PAWEŁ RUS

PIOTR KONDZIOŁKA

SZCZEPAN DWORAK

CTF: Praktyczna nauka bezpieczeństwa aplikacji webowych

Czym są aplikacje webowe i dlaczego bezpieczeństwo jest ważne?

Aplikacje webowe to programy, które działają w przeglądarkach internetowych i są dostępne przez sieć. Przykłady to sklepy online, systemy bankowości internetowej, platformy społecznościowe czy narzędzia do zarządzania treścią (CMS). Działają dzięki połączeniu frontendu (interfejs użytkownika) i backendu (serwer, baza danych).

Rola CTF w edukacji bezpieczeństwa

Praktyczna nauka: Uczestnicy zdobywają doświadczenie w realnych scenariuszach zagrożeń.

Zrozumienie podatności: Analizowanie i wykorzystywanie luk w kontrolowanym środowisku pomaga lepiej rozumieć mechanizmy ataków.

Rozwój umiejętności technicznych: Poprawia umiejętności z zakresu kryptografii, inżynierii odwrotnej, eksploitacji i analizy kodu.

Praca zespołowa: CTF-y często wymagają współpracy i komunikacji w grupie.

Motywacja do nauki: Element rywalizacji sprawia, że uczestnicy chętniej poszerzają wiedzę.

Budowanie świadomości bezpieczeństwa: Pomaga lepiej zabezpieczać aplikacje i systemy.

Kluczowe aspekty bezpieczeństwa

Uwierzytelnianie i autoryzacja: Bezpieczne hasła, kontrola dostępu.

Walidacja danych wejściowych: Ochrona przed XSS i SQL Injection.

Bezpieczne sesje: Ciasteczka (Secure, HttpOnly), unikanie sesji w URL.

Szyfrowanie: HTTPS (SSL/TLS), szyfrowanie danych w tranzycie i spoczynku.

Zarządzanie lukami: Testy penetracyjne, szybkie aktualizacje.

Bezpieczny kod: Przeglądy i unikanie niebezpiecznych funkcji.

Monitorowanie: Logi i alerty podejrzanych działań.

SQL Injection

To atak polegający na wstrzyknięciu złośliwego kodu SQL do zapytań wysyłanych do bazy danych. Najczęściej wykorzystuje podatności w walidacji danych wejściowych. Skutkiem takiego ataku może być kradzież danych, manipulacja nimi lub przejęcie kontroli nad systemem. Aby zapobiegać atakom, należy stosować parametryzowane zapytania, unikać konkatenacji danych użytkownika w SQL oraz regularnie testować aplikację pod kątem podatności.

$query = "SELECT * FROM users WHERE username = '$username' AND password = '$password'";

Atak: Użytkownik wprowadza w polu "hasło" następujący ciąg: ' OR '1'='1

Zmieniony zapytanie SQL: SELECT * FROM users WHERE username = 'admin' AND password = '' OR '1'='1';

Skutek:

Zapytanie zwraca dane użytkownika "admin" niezależnie od hasła, ponieważ warunek `'1'='1'` zawsze jest prawdziwy.

Przykład ataku SQL Injection

Cross-Site Scripting (XSS)

To atak polegający na wstrzyknięciu złośliwego kodu (np. JavaScript) do treści wyświetlanej przez aplikację webową. Atakujący wykorzystuje tę lukę, aby kraść dane użytkowników, przechwytywać sesje lub wykonywać działania w ich imieniu. Aby zapobiegać XSS, należy prawidłowo filtrować i kodować dane wejściowe oraz stosować polityki zabezpieczeń, takie jak Content Security Policy (CSP).

Przykład ataku XSS

W przypadku podatności na XSS, atakujący może wstrzyknąć złośliwy kod JavaScript, który zostanie wykonany przez przeglądarkę użytkownika.

Podatny kod (brak walidacji danych wejściowych):

```
<input type="text" name="comment" value="<?php echo $_GET['comment']; ?>" />
```

Atak: Użytkownik wprowadza w polu komentarza następujący ciąg:

```
<script>alert('Zostałeś zaatakowany!');</script>
```

Skutek:

Przeglądarka użytkownika wykona wstrzyknięty kod, wyświetlając alert z komunikatem "Zostałeś zaatakowany!". W bardziej zaawansowanych atakach, atakujący może kraść ciasteczka sesyjne lub przejąć konto użytkownika.

Cross-Site Request Forgery (CSRF)

CSRF to atak, w którym złośliwa strona internetowa wysyła nieautoryzowane żądania do aplikacji, w której użytkownik jest zalogowany. Atakujący wykorzystuje to, by wykonać niepożądane akcje w imieniu ofiary, takie jak zmiana ustawień konta lub wykonanie transakcji. Aby chronić się przed CSRF, należy używać tokenów CSRF w formularzach, weryfikować źródło żądań i stosować metody autoryzacji, które nie polegają na prostym ciasteczku sesyjnym.

Przykład ataku CSRF

Atak CSRF polega na wysyłaniu nieautoryzowanych żądań w imieniu zalogowanego użytkownika. Aplikacja, która nie weryfikuje źródła żądania, może być podatna na ten typ ataku.

Podatny kod (brak weryfikacji CSRF):

```
<form action="change_password.php" method="POST">

  <input type="text" name="new_password" value="newpassword123" />

  <input type="submit" value="Change Password" />

</form>
```

Atak: Atakujący może stworzyć złośliwą stronę z poniższym kodem HTML:

```
<img src="http://example.com/change_password.php?new_password=attackerpassword" />
```

Skutek:

Jeśli ofiara odwiedzi stronę atakującego, jej przeglądarka wyśle nieautoryzowane żądanie do aplikacji, zmieniając hasło na "attackerpassword", bez jej wiedzy.

Ochrona:

Użycie tokenów CSRF w formularzach i weryfikacja źródła żądania zapobiega takim atakom.