Вісник ЛДУБЖД
Bulletin of Lviv State University of Life Safety
https://journal.ldubgd.edu.ua/index.php/Visnuk
ISSN 2078-4643 (print), ISSN 2708-1389 (online)

UDC 004.056
DOI: 10.32447/20784643.27.2023.08

А. Ю. Толкачова, О. І. Гарасимчук, І. Р. Опірський
Національний університет «Львівська політехніка», м. Львів, Україна
ORCID: https://orcid.org/0000-0002-8742-8872 –О. І. Гарасимчук
https://orcid.org/0000-0002-8461-8996 – І. Р. Опірський
oleh.i.harasymchuk@lpnu.ua

АНАЛІЗ БЕЗПЕКИ ПРОТОКОЛУ OAUTH

Постановка проблеми: В останні роки стрімко зростає використання протоколів авторизації та автентифікації в різних сервісах, що зумовлює підвищений інтерес до таких протоколів і до гарантій захисту, які вони можуть надавати. Робота присвячена огляду безпеки протоколу OAuth, який на сьогодні є одним з найбільш поширених механізмів авторизації, дозволяє користувачам надавати доступ до своїх захищених ресурсів без надання прямого доступу до свого пароля та лежить в основі стандарту SSO OpenID Connect. Він став стандартом для багатьох веб-додатків та API, які забезпечують доступ до обмежених ресурсів. Останнім часом стрімко зростає кількість зловживань під час використання цього протоколу, а також різноманітні кібератаки на нього, що створює критичні проблеми для його користувачів. В першу чергу такі атаки здійснюються через проблеми безпеки цього протоколу.

Метою дослідження є розроблення рекомендацій з реалізації та пошуку вразливостей методом тестування на проникнення Web-сервісів в контексті протоколу Oauth 2.0.

Результати дослідження. В роботі розглянуті базові аспекти та механізми використання цього протоколу. Детально проаналізовані його основні відомі вразливості. Зокрема досліджено, що в результаті недостатньої перевірки URI перенаправлення може бути порушена ідентифікація або автентифікація клієнта, що дозволить зловмиснику отримати код авторизації. Описано сценарії можливих атак на надання коду авторизації та на неявний грант, загрозу витоку кодів станів через сторони клієнта або сервера авторизації через Referrer-заголовки, які також часто використовуються зловмисниками для порушення роботи протоколу OAuth. Також, продемонстровано, що через протокол OAuth може бути скомпрометований сервер ресурсів. Проведено оцінювання вразливостей з використанням CVSS-калькулятора, яке показало певні серйозні моменти з безпекою протоколу OAuth.

Висновки. Отже, хоча протокол OAuth дозволяє зручну та безпечну авторизацію та автентифікацію в різних веб-додатках та сервісах, він має свої потенційні загрози безпеці, які необхідно враховувати при його використанні. Важливо дотримуватися високих стандартів безпеки та забезпечувати належну конфігурацію сервера, щоб захистити дані користувачів від можливих загроз. Оскільки дослідження в роботі були спрямовані загалом на сам протокол OAuth, а не на його окремі реалізації, то були надані загальні чіткі рекомендації, дотримання яких дасть змогу покращити безпеку авторизації та автентифікації користувачів в Інтернет, а також забезпечити захист інформації на належному рівні.

Ключові слова: безпека, протокол OAuth, доступ до ресурсів, автентифікація, авторизація, атака, вразливості**.**

A. Yu. Tolkachova, O. I. Harasymchuk, I. R. Opirskyy
Lviv Polytechnic National University, Lviv, Ukraine

SECURITY ANALYSIS OF THE OAUTH PROTOCOL

Introduction. In recent years, the use of authorisation and authentication protocols in various services has been growing rapidly, which leads to increased interest in such protocols and the protection guarantees they can provide. The work is devoted to an overview of the security of the OAuth protocol, which is currently one of the most widely used authentication mechanisms, allowing users to grant access to their protected resources without providing direct access to their passwords, and is the basis of the OpenID Connect SSO standard. It has become the standard for many web applications and APIs that provide access to restricted resources. Recently, the number of abuses during the use of this protocol and various cyber-attacks on it has been rapidly increasing, which poses critical problems for users who use it. Such attacks are primarily carried out through existing security issues of this protocol.

This study aims to develop recommendations for the implementation and search for vulnerabilities by the method of penetration testing of Web services in the context of the Oauth 2.0 protocol. The paper considers the basic aspects and mechanisms of using this protocol. Its main known vulnerabilities are analysed in detail. In particular, it is investigated that as a result of insufficient URI redirection validation, client identification or authentication may be violated, allowing an attacker to obtain an authorization code. The scenarios of possible attacks on the provision of authorization code and

implicit grant, and the threat of leakage of state codes through client or authorisation server referrer headers, which are also often used by attackers to disrupt the OAuth protocol, are described. It is also demonstrated that the OAuth protocol can compromise the resource server. The vulnerability assessment was conducted using the CVSS calculator, which showed certain serious security issues with the OAuth protocol.

Conclusions. So, although the OAuth protocol allows convenient and secure authorisation and authentication in various web applications and services, it has its potential security threats that must be considered when using it. It is important to maintain high-security standards and ensure proper server configuration to protect user data from possible threats. Since the research in the paper was generally aimed at the OAuth protocol itself, not its implementations, clear general recommendations were provided, compliance with which will improve the security of user authentication and authentication on the Internet, as well as ensure information protection at an appropriate level.

