokumentów, danych finansowych i informacji osobistych pracowników.

- **Incident Capital One (2019):** Pracownica firmy Capital One włamała się do systemów banku, powodując wyciek danych z ponad 100 milionów klientów. W skradzionych danych znalazły się m.in. numery kont bankowych i informacje dotyczące kredytów.

- **Wyciek danych Experian-T-Mobile (2015):** Firma Experian, agencja oceniająca zdolność kredytową, doświadczyła ataku, w wyniku którego skradziono dane ponad 15 milionów klientów T-Mobile. Wyciekły informacje takie jak numery społeczne, imiona i nazwiska.

3. Mechanizmy zabezpieczeń w aplikacjach webowych

Mechanizmy zabezpieczeń w aplikacjach webowych są kluczowym elementem w ochronie przed różnorodnymi zagrożeniami. Rola autoryzacji polega na kontrolowaniu, co użytkownik może zrobić na podstawie nadanych mu uprawnień. Uwierzytelnianie potwierdza tożsamość użytkownika za pomocą loginu, hasła czy też zaawansowanych metod, takich jak dwuskładnikowa autentykacja.

Kontrola dostępu i zarządzanie tożsamością to kolejne istotne aspekty. Skupiają się na określaniu, kto ma dostęp do jakich zasobów i w jaki sposób. Single Sign-On

(SSO) ułatwia użytkownikom korzystanie z wielu usług bez konieczności wielokrotnego logowania, ale wymaga odpowiednich zabezpieczeń.

Przetwarzanie danych wejściowych to obszar szczególnie narażony na ataki. Skuteczna walidacja danych, czyli sprawdzenie, czy są one zgodne z oczekiwanym formatem, jest kluczowym krokiem w zapobieganiu atakom, takim jak SQL Injection czy Cross-Site Scripting. Wykorzystanie parametrów bezpieczeństwa, takich jak używanie przygotowanych instrukcji SQL, wzmacnia ochronę przed manipulacją danymi.

3.1. Autoryzacja i Uwierzytelnianie

Uwierzytelnianie jest procesem potwierdzania tożsamości użytkownika. Wykorzystuje różne metody, takie jak loginy, hasła, a także bardziej zaawansowane techniki, np. biometrię czy dwuskładnikową autentykację. Przez to użytkownik musi dostarczyć pewne unikalne dane, aby uzyskać dostęp do systemu.

Autoryzacja koncentruje się na kontrolowaniu tego, co dany użytkownik może robić po poprawnym uwierzytelnieniu. Przydzielane są mu określone uprawnienia, które decydują, do których zasobów czy funkcji ma dostęp. To ograniczenie dostępu zabezpiecza przed nieuprawnionym korzystaniem z aplikacji.

3.1.1. Rola autoryzacji i uwierzytelniania

Rola autoryzacji i uwierzytelniania w aplikacjach webowych jest istotna dla zapewnienia bezpieczeństwa i ochrony zasobów. Jak już wcześniej wspomniano uwierzytelnianie, służy do potwierdzenia tożsamości użytkownika, wymagając dostarczenia unikalnych danych, takich jak login i hasło, bądź wykorzystując bardziej zaawansowane metody. Dzięki temu system może jednoznacznie zidentyfikować, kto próbuje uzyskać dostęp.

Po poprawnym uwierzytelnieniu pojawia się rola autoryzacji, która decyduje o przyznawaniu uprawnień użytkownikowi. Uprawnienia te określają, do których zasobów, funkcji czy danych użytkownik ma dostęp. Autoryzacja zapobiega nieautoryzowanemu korzystaniu z aplikacji oraz kontroluje, jakie czynności może wykonywać dany użytkownik.

W praktyce, kombinacja tych dwóch mechanizmów pozwala na skonstruowanie bezpiecznego środowiska. Jednak rola uwierzytelniania i autoryzacji nie kończy się na prostym potwierdzeniu tożsamości i przyznaniu uprawnień. Istotne jest także świa-

dome zarządzanie danymi uwierzytelniającymi, takimi jak hasła, poprzez zastosowanie bezpiecznych praktyk, np. haszowanie.

