КІБЕРБЕЗПЕКА: освіта, наука, техніка
CYBERSECURITY: EDUCATION, SCIENCE, TECHNIQUE
№ 3 (11), 2021
ISSN 2663 - 4023
DOI 10.28925/2663-4023.2021.11.6172
УДК 004.03

Ільєнко Анна Вадимівна
Кандидат технічних наук, доцент, доцент кафедри комп'ютеризованих систем захисту інформації
Національний авіаційний університет університет, Київ, Україна
ORCID ID: 0000-0001-8565-1117
ilyenko.a.v@nau.edu.ua

Ільєнко Сергій Сергійович
Кандидат технічних наук, доцент, доцент кафедри автоматизації та енергоменеджменту
Національний авіаційний університет університет, Київ, Україна
ORCID ID: 0000-0002-0437-0995
ilyenko.s.s@nau.edu.ua

Сташевський Данило Сергійович
магістр кафедри комп'ютеризованих систем захисту інформації
Національний авіаційний університет університет,Київ, Україна
ORCID ID: 0000-0001-9910-5504
121lak0sta121@gmail.com

ПРОГРАМНИЙ МОДУЛЬ ВІДСЛІДКОВУВАННЯ ПОМИЛОК У ВИСОКОНАВАНТАЖЕНИХ ВЕБ-ДОДАТКАХ НА БАЗІ ВИКОРИСТАННЯ АВТОРСЬКОГО АЛГОРИТМУ ЛОГЕРУ

Анотація. Дана стаття присвячена розгляду подальших актуальних шляхів забезпечення процедури відслідковування помилок у високонавантажених веб-додатках реалізованих мовою програмування Javascript. У статті досліджено та визначено, що помилки, які виникають при розробці та використанні сучасних високонавантажених веб-додатків є дуже небезпечними, оскільки впливають на повноцінну життєдіяльність інформаційної системи в цілому та можуть призводити до порушення конфіденційності та цілісності персональної інформації користувачів. В статті авторами розглянуті питання обробки помилок у мові програмування Javascript, проблема та необхідність відслідковування помилок у високонавантажених веб-додатках, поняття високонавантажених веб-додатків, існуючі методи та підходи відслідковування помилок, принципи побудови сучасних високонавантажених веб-додатків, порівняння існуючих рішень для відслідковування помилок у високонавантежних веб-додатках реалізованих мовою програмування Javascript. Результатом даних досліджень стало створення авторського програмного модулю відслідковування помилок у висконавантажених веб-додатках для вирішення проблеми логування помилок, аналіз логів на повноту, обробку помилок та вирішення їх в майбутньому. Також впровадження такого рішення дозволяє зменшити розмір програмного додатку для завантаження до 5 кілобайт та зберігати історію помилок. Розроблений програмний модуль відслідковування помилок у високонавантажених веб-додатках складається з двох частин клієнтської та серверної. Кожна частина є незалежним програмним модулем та може бути переконфігурована з мінімальними змінами конфігурації на будь-якому іншому ресурсі. Така реалізація дає змогу повністю збирати метрики про кожен XMLHTTP запит, збирати інформацію про оточення користувача в якому сталася помилка, збирати інформація про те, чим саме була викликана помилка, визначати конкретне місце, де сталася помилка при виконанні програмного коду, за допомогою власноруч розробленого алгоритму, зберігати історії помилок у журналі Kibana. Можливі напрямки розвитку цієї роботи пов'язані із розширенням алгоритму відслідковування помилок, для збору більшої кількості даних та удосконалення їх агрегації, на основі розширення метрик. Авторами в подальшому планується ряд науково технічних рішень розробки та впровадження ефективних методів, засобів забезпечення вимог, принципів та підходів забезпечення кібернетичної безпеки та організації захисту на основі використання авторських підходів відслідковування помилок у високонавантажених веб-додатках в дослідних комп'ютерних системах та мережах.

Ключові слова: високонавантажений веб-додаток, логер, javascript, kibana, відслідковування помилок.