Keywords: security, OAuth protocol, access to resources, authentication, authorisation, attack, vulnerabilities.;

Вступ. Питання автентифікації та авторизації користувачів мають надзвичайно важливе значення для захисту даних та інформаційної безпеки. Автентифікація – це процес, за допомогою якого ви надаєте докази того, що ви є тим, за кого себе видаєте. Авторизація надає вам дійсні дозволи [1-4]. Веб-додатки запровадили потребу в більш глибокому розумінні цих двох процесів як користувачам, так і розробникам. У переважній більшості взаємодій між користувачем та комп'ютером чи інформаційною системою застосовується автентифікація користувача. Дослідженню проблем безпеки під час автентифікації та авторизації присвячена велика кількість праць і в сучасних умовах необхідно постійно оновлювати та вдосконалювати системи авторизації користувачів, щоб захистити їх від різноманітних існуючих загроз.

Протокол OAuth [5-8] є одним з найбільш поширених механізмів авторизації, який використовуються в сучасному світі, а також виступає в якості основи для нового стандарту SSO OpenID Connect. Він дозволяє користувачам надавати доступ до своїх облікових записів на сторонніх сервісах без необхідності відкривати свій пароль. OAuth доволі поширений в різних сервісах, таких як соціальні мережі, онлайн магазини, електронна пошта і т.д. Цей протокол є стандартом відкритого програмного забезпечення і має велику спільноту розробників та організацій, що його підтримують. Постійно зростаюча кількість кібератак та зловживань під час використання протоколу OAuth створює серйозні проблеми для безпеки користувачів. У роботі ми детально проаналізуємо безпеку протоколу OAuth, розглянемо основні загрози, пов'язані з його використанням, а саме: атаки типу CSRF, а також дії, які можуть бути вжиті для запобігання цим загрозам. Також буде проведено оцінювання вже відомих вразливостей. Наш аналіз спрямований на встановлення надійних гарантій авторизації, автентифікації користувачів та цілісності даних.

Актуальність теми безпеки протоколу OAuth зростає з кожним днем, оскільки все більше компаній та сервісів використовують його для забезпечення доступу до особистих даних користувачів на сторонніх сервісах. Недотримання правил безпеки при використанні OAuth може призвести до витоку персональної інформації користувачів та порушення їхньої конфіденційності. Безпека даних є надзвичайно важливою в сучасному світі, тому проведення аналізу безпеки протоколу OAuth допоможе забезпечити захист від можливих загроз та забезпечити безпеку користувачів. протокол OAuth широко використовується в різних веб-додатках, у тому числі в таких популярних сервісах, як Google, Twitter, LinkedIn [9,10], Dropbox та багато інших. Наприклад, Google використовує протокол OAuth для авторизації користувачів у своїх сервісах, таких як Gmail, Google Drive, YouTube, Facebook тощо. Twitter використовує OAuth для авторизації та автентифікації користувачів на своїй платформі, а LinkedIn використовує OAuth для авторизації та доступу до свого API. Dropbox також використовує OAuth для авторизації та отримання доступу до сховища файлів користувача. Соціальна мережа Facebook використовує протокол OAuth для авторизації користувачів на сторонніх веб-сайтах та додатках. Користувачі можуть використовувати свій обліковий запис Facebook для авторизації на інших веб-сайтах та додатках, що дозволяє їм швидко та легко зареєструватись та авторизуватись на нових сервісах без необхідності створювати нові облікові записи та вводити повторно свої конфіденційні дані. Варто зазначити, що протокол OAuth став стандартом для багатьох сервісів, які надають доступ до API та особистих даних користувачів. Це дає змогу користувачам безпечно та зручно використовувати різні сервіси в Інтернеті, не надаючи при цьому свої паролі та інші конфіденційні дані. Оцінка практичної безпеки є доволі складним завданням, особливо з огляду на те, що робота системи залежить від закритого коду, власних специфікацій та інструкцій з реалізації. За відсутності детальних специфікацій оцінка безпеки потребуватиме вичерпної експериментальної оцінки та аналізу.

Виклад основного матеріалу. Спочатку варто проаналізувати основні вразливості та тими

можливих атак, які зловмисники можуть здійснити на даний протокол

Публічні вразливості. Дослідження використання OAuth компаніями Facebook, Google та Microsoft показало наявність певних проблем. Виявлені вразливості в протоколі OAuth, які стосуються його використання на різних веб-сайтах та додатках. Вони можуть призвести до зловживання, злому облікового запису або втрати конфіденційних даних користувача. Зокрема в [12] проаналізовано 27 провідних постачальників ідентифікації OAuth2.0 і виявлено, що 19 з них є вразливим до атак.

Facebook та Google були у фокусі уваги з приводу вразливостей OAuth. Наприклад, у 2018 році Facebook заявив про викриття вразливості в протоколі OAuth, яка давала можливість зловмисникам отримати доступ до персональних даних користувачів [12]. У результаті було скомпрометовано близько 50 мільйонів облікових записів користувачів.

У 2019 році Google заявив про вразливість в протоколі OAuth, яка дозволяла зловмисникам отримувати доступ до електронної пошти, календарів та інших конфіденційних даних користувачів. Ця вразливість знайшла своє застосування в атаках на політиків та активістів, а також у шпигунських операціях. У обох випадках компанії прийняли заходи для виправлення вразливостей та підвищення безпеки своїх систем. Однак, ці приклади показують, що протокол OAuth може бути вразливим і необхідно вживати заходів для його захисту. У лютому 2021 року було опубліковано доповідь від фахівців з безпеки, де були виявлені п'ять вразливостей у реалізації протоколу OAuth2.0 на більшості веб-сайтів, що використовують Google як провайдера [12]. Деталі цих вразливостей були передані Google та відповідним веб-сайтам для виправлення. Одна з вразливостей, яку було виявлено, полягає у недостатній перевірці доменного імені, яке повертається під час автентифікації. Це може дати змогу зловмиснику створити фішинговий сайт зі схожим доменним іменем та залучити користувачів до введення своїх облікових даних. Інша вразливість полягає у можливості отримання токенів доступу без необхідності отримувати підтвердження від користувача. Це може дати змогу зловмиснику отримати доступ до ресурсів користувача без його згоди. Крім того, було виявлено вразливість у методі автентифікації "Implicit Grant" [12], який дозволяє отримувати токени доступу без отримання "Client Secret". Це може дати змогу зловмисникам отримати токен доступу і зловживати ним без знання власника ресурсу. Інші вразливості були пов'язані з використанням неправильних URI-адрес для перенаправлення користувача на веб-сайт провайдера, недостатньою перевіркою внутрішніх токенів та використанням застарілих версій протоколу OAuth. Загалом, ці вразливості підкреслюють необхідність дотримання належної безпеки при використанні протоколу OAuth, а також важливість забезпечення безпеки від провайдерів послуг, які надають доступ до ресурсів та належно конфігурувати сервер.

