УДК 004.4

Подтьопа Сергій

ПОМИЛКИ ПРИ РОБОТІ З WEB-СИСТЕМАМИ АВТЕНТИФІКАЦІЇ

У статті розглядаються основні помилки починаючих та досвідчених web-розробників при роботі з системами автентифікації. Розглянуті помилки актуальні для всіх мов програмування, хоча стаття зосереджена над мовою JavaScript і її бібліотеками.

Ключові слова: автентифікація, web-додаток, JavaScript, захист даних.
Бібл.: 13.

The paper deals with the issues the major errors of beginning and experienced web-developers when working with authentication systems. The bugs discussed are relevant to all programming languages, although the paper deals focuses on the language of JavaScript and its libraries.

Key words: authentication, web-application, JavaScript, data protection
Bibl.: 13.

Актуальність теми дослідження. У зв'язку з активним розвитком web-індустрії, зростає потреба в якісних системах автентифікації для web-ресурсів. На сам перед дана стаття буде актуальна для JS-розробників, але помилки вказані в ній часто зустрічаються і в інших платформах.

Постанова проблеми. Часто при роботі з системами автентифікації виникають помилки, що призводить до незахищеності даних користувачів.

Аналіз останніх досліджень і публікацій. Незважаючи на значну кількість робіт, присвячених питанню автентифікації та захисту даних в цілому, проблема роботи з web-системами автентифікації залишається мало дослідженою.

Мета дослідження. Метою даного дослідження є виділити основні помилки при роботі з web-системами автентифікації. Стаття буде зосереджена на найбільш розповсюджених помилках та їхньому уникненні при створенні власної системи автентифікації чи використанні існуючої.

Викладення основного матеріалу. На сьогоднішній деньіснує безліч навчальних посібників, які спрямовані на допомогу в установці та налагодженні системи автентифікації для web-додатків. Практично всі вони містять ті чи інші помилки. Жодне з них не дозволяє створити повномасштабне рішення, необхідне для працюючого веб-додатку.

Сховище даних користувачів. Запис і читання облікових даних є доволі звичним завданням для систем по управлінню автентифікацією, традиційним способом вирішення цих завдань є використання власної бази даних. Більшість існуючих систем автентифікації є лише проміжним програмним забезпеченням, що зазвичай просто повідомляє нашому додатку, що користувач пройшов або не пройшов перевірку, вимагаючи відповіді для роботи зі сховищем паролів в локальній базі даних.

Звернувшись до системи Passport.js [1], а саме до посібника, можна побачити, що в прикладах використання, не виявилося підсистеми підтримки бази даних. Приклад мав на увазі просте використання деякого набору акаунтів.

На перший погляд все доволі логічно. Нам наведений в приклад звичайний web-додаток. Але розробник, який скористається ним, через недосвідченість чи брак часу не буде нічого серйозно поліпшувати. Зауваживши, що паролі в прикладі не шифруються, а зберігаються у вигляді тексту поруч з кодом логіки валідації. Про сховище облікових даних в прикладі навіть не згадується. Дуже цікава вийде система автентифікації.

Далі наведений ще один посібник по Passport.js , написаний в 2015 році [2]. Даний посібник використовує Mongoose ODM і читає облікові дані з бази даних. Тут є все, зокрема інтеграційні тести, а також ще один шаблон, що можна використовувати. Проте, Mongoose ODM зберігає паролі, використовуючи тип даних String, як і в попередній інструкції, у вигляді звичайного тексту, тільки на цей раз в екземплярі MongoDB. Добре відомо, що екземпляри MongoDB зазвичай дуже добре захищені.

Звернемось тепер до всіма улюбленого пошукового сервісу. Верхнім результатом є керівництво від scotch.io [3], даний посібник є кращим, в ньому використовується bcrypt з коефіцієнтом трудомісткості рівним 8 для шифрування паролів. Але 8 – це мало, навіть дуже мало. Більшість сучасних bcrypt-бібліотек використовують 12. Коефіцієнт трудомісткості 8 був хороший для адміністративних облікових записів вісімнадцять років тому, відразу після випуску першої специфікації bcrypt.

Якщо навіть не говорити про зберігання секретних даних, жодний з посібників не показує реалізацію механізму скидання паролів. Найважливіша частина системи автентифікації, де багато власних помилок, була залишена в якості завдання для розробника.

Система скидання паролів. Схожа проблема в області безпеки – скидання пароля. Жодний з посібників, що знаходяться у верхній частині пошукового сервісу, абсолютно нічого не говорить про те, як робити це з використанням Passport.

При реалізації системи скидання паролів є тисячі способів все зіпсувати. Ось найпоширеніші помилки в рішенні цієї задачі, що можна зустріти:

1. ***Передбачувані токени.*** Гарним прикладом є токени, засновані на поточному часі. Токен, що побудований на базі поганого генератора псевдорандомних чисел, хоча і виглядає краще, проблему не вирішують теж.