ВСТУП

Актуальність. На сьогодні відбувається стрімкий розвиток інформаційних технологій та інтенсивна розробка високонавантажених веб-додатків. Таким чином з великими об'ємами кодової бази постає потреба у відслідковуванні помилок у високонавантажених веб-додатках. Оскільки велика кількість веб-додатків написані мовою програмування Javascript, у цій мові програмування вже є деякі інструменти для відслідковування помилок. Проте, коли розміри веб-додатку починають збільшуватися, відслідковувати усі помилки та аналізувати їх дуже важко. Мова Javascript та інструменти відслідковування помилок у готових веб-додатках написаних на мові програмування Javascript є основою даної статті. В роботі враховано, що помилки можуть мати різних характер та природу виникнення, від звичайних помилок які порушують цілісніть інтерфейсу, до вразливостей у системі безпеки веб-додатку. До найпоширених помилок у високонавантажених веб-додатках можна віднести помилки при взаємодії користувача з веб-додатком, помилки з авторизацією та автентифікацією користувачів, помилки які виникають при високому навантаженні на додаток, помилки з кешуванням даних, що призводить до порушення конфіденційності та цілісності персональної інформації користувачів. Тому можна сказати, що механізм для вчасного та якісного відслідковування помилок, на сьогоднішній день – це необхідний компонент у будь-якому веб-додатку.

Аналіз останніх досліджень і публікацій. Проаналізувавши існуючі рішення для відслідковування помилок у високонавантажених веб-додатках [1-3], можна дійти до висновку, що на сьогоднішній день існує лише два рішення які більш-менш задовольняють потреби відслідковування помилок. Проте навіть, ці два рішення мають свої недоліки. Аналіз останніх досліджень і публікацій дозволив сформувати вимоги, яке висуваються до програмного модулю відслідковування помилок у високонавантажених веб-додатках [5-8]. Враховуючи аналіз останніх досліджень і публікацій, актуальним є у розробці та впровадженні удосконаленого авторського модуля відслідковування помилок у високонавантажених веб додатках, який буде задовольняти вимогам сучасності.

Мета статті. Метою даної статті є дослідження відомих підходів відслідковування помилок у високонавантажених веб-додатках та існуючих практичних рішень, формування власних вимог до розробки авторського програмного модулю та визначення нового рішення відслідковування помилок у високонавантажених веб-додатках за рахунок впровадження авторського логеру. Такий підхід направлений на виявлення помилок та є захищеним від стороннього втручання, дозволяє аналізувати місце, де саме сталася помилка та фіксувати будь-які інші помилки на серверній та клієнтській частинах. Кінцевим результатом роботи планується представлення власної реалізації удосконаленого модуля відслідковування помилок у високонавантажених вебхдодатках на основі використання власного алгоритму логеру, що дозволяє визначати місце у додатку де саме сталася помилка, та відправляти інформації у журнал подій з використанням мови програмування Javascript. Дане рішення дозволить зменшити розмір програмного додатку для завантаження до 5 кілобайт та зберігати історію помилок. Для досягнення поставленої мети потрібно розв'язати основні задачі: дослідити відомі підходи відслідковування помилок у високонавантажених веб-додатках та існуючі практичні рішення які можуть

відслідковувати помилки у високонавантажених веб-додатках та сформувати вимоги до програмного модулю; розробити алгоритм відслідковування помилок у високонавантажених веб-додатках; розробити програмну реалізацію модуля відслідковування помилок у високонавантажених веб-додатках з використанням мови програмування Javascript; провести тестування на доцільність використання розробленого програмного модулю.

ТЕОРЕТИЧНІ ОСНОВИ ДОСЛІДЖЕННЯ

Під поняттям сучасний високонавантажений веб-додаток авторами наділі буде розумітися унікальна розробка, орієнтована на рішення значної кількості прикладних задач в інформаційних системах та мережах. Для розширення послуг, підвищення їх якості та простоти уявлення, сучасні компанії широко впроваджують та використовують сучасні веб-додатки. Перевагою використання сучасних клієнт-серверних веб-додатків є те, що учасникам інформаційного обміну не потрібно додатково встановлювати спеціалізоване програмне забезпечення, оскільки для простоти реалізації всі дії відбуваються у браузері [1-3]. Для розробки серверної частини додатку на сьогодні використовуються різні мови та інструменти програмування такі як: Ruby; Perl; PHP; Python; NodeJS; та інші. Розробка клієнтської частини передбачає використання: JavaScript; Vue; React; та інші. Сфера використання та призначення сучасного веб-додатку може бути різноманітна, все залежить від умов та направлень для чого він буде призначений, в якому середовищі він створений і запущений. Найпростішими прикладами реалізації та використання веб-додатків є, наприклад: бронювання готелів та квитків, автомобілів і т.д.; купівля товарів; замовлення їжі, різних послуг; розваги, ігри; фінансові, банківські послуги різного типу – починаючи від калькулятора валют і закінчуючи повноцінним онлайн-банкінгом; соціальні мережі; освітні програми – наприклад, з вивчення іноземної мови; біржі фрілансу; CRM та багато іншого.