Протокол OAuth 2.0 [5,6,8] – це відкритий протокол авторизації, який дозволяє користувачам надавати доступ до своїх облікових записів на різних веб-сайтах та додатках, не передаючи при цьому свої логіни та паролі. Протокол OAuth 2.0 є розширенням протоколу OAuth 1.0, відмінність якого полягає в тому, що OAuth 2.0 є простішим у використанні та підтримує більш широкий спектр застосувань.

Рисунок 1 – Абстрактна схема протоколу OAuth2.0

Розглянемо опис послідовності кроків на цій діаграмі:

– Клієнт запитує у власника ресурсу доступ до ресурсів, вказуючи тип доступу, який необхідний для виконання запиту.

– Власник ресурсу надає дозвіл на доступ до ресурсів, використовуючи спеціальне діалогове вікно авторизації. Це діалогове вікно може бути спеціальним для кожного ресурсу, або ж реалізуватися універсальним шляхом, наприклад, через механізм Single Sign-On.

– Клієнт надсилає до сервера авторизації (API) запит щодо авторизаційного токену. При цьому клієнт надає про себе інформацію.

– Сервер авторизації виконує перевірку клієнта і у випадку успішного її результату та отримання дозволу на авторизацію створює токен доступу для програми. На цьому відбувається завершення процесу авторизації.

– Клієнт здійснює запит захищеного ресурсу у сервера ресурсів, при цьому надає відповідний токен доступу.

– Сервер ресурсів виконує перевірку токену доступу, переданого клієнтом, та, якщо токен коректний, надає доступ до ресурсів.

Фактичний порядок кроків описаного процесу може відрізнятися залежно від типу дозволу на авторизацію, що використовується, але загалом процес буде виглядати описаним чином.

Перед тим, як почати використовувати OAuth в додатку, в розділі "developer" або "API" сайту сервісу необхідно здійснити реєстрацію, де вказуються такі дані додатку:

– його назва;

– сайт;

– Redirect URL (callback URL) – це URL, за яким буде відбуватися перенаправлення сервісом користувача за результатом успішного проходження авторизації або ж відмови в ній.

У разі успішної реєстрації додатка сервісом будуть створені наступні

Після реєстрації додатка сервіс створить профіль клієнта у вигляді:

– ідентифікатора клієнта (client ID) – рядка з публічним доступом, який буде використовувати API сервісом при ідентифікації додатка та для формування URL для авторизації користувачів.

– секрет клієнта (client secret), який може бути відомим лише додатку та API і призначений для автентифікації достовірності додатка для сервісу API при запиті цим додатком доступу до акаунту користувача.

Дозвіл на авторизацію залежить від методу запиту авторизації, що використовується додатком, та від того, які типи дозволу підтримує API. В рамках OAuth 2 доступні чотири типи запитів, кожен з яких є ефективним у певних ситуаціях [8]:

– запит коду авторизації (Authorization Code), що застосовується з серверними додатками (server-side applications).

– неявний (Implicit) запит, який стосується додатків, що працюють на пристроях користувачів, таких як мобільні та веб-додатки.

– Resource Owner Password Credentials – це тип запиту у OAuth 2, який використовують довірені додатки, наприклад ті, що є частиною сервісу. В цьому випадку, облікові дані власника ресурсу передаються безпосередньо через додаток для отримання доступу до ресурсу.

– Облікові дані клієнта (Client Credentials), які можуть бути використані у випадку доступу додатка до API.

OAuth2.0 включає в себе ряд типів дозволів (grants), які дозволяють забезпечити різні рівні доступу до ресурсів [8]:

1. *Authorization Code Grant* – цей тип дозволу використовується для взаємодії між додатком та сервером авторизації. Після того, як користувач дозволяє додатку отримати доступ до своїх ресурсів, сервер авторизації надсилає додатку authorization code, який дозволяє додатку отримати access token.

До найбільш поширених типів запитів належить код авторизації. Це особливо актуально, для серверних додатків, для яких вихідний код програми та секрети клієнта не можуть бути доступні стороннім особам. У цьому випадку процес авторизації базується на перескеруванні (redirection), що передбачає взаємодію програми зі спеціальним агентом (user-agent), наприклад веб-браузером, щоб отримати коди авторизації API та передати їх через зазначений агент для користувача.

2. *Implicit Grant* – механізм авторизації, який використовується для клієнтів, які працюють на стороні клієнта, наприклад, JavaScript-додатки, які працюють у веб-браузері. Після того, як користувач успішно авторизувався, сервер авторизації відправляє токен доступу клієнту без отримання authorization code.

3. *Resource Owner Password Credentials Grant* – цей тип дозволу використовуєтьсядля додатків, які мають повну довіру до користувача. Додаток отримує логін та пароль від користувача та обмінює їх на access token. Цей механізм авторизації не рекомендується використовувати, оскільки він передбачає передачу пароля користувача на сторонній сервер.

