УДК 004.4

Тизунь Віталій

ВРАЗЛИВОСТІ БЕЗПЕКИ ТА КОНФІДЕНЦІЙНОСТІ ІНФОРМАЦІЇ ПРИ ВИКОРИСТАННІ ПОПУЛЯРНИХ JAVASCRIPT БІБЛІОТЕК

У статті розглядаються основні вразливості безпеки та конфіденційності інформації на web-сервісах, пов'язані з використанням популярних JavaScript бібліотек. Особливо увагу звернено на вразливості, що приводять до витоку користувацької інформації.

Ключові слова: захист даних, web-сервіс, JavaScript, вразливість безпеки.
Бібл.: 13.

The article discusses about the main vulnerabilities of security and privacy of information on web services associated with the use of popular JavaScript libraries. Particular attention is paid to vulnerabilities that lead to the leak of user information.

Key words: data protection, web-service, JavaScript, security vulnerability.
Bibl.: 13.

Актуальність теми дослідження. При розробці багатьох web-сервісів для полегшення самого завдання розробки часто використовуються сторонні бібліотеки, які не завжди гарантують те, що реалізовані в них методи будуть абсолютно безпечними для використання. Дана проблема є доволі актуальною, так як навіть в популярних бібліотеках, що використовуються багатьма ресурсами часто знаходять вразливості.

Постанова проблеми. При роботі з сторонніми бібліотеками розробник ресурсу може не знати про вразливості, що містять певні методи використовуваної бібліотеки.

Аналіз останніх досліджень і публікацій. При дослідженнях великої кількості web-ресурсів було виявлено, що більшість із них містять вразливості, пов'язані з використанням популярних бібліотек.

Мета дослідження. Метою даного дослідження є виділення найвідоміших вразливостей безпеки, бібліотек, використання яких може призвести до них та методи запобігання даних вразливостей.

Викладення основного матеріалу. В даний час розробка засобами JavaScript неможлива без використання принаймні однієї з тисяч доступних бібліотек JavaScript із відкритим кодом. Бібліотеки роблять програмування в JavaScript простішим та ефективнішим шляхом прийняття процесів, які вимагають написання кількох рядків коду, зазвичай шляхом створення єдиної функції, яку можна викликати.

Проте ці переваги піддаються ризикам. Бібліотеки JavaScript можуть мати дефекти безпеки, які можуть зробити вразливим веб-сайт, що використовує їх.

Аналіз веб-ресурсів на наявність вразливостей. Згідно з дослідженням, проведеним Північно-Східним університетом в Бостоні [1], більше 37% веб-сайтів використовують щонайменше одну версію бібліотеки з відомими вразливостями.

Загальні вразливі місця безпеки в JavaScript включають міжсторінкові сценарії, перехресні запити через перемички та переповнення буферів. Перехресні сценарії

дозволяють злодіям вставляти шкідливий код у довірені веб-сторінки, які потім передають цей шкідливий код користувачам, які переглядають сайт. Підробка міжсторінкового запиту дозволяє злодіям використовувати браузер користувача для відстеження його активності на інших сайтах. Переповнення буфера відбувається, коли зловмисники надсилають дані, що занадто великі для зберігання в буфері, внаслідок чого дані записуються за межі буфера та дозволяють зловмисникам вставляти шкідливий код, пошкоджені дані або аварійно завершити роботу програми.

Згідно з публікацією блогу npm [2] загальноприйнятим є те, що сучасний проект JavaScript залежить від 700 до 1200 сторонніх пакетів, і задача пошуку інформації про вразливості в інтернеті являється неможливою для розробника. Саме тоді інструменти для аналізу складу програм стають необхідними та надзвичайно корисними.

Інструменти аналізу складу аналізують ваш код та знаходять вразливі компоненти. Це прискорює процес виявлення вразливостей вашого сайту, а також зменшує ризик людської помилки. В більшості випадків такі інструменти активно аналізують проект та сповіщають розробника при знаходженні вразливостей.

У дослідженні Північно-Східного університету було встановлено, що в середньому веб-сайт у своєму наборі даних використовує версії бібліотек, які на 1177 днів старіші, ніж найновіший реліз. Перехід на нову версію вимагає певного часу, тому що потрібно виконати певні зміни програмного коду та їх тестування, щоб перевірити сумісність нових версій пакетів з ресурсом.

Постійне оновлення до більш нових версій бібліотек старих проектів може заважати розробникам приділяти свій час новим проектам, але не можна залишати поточні сайти вразливими.

Інше дослідження, проведене командою Snyk [3], говорить про те, що 77% із досліджених командою 433000 сайтів містять певні вразливості. При цьому 51% з них дозволили собі дві відомі вразливості, а 9.2% - більше чотирьох.

Командою був складений список, які бібліотеки, що використовувались даними ресурсами містять в собі вразливості. Найпопулярнішою в списку являється бібліотека jQuery. 92.5% з числа веб-ресурсів з вбудованою jQuery використовують старі версії бібліотек з великим числом вразливостей. Друге місце займає jQuery UI, що застосовується в 19.9% досліджених ресурсів.

Серед інших вразливих бібліотек знаходяться Modernizr (використовується в 15.1% досліджених ресурсів), Bootstrap (13.7% ресурсів), yepnope (9.9% ресурсів).

Вразливості в перерахованих бібліотеках у більшості випадків пов’язана з міжсторінковим скриптуванням і підстановкою контенту. Вони можуть бути використані для визначення вмісту куків браузера при відкритті користувачем спеціально оформленого посилання.