Розглянемо кілька найпопулярніших моделей високонавантажених веб-додатків:

– ERP (Enterprise resource planning) система - планування корпоративних ресурсів. Планування корпоративних ресурсів відноситься до типу програмного забезпечення, яке організації використовують для управління повсякденною комерційною діяльністю, такою як бухгалтерський облік, закупівлі, управління проектами, управління ризиками та дотримання вимог, а також операції з ланцюгами поставок

– Корпоративний портал. Корпоративний портал управляє внутрішнім інформаційним середовищем компанії для організації колективної роботи над завданнями, проектами та документами, а також для встановлення ефективних внутрішніх комунікацій.

– Ecommerce (електронная комерція). Ecommerce, також відома як електронна комерція або інтернет-торгівля, стосується купівлі-продажу товарів або послуг за допомогою Інтернету, а також передачі грошей та даних для здійснення цих операцій

– CRM (Customer relationship management) – управління взаємовідносинами з клієнтами. Це технологія управління всіма відносинами та взаємодією вашої компанії із клієнтами та потенційними клієнтами [4-8, 10-12].

Отже основний механізм роботи високонавантажених веб-додатків, веб-додаток отримує запит від клієнта і виконує обчислення, після цього формує веб-сторінку і відправляє її клієнту через мережу з використанням протокола HTTP. При побудові високонавантажених веб-додатків є декілька спільних технік, наприклад кешування,

асинхроне завантаження даних так фонові воркери, розділена архітектура, відказостійкі система, балансувач навантаження [5].

Розглянувши теоретичні принципи поняття високонавантажених веб-додатків перейдемо до розгляду принципів відслідковування помилок у високонавантежних веб-додатках реалізованих мовою програмування Javascript. В даній роботі увага буде сконцентрована на роботу клієнтської частини реалізованої мовою програмування JavaScript.

На сьогоднішній день є два готових рішення для відслідковувааня помилок у високонавантажених веб-додатках, це Sentry.io та Catch.js. Далі буде розглянуто більш детально кожен з вищезгаданих варіантів.

Sentry.io – це бібліотека для відслідковування помилок у високонавантажених веб-додатках. На даний момент це рішення є безшкотовним для пробного періоду, проте якщо використовувати його у великій екосистемі, то його послуги та масштабування коштувати чималих коштів для організвації.Sentry.io без додаткових налаштувань пропонує наступні підходи відслідковування помилок у високонавантажених веб-додатках: відслідковування HTTP запитів; відслідковування змін у адресній строці браузера; дії користувача у браузері; збір консольних метрик; посилання будь-яких помилок, які сталися під час користування веб-додатком. Отже, Sentry.io, досить непогане рішення для відслідковування помилок у високонавантажених веб-додатках написаних на мові програмування Javascript. Прото є декілька недоліків, це спеціальний синтаксис для відслідковування помилок, також це рішення є платним і потрібно завантажувати на ваш веб-додаток ще один скрипт, яких і так у високонавантажених веб-додатках вистачає [13].

Catch.js – це бібліотека для відслідковування помилок у високонавантажених веб-додатках. На даний момент це рішення є безшкотовним для пробного періоду який триває лише 14 днів, проте якщо використовувати його у великій екосистемі, то його послуги та масштабування будут коштувати чималих коштів. Catch.js без додаткових налаштувань пропонує наступні підходи відслідковування помилок у високонавантажених веб-додатках: відслідковування HTTP запитів; знімки екранів користувачів при помилці; дії користувача у браузері; збір консольних метрик; інформація про браузер користувача де сталася помилка. Отже, Catch.js, також непогане рішення для відслідковування помилок у високонавантажених веб-додатках, проте воно також платне, и досить велике по розміру [12].