4. *Client Credentials Grant* – цей тип дозволу використовується для додатків, які не потребують доступу до ресурсів конкретного користувача. Додаток отримує access token без участі користувача.

Крім того, в OAuth 2.0 передбачена можливість використовувати декілька механізмів авторизації одночасно, наприклад, дозвіл на основі коду та дозвіл на основі логіна та пароля користувача. Такий підхід називається комбінованим (hybrid) механізмом.

Усі ці механізми дають змогу забезпечити безпеку при доступі до різних ресурсів та API. Однак, недбале використання цих механізмів може призвести до вразливостей та можливості несанкціонованого доступу до даних користувачів. Тому важливо використовувати OAuth 2.0 з дотриманням усіх вимог та правил застосування.

Аналіз відомих вразливостей та атак. Оскільки, як було згадано вище, в OAuth 2.0 передбачена можливість авторизації за допомогою пароля не належить до надійних методів, тому, що доволі часто паролі легко можна підібрати, і користувачі переважно використовують прості й однакові паролі в різних системах або записують їх на папері. Коли зловмисник дізнається пароль, користувач зазвичай про це навіть може не здогадуватися. Крім того, розробники додатків інколи допускають концептуальні помилки, що роблять облікові записи уразливими. Найпоширеніші вразливості при використанні автентифікації за паролем можуть бути такими:

– паролі можуть бути підібрані, особливо якщо вони слабкі, якщо веб-додаток не захищений від можливості перебору паролів, чи якщо паролі згенеровані та поширені безпосередньо веб-додатком та не змінені користувачем після першого входу;

– розробники можуть припуститися помилки в захисті облікових даних, наприклад, зберігати паролі у відкритому вигляді або передавати їх по мережі у незахищеному вигляді, по незахищеному HTTP з'єднанню або в рядку URL;

– атаки на логін-форми можуть включати фішингові атаки, в яких користувачів можна переконати вводити свої облікові дані на фальшивих сторінках;

– для зберігання паролів веб-додатком не використовуються безпечні хеш-функції;

– міжсайтовий скриптінг (XSS) може використовуватись для крадіжки паролів, особливо якщо вони зберігаються у браузері користувача;

– атаки на сесії можуть використовуватись для отримання доступу до облікового запису, особливо, якщо сесійні токени передаються у незахищеному вигляді;

– для відновлення паролю у веб-додатку використовується вразлива функція, через яку можна отримати несанкціонований доступ до інших облікових записів;

– session tokens створені веб-додатком таким чином, що їх можна підібрати або передбачити для інших користувачів;

– веб-додаток вразливий до атак з фіксації сесії (session fixation), оскільки не генерує новий токен сесії при переході від анонімної до автентифікованої сесії користувача;

– веб-додаток не завершує автоматично сесію користувача після того, як він довший час є неактивним або не має можливості здійснити вихід з автентифікованої сесії;

– зловмисники можуть використовувати XSS атаки для вставки скриптів на сторінки веб-додатку, що дозволить отримати доступ до облікових даних користувачів, які входять на сайт;

– зловмисники можуть перехоплювати комунікацію між користувачем і сервером, і отримувати облікові дані користувача використовуючи Man-in-the-middle атаки.

Розглянемо найбільш поширені атаки на реалізації OAuth [13]:

1. Неповна перевірка URI перенаправлення

У деяких серверів авторизації передбачена можливість реєстрації шаблонів для URI перенаправлення замість повних URI. У такому разі відбувається порівняння фактичного значення параметра URI перенаправлення в кінцевій точці авторизації із даним шаблоном. Завдяки цьому клієнти мають можливість закодувати додаткову інформацію про транзакцію у параметрах URI перенаправлення або виконати реєстрацію одного шаблону для декількох URI перенаправлення, що забезпечує більшу гнучкість. З одного боку, це прискорює процес перевірки, але з іншого боку, такий метод є складнішим для втілення і може призвести до більшої кількості помилок, ніж при перевірці звичайних URI перенаправлень. Також є відомі ряд атак, що базуються на уразливостях в реалізації шаблону. Даний недолік ефективно компрометує процес ідентифікації або автентифікації клієнта (в залежності від типу гранту та клієнта) та дає можливість зловмиснику отримати код авторизації або токен доступу:

– використовуючи безпосередню відправку агента користувача до URI, що контролюється зловмисниками;

– коли відкрите перенаправлення на клієнті поєднується з обробкою фрагментів URL-адреси агентами користувача, це може призвести до розкриття облікових даних OAuth зловмисникам.

2. Атака на надання коду авторизації

Якщо клієнт є загальнодоступним та використовує кодовий тип гранту, то атака буде виглядати таким чином. Припустимо, що URL-адреса перенаправлення «https://*. page.sample/*» зареєстрована для клієнта ID «n3HfLNpqw7».

Даний патерн дає змогу виконувати перенаправлення URI, які містять будь-який хост у домені page.sample. Тому, якщо зловмиснику вдається створити свій хост або субдомен у домені page.sample, то він може маскуватися під законного клієнта. Розглянемо ситуацію, коли зловмисник створює хост «intruder. page.sample». Тоді, щоб успішно провести атаку:

– Для успішної атаки зловмиснику необхідно переконати користувача відкрити підроблений URL у своєму браузері, що запускає сторінку, під спостереженням атакуючого, наприклад, «https://www. intruder.sample».

– Даною URL-адресою буде ініційований запит авторизації з ідентифікатором законного клієнта до фінальної точки авторизації. Як прикад можна навести наступний запит:

GET/authorize?response_type=code&client_id =n3HfLNpqw7&state=xyz&redirect_uri= https%3 A%2F%2Fintruder.page.sample% 2Fcb HTTP/1.1

Host: server.page.sample

– Сервером авторизації буде проведена перевірка URI перенаправлення для ідентифікації клієнта. Так як шаблон дозволяє використовувати будь-які імена доменів в форматі "page.sample", запит на авторизацію обробляється на основі ідентифікації законного клієнта. Можливість зловмисника виконати атаку залежить від того, чи є можливість автоматично схвалювати запити на згоду користувача. Якщо цей варіант дозволений (але це не рекомендується для публічних клієнтів), то атака може бути проведена ще простіше.