2. ***Погане сховище даних***. Зберігання незашифрованих токенів скидання пароля в базі даних означає, що якщо база буде зламана, ці токени – фактично паролі, що зберігаються у вигляді відкритого тексту. Створення довгих токенів за допомогою криптографічно стійкого генератора псевдовипадкових чисел дозволяє запобігти грубих атак на токени скидання паролів, але не захищає від локальних атак. Токени скидання паролів є обліковими даними і їх слід розглядати як такі.

3. ***Токени, без обмеженого терміну дії.*** Якщо термін дії токенів не закінчується, то атакуючий має час, щоб скористатися тимчасовим вікном скидання пароля.

4. ***Відсутність додаткових перевірок***. Додаткові питання при скиданні пароля – це стандарт верифікації даних де-факто. Звичайно, це працює як треба лише в тому випадку, якщо розробники вибирають хороші питання. У подібних питань є власні проблеми. Тут варто сказати і про використання електронної пошти для відновлення пароля, хоча міркування про це можуть здатися зайвої перестраховкою. Адреса електронної пошти – це те, що у вас є, а не те, що ви

знаєте. Адреса об'єднує різні чинники автентифікації. Як результат, електронна пошти стає ключем до будь-якого облікового запису, яка просто відправляє на нього токен скидання пароля.

Наголосивши загальні питання, перейдемо до конкретики.

Можна звернути увагу на бібліотеку для скидання паролів п'ятирічної давності від чудового видавця substack [4]. З огляду на швидкість розвитку web-технологій, цей пакет нагадує дідуся, розглянувши деталі, можна сказати, що функція Math.random() доволі передбачувана, тому її не слід використовувати для створення токенів, тому йдемо далі.

Отож, повертаємося до пошукового сервісу. Виникає відчуття, що тема, котра цікавить нас, розкрита лише в одному матеріалі. Візьмемо перший результат, знайдений за запитом [5]. Знову зустрічаємо bcrypt, з меншим коефіцієнтом трудомісткості, ніж потрібно в сучасних умовах, рівним 5.

Посібник виглядає досить-таки цілісним в порівнянні з іншими, так як використовує crypto.randomBytes для створення по-справжньому випадкових токенів, термін дії яких закінчується, якщо вони не були використані. Але пункти 2 і 4 з вищенаведеного списку помилок при скиданні пароля в посібнику не враховані. Токени зберігаються ненадійно - згадуємо першу помилку керівництв з автентифікації, пов'язану зі зберіганням облікових даних.

Позитивним є те, що викрадені з такої системи токени мають обмежений термін дії. Проте, ці токени особливо приємні, якщо у атакуючого є доступ до об'єктів користувачів в базі даних через BSON, або може отримати вільний доступ до Mongo через невірну конфігурацію. Атакуючий може просто запустити процес скидання пароля для кожного користувача, прочитати незашифровані токени з бази даних і створити власні паролі для облікових записів користувачів, замість ресурсномістких атак по словнику bcrypt, використовуючи потужний комп'ютер.

Токени API. Токени API є обліковими даними. Вони так само важливі, як паролі та токени скидання паролів. Практично всі розробники знають це і намагаються дуже надійно зберігати свої ключі та коди, однак, програм, над якими вони працюють, це доволі часто не стосується.

Справи складаються так, що в пошуковому сервісі важко знайти вартий уваги матеріал, а з популярних посібників [6] можна відмітити деякі помилки в зберіганні облікових даних:

1. Ключі токенів API зберігаються у вигляді звичайного тексту в наведеній репозиторії.

2. Для зберігання паролів використовується симетричний шифр. Це означає, хтось може заволодіти ключем шифрування і розшифрувати всі паролі. До того ж, тут спостерігаються неправильні взаємини між ключем шифрування і секретним ключем токенів API.

3. Для шифрування даних в сховищі паролів, використовується алгоритм AES-256-CTR. Даний алгоритм взагалі не варто використовувати, і вказаний його варіант нічого не змінює [7].

На жаль більшість посібників [6, 8, 9] мають схожі помилки з захисту облікових даних користувачів та в них спостерігається та ж уразливість, пов'язана з розкриттям інформації.

Обмеження кількості спроб автентифікації. В жодному серед наведених вище посібників зі створення систем автентифікації не знайшлося згадок про обмеження числа спроб автентифікації або про блокування облікового запису.

Без обмеження кількості спроб автентифікації, атакуючий може виконувати онлайн атаку по словнику, в надії отримати доступ до облікового запису зі слабким паролем. Блокування облікового запису так само допомагає у вирішенні цієї проблеми завдяки вимогам введення додаткової інформації при наступних спробах входу в систему.

Слід пам'ятати про те, що обмеження числа спроб автентифікації сприяє доступності сервісу. Так, використання bcrypt створює серйозне навантаження на процесор. Без обмежень, функції, в яких викликають bcrypt, стають вектором відмови в обслуговуванні рівня додатки, особливо при використанні високих значень коефіцієнта трудомісткості. В результаті обробка безлічі запитів на реєстрацію користувачів або на вхід в систему з перевіркою пароля створює високе навантаження на сервер.