Отже, проаналізувавши вищезагадані вимоги, можна зробити висновок, що для того, щоб якісно слідкувати за помилками у високонавантажених веб-додатках написаних на мові програмування Javascript, потрібно визначити вимоги щодо розробки власного програмного модуля. Далі представимо вимоги, які висуваються до авторського програмного модуля відслідковування помилок у високонавантажених веб-додатках:

1. потрібно проводити відслідковування кожного XMLHttp запит, який відбувається на стороні серверу або клієнтській частині, щоб можливо було при виникненні помилки переглянути зв’язані запити, або який запит передував помилковому;

2. потрібно мати інформацію про браузер та user-agent користувача, щоб зрозуміти як відтворити помилку та у якому середовищі виправляти цю помилку;

3. потрібно мати інформацію про тип помилки, яка сталася, тобто це може бути помилка при виконанні коду, або помилка при запиті на будь-який ресурс;

4. програмне рішення має бути гнучким та мати невеликий розмір, щоб не відбувалося приривання виконання програмного коду веб-сторніки, тобто це дозволить в реальному часі відслідковувати помилки користувача, серверу, завантаження сторонніх скриптів та бібліотек.

5. повинна бути зв'язанність інформації про помилки, та збір статистики, отже кожен запис про помилку, повинен мати певний ідентифікатор для того, щоб переглянути зв'язані запити або помилки.

6. програмне рішення має бути безшкоштовним, та зберігати історію помилок для якісного та кількісного порівняння інформації про помилки.

З вище сформульованих вимог можна прийняти рішення про необхідність розробки власного авторського програмного модулю для відслідковування помилок у високонавантажених веб-додатках, який буде в повній мірі задовольняти всім критеріям та вимогам сучасності.

ПРАКТИЧНІ РЕЗУЛЬТАТИ ДОСЛІДЖЕННЯ

Перш ніж розробляти алгоритм програмного модулю, перш за все слід обрати технології для його реалізації. Для розробки данного програмного модулю було взято за основну мову програмування – Javascript. Також слід зазначити, щоб розробка алгоритму буде відбуватися, до вже функціонуючого високонавантаженого веб-додатку, написаного на мові програмування Javascript. Для того, щоб вдало розробити алгоритм відслідковування помилок, слід знати, що у висконавантаженому веб-додатку використовуються підхід SSR(Server side Rendering) рендеринг веб-сторінки на сервері та технологія кешування. Технологія для клієнтської логіки – це Vue.js, та декілька бібліотек які допоможуть відправляти запити та взаємодіяти з користувачем такі як axios, vuex, vue-router, localStorage. Scss, lodash. Vue є цілою екосистемою для побудови інтерфейсів користувача. Vue цілком здатний запускати складні програми з однією сторінкою (SPA – single page application), використовуючи їх у поєднанні з сучасними інструментами та бібліотеками. Технологія для серверної логіки – це Node.js, Express.js та Typescript, які на сьогоднішній день є найбільш вдалими та популярними рішеннями для побудови серверної логіки, та взаємодії з SSR технологією.

Розробку програмного модулю можна умовно поділити на дві частини:

1. Розробка алгоритму спеціального логеру, для збирання інформації про помилки та різних метрик.
2. Розробка алгоритму відслідковування місця де саме сталася помилка.

Авторський програмний модуль, реалізований мовою програмування Javascript та Typescript. Запропонований програмний модуль вирішує проблему необхідності відслідковування помилок у висконавантажених веб-додатках, та відповідає усім критеріям які було згадано вище. Проте, це рішення зроблено конкретно під потреби високонавантаженого веб-додатка. Модуль складається з трьох частнин: алгоритм логеру, алгоритм відслідковування помилок у конкретному місці де сталася помилка та програмний інтерфейс з Kibana. Велика перевага цього модулю в тому, що кожна частина може використовуватися незалежно від інших, та може масштабуватися у майбутньому. Спеціальний логер, для збирання інформації про помилки та різних метрик в даному випадку буде викликатися безпосередньо на стороні серверу, проте, його можна буде використовувати і на клієнтській частині, оскільки ми використовуємо технологію серверного рендерингу. Для початку слід зазначити, що даний логер буде окрім логування помилок, ще записувати, наприклад повідомлення інформаційного характеру, які можуть бути пов'язані з помилками, для того, щоб покроково відтворити сценарій помилки. Авторський логер буде розмежовувати типи повідомлень, які будуть логуватися. У нашому випадку, це буде відбуватися зо допомогою параметру level_name, він може приймати такі значенням: INFO, для того