Wprowadzenie dwuskładnikowej autentykacji, monitorowanie aktywności użytkowników czy stosowanie silnych polityk haseł to dodatkowe elementy, które podnoszą poziom bezpieczeństwa. Rola autoryzacji i uwierzytelniania jest zatem nie tylko technicznym aspektem, ale także strategicznym, wpływającym na całkowite bezpieczeństwo aplikacji webowej i danych w niej przechowywanych.

3.1.2. Metody uwierzytelniania

- **Login i hasło:** Standardowa metoda, jednak wymaga odpowiednich praktyk, takich jak silne hasła czy haszowanie.

- **Dwuskładnikowa autentykacja (2FA):** Dodatkowy poziom zabezpieczeń, wykorzystujący co najmniej dwie różne metody uwierzytelniania, na przykład hasło i kod generowany na urządzeniu mobilnym.

- **Biometria:** Wykorzystanie unikalnych cech fizycznych, takich jak odciski palców, rozpoznawanie twarzy czy skanowanie siatkówki oka.

- **Tokeny autoryzacyjne:** Generowane kody czasowe, które są używane do jednorazowego uwierzytelniania. Zwiększają bezpieczeństwo, eliminując potrzebę przechowywania długotrwałego hasła.

- **Single Sign-On (SSO):** Umożliwia użytkownikowi zalogowanie się raz i uzyskanie dostępu do wielu powiązanych usług bez ponownego logowania.

- **OAuth i OpenID Connect:** Standardy protokołów uwierzytelniania dla aplikacji webowych, pozwalające na bezpieczne udostępnianie zasobów i uwierzytelnianie.

- **FIDO2 (Fast Identity Online):** Nowoczesna technologia oparta na kluczach publicznych, eliminująca konieczność korzystania z haseł i zapewniająca bezpieczne uwierzytelnianie.

3.2. Przetwarzanie danych wejściowych

Przetwarzanie danych wejściowych w aplikacjach webowych jest ważnym elementem dla zapewnienia bezpieczeństwa i integralności systemu. Jak już wcześniej

wspominano zagrożenia związane z danymi wejściowymi, takie jak ataki SQL Injection, Cross-Site Scripting (XSS) czy Cross-Site Request Forgery (CSRF), wymagają skutecznego przeciwdziałania.

Zaczynając od identyfikacji zagrożeń, należy dokładnie analizować każdy rodzaj danych wejściowych i potencjalne ryzyko związane z manipulacją nimi. Skuteczna walidacja danych to kluczowy krok, który zapewnia, że dane są zgodne z oczekiwanym formatem, eliminując tym samym możliwość wstrzykiwania złośliwego kodu.

3.2.1. Zagrożenia związane z danymi wejściowymi

Do zagrożeń związanych z danymi wejściowymi można zaliczyć:

- **SQL Injection (SQLi):** Wcześniej wspomniany atak polegający na wstrzykiwaniu złośliwego SQL do danych wejściowych, co może prowadzić do dostępu do bazy danych i wycieku lub modyfikacji danych.

- **Cross-Site Scripting (XSS):** Również wcześniej podany atak, w którym atakujący wstrzykuje złośliwy kod JavaScript do treści strony, który jest wykonywany przez przeglądarkę użytkownika. To może prowadzić do kradzieży sesji, przechwycenia danych użytkownika i innych ataków.

- **Cross-Site Request Forgery (CSRF):** Jak już wcześniej wspomniano atak ten polegaja na wymuszeniu wykonania nieautoryzowanych działań w imieniu zalogowanego użytkownika poprzez wykorzystanie podrobionego żądania HTTP.