Хоча відповідного навчального керівництва на цю тему майже немає, є безліч допоміжних бібліотек для обмеження числа запитів [10, 11, 12]. Важко сказати про рівень безпеки цих модулів, але свою функцію мають цілком виконати.

Висновки. Автентифікація – важка задача. Багато хто може сказати, що посібники – це лише пояснення основ, але недосвідчені читачі вірять словам авторів посібників, у яких набагато більше досвіду, ніж у тих, хто читає керівництва.

Якщо ви – розробник початківець – не довіряйте навчальним посібникам. Копіювання коду з таких матеріалів, часто може, привести до проблем у сфері web-автентифікації. Якщо вам дійсно потрібні надійні, готові до використання в коді бібліотеки для автентифікації, погляньте на щось, чим вам буде зручно користуватися, на те, що володіє більшою стабільністю і краще випробувано часом.

Список використаних джерел

1.Passport.Simple, unobtrusiveauthenticationforNode.js. [Електронний ресурс]. — Режим доступу: http://www.passportjs.org/docs/

2. Michael Herman. User Authentication with Passport and Express 4. [Електронний ресурс]. — Режим доступу:http://mherman.org/blog/2015/01/31/local-authentication-with-passport-and-express-4/

3. Chris Sevilleja. Easy Node Authentication: Setup and Local. [Електронний ресурс]. — Режим доступу:https://scotch.io/tutorials/easy-node-authentication-setup-and-local

4. Substack. Middleware for managing password reset emails. [Електроннийресурс]. — Режим доступу:https://npmjs.com/package/password-reset/

5. Sahat Yalkabov. How To Implement Password Reset In Node.js. [Електронний ресурс]. — Режим доступу:http://sahatyalkabov.com/how-to-implement-password-reset-in-nodejs/

6. Soni Pandey. User Authentication using JWT (JSON Web Token) in Node.js. [Електронний ресурс]. — Режим доступу:https://medium.com/@pandeysoni/user-authentication-using-jwt-json-web-token-in-node-js-using-express-framework-543151a38ea1/

7. Is regular CTR mode vulnerable to any attacks? [Електронний ресурс]. — Режим доступу: https://crypto.stackexchange.com/questions/33846/is-regular-ctr-mode-vulnerable-to-any-attacks/33861#33861

8. JonathanMH. About Express, Passport and JSON Web Token Authentication for Beginners. [Електронний ресурс]. — Режим доступу:https://jonathanmh.com/express-passport-json-web-token-jwt-authentication-beginners/

9. Joshua Slate. Creating a Simple Node/Express API Authentication System with Passport and JWT. [Електронний ресурс]. — Режим доступу:http://blog.slatepeak.com/creating-a-simple-node-express-api-authentication-system-with-passport-and-jwt/

10. ExpressRateLimit. [Електронний ресурс]. — Режим доступу:https://npmjs.com/package/express-rate-limit/

11. Expresslimiter. [Електронний ресурс]. — Режим доступу:https://npmjs.com/package/express-limiter/

12. Adam Pflug. Express-brute. [Електронний ресурс]. — Режим доступу:https://github.com/AdamPflug/express-brute

13. Your Node.js authentication tutorial is (probably) wrong. [Електронний ресурс]. — Режим доступу:https://hackernoon.com/your-node-js-authentication-tutorial-is-wrong-f1a3bf831a46/

ДОВІДКА ПРО АВТОРА

Подтьопа Сергій Андрійович – студент, кафедра обчислювальної техніки, Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського».

Podtopa Serhii –student, Department of Computer Engineering, National Technical University of Ukraine “Igor Sikorsky Kyiv Polytechnic Institute”.

E-mail: seriy_gry@ukr.net

Podtopa Serhii

ERRORS IN WORK WITH WEB SYSTEMS AUTHENTICATION

Relevance of the research. Due to the active development of the web-industry, there is a growing need for high-quality authentication systems for web-resources. The article itself will be relevant to JS developers, but the errors mentioned in it often found on other platforms as well.

Target setting. The lack of comparative studies on selection of a parallelism degree of granularity to solve specific mathematical problems that require significant computation, and tools that implement parallelism.

Actual scientific researches and issues analysis. Despite the large number of works devoted to the issue of authentication and data protection in general, the problem of working with web-system authentication systems remains poorly researched.

The research objective. The purpose of this study is to identify the main errors in working with web-authentication systems. The article will focus on the most common mistakes and avoid them when creating their own authentication system or using an existing one.

The statement of basic materials. The four main errors that occur when working with the authentication system identified. Examples of the occurrence of these errors and the ways of avoiding them are given.

Conclusions. Authentication is a difficult task. If you are a beginner developer, do not trust the tutorials. Copying the code from such materials can often lead to problems in the field of web-authentication.

Key words: authentication, web-application, JavaScript, data protection.