щоб збирати будь-яку інформацію про продукт на стороні серверу; CLIENT_INFO, для того щоб збирати будь-яку інформацію про продукт на стороні клієнту; WARNING, для того щоб збирати будь-яку інформацію про попередження на стороні серверу; CLIENT_CRITICAL, для того щоб збирати інформацію про критичні помилки на стороні клієнту; CRITICAL, для того щоб збирати інформацію про критичні помилки на стороні серверу; DEBUG, для того щоб збирати інформацію про код для розробки на стороні клієнту або серверу; CLIENT_WARNING, для того щоб збирати інформацію про попередження на стороні клієнту. Далі опишемо інтерфейс, повідомлень про помилки, які будемо безпосередньо записувати в журнал при використанні авторського логеру. Інтрефейс складається з параметрів: label, параметр, який дає загальну інформацію про повідомлення; message, поле, яке буде містити у собі повідомлення про помилку або будь-яку інформацію яку ми будемо передавати в логер; context, необов'язкове поле, яке міститиме у собі інформацію про контекс виклику нашого логеру;level_name, поле, в якому потрібно вказати тип повідомлення, можливі типи (INFO, CLIENT_INFO, WARNING, CLIENT_CRITICAL, CRITICAL, DEBUG, CLIENT_WARNING); scope, поле в якому буде вказано об'єм помилки, та де вона була викликана; stack, стек, який був створений при умовах коли відбулася помилка.

Проведемо тестування програмного модулю, для цього буде потрібно підключити наш модуль до вже працюючого високонавантаженого веб-додатку та навмисно зробити помилку, для того щоб перевірити модуль на працездатність та повноту інформації, яка буде потрапляти в Kibana.

Додамо фрагмент коду до головного Javascript файлу проекту main.js

```
window.onerror = function (err, info, vm) {
        const errorInfo = {
            message: err.message,
            stack: err.stack,
            level_name: "ERROR",
            context: {
                info,
                vm: vm.$el.className || "",
                uri: vm.$el.baseURI,
            },
        };

        log.error("GLOBAL_ERROR_HANDLER", errorInfo);
    }
```

Проведемо тестування, а саме запустимо сторінку додатку з повільним інтернет з'єднанням, для того, щоб нариклад деякі фрагменти коду повільно завантажувалися, або спрацював таймаут завантаження та була викличена помилка. Далі подивимося на наші логи в Kibana, які зображено на Рисунках 1 та 2.

Рис. 1 Приклад логу про помилку в Kibana

ISSN 2663 - 4023

Із даної помилки бачимо, дату, коли вона сталася, дані про те в якому браузері відбулася помилка, IP клієнта, географічні координати звідку було зроблено запит та країну, повідомлення про те, що сталася помилка при завантаженні скрипту google-analytics.js, та код помилки 451, також видно звідки користувач переходив на наш веб-додаток.

Щоб оцінити ефективність впровадження програмного модулю відслідковування помилок, достатньо поглянути на статистику помилок після його впровадження. Почнемо з статистики, оскільки раніше був впроваджений тільки логер, без відслідковування помилок, загальна кількість логів за один день в середньому була від 5 до 100 записів. Після впровадження даного модулю кількість помилок стала близько 500 в день. Тому що, стало відомо про багато помилок, про наявність яких не було гадки, і виявлено, що в багатьох випадках користувачі веб-додатку не отримували потрібний контент. Проте тепер стало видно в яких випадках відбуваються помилки, та те, що вони взагалі є. Таким чином, на протязі близько місяця, була проведена масштабна робота над помилками, та знижена їх кількість до 50 в день, що дало дуже значний ефект на продуктивність веб-додатку, та кількість звернень користувачів до служби підтримки з технічних питань зменшилась у два рази. Даний модуль дозволяє відслідковувати такі типи помилок, а саме запити від клієнтської частини на серверну під час виконання яких відбулася помилка, помилки на серверній частині додатку; помилки під час кешування та дозволяє знаходити місце у коді, де саме сталася помилка.