- **Injection Attacks:** Obejmują również ataki takie jak LDAP Injection czy OS Commanding, gdzie intruz próbuje wstrzyknąć złośliwy kod do innych obszarów systemu.

- **Data Validation Bypassing:** Próby obejścia mechanizmów walidacji danych, które pozwalają na wprowadzenie do systemu niepoprawnych danych, co może prowadzić do nieautoryzowanego dostępu lub innego rodzaju ataków.

- **File Upload Vulnerabilities:** Nieprawidłowe przetwarzanie danych wejściowych w przypadku przesyłania plików może prowadzić do załadowania złośliwego oprogramowania na serwer.

- **Brute Force Attacks:** Ataki, w których intruz próbuje odgadnąć dane uwierzytelniające poprzez wielokrotne próby, wykorzystując różne kombinacje loginu i hasła.

- **Security Misconfigurations:** Niewłaściwe konfiguracje związane z obsługą danych wejściowych, takie jak nieodpowiednie ustawienia serwera czy bazy danych, mogą prowadzić do luk w zabezpieczeniach.

3.2.2. Walidacja danych

Walidacja danych to ważny element procesu zabezpieczania aplikacji webowych przed różnorodnymi zagrożeniami. Obejmuje ona sprawdzanie, czy dane wprowadzone przez użytkowników są zgodne z oczekiwanym formatem i spełniają określone kryteria, co eliminuje potencjalne luki w bezpieczeństwie. Przykłady walidacji danych obejmują:

- **Sprawdzenie poprawności formularzy:** Weryfikacja, czy dane wprowadzone w formularzach są zgodne z oczekiwanym typem (np. tekst, liczba, e-mail) oraz czy nie zawierają złośliwego kodu.

- **Ochrona przed SQL injection:** Unikanie bezpośredniego wstawiania danych wejściowych do zapytań SQL. Zamiast tego możliwe jest stosowanie przygotowanych instrukcji SQL (Prepared Statements), które eliminują ryzyko wstrzykiwania złośliwego kodu SQL.

- **Filtrowanie danych:** Usuwanie lub zastępowanie potencjalnie niebezpiecznych danych, takich jak znaki specjalne, w celu zminimalizowania ryzyka ataków typu XSS czy SQL Injection.

- **Walidacja na poziomie serwera i klienta:** Walidacja po stronie serwera jest obowiązkowa, ale walidacja po stronie klienta może zapewnić szybszą informację zwrotną dla użytkownika, zanim dane zostaną przesłane na serwer.

- **Użycie frameworków i bibliotek:** Korzystanie z wbudowanych funkcji walidacyjnych dostępnych w popularnych frameworkach i bibliotekach programistycznych ułatwia implementację bezpiecznej walidacji.

- **Zarządzanie błędami:** Odpowiednie zarządzanie błędami walidacji, które nie ujawnia nadmiernych informacji, jest kluczowe dla bezpieczeństwa. Użytkownik

nie powinien uzyskać dostępu do szczegółów implementacji bądź struktury bazy danych.

- **Regularne wyrażenia:** Skuteczne wykorzystanie regularnych wyrażeń pozwala na precyzyjne określenie oczekiwanych wzorców danych, co wspomaga skuteczną walidację.

- **Aktualizacja zabezpieczeń:** Regularna aktualizacja mechanizmów walidacyjnych w odpowiedzi na zmieniające się zagrożenia i standardy bezpieczeństwa.

3.3. Protokół HTTPS

Protokół HTTPS (HTTP Secure) to rozszerzenie standardowego protokołu HTTP (Hypertext Transfer Protocol), który służy do zabezpieczania komunikacji w Internecie. Jest to podstawowy protokół używany do bezpiecznej komunikacji w sieci World Wide Web i jest szeroko stosowany w transakcjach online, bankowości i wymianie innych poufnych informacji.