Також слід відзначити випуск нового масштабного звіту від проекту Open Web Application Security [4]. В ньому дослідники в області інформаційної безпеки спробували класифікувати відповідні ризики, а також запропонували рейтинг найістотніших і популярних проблем, з якими доводиться стикатися веб-спільноті.

Лідируючі позиції в цьому списку впевнено утримують вразливості, пов'язані з появою неперевірених даних в складі виконуваних команд або запитів (SQL, NoSQL, OS, LDAP Injection). За ними слідують вразливості від

некоректної роботи механізмів ідентифікації та автентифікації, що дозволяють отримати доступ без авторизації.

Все більш актуальними стають вразливості, що призводять до витоку користувацьких персональних даних, вони займають третє місце. Також стала популярною нова категорія вразливостей, пов'язаних з некоректною обробкою зовнішніх посилань у XML-документах (атака класу XXE). П'яте місце в новому рейтингу відведено проблемам, що з'являються в оброблювачах списків доступу. Вони дозволяють виконувати операції, не передбачені діючими повноваженнями.

Інструменти аналізу вразливостей. Одним із інструментів пошуку вразливостей являється Snyk. Цей інструмент представляє чудовий інтерфейс для пошуку та перегляду вразливостей системи безпеки. Він також забезпечує інтеграцію з GitHub. Інструменти командного рядка, надані Snyk, не тільки повідомлять про вразливості компонентів, але також запропонують виправити їх за допомогою спеціального інструмента. Переходячи з командного рядка на веб-сайт Snyk можна легко перевірити модулі NPM на вразливості, переглянути інформаційну панель, що відображає вразливості у вашому поточному проекті та налаштувати параметри Github.

Іншим хорошим засобом являється Retire.js, який теж являється чудовим сканером вразливостей для бібліотек JavaScript. Хоча Retire.js не надає функцій веб-додатків або інтеграції Github, таких як Snyk, він використовує інші засоби. Даний інструмент перевіряє на вразливості не лише модулі NPM, які використовуються в Node.js, а й перевіряє бібліотеки JavaScript.

Retire.js запускається насамперед за допомогою інструмента командного рядка, але його можна використовувати різними способами:

- як grunt плагін,
- як gulp завдання,
- як розширення Chrome,
- як розширення Firefox,
- як burp плагін,
- як плагін OWASP Zap.

Висновки. Пошук та виявлення вразливостей в web-сервісах, що використовують JavaScript являється складною задачею із-за того, що при розробці часто бувають задіяні сторонні бібліотеки. Також при розробці серверної сторони проекту з використанням Node.js та NPM модулей задача сильно ускладнюється тим, що NPM модулі часто мають безліч залежностей від інших модулів і дуже важко прослідкувати за актуальністю версій та відсутністю вразливостей у всіх залежностях.

Однак існують засоби, які дозволяють спростити та автоматизувати процес пошуку та виправлення вразливостей в проекті.

Список використаних джерел

1. Thou Shalt Not Depend on Me: Analysing the Use of Outdated JavaScript Libraries on the Web. [Електронний ресурс]. — Режим доступу: http://www.ccs.neu.edu/home/arshad/publications/ndss2017jslibs.pdf

2. The npm Blog. Why use SemVer? [Електронний ресурс]. — Режим доступу: https://blog.npmjs.org/post/162134793605/why-use-semver

3. Snyk Blog: 77% of 433,000 Sites Use Vulnerable JavaScript Libraries. [Електронний ресурс]. — Режим доступу: https://snyk.io/blog/77-percent-of-sites-still-vulnerable/

4. OWASP Top 10 - 2017 The Ten Most Critical Web Application Security Risks. [Електронний ресурс]. — Режим доступу: https://www.owasp.org/images/7/72/OWASP_Top_10-2017_%28en%29.pdf.pdf

ДОВІДКА ПРО АВТОРІВ

Тизунь Віталій Юрійович – студент, кафедра обчислювальної техніки, Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського».

Tyzun Vitalii Yuriyovych – student, Department of Computer Engineering, National Technical University of Ukraine “Igor Sikorsky Kyiv Polytechnic Institute”.

E-mail: vitaliy.tyzun@gmail.com

Tyzun Vitalii

VULNERABILITIES OF SAFETY AND CONFIDENTIALITY OF INFORMATION WHEN USING POPULAR JAVASCRIPT LIBRARIES

Relevance of the research. When developing many web-services use third-party libraries, which does not always guarantee that the methods implemented in them will be completely safe for use

Target setting. When working with third-party libraries, the resource developer may not be aware of vulnerabilities that contain certain methods of the library used.

Actual scientific researches and issues analysis. When investigating a large number of web-services, it has been found that most of them contain vulnerabilities associated with the use of popular libraries.

The research objective. The purpose of this study is to identify the most prominent security vulnerabilities, libraries that provide them, and methods for preventing these vulnerabilities.

The statement of basic materials. At the moment, the development of JavaScript tools is not possible without the use of at least one of the thousands of available open source JavaScript libraries.

However, these benefits are risky, because JavaScript libraries may have security vulnerabilities.

Conclusions. Finding and detecting vulnerabilities in web services using JavaScript is a daunting task because third-party libraries are often involved in development. However, there are tools that can simplify and automate the process of finding and correcting vulnerabilities in the project.

Key words: data protection, web-service, JavaScript, security vulnerability.