@timestamp	Nov 24, 2020 @ 23:02:33.737
_id	zYWc03UBJ_ynl2NmOiPp
_index	kibana_sample_data_logs
_score	-
_type	_doc
agent	Mozilla/5.0 (X11; Linux i686) AppleWebKit/534.24 (KHTML, like Gecko) Chrome/11.0.696.50 Safari/534.24
bytes	3,758
clientip	193.185.70.83
event.dataset	sample_web_logs
extension	deb
geo.coordinates	{ "lat": 38.70042333, "lon": -87.12973222 }
geo.dest	IN
geo.src	BI
geo.srcdest	BI:IN
host	artifacts.elastic.co
hour_of_day	21
index	kibana_sample_data_logs
ip	193.185.70.83
machine.os	win xp
machine.ram	8,589,934,592
memory	-
message	Network error, timeout exceeded while loading google-analytics.js
phpmemory	-
referer	http://twitter.com/success/michael-p-anderson
request	/elasticsearch/elasticsearch-6.3.2.deb
response	451
tags	error, info
timestamp	Nov 24, 2020 @ 23:02:33.737
url	https://artifacts.elastic.co/downloads/elasticsearch/elasticsearch-6.3.2.deb
utc_time	Nov 24, 2020 @ 23:02:33.737

Рис. 2 Детальна інформація про помилку

Нижче будуть приведені зображення до та після введення модулю відслідковування помилок по кількості.

Рис. 3 Кількість логів до введення програмного модулю

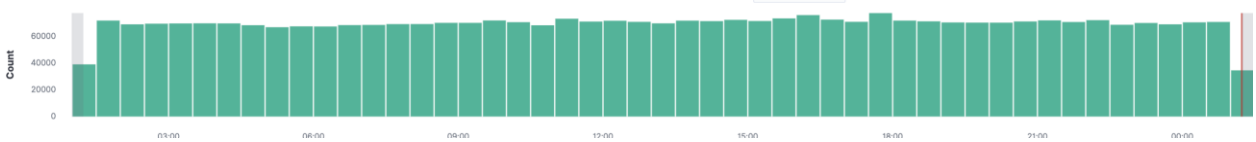

Рис. 4 Кількість логів після введення програмного модулю

Повернемося до порівняльної таблиці будуть приведені зображення до та після введення модулю відслідковування помилок по кількості.

Таблиця 1.

Порівняльна таблиця рішень для відслідковування помилок у високонавантажених веб-додатках написаних на мові програмування Javascript

Назва рішення / Вимога	Catch.js	Sentry.io	Авторська реалізація
Інформація про кожен XMLHTTP запит	-	-	+
Інформація про оточення користувача	+	+	+
Детальна інформація про тип помилки	+	+	+
Гнучкість	-	-	+
Зв’язаність інформації про помилку з іншими запитами	-	-	+
Визначення конкретного місця де сталася помилка при виконанні програмного коду	-	+	+
Збереження історії помилок та безкоштовність рішення	-	-	+
Розмір програмного модулю	300кб	200кб	5кб

З порівняльної таблиці можна зробити висновок, що авторська реалізація цілком задовольняє усі вимоги, які було перераховано вище, а саме збираються метрики про кожен XMLHTTP запит, збирається інформація про оточення користувача в якому сталася помилка, збирається інформація про те, чим саме була викликана помилка, рішення гнучке та масштабоване, присутня зв’язанність інформації про помилку з іншими запитами за допомогою поля id, проводиться детальне визначення конкретного місця, де сталася помилка при виконанні програмного коду, за допомогою власноруч розробленого алгоритму, відбувається збереження історії помилок у журналі Kibana.