– Якщо користувачем атака не буде ідентифікована, то «код» авторизації буде видано та передано безпосередньо до клієнта зловмисника.

– Зловмисник, який уособлює публічного клієнта, може отримати токени, обмінявши код на відповідній кінцевій точці токена. Однак ця атака не буде працювати для конфіденційних клієнтів, оскільки обмін кодами потребує автентифікації з легітимним секретом клієнта. Для того, щоб отримати код зловмисник змушений буде використати законного клієнта (наприклад, використовуючи ін'єкції коду). Також варто зазначити, що уразливості такого роду також можуть існувати, якщо сервер авторизації правильно обробляє символи підстановки.

3. Атака на неявний грант

Ця атака може бути використана для отримання доступу до ресурсів, які потребують авторизації, за допомогою неявного надання гранту. Якщо зловмисник може надіслати відповідь серверу авторизації, в якій міститься URI-адреса під його управлінням, то він може одержати фрагмент з токеном доступу безпосередньо. Також, неявні клієнти можуть стати жертвами іншого типу атак. Цей вид атак використовує той факт, що користувачі додають фрагменти до URL-адреси при переході за посиланням, якщо вони не містяться у заголовку розташування. Ця атака використовує функцію відкритого переадресування, коли клієнт переадресовує користувача на іншу сторінку, і поєднує її з тим, що клієнт може додати фрагмент до URL-адреси, який може обійти обмеження на шаблони URI переадресації, хоча перевірка URL все ще застосовується. Нехай для клієнта «n3HfLNpqw7» маємо наступний шаблон «https://client.page.sample/at?*», Це означає, що в «https://client.page.sample/at» може бути перенаправлено будь-який параметр. Негативною стороною є те, що клієнт, який буде виконувати відкриту переадресацію відкриє себе для атаки. Цей кінцевий вузол виконує підтримку параметра "redirect_to", який отримує кінцеву URL-адресу. На жаль, клієнт який виступає в ролі відкритого переадресувача, відкриває себе для атаки. Ця кінцева точка підтримує параметр "redirect_to", який приймає цільову URL-адресу, і за допомогою HTTP-заголовка перенаправлення "Location 303", браузер буде автоматично перенаправлений на цю URL-адресу. Для успішної реалізації атаки зловмиснику необхідно:

– Подібно до вищенаведеної атаки, зловмисник змушений обманути користувача, переконуючи його відкрити підроблену URL-адресу у браузері, який виконає запуск сторінки під спостереженням атакуючого, наприклад, "https://www.intruder.sample".

– URL ініціює запит авторизації, який дуже схожий на запит під час атаки кодового потоку. Як відмінності, він використовує відкритий переадресовувач, кодуючи «redirect_to=https://intruder.sample» в параметрах URI перенаправлення, і використовує тип відповіді «токен»:

GET /authorize?response_type=token&client_id=n3HfL Npqw7&state=xyz&redirect_uri= https%3A%2F%2Fclient.page.sample%2Fat%26red irect_to%253Dhttps%252A%252 F%252Fintruder.sample%252Fat HTTP/1.1

Host: server.page.sample

– Враховуючи те, що URI перенаправлення має повну відповідність шаблону, який зареєстрований, то сервер авторизації дозволить запит і надішле отриманий токен доступу з перенаправленням 303

HTTP/1.1 303 Found

Location: https://client.page.sample/at?

redirect_to%3Dhttps%3A%2F%2Fclient.intrud er.sample%2Fat

#access_token=2YotnFZFEjr1zCsicMWpAA&..

На сайті sample.com запит попаде до

переадресовувача, що є відкритим, який зчитає параметри перенаправлення та згенерує HTTP заголовок перенаправлення з кодом 303 на вказану URL-адресу "https://intruder.sample.com/".

HTTP/1.1 303 Found

Location: https://intruder.sample/

– Оскільки переадресація на client.page.sample не містить фрагмент у заголовку Location, то агентом користувача буде повторне приєднаня оригінального фрагменту «#access_token=2FsPcgVFRjr5dCgfdXZlnFF&...» до URL і перейде до URL-адреси: https://intruder.sample/#access_token=2FsPcgVFRjr5dCgfdXZlnFF&...

– Сторінка зловмисника на intruder.sample може отримати доступ до фрагмента і отримати токен доступу.

4. Можливість витоку «кодів» або станів через Referrer заголовки зі сторони клієнта або сервера авторизації

Якщо сторонній сайт містить посилання на веб-сайт клієнта або на веб-сайт сервера авторизації, то навіть без навмисних дій може статися витік кодів авторизації або значень стану через Referrer заголовок.

OAuth може бути вразливим до витоку з клієнта, якщо після успішної авторизації, клієнт передає сторінку, яка:

– містить треті сторони (зображення), одним з прикладів може бути розміщений на сторінці створений користувачем блог;

– вміщає в собі посилання на інші сторінки, що знаходяться під спостереженням зловмисника (оголошення, FAQ тощо), а користувач перейде за таким посиланням.

При переході браузера на сторінку атакуючого або завантаженні вмісту з третіх сторін, зловмисник дізнається URL-адресу відповіді на авторизацію та може отримати доступ до коду або стану.

Якщо сервер авторизації містить посилання або вміст третіх сторін на кінцевій точці авторизації, то в разі витоку з сервера, зловмисник може отримати доступ до стану. Як результат, якщо зловмисник отримує доступ до стану, захист CSRF, який було здійснено за допомогою цього стану, порушується, що може призвести до атак CSRF.