HTTPS wykorzystuje tę samą podstawową strukturę co HTTP, ale jest implementowany za pośrednictwem protokołów Secure Sockets Layer (SSL) lub Transport Layer Security (TLS). Protokoły te zapewniają szyfrowanie i uwierzytelnianie danych wymienianych między klientem a serwerem, zapewniając, że dane są przesyłane bezpiecznie i mogą być odczytane tylko przez zamierzonego odbiorcę.

Gdy klient łączy się z serwerem HTTPS, serwer wysyła do klienta swój certyfikat SSL/TLS. Następnie klient weryfikuje certyfikat, aby upewnić się, że jest ważny i wydany przez zaufany urząd. Po zweryfikowaniu certyfikatu klient i serwer ustanawiają sesję SSL/TLS, która umożliwia im bezpieczną wymianę danych. HTTPS domyślnie używa portu 443 i jest używany przez wiele usług i aplikacji internetowych, takich jak bankowość internetowa, handel elektroniczny i media społecznościowe, w celu zabezpieczenia transferu danych między klientem a serwerem. Ponadto większość przeglądarek internetowych wyświetla ikonę kłódki i prefiks „https” w pasku adresu witryny, aby wskazać, że witryna korzysta z protokołu HTTPS.

HTTPS stał się standardem bezpiecznej komunikacji w Internecie i jest obecnie szeroko stosowany, ponieważ zapewnia poufność, integralność i autentyczność przesyłanych danych.

3.4. Nawiązywanie połączenia przez protokół HTTPS

Jak już wspomnaino HTTPS (HTTP Secure) wykorzystuje tę samą podstawową strukturę co HTTP, ale do ustanowienia połączenia używa protokołów Secure Sockets Layer (SSL) lub Transport Layer Security (TLS). Proces nawiązywania połączenia przy użyciu protokołu HTTPS jest powszechnie znany jako "TLS handshake"i zwykle obejmuje następujące kroki:

1) Klient inicjuje połączenie, wysyłając do serwera komunikat „Client Hello”. Ta wiadomość zawiera informacje o wersji SSL/TLS klienta, obsługiwanych zestawach szyfrów i unikalnym identyfikatorze sesji.

2) Serwer odpowiada komunikatem „Server Hello”, który zawiera informację o wersji SSL/TLS serwera, wybrany zestaw szyfrów oraz certyfikat SSL/TLS. Certyfikat zawiera informacje o kluczu publicznym serwera i tożsamości urzędu certyfikacji, który go wystawił.

3) Klient weryfikuje certyfikat serwera, aby upewnić się, że jest ważny i wydany przez zaufany urząd. Jeśli certyfikat jest ważny, klient generuje "premaster secret"i szyfruje go przy użyciu klucza publicznego.

4) Serwer używa swojego klucza prywatnego do odszyfrowania klucza "premaster secret"a następnie razem z klientem generuje klucz główny. Zarówno klient, jak i serwer używają klucza głównego do generowania kluczy sesyjnych do szyfrowania i odszyfrowywania danych.

5) Klient wysyła do serwera komunikat „Client Key Exchange”, który potwierdza, że klient zakończył uzgadnianie SSL/TLS. W odpowiedzi serwer wysyła do klienta komunikat „Server Hello Done”.

6) Klient i serwer wymieniają komunikaty „Change Cipher Spec”, aby potwierdzić, że będą używać kluczy sesji do szyfrowania i deszyfrowania danych.

7) Klient wysyła do serwera komunikat „Zakończono”, na który Serwer odpowiada tym samym komunikatem.

8) Po zakończeniu uzgadniania SSL/TLS klient i serwer mogą bezpiecznie wymieniać dane przy użyciu kluczy sesji.