ВИСНОВКИ

В даній статті було проведено детальний аналіз сучасних підходів відслідковування помилок у високонавантажених веб-додатках, ознайомлено з проблемою та

необхідністю відслідковування помилок у сучасних програмних модулях. Було приділено увагу основним інструментам, які існують на сьогоднішній день для відслідковування помилок у високонавантажених веб-додатках реалізованою мовою програмування Javascript, такі як Sentry.io, Catch.js. Сформовано авторські вимоги та на базі них авторський програмний модуль відслідковування помилок. На базі цих досліджень авторами досягнуто таких результатів: досліджено відомі підходи відслідковування помилок у високонавантажених веб-додатках та існуючі практичні рішення, які можуть відслідковувати помилки у високонавантажених веб-додатках та сформовано вимоги до програмного модулю; розроблено алгоритм відслідковування помилок у високонавантажених веб-додатках; розроблено програмну реалізацію модуля відслідковування помилок у високонавантажених веб-додатках з використанням мови програмування Javascript; проведено тестування на доцільність використання розробленого програмного модулю.Також було реалізовано власний програмний модуль відслідковування помилок мовою програмування Javascript, та проведено його тестування на реальній системі. Авторський модуль складається з трьох частнин: алгоритм логеру, алгоритм відслідковування помилок у конкретному місці де сталася помилка та програмний інтерфейс з Kibana. Цей комплекс дозволяє встановлювати місце у додатку де саме сталася помилка, та відправляти інформації у журнал подій та зберугати історію виникнення посилки. Впровадження даного рішення дозволяє зменшити розмір програмного додатку для завантаження до 5 кілобайт, що дозволяє прискорити роботу процедури відслідковування посилок та зменшити навантаження на інформаційну систему.

Зважаючи на досягнення мети роботи практична цінність цих рішень є актуальною оскільки наявні посилки високонавантажених веб-додатків здатні призвести до порушення чи цілісності інтерфейсу веб-додатку чи до порушення життєдіяльності інформаційної системи в цілому.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1 *Error Handling -- Eloquent JavaScript*. (б. д.). Eloquent JavaScript. https://eloquentjavascript.net/1st_edition/chapter5.html
2 *Control flow and error handling - JavaScript | MDN*. (б. д.). MDN Web Docs. https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Control_flow_and_error_handling
3 Gisder-Dubé, L. (2018, 11 листопада). *Handling Errors in JavaScript: The Definitive Guide*. Medium. https://levelup.gitconnected.com/the-definite-guide-to-handling-errors-gracefully-in-javascript-58424d9c60e6
4 *Error handling — a modern introduction to programming with javascript and jquery*. (б. д.). Open Book Project. https://www.openbookproject.net/books/mi2pwjs/ch04.html
5 https://docs.swift.org/swift-book/LanguageGuide/ErrorHandling.html
6 *Building high performance, scalable web applications*. (б. д.). HashJar. https://www.hashjar.dev/blog/building-high-performance-scalable-applications
7 *Why is error handling important?* (б. д.). Stack Overflow. https://stackoverflow.com/questions/368139/why-is-error-handling-important
8 Maglovanyi, A. (2019, 3 березня). *Error handling in javascript*. Medium. https://itnext.io/error-handling-in-javascript-3e444ccae117
9 *What is erp? | oracle*. (б. д.). Oracle | Integrated Cloud Applications and Platform Services. https://www.oracle.com/applications/erp/what-is-erp.html
10 *Ecommerce definition - what is ecommerce*. (б. д.). Shopify. https://www.shopify.com/encyclopedia/what-is-ecommerce
11 *Corporate portal | "IT-Інтегратор"*. (б. д.). IT Integrator – Партнерство IT та бізнесу. https://it-integrator.ua/en/corporate-portal

12 Warden, J. (2017, 11 листопада). *Error Handling Strategies - DZone Performance*. dzone.com. https://dzone.com/articles/error-handling-strategies

13 *Content Security Policy (CSP) - HTTP | MDN*. (б. д.). MDN Web Docs. https://developer.mozilla.org/en-US/docs/Web/HTTP/CSP

КІБЕРБЕЗПЕКА: освіта, наука, техніка
CYBERSECURITY: EDUCATION, SCIENCE, TECHNIQUE
№ 3 (11), 2021
ISSN 2663 - 4023

Ilyenko Anna
Candidate of Technical Sciences, assistant professor ,
assistant professor of Information Security Systems Department
National Aviation University of Kyiv, Ukraine
ORCID ID: 0000-0001-8565-1117
ilyenko.a.v@nau.edu.ua