5. Скомпрометований сервер ресурсів

Зловмисник може використати різні методи для зламування сервера ресурсів і отримання доступу до його вмісту та вмісту інших серверів, що пов'язані з ним. Ці методи можуть включати доступ до журналів сервера або навіть повне придбання контролю над сервером. Якщо зловмисник отримує повний доступ до сервера, включаючи оболонку, він може уникнути всіх контролів доступу та без обмежень отримати доступ до ресурсів. У разі компрометації системи зловмисник може отримати доступ до токенів доступу, які передаються до цієї системи і можуть бути використані для отримання доступу до інших серверів ресурсів. Навіть якщо зловмиснику вдасться отримати обмежений доступ до лог-файлів або баз даних серверної системи, він може взяти на себе контроль над дійсними токенами доступу. Зміцнення і моніторинг серверних систем є стандартною процедурою для запобігання порушенням сервера. Важливо звернути увагу на наслідки порушень безпеки OAuth, які можуть призвести до відтворення викрадених токенів доступу на скомпрометованих серверах ресурсів та інших серверах відповідного розгортання. Щоб запобігти відтворенню токенів, рекомендується застосовувати такі заходи захисту:

– Сервер ресурсів повинен відноситися до токенів доступу, аналогічно як і до будь-яких інших облікових даних. З метою підвищення безпеки бажано їх не реєструвати та не зберігати у форматі звичайного тексту.

– З метою недопущення відтворення токенів доступу на інших серверах ресурсів необхідно використовувати токени доступу обмеження відправника. Залежно від глибини проникнення, це також запобігає відтворенню компрометованої системи.

– Обмеження аудиторії, може бути використано для запобігання відтворенню захоплених токенів доступу на інших серверах ресурсів.

6. Фішинг токенів доступу шляхом підробки серверу ресурсів

Зловмисник може створити власний сервер ресурсів та надіслати клієнту підроблені токени доступу до інших серверів ресурсів. Якщо клієнт відправить дійсний токен доступу на цей підроблений сервер ресурсу, то зловмисник матиме можливість використовувати цей токен для доступу до інших сервісів від імені власника ресурсу. Ця атака використовує той факт, що клієнт не пов'язаний з конкретною URL-адресою сервера ресурсів під час розробки, але пізніше, під час виконання, екземпляри клієнта уже пов'язуються з нею. Цей пізній зв'язок є типовим для випадків, коли клієнт використовує стандартний API і налаштований користувачем або адміністратором для використання відповідно до стандартів, що запроваджені в компанії або конкретним користувачем.

7. Метадані

Для клієнта існує можливість отримання від сервера авторизації додаткової інформації про місцезнаходження, де безпечно використовувати його токени доступу. Найпростіший варіант передбачає, що для реалізації цього методу необхідно, щоб сервер авторизації мав

опублікований список всіх відомих серверів ресурсів. Також сервер авторизації може повернути URL-адреси, для яких підходитиме токен доступу. Проте варто зазначити, що ця стратегія пом'якшення покладатиметься на те, що клієнт запровадить політику безпеки та надсилатиме лише маркери доступу законним адресатам. Хоча як свідчить практика в реальності це не завжди так.

CVSS calculator та оцінка вразливостей протоколу OAuth. Для дослідження вразливостей протоколу OAuth ми використовуємо CVSS калькулятор [14, 15].

CVSS (Common Vulnerability Scoring System) калькулятор використовує числову оцінку, щоб допомогти аналізувати та оцінювати ризики, пов'язані з вразливостями в програмному забезпеченні. Цей інструмент використовується для стандартизації оцінки вразливостей та допомагає вирішити, наскільки серйозною є вразливість та які заходи потрібно прийняти для зниження ризику. CVSS калькулятор використовує формулу, щоб призначити числову оцінку вразливості. Ця формула враховує різні фактори, зокрема:

1. Базові метрики. Ці метрики включають наскільки легко вразливість може бути експлуатована, які можливі наслідки для конфіденційності, цілісності та доступності даних, а також наскільки складно експлуатувати вразливість.

– Конфіденційність відображає наскільки легко атакуючий може отримати доступ до конфіденційної інформації, яка зберігається в системі або передається через неї.

– Цілісність відображає наскільки легко атакуючий може змінити, чи впровадити недостовірну інформацію або видалити інформацію з системи.

– Доступність відображає наскільки легко атакуючий може перешкоджати нормальному функціонуванню системи, шляхом блокування доступу користувачів до неї.

2. Темпоральні метрики. Ці метрики включають терміни, пов'язані з виправленням вразливості, тобто наскільки швидко стане відомо про вразливість та наскільки часто вона використовується.

3. Окремі метрики. Ці метрики допомагають врахувати особливості середовища, такі як важливість атакованих даних та доступність інструментів для проведення атаки.

Кожна з цих метрик має свій власний рівень важливості, який враховується при розрахунку оцінки вразливості. Після обчислення кожної метрики, CVSS калькулятор визначає числову оцінку від 0 до 10, де 10 – найбільш серйозна вразливість.

При оцінюванні кожної метрики були використані такі параметри і їх можливі значення:

– Атакування складність (Attack Complexity). Низький рівень (Low) – оскільки атака вимагає виконання спеціально підготовленого запиту.

– Складність автентифікації (Authentication). Низький рівень (Low) – зловмиснику не потрібно автентифікуватися для експлуатації вразливості.

– Вплив на конфіденційність (Confidentiality Impact). Високий рівень (High) – оскільки зловмисник може отримати доступ до конфіденційної інформації, такої як логіни та паролі користувачів.

– Вплив на цілісність (Integrity Impact). Низький рівень (Low) – зловмисник не може змінювати дані на ресурсі.