4. Metody Testowania Bezpieczeństwa w Aplikacjach Webowych

Testowanie bezpieczeństwa w aplikacjach webowych ma na celu zidentyfikowanie potencjalnych luk w zabezpieczeniach i zagrożenia dla danych. Testy te służą ocenie trwałości systemu przed rzeczywistymi atakami. Metody testowania bezpieczeństwa obejmują analizę statyczną i dynamiczną kodu, testy przeciążenia, testy penetrujące oraz testy aplikacji mobilnych. Ważnym aspektem jest również kontrola zgodności z normami bezpieczeństwa i przepisami prawnymi. Wszystkie te metody pozwalają na identyfikację i likwidację potencjalnych słabych punktów oraz poprawę ogólnego poziomu bezpieczeństwa aplikacji webowych.

4.1. Rodzaje testów bezpieczeństwa

Jako rodzaje testów bezpieczeństwa można wymienić:

- **Testy penetracyjne (penetration testing):**Przeprowadzanie kontrolowanych ataków na aplikację w celu identyfikacji słabości i podatności, które mogą być wykorzystane przez potencjalnych atakujących.

- **Skanowanie bezpieczeństwa (security scanning):**Automatyczne skanowanie aplikacji w celu wykrycia luk w zabezpieczeniach, nieaktualnych komponentów, czy też innych podatności.

- **Analiza kodu źródłowego (static code analysis):**Ocena kodu aplikacji w poszukiwaniu potencjalnych podatności i niezabezpieczonych fragmentów kodu.

- **Testy wydajności przy obciążeniu (load testing):**Sprawdzanie, jak aplikacja radzi sobie podczas obciążenia, w celu zidentyfikowania potencjalnych luk w bezpieczeństwie związanych z obciążeniem.

- **Analiza ruchu sieciowego (network traffic analysis:** Monitorowanie i analiza ruchu sieciowego w celu wykrywania niezwykłych lub potencjalnie szkodliwych aktywności.

- **Testy integracyjne:**Weryfikacja bezpieczeństwa integracji aplikacji z innymi systemami, w tym zewnętrznymi usługami, API czy też bazami danych.

- **Analiza konfiguracji bezpieczeństwa:**Sprawdzanie, czy konfiguracja serwerów, baz danych i innych składników aplikacji jest zgodna z najlepszymi praktykami bezpieczeństwa.

- **Testy uwierzytelniania i autoryzacji:** Ocena skuteczności mechanizmów uwierzytelniania i autoryzacji, aby upewnić się, że tylko uprawnione osoby mają dostęp do określonych zasobów.

- **Testy bezpieczeństwa mobilnego:** Jeśli aplikacja posiada komponenty mobilne, przeprowadzane są testy specyficzne dla zagrożeń związanych z urządzeniami mobilnymi.

- **Analiza raportów i logów:** Monitorowanie i analiza raportów z testów bezpieczeństwa oraz logów aplikacji w celu identyfikacji nieprawidłowości, ataków lub podejrzanej aktywności.

4.2. Narzędzia do przeprowadzania testów bezpieczeństwa

1) **Narzędzia do Testów Penetracyjnych:**

    - **Metasploit:** Platforma do testów penetracyjnych, która zawiera szereg narzędzi i exploitów do testowania bezpieczeństwa systemów.
    - **Burp Suite:** Narzędzie do testowania aplikacji webowych, obejmujące funkcje takie jak skanowanie podatności, analiza ruchu HTTP i wspieranie testów bezpieczeństwa.

2) **Narzędzia do Testów Aplikacji Webowych:**

    - **OWASP ZAP (Zed Attack Proxy):** Narzędzie do testowania bezpieczeństwa aplikacji webowych, oferujące funkcje skanowania podatności, analizy bezpieczeństwa i ataków man-in-the-middle.
    - **Netsparker:** Automatyczne narzędzie do skanowania podatności w aplikacjach webowych, identyfikujące luki w zabezpieczeniach.

3) **Narzędzia do Analizy Statycznej Kodu:**

    - **Checkmarx:** Narzędzie do analizy statycznej kodu, identyfikujące podatności w kodzie źródłowym aplikacji.

- **Fortify Static Code Analyzer:** Narzędzie firmy Micro Focus do statycznej analizy kodu, pomagające w identyfikacji i eliminacji podatności.