Ilyenko Sergii
Candidate of Technical Sciences, assistant professor ,
assistant professor of Automation and Energy Management Department
National Aviation University of Kyiv, Ukraine
ORCID ID: 0000-0002-0437-0995
ilyenko.s.s@nau.edu.ua

Stashevskyi Danylo
Student Information Security Systems Department
National Aviation University of Kyiv, Ukraine
ORCID ID: 0000-0001-9910-5504
121lak0sta121@gmail.com

SOFTWARE ERROR TRACKING MODULE IN WEB APPLICATIONS BASED ON THE USE OF LOGGER ALGORITHM

Abstract. This article is devoted to the consideration of further relevant ways to ensure the procedure of error tracking in high-load web applications implemented in the Javascript programming language. The article investigates and identifies that errors that occur when developing and using modern high-load web applications are very dangerous because they affect the full functioning of the information system as a whole and can lead to breaches of confidentiality and integrity of personal information. In the article the authors consider the issues of error handling in Javascript programming language, the problem and need to track errors in high-load web applications, the concept of high-load web applications, existing methods and approaches to error tracking, principles of modern high-load web applications and comparison of existing error tracking solutions. in high-load web applications implemented in the Javascript programming language. The result of this research was the creation of an author's software module for error tracking in advanced web applications to solve the problem of logging errors, analysis of logs for completeness, error handling and solving them in the future. Also, the implementation of such a solution allows you to reduce the size of the software application to download up to 5 kilobytes and save the error history. The developed software module for error tracking in highly loaded web applications consists of two parts: client and server. Each part is an independent software module and can be reconfigured with minimal configuration changes on any other resource. This implementation allows you to fully collect metrics for each XMLHTTP request, collect information about the user environment in which the error occurred, collect information about what exactly caused the error, determine the specific location where the error occurred while executing program code, using a custom algorithm , save error stories in Kibana log. Possible areas of development of this work are related to the expansion of the error tracking algorithm, to collect more data and improve their aggregation, based on the expansion of metrics. The authors plan a number of scientific and technical solutions to develop and implement effective methods, tools, requirements, principles and approaches to cyber security and protection based on the use of author's approaches to error tracking in high-load web applications in experimental computer systems and networks.

Keywords: high-load web application, logger, javascript, kibana, bug tracking.

ISSN 2663 - 4023

REFERENCES

1 *Error Handling -- Eloquent JavaScript*. Eloquent JavaScript. https://eloquentjavascript.net/1st_edition/chapter5.html
2 *Control flow and error handling - JavaScript | MDN*. MDN Web Docs. https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Control_flow_and_error_handling
3 Gisder-Dubé, L. (2018). *Handling Errors in JavaScript: The Definitive Guide*. Medium. https://levelup.gitconnected.com/the-definite-guide-to-handling-errors-gracefully-in-javascript-58424d9c60e6
4 *Error handling — a modern introduction to programming with javascript and jquery*. Open Book Project. https://www.openbookproject.net/books/mi2pwjs/ch04.html
5 https://docs.swift.org/swift-book/LanguageGuide/ErrorHandling.html
6 *Building high performance, scalable web applications*. HashJar. https://www.hashjar.dev/blog/building-high-performance-scalable-applications
7 *Why is error handling important?* Stack Overflow. https://stackoverflow.com/questions/368139/why-is-error-handling-important
8 Maglovanyi, A. (2019). *Error handling in javascript*. Medium. https://itnext.io/error-handling-in-javascript-3e444ccae117
9 *What is erp? | oracle*. Oracle | Integrated Cloud Applications and Platform Services. https://www.oracle.com/applications/erp/what-is-erp.html
10 *Ecommerce definition - what is ecommerce*. Shopify. https://www.shopify.com/encyclopedia/what-is-ecommerce
11 *Corporate portal | " IT Integrator* - Partnership of IT and businessy. https://it-integrator.ua/en/corporate-portal
12 Warden, J. (2017, 11 листопада). *Error Handling Strategies - DZone Performance*. dzone.com. https://dzone.com/articles/error-handling-strategies
13 *Content Security Policy (CSP) - HTTP | MDN*. MDN Web Docs. https://developer.mozilla.org/en-US/docs/Web/HTTP/CSP