– Вплив на доступність (Availability Impact). Низький рівень (Low) – атака не впливає на доступність ресурсу.

Після обчислень калькулятор видає коротку абревіатуру (CVSS Vector: CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H), яка пояснює параметри оцінки. Таким чином ми вираховуємо кожну оцінку. Загальна оцінка вразливості визначається на основі розрахунку оцінки для кожної з цих категорій, а також врахування рівня важливості кожної з них. Оцінка вища за 7.0 вважається серйозною, а оцінка більше 9.0 може бути визнана критичною. Результати проведеного оцінювання вразливостей наведено в таблиці нижче.

Таблиця 1

Оцінювання вразливостей протоколу OAuth 2.0 за допомогою CVSS калькулятора

Вразливість	Оцінка
Недостатня перевірка URI перенаправлення	6.1
Атака на надання коду авторизації	8.8
Атака на неявний грант	6.5
Можливість витоку «кодів» або станів через Referrer заголовки зі сторони клієнта або сервера авторизації	5.3
Скомпрометований сервер ресурсів	9.1
Фішинг токенів	6.1
Метадані	6.1

Рекомендації із захисту. OAuth 2.0 – це потужний інструмент для авторизації та автентифікації користувачів у додатках. Проте, недотримання стандартів безпеки може призвести

до небезпек для користувачів і систем. Ми пропонуємо такі рекомендації щодо безпечної реалізації OAuth 2.0:

– Слід використовувати найновіші версії OAuth 2.0. Це дозволяє уникнути вразливостей, які вже були виявлені та виправлені у більш нових версіях.

– Необхідно конфіденційні дані, такі як ключі та паролі, захищати від несанкціонованого доступу. Для цього використовується шифрування та зберігання даних в безпечних місцях.

– Потрібно використовувати тільки протокол HTTPS. Це дозволить захистити передачу даних між користувачем та сервером від несанкціонованого доступу.

– Необхідно виконувати перевірку URI перенаправлень. URI перенаправлення повинно відповідати тому, що було відправлено під час запиту авторизації. Це запобігає атакам, пов'язаним з перенаправленням.

– Варто обмежити права доступу. Для користувачів мають бути надані тільки необхідні права доступу. Це зменшує ризик несанкціонованого доступу до систем.

– Необхідно виконувати моніторинг активності систем та реагувати на будь-яку підозрілу діяльність. Це дозволяє вчасно виявляти можливі вразливості.

Висновки. В даній роботі досліджувалася безпека OAuth 2.0 – доволі популярного протоколу авторизації, який дозволяє користувачам надавати доступ до своїх ресурсів іншим додаткам без необхідності надання свого логіну та паролю. Даний протокол має кілька типів дозволів, які надають можливість забезпечити різні рівні доступу до ресурсів.

В результаті проведеного на основі оцінки вразливостей, використовуючи CVSS калькулятор, аналізу було виявлено, що протокол OAuth 2.0 загалом є безпечним, оскільки забезпечує надійну автентифікацію, авторизацію та цілісність сеансу. Однак, це можливо тільки у випадку, якщо користувачі дотримуються рекомендацій та найкращих практик безпеки OAuth. Аналіз охоплював сам стандарт протоколу, його режими (типи грантів) і доступні опції, враховував зловмисні ресурсні провайдери, а також пошкоджені браузери.

За результатом перевірки атак та ризиків, який вони несуть ми запропонували конкретні рекомендації, які мають особливу актуальність, оскільки OAuth є однією з найбільш широко розгорнутих систем авторизації та автентифікації в Інтернеті та є основою для інших протоколів. Виконання рекомендацій забезпечує безпеку реалізації OAuth, тоді як їх ігнорування не може гарантувати безпеку протоколу.

Список літератури:

1. Bucko A, Vishi K, Krasniqi B, Rexha B. Enhancing JWT Authentication and Authorization in Web Applications Based on User Behavior History. Computers. 2023, 12(4):78. https://doi.org/10.3390/computers12040078

2. Maksymovych V, Nyemkova E, Justice C, Shabatura M, Harasymchuk O, Lakh Y, Rusynko M. Simulation of Authentication in Information-Processing Electronic Devices Based on Poisson Pulse Sequence Generators. Electronics. 2022, 11(13):2039. https://doi.org/10.3390/electronics11132039.

3. Papathanasaki M, Maglaras L and Ayres N (2022) Modern Authentication Methods: A Comprehensive Survey. AI, Computer Science and Robotics Technology. IntechOpen. DOI: 10.5772/acrt.08, p.24.

4. О. Гарасимчук, І.Р. Опірський, Я. Совин, І. Тишик, Є. Штефанюк. Організація захисту результатів контролю знань в системах дистанційного навчання» / Кібербезпека: освіта, наука, техніка. 2020, вип. 2, вип. 10. С. 144-157.

5. Parecki A. The Little Book of OAuth 2.0 RFCs, (2022), p. 412. ISBN: 979-8-607-50395-6.

6. OAuth 2.0 Simplified, (2018), p. 240. (Lulu.com). – (ISBN 1387751514).

7. Chae, Cheol-Joo, Ki-Bong Kim and Han-Jin Cho. "A study on secure user authentication and authorization in OAuth protocol." Cluster Computing, Volume 22, Issue 1, (2019) pp 1991–1999, Available at:https://doi.org/10.1007/s10586-017-1119-6.

8. Hardt D., The OAuth 2.0 Authorization Framework. Microsoft Internet Engineering Task Force (IETF). Ed. Request for Comments: 6749 *Microsoft* Obsoletes: 5849 (October 2012) Category: Standards Track ISSN 2070-1721. P.76. Available at: datatracker.ietf.org/doc/html/rfc6749 #section-2.1/.