4) **Narzędzia do Analizy Dynamicznej Kodu:**

   - **AppScan (IBM Security):** Narzędzie do analizy dynamicznej kodu, skanowania podatności i testowania bezpieczeństwa aplikacji webowych.
   - **Veracode:**Platforma do analizy dynamicznej i statycznej kodu, dostarczająca narzędzia do oceny bezpieczeństwa aplikacji.

5) **Narzędzia do Testów Penetracyjnych Sieci:**

   - **Wireshark:** Narzędzie do analizy ruchu sieciowego, pozwalające na monitorowanie i analizę pakietów danych w czasie rzeczywistym.
   - **Nmap:** Narzędzie do skanowania sieci, identyfikujące otwarte porty i usługi, a także zbierające informacje o hostach w sieci.

6) **Narzędzia do Testów Przeciążeniowych:**

   - **Apache JMeter:** Narzędzie do testowania wydajności, przeciążenia i stabilności aplikacji, wspierające scenariusze obciążeniowe.
   - **Loader.io:** Narzędzie do testowania obciążenia serwerów poprzez symulowanie dużej liczby użytkowników.

7) **Narzędzia do Testów Bezpieczeństwa Aplikacji:**Mobilnych:

   - **MobSF (Mobile Security Framework):** Platforma do analizy bezpieczeństwa aplikacji mobilnych, oferująca skanowanie podatności i analizę zachowania.
   - **Drozer (MWR InfoSecurity):** Narzędzie do testowania bezpieczeństwa aplikacji Android, umożliwiające analizę interakcji między aplikacjami.

8) **Narzędzia do Testów Zgodności:**

   - **OpenSCAP:** Narzędzie do skanowania systemów pod kątem zgodności z normami bezpieczeństwa.
   - **Nessus:** Narzędzie do skanowania podatności, obejmujące funkcje związane z testowaniem zgodności z normami branżowymi.

4.3. Penetrowanie Aplikacji Webowych

4.3.1. Cel i zakres testów penetracyjnych

4.3.2. Praktyki bezpiecznego penetrowania

5. Technologie wykorzystane do utworzenia aplikacji

Do utworzenia aplikacji wykorzystane zostały technologie takie jak JavaScript, TypeScript, React i Node.js. JavaScript jest używany po stronie klienta do dynamicznego renderowania interfejsu, natomiast TypeScript wprowadza statyczne typy i dodatkowe funkcje, poprawiając jakość i skalowalność kodu. React, jako biblioteka do budowy interfejsów użytkownika, ułatwia deklaratywne tworzenie komponentów, co przyspiesza rozwój aplikacji. Node.js, jako środowisko uruchomieniowe, umożliwia skryptowe wykonywanie kodu JavaScript po stronie serwera.

5.1. JavaScript

JavaScript to wszechstronny język programowania stosowany głównie do tworzenia interaktywnych i dynamicznych stron internetowych. Jest skryptowym językiem, co oznacza, że jego kod jest wykonywany w przeglądarce internetowej użytkownika. JavaScript umożliwia manipulację treścią strony, obsługę zdarzeń, komunikację z serwerem, a także tworzenie bogatych interfejsów użytkownika.

5.2. TypeScript

TypeScript to rozszerzenie języka JavaScript, które wprowadza statyczne typy oraz nowe funkcje do poprawy jakości i skalowalności kodu. Jest rozwijane przez Microsoft i przed użyciem w przeglądarce jest kompilowany do zwykłego kodu javaScript, co znaczy że każdy poprawny kod JavaScript jest również poprawnym kodem TypeScript.

Główną cechą TypeScript jest wprowadzenie statycznego systemu typów. Programiści mogą określać typy zmiennych, parametrów funkcji, a także zwracanych wartości. To umożliwia wykrywanie błędów już na etapie kompilacji, co ułatwia debugowanie i zwiększa pewność co do poprawności kodu.

5.3. React

React to biblioteka JavaScript stworzona przez Facebooka, skoncentrowana na budowaniu interfejsów użytkownika. Jest powszechnie stosowana do tworzenia dynamicznych i efektywnych aplikacji internetowych. React wprowadza podejście oparte na komponentach, co oznacza, że interfejs użytkownika jest dekomponowany na mniejsze, łatwiejsze do zarządzania części, nazywane komponentami. Te komponenty są następnie łączone, tworząc pełen interfejs.

Jedną z głównych cech React jest wirtualny DOM (Document Object Model), który umożliwia efektywne aktualizacje interfejsu użytkownika bez konieczności manipulacji całym drzewem DOM. React obsługuje także jednokierunkowy przepływ danych, co ułatwia śledzenie i zarządzanie stanem aplikacji.

Dodatkowo, wspierana jest funkcja reaktywnego aktualizowania interfejsu, co oznacza, że zmiany w stanie komponentów automatycznie prowadzą do aktualizacji widoku.

5.4. Node.JS

Node.js to środowisko wykonawcze dla języka JavaScript, które umożliwia uruchamianie skryptów po stronie serwera. Zostało stworzone na bazie silnika V8 od Google Chrome, co pozwala na efektywne wykonywanie kodu JavaScript poza przeglądarką internetową. Node.js jest często używane do budowy skalowalnych aplikacji sieciowych, serwerów HTTP, czy aplikacji internetowych w oparciu o technologię JavaScript.

Jednym z kluczowych aspektów Node.js jest jego asynchroniczność i nieblokujące wejścia/wyjścia. To oznacza, że jest w stanie obsługiwać wielu użytkowników jednocześnie bez blokowania jednostki wykonawczej. Node.js używa modelu zdarzeń, dzięki czemu sprawnie radzi sobie z obsługą dużych ilości równoczesnych połączeń, co jest szczególnie ważne w dziedzinie serwerów HTTP.

6. Efekty pracy

7. Podsumowanie

Literatura

[1] https://www.rfc-editor.org/rfc/rfc2818

[2] https://nordvpn.com/pl/blog/ataki-hakerskie/

[3] https://www.upguard.com/blog/biggest-data-breaches-us

[4] https://www.sunmark.org/connect/sunmark-360/12-worst-data-breaches-last-decade

[5] https://www.imperva.com/learn/data-security/data-security/

[6] https://learn.microsoft.com/en-us/entra/identity-platform/authentication-vs-authorization

[7] https://learn.microsoft.com/en-us/entra/identity/authentication/concept-authentication-methods

[8] https://hevodata.com/learn/data-validation/

[9] https://developer.mozilla.org/en-US/docs/Web/JavaScript

[10] https://www.typescriptlang.org/

[11] https://www.patterns.dev/react

[12] https://nodejs.org/en/about

POLITECHNIKA RZESZOWSKA im. I. Łukasiewicza Rzeszów, 2024
Wydział Elektrotechniki i Informatyki

STRESZCZENIE PRACY DYPLOMOWEJ MAGISTERSKIEJ

BEZPIECZEŃSTWO DANYCH W APLIKACJACH WEBOWYCH

Autor: Dawid Pnidara, nr albumu: EF-163994
Opiekun: dr inż. Bogdan Kwiatkowski
Słowa kluczowe: (max. 5 słów kluczowych w 2 wierszach, oddzielanych przecinkami)

Treść streszczenia po polsku

RZESZOW UNIVERSITY OF TECHNOLOGY Rzeszow, 2024
Faculty of Electrical and Computer Engineering

MSC THESIS ABSTRACT

TEMAT PRACY PO ANGIELSKU

Author: Dawid Pnidara, nr albumu: EF-163994
Supervisor: (academic degree) Imię i nazwisko opiekuna
Key words: (max. 5 słów kluczowych w 2 wierszach, oddzielanych przecinkami)

Treść streszczenia po angielsku