9. Yang, Feng and Sathiamoorthy Manoharan. "A security analysis of the OAuth protocol." 2013 IEEE Pacific Rim Conference on Communications, Computers and Signal Processing (PACRIM) (2013): pp. 271-276.

10. Srikanth, V., Jupalli Sneha Latha, Dinne Ajay Kumar, and Kakarla Uma Maheswari. "A survey on OAUTH protocol for security." International Journal of Engineering & Technology 7, no. 1.1 (December 21, 2017): 692. Available at: http://dx.doi.org/10.14419/ijet.v7i1.1.10834.

11. Li, W., Mitchell, C. J. & Chen, T. Your Code Is My Code: Exploiting a Common Weakness in OAuth 2.0 Implementations. In: Security Protocols XXVI. Security Protocols (2018). pp. 24-41. Cham, Switzerland: Springer (2018). ISBN 9783030032500 doi: 10.1007/978-3-030-03251-7_3.

12. Rahat, Tamjid Al, Yu Feng and Yuan Tian. "Cerberus: Query-driven Scalable Vulnerability Detection in OAuth Service Provider Implementations." Proceedings of the 2022 ACM

SIGSAC Conference on Computer and Communications Security (2021): n. pag.

13. Lodderstedt T., Bradley J., Labunets A., and Fett D. OAuth 2.0 Security Best Current Practice, (2022), p.16. Available at: tools.ietf.org/html/draft-ietf-oauth-security-topics-19.

14. Mell, Peter. “Measuring the Common Vulnerability Scoring System Base Score Equation.” (2022), p. 52. Available at: https://doi.org/10.6028/NIST.IR.8409.

15. Mell P., Scarforne K., Romanosky S. A Complete Guide to the Common Vulnerability Scoring System Version 2.0 (CVSS). (2007). 23 p. Available at: https://www.first.org/cvss/v2/guide.

References:

1. Bucko A, Vishi K, Krasniqi B, Rexha B. Enhancing JWT Authentication and Authorization in Web Applications Based on User Behavior History. Computers. 2023; 12(4):78. https://doi.org/10.3390/computers12040078

2. Maksymovych V, Nyemkova E, Justice C, Shabatura M, Harasymchuk O, Lakh Y, Rusynko M. Simulation of Authentication in Information-Processing Electronic Devices Based on Poisson Pulse Sequence Generators. Electronics. 2022, 11(13):2039. https://doi.org/10.3390/electronics11132039.

3. Papathanasaki M, Maglaras L and Ayres N (2022) Modern Authentication Methods: A Comprehensive Survey. AI, Computer Science and Robotics Technology. IntechOpen. DOI: 10.5772/acrt.08, p.24.

4. Harasymchuk O.I., Opirskyy I.R., Sovyn Ya.R., Tyshyk I.Ya., Shtefanyuk Ye.F. Organization of protection of knowledge control results in distance learning systems" / Cybersecurity: education, science, technology. 2020, vol. 2, issue 10. P. 144-157,.

5. Parecki A. The Little Book of OAuth 2.0 RFCs, (2022), p. 412. ISBN: 979-8-607-50395-6.

6. OAuth 2.0 Simplified, (2018), p. 240. (Lulu.com). – (ISBN 1387751514).

7. Chae, Cheol-Joo, Ki-Bong Kim and Han-Jin Cho. “A study on secure user authentication and authorization in OAuth protocol.” Cluster Computing, Volume 22, Issue 1, (2019). Pp. 1991–1999, Available at:https://doi.org/10.1007/s10586-017-1119-6.

8. Hardt D., The OAuth 2.0 Authorization Framework. Microsoft Internet Engineering Task Force (IETF). Ed. Request for Comments: 6749 Microsoft Obsoletes: 5849 (October 2012) Category: Standards Track ISSN 2070-1721. P.76. Available at: datatracker.ietf.org/doc/html/rfc6749 #section-2.1/.

9. Yang, Feng and Sathiamoorthy Manoharan. “A security analysis of the OAuth protocol.” 2013 IEEE Pacific Rim Conference on Communications, Computers and Signal Processing (PACRIM) (2013). Pp. 271-276.

10. Srikanth, V., Jupalli Sneha Latha, Dinne Ajay Kumar, and Kakarla Uma Maheswari. "A survey on OAUTH protocol for security." International Journal of Engineering & Technology 7, no. 1.1 (December 21, 2017): 692. Available at http://dx.doi.org/10.14419/ijet.v7i1.1.10834.

11. Li, W., Mitchell, C. J. & Chen, T. Your Code Is My Code: Exploiting a Common Weakness in OAuth 2.0 Implementations. In: Security Protocols XXVI. Security Protocols (2018). pp. 24-41. Cham, Switzerland: Springer (2018). ISBN 9783030032500 doi: 10.1007/978-3-030-03251-7_3.

12. Rahat, Tamjid Al, Yu Feng and Yuan Tian. “Cerberus: Query-driven Scalable Vulnerability Detection in OAuth Service Provider Implementations.” Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security (2021): n. pag.

13. Lodderstedt T., Bradley J., Labunets A., and Fett D. OAuth 2.0 Security Best Current Practice, (2022), p.16. Available at: https://tools.ietf.org/html/draft-ietf-oauth-security-topics-19.

14. Mell, Peter. “Measuring the Common Vulnerability Scoring System Base Score Equation.” (2022), p. 52. Available at: https://doi.org/10.6028/NIST.IR.8409.

15. Mell P., Scarforne K., Romanosky S. A Complete Guide to the Common Vulnerability Scoring System Version 2.0 (CVSS). (2007). 23 p. Available at: https://www.first.org/cvss/v2/guide.

Науково-методична стаття.
Надійшла до редакції 26.04.2023.
Прийнято до публікації 18.05.2023.